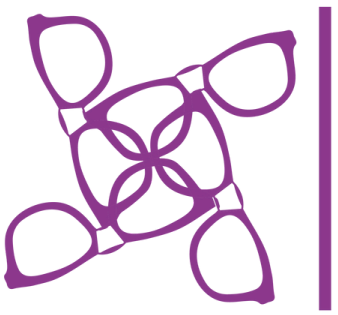
tech station padova

MOLEN Geek

techgrounds



Molengeek International

AWS re/Start

Co-funded by
the European Union

General Program Information

Program Title

AWS re/Start

The AWS re/Start program is a practical, career-oriented cloud-training initiative delivered by MolenGeek in partnership with Amazon Web Services. It introduces participants with little or no IT experience to the fundamentals of cloud computing and prepares them for entry-level cloud or IT support roles. Learners acquire hands-on technical skills, foundational knowledge of IT systems, and the professional behaviour required to succeed in a digital environment. The course follows the AWS global framework while integrating local coaching, employability support, and community learning.

Organisation

MolenGeek (asbl / non-profit organisation)

MolenGeek is an international technology ecosystem that bridges education and employment. Founded in Molenbeek (Brussels), it now operates across Belgium and France with active hubs in Brussels, Antwerp, Charleroi, Gaume and Roubaix, and partnerships throughout Europe. Its mission is to make technology accessible to everyone, regardless of their background, age, or education level.

MolenGeek offers a broad portfolio of digital and cloud programs—ranging from short bootcamps to full reskilling trajectories—and collaborates with industry leaders such as AWS, Google, Salesforce, Meta, and Proximus.
Through its community-driven approach, MolenGeek combines technical upskilling with personal development, mentoring, and entrepreneurship, creating an inclusive gateway to digital careers.

Country / Local Context

Belgium and France – Hybrid Learning Model with European Reach

AWS re/Start is offered in both Belgium and France through a hybrid delivery format that blends online learning with optional in-person participation at MolenGeek hubs. Learners can attend live classes and coaching sessions on-site or join fully online from anywhere.

Both countries face a continued shortage of skilled IT professionals and a growing need for diversity in tech. By maintaining local training centers in disadvantaged neighbourhoods while providing full online accessibility, MolenGeek reaches 'hidden talent'—individuals who may lack access to traditional education yet show strong motivation and aptitude. This approach maximises inclusion and scalability while staying connected to local labour-market realities.

Graduates benefit from MolenGeek's strong network of partner companies and its career-placement support, which includes coaching, mentorship, and introductions to employers looking for entry-level cloud professionals.

Target Audience

Reskillers, job seekers, career changers, students, and underrepresented talent

The program is designed for people who want to start or restart a career in technology, particularly in cloud. It welcomes:
- Career changers transitioning from non-technical backgrounds;
- Job seekers registered with employment agencies;
- Students and graduates seeking professional orientation;
- Women and individuals from diverse cultural backgrounds, refugees, and underrepresented groups in tech.

No formal diploma or IT experience is required. Admission is based on motivation, reliability, and learning potential. While accessible to complete beginners, the program includes activities that help participants assess whether technical roles suit their interests and cognitive comfort level. This reflective orientation reduces dropout rates and ensures that learners move on to follow-up programs or jobs that match their strengths.

Languages of Delivery

English – French – Dutch

Reflecting Belgium's and France's multilingual environment, AWS re/Start at MolenGeek is delivered in three languages. Participants may follow the course in their preferred language or combine materials to improve multilingual professional communication. Core AWS resources are available in English, while all instructions, coaching, and assessments are supported in English, French, and Dutch. This trilingual model ensures full accessibility for a broad audience across both countries.

Delivery Mode and Learning Environment

- Hybrid structure: learners choose between full online access and on-site participation at MolenGeek hubs.
- Flexible learning: self-paced labs combined with live instructor sessions, group discussions, and mentoring.
- Community support: each cohort is connected via Slack / Discord for peer-learning, Q&A, and collaborative projects.
- Practical focus: every module includes guided exercises in an AWS sandbox environment, ensuring hands-on skill development.
- Career integration: participants receive coaching on CV writing, LinkedIn branding, interview preparation, and direct networking with partner employers.

Key Facts Summary

Aspect	Details
Program Title	AWS re/Start
Organisation	MolenGeek (asbl)
Delivery Mode	Hybrid (online + on-site)
Duration	12 weeks (approx. 240 hours) + career coaching
Audience	Reskillers, job seekers, students, and underrepresented groups
Languages	English, French, Dutch
Learning Format	Practical labs, live sessions, community learning
Outcome	Entry-level cloud skills + career readiness
Main Partners	AWS, Google, Microsoft, Meta, Proximus

Section 2 – Program Objectives and Description

Overall Goal of the Program

The AWS re/Start program delivered by MolenGeek aims to help individuals—especially those with little or no technical background—build a solid foundation in IT and cloud computing, develop employability skills, and gain the confidence needed to pursue a career in the technology sector. The program combines technical learning, soft skills, and career planning in a supportive community environment.

MolenGeek positions AWS re/Start as part of its broader reskilling strategy: first providing an accessible introduction to cloud and IT concepts, followed by deeper specialisation or employment pathways. The immediate goal is not job placement alone but ensuring that participants are well-informed, motivated, and technically prepared for entry-level cloud roles or further certification programs. This structure reduces dropouts and mismatches, improving success rates for learners and employers alike.

Specific Learning Objectives

- Understand essential IT and cloud terminology, including key AWS concepts.

- Experience hands-on labs to assess affinity for technical work and cloud environments.

- Identify which IT and cloud roles (such as Cloud Support, DevOps, or Security) best match their skills and interests.

- Acquire core knowledge of IT fundamentals—operating systems, networking, databases, and cybersecurity.

- Gain introductory programming skills in Python and automation for AWS environments.

- Develop a personal career and reskilling plan outlining next steps and certification goals.

- Understand available funding options and resources for continued learning or employment.

- Recognise the impact of Artificial Intelligence (AI) on the modern workplace and its relation to cloud operations.

Brief Overview of Curriculum Topics

Module / Learning Block	Key Topics	Learning Method
Learning Principles & Orientation	Computational thinking, effective learning strategies, responsible use of AI tools such as ChatGPT, and prompt engineering.	Interactive workshops, guided self-study, and applied labs
Career Planning	Building a personal reskilling and career plan with dedicated templates and individual coaching sessions.	Interactive workshops, guided self-study, and applied labs
Cloud & IT Domains	Exploration of cloud, data, security, DevOps, and software development domains to identify a personal career direction.	Interactive workshops, guided self-study, and applied labs
IT Fundamentals	Basic understanding of networks, operating systems, and programming languages.	Interactive workshops, guided self-study, and applied labs
Programming Basics	Python essentials with practical exercises using AWS SDK (boto3) to automate real-world tasks.	Interactive workshops, guided self-study, and applied labs
AWS Core Services	Hands-on experience with EC2, S3, RDS, IAM, VPC, and CloudWatch, focusing on scalability, cost, and security.	Interactive workshops, guided self-study, and applied labs
Security & Compliance	Introduction to IAM, encryption, shared responsibility, and GDPR awareness.	Interactive workshops, guided self-study, and applied labs
AI & Automation	Understanding how AI supports cloud operations and responsible automation practices.	Interactive workshops, guided self-study, and applied labs

Soft Skills Development	Communication, teamwork, and professional collaboration embedded across modules.	Interactive workshops, guided self-study, and applied labs

Duration (Total Hours and Weeks)

The full AWS re/Start program spans approximately 12 weeks (around 240 hours) of guided learning, practice labs, and professional coaching. Learners can access online materials flexibly for revision and independent practice. The structure typically includes:
- 40 hours of IT fundamentals and orientation modules;
- 60 hours of AWS services and practical labs;
- 40 hours of projects and assessments;
- 20 hours of soft skills and career development;
- Additional optional material on AI and automation.

This modular setup allows learners to progress gradually from theory to application, supported by instructors and mentors.

Prerequisites (Knowledge and Experience)

No prior IT knowledge or formal education is required. Admission is based on motivation, curiosity, and learning potential. Participants should demonstrate basic digital literacy—such as using email, browsers, and online communication tools. All required tools, including access to the AWS sandbox environment, Slack/Discord channels, and GitHub, are provided at the start of the course.

This inclusive approach ensures accessibility for individuals from all backgrounds while assembling motivated learners who can successfully complete the program and transition into further training or employment.

Section 3 – Structure and Modules

The AWS re/Start program at MolenGeek is composed of multiple components that together offer a comprehensive and practical learning journey. The structure is designed to guide participants progressively—from IT and cloud fundamentals to hands-on technical skills and professional development. Through a blend of self-study, live interaction, peer support, and

career coaching, the program delivers a multimodal experience well-suited for learners of diverse backgrounds, including those entirely new to technology.

Every block focuses on a specific domain—technical, professional, or reflective—and includes applied assignments and AWS-based labs. By integrating topics like Artificial Intelligence (AI), automation, and LinkedIn-based employability tools, the curriculum remains aligned with current industry needs and future job-market trends.

Table: AWS re/Start Program – Module Overview

Module / Component	Key Topics / Activities	Learning Method	Intended Outcomes
Learning Block 1: Learning Principles & Orientation	Computational thinking, learning strategies, prompt engineering, responsible AI usage.	Interactive self-study, instructor-led workshops.	Develop metacognitive and analytical skills crucial for IT problem-solving.
Learning Block 2: Career Planning	Creation of a personal reskilling and career plan; exploration of IT roles, certifications, and pathways.	Guided assignments, templates, and mentoring.	Gain direction through a personal roadmap for professional growth.
Learning Block 3: Cloud & IT Domains	Overview of Cloud, Data, Security, DevOps, Software, and AI domains.	Self-study modules, expert Q&A, webinars.	Understand IT specialisations to identify preferred career directions.
Learning Block 4: IT Fundamentals	Basic computing concepts: operating systems, networking, databases, and security.	Online modules with AWS sandbox labs.	Build foundational IT knowledge applicable to cloud environments.
Learning Block 5: Programming Basics	Python scripting for automation, APIs, and AWS SDK (boto3).	Practice-based exercises and tutorials.	Develop the ability to automate cloud processes and basic code logic understanding.
Learning Block 6: AWS Core Services	Hands-on labs for EC2, S3, RDS, IAM,	Instructor-led sessions and	Gain operational knowledge of AWS compute, storage,

	VPC, Lambda, and CloudWatch.	sandbox environments.	database, and monitoring services.
Learning Block 7: Cloud Security & Compliance	IAM roles/policies, encryption, CloudTrail, Config, GDPR compliance.	Case studies and scenario-based labs.	Implement secure and compliant architectures following AWS best practices.
Learning Block 8: AI & Automation in Cloud	AI-driven monitoring, automated scaling, cost optimisation, responsible AI use.	Self-study, guided projects, applied labs.	Apply AI concepts responsibly and use automation to improve efficiency.
Learning Block 9: Soft Skills & Professional Development	Communication, teamwork, problem-solving, and professional behaviour.	Workshops and peer-learning activities.	Demonstrate effective collaboration and communication within a tech environment.
Learning Block 10: Final Project & Career Integration	Capstone AWS deployment, team presentation, career coaching.	Hands-on project, mock interviews, LinkedIn enhancement.	Showcase technical competence and job readiness with a professional portfolio.

Weekly Live Sessions and Community Interaction

Participants join up to five live sessions weekly, including technical masterclasses, troubleshooting clinics, peer review workshops, and professional development sessions. All are delivered online via platforms like Zoom or Google Meet, with recordings available for flexibility. Learners interact continuously through Slack or Discord, where they share progress, exchange knowledge, and collaborate on projects. This structure encourages motivation, accountability, and collective learning.

Career Assessment and Peer Learning Tools

The program includes a formal career assessment to identify each participant's interests, strengths, and professional goals. Learners gain access to tools such as LinkedIn Premium and the IT Navigator platform to explore job roles, map training pathways, and expand professional networks. Continuous peer support fosters collaboration and knowledge exchange across diverse cohorts.

Module: Impact of Artificial Intelligence

This module explores how Artificial Intelligence shapes modern IT and cloud careers. Participants study the impact of AI on automation, security, and data management, and learn to

use AI responsibly within cloud contexts. The goal is to understand how emerging technologies influence work processes, efficiency, and ethical decision-making in the digital economy.

Section 4 – Delivery Methods and Tools

Mode of Delivery

AWS re/Start at MolenGeek follows a hybrid learning model that blends both online and in-person components. This flexible approach allows participants from across Belgium and France to join the program according to their availability and preference. Learners may attend classes and coaching sessions at MolenGeek hubs (Brussels, Antwerp, Charleroi, Roubaix) or follow the full curriculum online. All learning materials, assignments, and recordings are hosted digitally, ensuring accessibility for every participant. The hybrid format maximises inclusion and scalability while maintaining MolenGeek's supportive, community-based learning environment.

Digital Tools and Platforms Used

- MolenGeek Learning Platform – the central platform that hosts all course modules, video lessons, and assignments.
- AWS Skill Builder and Sandbox – the official AWS environment used for hands-on labs, cloud exercises, and project work.
- Slack / Discord – communication platforms for collaboration, Q&A, peer feedback, and community engagement.
- LinkedIn Premium – one-year access granted to support learners' professional networking and career visibility.
- IT Navigator – a digital exploration tool for mapping IT roles, certifications, and training opportunities.
- Video conferencing (Zoom / Google Meet) – used for weekly live sessions, Q&A discussions, and masterclasses.

- ChatGPT and AI Tools – introduced during Learning Principles to support prompt engineering, documentation, and research.

Teaching Methods

MolenGeek applies an active, practice-driven methodology inspired by its core philosophy of "learning by doing" ('C'est en forgeant qu'on devient forgeron'). This means that participants learn by directly experimenting, building, and reflecting on what they create. The program combines self-paced modules with instructor guidance, hands-on labs, and collaborative projects.

- Self-study modules – learners study curated digital materials and recorded sessions independently.
- Webinars and live sessions – weekly interactive lessons covering core AWS concepts, use cases, and demonstrations.
- Tech workshops and practical assignments – guided exercises in networking, security, and automation.
- Q&A sessions – opportunities to interact directly with instructors, mentors, and AWS experts for real-time support.
- Mentorship and coaching – one-on-one or group support to guide professional growth and clarify career goals.
- Peer learning – collaboration through Slack channels, team projects, and peer review to reinforce teamwork and confidence.
- Career assessment – structured self-evaluation to identify strengths, learning goals, and career aspirations.

This diverse teaching mix encourages practical understanding and confidence rather than rote memorisation. It provides learners with the opportunity to 'taste' the IT world through guided experimentation, peer learning, and community engagement.

Assessment Methods

Evaluation within the AWS re/Start program at MolenGeek focuses on continuous feedback, reflection, and applied performance rather than formal testing. The emphasis is on personal growth, skill mastery, and employability readiness.

- Career Assessment – a structured reflection activity to help learners identify interests, strengths, and potential IT pathways.
- Practical Labs and Projects – applied exercises and challenges that demonstrate understanding of cloud concepts through practice.
- Personal Career Plan – an individual document outlining goals, skill development, and career strategies.
- Peer and Mentor Feedback – regular feedback loops through community discussions and mentoring sessions.

- No formal exams – evaluation is based on participation, completion of labs, and demonstrated growth rather than written tests.

Section 5 – Localization and Adaptation

Overview

MolenGeek was founded in Molenbeek, Brussels, with the mission to make technology and innovation accessible to everyone — regardless of background, gender, or prior education. From its inception, MolenGeek has combined education, entrepreneurship, and employability in a single ecosystem guided by the principle “C’est en forgeant qu’on devient forgeron” (It is by forging that one becomes a blacksmith). This hands-on, community-driven approach has grown into an international reference model for inclusive and practice-oriented digital training.

Localization Philosophy

Rather than exporting a fixed curriculum, MolenGeek adapts each program to the social, linguistic, and economic context of the region where it operates. Its educational philosophy — accessibility, experiential learning, and community building — remains constant, while delivery methods, languages, and partnerships are adjusted locally to ensure inclusion, impact, and alignment with the regional tech ecosystem.

Current Localizations

1. Belgium – Headquarters (Molenbeek, Brussels)

• Training available in French, Dutch, and English.
• Programs focused on cloud computing, cybersecurity, and digital entrepreneurship.
• Blended learning format combining onsite sessions with full online accessibility.

2. France – Roubaix

• Training delivered primarily in French, with technical content available in English for international relevance.
• Focus on youth employability, inclusion, and access to technology in underrepresented communities.
• Courses mirror the Belgian model while integrating local community collaboration to increase regional impact.

3. Netherlands – Partner Initiatives

• MolenGeek's methodology inspired Dutch implementations based on a two-phase structure: orientation followed by reskilling.
• Programs include online introductions to IT fundamentals and cloud technologies, supported by hybrid workshops.
• Local adaptations integrate modules on Azure, data engineering, and cybersecurity in response to national labor-market needs.

4. Italy and Morocco – Emerging Collaborations

• New initiatives are being explored to extend the MolenGeek model to Southern Europe and North Africa.
• Focus on building partnerships with local organizations and governments to promote inclusion and digital literacy.
• These collaborations aim to empower youth and bridge skill gaps through culturally relevant, community-based learning environments.

Core Elements of Adaptation

• Multilingual Delivery – Training offered in the dominant regional languages to ensure accessibility.

• Hybrid Learning – Participants can follow courses onsite or entirely online, with full digital access to recordings and resources.

• Local Partnerships – Collaboration with municipalities, educational institutions, and local organizations to support inclusion and outreach.

• Cultural Relevance – Integration of soft skills, mentoring, and coaching adapted to regional professional cultures.

• Industry Alignment – Ongoing curriculum updates reflecting market demand in AWS, Azure, AI, and cybersecurity.

• Legal Compliance – Full adherence to GDPR and local education quality standards.

Challenges and Continuous Adaptation

Scaling a community-based training ecosystem internationally requires balancing authenticity with flexibility. MolenGeek's main challenges include adapting administrative and funding frameworks to diverse contexts, ensuring consistent educational quality across languages, and maintaining strong industry relationships while expanding globally. Each localization serves as a learning laboratory — reinforcing MolenGeek's international reach while preserving its inclusive, community-centered identity.

Future Outlook

Localization at MolenGeek is a continuous and evolving process. The organization will continue to refine its model through cross-border collaboration, technological innovation, and partnerships. Future priorities include:

• Regular updates of curricula to align with emerging technologies and job-market trends.

• Strengthening cooperation with local governments, universities, and tech industries.

• Encouraging cross-hub collaboration between Belgium, France, the Netherlands, Italy, and Morocco to share best practices.

• Expanding global visibility while maintaining the local community spirit that defines MolenGeek's identity.

• Remaining faithful to its founding mission — making technology a source of opportunity for everyone.

Section 6 – Outcomes and Evaluation

Number of Participants Trained

Since its launch, the AWS re/Start program at MolenGeek has trained 91 participants across several cohorts in Belgium. Participants come from diverse backgrounds — including job seekers, career changers, and motivated young graduates — all seeking to build a strong foundation in cloud computing and enter the IT job market.

Completion / Certification Rate

The program maintains an exceptional completion rate of 91.6 %, demonstrating the effectiveness of MolenGeek's learner-centered approach and the commitment of its participants. Upon successful completion, learners earn the AWS Certified Cloud Practitioner certification. Highly motivated graduates also have the opportunity to progress toward the AWS Solutions Architect – Associate certification, supported by mentors and post-course study groups facilitated by AWS and MolenGeek instructors.

These certifications confirm not only the learners' understanding of cloud fundamentals but also their ability to apply technical knowledge in real-world professional environments.

Post-Training Outcomes

The AWS re/Start program at MolenGeek emphasizes career readiness, employability, and long-term growth. Within six months after completion:

• Approximately 70–75 % of graduates secure employment in IT, obtain internships, or continue into advanced training programs.

• Many enter junior roles such as Cloud Support Associate, DevOps Trainee, or Junior System Engineer.

• Others continue developing their technical expertise through associate-level AWS certifications or specialization in cybersecurity and automation.

After graduation, participants gain access to exclusive post-training resources provided by AWS and MolenGeek, which include:

• AWS expert sessions and alumni cohorts for continued guidance and mock interviews.

• A network of professionals and resources to help graduates transition into real-world projects.

• The SideGeek platform, developed by MolenGeek itself, connects graduates directly with partner companies and employers looking for new cloud talent.

This unique ecosystem extends learning beyond the classroom, ensuring that graduates remain supported as they integrate into the job market and build professional experience.

Participant Feedback Summary

Feedback from participants is consistently positive and enthusiastic. Graduates highlight the hands-on nature of the AWS labs, the accessibility of the instructors, and the practical focus of the program. They emphasize that MolenGeek's community-based approach — where learners support one another and engage directly with industry mentors — creates a sense of belonging and confidence.

• "The program gave me my first real chance in IT."

• "The instructors truly care about your success."

• "It's not just a course — it's a community that continues after you graduate."

This culture of inclusion and ongoing support has been one of the main reasons for the program's high retention and success rates.

Lessons Learned and Recommendations

Delivering AWS re/Start at MolenGeek has provided important insights into how inclusive IT education can lead to long-term professional outcomes.

• Combining technical instruction with career and mentoring support greatly enhances learner success.

• Peer learning and community engagement foster motivation and self-confidence.

• Post-training support through AWS resources and MolenGeek's SideGeek platform is vital to sustain employment outcomes.

• Keeping the curriculum aligned with evolving AWS technologies ensures relevance and job-market competitiveness.

Recommendations for continuous improvement:

• Expand employer partnerships to strengthen job-placement pipelines.

• Introduce more real-world, project-based assignments to simulate workplace environments.

• Enhance alumni mentorship within SideGeek to connect graduates with hiring managers and project collaborations.

• Maintain periodic updates of course materials to include new AWS services and automation trends.

Conclusion

With a 91.6 % completion rate and strong post-training employment outcomes, the AWS re/Start program at MolenGeek continues to demonstrate excellence in inclusive, cloud-based education. By integrating recognized AWS certifications, personalized coaching, and innovative tools such as MolenGeek's own SideGeek platform, the program ensures that graduates are not only technically skilled but also professionally empowered to succeed in the digital economy.

Section 7 – Supporting Materials

Overview

The AWS re/Start program at MolenGeek provides participants with a wide range of digital resources, learning materials, and community tools. These materials ensure that learners receive continuous guidance, structured learning, and direct access to real AWS environments and experts. All resources are designed to promote practical learning, career orientation, and peer collaboration beyond the classroom.

Official Program Platforms and Resources

• Official MolenGeek Education Website: https://molengeek.education/en – Central hub for all MolenGeek training programs, including AWS re/Start, providing information about registration, schedules, and ongoing initiatives.

• AWS re/Start Learning Platform (Canvas): https://awsrestart.instructure.com/courses/3463/modules#module_93202 – Official AWS re/Start digital environment containing structured modules, quizzes, and lab exercises. The platform provides approximately 80 hours of guided learning on IT fundamentals, networking, security, Python programming, and AWS core services.

• AWS Skill Builder and Sandbox Environment – Used for hands-on labs and real-time practice with cloud resources. Participants gain practical experience by deploying, monitoring, and securing cloud infrastructure directly within AWS.

• Recorded Live Sessions and Masterclasses – Weekly webinars, Q&A sessions, and technical deep-dives led by AWS certified instructors and industry professionals. All recordings are available for later review, enabling learners to learn at their own pace.

Career and Community Resources

• Career Development Tools – Templates for creating personal reskilling and career plans, including goal-setting frameworks and self-assessment questionnaires. These tools help learners define their professional path and align training outcomes with long-term objectives.

• SideGeek Platform (by MolenGeek) – An internal digital platform developed by MolenGeek to connect graduates directly with employers, recruiters, and partner companies seeking entry-level cloud talent. SideGeek also serves as a collaboration hub for alumni and current students.

• Peer-Learning and Communication Channels – A dedicated Slack workspace for AWS re/Start participants encourages daily collaboration, Q&A, and knowledge exchange between learners and instructors. This community-driven model reinforces motivation and fosters continuous support.

• AWS Post-Training Resources – Exclusive access to AWS expert sessions, alumni cohorts, and career-readiness materials designed to prepare graduates for real-world employment opportunities.

Additional Supporting Materials

• Learning Slides and Exercises – Comprehensive slide decks, guided labs, and assignments covering key AWS topics such as IAM, EC2, S3, VPC, CloudTrail, and CloudFormation.

• Soft Skills Workshops – Sessions on communication, teamwork, and professional behavior in a digital workplace.

• Career Workshops – LinkedIn optimization, mock interviews, and career-branding guidance to improve employability.

• Certification Guides – Official AWS exam preparation materials for the Cloud Practitioner and Solutions Architect – Associate exams.

• Community Networking Events – On-site or virtual meet-ups where learners connect with mentors, alumni, and local companies.

Access and Availability

All materials are accessible online through MolenGeek's learning ecosystem and remain available for a minimum of three months after program completion. Graduates continue to

benefit from community access through Slack, the SideGeek platform, and AWS post-training resources, ensuring long-term support and connection to the digital industry.

Conclusion

The AWS re/Start program at MolenGeek stands as a flagship initiative for inclusive and practice-oriented cloud education. Rooted in MolenGeek's mission to make technology accessible to everyone, regardless of background or prior education, the program empowers diverse learners to take meaningful steps toward sustainable careers in IT and cloud computing.

Key strengths of the program lie in its hands-on methodology, career-driven structure, and community-centered philosophy. By combining theoretical understanding with intensive, real-world AWS labs, mentorship, and post-training support, the program ensures that participants gain both technical expertise and professional confidence. This dual emphasis on skill mastery and employability enables graduates to transition smoothly into the digital workforce and to continue advancing through AWS certifications and industry collaboration.

MolenGeek's ecosystem of learning, innovation, and employment reinforces the long-term success of AWS re/Start graduates. The integration of initiatives such as SideGeek — a digital platform developed by MolenGeek to connect alumni directly with employers and partners — exemplifies a forward-thinking model where education and employability are seamlessly connected. This community-driven approach not only strengthens professional outcomes but also cultivates a culture of collaboration, inclusion, and lifelong learning.

Beyond technical training, the program contributes to social mobility and economic inclusion. By equipping underrepresented and reskilling individuals with cloud and digital competencies, MolenGeek bridges the gap between talent and opportunity — fostering a more diverse, innovative, and future-ready tech workforce.

As MolenGeek continues to expand its impact across Europe and North Africa, the AWS re/Start program serves as a scalable model for inclusive cloud education — combining industry partnerships, accessible pedagogy, and a strong post-training network. Its success demonstrates that empowerment through education, when grounded in community and practice, is the key to unlocking hidden talent and shaping the next generation of cloud professionals.


aws re/start
Introduction to Linux
Linux Fundamentals

Linux Fundamentals

Lesson 1 – Welcome to the Shell

When you log in to a graphical Linux desktop you usually click an icon labelled **Terminal**. On servers you might SSH in from another machine. In either case the black-and-white window that appears is an *interactive shell session*. Historically Linux inherited the original UNIX **sh** shell, but today the overwhelmingly common default is **Bash** (the "Bourne-Again SHell").

Think of the shell as a combination of *command runner* and *mini programming language*. You enter a command, it executes, the kernel furnishes output, and the shell redraws a prompt so you can type the next instruction. Unlike a GUI where you must hunt for the right button, the CLI rewards recall: once a command sits in your memory you can launch it in milliseconds.

A typical prompt might read:

```
None
sam@example-pc:/home/sam$
```

- **sam** – your current username (use `whoami` to print it).
- **example-pc** – the system's network hostname (`hostname`).
- **/home/sam** – the *working directory* returned by `pwd`.
- The trailing `$` indicates an ordinary user; a root session shows `#`.

A command line often comprises three parts:

```
None
command [OPTIONS] [ARGUMENTS]
```

Take `ls -ahl /var/log`:

- **ls** is the program,
- **-a -h -l** are short options bundled together (show hidden files, human-readable sizes, long format),
- **/var/log** is the path argument specifying the target directory.

Options may also be long form (`--human-readable`). After execution every program exits with a *status code*. By convention **0** means success and any non-zero integer loosely represents an error class. You can inspect the previous command's status using `echo $?`.

Finally, two shell super-powers accelerate your workflow:

- **Tab completion** – Type the first few letters of a filename or command then press **Tab**; Bash completes or lists possibilities.
- **History** – Re-run past commands with the **Up/Down** arrows or search interactively with **Ctrl-r** followed by part of the command.

Exercise 1.1:

1. Open a terminal.
2. Run `echo $SHELL` to check which shell you are using.
3. Type `date` to print current system time, then inspect the exit status (`echo $?`).
4. Press **Ctrl-l** to clear the screen, then retrieve `date` from history using **Ctrl-r**.

Spend a few minutes experimenting with history and tab completion; you will use them constantly throughout the remainder of this course.

Lesson 2 – Navigating the Filesystem

The Linux filesystem resembles an upside-down tree. At the very top sits the **root directory** (`/`). All other directories—whether physical disks, USB sticks, network shares, or pseudo-filesystems like `/proc`—attach somewhere beneath this root.

Important top-level directories you will encounter frequently:

Directory	Purpose (simplified)

`/bin`	Essential user commands required for single-user mode
`/sbin`	System binaries (administration programs)
`/etc`	System-wide configuration files
`/home`	Personal directories for each user
`/var`	Variable data such as logs, caches, spool files
`/tmp`	Temporary files cleared on reboot or by housekeeping jobs
`/usr`	Userland software and libraries (gigantic hierarchy)

To discover where you are, type `pwd` (**p**rint **w**orking **d**irectory). New terminals open inside your home (`/home/yourname`).

Changing directory uses `cd`. Give it an **absolute path** (starts with `/`) or a **relative path** (interpreted from the current location):

```
None
cd /etc       # absolute
cd ../..      # relative: go up two levels
cd            # with no argument returns to $HOME
cd -          # jump back to previous dir
```

List directory contents with `ls`. Add options to reveal more information:

- `-l` long format (permissions, owner, size, date)
- `-a` include dotfiles (`.gitignore`)
- `-h` human-readable sizes (when used with `-l`)
- `-R` traverse sub-directories recursively

Combine them as short flags (`ls -lah`). Colourised output is enabled by default on most distros via aliases.

Filenames containing spaces must be quoted (`"My File.txt"`) or escaped (`My\ File.txt`). Instead, prefer small dash-separated names like `project-notes.txt`.

Globbing

Bash expands wildcards **before** the command executes:

- `*` any string (even empty)
- `?` any single character
- `[abc]` any one of the listed characters

Example:

```
None
ls /var/log/*.log
```

Productivity tips

- Hit **Tab** twice after typing `cd /u` to let the shell list all paths beginning with `/u`.
- Press **Ctrl-a** to jump to the line start, **Ctrl-e** to the end.
- Use `pushd /path` and `popd` as a stack-based alternative to `cd -`.

Exercise 2.1:

1. Navigate to `/usr/bin`, count how many regular files exist with `ls | wc -l`.
2. Return home, then create a directory tree `projects/linux101/week1` in a single command (`mkdir -p`).
3. Use globbing to list all `.conf` files under `/etc` that start with “ssh”.

Mastery of navigation and listing commands underpins every other CLI workflow. Practise until they feel automatic.

Lesson 3 – Inspecting Files and Directories

Once you can move around the filesystem the next step is *looking inside*. Linux distinguishes between **plain text** files, **binary** files, **directories** and **special** files (sockets, devices, named pipes). Many configuration and log files are just text, which makes them easy to read from the shell.

Quick Peeks

- `cat file` prints the entire file to standard output. Great for short snippets, risky for big logs.
- `less file` opens a paging viewer. Navigate with arrow keys, PgUp/PgDn, search `/pattern`, jump to end `G`, quit `q`.

None

```
less +F /var/log/syslog   # follow mode (like tail -f)
```

- `head` and `tail` show the first or last *n* lines (`-n 20`).
  Combining them is useful: `head -n 1 file && tail -n 1 file` reveals the earliest and latest entries.

Metadata

`stat file` prints explicit timestamps (access, modify, change), inode number, permissions and device id. Meanwhile the humble `ls -l` already exposes many of these fields along with hard link count and file size.

`file` performs *magic number* inspection rather than relying on filenames:

None

```
$ file /bin/ls
/bin/ls: ELF 64-bit LSB executable, x86-64, dynamically linked (...)
```

Counting and Summarising

The `wc` (*word count*) utility can tally lines, words and bytes. For example, to quickly approximate log growth rate:

None

```
watch -n5 'wc -l /var/log/auth.log'
```

Binary Data

Attempting to `cat` a compiled program spits gibberish to your terminal. Instead pipe through `xxd` (hex dump) or inspect with `strings` to pull out printable sequences—handy when reverse-engineering malware.

Exercises

1. Use `head -n 50 /etc/services` to view the first fifty network service definitions.
2. Run `stat ~/.bashrc` and identify when you last modified your shell configuration.
3. Determine if `/usr/bin/grep` is statically or dynamically linked (`file /usr/bin/grep`).
4. Create a 1 MiB dummy file: `dd if=/dev/zero of=blank.img bs=1M count=1` then verify size with `ls -lh` and bytes with `stat`.

Understanding how to inspect both content and metadata arms you with the diagnostic skills needed to debug misconfigurations, program behaviour and disk usage anomalies.

Lesson 4 – Manipulating Files and Directories

Creation, duplication, renaming and deletion are everyday tasks. Linux provides small, composable tools rather than a monolithic file manager.

Creating Things

- `touch file` updates timestamps but also creates an empty file when it does not yet exist.
- `mkdir dir` makes a directory. Add `-p` to create intermediate directories without complaint.

```
None
mkdir -p reports/2025/Q2
touch reports/2025/Q2/todo.md
```

Copying

`cp SOURCE DEST` duplicates files. Important options:

- `-r` or `-R` copy directories recursively,
- `-p` preserve mode, ownership and timestamps,
- `-a` archive mode (`-r -p --preserve=links` etc.), ideal for backups.

You can copy multiple items into a directory:

```
None
cp -av *.txt /tmp/backup/
```

Moving & Renaming

`mv` either relocates or renames. Unix treats renaming as moving within the same directory entry.

```
None
mv draft.txt final.txt          # rename
mv *.csv data/                  # move many files
mv old_project ../archive/old_project_$(date +%Y-%m-%d)
```

If DEST is an existing directory, SOURCE moves inside; if not, the file/directory changes its path.

Deleting

- `rm` removes files. `rm -r` removes directories recursively.
- `rm -i` prompts for confirmation; many admins alias `rm` to `rm -i` for safety.
- `rmdir` only deletes empty directories.

- Graphical trash semantics are implemented by **trash-cli** (`sudo apt install trash-cli`) which provides `trash-put`, `restore-trash` etc.

Archiving and Compression

The venerable **tar** packs multiple files into one archive. Add gzip or zstd compression via flags:

```
None
tar -czvf project.tar.gz project/
tar -tzf project.tar.gz            # list contents
tar -xzvf project.tar.gz -C /tmp   # extract elsewhere
```

Other compressors: `gzip`, `bzip2`, `xz`, `zstd`. Remember that these tools often operate *in place*: `gzip big.log` replaces the original file with `big.log.gz`.

Atomic Operations

Copying huge datasets can race with other processes. *rsync* solves this by transferring deltas and writing to temporary files before swapping them into place:

```
None
rsync -av --progress --delete /src/ /dest/
```

The trailing slashes matter (`/src/` syncs the *contents* whereas `/src` would create another subdirectory).

Scripting Patterns

It is common to chain manipulation commands via `&&` for short deployment scripts:

```
None
mkdir -p build && cp -a src/. build/ && tar -caf
build.tar.zst build
```

If any command fails (non-zero status) the chain aborts, preventing partial setups.

Exercises

1. Download the GNU Coreutils source tarball (`curl -LO https://ftp.gnu.org/gnu/coreutils/coreutils-9.5.tar.xz`) and extract it to `~/src`.
2. Copy only files modified within the last day to `~/src/today` using `find` combined with `cp --parents`.
3. Delete all `.o` object files from within the extracted tree, but first *dry-run* with `echo rm` substitution to verify which paths will vanish.

Handling files confidently allows you to automate packaging, backups and deployments without ever leaving the terminal. Practise until flags such as `-av`, `--delete`, `-p` and `-r` are second nature.

Lesson 5 – User and Group Permissions

Multitasking, multi-user systems like Linux rely on a robust permission model to isolate users and safeguard vital components. Understanding permission bits is crucial for both security and troubleshooting.

Ownership Triplet

Each filesystem object has:

1. **User owner** – usually the creator (`ls -l` column 3),
2. **Group owner** – determines additional access (`ls -l` column 4),
3. **Mode bits** – three sets of `rwx` flags:

```
None
r (read)  = 4
w (write) = 2
x (exec)  = 1
```

Matrix:

Scope	r	w	x
User	✔	✔	✔

Group	✔	✖	✖
Other	✔	✖	✖

The above corresponds to octal 744 (`4+2+1`, `4+0+0`). Displayed symbolically as `rwxr--r--`.

Changing Modes

`chmod` modifies the mode:

```
None
chmod u+x script.sh          # add execute for user
chmod go-w file              # remove write for
group+other
chmod 600 secrets.gpg        # read/write for owner only
```

Recursive flag `-R` changes whole trees—use sparingly.

Special Bits

- SUID (4 000) – executable runs with *file owner* privileges. Example: `/usr/bin/passwd`.
- SGID (2 000) – executable inherits *group* privileges; when set on directories newly created files adopt the directory's group.
- Sticky (1 000) – on directories allows only owners to delete inside them (`/tmp`).

```
None
chmod 4755 mysetuidhelper
```

Changing Ownership

`chown user:group target`. Omitting group keeps it unchanged. Recursive variant `-R` is common when un-archiving external projects:

```
None
sudo chown -R $USER:$USER ~/src/coreutils
```

Elevating Privileges

Editing `/etc`, installing kernel modules and binding to ports <1024 all require *superuser* rights. Modern distros favour **sudo** over logging in as root:

```
None
sudo apt update
sudoedit /etc/fstab    # uses $EDITOR safely
```

Admin accounts are enumerated in `/etc/sudoers` via the `%sudo` group. Limit membership wisely.

Debugging Permission Denials

When a command returns `Permission denied`:

1. Check path permissions with `ls -ld path`.
2. Use `` `namei -l /path´ ``

aws re/start

Introduction to networking

Networking Fundamentals

Module 1 – What Is a Network?

A **network** is any collection of two or more nodes that exchange information through some medium. Humans have created networks for millennia—smoke signals, semaphore, telegraph—but the modern computer network traces its ancestry to the ARPANET of 1969. At its heart, a network solves three problems:

- **Connectivity:** physically linking devices.
- **Addressing:** uniquely identifying endpoints.
- **Delivery:** moving data from sender to receiver.

Networks scale from a personal area network (PAN) on your smartwatch to the globe-spanning Internet. They can be wired, wireless, or a hybrid; public, private, or virtual. Regardless of size they all rely on layered abstractions to hide complexity. You needn't know which fiber routes your emoji across the ocean; the layers below handle it. As you work through the course, notice how each layer adds specific capabilities while depending on the services of the layer underneath.

Analogy – The City Road System

Think of each device as a house and the medium as the asphalt. An address painted on the mailbox allows the courier (the network) to find the correct door, while traffic

signs (protocols) ensure cars don't crash. If the road washes out, letters never arrive—mirroring cable cuts or wireless dead zones.

Key Terms
`Node`, `Host`, `Medium`, `Bandwidth`, `Latency`, `Topology`.

Mini-Lab
Draw a simple map connecting two laptops and a smartphone to a home router. Label the medium (Ethernet, Wi-Fi) and mark where addressing and delivery occur.

Module 2 – The OSI and TCP/IP Reference Models

In 1984 the International Organization for Standardization (ISO) published the **Open Systems Interconnection (OSI)** model—seven conceptual layers that describe how data moves from application to physical wire:

1. **Application**
2. **Presentation**
3. **Session**
4. **Transport**
5. **Network**
6. **Data Link**
7. **Physical**

In practice, the simpler four-layer **TCP/IP** model dominates real-world implementations:

- **Application** (OSI 5-7)
- **Transport** (OSI 4)
- **Internet** (OSI 3)
- **Link** (OSI 1-2)

Memorization is less important than internalizing *responsibility boundaries*. Applications should not care about routing decisions, and routers should not parse HTTP payloads. Understanding these separations enables efficient troubleshooting: if packets leave Layer 4 but never reach Layer 3 on the far side, you investigate routing, not the web server.

Analogy – Postal Service Layers

A birthday card begins at the kitchen table (Application), is placed into an envelope (Presentation), given to a local courier (Session/Transport), carried by a regional sorting center (Network), loaded onto a truck (Data Link), and finally driven on asphalt (Physical). Each layer only needs to know how to hand the card to the layer below it.

Key Terms

`Encapsulation`, `Decapsulation`, `PDU`, `Service Access Point`.

Mini-Lab

Use Wireshark to capture packets while browsing a website. Expand a single frame and observe how Ethernet, IP, TCP, and HTTP headers nest like Russian dolls.

Module 3 – The Physical Layer (Layer 1)

Layer 1 concerns the **raw transmission of bits**. Its parameters include voltage levels, light wavelengths, radio frequencies, pinouts, and connector types. Common media and their approximate speed ceilings (as of 2025):

- **Unshielded Twisted Pair (UTP)** – Cat 5e (1 Gb/s, 100 m), Cat 6 (10 Gb/s, 55 m), Cat 6A (10 Gb/s, 100 m)
- **Fiber Optic** – Multimode OM4 (100 Gb/s, 150 m), Single-mode OS2 (>400 Gb/s, tens of km)
- **Wireless 802.11ax (Wi-Fi 6/6E)** – 9.6 Gb/s theoretical, real-world 1-2 Gb/s

Topologies describe how nodes connect: **bus**, **star**, **ring**, **mesh**. Today's Ethernet LANs favor a star—each endpoint home-runs to a switch. Note that Layer 1 includes repeaters and media converters but not switches (Layer 2) or routers (Layer 3).

Signal degradation (attenuation), noise (crosstalk), and interference (EMI/RFI) can corrupt bits. Engineers mitigate these with shielding, proper grounding, and error-detection codes at higher layers.

Real-World Story

In 2023 Meta's data center in Gallatin, Tennessee suffered intermittent packet loss traced to bent Cat 6 patch cables that exceeded the 100 meter standard by being unofficially "extended" through two jumpers. Re-crimping and replacing the run restored full 10 Gb/s throughput.

Key Terms
`Attenuation`, `Signal-to-Noise Ratio (SNR)`, `Plenum`, `Simplex`, `Duplex`.

Mini-Lab
Borrow a cable tester and verify the pinout of two patch leads. Note the difference between straight-through and crossover wiring.

Module 4 – Data Link Layer (Layer 2)

Layer 2 packages raw bits into **frames** and handles **node-to-node delivery** on a single broadcast domain. Ethernet dominates wired LANs; 802.11 governs WLANs. Each network interface has a 48-bit **MAC address** (e.g., *00-1A-2B-3C-4D-5E*) burned into firmware. Switches learn which MAC lives on which port and build a **MAC address table** to forward frames intelligently.

Important Layer 2 concepts:

- **Collision Domains** – modern full-duplex switches eliminate collisions, but hubs did not.
- **VLANs (802.1Q)** – logical segmentation atop the same physical switch fabric.
- **Spanning Tree Protocol (STP)** – prevents loops when redundant links exist.
- **Link Aggregation (LACP)** – combines multiple physical links for higher throughput and resilience.

Wireless adds complexity: access points mediate contention via CSMA/CA, deal with roaming clients, and implement security suites like WPA3.

Real-World Story
A finance firm once connected two redundant core switches without STP. During a firmware update both reacted to a sudden topology change, forming a switching loop that flooded the VLAN and halted trades for thirteen minutes. Enabling Rapid PVST+ and setting one switch as the root avoided a repeat.

Key Terms
`Frame`, `MAC Address`, `Broadcast`, `STP`, `BPDU`, `VLAN Tag`.

Mini-Lab
Configure two switches and create VLAN 10 and VLAN 20. Connect a laptop to each

VLAN and demonstrate that they cannot ping each other until you add an 802.1Q trunk to a router-on-a-stick.

Module 5 – Network Layer (Layer 3)

Layer 3 provides **logical addressing and routing**. The most common protocol is **Internet Protocol version 4 (IPv4)**, a 32-bit address space that will eventually give way to **IPv6** with 128 bits. Key terms:

- **Subnet Mask / Prefix Length** – divides network and host portions.
- **Default Gateway** – the router that relays packets to other networks.
- **Routing Table** – a set of rules that map destinations to next hops.
- **ICMP** – diagnostics and error reporting (e.g., Echo Request/Reply used by *ping*).

Routers consult their tables to choose the best path. They decrement each packet's **Time-to-Live (TTL)** to prevent loops; when TTL hits zero, an ICMP Time Exceeded message is sent back.

Subnetting lets administrators right-size networks. For example, 192.168.10.0/24 provides 254 host addresses, whereas /30 supports only two—ideal for point-to-point links.

Analogy – GPS Navigation

When you drive to a new city you type the street address (IP) into a GPS, which selects one of many possible highways (routes). If a bridge is closed, the GPS recalculates—similar to dynamic routing protocols adapting to failures.

Key Terms

`CIDR`, `Prefix`, `Default Gateway`, `TTL`, `ARP`, `MTU`.

Mini-Lab

Calculate the subnet mask needed to support 62 hosts and implement it on two routers that share OSPF. Verify with `show ip route`.

Module 6 – Transport Layer (Layer 4)

The Transport layer end-to-end delivers segments between processes (ports) on host devices. Two primary protocols:

- TCP (Transmission Control Protocol)
  - Connection-oriented (three-way handshake)
  - Reliable delivery via sequence numbers, acknowledgments, and retransmissions
  - Flow control (window size) and congestion avoidance (e.g., Cubic, BBR)
- UDP (User Datagram Protocol)
  - Connectionless, minimal overhead
  - No reliability guarantees, but faster and ideal for real-time (VoIP, video)

Ports identify services: 80 and 443 for HTTP/HTTPS, 53 for DNS, 22 for SSH. *Netstat* or *ss* lists active sockets; *iptables* or *nftables* can filter them.

Real-World Story
Netflix's custom congestion control, BBR, improved global streaming throughput by 14 % versus CUBIC in 2024, illustrating how kernel-level tweaks at Layer 4 translate into happier viewers at Layer 7.

Key Terms
`Three-Way Handshake`, `Window Size`, `Retransmission Timeout`, `Checksum`, `MUX`, `Demux`.

Mini-Lab
Open two terminal windows. In the first run `iperf3 -s`. In the second run `iperf3 -c <server>` twice: once with `--udp` and once default (TCP). Compare throughput and packet loss.

Module 7 – Application Layer (Layers 5-7 in OSI)

This top layer houses the protocols humans interact with. Highlights:

Protocol	Purpose	Default Port
HTTP / HTTPS	Web browsing, APIs	80 / 443
DNS	Name resolution	53 (UDP/TCP)
SMTP, IMAP, POP3	Email transport and retrieval	25, 143, 110

FTP / SFTP	File transfer	21 / 22
DHCP	Automatic IP assignment	67/68
SNMP	Device management	161

Many application protocols now ride atop **TLS** for encryption—"HTTPS" is merely HTTP over TLS. Packet analyzers reveal the handshake but not the ciphertext inside.

Analogy – Restaurant Menu

Ordering in a restaurant uses a well-understood vocabulary: "medium-rare steak" means the same worldwide. Protocols are menus for computers; DNS is the maître d' translating human names to kitchen tickets (IP addresses).

Key Terms

`URI`, `TLS Handshake`, `MX Record`, `REST`, `Stateful`, `Stateless`.

Mini-Lab

Use `dig` to query the TXT records of google.com, then fetch a web page with `curl -v https://example.com` and interpret the TLS handshake.

Module 8 – Network Devices and Infrastructure

- **Hub** – obsolete Layer 1 repeater.
- **Switch** – Layer 2 device that forwards frames based on MAC addresses.
- **Router** – Layer 3 device that routes packets between networks.
- **Wireless Access Point (AP)** – bridges 802.11 wireless to wired LANs.
- **Firewall** – inspects traffic against policy, ranging from stateless ACLs to NGFWs with Layer 7 inspection.
- **Load Balancer** – distributes traffic across server pools for scale and resilience.
- **Gateway** – performs protocol translation (e.g., VoIP gateways).

Choosing gear involves throughput, latency, port density, PoE requirements, redundancy, and budget.

Real-World Story

In February 2025 Reddit scaled to 200 million daily active users by replacing hardware load balancers with an Envoy-based software mesh, cutting latency 18 ms at the 95th percentile.

Key Terms
`Throughput`, `Forwarding Rate`, `PoE`, `Backplane`, `ASIC`, `Control Plane`.

Mini-Lab
Rack a small lab: home router, 8-port switch, Wi-Fi AP, and Raspberry Pi firewall. Label where each OSI layer's primary function happens.

Module 9 – Network Address Translation (NAT) and Private IPs

IPv4 exhaustion forced creative reuse of address space. **RFC 1918** reserves three private ranges:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

These addresses are **not routable on the public Internet**. Instead, routers use **Network Address Translation (NAT)** to map internal private IPs to a single public IP. The most common variant is **PAT (Port Address Translation)**, where many devices share one external IP, distinguished by port numbers.

When a user inside a private network makes an outbound request (e.g., to a website), the NAT device records the mapping in a translation table. Incoming replies are matched and delivered to the correct internal host. This method conserves public addresses and adds a layer of basic firewall-like isolation.

Types of NAT:

- **Static NAT**: 1:1 mapping of internal to external IP
- **Dynamic NAT**: maps from a pool of public IPs
- **PAT (NAT Overload)**: many-to-one using ports

While NAT solves address exhaustion, it complicates protocols that embed IPs in payloads (e.g., SIP, FTP) and breaks true end-to-end connectivity. Solutions include **Application Layer Gateways (ALGs)** or transitioning to **IPv6**, where every device can have a globally unique address.

Real-World Story
A university lab deployed 200 Raspberry Pi devices behind a NAT. Due to misconfigured port forwarding, only one device could be remotely accessed at a time.

Switching to IPv6 with unique addressing and proper firewall rules solved the scalability issue.

Key Terms
`RFC 1918`, `PAT`, `Static NAT`, `Dynamic NAT`, `ALG`, `Carrier-Grade NAT (CGNAT)`

Mini-Lab
Configure PAT on a home router. Use one device to browse to `whatismyip.com` and note the shared public IP. On another device, open a connection and verify it uses the same external address.

Module 10 – Firewalls and Security Basics

Firewalls control traffic between network segments. At their core, firewalls enforce rules that allow or deny packets based on IP addresses, ports, and protocols.

Types of firewalls:

- **Packet-filtering firewalls**: basic rules at Layer 3 and 4 (IP, TCP/UDP)
- **Stateful firewalls**: track active connections
- **Next-Gen Firewalls (NGFW)**: include deep packet inspection, intrusion detection/prevention, and application awareness

Basic security principles include:

- **Default Deny**: Only explicitly allowed traffic is permitted
- **Least Privilege**: Grant only the access needed
- **Defense in Depth**: Use multiple overlapping security layers

Real-World Story
A company exposed a printer's web management interface to the Internet without a firewall rule. Within hours, it was added to a botnet. Implementing a firewall with proper deny rules stopped the intrusion.

Key Terms
`Stateful`, `ACL`, `NGFW`, `DPI`, `DMZ`, `Default Deny`

Mini-Lab
Set up a basic firewall rule on a home router to block all inbound traffic except port 443. Try accessing the router from another network.

Module 11 – Wireless Networking Fundamentals

Wireless networking uses **radio waves** to connect devices without physical cables. The most common standard is **IEEE 802.11** (Wi-Fi).

Frequency bands:

- **2.4 GHz**: longer range, more interference
- **5 GHz**: faster speeds, shorter range

Security standards:

- **WEP** (obsolete)
- **WPA/WPA2** (still common)
- **WPA3** (modern, recommended)

Wireless networks use a central device called an **Access Point (AP)**. APs broadcast an **SSID** (network name) and handle authentication.

Real-World Story

A café allowed open Wi-Fi without encryption. An attacker used a packet sniffer to capture login credentials. Switching to WPA2 and disabling SSID broadcast reduced unauthorized access.

Key Terms

`SSID`, `WPA3`, `802.11`, `Access Point`, `MAC Filtering`, `Wi-Fi 6`

Mini-Lab

Use a Wi-Fi analyzer app to survey available networks. Note the channel, frequency, and signal strength. Identify overcrowded channels.

Module 12 – IPv6 and the Future of Networking

IPv6 was designed to replace IPv4 and solve address exhaustion. It uses 128-bit addresses, allowing for 340 undecillion unique addresses.

An IPv6 address looks like: `2001:0db8:85a3:0000:0000:8a2e:0370:7334`
(Zeroes can be compressed: `2001:db8:85a3::8a2e:370:7334`)

Benefits:

- No NAT needed — every device can have a unique global IP
- Simplified header for faster processing
- Built-in security with **IPSec**

Adoption has been slow due to legacy systems and lack of demand, but is growing as ISPs and cloud providers support IPv6.

Real-World Story

A major streaming platform moved to IPv6 for mobile clients to reduce latency and avoid NAT traversal issues. This improved performance during peak usage.

Key Terms

`IPv6`, `CIDR`, `Stateless Autoconfiguration`, `Link-local`, `Global Unicast`, `EUI-64`

Mini-Lab

Enable IPv6 on your device and browse to `ipv6-test.com` or `test-ipv6.com` to see if you're using it. Note the address format.

Appendix – Networking Commands Cheat Sheet

- `ipconfig / ifconfig`: show IP config
- `ping`: test reachability
- `traceroute / tracert`: path to a host
- `nslookup / dig`: DNS queries
- `netstat`: active connections
- `tcpdump / Wireshark`: packet capture
- `arp`: view address resolution cache

Glossary

- **Router** – Directs packets between networks
- **Switch** – Forwards frames based on MAC addresses
- **Firewall** – Filters traffic based on rules
- **IP Address** – Logical address for a device
- **MAC Address** – Physical hardware identifier
- **DHCP** – Automatically assigns IPs
- **DNS** – Translates names to IPs
- **NAT** – Translates internal to external IPs
- **SSID** – Wireless network name
- **CIDR** – Notation for IP ranges
- **Port** – Endpoint for communication (e.g., TCP 80 for HTTP)

Introduction to Programming

Python Fundamentals

Python

Part 1: Introduction to Python

1.1 What is Python?

Python is a **high-level, interpreted programming language** known for its readability and simplicity. It was created by **Guido van Rossum** and first released in **1991**. Python emphasizes code readability with its clean and straightforward syntax, making it ideal for beginners.

Python is used in various fields, including:

- Web development (e.g., Django, Flask)
- Data science and machine learning (e.g., pandas, scikit-learn)
- Automation and scripting
- Game development
- Cybersecurity and ethical hacking
- Desktop applications
- Internet of Things (IoT)

Why Learn Python?

- **Beginner-friendly** syntax
- **Large community** and vast libraries
- Widely used in **job markets**
- Great for both small scripts and large applications

- **Cross-platform** (Windows, MacOS, Linux)

1.2 How Python Works

Unlike compiled languages (like C or Java), Python is an **interpreted language**. This means:

- You write Python code in a `.py` file.
- The Python interpreter reads and executes the code **line by line**.
- This allows for quicker development and debugging.

Python also supports two programming paradigms:

1. **Procedural Programming** – writing functions and procedures.
2. **Object-Oriented Programming** – organizing code using classes and objects.

1.3 Installing Python

Python comes pre-installed on most Linux and macOS systems, but Windows users need to install it manually.

Step-by-Step Installation (Windows):

1. Go to https://www.python.org/downloads/
2. Download the latest version (e.g., Python 3.12.x).
3. **Important:** During installation, check the box that says **"Add Python to PATH"**.
4. Click "Install Now" and follow the instructions.

Verifying Installation

Open a terminal or command prompt and type:

```shell
python --version
```

If installed correctly, you'll see something like:

```none
Python 3.12.0
```

You can also open the interactive Python shell by typing:

```shell
python
```

You'll see the Python prompt:

```python
>>>
```

Here, you can write Python code directly.

1.4 Setting Up a Development Environment

You can write Python code in multiple ways. Here are three common options:

Option 1: IDLE

Python ships with **IDLE**, its own basic editor. You can open it from your start menu or applications folder. It's simple and great for beginners.

Option 2: Text Editor (VS Code)

Visual Studio Code (VS Code) is a free, powerful code editor that supports Python. It features:

- Syntax highlighting
- IntelliSense (auto-completion)
- Extensions for debugging, linting, etc.

Install the **Python extension** in VS Code after installing Python.

Option 3: Jupyter Notebook (Optional for Data Science)

Jupyter lets you write code in cells and see outputs instantly—great for data science.

Install with:

```shell
pip install notebook
```

Then run:

```shell
jupyter notebook
```

1.5 Writing Your First Python Program

Let's write the classic "**Hello, World!**" program.

Option A: In the Interactive Shell

Open your terminal or command prompt and type:

```python
print("Hello, World!")
```

You'll see:

```
None
Hello, World!
```

Option B: In a Python File

1. Open your text editor or IDLE.
2. Create a new file and save it as `hello.py`.
3. Type the following code:

```
Python
print("Hello, World!")
```

4. Save the file.
5. Run it from the terminal:

```
Shell
python hello.py
```

You'll get the same output.

1.6 Understanding the `print()` Function

The `print()` function is used to display output on the screen. It takes one or more arguments and prints them.

Examples:

```python
print("Welcome to Python!")
print("The answer is", 42)
print("Hello", "World", sep="-")
```

Output:

```
Welcome to Python!
The answer is 42
Hello-World
```

- `sep` defines the separator between values (default is a space).
- You can also format strings using f-strings:

```python
name = "Alice"
print(f"Hello, {name}!")
```

1.7 Comments in Python

Comments are notes in your code that are not executed. They help explain what your code does.

Single-line comment:

```python
# This is a comment
print("Hello")  # This is also a comment
```

Multi-line comment (technically a string):

```python
"""
This is a multi-line comment.
Useful for documentation.
"""
```

1.8 Errors and Debugging

Python has two main types of errors:

- **Syntax Errors**: Mistakes in the structure of code.

```python
print("Hello   # Missing closing quote
```

- **Runtime Errors**: Errors that occur while the program is running.

```python
print(1 / 0)  # Division by zero
```

Python's error messages are helpful. Pay attention to:

- The **line number**
- The **error type** (e.g., `SyntaxError`, `ZeroDivisionError`)
- A **description** of the problem

Tip: Read errors carefully! They often tell you exactly what went wrong.

1.9 Python Versions: Python 2 vs Python 3

Python 2 is **deprecated** and no longer supported. Always use **Python 3**.

Key differences:

- `print` is a **function** in Python 3: `print("Hello")`
- Integer division behaves differently:
    - Python 2: `5 / 2 = 2`
    - Python 3: `5 / 2 = 2.5`

1.10 How to Practice Python

Here are some great ways to practice:

1. Online Interpreters:

- Repl.it
- Programiz
- Google Colab (great for data science)

2. Challenges for Beginners:

- Print your name, age, and favorite color.
- Write a script that calculates your age in months.
- Create a greeting program that takes a name and prints `Hello, <name>`.

1.11 Summary

In this lesson, you learned:

✅ What Python is and why it's useful
✅ How to install Python and set up your environment
✅ How to write and run your first Python script
✅ The `print()` function and using comments
✅ Basic error handling and debugging
✅ Python versions and how to practice

Practice Tasks

Try solving these to reinforce what you've learned:

1. Write a program that prints your name and age.

2. Write a program that prints:

```
None
Welcome to Python!
Let's start coding.
```

3. Write a program that prints the result of `5 + 10`, `5 * 10`, and `10 / 2`.

4. Use `print()` to display:

```
None
Name: Alice
Age: 25
Hobby: Reading
```

Part 2: Variables and Data Types

2.1 What is a Variable?

A **variable** is like a container that holds data. You give it a name, and you can store or retrieve information from it at any time during your program.

Think of it like this:

```python
box = "apple"
```

Now, `box` contains the word `"apple"`.

Why use variables?

- To store data temporarily
- To make code readable and maintainable
- To perform calculations or logic using stored values

2.2 Creating Variables in Python

You don't need to declare a variable's type. Python is **dynamically typed**, which means it figures out the type based on the value you assign.

Example:

```python
name = "Alice"
age = 25
```

```
height = 5.7
is_student = True
```

Naming Rules:

- Must start with a letter or underscore `_`
- Can contain letters, digits, and underscores
- **Cannot** start with a number
- **Case-sensitive** (`Name` and `name` are different)

Valid examples:

```python
my_name = "John"
_age = 20
year2025 = 2025
```

Invalid examples:

```python
2cool = "no"        # Starts with number ❌
my-name = "John"    # Hyphen not allowed ❌
```

2.3 Data Types in Python

Python has several **built-in data types**. The most common ones for beginners are:

Type	Example	Description
`int`	`10`, `-5`	Whole numbers

`float`	`3.14`, `-0.01`	Decimal numbers
`str`	`"hello"`	Text (strings)
`bool`	`True`, `False`	Boolean (yes/no, on/off)

2.4 Type Checking

Use the `type()` function to check what type a variable is:

Python
```
x = 42
print(type(x))  # <class 'int'>
```

Python
```
y = "Hello"
print(type(y))  # <class 'str'>
```

2.5 Working with Strings

Strings are **sequences of characters** enclosed in either single or double quotes.

Python
```
message = "Hello, World!"
```

Common String Operations:

Python
```
# Concatenation
```

```python
first = "Hello"
second = "World"
print(first + " " + second)  # Hello World

# Repetition
print("Hi! " * 3)  # Hi! Hi! Hi!

# Length
print(len("Python"))  # 6

# Indexing (starts at 0)
text = "Python"
print(text[0])  # P
print(text[-1])  # n

# Slicing
print(text[0:3])  # Pyt
print(text[2:])   # thon
```

2.6 String Methods

Python gives you useful built-in functions (called **methods**) for working with strings.

Python

```python
name = "alice"

print(name.upper())    # ALICE
print(name.capitalize())  # Alice
print(name.isalpha())  # True
print("123".isdigit()) # True
```

2.7 Working with Numbers

Integers (`int`) and Floating-point numbers (`float`):

```python
x = 10        # Integer
y = 3.14      # Float
z = -5        # Negative integer
```

Arithmetic Operations:

Operator	Meaning	Example
`+`	Addition	`2 + 3 = 5`
`-`	Subtraction	`5 - 1 = 4`
`*`	Multiplication	`4 * 2 = 8`
`/`	Division (float)	`5 / 2 = 2.5`
`//`	Floor Division	`5 // 2 = 2`
`%`	Modulus (remainder)	`5 % 2 = 1`
`**`	Exponentiation	`2 ** 3 = 8`

2.8 Type Conversion (Casting)

Sometimes, you need to convert between data types. Python provides built-in functions:

Convert to string:

```python
age = 25
print("Age: " + str(age))  # Age: 25
```

Convert to integer:

```python
num = "10"
print(int(num) + 5)  # 15
```

Convert to float:

```python
value = "3.14"
print(float(value) + 1)  # 4.14
```

Convert to boolean:

```python
print(bool(0))    # False
print(bool(1))    # True
print(bool(""))   # False
print(bool("Hi")) # True
```

2.9 Input from the User

Use the `input()` function to get user input. It always returns a **string**, so you may need to convert it.

```python
name = input("Enter your name: ")
```

```
print("Hello, " + name)
```

Python
```
age = int(input("Enter your age: "))
print("In 5 years, you'll be", age + 5)
```

2.10 Constants

Python doesn't have true constants, but by convention, we use **uppercase variable names** to indicate values that should not change:

Python
```
PI = 3.14159
GRAVITY = 9.8
```

While these can still be changed, developers treat them as constants by agreement.

2.11 Multiple Assignments

Python allows you to assign values to multiple variables in one line:

Python
```
a, b, c = 1, 2, 3
print(a, b, c)  # 1 2 3

x = y = z = 0
print(x, y, z)  # 0 0 0
```

2.12 Best Practices

- Use **descriptive names**: `score`, `user_name`, `price`
- Use **snake_case** for variable names (`my_variable`)
- Avoid reserved words like `class`, `if`, `print` as variable names
- Add comments to explain your code

Practice Tasks

Try solving these to test your understanding:

1. Store your name, age, and favorite color in variables and print them.
2. Write a program that asks the user for two numbers and prints their sum.
3. Write a program that calculates the area of a rectangle.
4. Ask the user for their birth year and calculate their age.
5. Convert a string like `"42"` to an integer and multiply it by 2.

Summary

In this part, you learned:

✅ What variables are and how to use them
✅ Python's basic data types: `int`, `float`, `str`, `bool`
✅ How to work with and manipulate strings and numbers
✅ How to convert between data types
✅ How to take user input
✅ Best practices for writing clean and clear code

Part 3: Operators in Python

3.1 What Are Operators?

Operators are **symbols or keywords** in Python that perform operations on values or variables. They're essential in writing logic, performing calculations, and comparing data.

Example:

```python
a = 10

b = 5

print(a + b)  # Output: 15
```

Python has several categories of operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Identity and Membership Operators (covered later in advanced sections)

Let's explore each one step-by-step.

3.2 Arithmetic Operators

These are used for mathematical operations:

Operator	Description	Example	Result
`+`	Addition	`3 + 2`	`5`
`-`	Subtraction	`5 - 2`	`3`
`*`	Multiplication	`4 * 3`	`12`
`/`	Division (float)	`10 / 3`	`3.3333`
`//`	Floor Division	`10 // 3`	`3`
`%`	Modulus (remainder)	`10 % 3`	`1`
`**`	Exponentiation	`2 ** 3`	`8`

Examples:

Python

```
a = 15

b = 4
```

```
print(a + b)   # 19
print(a - b)   # 11
print(a * b)   # 60
print(a / b)   # 3.75
print(a // b)  # 3
print(a % b)   # 3
print(a ** b)  # 50625
```

3.3 Assignment Operators

These are used to **assign** values to variables and to **update** them.

Operator	Example	Same as
`=`	`x = 5`	`x = 5`
`+=`	`x += 2`	`x = x + 2`
`-=`	`x -= 3`	`x = x - 3`

*=	x *= 4	x = x * 4
/=	x /= 2	x = x / 2
//=	x //= 2	x = x // 2
%=	x %= 3	x = x % 3
**=	x **= 2	x = x ** 2

Example:

```python
x = 10

x += 5   # x = 15

x *= 2   # x = 30

print(x) # 30
```

These save time and make code cleaner.

3.4 Comparison Operators (Relational Operators)

These compare two values and return a **boolean** result (`True` or `False`).

Operator	Description	Example	Result
`==`	Equal to	`5 == 5`	`True`
`!=`	Not equal to	`5 != 3`	`True`
`>`	Greater than	`10 > 7`	`True`
`<`	Less than	`5 < 10`	`True`
`>=`	Greater than or equal	`7 >= 7`	`True`
`<=`	Less than or equal	`3 <= 4`	`True`

Example:

Python

```
a = 8

b = 12


print(a == b)   # False

print(a != b)   # True
```

```
print(a > b)    # False
print(a <= b)   # True
```

These are often used in **if statements** and loops.

3.5 Logical Operators

Logical operators allow you to **combine conditions** or invert them. They return boolean values.

Operator	Description	Example
`and`	True if both are true	`True and True → True`
`or`	True if at least one is true	`True or False → True`
`not`	Inverts the result	`not True → False`

Examples:

Python

```
x = 10
print(x > 5 and x < 20)   # True
```

```
print(x < 5 or x > 8)      # True
print(not(x == 10))        # False
```

Use logical operators when you want to evaluate **multiple conditions**.

3.6 Order of Operations (PEMDAS in Python)

Python follows an **order of precedence** for evaluating expressions:

1. Parentheses `()`
2. Exponents `**`
3. Multiplication / Division / Modulus `* / // %`
4. Addition / Subtraction `+ -`
5. Comparison Operators `==, !=, >, <, >=, <=`
6. Logical NOT `not`
7. Logical AND `and`
8. Logical OR `or`

Example:

Python

```
result = 3 + 4 * 2 ** 2

# Step-by-step:
```

```
# 2 ** 2 = 4

# 4 * 4 = 16

# 3 + 16 = 19

print(result)  # 19
```

Use parentheses to make your logic clearer and avoid mistakes.

3.7 Combining Operators in Real Examples

Example 1: Check if a number is within a range

Python

```
x = 25

print(x >= 10 and x <= 30)  # True
```

Example 2: Check if a password is valid

Python

```
password = "abc123"

print(len(password) >= 6 and " " not in password)  # True
```

Example 3: Voting eligibility

Python

```
age = int(input("Enter your age: "))

citizen = input("Are you a citizen? (yes/no): ").lower()
```

```
if age >= 18 and citizen == "yes":
    print("You are eligible to vote.")
else:
    print("Sorry, you're not eligible.")
```

3.8 Common Mistakes with Operators

Mistake	Why it's wrong	Correct version
`=` instead of `==`	`=` is assignment, not comparison	`if x == 5:`
Using `and` instead of `or`	Logic error	Check conditions carefully
Forgetting parentheses	Changes evaluation order	Use `()` to group conditions

---

Practice Time!

Beginner Challenges:

1. Write a program that checks if a number is even or odd using %.
2. Ask the user for a number and check if it is between 1 and 100.
3. Create a calculator that performs basic arithmetic operations.
4. Write a program that checks if two numbers are both divisible by 3.
5. Ask for a user's name and age, and display if they're old enough to drive (age >= 18).

Mini Project: Simple Grade Evaluator

Write a program that asks for a student's test score and prints their grade based on the following:

Score Range	Grade
90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
Below 60	F

Example:

```python
score = int(input("Enter your score: "))


if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
elif score >= 60:
    print("Grade: D")
else:
    print("Grade: F")
```

Try modifying it to include validation (e.g., scores between 0–100 only).

Summary

In this part, you learned about:

✅ **Arithmetic operators** for math operations
✅ **Assignment operators** for updating values
✅ **Comparison operators** to compare data
✅ **Logical operators** for combining conditions
✅ **Operator precedence** and best practices

Part 4: Control Flow (Making Decisions in Code)

4.1 What is Control Flow?

Control flow allows your program to **make decisions** and **choose different paths** based on certain conditions. It's how your code can act **intelligently**.

Without control flow, a program just runs straight from top to bottom. With it, you can:

- Respond to user input
- Handle different situations
- Skip or repeat code

4.2 The `if` Statement

The most basic control structure is the `if` statement. It lets you run code **only if** a condition is `True`.

Syntax:

```python
if condition:
    # code block
```

Example:

```python
age = 18


if age >= 18:
    print("You're an adult.")
```

If the condition is `True`, the indented code runs. If not, it's skipped.

4.3 `if...else` Statement

Sometimes you want to run one block of code **if a condition is true**, and a different one **if it's false**.

Syntax:

```python
if condition:
    # true block
else:
    # false block
```

Example:

```python
temperature = 25


if temperature > 30:
```

```
    print("It's hot today!")
else:
    print("It's not that hot.")
```

4.4 `if...elif...else`

Use `elif` (short for “else if”) to test multiple conditions in order.

Syntax:

Python

```
if condition1:
    # runs if condition1 is true
elif condition2:
    # runs if condition1 is false, and condition2 is true
else:
    # runs if none of the above are true
```

Example:

Python

```
score = 75


if score >= 90:
```

```
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: D or F")
```

Output:

```
None
Grade: C
```

Note: Only the **first true** condition is executed.

4.5 Nested `if` Statements

You can put one `if` block **inside another**. This is useful when one condition depends on another.

Example:

```
Python
age = 20

has_license = True
```

```
if age >= 18:
    if has_license:
        print("You can drive.")
    else:
        print("You need a license to drive.")
else:
    print("You're too young to drive.")
```

This checks both age and whether the person has a license.

4.6 Logical Operators with `if`

You can combine conditions using:

- `and`: both must be true
- `or`: at least one must be true
- `not`: flips the truth value

Example:

Python

```
username = "admin"
password = "1234"
```

```
if username == "admin" and password == "1234":
    print("Access granted.")
else:
    print("Access denied.")
```

4.7 Comparison and Boolean Recap

You can use any expression that returns a boolean (`True` or `False`) in `if` statements:

Python
```
# Examples of conditions
x == 10
x != 5
x > 0
len(name) > 3
age >= 18 and citizen == "yes"
```

4.8 Truthy and Falsy Values

Python treats some values as automatically `True` or `False`.

Falsy values:

- `0`

- `0.0`
- `""` (empty string)
- `[]`, `{}`, `set()` (empty collections)
- `None`
- `False`

Everything else is **truthy**.

Example:

```python
name = ""


if name:
    print("Hello,", name)
else:
    print("Please enter your name.")
```

4.9 Indentation Is Critical

Python uses **indentation** (spaces or tabs) to define code blocks.

Example:

```python
x = 5
```

```
if x > 0:
    print("Positive")
    print("Still in if")
print("Out of if")
```

Make sure all code inside the `if` is **indented equally**.

4.10 Practical Examples

Example 1: Even or Odd

Python

```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

Example 2: BMI Calculator

Python

```python
weight = float(input("Enter your weight (kg): "))

height = float(input("Enter your height (m): "))


bmi = weight / (height ** 2)


if bmi < 18.5:

    print("Underweight")

elif bmi < 25:

    print("Normal weight")

elif bmi < 30:

    print("Overweight")

else:

    print("Obese")
```

Example 3: Leap Year Checker

Python

```python
year = int(input("Enter a year: "))


if (year % 4 == 0 and year % 100 != 0) or (year % 400 ==
0):
```

```
    print("Leap year")
else:
    print("Not a leap year")
```

4.11 Common Mistakes to Avoid

Mistake	Why It's Wrong	Correct
Missing colon `:`	Python needs it after `if`, `else`, etc.	`if x > 5:`
Wrong indentation	Code won't run or behave unexpectedly	Use consistent indentation (4 spaces)
Using `=` instead of `==`	`=` is assignment, not comparison	`if x == 10:`
Writing conditions that always evaluate the same	Learn truthy/falsy behavior	`if name != "":` or `if name:`

Practice Challenges

1. Write a program that asks for a number and prints if it's positive, negative, or zero.

2. Ask the user to input their age. Print:
    - “Child” if under 13,
    - “Teen” if 13–19,
    - “Adult” if 20–64,
    - “Senior” if 65 or older.
3. Ask for a username and password. Print “Login successful” only if both match expected values.
4. Create a program that checks if a number is divisible by both 3 and 5.
5. Build a mini program that checks if a character is a vowel or consonant.

Mini Project: Basic Login System

Write a simple program that:

- Asks for a username and password
- If both are correct, prints a welcome message
- Else, prints an error

Example:

```python
correct_username = "user1"

correct_password = "pass123"
```

```
username = input("Enter username: ")

password = input("Enter password: ")


if username == correct_username and password ==
correct_password:

    print("Welcome,", username)

else:

    print("Invalid login.")
```

Extend it:

- Allow for case-insensitive login
- Give a maximum of 3 attempts (advanced)

Summary

✅ You learned how to use control flow to make decisions
✅ Mastered `if`, `elif`, and `else` statements
✅ Practiced writing nested and combined conditionals
✅ Used real-world logic in your Python programs
✅ Explored truthy and falsy values in decision-making

Part 5: Loops in Python

5.1 What Are Loops?

Loops are used to **repeat blocks of code**. Instead of writing the same line over and over, you can tell Python:

> "Repeat this code a certain number of times or while a condition is true."

Python supports two main types of loops:

- `for` loops — great for iterating over a **sequence**
- `while` loops — run as long as a **condition is true**

5.2 The `for` Loop

A `for` loop is used to **iterate** (loop through) a sequence such as:

- Lists
- Strings
- Ranges
- Tuples

Basic Syntax

```python
for variable in sequence:
    # code block
```

Example 1: Loop over a list

```python
fruits = ["apple", "banana", "cherry"]
```

```
for fruit in fruits:
    print(fruit)
```

Output:

```
None
apple
banana
cherry
```

Here, `fruit` takes on each value from the list, one by one.

5.3 Using `range()` with `for` Loops

The `range()` function creates a sequence of numbers.

Syntax:

```
Python
range(start, stop, step)
```

- `start` (optional): the first number (default is 0)
- `stop`: the sequence goes up to (but doesn't include) this number
- `step` (optional): how much to increase each time (default is 1)

Examples:

```python
for i in range(5):
    print(i)  # prints 0 to 4


for i in range(1, 6):
    print(i)  # prints 1 to 5


for i in range(10, 0, -2):
    print(i)  # prints 10, 8, 6, 4, 2
```

5.4 The `while` Loop

A `while` loop repeats as long as a **condition is true**.

Syntax:

```python
while condition:
    # code block
```

Example:

```python
x = 1
```

```
while x <= 5:
    print(x)
    x += 1
```

Output:

```
None
1
2
3
4
5
```

5.5 Infinite Loops and How to Avoid Them

A **common mistake** with `while` loops is creating an **infinite loop**—one that never ends.

Example of an infinite loop:

```python
x = 5
while x > 0:
    print("Counting down...")
    # Forgot to update x
```

Always make sure the condition **will eventually become false**.

5.6 `break` and `continue` Statements

`break`: Exits the loop completely

```python
while True:
    name = input("Enter your name (or 'q' to quit): ")
    if name == "q":
        break
    print("Hello,", name)
```

`continue`: Skips the rest of the current loop and goes to the next iteration

```python
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

Output:

```
1
2
```

```
4

5
```

5.7 `else` Clause in Loops

Loops in Python can have an `else` clause. It runs **only if the loop is not terminated by `break`**.

Example:

```python
for i in range(5):
    print(i)
else:
    print("Finished!")
```

But:

```python
for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("Finished!")  # Won't run because of break
```

5.8 Nested Loops

You can place one loop **inside another**.

Example: Multiplication Table

```python
for i in range(1, 4):
    for j in range(1, 4):
        print(i * j, end="\t")
    print()
```

Output:

```
1    2    3
2    4    6
3    6    9
```

5.9 Looping Over Strings

Strings are **sequences of characters**, so you can loop over them.

```python
for letter in "hello":
    print(letter)
```

5.10 Real-World Loop Examples

Example 1: Countdown Timer

Python

```python
import time


for i in range(5, 0, -1):
    print(i)
    time.sleep(1)
print("Time's up!")
```

Example 2: Password Attempts

Python

```python
correct_password = "letmein"
attempts = 3


while attempts > 0:
    guess = input("Enter password: ")
    if guess == correct_password:
        print("Access granted.")
        break
```

```
    else:
        attempts -= 1
        print("Wrong! Attempts left:", attempts)
else:
    print("Too many attempts. Access denied.")
```

Example 3: Sum of Numbers

```python
total = 0

for i in range(1, 6):
    total += i

print("Sum from 1 to 5 is:", total)
```

5.11 Common Mistakes with Loops

Mistake	Why It's Wrong	Fix
Forgetting to update variable in `while`	Causes infinite loop	Add increment or update

Off-by-one errors	Wrong range start/stop	Double-check `range()`
Misusing `break` or `continue`	Logic doesn't work	Use them only when necessary
Improper indentation	Code doesn't run as expected	Be consistent with 4-space indentation

---

Practice Challenges

1. **Countdown Timer**: Ask the user for a number and count down to 0.
2. **Multiples of 3**: Print all numbers between 1 and 100 divisible by 3.
3. **Sum of Even Numbers**: Use a loop to sum even numbers from 1 to 100.
4. **Reverse a String**: Print each character of a string in reverse order.
5. **Guessing Game**: Make a number guessing game where the user has 5 attempts.

