# AWS Certified Cloud Practitioner

## By Stéphane Maarek


COURSE →

EXTRA PRACTICE EXAMS →

# Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the AWS Certified Cloud Practitioner Course

- **Please do not share this document**, it is intended for personal use and exam preparation only, thank you.

- Best of luck for the exam and happy learning!

# AWS Certified Cloud Practitioner Course

# CLF-C01

# Welcome! We're starting in 5 minutes


- We're going to prepare for the **Cloud Practitioner exam** – CLF-C01
- It's a challenging certification, so this course will be long and interesting
- Basic IT knowledge is helpful, but I will explain everything

- We will cover over **40 AWS services** (out of the 200+ in AWS)
- AWS / IT Beginners welcome! (but take your time, it's not a race)
- **Learn by doing – key learning technique!**
  **This course mixes both theory & hands on**

# Sample question: Certified Cloud Practitioner

**Which AWS service would simplify the migration of a database to AWS?**

- A) AWS Storage Gateway <= we will learn
- B) AWS Database Migration Service <= correct answer
- C) Amazon EC2 <= we will learn
- D) Amazon AppStream 2.0 <= distractor (over 200 services in AWS)

- https://d1.awsstatic.com/training-and-certification/docs-cloud-practitioner/AWS-Certified-Cloud-Practitioner_Sample-Questions.pdf

# About me

- I'm Stephane!
- 9x AWS Certified (so far!)
- Worked with AWS many years: built websites, apps, streaming platforms
- Veteran Instructor on AWS (Certifications, CloudFormation, Lambda, EC2…)

- You can find me on
  - LinkedIn: https://www.linkedin.com/in/stephanemaarek
  - Medium: https://medium.com/@stephane.maarek
  - Twitter: https://twitter.com/stephanemaarek
  - GitHub: https://github.com/simplesteph

**4.6** Instructor Rating

**107,634** Reviews

**356,650** Students

**31** Courses

# Your AWS Certification journey

## FOUNDATIONAL

**Six months** of fundamental AWS Cloud and industry knowledge


## PROFESSIONAL

**Two years** of experience designing, operating, and troubleshooting solutions using the AWS Cloud


## ASSOCIATE

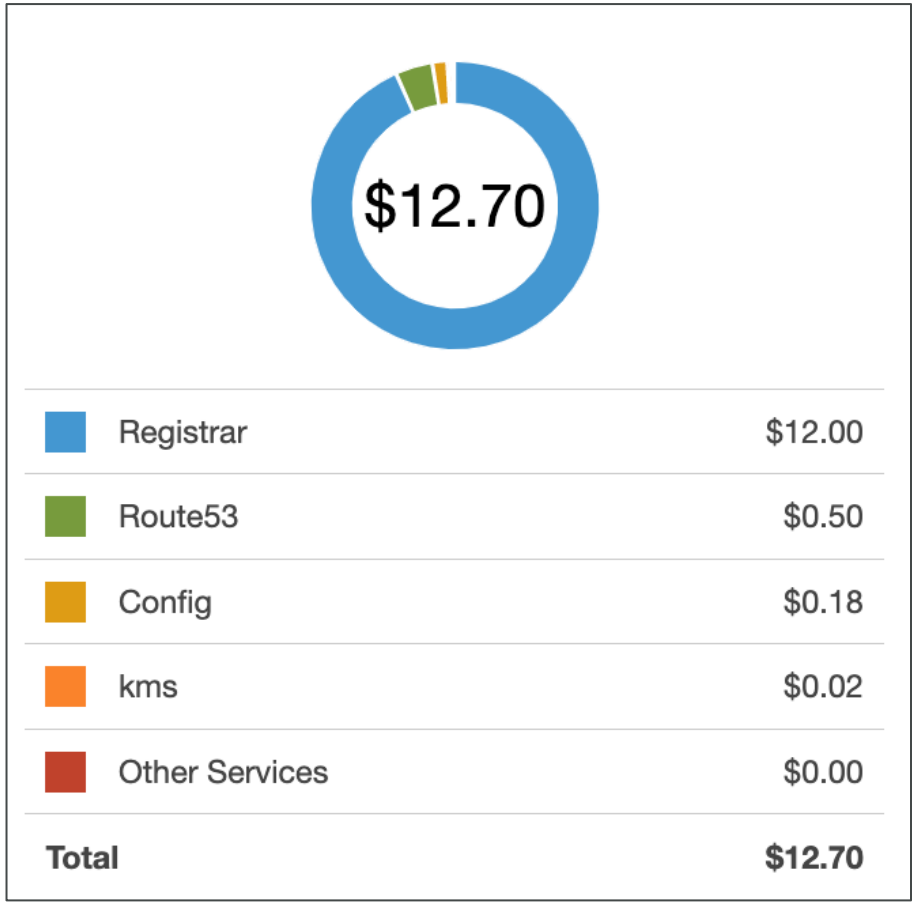
**One year** of experience solving problems and implementing solutions using the AWS Cloud


## SPECIALTY

Technical AWS Cloud experience in the Specialty domain as specified in the exam guide

# Estimated Cost for this Course

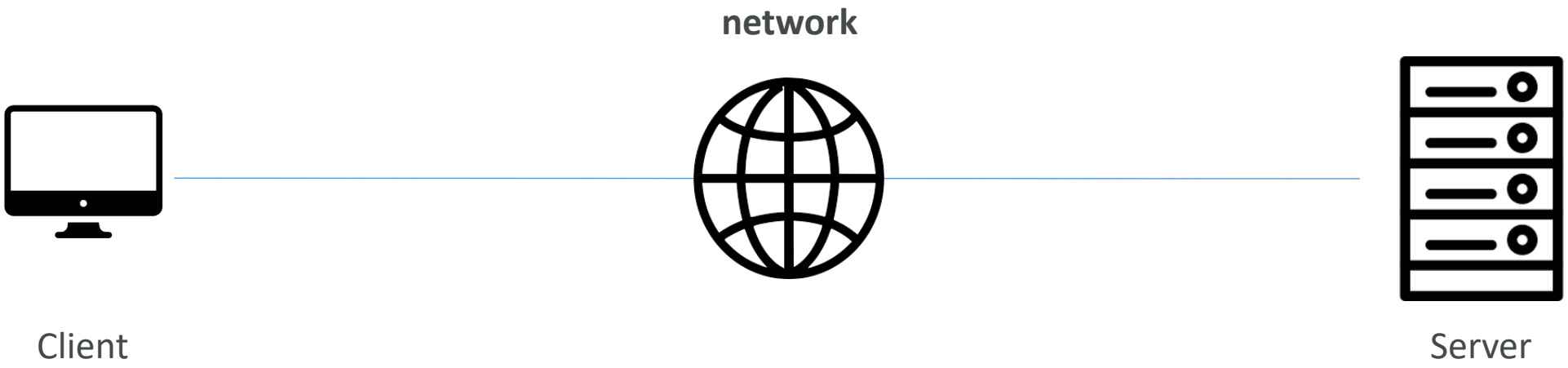
- Most of the services we'll use will be within the AWS Free Tier = $0
- If I use a service which will cost you money, I will mention it
- You can read more about the AWS Free Tier at: https://aws.amazon.com/free/



# Udemy Tips

# What is Cloud Computing Section

# How websites work



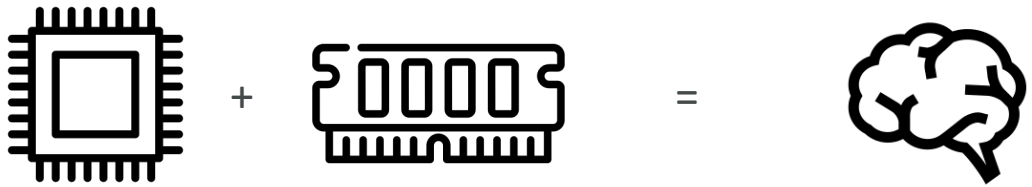

**Clients have IP addresses**

**Servers have IP addresses**

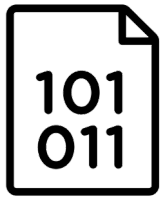
# Just like when you're sending post mail!

# What is a server composed of?

- Compute: CPU
- Memory: RAM



- Storage: Data

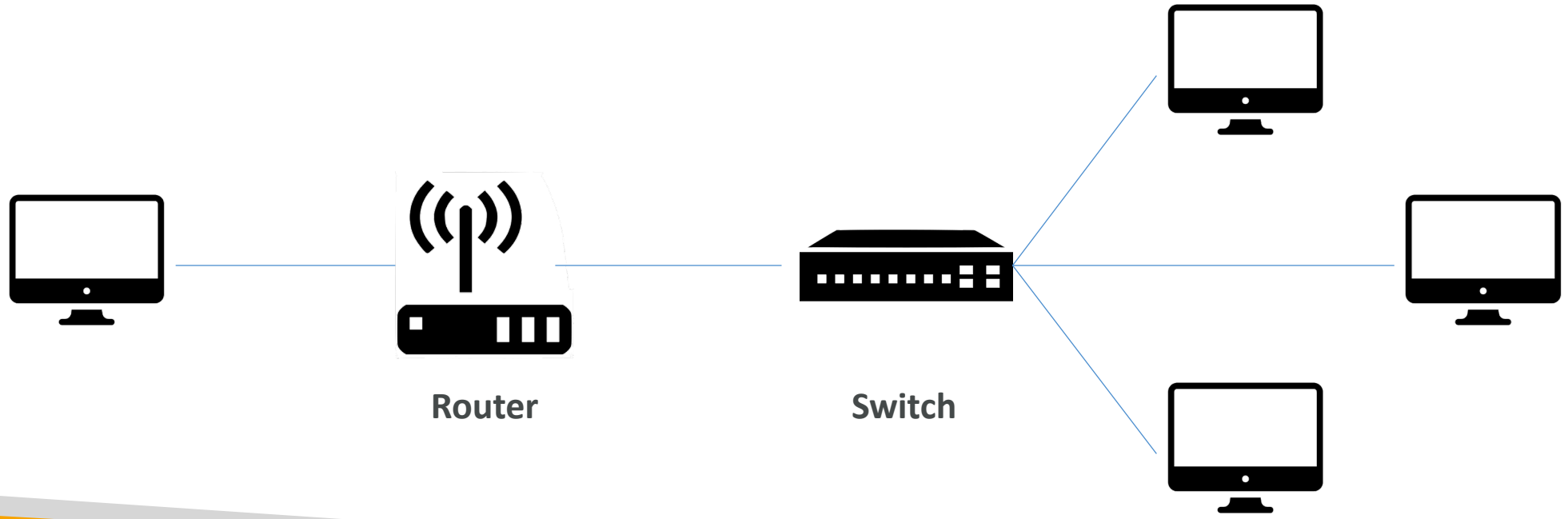



- Database: Store data in a structured way

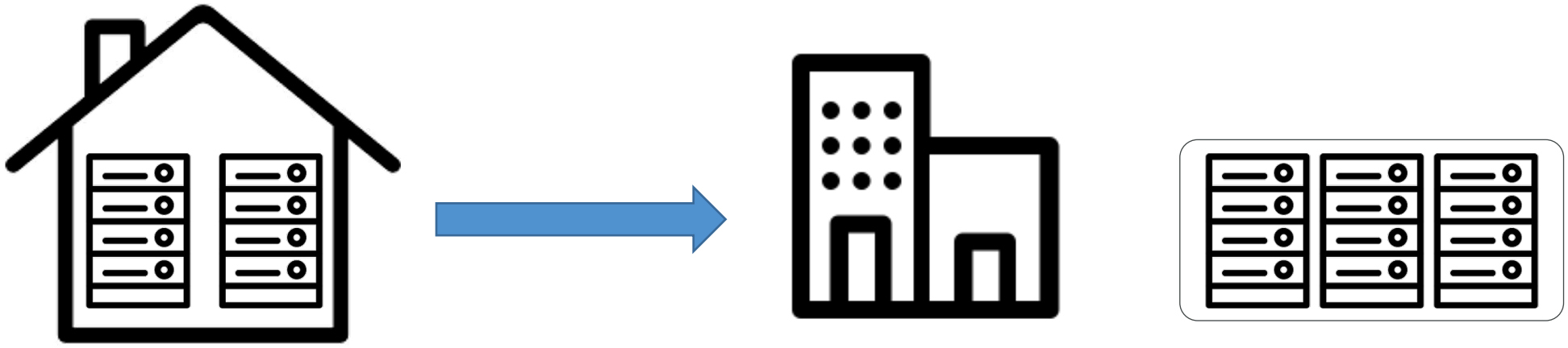
- Network: Routers, switch, DNS server

# IT Terminology

- **Network:** cables, routers and servers connected with each other
- **Router:** A networking device that forwards data packets between computer networks. They know where to send your packets on the internet!
- **Switch:** Takes a packet and send it to the correct server / client on your network



# Traditionally, how to build infrastructure



Home or Garage

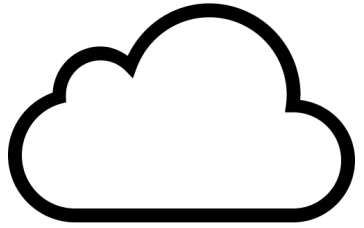
Office

Data center

# Problems with traditional IT approach

- Pay for the rent for the data center
- Pay for power supply, cooling, maintenance
- Adding and replacing hardware takes time
- Scaling is limited
- Hire 24/7 team to monitor the infrastructure
- How to deal with disasters? (earthquake, power shutdown, fire…)

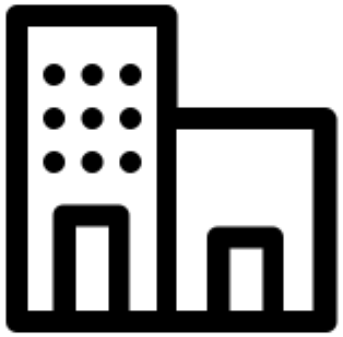
- Can we externalize all this?



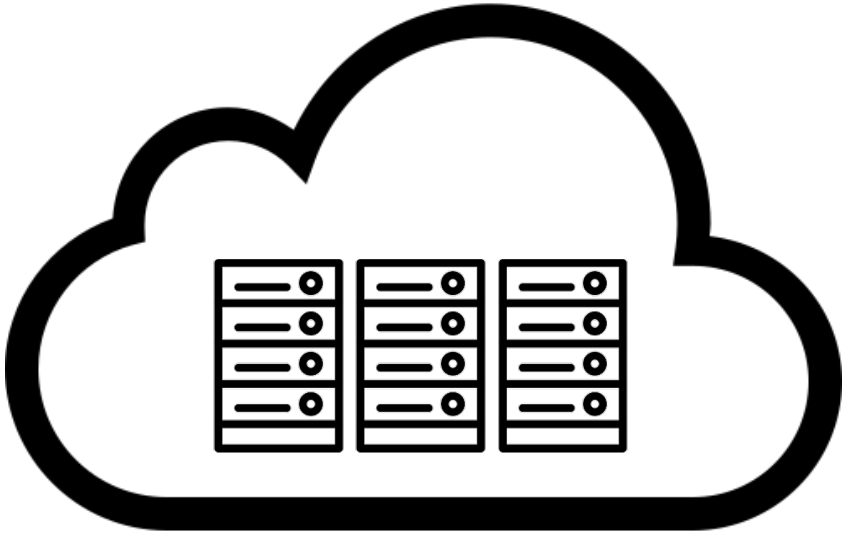
# What is Cloud Computing?


- Cloud computing is the **on-demand delivery** of compute power, database storage, applications, and other IT resources
- Through a cloud services platform with **pay-as-you-go pricing**
- You can **provision exactly the right type and size of computing** resources you need
- You can access as many resources as you need, **almost instantly**
- Simple way to access **servers, storage, databases** and a set of **application services**

- Amazon Web Services owns and maintains the network-connected hardware required for these application services, while you provision and use what you need via a web application.



Office



The Cloud

# You've been using some Cloud services

**Gmail**

- E-mail cloud service
- Pay for ONLY your emails stored (no infrastructure, etc.)


**Dropbox**

- Cloud Storage Service
- Originally built on AWS

**Netflix**

- Built on AWS
- Video on Demand

# The Deployment Models of the Cloud

## Private Cloud:

- Cloud services used by a single organization, not exposed to the public.
- Complete control
- Security for sensitive applications
- Meet specific business needs

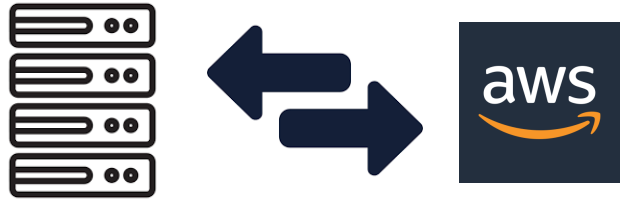
## Public Cloud:

- Cloud resources owned and operated by a third-party cloud service provider delivered over the Internet.
- Six Advantages of Cloud Computing

## Hybrid Cloud:

- Keep some servers on premises and extend some capabilities to the Cloud
- Control over sensitive assets in your private infrastructure
- Flexibility and cost-effectiveness of the public cloud





# The Five Characteristics of Cloud Computing

- On-demand self service:
  - Users can provision resources and use them without human interaction from the service provider
- Broad network access:
  - Resources available over the network, and can be accessed by diverse client platforms
- Multi-tenancy and resource pooling:
  - Multiple customers can share the same infrastructure and applications with security and privacy
  - Multiple customers are serviced from the same physical resources
- Rapid elasticity and scalability:
  - Automatically and quickly acquire and dispose resources when needed
  - Quickly and easily scale based on demand
- Measured service:
  - Usage is measured, users pay correctly for what they have used

# Six Advantages of Cloud Computing

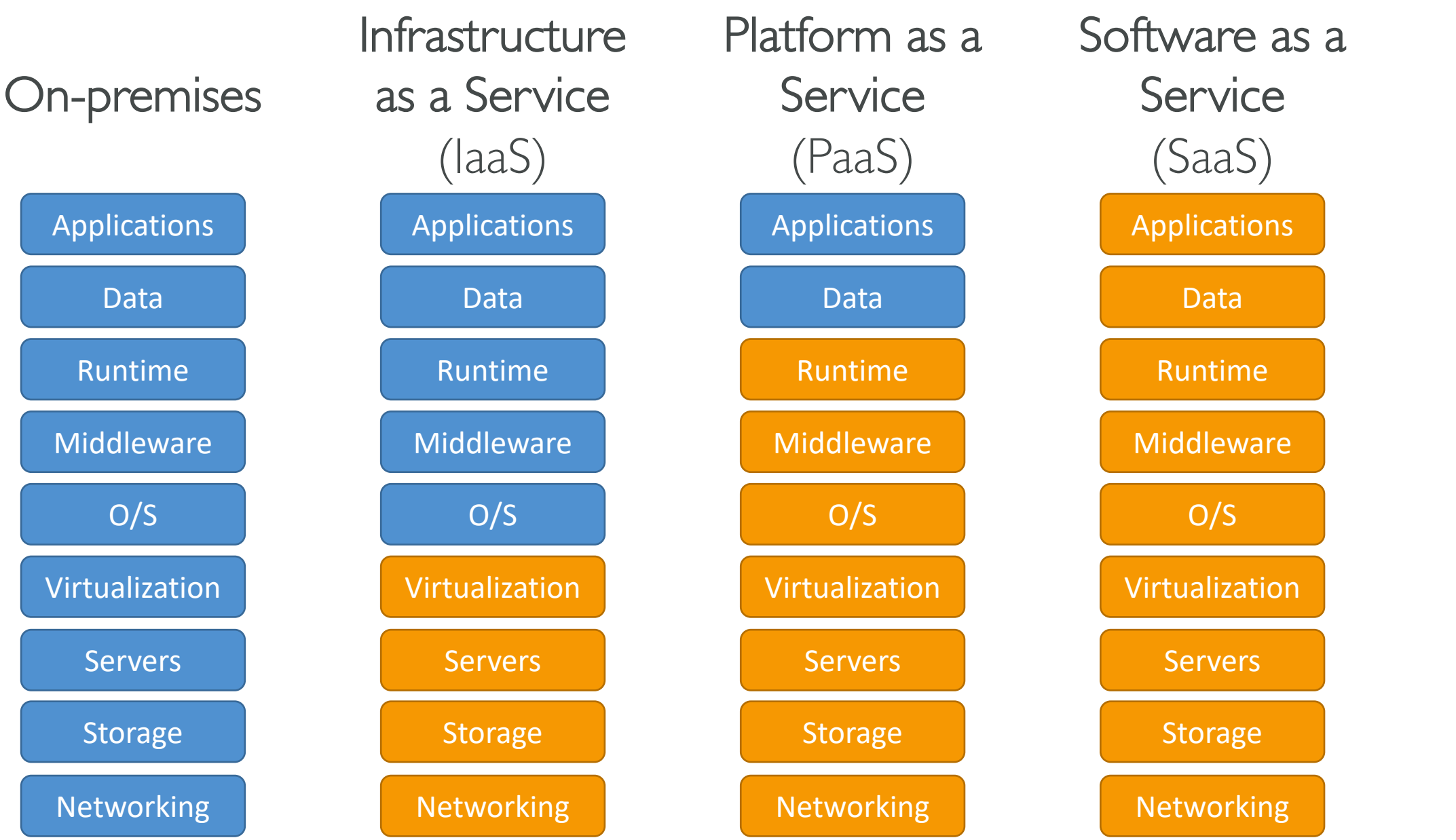
- Trade capital expense (CAPEX) for operational expense (OPEX)
  - Pay On-Demand: don't own hardware
  - Reduced Total Cost of Ownership (TCO) & Operational Expense (OPEX)
- Benefit from massive economies of scale
  - Prices are reduced as AWS is more efficient due to large scale
- Stop guessing capacity
  - Scale based on actual measured usage
- Increase speed and agility
- Stop spending money running and maintaining data centers
- Go global in minutes: leverage the AWS global infrastructure

# Problems solved by the Cloud

- **Flexibility:** change resource types when needed
- **Cost-Effectiveness:** pay as you go, for what you use
- **Scalability:** accommodate larger loads by making hardware stronger or adding additional nodes
- **Elasticity:** ability to scale out and scale-in when needed
- **High-availability and fault-tolerance:** build across data centers
- **Agility:** rapidly develop, test and launch software applications

# Types of Cloud Computing

- Infrastructure as a Service (IaaS)
  - Provide building blocks for cloud IT
  - Provides networking, computers, data storage space
  - Highest level of flexibility
  - Easy parallel with traditional on-premises IT
- Platform as a Service (PaaS)
  - Removes the need for your organization to manage the underlying infrastructure
  - Focus on the deployment and management of your applications
- Software as a Service (SaaS)
  - Completed product that is run and managed by the service provider

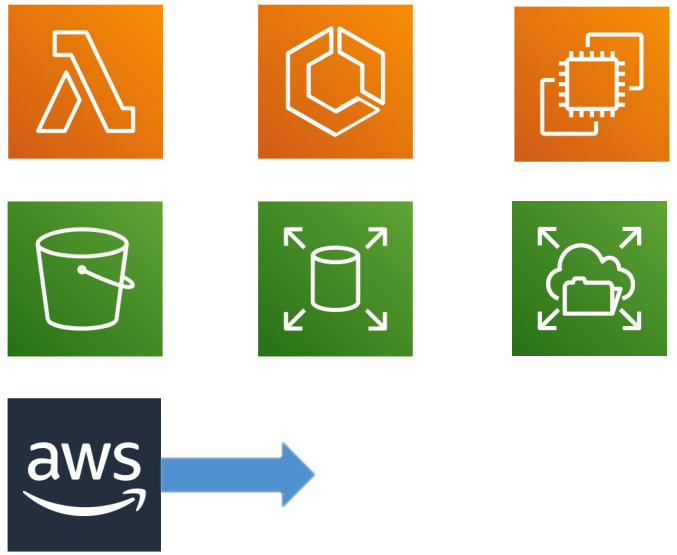


# Example of Cloud Computing Types

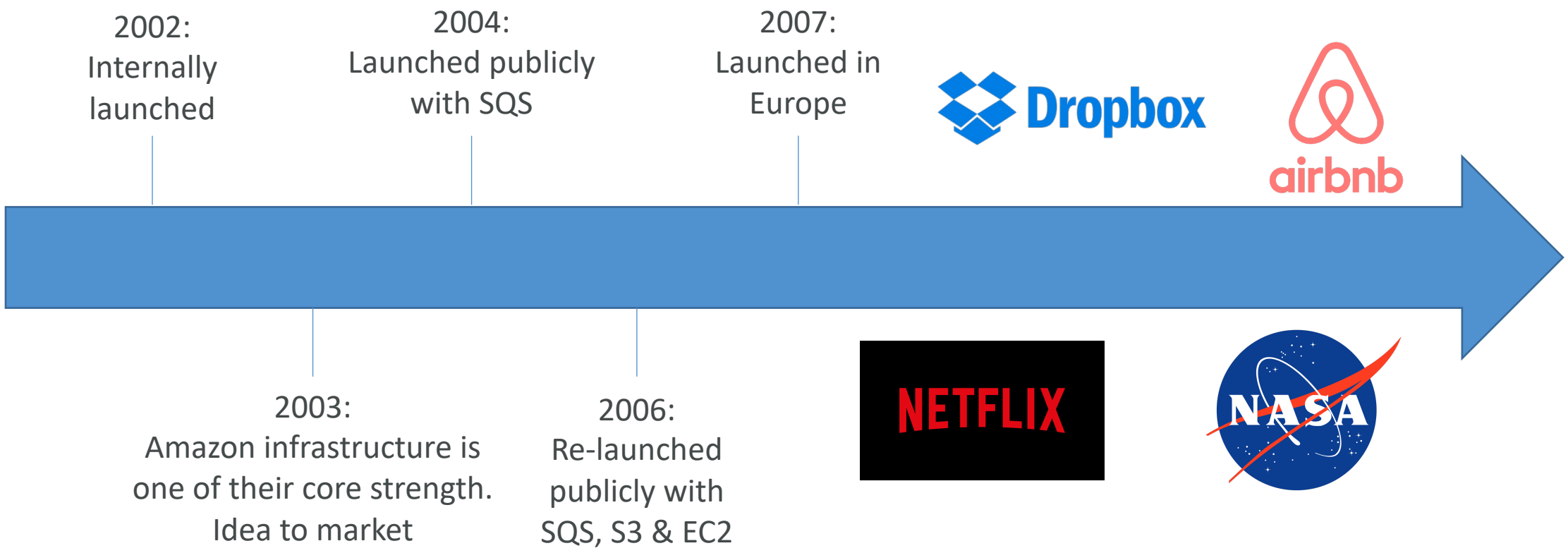
- Infrastructure as a Service:
  - Amazon EC2 (on AWS)
  - GCP, Azure, Rackspace, Digital Ocean, Linode
- Platform as a Service:
  - Elastic Beanstalk (on AWS)
  - Heroku, Google App Engine (GCP), Windows Azure (Microsoft)
- Software as a Service:
  - Many AWS services (ex: Rekognition for Machine Learning)
  - Google Apps (Gmail), Dropbox, Zoom

# Pricing of the Cloud – Quick Overview

- AWS has 3 pricing fundamentals, following the pay-as-you-go pricing model
- **Compute:**
  - Pay for compute time
- **Storage:**
  - Pay for data stored in the Cloud
- **Data transfer OUT of the Cloud:**
  - Data transfer IN is free
- Solves the expensive issue of traditional IT



# AWS Cloud History



# AWS Cloud Number Facts

- In 2019, AWS had $35.02 billion in annual revenue
- AWS accounts for 47% of the market in 2019 (Microsoft is 2nd with 22%)
- Pioneer and Leader of the AWS Cloud Market for the 9th consecutive year
- Over 1,000,000 active users


**Gartner Magic Quadrant**

# AWS Cloud Use Cases

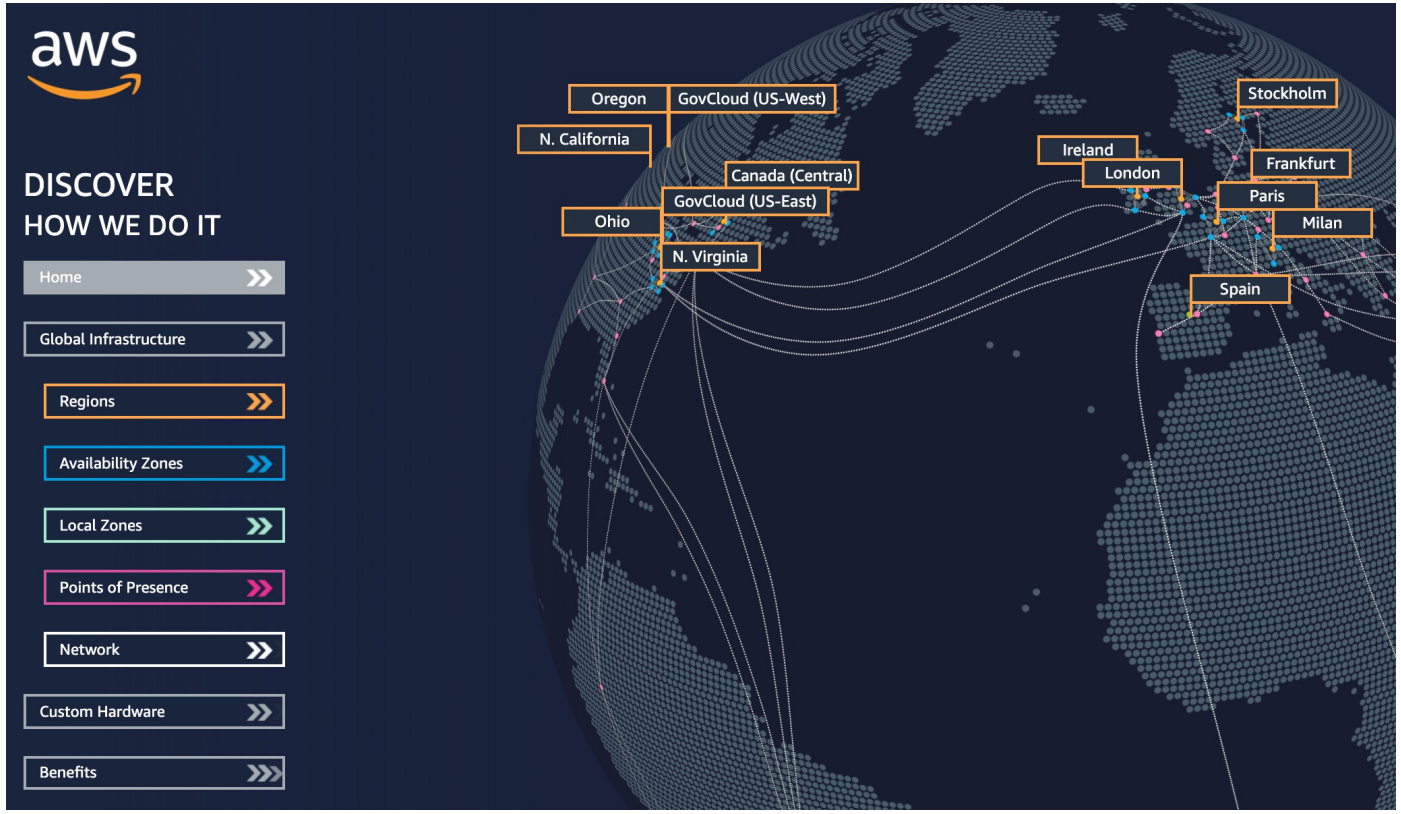
- AWS enables you to build sophisticated, scalable applications
- Applicable to a diverse set of industries
- Use cases include
  - Enterprise IT, Backup & Storage, Big Data analytics
  - Website hosting, Mobile & Social Apps
  - Gaming



# AWS Global Infrastructure

- AWS Regions
- AWS Availability Zones
- AWS Data Centers
- AWS Edge Locations / Points of Presence

- https://infrastructure.aws/



# AWS Regions

- AWS has **Regions** all around the world
- Names can be us-east-1, eu-west-3…
- A region is a **cluster of data centers**
- Most AWS services are region-scoped


https://aws.amazon.com/about-aws/global-infrastructure/

| | |
|---|---|
| **US East (N. Virginia)** | us-east-1 |
| US East (Ohio) | us-east-2 |
| US West (N. California) | us-west-1 |
| US West (Oregon) | us-west-2 |
| Africa (Cape Town) | af-south-1 |
| Asia Pacific (Hong Kong) | ap-east-1 |
| Asia Pacific (Mumbai) | ap-south-1 |
| Asia Pacific (Seoul) | ap-northeast-2 |
| Asia Pacific (Singapore) | ap-southeast-1 |
| Asia Pacific (Sydney) | ap-southeast-2 |
| Asia Pacific (Tokyo) | ap-northeast-1 |
| Canada (Central) | ca-central-1 |
| Europe (Frankfurt) | eu-central-1 |
| Europe (Ireland) | eu-west-1 |
| Europe (London) | eu-west-2 |
| Europe (Paris) | eu-west-3 |
| Europe (Stockholm) | eu-north-1 |
| Middle East (Bahrain) | me-south-1 |
| South America (São Paulo) | sa-east-1 |

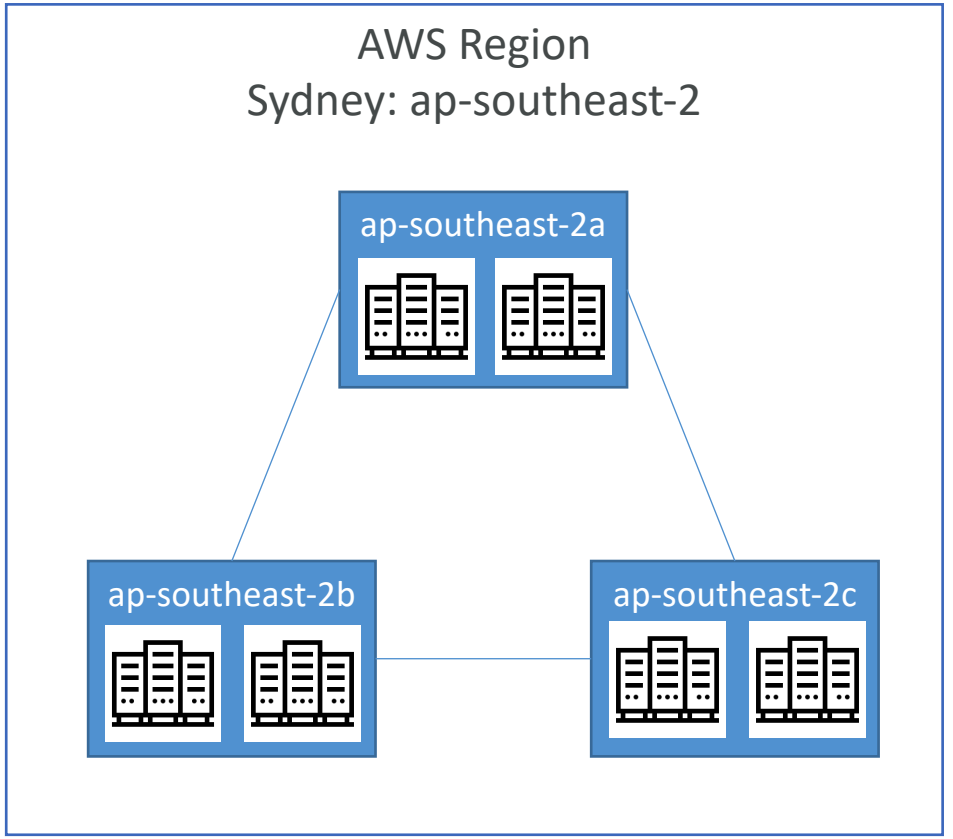
# How to choose an AWS Region?