Mini Project: Number Guessing Game

Requirements:

- Program chooses a random number between 1–20
- User has 5 guesses to find it

- Gives feedback (“too high” or “too low”)

Example Code:

```python
import random


secret_number = random.randint(1, 20)

attempts = 5


while attempts > 0:

    guess = int(input("Guess the number (1–20): "))


    if guess == secret_number:

        print("Correct! You win!")

        break

    elif guess < secret_number:

        print("Too low.")

    else:

        print("Too high.")


    attempts -= 1
```

```
if attempts == 0:
    print("You lose! The number was:", secret_number)
```

Summary

✅ You learned how to repeat actions using `for` and `while` loops
✅ Used `range()` to control loops
✅ Applied `break`, `continue`, and `else` with loops
✅ Practiced looping through sequences like lists and strings
✅ Wrote practical programs using loops

Part 6: Data Structures in Python

6.1 What Are Data Structures?

Data structures are **containers** for storing and organizing data. They allow you to:

- Group related data together
- Access, update, and remove items efficiently
- Iterate through items

In this part, you'll learn the four core Python data structures:

1. **Lists**
2. **Tuples**
3. **Sets**
4. **Dictionaries**

6.2 Lists – Ordered, Changeable Sequences

A **list** is an ordered collection of items that can be changed.

Create a List:

```python
fruits = ["apple", "banana", "cherry"]
```

Access Elements:

```python
print(fruits[0])   # apple

print(fruits[-1])  # cherry
```

Modify Elements:

```python
fruits[1] = "blueberry"

print(fruits)  # ['apple', 'blueberry', 'cherry']
```

List Methods:

```python
fruits.append("orange")     # Add to end

fruits.insert(1, "kiwi")    # Insert at index

fruits.remove("apple")      # Remove by value

fruits.pop()                # Remove last item
```

```
fruits.sort()                # Sort in place
```

Loop Through List:

```python
for fruit in fruits:
    print(fruit)
```

Check Membership:

```python
if "banana" in fruits:
    print("Found banana")
```

6.3 Tuples – Ordered, Unchangeable Sequences

Tuples are like lists, but **immutable** (can't be changed).

Create a Tuple:

```python
coordinates = (10, 20)
```

Access Elements:

```python
print(coordinates[0])  # 10
```

Tuple Packing & Unpacking:

```python
person = ("Alice", 30)

name, age = person

print(name)  # Alice
```

✅ Use tuples when you want a fixed group of related items.

6.4 Sets – Unordered, Unique Items

A **set** is an unordered collection of **unique** items.

Create a Set:

```python
colors = {"red", "green", "blue"}
```

Add & Remove:

```python
colors.add("yellow")

colors.remove("red")
```

Check Membership:

```python
if "blue" in colors:

    print("Blue is here")
```

Useful for Removing Duplicates:

```python
nums = [1, 2, 2, 3, 3, 3]

unique_nums = set(nums)

print(unique_nums)  # {1, 2, 3}
```

6.5 Dictionaries – Key-Value Pairs

A **dictionary** stores data in **key-value** pairs.

Create a Dictionary:

```python
person = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}
```

Access Values:

```python
print(person["name"])  # Alice
```

Modify/Add Items:

```python
person["age"] = 31

person["job"] = "Engineer"
```

Remove Items:

```python
del person["city"]
```

Loop Through Dictionary:

```python
for key, value in person.items():

    print(key, "->", value)
```

6.6 Practical Examples

Example 1: Shopping List (List)

```python
shopping_list = []


shopping_list.append("milk")

shopping_list.append("bread")


for item in shopping_list:
```

```
    print("Buy:", item)
```

Example 2: Coordinate System (Tuple)

```python
location = (5, 10)

x, y = location

print("X:", x, "Y:", y)
```

Example 3: Unique Usernames (Set)

```python
usernames = ["alice", "bob", "alice", "carol"]

unique_usernames = set(usernames)

print(unique_usernames)
```

Example 4: Student Record (Dictionary)

```python
student = {

    "name": "John",

    "grades": [90, 85, 92]
```

```
}


average = sum(student["grades"]) / len(student["grades"])

print("Average grade:", average)
```

6.7 Conversion Between Types

You can convert between list, tuple, set:

```python
my_list = [1, 2, 3, 3]

my_set = set(my_list)        # Remove duplicates

my_tuple = tuple(my_set)     # Make it immutable

my_list_again = list(my_tuple)
```

6.8 Summary Table

Type	Ordered	Mutable	Allows Duplicates	Example
List	✅	✅	✅	`["a", "b", "c"]`

Tuple	✅	❌	✅	`("a", "b", "c")`
Set	❌	✅	❌	`{"a", "b", "c"}`
Dictionary	✅*	✅	❌ (unique keys)	`{"name": "Alice"}`

*Dictionaries maintain insertion order in Python 3.7+

Practice Challenges

1. **List Practice**: Create a list of 5 favorite movies. Print them one by one.
2. **Tuple Practice**: Create a tuple with your birth year, month, and day. Unpack and print them.
3. **Set Practice**: Convert a list with duplicate numbers into a set and print the unique numbers.
4. **Dictionary Practice**: Create a dictionary with keys `"name"`, `"age"`, and `"hobby"`. Print a sentence using those values.
5. **Nested Dictionary**: Make a dictionary with students' names as keys and a list of their marks as values. Compute and print their averages.

Mini Project: Simple Address Book

Build a dictionary-based address book where users can:

- Add a contact
- View all contacts
- Search by name

Example Code:

```python
address_book = {}


while True:
    print("\n1. Add Contact\n2. View Contacts\n3. Search Contact\n4. Exit")
    choice = input("Choose an option: ")


    if choice == "1":
        name = input("Name: ")
        phone = input("Phone: ")
        address_book[name] = phone
    elif choice == "2":
        for name, phone in address_book.items():
            print(name, "->", phone)
    elif choice == "3":
        search = input("Enter name to search: ")
```

```
        if search in address_book:
            print("Phone:", address_book[search])
        else:
            print("Contact not found.")
    elif choice == "4":
        break
    else:
        print("Invalid choice.")
```

Common Mistakes

Mistake	Problem	Fix
Accessing list index out of range	Crashes the program	Check list length
Modifying tuples	Tuples are immutable	Use lists if modification is needed
Assuming order in sets	Sets are unordered	Don't rely on element order
Using mutable types as dict keys	Keys must be immutable	Use strings, numbers, or tuples as keys

---

Summary

✅ You learned about **lists**, **tuples**, **sets**, and **dictionaries**
✅ You practiced creating, accessing, modifying, and looping through them
✅ You applied them in real-life examples and mini projects
✅ You now understand how to **store and organize data** effectively

Part 7: Functions in Python

7.1 What Is a Function?

A **function** is a reusable block of code that performs a specific task. Functions help you:

- **Avoid repeating code**
- **Break down problems** into smaller parts
- **Organize code** for clarity and reuse

Python includes many **built-in functions** (like `print()`, `len()`, etc.), but you can also create your own.

7.2 Defining a Function

Use the `def` keyword to create a function.

Syntax:

```python
def function_name():
    # code block
```

Example:

```python
def greet():
    print("Hello, world!")


greet()  # call the function
```

7.3 Function Parameters

Parameters allow functions to accept input.

Example with One Parameter:

```python
def greet(name):
    print("Hello,", name)


greet("Alice")
greet("Bob")
```

Example with Multiple Parameters:

```python
def add(x, y):
    print(x + y)
```

```
add(3, 5)
```

7.4 Return Values

Functions can return results using the `return` keyword.

Example:

```python
def square(x):
    return x * x


result = square(4)
print(result)  # 16
```

Without `return`, the function just performs actions but gives no output to reuse.

7.5 Function with Default Parameters

You can assign default values to parameters.

Example:

```python
def greet(name="Guest"):
```

```
    print("Hello,", name)


greet()         # Hello, Guest
greet("Alice")  # Hello, Alice
```

7.6 Function with Keyword Arguments

Python allows **named arguments**, which can improve readability.

Example:

```
Python
def describe_pet(animal, name):
    print(f"{name} is a {animal}")


describe_pet(animal="dog", name="Rex")
```

7.7 Practical Examples

Example 1: Calculate Area of Rectangle

```
Python
def area(width, height):
    return width * height
```

```
print(area(5, 10))  # 50
```

Example 2: Even or Odd Checker

Python

```
def is_even(num):
    return num % 2 == 0

print(is_even(4))  # True
print(is_even(7))  # False
```

Example 3: Personalized Message

Python

```
def send_message(name, message):
    print(f"To {name}: {message}")

send_message("Alice", "Welcome!")
```

7.8 Scope of Variables

Scope determines **where a variable is accessible**.

Local Scope:

Variables created inside a function exist **only inside that function**.

```python
def greet():
    name = "Alice"  # local variable
    print("Hello", name)

greet()
# print(name)  # Error: name is not defined
```

Global Scope:

Variables created outside functions are **global**.

```python
name = "Bob"

def greet():
    print("Hello", name)  # uses global variable

greet()
```

Modifying Global Variables:

You must declare them with `global`:

```python
count = 0


def increment():
    global count
    count += 1


increment()
print(count)
```

7.9 Nested Functions

You can define functions inside other functions.

```python
def outer():
    print("Outer function")


    def inner():
        print("Inner function")


    inner()
```

```
outer()
```

7.10 Lambda Functions (Anonymous Functions)

Python allows short, one-line functions with `lambda`.

Syntax:

```python
lambda arguments: expression
```

Example:

```python
square = lambda x: x * x

print(square(5))  # 25
```

Another example:

```python
add = lambda a, b: a + b

print(add(2, 3))  # 5
```

> Use `lambda` for simple, throwaway functions (e.g., for sorting or filtering lists).

7.11 Practice Exercises

1. **Greeting Function**: Create a function `say_hello(name)` that prints "Hello, [name]!"
2. **Math Function**: Create `multiply(x, y)` that returns the product.
3. **Maximum Number**: Write a function `max_of_three(a, b, c)` that returns the largest number.
4. **Palindrome Checker**: Write `is_palindrome(word)` that returns `True` if the word reads the same backward.
5. **Fibonacci Generator**: Write a function that prints the first N Fibonacci numbers.

Mini Project: Simple Calculator

Build a calculator that supports addition, subtraction, multiplication, and division using functions.

Example Code:

Python

```python
def add(x, y):
    return x + y


def subtract(x, y):
    return x - y


def multiply(x, y):
    return x * y
```

```
def divide(x, y):
    if y == 0:
        return "Error: Division by zero"
    return x / y

while True:
    print("\nOptions: + - * / q (quit)")
    choice = input("Choose operation: ")

    if choice == "q":
        break

    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))

    if choice == "+":
        print("Result:", add(a, b))
    elif choice == "-":
        print("Result:", subtract(a, b))
    elif choice == "*":
```

```
        print("Result:", multiply(a, b))
    elif choice == "/":
        print("Result:", divide(a, b))
    else:
        print("Invalid option.")
```

7.12 Common Mistakes with Functions

Mistake	Description	Fix
Not using `return`	Function runs but doesn't return value	Use `return` to give output
Forgetting parentheses	Refers to function, not call	Use `greet()` instead of `greet`
Confusing local/global variables	Modifying global variable inside function	Use `global` keyword
Missing arguments	Causes `TypeError`	Match function signature correctly
Naming conflict	Function name same as built-in	Avoid naming your function `print`, `list`, etc.

Summary

✅ You learned how to define and call functions
✅ You used parameters and return values
✅ You understood local and global scope
✅ You practiced using `lambda` for simple functions
✅ You built a reusable calculator with function logic

Introduction to Security

SECURITY

Offensive Security

Proactive and adversarial cybersecurity practices aimed at identifying and exploiting vulnerabilities in systems, networks, or applications before malicious hackers can. Unlike defensive security, which focuses on protecting assets, offensive security involves activities like:

- **Penetration testing** (ethical hacking): Simulating attacks to find weaknesses.
- **Red teaming**: Emulating real-world attack scenarios to test defenses.
- **Vulnerability assessment**: Identifying and evaluating potential security flaws.
- **Exploitation development**: Creating and testing custom exploits to assess risks.

Its goal is to improve security by understanding how systems can be breached, helping organizations strengthen their defenses.

Defensive Security

The practice of protecting computer systems, networks, and data from cyber threats. It focuses on **prevention, detection, and response** to attacks, ensuring systems remain secure and operational.

Key Areas of Defensive Security:

- **Firewalls and Intrusion Detection Systems (IDS):** Block or monitor unwanted traffic.
- **Security Information and Event Management (SIEM):** Centralizes and analyzes security logs.
- **Endpoint Protection:** Secures user devices from malware and intrusions.
- **Patch Management:** Keeps systems up to date with security fixes.
- **Incident Response:** Detects and responds to attacks in real-time.
- **Access Control:** Ensures only authorized users can access resources.
- **Security Awareness Training:** Educates users to prevent human error exploits.

Goal:

To reduce the risk of breaches and ensure quick recovery if one occurs. Defensive security is the backbone of an organization's cybersecurity strategy, supporting business continuity and compliance.

Careers in Cyber Security

Entry-Level Roles

1. Security Analyst / SOC Analyst

Description: Monitors networks and systems for security breaches, investigates alerts, and escalates threats.
Qualifications:

- Bachelors in cybersecurity or IT
- Basic understanding of networking and SIEM tools (e.g., Splunk)
- CompTIA Security+, CySA+

2. IT Security Administrator

Description: Maintains security systems and implements access controls and firewalls.
Qualifications:

- Bachelor's degree in IT
- Familiarity with firewalls, VPNs, and security policies
- CompTIA Security+, Cisco CCNA Security

3. Incident Response Technician

Description: First responder to cyber incidents, contains threats, and documents findings.
Qualifications:

- Bachelors in cybersecurity or related field
- Knowledge of incident response frameworks (e.g., NIST)
- CompTIA Security+, GIAC Certified Incident Handler (GCIH)

4. Network Security Technician

Description: Assists in protecting and configuring secure network infrastructure.
Qualifications:

- Basic networking knowledge (TCP/IP, subnets)
- CCNA, CompTIA Network+

5. Vulnerability Analyst

Description: Scans and analyzes systems for vulnerabilities and reports risk.
Qualifications:

- Understanding of common vulnerabilities (CVEs)
- Experience with scanning tools (Nessus, Qualys)
- CompTIA Security+, OSCP (optional)

6. Cybersecurity Support Specialist

Description: Provides technical support on security software and access issues.
Qualifications:

- Associate's or Bachelor's in IT
- Customer service skills + technical knowledge
- Entry-level certifications (Security+, A+)

Mid-Level Roles

7. Penetration Tester (Ethical Hacker)

Description: Simulates real-world attacks to find and exploit vulnerabilities.
Qualifications:

- Strong knowledge of networking, scripting, and OS internals
- CEH, OSCP, eJPT (for junior roles)

8. Threat Intelligence Analyst

Description: Gathers and analyzes cyber threat data to prevent attacks.
Qualifications:

- Knowledge of threat actor tactics (MITRE ATT&CK)
- Experience with threat feeds and indicators of compromise (IOCs)
- CompTIA CySA+, GIAC Cyber Threat Intelligence (GCTI)

9. Digital Forensics Analyst

Description: Investigates digital crimes by analyzing devices and logs.
Qualifications:

- Strong investigative mindset
- Tools: EnCase, FTK, Autopsy
- GIAC Certified Forensic Analyst (GCFA)

10. Security Engineer

Description: Designs and implements secure infrastructure and mitigations.
Qualifications:

- Proficiency in security tools and systems architecture
- CISSP, GIAC certifications

11. Security Consultant

Description: Advises clients or companies on best security practices.
Qualifications:

- Broad knowledge of security standards (ISO 27001, NIST)
- CISA, CISSP

12. Red Team Operator

Description: Simulates advanced attacks to test organizational readiness.
Qualifications:

- Advanced knowledge of offensive tactics
- OSCP, CRTO, Red Team Ops certifications

13. Blue Team Defender

Description: Focuses on detection, defense, and response to cyber threats.
Qualifications:

- Familiarity with SIEM, EDR, and log analysis
- CySA+, GCIA, or Blue Team Level certifications

Advanced / Senior Roles

14. Security Architect

Description: Designs complex security infrastructure and system architecture.
Qualifications:

- Years of experience in various security domains
- CISSP-ISSAP, TOGAF (for architecture)

15. Cybersecurity Manager

Description: Leads teams and oversees cybersecurity operations.
Qualifications:

- Management experience + technical background
- CISSP, CISM

16. Incident Response Lead

Description: Manages the team during active cyber incidents.
Qualifications:

- Experience in IR, malware analysis

- GCIH, GCFA, CISSP

17. Malware Analyst / Reverse Engineer

Description: Dissects malware to understand its functionality and origins.
Qualifications:

- Deep understanding of assembly, Windows internals
- GREM, OSCE, RE tools (IDA Pro, Ghidra)

18. Cybersecurity Researcher

Description: Explores new threats, vulnerabilities, and attack techniques.
Qualifications:

- Strong R&D skills, often a PhD or advanced degree preferred
- OSCP, or specialized exploit dev experience

19. Application Security Engineer

Description: Secures web, mobile, and desktop applications against vulnerabilities.
Qualifications:

- Experience with secure coding practices (OWASP)
- Certifications: CSSLP, GWAPT

20. Cloud Security Engineer

Description: Focuses on securing cloud environments (AWS, Azure, GCP).
Qualifications:

- Experience with cloud security controls
- Certifications: AWS Certified Security – Specialty, Azure SC-300

Specialized Roles

21. GRC Analyst

Description: Manages risk, compliance, and regulatory frameworks.
Qualifications:

- Knowledge of frameworks (NIST, ISO, HIPAA)
- CISA, CRISC

22. Privacy Officer / DPO

Description: Ensures data privacy compliance (GDPR, CCPA).
Qualifications:

- Legal and regulatory knowledge
- CIPP/E, CIPM

23. Cryptographer

Description: Develops secure encryption systems and algorithms.
Qualifications:

- Background in mathematics/computer science
- Advanced degree + cryptographic experience

24. DevSecOps Engineer

Description: Integrates security into DevOps pipelines.
Qualifications:

- Knowledge of CI/CD, IaC, and security tooling
- Certifications: DevSecOps Foundation, Kubernetes Security

25. Security Automation Engineer

Description: Builds automation tools for threat detection and response.
Qualifications:

- Programming (Python, PowerShell)

- Experience with SOAR platforms (Splunk Phantom, Palo Alto XSOAR)

26. ICS/SCADA Security Specialist

Description: Secures industrial systems used in utilities and manufacturing.
Qualifications:

- Knowledge of OT environments, protocols (Modbus, DNP3)
- GIAC Global Industrial Cyber Security Professional (GICSP)

Leadership Roles

27. Chief Information Security Officer (CISO)

Description: Executive responsible for overall cybersecurity strategy.
Qualifications:

- Years of leadership experience
- MBA or similar + CISSP, CISM

28. Security Program Director

Description: Oversees large-scale security programs and projects.
Qualifications:

- Project management + cybersecurity background
- PMP + CISM or CISSP

29. Cybersecurity Operations Director

Description: Leads SOC, incident response, and day-to-day security ops.
Qualifications:

- Deep operational experience
- CISSP, GIAC, leadership certifications

Cryptography

1. Introduction

Cryptography is the discipline that turns mathematical abstractions into practical tools for securing communication and data. It is the reason your phone can send a credit-card number over café Wi-Fi without fear, why spacecraft can receive commands from Earth without hostile tampering, and how dissidents can organise under repressive regimes. At its core sit four security goals:

- **Confidentiality** – keeping content secret from unauthorised parties.
- **Integrity** – detecting any unauthorised modification.
- **Authenticity** – proving the origin of a message.
- **Non-repudiation** – preventing the sender from later denying authorship.

Cryptography achieves these goals with *primitives* such as ciphers, hash functions and digital signatures, which are combined into higher-level *protocols*. Because threat models evolve—new mathematics, faster hardware, novel side-channel leaks—designers emphasise public standards, open peer review and conservative margins of safety. Mastery of the fundamentals therefore begins not with exotic research topics but with a clear grasp of the basic building blocks and how they interact inside real systems. This chapter-length overview sets out to deliver that grasp, guiding you through the essential ideas, algorithms, protocols and implementation pitfalls that make modern cryptography work.

---

2. A Brief Historical Arc

Humanity has sought private communication for as long as it has written language. Spartan generals (ca. 400 BCE) wrapped parchment around a rod—the *scytale*—so that letters aligned only on a rod of identical diameter. Julius Caesar shifted each letter three places along the Latin alphabet, a simple substitution cipher immortalised as the "Caesar shift." The Renaissance brought Blaise de Vigenère's polyalphabetic table, which resisted frequency analysis for three centuries until Charles Babbage and Friedrich Kasiski exposed its weakness.

Mechanisation arrived during the world wars. Germany's electro-mechanical Enigma relied on rotating rotors but succumbed to Polish mathematicians and Alan Turing's *bombe*. Claude Shannon's 1949 paper "Communication Theory of Secrecy Systems" provided the first rigorous definition of *perfect secrecy*. The modern era dawned in 1976, when Whitfield Diffie and Martin Hellman introduced public-key cryptography and, with Ralph Merkle, the first practical key-exchange protocol. RSA followed in 1977, and by 2001 the Advanced Encryption Standard (AES) replaced DES, marking the beginning of today's standards-based landscape.

---

3. Core Security Properties

Every real-world cryptographic design is evaluated against four intertwined properties:

- **Confidentiality** – without the correct key, an adversary should gain no information about plaintext from ciphertext (*semantic security*).
- **Integrity** – any bit-flips, insertions or deletions must be detectable; Message Authentication Codes (MACs) and authenticated-encryption modes provide this guarantee.
- **Authenticity** – the receiver must be certain who sent the message; public-key certificates tie cryptographic keys to human or organisational identities.
- **Non-repudiation** – a validly signed message can be proven in court even if the signer later protests.

Systems also value **availability** (keys remain usable) and **forward secrecy** (compromise of long-term keys does not decrypt past traffic). These goals are realised by orchestrating primitives into carefully analysed protocols whose security proofs relate system behaviour to well-studied mathematical problems.

---

4. Symmetric-Key Cryptography

Symmetric ciphers use the same secret key for both encryption and decryption, offering speed and constant-time operations suitable for high-throughput hardware. The workhorse is the **Advanced Encryption Standard (AES)**—a substitution–permutation network operating on 128-bit blocks with 10, 12 or 14

rounds for 128-, 192- and 256-bit keys respectively. AES was selected in 2001 to replace the ageing Data Encryption Standard (DES), whose 56-bit key succumbed to exhaustive search.

Block ciphers alone are insufficient because identical plaintext blocks would produce identical ciphertext. *Modes of operation* blend unpredictable *initialisation vectors* (IVs) or counters with each block:

- **CBC (Cipher Block Chaining)** – XORs each plaintext block with the previous ciphertext block.
- **CTR (Counter)** – treats the block cipher as a keystream generator by encrypting a monotonically increasing counter.
- **GCM (Galois/Counter Mode)** – adds polynomial MACs, yielding *authenticated encryption with associated data* (AEAD).

Stream ciphers such as **ChaCha20** output pseudorandom keystream bytes that are XORed with plaintext. Because reusing a keystream fatally reveals information, unique nonces and robust random number generators are essential ingredients alongside the cipher itself. (NIST Computer Security Resource Center)

---

5. Asymmetric-Key Cryptography

Asymmetric cryptography separates keys into a public component for encryption or verification and a private component for decryption or signing, removing the logistical nightmare of sharing a secret ahead of time. The original scheme, **RSA**, relies on factoring a composite modulus N=pqN=pq. Security scales with modulus size: 2048-bit keys are common, while 3072-bit or stronger are mandated for long-lived certificates.

Key exchange is handled by **Diffie–Hellman (DH)**: two parties choose secrets *a* and *b*, exchange gag^a and gbg^b, and compute gabg^{ab} as their shared secret. Replacing integers modulo pp with points on an elliptic curve yields **ECDH**, providing equivalent security with far shorter keys (e.g., Curve 25519 uses 256-bit keys). Smaller keys mean faster handshakes and lighter certificates—critical for mobile devices. Asymmetric primitives also underpin digital signatures (Section 7) and PKI (Section 8). Their downside is computational cost—several orders of magnitude slower than

AES—so protocols negotiate an asymmetric handshake, then switch to symmetric encryption for bulk data.

---

6. Cryptographic Hash Functions

A cryptographic hash function HH maps an input of arbitrary length to a fixed-length digest in a way that is deterministic yet practically irreversible. Strong hashes satisfy:

1. *Preimage resistance*: given a digest, finding any input that maps to it is infeasible.

2. *Second-preimage resistance*: given one input, finding a different input with the same digest is infeasible.

3. *Collision resistance*: finding *any* two inputs that collide is infeasible.

SHA-2 (SHA-256, SHA-512) remains dominant for TLS and cryptocurrencies, while SHA-3, standardised in 2015, uses the Keccak sponge construction to resist length-extension attacks. Hashes power integrity checks, password storage (with salt and key-stretching), deterministic randomness generators, commitment schemes and HMACs. When combined with a secret key, a hash yields a Message Authentication Code; HMAC-SHA-256 underpins OAuth tokens, JSON Web Signatures and countless APIs.

---

7. Digital Signatures

Digital signatures bind identities to messages, delivering authenticity and non-repudiation. The workflow hashes the message and applies a private-key operation; verification recomputes the hash and uses the public key.

- RSA-PSS – signs by raising the padded hash to the secret exponent dd modulo NN.

- DSA / ECDSA – leverage discrete logarithms in integer and elliptic-curve groups respectively; reused nonces leak the private key.

- EdDSA (e.g., *Ed25519*) – a modern, deterministic variant on twisted Edwards curves that avoids nonce-reuse disasters and supports efficient batch verification.

To sign large binaries or structured data, one signs the hash rather than raw bytes, keeping signature sizes manageable. Later (Section 11) we revisit signatures designed to survive quantum adversaries, but for most contemporary applications Ed25519 offers an appealing balance of speed, key size and security.

---

8. Key Management and Public Key Infrastructure

Cryptographic strength is meaningless if keys leak or go stale. Key management spans:

- Generation – harvest entropy from hardware noise, then seed OS pools.
- Storage – keep private keys inside HSMs, TPMs or Secure Enclaves.
- Distribution – use DH/ECDH during TLS handshakes or ship public keys inside X.509 certificates.
- Rotation & Revocation – certificates expire; OCSP and CRLs announce premature revocation.
- Backup & Destruction – encrypted backups guard against loss; shredding media prevents forensic recovery.

PKI translates public keys into meaningful identities. Certificate Authorities sign certificates binding subject names to public keys. Browsers ship with trust stores of hundreds of CAs; compromise of any CA undermines the chain. Certificate Transparency logs and domain-validation audits mitigate mis-issuance by adding public, append-only ledgers anyone can monitor.

---

9. Cryptographic Protocols

Primitives become useful only when stitched into *protocols* that specify message formats, handshake sequences and error handling. TLS 1.3 proceeds:

1. ClientHello – client sends random bytes, supported cipher suites and an ECDH key-share.

2. ServerHello – server chooses parameters, replies with its certificate and a key-share.

3. Key derivation – both derive a shared secret, expand it via HKDF into traffic keys.

4. Encrypted handshake – identities and parameters are authenticated before application data flows.

TLS 1.3 removed fragile constructs and halved handshake latency, enabling *0-RTT* resumes. Other protocols include IPsec (layer-3 VPNs), SSH (remote shell) and Signal's Double Ratchet (secure messaging with post-compromise forward secrecy). Their security rests not only on sound mathematics but also on exact state machines, constant-time coding and graceful fallback rules that avoid downgrade attacks.

---

10. Cryptanalysis and Attack Vectors

Cryptanalysis is the counterpoint to cryptography, probing for weaknesses and ensuring primitives meet advertised strength. Classical substitution systems fell to *frequency analysis*. Modern symmetric ciphers withstand analogous attacks, yet reduced-round variants let academics measure security margins.

- Differential & linear cryptanalysis – track input differences or linear approximations through S-boxes.

- Algebraic attacks – model stream ciphers as polynomial systems solved by Gröbner bases.

- Impossible-differential & integral attacks – break some lightweight ciphers with combinatorial arguments.

More alarming are **side-channel attacks** that bypass mathematics entirely: timing (Lucky-13), power analysis, electromagnetic leakage and deliberate fault injection. Countermeasures include constant-time code, masking, blinding and tamper-resistant packaging, reminding us that real security demands a holistic view of software, hardware and environment.

---

11. Post-Quantum Cryptography

Quantum computers equipped with Shor's algorithm could factor 2048-bit RSA in hours and break elliptic-curve Diffie–Hellman outright, jeopardising TLS and software updates worldwide. Symmetric ciphers suffer only quadratic speed-ups under Grover's algorithm, so doubling key sizes (e.g., AES-256) suffices. The greater challenge lies in replacing public-key primitives.

The U.S. National Institute of Standards and Technology began a public competition in 2016 and, in July 2022, announced four frontrunner algorithms for standardisation: **CRYSTALS-Kyber** for key encapsulation and **CRYSTALS-Dilithium**, **FALCON** and **SPHINCS+** for digital signatures. Draft FIPS documents for the first three appeared in 2023, with FALCON following in 2024.(NIST) All rely on lattice- or hash-based problems believed resistant to quantum attacks.

Migration is under way: Google has trialled Kyber-hybrid TLS in Chrome, and OpenSSH 9 enables optional PQC key exchange. Best practice is *hybridisation*—negotiating both classical and post-quantum keys—so that compromise of one primitive does not collapse the session. Transition timelines span a decade because hardware tokens, embedded systems and certificate lifecycles change slowly.

---

12. Advanced Topics: Homomorphic, Zero-Knowledge and Threshold Schemes

Research pushes cryptography beyond secrecy into *functional* security—allowing computations and assurances impossible a decade ago.

- **Fully Homomorphic Encryption (FHE)** lets cloud servers compute arbitrary functions on ciphertexts. Although bootstrapping noise still slows throughput by orders of magnitude, specialised hardware and schemes (CKKS, BFV, TFHE) are closing the gap.

- **Zero-Knowledge Proofs (ZKPs)** let a prover convince a verifier that a statement is true without revealing *why*. Succinct non-interactive proofs (zk-SNARKs) power privacy coins and roll-ups; transparent zk-STARKs remove trusted setups at the cost of larger proofs.

- **Threshold cryptography** splits a private key into *n* shares so that any *t* of them can sign or decrypt. This guards cryptocurrency cold wallets and election systems against insider compromise.

These innovations show cryptography's trajectory: not merely hiding data but enabling rich, verifiable computation under strong privacy constraints.

---

13. Implementing Cryptography Safely

History shows that flawed implementation, not broken maths, causes most compromises. The *Debian-OpenSSL* bug (2008) reduced entropy to 2^{15} possibilities; *Heartbleed* (2014) let attackers read 64 kB of server memory—including private keys—because of a missing bounds check; *ROCA* (2017) stemmed from weak RSA key generation in a hardware library.

Secure practice means:

- **Use vetted libraries** – libsodium, BoringSSL, RustCrypto.

- **Prefer AEAD APIs** – avoid manual encrypt-then-MAC sequences.

- **Employ memory-safe languages** or hardened C subsets.

- **Automate testing** – formal verification, fuzzing, CI.

- **Monitor side-channels** – constant-time coding, masking, blinding.

Cryptography must integrate with secure design patterns—least privilege, defence in depth and secure defaults—because perfect ciphers cannot salvage an application riddled with injection flaws or misconfigured access controls.

---

14. Legal, Policy and Ethical Dimensions

Cryptography shapes and is shaped by law, politics and ethics. Early U.S. export rules treated strong encryption as munitions, restricting browsers to 40-bit keys. Today, the EU's GDPR incentivises encryption by fining data leaks, while HIPAA sets sector-specific requirements. Efforts to mandate "exceptional access," from the 1990s Clipper Chip to modern client-side scanning proposals, raise fears of systemic vulnerability. Civil-society groups argue backdoors erode privacy and chill speech, whereas some law-enforcement bodies warn of "going dark."

Export control regimes, patent landscapes and standards bodies (IETF, ISO, NIST) influence which algorithms dominate. Ethical practice demands transparency, open algorithms and the right of individuals to secure their data from both criminals and overreaching states.

---

15. Conclusion

From the wooden scytale to quantum-resistant lattices, cryptography has progressed through an unending duel between inventors and attackers. Yet its basic pillars remain recognisable: symmetric keys for speed, asymmetric keys for open networking, cryptographic hashes for integrity and digital signatures for verifiable identity. Understanding these pillars—together with disciplined key management, protocol design and defensive implementation—empowers engineers to build the secure systems that underpin modern life.

The future promises both upheaval and opportunity. Quantum computing looms, while privacy-preserving computation, homomorphic encryption and zero-knowledge proofs open entire new categories of application. Behind every algorithm stands a global community of mathematicians, engineers and security professionals committed to rigorous peer review, responsible disclosure and open standards. Whether you aspire to design new primitives, implement secure applications or audit mission-critical systems, the journey begins with the basics—and those basics are now in your hands.

Chapter 1

Introduction - Cloud Foundations

You will learn how to:

- Describe cloud computing
- Define different models of cloud computing
- Distinguish between deployment methods and determine the difference between on-premises and cloud

You will discuss:

- The origins of cloud computing

Key terms:

- High availability
- Fault tolerance
- Low latency
- Scalability Elasticity

Important cloud terminology

- **High availability (highly available)**

High availability is a quality of computing infrastructure that allows it to continue functioning, even when some of its components fail. (Accessible when you need it)

- **Fault tolerance (fault tolerant)**

Ability to withstand a certain amount of failure and still remain functional

- **Scalability (scalable)**

Ability to easily grow in size, capacity, or scope when required
Growth is (usually) based on demand

- **Elasticity (elastic)**

Ability to grow (scale) when required and to reduce in size when resources are no longer needed

Understanding cloud computing

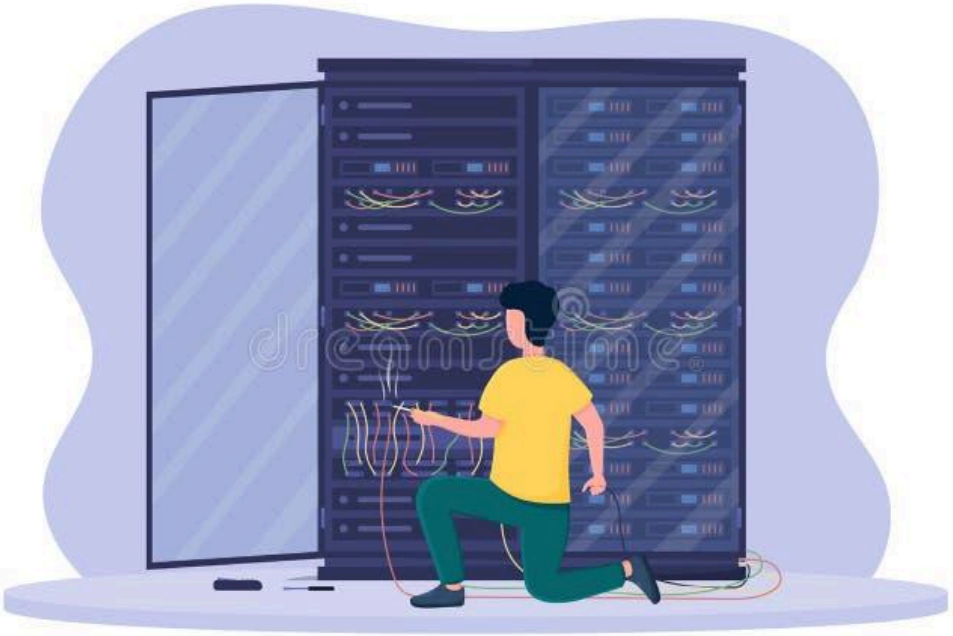
1. What is Cloud Computing

What does cloud computing mean to you?

Answer:

Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources. These resources are delivered through a cloud services platform via the internet with pay-as-you-go pricing.

2. Before cloud computing



Infrastructure as code

Before cloud computing, companies had to store all their data and software on their own hard drives and servers. The bigger the company, the more storage they needed. This way of treating data is not scalable at speed

Hardware solutions are physical, and they require:

- Space
- Staf
- Physical security

- Planning
- Capital expenditure

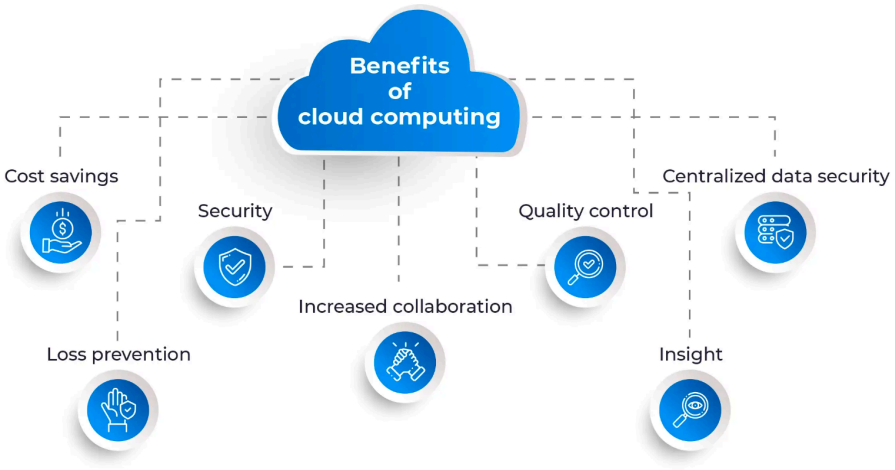
You must guess at theoretical maximum peaks:

- Is there enough resource capacity?
- Do we have sufficient storage?

What if your needs change?

- You must go through the time, effort, and cost required to change all these.

3. Using cloud computing

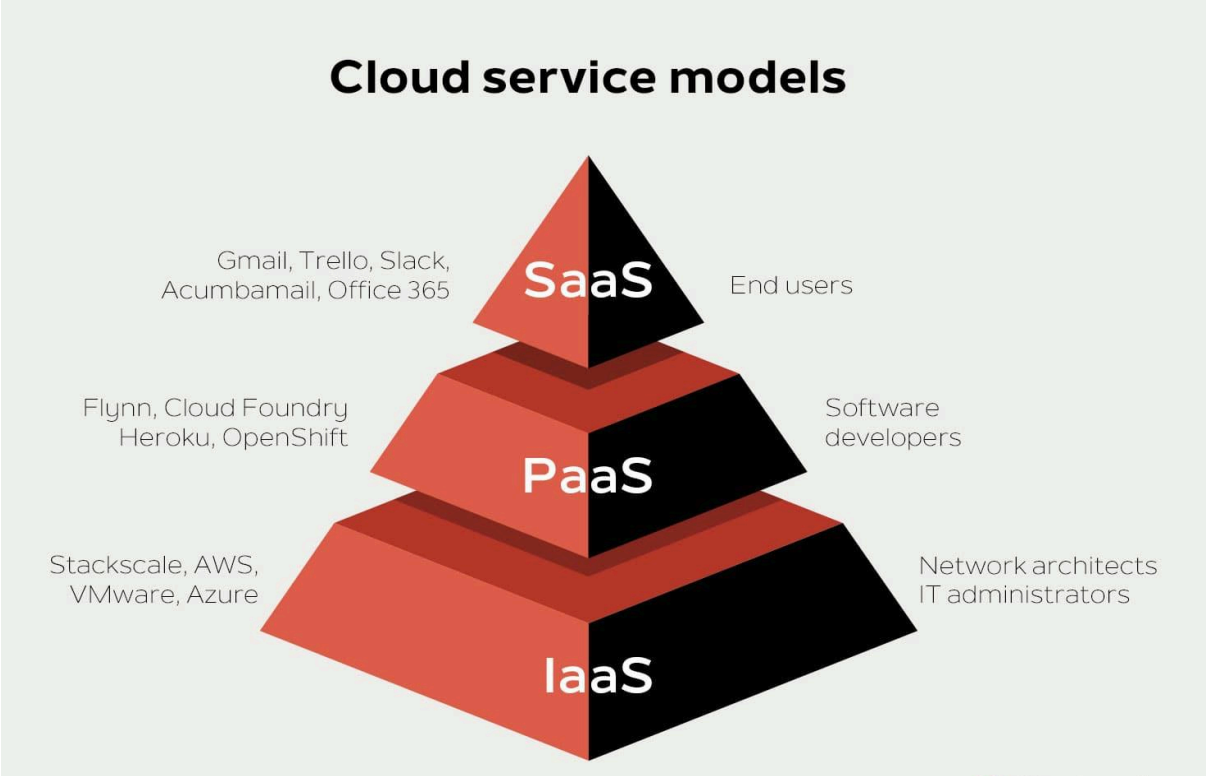


Cloud computing offers scalability, allowing businesses to easily scale their resources up or down based on demand. It promotes cost efficiency by eliminating the need for extensive on-premises hardware and maintenance expenses. Additionally, cloud services provide flexibility and accessibility, enabling users to access data and applications from anywhere with an internet connection, fostering collaboration and remote work.

We will go into more detail about the advantages of using cloud computing in the next chapters.

Models of cloud computing (IaaS vs PaaS vs SaaS)

1. Three models of cloud computing



There are three major cloud service models: software as a service (SaaS), infrastructure as a service (IaaS) and platform as a service (PaaS). Cloud service pricing models are

categorized into pay per use, subscription-based and hybrid, which is a combination of pay-per-use and subscription pricing models.

Software as a Service

Software as a service vendors host the applications, making them available to users via the internet. With SaaS, businesses don't have to install or download any software to their existing IT infrastructures. SaaS ensures that users are always running the most up-to-date versions of the software. The SaaS provider handles maintenance and support.

Platform as a Service

Platform as a service offers developers a platform for software development and deployment over the internet, enabling them to access up-to-date tools. PaaS delivers a framework that developers can use to create customized applications. The organization or the PaaS cloud vendor manages the servers, storage and networking, while the developers manage the applications.

Infrastructure as a Service

Infrastructure as a service is used by companies that don't want to maintain their own on-premises data centers. IaaS provides virtual computing resources over the Internet. The IaaS cloud vendor hosts the infrastructure components that typically exist in an on-premises data center, including servers, storage and networking hardware, as well as the hypervisor or virtualization layer.

Types of Cloud Services - IaaS, PaaS, & SaaS

2. Three cloud deployment models

The term cloud computing spans a range of classifications, types, and architecture models. This networked computing model has transformed how we work—you're likely already using the cloud. But the cloud isn't one thing—cloud computing can be categorized into three general types:

- **Public cloud** is cloud computing that's delivered via the internet and shared across organizations.
- **Private cloud** is cloud computing that is dedicated solely to your organization.
- **Hybrid cloud** is any environment that uses both public and private clouds.


Public Cloud
Private Cloud
Hybrid Cloud
No maintenance costs
Dedicated, secure
Policy-driven deployment
High scalability, flexibility
Regulation compliant
High scalability, flexibility
Reduced complexity
Customizable
Minimal security risks
Flexible pricing
High scalability
Workload diversity supports high reliability
Agile for innovation
Efficient
Improved security
Potential for high TCO
Expensive with high TCO
Potential for high TCO
Decreased security and availability
Minimal mobile access
Compatibility and integration
Minimal control
Limiting infrastructure
Added complexity
Benefits
Drawbacks

What is the public cloud?

The public cloud refers to the cloud computing model in which IT services are delivered via the internet. As the most popular model of cloud computing services, the public cloud offers vast choices in terms of solutions and computing resources to address the growing needs of organizations of all sizes and verticals.

The defining features of a public cloud solution include:

- High elasticity and scalability
- A low-cost subscription-based pricing tier

Services on the public cloud may be free, freemium, or subscription-based, wherein you're charged based on the computing resources you consume.

The computing functionality may range from common services—email, apps, and storage—to the enterprise-grade OS platform or infrastructure environments used for software development and testing.

The cloud vendor is responsible for developing, managing, and maintaining the pool of computing resources shared between multiple tenants from across the network.

What is a private cloud?

The private cloud refers to any cloud solution dedicated for use by a single organization. In the private cloud, you're not sharing cloud computing resources with any other organization.

The data center resources may be located on-premise or operated by a third-party vendor off-site. The computing resources are isolated and delivered via a secure private network, and not shared with other customers.

Private cloud is customizable to meet the unique business and security needs of the organization. With greater visibility and control into the infrastructure, organizations can operate compliance-sensitive IT workloads without compromising on the security and performance previously only achieved with dedicated on-premise data centers.

What is a hybrid cloud?

The hybrid cloud is any cloud infrastructure environment that combines both public and private cloud solutions.

The resources are typically orchestrated as an integrated infrastructure environment. Apps and data workloads can share the resources between public and private cloud deployment based on organizational business and technical policies around aspects like:

- Security
- Performance
- Scalability
- Cost
- Efficiency

This is a common example of hybrid cloud: Organizations can use private cloud environments for their IT workloads and complement the infrastructure with public cloud resources to accommodate occasional spikes in network traffic.

Or, perhaps you use the public cloud for workloads and data that aren't sensitive, saving cost, but opt for the private cloud for sensitive data.

As a result, access to additional computing capacity does not require the high CapEx of a private cloud environment but is delivered as a short-term IT service via a public cloud solution. The environment itself is seamlessly integrated to ensure optimum performance and scalability to changing business needs.

When you do pursue a hybrid cloud, you may have another decision to make: whether to be homogeneous or heterogeneous with your cloud. That is—are you using cloud services from a single vendor or from several vendors?

Public Cloud vs Private Cloud vs Hybrid Cloud

Chapter 2

Introduction - Optional Resources

1. AWS Free Tier

About AWS Free Tier

The free tier applies to certain participating AWS services up to a specific maximum amount of usage each month. The AWS Free Usage Tier consists of three different types of pricing models, a 12-month Free Tier, an Always Free offer, and short term trials.

The AWS Free Usage Tier is available to everyone – students, entrepreneurs, small businesses, and Fortune 500 companies. More importantly, the AWS free usage tier is available to new AWS accounts created on or after October 21, 2010.

Services that are available in the AWS Free Usage Tier

- 750 hours of Amazon EC2 Linux or RHEL or SLES t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month
- 750 hours of an Elastic Load Balancer plus 15 GB data processing
- 750 hours of Amazon RDS Single-AZ Micro DB Instances, running MySQL, MariaDB, PostgreSQL, Oracle BYOL or SQL Server Express Edition – enough hours to run a DB Instance continuously each month. You also get 20 GB of database storage and 20 GB of backup storage
- 750 hours of Amazon ElastiCache Micro Cache Node usage – enough hours to run continuously each month.
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with EBS Magnetic) and 1 GB of snapshot storage
- 5 GB of Amazon S3 standard storage, 20,000 Get Requests, and 2,000 Put Requests
- 25 GB of Storage, 25 Units of Read Capacity and 25 Units of Write Capacity, enough to handle up to 200M requests per month with Amazon DynamoDB
- 25 Amazon SimpleDB Machine Hours and 1 GB of Storage
- 1,000 Amazon SWF workflow executions can be initiated for free. A total of 10,000 activity tasks, signals, timers and markers, and 30,000 workflow-days can also be used for free
- 100,000 Requests of Amazon Simple Queue Service
- 100,000 Requests, 100,000 HTTP notifications and 1,000 email notifications for Amazon Simple Notification Service
- 10 Amazon Cloudwatch metrics, 10 alarms, and 1,000,000 API requests
- 50 GB Data Transfer Out, 2,000,000 HTTP and HTTPS Requests for Amazon CloudFront
- 15 GB of bandwidth out aggregated across all AWS services

Note: If you are linked to an Organization (under AWS Organizations), only one account within the organization can benefit from the Free Tier offers.

How to monitor your AWS Free Usage Tier?

You receive fairly generous credits of AWS resources as part of the Free Tier and you will not be billed unless your usage exceeds those credits. Additionally, AWS has brought in a new feature the Billing and Cost Management Dashboard to help you keep better track of your AWS usage, and see where you are at with respect to the Free Tier credits for each service. It's easy to view your actual usage (month to date) and your forecasted usage (up to the end of the month).
This feature should be used to estimate and plan your AWS costs, ensuring you stay within your free tier limits. You can even receive alerts if your costs exceed a threshold that you set, which could be $0. All of this information is available to you in the AWS Billing and Cost Management Dashboard.

Limits on the AWS Free Tier

The AWS free usage tier expires 12 months from the date you sign up. When your free usage expires, you simply pay standard, pay-as-you-go service rates.
The AWS free usage tier is available to new AWS accounts created on or after October 21, 2010.
Amazon Simple Workflow Service, Amazon DynamoDB, Amazon SimpleDB, Amazon Simple Notification Service(SNS), and Amazon Simple Queue Service(SQS) free tiers are some of the services that are available to both existing and new AWS customers indefinitely.

Services not available in the AWS Free Usage Tier

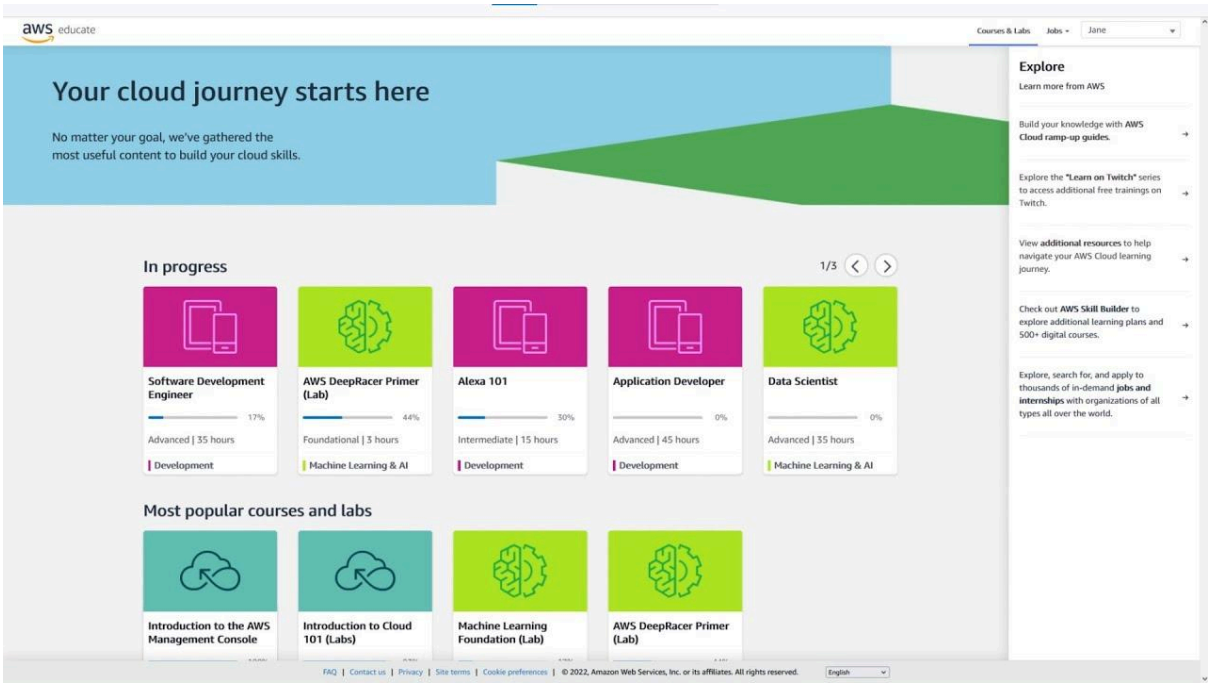
- AWS Reserved Instances
- AWS Support subscription
- Route 53 is not included so you'll still need to pay for hosted availability zones and domains
- Any service that uses an instance size, like EC2, is generally limited to a rather small instance
- Many of the amounts are high enough that you can get locked in — like 50GB Glacier storage or 1M calls/month to API Gateway
- No transfers from existing solutions are included
- You cannot use a free tier to mine for crypto. If you are caught, Amazon will charge you the normal rate and may suspend your account.
- Not all AMIs (machine images) are available so, if you plan on using a larger instance in the future, you might not be able to start with the free tier, or you might need to move your image
- The free 12 months are time-based, not usage-based so the clock is ticking after sign up
- There is no rollover of quota from one month to the next
- It does not include Amazon S3 RRS storage.

- The free tier is not restricted. There are no guard rails. If you start using the services that aren’t free, Amazon will charge you the necessary expense.

AWS Educate

AWS Educate is a free, online learning program that teaches students about cloud computing. The program is available to students with any level of experience, but it's designed particularly for preprofessionals with little or no experience working with the cloud. The AWS Educate program offers on-demand, self-directed learning content and training resources that let students practice and evaluate their cloud skills. AWS Educate is part of the larger AWS Training and Certification program. Qualified students can earn AWS certifications, gaining important proficiencies as they enter the information technology (IT) workforce.
To qualify, AWS Educate candidates must meet their home country's minimum age requirement. Previously, the program was designed as a grant program and was only available to educators with a .edu email address and military veterans, enabling them to receive AWS credits.

How does AWS Educate work?



AWS Educate offers hundreds of hours of self-paced training and labs that give students hands-on practice with the AWS console. Students get a fixed number of hours to complete a lab. However, they get unlimited tries, as long as they complete the lab in the allotted time. As students complete courses, they earn learner badges that show their understanding of a skill.

Start your Cloud Journey with AWS Educate | Amazon Web Services

Chapter 3

Introduction - Six Advantages Of Cloud computing

You will learn how to:

- Identify the advantages of cloud computing
- Analyze the difference between capex and variable expense
- Summarize the concept of Economies of Scale

You will discuss:

- The benefits of cloud computing
- The advantages of moving to the cloud

Key terms:

• Content delivery network (CDN) • Amazon CloudFront
• Capital expense (capex)
• Variable expense
• Economies of scale
• Edge locations

Benefits of adopting the cloud

According to AWS, there are six main benefits of adopting the cloud:

- Trade capital expense for variable expense
- Benefit from massive economies of scale
- Stop guessing capacity
- Increase speed and agility
- Stop spending money running/maintaining data centers
- And go global in minutes

Advantage 1: Capex to variable expense

Trade capital expense for variable expense

Capex to variable expense
Instead of having to invest heavily in data centers and servers before you know how you're going to use them, you can pay only when you consume computing resources, and pay only for how much you consume.

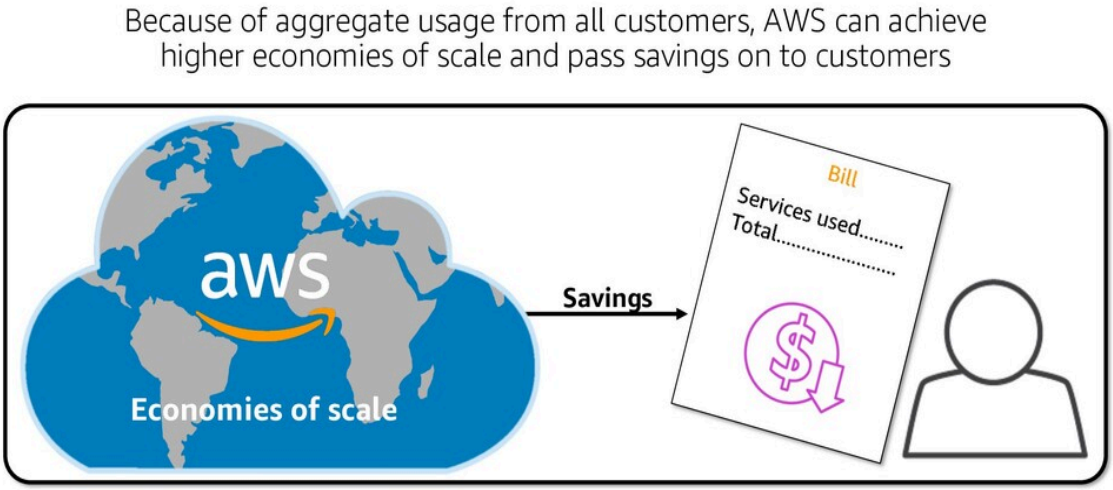
Real-World Example:
A big part of my time working for MSPs was spent preparing quotes for upgrades or new builds. Looking back, I spent a ridiculous amount of time reviewing ESXi hosts, SAN storage, and other hardware my clients had to purchase upfront. All of this was due to the fact we never wanted to run out of space, or not have enough compute power to run proper workloads.
Shifting to the cloud, instead of purchasing expensive SAN storage upfront, clients moved to storage services like AWS S3 which allowed them to pay only for what they used. This avoided large hardware purchases. Many CFOs I've worked with prefer this model aligns with a cost-effective and agile approach to resource allocation and allows the business to adapt to changing needs and optimize their budgets.

Capital expense (capex):
• Funds that a company uses to acquire, upgrade, and maintain physical assets such as property, industrial buildings, or equipment.

Variable expense
• An expense that the person who bears the cost can easily alter or avoid.

6 Advantages of Cloud Computing

Advantage 2: massive economies of scale

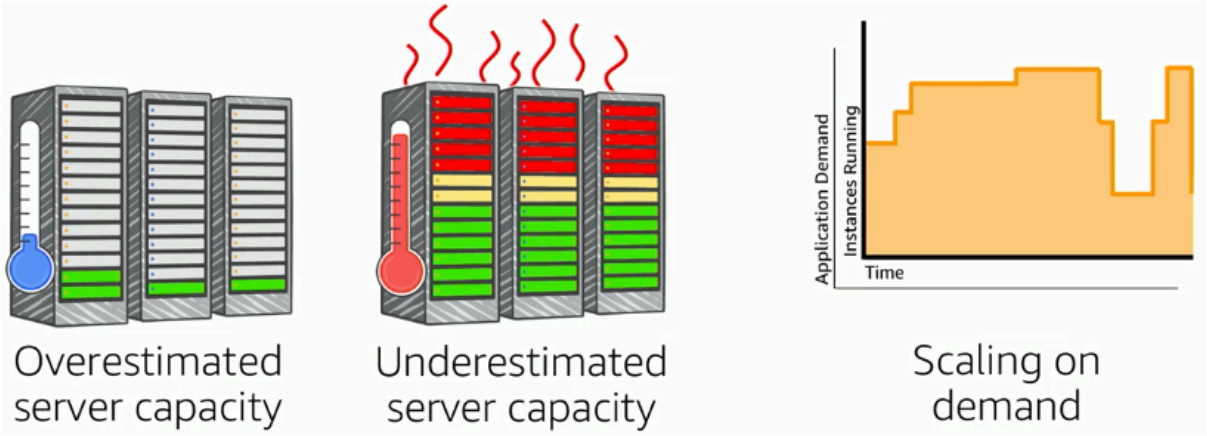


When it comes to the cloud, imagine hundreds of thousands of customers seeking the same services. This allows providers like AWS to achieve higher economies of scale and by consequence lower pay-as-you-go prices, which are difficult or even impossible for companies to achieve such low variable costs in their own data centers.

Real-World Example:

Rather than massive upfront infrastructure investments, customers take advantage of AWS's vast economies of scale from its millions of users to obtain on-demand access to a wide range of enterprise-level solutions at bargain prices. One manufacturing client I worked with was able to reduce IT costs by over 50% by shutting down their data centers and leveraging AWS's infrastructure. My consulting clients enjoyed major cost reductions, efficiency gains, and agility by tapping into AWS's customer base.

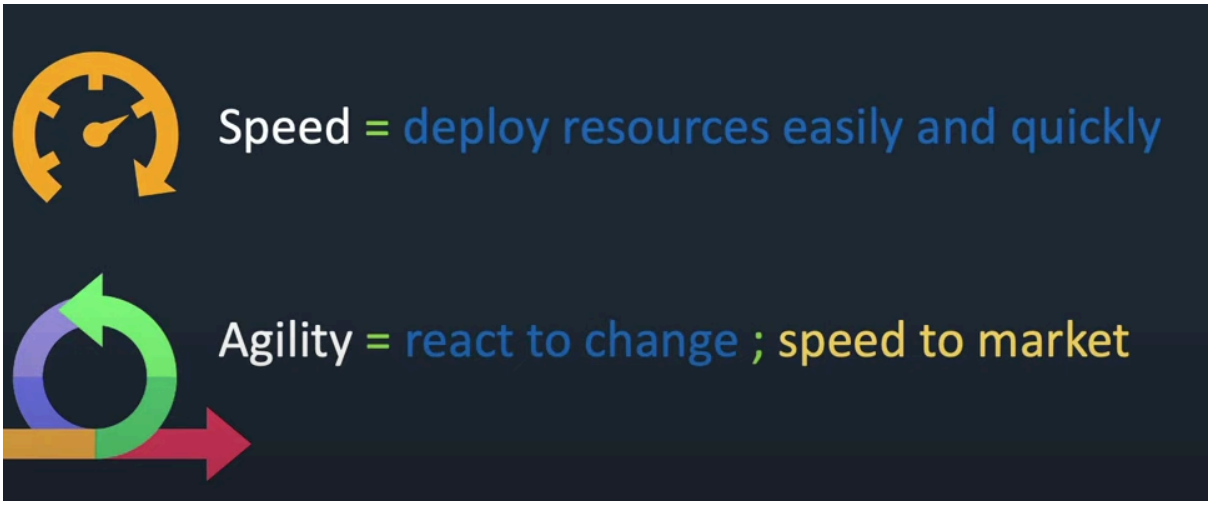
Advantage 3: Stop guessing capacity



Eliminate guessing on your infrastructure capacity needs. When you make a capacity decision before deploying an application, you often end up either sitting on expensive idle resources or dealing with limited capacity. With cloud computing, these problems go away. You can access as much or as little capacity as you need, and scale up and down as required with only a few minutes' notice.

Real-World Example:
Many clients faced issues with guessing the needed capacity in terms of servers, storage, and bandwidth. Every time, I had to overprovision the resources in case there was a spike in usage and to make the hardware last its normal lifecycle. By leveraging AWS's scalable resources, my clients were able to adapt on the fly to changing business needs without big forecasts. One media client easily handled a 10X spike in traffic to its site by leveraging AWS's autoscaling, avoiding previous crashes under peak loads.

Advantage 4: Increase speed and agility

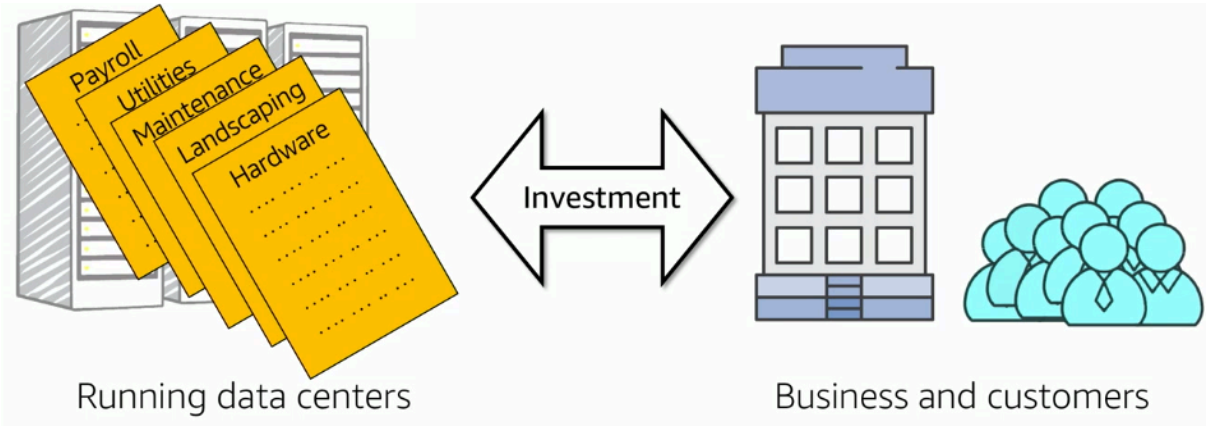


In a cloud computing environment, new IT resources are only a click away, which means that you reduce the time to make those resources available to your developers from weeks to just minutes. This results in a dramatic increase in agility for the organization since the cost and time it takes to experiment and develop is significantly lower.

Real-World Example:

The ability to instantly spin up resources is a huge benefit to migrating resources to the cloud. One startup client was able to rapidly test and iterate their MVP in a matter of weeks and launch months faster by leveraging AWS instead of traditional on-premises procurement cycles.

Advantage 5: Stop spending money running and maintaining data center



Focus on projects that differentiate your business, not the infrastructure. Cloud computing lets you focus on your customers, rather than on the heavy lifting of racking, stacking, and powering servers.

Real-World Example:

Freeing clients from the overhead of managing their infrastructure and facilities has been a major win. One retailer I worked with was able to reallocate IT resources to customer-focused projects after migrating fully to AWS cloud-managed services.

Advantage 6: Go global in minutes

Easily deploy your application in multiple regions around the world with just a few clicks. This means you can provide lower latency and a better experience for your customers at a minimal cost.

Real-World Example:

AWS's global footprint has enabled clients to easily expand internationally on demand versus the significant investment previously needed. One of my previous clients rolled out targeted campaigns in Europe and Asia within days by leveraging AWS edge locations and cloud infrastructure versus months previously.

Conclusion

By leveraging AWS's global scale and on-demand model, organizations can gain agility, reduce costs, and focus IT resources on innovation versus maintenance. I've seen firsthand how small businesses and large enterprises alike have benefited from moving to the cloud - faster time to market, expanded global reach, and greater adaptability to changing needs. At the end of the day, AWS cloud delivers tangible advantages by allowing companies to tap into world-class infrastructure and services without massive capital investments. The numbers speak for themselves, as client after client achieves their goals by embracing the flexibility and opportunities of cloud computing.

Chapter 4

Introduction - Amazon Web Services

You will learn how to:

- Explain in general what a web service is
- Explore services that Amazon Web Services (AWS) offers
- Examine ways to access AWS services You will discuss:

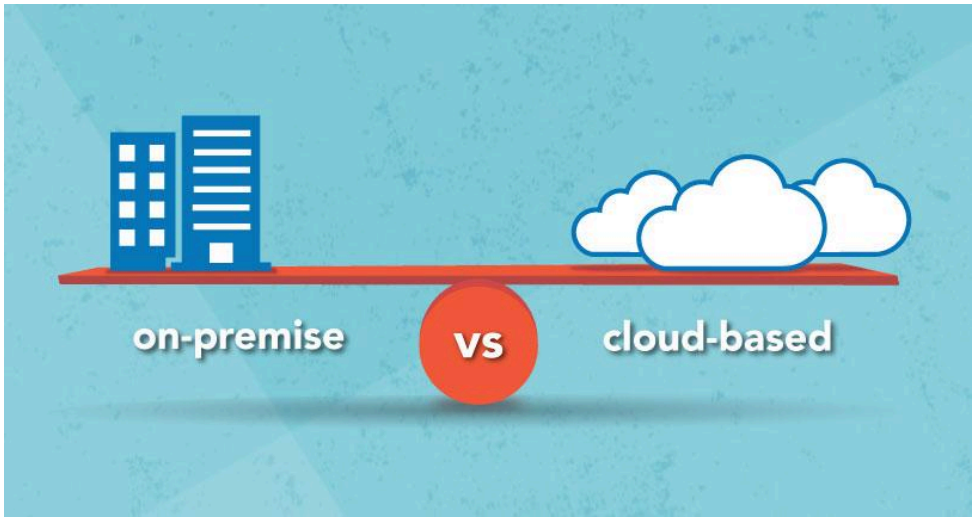
You will discuss:

- What web services are

Key terms:

- AWS Core Services
- AWS Foundational Services
- Software development kits (SDKs)
- AWS Command Line Interface (AWS CLI)

On premises and AWS comparison

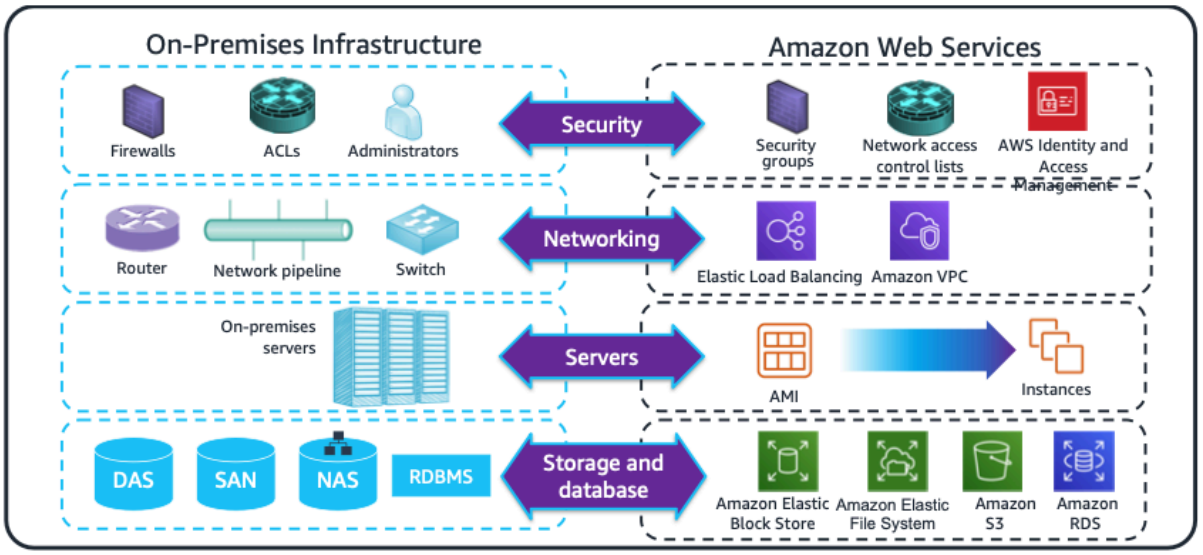


Let us first understand the meaning of the words On-Premises and On Cloud.

On-Premises : In on-premises, from use to the running of the course of action, everything is done inside; whereby backup, privacy, and updates moreover should be managed in-house. At the point when the item is obtained, it is then installed on your servers, requiring additional power laborers, database programming software and operating systems to be purchased. With no prior commitment, you anticipate complete ownership.

On Cloud : Cloud refers to the delivery of on-demand computing services over the internet on “Pay As U Use “services, in simple words rather than managing files and Services on the local storage device you can do the same over the Internet in a cost-efficient manner. With a Cloud-based enrolment model, there is no convincing motivation to purchase any additional establishment or licenses.

Now let's explore the differences between on-premise and cloud solutions, particularly focusing on AWS (Amazon Web Services), in terms of *Security, Networking, Servers, and Storage/Databases.*



Security:

On-Premise:

- Security is managed entirely by the organization's IT team.
- Physical security measures such as access control systems, surveillance cameras, and security personnel are crucial.
- Organizations are responsible for ensuring compliance with regulations and implementing security patches and updates.

AWS (Cloud):

- Security is a shared responsibility between AWS and the customer. AWS manages security of the cloud, while customers are responsible for security in the cloud.
- AWS provides a range of security services, such as AWS Identity and Access Management (IAM), AWS Key Management Service (KMS), and AWS WAF (Web Application Firewall).

Example: AWS Key Management Service (KMS) for managing encryption keys securely.

Networking:

On-Premise:

- Networking infrastructure is physically owned and maintained by the organization.
- Scaling may require purchasing additional hardware, which can be time-consuming and expensive.
- Organizations have direct control over the network configuration and settings.

AWS (Cloud):

- Networking is virtualized and managed by AWS.
- AWS offers services like Amazon VPC (Virtual Private Cloud) for creating isolated networks and Amazon Route 53 for domain registration and DNS.
- Scaling is dynamic and can be achieved through AWS services like Auto Scaling.

Example: Amazon VPC for creating a logically isolated section of the AWS Cloud.

Servers:

On-Premise:

- Physical servers are located on-site, and organizations are responsible for their maintenance, upgrades, and scalability.
- Server provisioning can be time-consuming and may involve purchasing new hardware.

AWS (Cloud):

- Virtual servers (EC2 instances) are used, and AWS manages the underlying hardware.
- EC2 instances can be provisioned and scaled quickly based on demand.
- Various instance types are available for different workloads.

Example: Amazon EC2 (Elastic Compute Cloud) for scalable compute capacity in the cloud.

Storage/Databases:

On-Premise:

- Organizations manage their own storage infrastructure, including backups and redundancy.
- Scalability may involve adding physical storage devices.

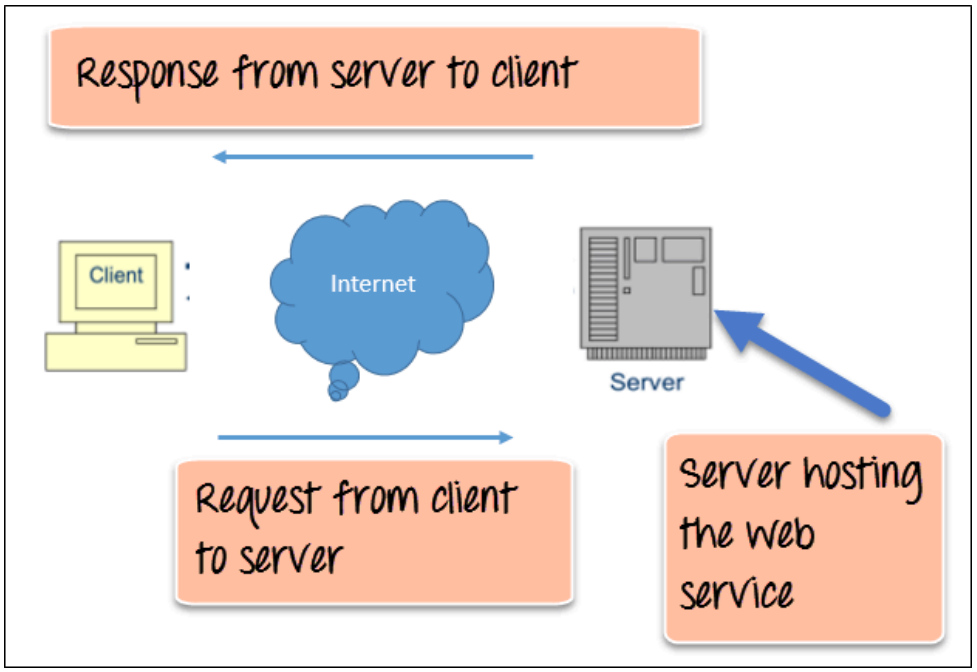
AWS (Cloud):

- AWS offers scalable storage solutions like Amazon S3 (Simple Storage Service) and Amazon EBS (Elastic Block Store).
- Managed database services like Amazon RDS (Relational Database Service) and Amazon DynamoDB simplify database administration.
- Storage can be dynamically scaled based on demand.

Example: Amazon S3 for scalable and durable object storage.

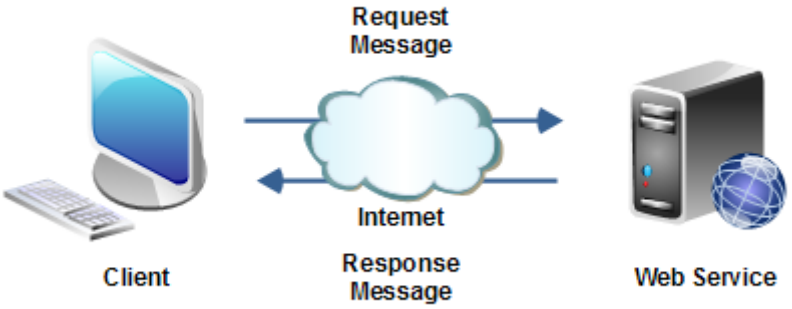
In summary, while on-premise solutions provide direct control over infrastructure, they may lack the scalability and flexibility of cloud solutions. AWS, being a leading cloud service provider, offers a wide range of services that allow organizations to focus on their applications and data rather than the underlying infrastructure. The choice between on-premise and cloud often depends on specific business requirements, cost considerations, and the need for scalability and flexibility

What are Web Services



A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service.

For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.



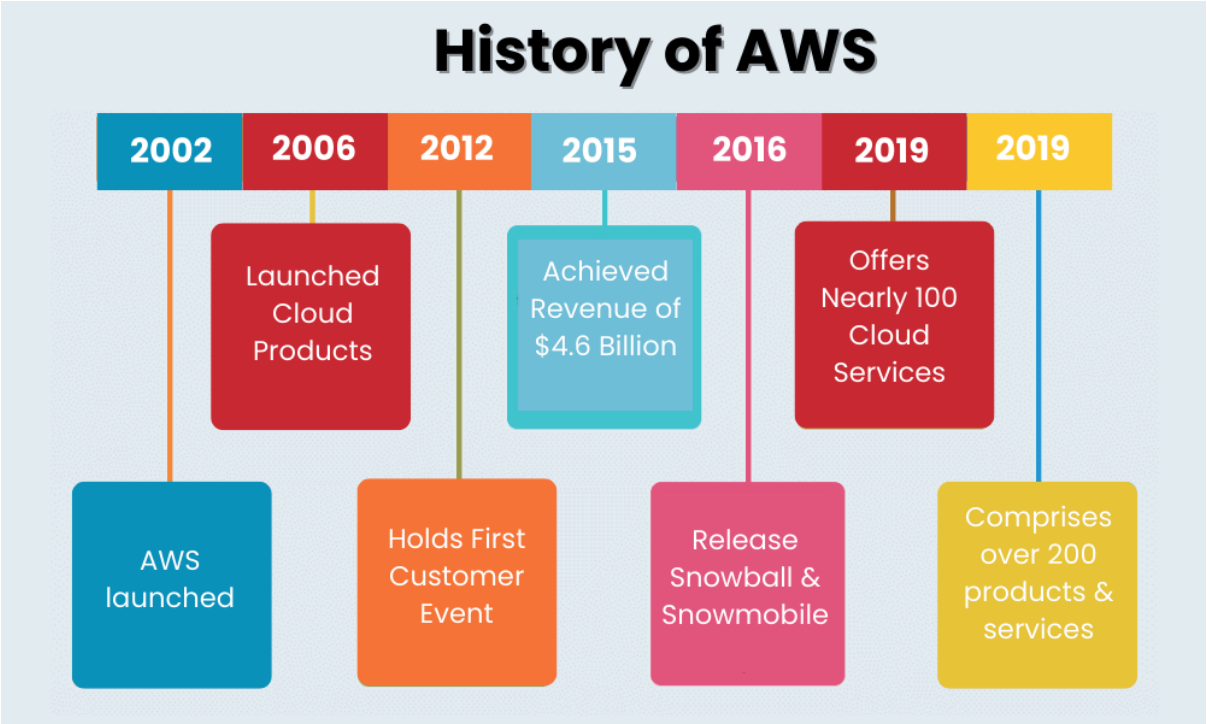
What is AWS ?

Amazon Web Services (AWS) offers pay-as-you-go cloud computing platforms and APIs to people, businesses, and governments. AWS server farms provide a variety of networking, compute, storage, middleware, IOT, and other processing capacity services, as well as software tools. These cloud computing web services free clients from managing, scaling, and patching hardware and operating systems. AWS has become the most successful cloud infrastructure company on the planet, garnering more than 30 percent of the market.

In this part, we will be exploring the basics of Amazon Web Services.

History

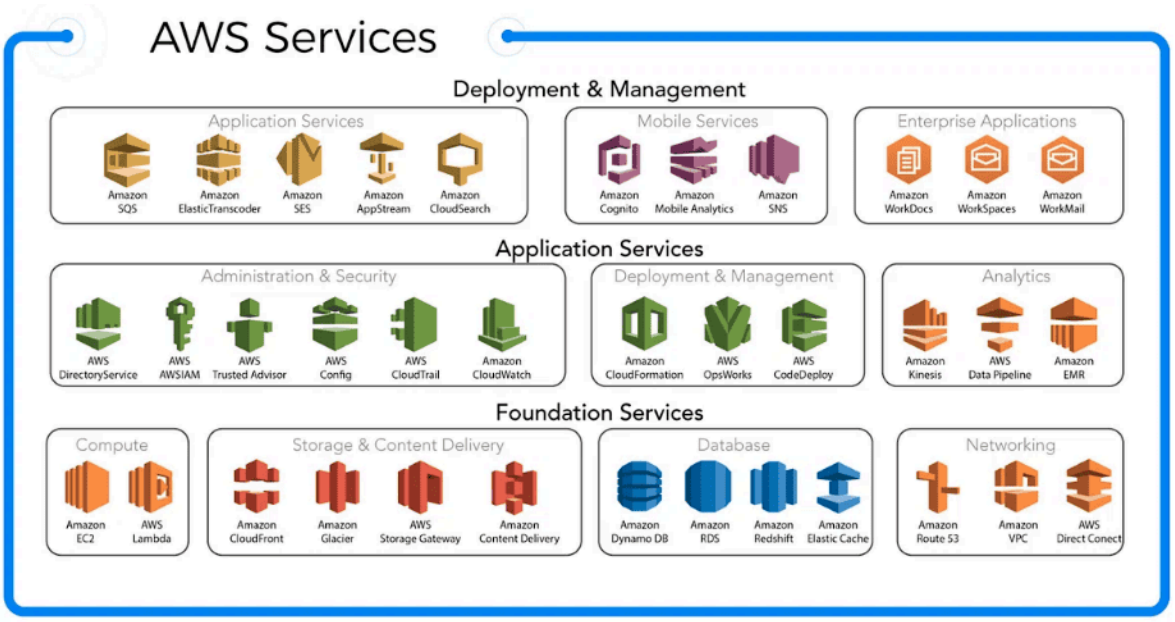
The origins of AWS as a developer tool can be traced all the way back to 2002 when an initial beta was released (named Amazon.com Web Service) that offered SOAP and XML interfaces for the Amazon product catalogue1. In 2006, AWS was officially launched with two core services: Amazon Elastic Compute Cloud (EC2), which allows users to have at their disposal a virtual cluster of computers, and Amazon Simple Storage Service (S3), which provides online storage space. Amazon had almost 180,000 developers sign up in 2007. Amazon.com's retail web services were transferred to AWS in 2010, therefore Amazon.com is currently running on AWS. Since then, Amazon has expanded its portfolio of services and regions and has attracted millions of customers from startups to enterprises to public sector organizations.



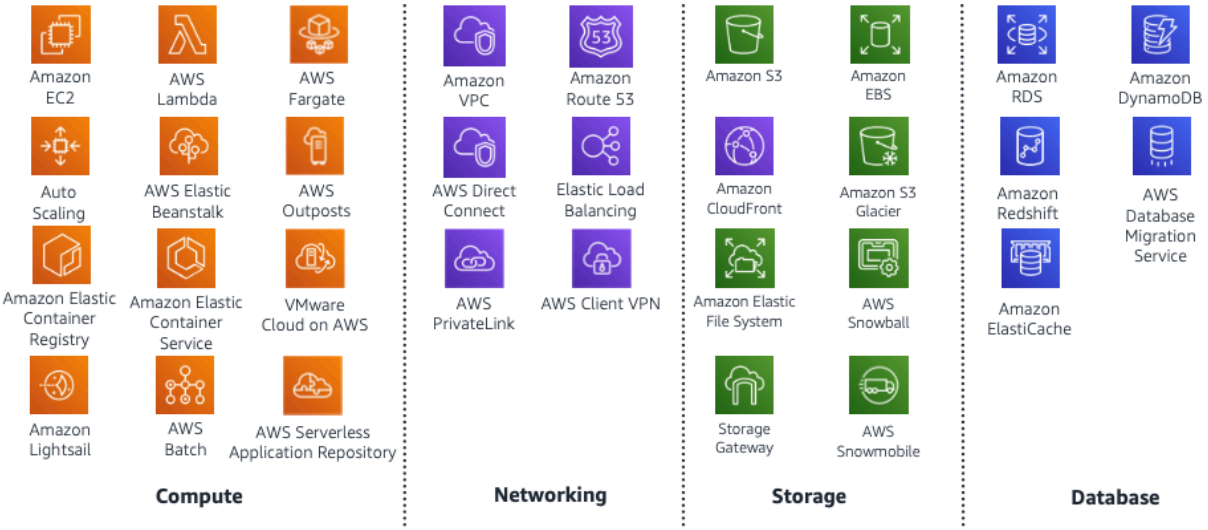
AWS Introduction

AWS is a cloud computing platform that provides various services that can be used to build and deploy applications on the Internet. AWS allows users to access these services through web-based interfaces, command-line tools, or software development kits (SDKs). Users can choose from a wide range of services that cover different aspects of cloud computing, such as:

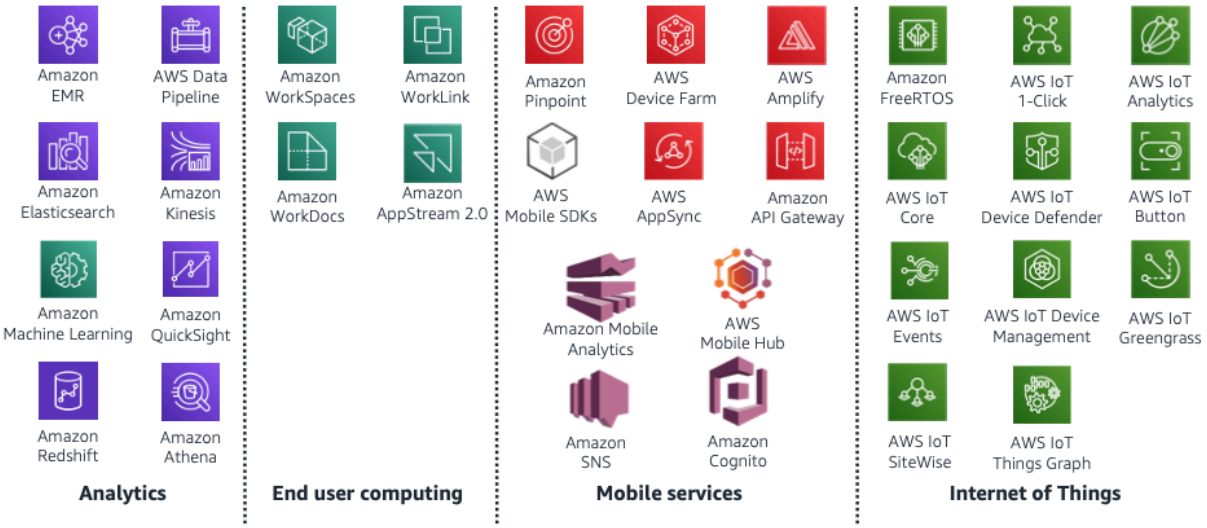
- **Compute**: Services that provide processing power for running applications and workloads. Examples include Amazon EC2, Lambda, and Amazon Elastic Container Service (ECS).
- **Storage**: Services that provide online storage space for data and objects. Examples include Amazon S3, Amazon Elastic Block Store (EBS), and Amazon Glacier.
- **Networking**: Services that enable users to connect their applications and resources across different regions and locations. Examples include Amazon Virtual Private Cloud (VPC), Amazon Route 53, and AWS Direct Connect.
- **Middleware**: Services that provide common functionality and features for applications. Examples include Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS), and Amazon Relational Database Service (RDS).
- **IOT**: Services that enable users to connect and manage devices and sensors on the internet. Examples include AWS IoT Core, IoT Greengrass, and IoT Analytics.
- **Other**: Services that provide specialized capabilities for specific use cases or domains. Examples include Amazon Machine Learning, Amazon Lex, and Amazon Polly.



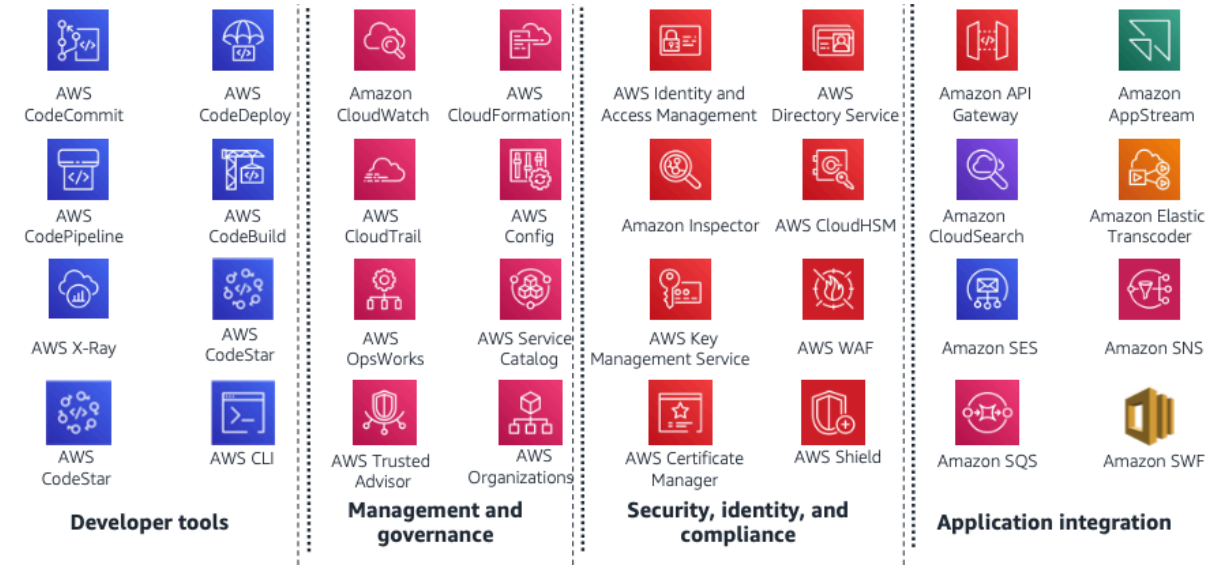
AWS by category: Core services



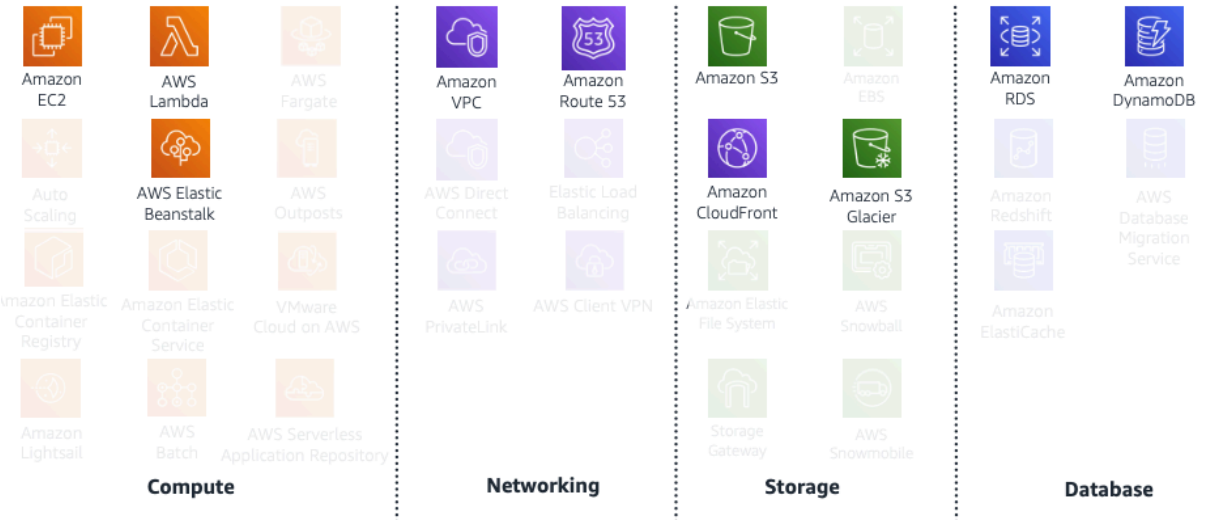
AWS by category: Foundational services



AWS by category: Developer and operations services

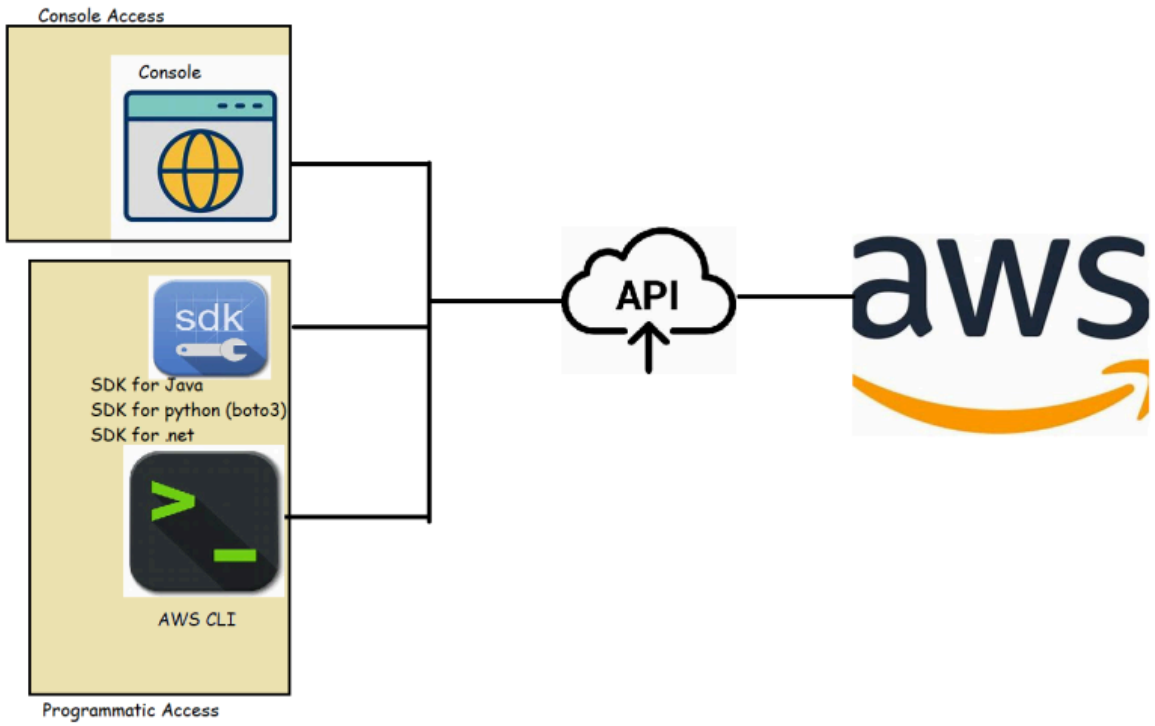


Core services: Focus on key services



Access to AWS services

1. **AWS Management Console**
2. **AWS Command Line Interface (AWS CLI)**
3. **Software development kits (SDK)**



There are a few different ways to access AWS services.

1. One way is to use the AWS Management Console, which is a web-based interface that provides access to each service console and offers a single place to access the information you need to perform your AWS-related tasks. You can also customize the Console Home page by adding, removing, and rearranging widgets, such as recently visited, AWS Health, Trusted Advisor, and more.

2. Another way is to use the AWS Command Line Interface (CLI), which is a tool that allows you to interact with AWS services from a command prompt. You can use the AWS CLI to automate common tasks and scripts, as well as to access advanced features that are not available in the console. The AWS CLI supports multiple platforms, such as Windows, Linux, and macOS.

3. A third way is to use Amazon Software Development Kits (SDKs), which are libraries that enable you to integrate AWS services into your applications using your preferred

programming language. Amazon SDKs provide high-level APIs that simplify common tasks, such as authentication, request signing, and error handling. Amazon SDKs are available for various languages, such as Java, Python, Ruby, .NET, Node.js, and more.

AWS Access Keys, CLI and SDK Introduction

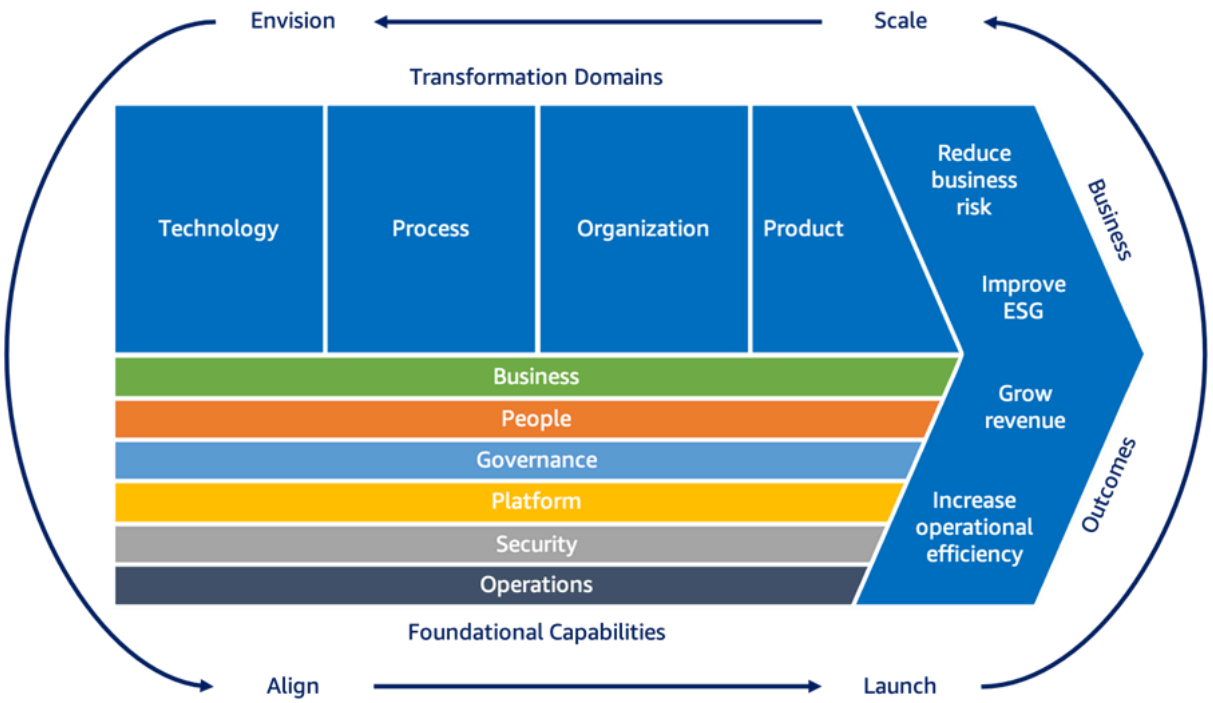
Chapter 5

Cloud Adoption Framework

AWS Cloud Adoption Framework (AWS CAF)

The AWS Cloud Adoption Framework, or AWS CAF for short, is simply a framework provided by AWS to assist you in adopting cloud computing for your enterprise infrastructure. It is a framework that contains various perspectives that are based on years of extensive experience and best practices in AWS. This can help you digitally transform and accelerate your digital transformation as well as business outcomes through the innovative use of the AWS Cloud.

AWS CAF zeroes in on specific organizational capabilities that are vital in successful cloud transformations. The capabilities and perspectives of this framework provide best-practice guidance that assists companies in improving their total cloud readiness.



Business Perspective

Instead of making specific strategies to achieve specific business benefits, you may utilize the AWS Cloud Adoption Framework (AWS CAF) to establish a global business model with a long-term vision to expand and adapt to changing market demands. Managers and business owners will gain from this perspective in that they will:

- Give excellent planning opportunities,
- Improve end-user wants satisfaction,
- Optimize the distribution of funds, and
- Control company risks from beginning to end, both external and strategic.

Governance Perspective

This perspective aids in the organization and coordination of non-IT business processes and software. IT managers, architects, and business analysts will find it useful because:

- Scheduling cloud workloads and correctly setting service priorities;
- Selecting the appropriate cloud migration techniques;
- Cost reduction;
- Licensing administration.

Operations Perspective

This AWS Cloud Adoption Framework perspective assists in managing workloads to the current and future needs of the business. You may, therefore, quickly create and carry out a new business strategy with little risk. During this stage, the following will be advantageous for operators in technical support:

- Service monitoring to determine the IT operations' level of efficiency and to ensure corporate standards are met when using cloud-based applications;
- Release/change management for a wise choice of continuous integration/continuous delivery (CI/CD) techniques, resource inventory management to rationalize the use of virtual IT assets;
- Reporting and analytics for tracking performance;
- Establishing backups and evaluating downtime for ongoing business operations.

Security Perspective

This perspective of the AWS Cloud Adoption Framework aids in managing workloads by present and foreseeable business requirements. You may quickly create and carry out a new business strategy with little risk. These are the security advantages they receive from the AWS Cloud Adoption Framework:

- IAM (Identity and Access Management), which offers thorough instructions on incorporating AWS into user authentication and authorization operations;
- Detecting suspect network activity is the responsibility of detective control;
- Infrastructure security, which is in charge of creating rules and security specifications;
- Data protection, which aids in the development of effective methods for securing data during transport and storage operations;
- Service for an incident response that enables quick reaction to unusual network activity.

People Perspective

HR professionals can better prepare their staff for the move to the cloud by taking this into account. In particular, HR specialists receive:

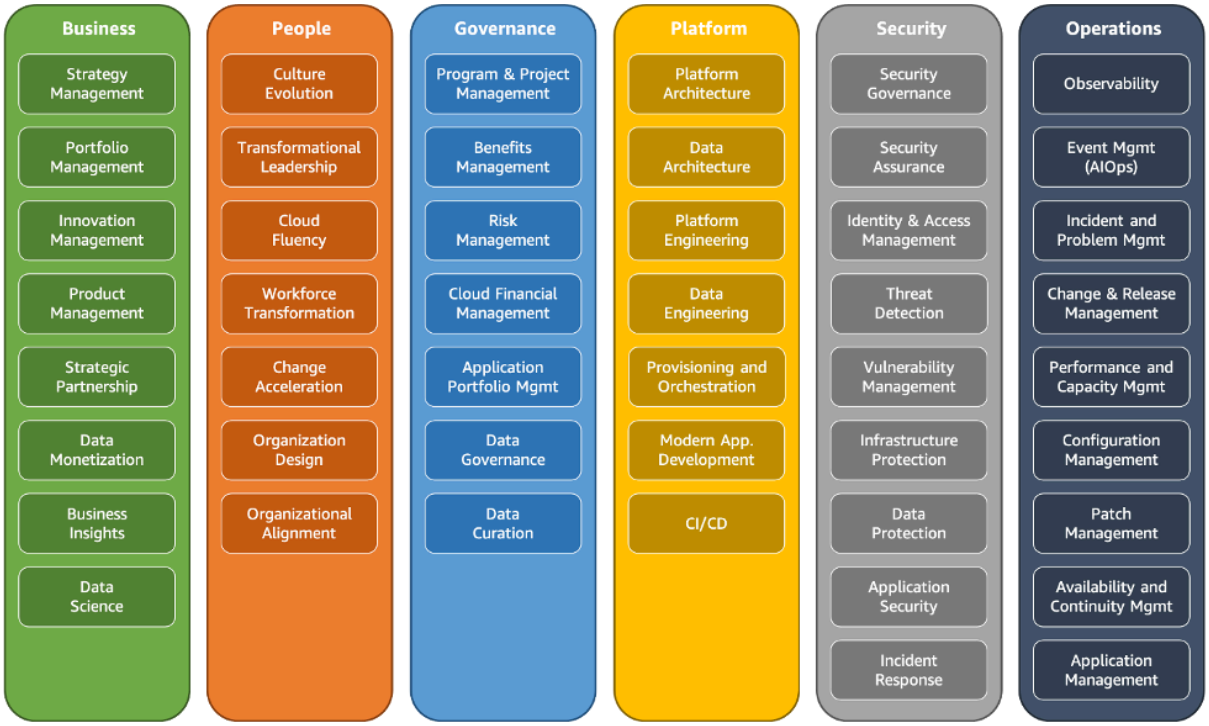
- The improved network architecture's capacity to forecast new openings;
- Putting together a shared set of abilities for cloud migration;

- Creating a training program to prepare staff to perform their regular duties in a modern setting.

Platform Perspective

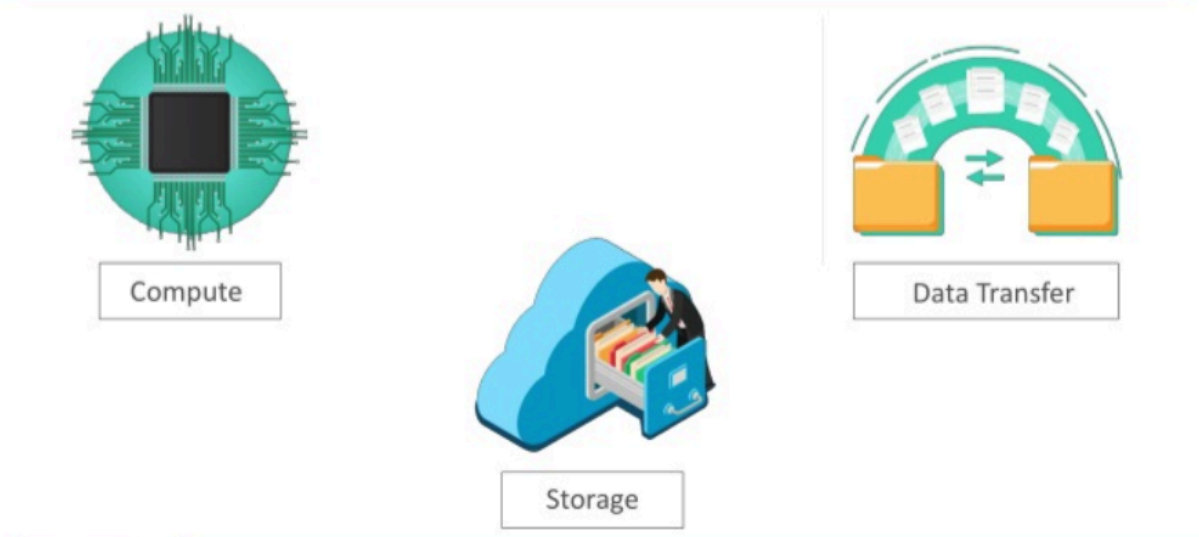
The platform perspective enhances the network infrastructure built using AWS. For network architects, CTOs, and managing IT team members, it is essential. These are the advantages they receive:

- Knowing the specifications that the system's design should adhere to;
- The outlook for potential growth and the development of new cloud-based workflows;
- Coordinating new business objectives with the increased processing power provided by the cloud.



Chapter 6:Cloud Economics - AWS Pricing

AWS Pricing fundamentals



1. **Pay for AWS fundamentals:**

    - Compute
    - Storage
    - Outbound data transfer

2. **No charge:**

    - Inbound data transfer
    - Data transfer between services within the same region

3. **Charge for aggregated outbound**

There are three fundamental drivers of cost with AWS: compute, storage, and outbound data transfer. These characteristics vary somewhat, depending on the AWS product and pricing model you choose. In most cases, there is no charge for inbound data transfer or for data transfer between other AWS services within the same Region. There are some exceptions, so be sure to verify data transfer rates before beginning. Outbound data transfer is aggregated across services and then charged at the outbound data transfer rate. This charge appears on the monthly statement as AWS Data Transfer Out. The more data you transfer, the less you pay per GB. For Compute resources, you pay by the hour or by the second from the time you launch a resource until the time you stop or terminate it, unless you have made a reservation for which the cost is agreed upon beforehand. For data storage and transfer, you typically pay per GB.

AWS pricing model:

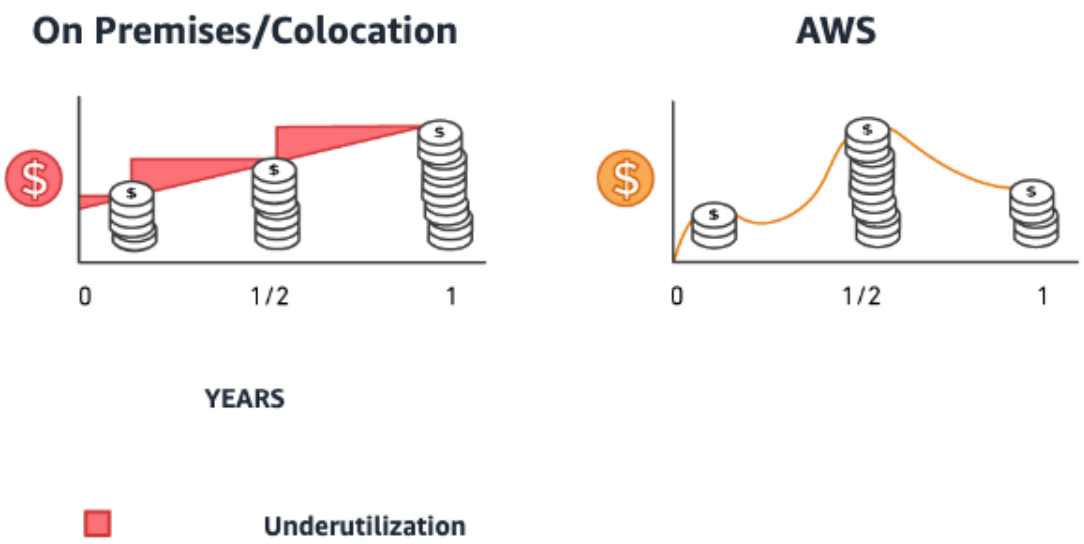
While the number and types of services offered by AWS have increased, the AWS philosophy on pricing has not changed.

At the end of each month, you pay for what you use. You can start or stop using a product at any time. No long-term contracts are required.

Advantages of the model:

- Pay as You go
- Pay less when you reserve
- Pay less when you use more
- Pay less as AWS grows

1. Pay as You Go



With AWS you only pay for what use, helping your organization remain agile, responsive and always able to meet scale demands.

Pay-as-you-go pricing allows you to easily adapt to changing business needs without overcommitting budgets and improving your responsiveness to changes. With a pay as you go model, you can adapt your business depending on need and not on forecasts, reducing the risk or overprovisioning or missing capacity.

Benefits

By paying for services on an as needed basis, you can redirect your focus to innovation and invention, reducing procurement complexity and enabling your business to be fully elastic.

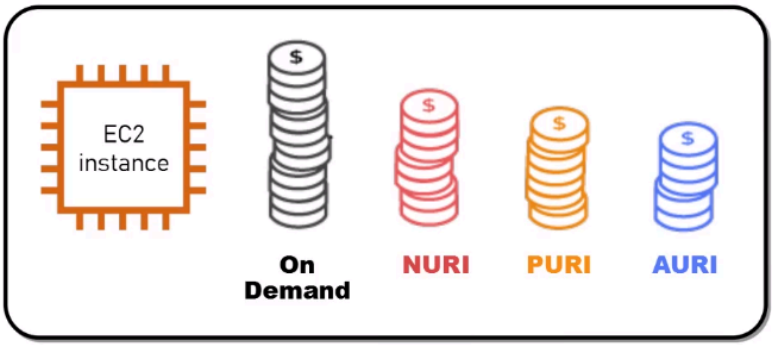
2. Pay less when you reserve

Savings Plans is a flexible pricing model that provides significant savings on your AWS usage. This pricing model offers lower prices on AWS Compute and AWS Machine Learning. Savings Plans offer savings over On-Demand in exchange for a commitment to use a specific amount (measured in $/hour) of an AWS service or a category of services, for a one- or three-year period.

You can sign up for Savings Plans for a 1- or 3-year term and easily manage your plans by taking advantage of recommendations, performance reporting, and budget alerts in the AWS Cost Explorer.

Invest in Reserved Instances (RIs):

- Save up to 75 percent
- Options:
  - All Upfront Reserved Instance **(AURI)** → **largest discount**
  - Partial Upfront Reserved Instance **(PURI)** → **lower discounts**
  - No Upfront Payments Reserved Instance **(NURI)** → **smaller discount**



What Is PURI NURI AURI?

PURI NURI AURI are just that: ways to structure your cloud account to minimize expenses. With AWS reserved instances (RI), you pre-pay for a commitment to use an Amazon Elastic Compute (EC2) or Relational Database Service (RDS) instance for a 12 or 36 month term. In exchange, AWS discounts the price of the RI (up to 75%) as compared to regular On-Demand services. Reserved instances are ideal for applications that have a steady state and predictable usage patterns.

You can choose to pay for reserved instances in one of three ways:

PURI: Partial Upfront Reserved Instance

At the start of the term, you pay a portion of the RI's cost and the remaining hours in the term are billed at a discounted hourly rate, regardless of whether the Reserved Instance is being used.

NURI: No Upfront Reserved Instance

Without paying ahead, you are billed a discounted hourly rate for every hour within the term, regardless of whether the Reserved Instance is being used.

AURI: All Upfront Reserved Instance

At the start of the term, you pay in full, with no other costs or additional hourly charges incurred for the remainder of the term, regardless of hours used.

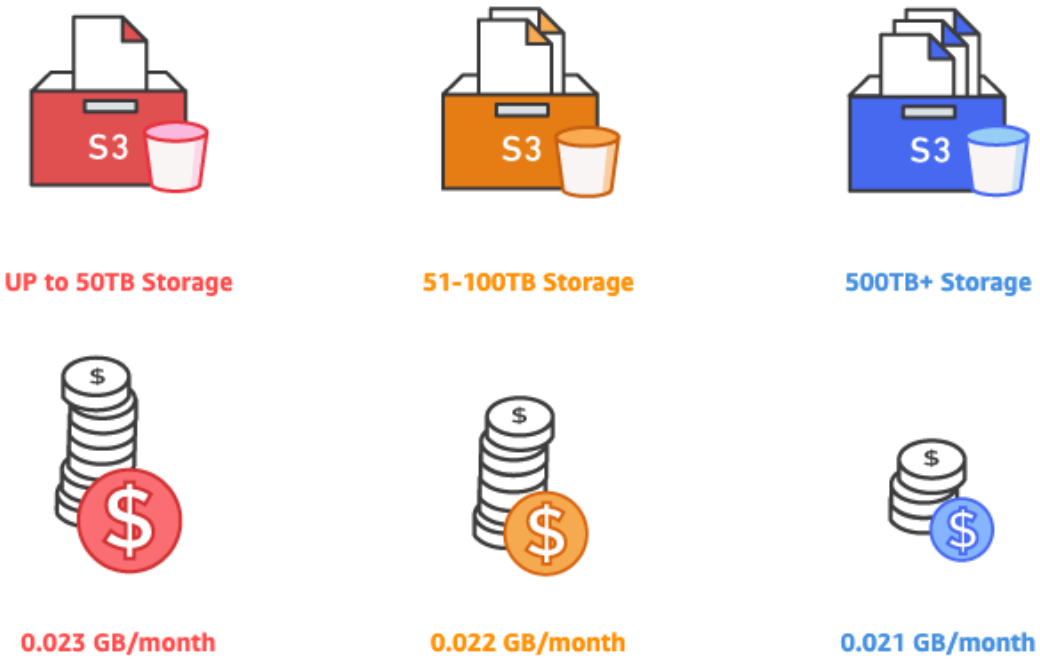
Conclusion

The best way to optimize AWS costs is to let your usage data guide you. If you have a new system, it's safest to start with On-Demand pricing. Once you've established a baseline, you can choose to use Reserved Instances, and then PURI, NURI, or AURI…

3. Pay less by using more

With AWS, you can get volume based discounts and realize important savings as your usage increases. For services such as S3 and data transfer OUT from EC2, pricing is tiered, meaning the more you use, the less you pay per GB. In addition, data transfer IN is always free of charge. As a result, as your AWS usage needs increase, you benefit from the economies of scale that allow you to increase adoption and keep costs under control.

As your organization evolves, AWS also gives you options to acquire services that help you address your business needs. For example, AWS' storage services portfolio, offers options to help you lower pricing based on how frequently you access data, and the performance you need to retrieve it. To optimize your savings, choose the right combinations of storage solutions that help you reduce costs while preserving performance, security and durability.



4. Pay even less as AWS Grows

- AWS focuses on lowering cost of doing business
- This pratice results in AWS passing savings from economies of scale to you
- Since 2006, AWS has lowered pricing 75 times (as of Septembre 2019)
- Future higher-performing resources replace current resources for no extra charge

Custom pricing

- Meet varying needs through custom pricing.
- Available for high-volume projects with unique requirements.

Free Usage Tier

AWS offers new customers the option to use many services for free, for the duration of one year or until you reach the allotted quota. For example, you can get a maximum of 750 hours worth of EC2 instances or 5 GB of storage on Amazon Simple Storage Service (Amazon S3). However, to ensure you properly leverage free resources, you should plan your budget in advance. AWS offers a pricing calculator that can help you estimate your costs.
Now that we have introduced AWS pricing principles, let's dive into each one of their actual pricing models.

AWS Free Tier

An important part of Amazon's pricing is the AWS Free Tier, which allows organizations to try Amazon services at no cost. The Free Tier has three levels:

- 12 months free—a bundle of services that a new Amazon user can use for a year from their initial registration. Users pay the regular price after 12 months, or if they exceed the Free Tier limits on any of the services.
- **Always free**—this is a more limited bundle of services that all AWS users can use for free, even after their initial 12 months expire.
- **Trials**—short term trial for a specified period after activating an Amazon service

The following table shows key services available in the 12 month and always free tiers. To see the full list of services for all three tiers see the official free tier pricing.

	12 Months Free	Always Free
Compute	750 hours of EC2 Linux t2.micro instance and Elastic Load Balancer (ELB)	

Database	750 hours of Amazon Relational Database Service (RDS)* with 20GB storage and 20GB backup 25GB of storage and 25 Units of Read/Write Capacity on DynamoDB	25GB of storage on DynamoDB
Caching	750 hours of Amazon ElastiCache Micro Cache Node	
Storage	EBS volume with 30GB, GB of storage on S3, 1 GB of snapshot storage	
Serverless	(same as Always Free)	1 million requests per month on AWS Lambda
Messaging	(same as Always Free)	1M publishes on Amazon SNS
Logging	(same as Always Free)	10 custom metrics and alarms on CloudWatch
Archival	(same as Always Free)	10GB of retrieval from Amazon Glacier

(*) Amazon RDS provides access to free database engines including MySQL, PostgreSQL, and SQL Server Express Edition.

Services with no charge

- Amazon Virtual Private Cloud (Amazon VPC)
- AWS Identity and Access Management (IAM)

- Consolidated Billing
- AWS Elastic Beanstalk**
- AWS CloudFormation**
- Amazon EC2 Auto Scaling**
- AWS OpsWorks**

**Note: There may be charges associated with other AWS services used with these services.

Chapter 7

Cloud Economics - Cost of Ownership

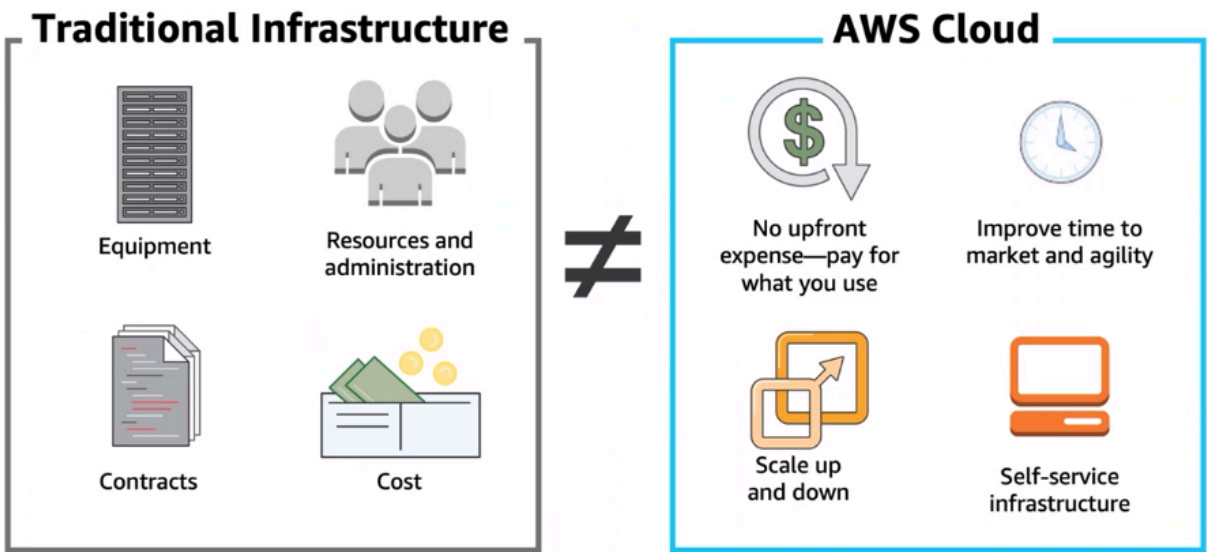
Total cost of Ownership

What is Total Cost of Ownership (TCO) ?

Total Cost of Ownership (TCO) is the financial estimate to help identify direct and indirect costs of a system.

Why use TCO ?

- To compare the costs of running an entire infrastructure environmnet of specific workload on-premises versus on AWS
- To budget and build the business case for moving to the cloud



Cloud TCO involves calculating the costs required to host, run, integrate, secure and manage workloads in the cloud over their lifetime. These include fees associated with the underlying infrastructure, such as compute, data transfer and storage. It also includes the cost of supporting cloud services, ranging from security and management tools to data analytics. Manpower costs for cloud engineers should also be part of a cloud TCO equation. Read on to learn how to estimate your AWS costs, understand the TCO of your current on-premises infrastructure, and understand the financial viability of migrating to the AWS cloud.

Calculating Total Cost of Ownership on AWS

AWS costs depend on three key components: compute, outbound data transfer, and storage. The AWS pricing model and the product's specific characteristics determine the charges for each of these components.

Data transfer

Generally, AWS does not charge for inter-service data transfer within a region or inbound transfer. However, there are certain exceptions, so it's important to check the data transfer costs before starting. AWS calculates your aggregated outbound data transfer across all services and charges based on the specified rate. The monthly AWS statement specifies this charge under "Data Transfer Out." The amount of data transferred determines the cost per GB.

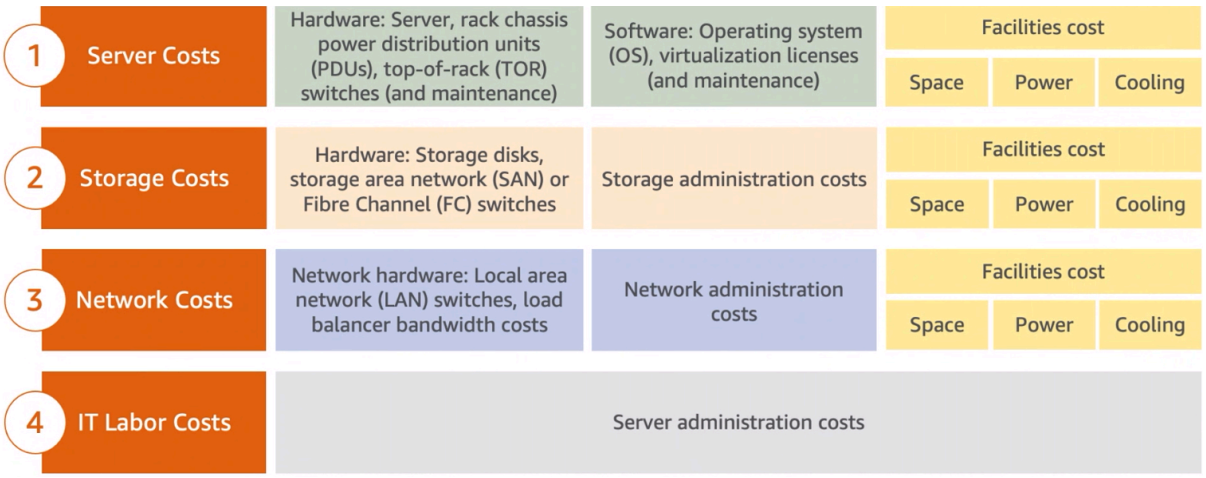
Compute

AWS charges per hour for compute resources, billing you from when you launch each resource until termination (for on-demand resources). There is also an option to make reservations with predetermined, set costs. AWS offers several compute instance types, such as the Elastic Compute Cloud (EC2) instances, which allow you to optimize your computing costs.

Storage

AWS usually charges for each GB of data storage. There are several storage classes for different use cases and data types, so choosing the right storage class for your needs is important. Cold storage is cheaper but less accessible, while hot storage is more readily available but more expensive, making it unsuitable for long-term storage.
The AWS cloud offers fixed and variable pricing models, providing flexibility and scalability at optimized costs. AWS offers several tools for calculating and managing costs, including resources to help you plan budgets, forecast spending, and track your usage and costs. These tools can provide recommendations for optimizing costs.
When calculating your TOC, you must consider the hidden, indirect costs that are difficult to predict, such as the cost of downtime, reduced productivity and other issues. The TOC assessment should include an evaluation of what-if scenarios, such as over-provisioning and changing requirements.

TCO Considerations

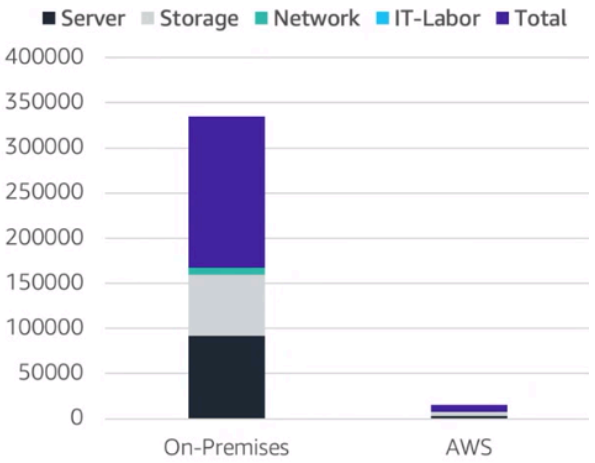


On-premises versus all-in-cloud

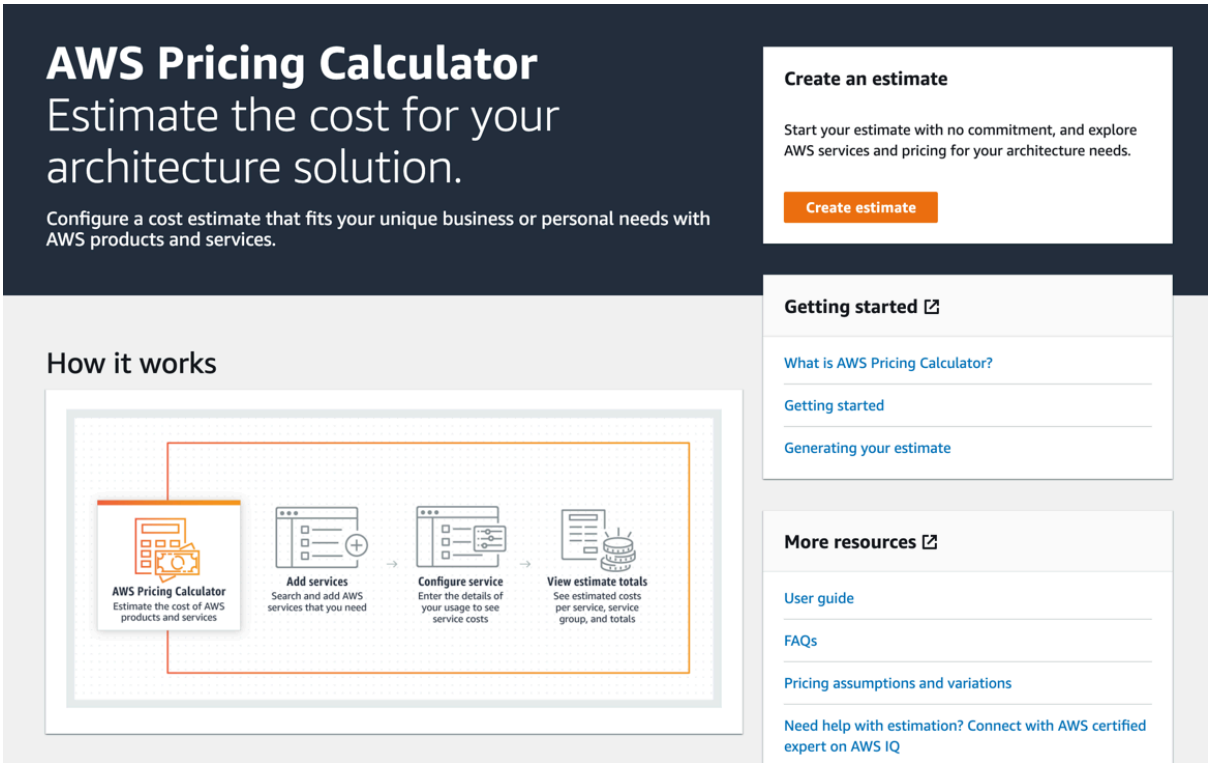
You cloud cave up to 96 percent a year by moving your infratstructure to AWS. Your 3-year total savings would be $159,913

3-Year Total Cost of Ownership		
	On-Premises	AWS
Server	$91,922	$2,547
Storage	$67,840	$4,963
Network	$7,660	$-------
IT – Labor	$ -------- --	$-------
Total	$167, 422	$7,509

AWS cost includes business-level support and a 3-year PURI EC2 instance



AWS Pricing Calculator



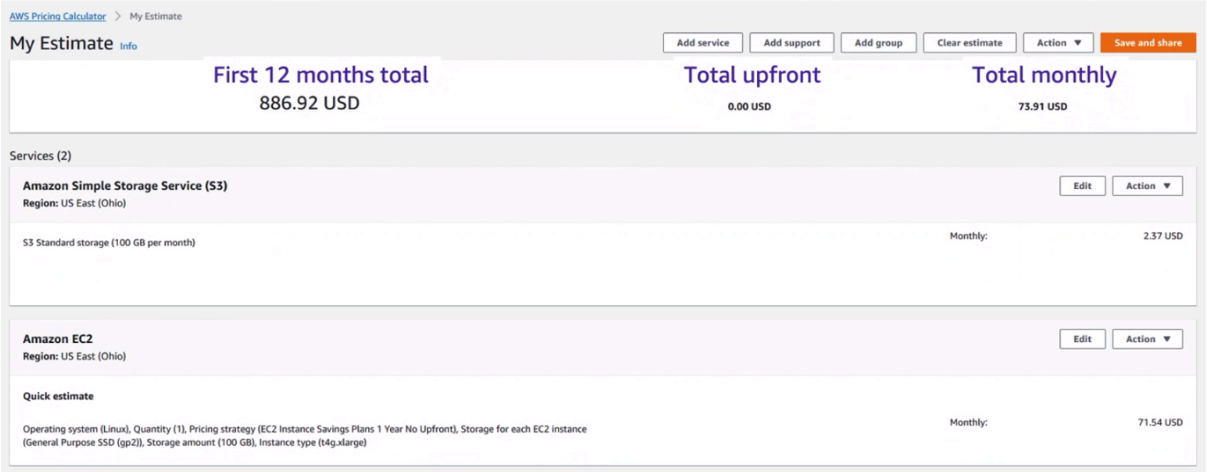
Use the AWS Pricing Calculator to:

- Estimate monthly costs
- Identify opportunities to reduce monthly costs
- Model your solutions before building them
- Explore price points and calculations behind your estimate
- Find the available instance types and contract terms that meet your needs
- Name your estimate and create name groups of services

Reading an estimate

Your estimate is broken into:

- first 12 months total
- total upfront
- total monthly



Additional benefit considerations

- Cloud Total Cost of Ownership: what will be spent to run the solution
- Return on Investment analysis (ROI): determine the value generated while considering savings

Case study: Delaware North

Background:

- Growing global company with over 200 locations
- 500 million customers: $3 billion USD annual revenue

Challenge:

- Meet demand to rapidly deploy new solutions
- Constantly upgrade aging equipment

Criteria:

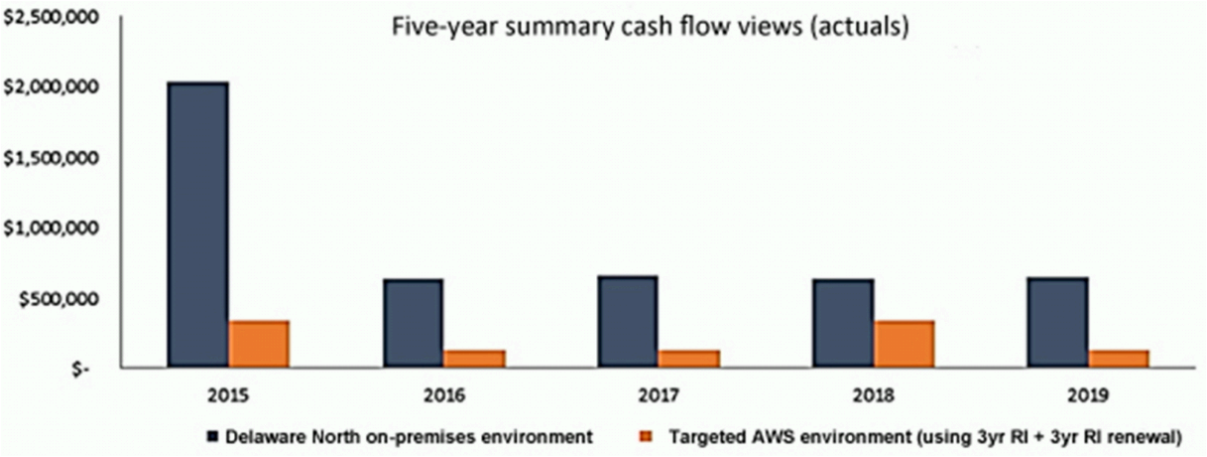
- Have a broad solution to handle all workloads
- Be able to modify processes to improve efficiency and lower costs
- Eliminate busy work (such as patching software)
- Achieve a positive return on investment (ROI)

Solution:

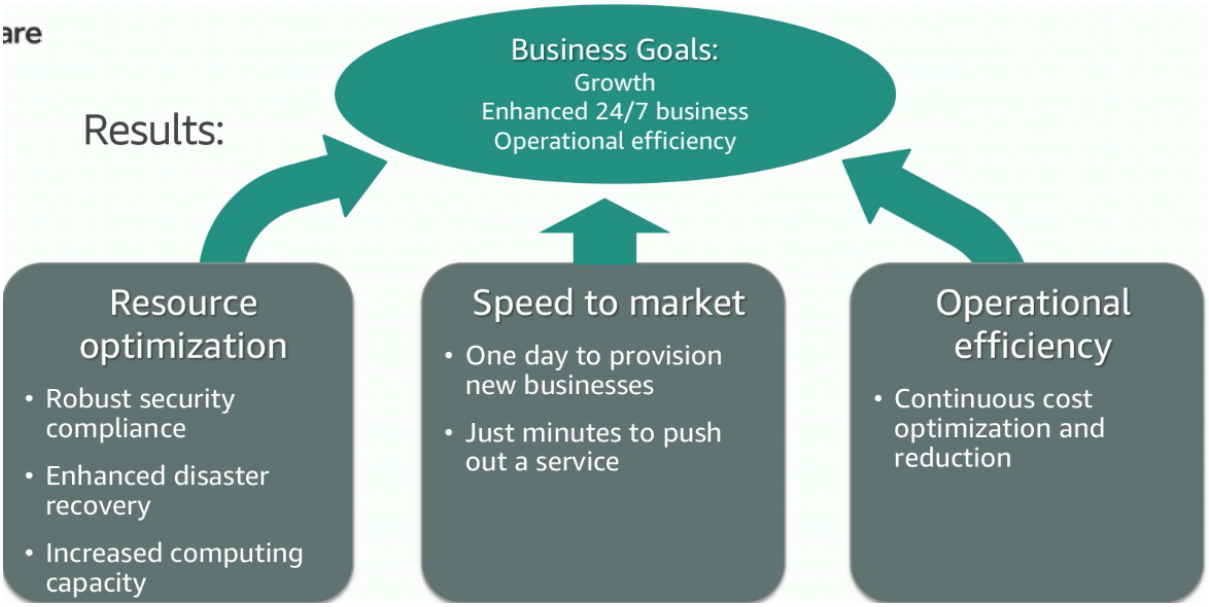
- Move their on-premises data center to AWS
    - Eliminated 205 servers (90%)
    - Moved nearly all aplications to AWS

- Used 3-year Amazon ECE2 Reserved Instances

Cost comparison



Results



Chapter 8

Cloud Economics - AWS Infrastructure

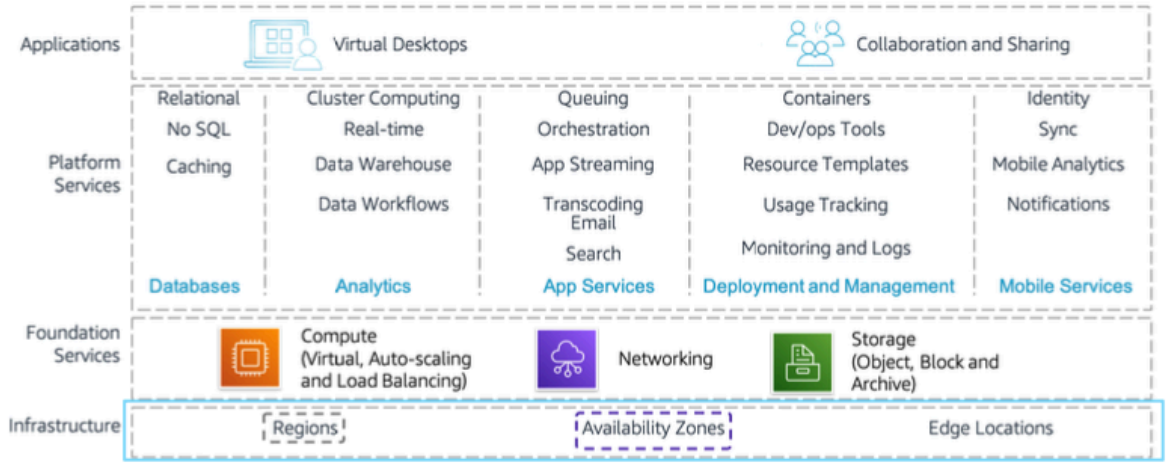
AWS Global Cloud Infrastructure

AWS provides the most extensive global footprint compared to any other cloud providers in the market, and it opens up new regions faster than others/. AWS maintains

numerous geographic regions around the globe, from North America, South America, Europe, Asia Pacific, and the Middle East. AWS serves over a million active customers in more than 190 countries.

AWS is able to support this massive workload thanks to its Global Cloud Infrastructure, which consists of the following:

- Availability Zones
- Regions
- Edge Networks



The AWS Global Cloud Infrastructure is the most secure, extensive, and reliable cloud platform in the industry today, which offers a wide range of cloud service offerings. AWS is the top choice of small and medium enterprises for deploying their application workloads across the globe and for distributing content closer to their end-users with low latency. It provides you with a highly available and fault-tolerant cloud infrastructure where and when you need it.

AWS owns and operates thousands of servers and networking devices that are running in various data centers scattered around the globe. A data center is a physical facility that houses hundreds of computer systems, network devices, and storage appliances. You can run your applications in two or more data centers to achieve high availability, so if there is an outage in one of the data centers, you still have other servers running in another data center. A data center can also deliver cached content to your global end-users to improve response times.

AWS data centers

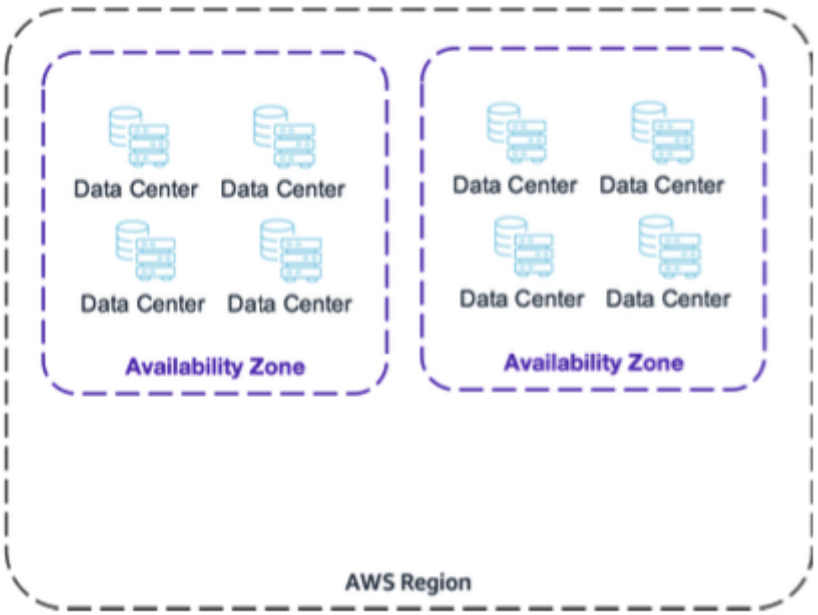
- AWS data centers are designed for security
- Data centers are where the data resides and data processing occurs
- Each data has redundant power, networking and connectivity, and is housed in a separate facility
- A data center typically has 50,000 to 80,000 physical servers

AWS uses custom networking equipment sourced from multiple ODMs.

ODM: Original Device Manufacturers Design and manufacture products based on specifications from a second company.
The second company rebrands the products for sale.

AWS Regions

AWS Regions are separate geographic areas that AWS uses to house its infrastructure. These are distributed around the world so that customers can choose a region closest to them in order to host their cloud infrastructure there. The closer your region is to you, the better, so that you can reduce network latency as much as possible for your end-users. You want to be near the data centers for fast service.



- An AWS Region is a geographical area.
- Each Region is made up of two or more Availability Zones.

- AWS has 32 Regions worldwide. (November 2023)
- You enable and control data replication across Regions.
- Communication between Regions uses AWS backbone network connections infrastructure.

What AWS Regions have the most services?

Not all regions are created equally. These regions have more services than others in their general areas:

- Americas: US East (N. Virginia), US West (N. California)
- Asia Pacific: Singapore, Sydney, Tokyo
- EU: Frankfurt, Ireland

Best practices for choosing AWS Regions

In general, try to follow these best practices when you choose a region, to help ensure top performance and resilience:

Proximity: Choose a region closest to your location and your customers' location to optimize network latency.

Services: Try and think about what are your most needed services. Usually, the newest services start on a few main regions then pop up in other regions later.

Cost: Certain regions will cost more than others, so use built-in AWS calculators to do rough cost estimates to inform your choices.
Service Level Agreement (SLA): Just as with cost, your SLA details will vary by region, so be sure to be aware of what your needs are and if they're being met.
Compliance: You may need to meet regulatory compliance needs such as GDPR by hosting your deployment in a specific — or multiple regions.

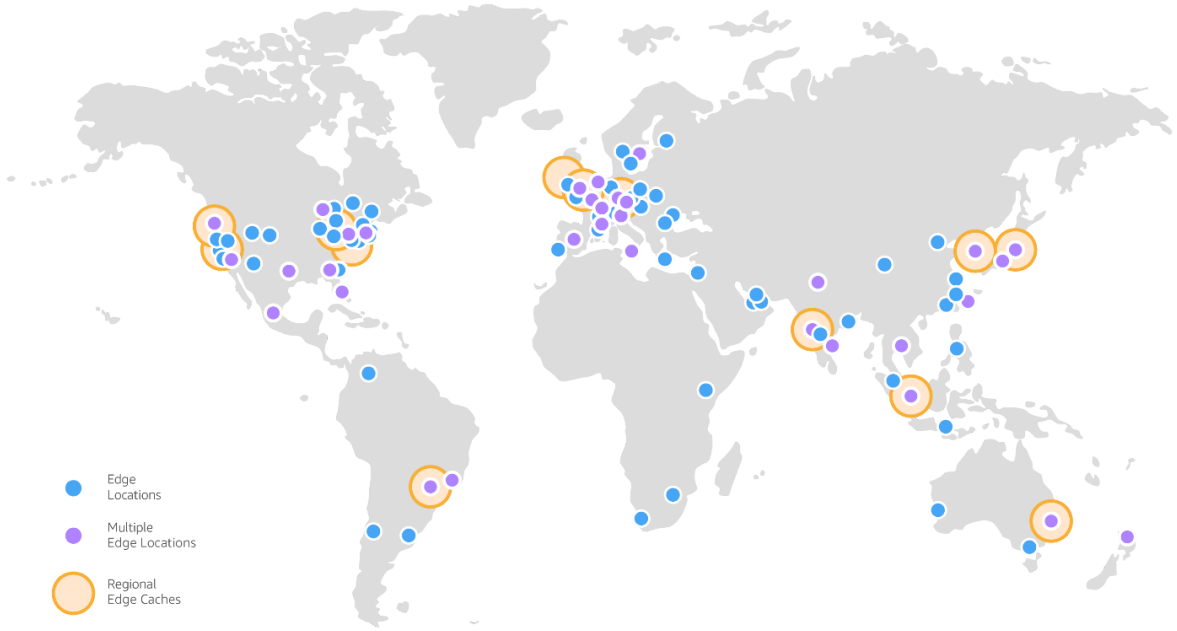
AWS Availability Zones?

An AWS Availability Zone (AZ) is the logical building block that makes up an AWS Region. There are currently 102 AZs, which are isolated locations— data centers — within a region. Each region has multiple AZs and when you design your infrastructure to have backups of data in other AZs you are building a very efficient model of resiliency, i.e. a core concept of cloud computing.

- Each Availability Zone is:
  - Made up of one or more data centers.
  - Designed for fault isolation.
  - Interconnected with other Availability Zones using high-speed private links.
- You choose your Availability Zones.
- AWS recommends replicating across Availability Zones for resiliency.

Points of Presence

In addition to the AWS Regions and Availability Zones, AWS also operates a globally distributed point of presence (PoP) network. These PoPs host Amazon CloudFront, a content delivery network (CDN); Amazon Route 53, a public Domain Name System (DNS) resolution service; and AWS Global Accelerator (AGA), an edge networking optimization service. The global edge network currently consists of over 410 PoPs, including more than 400 Edge Locations, and 13 regional mid-tier caches in over 90 cities across 48 countries (current status can be found here: Amazon CloudFront Key Features).

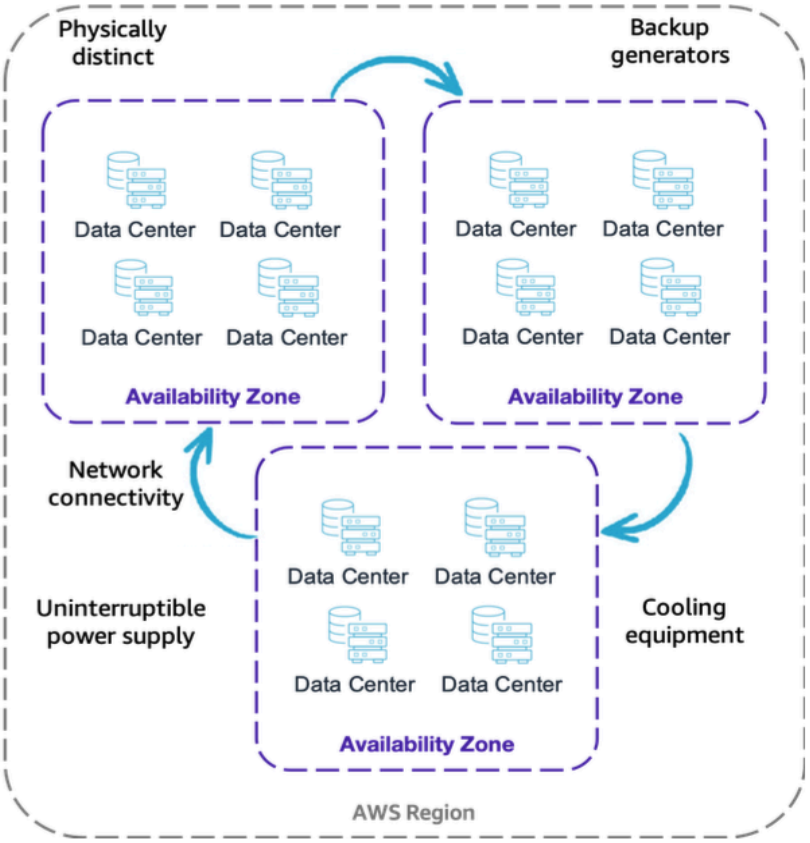


Amazon CloudFront global edge network

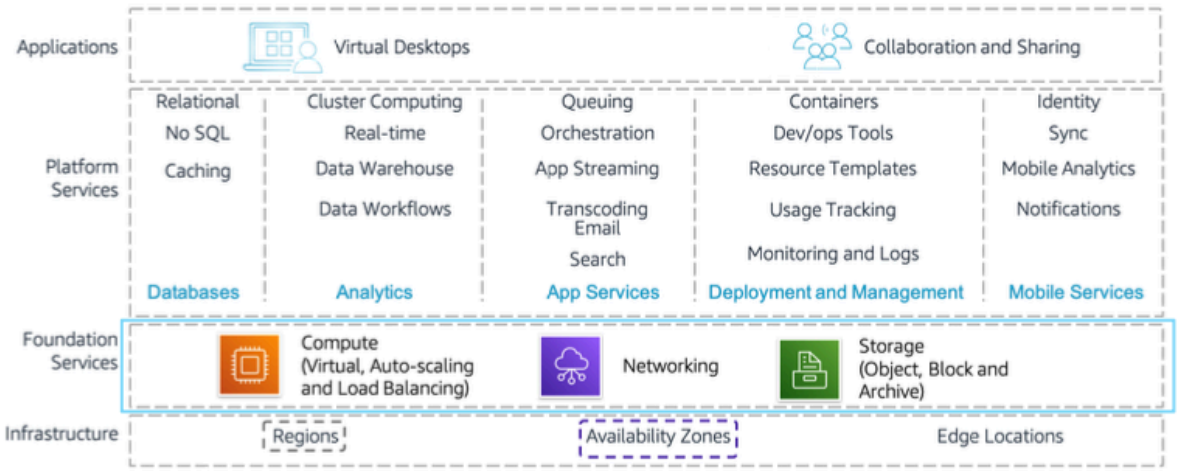
Each PoP is isolated from the others, which means a failure affecting a single PoP or metropolitan area does not impact the rest of the global network. The AWS network peers with thousands of Tier 1/2/3 telecom carriers globally, is well connected with all major access networks for optimal performance, and has hundreds of terabits of deployed capacity. Edge locations are connected to the AWS Regions through the AWS network backbone, a fully redundant, multiple 100GbE parallel fiber that circles the globe and links with tens of thousands of networks for improved origin fetches and dynamic content acceleration.

AWS infrastructure features

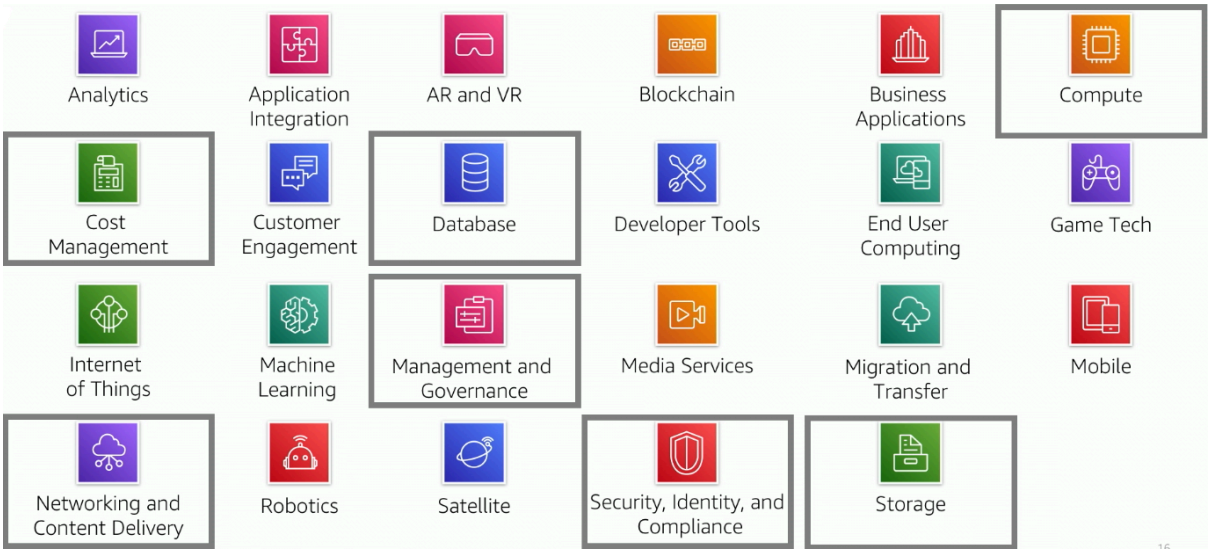
- Elasticity and scalability
  - Elastic infrastructure, dynamic adaption of capacity
  - Scalable infrastructure, adapts to accommodate growth
- Fault-tolerance
  - Continues operating properly in the presence of a failure
  - Built-in redundancy of components
- High availability
  - High level of operational performance
  - Minimize downtime
  - No human intervention



AWS Foundational Services



AWS categories of services



Storage service category

- Amazon Simple Storage Service (Amazon S3)
  - Object storage
  - Scalability, data availbility and performance
- Amazon Elastic Block Store (Amazon EBS)
  - high performance block storage
  - Used with Amazon EC2
- Amazon Elastic File System (Amazon EFS)
  - Scalable file system (NFS)
  - Use with AWS Cloud Services
- Amazon Simple Storage Service Glacier
  - Extremely low-cost
  - Data archiving

Compute service category

- Amazon EC2
  - Resizable compute capacity
- Amazon EC2 Auto Scaling
  - Automatically add or remove EC2 instances
- Amazon Elastic Container Service
  - Supports docker container
- Amazon EC2 Container Registry (ECR)
  - Fully managed docker container registry
- AWS Elastic Beanstalk
  - Deploying and scaling web applications
- AWS Lambda

    - Run code without servers
    - No charge when the code is not running
- Amazon Elastic Kubernetes Service (Amazon EKS)
    - Deploy, manage and scale applications using Kubernetes
- AWS Fargate
    - Run container without having to manage servers

Database service category

- Amazon Relational Database Service (RDS)
    - Relational database in the cloud
    - Scalable
    - Automating database setup, patching, back-ups
- Amazon Aurora
    - MySQL and PostgreSQL
    - 5 time faster than MySQL
    - 3 times faster than PostgreSQL
- Amazon Redshift
    - Analytic queries against petabytes of data
    - Fast
- Amazon DynamoDB
    - NoSQL database
    - Single digit performance

Networking and content delivery service category

- Amazon VPC
    - Isolated sections AWS Cloud
- Elastic Load Balancing
    - Automatically distributes incoming application traffic
- Amazon CloudFront
    - Delivery network (CDN)
    - Secures data to customers
- AWS Transit Gateway
    - Connect Amazon VPC and on-premises network
- Amazon Rout 53
    - Scalable cloud domain name system
    - Translate URL to IP addresses
- AWS Direct Connect
    - Established dedicated private network
- AWS VPN
    - Secure private tunnel to AWS global network

Security, identity and compliance service category

- AWS Identity and Access Management (IAM)
  - Enables you to manage access
- AWS Organizations
  - Restricts actions and services allowed in your account
- Amazon Cognito
  - Let you add user authentication and access control to web and mobile apps
- AWS Artifact
  - On-demand access to AW security and compliance reports
- AWS Key Management Service (KMS)
  - Create and manage encryption keys
- AWS Shield
  - Managed distributed denial of service protection service

AWS cost management category

- AWS Cost and Usage Report
  - Set AWS cost and usage data
- AWS Budget
  - Set custom budget
- AWS Cost Explorer
  - Visualize and manage AWS cost and usage

Management and governance service category

- AWS Management Console
  - Web-based user interface for accessing your AWS account
- AWS Config
  - Track resource inventory
- Amazon CloudWatch
  - Monitor resources and app
- AWS Auto Scaling
  - Scale multiple resources to meet demand
- AWS Command Line Interface (CLI)
  - Unified tool to manage AWS services
- AWS Trusted Advisor
  - Optimize performance and security
- AWS Well-Architected Tool
  - Reviewing and improving workloads
- AWS CloudTrail
  - Track user activity an API usage

Chapter 9

Core Services - Compute - EC2

Compute services overview

AWS compute services

- Amazon EC2:
  - resizable virtual machine
- Amazon EC2 auto-scaling:
  - define conditions to launch or terminate EC2 instances
- Amazon ECR:
  - store and retrieve Docker images
- Amazon ECS:
  - Container orchestration service that supports Docker
- VMWare Cloud on AWS:
  - hybrid cloud without custom hardware
- AWS Elastic Beanstalk:
  - run and manage web app
- AWS Lambda:
  - serverless compute solution
- Amazon EKS:
  - run managed kubernetes on AWS
- Amazon LightSail:
  - building app or website
- AWS Batch:
  - running batch job at any scale
- AWS Fargate:
  - run containers
- AWS Outpost:
  - run AWS services in your on-premises data center
- AWS Serverless Repository:
  - discover, deploy and publish application

Categorizing compute services

Services	Key Concepts	Characteristics	Ease of Use
• Amazon EC2	• Infrastructure as a service (IaaS) • Instance-based • Virtual machines	• Provision virtual machines that you can manage as you choose	A familiar concept to many IT professionals.
• AWS Lambda	• Serverless computing • Function-based • Low-cost	• Write and deploy code that executes on a schedule or that can be triggered by events • Use when possible (architect for the cloud)	A relatively new concept for many IT staff members, but easy to use after you learn how.
• Amazon ECS • Amazon EKS • AWS Fargate • Amazon ECR	• Container-based computing • Instance-based	• Spin up and execute jobs more quickly	AWS Fargate reduces administrative overhead, but you can use options that give you more control.
• AWS Elastic Beanstalk	• Platform as a service (PaaS) • For web applications	• Focus on your code (building your application) • Can easily tie into other services—databases, Domain Name System (DNS), etc.	Fast and easy to get started.