If you need to launch a new application, where should you do it?


- Compliance with data governance and legal requirements: data never leaves a region without your explicit permission
- Proximity to customers: reduced latency
- Available services within a Region: new services and new features aren't available in every Region
- Pricing: pricing varies region to region and is transparent in the service pricing page

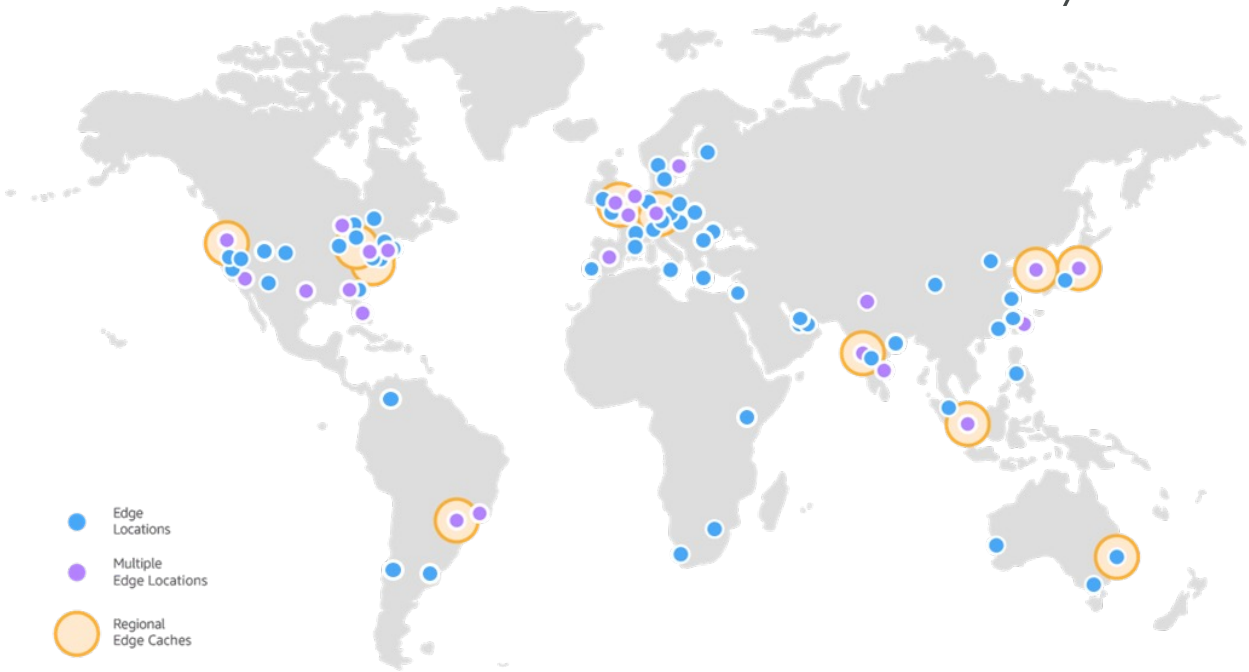
# AWS Availability Zones

- Each region has many availability zones (usually 3, min is 2, max is 6). Example:
  - ap-southeast-2a
  - ap-southeast-2b
  - ap-southeast-2c
- Each availability zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity
- They're separate from each other, so that they're isolated from disasters
- They're connected with high bandwidth, ultra-low latency networking



# AWS Points of Presence (Edge Locations)

- Amazon has 216 Points of Presence (205 Edge Locations & 11 Regional Caches) in 84 cities across 42 countries
- Content is delivered to end users with lower latency

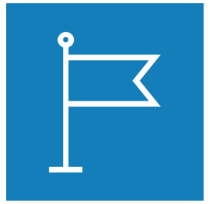



https://aws.amazon.com/cloudfront/features/

# Tour of the AWS Console


- AWS has Global Services:
  - Identity and Access Management (IAM)
  - Route 53 (DNS service)
  - CloudFront (Content Delivery Network)
  - WAF (Web Application Firewall)

- Most AWS services are Region-scoped:
  - Amazon EC2 (Infrastructure as a Service)
  - Elastic Beanstalk (Platform as a Service)
  - Lambda (Function as a Service)
  - Rekognition (Software as a Service)

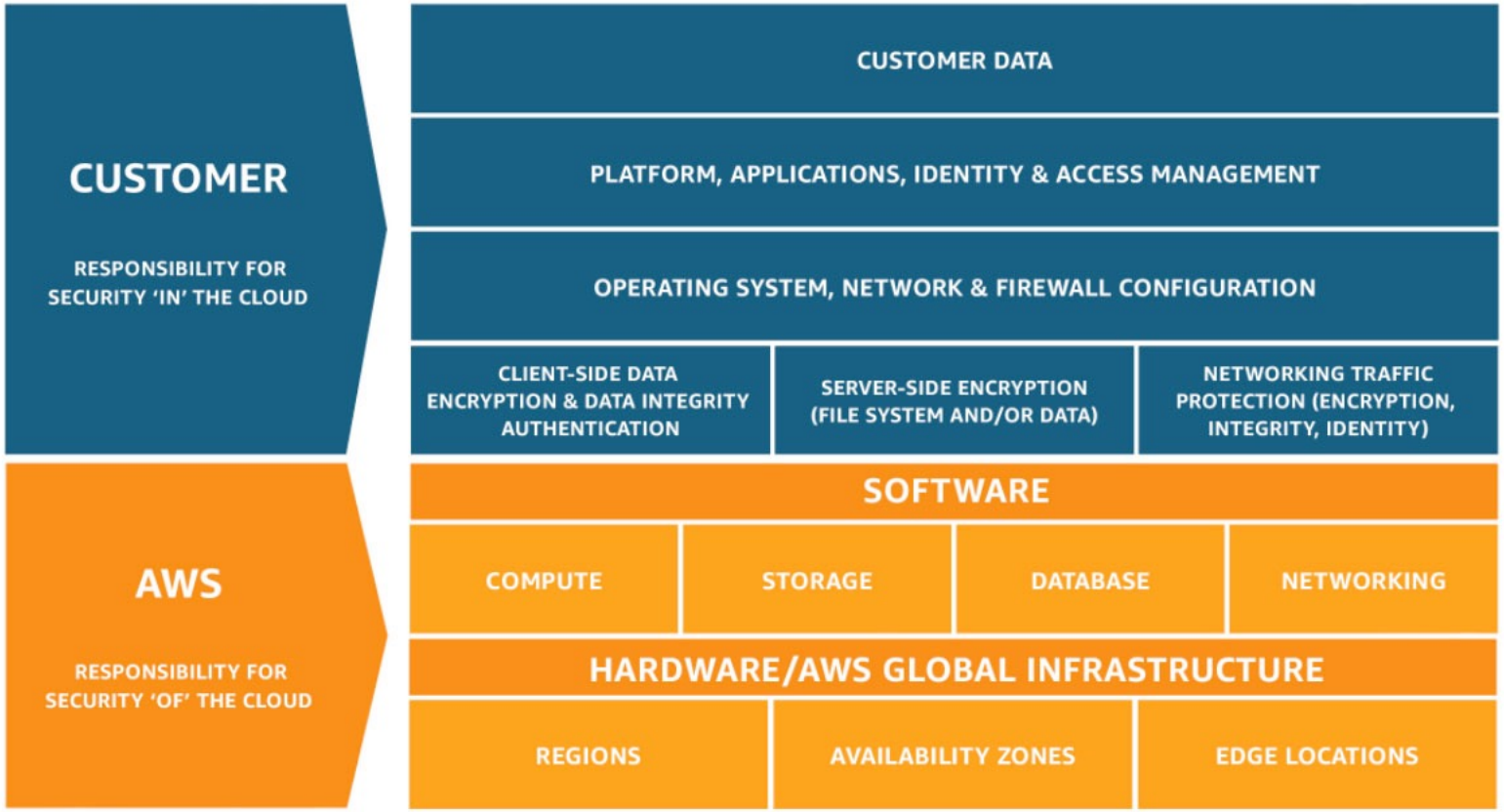


- Region Table: https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services

# Shared Responsibility Model diagram

CUSTOMER = RESPONSIBILITY FOR THE SECURITY **IN** THE CLOUD

AWS = RESPONSIBILITY FOR THE SECURITY **OF** THE CLOUD




https://aws.amazon.com/compliance/shared-responsibility-model/

# AWS Acceptable Use Policy

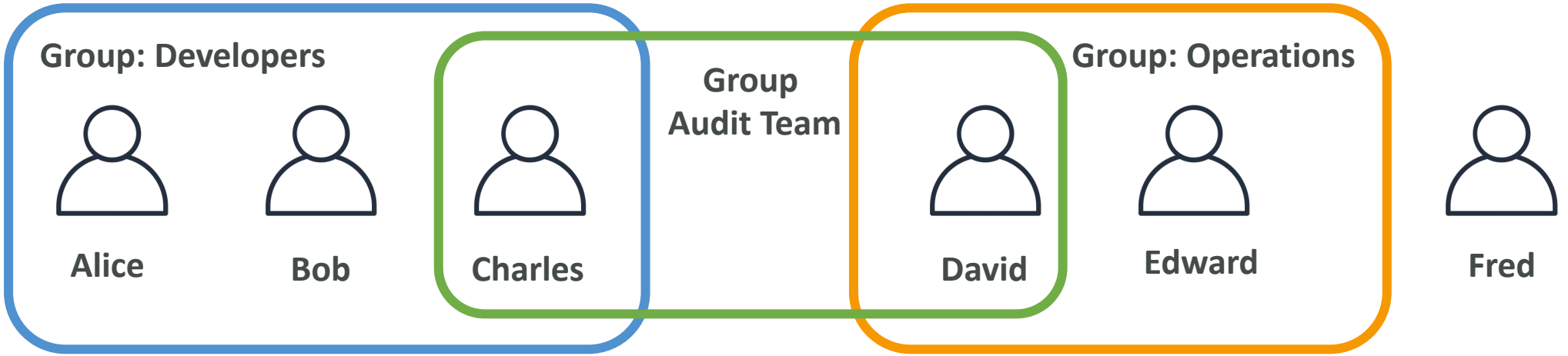
- https://aws.amazon.com/aup/

- No Illegal, Harmful, or Offensive Use or Content
- No Security Violations
- No Network Abuse
- No E-Mail or Other Message Abuse

# IAM Section

# IAM: Users & Groups

- IAM = Identity and Access Management, **Global** service
- **Root account** created by default, shouldn't be used or shared
- **Users** are people within your organization, and can be grouped
- **Groups** only contain users, not other groups
- Users don't have to belong to a group, and user can belong to multiple groups

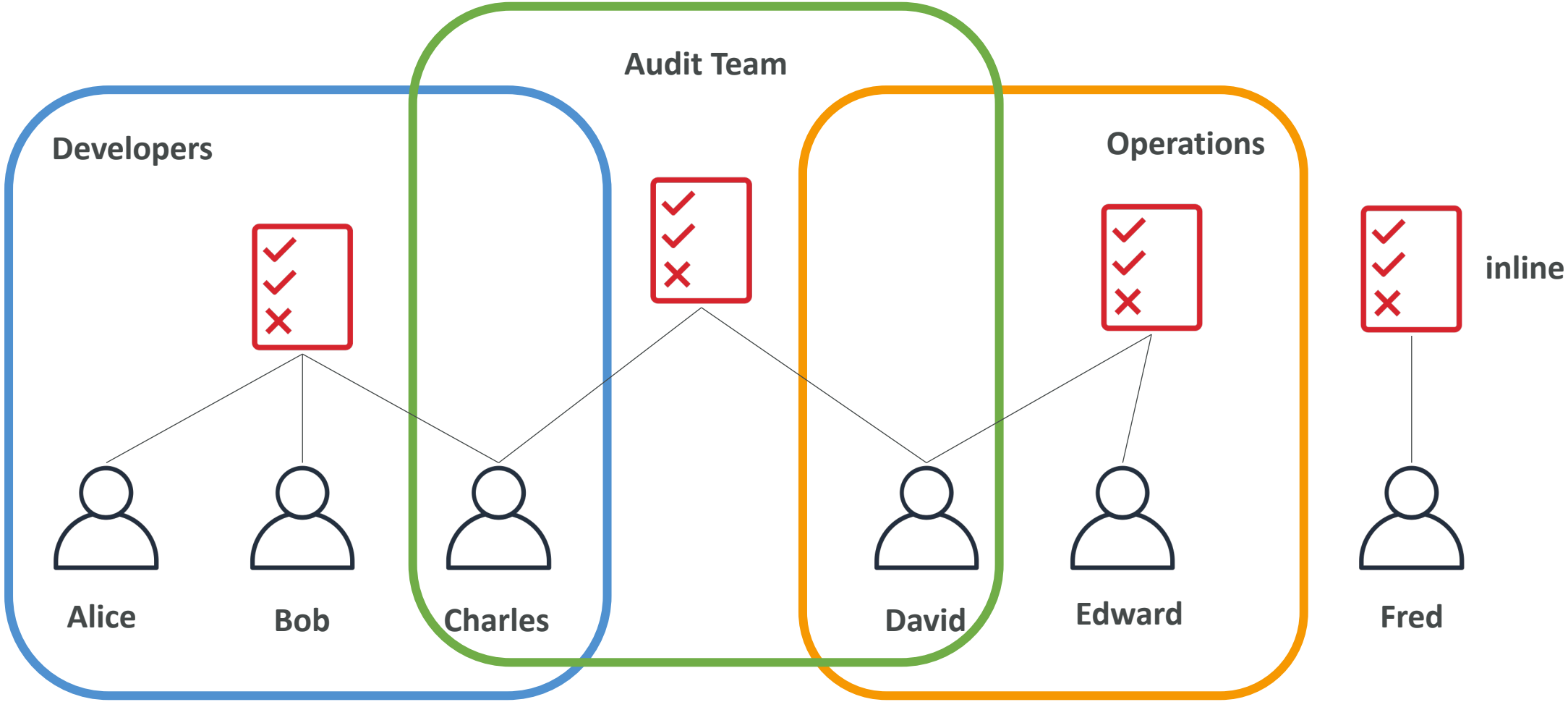


# IAM: Permissions

- **Users or Groups** can be assigned JSON documents called policies
- These policies define the **permissions** of the users
- In AWS you apply the **least privilege principle**: don't give more permissions than a user needs

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

# IAM Policies inheritance



# IAM Policies Structure

- Consists of
  - **Version:** policy language version, always include "2012-10-17"
  - **Id:** an identifier for the policy (optional)
  - **Statement:** one or more individual statements (required)
- Statements consists of
  - **Sid:** an identifier for the statement (optional)
  - **Effect:** whether the statement allows or denies access (Allow, Deny)
  - **Principal:** account/user/role to which this policy applied to
  - **Action:** list of actions this policy allows or denies
  - **Resource:** list of resources to which the actions applied to
  - **Condition:** conditions for when this policy is in effect (optional)

```
{
    "Version": "2012-10-17",
    "Id": "S3-Account-Permissions",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": ["arn:aws:iam::123456789012:root"]
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": ["arn:aws:s3:::mybucket/*"]
        }
    ]
}
```

# IAM – Password Policy

- Strong passwords = higher security for your account
- In AWS, you can setup a password policy:
  - Set a minimum password length
  - Require specific character types:
    - including uppercase letters
    - lowercase letters
    - numbers
    - non-alphanumeric characters
  - Allow all IAM users to change their own passwords
  - Require users to change their password after some time (password expiration)
  - Prevent password re-use

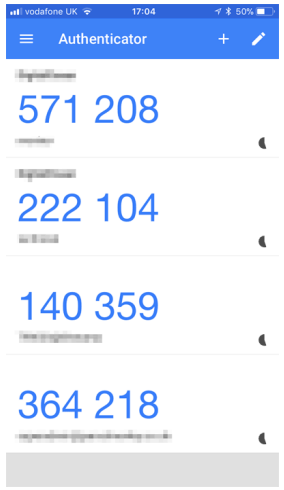
# Multi Factor Authentication - MFA


- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- You want to protect your Root Accounts and IAM users
- MFA = password *you know* + security device *you own*

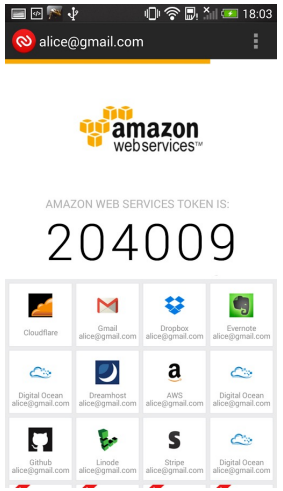

- Main benefit of MFA:
  if a password is stolen or hacked, the account is not compromised

# MFA devices options in AWS

## Virtual MFA device




**Google Authenticator**
(phone only)




**Authy**
(multi-device)

Support for multiple tokens on a single device.

## Universal 2nd Factor (U2F) Security Key

**YubiKey** by Yubico (3rd party)

Support for multiple root and IAM users using a single security key

# MFA devices options in AWS

## Hardware Key Fob MFA Device


Provided by Gemalto (3rd party)

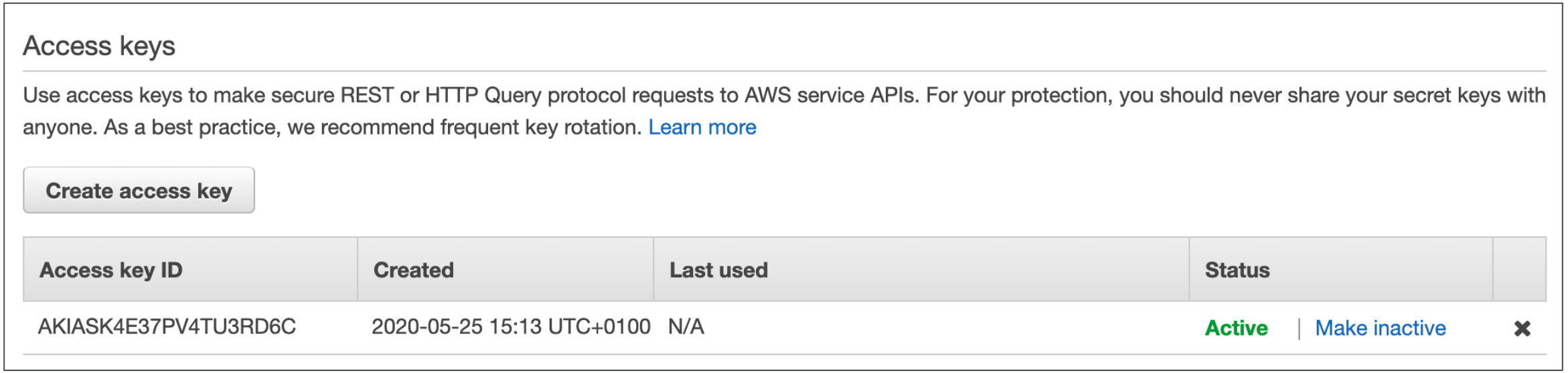
## Hardware Key Fob MFA Device for AWS GovCloud (US)


Provided by SurePassID (3rd party)

# How can users access AWS ?

- To access AWS, you have three options:
  - **AWS Management Console** (protected by password + MFA)
  - **AWS Command Line Interface (CLI):** protected by access keys
  - **AWS Software Developer Kit (SDK)** - for code: protected by access keys
- Access Keys are generated through the AWS Console
- Users manage their own access keys
- Access Keys are secret, just like a password. Don't share them
- Access Key ID ~= username
- Secret Access Key ~= password

# Example (Fake) Access Keys



Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. Learn more

Create access key

| Access key ID | Created | Last used | Status | |
|---|---|---|---|---|
| AKIASK4E37PV4TU3RD6C | 2020-05-25 15:13 UTC+0100 | N/A | Active \| Make inactive | ✖ |

- Access key ID: AKIASK4E37PV4983d6C
- Secret Access Key: AZPN3zojWozWCndIjhB0Unh8239a1bzbzO5fqqkZq
- Remember: don't share your access keys

# What's the AWS CLI?

- A tool that enables you to interact with AWS services using commands in your command-line shell
- Direct access to the public APIs of AWS services
- You can develop scripts to manage your resources
- It's open-source https://github.com/aws/aws-cli
- Alternative to using AWS Management Console

```
➜  ~ aws s3 cp myfile.txt s3://ccp-mybucket/myfile.txt
upload: ./myfile.txt to s3://ccp-mybucket/myfile.txt
➜  ~ aws s3 ls s3://ccp-mybucket
2021-05-14 03:22:52          0 myfile.txt
➜  ~
```

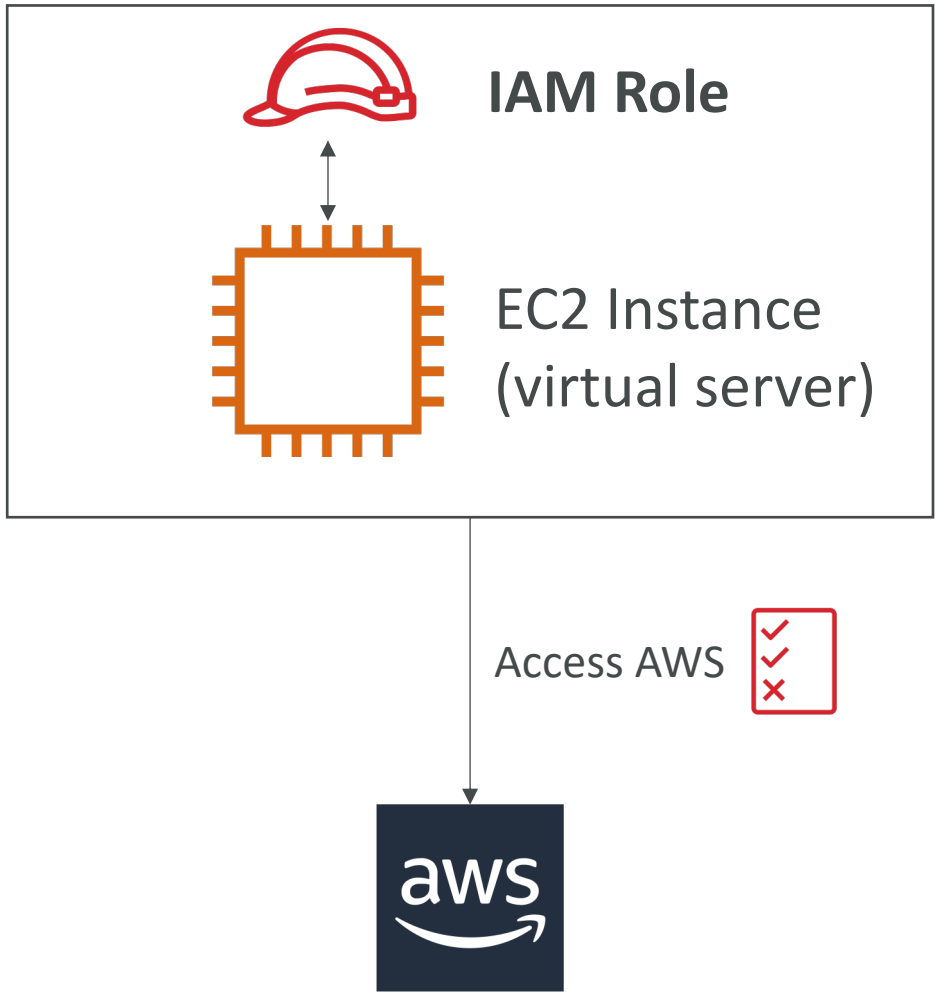
# What's the AWS SDK?

- AWS Software Development Kit (AWS SDK)
- Language-specific APIs (set of libraries)
- Enables you to access and manage AWS services programmatically
- Embedded within your application
- Supports
  - SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)
  - Mobile SDKs (Android, iOS, …)
  - IoT Device SDKs (Embedded C, Arduino, …)
- Example: AWS CLI is built on AWS SDK for Python


Your Application

# IAM Roles for Services

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign **permissions** to AWS services with **IAM Roles**
- Common roles:
  - EC2 Instance Roles
  - Lambda Function Roles
  - Roles for CloudFormation



# IAM Security Tools

- IAM Credentials Report (account-level)
  - a report that lists all your account's users and the status of their various credentials

- IAM Access Advisor (user-level)
  - Access advisor shows the service permissions granted to a user and when those services were last accessed.
  - You can use this information to revise your policies.

# IAM Guidelines & Best Practices

- Don't use the root account except for AWS account setup
- One physical user = One AWS **user**
- **Assign users to groups** and assign permissions to groups
- Create a **strong password policy**
- Use and enforce the use of **Multi Factor Authentication (MFA)**
- Create and use **Roles** for giving permissions to AWS services
- Use Access Keys for Programmatic Access (CLI / SDK)
- Audit permissions of your account with the IAM Credentials Report
- Never share IAM users & Access Keys

# Shared Responsibility Model for IAM


- Infrastructure (global network security)
- Configuration and vulnerability analysis
- Compliance validation

You

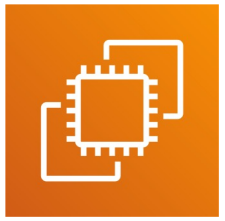
- Users, Groups, Roles, Policies management and monitoring
- Enable MFA on all accounts
- Rotate all your keys often
- Use IAM tools to apply appropriate permissions
- Analyze access patterns & review permissions

# IAM Section – Summary

- **Users:** mapped to a physical user, has a password for AWS Console
- **Groups:** contains users only
- **Policies:** JSON document that outlines permissions for users or groups
- **Roles:** for EC2 instances or AWS services
- **Security:** MFA + Password Policy
- **AWS CLI:** manage your AWS services using the command-line
- **AWS SDK:** manage your AWS services using a programming language
- **Access Keys:** access AWS using the CLI or SDK
- **Audit:** IAM Credential Reports & IAM Access Advisor

# EC2 Section

# Amazon EC2



- EC2 is one of the most popular of AWS' offering
- EC2 = Elastic Compute Cloud = Infrastructure as a Service
- It mainly consists in the capability of :
  - Renting virtual machines (EC2)
  - Storing data on virtual drives (EBS)
  - Distributing load across machines (ELB)
  - Scaling the services using an auto-scaling group (ASG)

- Knowing EC2 is fundamental to understand how the Cloud works

# EC2 sizing & configuration options

- Operating System (**OS**): Linux, Windows or Mac OS
- How much compute power & cores (**CPU**)
- How much random-access memory (**RAM**)
- How much storage space:
  - Network-attached (**EBS & EFS**)
  - hardware (**EC2 Instance Store**)
- Network card: speed of the card, Public IP address
- Firewall rules: **security group**
- Bootstrap script (configure at first launch): EC2 User Data

# EC2 User Data

- It is possible to bootstrap our instances using an EC2 User data script.
- bootstrapping means launching commands when a machine starts
- That script is only run once at the instance first start
- EC2 user data is used to automate boot tasks such as:
  - Installing updates
  - Installing software
  - Downloading common files from the internet
  - Anything you can think of
- The EC2 User Data Script runs with the root user

# Hands-On:
# Launching an EC2 Instance running Linux

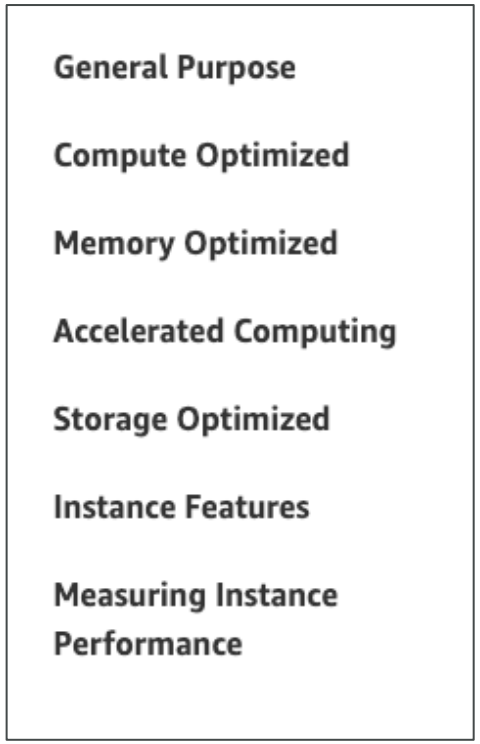
- We'll be launching our first virtual server using the AWS Console
- We'll get a first high-level approach to the various parameters
- We'll see that our web server is launched using EC2 user data
- We'll learn how to start / stop / terminate our instance.