Choosing the optimal compute service

- The optimal compute service or services that you use will depend on your use case
- Some aspects to consider
    - What is your application design ?
    - What are your usage patterns ?
    - Which configuration settings wll you want to manage ?
- Selecting the wrong compute solution for an architecture can lead to lower performance efficiency
    - A good starting place: understand the available compute options

Amazon Elastic Compute Cloud (Amazon EC2)

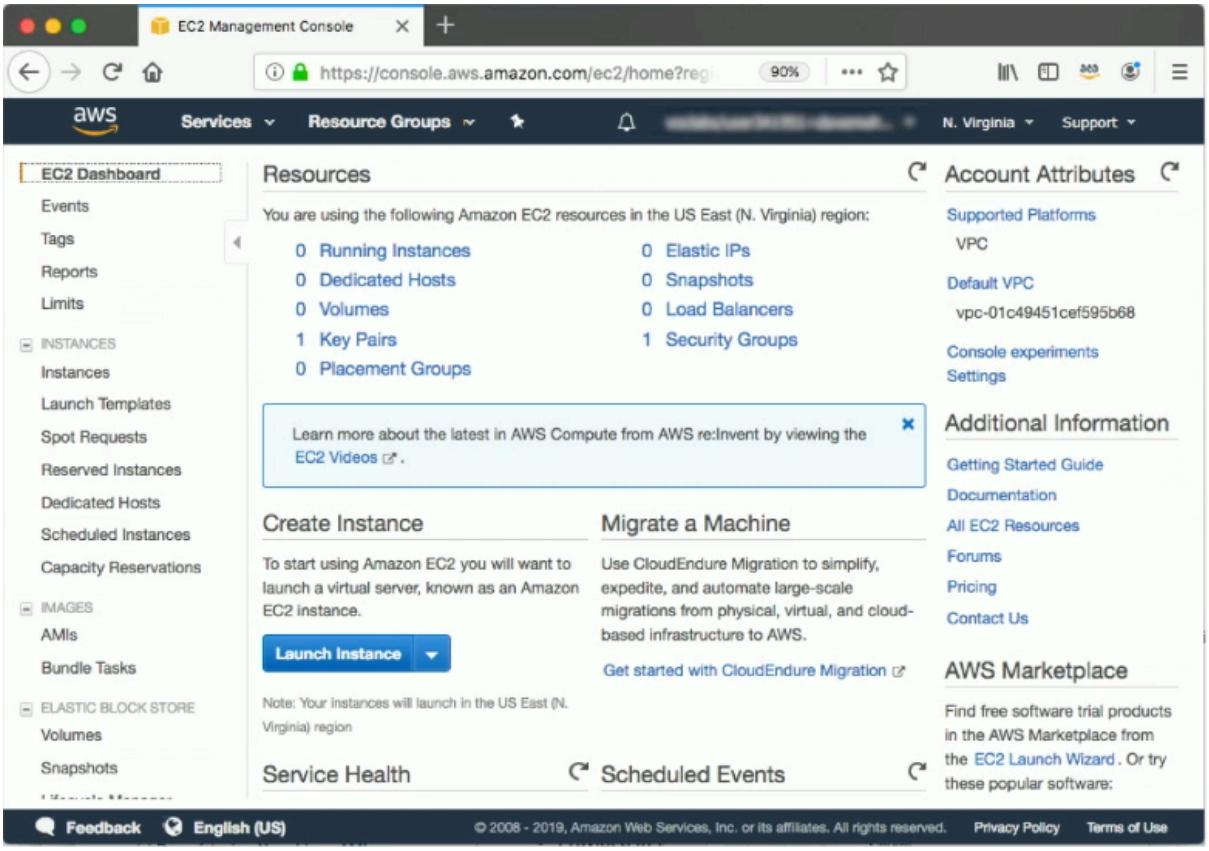
Example uses of Amazon EC2 instances:

- App server
- web server
- Database server
- Game server
- Mail server
- Media server
- Catalog server
- File server
- Computing server
- Proxy server

Amazon EC2 overview

- Amazon Elastic Compute Cloud (Amazon EC2)
    - Provides virtual machines (EC2 instance) in the cloud
    - Fives you full control over the guest operating system (Windows or Linux) on each instance
- You can launch instances of any size into and Availability Zone anywhere in the world
    - Launch instance from Amazon Machine Images (AMIs)
    - Launch instances with a few clicks or a line of code, and they are ready in minutes
- You can control traffic to and from instances

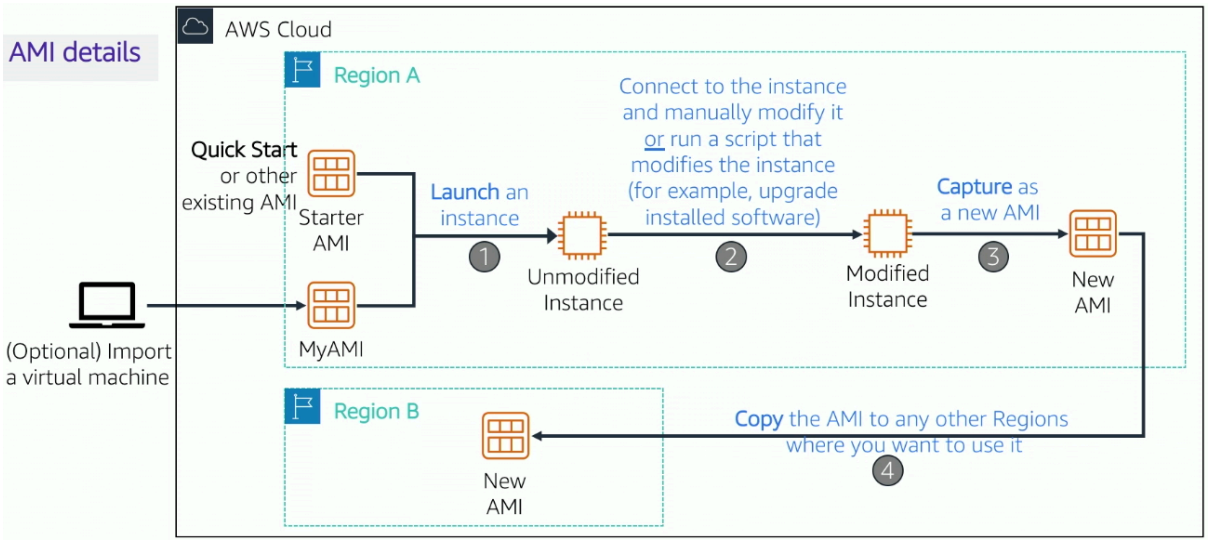
Launching an amazon EC2 instance



Nine key decisions when creating a EC2 instance.

1. Select an AMI

- Amazon Machine Image (AMI)
    - Is a template that is used to create an EC2 instance
    - Contains a Windows or Linux OS
    - Often has some software pre-installed
- AMI choices:
    - Quick Start
        - Linux and Windows AMIs provided by AWS
    - My AMIs
        - Any AMIs that you created
    - AWS Marketplace
        - Pre-configured templates from third parties
    - Community AMIs
        - AMIs shared by others; use at you own risk



2. Select an instance type

- Consider you use case
  - How will the EC2 instance you create be used ?
- The instance type that you choose determines
  - Memory (RAM)
  - Processing power (CPU)
  - Disk space and disk type (Storage)
  - Network performance
- Instance type categories
  - General purpose
  - Compute optimized
  - Memory optimized
  - Storage optimized
  - Accelerated computed
- Instance types offer family, generation and size

Instance type naming and sizes

Instance type details

Instance type naming

- Example: t3.large
  - T is the family name
  - 3 is the generation number
  - Large is the size

Example instance sizes

Instance Name	vCPU	Memory (GB)	Storage
t3.nano	2	0.5	EBS-Only
t3.micro	2	1	EBS-Only
t3.small	2	2	EBS-Only
t3.medium	2	4	EBS-Only
t3.large	2	8	EBS-Only
t3.xlarge	4	16	EBS-Only
t3.2xlarge	8	32	EBS-Only

Based on use case

Instance type details

	General Purpose	Compute Optimized	Memory Optimized	Accelerated Computing	Storage Optimized
Instance Types	a1, m4, m5, t2, t3	c4, c5	r4, r5, x1, z1	f1, g3, g4, p2, p3	d2, h1, i3
Use Case	Broad	High performance	In-memory databases	Machine learning	Distributed file systems

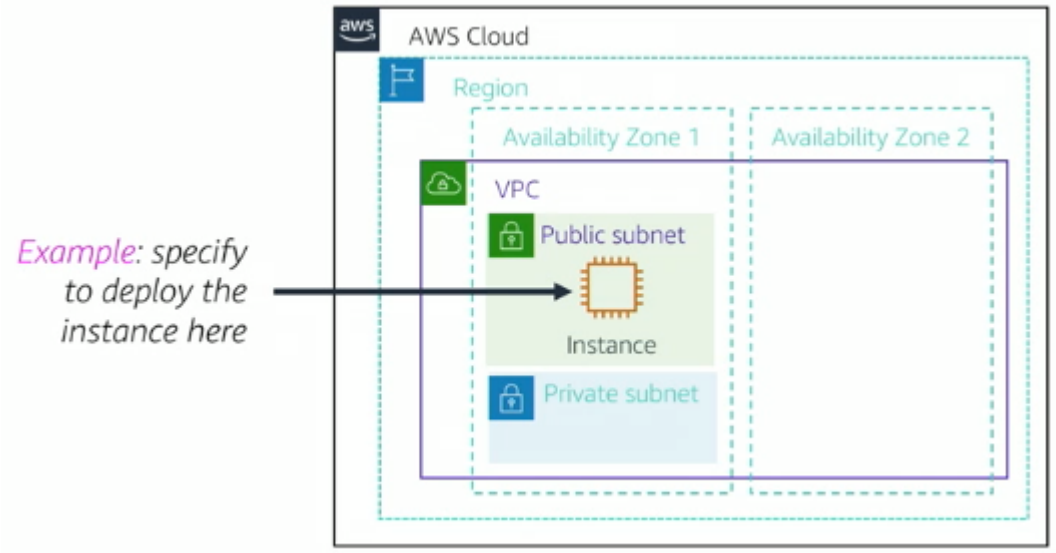
Networking features

- The network bandwidth (GBps) varies by instance type
- To maximize networking and bandwidth performance of your instance type
  - If you have interdependent instances, launch them into a cluster placement group
  - Enable enhanced networking
- Enhanced networking types are supported on most instance types
- Enhanced networking types
  - Elastic Network Adapter (ENA): Supports network speeds of up to 100 Gpbds

- Intel 82599 Virtual Function interface: Supports network speeds of up to 10 Gbps

3. Specify network settings

- Where should the instance be deployed ?
  - Identify the VPC and optinally the subnet
- Should a public IP address be automatically assigned ?
  - To make it internet-accessible

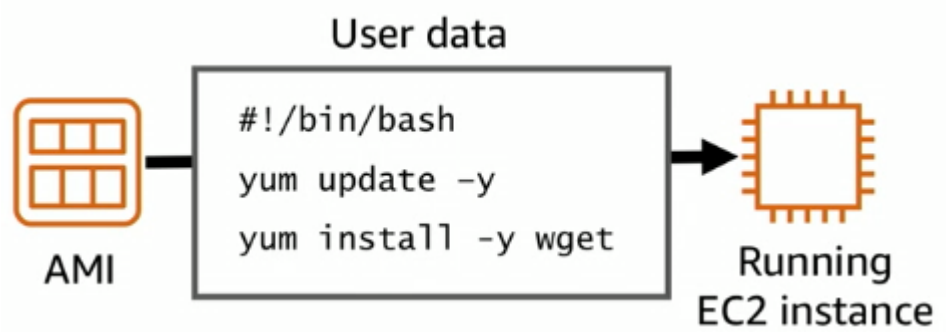


4. Attach IAM role (optional)

- Will software on the EC2 insrance need to interact with other AWS services ?
  - If yes, attach an appropriate IAM Role
- An AWS Identity and Access Management (IAM) role that is attache to an EC2 instance is kept in an instance profile
- You are not restricted to attaching a role only at instance launch
  - You can also attach a role to an instance that already exists

5. User data script (optional)

- Optionally specify a user data script at instance launch
- Use user data scripts to customize the runtime environment of your instance
  - Script executes the first time the instance starts
- Can be used strategically
  - Reduce the number of custom AMIs that you build and maintain



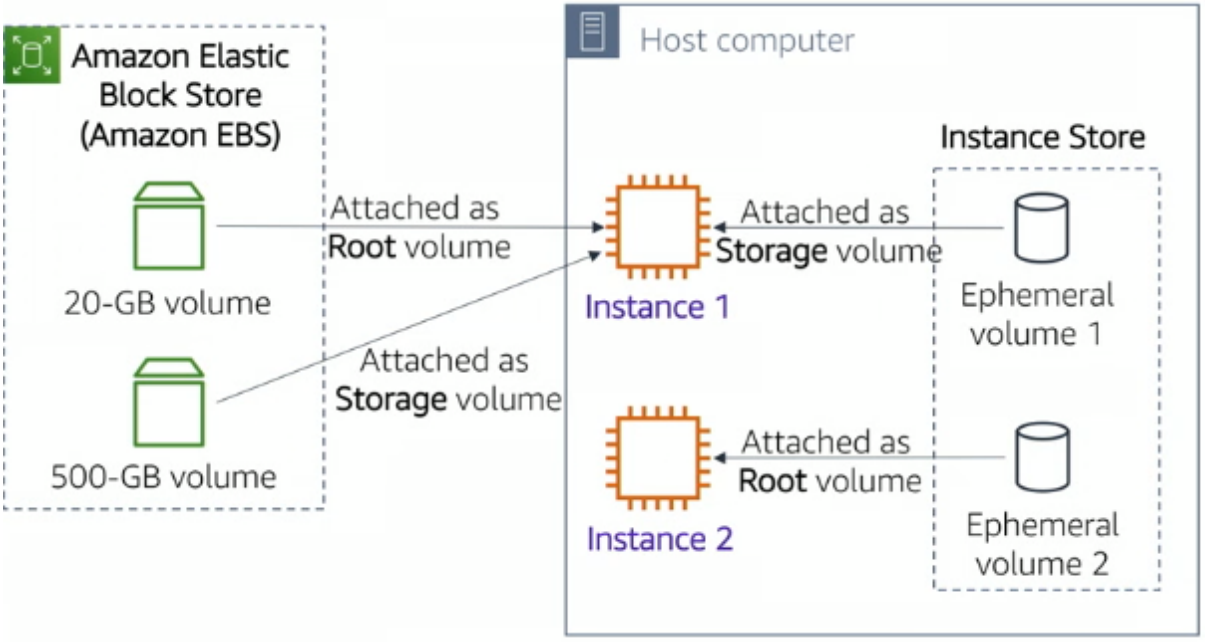
6. Specify storage

- Configure the root volume
  - Where the guest operating system is installed
- Attach additional storage volumes (optional)
  - AMI might already include more than one volume
- For each volume, specify:
  - The size of the disk (in GB)
  - The volume type
    - Different types of SSDs and HDDs are available
  - If the volume will be deleted when the instance is terminated
  - If encryption should be used

Amazon EC2 storage options

- Amazon Elastic Block Store (Amazon EBS)
  - Durable, block-level storage volumes
  - You can stop the instance and start it again, and the data will still be there
- Amazon Elastic Block Store
  - Storage is provided on disls that are attached to the host computer where the EC2 instance is running
  - If the instance stops, data stored here is deleted
- Other options for storage (not for root volume)
  - Mount an Amazon Elastic File System (Amazon EFS) file system
  - Connect to Amazon Simple Storage Service (Amazon S3)

Example storage options



- Instance 1 characteristics
  - It has an Amazon EBS root volume type for the operating system
  - What will happen if the instance is stopped and then started again ?
    - The OS volume would survive
    - Any data stored on Amazon EBS would remain intact
    - Any data stored in ephemeral volume 1 would be lost
- Instance 2 characteristics
  - It has an Instance Store root volume type for the operating system
  - What will happen if the instance stops (because of user error or a system malfunction)?
    - All data stored in ephemeral volume 2 would be lost, including the OS

7. Add tags

A tag is a label that you can assign to an AWS resource

- Consists of a key and an optional value
- Tagging is how you can attach metadata to an EC2 instance
- Potential benefits from tagging - Filtering, automation, cost allocation and access control

8. Security group settings

A security group is a set of firewall rules that control traffic to the instance.

- It exists outside of the instance's guest OS

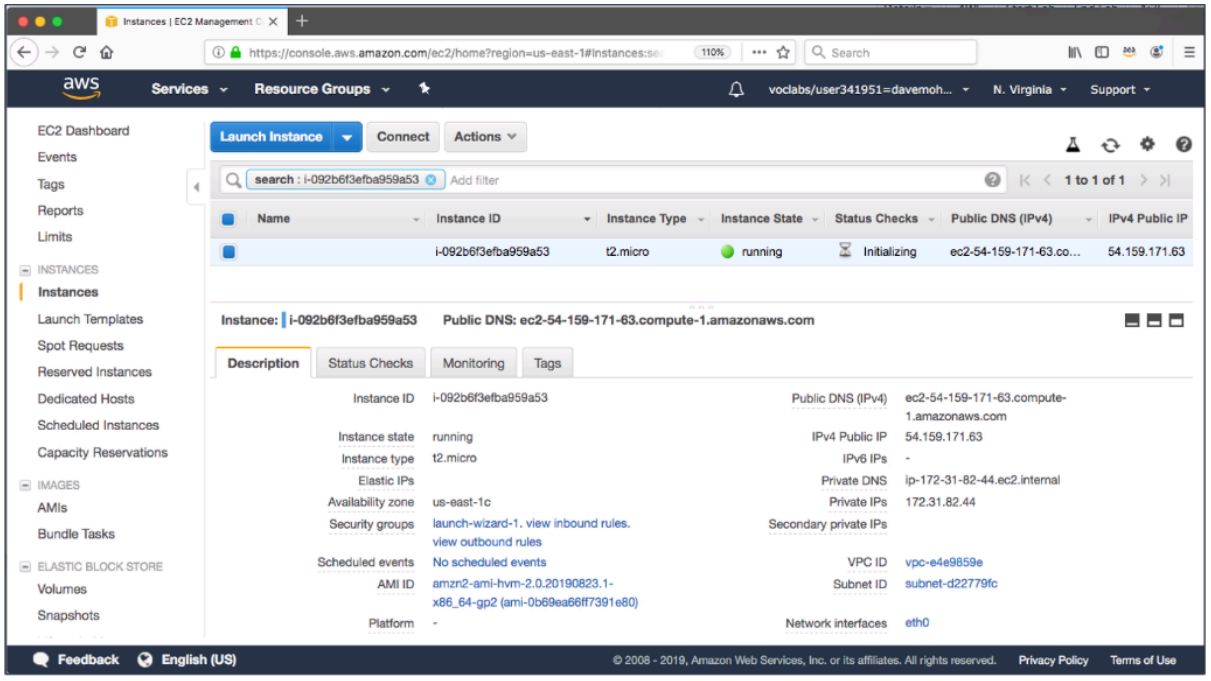
Create rules that specify the source and which ports that network communications can use.

- Specify the port number and the protocol, such as TCP, UDP or ICMP
- Specify the source that is allowed to use the rule

9. Identify the key pair

- At instance launch, you specify an existing key pair or create a new key pair
- A key pair consists of
    - A public key that AWS stores
    - A private key file that you store
- It enables secure connections to the instance
- For Windows AMIs
    - Use the private key to obtain the administrator password that you need to log in to your instance
- For Linux AMIs
    - Use the private key to use SSH to securely connect to your instance

Amazon EC2 console view of a running EC2 instance



Another option: Launch an EC2 instance with the AWS CLI

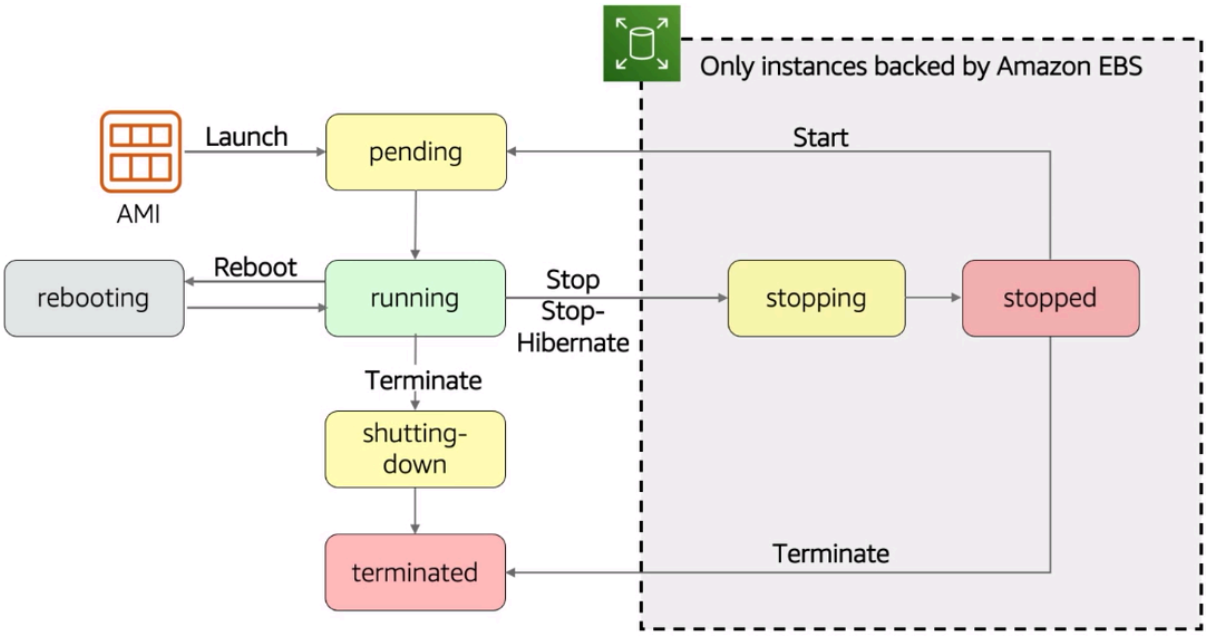
- EC2 instances can also be created programmatically

```
aws ec2 run-instances --image0id ami-1a2b3c4d --count 1
--instance-type c3.large \
--key-name MyKeyPair --security-groups MySecurityGroup
--region us-east-1
```

This example shows how simple the command can be.

- This command assumes that the key pair and security group already exists
- More option could be specified

Amazon EC2 instance lifecycle



Consider using an Elastic IP address

- Rebooting an instance will not change any IP addresses or DNS hostnames
- When an instance will not change any IP addresses or DNS hostnames
- When an instance is stopped and then started again
  - The public IPv4 address and external DNS hostname will change
  - The private IPv4 address and internal DNS hostname do not change
- If you require a persistent public IP address
  - Associate an Elastic IP address with the instance
- Elastic IP address characteristics
  - Can be associated with instances in the Region as needed

    - Remains allocated to your account until you choose to release it

EC2 instance metadata

It is data about your instance

- While you are connected to the instance, you can view it
    - In a browser: http://169.254.169.254/latest/meta-data/
    - In a terminal window: curl http://169.254.169.254/latest/meta-data/
- Example retrievable values
    - Public IP address, private IP address, public hostname, instance ID, security groups, Region, Availability zone
    - Any user data specified at instance launch can also be accessed at: http://169.254.169.254/latest/user-data/
- It can be used to configure or manage a running instance
    - For example, author a configuration script that read the metadata and uses to configure applications or OS settings

Amazon CloudWatch for monitoring

- Use Amazon CloudWatch to monitor EC2 instances
    - Provides near-real-time metrics
    - Provides charts in the Amazon EC2 console Monitoring tab
    - Maintains 15 months of historical data
- Basic monitoring
    - Default, no additional cost
    - Metric data sent to CloudWatch every 5 minutes
- Detailed monitoring
    - Fixed monthly rate for seven pre-selected metrics
    - Metric data delivered every 1 min

Amazon EC2 Cost Optimization

Amazon EC2 pricing models

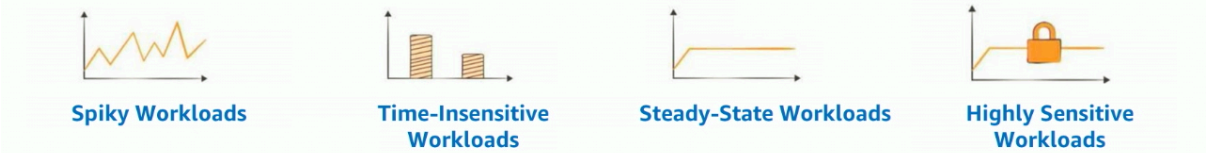
- On-Demand Instances
    - Pay by the hour
    - No long-term commitments
    - Eligible for the AWS Free Tier
- Dedicated Hosts
    - A physical server with EC2 instance capacity fully dedicated to your use
- Dedicated instances
    - Instances that run in a VPC on a hardware that is dedicated to a single customer

- Reserved Instances
    - Full, partial, or no upfront payment for instance you reserve
    - Discount on hourly charge for that instance
    - 1-year or 3-year term
- Scheduled Reserved Instances
    - Purchase a capacity reservation that is always available on a recurring schedule you specify
    - 1-year term
- Spot Instances
    - Instances run as long as they are available and your bid is above the Spot Instance price
    - They can be interrupted by AWS with a 2-minute notification
    - Interruption options include terminated, stopped or hibernated
    - Prices can be significantly less expensive compared to On-Demand Instances
    - Good choice when you have flexibility in when your applications can run

Benefits

On-demand Instances	Spot Instances	Reserved Instances	Dedicated Instances
Low cost and flexibility	Large scale, dynamic workload	Predictability ensures compute capacity is available when needed	Save money on licensing costs </br> Help meet compliance and regulatory requirements

Use cases



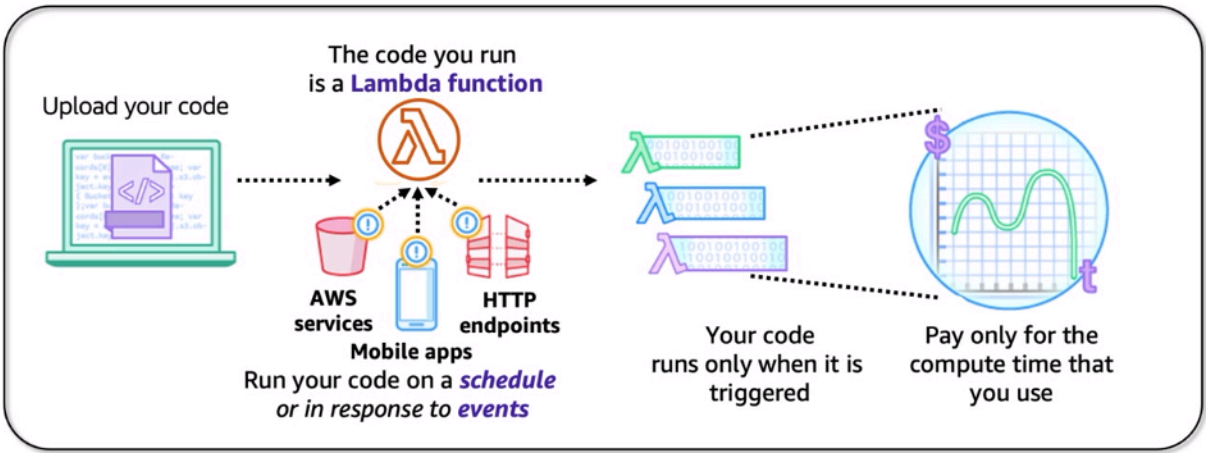
On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
• Short-term, spiky, or unpredictable workloads • Application development or testing	• Applications with flexible start and end times • Applications only feasible at very low compute prices • Users with urgent computing needs for large amounts of additional capacity	• Steady state or predictable usage workloads • Applications that require reserved capacity, including disaster recovery • Users able to make upfront payments to reduce total computing costs even further	• Bring your own license (BYOL) • Compliance and regulatory restrictions • Usage and licensing tracking • Control instance placement

Chapter 10

Core Services - Compute - Lambda and Beanstalk

AWS Lambda: Run code without servers

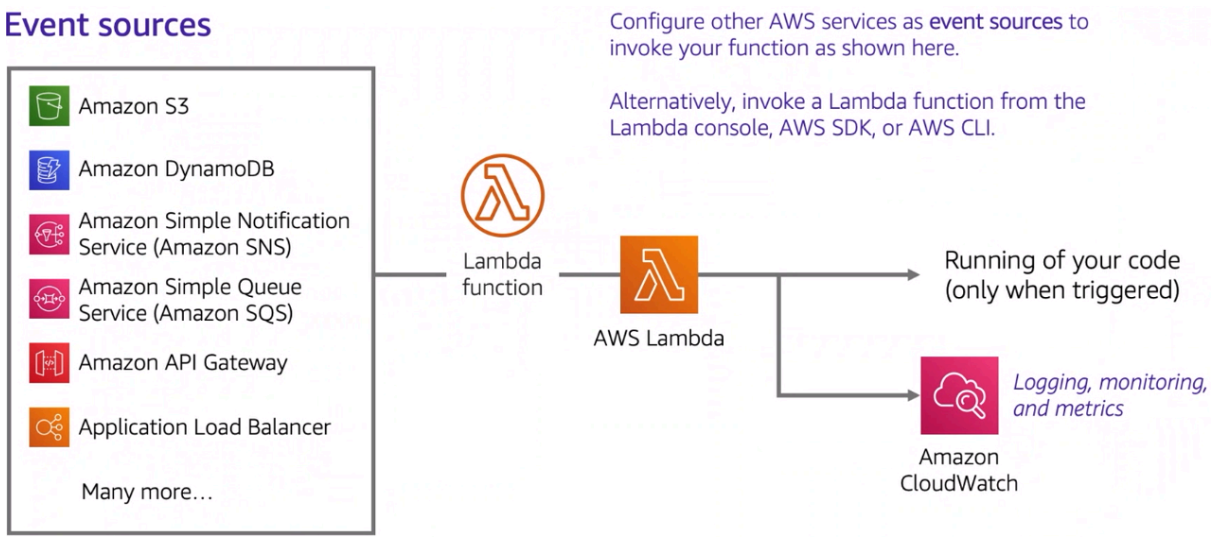
AWS Lambda is a serverless compute service.



Benefits of Lambda

- Supports multiple programming languages
- Completely automated administration
- Built-in fault tolerance
- Supports orchestration of multiple functions
- Pay-per-use pricing

AWS Lambda event sources

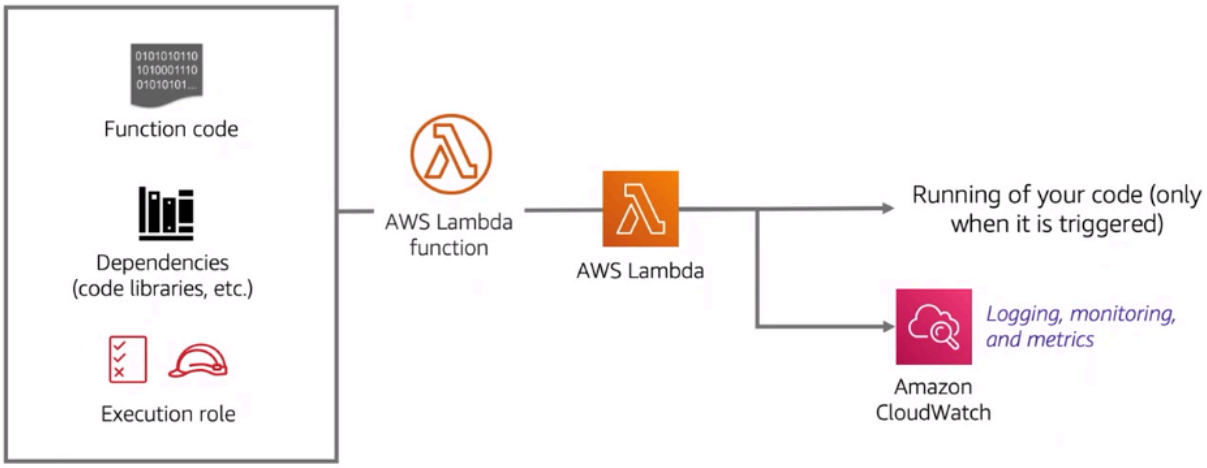


AWS Lambda function configuration

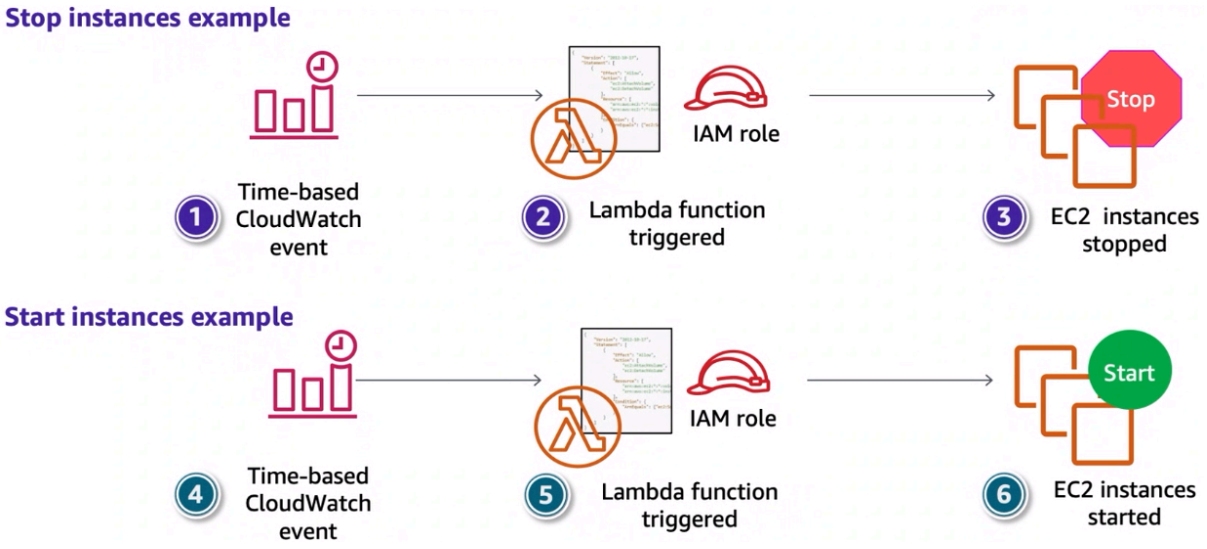
- Create lambda function: give a name
- Runtime environment
    - Python
    - Node.js

- Execution role to grant IAM permission to the function to interact with other services
- Configure the function
    - adding a trigger

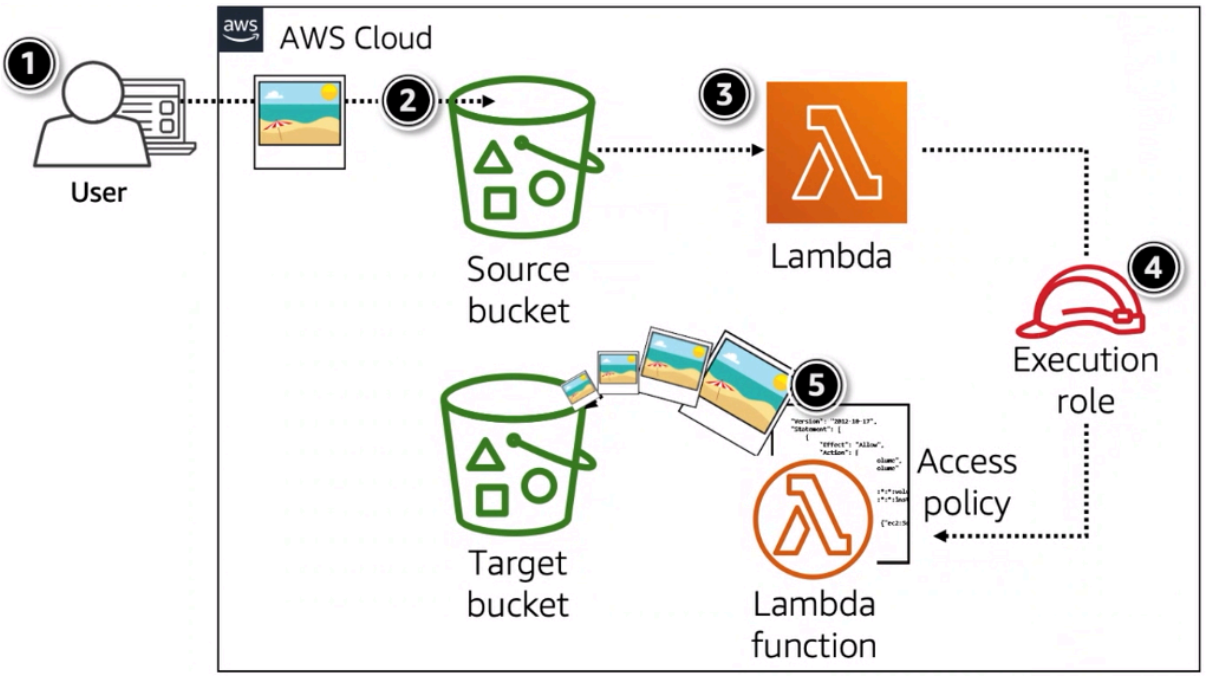
- Add function code
- Specify the memory in megabytes (up to 3008MGB)
- Specify env variable



Schedule-based Lambda function example: start and stop EC2 instances



Envent-based Lambda function example: create thumbnail images



AWS Lambda limits

Soft limits per Region

- Concurrent executions = 1,000
- Function and layer storage = 75GB

Hard limits for individual function:

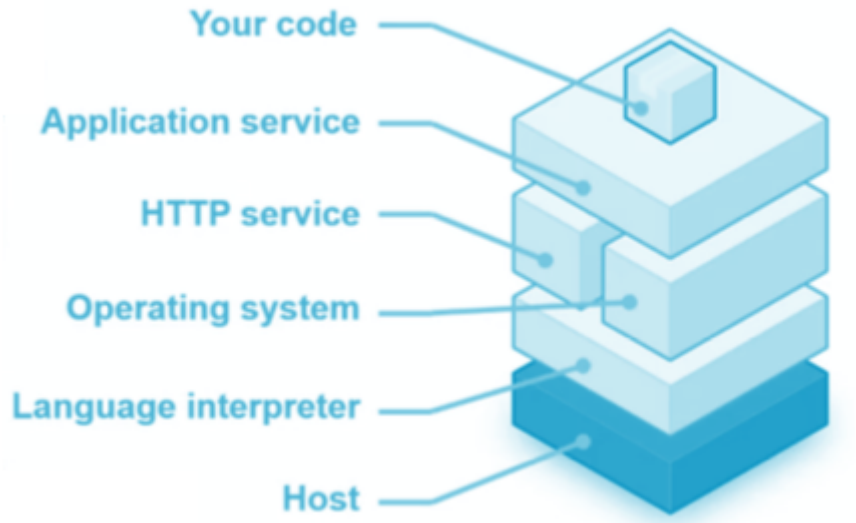
- Max function memory alloc = 3,008 MB
- Function tiemout = 15 min
- Deployement package size = 250 MB unzipped, including layers

AWS Elastic Beanstalk

- An easy way to get web app up and running
- A managed service that automatically handles
  - Infra provisionning and config
  - Deployement
  - Load balancing
  - Automatic scaling
  - Health monitoring
  - Analysis and debugging

    - Logging
- No additional charge for Elastic Beanstalk
    - Pay only for the underlying resources that are used

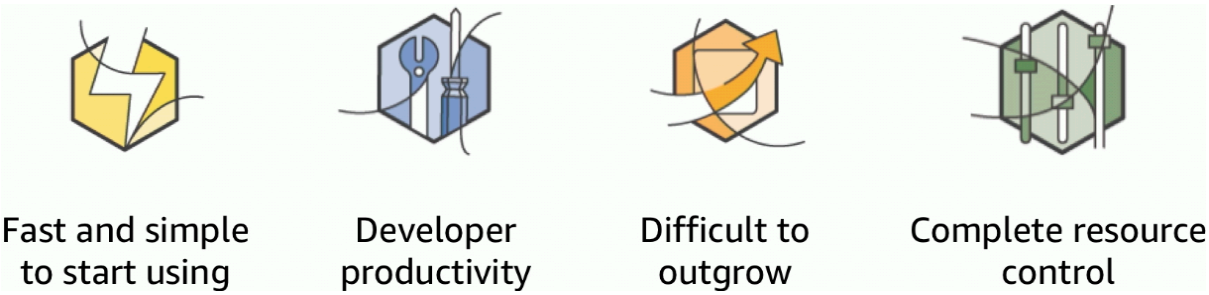
Elastic Beanstalk provides all the services that you need



AWS Elastic Beanstalk deployments

- Supports web app written for common platforms
    - Java, .NET, PHP, Node.js, Python, Ruby, Go and Docker
- You upload your code
    - Elastic Beanstalk automatically handles the deployment
    - Deploys on servers such as Apache, NGINX, Passenger, Puma, and Microsoft Internet Information Services (IIS)

Benefits of Elastic Beanstalk



Chapter 11

Core Services - Storage - EBS

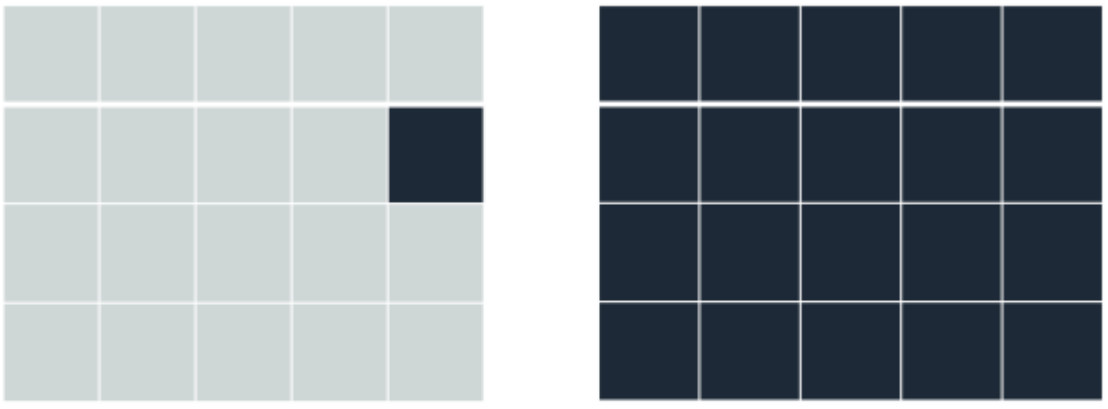
Amazon Elastic Block Store (Amazon EBS)

Storage

- Provides persistent block storage volumes with Amazon EC2 instances
- Called non-volatile storage
- Replicated within AZ

Block-level versus object-level storage

What if you want to change one character in a 1-GB file ?



- With block storage, you change only the block that contains the character.
- With object storage, the entire file must be updated.

Difference

This difference has a major impact on the throughput, latency, and cost of your storage solution.

Amazon EBS

Amazon EBS enables you to create individual storage volumes and attach them to an Amazon EC2 instance

- Amazon EBS offers block-level storage
- Volumes are automatically replicated within its AZ
- Can be backed up automatically to Amazon S3 through snapshots
- Uses include
    - Boot volumes and storage for Amazon Elastic Compute Cloud (Amazon EC2) instance
    - Data storage with a file system
    - Database hosts
    - Enterprise app

Amazon EBS volume types

	Solid State Drives (SSD)		Hard Disk Drives (HDD)	
	General Purpose	Provisioned IOPS	Throughput-Optimized	Cold
Maximum Volume Size	16 TiB	16 TiB	16 TiB	16 TiB
Maximum IOPS/Volume	16,000	64,000	500	250
Maximum Throughput/Volume	250 MiB/s	1,000 MiB/s	500 MiB/s	250 MiB/s

Snapshots, encryption, elasticity

- Snapshots
    - Point-in-time snapshots
    - Recreate a new volume at any time
- Encryption
    - Encrypted Amazon EBS volumes
    - No additional cost
- Elasticity
    - Increase capacity
    - Change to different types

Volumes, IOPS and pricing

1. Volumes
    - Amazon EBS volumes persist independently from the instance
    - All volume types are charged by the that is provisioned per month
2. IOPS
    - General Purpose SSD
        - Charged by the amount that you provision in GB per month until storage is released
    - Magnetic
        - Charged by the number of requests to the volume
    - Provisioned IOPS SSD
        - Charged by the amount that you provision in IOPS (multiplied by the percentage of days that you provision for the month
3. Snapshots
    - Added cost of Amazon EBS snapshots to Amazon S3 is per GB-month of data stored
4. Data transfer
    - Inbound data transfer is free
    - Outbound data transfer across Regions incurs charges

Chapter 12

Core Services - Storage - S3

Amazon Simple Storage Service (Amazon S3)

Storage

Amazon S3 is object-level storage.

- If want to change part of a file, must do the change and reupload the entire file

Amazon S3 overview

- Data stored as objects in buckets
- Virtually unlimited storage
  - Single object is limited to 5 TB
- Designed for 11 9s of durability
- Granular access to bucket and objects
- Data private per default
- Can set up notification
  - When object is added
  - When object is deleted

Amazon S3 storage classes

Amazon S3 offers a range of object-level storage classes that are designed for different use cases

- Amazon S3 standard
    - High availability
    - High durability
    - Performance
    - Frequently access data
- Amazon S3 Intelligent-Tiering
    - Optimize cost
    - Moving data to the most cost-effective access tier
    - long-live data with unpredictable access pattern
- Amazon S3 Standard-Infrequent Access (Amazon S3 Standard-IA)
    - Data accessed less frequently
    - long-term storage
- Amazon S3 One Zone-Infrequent Access (Amazon S3 One Zone-IA)
    - Data accessed less frequently
    - Stores data in a single availability zone
- Amazon S3 Glacier
    - Secure
    - Durable
    - low cost
    - data archiving
    - three retrieval options
        - min to hours
- Amazon S3 Glacier Deep Archive
    - Lowest cost
    - long-term detention
    - retrieved once or twice a year

Amazon S3 bucket URLS (two styles)

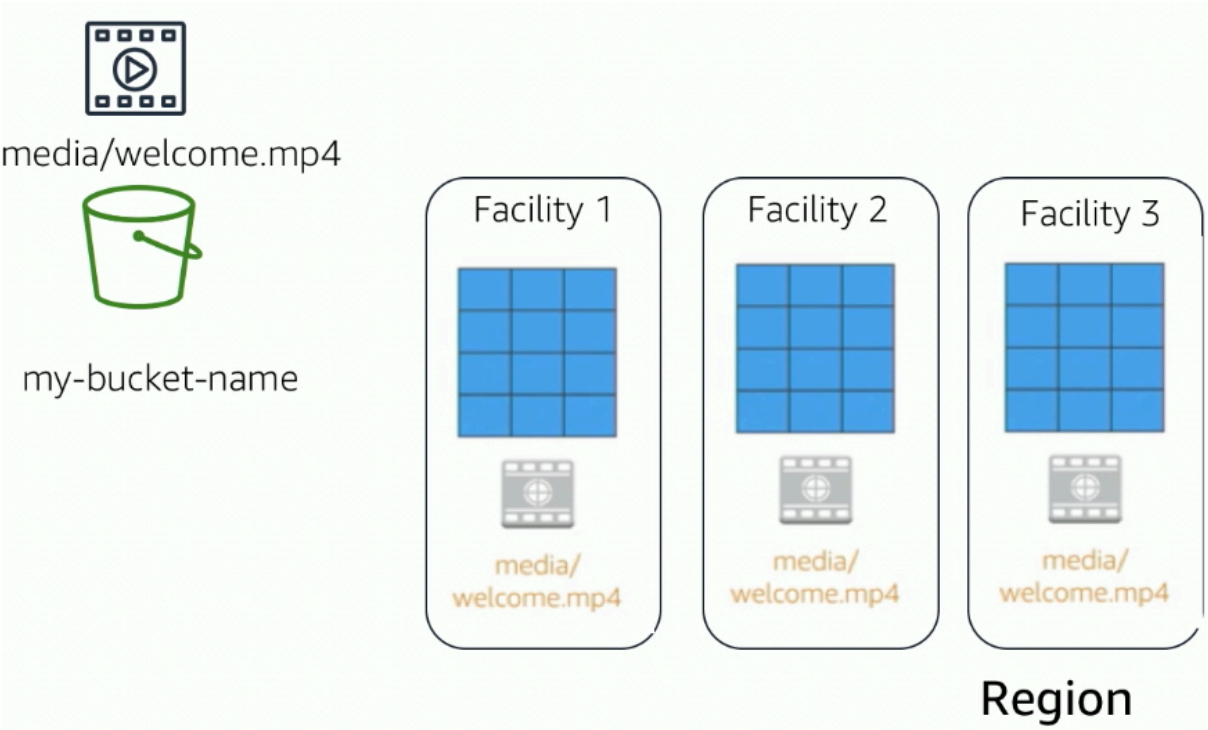
To upload your data:

1. Create a bucket in an AWS Region
2. Upload almost any number of objects to the bucket

Bucket path-style URL endpoint: https://s3.ap-northeast-1.amazonaws.com/bucket-name
Bucket virtual-hosted-style URL endpoint
https://bucket-name.s3-ap-northeast-1.amazonaws.com

Data is redundantly stored in the Region

Prevent data loss



Designed for seamless scaling

Amazon S3:

- automatically manage the storage
- scales to handle high volume of request
- billed for what you use

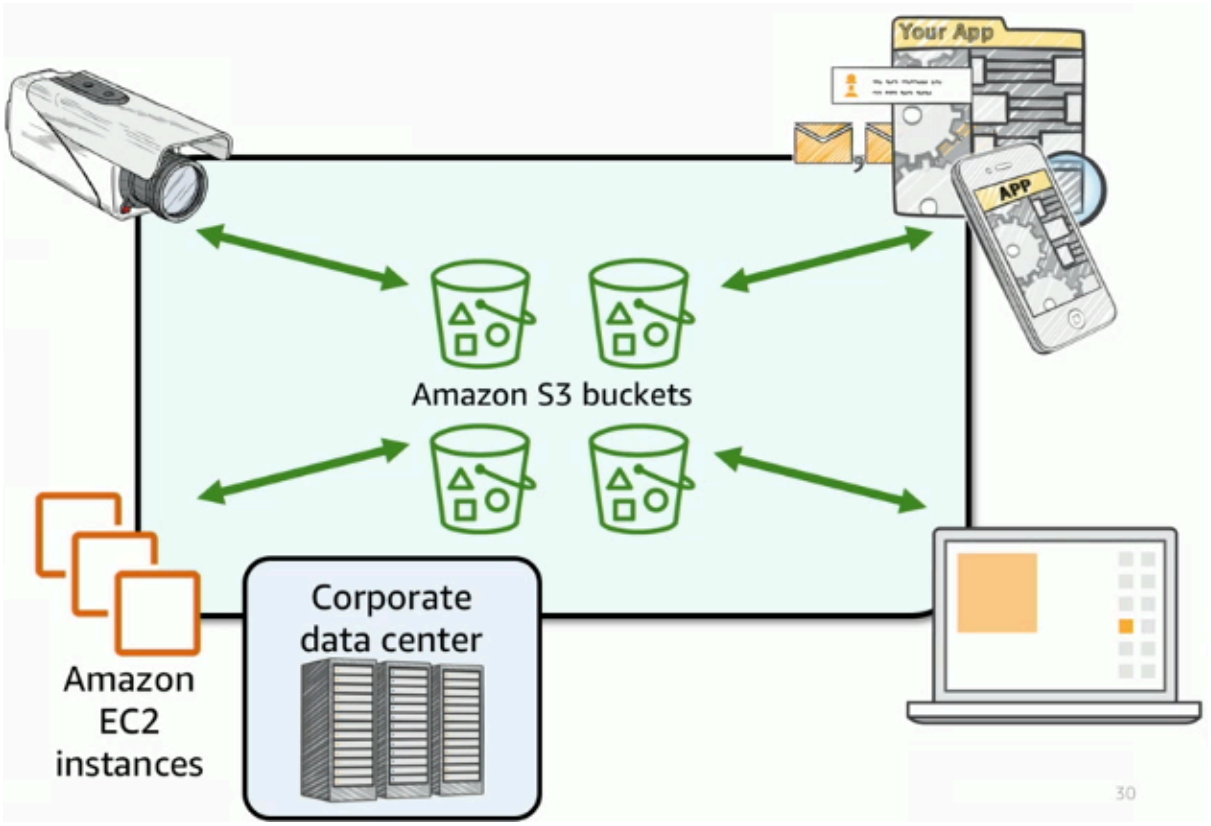
Access the data anywhere

- AWS CLI
- AWS Management Console
- SDK

Bucket names must be globally unique and DNS compliant: all lowercase, only letters, numbers and dashes

Amazon S3 common scenarios

- Backup and storage
- Application hosting
- Media hosting
- Software



Amazon S3 pricing

- Pay for what you use
  - GBs per month
  - Transfer OUT to other Regions
  - PUT, COPY, POST, LIST and GET requests
- You do not pay for
  - Transfers IN to Amazon S3
  - Transfers OUT from Amazon S3 to Amazon CloudFront or Amazon EC2 in the same region

Amazon S3: Storage pricing

To estimate Amazon S3 costs:

1. Types of storage classes
   - Standard storage is for
     - 11 9s of durability
     - 4 9s of availability
   - S3 Standard-Infrequent Access (S-IA) is for
     - 11 9s of durability
     - 3 9s of availability
2. Amount of storage
   - The number and size of objects

3. Requests
    - Number of requests (GET, PUT, COPY)
    - Type of requests
        - Different rates for GET requests
4. Data transfer
    - Pricing based on amount of data transferred out of Amazon S3 Region
        - Data transfer in is free, but incur charges for data transferred out

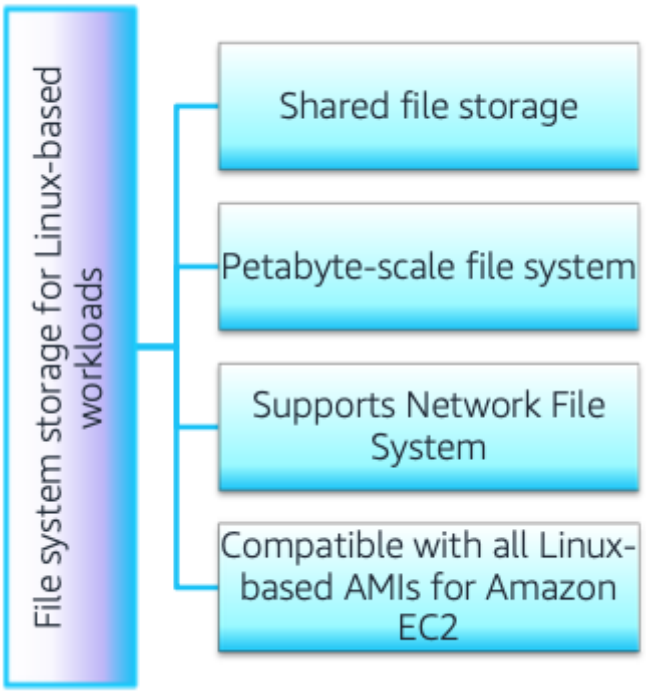
Chapter 13

Core Services - Storage - EFS

Amazon Elastic File System (Amazon EFS)

Storage

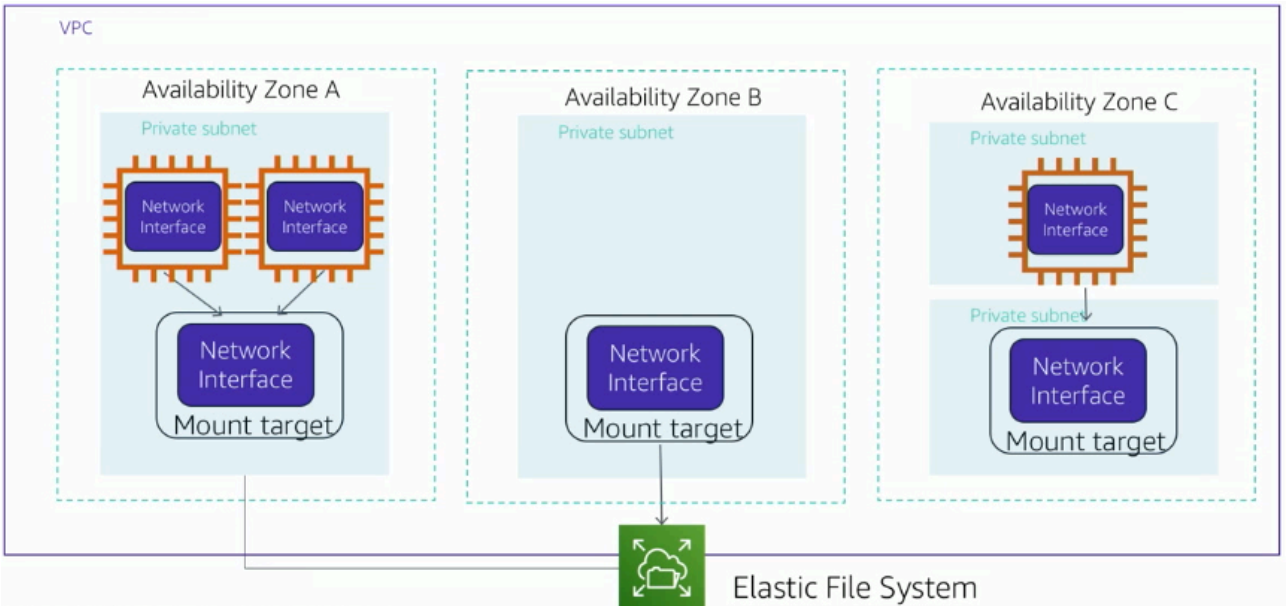
EFS is a file storage service for use with Amazon compute (EC2, containers, serverless) and on-premises servers. EFS provides a file system interface, file system access semantics (such as strong consistency and file locking), and concurrently accessible storage for up to thousands of EC2 instances.



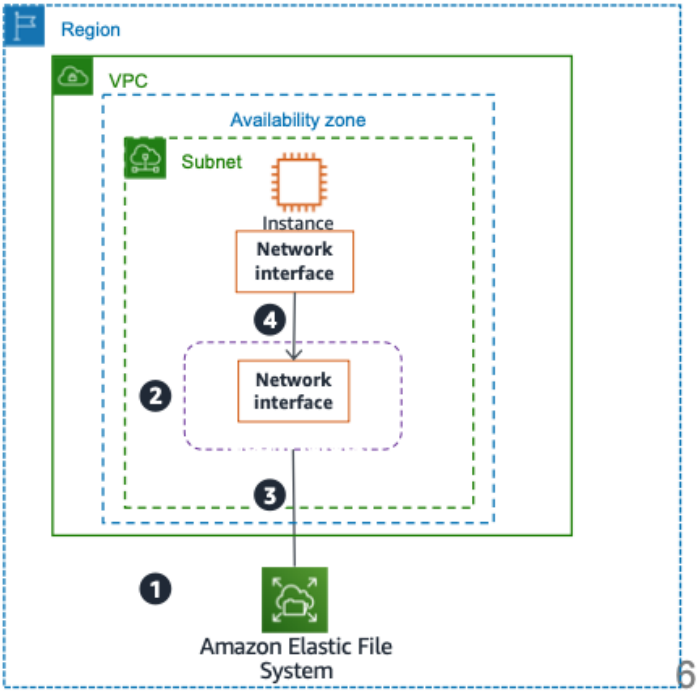
Features

- File storage in the AWS Cloud
- Works well for big data and analytics, media processing workflows, content management, web serving and home directories
- Petabyte-scale, low-latency file system
- Shared storage
- Elastic capacity
    - Gigabytes to petabytes of data
- Supports Network File System (NFS) versions 4.0 and 4.1 (NFSv4)
- Compatible with all Linux-based AMIs for Amazon EC2
- Pay for what you use

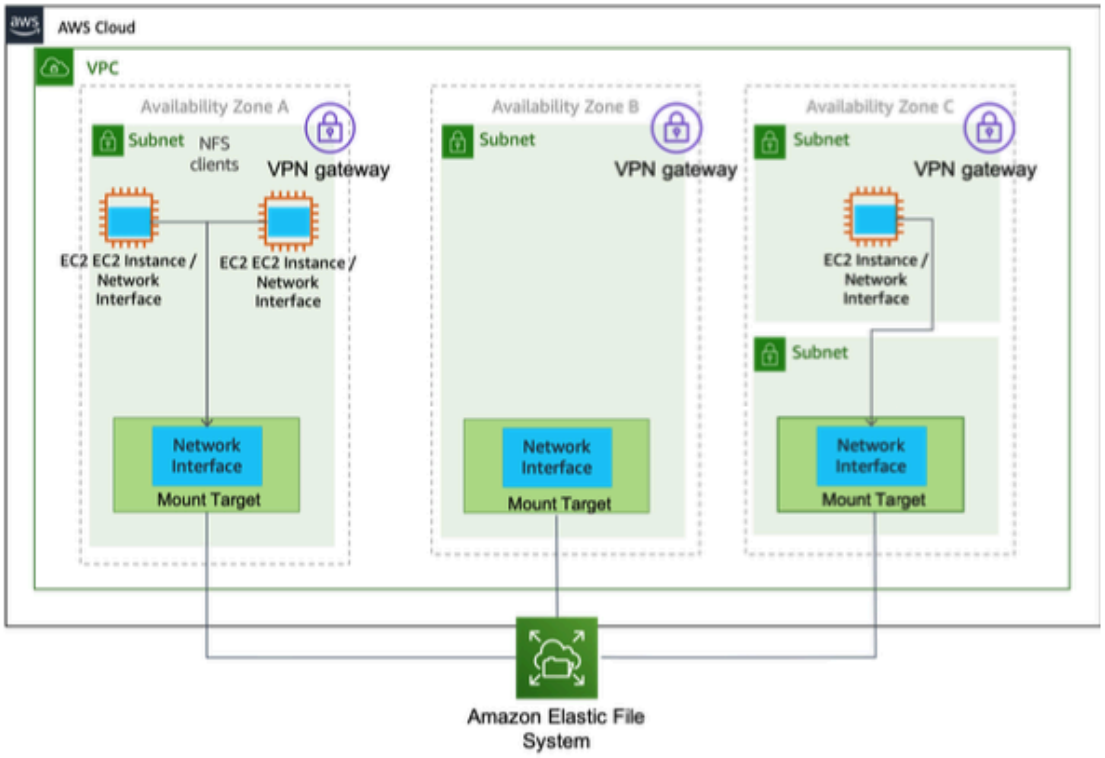
Amazon EFS architecture



Amazon EFS file system setup



Amazon EFS implementation



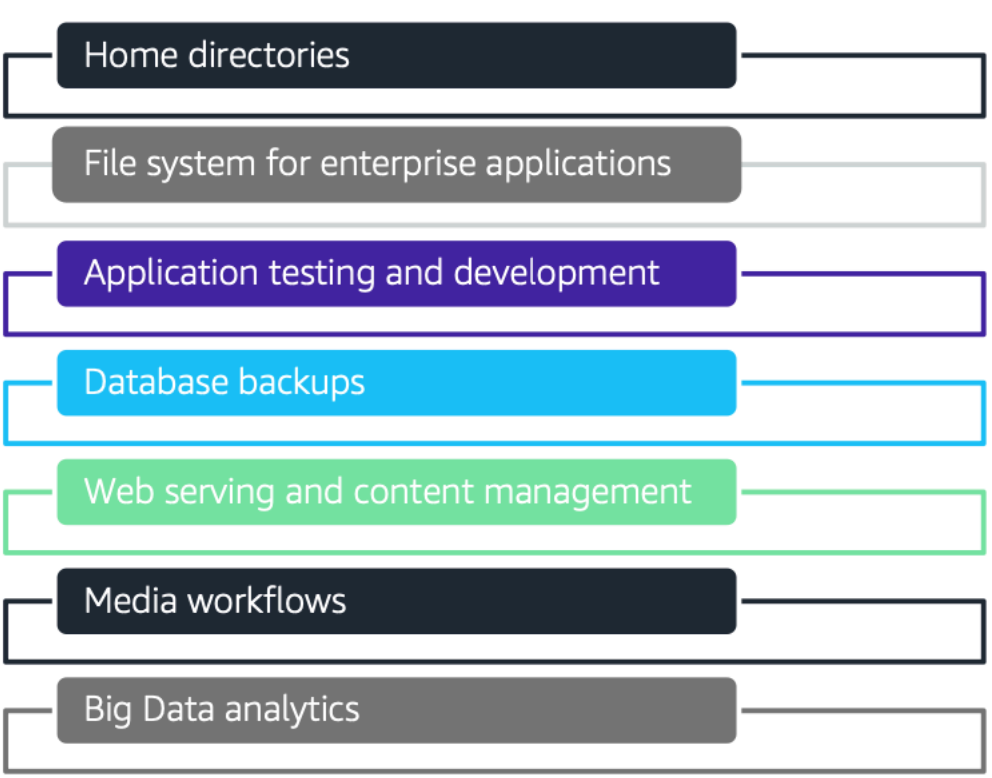
1. Create your Amazon EC2 resources and launch your instance
2. Create your Amazon EFSfile system
3. Create your mount targets in the appropriate subnets
4. Connect your Amazon EC2 instances to the mount targets
5. Verify the resources and protection of your AWS account

Amazon EFS resources

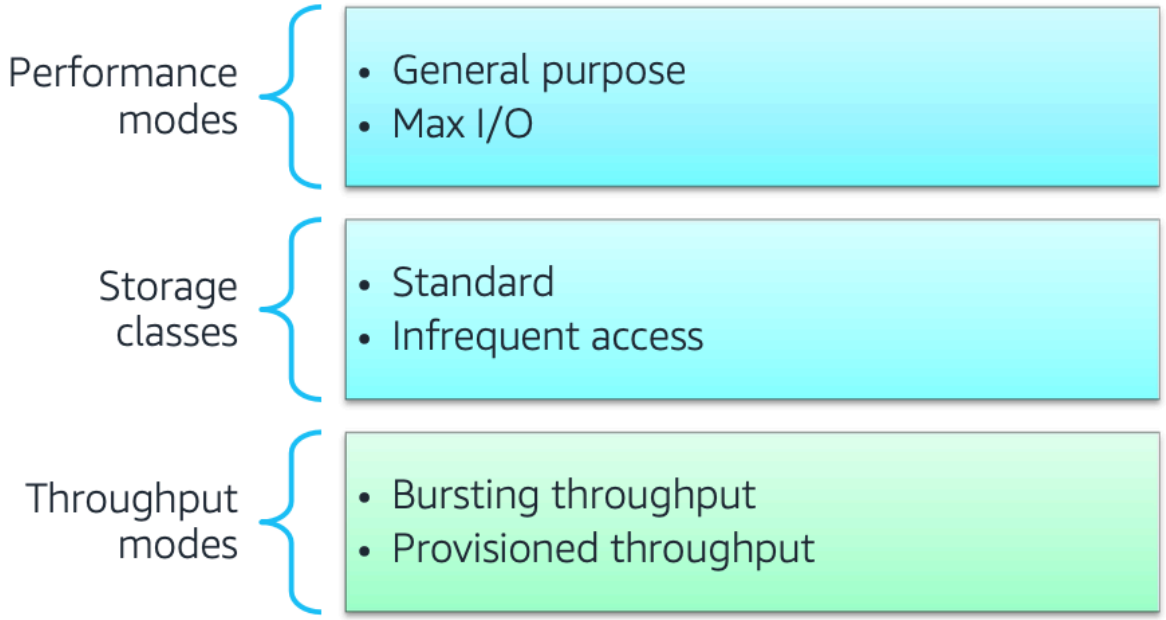
- Mount target
  - Subnet ID
  - Security groups
  - One or more per file system
  - Create in a VPC subnet
  - One per AZ
  - Must be in the same VPC
- Tags
  - Key0value pairs

Amazon EFS use cases

Amazon EFS is designed to provide performance for a broad spectrum of workloads and applications.



Amazon EFS performance attributes



Chapter 14

Core Services - Storage - Glacier

Amazon S3 Glacier

Storage

The Amazon S3 Glacier storage classes are purpose-built for data archiving, providing you with the highest performance, most retrieval flexibility, and the lowest cost archive storage in the cloud.

- Archive
  - Any object such as photo, video, file or document stored in Amazon S3 Glacier
  - Bas unit of storage
  - unique ID
- Vault
  - Container for storing archive
  - Specifies vault name
  - Permissions access policy
  - Vault lock policy

Amazon S3 Glacier review

- Designed to provide 11 9s of durability for objects
- Supports encryption of data in transit/at rest through Secure Sockets Layer (SSL) or Transport Layer Security (TLS)
- Vault lock: enforces compliance through a policy
- Extremely low-cost for long-term archiving
  - Three options: expedited, standard or bulk
  - Retrieval times from a few minutes to hours

Amazon S3 Glacier

- Storage service for low-cost data archiving and long-term backup
- Configure lifecycle archiving Amazon S3 content to Amazon S3 Glacier
- Retrieval options:
    - Standard: 3-5 hours
    - Bulk: 5-12 hours
    - Expedited: 1-5 min

Amazon S3 Glacier use cases

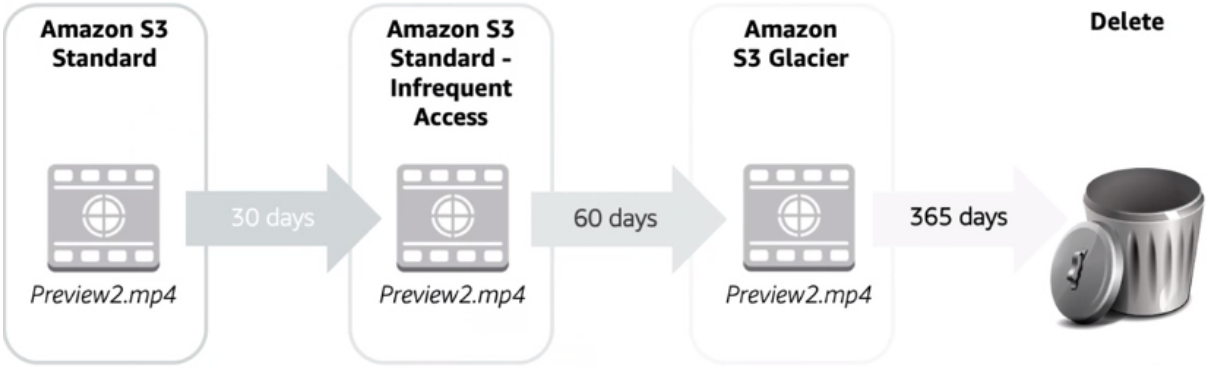
- Media asset archiving
- Healthcare info archiving
- Regulatory and compliance archiving
- Sicentific data archiving
- Digital preservation
- Magnetic tape replacement

Using Amazon S3 Glacier

- RESTful web services
- Java or .NET SDKs
- Amazon S3 with lifecycle policies

Lifecycle policies

Amazon S3 lifecycle policies enable you to delete or move objects based on age.



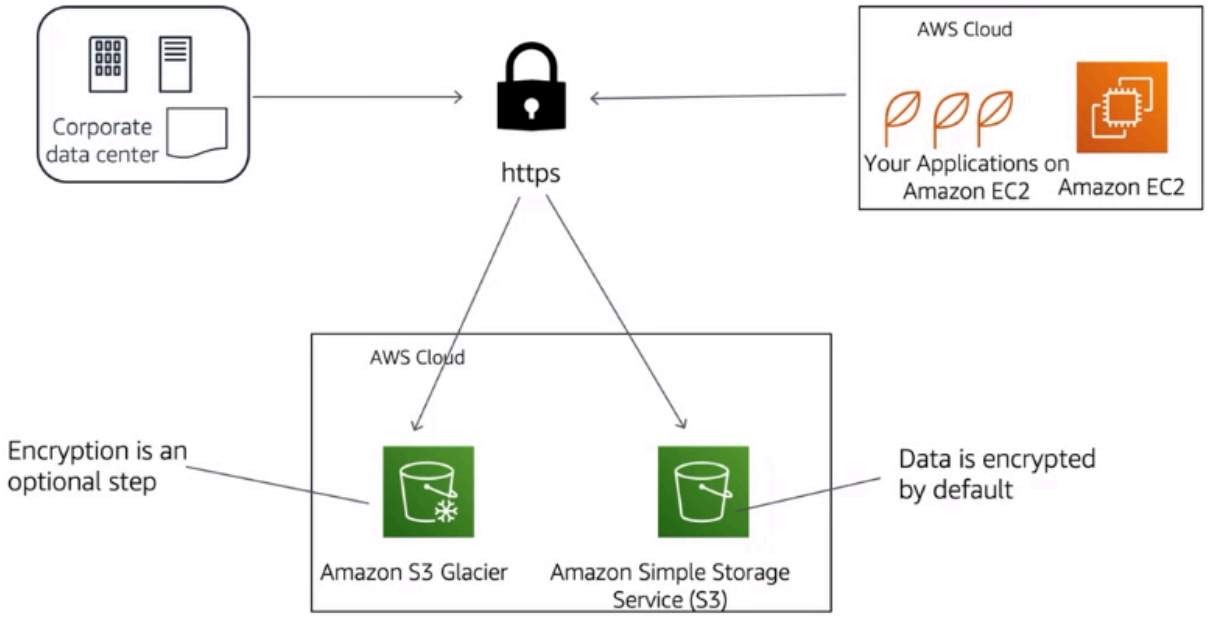
Amazon S3 storage classes

Storage Class	Features
S3 Standard	• ≥ three Availability Zones
S3 Standard-Infrequent Access (IA)	• Retrieval fee is associated with objects • Most suitable for infrequently accessed data
S3 Intelligent-Tiering	• It automatically moves objects between tiers based on access patterns • ≥3 Availability Zones
S3 One Zone-IA	• One Availability Zone • Costs less than Amazon S3 Standard-IA
S3 Glacier	• It is not available for real-time access • You must restore objects before you can access them • Restoring objects can take between 1 minute and 12 hours
S3 Glacier Deep Archive	• Lowest cost storage for long-term retention (7–10 years) • ≥ three Availability Zones • Retrieval time within 12 hours

Storage comparison

	Amazon S3	Amazon S3 Glacier
Data Volume	No limit	No limit
Average Latency	ms	minutes/hours
Item Size	5 TB maximum	40 TB maximum
Cost/GB per Month	Higher cost	Lower cost
Billed Requests	PUT, COPY, POST, LIST, and GET	UPLOAD and retrieval
Retrieval Pricing	¢ Per request	¢¢ Per request and per GB

Server-side encryption



Server-side encryption:

- SSE S3
  - each objects has unique key
  - AES 256
- SSE-C
  - Own encryption keys
- AWS Key Management Service
  - Scaled for the cloud
  - Customer master keys
  - IAM Console or API

    - Access keys
    - How keys can be used

Security with Amazon S3 Glacier

- Controle access with IAM
- Amazon S3 Glacier encrypts your data with AES-256
- Amazon S3 Glacier manages your keys for you

Chapter 15

Core Services - VPC - Virtual Private Cloud

Chapter Overview

At the core of this lesson, you will:

- Gain an understanding of key concepts related to Amazon Virtual Private Cloud (Amazon VPC) and security groups.

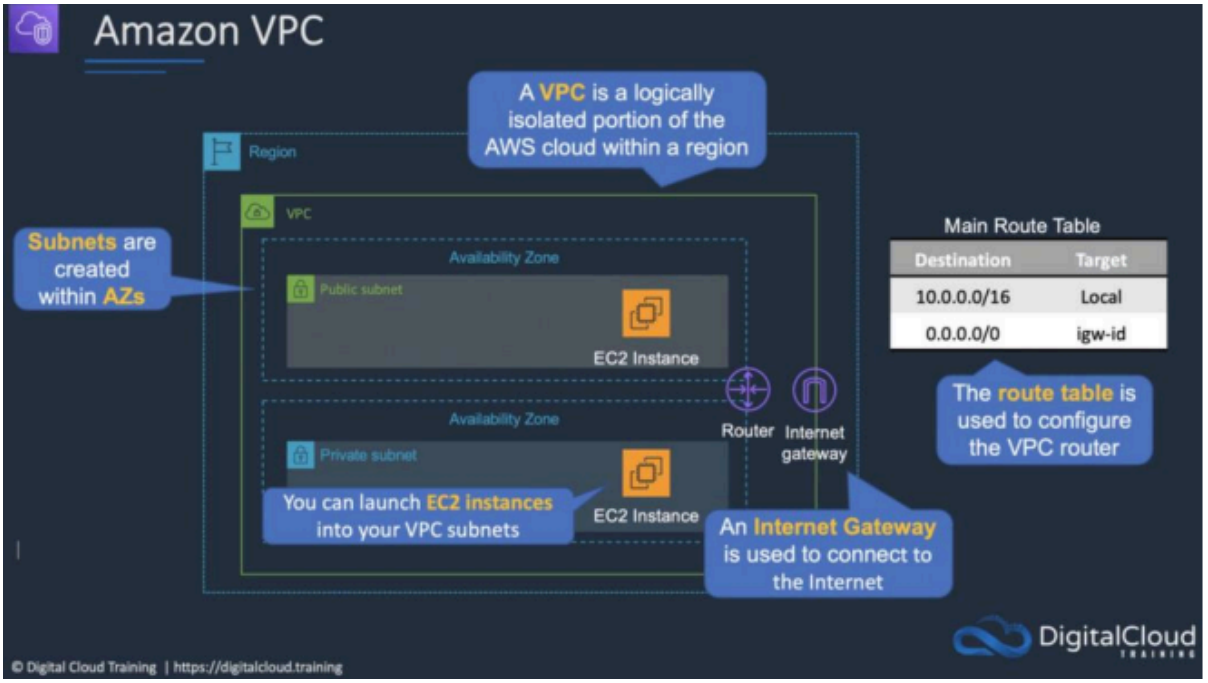
- Explore the concept of virtual networking in the cloud using Amazon VPC.

- Learn the process of creating virtual firewalls with security groups.

Introduction

Understanding Amazon VPC

Amazon VPC is a private section of AWS that allows users to control and place AWS resources, such as EC2 instances and databases, within a logically isolated virtual network. It provides full control over who can access the resources placed inside the VPC, offering a secure and customizable environment for deploying cloud resources.



Hands-On Experience

As part of the learning process, you will have the opportunity to gain hands-on experience with Amazon VPC through guided labs and practical exercises. This interactive approach will enable you to apply the knowledge in a real-world context and develop essential skills in cloud networking.
By the end of this chapter, you will have a solid understanding of Amazon VPC, its components, and its role in cloud computing, setting the stage for further exploration and practical application of VPC in AWS environments.

Network environment in the cloud

Control of Network Configuration in AWS Cloud

In the AWS cloud environment, the control of network configuration is a fundamental aspect that provides users with complete control over various network settings and policies. This includes the management of Internet Protocol (IP) address ranges, subnet creation, route table creation, network gateways, and security settings.

.
AWS offers the Amazon Virtual Private Cloud (VPC) service, which allows users to provision a logically isolated section of the AWS Cloud, providing complete control over the virtual networking environment. This includes the selection of IP address ranges, creation of subnets, and configuration of route tables and network gateways.