# EC2 Instance Types - Overview

- You can use different types of EC2 instances that are optimised for different use cases (https://aws.amazon.com/ec2/instance-types/)
- AWS has the following naming convention:

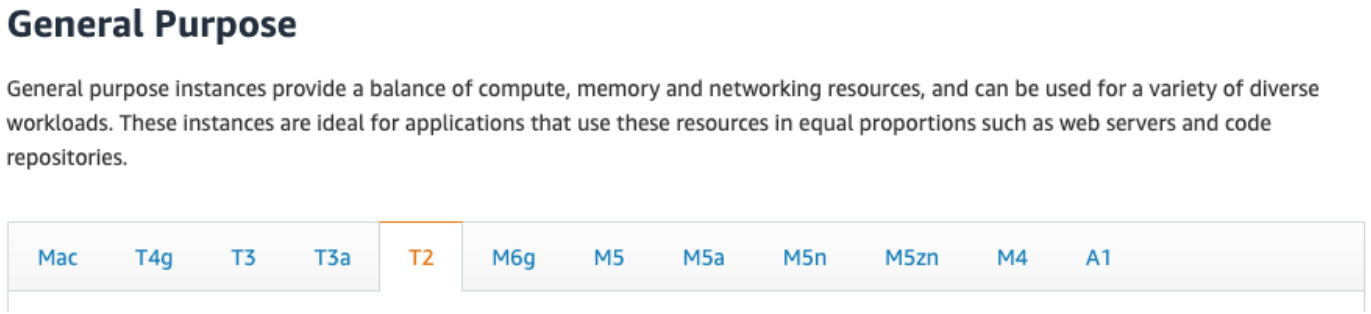
m5.2xlarge

- m: instance class
- 5: generation (AWS improves them over time)
- 2xlarge: size within the instance class



# EC2 Instance Types – General Purpose

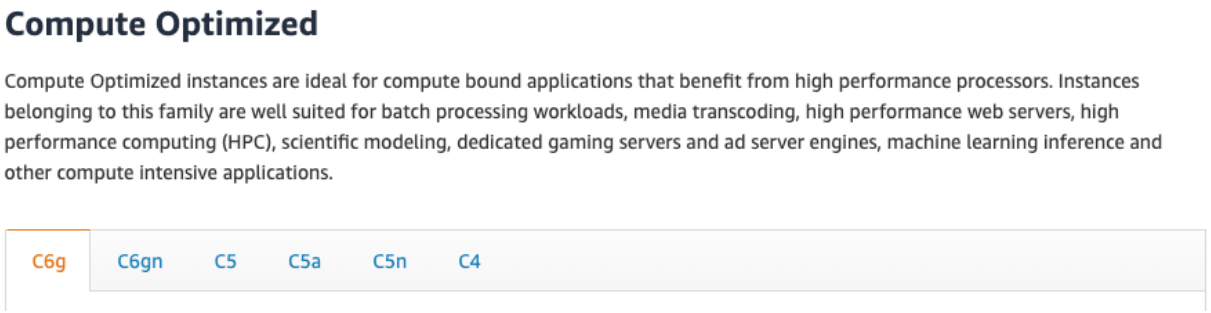
- Great for a diversity of workloads such as web servers or code repositories
- Balance between:
  - Compute
  - Memory
  - Networking
- In the course, we will be using the t2.micro which is a General Purpose EC2 instance




** this list will evolve over time, please check the AWS website for the latest information*

# EC2 Instance Types – Compute Optimized

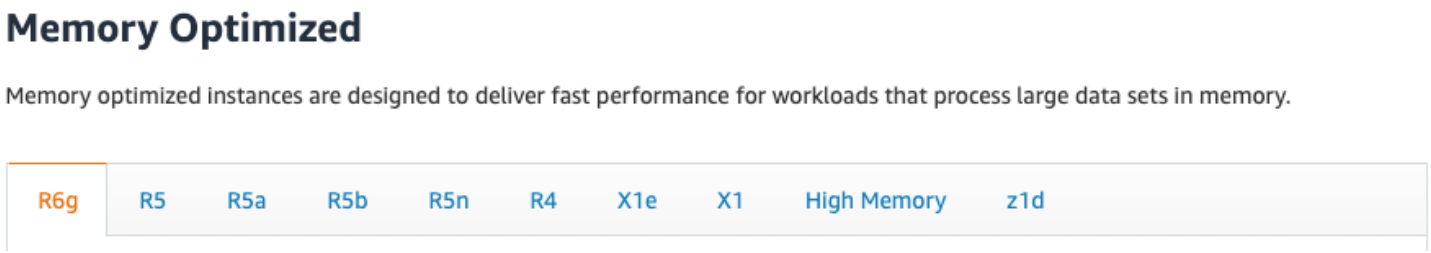
- Great for compute-intensive tasks that require high performance processors:
  - Batch processing workloads
  - Media transcoding
  - High performance web servers
  - High performance computing (HPC)
  - Scientific modeling & machine learning
  - Dedicated gaming servers




** this list will evolve over time, please check the AWS website for the latest information*

# EC2 Instance Types – Memory Optimized

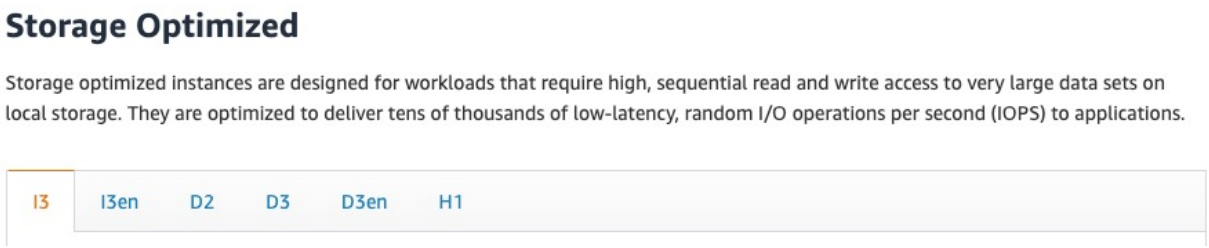
- Fast performance for workloads that process large data sets in memory
- Use cases:
  - High performance, relational/non-relational databases
  - Distributed web scale cache stores
  - In-memory databases optimized for BI (business intelligence)
  - Applications performing real-time processing of big unstructured data




** this list will evolve over time, please check the AWS website for the latest information*

# EC2 Instance Types – Storage Optimized

- Great for storage-intensive tasks that require high, sequential read and write access to large data sets on local storage
- Use cases:
  - High frequency online transaction processing (OLTP) systems
  - Relational & NoSQL databases
  - Cache for in-memory databases (for example, Redis)
  - Data warehousing applications
  - Distributed file systems




** this list will evolve over time, please check the AWS website for the latest information*

# EC2 Instance Types: example

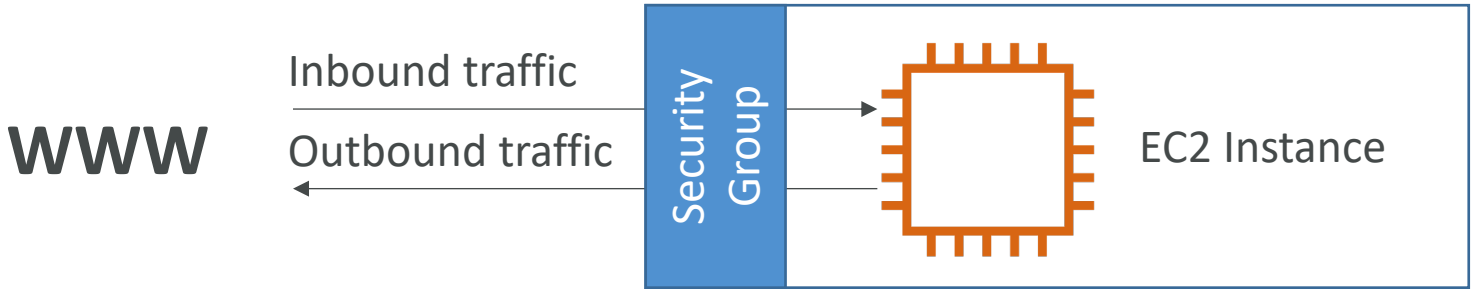
| Instance | vCPU | Mem (GiB) | Storage | Network Performance | EBS Bandwidth (Mbps) |
|---|---|---|---|---|---|
| t2.micro | 1 | 1 | EBS-Only | Low to Moderate | |
| t2.xlarge | 4 | 16 | EBS-Only | Moderate | |
| c5d.4xlarge | 16 | 32 | 1 x 400 NVMe SSD | Up to 10 Gbps | 4,750 |
| r5.16xlarge | 64 | 512 | EBS Only | 20 Gbps | 13,600 |
| m5.8xlarge | 32 | 128 | EBS Only | 10 Gbps | 6,800 |

**t2.micro is part of the AWS free tier (up to 750 hours per month)**

Great website: https://instances.vantage.sh

# Introduction to Security Groups

- Security Groups are the fundamental of network security in AWS
- They control how traffic is allowed into or out of our EC2 Instances.



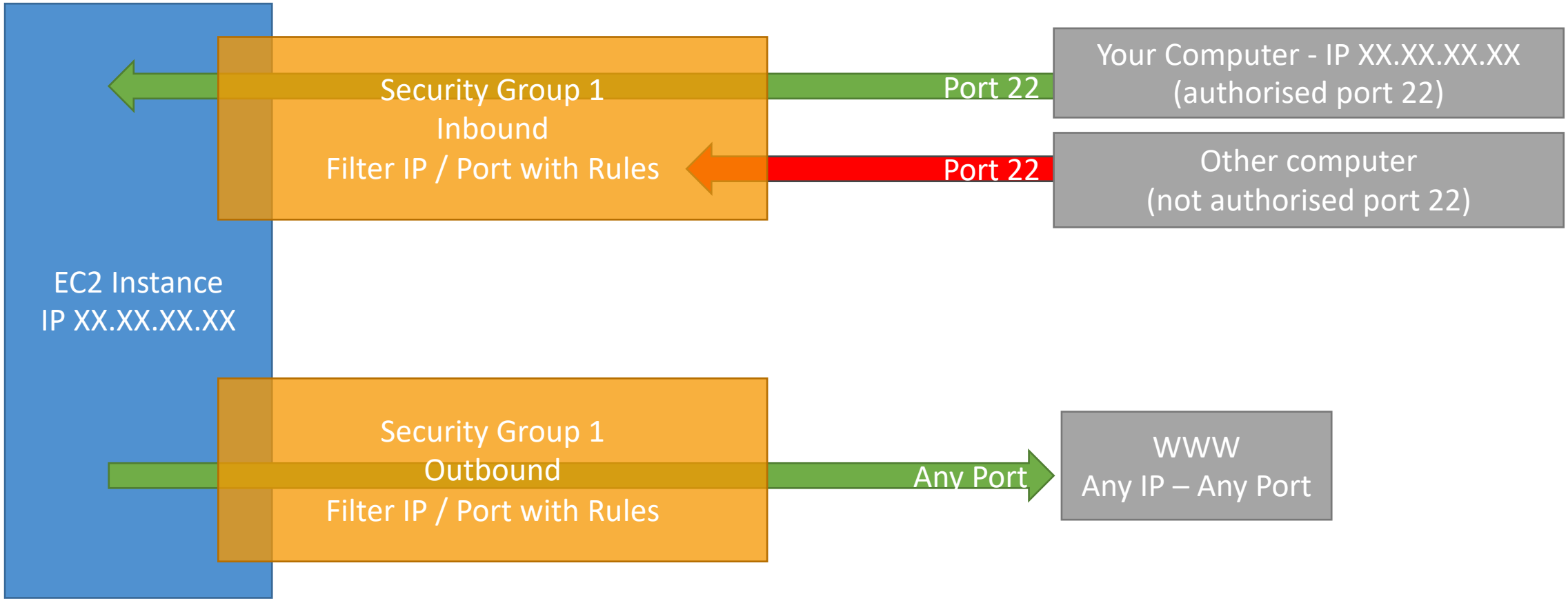

- Security groups only contain allow rules
- Security groups rules can reference by IP or by security group

# Security Groups Deeper Dive

- Security groups are acting as a "firewall" on EC2 instances
- They regulate:
  - Access to Ports
  - Authorised IP ranges – IPv4 and IPv6
  - Control of inbound network (from other to the instance)
  - Control of outbound network (from the instance to other)

| Type | Protocol | Port Range | Source | Description |
|---|---|---|---|---|
| HTTP | TCP | 80 | 0.0.0.0/0 | test http page |
| SSH | TCP | 22 | 122.149.196.85/32 | |
| Custom TCP Rule | TCP | 4567 | 0.0.0.0/0 | java app |

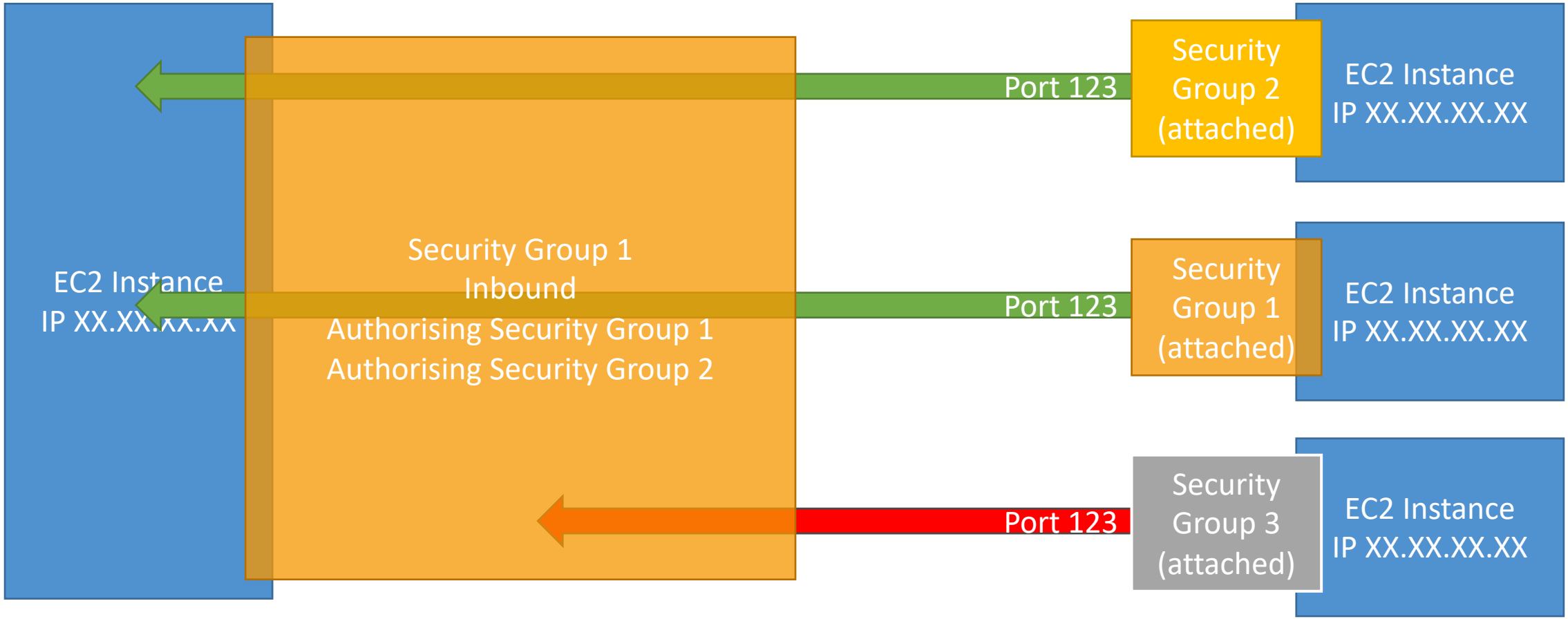
# Security Groups Diagram



# Security Groups
# Good to know

- Can be attached to multiple instances
- Locked down to a region / VPC combination
- Does live "outside" the EC2 – if traffic is blocked the EC2 instance won't see it
- It's good to maintain one separate security group for SSH access
- If your application is not accessible (time out), then it's a security group issue
- If your application gives a "connection refused" error, then it's an application error or it's not launched
- All inbound traffic is blocked by default
- All outbound traffic is authorised by default

# Referencing other security groups Diagram



# Classic Ports to know

- 22 = SSH (Secure Shell) - log into a Linux instance
- 21 = FTP (File Transfer Protocol) – upload files into a file share
- 22 = SFTP (Secure File Transfer Protocol) – upload files using SSH
- 80 = HTTP – access unsecured websites
- 443 = HTTPS – access secured websites
- 3389 = RDP (Remote Desktop Protocol) – log into a Windows instance

# SSH Summary Table

| | SSH | Putty | EC2 Instance Connect |
|---|---|---|---|
| Mac | ✓ | | ✓ |
| Linux | ✓ | | ✓ |
| Windows < 10 | | ✓ | ✓ |
| Windows >= 10 | ✓ | ✓ | ✓ |

# Which Lectures to watch

- **Mac / Linux:**
  - SSH on Mac/Linux lecture
- **Windows:**
  - Putty Lecture
  - **If Windows 10:** SSH on Windows 10 lecture
- **All:**
  - EC2 Instance Connect lecture

# SSH troubleshooting

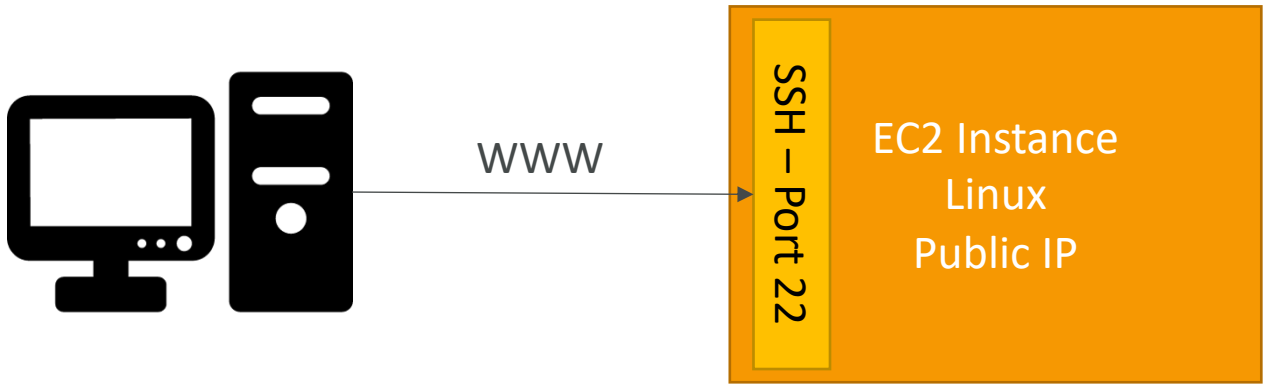
- Students have the most problems with SSH

- If things don't work…
  1. Re-watch the lecture. You may have missed something
  2. Read the troubleshooting guide
  3. Try EC2 Instance Connect

- If one method works (SSH, Putty or EC2 Instance Connect) you're good
- If no method works, that's okay, the course won't use SSH much

# How to SSH into your EC2 Instance
## Linux / Mac OS X

- We'll learn how to SSH into your EC2 instance using Linux / Mac
- SSH is one of the most important function. It allows you to control a remote machine, all using the command line.

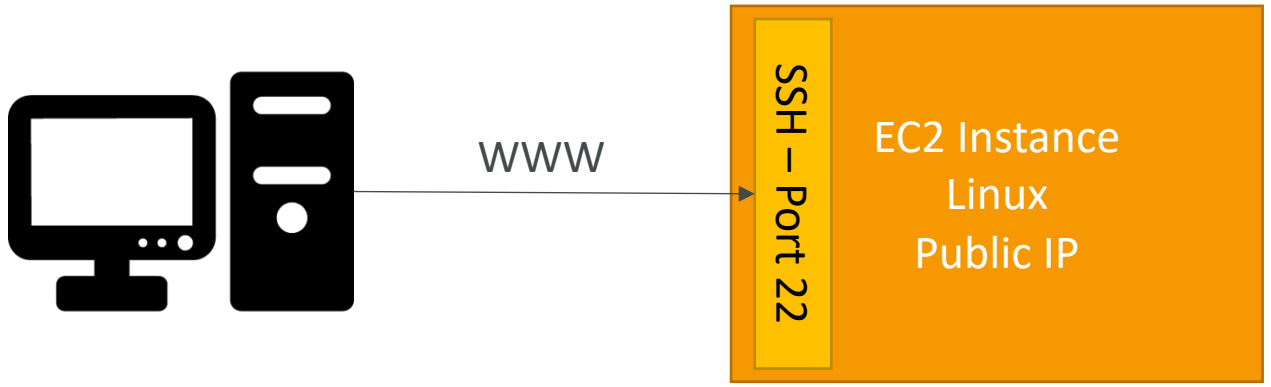



- We will see how we can configure OpenSSH ~/.ssh/config to facilitate the SSH into our EC2 instances

# How to SSH into your EC2 Instance

## Windows

- We'll learn how to SSH into your EC2 instance using Windows
- SSH is one of the most important function. It allows you to control a remote machine, all using the command line.




- We will configure all the required parameters necessary for doing SSH on Windows using the free tool Putty.

# EC2 Instance Connect

- Connect to your EC2 instance within your browser
- No need to use your key file that was downloaded
- The "magic" is that a temporary key is uploaded onto EC2 by AWS

- Works only out-of-the-box with Amazon Linux 2

- Need to make sure the port 22 is still opened!

# EC2 Instances Purchasing Options

- **On-Demand Instances** – short workload, predictable pricing, pay by second
- **Reserved** (1 & 3 years)
  - **Reserved Instances** – long workloads
  - **Convertible Reserved Instances** – long workloads with flexible instances
- **Savings Plans** (1 & 3 years) –commitment to an amount of usage, long workload
- **Spot Instances** – short workloads, cheap, can lose instances (less reliable)
- **Dedicated Hosts** – book an entire physical server, control instance placement
- **Dedicated Instances** – no other customers will share your hardware
- **Capacity Reservations** – reserve capacity in a specific AZ for any duration

# EC2 On Demand

- Pay for what you use:
  - Linux or Windows - billing per second, after the first minute
  - All other operating systems - billing per hour
- Has the highest cost but no upfront payment
- No long-term commitment

- Recommended for **short-term** and **un-interrupted workloads**, where you can't predict how the application will behave

# EC2 Reserved Instances

- Up to 72% discount compared to On-demand
- You reserve a specific instance attributes **(Instance Type, Region, Tenancy, OS)**
- **Reservation Period** – **1 year** (+discount) or **3 years** (+++discount)
- **Payment Options** – **No Upfront** (+), **Partial Upfront** (++), **All Upfront** (+++)
- **Reserved Instance's Scope** – **Regional** or **Zonal** (reserve capacity in an AZ)
- Recommended for steady-state usage applications (think database)
- You can buy and sell in the Reserved Instance Marketplace

- **Convertible Reserved Instance**
  - Can change the EC2 instance type, instance family, OS, scope and tenancy
  - Up to 66% discount

**Note:** the % discounts are different from the video as AWS change them over time – the exact numbers are not needed for the exam. This is just for illustrative purposes ☺

# EC2 Savings Plans

- Get a discount based on long-term usage (up to 72% - same as RIs)
- Commit to a certain type of usage ($10/hour for 1 or 3 years)
- Usage beyond EC2 Savings Plans is billed at the On-Demand price

- Locked to a specific instance family & AWS region (e.g., M5 in us-east-1)
- Flexible across:
  - Instance Size (e.g., m5.xlarge, m5.2xlarge)
  - OS (e.g., Linux, Windows)
  - Tenancy (Host, Dedicated, Default)

# EC2 Spot Instances

- Can get a **discount of up to 90%** compared to On-demand
- Instances that you can "lose" at any point of time if your max price is less than the current spot price
- The **MOST cost-efficient** instances in AWS

- **Useful for workloads that are resilient to failure**
  - Batch jobs
  - Data analysis
  - Image processing
  - Any **distributed** workloads
  - Workloads with a flexible start and end time

- **Not suitable for critical jobs or databases**

# EC2 Dedicated Hosts

- A physical server with EC2 instance capacity fully dedicated to your use
- Allows you address **compliance requirements** and **use your existing server-bound software licenses** (per-socket, per-core, pe—VM software licenses)
- Purchasing Options:
  - **On-demand** – pay per second for active Dedicated Host
  - **Reserved** - 1 or 3 years (No Upfront, Partial Upfront, All Upfront)
- The most expensive option

- Useful for software that have complicated licensing model (BYOL – Bring Your Own License)
- Or for companies that have strong regulatory or compliance needs

# EC2 Dedicated Instances

- Instances run on hardware that's dedicated to you
- May share hardware with other instances in same account
- No control over instance placement (can move hardware after Stop / Start)

| Characteristic | Dedicated Instances | Dedicated Hosts |
|---|---|---|
| Enables the use of dedicated physical servers | X | X |
| Per instance billing (subject to a $2 per region fee) | X | |
| Per host billing | | X |
| Visibility of sockets, cores, host ID | | X |
| Affinity between a host and instance | | X |
| Targeted instance placement | | X |
| Automatic instance placement | X | X |
| Add capacity using an allocation request | | X |

# EC2 Capacity Reservations

- Reserve **On-Demand** instances capacity in a specific AZ for any duration
- You always have access to EC2 capacity when you need it
- **No time commitment** (create/cancel anytime), **no billing discounts**
- Combine with Regional Reserved Instances and Savings Plans to benefit from billing discounts
- You're charged at On-Demand rate whether you run instances or not

- Suitable for short-term, uninterrupted workloads that needs to be in a specific AZ

# Which purchasing option is right for me?

- **On demand:** coming and staying in resort whenever we like, we pay the full price
- **Reserved:** like planning ahead and if we plan to stay for a long time, we may get a good discount.
- **Savings Plans:** pay a certain amount per hour for certain period and stay in any room type (e.g., King, Suite, Sea View, …)
- **Spot instances:** the hotel allows people to bid for the empty rooms and the highest bidder keeps the rooms. You can get kicked out at any time
- **Dedicated Hosts:** We book an entire building of the resort
- **Capacity Reservations:** you book a room for a period with full price even you don't stay in it

# Price Comparison
# Example – m4.large – us-east-1

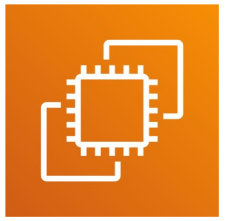
| Price Type | Price (per hour) |
|---|---|
| On-Demand | $0.10 |
| Spot Instance (Spot Price) | $0.038 - $0.039 (up to 61% off) |
| Reserved Instance (1 year) | $0.062 (No Upfront) - $0.058 (All Upfront) |
| Reserved Instance (3 years) | $0.043 (No Upfront) - $0.037 (All Upfront) |
| EC2 Savings Plan (1 year) | $0.062 (No Upfront) - $0.058 (All Upfront) |
| Reserved **Convertible** Instance (1 year) | $0.071 (No Upfront) - $0.066 (All Upfront) |
| Dedicated Host | On-Demand Price |
| Dedicated Host Reservation | Up to 70% off |
| Capacity Reservations | On-Demand Price |

# Shared Responsibility Model for EC2


- Infrastructure (global network security)
- Isolation on physical hosts
- Replacing faulty hardware
- Compliance validation

- Security Groups rules
- Operating-system patches and updates
- Software and utilities installed on the EC2 instance
- IAM Roles assigned to EC2 & IAM user access management
- Data security on your instance

# EC2 Section – Summary



- **EC2 Instance:** AMI (OS) + Instance Size (CPU + RAM) + Storage + security groups + EC2 User Data
- **Security Groups:** Firewall attached to the EC2 instance
- **EC2 User Data:** Script launched at the first start of an instance
- **SSH:** start a terminal into our EC2 Instances (port 22)
- **EC2 Instance Role:** link to IAM roles
- **Purchasing Options:** On-Demand, Spot, Reserved (Standard + Convertible + Scheduled), Dedicated Host, Dedicated Instance

# EC2 Instance Storage Section

# What's an EBS Volume?

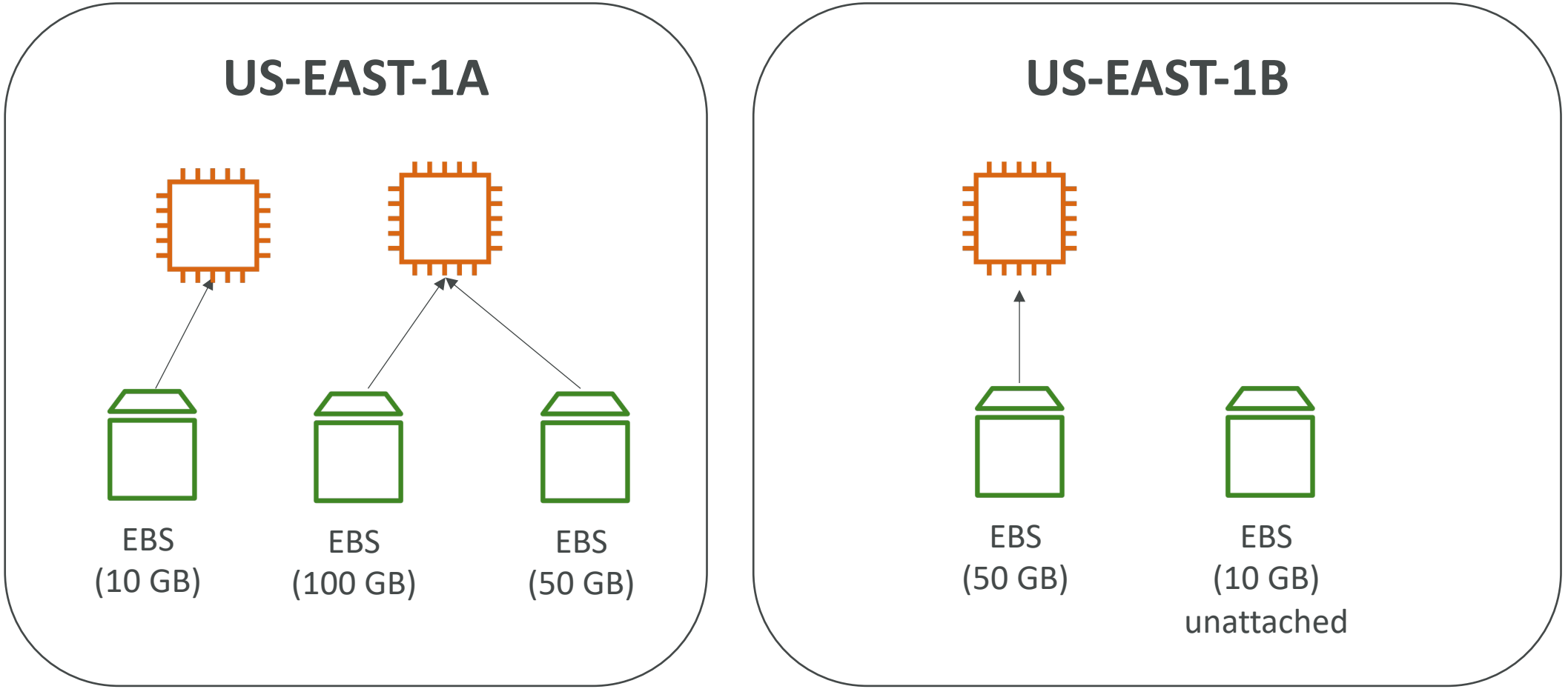
- An EBS (Elastic Block Store) Volume is a network drive you can attach to your instances while they run
- It allows your instances to persist data, even after their termination
- **They can only be mounted to one instance at a time** (at the CCP level)
- They are bound to **a specific availability zone**

- Analogy: Think of them as a "network USB stick"
- Free tier: 30 GB of free EBS storage of type General Purpose (SSD) or Magnetic per month

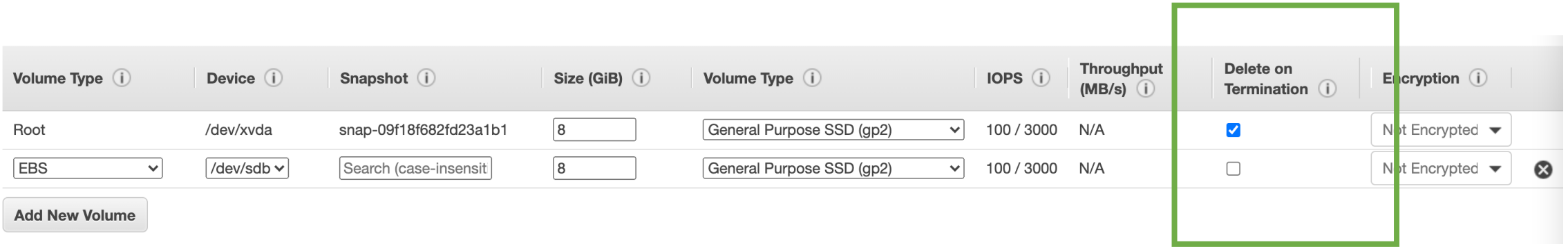
# EBS Volume

- It's a network drive (i.e. not a physical drive)
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
  - An EBS Volume in us-east-1a cannot be attached to us-east-1b
  - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
  - You get billed for all the provisioned capacity
  - You can increase the capacity of the drive over time

# EBS Volume - Example



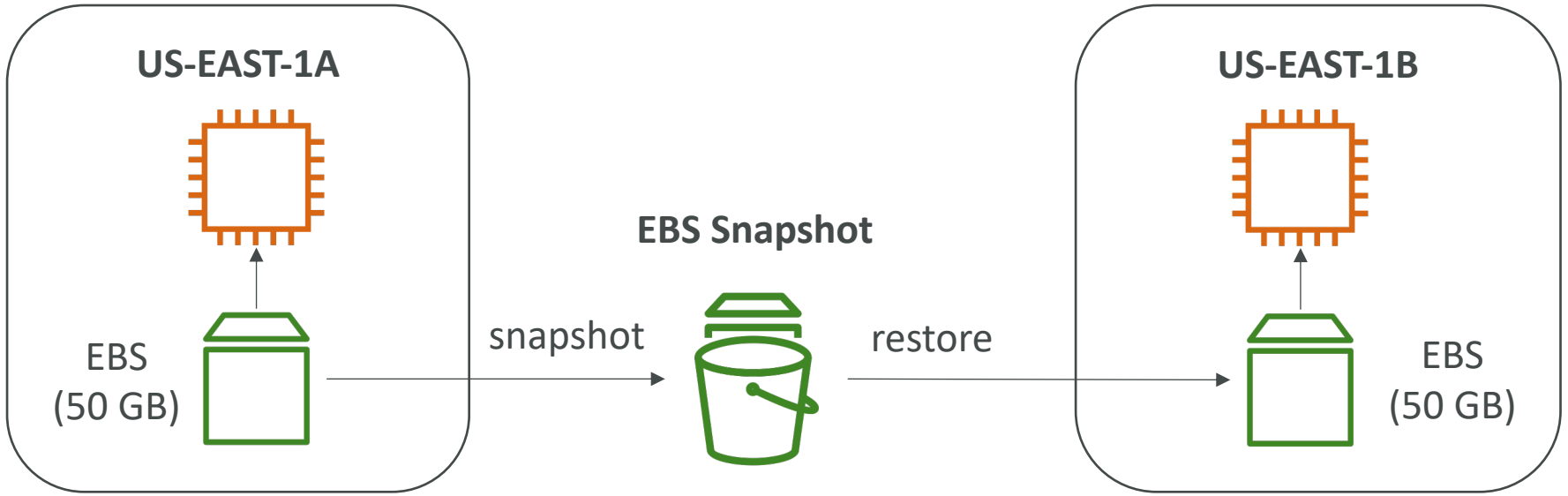
# EBS – Delete on Termination attribute




- Controls the EBS behaviour when an EC2 instance terminates
  - By default, the root EBS volume is deleted (attribute enabled)
  - By default, any other attached EBS volume is not deleted (attribute disabled)
- This can be controlled by the AWS console / AWS CLI
- Use case: preserve root volume when instance is terminated

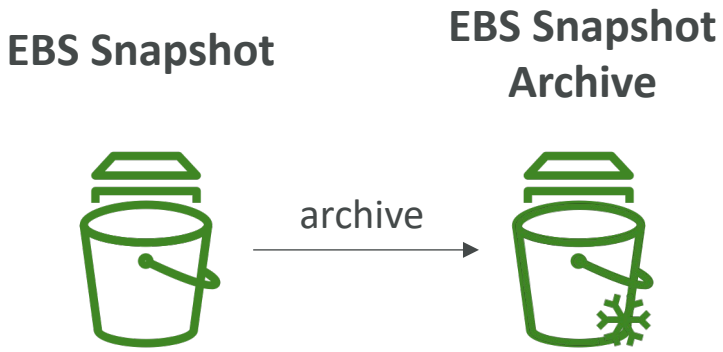
# EBS Snapshots

- Make a backup (snapshot) of your EBS volume at a point in time
- Not necessary to detach volume to do snapshot, but recommended
- Can copy snapshots across AZ or Region

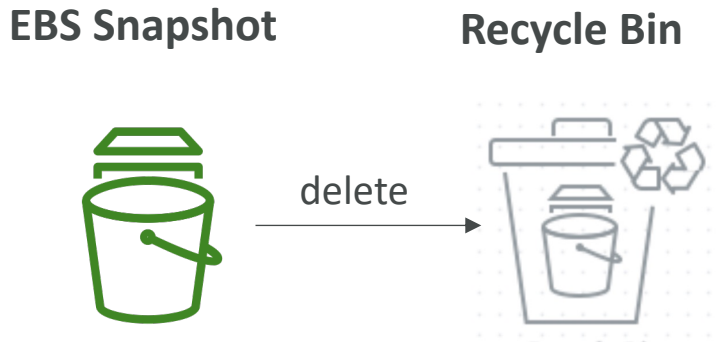


# EBS Snapshots Features

- EBS Snapshot Archive
  - Move a Snapshot to an "archive tier" that is 75% cheaper
  - Takes within 24 to 72 hours for restoring the archive




- Recycle Bin for EBS Snapshots
  - Setup rules to retain deleted snapshots so you can recover them after an accidental deletion
  - Specify retention (from 1 day to 1 year)

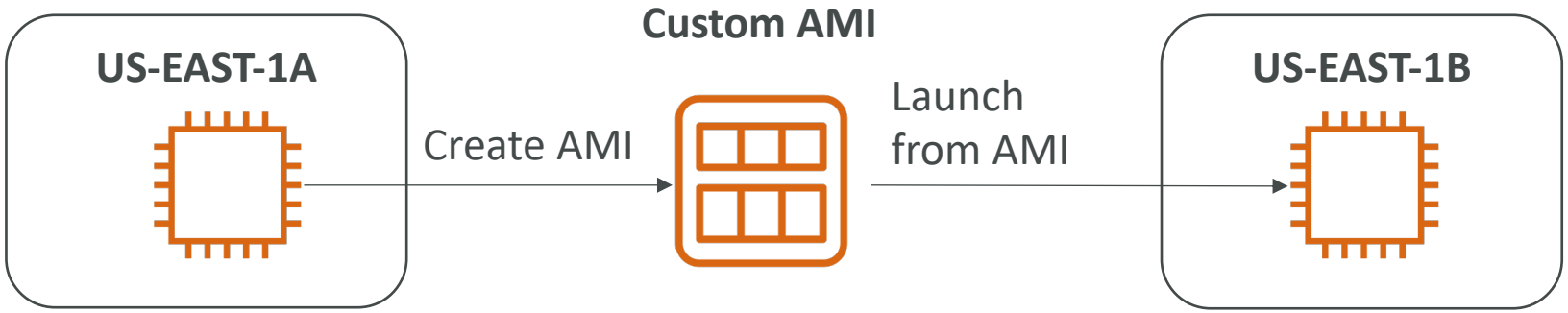


# AMI Overview

- AMI = Amazon Machine Image
- AMI are a **customization** of an EC2 instance
  - You add your own software, configuration, operating system, monitoring…
  - Faster boot / configuration time because all your software is pre-packaged
- AMI are built for a **specific region** (and can be copied across regions)
- You can launch EC2 instances from:
  - **A Public AMI:** AWS provided
  - **Your own AMI:** you make and maintain them yourself
  - **An AWS Marketplace AMI:** an AMI someone else made (and potentially sells)

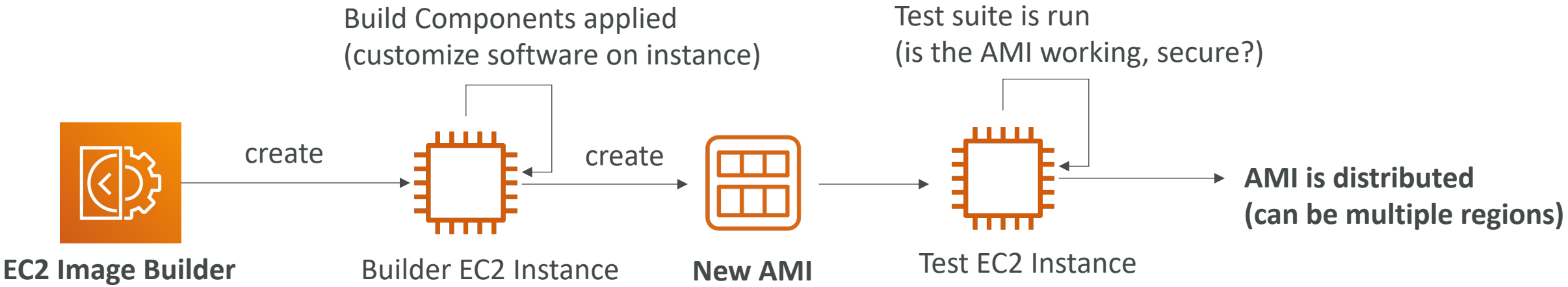
# AMI Process (from an EC2 instance)

- Start an EC2 instance and customize it
- Stop the instance (for data integrity)
- Build an AMI – this will also create EBS snapshots
- Launch instances from other AMIs

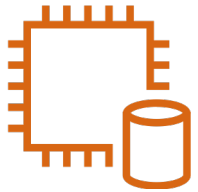


# EC2 Image Builder

- Used to automate the creation of Virtual Machines or container images
- => Automate the creation, maintain, validate and test **EC2 AMIs**
- Can be run on a schedule (weekly, whenever packages are updated, etc…)
- Free service (only pay for the underlying resources)



# EC2 Instance Store



- EBS volumes are **network drives** with good but "limited" performance
- If you need a high-performance hardware disk, use EC2 Instance Store

- Better I/O performance
- EC2 Instance Store lose their storage if they're stopped (ephemeral)
- Good for buffer / cache / scratch data / temporary content
- Risk of data loss if hardware fails
- Backups and Replication are your responsibility

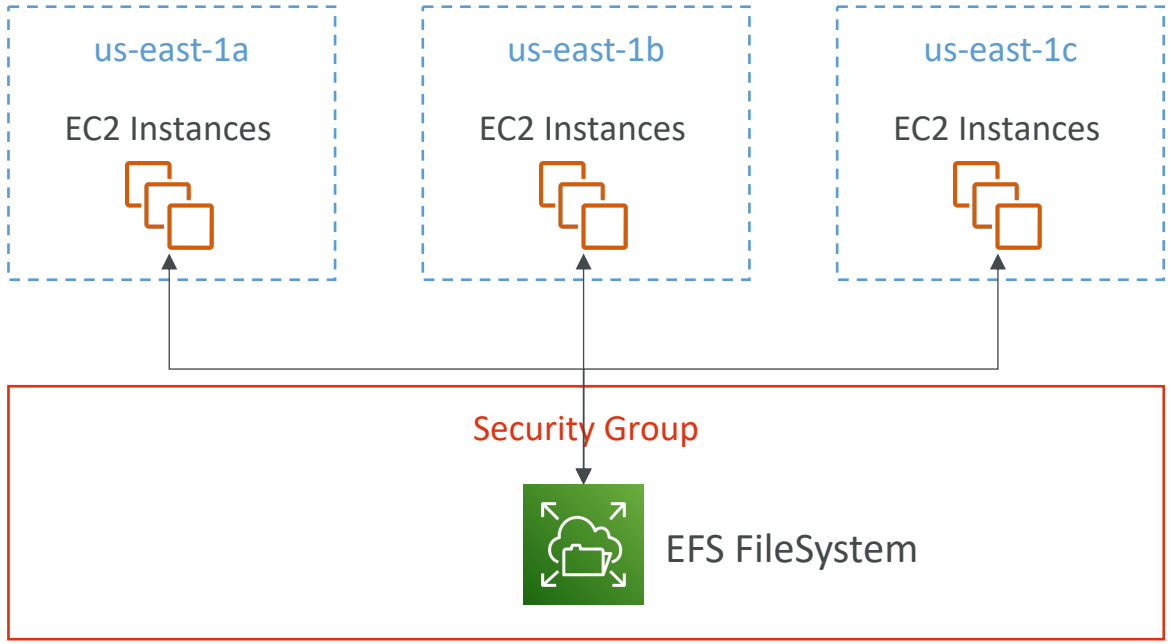
# Local EC2 Instance Store

**Very high IOPS**

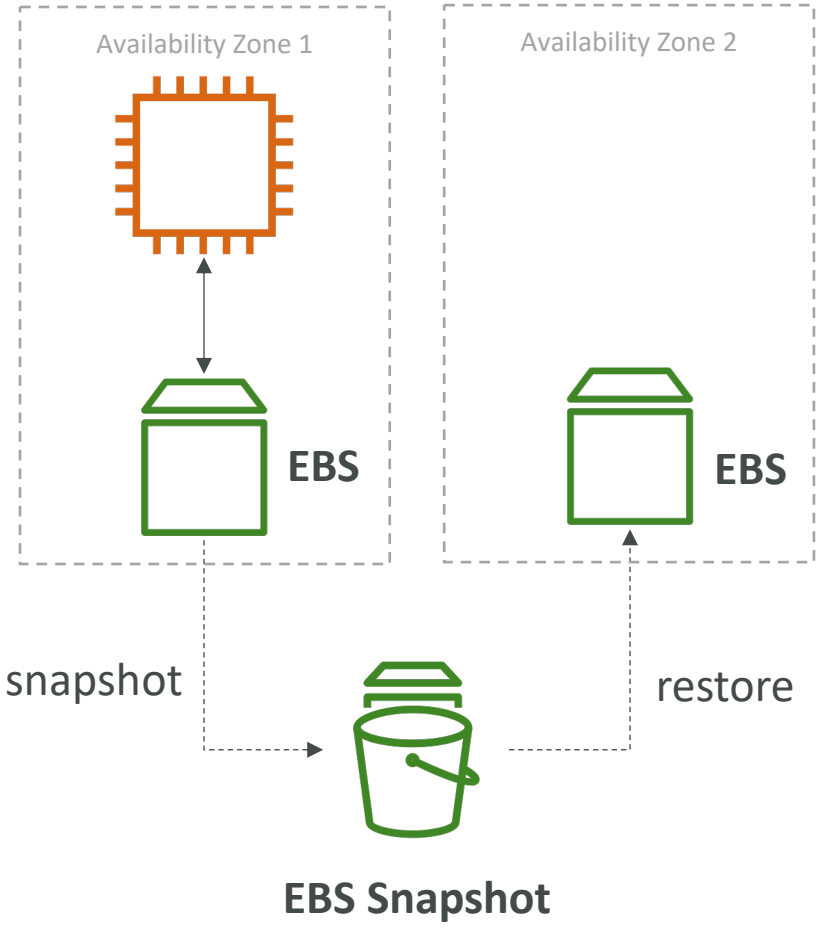
| Instance Size | 100% Random Read IOPS | Write IOPS |
|---|---|---|
| i3.large * | 100,125 | 35,000 |
| i3.xlarge * | 206,250 | 70,000 |
| i3.2xlarge | 412,500 | 180,000 |
| i3.4xlarge | 825,000 | 360,000 |
| i3.8xlarge | 1.65 million | 720,000 |
| i3.16xlarge | 3.3 million | 1.4 million |
| i3.metal | 3.3 million | 1.4 million |
| i3en.large * | 42,500 | 32,500 |
| i3en.xlarge * | 85,000 | 65,000 |
| i3en.2xlarge * | 170,000 | 130,000 |
| i3en.3xlarge | 250,000 | 200,000 |
| i3en.6xlarge | 500,000 | 400,000 |
| i3en.12xlarge | 1 million | 800,000 |
| i3en.24xlarge | 2 million | 1.6 million |
| i3en.metal | 2 million | 1.6 million |

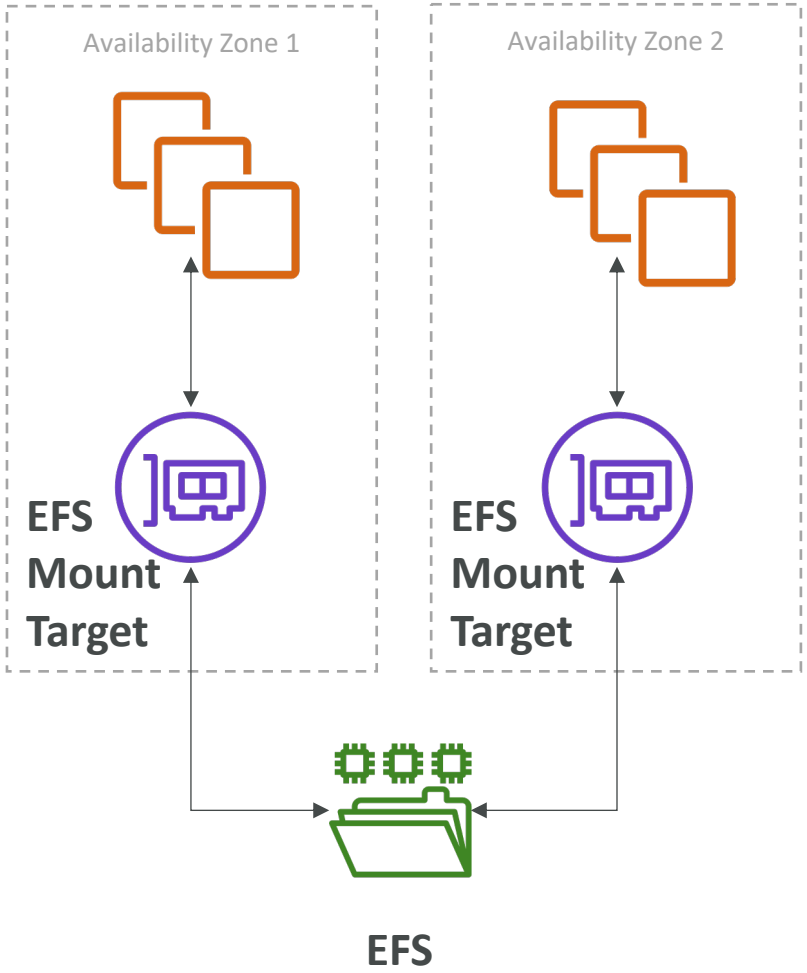
# EFS – Elastic File System

- Managed NFS (network file system) that **can be mounted on 100s of EC2**
- EFS works with **Linux** EC2 instances in **multi-AZ**
- Highly available, scalable, expensive (3x gp2), pay per use, no capacity planning



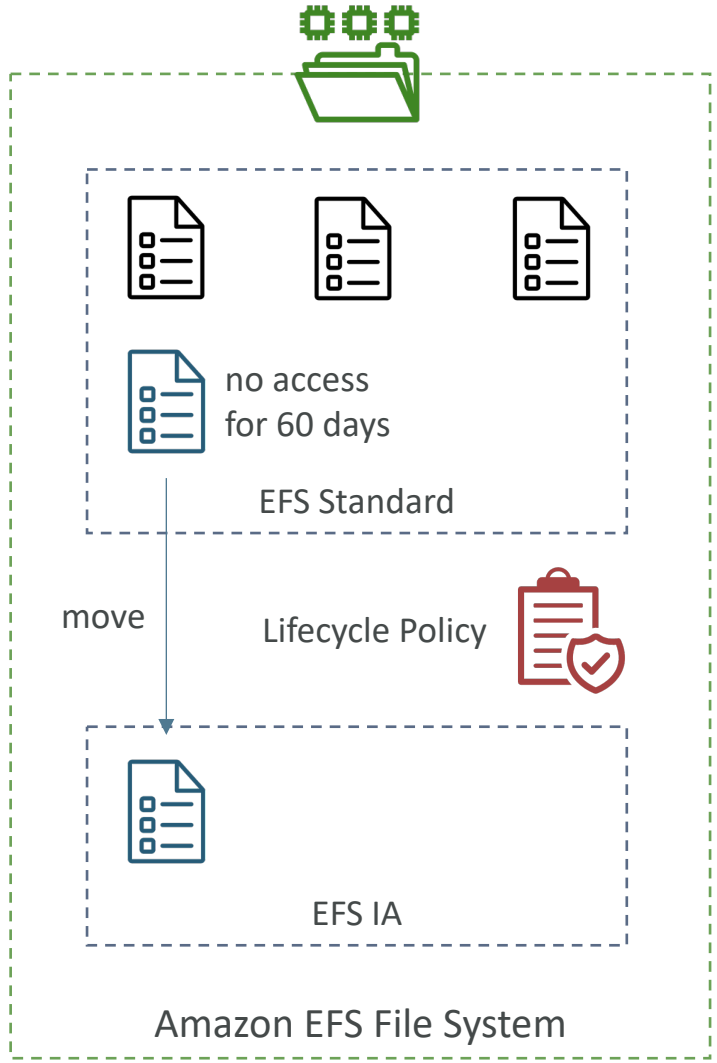
# EBS vs EFS



# EFS Infrequent Access (EFS-IA)

- **Storage class** that is cost-optimized for files not accessed every day
- Up to 92% lower cost compared to EFS Standard
- EFS will automatically move your files to EFS-IA based on the last time they were accessed
- Enable EFS-IA with a Lifecycle Policy
- Example: move files that are not accessed for 60 days to EFS-IA
- Transparent to the applications accessing EFS



# Shared Responsibility Model for EC2 Storage


- Infrastructure
- Replication for data for EBS volumes & EFS drives
- Replacing faulty hardware
- Ensuring their employees cannot access your data

- Setting up backup / snapshot procedures
- Setting up data encryption
- Responsibility of any data on the drives
- Understanding the risk of using EC2 Instance Store

# Amazon FSx – Overview


- Launch 3rd party high-performance file systems on AWS
- Fully managed service

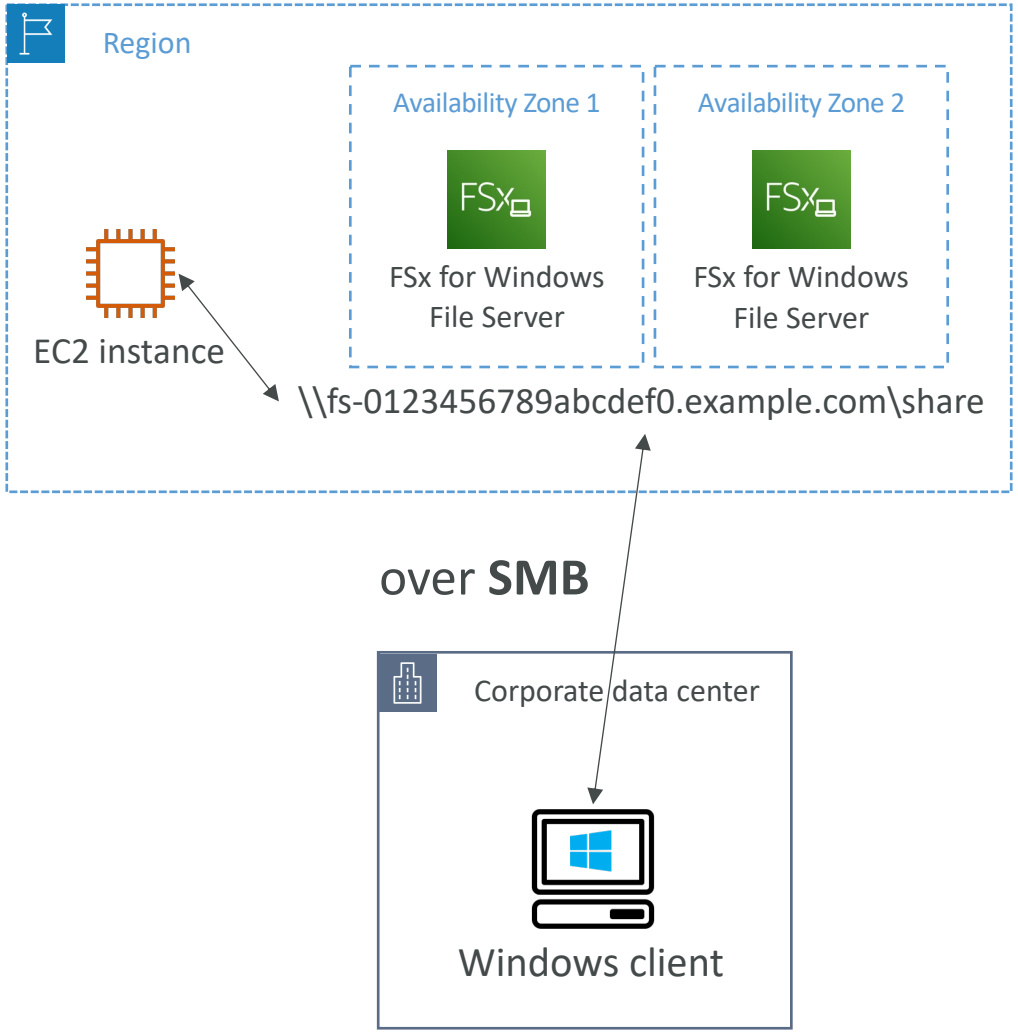

**FSx for Lustre**


**FSx for Windows File Server**


**FSx for NetApp ONTAP**

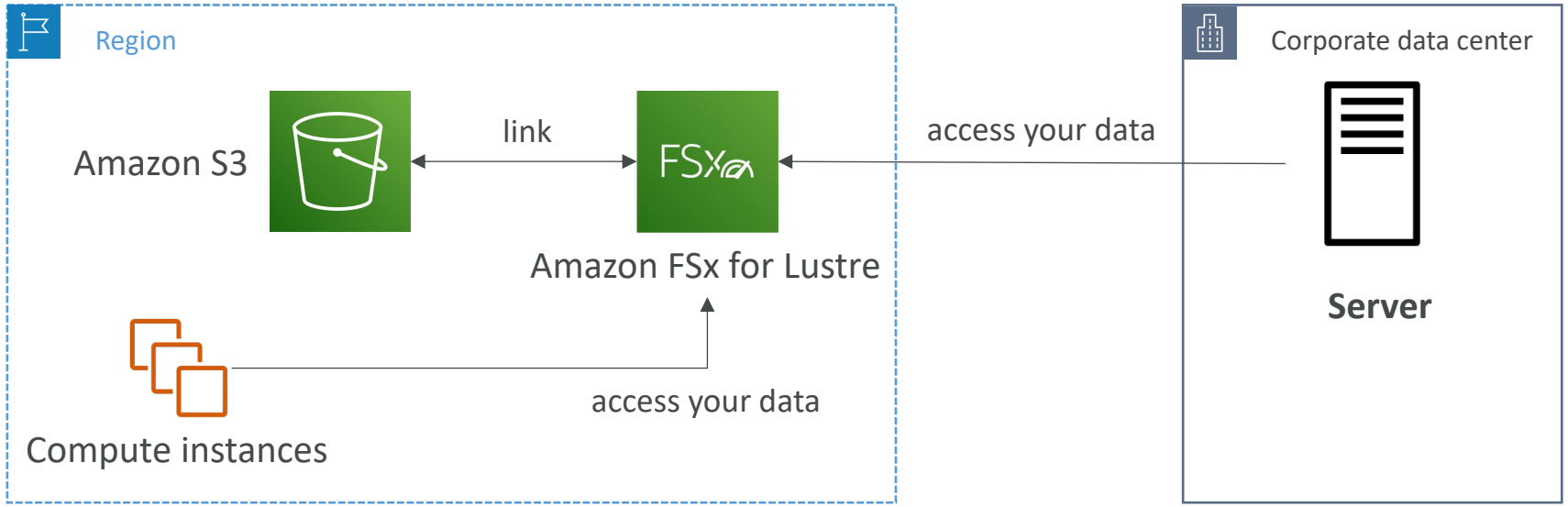
# Amazon FSx for Windows File Server


- A fully managed, highly reliable, and scalable **Windows native** shared file system
- Built on **Windows File Server**
- Supports **SMB protocol** & Windows NTFS
- Integrated with Microsoft Active Directory
- Can be accessed from AWS or your on-premise infrastructure



# Amazon FSx for Lustre


- A fully managed, high-performance, scalable file storage for **High Performance Computing (HPC)**
- The name Lustre is derived from "Linux" and "cluster"
- Machine Learning, Analytics, Video Processing, Financial Modeling, …
- Scales up to 100s GB/s, millions of IOPS, sub-ms latencies



# EC2 Instance Storage - Summary

- EBS volumes:
  - network drives attached to one EC2 instance at a time
  - Mapped to an Availability Zones
  - Can use EBS Snapshots for backups / transferring EBS volumes across AZ
- AMI: create ready-to-use EC2 instances with our customizations
- EC2 Image Builder: automatically build, test and distribute AMIs
- EC2 Instance Store:
  - High performance hardware disk attached to our EC2 instance
  - Lost if our instance is stopped / terminated
- EFS: network file system, can be attached to 100s of instances in a region
- EFS-IA: cost-optimized storage class for infrequent accessed files
- FSx for Windows: Network File System for Windows servers
- FSx for Lustre: High Performance Computing Linux file system

# Elastic Load Balancing & Auto Scaling Groups Section

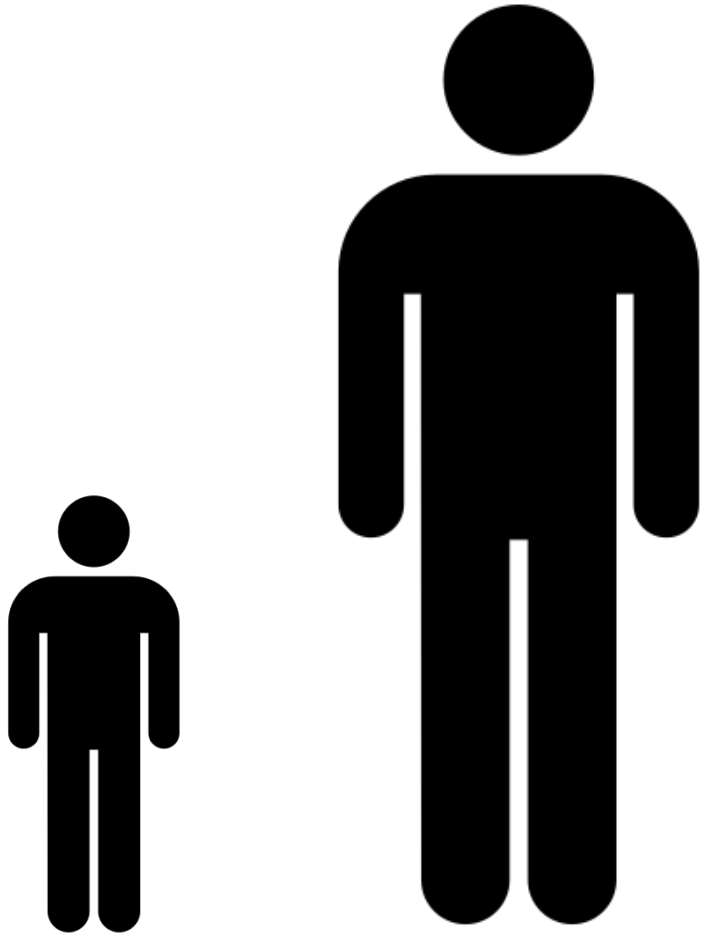
# Scalability & High Availability

- Scalability means that an application / system can handle greater loads by adapting.
- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (= elasticity)
- **Scalability is linked but different to High Availability**

- Let's deep dive into the distinction, using a call center as an example

# Vertical Scalability

- Vertical Scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- There's usually a limit to how much you can vertically scale (hardware limit)

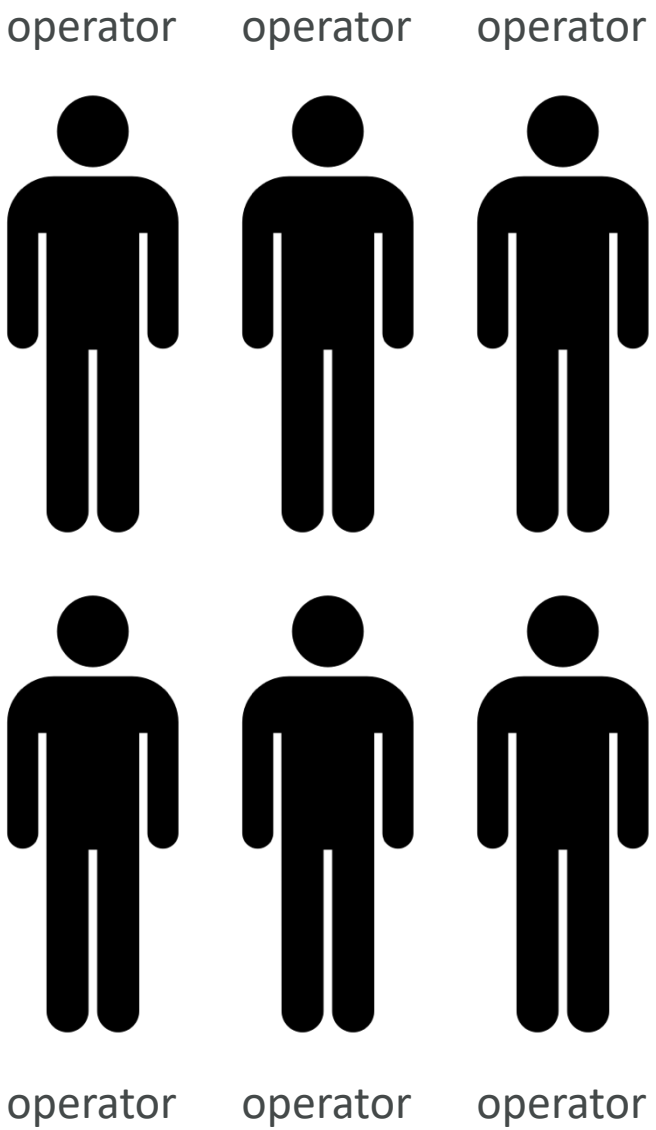


# Horizontal Scalability

- Horizontal Scalability means increasing the number of instances / systems for your application