Deployment and Security Controls

In addition to network configuration, AWS provides several layers of security controls for deployment. Users have the ability to allow and deny specific internet and internal traffic, enhancing the security of the network environment.

.Furthermore, when deploying other AWS services into Amazon VPC, these services inherit the security built into the network, ensuring a secure and protected environment for the deployment of various applications and resources.

Features

The AWS cloud network environment builds on the high availability of AWS Regions and Availability Zones. Each VPC lives in a single Region, and multiple VPCs can be created per account. Subnets are used to divide VPCs, enabling Amazon VPC to span multiple Availability Zones, thereby enhancing fault tolerance and resilience.
.

Classless Inter-Domain Routing (CIDR)

Imagine Amazon Web Services (AWS) as a giant city in the cloud. Every service and resource in AWS needs a unique address, just like every house in a city.

1. Why is CIDR Important?

- Efficient Addressing

Think of AWS as a vast neighborhood. CIDR helps AWS organize its address space smartly. It's like saying, "This neighborhood starts with 10, and here's a map to efficiently find each house."

- Resource Management

The CIDR block (like 10.0.0.0/16) is AWS's way of saying, "This is the size of our neighborhood." It ensures that AWS doesn't waste addresses or run out of space, making it easier to manage resources effectively.

2. Breaking Down "10.0.0.0/16"

Choosing a Neighborhood

The "10" at the beginning is like choosing a big neighborhood. It's a broad area in AWS.
Dividing the Neighborhood:

The "/16" tells us how AWS divides this neighborhood. In this case, it's like saying, "Let's have streets that start with 10.0, and the rest can be individual houses on those streets."
The Range of Houses:

So, the addresses go from 10.0.0.1 to 10.0.255.254. Each house in this range belongs to the AWS neighborhood we've defined.

3. Why This Matters in AWS

Isolation

Each AWS service or project can have its own neighborhood (CIDR block), preventing them from interfering with each other. It's like having separate districts in a city.
Easy Routing:

When data (like messages or files) needs to travel within AWS, CIDR helps the system know exactly where to go. It's like having clear signs on the roads of our AWS city.

4. The Big Picture

In a real city, city planners decide the size of neighborhoods to ensure efficient use of space. Similarly, AWS uses CIDR to plan its city in the cloud, making sure every service has its space without wasting resources.
By understanding CIDR, we're essentially learning how AWS organizes its virtual city, ensuring it runs smoothly, efficiently, and without running out of addresses!

.
In summary, the AWS cloud environment provides robust capabilities for controlling network configuration, deploying services securely, leveraging high availability features, and efficiently managing IP address allocation through CIDR blocks.
.

Amazon VPC components

Understanding the components of Amazon VPC is essential for effectively managing and securing cloud-based networks and resources. Let's explore the key components of Amazon VPC:

1. Subnets

Subnets are segments of a VPC's IP address range where AWS services can be launched. It's important to note that subnets in an Availability Zone cannot span Availability Zones, meaning one subnet equals one Availability Zone. Subnets can be classified as public, private, or VPN only. Default VPCs contain one public subnet in every Availability Zone within the Region with a netmask of /20.

2. Route Tables

Route tables are used to control traffic going out of the subnets. They define the rules for routing network traffic.

3. Dynamic Host Configuration Protocol (DHCP) Option Sets

DHCP option sets provide a standard for passing configuration information to hosts on a TCP/IP network. They allow for the automatic configuration of network devices.

4. Security Groups

Security groups act as virtual, stateful firewalls, controlling inbound and outbound traffic for instances.

5. Network Access Control Lists (Network ACLs)

Network ACLs control access to subnets and are stateless, meaning they do not keep track of the state of the traffic.

6. Internet Gateway

An internet gateway allows access to the internet from the VPC, enabling communication with the internet and other AWS services.

7. Elastic IP Address

An Elastic IP address is a static, public IP address that can be pulled from a pool for use on a temporary basis. It provides a consistent IP address for dynamic cloud computing.

8. Elastic Network Interface

An Elastic Network Interface is a virtual network interface that can be attached to an instance in a VPC.

9. Endpoints

Endpoints provide a direct connection to another AWS service from your VPC without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect.

10. Peering

Peering allows two VPCs to communicate with each other using private IP addresses as if they were part of the same network.

11. Network Address Translation (NAT) Instances and NAT Gateways

NAT instances and NAT Gateways accept, translate, and forward traffic within a private subnet, allowing instances in a private subnet to initiate outbound traffic to the internet. Understanding these components is crucial for effectively managing and securing Amazon VPCs, ensuring optimal performance and security for cloud-based networks and resources

Amazon VPC Connections

AWS Hardware VPN

AWS Hardware VPN allows customers to establish secure IPsec VPN connections between their on-premises networks and Amazon VPCs. This option involves configuring customer gateway devices and leveraging IPsec VPN connections over private lines to establish secure and reliable connectivity. It provides predictable network performance, reduced bandwidth costs, and supports BGP peering and routing policies on AWS Direct Connect. Customers can reuse existing VPN equipment and processes, but may require additional telecom and hosting provider relationships or new network circuits to be provisioned

AWS Direct Connect

AWS Direct Connect provides a dedicated network connection over private lines to regional routers for multiple VPCs. It offers a private, logical connection from a remote network to Amazon VPC, ensuring consistent network performance, reduced bandwidth costs, and support for BGP peering and routing policies. AWS Direct Connect can be used in conjunction with AWS Transit Gateway and AWS Site-to-Site VPN for enhanced connectivity options.

AWS VPN CloudHub

AWS VPN CloudHub facilitates the establishment of a hub-and-spoke model for connecting remote branch offices, providing primary or backup connectivity. It offers high availability and scalability, allowing reuse of existing internet connections and AWS VPN equipment and processes. This option is suitable for organizations looking to connect multiple VPCs in a hub-and-spoke model for efficient network communication

Software VPN

Software VPN options include AWS Client VPN, which is an AWS-managed high availability and scalability service enabling secure software remote access. It provides the option of creating a secure TLS connection between remote clients and Amazon VPCs, allowing secure access to AWS resources and on-premises networks over the internet. Remote clients can utilize the AWS Client VPN for Desktop or third-party OpenVPN VPN clients, with authentication via Active Directory or mutual certificate authentication

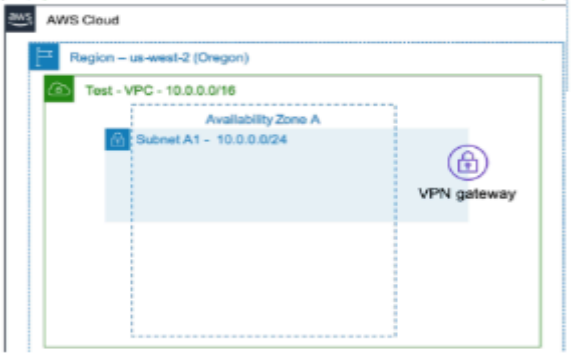
Design a VPC

Designing a Virtual Private Cloud (VPC) involves several steps, including creating the VPC, defining subnets, configuring route tables, and setting up security groups. The following is a step-by-step guide for designing a VPC:

1

Design VPC: Part 1

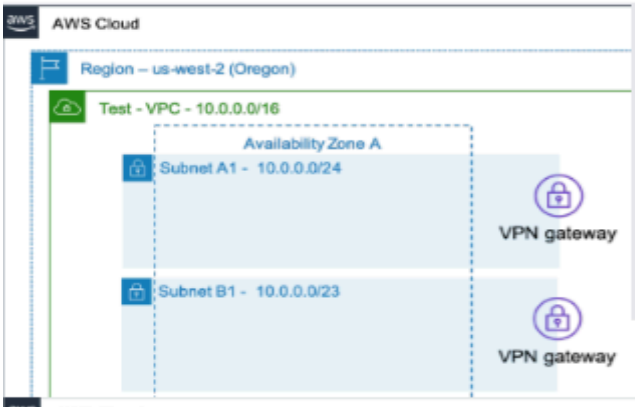
- Create the VPC and give it a name, such as Test VPC.
- Give your VPC any IP addresses in the following CIDR block: 10.0.0.0/16.



2

Design VPC: Part 2

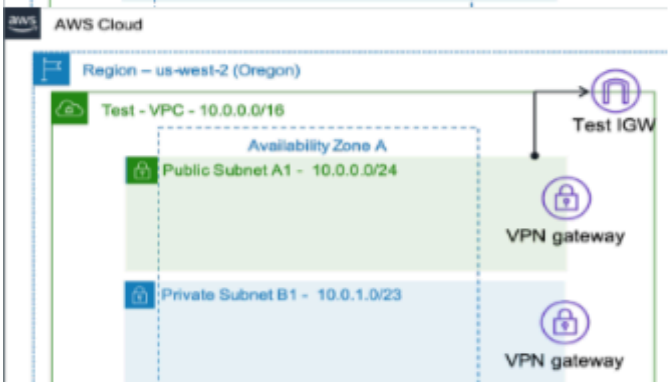
- Create a subnet named Subnet A1 and assign an IP address space that contains 256 IP addresses.



3

Design VPC: Part 3

- Create another subnet called Subnet B1 and assign an IP address space to it.



4

Design VPC: Part 4

- To accomplish this, add an internet gateway called Test IGW.
- Subnet A1 now becomes a public subnet where non-local traffic is routed through the internet gateway.
- Subnet B1 will be your private subnet that is isolated from the internet.

Chapter 16 : AWS CloudFront

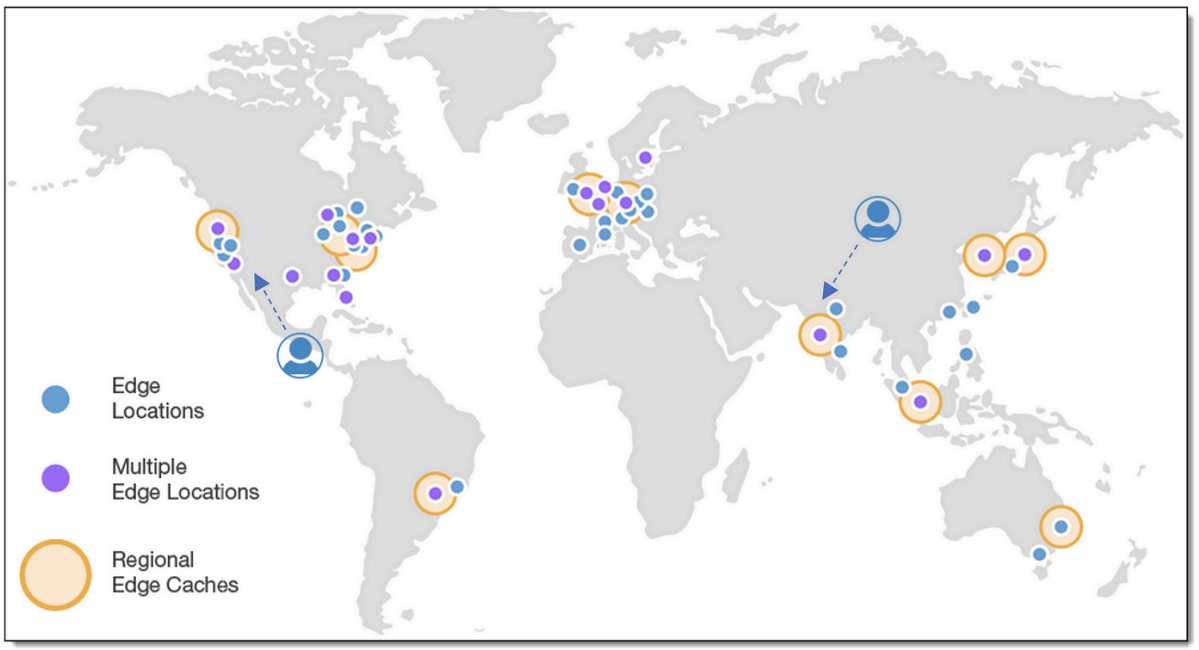
Chapter Overview

You'll learn:
- Global, Growing Content Delivery Network
- Secure Content at the Edge Location
- Programmable Content Delivery Network (CDN)

AWS CloudFront

What is Amazon CloudFront

Amazon CloudFront is a web service that accelerates the distribution of static and dynamic web content, such as .html, .css, .js, and image files, to users. It operates through a global network of data centers called edge locations. This service was designed to provide a highly programmable content delivery network (CDN) that caters to a wide array of content types, including websites, videos, music, applications, software, games, and APIs. It aims to enhance access speed for downloading content by caching it closer to end-users, thereby improving the overall user experience.



Key Features

Amazon CloudFront offers several key features that make it a popular choice for content delivery. These features include:

- **Global, Growing Content Delivery Network:** Amazon CloudFront has servers located in many countries around the world, including the United Kingdom, Ireland,

The Netherlands, Germany, Spain, Hong Kong, Singapore, Japan, Taiwan, Vietnam, Indonesia, India, Australia, South America, Africa, and several major cities in the United States. In November 2022, the service operated from 400 edge locations on six continents.

- **Secure Content at the Edge Location:** Amazon CloudFront offers traffic encryption and access controls, and uses AWS Shield Standard to defend against distributed denial-of-service (DDoS) attacks at no additional charge.

- **Programmable Content Delivery Network (CDN):** Amazon CloudFront offers programmable and secure edge CDN computing capabilities through CloudFront Functions and AWS Lambda@Edge.

- **High Performance:** Amazon CloudFront provides low latency and high data transfer speeds, making it a popular choice for content delivery

.

- **Cost-Effective:** Amazon CloudFront operates on a pay-as-you-go basis, meaning that you only pay for data transfer and requests to deliver content to customers. There are no upfront or minimum commitments

.

Cost Estimation

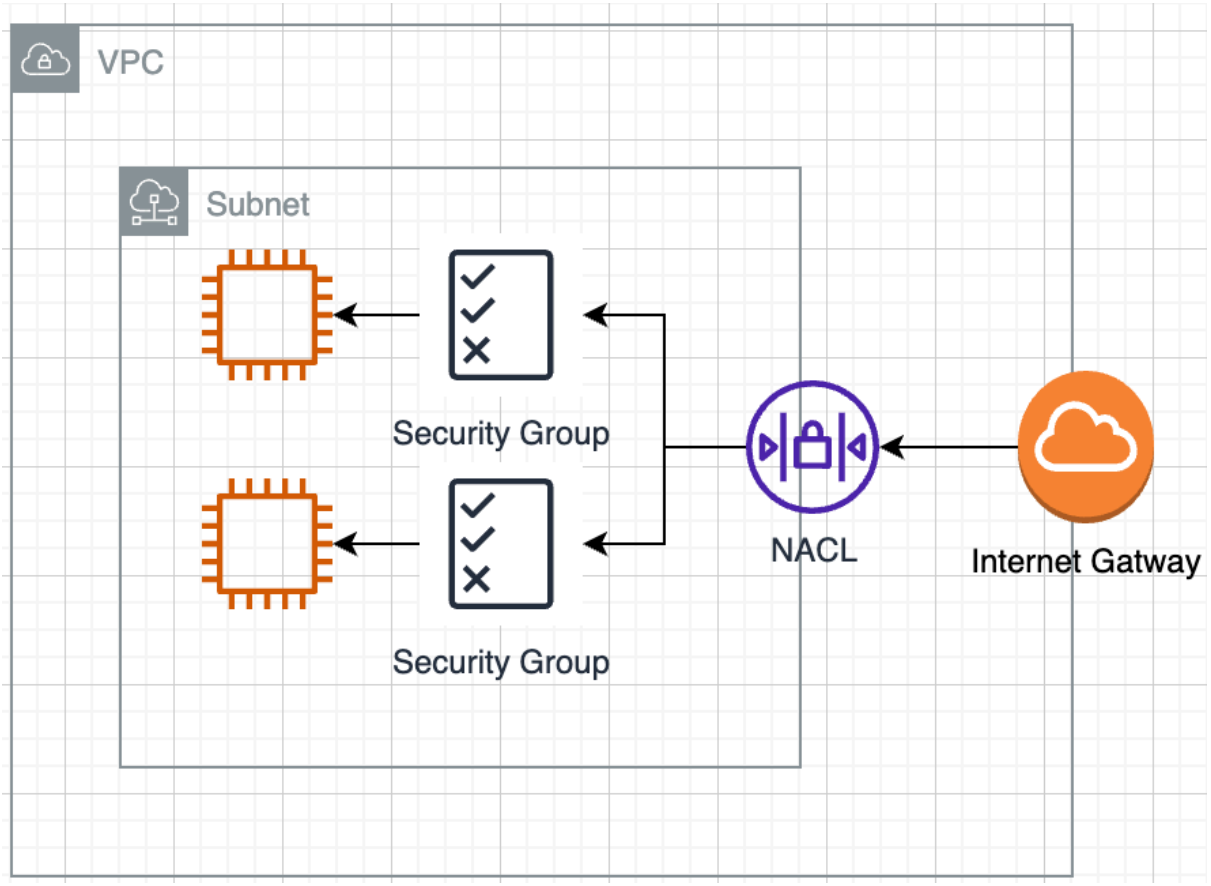
The cost of using Amazon CloudFront varies depending on several factors, including traffic distribution, requests, and data transfer out. Here are some key factors that affect the cost of using Amazon CloudFront:

- **Traffic Distribution:** Pricing varies across geographic regions and is based on the edge location.
- **Requests:** The number and type of requests, as well as the geographic region, can affect the cost of using Amazon CloudFront.
- **Data Transfer Out:** The amount of data transferred out of Amazon CloudFront edge locations can also affect the cost of using the service.

Conclusion

Amazon CloudFront is a highly programmable content delivery network (CDN) that distributes websites, videos, music, applications, software, games, and APIs to end-users. It offers several key features, including a global, growing content delivery network, secure content at the edge location, and programmable CDN capabilities. Amazon CloudFront is cost-effective and operates on a pay-as-you-go basis. The cost of using Amazon CloudFront

varies depending on several factors, including traffic distribution, requests, and data transfer out.



Chapter 17: AWS Security Groups

Chapter overview

- Understanding Security Groups and their role in securing data.
- Exploring Network Access Control Lists (NACLs) and their significance in network security.
- Learning about Key Pairs and their use in secure access management.

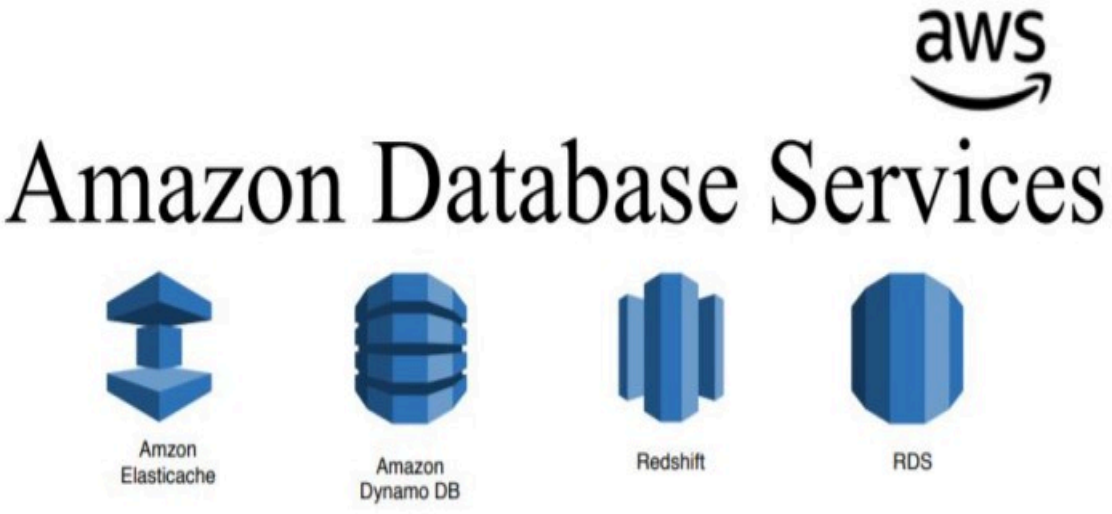
AWS security groups

Amazon Web Services (AWS) offers a suite of security features to safeguard user data, with Security Groups serving as a fundamental component. These groups function as built-in firewalls for virtual servers, controlling inbound and outbound traffic and determining access to instances. Additionally, Security Groups are stateful, meaning they retain information about traffic. In conjunction with Security Groups, Network Access Control Lists (NACLs) act as firewalls for associated subnets, while Key Pairs provide cryptographic capabilities for login information. Furthermore, Amazon VPC and Security Groups provide an additional layer of security, exemplified by the AWS multi-tier security group, which demonstrates the creation of multiple security group rules to accommodate a multi-tiered web architecture. NACLs allow or deny specific inbound or outbound traffic at the subnet level and are stateless, meaning that information about previously sent or received traffic is not saved.

The differences between NACLs and Security Groups have been discussed in various sources. NACLs are considered an optional form of defense for instances, and a subnet must have a NACL, but by default, it is configured to allow all traffic in and out. In contrast, Security Groups are locked down by default. Security Groups are stateful, automatically allowing return traffic, regardless of the specified rules, while NACLs are stateless, requiring explicit rules for traffic to be allowed. Combining Security Groups and NACLs effectively enhances security filtering capabilities, allowing for a specific set of rules at the subnet level using NACLs and a different set of instance-specific rules at the Security Group level.

In summary, Security Groups and NACLs are crucial components of AWS security, providing a built-in firewall for Amazon EC2 instances, controlling inbound and outbound traffic, and determining who has access to instances. By leveraging these security features, AWS users can ensure the safety and security of their data within the Amazon VPC environment.

Chapter 18: Amazon Database Services Overview



Chapter overview

- Understanding the challenges of relational databases
- Exploring the available database options offered by AWS
- Examining the difference between unmanaged and managed database solutions
- Describing the differences between a structured query language (SQL) database and a NoSQL database
- Explaining how to choose an AWS database that meets the needs of different business scenarios

Key Terms

- SQL
- NoSQL
- Database instance
- Relational database
- Managed services
- Unmanaged services

Challenges of relational databases

Relational databases pose several challenges:

- **Server Maintenance and Energy Footprint:** Relational databases often require significant server maintenance, including hardware upkeep and energy consumption. AWS offers managed database services that alleviate the burden of server maintenance, allowing users to focus on application development rather than infrastructure management.

- **Software Installation and Patches:** Installing and patching database software can be time-consuming and complex. AWS managed database services handle software installation and patching, ensuring that databases are up-to-date and secure without requiring manual intervention.

- **Database Backups and High Availability:** Managing database backups and ensuring high availability can be challenging. AWS provides automated backup solutions and high availability features for databases, reducing the operational overhead and enhancing data protection.

- **Limits on Scalability:** Traditional relational databases may encounter scalability limitations. AWS purpose-built databases, including relational and non-relational options, are designed for scalability, offering the ability to scale resources based on demand and workload requirements.

- **Data Security:** Data security is paramount, and AWS offers a range of security features, including encryption, access controls, and compliance certifications, to ensure the confidentiality, integrity, and availability of data stored in relational databases.

- **Operating System (OS) Installation and Patches:** Managing the operating system installation and patches for database servers can be complex. AWS managed database services handle the underlying infrastructure, including OS management, allowing users to focus on database usage and application development.

AWS database options

Amazon Relational Database Service (RDS)

Description: RDS stands out as a fully managed relational database service on AWS, catering to diverse database engines like MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
Use Case: Tailored for applications necessitating a conventional relational database structure. RDS automates routine database tasks, including backups and patch management, enabling developers to concentrate on application development.

Amazon DynamoDB

Description: DynamoDB, a fully managed NoSQL database service, boasts seamless scalability and predictable performance for both document and key-value data models.
Use Case: Suited for applications with high read and write demands, and those requiring a flexible schema design. It finds common application in web and mobile development, gaming, and Internet of Things (IoT) scenarios.

Amazon Redshift

Description: Redshift, a fully managed data warehouse service, is tailored for high-performance data analysis through SQL queries. It facilitates the analysis of large datasets and excels at handling complex queries across multiple tables.
Use Case: Suited for data warehousing and analytics, particularly in business intelligence and reporting scenarios where extensive data analysis is required.

Amazon Neptune

Description: Neptune, a fully managed graph database service, supports both Property Graph and RDF models. It is designed for applications requiring highly interconnected data structures.

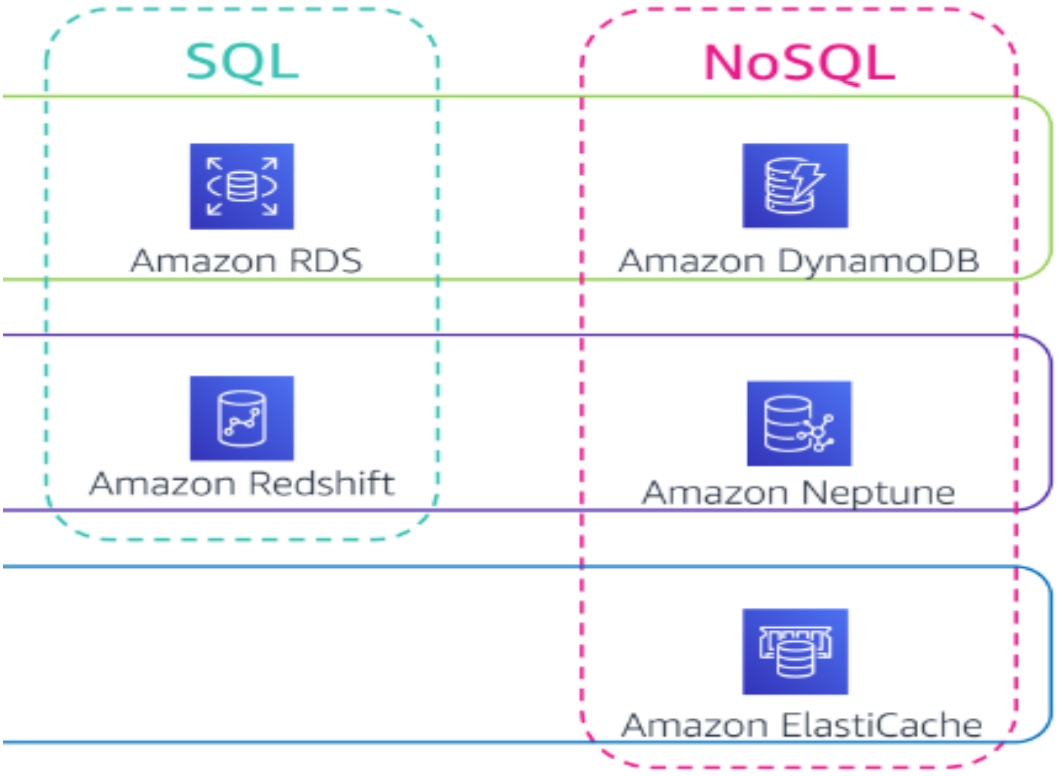
Use Case: Ideal for applications with data-centric relationships, such as social networking, fraud detection, and knowledge graphs. Neptune facilitates easy querying and navigation of relationships between data points.

Amazon ElastiCache

Description: ElastiCache, a fully managed in-memory caching service, supports popular caching engines like Redis and Memcached.
Use Case: Beneficial for enhancing web application performance by caching frequently accessed data, thereby reducing the load on backend databases.

Comprehending these diverse AWS database options empowers you to make informed choices based on the specific needs of your applications, be it scalability, performance, or data model flexibility.

SQL and NoSQL databases



SQL databases, often referred to as Relational Database Management Systems (RDBMS), and NoSQL databases represent two distinct approaches to data storage and management. In the realm of SQL databases, a structured, table-based system prevails, providing a well-defined framework for organizing and relating data. On the flip side, NoSQL databases take on a more diverse landscape, encompassing document-oriented, key-value pairs, and graph structures.

SQL databases adhere rigorously to the principles of ACID (Atomicity, Consistency, Isolation, and Durability), ensuring the integrity and validity of data transactions. In contrast, NoSQL databases align with the CAP theorem (Consistency, Availability, and Partition Tolerance), prioritizing availability and the ability to function seamlessly in the face of network partitioning.

	SQL	NoSQL
Data Storage	Rows and columns	Key-value
Schemas	Fixed	Dynamic
Querying	Uses SQL	Focuses on collection of documents
Scalability	Vertical	Horizontal

SQL

ISBN	Title	Author	Format
9182932465265	Cloud Computing Concepts	Jackson, Mateo	Paperback
3142536475869	The Database Guru	García, María	Ebook

NoSQL

```
{
    "ISBN": 9182932465265,
    "Title": "Cloud Computing Concepts",
    "Author": "Jackson, Mateo",
    "Format": "Paperback"
}
```

The strengths of SQL databases shine when dealing with related data and multi-row transactions, making them a stalwart choice for applications like accounting systems where data accuracy is paramount. On the other hand, NoSQL databases come into their own when tasked with managing extensive or constantly evolving datasets. They offer high performance, exceptional flexibility, and user-friendly interfaces, making them well-suited for applications in dynamic environments.

In the realm of support, SQL databases boast a broad network of vendors and independent consultants, providing a robust ecosystem for users. NoSQL databases, meanwhile, often rely more heavily on community support, fostering collaboration and innovation within the user community.

The language of interaction sets SQL and NoSQL databases apart. SQL databases utilize the Structured Query Language (SQL), a standardized language for managing and querying relational databases. On the contrary, NoSQL databases employ diverse query languages and data structures, such as JSON, XML, and YAML, providing adaptability to different types of data.

An additional feather in the cap of NoSQL databases is their horizontal scalability, a feature that makes them particularly well-suited for the demands of modern cloud-based infrastructures. This capability enables seamless expansion of database resources to meet the evolving needs of applications in the dynamic and scalable cloud environment.

To sum it up, SQL databases exhibit efficiency in handling structured and related data, making them a go-to choice for applications with strict data organization requirements. NoSQL databases, on the other hand, shine in managing structured, semi-structured, and unstructured data, offering a blend of flexibility and scalability that aligns well with the demands of contemporary and dynamic data landscapes.

Managed Services

AWS Managed Services (AMS) is a comprehensive solution that simplifies the management of AWS infrastructure, allowing users to focus on their applications. It aids organizations in scaling their AWS adoption efficiently and securely, leveraging standard AWS services and operational best practices. AMS ensures governance transparency through detailed reporting, including key performance metrics and cost-saving insights. It enforces compliance with enterprise governance standards using tags, and it can take over administration and maintenance tasks for Control Tower environments.

Automation is a key feature, streamlining activities like change requests, monitoring, patch management, security, and backups throughout the infrastructure's entire lifecycle. AMS facilitates swift and cost-effective migrations to AWS while improving compliance and security. The service introduces a secure AWS landing zone to meet various compliance program requirements and reduce operational overhead and risk.

Unmanaged vs. Managed Services

Unmanaged services empower users with full control over provisioning, scaling, fault tolerance, and availability. This control, however, requires a deep understanding of infrastructure management and places the responsibility for optimization squarely on the user.

On the other hand, **managed services** introduce a more user-friendly approach. Users configure the service to their needs, but crucial aspects like scaling, fault tolerance, and availability are seamlessly integrated into the service itself. This means that the service provider takes on the operational responsibilities, allowing users to focus on configuring the service for their applications without delving into the intricacies of infrastructure management.

In essence, the choice between unmanaged and managed services hinges on the user's preference for control and customization versus a desire for simplicity and offloading operational complexities. Both approaches have their advantages and considerations.

Managed Services Responsibilities

Managed Services outline the distinct responsibilities between customers and Amazon Web Services (AWS). This collaborative framework ensures the optimal performance and reliability of applications. Here's a breakdown:

You Manage: Application Optimization

As a customer, your primary focus lies in fine-tuning and optimizing your applications. This involves tailoring them to meet specific business needs and enhance overall efficiency.

AWS Manages: Infrastructure Foundations

AWS takes charge of foundational aspects critical to a robust infrastructure:

- OS Installation and Patches: Ensuring the operating system is installed and updated with the latest security patches.
- Database Software Install and Patches: Managing the installation and patching of database software to maintain data integrity and security.
- Database Backups: Implementing and maintaining robust backup systems to safeguard against data loss.
- High Availability: Ensuring applications are highly available through redundancy and failover mechanisms.
- Scaling: Automatically adjusting computing capacity to handle varying workloads seamlessly.
- Power and Rack & Stack: Managing physical aspects such as power distribution and rack configuration.
- Server Maintenance: Overseeing routine server maintenance, including hardware updates and replacements.

This collaborative approach allows customers to focus on optimizing their applications, while AWS handles the critical infrastructure elements. Clarity in responsibilities ensures a streamlined and efficient partnership in the realm of Managed Services.

AWS database recommendations

Amazon Web Services (AWS) offers a comprehensive suite of database services tailored to diverse needs, ranging from relational databases like Amazon RDS to NoSQL databases like Amazon DynamoDB.

Managed Relational Database: Amazon RDS

- **Description:** A managed relational database service launched with a choice of six popular database engines.
- **Suggested Use Case:** Ideal for applications that require a traditional relational database management system with the flexibility to choose from various engines.

MySQL- and PostgreSQL-Compatible Relational Database: Amazon Aurora

- **Description:** A relational database built for the cloud, compatible with MySQL and PostgreSQL.
- **Suggested Use Case:** Suited for applications that demand high performance and compatibility with MySQL or PostgreSQL, providing the benefits of a cloud-native architecture.

Fully Managed NoSQL Database: Amazon DynamoDB

- **Description:** A fully managed NoSQL database service.
- **Suggested Use Case:** Excellent for applications with high scalability requirements, particularly suited for those that require low-latency access to data, such as gaming or mobile applications.

Managed Graph Database: Amazon Neptune

- **Description:** A managed graph database service.
- **Suggested Use Case:** Suitable for applications that need to navigate and analyze highly connected data, such as social networking or fraud detection systems.

Managed Redis Key-Value Store: Amazon ElastiCache

- **Description:** A managed Redis key-value store.
- **Suggested Use Case:** Ideal for caching frequently accessed data, session storage, and real-time analytics.

Columnar Storage SQL Data Warehouse: Amazon Redshift

- **Description:** A columnar storage SQL data warehouse.
- **Suggested Use Case:** Suited for complex analytical queries and large-scale data warehousing, often used for business intelligence and data analytics.

Customer Needs	Recommended Database
MySQL- and PostgreSQL-compatible relational database built for the cloud	Amazon Aurora
Managed relational database that is launched with a choice of six popular database engines	Amazon RDS
Fully managed NoSQL database	Amazon DynamoDB
Managed graph database	Amazon Neptune

Managed Redis key-value store	Amazon ElastiCache
Columnar storage SQL data warehouse	Amazon Redshift

AWS database usage scenarios

We will explore the practical applications of AWS databases. We'll take a closer look at real-world scenarios to understand how different AWS databases can be effectively employed.

1. Amazon RDS and Amazon Aurora: Transactional Excellence

Typical Applications

- **ERP (Enterprise Resource Planning):** Used for managing and integrating business processes, including financials, supply chain, and human resources.
- **CRM (Customer Relationship Management):** Streamlines customer interactions, storing and managing customer data effectively.
- **E-commerce Platforms:** Logs transactions and structures data for seamless online retail experiences.

Amazon Aurora Differentiator

When we say "Amazon Aurora Differentiator," we're highlighting features that distinguish it from standard Amazon RDS services:

- **Open Source, High Performance, and Low Cost:** Amazon Aurora stands out for being open source, delivering high performance, and offering cost-effectiveness. This is particularly beneficial in transactional applications like ERP, CRM, and e-commerce.

- **Compatibility:** Amazon Aurora is designed to be compatible with MySQL and PostgreSQL, providing a seamless transition for users of these databases within the Amazon RDS family.

- **Performance:** Amazon Aurora excels in terms of performance, especially in write-intensive workloads, making it a preferred choice for applications with demanding transactional requirements.

Example

In an ERP system handling a high volume of financial transactions, Amazon Aurora's compatibility with MySQL and PostgreSQL, along with its enhanced performance, make it a compelling choice over standard Amazon RDS.

2. Amazon Redshift: Powering Analytics at Scale

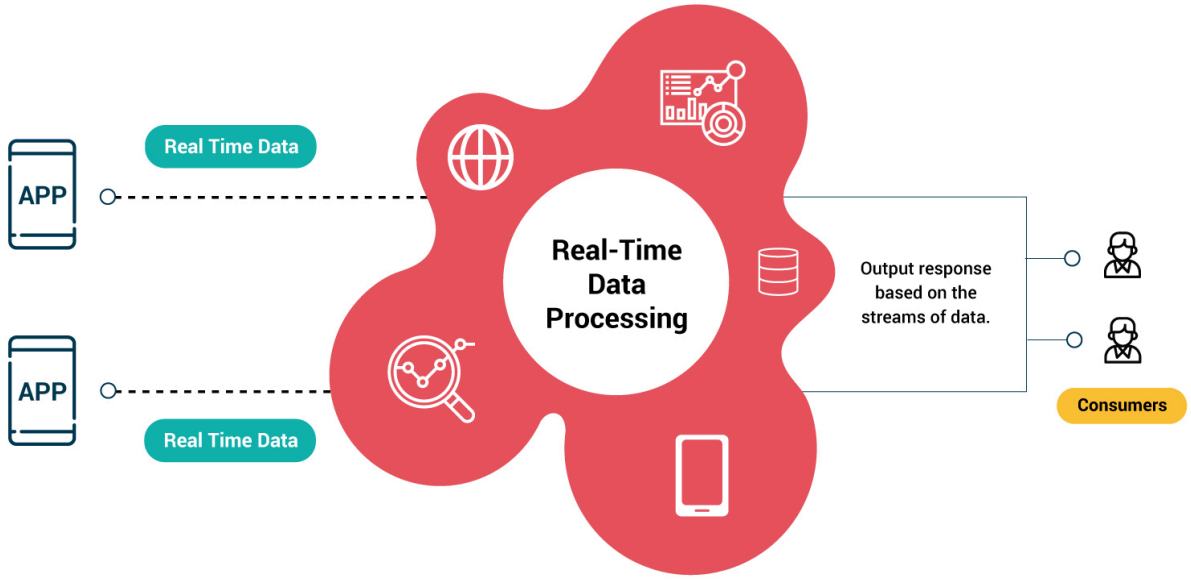
Typical Applications

- **Operational Reporting:** Supports the generation of insights from day-to-day operations.
- **Querying Large-scale Data:** Ideal for applications dealing with terabytes to exabytes of data, such as data warehousing for business intelligence.

Example

In a retail setting, Amazon Redshift could be employed to analyze large datasets for operational reporting, providing insights into sales trends and inventory management.

3. Amazon ElastiCache: Real-time Responsiveness



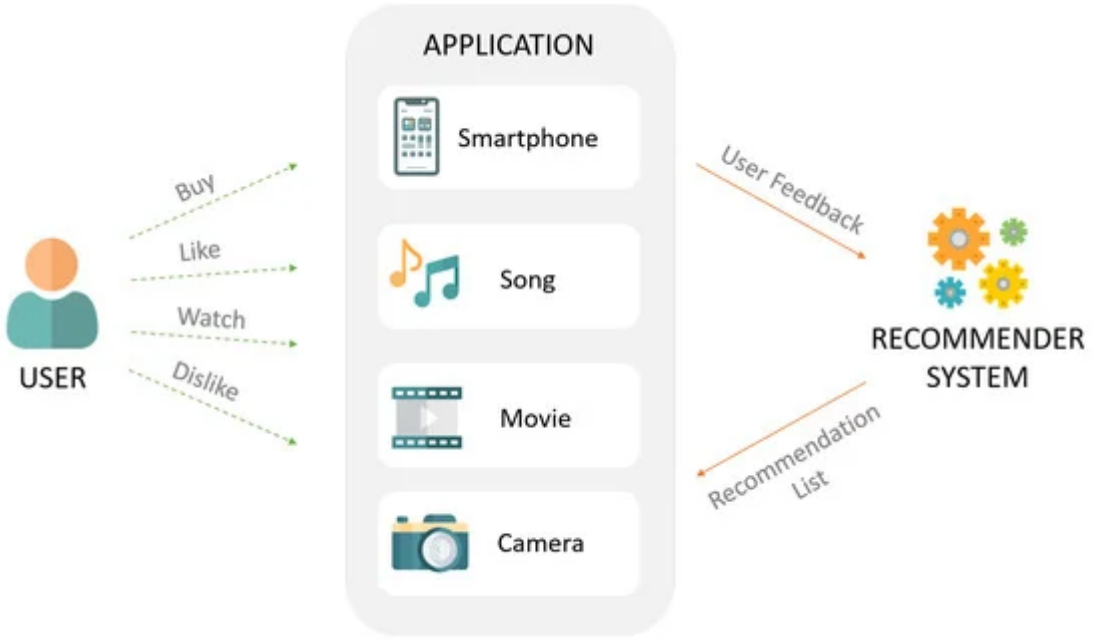
Typical Applications

- **Gaming Leaderboards:** Requires submillisecond latency for real-time updates in gaming environments.
- **Chat or Messaging Platforms:** Demands instantaneous communication for real-time messaging.
- **Streaming Services and IoT:** Supports real-time data processing for applications like streaming services and the Internet of Things (IoT).

Example

In a gaming application, ElastiCache could be utilized to store and quickly retrieve leaderboard data for real-time updates during gameplay.

4. Amazon Neptune: Navigating Connected Data



Typical Applications

- **Social News Feeds:** Navigates highly connected data for timely updates in social platforms.
- **Recommendation Systems:** Analyzes relationships for personalized suggestions.
- **Fraud Detection:** Navigates complex networks of data to identify anomalies.

Example

In a social media platform, Amazon Neptune could be employed to analyze the connections between users, ensuring relevant content appears in their news feeds.

5. Amazon DynamoDB: Internet-Scale Dynamo

Typical Applications

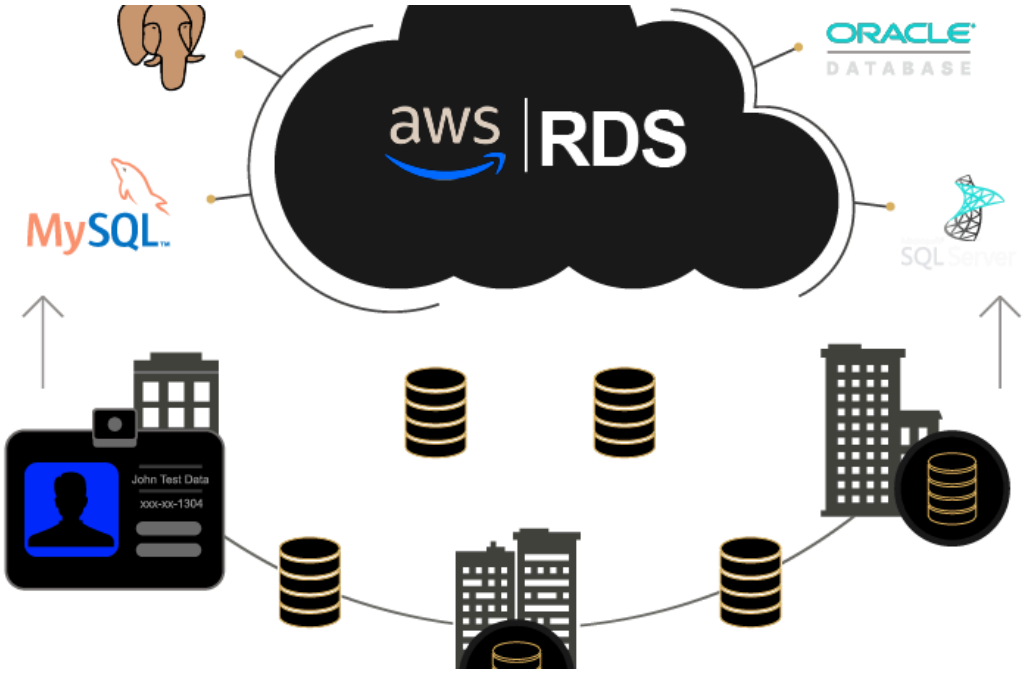
- **Hospitality Apps:** Efficiently serves content in hotel and travel applications.
- **Dating Platforms:** Handles large-scale data interactions for matchmaking.
- **Ridesharing Services:** Ensures responsive and scalable content delivery for dynamic, location-based services.

Example

In a ridesharing app, DynamoDB could store and quickly retrieve user and ride data, ensuring real-time updates and efficient matchmaking.

These examples provide a comprehensive understanding of how each AWS database can be applied in real-world scenarios, showcasing their versatility and efficiency in various domains.

Chapter 19: Amazon Relational Database Service (RDS)



Chapter overview

You'll learn to:

• discover one type of data services, Amazon Relational Database Service (RDS)

What is Amazon RDS?

aws re/start

Amazon Relational Database Service (Amazon RDS) is a powerful and versatile managed service provided by Amazon Web Services (AWS). It simplifies the process of setting up, operating, and scaling relational databases in the cloud. In this chapter, we'll explore the key features and benefits of Amazon RDS.

1. Managed Service

Amazon RDS is a fully managed service, meaning AWS takes care of many time-consuming tasks associated with traditional database management. This includes routine database tasks such as backups, software patching, and database scaling. This allows developers and administrators to focus more on optimizing performance and less on the operational intricacies of database management.

2. Relational Database in the Cloud

Amazon RDS supports popular relational database engines such as MySQL, PostgreSQL, Oracle, MariaDB, and Microsoft SQL Server. Users can choose the database engine that best suits their application requirements. Amazon RDS handles the heavy lifting associated with database provisioning, making it easier to deploy and manage databases in the cloud.

3. Setup and Configuration

Setting up a relational database with Amazon RDS is a straightforward process. Users can launch a fully functional and scalable database instance with just a few clicks through the AWS Management Console, Command Line Interface (CLI), or API. This simplicity accelerates the development and deployment of applications that rely on relational databases.

4. Operational Tasks

Amazon RDS automates routine operational tasks, such as backups, software patching, and monitoring. Automated backups are performed regularly, providing point-in-time recovery options. This managed approach enhances data durability and minimizes the risk of data loss.

5. Scalability

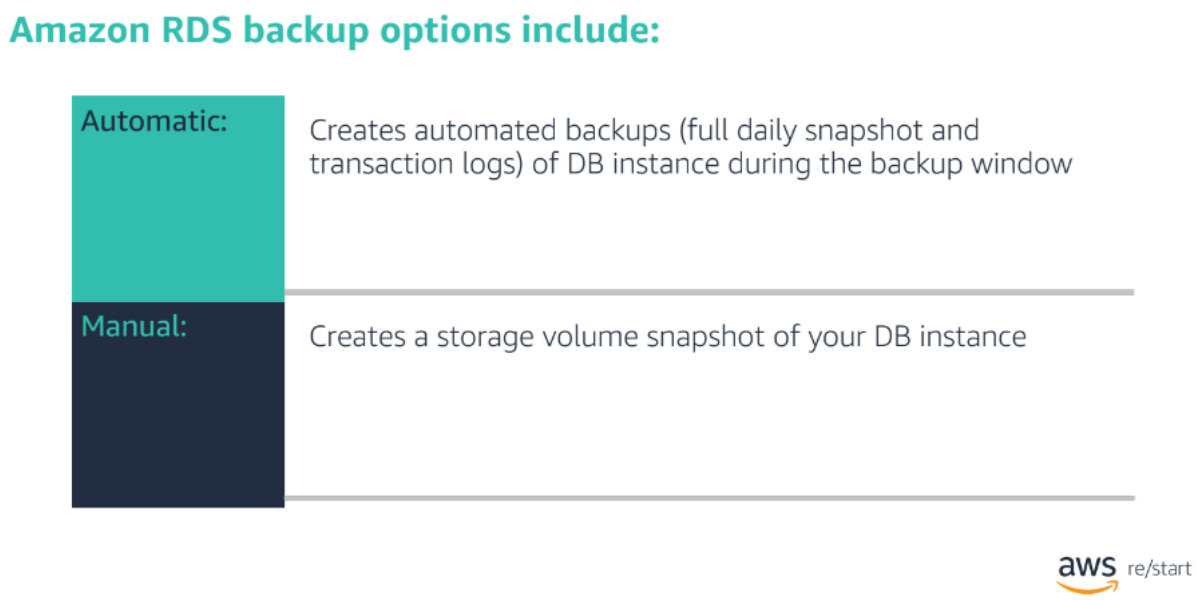
Scaling databases to accommodate changing workloads is seamless with Amazon RDS. Users can easily scale compute and storage resources up or down based on application demands. This flexibility ensures optimal performance and cost efficiency, allowing organizations to adapt to evolving requirements.

6. Security

Amazon RDS prioritizes security, providing features such as encryption at rest and in transit, automated software patching for security updates, and integration with AWS Identity and

Access Management (IAM). These security measures help users meet compliance standards and safeguard sensitive data.

Amazon RDS backup



1. Automatic Backups

Amazon RDS provides an automatic backup feature that takes the complexity out of the backup process. During the defined backup window, Amazon RDS automatically creates full daily snapshots of your DB instance. These snapshots capture the entire database, allowing for point-in-time recovery. Additionally, transaction logs are continuously backed up, providing the ability to restore your database to any specific second within the retention period.

Key Features of Automatic Backups:

- **Full Daily Snapshot:** Amazon RDS generates a complete snapshot of the DB instance every day. This snapshot serves as a baseline for recovery.
- **Transaction Log Backups:** Continuous transaction log backups enable point-in-time recovery, allowing you to restore your database to a specific moment within the retention period.

- **Backup Window:** Administrators can define a backup window during off-peak hours to minimize the impact on database performance.

2. Manual Backups

In addition to automatic backups, Amazon RDS allows users to create manual backups by taking storage volume snapshots of their DB instances. While automatic backups are crucial for routine recovery scenarios, manual backups provide an extra layer of control and flexibility.

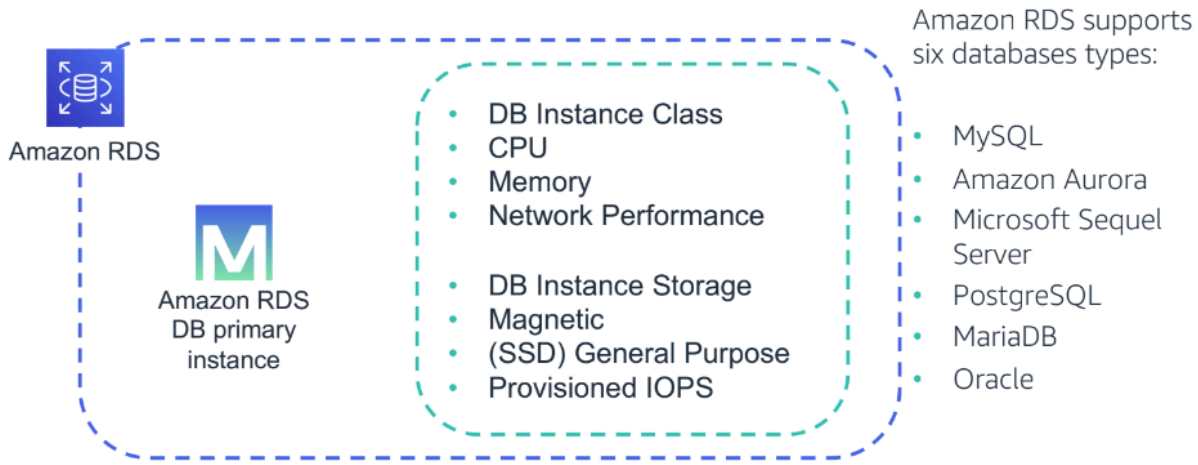
Key Features of Manual Backups:

- **Storage Volume Snapshot:** Manual backups involve capturing a snapshot of the storage volume associated with your DB instance. This snapshot represents a point-in-time image of the database.
- **User-Initiated:** Administrators can trigger manual backups at any time, independent of the defined automatic backup window.
- **Customization:** Users can choose specific times to create manual backups, aligning with maintenance schedules or critical updates.

3. Best Practices

- **Combine Automatic and Manual Backups:** Leverage both automatic and manual backup strategies for comprehensive data protection. Automatic backups provide continuous protection, while manual backups offer additional control for specific scenarios.
- **Regularly Test Restores:** Periodically test the restoration process using both automatic and manual backups to ensure the recoverability of your database.
- **Define Appropriate Retention Periods:** Set retention periods for automatic backups based on your business requirements. Consider factors like compliance, recovery point objectives, and storage costs.

Amazon RDS DB Instances



DB Instance Class

The DB instance class determines the computing and memory capacity of an Amazon RDS instance. Instances come in various classes, ranging from small instances suitable for lightweight applications to larger, more powerful instances for high-performance workloads. Selecting the right instance class is crucial for optimizing performance and cost.

CPU

The CPU (Central Processing Unit) of an Amazon RDS DB instance determines its processing power. Different instance classes offer varying levels of CPU capacity to meet the demands of specific workloads. Users can choose an instance with an appropriate CPU configuration based on the performance requirements of their applications.

Memory

The amount of memory available in an Amazon RDS DB instance is a critical factor in determining its ability to handle concurrent database transactions and queries efficiently. Instance classes with larger memory capacity are better suited for memory-intensive applications.

Network Performance

Amazon RDS instances come with different levels of network performance. This includes the amount of network bandwidth available for data transfer. Understanding and selecting the right network performance level is essential, especially for applications with high data transfer requirements.

DB Instance Storage

Amazon RDS offers different types of storage for DB instances, catering to diverse performance needs. The two main types are Magnetic storage, suitable for workloads with moderate I/O requirements, and (SSD) General Purpose storage, offering higher I/O performance for a wide range of workloads.

Provisioned IOPS

For workloads with demanding I/O requirements, Amazon RDS provides Provisioned IOPS (Input/Output Operations Per Second). This feature allows users to specify the level of I/O performance they need, ensuring consistent and predictable database performance for applications that rely heavily on disk I/O.

Amazon RDS Supports Six Database Types:

MySQL: An open-source relational database management system.

Amazon Aurora: A MySQL and PostgreSQL-compatible relational database built for the cloud.

Microsoft SQL Server: A widely used relational database management system developed by Microsoft.

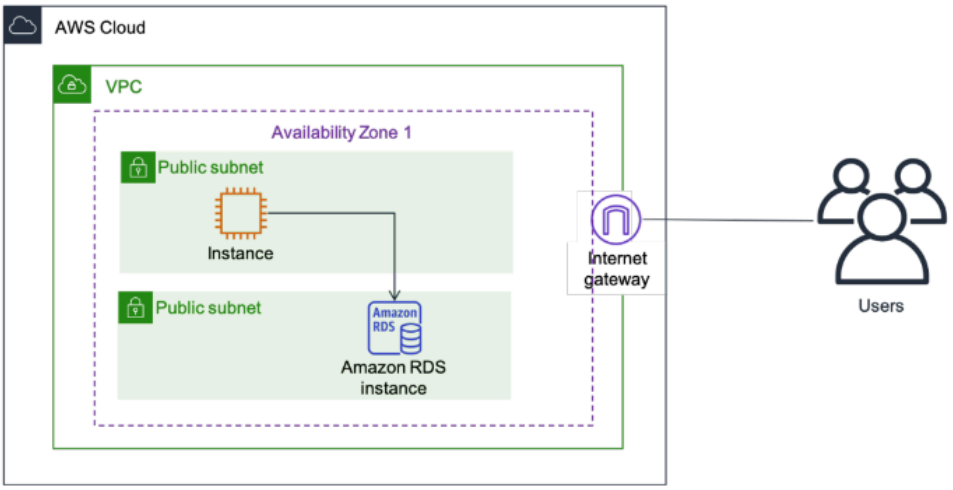
PostgreSQL: An open-source object-relational database system.

MariaDB: A fork of MySQL, designed for improved performance and ease of use.

Oracle: A comprehensive and widely used relational database management system.

Amazon RDS DB instances provide a flexible and scalable solution for managing relational databases in the AWS cloud. Understanding the various components such as instance class, CPU, memory, network performance, and storage options is crucial for optimizing the performance of your databases. With support for six different database types, Amazon RDS caters to a wide range of application requirements, making it a versatile choice for businesses seeking a managed database solution in the cloud.

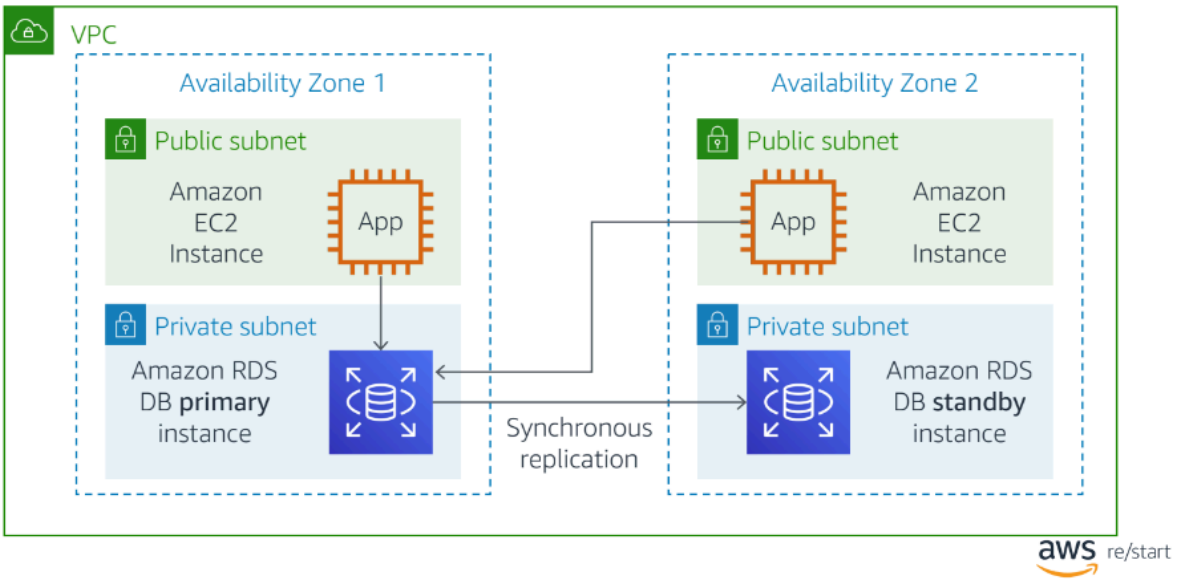
Amazon RDS In a Virtual Private Cloud



aws re/start

Amazon RDS in a Virtual Private Cloud combines the flexibility of a fully managed relational database service with the security and customization capabilities of AWS VPC. This integration empowers organizations to build scalable, secure, and resilient database architectures, meeting the demands of modern applications while adhering to best practices in cloud infrastructure management.

High availability with multi-AZ deployment: Replication



High availability is a critical consideration for databases, and AWS offers a Multi-AZ (Availability Zone) deployment option for Amazon RDS to enhance resilience. We will explore the concept of high availability with a focus on replication in Amazon RDS Multi-AZ deployments.

Multi-AZ Deployment Overview

Multi-AZ deployment is a feature of Amazon RDS designed to enhance database availability and durability. In a Multi-AZ setup, your primary database is synchronously replicated to a standby instance located in a different Availability Zone. If the primary instance fails, Amazon RDS automatically fails over to the standby instance, minimizing downtime and ensuring continued database operations.

Replication in Multi-AZ Deployment

Replication is a fundamental aspect of Multi-AZ deployment. The primary database instance replicates its data to a secondary, standby instance in real-time. This replication ensures that the standby instance is an exact copy of the primary, providing data redundancy and enabling seamless failover in the event of a primary instance failure.

Synchronous Replication

One key characteristic of Multi-AZ deployment is synchronous replication. Changes made to the primary database are immediately replicated to the standby instance. This ensures that the standby is always up-to-date, and in the event of a failover, there is minimal data loss. Synchronous replication is crucial for maintaining data integrity and consistency across both instances.

Automatic Failover

In the event of a failure of the primary database instance, Amazon RDS automatically initiates a failover to the standby instance. This process is transparent to the application, and the DNS endpoint is automatically updated to point to the standby instance. Automatic failover minimizes the impact of outages, providing a resilient and highly available database solution.

Read Replicas for Scalability

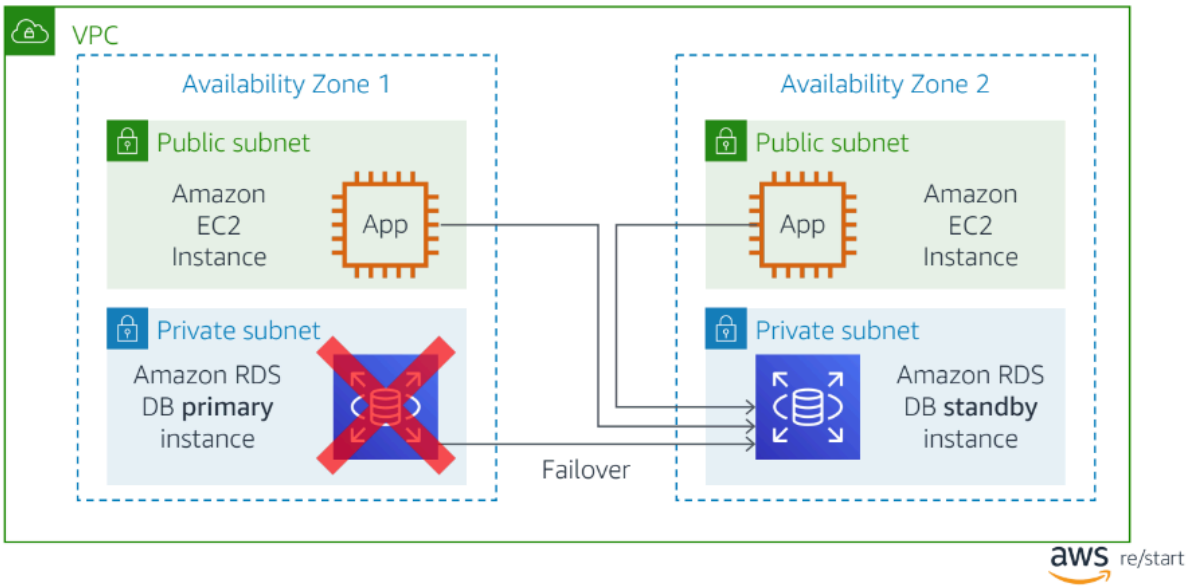
In addition to Multi-AZ deployment, Amazon RDS allows for the creation of read replicas. Read replicas are separate database instances that replicate data from the primary database asynchronously. While not part of the automatic failover process, read replicas can be leveraged for read scalability, distributing read traffic across multiple instances.

Best Practices

- Regularly monitor the health of your RDS instances and set up CloudWatch alarms for automated alerts.
- Test failover procedures periodically to ensure a smooth transition in case of an actual failure.
- Consider implementing read replicas to offload read traffic and enhance overall database performance.

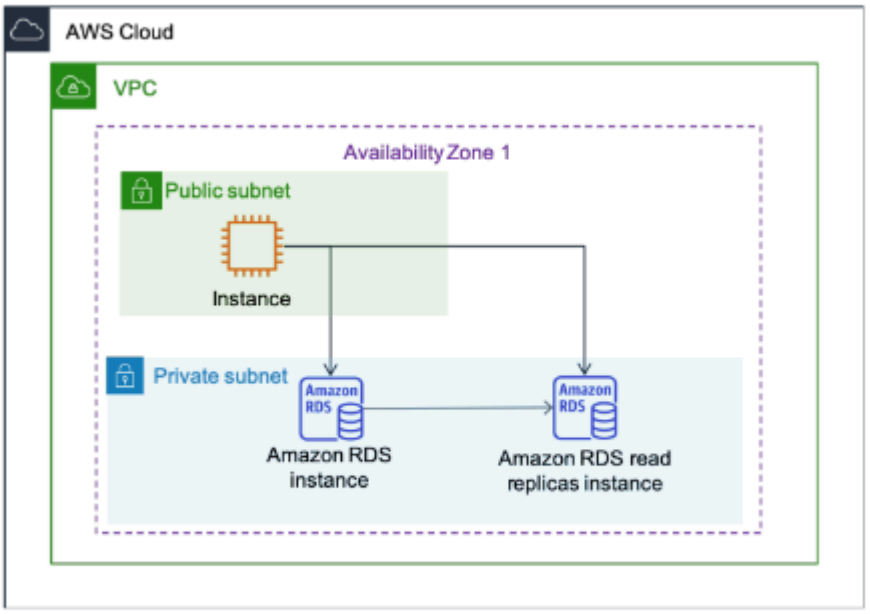
Amazon RDS Multi-AZ deployment with replication is a robust solution for achieving high availability and data durability in the AWS cloud. By understanding the principles of synchronous replication, automatic failover, and the role of read replicas, organizations can design resilient database architectures that meet their availability and scalability requirements. Implementing best practices ensures a well-managed and highly available Amazon RDS environment for critical applications and services.

High availability with multi-AZ deployment: Failover



Automatic failover is a key component of Multi-AZ deployment. If the primary database instance becomes unavailable due to hardware failure, software failure, or maintenance activities, Amazon RDS automatically promotes the standby instance to the new primary instance. This process is seamless and typically takes only a few minutes, reducing the impact on applications relying on the database.

Amazon RDS Read Replicas and Scaling



aws re/start

One powerful feature of Amazon RDS is the ability to use Read Replicas for scaling read-intensive workloads.

1. Understanding Amazon RDS Read Replicas

Read Replicas in Amazon RDS are copies of the primary database instance that can be used to offload read traffic from the primary instance. These replicas are read-only, meaning they are not intended for write operations. By distributing read queries across multiple replicas, organizations can enhance performance and scale their database infrastructure to handle a larger number of read requests.

2. Benefits of Using Read Replicas for Scaling

- **Improved Read Performance:** Read Replicas distribute read traffic, reducing the load on the primary instance and improving overall read performance. This is particularly beneficial for applications with heavy read workloads.

- **Scalability without Impacting the Primary Instance:** Read Replicas allow for horizontal scaling without affecting the performance of the primary database. This

means that as read demands increase, additional Read Replicas can be added to handle the load.

- **High Availability:** Read Replicas contribute to high availability by providing a failover option. In the event of a primary instance failure, one of the replicas can be promoted to become the new primary instance.

3. Configuring and Managing Read Replicas

- **Creating Read Replicas:** Creating Read Replicas in Amazon RDS is a straightforward process. Users can specify the number of replicas needed and configure them to be in the same or different availability zones for enhanced fault tolerance.

- **Monitoring and Scaling:** AWS provides monitoring tools that allow users to track the performance of both the primary instance and Read Replicas. Based on performance metrics, organizations can make informed decisions about scaling by adding or removing replicas as needed.

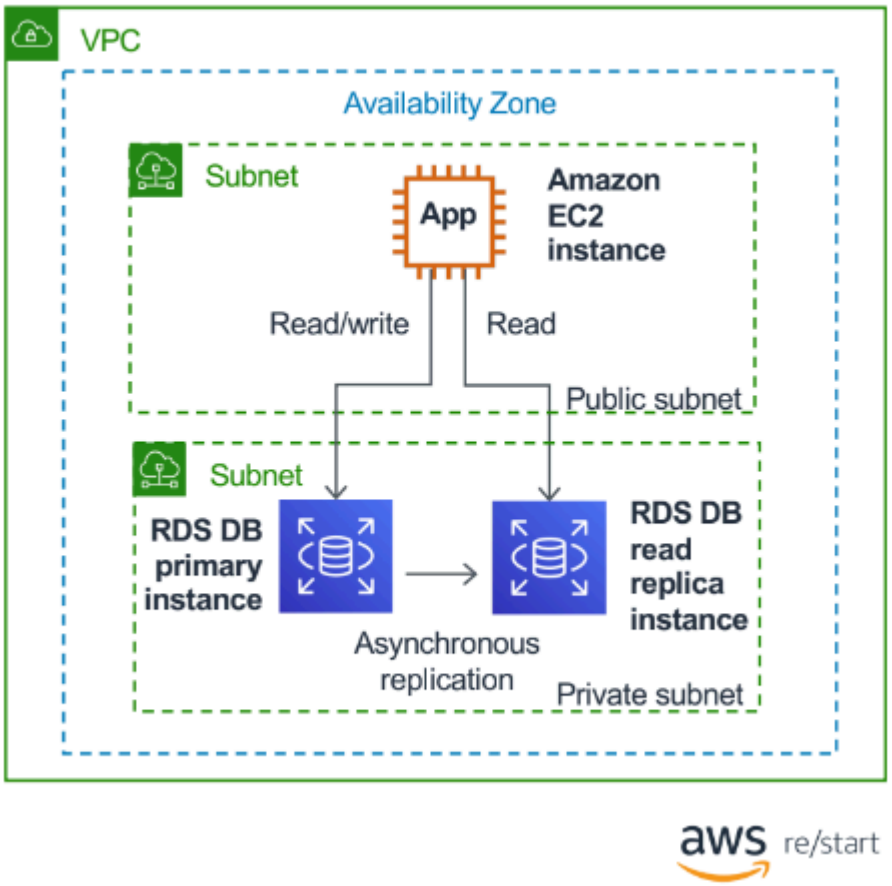
4. Considerations and Best Practices

- **Read Consistency:** Read Replicas operate asynchronously, and there might be a slight lag between the primary instance and the replicas. Applications should be designed to handle eventual consistency.

- **Security and Access Control:** Access control policies should be carefully managed to ensure that Read Replicas are appropriately secured. IAM roles and security groups can be used to control access.

- **Choosing the Right Instance Types:** Selecting the appropriate instance types for both the primary instance and Read Replicas is essential for optimizing performance and cost.

Amazon RDS scaling



Scaling your database is a critical aspect of managing growing workloads, and Amazon RDS offers a range of options to scale your database seamlessly.

Vertical Scaling (Scaling Up)

Vertical scaling, also known as scaling up, involves increasing the compute and memory capacity of your Amazon RDS instance. With Amazon RDS, you can easily modify the instance class to a higher-tier instance to handle increased workloads. This is a straightforward way to boost performance by providing more CPU and memory resources to your database without changing the underlying architecture.

Horizontal Scaling (Scaling Out)

Amazon RDS supports horizontal scaling, also known as scaling out, through the use of read replicas. Read replicas are copies of the primary database instance that can handle read-only queries, distributing the read workload and improving overall performance. By creating read replicas, you can scale horizontally to accommodate higher read traffic without impacting the primary database's write operations.

Multi-AZ Deployments for High Availability

Scaling in terms of availability is crucial for maintaining a resilient database infrastructure. Amazon RDS offers Multi-AZ (Availability Zone) deployments, where a standby replica is automatically created in a different Availability Zone. In the event of a failure, Amazon RDS can automatically failover to the standby replica, minimizing downtime and providing enhanced availability.

Amazon Aurora: Serverless and Global Databases

For those seeking a serverless approach, Amazon Aurora Serverless allows your database to automatically start and stop based on actual demand. This is a cost-effective way to handle intermittent or unpredictable workloads. Additionally, Amazon Aurora Global Databases enable you to deploy Aurora replicas in multiple regions globally, providing low-latency access to users around the world.

Performance Insights and Monitoring

Amazon RDS provides Performance Insights, a feature that helps you identify and analyze performance bottlenecks. With detailed metrics and visualizations, you can gain insights into database activity, making informed decisions about scaling based on actual performance data. Monitoring tools like Amazon CloudWatch further enhance your ability to track and manage database resources effectively.

Best Practices

- Regularly monitor database performance using tools like Performance Insights and CloudWatch.
- Consider vertical scaling for increased CPU and memory resources and horizontal scaling with read replicas for improved read performance.
- Implement Multi-AZ deployments for high availability and reliability.

Use Cases

We will explore the versatile use cases of Amazon RDS and its key features that cater to the diverse needs of web and mobile applications, e-commerce platforms, and mobile and online games.

1. Web and Mobile Applications

- **High Throughput:** Amazon RDS is well-suited for web and mobile applications that require high throughput. It efficiently handles a large number of concurrent transactions, ensuring optimal performance for applications with heavy user traffic.
- **Massive Storage Scalability:** As data requirements grow, RDS allows seamless scaling of storage capacity. This ensures that web and mobile applications can accommodate expanding datasets without compromising on performance or responsiveness.
- **High Availability:** RDS offers high availability features, including automatic backups and failover support. This ensures that web and mobile applications enjoy reliable and continuous access to the database, minimizing downtime and enhancing user experience.

2. E-commerce Applications

- **Low-Cost Database:** Cost efficiency is crucial for e-commerce applications. RDS provides a cost-effective solution by offloading database management tasks, allowing

businesses to focus on delivering a seamless shopping experience without the overhead of database maintenance.

- **Data Security:** E-commerce applications handle sensitive customer information. RDS addresses this concern by providing robust security features, including encryption at rest and in transit, ensuring the confidentiality and integrity of data.

- **Fully Managed Solution:** RDS automates routine database administration tasks such as backups, patch management, and database tuning. This fully managed approach allows e-commerce businesses to allocate resources strategically and concentrate on business growth.

3. Mobile and Online Games

- **Rapidly Grow Capacity:** Games often experience sudden spikes in user activity. RDS enables game developers to rapidly scale database capacity to handle increased player loads, ensuring a smooth gaming experience during peak times.

- **Automatic Scaling:** With RDS, database scaling becomes an automated process. The system can dynamically adjust resources based on demand, providing flexibility for mobile and online games with varying usage patterns.

- **Database Monitoring:** RDS offers comprehensive monitoring tools that allow game developers to track database performance in real-time. This proactive monitoring helps identify and address issues before they impact the gaming experience.

When to Use Amazon RDS

When considering whether to use Amazon RDS, various factors come into play, and understanding the specific use cases can help you make informed decisions. We will explore scenarios when Amazon RDS is an ideal choice based on different requirements.

Complex Transactions or Complex Queries

Amazon RDS is well-suited for scenarios involving complex transactions or queries typical of relational databases. If your application requires the use of JOIN operations, ACID compliance, and other relational database features, RDS provides the necessary support.

Medium to High Query/Write Rate (Up to 30K IOPS)

For workloads with a moderate to high query/write rate, Amazon RDS is a suitable choice, supporting up to 30,000 I/O operations per second (15,000 reads and 15,000 writes). This makes it effective for applications with demanding transactional requirements.

No More Than a Single Worker Node/Shard

Amazon RDS is designed for use cases where a single worker node or shard is sufficient. It is particularly well-suited for scenarios that do not require horizontal scaling and can operate effectively within the constraints of a single instance.

High Durability

Amazon RDS provides high durability, ensuring that your data is consistently backed up and available. This is crucial for applications where data integrity and durability are paramount, such as in financial or healthcare systems.

Massive Read/Write Rates (e.g., 150K Write/Second)

In scenarios demanding extremely high read and write rates, other database solutions may be more appropriate than Amazon RDS. RDS is optimal for workloads within the specified IOPS limits, making it an efficient choice for applications with substantial but not extreme throughput requirements.

Sharding Due to High Data Size or Throughput Demands

When your data size or throughput demands surpass the capabilities of a single RDS instance, and horizontal scaling becomes necessary, alternative solutions or database architectures may be more suitable. Consider sharding or other scaling strategies in such cases.

Simple GET/PUT Requests and Queries for NoSQL Databases

If your application predominantly involves simple GET/PUT requests and queries that align with the capabilities of NoSQL databases, Amazon RDS may not be the optimal choice. NoSQL databases might better suit scenarios with less structured data and more flexible query patterns.

Relational Database Management System (RDBMS) Customization

Amazon RDS allows some level of customization for RDBMS engines. If your application requires specific optimizations or configurations for the underlying database engine (e.g., MySQL, PostgreSQL, SQL Server), Amazon RDS provides the necessary flexibility.

Clock Hours and Service Time

As users interact with RDS instances, it's essential to grasp the concepts of RDS Clock Hours and Service Time to optimize usage, performance, and costs.

RDS Clock Hours

RDS Clock Hours refer to the total time an Amazon RDS instance is operational within a billing period. It includes both the time the instance is actively running and the time it remains in a stopped state. Clock Hours play a crucial role in determining the cost associated with an RDS instance, as AWS bills based on the total hours an instance is available, regardless of whether it is actively processing requests or not.

Understanding and monitoring RDS Clock Hours is vital for cost management. Users should be strategic in starting and stopping instances based on actual usage patterns to optimize expenses and prevent unnecessary charges during periods of inactivity.

Service Time

Service Time in the context of RDS refers to the time an instance spends actively processing requests and serving database operations. Unlike Clock Hours, which account for the total operational time, Service Time focuses specifically on the periods when the database is actively engaged in handling queries, inserts, updates, and other transactions.

Monitoring Service Time is crucial for assessing database performance and efficiency. A low Service Time suggests that the database is quickly processing requests, while prolonged

Service Time may indicate performance bottlenecks that need attention. Optimizing Service Time is essential for ensuring responsive and efficient database operations.

Best Practices

- **Optimize RDS Instance Utilization:** Periodically review and adjust RDS instance configurations based on actual usage patterns to ensure optimal resource utilization.
- **Implement Automated Start/Stop:** Leverage AWS tools, such as AWS Lambda and CloudWatch Events, to automate the start and stop of RDS instances during specified periods of inactivity, minimizing unnecessary Clock Hours.
- **Monitor and Analyze Service Time:** Regularly monitor Service Time metrics to identify performance bottlenecks, optimize queries, and enhance the overall efficiency of your RDS instances.
- **Leverage RDS Performance Insights:** Utilize AWS RDS Performance Insights to gain deeper insights into database performance, query execution times, and resource utilization.

DB Purchase Type and Multiple DB Instances

Choosing the right database solution is a critical decision for organizations seeking efficient data management. We will delve into the concept of database purchase types and the advantages of deploying multiple database instances. Understanding these aspects is essential for optimizing database performance, scalability, and cost-effectiveness.

Database Purchase Types

In the realm of databases, organizations have the option to choose between different purchase types, primarily focusing on two models: On-Demand and Reserved Instances.

- **On-Demand Instances:** Offer flexibility by allowing users to pay for database capacity on an hourly or per-second basis, without any upfront costs or long-term commitments. This model is ideal for unpredictable workloads or short-term projects.
- **Reserved Instances:** Involve a commitment to a one- or three-year term, providing significant cost savings compared to On-Demand pricing. Reserved Instances are suitable for workloads with predictable and steady usage patterns.

Understanding the nature of your organization's workloads and requirements is crucial for selecting the most cost-effective database purchase type.

Multiple Database Instances

Deploying multiple database instances refers to the practice of running several independent database environments within an organization. This approach offers several advantages:

- **Isolation of Workloads:** Multiple instances allow organizations to isolate different workloads, preventing resource contention and ensuring optimal performance for each application or service.

- **Scalability:** Each instance can be scaled independently, providing the flexibility to allocate resources based on the specific needs of different applications. This ensures that resources are allocated efficiently, avoiding over-provisioning and unnecessary costs.

- **High Availability:** Distributing databases across multiple instances enhances availability. In the event of a failure in one instance, others can continue to operate, minimizing downtime and ensuring continuous service availability.

- **Security and Compliance:** Multiple instances provide an additional layer of security by isolating sensitive data or applications. This is particularly important for organizations with strict compliance requirements.

Best Practices

- **Assess Workload Characteristics:** Evaluate the nature of your workloads to determine the most suitable database purchase type – On-Demand or Reserved Instances.

- **Plan for Growth:** Design database architectures with scalability in mind. Multiple instances can accommodate future growth by allowing organizations to scale horizontally.