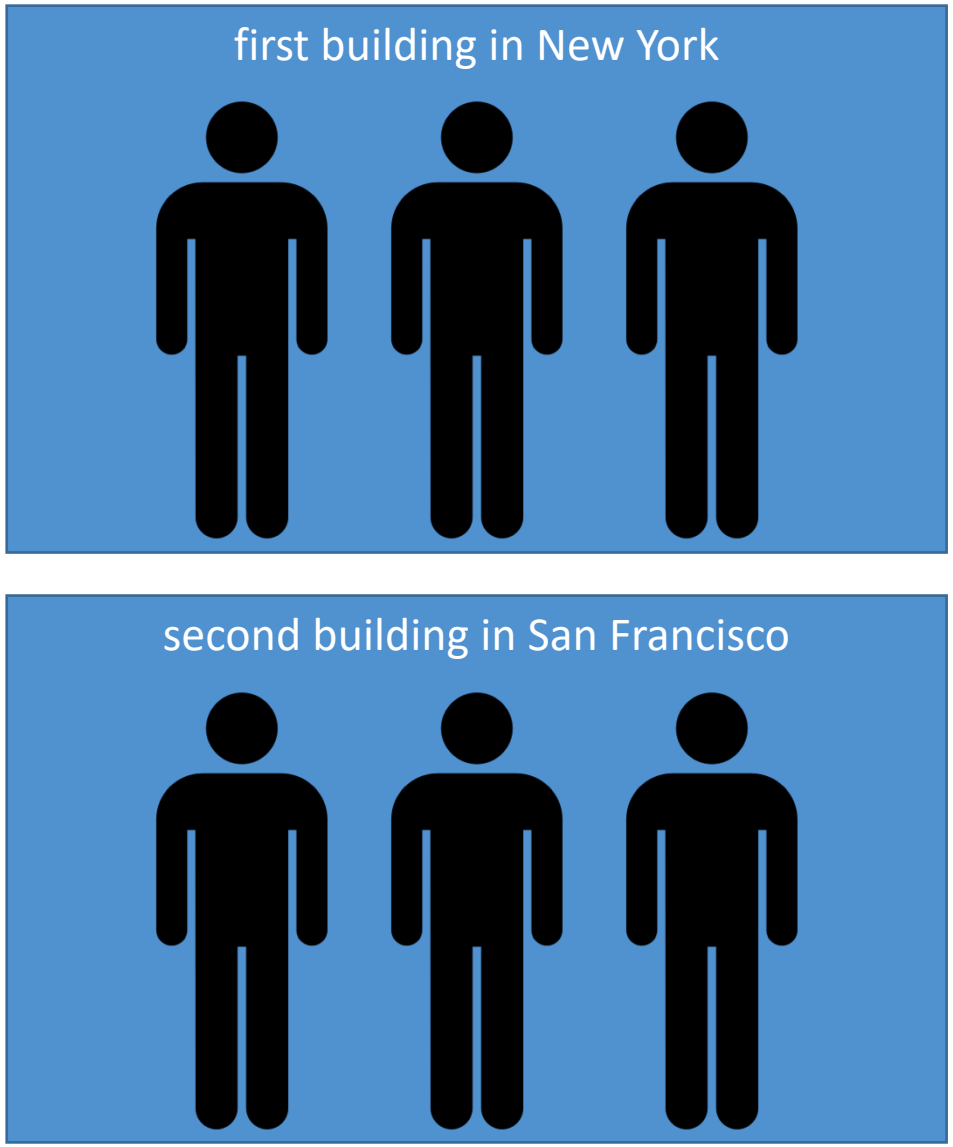
- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications

- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2



# High Availability

- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 Availability Zones
- The goal of high availability is to survive a data center loss (disaster)



# High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up / down)
  - From: t2.nano - 0.5G of RAM, 1 vCPU
  - To: u-12tb1.metal – 12.3 TB of RAM, 448 vCPUs

- Horizontal Scaling: Increase number of instances (= scale out / in)
  - Auto Scaling Group
  - Load Balancer

- High Availability: Run instances for the same application across multi AZ
  - Auto Scaling Group multi AZ
  - Load Balancer multi AZ

# Scalability vs Elasticity (vs Agility)

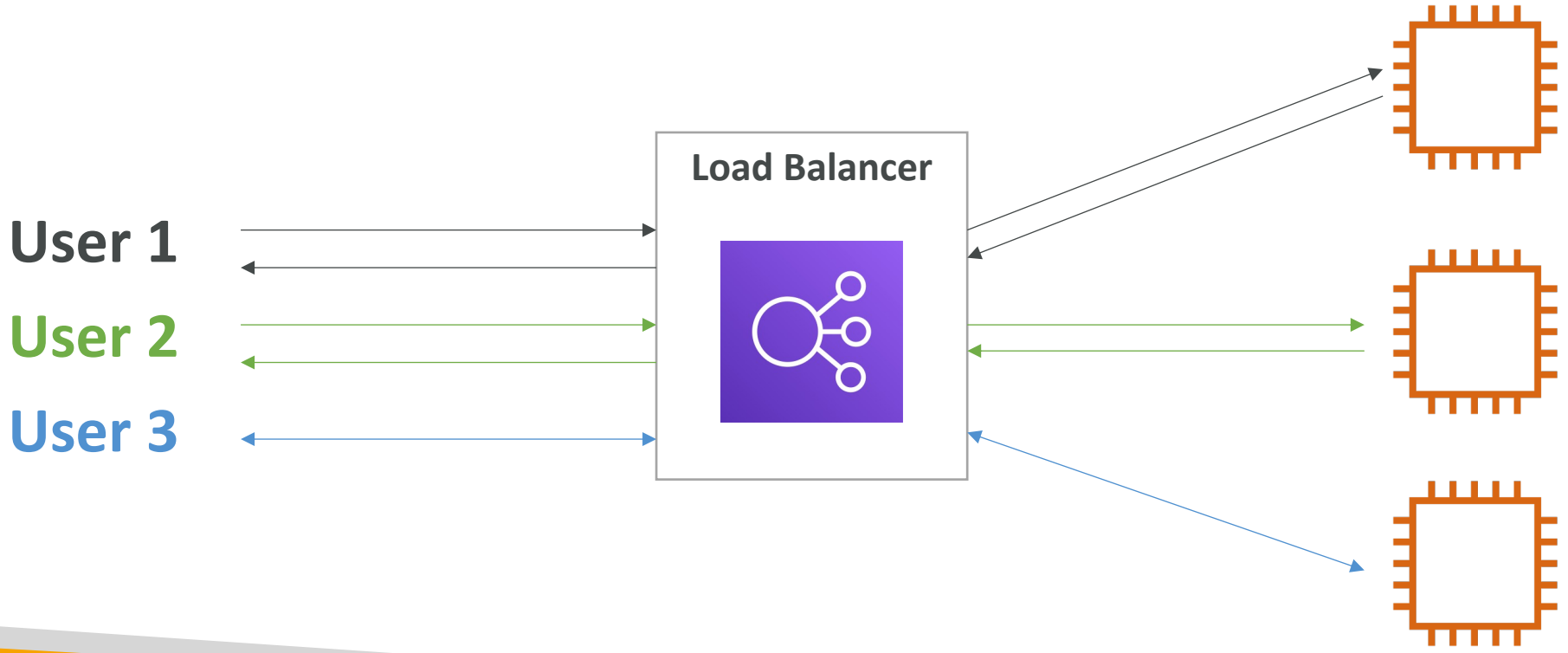
- **Scalability:** ability to accommodate a larger load by making the hardware stronger (scale up), or by adding nodes (scale out)

- **Elasticity:** once a system is scalable, elasticity means that there will be some "auto-scaling" so that the system can scale based on the load. This is "cloud-friendly": pay-per-use, match demand, optimize costs

- **Agility:** (not related to scalability - distractor) 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.

# What is load balancing?

- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



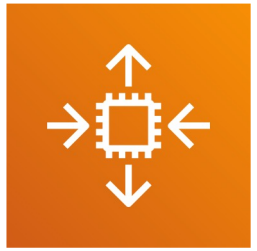
# Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- High availability across zones

# Why use an Elastic Load Balancer?

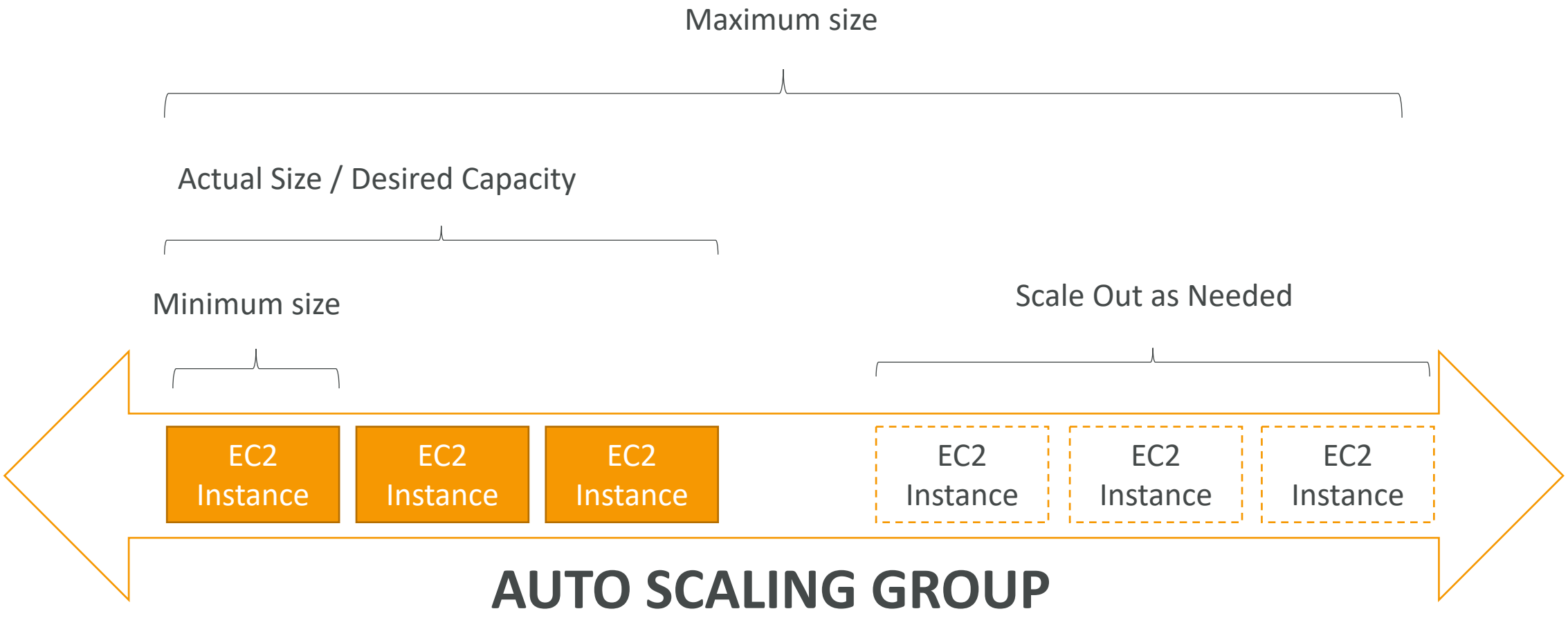
- An ELB (Elastic Load Balancer) is a managed load balancer
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end (maintenance, integrations)
- 3 kinds of load balancers offered by AWS:
  - Application Load Balancer (HTTP / HTTPS only) – Layer 7
  - Network Load Balancer (ultra-high performance, allows for TCP) – Layer 4
  - Classic Load Balancer (slowly retiring) – Layer 4 & 7

# What's an Auto Scaling Group?

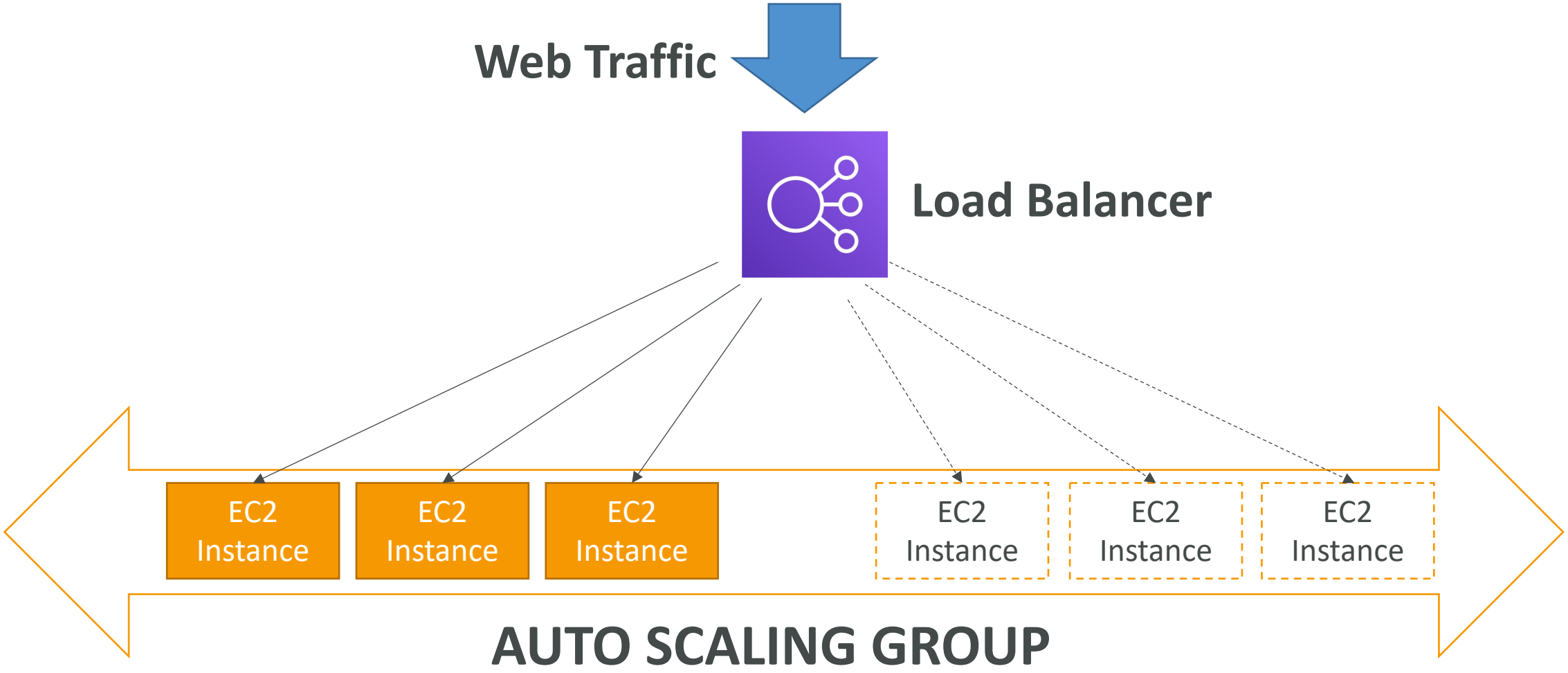


- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically register new instances to a load balancer
  - Replace unhealthy instances
- Cost Savings: only run at an optimal capacity (principle of the cloud)

# Auto Scaling Group in AWS



# Auto Scaling Group in AWS With Load Balancer



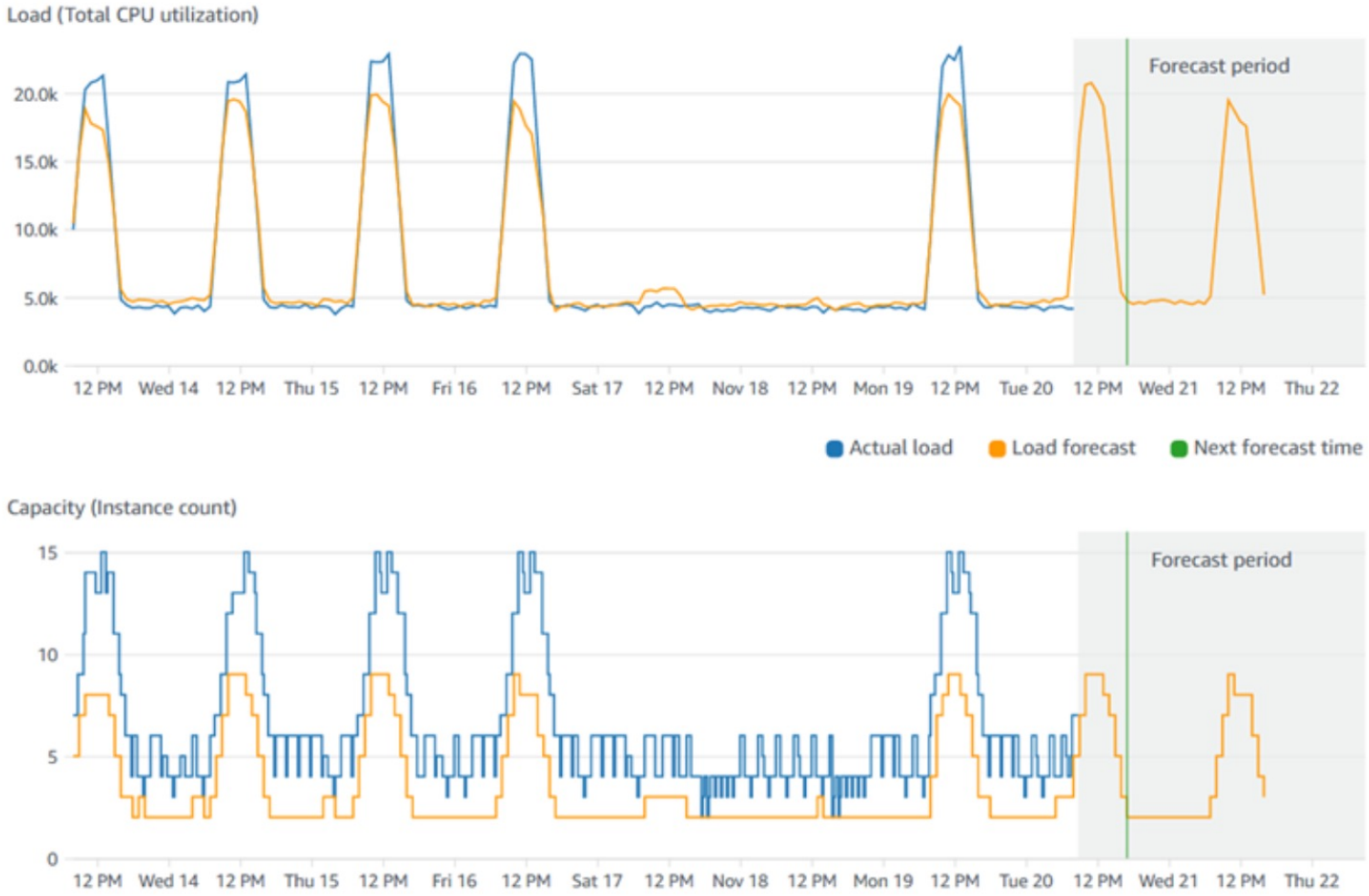
# Auto Scaling Groups – Scaling Strategies

- **Manual Scaling:** Update the size of an ASG manually

- **Dynamic Scaling:** Respond to changing demand
  - **Simple / Step Scaling**
    - When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
    - When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1
  - **Target Tracking Scaling**
    - Example: I want the average ASG CPU to stay at around 40%
  - **Scheduled Scaling**
    - Anticipate a scaling based on known usage patterns
    - Example: increase the min. capacity to 10 at 5 pm on Fridays

# Auto Scaling Groups – Scaling Strategies

- **Predictive Scaling**
  - Uses Machine Learning to predict future traffic ahead of time
  - Automatically provisions the right number of EC2 instances in advance
- Useful when your load has predictable time-based patterns



# ELB & ASG – Summary

- **High Availability** vs **Scalability** (vertical and horizontal) vs **Elasticity** vs **Agility** in the Cloud
- **Elastic Load Balancers (ELB)**
  - Distribute traffic across backend EC2 instances, can be Multi-AZ
  - Supports health checks
  - 3 types: Application LB (HTTP – L7), Network LB (TCP – L4), Classic LB (old)
- **Auto Scaling Groups (ASG)**
  - Implement Elasticity for your application, across multiple AZ
  - Scale EC2 instances based on the demand on your system, replace unhealthy
  - Integrated with the ELB

# Amazon S3 Section

# Section introduction

- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage

- Many websites use Amazon S3 as a backbone
- Many AWS services use Amazon S3 as an integration as well

- We'll have a step-by-step approach to S3

# Amazon S3 Use cases

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Software delivery
- Static website


Nasdaq stores 7 years of data into S3 Glacier


Sysco runs analytics on its data and gain business insights

# Amazon S3 - Buckets

- Amazon S3 allows people to store objects (files) in "buckets" (directories)
- Buckets must have a **globally unique name (across all regions all accounts)**
- Buckets are defined at the region level
- S3 looks like a global service but buckets are created in a region
- Naming convention
  - No uppercase, No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number
  - Must NOT start with the prefix **xn--**
  - Must NOT end with the suffix **-s3alias**

S3 Bucket

# Amazon S3 - Objects

- Objects (files) have a Key
- The key is the FULL path:
  - s3://my-bucket/my_file.txt
  - s3://my-bucket/my_folder1/another_folder/my_file.txt
- The key is composed of prefix + object name
  - s3://my-bucket/my_folder1/another_folder/my_file.txt
- There's no concept of "directories" within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("/")

Object

S3 Bucket with Objects

# Amazon S3 – Objects (cont.)

- Object values are the content of the body:
  - Max. Object Size is 5TB (5000GB)
  - If uploading more than 5GB, must use "multi-part upload"

- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

# Amazon S3 – Security

- User-Based
  - **IAM Policies** – which API calls should be allowed for a specific user from IAM
- Resource-Based
  - **Bucket Policies** – bucket wide rules from the S3 console - allows cross account
  - **Object Access Control List (ACL)** – finer grain (can be disabled)
  - **Bucket Access Control List (ACL)** – less common (can be disabled)
- **Note:** an IAM principal can access an S3 object if
  - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
  - AND there's no explicit DENY
- **Encryption:** encrypt objects in Amazon S3 using encryption keys

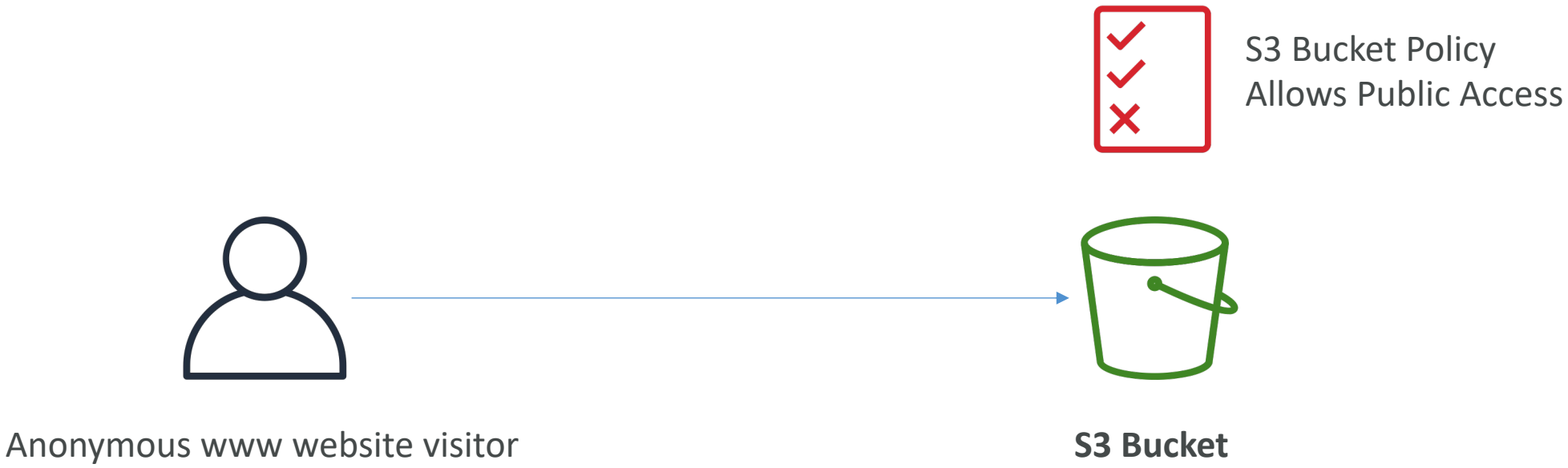
# S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Effect: Allow / Deny
  - Actions: Set of API to Allow or Deny
  - Principal: The account or user to apply the policy to

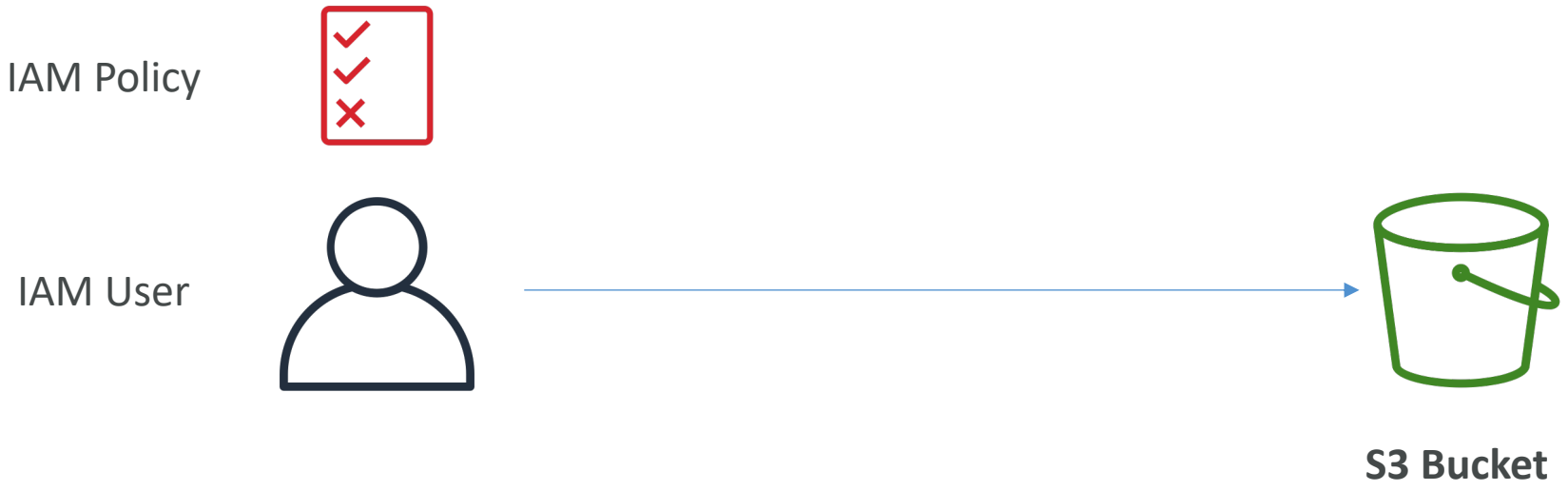
- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ]
        }
    ]
}
```

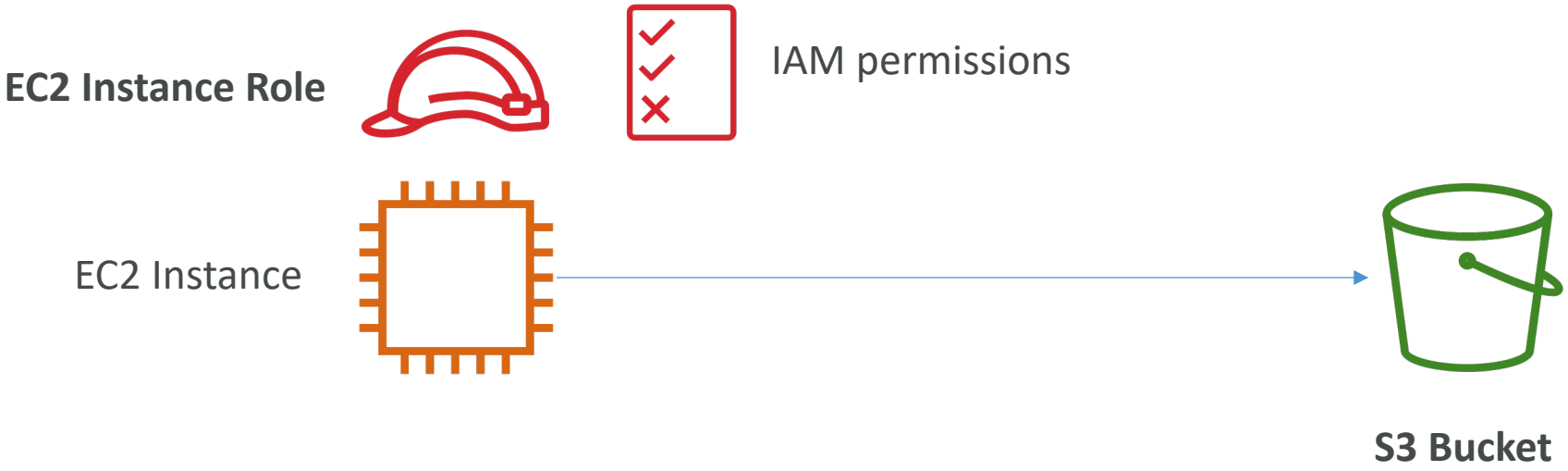
# Example: Public Access - Use Bucket Policy



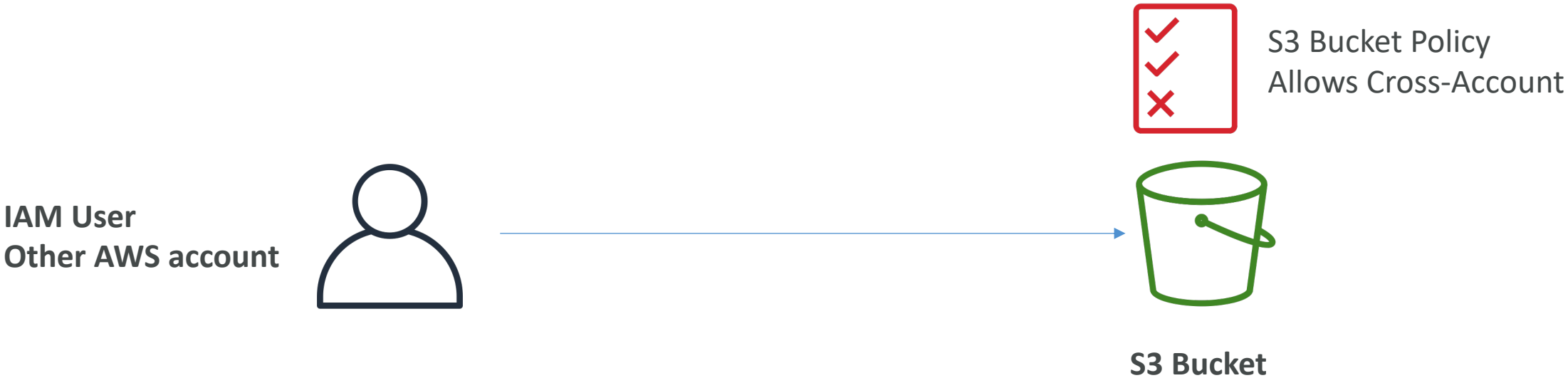
# Example: User Access to S3 – IAM permissions



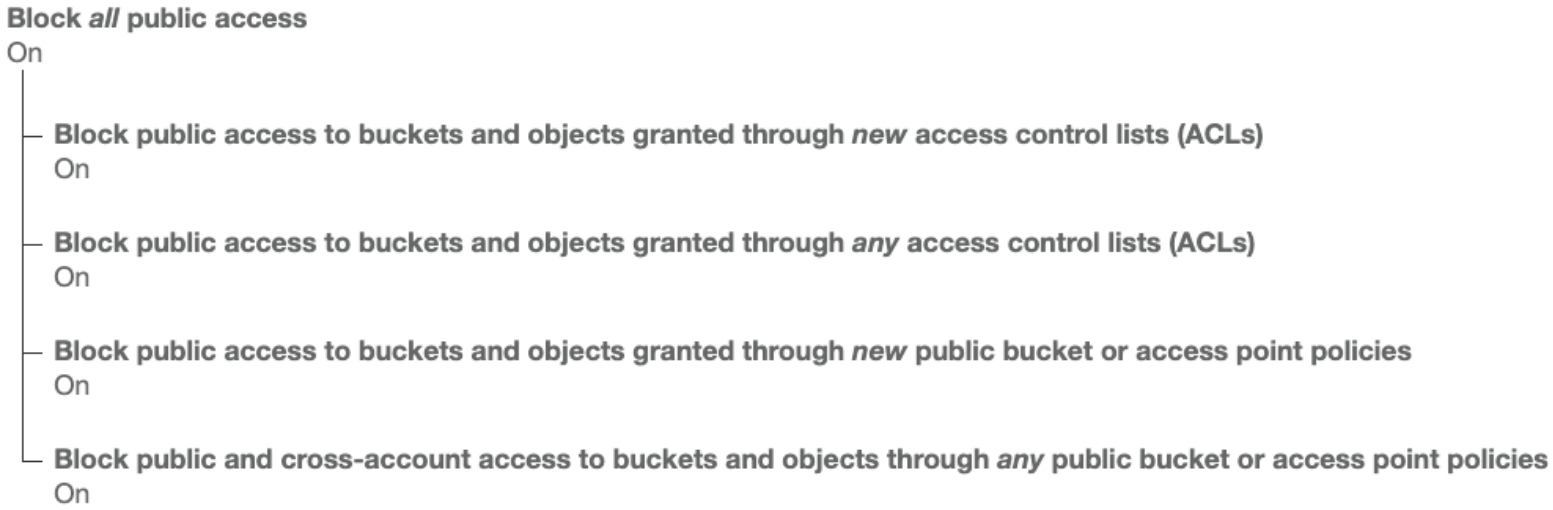
# Example: EC2 instance access - Use IAM Roles



# Advanced: Cross-Account Access – Use Bucket Policy



# Bucket settings for Block Public Access




- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on
- Can be set at the account level

# Amazon S3 – Static Website Hosting

- S3 can host static websites and have them accessible on the Internet

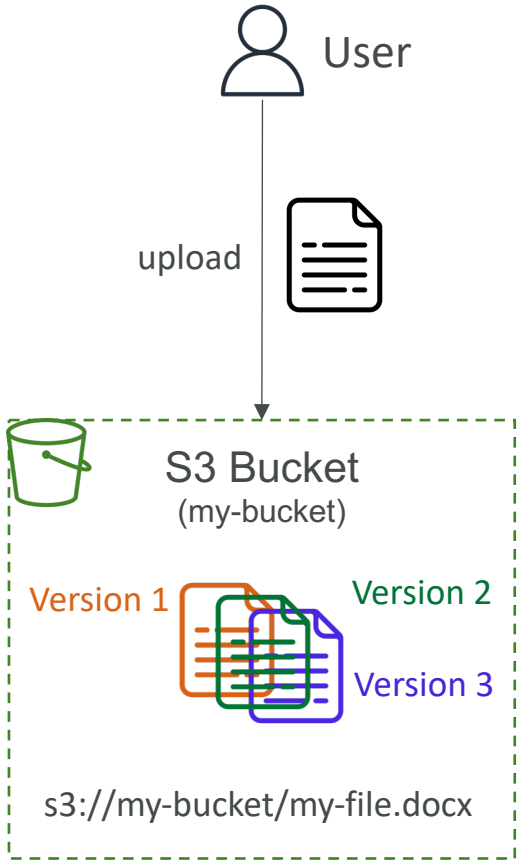
- The website URL will be (depending on the region)
  - http://*bucket-name*.s3-website-*aws-region*.amazonaws.com

  OR
  - http://*bucket-name*.s3-website.*aws-region*.amazonaws.com

- If you get a **403 Forbidden** error, make sure the bucket policy allows public reads!

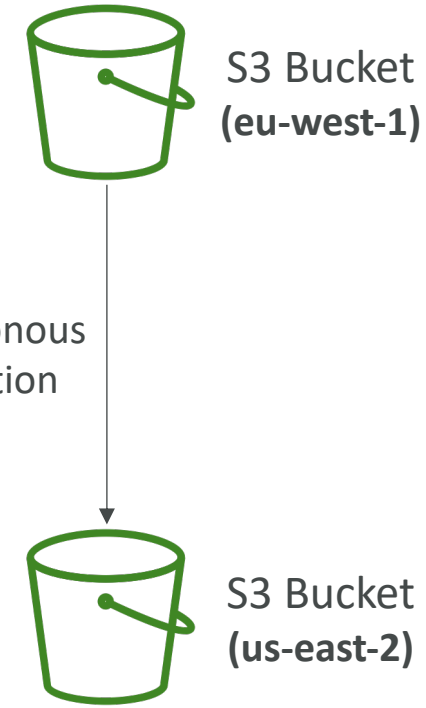
# Amazon S3 - Versioning

- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will change the "version": 1, 2, 3….
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - Any file that is not versioned prior to enabling versioning will have version "null"
  - Suspending versioning does not delete the previous versions



# Amazon S3 – Replication (CRR & SRR)

- **Must enable Versioning** in source and destination buckets
- **Cross-Region Replication (CRR)**
- **Same-Region Replication (SRR)**
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3




- Use cases:
  - **CRR** – compliance, lower latency access, replication across accounts
  - **SRR** – log aggregation, live replication between production and test accounts

# S3 Storage Classes

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Glacier Instant Retrieval
- Amazon S3 Glacier Flexible Retrieval
- Amazon S3 Glacier Deep Archive
- Amazon S3 Intelligent Tiering

- Can move between classes manually or using S3 Lifecycle configurations

# S3 Durability and Availability

- Durability:
  - High durability (99.999999999%, 11 9's) of objects across multiple AZ
  - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
  - Same for all storage classes

- Availability:
  - Measures how readily available a service is
  - Varies depending on storage class
  - Example: S3 standard has 99.99% availability = not available 53 minutes a year

# S3 Standard – General Purpose

- 99.99% Availability
- Used for frequently accessed data
- Low latency and high throughput
- Sustain 2 concurrent facility failures

- Use Cases: Big Data analytics, mobile & gaming applications, content distribution…

# S3 Storage Classes – Infrequent Access

- For data that is less frequently accessed, but requires rapid access when needed
- Lower cost than S3 Standard

- Amazon S3 Standard-Infrequent Access (S3 Standard-IA)
  - 99.9% Availability
  - Use cases: Disaster Recovery, backups

- Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)
  - High durability (99.999999999%) in a single AZ; data lost when AZ is destroyed
  - 99.5% Availability
  - Use Cases: Storing secondary backup copies of on-premise data, or data you can recreate

# Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost

- Amazon S3 Glacier Instant Retrieval
  - Millisecond retrieval, great for data accessed once a quarter
  - Minimum storage duration of 90 days
- Amazon S3 Glacier Flexible Retrieval (formerly Amazon S3 Glacier):
  - Expedited (1 to 5 minutes), Standard (3 to 5 hours), Bulk (5 to 12 hours) – free
  - Minimum storage duration of 90 days
- Amazon S3 Glacier Deep Archive – for long term storage:
  - Standard (12 hours), Bulk (48 hours)
  - Minimum storage duration of 180 days

# S3 Intelligent-Tiering

- Small monthly monitoring and auto-tiering fee
- Moves objects automatically between Access Tiers based on usage
- There are no retrieval charges in S3 Intelligent-Tiering

- *Frequent Access tier (automatic):* default tier
- *Infrequent Access tier (automatic)*: objects not accessed for 30 days
- *Archive Instant Access tier (automatic):* objects not accessed for 90 days
- *Archive Access tier (optional):* configurable from 90 days to 700+ days
- *Deep Archive Access tier (optional):* config. from 180 days to 700+ days

# S3 Storage Classes Comparison

| | Standard | Intelligent-Tiering | Standard-IA | One Zone-IA | Glacier Instant Retrieval | Glacier Flexible Retrieval | Glacier Deep Archive |
|---|---|---|---|---|---|---|---|
| Durability | 99.999999999% == (11 9's) | | | | | | |
| Availability | 99.99% | 99.9% | 99.9% | 99.5% | 99.9% | 99.99% | 99.99% |
| Availability SLA | 99.9% | 99% | 99% | 99% | 99% | 99.9% | 99.9% |
| Availability Zones | >= 3 | >= 3 | >= 3 | 1 | >= 3 | >= 3 | >= 3 |
| Min. Storage Duration Charge | None | None | 30 Days | 30 Days | 90 Days | 90 Days | 180 Days |
| Min. Billable Object Size | None | None | 128 KB | 128 KB | 128 KB | 40 KB | 40 KB |
| Retrieval Fee | None | None | Per GB retrieved | Per GB retrieved | Per GB retrieved | Per GB retrieved | Per GB retrieved |

https://aws.amazon.com/s3/storage-classes/

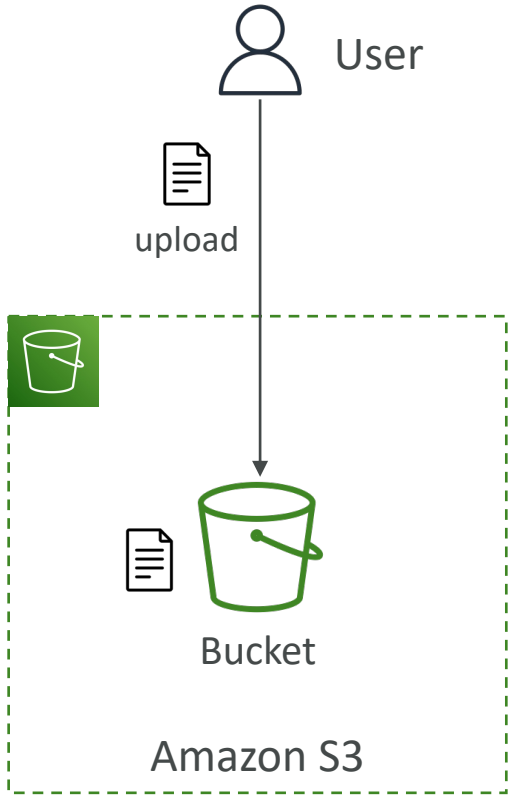
# S3 Storage Classes – Price Comparison Example: us-east- l

| | Standard | Intelligent-Tiering | Standard-IA | One Zone-IA | Glacier Instant Retrieval | Glacier Flexible Retrieval | Glacier Deep Archive |
|---|---|---|---|---|---|---|---|
| Storage Cost (per GB per month) | $0.023 | $0.0025 - $0.023 | %0.0125 | $0.01 | $0.004 | $0.0036 | $0.00099 |
| Retrieval Cost (per 1000 request) | **GET:** $0.0004<br>**POST:** $0.005 | **GET:** $0.0004<br>**POST:** $0.005 | **GET:** $0.001<br>**POST:** $0.01 | **GET:** $0.001<br>**POST:** $0.01 | **GET:** $0.01<br>**POST:** $0.02 | **GET:** $0.0004<br>**POST:** $0.03<br><br>**Expedited:** $10<br>**Standard:** $0.05<br>**Bulk:** free | **GET:** $0.0004<br>**POST:** $0.05<br><br>**Standard:** $0.10<br>**Bulk:** $0.025 |
| Retrieval Time | Instantaneous | | | | | **Expedited** (1 – 5 mins)<br>**Standard** (3 – 5 hours)<br>Bulk (5 – 12 hours) | **Standard** (12 hours)<br>**Bulk** (48 hours) |
| Monitoring Cost (pet 1000 objects) | | $0.0025 | | | | | |

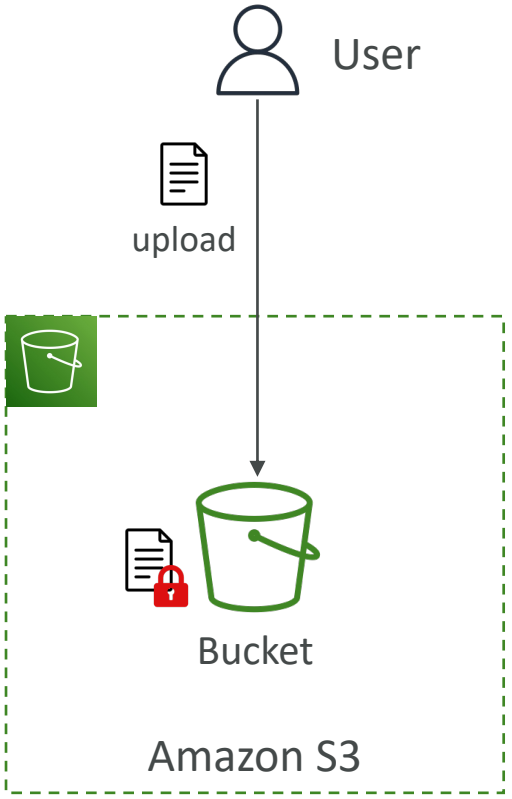
https://aws.amazon.com/s3/pricing/

# S3 Encryption

## No Encryption

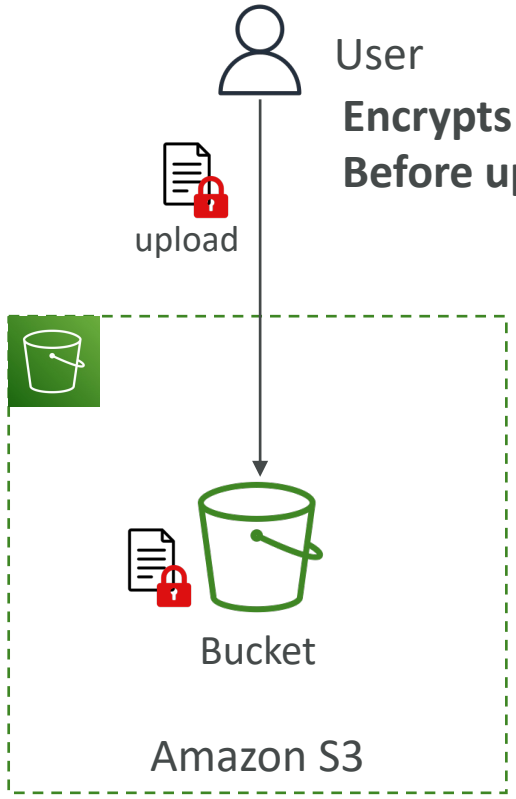



## Server-Side Encryption




**Server encrypts the file after receiving it**

## Client-Side Encryption




**Encrypts the file**
**Before uploading it**

# Shared Responsibility Model for S3


- Infrastructure (global security, durability, availability, sustain concurrent loss of data in two facilities)
- Configuration and vulnerability analysis
- Compliance validation

- S3 Versioning
- S3 Bucket Policies
- S3 Replication Setup
- Logging and Monitoring
- S3 Storage Classes
- Data encryption at rest and in transit

# AWS Snow Family

- Highly-secure, portable devices to **collect and process data at the edge**, and **migrate data into and out of AWS**

- Data migration:


Snowcone


Snowball Edge


Snowmobile

- Edge computing:


Snowcone


Snowball Edge

# Data Migrations with AWS Snow Family

| | Time to Transfer | | |
|---|---|---|---|
| | **100 Mbps** | **1Gbps** | **10Gbps** |
| **10 TB** | 12 days | 30 hours | 3 hours |
| **100 TB** | 124 days | 12 days | 30 hours |
| **1 PB** | 3 years | 124 days | 12 days |

Challenges:

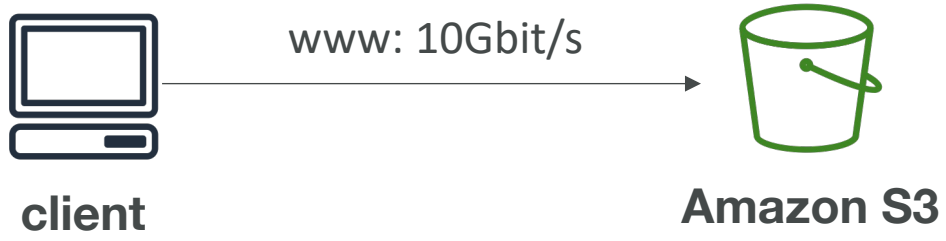
- Limited connectivity
- Limited bandwidth
- High network cost
- Shared bandwidth (can't maximize the line)
- Connection stability

AWS Snow Family: offline devices to perform data migrations

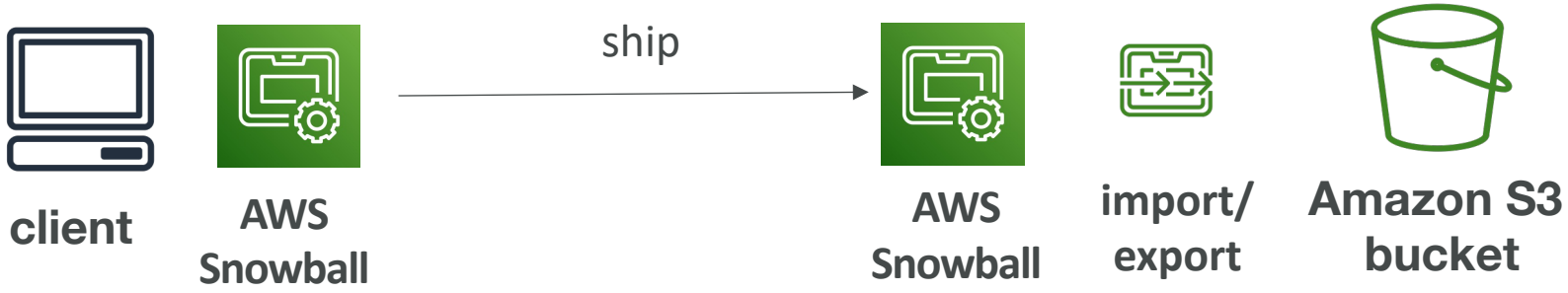
If it takes more than a week to transfer over the network, use Snowball devices!

# Diagrams

- Direct upload to S3:




- With Snow Family:



# Snowball Edge (for data transfers)

- Physical data transport solution: move TBs or PBs of data in or out of AWS
- Alternative to moving data over the network (and paying network fees)
- Pay per data transfer job
- Provide block storage and Amazon S3-compatible object storage
- Snowball Edge Storage Optimized
  - 80 TB of HDD capacity for block volume and S3 compatible object storage
- Snowball Edge Compute Optimized
  - 42 TB of HDD capacity for block volume and S3 compatible object storage
- Use cases: large data cloud migrations, DC decommission, disaster recovery

# AWS Snowcone

- **Small, portable computing, anywhere, rugged & secure, withstands harsh environments**
- Light (4.5 pounds, 2.1 kg)
- Device used for edge computing, storage, and data transfer
- **8 TBs of usable storage**
- Use Snowcone where Snowball does not fit (space-constrained environment)
- Must provide your own battery / cables

- Can be sent back to AWS offline, or connect it to internet and use **AWS DataSync** to send data

# AWS Snowmobile


- Transfer exabytes of data (1 EB = 1,000 PB = 1,000,000 TBs)
- Each Snowmobile has 100 PB of capacity (use multiple in parallel)
- High security: temperature controlled, GPS, 24/7 video surveillance
- **Better than Snowball if you transfer more than 10 PB**

# AWS Snow Family for Data Migrations

**Snowcone**

**Snowball Edge**

**Snowmobile**

| | Snowcone | Snowball Edge Storage Optimized | Snowmobile |
|---|---|---|---|
| Storage Capacity | 8 TB usable | 80 TB usable | < 100 PB |
| Migration Size | Up to 24 TB, online and offline | Up to petabytes, offline | Up to exabytes, offline |
| DataSync agent | Pre-installed | | |
| Storage Clustering | | Up to 15 nodes | |

# Snow Family – Usage Process

1. Request Snowball devices from the AWS console for delivery
2. Install the snowball client / AWS OpsHub on your servers
3. Connect the snowball to your servers and copy files using the client
4. Ship back the device when you're done (goes to the right AWS facility)
5. Data will be loaded into an S3 bucket
6. Snowball is completely wiped

# What is Edge Computing?

- Process data while it's being created on **an edge location**
  - A truck on the road, a ship on the sea, a mining station underground...



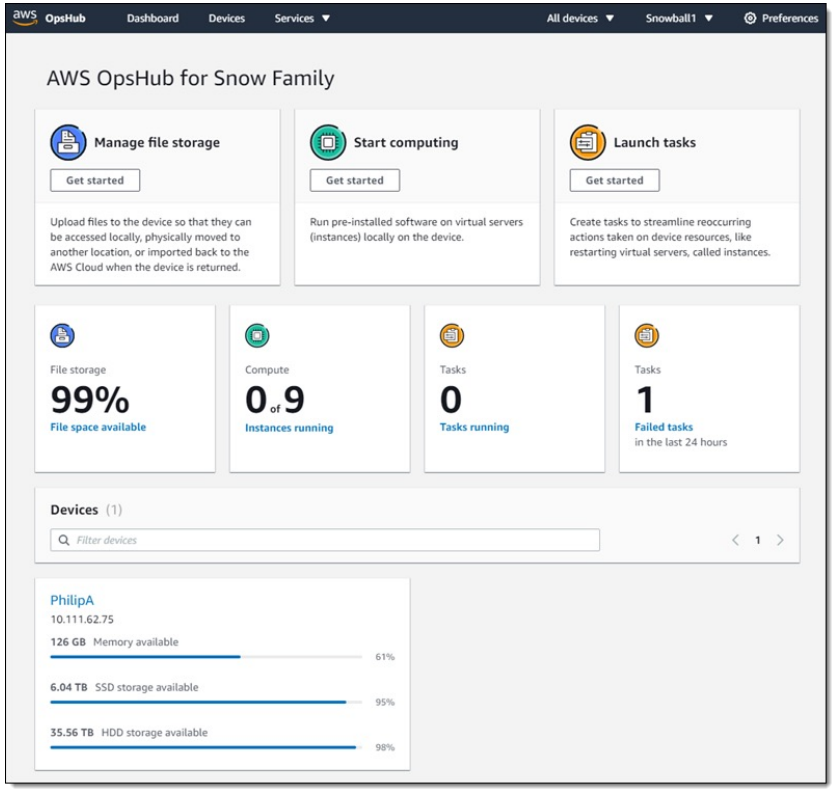
- These locations may have
  - Limited / no internet access
  - Limited / no easy access to computing power
- We setup a **Snowball Edge / Snowcone** device to do edge computing
- Use cases of Edge Computing:
  - Preprocess data
  - Machine learning at the edge
  - Transcoding media streams
- Eventually (if need be) we can ship back the device to AWS (for transferring data for example)

# Snow Family – Edge Computing

- Snowcone (smaller)
  - 2 CPUs, 4 GB of memory, wired or wireless access
  - USB-C power using a cord or the optional battery
- Snowball Edge – Compute Optimized
  - 52 vCPUs, 208 GiB of RAM
  - Optional GPU (useful for video processing or machine learning)
  - 42 TB usable storage
- Snowball Edge – Storage Optimized
  - Up to 40 vCPUs, 80 GiB of RAM
  - Object storage clustering available
- All: Can run EC2 Instances & AWS Lambda functions (using AWS IoT Greengrass)
- Long-term deployment options: 1 and 3 years discounted pricing

# AWS OpsHub

- Historically, to use Snow Family devices, you needed a CLI (Command Line Interface tool)
- Today, you can use **AWS OpsHub** (a software you install on your computer / laptop) to manage your Snow Family Device
  - Unlocking and configuring single or clustered devices
  - Transferring files
  - Launching and managing instances running on Snow Family Devices
  - Monitor device metrics (storage capacity, active instances on your device)
  - Launch compatible AWS services on your devices (ex: Amazon EC2 instances, AWS DataSync, Network File System (NFS))




https://aws.amazon.com/blogs/aws/aws-snowball-edge-update/

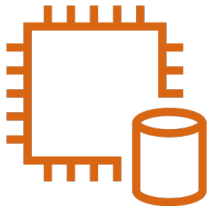
# Hybrid Cloud for Storage

- AWS is pushing for "hybrid cloud"
  - Part of your infrastructure is on-premises
  - Part of your infrastructure is on the cloud
- This can be due to
  - Long cloud migrations
  - Security requirements
  - Compliance requirements
  - IT strategy
- S3 is a proprietary storage technology (unlike EFS / NFS), so how do you expose the S3 data on-premise?
- AWS Storage Gateway!

# AWS Storage Cloud Native Options

**BLOCK**

Amazon EBS



EC2 Instance Store

**FILE**

Amazon EFS

**OBJECT**

Amazon S3

Glacier

# AWS Storage Gateway

- Bridge between on-premise data and cloud data in S3
- Hybrid storage service to allow on-premises to seamlessly use the AWS Cloud
- Use cases: disaster recovery, backup & restore, tiered storage

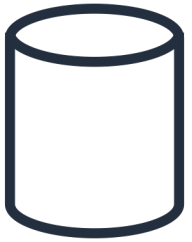
- Types of Storage Gateway:
  - File Gateway
  - Volume Gateway
  - Tape Gateway
- No need to know the types at the exam

# Amazon S3 – Summary

- **Buckets vs Objects:** global unique name, tied to a region
- **S3 security:** IAM policy, S3 Bucket Policy (public access), S3 Encryption
- **S3 Websites:** host a static website on Amazon S3
- **S3 Versioning:** multiple versions for files, prevent accidental deletes
- **S3 Replication:** same-region or cross-region, must enable versioning
- **S3 Storage Classes:** Standard, IA, 1Z-IA, Intelligent, Glacier (Instant, Flexible, Deep)
- **Snow Family:** import data onto S3 through a physical device, edge computing
- **OpsHub**: desktop application to manage Snow Family devices
- **Storage Gateway:** hybrid solution to extend on-premises storage to S3

# Databases Section

# Databases Intro

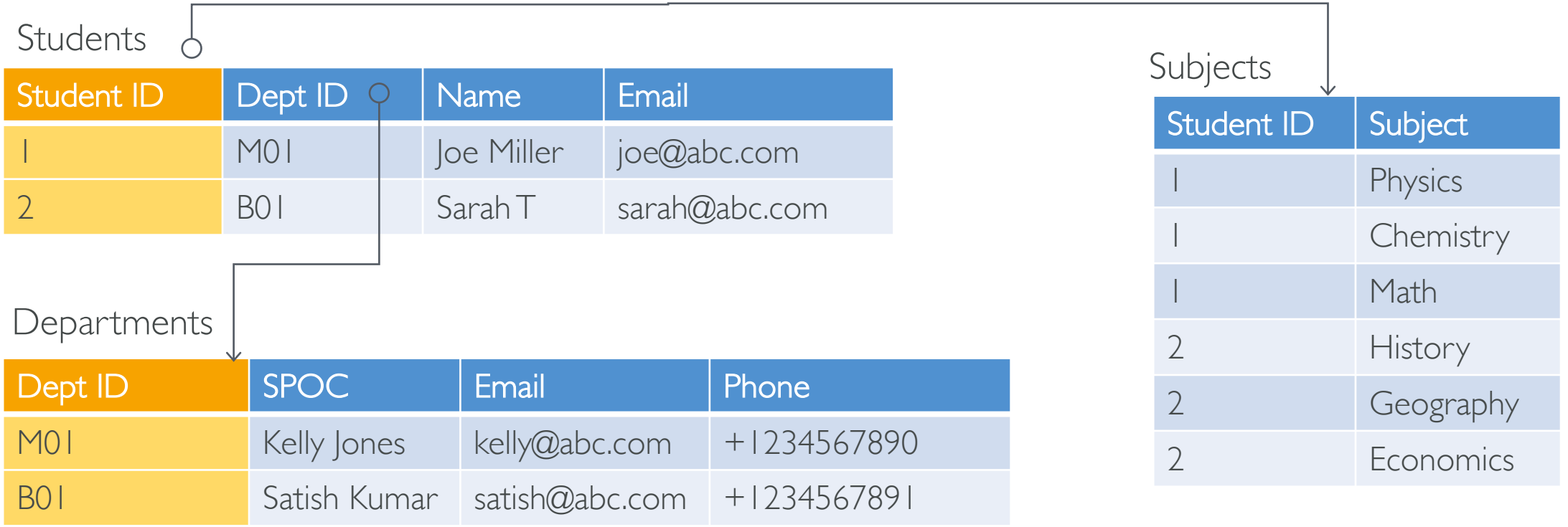


- Storing data on disk (EFS, EBS, EC2 Instance Store, S3) can have its limits
- Sometimes, you want to store data in a database…
- You can **structure** the data
- You build **indexes** to efficiently **query** / **search** through the data
- You define **relationships** between your **datasets**

- Databases are **optimized for a purpose** and come with different features, shapes and constraints

# Relational Databases

- Looks just like Excel spreadsheets, with links between them!
- Can use the SQL language to perform queries / lookups



Students

| Student ID | Dept ID | Name | Email |
|---|---|---|---|
| 1 | M01 | Joe Miller | joe@abc.com |
| 2 | B01 | Sarah T | sarah@abc.com |

Departments

| Dept ID | SPOC | Email | Phone |
|---|---|---|---|
| M01 | Kelly Jones | kelly@abc.com | +1234567890 |
| B01 | Satish Kumar | satish@abc.com | +1234567891 |

Subjects

| Student ID | Subject |
|---|---|
| 1 | Physics |
| 1 | Chemistry |
| 1 | Math |
| 2 | History |
| 2 | Geography |
| 2 | Economics |

# NoSQL Databases

- NoSQL = non-SQL = non relational databases
- NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications.
- Benefits:
  - Flexibility: easy to evolve data model
  - Scalability: designed to scale-out by using distributed clusters
  - High-performance: optimized for a specific data model
  - Highly functional: types optimized for the data model
- Examples: Key-value, document, graph, in-memory, search databases

# NoSQL data example: JSON

- JSON = JavaScript Object Notation
- JSON is a common form of data that fits into a NoSQL model
- Data can be **nested**
- Fields can **change** over time
- Support for new types: **arrays**, etc…