- **Implement High Availability:** Ensure that critical databases are distributed across multiple instances to enhance fault tolerance and maintain high availability.

- **Regularly Review and Optimize:** Periodically review database usage patterns and adjust the number and type of instances to optimize costs and performance.

Amazon RDS: Storage

1. Storage Types

Amazon RDS offers various storage types to cater to different performance and cost requirements:

- **General Purpose (SSD):** Suitable for a broad range of workloads, this storage type delivers balanced performance and cost-effectiveness.
- **Provisioned IOPS (SSD):** Ideal for I/O-intensive database workloads, providing consistent and predictable performance through provisioned I/O operations per second (IOPS).
- **Magnetic:** A cost-effective option for workloads with moderate I/O requirements, such as development and testing environments.
- **IO1 (Provisioned IOPS):** Offers high-performance storage for I/O-intensive workloads, allowing users to provision a specific amount of IOPS.

2. Automatic Backups

Amazon RDS automatically backs up your database and transaction logs. The backups are retained for a specified retention period, providing point-in-time recovery options. This feature ensures data durability and facilitates easy recovery in case of accidental data loss or corruption.

3. Snapshots

Users can create manual snapshots of their databases. These snapshots serve as backups that can be restored at any time. Snapshot management allows users to control costs by retaining only the necessary backups.

4. Scaling Storage

As your database storage needs grow, Amazon RDS makes it easy to scale your storage capacity. Users can modify the storage size of their DB instances or change the storage type to adapt to changing workload requirements.

5. Multi-AZ Deployments

For enhanced durability and availability, users can opt for Multi-AZ (Availability Zone) deployments. In this setup, Amazon RDS automatically replicates the database to a standby instance in a different Availability Zone. If the primary instance fails, the system automatically fails over to the standby instance.

6. Encryption

Amazon RDS provides options for encrypting both data at rest and data in transit. This ensures that sensitive information is protected from unauthorized access, meeting security and compliance requirements.

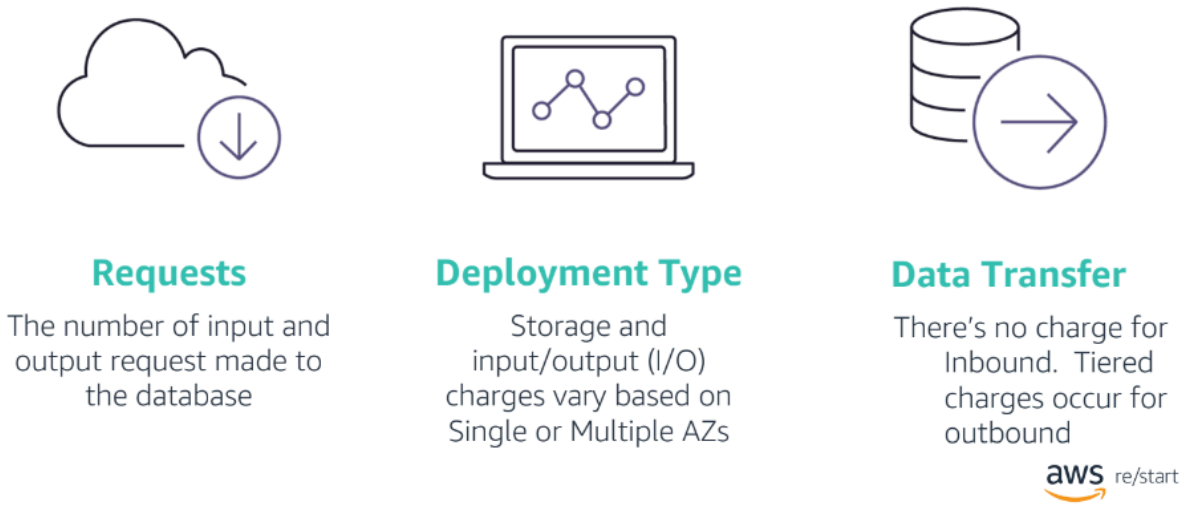
7. Performance Monitoring

Amazon RDS offers performance monitoring tools that allow users to assess the health and performance of their databases. Monitoring features include Amazon CloudWatch metrics, enhanced monitoring, and the ability to view and analyze database logs.

8. Best Practices

- Regularly monitor storage usage and performance metrics.
- Implement a backup and recovery strategy, leveraging both automatic backups and manual snapshots.
- Adjust storage size and type based on the evolving needs of your database workload.

Amazon RDS: Deployment Type and Data Transfer



When utilizing Amazon RDS, it's essential to comprehend deployment types and data transfer aspects, as they directly impact the performance, cost, and overall efficiency of your database.

Deployment Types

Amazon RDS provides flexibility in choosing the deployment type that best suits your application's needs. The deployment types primarily involve deciding whether to run your database instance in a Single-Availability Zone (Single-AZ) or Multiple-Availability Zones (Multi-AZ). This choice has implications for both the availability and durability of your database.

Single-AZ Deployment: Your database runs in a single availability zone. Suited for development and test environments or applications that do not require high availability.

Multi-AZ Deployment: Your database is replicated across multiple availability zones. Offers enhanced availability and fault tolerance. Suited for production environments where high availability is critical.

Data Transfer

1. **Input and Output Requests**

The number of input and output requests made to the database is a crucial metric for assessing performance. It directly impacts the responsiveness and scalability of your database. Monitoring and optimizing input/output operations are essential for efficient database operations.

2. **Storage and Input/Output (I/O) Charges**

Storage and I/O charges in Amazon RDS vary based on your chosen deployment type. Single-AZ deployments generally have lower costs compared to Multi-AZ deployments. Understanding your application's needs and performance requirements is key to selecting an appropriate deployment type.

3. **Data Transfer Charges**

Inbound data transfer to your Amazon RDS instance is typically free. Outbound data transfer incurs tiered charges, meaning costs increase as the volume of outbound data grows. Monitoring and optimizing outbound data transfer can help control costs.

Best Practices

1. Assess your application's availability requirements to determine the appropriate deployment type.

2. Regularly monitor input/output requests and adjust resources accordingly to maintain optimal performance.

3. Understand and optimize data transfer patterns to minimize costs.

RDS Commands

Create snapshots: Command

Use the following:

```
$ aws rds copy-db-snapshot --
source-db-snapshot-identifier
mydbsnapshot --target-db-
snapshot-identifier
mydbsnapshot-copy --copy-tags
```

The expected result is:


```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot",
        "DBInstanceIdentifier": "mytestdb",
        "Engine": "mysql",
        "AllocatedStorage": 20,
        "Status": "creating",
        "Port": 3306,
        "AvailabilityZone": "us-east-1f",
        "VpcId": "vpc-123456789012",
        "InstanceCreateTime": "2019-04-
03T17:18:34.920Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual",
        "OptionGroupName": "default:mysql-5-6",…
```


Restore DB from snapshot: Command

Use the following:

```
$ aws rds create-db-snapshot
--db-instance-identifier
mytestdb --db-snapshot-
identifier mydbsnapshot
```

The expected result is:

```
DBINSTANCE  mytestdb-new  db.t2.micro  MySQL     50
sa
creating  3  n  5.6.40  general-public-license
```

Copy RDS Snapshot: Command

Use the following:

```
$ aws rds copy-db-snapshot --
source-db-snapshot-identifier
mydbsnapshot --target-db-
snapshot-identifier
mydbsnapshot-copy --copy-tags
```

The expected result is:


```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot-copy",
        "DBInstanceIdentifier": "mytestdb",
        "Engine": "mysql",
        "AllocatedStorage": 20,
        "Status": "creating",
        "Port": 3306,
        "AvailabilityZone": "us-east-1f",
        "VpcId": "vpc-0123456789012",
        "InstanceCreateTime": "2019-04-
03T17:18:34.920Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual",
        "OptionGroupName": "default:mysql-5-6",
        "PercentProgress": 0,
        "SourceRegion": "us-east-1", …
```

Deleting a snapshot: Command

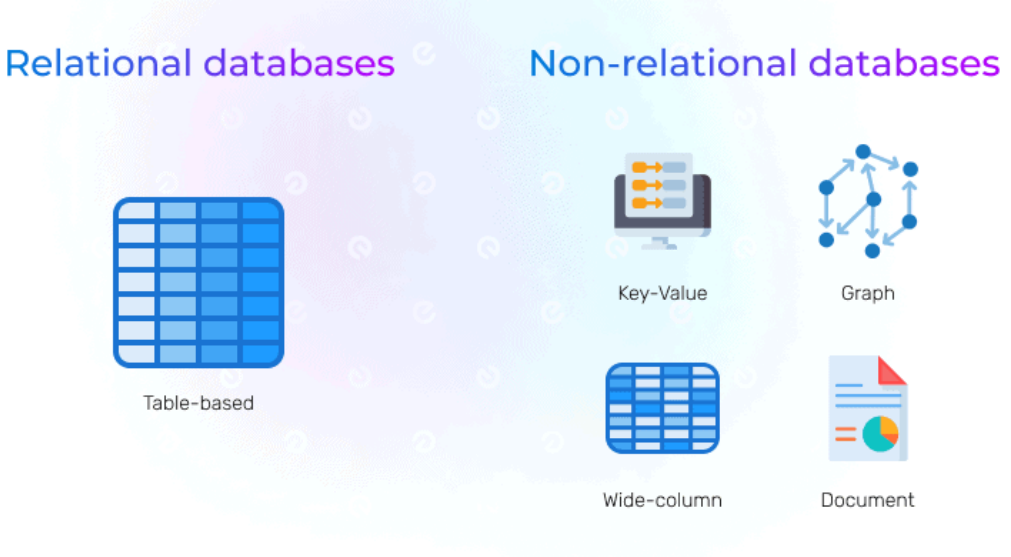
Use the following:

```
$ aws rds delete-db-snapshot
--db-snapshot-identifier
mydbsnapshot
```

The expected result is:


```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot",
        "DBInstanceIdentifier": "mytestdb",
        "SnapshotCreateTime": "2019-04-
03T21:29:21.338Z",
        "Engine": "mysql",
        "AllocatedStorage": 20,
        "Status": "deleted",
        "Port": 3306,
        "AvailabilityZone": "us-east-1f",
        "VpcId": "vpc-012345678912",
        "InstanceCreateTime": "2019-04-
03T17:18:34.920Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual", ….
```


Chapter 20: Amazon DynamoDB



Databases are essential tools used to organize, manage, and retrieve data efficiently. They come in various types, each designed to address specific data management needs. Two primary categories of databases are Relational Databases (RDB) and Non-Relational Databases (NoSQL).

Relational Databases (RDB)

A Relational Database is a structured data system that organizes information into tables, records, and columns. Think of it as a collection of spreadsheets interconnected by

relationships. These databases establish a well-defined relationship between different tables, allowing for robust data querying and management. The Structured Query Language (SQL) serves as the backbone of RDBs, providing a standardized interface for users to interact with the database. SQL enables users to perform various operations, including data insertion, retrieval, update, and deletion, making it an integral part of relational databases.

One of the defining features of RDBs is their ability to maintain data integrity and enforce relationships between different entities within the database. This structured approach facilitates efficient storage, retrieval, and manipulation of data, making them suitable for scenarios where data consistency and structured querying are crucial, such as in finance, e-commerce, and enterprise systems.

Non-Relational Databases (NoSQL)

Contrary to Relational Databases, Non-Relational Databases, commonly referred to as NoSQL databases, do not adhere strictly to the traditional relational model. These databases encompass various data models, such as document-based, key-value pairs, columnar, and graph databases, among others. NoSQL databases are designed to handle unstructured, semi-structured, or rapidly changing data formats, providing more flexibility compared to the rigid structure of RDBs.

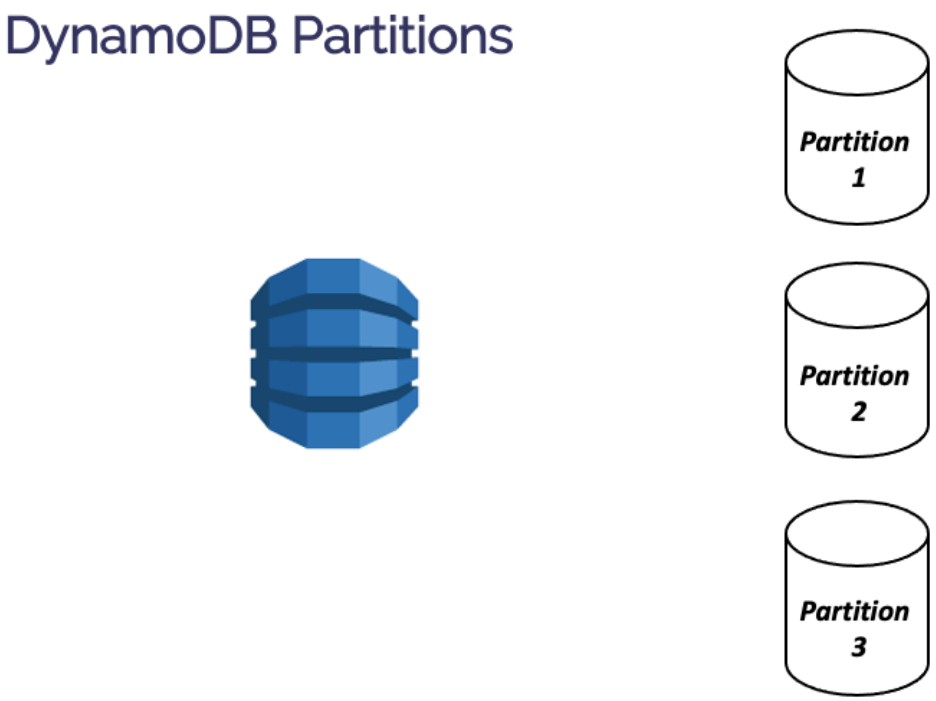
The absence of a fixed schema and the ability to scale horizontally make NoSQL databases ideal for scenarios where data volume is vast and dynamic, like in social media, IoT (Internet of Things), and real-time analytics. These databases prioritize high performance, scalability, and flexibility over the strict adherence to relational constraints, allowing for agile development and adaptation to evolving data requirements.

Feature	Non Relational Database	Relational Database
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimised for Huge Data	Medium - Large Data Size
Performance	High	Low
Reliability	Poor	Good
Scalability	High	High (but more expensive)

In summary, while Relational Databases offer structured and organized data management suited for applications requiring well-defined relationships and data integrity, Non-Relational Databases offer flexibility, scalability, and efficiency in handling diverse data formats and high volumes.

Understanding the differences between these database types helps in selecting the most suitable option based on the specific needs of a given project or application.

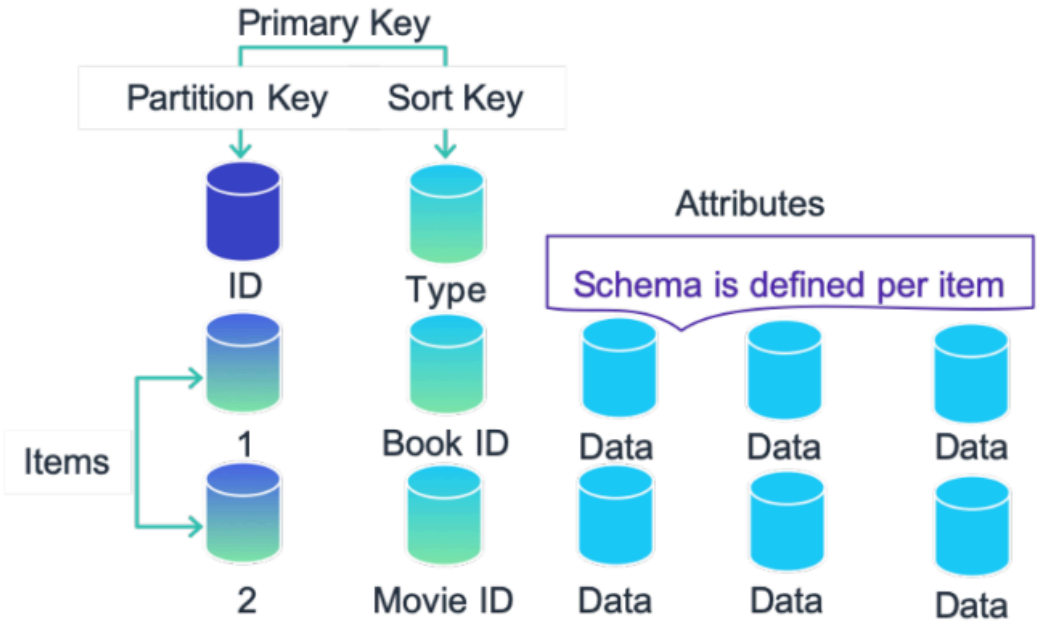
The concept of partitioning



Understanding Database Partitions

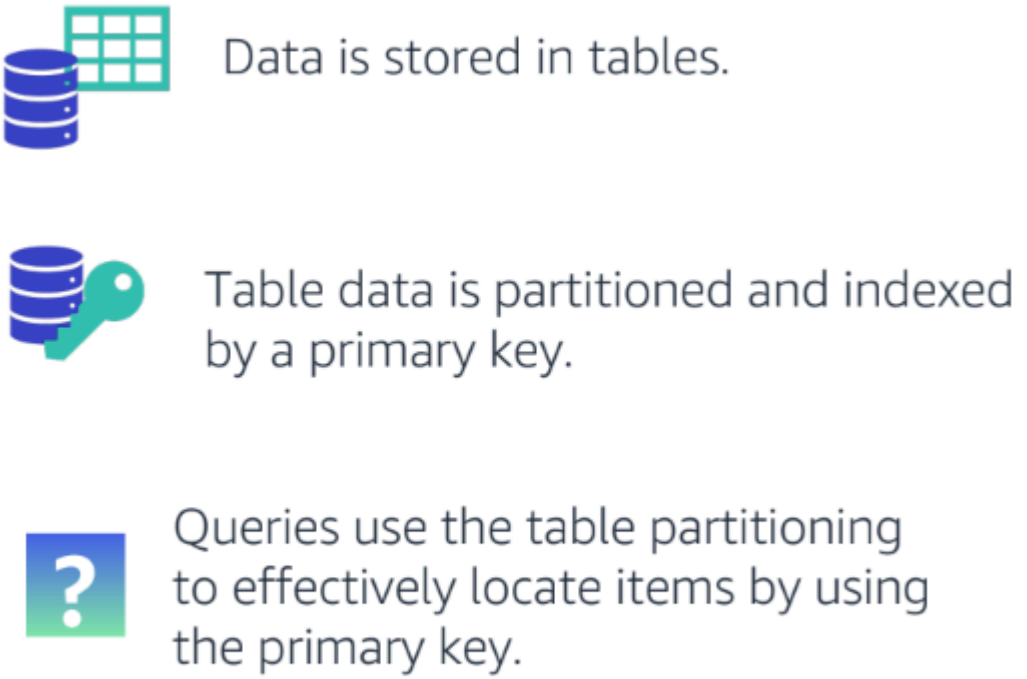
In databases, data is stored in organized structures called tables. These tables hold various types of information, and efficient management of this data is crucial for fast and effective retrieval. Partitioning is a technique used to enhance the storage and retrieval of data within these tables.

What is Partitioning?



Imagine a library with thousands of books. Instead of stacking all the books on a single shelf, they might be divided among different sections: fiction, non-fiction, reference, and so on. Similarly, in databases, partitioning involves dividing a table's data into smaller, more manageable parts based on certain criteria.

Partitioning and Indexing



When a table is partitioned, the data is organized into separate segments, and each segment can be indexed by a primary key. A primary key is a unique identifier for each record in the table. By indexing these partitions based on the primary key, databases can efficiently locate and retrieve specific items within these segments.

Optimizing Queries with Table Partitioning

Queries in a database involve searching for specific information. With table partitioning, the database system can leverage this segmentation to locate items more effectively. Instead of scanning through the entire table, the system can pinpoint the relevant partition based on the primary key provided in the query. This targeted search within a smaller subset of data significantly speeds up the retrieval process.

For instance, think of a massive phone directory organized alphabetically. If you're looking for a specific surname, you'd immediately jump to that section rather than scanning the entire directory from start to finish. Table partitioning works in a similar way, allowing databases to quickly pinpoint the section where the requested information is stored.

Benefits of Partitioning

Partitioning offers several advantages:

- **Improved Performance:** By narrowing down the search area, queries execute faster, enhancing overall database performance.
- **Ease of Maintenance:** Partitioning allows for easier management of large volumes of data, making tasks like backup, restore, and data manipulation more efficient.
- **Scalability:** It facilitates scaling by distributing data across different storage devices or servers, accommodating growing data volumes.

In conclusion, table partitioning optimizes database performance by segmenting data into smaller, more manageable sections indexed by a primary key. This technique enhances query speed and database management, offering efficiency and scalability in handling large volumes of information.

Amazon DynamoDB

Amazon DynamoDB stands as a robust NoSQL database service provided by Amazon Web Services (AWS), offering dynamic capabilities for managing vast amounts of data with exceptional speed and flexibility.

Key Features of DynamoDB

NoSQL Database Structure: Unlike traditional relational databases, DynamoDB operates with NoSQL database tables, allowing for more flexibility in handling diverse data structures and formats. This design enables DynamoDB to efficiently manage unstructured or semi-structured data.

Virtually Unlimited Storage: DynamoDB is built to handle massive volumes of data with ease. It provides virtually limitless storage capacity, ensuring that as data grows, the system can seamlessly accommodate it without compromising performance.

Flexible Attributes: Within DynamoDB, items (the individual entities stored in tables) can possess varying attributes. This means each item can have different sets of characteristics, providing versatility in data organization.

Low-Latency Queries: DynamoDB is optimized for low-latency queries, allowing for rapid retrieval of data. This swift responsiveness ensures that applications relying on DynamoDB can deliver quick and efficient responses to user requests.

Scalable Read/Write Throughput: DynamoDB offers the flexibility to adjust the read and write capacity based on demand. This scalability feature allows the system to handle fluctuations in traffic and workload effectively.

Core Components of DynamoDB

Tables, Items, and Attributes: DynamoDB revolves around these core components. Tables serve as containers for storing related items, which represent individual data entries. Each item comprises various attributes that define its characteristics.

Primary Keys: DynamoDB utilizes two primary key options for organizing data:

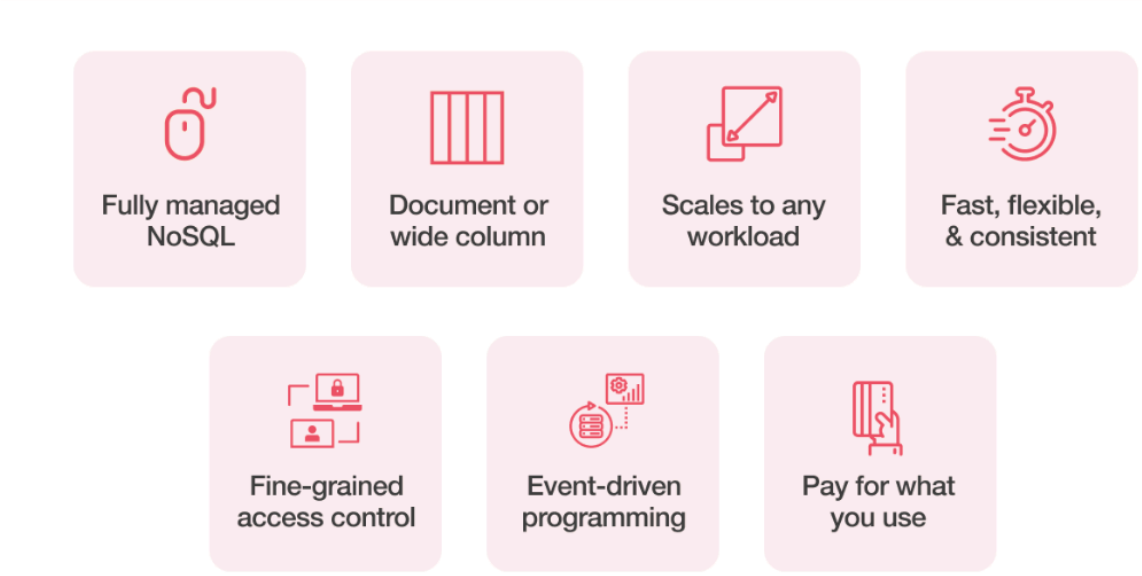
Partition Key: This primary key uniquely identifies items within a table. It partitions the data for efficient storage and retrieval, just like sorting books into different sections in a library. Partition and Sort Key: Also known as a composite key, this combination of partition and sort keys enables more precise querying and sorting of data within partitions. It's like organizing books by different categories and then arranging them in order within each category, making it easier to find a specific book quickly.

Why Primary Keys Matter?

Primary keys are like special tags that keep everything organized in DynamoDB. They ensure each item is unique, like a book having its unique library code. These keys help the database quickly find and sort data, just like a librarian finding a book using its special code in a well-organized library. This makes searches lightning fast and super organized in DynamoDB!

In summary, Amazon DynamoDB stands out as a powerful NoSQL database service offering virtually limitless storage, flexible data attributes, low-latency queries, and scalable throughput. Its core components and primary key options make it a versatile and efficient choice for managing diverse datasets in various applications.

Amazon DynamoDB technical benefits



Amazon DynamoDB is a dynamic non-relational database service renowned for its speed and scalability in handling vast amounts of data. Key Benefits of Amazon DynamoDB:

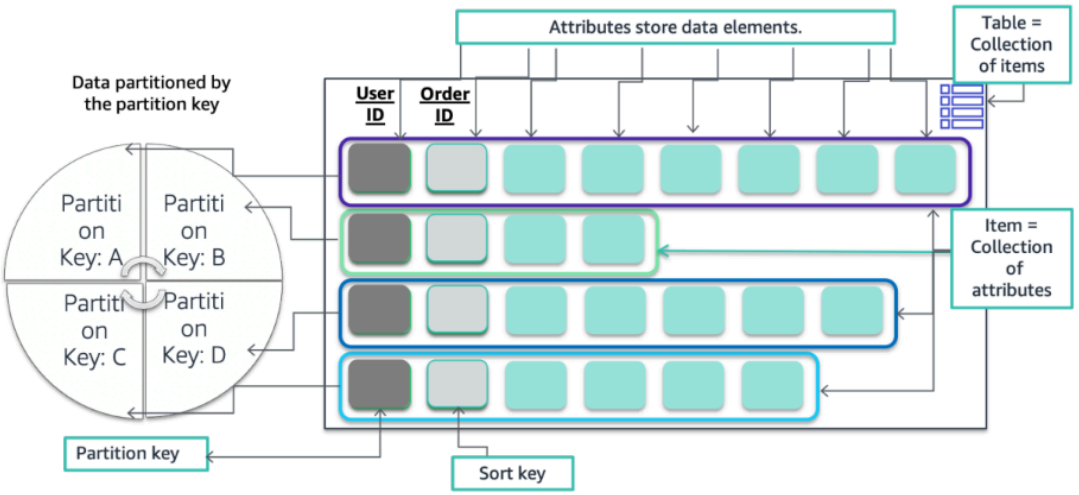
Fully Managed Service: DynamoDB provides a fully managed solution, freeing users from the complexities of infrastructure management. This allows for streamlined data operations without the burden of maintaining backend systems.

Low-Latency Queries: With DynamoDB, accessing data is lightning-fast. Its architecture is optimized for low-latency queries, ensuring swift retrieval of information, which is crucial for applications demanding quick responses.

Fine-Grained Access Control: Security is paramount, and DynamoDB excels in providing precise control over who can access specific data. This fine-grained access control allows users to set detailed permissions, ensuring data confidentiality and integrity.

Flexibility in Data Handling: DynamoDB is designed to be flexible, accommodating diverse data structures and formats. Whether dealing with structured or unstructured data, DynamoDB adapts seamlessly, offering versatility in managing various data types.

Amazon DynamoDB data model



aws re/start

Understanding the DynamoDB Data Model

Tables as Containers: In DynamoDB, data is organized into tables, acting as containers for information. Each table stores multiple items, which represent individual pieces of data, similar to rows in a traditional database.

Items and Attributes: Items within DynamoDB tables can vary in structure. Each item contains attributes, which are like data fields providing specific information. These attributes can differ between items within the same table, offering a flexible approach to data storage.

Primary Keys for Uniqueness: Every item in a DynamoDB table must have a primary key. The primary key uniquely identifies each item within the table and is made up of either a partition key or a combination of a partition key and a sort key. This uniqueness ensures efficient data retrieval and organization.

Partition Key: It's the primary identifier for data distribution across DynamoDB partitions. It helps distribute data among partitions for efficient storage and retrieval.
Sort Key: When combined with a partition key, it creates a composite key that enables sorting items within the partition, facilitating ordered retrieval of data.

Nested Data Structures: DynamoDB supports nested data structures within attributes. This means an attribute can hold complex data types like lists or maps, allowing for versatile data representation within items.

Benefits of the DynamoDB Data Model

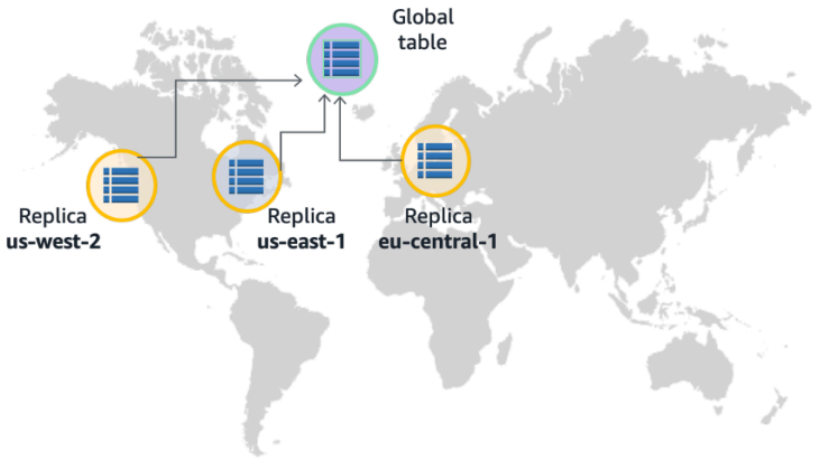
Flexibility: The flexible structure allows for diverse data representations within tables, accommodating various data types and structures.

Efficiency: The partitioning strategy based on primary keys ensures efficient data storage, retrieval, and scalability, making DynamoDB an excellent choice for high-performance applications.

Scalability: With a distributed architecture based on partition keys, DynamoDB seamlessly scales to accommodate growing data volumes and traffic demands.

In summary, the DynamoDB data model revolves around tables, items, and attributes, utilizing primary keys for uniqueness and enabling flexibility and scalability in data storage and retrieval.

Amazon DynamoDB global tables



aws re/start

Understanding DynamoDB Global Tables

- **Multi-Region Replication:** Global Tables enable the automatic replication of DynamoDB tables across multiple AWS regions worldwide. This replication ensures that data remains available and consistent across different geographic locations, minimizing latency for users accessing data from various regions.

- **Synchronous Replication:** Changes made to a Global Table in one region are replicated near-instantaneously to all other regions where the table exists. This synchronous replication ensures that data across regions is always up-to-date and consistent.

- **Resilience and High Availability:** With Global Tables, applications gain resilience against regional outages or disruptions. Even if one region experiences issues, users can seamlessly access data from other regions where the replicated table exists, ensuring high availability.

- **Scalability and Performance:** Global Tables maintain DynamoDB's scalability and performance benefits, allowing applications to handle growing workloads and maintain responsiveness regardless of the geographic distribution of users.

Benefits of DynamoDB Global Tables

- **Global Reach:** Applications can cater to users worldwide while maintaining data consistency and low-latency access across regions.

- **Disaster Recovery:** The multi-region replication capability ensures data availability in case of region-specific failures or disasters, contributing to disaster recovery strategies.

- **Simplified Development:** Developers can build global applications without worrying about the complexity of data replication and consistency across multiple regions, streamlining the development process.

In summary, Amazon DynamoDB offers managed simplicity, consistent speed with single-digit millisecond latency, limitless scaling without constraints, and simplified global replication through Global Tables.it's a managed, high-speed, and scalable database service ideal for modern applications.

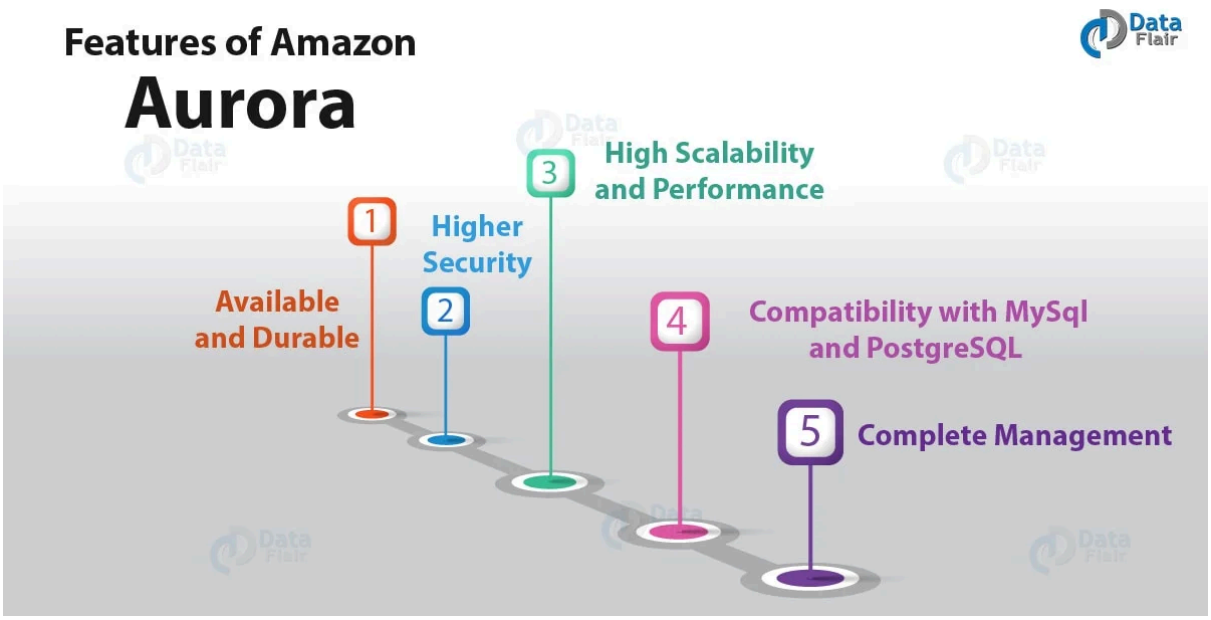
Chapter 21: Amazon Aurora

21_Core Services - Database - Aurora - Slides.pdf

Chapter overview

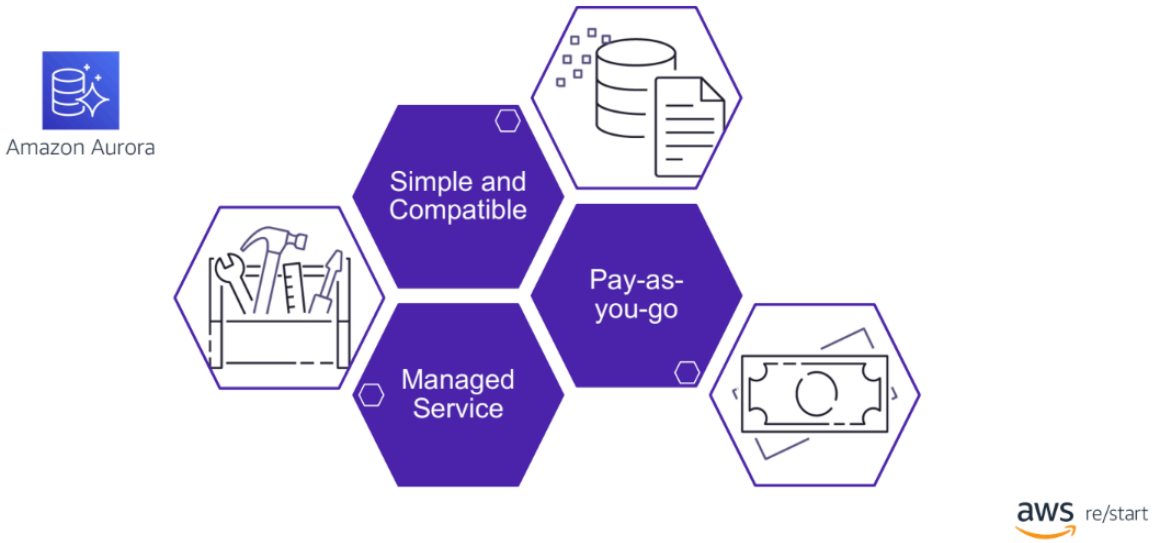
you'll gain insights into:

- Understanding the essence of Amazon Aurora, a robust database service.
- Exploring and uncovering the essential benefits that Amazon Aurora offers to its users.



Amazon Aurora

Key features



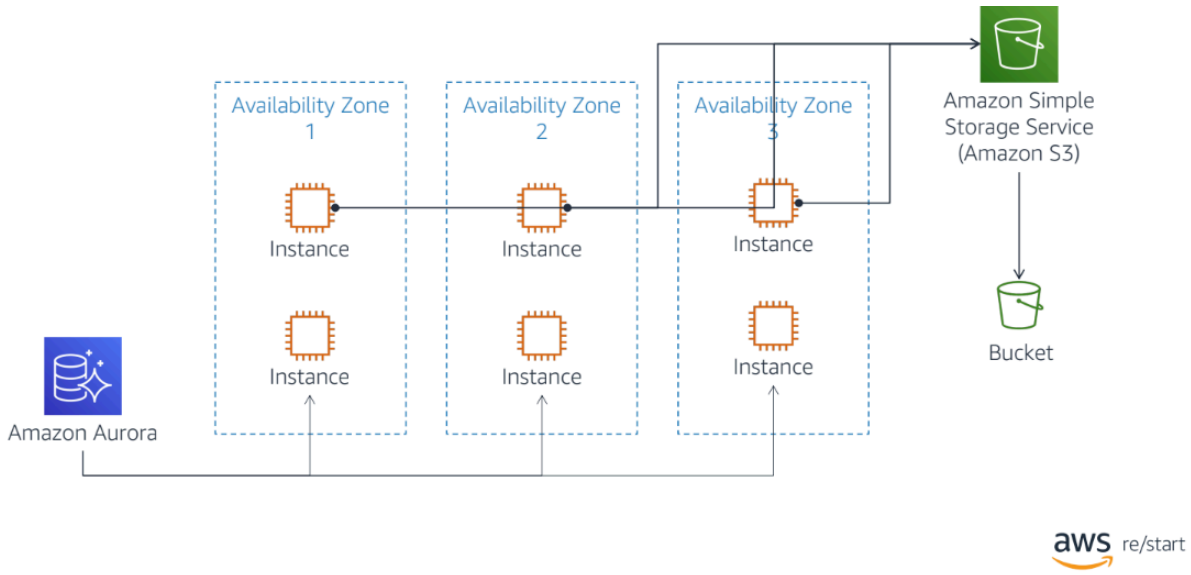
Amazon Aurora, a robust and high-performing relational database service, offers several distinctive features that set it apart in the world of database management:

1. Performance and Scalability

- **Efficient Performance:** Amazon Aurora delivers exceptional performance, providing five times the throughput of standard MySQL and three times that of PostgreSQL, making it an ideal choice for high-performance applications.

- **Auto-Scaling Capacity:** Its auto-scaling feature dynamically adjusts storage in 10GB increments, ensuring seamless scalability without compromising performance.

2. High Availability and Durability



- **Replication and Failover:** Aurora provides automatic and continuous replication across three availability zones, ensuring data durability and high availability.

- **Automated Failover:** In the event of a failure, Aurora automatically switches to a replicated instance within seconds, minimizing downtime and ensuring business continuity.

3. Compatibility and Integration

- **MySQL and PostgreSQL Compatibility:** Aurora is compatible with MySQL and PostgreSQL, allowing for easy migration and integration with existing applications and tools, providing a familiar environment for developers.

- **Native Integration with AWS Services:** It seamlessly integrates with various AWS services like AWS Lambda, Amazon Redshift, and Amazon S3, enabling enhanced functionality and data management capabilities.

4. Security and Cost Efficiency

- **Enhanced Security Features:** Aurora provides encryption at rest and in transit, ensuring data remains secure. It also offers fine-grained access control and audit logging for comprehensive security measures.

- **Cost-Effective Pricing:** With a pay-as-you-go pricing model, users only pay for the resources they consume, offering cost efficiency without compromising on performance or availability.

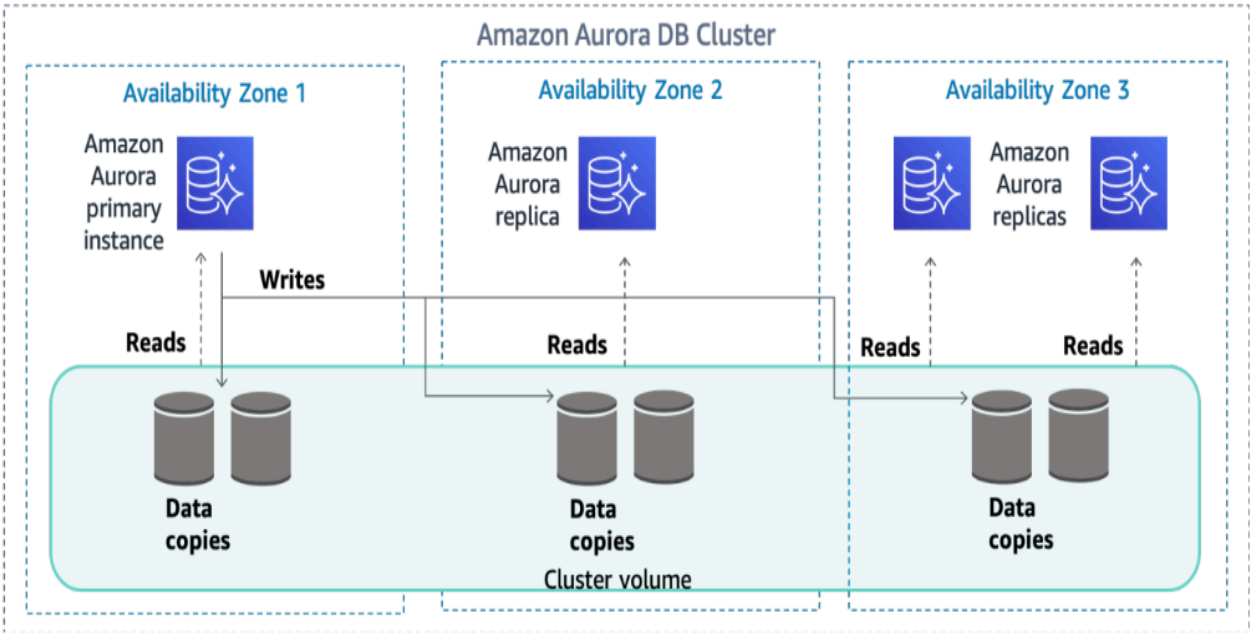
5. Backtrack and Point-In-Time Recovery

- **Backtrack Feature:** Aurora's backtrack feature allows for the rollback of databases to a specific point in time, enabling easy data recovery without impacting ongoing operations.

- **Point-In-Time Recovery:** It supports point-in-time recovery, enabling restoration of databases to a specific second within the retention period, ensuring data consistency and integrity.

Amazon Aurora's unique combination of performance, scalability, high availability, compatibility, security features, and cost efficiency makes it a compelling choice for businesses seeking a powerful and reliable database solution.

Amazon Aurora DB cluster architecture



aws re/start

Understanding Amazon Aurora DB Cluster Architecture

Amazon Aurora employs a unique and robust architecture designed for performance, high availability, and durability:

1. Clustered Storage: Aurora employs a clustered storage model. It separates compute and storage, creating a highly distributed and replicated storage layer across multiple nodes. This architecture contributes to enhanced performance and durability.

2. Primary Instance and Replicas: Each Aurora DB cluster comprises a primary instance and multiple read replicas. The primary instance manages write operations, while the replicas handle read queries, enabling horizontal scalability and load distribution.

3. Endpoints for Connectivity: Aurora provides multiple endpoints for connectivity. The cluster endpoint connects to the primary instance for read-write operations, while read-only endpoints facilitate connections to read replicas for distributed read access, enhancing performance.

4. Storage Volume and Redundancy: Data in Aurora is stored in a distributed and replicated volume across multiple Availability Zones (AZs). The data volume is automatically replicated six ways, ensuring durability and availability even in the event of AZ failures.

5. Quorum and Consistency: Aurora utilizes a quorum-based approach for writes. It replicates data across multiple storage nodes and maintains consistency by requiring

acknowledgement from a majority of nodes before confirming a write operation, ensuring data integrity.

6. Failover and Recovery: In the event of a failure, Aurora seamlessly performs automatic failover to a replica within the cluster. This swift failover process ensures minimal downtime and continuity of operations.

7. Multi-AZ Deployment: Aurora supports multi-Availability Zone (Multi-AZ) deployment, replicating data across different AZs for enhanced availability and disaster recovery capabilities.

8. Storage Scaling: The storage volume of an Aurora DB cluster dynamically scales in 10GB increments up to 64TB without impacting performance, providing seamless scalability as data requirements grow.

Chapter 22: Amazon Redshift

Chapter overview

At the core of this lesson, you will learn how to:

• Explain the details of Amazon Redshift.
• Explore the essential features of Amazon Redshift.

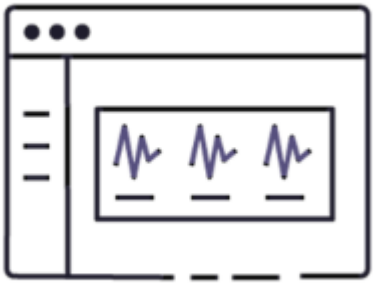
Amazon Redshift key features

Amazon Redshift stands out as a fully managed data warehouse solution, offering a suite of key features that empower users to handle data seamlessly. Let's explore these features to better understand how Amazon Redshift revolutionizes data warehousing.

Fully Managed Data Warehouse

Being fully managed means that Amazon Redshift handles administrative tasks such as hardware provisioning, configuration, and maintenance. This allows users to focus on utilizing the warehouse for their analytical needs without the burden of intricate system management.

Encryption for Security



Ensuring the security of your data is paramount, and Amazon Redshift recognizes this. It provides robust encryption measures both at rest and in transit. This means that whether your data is stored on disk or being transmitted between the data warehouse and your applications, it remains safeguarded through encryption, adding an extra layer of protection to sensitive information.

Compatibility with Standard SQL

Amazon Redshift facilitates a user-friendly experience by supporting standard Structured Query Language (SQL). This compatibility makes it easy for users familiar with SQL to seamlessly transition into using Amazon Redshift. SQL is a widely adopted language for managing and querying relational databases, making it accessible and versatile for users across different skill levels.

High-Performance Connectivity

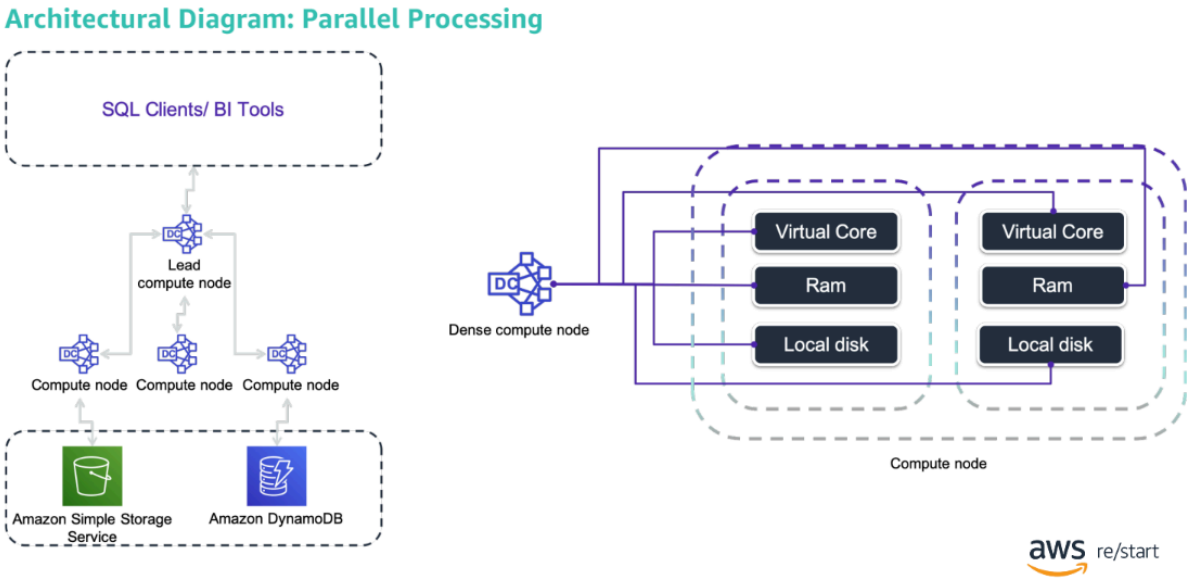
To enhance the accessibility of data, Amazon Redshift supports high-performance Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) connectors. These connectors enable efficient and reliable communication between the data warehouse and various applications, making it easier for developers to integrate Amazon Redshift into their existing workflows.

Conclusion

In summary, Amazon Redshift's key features, including its status as a fully managed data warehouse, robust encryption practices, compatibility with standard SQL, and high-performance connectivity options, collectively contribute to a user-friendly and secure environment for data analytics.

How Amazon Redshift works

Amazon Redshift Architecture



Amazon Redshift's architecture is designed to deliver high-performance analytics through a distributed and parallelized system. At its core, the architecture comprises clusters, nodes, and slices.

Clusters

A cluster is the foundational unit of Amazon Redshift and represents the entire data warehouse. It is composed of one leader node and multiple compute nodes.

Nodes

Nodes within the cluster serve distinct roles. The leader node manages query coordination and optimization, while the compute nodes handle data storage and processing.
The distribution of data across nodes is a key aspect of Amazon Redshift's parallel processing.

Slices

Each compute node is further divided into slices, which are segments of storage and processing capacity. The division into slices enables parallelization of queries, allowing multiple operations to be performed simultaneously on different slices.

Parallel Processing

Parallel processing is a fundamental concept in Amazon Redshift's functioning, contributing to its ability to swiftly process large datasets. Here's how parallel processing works within the architecture:

Data Distribution

Amazon Redshift distributes data across compute nodes using a method known as key-based distribution. This ensures that related data is stored on the same node, minimizing the need for data movement during query execution.

Parallel Execution

When a query is submitted to Amazon Redshift, the leader node optimizes and breaks down the query into smaller, parallelizable tasks. These tasks are then distributed across the compute nodes and their respective slices for simultaneous execution.

Aggregation and Reduction

As the parallelized tasks are executed, the results are aggregated and reduced back at the leader node. The final result set is then presented to the user, combining the outcomes of parallel processes into a cohesive response.

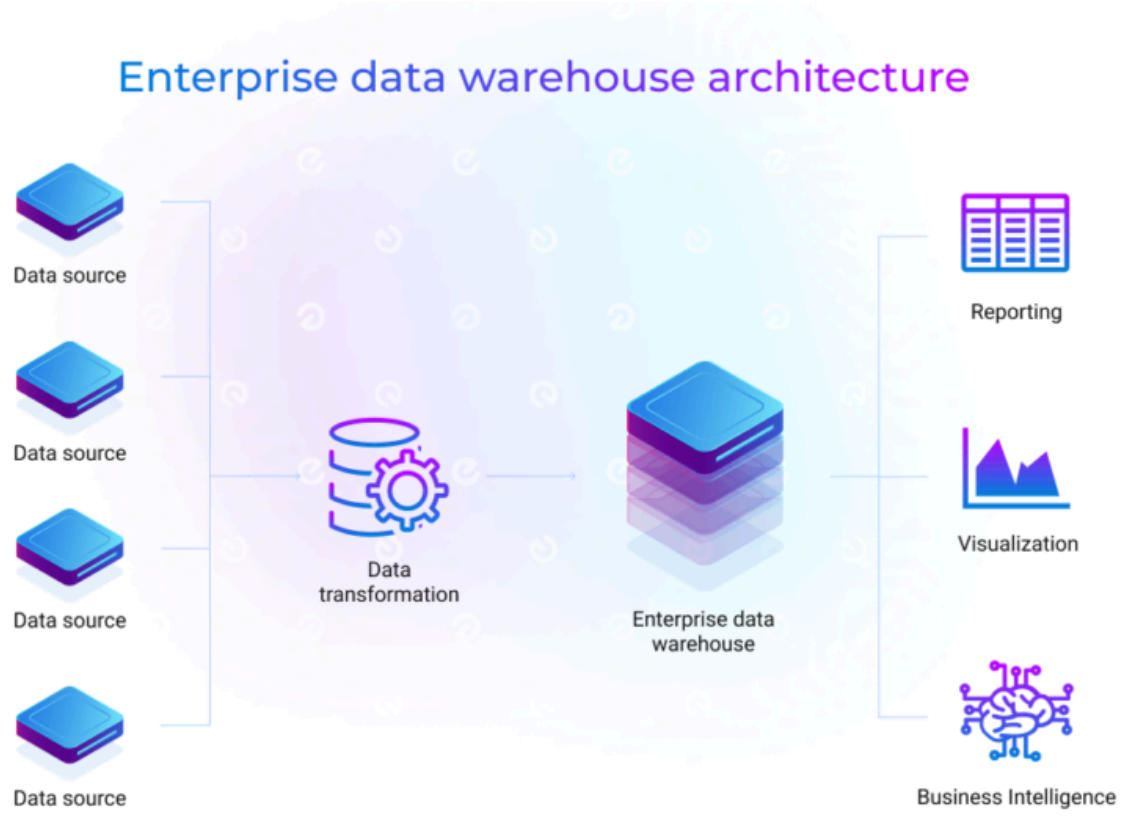
Conclusion

In summary, Amazon Redshift's architectural diagram, centered around parallel processing, forms the backbone of its high-performance analytics capabilities. By distributing data, optimizing queries, and executing tasks in parallel, Amazon Redshift enables efficient and swift processing of vast datasets, making it an invaluable tool for data warehousing and analytics.

Amazon Redshift use cases

In this exploration, we'll delve into the key use cases of Amazon Redshift, shedding light on how it serves as a transformative tool for enterprises, facilitates big data analytics, and seamlessly integrates with Software as a Service (SaaS) applications.

1. Enterprise Data Warehouse (EDW)

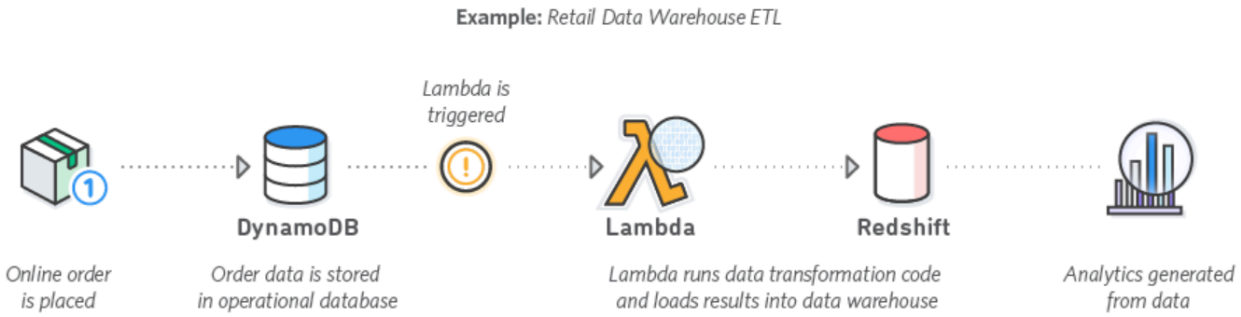


- **Migrate at a Comfortable Pace:** Amazon Redshift empowers organizations to migrate their data at a pace that aligns with their comfort level. This flexibility ensures a smooth transition, minimizing disruptions to ongoing operations during the migration process.

- **Experimentation without Upfront Commitment:** Organizations can experiment with Amazon Redshift without incurring large upfront costs or long-term commitments. This agile approach allows businesses to explore the benefits of Amazon Redshift before making a full-scale commitment.

- **Responsive to Business Needs:** Amazon Redshift enables organizations to respond swiftly to evolving business needs. Its scalable infrastructure allows for adjustments in real-time, ensuring that the data warehouse aligns with changing business requirements.

2. Big Data



- **Low Price Point for Small Customers:** Amazon Redshift offers a cost-effective entry point for small customers looking to harness the power of big data analytics. This affordability ensures that organizations of varying sizes can leverage the benefits of a sophisticated data warehouse.

- **Managed Service for Deployment and Maintenance Ease:** As a managed service, Amazon Redshift alleviates the complexities of deployment and ongoing maintenance. This enables organizations to focus more on utilizing data for strategic decision-making and less on the intricacies of database management.

- **Focus on Data, Not Database Management:** Amazon Redshift's design places a strong emphasis on data, allowing users to concentrate on deriving insights from their data rather than getting bogged down by database management intricacies.

3. Software as a Service (SaaS)

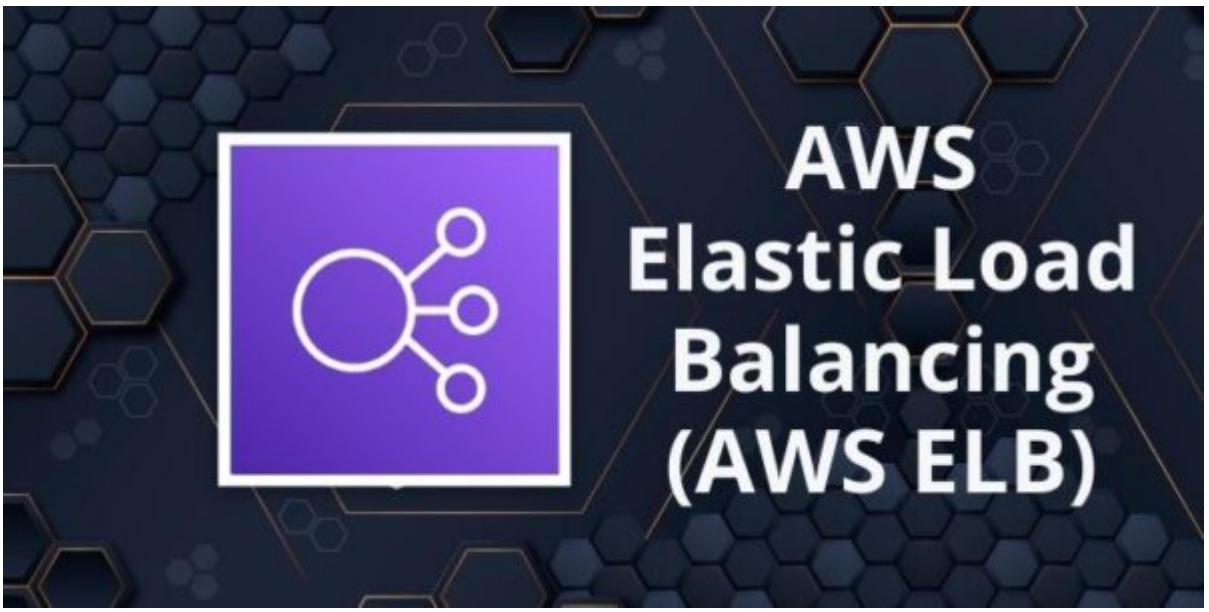
- **Scalability with Demand:** Amazon Redshift seamlessly scales its data warehouse capacity based on demand. This elasticity ensures that SaaS providers can accommodate varying workloads without compromising performance or incurring unnecessary costs.

- **Adding Analytic Functionality to Applications:** SaaS applications can enhance their offerings by integrating Amazon Redshift to add powerful analytic functionality. This integration empowers users to extract valuable insights directly within the SaaS environment.

- **Cost Reduction in Hardware and Software:** By leveraging Amazon Redshift, SaaS providers can substantially reduce hardware and software costs. The cloud-based nature of Amazon Redshift eliminates the need for extensive infrastructure investments, providing a more cost-effective solution.

23 Core Services - Balancing Scaling - ELB - Slides

Chapter 23: Elastic Load Balancing



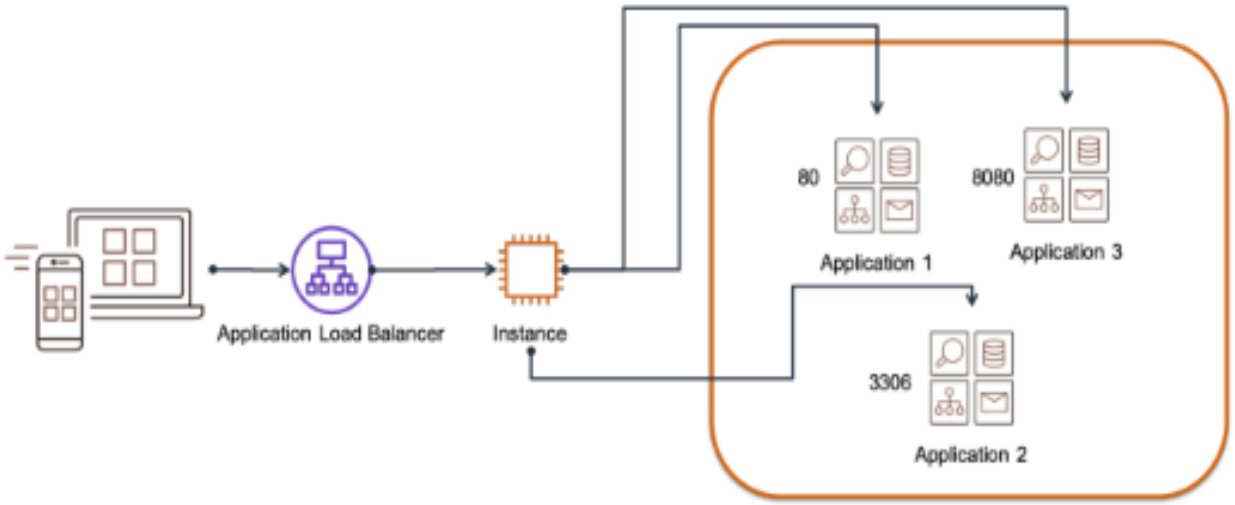
Chapter overview

you will acquire the skills to:

• Describe the details of one AWS load balancing service, Elastic Load Balancing (ELB).

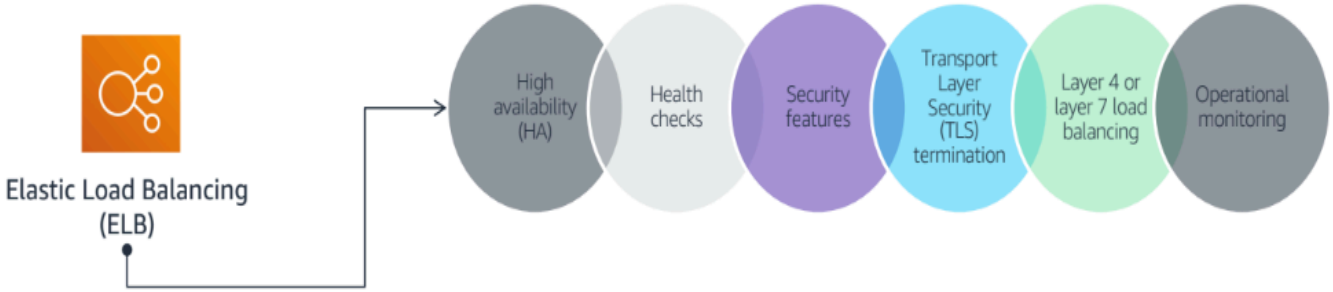
• Highlight the three different types of Elastic Load Balancing load balancers.

What is a Load Balancer?



A load balancer is a critical component within a network infrastructure designed to enhance the performance, reliability, and scalability of applications. Its primary function is to automatically distribute incoming application traffic across multiple targets. This strategic distribution ensures optimal resource utilization, prevents overload on any single target, and ultimately enhances the overall efficiency of the application.

Features



ELB load balancers automatically distribute incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, or IP addresses. This automated distribution ensures optimal resource utilization and prevents overload on any single target.

1. High Availability (HA)

- **Redundancy:** ELB is designed for high availability, offering redundancy in the distribution of incoming traffic. In the event of a failure or disruption in one availability zone, ELB redirects traffic to healthy targets in other zones, ensuring continuous operation and minimizing downtime.

2. Health Checks

- **Proactive Monitoring:** ELB conducts health checks on registered targets, regularly assessing their status. If a target fails a health check, the load balancer automatically reroutes traffic away from the unhealthy target to maintain the overall health and reliability of the application.

3. Security Features

- **SSL/TLS Termination:** ELB supports Transport Layer Security (TLS) termination, allowing it to decrypt traffic at the load balancer and then re-encrypt it before forwarding it to the targets. This enhances security by centralizing encryption processes and providing a secure communication channel.

4. Transport Layer Security (TLS) Termination

- **Secure Communication:** ELB can terminate SSL/TLS connections, handling the decryption of incoming encrypted traffic. This feature allows the load balancer to inspect and manipulate the traffic before forwarding it to the targets, enhancing security and facilitating communication between the load balancer and the application.

5. Layer 4 or Layer 7 Load Balancing

- **Flexibility in Load Balancing:** ELB provides the flexibility of operating at both Layer 4 (Transport layer) and Layer 7 (Application layer). Layer 4 load balancing involves distributing traffic based on IP addresses and ports, while Layer 7 load balancing adds the capability to make routing decisions based on content, such as HTTP headers.

6. Operational Monitoring

- **Insight into Performance:** ELB offers operational monitoring features that provide insight into the performance and health of the load balancer and its associated targets. Metrics such as request count, latency, and error rates are monitored, allowing administrators to make informed decisions and optimize the overall system.

Types of Elastic Load Balancers

Types

Application Load Balancer (ALB)	Network Load Balancer (NLB)	Classic Load Balancer (CLB)
HTTP HTTPS	TCP	PREVIOUS GENERATION for HTTP, HTTPS, and TCP
Flexible application management	Extreme performance and static IP for your application	Existing application that was built within the Amazon EC2-Classic network
Advanced load balancing of HTTP and HTTPS traffic	Load balancing of TCP traffic	Operates at both the request level and connection level
Operates at the request level (Layer 7)	Operates at the connection level (Layer 4)	

aws re/start

1. Application Load Balancer (ALB)

Purpose

ALB operates at the application layer (Layer 7) of the OSI model, making intelligent routing decisions based on content. It is ideal for applications that require advanced routing capabilities and support multiple domains.

Features

- **Content-Based Routing:** ALB can route traffic based on content, enabling it to direct requests to different backend services based on the content of the request.
- **WebSocket Support:** Ideal for applications utilizing WebSocket protocols for real-time communication.
- **Target Groups:** Allows grouping of targets based on different criteria, providing flexibility in managing application components.

2. Network Load Balancer (NLB)

Purpose

NLB operates at the transport layer (Layer 4) and is well-suited for applications that require high throughput and low-latency processing of TCP traffic.

Features

- **TCP and UDP Load Balancing:** NLB is capable of handling both TCP and UDP traffic, making it suitable for a wide range of applications.

- **Static IP Address:** Provides a static IP address for the load balancer, offering a stable endpoint for applications.

3. Classic Load Balancer

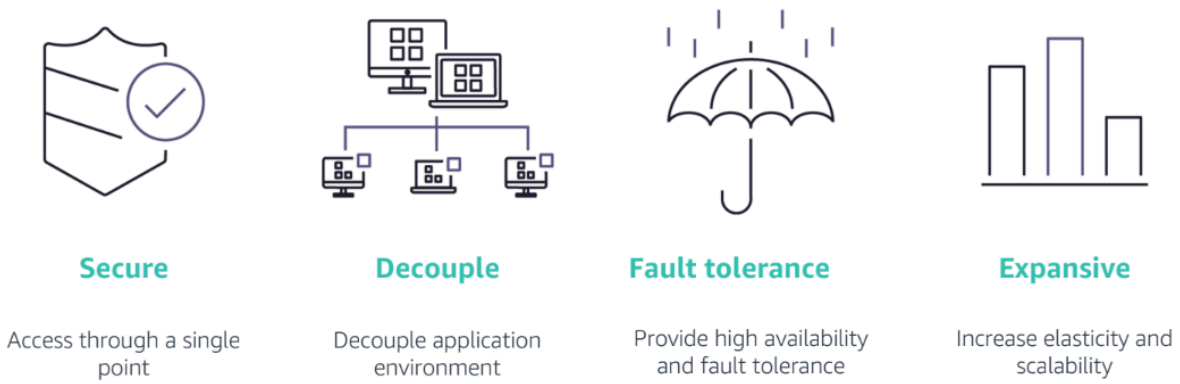
Purpose

Classic Load Balancer is the legacy version that provides basic load balancing across multiple Amazon EC2 instances.

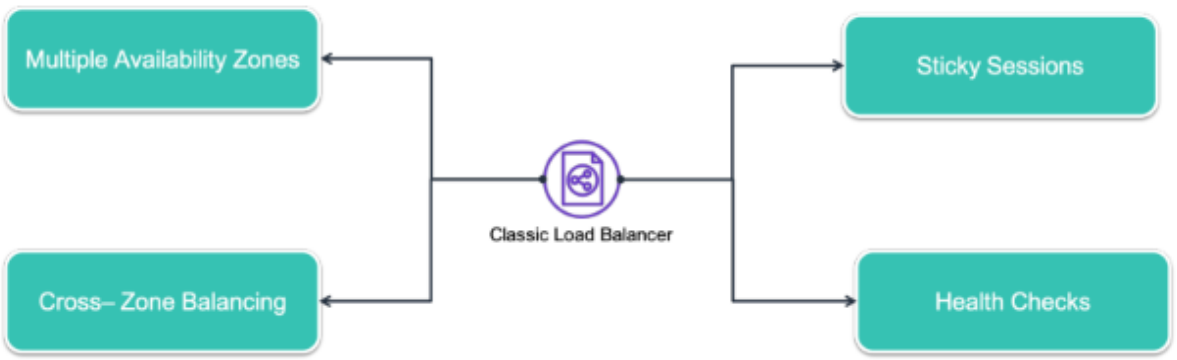
Features

- **Support for EC2-Classic:** Classic Load Balancer supports applications within the EC2-Classic network.
- **Simple Configuration:** Offers a straightforward configuration process suitable for basic load balancing needs.

Load balancer use cases



Classic Load Balancer



1. Access Servers through a Single Point

Classic Load Balancer serves as a central access point, directing incoming traffic to multiple servers. This ensures simplicity for end-users who interact with a single entry point, enhancing user experience and ease of access.

2. Decouple the Application Environment

Classic Load Balancer effectively decouples the application environment by distributing traffic among multiple servers. This decoupling enhances flexibility and resilience, allowing changes or updates to be made to individual components without affecting the overall application.

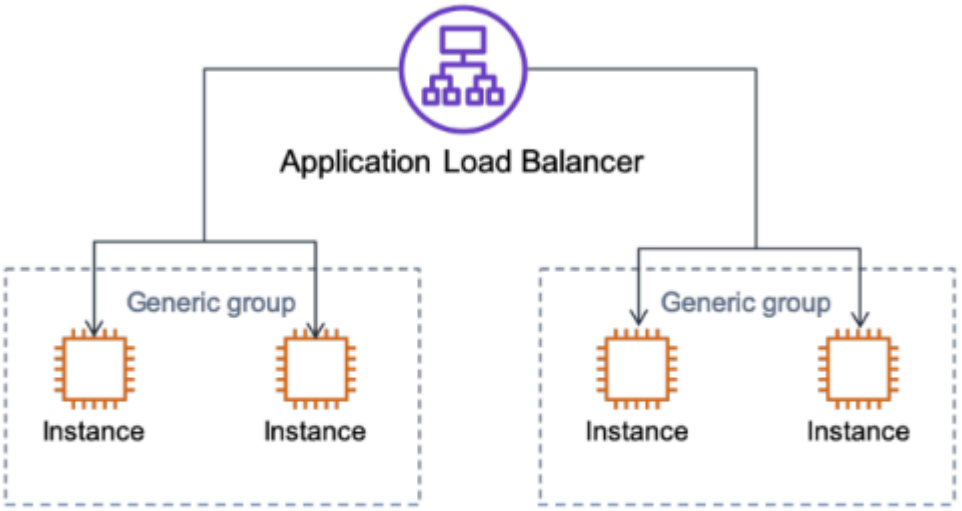
3. Provide High Availability and Fault Tolerance

Classic Load Balancer ensures high availability by distributing traffic evenly across servers. In the event of a server failure, traffic is automatically redirected to healthy servers, minimizing downtime and enhancing fault tolerance.

4. Increase Elasticity and Scalability

Classic Load Balancer facilitates the scalability of applications by dynamically distributing traffic across multiple servers. As the demand for an application increases, additional servers can be added, ensuring elastic scaling to meet varying workloads.

Application Load Balancer



1. Path-Based and Host-Based Routing

Application Load Balancer excels in directing traffic based on specific paths or hosts, allowing for granular control over how requests are handled. This is beneficial for applications with diverse content or multiple subdomains.

2. Native IPv6 Support

Application Load Balancer natively supports IPv6, accommodating the growing need for the next-generation Internet Protocol and ensuring seamless communication in IPv6-enabled networks.

3. Dynamic Ports

Unlike Classic Load Balancer, Application Load Balancer supports dynamic ports, providing flexibility in handling requests on different ports based on the application's requirements.

4. Additional Supported Request Protocols

Application Load Balancer supports a variety of request protocols beyond HTTP and HTTPS, making it versatile for applications with diverse communication needs.

5. Deletion Protection and Request Tracking

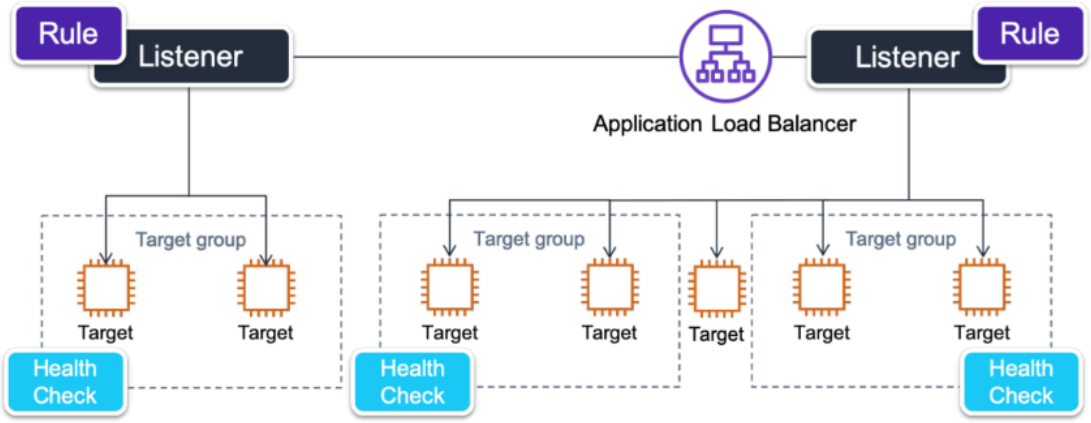
This load balancer type offers features such as deletion protection, preventing accidental deletion of resources, and request tracking, which aids in monitoring and analyzing incoming requests.

6. Enhanced Metrics and Access Logs

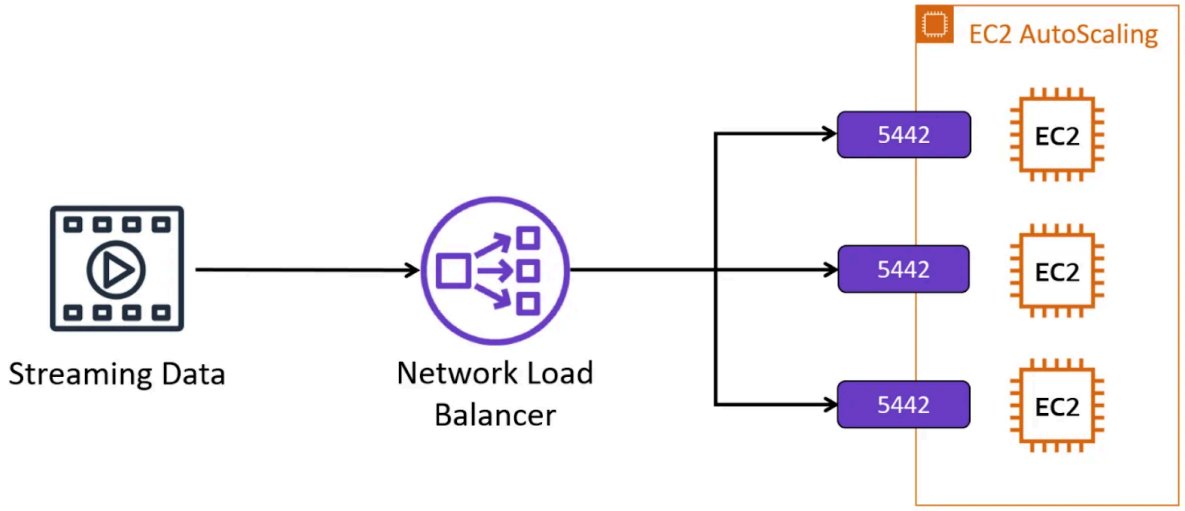
Application Load Balancer provides detailed metrics and access logs, offering deeper insights into application performance and user interactions.

7. Targeted Health Checks

Health checks are specifically tailored to targets, ensuring accurate monitoring and prompt response to the health status of individual components.



Network Load Balancer



1. Sudden and Volatile Traffic Patterns

Network Load Balancer is designed to handle sudden and volatile traffic patterns effectively, making it suitable for applications that experience unpredictable spikes in demand.

2. Single Static IP Address per Availability Zone

Network Load Balancer provides a single static IP address per Availability Zone, simplifying the management of network configurations and providing stability in dynamic environments.

3. Extreme Performance Requirements

Well-suited for applications that demand extreme performance, Network Load Balancer excels in efficiently distributing traffic while minimizing latency.

4. View HTTP Responses

Network Load Balancer allows administrators to view HTTP responses, providing valuable insights into how the application is interacting with incoming requests.

5. See Number of Healthy and Unhealthy Hosts

Administrators can easily monitor the health of hosts, gauging the number of healthy and unhealthy instances, facilitating proactive maintenance and troubleshooting.

6. Filter Metrics Based on Availability Zones or Load Balancer

Network Load Balancer allows metrics to be filtered based on Availability Zones or the load balancer itself, providing granular control over monitoring and analysis.

Conclusion

Load Balancing Feature	APPLICATION Load Balancer	NETWORK Load Balancer	CLASSIC load balancer	Use Cases
Platforms	VPC	VPC	EC2-Classic, VPC	Use CLB if using EC2 Classic
Protocol	HTTP, HTTPS	TCP	SSL, TCP, HTTP, HTTPS	Use NLB for TCP only, Use CLB for SSL
Elastic IP Address		Y		Use NLB when you need to support an IP address
Static IP Address		Y		
Preserve Source IP address		Y		
Health checks	Y	Y	Y	the common features all load balancers can deliver
CloudWatch metrics	Y	Y	Y	
Logging	Y	Y	Y	
Zone fail over	Y	Y	Y	
Connection draining	Y	Y	Y	
Load balance to more than one port on the same instance	Y	Y		Use ALB or NLB for container services and / or ECS
Web sockets	Y	Y		
IP address as targets	Y	Y		
Delete protection	Y	Y		
Path based / Host based routing	Y			Use ALB for specific routing
Native HTTP/2	Y			Use ALB to support native HTTP/2
Configurable idle time out	Y		Y	
Cross zone load balancing	Y		Y	
SSL offloading	Y		Y	Use ALB or CLB if you need to support SSL offloading
Sever name indication SNI	Y			
Sticky sessions	Y		Y	
Back end server encryption	Y		Y	

Chapter 24: Amazon CloudWatch

24 Core Services - Balancing Scaling - CloudWatch - Slides.pdf

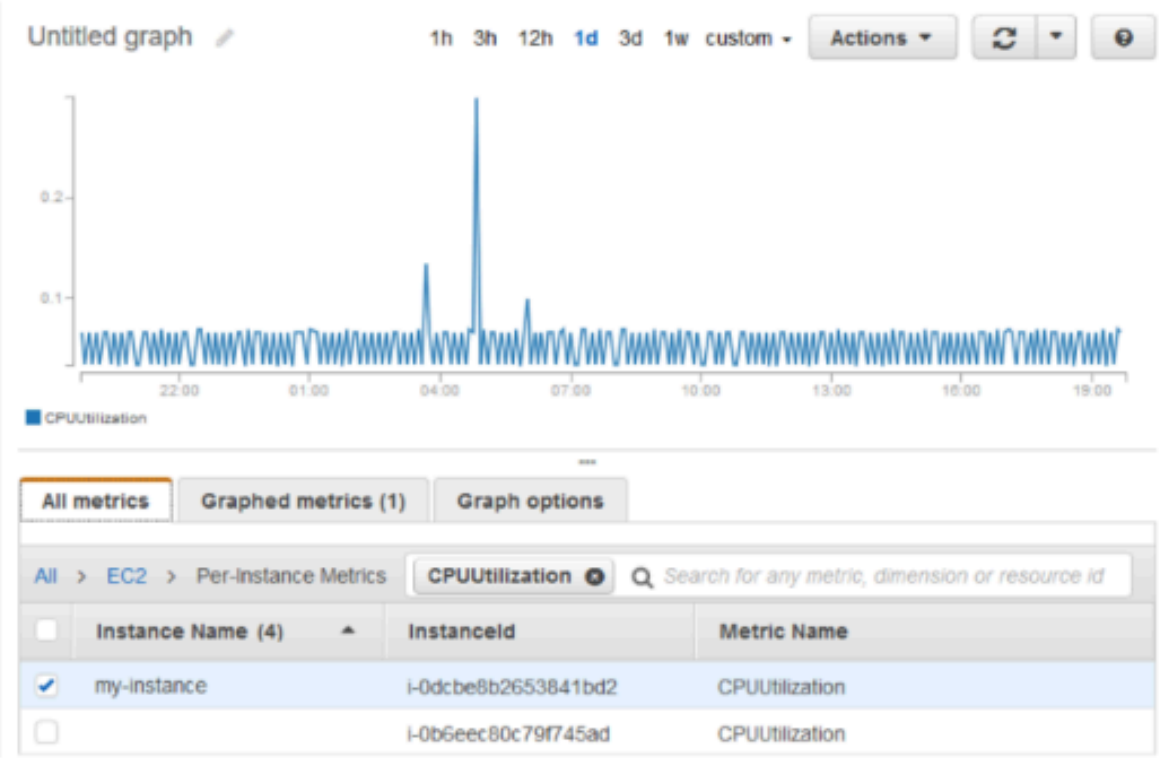
Amazon CloudWatch

Chapter Overview

You will learn how to:

- Describe an AWS monitoring service, Amazon CloudWatch
- Highlight the three components of CloudWatch

Leverage AWS efficiently and gain insight



➔ **How do I know when I should launch more Amazon EC2 instances?**

★ Amazon EC2 instances are the backbone of your computing power in AWS. Efficiently utilizing them involves understanding when to scale. Insightful monitoring tools, such as AWS CloudWatch, can provide metrics on CPU utilization, memory usage, and other performance indicators. By analyzing these metrics, you can determine if launching additional EC2 instances is necessary to meet the demand of your applications.

➔ **Is my application's performance or availability being affected by a lack of sufficient capacity?**

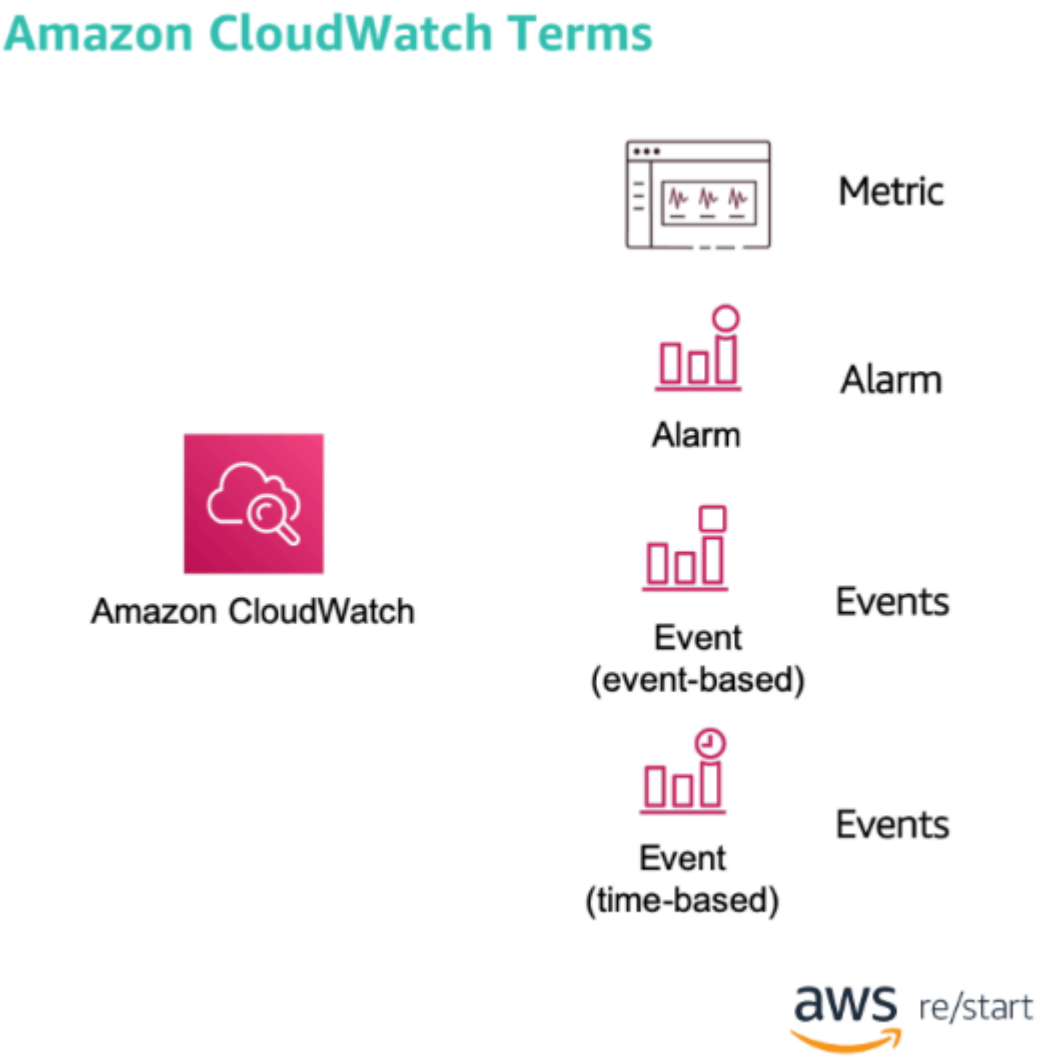
★ Efficient AWS usage requires a keen understanding of how your application's performance and availability are impacted by the available capacity. This involves monitoring factors such as network throughput, disk I/O, and response times. AWS CloudWatch, coupled with AWS Auto Scaling, allows you to dynamically adjust capacity based on these metrics, ensuring optimal

performance and availability. Think of your application as a busy highway. Monitoring capacity is like gauging traffic flow. If traffic is heavy, you might need to widen the road (increase capacity) to prevent congestion and keep things moving smoothly.

➔ **How much of my infrastructure is actually being used?**

★ Understanding the true utilization of your infrastructure involves assessing how much of your resources are actively in use. AWS provides tools like AWS CloudTrail and AWS Config that enable you to track changes, monitor usage patterns, and gain a comprehensive view of your infrastructure. Imagine your infrastructure as a library. Monitoring its usage is like tracking which books are being read and how often. This insight helps you optimize the library's layout or expand certain sections based on popularity.

What is Amazon CloudWatch?



Amazon CloudWatch stands as a service within the Amazon Web Services (AWS) ecosystem, designed to monitor, analyze, and optimize the state and utilization of a wide array of resources managed under AWS. Let's delve into the key concepts and functionalities of Amazon CloudWatch to gain a comprehensive understanding.

Key Concepts of Amazon CloudWatch

1. Standard Metrics

Standard metrics are pre-built performance metrics provided by AWS for various AWS resources. These metrics offer insights into the health and performance of your resources without any additional configuration. For example, CPU utilization, network activity, and disk I/O are standard metrics for Amazon EC2 instances. Think of standard metrics as the basic

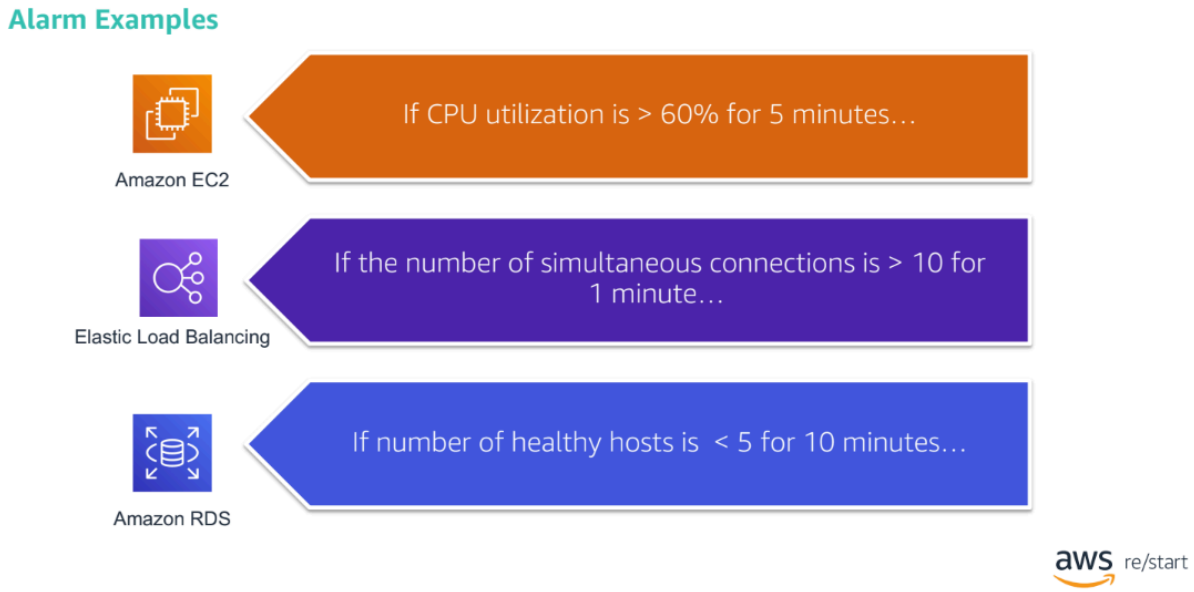
health indicators for your AWS resources. They are like built-in instruments that show how well your resources are performing.

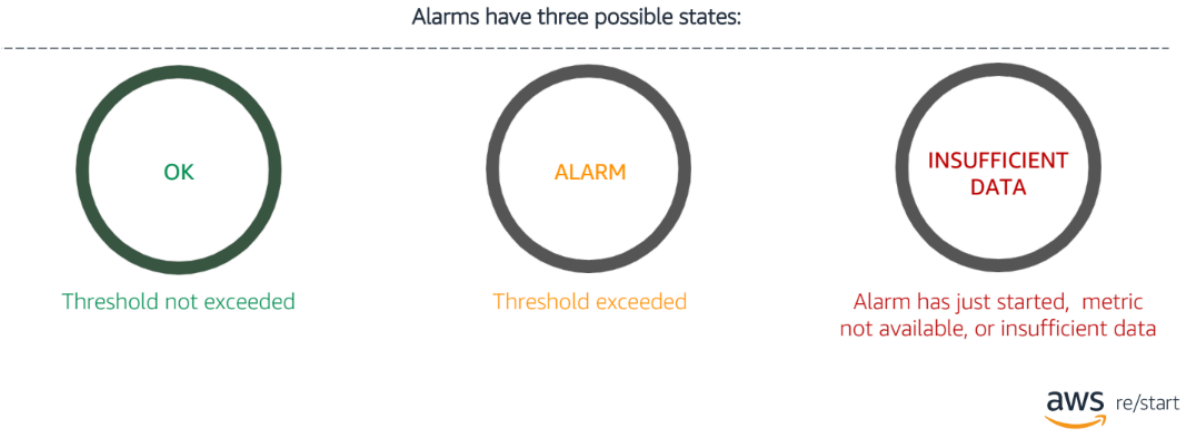
2. Custom Metrics

Custom metrics allow you to monitor specific aspects of your resources that might not be covered by standard metrics. You can create and publish custom metrics to CloudWatch, enabling you to track the exact parameters that are crucial for your specific applications. Custom metrics are like creating your own unique health checklist for resources. If there's a specific aspect you want to monitor, like a special feature of your application, you can design a custom metric for it.

3. Alarms

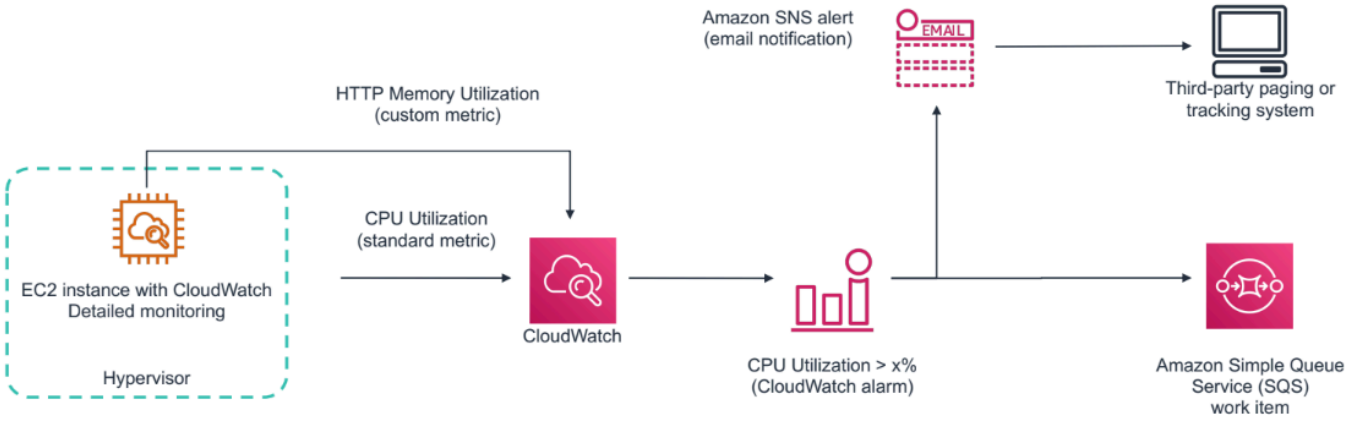
Alarms in CloudWatch enable you to set thresholds on metrics. If a metric breaches the defined threshold, an alarm is triggered, allowing you to take proactive actions. For instance, you can set up an alarm to notify you when CPU utilization on an EC2 instance exceeds a certain percentage. Alarms act as your personal alert system. When a metric misbehaves, an alarm rings, letting you know that something needs attention.





4. Notifications

CloudWatch allows you to set up notifications that are triggered when an alarm enters a particular state, such as "ALARM" or "OK." These notifications can be sent to various endpoints like Amazon SNS (Simple Notification Service), enabling you to stay informed about the status of your resources. Notifications are like messages that CloudWatch sends to your phone when there's something important happening. They keep you in the loop about the health of your resources.



CloudWatch Agent for System-Level Metrics

1. Amazon EC2 Instances

The CloudWatch Agent is a tool that facilitates the collection of system-level metrics from Amazon EC2 instances. It gathers data on performance parameters like CPU utilization, memory usage, and disk space, providing a comprehensive view of the health and performance of your EC2 instances. Imagine the CloudWatch Agent as a friendly assistant

on your EC2 instances. It keeps track of how busy your instances are and lets you know if they need extra help.

2. On-Premises Servers

The CloudWatch Agent extends its functionality beyond the AWS cloud to collect system-level metrics from on-premises servers. This ensures that whether your resources are in the cloud or on your premises, CloudWatch provides a unified monitoring solution. The CloudWatch Agent is like a traveler, capable of monitoring resources both at home and in a distant land. It ensures that, regardless of location, you have a consistent view of your resources.

Standard and custom metrics

Standard Metrics

1. Grouped by Service Name

Standard metrics are pre-built performance indicators specific to each AWS service. These metrics are grouped by the name of the service they monitor, making it easy to locate and understand them. For example, Amazon EC2 instances have standard metrics such as CPU utilization, network activity, and disk I/O. Think of standard metrics as ready-made tools designed for specific jobs. Each tool is labeled by the service it monitors, providing a clear and organized toolbox for understanding the performance of AWS services.

2. Display Graphically for Comparison

Standard metrics are displayed graphically in the CloudWatch console. This graphical representation allows users to compare different metrics visually, aiding in the assessment of resource performance over time. Visualizing standard metrics is like creating a performance

chart for each AWS service. You can compare these charts to see how different services are performing and identify any patterns or trends.

3. Time Limitation

Standard metrics only appear in the CloudWatch console if the associated AWS service has been used within the past 15 months. This limitation ensures that the console remains focused on currently relevant metrics. Standard metrics are like snapshots taken every 15 months. If a service hasn't been used recently, its metrics won't clutter the console, keeping things tidy and relevant.

4. Programmatic Accessibility

Explanation: Standard metrics can be accessed programmatically through the AWS Command Line Interface (CLI) or the application programming interface (API). This programmability allows users to incorporate metric data into scripts or applications for automated monitoring. Accessing standard metrics programmatically is akin to having a set of magical commands that fetch information about AWS services. It allows users to automate monitoring tasks and integrate metrics into custom workflows.

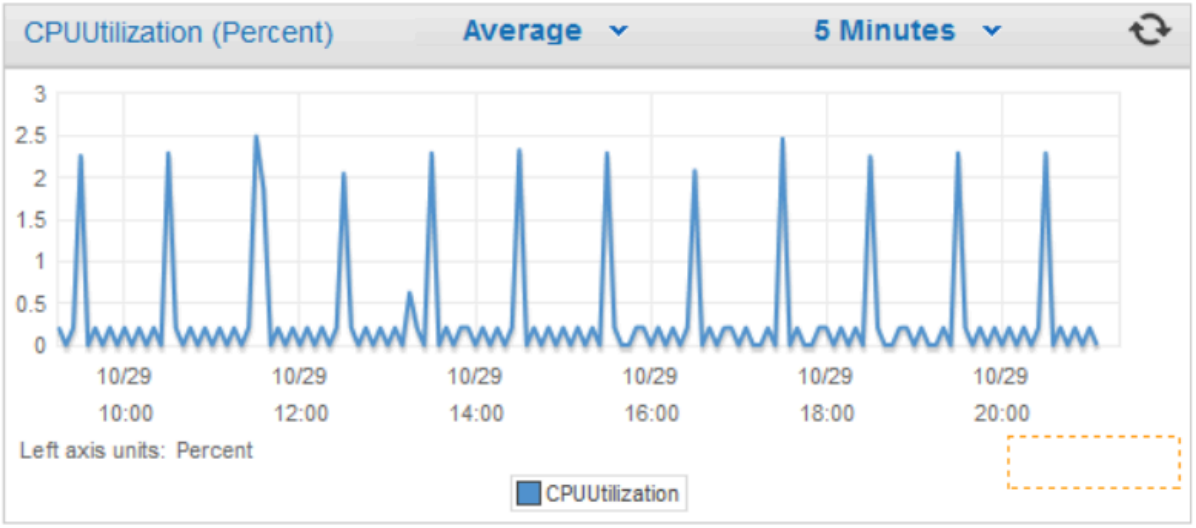
Custom Metrics

1. Grouped by User-Defined Namespace

Custom metrics are user-defined performance indicators that are grouped by a namespace specified by the user. A namespace is a container for metrics and helps organize them logically. For instance, you might create a namespace for a specific application or project. Custom metrics are like creating your own set of tools. You get to decide the categories (namespaces) and tools (metrics) that are most meaningful for your specific needs.

2. Publishing to CloudWatch

Users can publish custom metrics to CloudWatch using various methods, including the AWS CLI, API, or the CloudWatch Agent. This flexibility enables users to gather data from diverse sources and tailor metrics to their specific monitoring requirements. Publishing custom metrics is like becoming a data explorer. You decide what data to collect, how to collect it, and where to display it in CloudWatch, providing a personalized and dynamic approach to monitoring.



Monitoring for Suspicious Activity

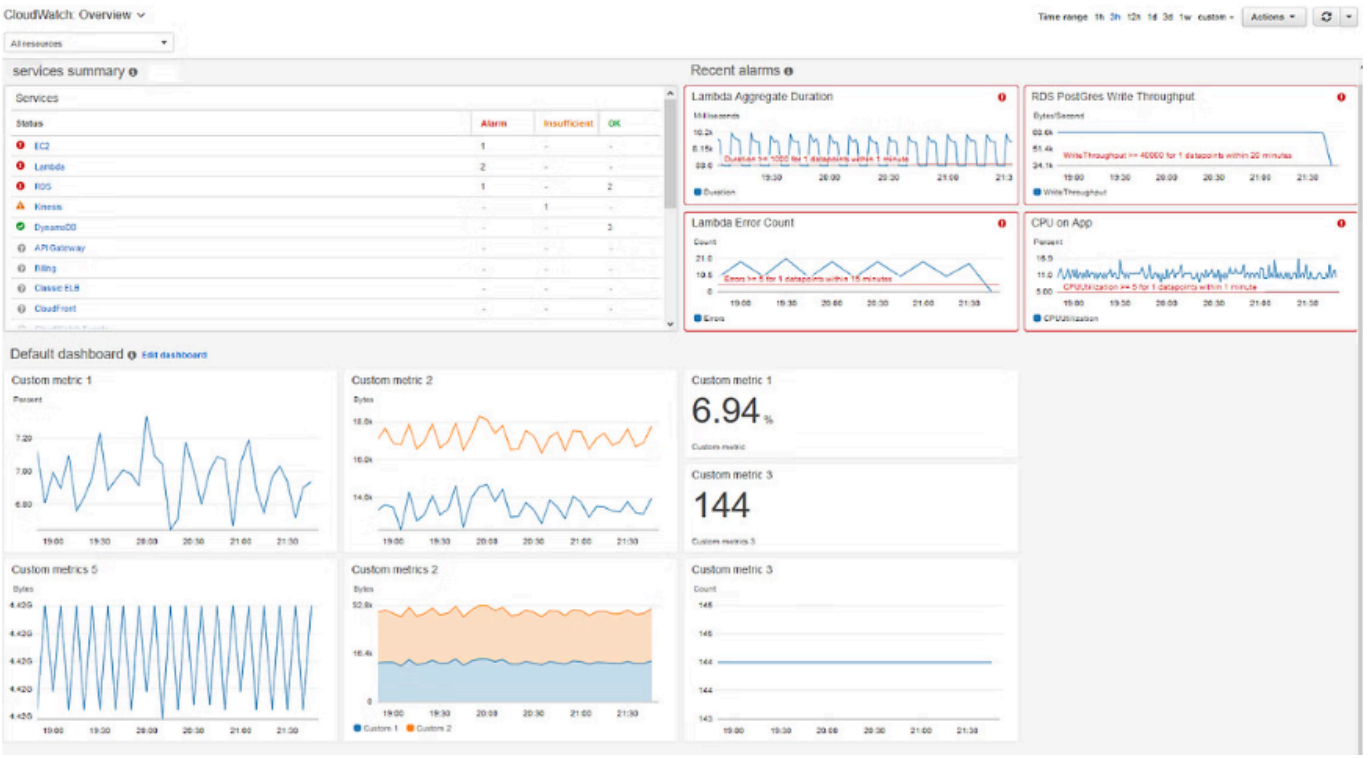
1. Unusual Spikes in Service Usage

CloudWatch allows you to set up alarms that monitor various metrics, such as CPU utilization, disk activity, and Amazon Relational Database Service (Amazon RDS) usage. Unusual, prolonged spikes in these metrics may indicate abnormal behavior or potential security threats. Think of it like having a watchful guardian for your resources. If there's an unexpected surge in activity, the guardian (CloudWatch) alerts you, helping you investigate and ensure everything is secure.

2. Alerts on Billing Metrics

CloudWatch enables you to set alerts on billing metrics. This is particularly useful for monitoring unexpected or suspicious changes in your billing patterns. By enabling these alerts in your AWS account settings, you can receive notifications if there are anomalies in your billing metrics. Imagine these alerts as a financial watchdog for your AWS account. If there are unusual charges or deviations from your regular billing patterns, the watchdog notifies you, allowing you to promptly address any potential security concerns.

CloudWatch automatic dashboards



Key Features of Amazon CloudWatch Dashboards

1. Surface Data about Your AWS Ecosystem

CloudWatch Dashboards act as centralized hubs that aggregate and display essential metrics and information about your AWS resources. These visualizations enable you to quickly grasp the overall health and status of your entire ecosystem. Imagine the dashboard as a master control panel for your AWS world. It consolidates information from various services into a single, easy-to-read display, allowing you to see the big picture at a glance.

2. Dynamic and Customizable

CloudWatch Dashboards are highly customizable, allowing you to tailor them to your specific monitoring needs. You can arrange and resize widgets, choose the metrics to display, and create layouts that suit your preferences. Think of the dashboard as your personalized monitoring canvas. You get to decide what metrics are important to you, where they are

displayed, and how you want to visualize them, providing a unique and flexible monitoring experience.

3. Integration with Existing Monitoring Tools

CloudWatch Dashboards can seamlessly integrate with existing monitoring tools and workflows. Whether you use third-party monitoring solutions or have in-house tools, the dashboards can complement and enhance your overall monitoring strategy. Integration is like creating a bridge between CloudWatch Dashboards and your favorite tools. It ensures that you can continue using the tools you're familiar with while leveraging the visual power of CloudWatch Dashboards for AWS-specific insights.

Chapter 25: Amazon EC2 Auto Scaling

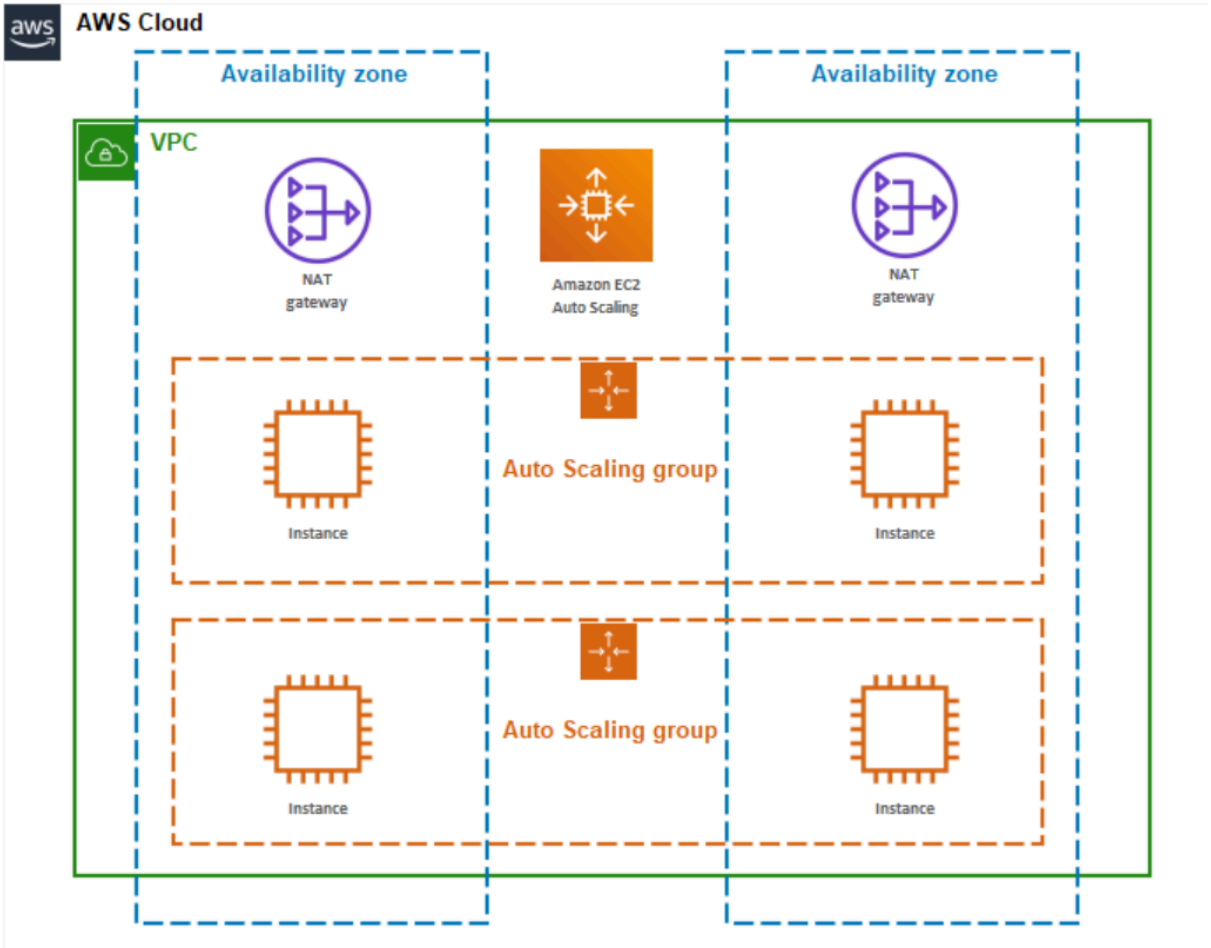
25_Core Services - Balancing _ Scaling - Scaling - Slides.pdf

Chapter Overview

You will learn how to:

- Describe the details of an AWS automatic scaling service, Amazon EC2 Auto Scaling
- Highlight the components of Amazon EC2 Auto Scaling

What is Amazon EC2 Auto Scaling?



Amazon EC2 Auto Scaling is a robust service designed to efficiently manage and scale computing resources in the cloud. This tool simplifies the process of dynamically adjusting the capacity of your applications to meet changing demand, ensuring optimal performance and cost-effectiveness.

User Interface

Amazon EC2 Auto Scaling boasts a user-friendly interface that facilitates the creation of scaling plans. This interface serves as a control center, allowing users to define how their resources should scale based on specific criteria, such as demand fluctuations or other performance metrics.

Scaling Plans

Scaling plans are at the core of Amazon EC2 Auto Scaling. These plans enable users to outline the rules and conditions under which their resources should automatically scale. This

includes scenarios like adding or removing EC2 instances or adjusting capacity for DynamoDB tables and Aurora replicas.

Supported Resources

Amazon EC2 Auto Scaling primarily caters to Amazon EC2 instances and Spot Fleets, providing flexibility in managing compute capacity. However, it goes beyond just virtual machines, extending its support to other critical resources like Amazon DynamoDB tables and indexes. This means users can ensure that their database resources also scale seamlessly to handle varying workloads.

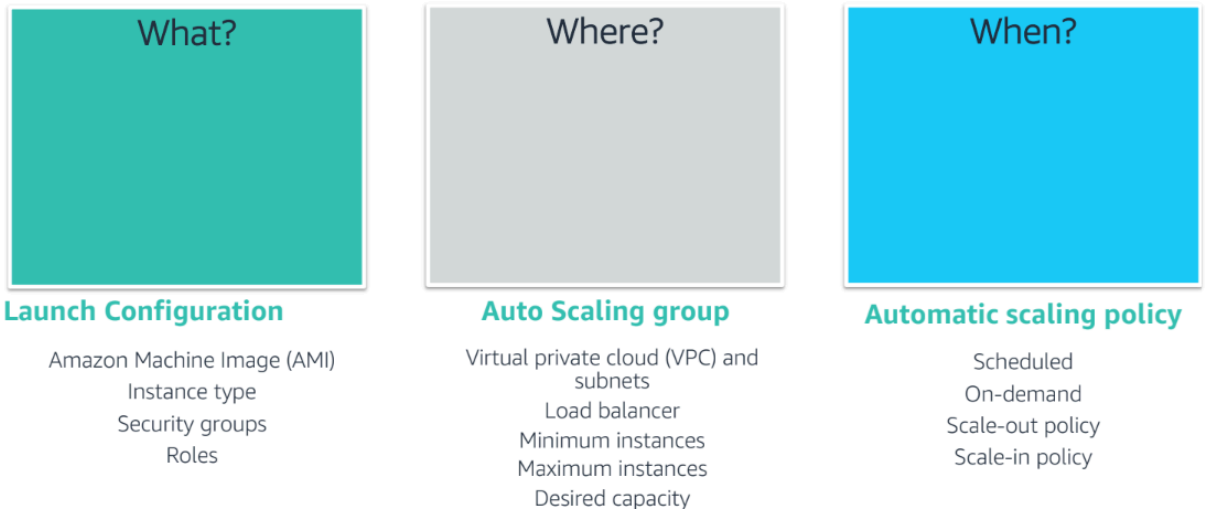
Spot Fleets

Spot Fleets refer to a collection of Spot Instances, which are spare compute capacity available at a potentially lower cost than On-Demand Instances. Amazon EC2 Auto Scaling's integration with Spot Fleets allows users to harness cost savings by intelligently utilizing these instances based on their application's needs.

DynamoDB and Aurora Integration

In addition to EC2 instances, Amazon EC2 Auto Scaling integrates with Amazon DynamoDB and Aurora. For DynamoDB, it can dynamically adjust the capacity of tables and indexes, ensuring that your NoSQL database scales efficiently. Similarly, with Aurora replicas, the service can manage the replication of your relational database, optimizing performance as demand fluctuates.

Automatic scaling components



Automatic scaling is a fundamental aspect of cloud computing that allows dynamic adjustment of resources to meet varying demand. To comprehend the components involved in automatic scaling, it's essential to explore the "What," "Where," and "When" of this process.

What?

Launch Configuration

At the core of automatic scaling is the launch configuration. It defines the specifications for the instances that will be automatically launched or terminated based on demand. This includes details such as the Amazon Machine Image (AMI), instance type, security groups, and roles.

Amazon Machine Image (AMI)

The AMI serves as the blueprint for the instances that will be launched. It encapsulates the necessary information, including the operating system, application server, and applications, providing a standardized environment for the instances.

Instance Type, Security Groups, and Roles

These components within the launch configuration define the characteristics and permissions of the instances. The instance type determines the computational capacity, security groups control inbound and outbound traffic, and roles grant necessary permissions to access other AWS services.

Where?

Auto Scaling Group

The Auto Scaling group is the orchestration mechanism that manages the automatic scaling of instances. It uses the launch configuration to maintain a specified number of instances, distributing them across availability zones for improved fault tolerance.

Virtual Private Cloud (VPC) and Subnets

The VPC and subnets define the networking environment in which the instances operate. This includes setting up private and public subnets to control traffic flow and ensure secure communication within the infrastructure.

Load Balancer

Load balancing is crucial for even distribution of incoming traffic among instances. The load balancer ensures that each instance in the Auto Scaling group shares the workload, enhancing both performance and fault tolerance.

Minimum Instances, Maximum Instances, Desired Capacity

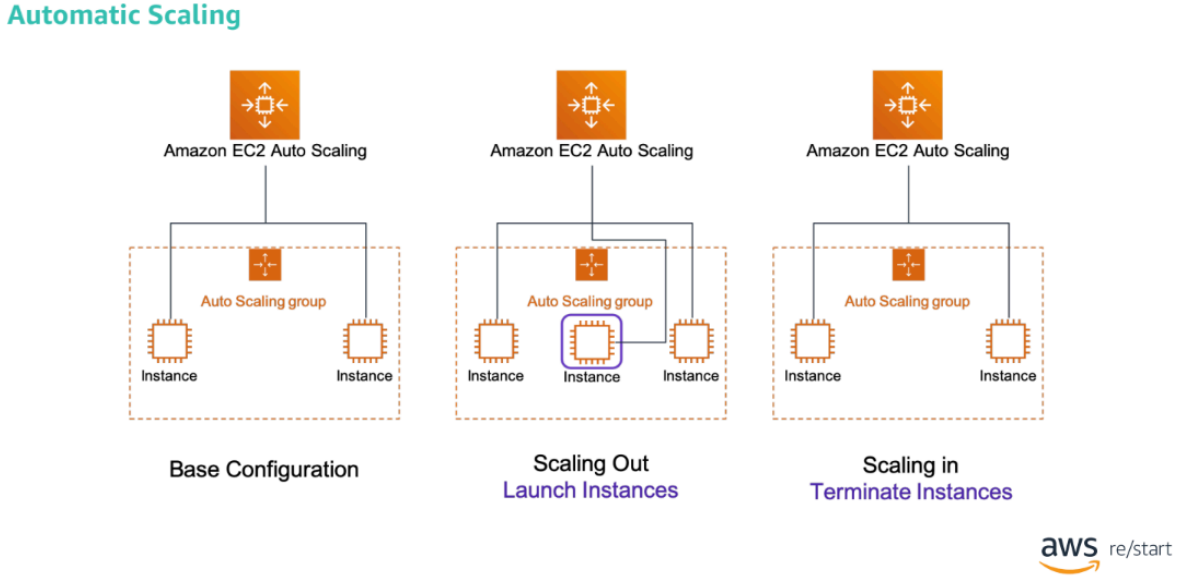
These parameters within the Auto Scaling group dictate the scaling limits. The minimum and maximum instances set boundaries for scaling, while the desired capacity represents the target number of instances, ensuring flexibility in responding to changing demand.

When?

Automatic Scaling Policy

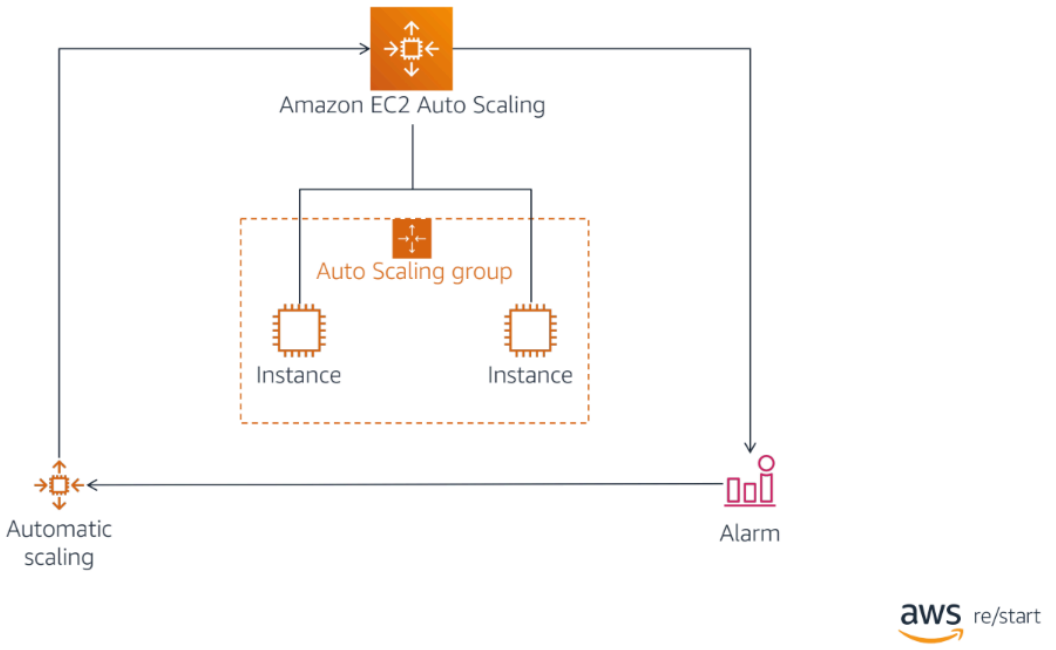
The "When" aspect involves defining policies that dictate when and how scaling actions should occur. These policies can be triggered in various ways:

- **Scheduled Scaling:** Based on predefined schedules, allowing for proactive adjustments.
- **On-Demand Scaling:** Triggered manually, responding to immediate needs.
- **Scale-Out Policy:** Initiates scaling out by adding instances to the Auto Scaling group based on predefined conditions.
- **Scale-In Policy:** Initiates scaling in by removing instances based on specified criteria.



In summary, automatic scaling involves a comprehensive set of components categorized into "What," "Where," and "When." This orchestrated approach ensures that cloud resources are dynamically adjusted, providing efficiency, reliability, and cost-effectiveness in response to fluctuating demand.

Dynamic automatic scaling



Dynamic automatic scaling is a sophisticated approach in the world of cloud computing that empowers systems to adapt to changing workloads seamlessly. Let's demystify this concept for a clearer understanding:

What is Dynamic Automatic Scaling?

In the dynamic landscape of computing, where demands fluctuate, dynamic automatic scaling is a smart strategy for optimizing resource utilization. It involves the automatic adjustment of computational resources based on real-time requirements, ensuring efficiency and performance without manual intervention.

How Does It Work?

Imagine you have a digital platform – maybe a website, app, or service. The demand for your platform is like a rollercoaster, sometimes high and bustling, other times calm. Dynamic automatic scaling enables your system to intelligently respond to these variations:

- **Real-Time Monitoring:** Systems equipped with dynamic automatic scaling continually monitor key performance metrics. This could include factors like CPU usage, network traffic, or the number of users accessing your application.

- **Automated Decision-Making:** When the system detects an increase in demand, it automatically scales up by adding more computational resources. Conversely, during periods of low demand, it scales down by reducing resources. This dynamic adjustment happens in the background, keeping your system responsive and efficient.

Why is it Important?

Optimized Performance: Dynamic automatic scaling ensures that your system always has the right amount of resources to handle the current workload. This responsiveness leads to optimal performance, preventing slowdowns or downtime during peak usage.

Cost Efficiency: By scaling resources up or down based on demand, you're using only what you need. This translates to cost savings, as you're not paying for idle resources during quieter periods.

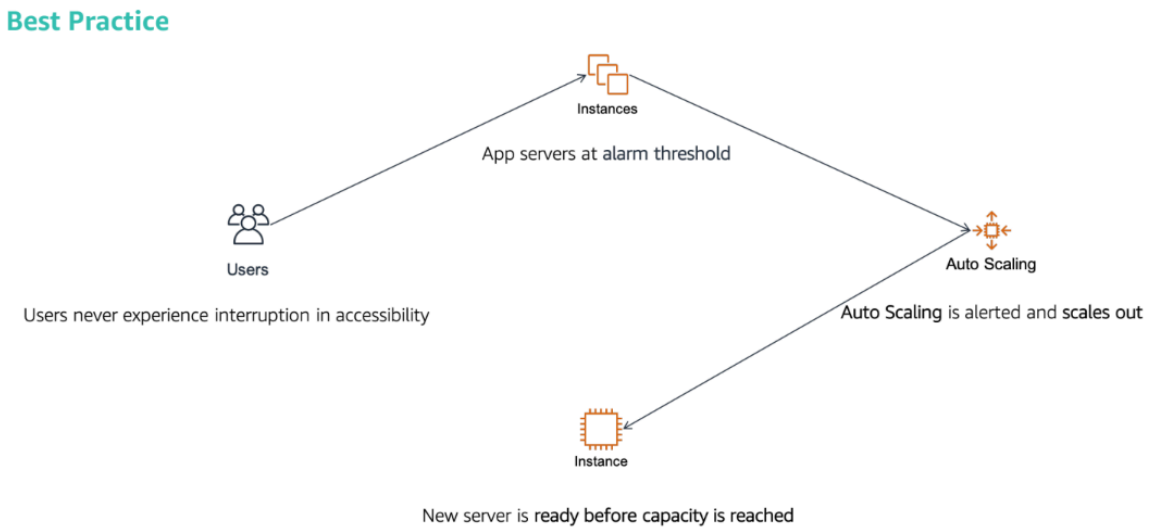
Enhanced Reliability: The ability to adapt to changing conditions enhances the overall reliability of your system. It becomes resilient to unexpected spikes in traffic or usage, providing a seamless experience for users.

Real-World Analogy

Think of dynamic automatic scaling like a smart thermostat in your home. When it's cold outside, the thermostat automatically adjusts the heating to keep the temperature comfortable. Conversely, when it's warm, it scales back to save energy. Similarly, dynamic automatic scaling fine-tunes your digital environment based on the 'temperature' of demand.

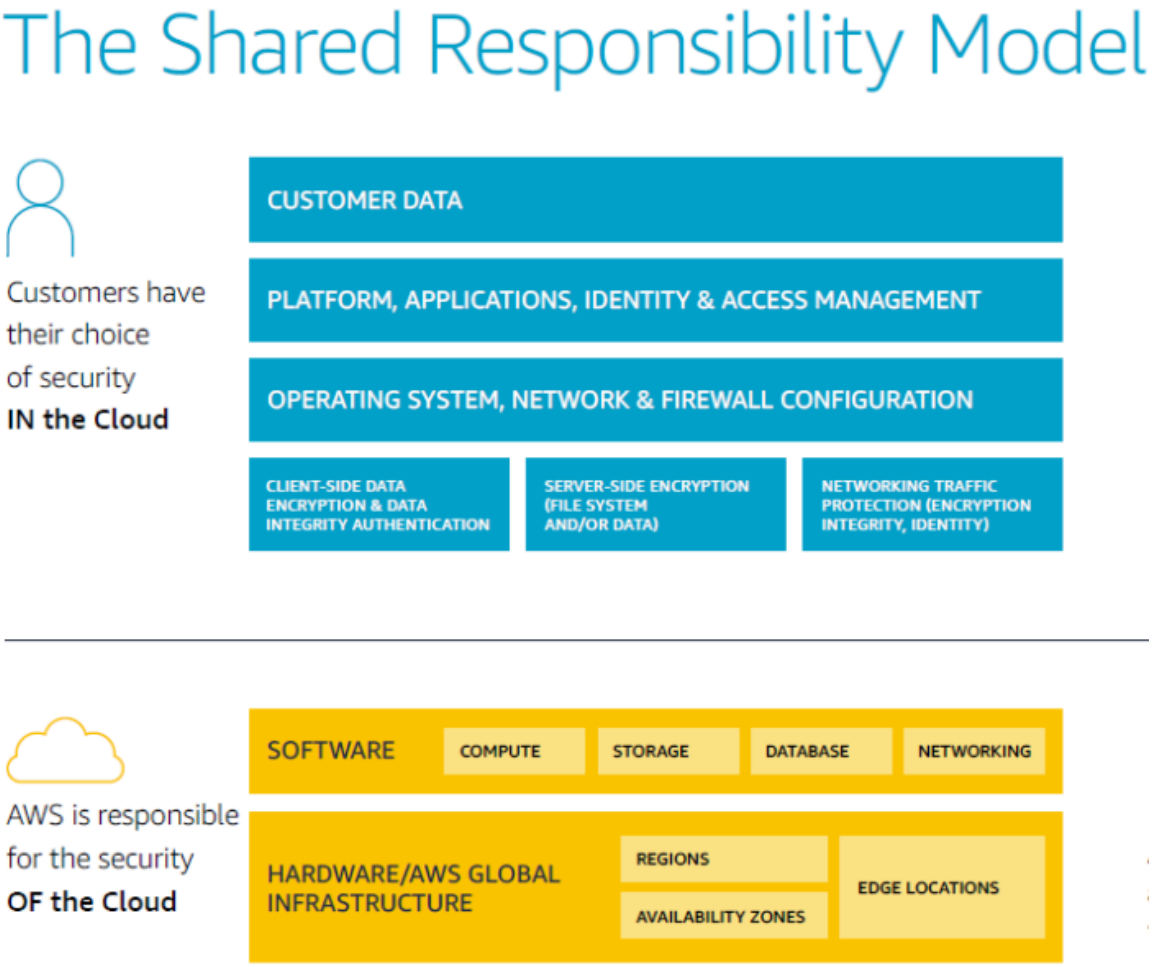
In conclusion, dynamic automatic scaling is a powerful tool in the cloud computing toolbox. It's like having a responsive, intelligent assistant that ensures your digital infrastructure is always operating at its best, efficiently handling both peaks and valleys in demand.

Ensure that your architecture can handle changes



Chapter 26: AWS Shared Responsibility Model

26_AWS Cloud Security - Shared Responsibility - Slides.pdf

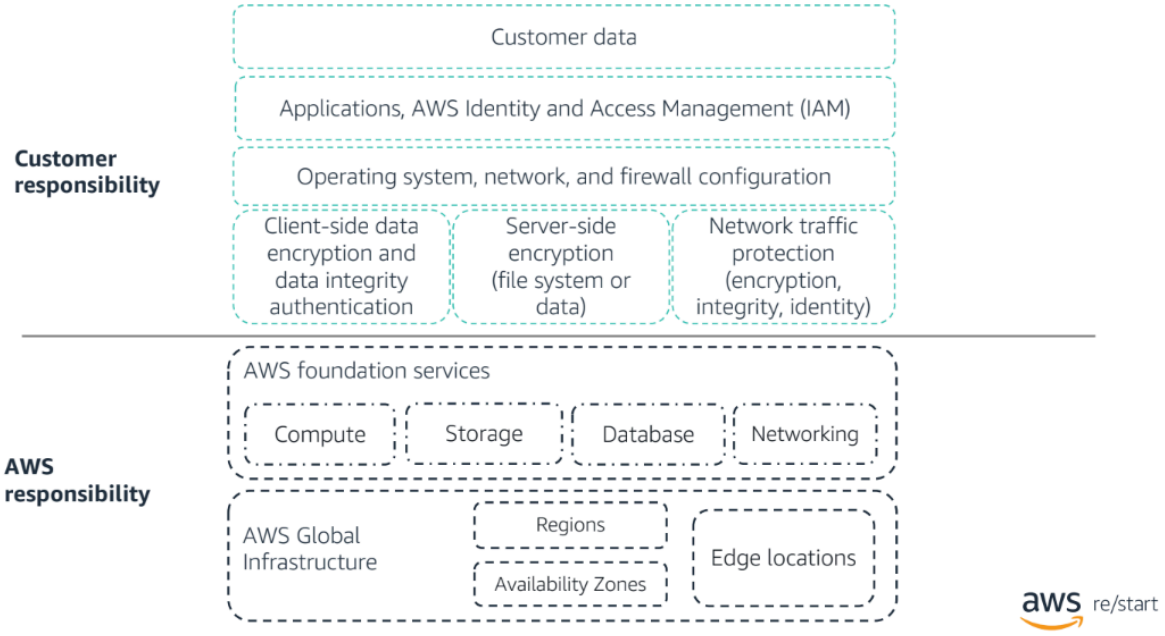


Chapter Overview

You will learn how to:

- Describe AWS Cloud security and the shared responsibility model
- Highlight the security responsibilities of AWS versus the security responsibilities customer responsibilities

Shared responsibility model



The Shared Responsibility Model is a foundational concept in cloud computing that defines the distribution of security responsibilities between cloud service providers (CSPs) and their customers. This model is crucial for understanding how security measures are divided and collaborated upon in a cloud environment.

Understanding the Shared Responsibility Model

Provider and Customer Roles

In a cloud setting, such as with providers like Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform, there is a clear distinction between what the cloud service provider manages and what the customer is responsible for. The responsibilities are distributed across infrastructure and application layers.

Provider Responsibilities

Cloud service providers take charge of securing the infrastructure layer. This includes the physical data centers, networking, and the hypervisor. Providers implement robust security measures to safeguard the hardware and ensure the availability and reliability of their services.

Customer Responsibilities

On the other side of the spectrum, customers are responsible for securing their data and applications. This involves configuring access controls, encrypting sensitive information, and managing user permissions. Customers are also accountable for the security of their applications, including patching and ensuring the proper configuration of virtual machines and databases.

Why it Matters?

- **Clarity in Accountability:** The Shared Responsibility Model provides clarity on who is responsible for what aspects of security. This transparency is crucial for organizations to understand their role in maintaining a secure cloud environment.

- **Collaborative Security:** By acknowledging shared responsibilities, the model encourages collaboration between cloud service providers and customers. This collaboration ensures a more comprehensive and effective security posture.

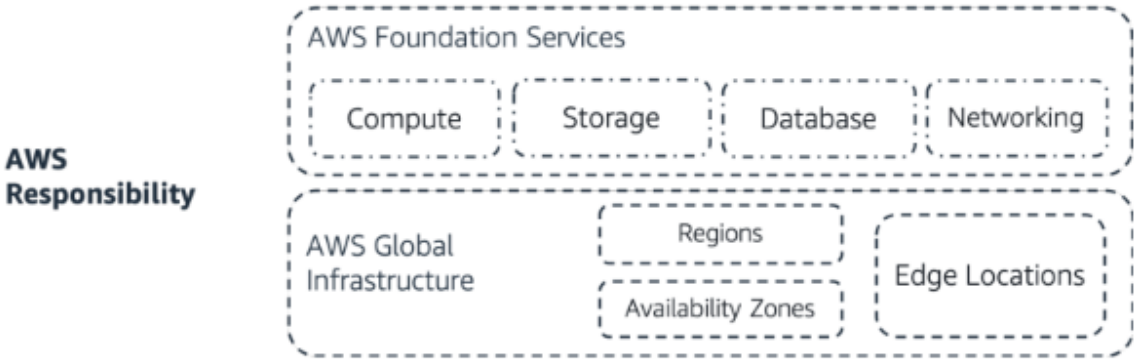
- **Adaptability to Different Cloud Models:** The Shared Responsibility Model is adaptable to various cloud deployment models, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). It provides a consistent framework for understanding security responsibilities regardless of the specific cloud service in use.

Real-World Analogy

Think of the Shared Responsibility Model as renting a house. The landlord (cloud provider) is responsible for maintaining the structural integrity of the building, fixing plumbing issues, and ensuring the roof doesn't leak (infrastructure layer). As a tenant (customer), you are responsible for securing your personal belongings, locking the doors and windows, and maintaining the cleanliness of the rented space (data and application layers).

In conclusion, the Shared Responsibility Model is a fundamental concept that empowers organizations to build a secure cloud environment collaboratively. It establishes a clear understanding of who is accountable for different aspects of security, fostering a shared commitment to maintaining a resilient and protected cloud infrastructure.

AWS security responsibilities: Security **OF** the cloud



AWS security responsibilities encompass a comprehensive framework that ensures the security of the cloud environment. Key components of these responsibilities include:

Physical Security of Data Centers

AWS takes the lead in securing its data centers physically. This involves stringent controls and need-based access to the facilities. Access is carefully managed, restricted, and monitored to prevent unauthorized entry, ensuring the physical integrity of the infrastructure.

Hardware and Software Infrastructure

AWS is responsible for maintaining the security of hardware and software infrastructure. This includes critical processes like storage decommissioning, where storage devices are securely retired to prevent data exposure. Additionally, AWS manages host operating system (OS) access logging and auditing, ensuring a robust security posture at the foundational level.

Network Infrastructure

AWS actively monitors and safeguards the network infrastructure. Intrusion detection systems are employed to identify and respond to any unauthorized or suspicious activities. This ensures the integrity and confidentiality of data in transit within the AWS environment.

Virtualization Infrastructure

The virtualization layer is a critical component in cloud computing. AWS takes charge of securing this infrastructure, with a focus on instance isolation. By ensuring that virtual instances are isolated from each other, AWS prevents potential security risks associated with shared environments, contributing to the overall security of customer data.

Why It Matters?

- **Data Integrity and Confidentiality:** AWS's commitment to physical security, infrastructure management, and network safeguards ensures the integrity and confidentiality of customer data. This is vital for maintaining trust and compliance with various data protection standards.

- **Prevention of Unauthorized Access:** AWS's strict access controls and monitoring mechanisms play a pivotal role in preventing unauthorized access to both physical facilities and the underlying cloud infrastructure. This mitigates the risk of security breaches and data compromise.

- **Foundational Security:** The security measures implemented by AWS in areas like hardware, software, and virtualization form the foundation of a secure cloud environment. This foundational security is essential for building robust, scalable, and resilient solutions on the AWS platform.

In summary, AWS takes on significant security responsibilities in safeguarding the cloud infrastructure. By addressing physical security, hardware and software integrity, network protection, and virtualization isolation, AWS ensures a secure foundation for customers to build and deploy their applications and data in the cloud. This collaborative security model allows organizations to focus on their specific security considerations within the AWS environment, fostering a secure and trusted cloud computing ecosystem.

AWS security responsibilities: Managed services

When it comes to managed services, specific responsibilities are allocated to each party.

AWS Responsibilities

1. OS and Database Patching

AWS takes charge of managing the patching of the operating system (OS) and databases within its managed services. This involves implementing timely updates and security patches to address vulnerabilities, ensuring that the underlying infrastructure remains resilient and protected against potential threats.

2. Firewall Configuration

The configuration of firewalls, a crucial component of network security, is handled by AWS. This responsibility includes setting up and managing firewall rules to control incoming and outgoing network traffic for managed services. Proper firewall configuration is essential for safeguarding against unauthorized access and potential cyber threats.

3. Disaster Recovery

AWS assumes a key role in managing disaster recovery for its services. This involves implementing and overseeing robust strategies to recover data and systems in the event of unforeseen incidents. AWS employs redundant architectures and backup mechanisms to ensure high availability and data resilience.

Customer Responsibilities

1. Logical Access Controls

Customers are responsible for implementing logical access controls within the managed services they utilize. This includes defining and managing user access permissions, ensuring that only authorized individuals have the necessary privileges to interact with the applications and data hosted on AWS.

2. Protect Account Credentials

Safeguarding account credentials is a critical responsibility placed on customers. This involves secure management of usernames, passwords, and access keys associated with AWS accounts. By protecting these credentials, customers contribute significantly to preventing unauthorized access and maintaining the overall security posture of their AWS environments.

3. Why It Matters?

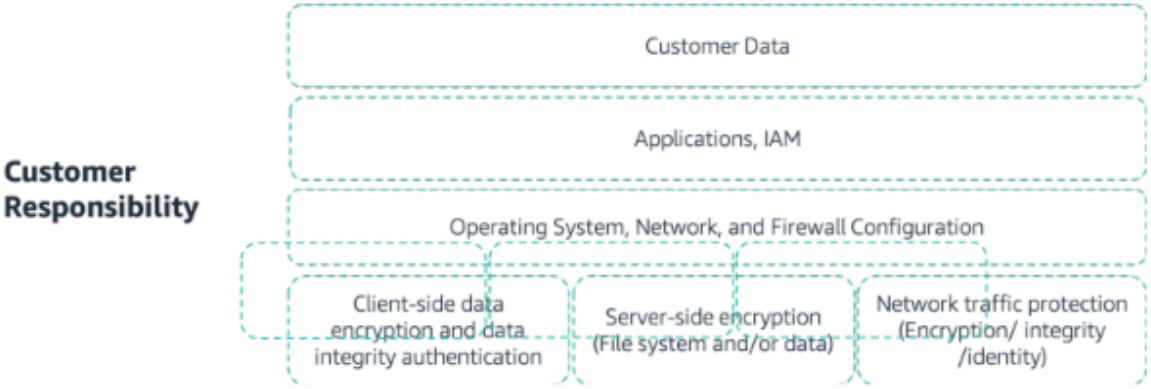
- **Efficient Patch Management:** AWS's management of OS and database patching ensures that customers benefit from efficient and timely updates, reducing the risk of security vulnerabilities. This collaborative effort promotes a secure foundation for hosted applications.

- **Network Security:** Firewall configuration by AWS is vital for creating a secure network environment. It establishes protective barriers against unauthorized access, preventing potential security breaches and ensuring the confidentiality and integrity of data.

- **Reliable Disaster Recovery:** AWS's role in disaster recovery planning and execution contributes to the resilience of managed services. This ensures that customers can rely on AWS for effective recovery strategies, minimizing downtime and data loss in case of emergencies.

- **Customer Control and Accountability:** Customer responsibilities, such as logical access controls and protecting account credentials, empower organizations with control over their specific security requirements. This distributed responsibility model encourages accountability and enables customers to tailor security measures to their unique needs.

Customer security responsibilities: Security **IN** the cloud



Ensuring the security of your data and applications in the cloud is a shared responsibility between customers and cloud service providers, exemplified by the Security IN the Cloud model. While the cloud service provider, in this case, oversees the Security OF the Cloud, customers play a pivotal role in safeguarding their specific assets within the shared environment.

Physical Security of Data Centers

Controlled, Need-Based Access

Customers are responsible for the security of their physical assets within the cloud data centers. This includes controlling access to these resources based on business needs. By defining and managing who has physical access to data center facilities, customers enhance the overall security of their infrastructure.
Hardware and Software Infrastructure:

Storage Decommissioning, Host OS Access Logging, and Auditing

The responsibility of ensuring the secure decommissioning of storage devices falls on the customer. This involves securely retiring storage hardware to prevent potential data exposure. Additionally, customers manage access to the host operating system (OS) by implementing robust logging and auditing practices, enhancing security at the infrastructure level.

Network Infrastructure

Intrusion Detection

Customers are entrusted with the implementation of intrusion detection measures within their network infrastructure. This involves setting up tools and processes to identify and respond to unauthorized or suspicious activities. By actively monitoring network traffic, customers contribute to the early detection of potential security threats.

Virtualization Infrastructure

Instance Isolation

Securing the virtualization layer is a customer responsibility, particularly in terms of instance isolation. Customers ensure that their virtual instances are appropriately isolated from each other to prevent security risks associated with shared environments. This proactive measure enhances the security posture of their virtualized resources.

Why It Matters?

Customized Security Measures

Customer responsibilities in Security IN the Cloud empower organizations to implement security measures tailored to their specific needs. This customization is essential for addressing unique business requirements and compliance standards.

Granular Control Over Access

Managing physical access to data centers provides customers with granular control. This ensures that only authorized personnel can physically interact with the infrastructure, reducing the risk of unauthorized tampering or data breaches.

Data Integrity and Compliance

Customer-managed processes like storage decommissioning and auditing contribute to data integrity and compliance. These practices help organizations adhere to industry regulations and maintain the trust of their stakeholders.

Proactive Threat Detection

By implementing intrusion detection measures, customers play a proactive role in identifying and responding to potential security threats. This contributes to a more resilient and secure network infrastructure.

In conclusion, the Security IN the Cloud model emphasizes the active role customers play in securing their assets within the cloud environment. By addressing physical security, infrastructure management, network monitoring, and virtualization security, customers contribute significantly to the overall security and integrity of their cloud-based resources. This collaborative approach ensures a robust and customized security posture for organizations leveraging cloud services.

27_AWS Cloud Security - IAM - Slides - vILT.pdf

Chapter 27: AWS Identity and Access Management (IAM)

Chapter Overview

You will learn how to:

• Describe AWS Identity and Access Management (IAM)
• Explore the different types of security credentials

What is IAM?

AWS Identity and Access Management (IAM) is a powerful service that enables the centralized management of authentication and access to AWS resources. As a fundamental feature of an AWS account, IAM is provided at no additional charge, offering a comprehensive solution for controlling and securing access to various services within the AWS cloud environment.

Key Features of AWS IAM

Centralized Management

IAM allows users to centrally manage authentication and access control for AWS resources. This means that administrators can efficiently oversee and regulate user access across multiple AWS services from a unified interface.

No Additional Cost

IAM is included as a core feature of an AWS account without incurring additional charges. This makes it a cost-effective and essential component for organizations seeking to implement robust security measures within their AWS infrastructure.

User, Group, and Role Management

IAM provides the capability to create and manage users, groups, and roles within the AWS environment. Users represent individuals, groups allow the grouping of users with similar access requirements, and roles define the set of permissions for entities assuming them.

This structured approach simplifies access management for different entities within an organization.

Policy Application

One of the key functionalities of IAM is the ability to apply policies to users, groups, and roles. Policies are sets of permissions that define what actions are allowed or denied on AWS resources. By associating policies with specific entities, administrators can finely control and tailor access permissions based on the principle of least privilege.

Why IAM Matters?

- **Enhanced Security:** IAM significantly contributes to the overall security posture of AWS environments by allowing organizations to implement robust access controls. This helps in preventing unauthorized access and securing sensitive data and resources.

- **Efficient Administration:** The centralized nature of IAM streamlines the administration of access and authentication. Administrators can efficiently manage user accounts, group memberships, and role assignments through a unified console.

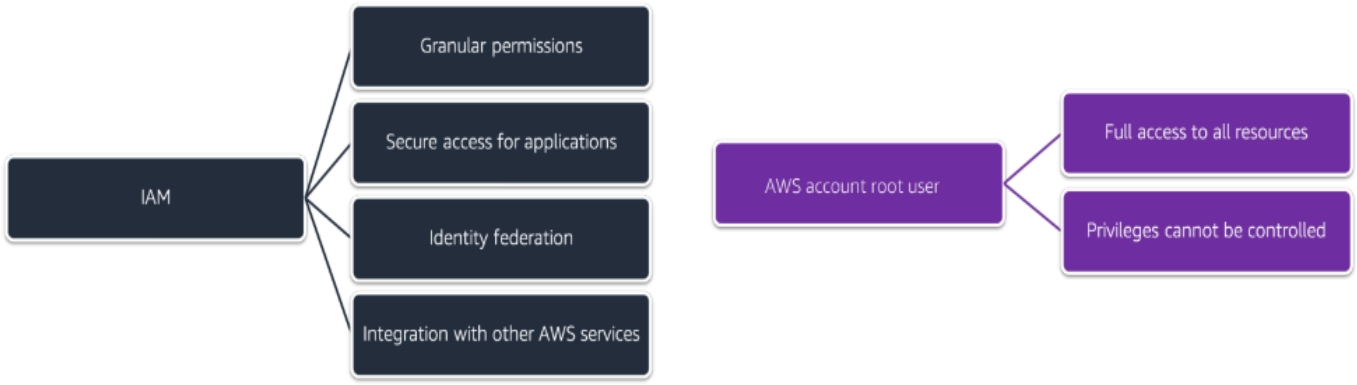
Scalability

IAM is designed to scale with the growth of AWS usage within an organization. As user numbers increase or change, IAM provides a flexible framework for adjusting access permissions, ensuring a scalable and adaptable security model.

Compliance and Auditing

IAM facilitates compliance with security and regulatory requirements by offering granular control over access. Additionally, it provides auditing capabilities, allowing organizations to monitor and review access events for accountability and compliance purposes.

AWS account root user access versus IAM access



When it comes to managing access within AWS, it's crucial to distinguish between the AWS account root user and Identity and Access Management (IAM) users. We will explore the differences between these two types of access, highlighting the importance of using IAM for enhanced security and control.

AWS Account Root User Access

The AWS account root user is created when an AWS account is first established. This user has unrestricted access to all AWS resources and services within the account. The root user is associated with the email address used to create the AWS account and is equipped with full administrative privileges.

However, relying solely on the root user for everyday tasks and administrative activities poses significant security risks. The root user has broad permissions, making it susceptible to misuse or compromise. Furthermore, it lacks the granularity needed for implementing the principle of least privilege, which is a fundamental security best practice.

IAM Access

IAM is a service that allows you to manage access to AWS services securely. IAM enables you to create and manage users, groups, and roles, each with specific permissions and policies tailored to their responsibilities. Unlike the root user, IAM provides a more fine-grained control over access, promoting the principle of least privilege.

Key Differences

a. Granularity of Permissions

- **Root user:** Has full access to all AWS services and resources.
- **IAM user:** Permissions are defined by policies attached to the user, enabling precise control over the actions and resources they can access.

b. Security Best Practices

Root user: Should only be used for initial setup and emergency situations due to its broad access.
IAM user: Designed for day-to-day activities, providing a secure and auditable way to manage access.

c. Auditing and Accountability

Root user: Actions taken by the root user are not as easily traceable to specific individuals.
IAM user: Actions are associated with specific users, allowing for better accountability and auditability.

Best Practices

a. Avoid using the root user for regular tasks.
b. Create IAM users with specific permissions based on job roles.
c. Implement multi-factor authentication (MFA) for added security, especially for the root user.
d. Regularly review and update IAM policies to align with changing business requirements.

The principle of least privilege

The Principle of Least Privilege revolves around granting individuals or systems the minimum level of access required to perform their duties, minimizing the potential impact of accidental or intentional misuse. IAM, as a comprehensive access management service, empowers AWS users to adhere to this principle by providing fine-grained control over permissions.

Best Practices for Implementing the Principle of Least Privilege

1. Delete Account Root User Access Keys

Avoid using root user credentials for routine tasks. Deleting root user access keys reduces the risk of unauthorized access.

2. Create an IAM User

Establish individual IAM users for each person requiring access.
Define permissions based on job responsibilities using IAM policies.

3. Grant Administrator Access Selectively

Assign administrator access cautiously and only to those who truly need it.
Regularly review and adjust permissions to align with evolving roles.

4. Enable Multi-Factor Authentication (MFA)

Add an extra layer of security by enabling MFA for IAM users.
MFA requires users to provide an additional authentication factor, enhancing overall account security.

5. Use IAM Credentials for AWS Interactions

Discourage the use of root user credentials for day-to-day operations.
Leverage IAM credentials for interactions with AWS services, ensuring accountability and traceability.

Types of security credentials

Types of Credentials	Associated with
Email address and password	Associated with AWS account (root)
IAM user name and password	Used for accessing the AWS Management Console.
Access and secret access keys	Typically used with AWS Command Line Interface (AWS CLI) and programmatic requests, like application programming interfaces (APIs) and software development kits (SDKs).
Multi-factor authentication (MFA)	Extra layer of security Can be enabled for AWS account root user and IAM users
Key pairs	Used only for specific AWS services like Amazon EC2.

Understanding the types of credentials and their applications is crucial for maintaining a robust security posture. We will delve into the different types of credentials associated with AWS, ranging from the foundational email address and password to advanced security measures like Multi-Factor Authentication (MFA) and key pairs.

Email Address and Password (AWS Account - Root)

Purpose: This is the primary credential associated with the AWS account's root user.

Application: Grants unrestricted access to all AWS resources and services within the account.

Best Practices: Limited use for initial setup and emergencies; avoid regular use to adhere to the principle of least privilege.

IAM User Name and Password

Purpose: Enables secure access to the AWS Management Console.
Application: IAM users are created with specific permissions and roles, providing fine-grained control over access.
Best Practices: Create IAM users for day-to-day tasks; regularly update passwords; implement least privilege principles.

Access and Secret Access Keys

Purpose: Used for programmatic access, such as AWS CLI, APIs, and SDKs.

Application: Enables automation and integration with AWS services through scripts and applications.

Best Practices: Securely manage and store keys; rotate keys regularly; avoid hardcoding keys in code.

Multi-Factor Authentication (MFA)

Purpose: Adds an extra layer of security on top of passwords.

Application: Can be enabled for both the AWS account root user and IAM users.

Best Practices: Highly recommended for enhanced security; provides an additional authentication step, typically involving a time-based token.

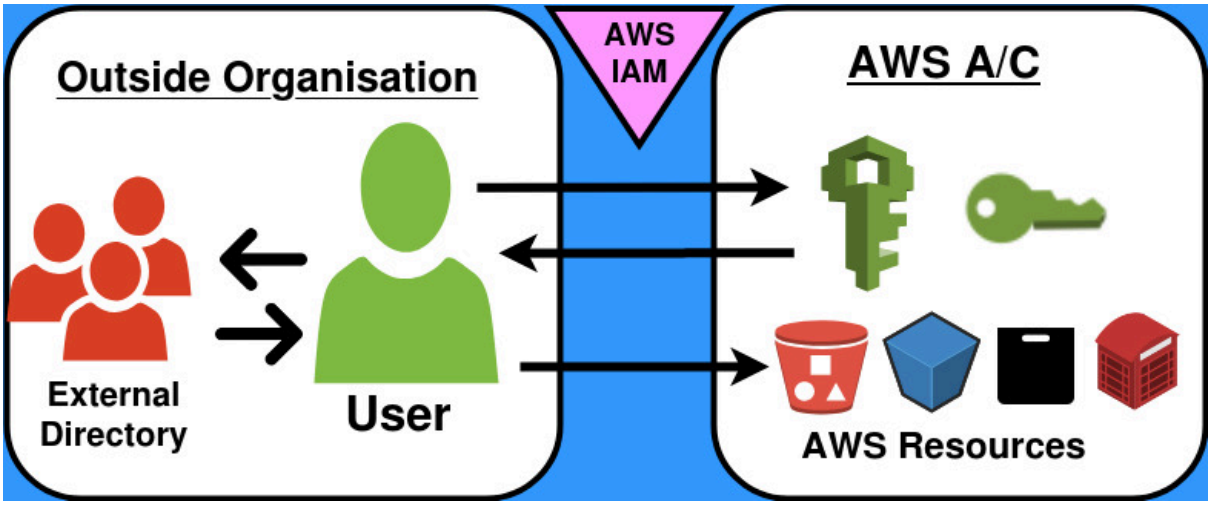
Key Pairs

Purpose: Specific to AWS services like Amazon EC2.

Application: Used for secure access to instances through Secure Shell (SSH) or Windows Remote Desktop Protocol (RDP).

Best Practices: Generate key pairs securely; associate key pairs with EC2 instances during launch; protect private keys.

IAM: Authorization



One key aspect of IAM is authorization, which involves granting permissions to users, allowing them to interact with AWS services. In this article, we'll explore how IAM enables authorization through the creation of IAM policies and the importance of understanding permissions for effective access control.

IAM Authorization Basics

IAM authorization revolves around the concept of assigning permissions to users, determining what AWS resources they can access and the actions they can perform. Permissions are granted by creating IAM policies, which are sets of rules defining the allowed or denied actions on specified resources.

Key Components of IAM Authorization

IAM Policies

IAM policies are the building blocks of authorization in AWS. These policies define permissions by specifying the actions allowed or denied on specific AWS resources. Policies can be attached to IAM users, groups, or roles.

Implicit Deny by Default

A crucial principle in IAM authorization is the concept of implicit deny.

By default, all permissions are implicitly denied unless explicitly granted through IAM policies.
This ensures a secure starting point where users have no access until permissions are intentionally assigned.

Explicit Deny Takes Precedence

If a permission is explicitly denied in an IAM policy, it takes precedence over any other permissions that might allow the action.
This "deny-first" approach enhances security by ensuring that specific actions are prohibited, even if they are allowed by other policies.

Global Nature of IAM

IAM is a global service in AWS, meaning it operates on a global scale and is not confined to specific regions. This global nature has several implications for authorization:

Consistency Across Regions

IAM policies apply uniformly across all AWS regions.
This ensures a consistent access control mechanism regardless of the geographic location of AWS resources.

Centralized Policy Management

Policy management is centralized, making it easier to maintain and enforce access control policies consistently.

Best Practices for IAM Authorization

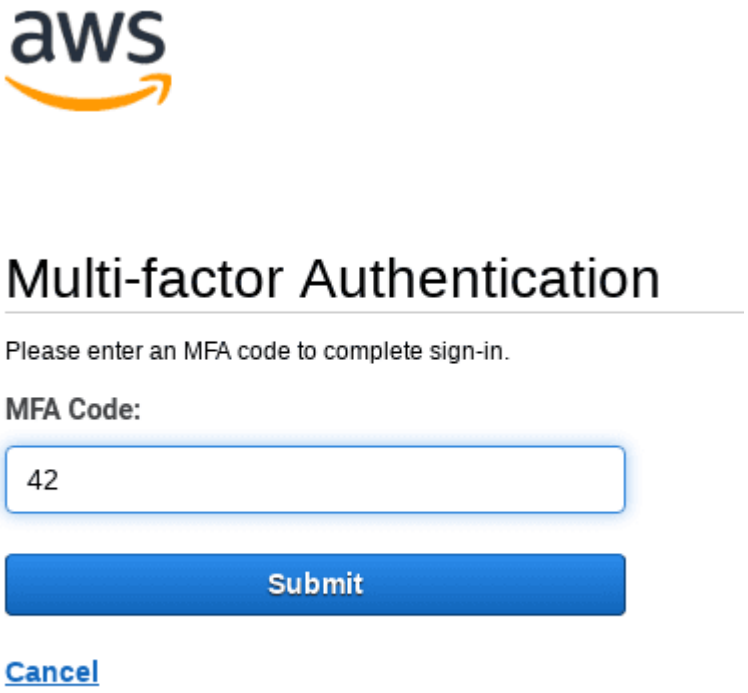
Follow the Principle of Least Privilege

Grant the minimum set of permissions required for users to perform their tasks.
Regularly review and update policies to align with changing roles and responsibilities.

Regularly Audit Permissions

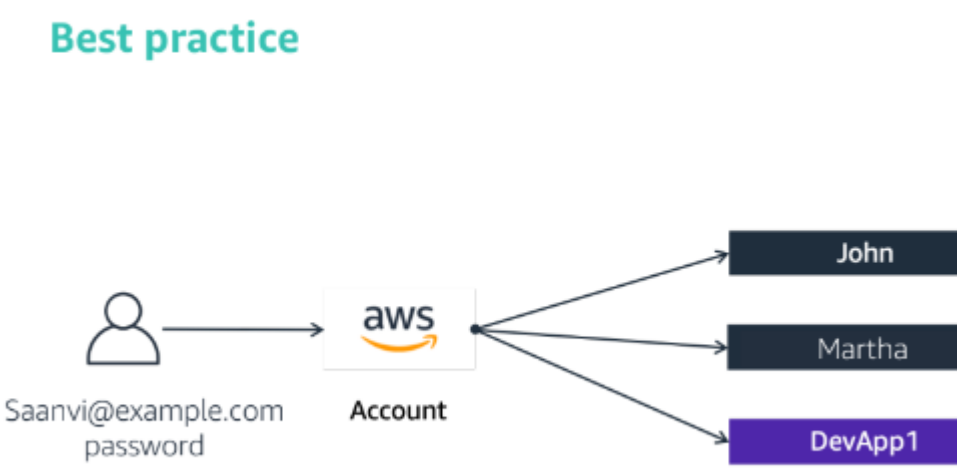
Conduct regular audits of IAM policies to ensure they align with the organization's security and compliance requirements.
Utilize AWS tools like AWS Identity and Access Management Access Analyzer for automated policy analysis.