```
{
    "name": "John",
    "age": 30,
    "cars": [
        "Ford",
        "BMW",
        "Fiat"
    ],
    "address": {
        "type": "house",
        "number": 23,
        "street": "Dream Road"
    }
}
```

# Databases & Shared Responsibility on AWS

- AWS offers use to **manage** different databases
- **Benefits** include:
  - Quick Provisioning, High Availability, Vertical and Horizontal Scaling
  - Automated Backup & Restore, Operations, Upgrades
  - Operating System Patching is handled by AWS
  - Monitoring, alerting

- Note: many databases technologies could be run on EC2, but you must handle yourself the resiliency, backup, patching, high availability, fault tolerance, scaling…

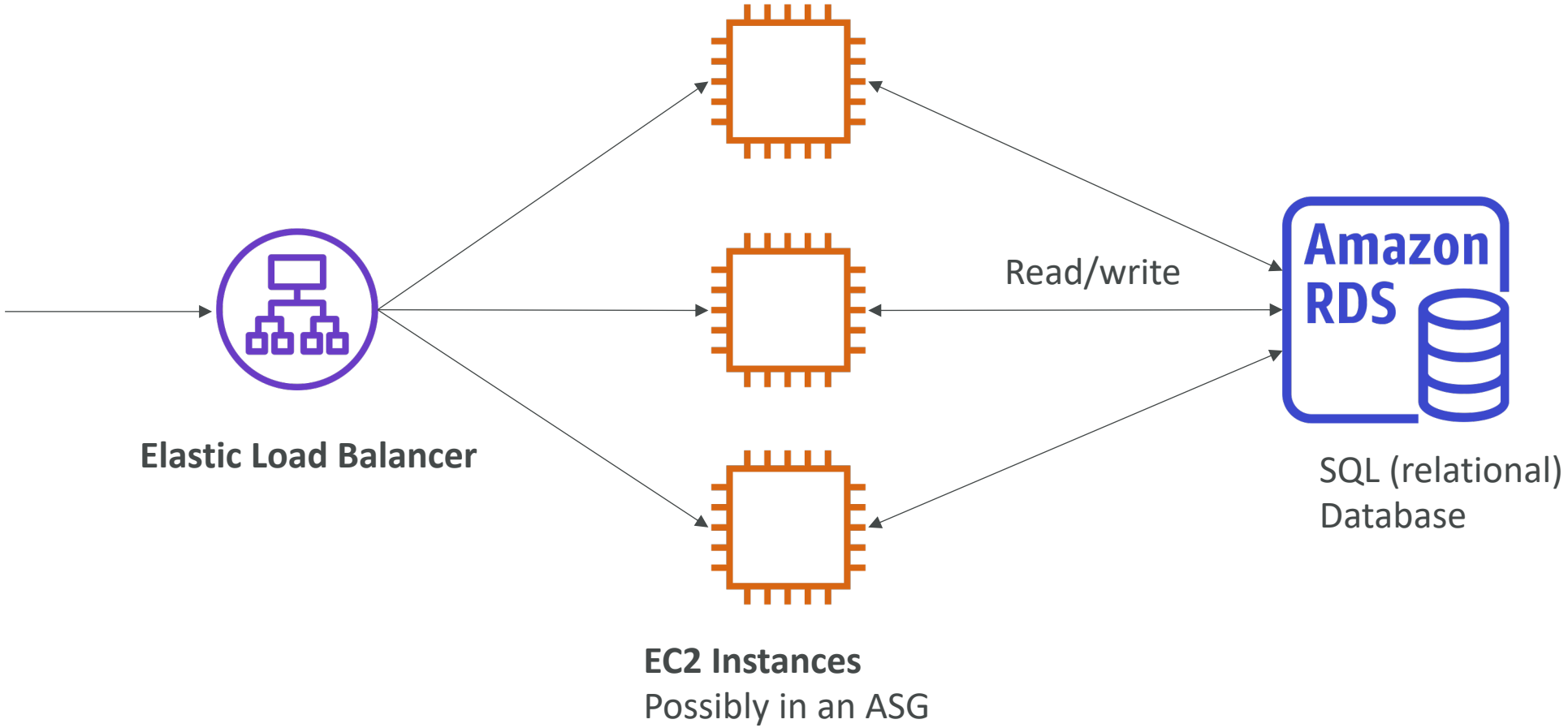
# AWS RDS Overview

- RDS stands for **Relational** Database Service
- It's a managed DB service for DB use **SQL** as a query language.
- It allows you to create databases in the cloud that are managed by AWS
  - Postgres
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
  - Aurora (AWS Proprietary database)

# Advantage over using RDS versus deploying DB on EC2

- RDS is a managed service:
  - Automated provisioning, OS patching
  - Continuous backups and restore to specific timestamp (Point in Time Restore)!
  - Monitoring dashboards
  - Read replicas for improved read performance
  - Multi AZ setup for DR (Disaster Recovery)
  - Maintenance windows for upgrades
  - Scaling capability (vertical and horizontal)
  - Storage backed by EBS (gp2 or io1)
- BUT you can't SSH into your instances

# RDS Solution Architecture

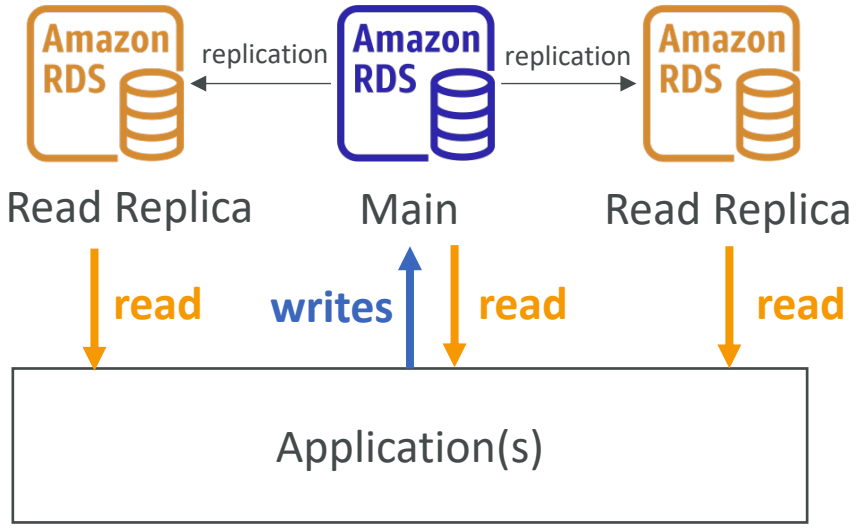


# Amazon Aurora

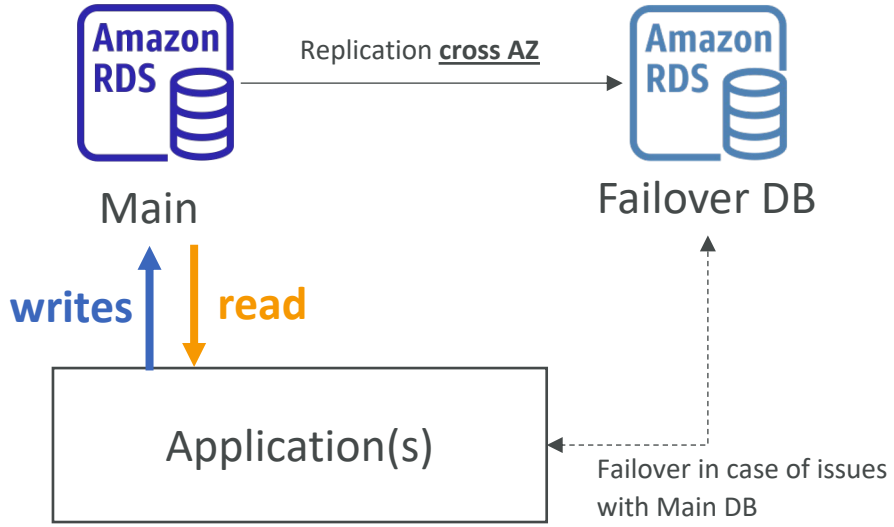
- Aurora is a proprietary technology from AWS (not open sourced)
- **PostgreSQL and MySQL** are both supported as Aurora DB
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64 TB.
- Aurora costs more than RDS (20% more) – but is more efficient
- Not in the free tier

# RDS Deployments: Read Replicas, Multi-AZ

- Read Replicas:
  - **Scale** the read workload of your DB
  - Can create up to 5 Read Replicas
  - Data is only written to the main DB

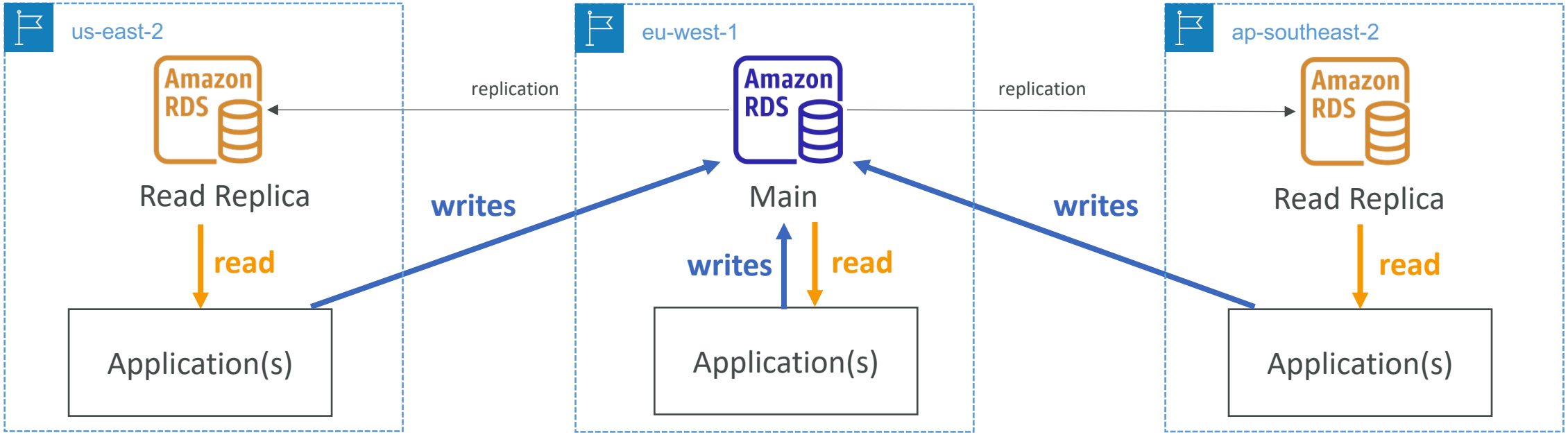



- Multi-AZ:
  - **Failover** in case of AZ outage (high availability)
  - Data is only read/written to the main database
  - Can only have 1 other AZ as failover



# RDS Deployments: Multi-Region

- Multi-Region (Read Replicas)
  - **Disaster recovery** in case of region issue
  - **Local performance** for global reads
  - Replication cost

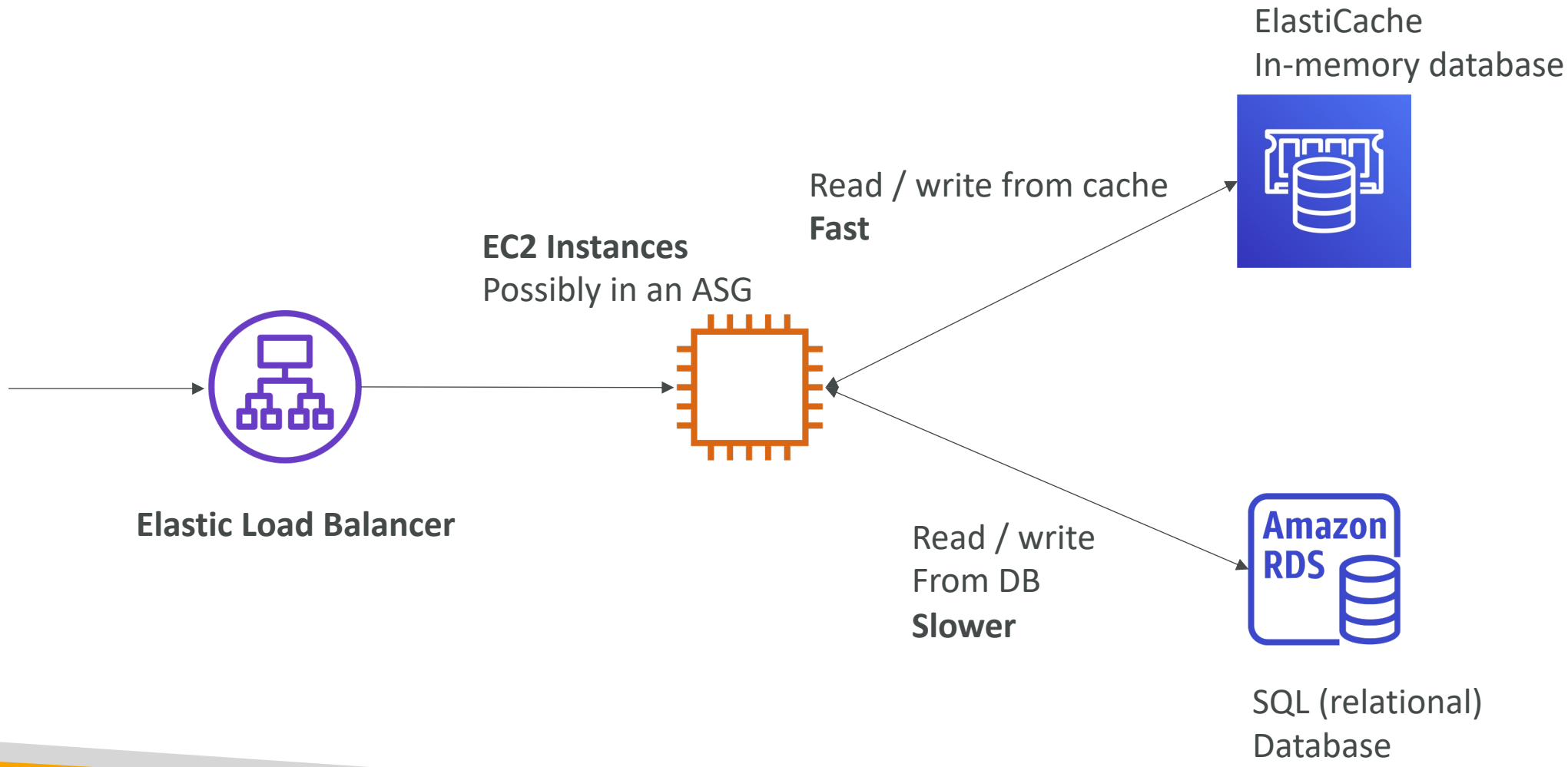


# Amazon ElastiCache Overview

- The same way RDS is to get managed Relational Databases…
- ElastiCache is to get managed Redis or Memcached
- Caches are **in-memory databases** with high performance, low latency
- Helps **reduce load off databases for read intensive workloads**

- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups

# ElastiCache
# Solution Architecture - Cache

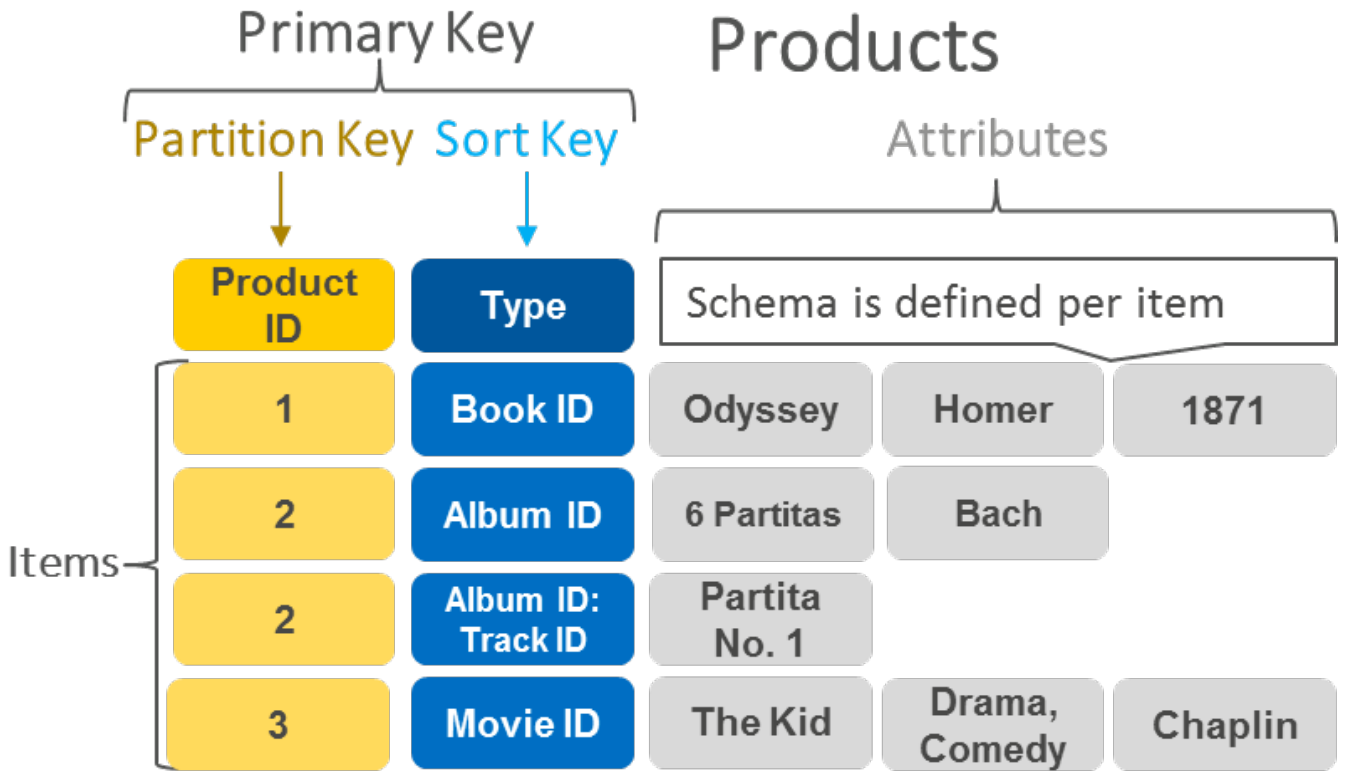


# DynamoDB

- Fully Managed Highly available with replication across 3 AZ
- **NoSQL database - not a relational database**
- Scales to massive workloads, distributed "**serverless**" database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance
- **Single-digit millisecond latency – low latency retrieval**
- Integrated with IAM for security, authorization and administration
- Low cost and auto scaling capabilities
- Standard & Infrequent Access (IA) Table Class

# DynamoDB – type of data

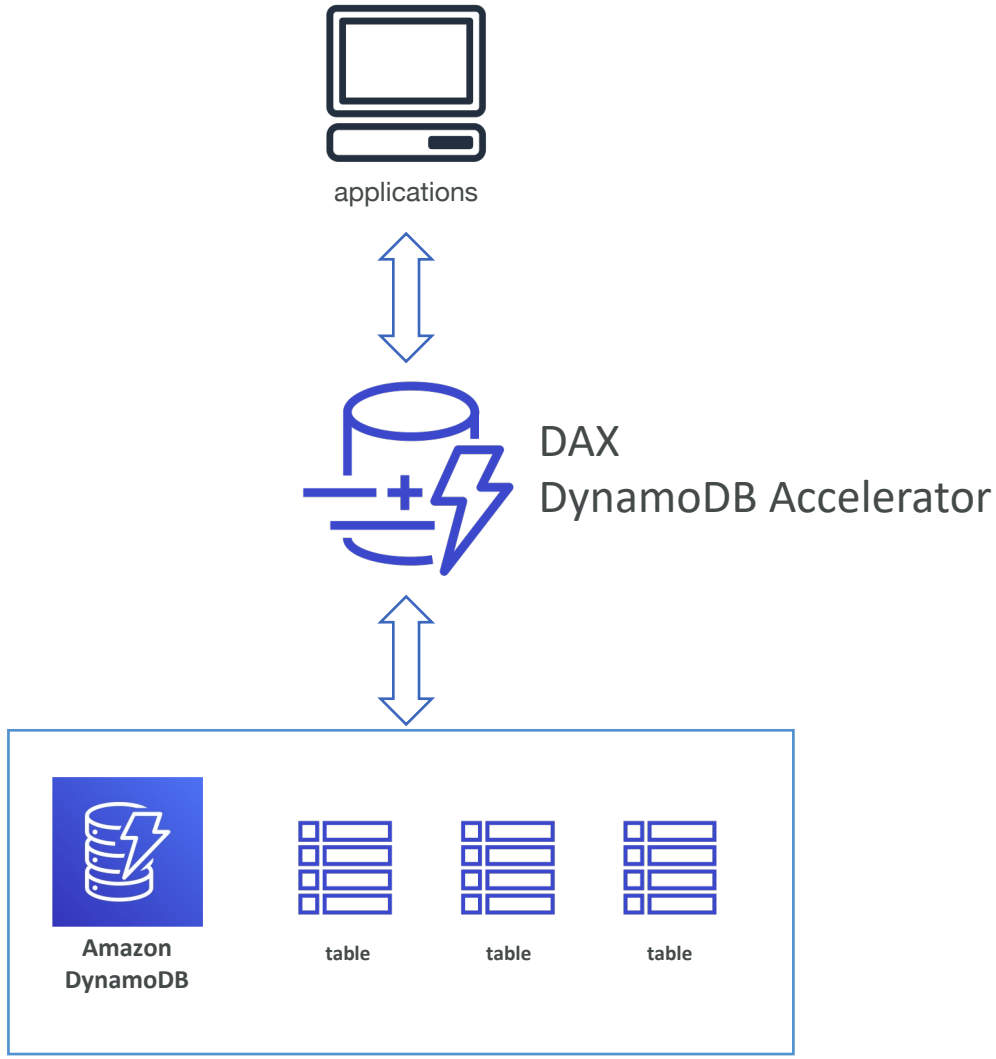
- DynamoDB is a key/value database




https://aws.amazon.com/nosql/key-value/

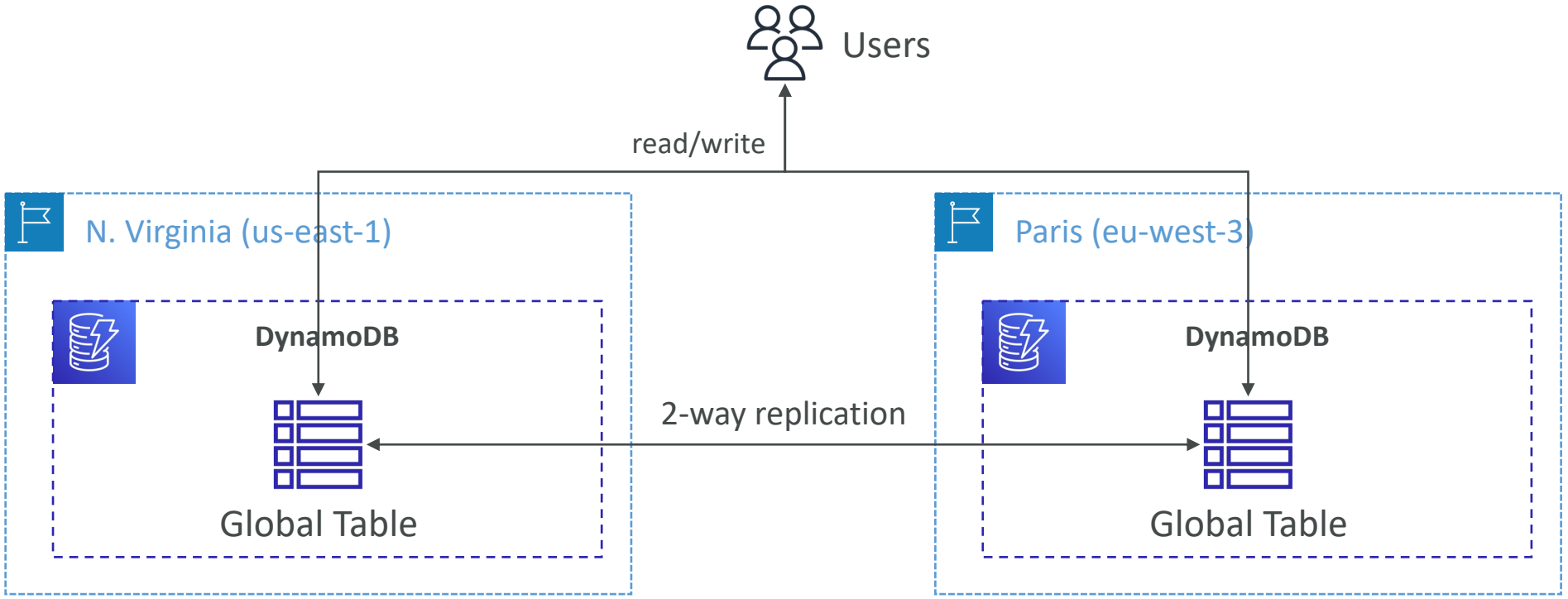
# DynamoDB Accelerator - DAX

- Fully Managed **in-memory cache** for DynamoDB
- **10x performance improvement** – single-digit millisecond latency to microseconds latency – when accessing your DynamoDB tables
- Secure, highly scalable & highly available
- Difference with ElastiCache at the CCP level: **DAX is only used for and is integrated with DynamoDB**, while ElastiCache can be used for other databases



# DynamoDB – Global Tables

- Make a DynamoDB table accessible with **low latency** in multiple-regions
- **Active-Active** replication (**read/write** to any AWS Region)



# Redshift Overview

- Redshift is based on PostgreSQL, but **it's not used for OLTP**
- **It's OLAP – online analytical processing (analytics and data warehousing)**
- Load data once every hour, not every second
- 10x better performance than other data warehouses, scale to PBs of data
- **Columnar** storage of data (instead of row based)
- Massively Parallel Query Execution (MPP), highly available
- Pay as you go based on the instances provisioned
- Has a SQL interface for performing the queries
- BI tools such as AWS Quicksight or Tableau integrate with it

# Amazon EMR

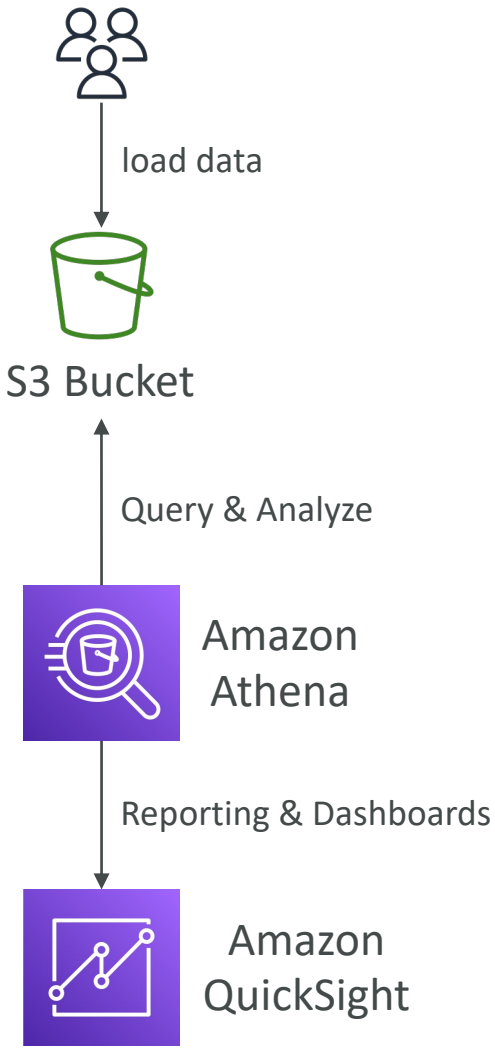
- EMR stands for "Elastic MapReduce"
- EMR helps creating **Hadoop clusters (Big Data)** to analyze and process vast amount of data
- The clusters can be made of **hundreds of EC2 instances**
- Also supports Apache Spark, HBase, Presto, Flink…
- EMR takes care of all the provisioning and configuration
- Auto-scaling and integrated with Spot instances
- **Use cases: data processing, machine learning, web indexing, big data…**

# Amazon Athena

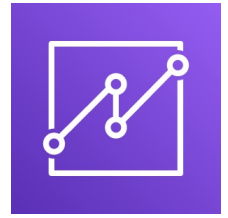
- **Serverless** query service to analyze data stored in Amazon S3
- Uses standard SQL language to query the files
- Supports CSV, JSON, ORC, Avro, and Parquet (built on Presto)
- Pricing: $5.00 per TB of data scanned
- Use compressed or columnar data for cost-savings (less scan)

- Use cases: Business intelligence / analytics / reporting, analyze & query VPC Flow Logs, ELB Logs, CloudTrail trails, etc...

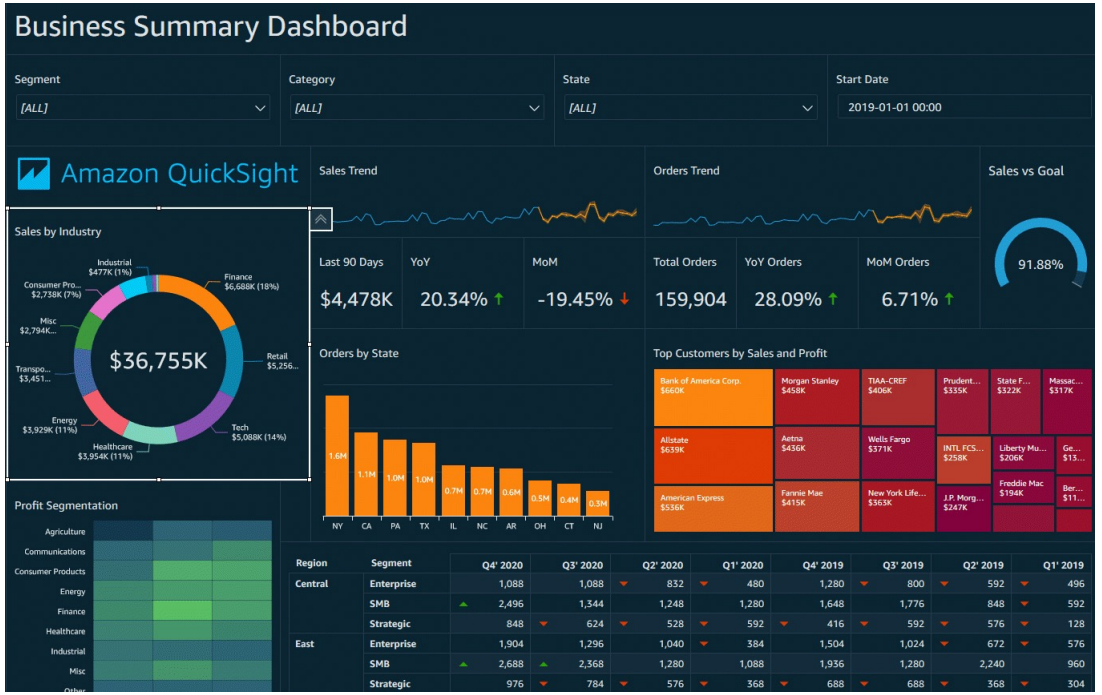
- <u>Exam Tip:</u> analyze data in S3 using serverless SQL, use Athena



# Amazon QuickSight



- Serverless machine learning-powered business intelligence service to create interactive dashboards
- Fast, automatically scalable, embeddable, with per-session pricing
- Use cases:
  - Business analytics
  - Building visualizations
  - Perform ad-hoc analysis
  - Get business insights using data
- Integrated with RDS, Aurora, Athena, Redshift, S3…




https://aws.amazon.com/quicksight/

# DocumentDB

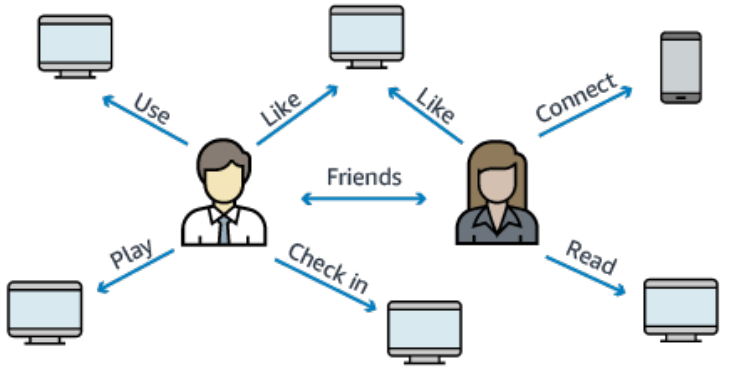

- Aurora is an "AWS-implementation" of PostgreSQL / MySQL …
- **DocumentDB is the same for MongoDB (which is a NoSQL database)**

- MongoDB is used to store, query, and index JSON data
- Similar "deployment concepts" as Aurora
- Fully Managed, highly available with replication across 3 AZ
- DocumentDB storage automatically grows in increments of 10GB, up to 64 TB.

- Automatically scales to workloads with millions of requests per seconds

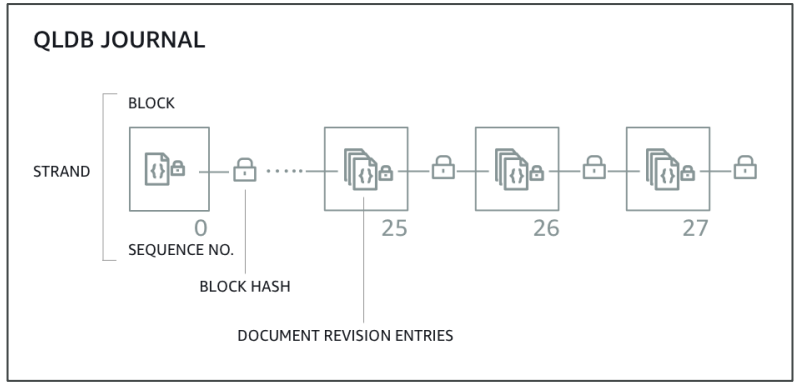
# Amazon Neptune

- Fully managed **graph** database
- A popular **graph dataset** would be a **social network**
  - Users have friends
  - Posts have comments
  - Comments have likes from users
  - Users share and like posts…
- Highly available across 3 AZ, with up to 15 read replicas
- Build and run applications working with highly connected datasets – optimized for these complex and hard queries
- Can store up to billions of relations and query the graph with milliseconds latency
- Highly available with replications across multiple AZs
- Great for knowledge graphs (Wikipedia), fraud detection, recommendation engines, social networking



# Amazon QLDB

- QLDB stands for "Quantum Ledger Database"
- A ledger is a book **recording financial transactions**
- Fully Managed, Serverless, High available, Replication across 3 AZ
- Used to **review history of all the changes made to your application data** over time
- **Immutable** system: no entry can be removed or modified, cryptographically verifiable




- 2-3x better performance than common ledger blockchain frameworks, manipulate data using SQL
- Difference with Amazon Managed Blockchain: **no decentralization component**, in accordance with financial regulation rules

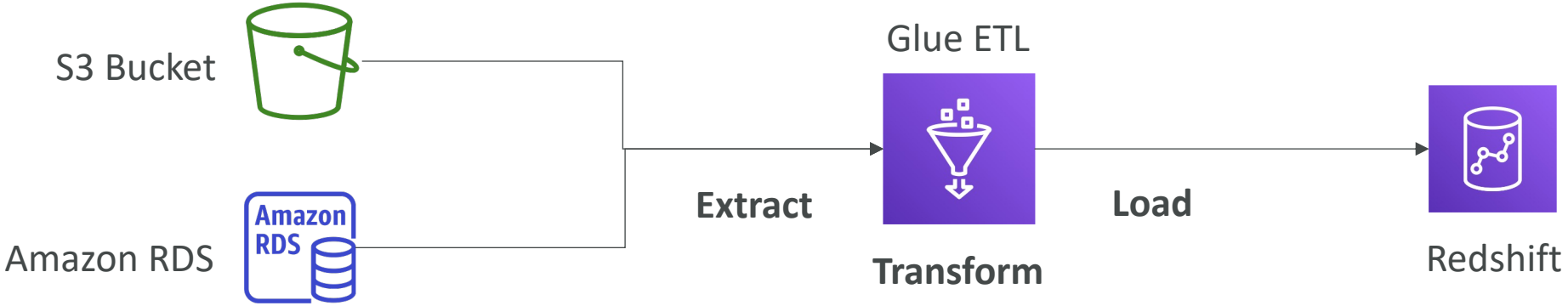
https://docs.aws.amazon.com/qldb/latest/developerguide/ledger-structure.html

# Amazon Managed Blockchain

- Blockchain makes it possible to build applications where multiple parties can execute transactions **without the need for a trusted, central authority.**
- Amazon Managed Blockchain is a managed service to:
  - Join public blockchain networks
  - Or create your own scalable private network
- Compatible with the frameworks Hyperledger Fabric & Ethereum

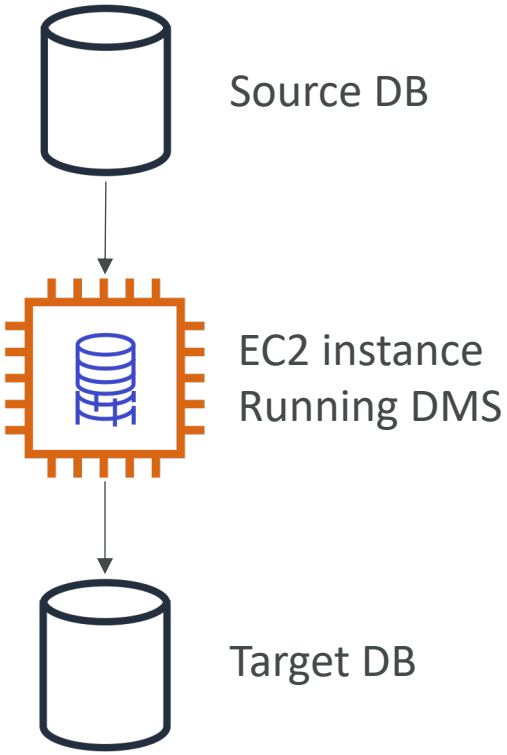

# AWS Glue

- Managed **extract, transform, and load (ETL)** service
- Useful to prepare and transform data for analytics
- Fully **serverless** service