IAM Multi-Factor Authentication (MFA)



IAM Multi-Factor Authentication (MFA) enhances security in AWS by introducing an additional layer beyond traditional username and password authentication. This extra layer involves a unique authentication code, often time-sensitive, which users must provide along with their credentials to access AWS services. MFA mitigates the risks of credential compromise, aligns with regulatory compliance standards, and ensures secure remote access. Implementing MFA is crucial for organizations managing sensitive data in AWS, providing a robust defense against unauthorized access and potential security threats.

IAM users



IAM users are entities created within AWS to facilitate interaction with its services. We will discover the key aspects of IAM users, emphasizing their significance, security considerations, and best practices for optimal access management.

Key Points about IAM Users

Entity Creation in AWS

IAM users serve as entities that you create within your AWS environment.
These entities are distinct from the AWS account root user and allow for better control and segmentation of access.

Interaction with AWS

IAM users provide a means to interact with AWS services.

They are instrumental in executing actions, whether it's deploying resources, managing configurations, or accessing specific services.

No Default Security Credentials

Unlike the AWS account root user, IAM users do not come with default security credentials. Security credentials, such as access key ID and secret access key, must be explicitly assigned to IAM users to grant them the necessary permissions.

IAM Users Are Not Necessarily People

IAM users are not limited to human users; they can represent applications, services, or other AWS resources. This flexibility allows for a diverse range of use cases, accommodating both human and non-human entities.

Best practices

1. **Create separate IAM user accounts**

One crucial best practice is to avoid relying on the AWS account root user for day-to-day operations. Instead, it is recommended to create separate IAM user accounts, each tailored to specific roles and responsibilities.

2. **Administrative Privileges**

IAM users can be assigned specific permissions through policies.
Creating a separate IAM user account with administrative privileges allows for the implementation of the principle of least privilege, reducing the risk of unauthorized access.

3. **Security and Accountability**

Assigning unique IAM user accounts enhances security by providing a clear audit trail of actions taken within AWS.
This practice improves accountability, as actions are traceable to specific IAM users, facilitating easier troubleshooting and monitoring.

4. **Easy Revocation of Access**

When an IAM user no longer requires certain privileges, access can be easily revoked by modifying the associated policies or by deactivating the IAM user account.
This agility in managing access helps maintain a dynamic and secure environment.

IAM groups

IAM groups play a crucial role in simplifying access management by offering a way to organize and assign permissions to multiple users collectively.

Collection of IAM Users

IAM groups are essentially collections of IAM users. Instead of assigning permissions to individual users one by one, you can streamline this process by grouping users based on their roles or responsibilities. This simplifies the administration of access policies, especially in environments with a large number of users.

Specifying Permissions for the Entire Group

One of the key advantages of using IAM groups is the ability to define permissions for the entire group. Instead of configuring permissions for each user separately, you can attach IAM policies directly to the group, making it efficient to manage access control at scale.

No Default Groups

Unlike some other systems, AWS does not come with default groups. This means that organizations have the flexibility to create IAM groups tailored to their specific needs and structures. Users can be added to these groups based on their roles, simplifying access management.

Groups Cannot Be Nested

It's important to note that IAM groups cannot be nested. Unlike some directory services that allow for hierarchical group structures, IAM groups in AWS operate as standalone entities. Each group is independent, and permissions are applied directly to the group without inheritance from other groups.

A User Can Belong to Multiple Groups

IAM users are not limited to belonging to a single group. In fact, a user can be a member of multiple IAM groups simultaneously. This flexibility allows organizations to accommodate diverse access requirements for users with varying roles and responsibilities.

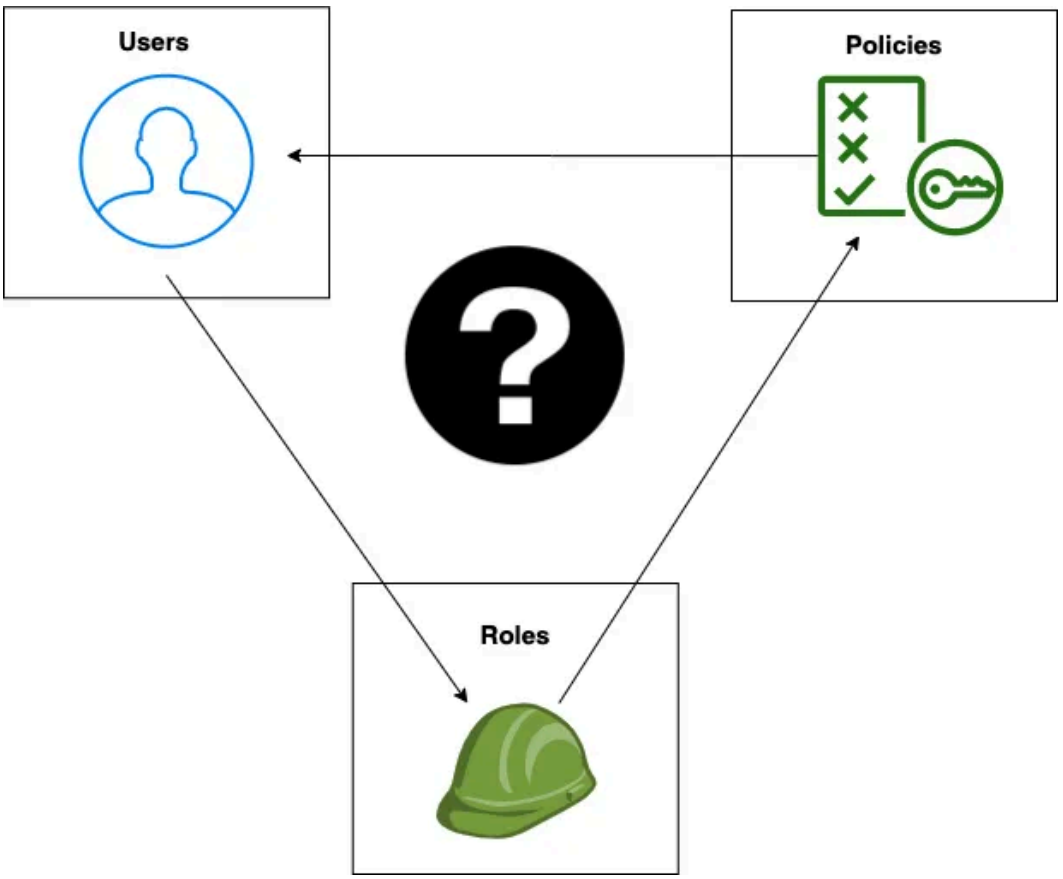
Permissions Defined Using IAM Policies

IAM groups derive their permissions from IAM policies. These policies define what actions users in the group are allowed or denied on specific AWS resources. By crafting granular policies, organizations can adhere to the principle of least privilege, ensuring that users only have the access they need to perform their tasks.

Best Practices

a. Create IAM groups based on roles or responsibilities within your organization.
b. Regularly review and update IAM group memberships and policies to align with changes in personnel or business requirements.
c. Leverage IAM policies to enforce the principle of least privilege, granting users in groups only the permissions necessary for their tasks.

IAM roles



IAM roles, a fundamental component of AWS IAM, offer a robust solution for delegating temporary access to AWS resources, eliminating the reliance on static credentials. We will explore the key features and benefits of IAM roles.

Delegating Access to AWS Resources

IAM roles are designed to delegate access within AWS environments. Instead of assigning permissions directly to individual users or groups, IAM roles provide a flexible mechanism for temporary access. This is particularly valuable in scenarios where users, applications, or services require short-term permissions to perform specific tasks.

Provides Temporary Access

One of the defining features of IAM roles is their ability to grant temporary access. Users or entities assuming an IAM role receive temporary security credentials, which are automatically rotated by AWS. This dynamic approach enhances security by reducing the exposure of long-lived credentials and mitigating the risks associated with compromised access keys.

Eliminates the Need for Static AWS Credentials

IAM roles contribute to a more secure AWS environment by eliminating the need for static AWS credentials. Unlike traditional access methods that involve long-term access keys, IAM roles grant temporary access through the use of short-lived security tokens. This not only enhances security but also simplifies the management of access credentials.

Permissions Defined Using IAM Policies

Similar to IAM users and groups, IAM roles derive their permissions from IAM policies. These policies define the actions users or entities can perform on specific AWS resources. However, what sets IAM roles apart is that these policies are attached directly to the role itself, not to an IAM user or group.

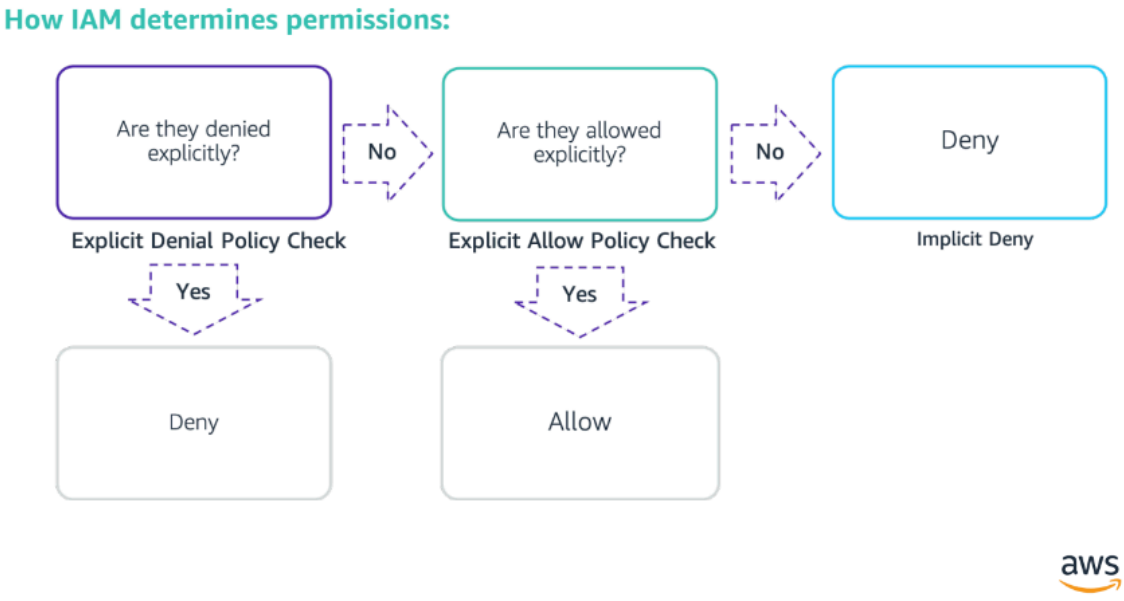
Attached to the Role, Not to an IAM User or Group

IAM role policies are associated directly with the role entity, allowing for a more versatile and scalable approach to access management. Users or services assuming the role inherit the associated permissions, enabling organizations to grant access across accounts or services without compromising security.

Best Practices

a. Use IAM roles for cross-account access and federated identity scenarios.
b. Regularly review and update IAM role policies to align with changing business requirements.
c. Leverage IAM roles for AWS services that support assuming roles, such as AWS Lambda functions or EC2 instances.

IAM permissions



How IAM Determines Permissions?

1. IAM Policies

IAM permissions are primarily defined through policies. IAM policies are JSON documents that explicitly state the actions allowed or denied on specific AWS resources. These policies are attached to IAM identities such as users, groups, or roles.

2. Policy Evaluation

IAM evaluates policies in a specific order:

- **Explicit Deny:** If there's an explicit deny in any policy, that takes precedence, and access is denied.
- **Explicit Allow:** If there's an explicit allow, access is granted.
- **Inherited Allow:** If no explicit allow or deny is found, IAM checks for any inherited allows. For example, if a user is in a group with an attached policy, they inherit the group's permissions.
- **Inherited Deny:** Lastly, if no explicit or inherited allows are found, IAM defaults to deny access.

3. Permissions Boundaries

IAM also considers permission boundaries when evaluating access. A permissions boundary is an advanced feature that sets the maximum permissions an IAM entity (such as a user or role) can have. It acts as an additional layer of control.

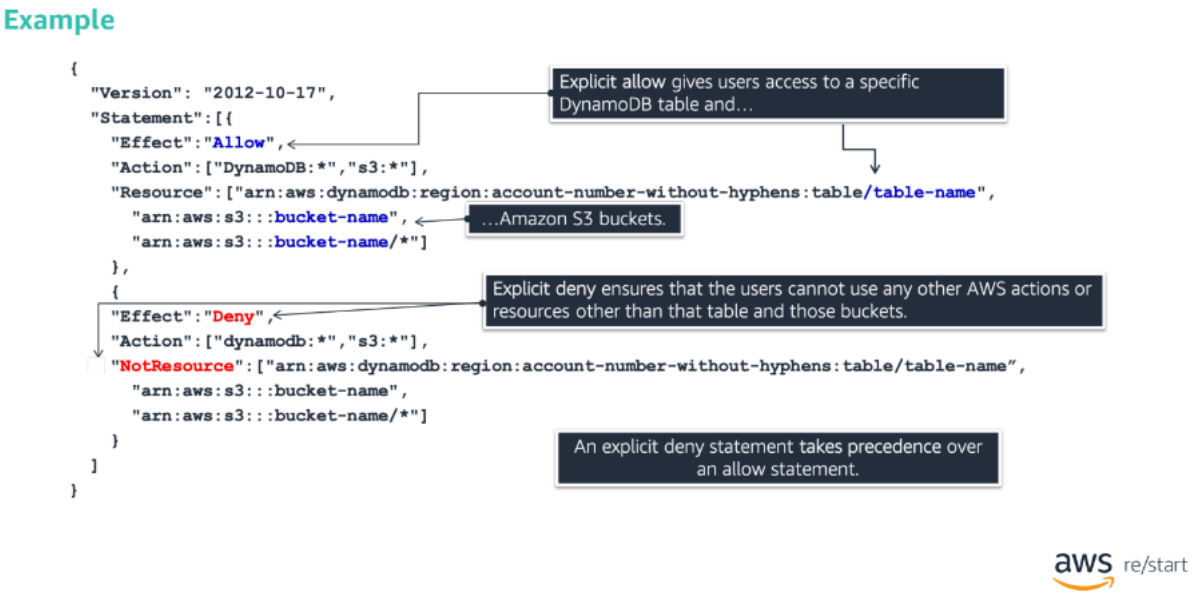
4. Resource Policies

Some AWS resources have their own resource-based policies, specifying who can access the resource and what actions they can perform. IAM permissions are evaluated in conjunction with these resource policies.

5. Trust Relationships

In the case of roles assumed by entities (e.g., EC2 instances or Lambda functions), IAM checks the trust relationship. The entity assuming the role must be trusted by the role's policy to gain access.

Example IAM Policy



Best Practices

1. **Principle of Least Privilege**

Grant users the minimum permissions required to perform their tasks. Avoid overly permissive policies.

2. **Regular Audits**

Periodically review and audit IAM policies to ensure they align with current business requirements and security standards.

3. **Use IAM Roles**

Leverage IAM roles for temporary access, especially for entities like EC2 instances or Lambda functions.

4. **Test Policies**

Utilize AWS IAM policy simulator to test and validate the effectiveness of policies before applying them in a production environment.

Understanding how IAM determines permissions is crucial for maintaining a secure AWS environment. By crafting well-defined policies and adhering to best practices, organizations can effectively control access to their AWS resources and mitigate potential security risks.

IAM policies

Identity and Access Management (IAM) policies serve as formal declarations of permissions, enabling organizations to define and control access to their AWS resources.

Formal Statements of Permissions

IAM policies are the foundation of access control in AWS. They are formal documents that articulate one or more permissions, specifying the actions that an IAM entity (such as a user, group, or role) is authorized to perform. Policies are written in JSON format, making them human-readable and easy to understand.

Attachment to IAM Entities

IAM policies can be attached to various IAM entities, including users, groups, and roles. This attachment is what grants or denies permissions to these entities. By associating policies with specific IAM entities, organizations can precisely tailor access control based on individual roles and responsibilities.

Fine-Grained Access Control

Policies play a pivotal role in achieving fine-grained access control. They define the boundaries of permissible actions, ensuring that IAM entities only have the access required

to perform their designated tasks. This granularity aligns with the security best practice known as the principle of least privilege.

Single Policy, Multiple Entities

A single IAM policy is not confined to a specific IAM entity. Instead, a policy can be attached to multiple entities. This feature promotes reusability and simplifies policy management, especially when the same set of permissions needs to be granted to different entities.

Multiple Policies for a Single Entity

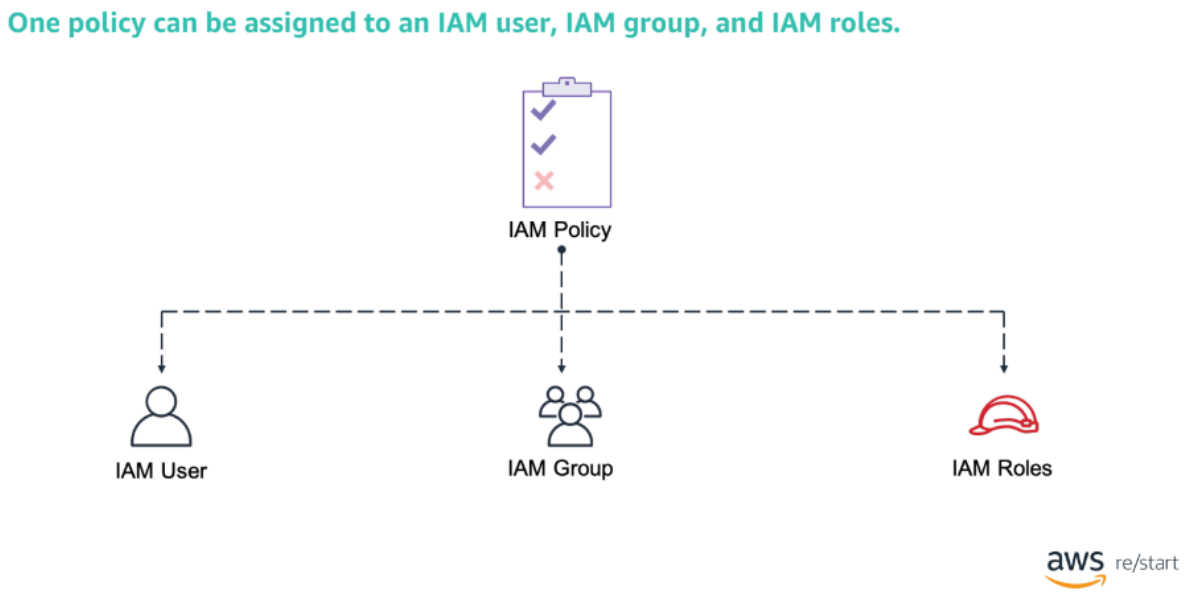
Conversely, a single IAM entity can have multiple policies attached to it. This flexibility allows organizations to layer permissions, accommodating complex access requirements for users, groups, or roles with diverse responsibilities.

Best Practice

Use Groups for Efficient Policy Management

When the same policy needs to be attached to multiple IAM users, it's recommended to group these users together and attach the policy to the group instead. This approach enhances manageability, as changes to the policy are reflected across all group members, streamlining the administration of access control.

IAM: Policy assignment



Policy Assignment to IAM Users

IAM policies can be directly attached to IAM users, specifying the actions they are allowed or denied on AWS resources. This one-to-one association ensures that each user has a personalized set of permissions tailored to their responsibilities. Policy assignment at the user level provides granularity and flexibility in access management.

Policy Assignment to IAM Groups

IAM policies can also be assigned to IAM groups, offering a streamlined approach to managing permissions for multiple users who share similar roles or responsibilities. By attaching policies to groups, administrators can easily update and enforce access controls across an entire group, simplifying the overall access management process.

Policy Assignment to IAM Roles

IAM roles are entities that define a set of permissions for making AWS service requests. Policies can be attached to IAM roles, and users or AWS services can assume these roles to temporarily acquire the associated permissions. Policy assignment to roles is especially useful for temporary access scenarios, such as during application development or cross-account access.

Policy Versioning and Evaluation

IAM policy assignment supports versioning, allowing administrators to manage policy changes over time. This ensures that modifications to policies are traceable, and the system

can evaluate which version of a policy applies to a particular user, group, or role. This feature enhances security and auditability within the AWS environment.

Peter's part (13 chapters)

28_AWS Cloud Security - Trusted Advisor - Slides.pdf

PURPOSE : Abide by Best Practices

AWS Trusted Advisor is a recommendation tool that provides real-time guidance to help you provision your resources following AWS best practices and to maximize efficiency.

BENEFITS

- **Cost Optimization:**
  - Identify and eliminate underutilized resources, reducing unnecessary costs.
  - Receive recommendations to make your AWS spending more efficient.
- **Performance Improvement:**

- Optimize the performance of your resources by following best practices.
- Enhance the overall efficiency and responsiveness of your applications.

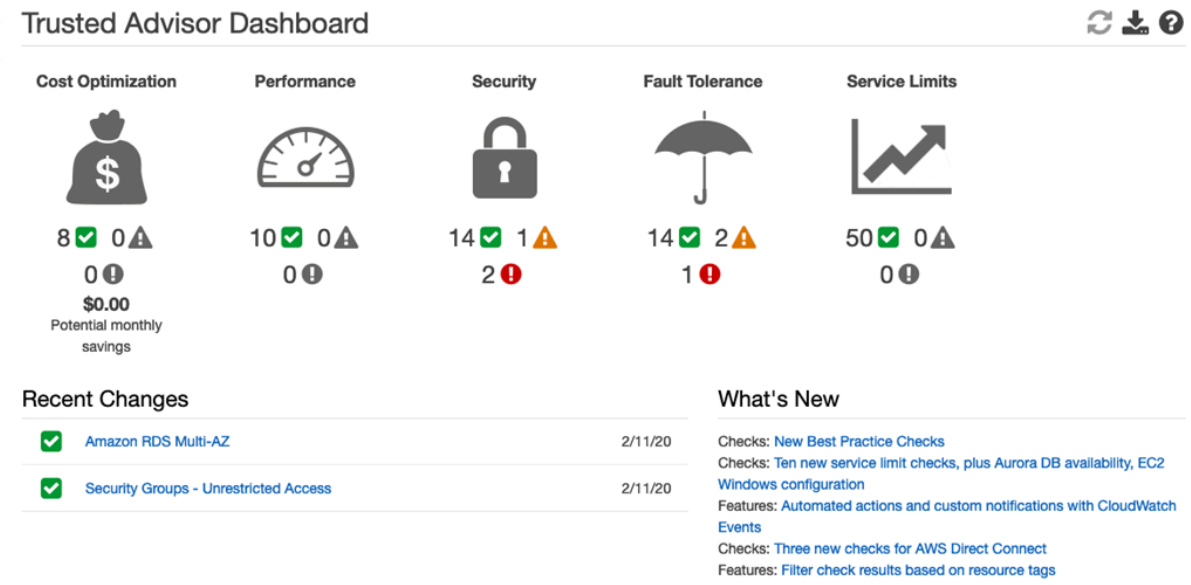
· **Security Enhancement:**
- Receive guidance on securing your AWS resources and adhering to best security practices.
- Identify and address potential security vulnerabilities.

· **Fault Tolerance:**
- Improve the resilience of your applications by following fault-tolerant design principles.
- Ensure that your infrastructure is distributed across multiple availability zones.

· **Service Limit Monitoring:**
- Stay aware of your AWS service limits and get recommendations to request limit increases when needed.

To use AWS Trusted Advisor, you'll need an AWS Support plan.

- The AWS Basic Support and Developer Support Plan gives access to basic security checks and checks on all service quotas.
- AWS Business and Enterprise customers have full access to all AWS Trusted Advisor checks.



HOW and WHAT

1)COST OPTIMIZATION

- **Idle Load Balancers:** Identifies unused Elastic Load Balancers.

- **Underutilized Amazon EBS Volumes:** Highlights EBS volumes that are not fully utilized.
- **Unassociated Elastic IP Addresses:** Flags Elastic IP addresses that are not associated with an EC2 instance.
- **Idle Amazon RDS DB Instances:** Identifies RDS instances that are not fully utilized.

2)PERFORMANCE

- **High Utilization Amazon EC2 Instances:** Flags instances with high CPU or network usage.
- **Amazon RDS Idle DB Instances:** Identifies RDS instances with low utilization.
- **Load Balancer Optimization:** Recommends adjustments to your load balancers for better performance.

3)SECURITY

- **Security Group Changes:** Alerts about recent changes to your security groups.
- **IAM Use:** Recommends best practices for Identity and Access Management (IAM).
- **Amazon S3 Bucket Permissions:** Flags publicly accessible S3 buckets.

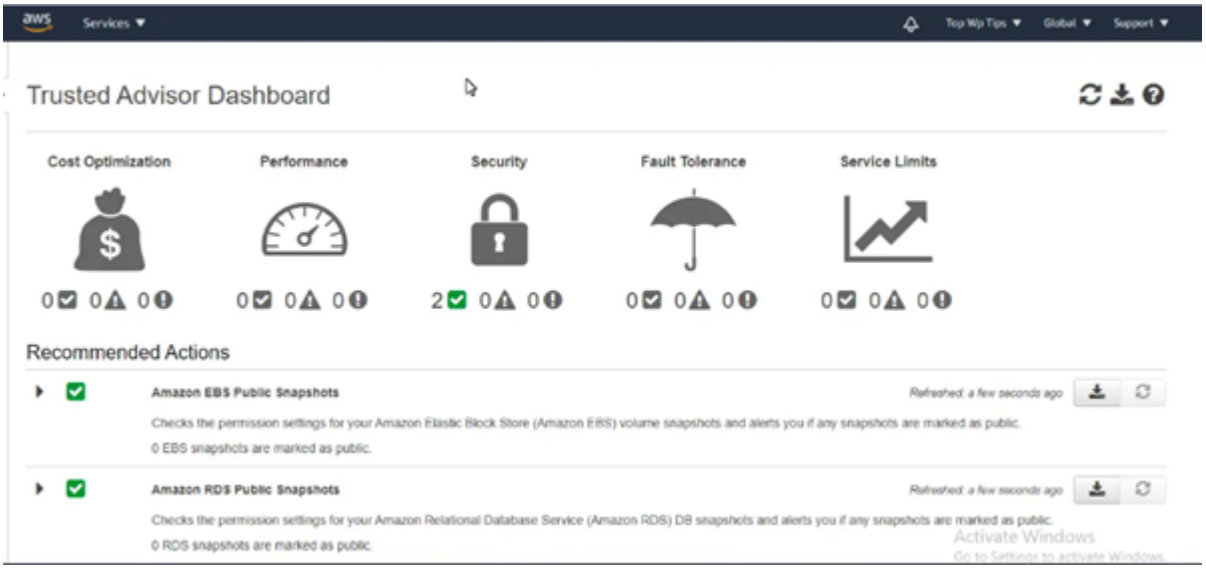
4)FAULT TOLERANCE

- **Amazon RDS Backups:** Ensures that automated backups are enabled for RDS instances.
- **Amazon EBS Snapshots:** Checks for recent EBS snapshots.
- **Amazon EC2 Availability Zones:** Verifies the distribution of instances across multiple availability zones.

5)SERVICE LIMITS

- **API Gateway Regional Endpoints:** Recommends regional endpoints for improved availability.
- **Lambda Function Concurrency:** Checks concurrency limits for AWS Lambda functions.

Auto Scaling Groups: Assesses the limits of Auto Scaling groups.



Getting Alerted on New Notifications:

To receive alerts on new Trusted Advisor notifications, you can set up CloudWatch Alarms. Here's a general guide on how to do this:

- **Open the AWS Management Console:**
- Navigate to the CloudWatch service.
- **Create a New Alarm:**
- Choose "Alarms" from the left-hand menu and click "Create Alarm."
- **Select Trusted Advisor Metrics:**
- Choose the "Trusted Advisor" namespace.
- **Choose a Metric:**
- Select the specific Trusted Advisor metric you want to monitor, such as "Action Recommendations."
- **Configure Conditions:**
- Set up conditions for the alarm based on your preferences (e.g., threshold values).
- **Configure Actions:**
- Define the actions to take when the alarm state changes, such as sending notifications via SNS (Simple Notification Service).
- **Review and Create:**
- Review your settings and click "Create Alarm."

With this setup, you'll receive notifications whenever Trusted Advisor identifies new recommendations or changes in your AWS environment. Adjust the alarm settings based on your specific needs and preferences.

HANDS ON EXAMPLE

Scenario:

Imagine you're a Solutions Architect responsible for a multi-tier web application hosted on AWS. Your goal is to strengthen the security posture of the infrastructure by implementing Trusted Advisor recommendations.

Steps:

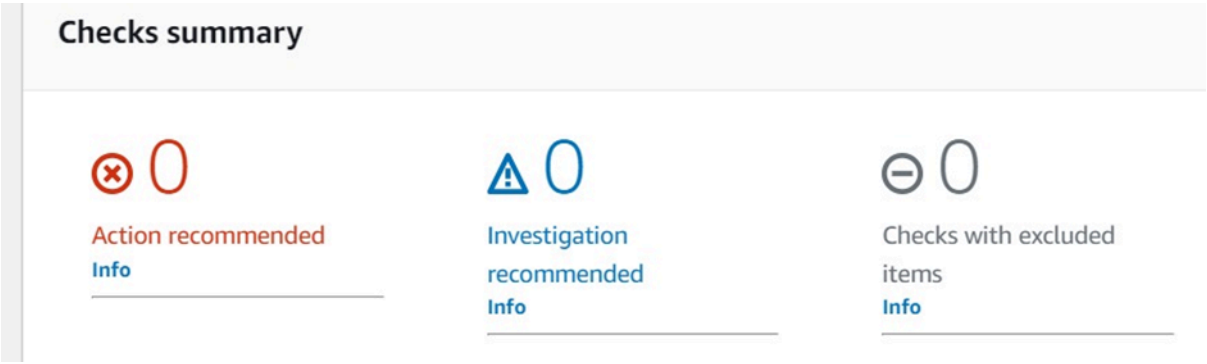
- **Accessing Trusted Advisor:**
- Navigate to the AWS Management Console.
- Open the Trusted Advisor service.
- **Reviewing Security Recommendations:**
- In the Trusted Advisor dashboard, select the "Security" category.
- Examine the security recommendations provided by Trusted Advisor.
- **Addressing IAM Best Practices:**
- Focus on Identity and Access Management (IAM) recommendations.
- Implement best practices like enforcing MFA, reviewing user permissions, and rotating access keys.
- **Enhancing Network Security:**
- Move to the "Network & Security" section.
- Implement recommendations related to network security groups, ensuring only necessary ports are open.
- **Securing Data at Rest:**
- Check for recommendations related to securing data at rest (e.g., encrypted Amazon EBS volumes).
- Implement encryption for any unencrypted volumes.
- **Monitoring Security Groups:**
- Utilize Trusted Advisor's insights on Security Groups.

- Review and modify security group rules to adhere to the principle of least privilege.
- **Enabling AWS CloudTrail:**
- Investigate Trusted Advisor's suggestions on AWS CloudTrail.
- Enable CloudTrail to capture API activity and enhance auditability.
- **Reviewing VPC Configurations:**
- Explore Trusted Advisor's VPC-related recommendations.
- Address any issues related to misconfigured VPCs or subnets.
- **Implementing Trusted Advisor Suggestions:**
- Go through each security recommendation provided by Trusted Advisor.
- Implement the suggestions systematically, ensuring a step-by-step improvement in the security posture.
- **Verifying Changes:**
- Regularly verify and audit the changes made based on Trusted Advisor recommendations.
- Confirm that the security enhancements align with your organization's policies and compliance requirements.

Benefits:

By following these steps, you're actively using AWS Trusted Advisor to strengthen the security architecture of the AWS environment. This hands-on example ensures that your infrastructure aligns with AWS best practices, enhancing its overall security and compliance.

Dashboard changes every time and is dependant on the view : account or organization



QUESTIONS AND ANSWERS

1. Cost Optimization:

- *Question:* Your organization is looking to reduce costs on AWS. Trusted Advisor recommends identifying and terminating idle load balancers. How would you use this recommendation to optimize costs, and what steps would you take to implement it?
- *Answer:* To optimize costs, you would follow Trusted Advisor's recommendation to identify and terminate idle load balancers. This involves reviewing your Elastic Load Balancers in the AWS Management Console, identifying those that are not actively serving traffic, and then terminating or downsizing them accordingly.

2. Performance Improvement:

- *Question:* How can Trusted Advisor help in optimizing the performance of an Amazon RDS database instance, and what specific recommendations might it provide in this context?
- *Answer:* Trusted Advisor can help optimize the performance of an Amazon RDS database instance by recommending adjustments to its configuration, such as increasing the allocated storage, upgrading the instance type for better performance, or adjusting parameters for optimal operation.

3. Security Enhancement:

- *Question:* Trusted Advisor flags an Amazon S3 bucket with public access. Explain the potential security risks associated with public S3 buckets, and how would you address this recommendation to enhance security?
- *Answer:* Publicly accessible S3 buckets pose security risks as they can be accessed by anyone on the internet. To enhance security, you would follow Trusted Advisor's recommendation to review and adjust the bucket permissions, ensuring that only

authorized users or services have access to the data.

4. Fault Tolerance:

- *Question:* In the context of fault tolerance, how does Trusted Advisor evaluate the distribution of Amazon EC2 instances across multiple availability zones, and what actions might it recommend to enhance fault tolerance?
- *Answer:* Trusted Advisor evaluates fault tolerance by checking the distribution of Amazon EC2 instances across multiple availability zones. If the distribution is not balanced, it may recommend adjusting Auto Scaling group configurations to ensure instances are spread across different availability zones for increased resilience.

5. Service Limit Monitoring:

- *Question:* Trusted Advisor alerts you that your Auto Scaling group is approaching its service limits. Describe the potential implications of hitting these limits and the steps you would take to address this situation.
- *Answer:* Hitting service limits can impact the scalability of your infrastructure. To address this, you would follow Trusted Advisor's recommendation to request limit increases for the affected resources, such as Auto Scaling groups. This involves submitting a limit increase request through the AWS Support Center.

6. CloudWatch Alarms Integration:

- *Question:* You want to receive real-time notifications when Trusted Advisor identifies new recommendations. How can you integrate Trusted Advisor with CloudWatch Alarms to achieve this, and what metrics might you monitor?
- *Answer:* To receive real-time notifications, you can integrate Trusted Advisor with CloudWatch Alarms by selecting the relevant Trusted Advisor metric (e.g., "Action Recommendations"). You would then set up CloudWatch Alarms with specific threshold values, actions, and notifications through Simple Notification Service (SNS) to alert you when Trusted Advisor identifies new recommendations.

29_AWS Cloud Security - AWS CloudTrail - Slides.pdf

PURPOSE : WHO DID WHAT ?

AWS CloudTrail is an essential service that records activity within an AWS account, enabling operational and risk auditing, governance, and compliance. It captures actions taken by users, roles, or AWS services as events, including those from the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs. The result is that we have a detailed history of what happened in our cloud infrastructure. Very handy for auditing purposes. CloudTrail is active in an AWS account upon creation, and it records event activity as they occur.

HOW ? Events, Trails, Logs, Lake, integrations

1. **Events :**
   Managed Events : what happened with a resource : CRUD of S3 bucket

Data Events : what happened within a resource : CRUD files within an S3 bucket

Event History : Managed events within 1 region are searchable for 90 days.

If you need more information, then you need to create a trail or an event data store.

2. **Trails :**

A Trail is a chronological record of activities and is multi-region by default:

A trail is a configuration that enables the delivery of events to an Amazon S3 bucket specified by the user. Question? : Does the creation of a trail impact the event history ?

A trail can be multi-region, multi-account

3. **Logs :**

Event History : Event Delivery and Analysis: CloudTrail publishes log files multiple times an hour, containing API calls from services in the account that support CloudTrail. These log files enable visibility into user activity and can be used for security analysis, resource change tracking, and compliance auditing. Logs are kept for 90 days as standard.

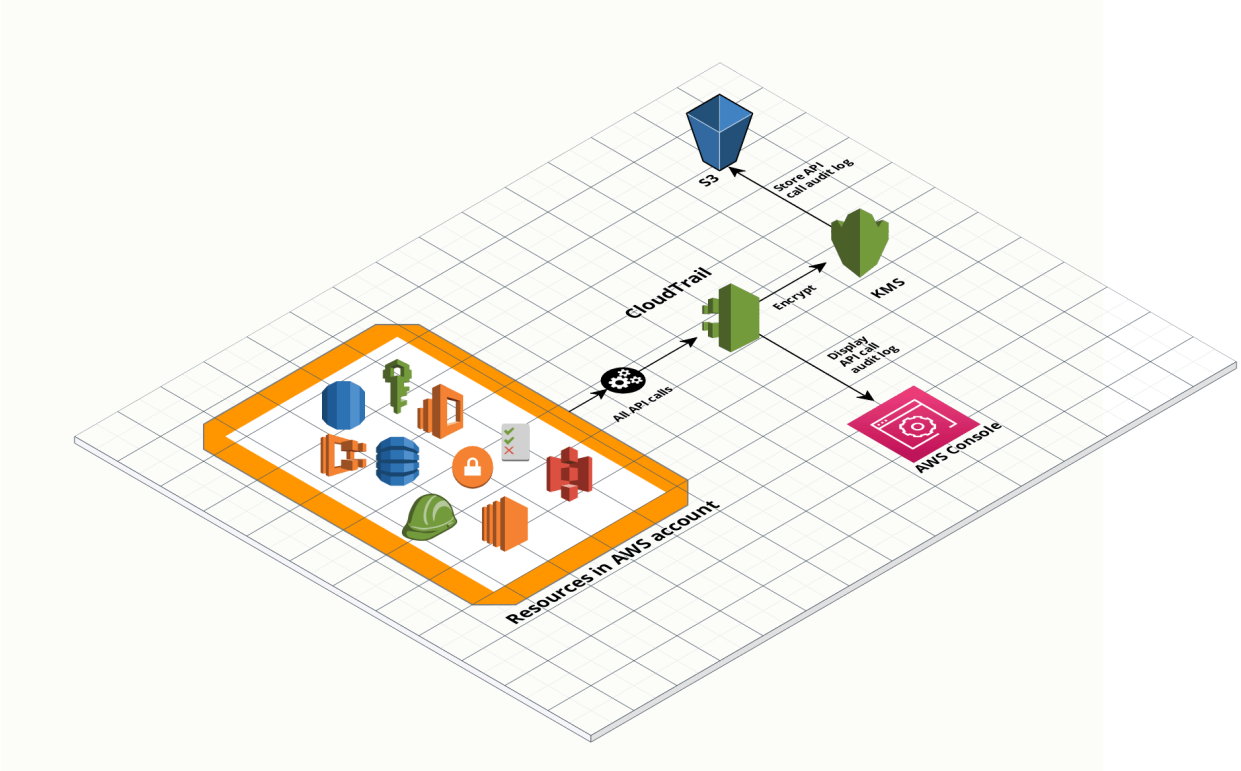
4. **CloudTrail Lake**:

AWS CloudTrail Lake is a managed data lake for capturing, storing, accessing, and analyzing user and API activity on AWS for audit and security purposes. It converts existing events in row-based JSON format to Apache ORC, and users pay for ingestion and storage based on the amount of uncompressed data ingested during the month

5. **Integrations :**

Example : with Amazon S3 and CloudWatch: The log of CloudTrail is delivered to an Amazon S3 bucket and can also be delivered to CloudWatch logs and CloudWatch events for further analysis and monitoring

Example : CloudTrail can log security issues, but we can also secure the security logs.



In summary, AWS CloudTrail works by capturing and recording activity as events, delivering these events to specified Amazon S3 buckets, and providing visibility into user activity for security analysis, resource change tracking, and compliance auditing. Additionally, CloudTrail Lake offers a managed data lake for capturing and analyzing user and API activity on AWS.
CloudTrail supports Multi-Regions tracking and tracking from multiple accounts. So the result is a centralized log.

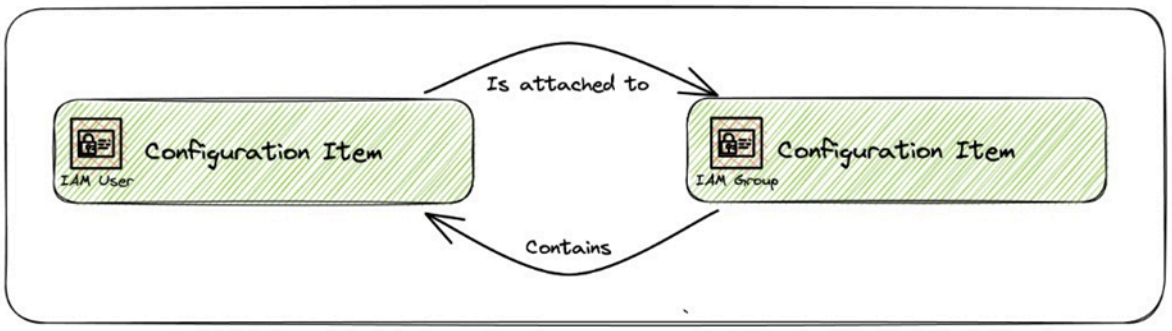
30_AWS Cloud Security - AWS Config - Slides.pdf

AWS Config is **a service that helps you assess, audit, and evaluate the configurations and relationships of your resources on AWS, on premises, and on other clouds.**

WHAT ?

- CUD Actions on Resources
- See the relationship between services
- Take snapshots : A snapshot is
- Store snapshots

AWS Service	Resource Type Value	Relationship	Related Resource
Amazon RDS	AWS::RDS::DBInstance	Is associated with	EC2 Security Group
			RDS DB Security Group
			RDS DB Subnet Group
	AWS::RDS::DBSecurity Group	Is associated with	EC2 Security Group
	AWS::RDS::DBCluster	Contains	RDS DB Instance



HOW ? RULES AND CONFORMANCE PACKS

- Specify Rules : Managed Rules (300 presets) and Custom Rules
  e.g. : Security best practices for service x
  e.g. : Operational best practices for service x
- A collection of Rules is called a conformance pack
- Use packs or rules per region and account or use AWS Organizations to deploy conformance packs in your entire organization

TIP : AGGREGATORS

- Is an AWS Config Resource type that collects multiple configuration data (from multiple accounts and regions into a single account and region
- A collection of Rules is called a conformance pack

WHY ? : USE CASES

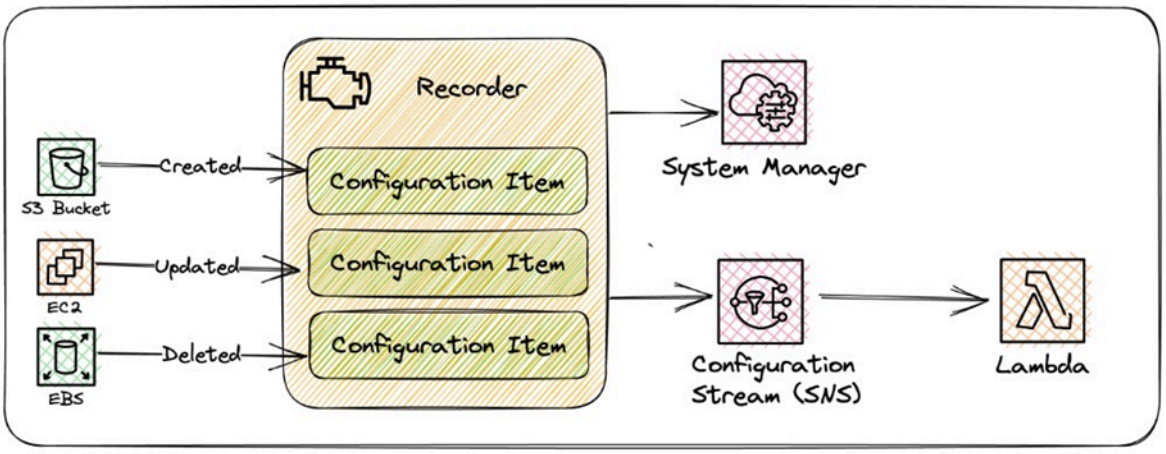
- Resource management
- Auditing and compliance
- Managing and troubleshooting configuration changes
- Security analysis

INTEGRATIONS

- S3 for storing snapshots
- SNS to send configuration stream notifications
- As a feed for Security Hub, Audit Manager, System Manager, etc.

CONFIGURATION

- Items : snapshot of a resource
- Recorder : responsible for storing all configuration items
- Snapshot : complete picture of all supported resources and configurations
- Stream : Overall Flow



31 AWS Cloud Security - Day1 - Slides.pdf

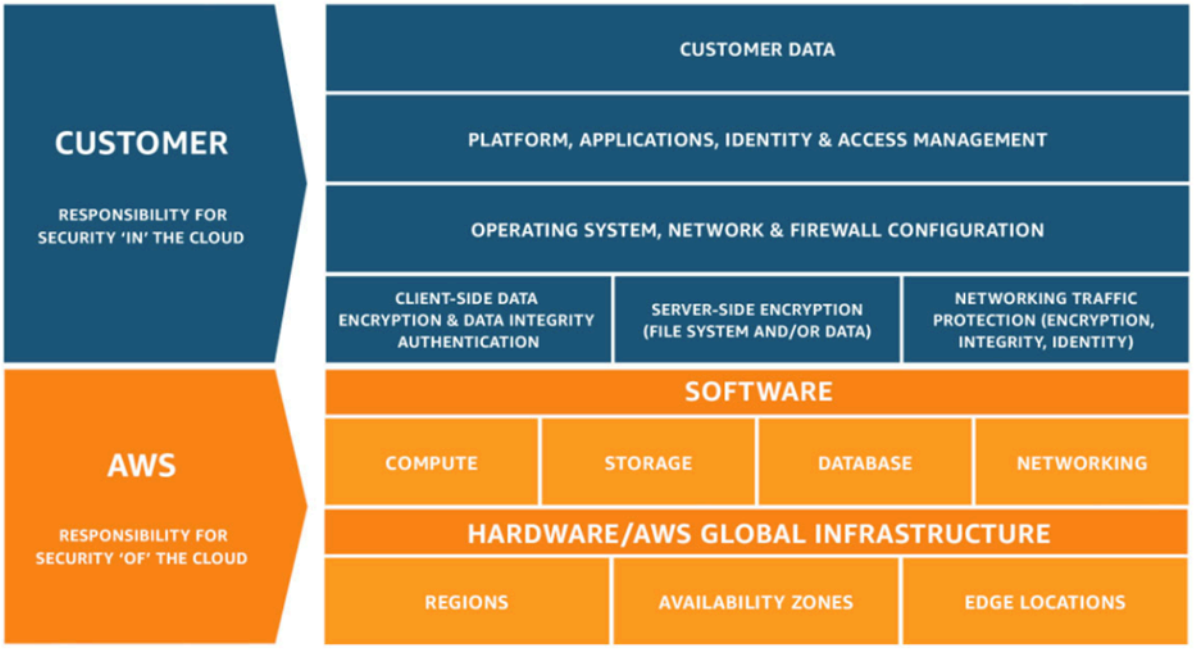
RECAP

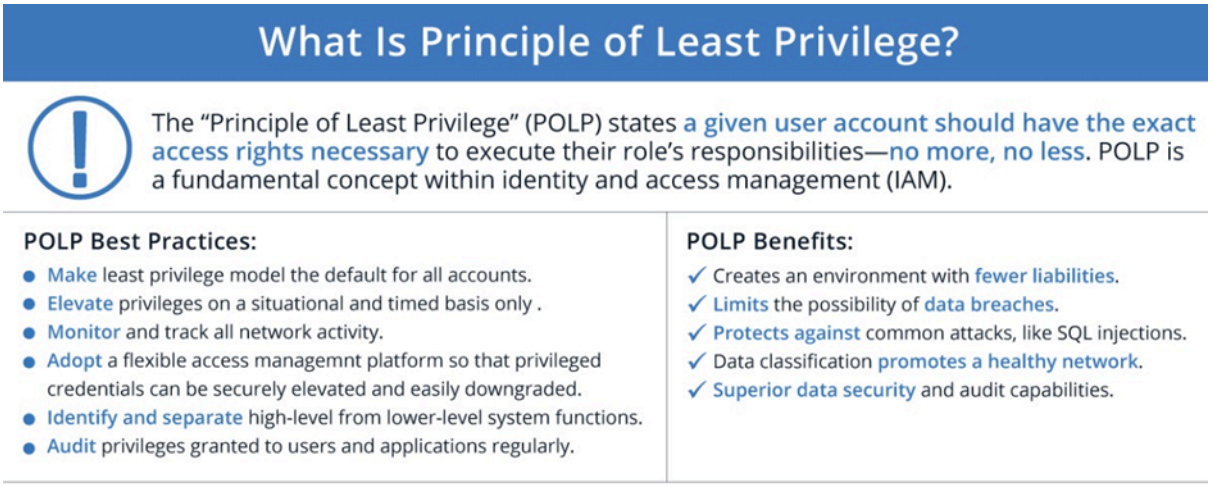
BASIC CONCEPTS

- Shared Responsibility Model
  Security **IN** the cloud versus **OF** the cloud

- Zero Trust model : the scope of verifying goes beyond your presence in the network.
  Packets flow in the network. Do we stop at checking the origin of the packet, or do we dive deeper into a more identity-centric model ?
  e.g. we might check device status and health, time of day, user and device combo, etc…

- Principle of Least privilege





HOW ?

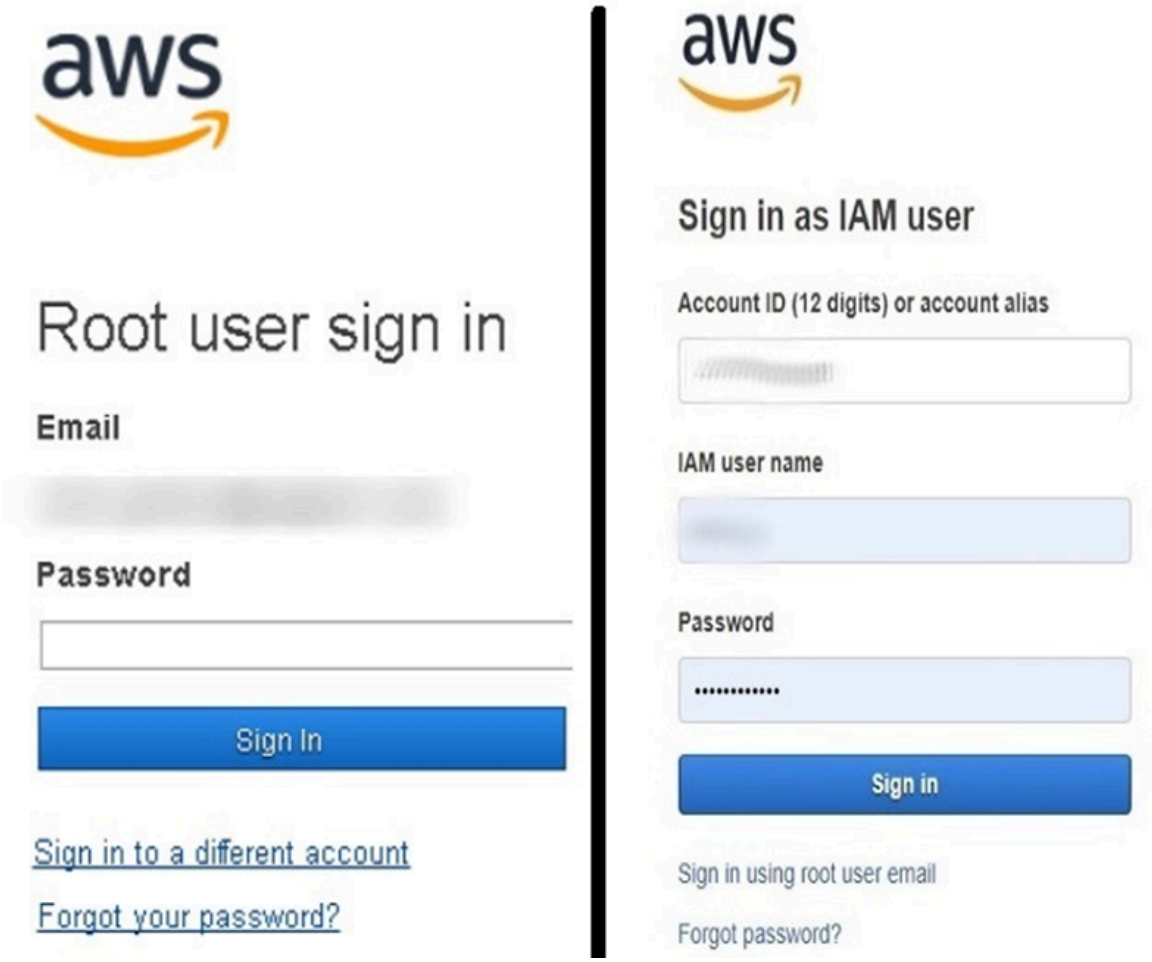
- Identity and Access Management
- Network Security
- Data Encryption

ACCESS TO AWS

- Via a freely downloadable SDK (Software Development Kit)

- Programmatic access : via the CLI (Command Line Interface) using access keys
- Via the AWS Management Console

https://console.aws.amazon.com



FIRST THINGS FIRST

- ROOT account security
    - Store Root User login credentials separately
    - Create a new user for yourself
    - Create Group Email alias with full administrative privileges so others can enter when you are absent
    - Enable multifactor authentication (software or hardware)
    - Delete root user access keys
    - Create a specific new users with his own access keys for programmatic access
    - Enable CloudTrail in all regions and limit access to the S3 bucket for administrators only

- IDENTITY AND ACCESS MANAGEMENT
    - Create individual IAM users
    - Use GROUPS to assign permission to IAM users
    - Use ROLES for applications on EC2's when they need to run
    - Use POLICY CONDITIONS for extra security.
      example : users have access to table, but only to the rows that they themselves created
    - Delegate by using ROLES instead of sharing credentials
    - Rotate Credentials
    - Monitor activity

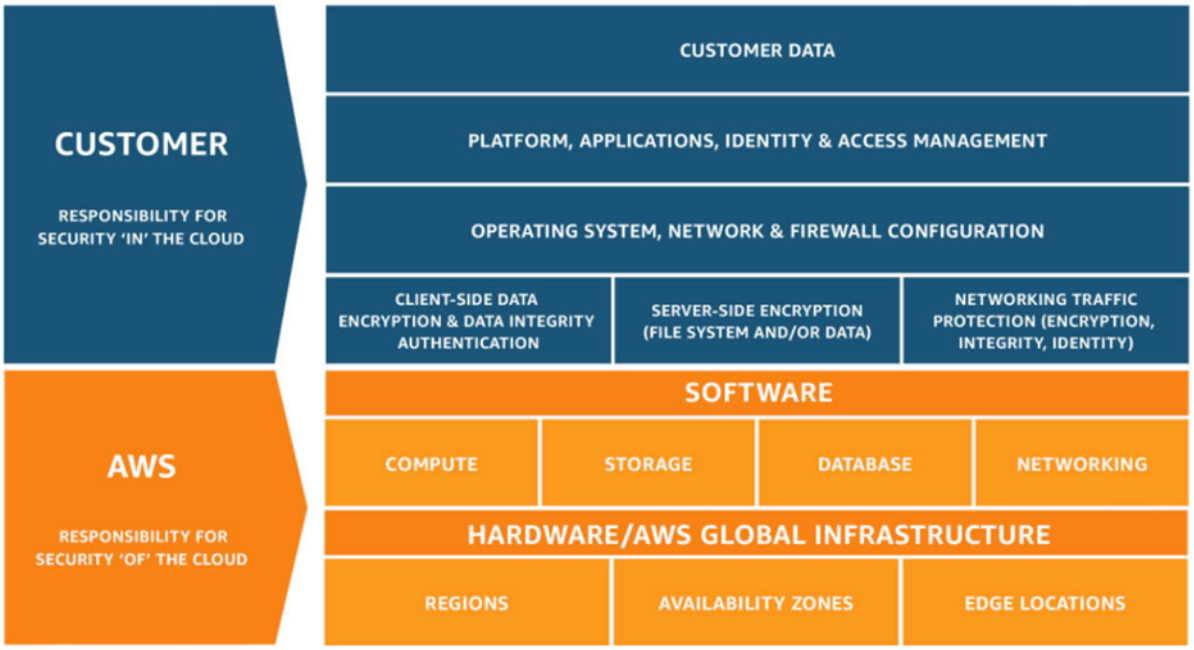
Question: What is the difference between a root user and an IAM user with Administrative Access ?
Answer: A root user is the only one to change his password, update account settings and billing information, and changing support plan.

For Practical Reasons it is also advised to immediately set up a Cost and Usage Report

- Provides hourly and daily usage data that serves as a feed for a Billing Report
- The Billing report can be posted daily to an S3 bucket.

32_AWS Cloud Security - Compliance - Slides.pdf

COMPLIANCE IS A SHARED RESPONSIBILITY



WHAT?

Provide assurance of effective Risk Management because of compliance with widely recognized frameworks, such as the Well Architected Framework.

HOW?

AWS Security Blog : Handy page that is updated daily with new articles.
AWS Architecture Center is a reference for diagrams, deployments and whitepapers.

GDPR Article 40:

CISPE : Cloud Infrastructure Service Providers in Europe

Data Protection Code of conduct

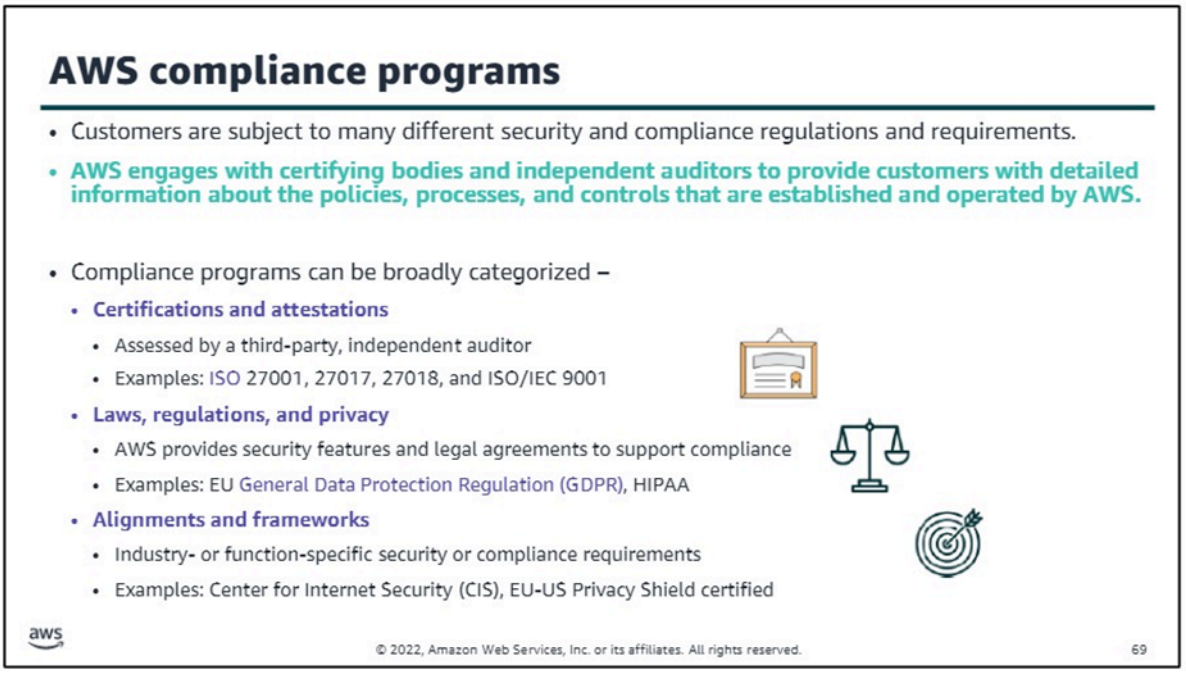
Well Architected Tool : Free service that enables the customer to review workloads and obtain compliance. Use Lenses to obtain more industry specific compliance.
AWS Artifact :

- Free service to access security and Compliance Reports
- Search and accept agreements
- Create configuration to receive notifications of updates

AWS Audit Manager : Paid service that provides a framework for automating assessments that check if your controls (policies, procedures and activities) are operating effectively. Testing procedures are documented and on activating the assessment, the gathering of evidence will generate an audit-ready report. You can also delegate to resource owners to validate.
AWS Security HUB : Paid service to automate checks and centralize security alerts, multiple regions.

Benefits of AWS

Third party validation for 1000's of global requirements in all sectors



Https://aws.amazon.com/compliance/programs

33_AWS Cloud Security - Resources - Slides.pdf

Security alert : WHAT NOW ?

AWS Account Teams

- Your FIRST point of contact
- They will guide you or direct you to the right information

AWS Enterprise support

- 15 min response time
- 24/7
- Technical Account Manager (TAM)

AWS Partner Network

- APN Partners

    - Implement and manage cloud deployment
    - Develop security policies
    - Help meet compliance requirements
    - Include system integrators and managed service providers
- APN Technology Partners
    - Provide Software tools and (integrated) AWS services
    - Include SAAS providers or Independent Software Vendors

Advisories and bulletins

- Current vulnerabilities and threats
- Customers and experts address
    - Reporting Abuseμ
    - Vulnerabilities
- Penetration Testing

AWS Auditor Learning Path

- Learn how to become compliant via labs, training and access to resources

AWS Compliance Solutions Guide

- Access the resources listed in the Compliance chapter

34_Cloud Architecting - Well-Arch - Slides.pdf

WHAT NOT ?

- Architectural Patterns
- Implementation Guide
- Relevant Case Studies

WHAT ?

- Questions centered on critically understanding architectural decisions
- Services and Solutions relevant to each question
- References to relevant resources

Operational Excellence : MONITOR AND INCREASE BUSINESS VALUE

- Continuous monitoring of systems to increase Business Value by managing and automating changes
- Improve processes and procedures by responding to events and responding to changes

Security : MONITOR AND PROTECT

- Monitor and protect Systems, Assets and Information
- Risk Assessments and mitigation strategies

5 Key Areas of AWS Cloud Security

- Identity and Access Management (IAM)
    - Centralized access control
    - Granular Permission managementμ
    - Enhanced security through MFA
- Detective Controls
    - CloudTrail : API calls : user actions
    - VPC flow Logs : network traffic flow
    - Guard Duty : unusual or malicious behaviour detection
- Infrastructure Protection
    - Security Groups : virtual firewall
    - NACL : control traffic at subnet level
    - Web Application Firewall : filter malicious requests to your applications
- Data Protection
    - Data Encryption
    - Backup and Recovery
    - Data Lifecycle Management
- Incident Response : Failing to Prepare is preparing to Fail
- Application Protection

RELIABILITY : MONITOR AND RECOVER

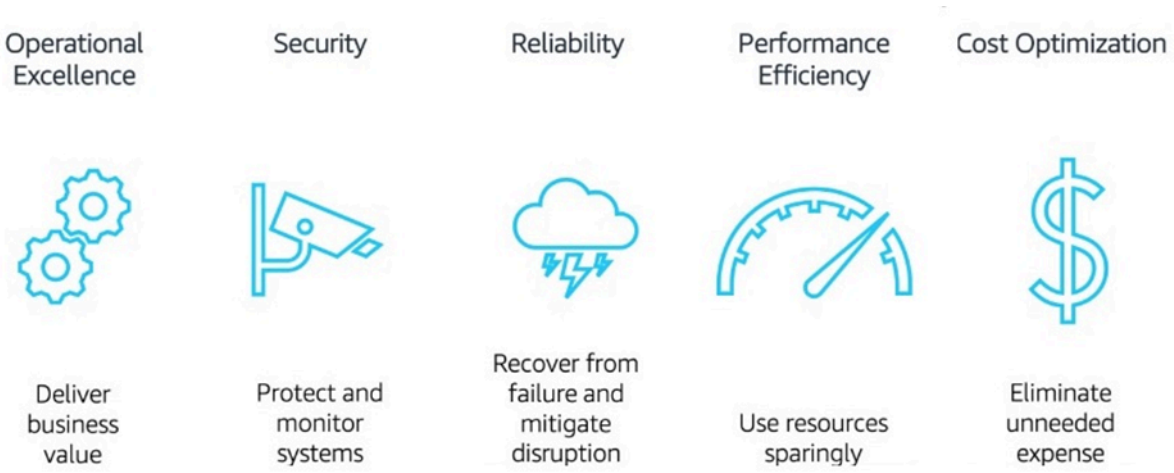
- From infrastructure or service failures
- Soften disruptions like misconfigurations or impermanent network issues
- Dynamically acquire extra resources if needed

PERFORMANCE EFFICIENCY : MONITOR TO OPTIMIZE

- Select Customizable solutions
- Review to innovate
- Consider the trade-offs

COST OPTIMIZATION : MONITOR TO SAVE MONEY

- Use cost-effective resources
- Monitor to match supply with demand
- Optimize over time



SUSTAINABILITY : MONITOR TO DECREASE IMPACT

The Sustainability pillar focuses on the environmental impacts, especially energy consumption and efficiency since they are important levers for architects to inform direct action to reduce resource usage.

35_Cloud Architecting - Deep Dive - Slides.pdf

• Stop guessing your capacity needs

Scale up and down manually or automatically

• Test systems at production scale

Create duplicate environment and erase when done

• Automate to make architectural experimentation easier

Replicate systems, audit the impact of your changes and revert if necessary

• Allow for evolutionary architectures

Automate on demand lowers the risk of impact of design changes

• Drive architectures using data

Collect data on workload and make fact-based decisions on architecture choices

• Improve through game days

Test performance while simulating events in production

OPERATIONAL EXCELLENCE

- Perform operations as code
- Make frequent, small, reversible changes
- Refine operations procedures frequently
- Anticipate failure
- Learn from all operational failures

SECURITY IN THE CLOUD

- Implement a strong identity foundation
- Enable traceability
- Apply security at all layers
- Automate security best practices
- Protect data in transit and at rest
- Keep people away from data

- Prepare for security events

RELIABILITY IN THE CLOUD

- Automatically recover from failure
- Test recovery procedures
- Scale horizontally to increase aggregate workload availability
- Stop guessing capacity
- Manage change in automation

PERFORMANCE EFFICIENCY

- Democratize advanced technologies
- Go global in minutes
- Use serverless architectures
- Experiment more often
- Consider mechanical sympathy

COST OPTIMIZATION

- Implement Cloud Financial Management
- Adopt a consumption model
- Measure overall efficiency
- Stop spending money on undifferentiated heavy lifting
- Analyze and attribute expenditure

SUSTAINABILITY IN THE CLOUD

- Understand your impact
- Establish sustainability goals
- Maximize utilization
- Anticipate and adopt new, more efficient hardware and software offerings
- Use managed services
- Reduce the downstream impact of your cloud workloads

36_Cloud Architecting - High Availability - Slides.pdf

RELIABILITY versus AVAILABILITY

RELIABILITY

Does the system perform as it was intended?

- Measure by MTBF : Mean Time Between Failures $\frac{\text{TOTAL TIME}}{\text{TOTAL FAILURES}}$
- Failure Rate $\frac{\text{TOTAL FAILURES}}{\text{TOTAL TIME}}$

AVAILABILITY

Is a percentage of uptime $\frac{\text{NORMAL OPERATION TIME}}{\text{TOTAL TIME}}$

We use a number of 9's to explain.

Example : if an application is 99,999% up, then we say : 5 nines available

Number of Nines	Percent of Uptime	Max Downtime per year	Equivalent Downtime per day
1	90%	36.5 Days	2.4 Hours
2	99%	3.65 Days	14 Minutes
3	99.9%	8.76 Hours	86 Seconds
4	99.99%	52.6 Minutes	8.6 Seconds
5	99.999%	5.25 Minutes	0.86 Seconds

Minimum in human intervention

FAULT TOLERANCE

Built-in redundancy (deletion) of components of an application and still remain operational.

SCALABILITY

The capability to accommodate growth without changing design

RECOVERABILITY

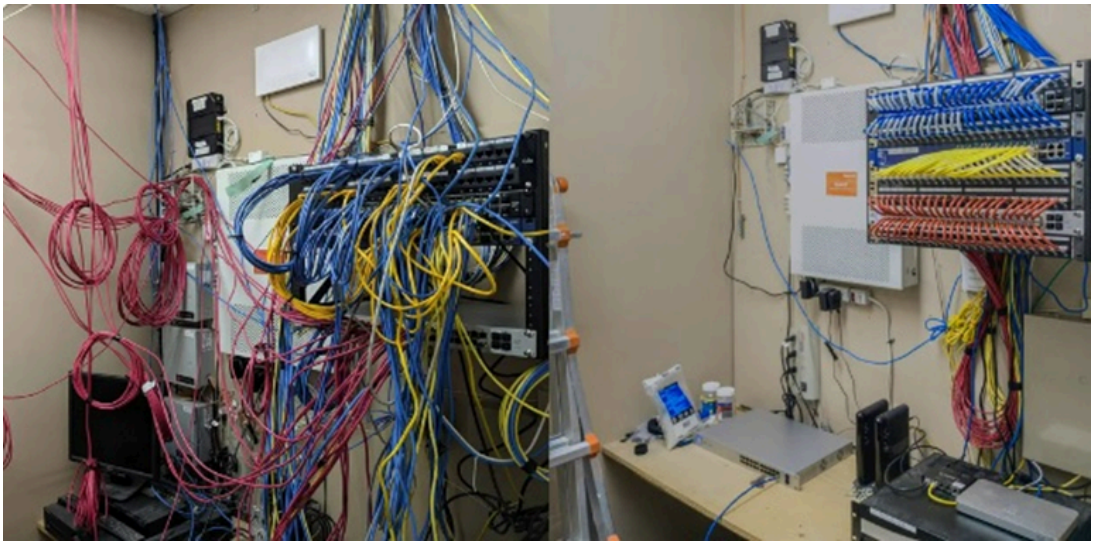
After a catastrophic event we review Processes, Policies and Procedures

ON PREMISES : expensive and only for mission-critical applications

IN THE CLOUD : AWS helps you thanks to

Multiple servers
Isolated redundant data centers within each Availability Zone
Multiple Availability Zones within each AWS Region
Multiple Regions around the world
Fault-tolerant services

37_Cloud Architecting - Data Center - Slides.pdf



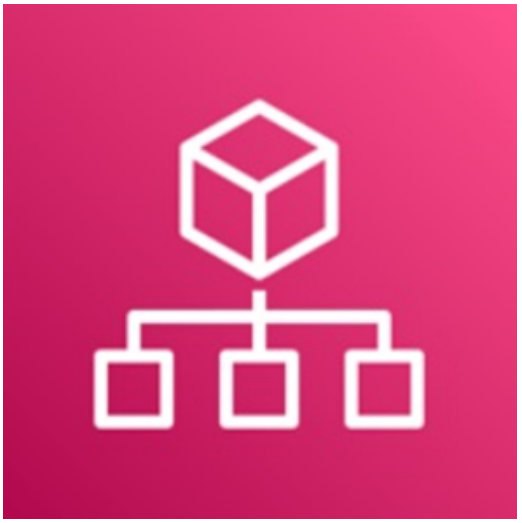
On-Premises

Server Rack
LDAP Server
Software based Load balancers
SAN solutions
NAS solutions
Databases

In The Cloud

EC2
AWS directory Service
Elastic Load Balancing (ELB)
Amazon EBS
Amazon EFS
Amazon RDS

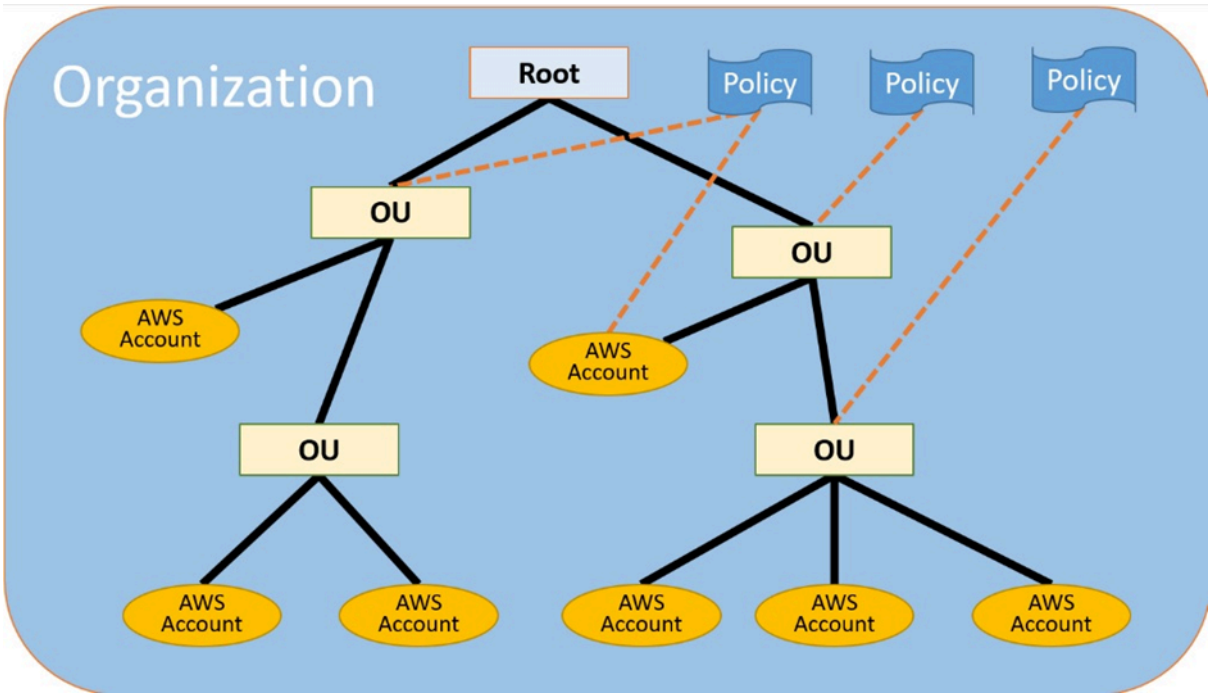
38_Cloud Billing - AWS Organizations - Slides.pdf



An Organization is the MASTER, the ROOT, the HIGHEST LEVEL.

An Organization controls the service policies for every

- Organizational Unit
- Account
- User
- Role
- Service



The new name is SERVICE CONTROL POLICIES (SCP)

Maximum 5 levels deep of Organizational Units
Access Organizations via the Management console, or via CLI, SDK, API

Now that we know this… What are the main features that become available ?

- POLICY based management
- GROUP account management
- Use API to create accounts
- Consolidated billing. Therefore we assign Tags to resources to get a more clear overview
  That will help us in obtaining goals regarding budget, security and compliance

SCP takes precedence over IAM Policy.
So if a higher level did not allow the use of a resource, then an IAM Policy in which you allow access will not give you access.

Explicit : included in the list

Implicit : not included in the list

- An explicit deny overrides an explicit allow.
- An explicit allow overrides an implicit deny.

An IAM can never restrict the root of an account
An SCP can restrict the root of an account.

39_Cloud Billing - Cost Mgmt - Slides.pdf

WHY ?

- Analyze data, identify trends
- Set up time intervals
- Forecast future usage
- Save money

WHAT ?

BILLING AND PAYMENTS

- Your bills
- Your Payments and credits

COST ANALYSIS

- Cost Explorer + reports
- Cost Anomaly detection
- Data Exports

COST ORGANIZATION

- Cost categories : to group and split costs
- Allocate tags : AWS generated or Custom defined

BUDGETS AND PLANNING

- Budgets + Reports : create thresholds and set alarms
- Pricing calculator

SAVINGS AND COMMITMENTS

- Cost Optimization Hub
- Saving Plans

Comparing RIs and Savings Plans

Savings plans offer all the benefits of RI as well as improved flexibility and reduced management

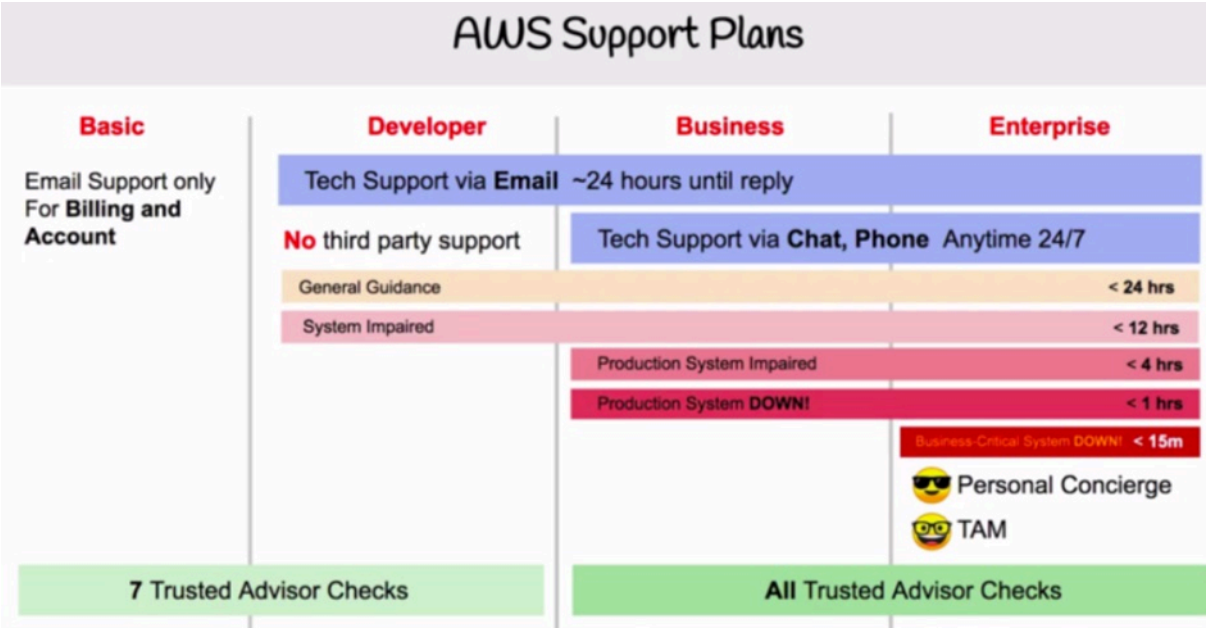
	Compute Savings Plans	EC2 Instance Savings Plans	Convertible RIs*	Standard RIs
Savings over On Demand	Up to 66%	Up to 72%	Up to 66%	Up to 72%
Low price in exchange for monetary commitment	✓	✓	✗	✗
Pricing automatically applies to any instance families	✓	✗	✗	✗
Pricing automatically applies to any instance size	✓	✓	✗**	✗**
Pricing automatically applies to any Tenancy, or OS	✓	✓	✗	✗
Automatically apply to Fargate Usage	✓	✗	✗	✗
Pricing automatically applies to across any AWS Region	✓	✗	✗	✗
1- and 3-year Term Length options	✓	✓	✓	✓

*convertible RIs can be changed across instance families, sizes, OS and tenancy they require customers to manually perform exchanges.
**Regional Convertible RIs and Regional Standard RIs provide instance flexibility

40_Cloud Billing - Support Services - Slides.pdf

FOR EVERYBODY

- Trusted Advisor : 7 checks
- Health Dashboard (list of AWS services that are having issues)



FREE 29$/month from 5500$/month from 15000$/month