- Glue Data Catalog: catalog of datasets
  - can be used by Athena, Redshift, EMR

# DMS – Database Migration Service




- Quickly and securely migrate databases to AWS, resilient, self healing
- The source database remains available during the migration

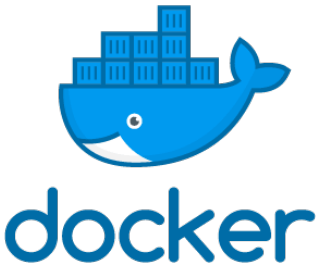
- Supports:
  - Homogeneous migrations: ex Oracle to Oracle
  - Heterogeneous migrations: ex Microsoft SQL Server to Aurora

# Databases & Analytics Summary in AWS

- **Relational Databases - OLTP:** RDS & Aurora (SQL)
- **Differences between Multi-AZ, Read Replicas, Multi-Region**
- **In-memory Database:** ElastiCache
- **Key/Value Database:** DynamoDB (serverless) & DAX (cache for DynamoDB)
- **Warehouse - OLAP:** Redshift (SQL)
- **Hadoop Cluster**: EMR
- **Athena:** query data on Amazon S3 (serverless & SQL)
- **QuickSight**: dashboards on your data (serverless)
- **DocumentDB**: "Aurora for MongoDB" (JSON – NoSQL database)
- **Amazon QLDB**: Financial Transactions Ledger (immutable journal, cryptographically verifiable)
- **Amazon Managed Blockchain:** managed Hyperledger Fabric & Ethereum blockchains
- **Glue:** Managed ETL (Extract Transform Load) and Data Catalog service
- **Database Migration:** DMS
- **Neptune:** graph database

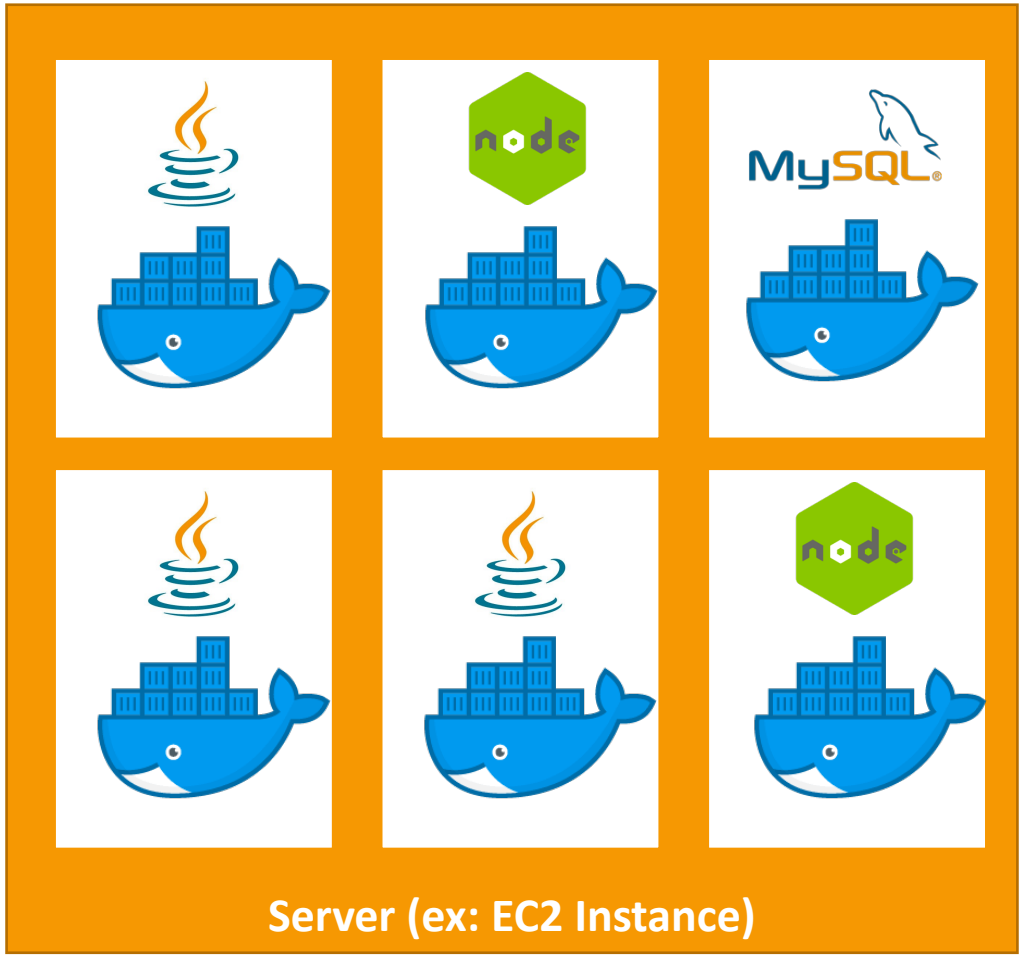
# Other Compute Section

# What is Docker?




- Docker is a software development platform to deploy apps
- Apps are packaged in **containers** that can be run on any OS
- **Apps run the same, regardless of where they're run**
  - Any machine
  - No compatibility issues
  - Predictable behavior
  - Less work
  - Easier to maintain and deploy
  - Works with any language, any OS, any technology
- Scale containers up and down very quickly (seconds)

# Docker on an OS



# Where Docker images are stored?

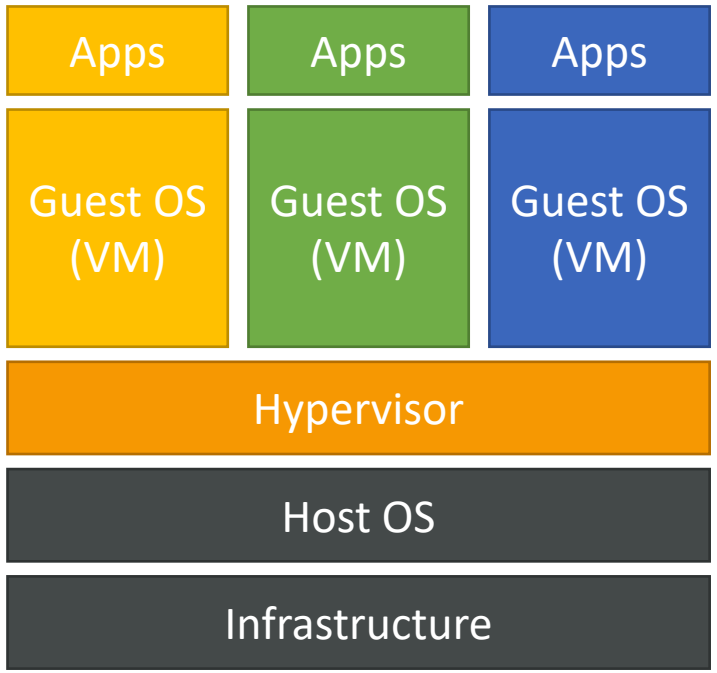
- Docker images are stored in Docker Repositories

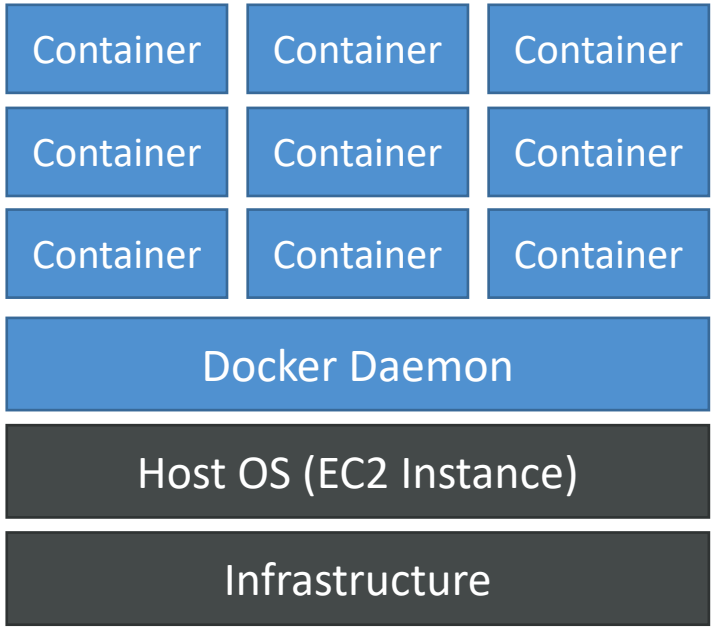
- Public: Docker Hub https://hub.docker.com/
  - Find base images for many technologies or OS:
  - Ubuntu
  - MySQL
  - NodeJS, Java…

- Private: Amazon ECR (Elastic Container Registry)

# Docker versus Virtual Machines

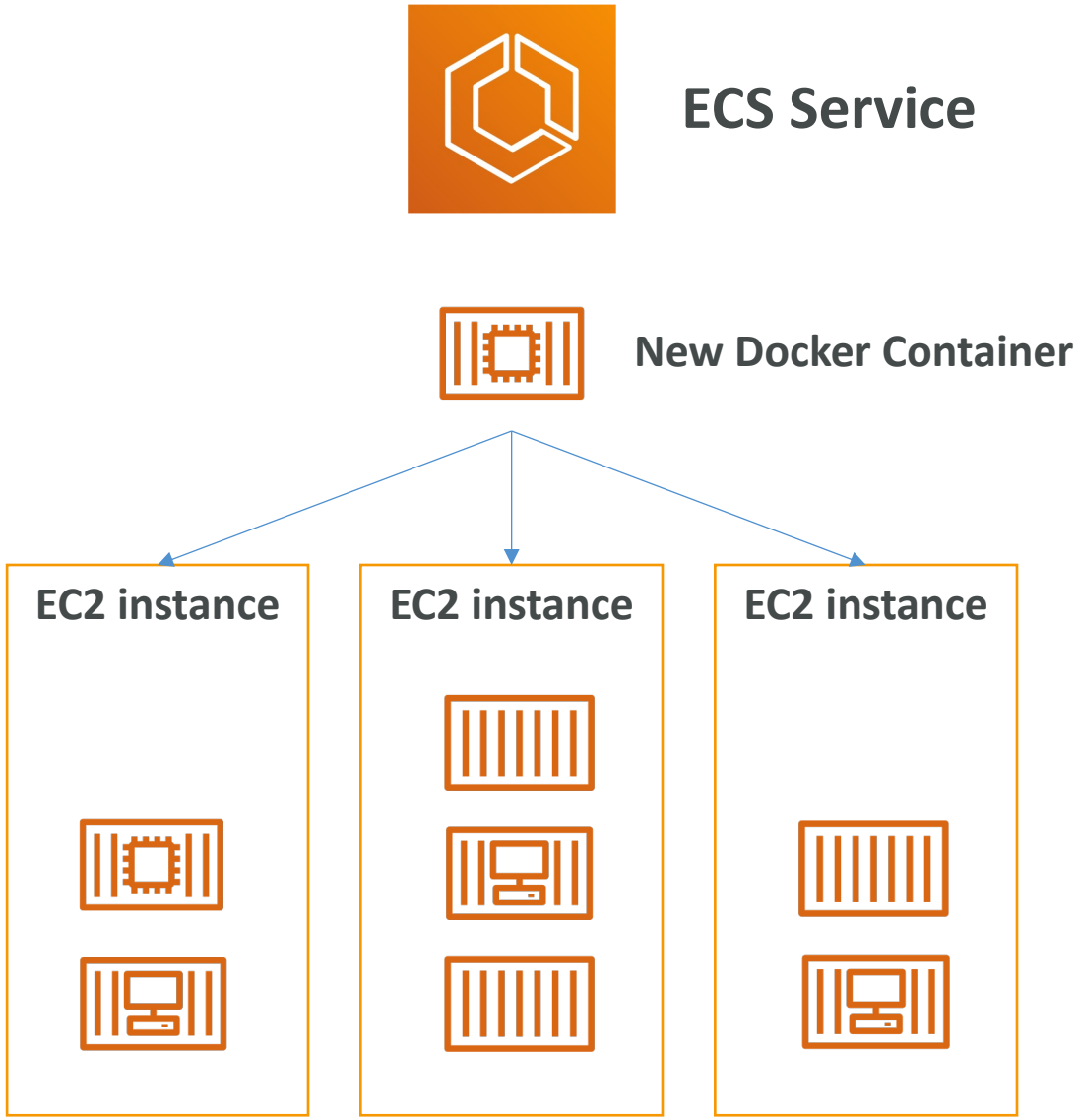
- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server



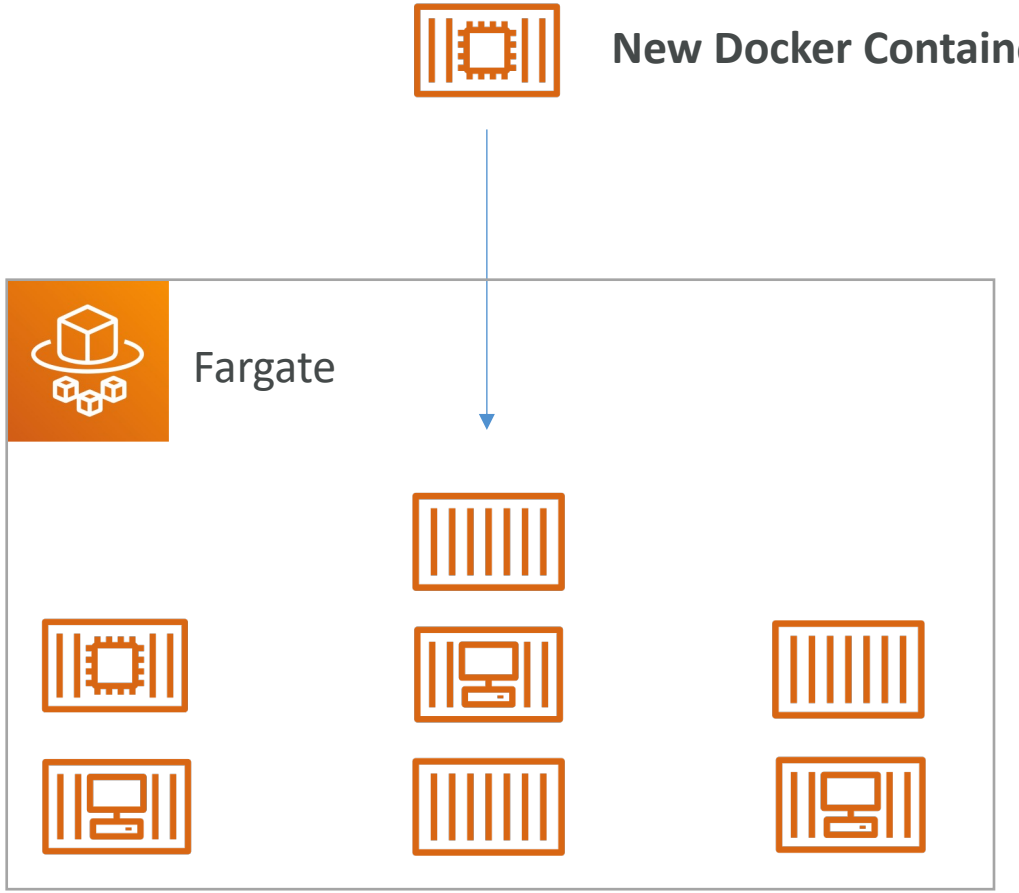
# ECS

- ECS = Elastic Container Service
- Launch Docker containers on AWS
- You must provision & maintain the infrastructure (the EC2 instances)
- AWS takes care of starting / stopping containers
- Has integrations with the Application Load Balancer



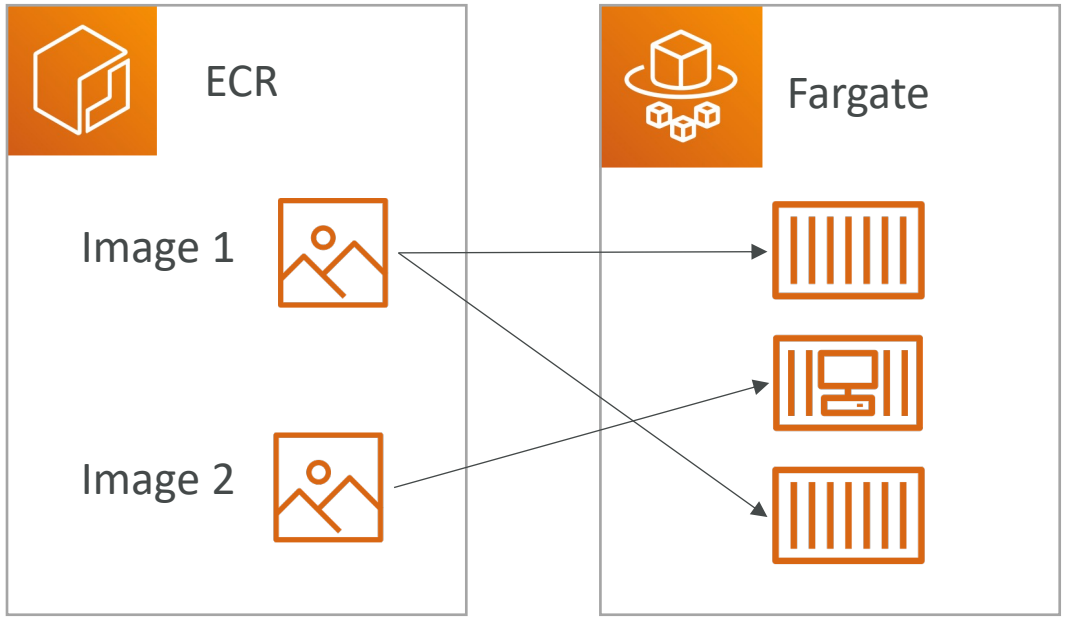
# Fargate

- Launch Docker containers on AWS
- You do not provision the infrastructure (no EC2 instances to manage) – simpler!
- Serverless offering
- AWS just runs containers for you based on the CPU / RAM you need



# ECR

- Elastic Container Registry
- Private Docker Registry on AWS
- This is where you **store your Docker images** so they can be run by ECS or Fargate



# What's serverless?

- Serverless is a new paradigm in which the developers don't have to manage servers anymore…
- They just deploy code
- They just deploy… functions !
- Initially... Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed: "databases, messaging, storage, etc."
- **Serverless does not mean there are no servers…**
  it means you just don't manage / provision / see them

# So far in this course…

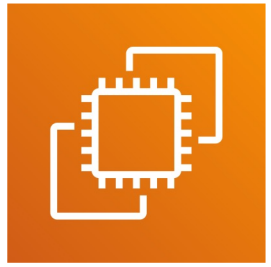
**Amazon S3**

**DynamoDB**

**Fargate**

**Lambda**

# Why AWS Lambda



**Amazon EC2**

- Virtual Servers in the Cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add / remove servers

---

**Amazon Lambda**

- Virtual **functions** – no servers to manage!
- Limited by time - **short executions**
- Run **on-demand**
- Scaling is automated!

# Benefits of AWS Lambda

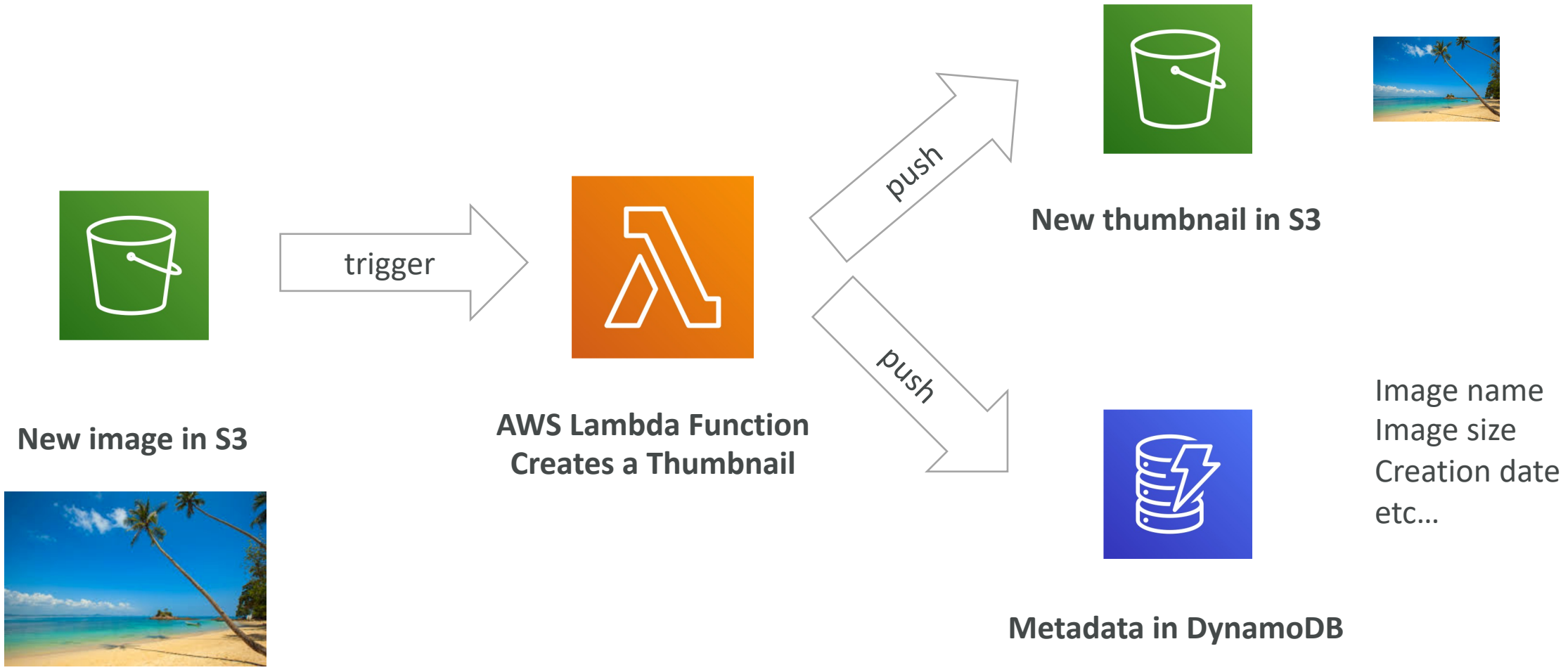
- Easy Pricing:
  - Pay per request and compute time
  - Free tier of 1,000,000 AWS Lambda requests and 400,000 GBs of compute time
- Integrated with the whole AWS suite of services
- **Event-Driven:** functions get invoked by AWS when needed
- Integrated with many programming languages
- Easy monitoring through AWS CloudWatch
- Easy to get more resources per functions (up to 10GB of RAM!)
- Increasing RAM will also improve CPU and network!

# AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)

- Lambda Container Image
  - The container image must implement the Lambda Runtime API
  - ECS / Fargate is preferred for running arbitrary Docker images

# Example: Serverless Thumbnail creation

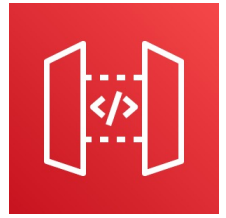


# Example: Serverless CRON Job

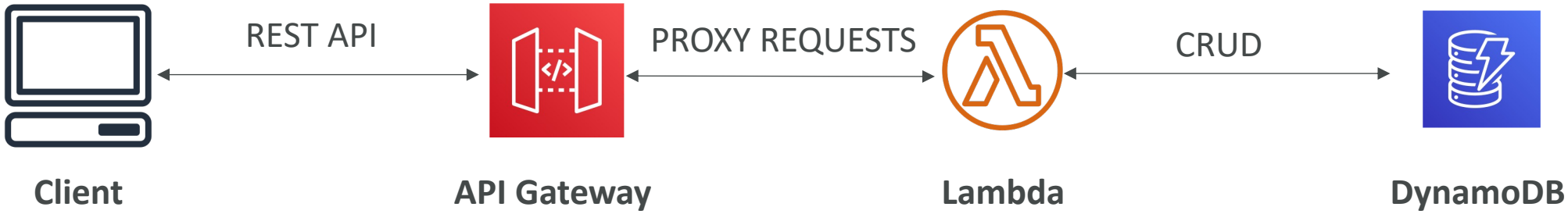
# AWS Lambda Pricing: example

- You can find overall pricing information here: https://aws.amazon.com/lambda/pricing/
- Pay per **calls**:
  - First 1,000,000 requests are free
  - $0.20 per 1 million requests thereafter ($0.0000002 per request)
- Pay per **duration**: (in increment of 1 ms)
  - 400,000 GB-seconds of compute time per month for FREE
  - == 400,000 seconds if function is 1GB RAM
  - == 3,200,000 seconds if function is 128 MB RAM
  - After that $1.00 for 600,000 GB-seconds
- It is usually very cheap to run AWS Lambda so it's very popular

# Amazon API Gateway



- Example: building a serverless API

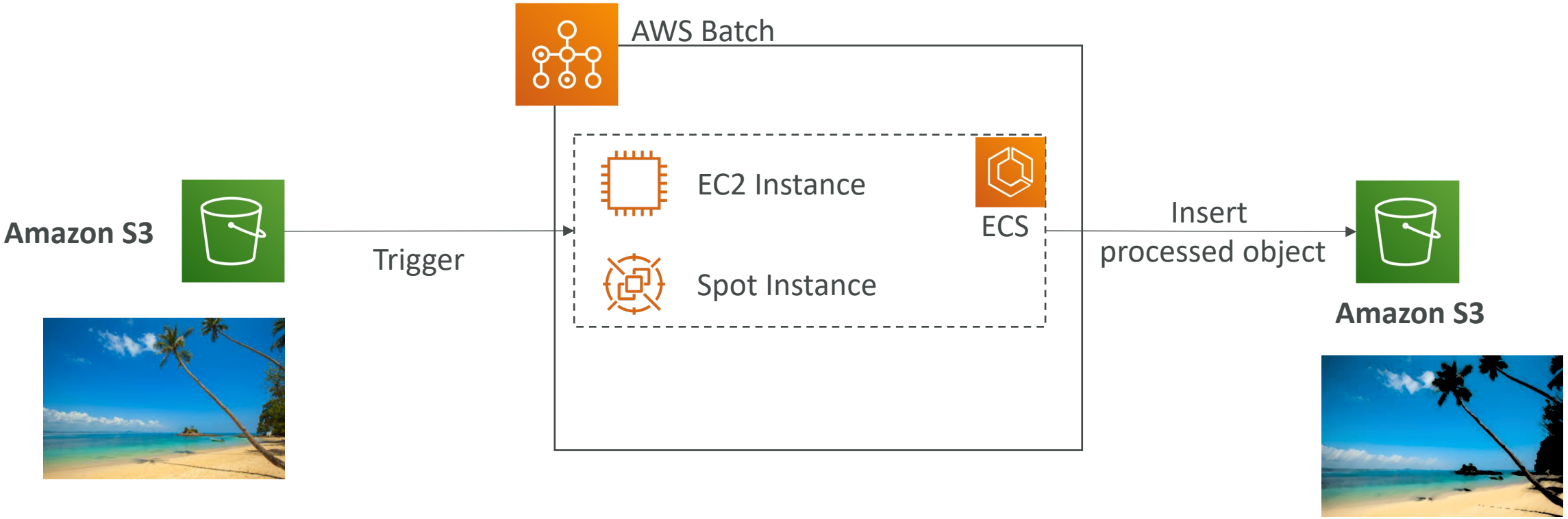



- Fully managed service for developers to easily create, publish, maintain, monitor, and secure APIs
- **Serverless** and scalable
- Supports RESTful APIs and WebSocket APIs
- Support for security, user authentication, API throttling, API keys, monitoring...

# AWS Batch

- **Fully managed** batch processing at **any scale**
- Efficiently run 100,000s of computing batch jobs on AWS
- A "batch" job is a job with a start and an end (opposed to continuous)
- Batch will dynamically launch **EC2 instances** or **Spot Instances**
- AWS Batch provisions the right amount of compute / memory
- You submit or schedule batch jobs and AWS Batch does the rest!
- Batch jobs are defined as **Docker images** and **run on ECS**
- Helpful for cost optimizations and focusing less on the infrastructure

# AWS Batch – Simplified Example



# Batch vs Lambda

- Lambda:
  - Time limit
  - Limited runtimes
  - Limited temporary disk space
  - Serverless

- Batch:
  - No time limit
  - Any runtime as long as it's packaged as a Docker image
  - Rely on EBS / instance store for disk space
  - Relies on EC2 (can be managed by AWS)

# Amazon Lightsail

- Virtual servers, storage, databases, and networking
- Low & predictable pricing
- Simpler alternative to using EC2, RDS, ELB, EBS, Route 53…
- Great for people **with little cloud experience**!
- Can setup notifications and monitoring of your Lightsail resources
- Use cases:
  - Simple web applications (has templates for LAMP, Nginx, MEAN, Node.js…)
  - Websites (templates for WordPress, Magento, Plesk, Joomla)
  - Dev / Test environment
- Has high availability but no auto-scaling, limited AWS integrations

# Other Compute - Summary

- **Docker:** container technology to run applications
- **ECS:** run Docker containers on EC2 instances
- **Fargate:**
  - Run Docker containers without provisioning the infrastructure
  - Serverless offering (no EC2 instances)
- **ECR:** Private Docker Images Repository
- **Batch:** run batch jobs on AWS across managed EC2 instances
- **Lightsail:** predictable & low pricing for simple application & DB stacks

# Lambda Summary

- Lambda is Serverless, Function as a Service, seamless scaling, reactive
- **Lambda Billing:**
  - By the time run x by the RAM provisioned
  - By the number of invocations
- **Language Support:** many programming languages except (arbitrary) Docker
- **Invocation time:** up to 15 minutes
- **Use cases:**
  - Create Thumbnails for images uploaded onto S3
  - Run a Serverless cron job
- **API Gateway**: expose Lambda functions as HTTP API

# Deploying and Managing Infrastructure at Scale Section

# What is CloudFormation

- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 instances using this security group
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

# Benefits of AWS CloudFormation (1/2)

- Infrastructure as code
  - No resources are manually created, which is excellent for control
  - Changes to the infrastructure are reviewed through code

- Cost
  - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

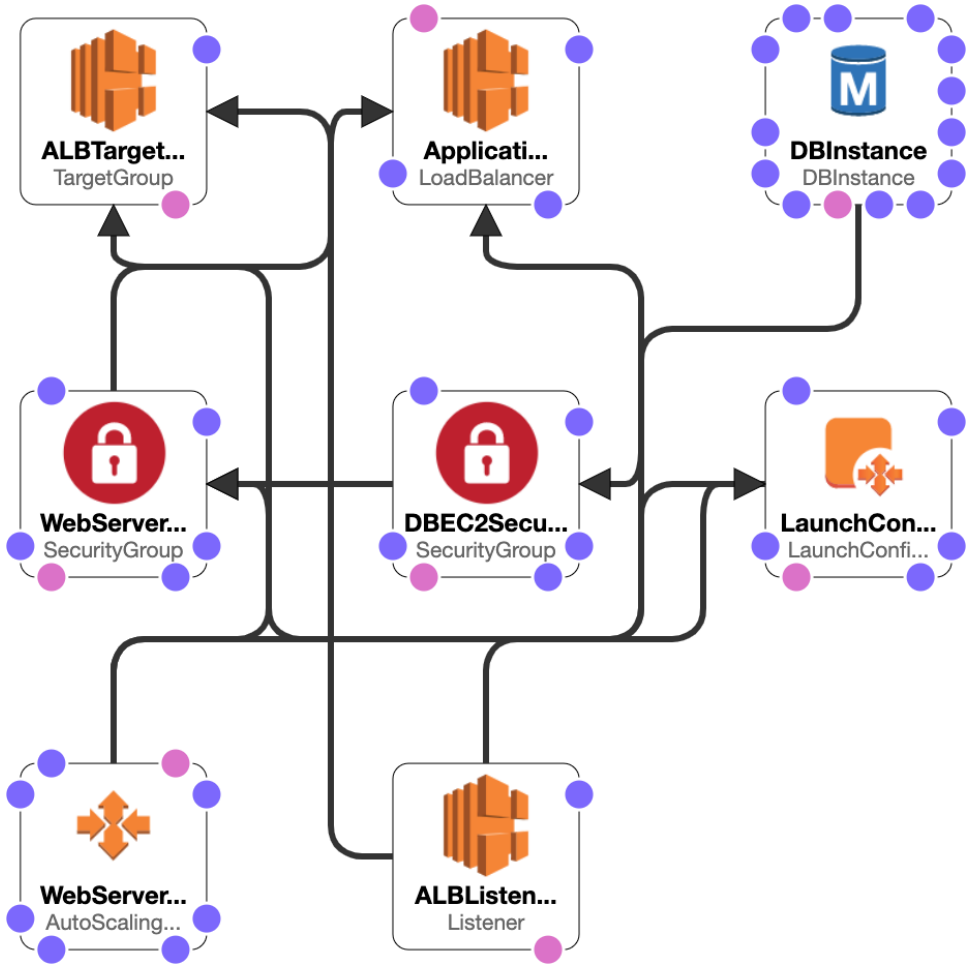
# Benefits of AWS CloudFormation (2/2)

- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)
- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation
- Supports (almost) all AWS resources:
  - Everything we'll see in this course is supported
  - You can use "custom resources" for resources that are not supported

# CloudFormation Stack Designer

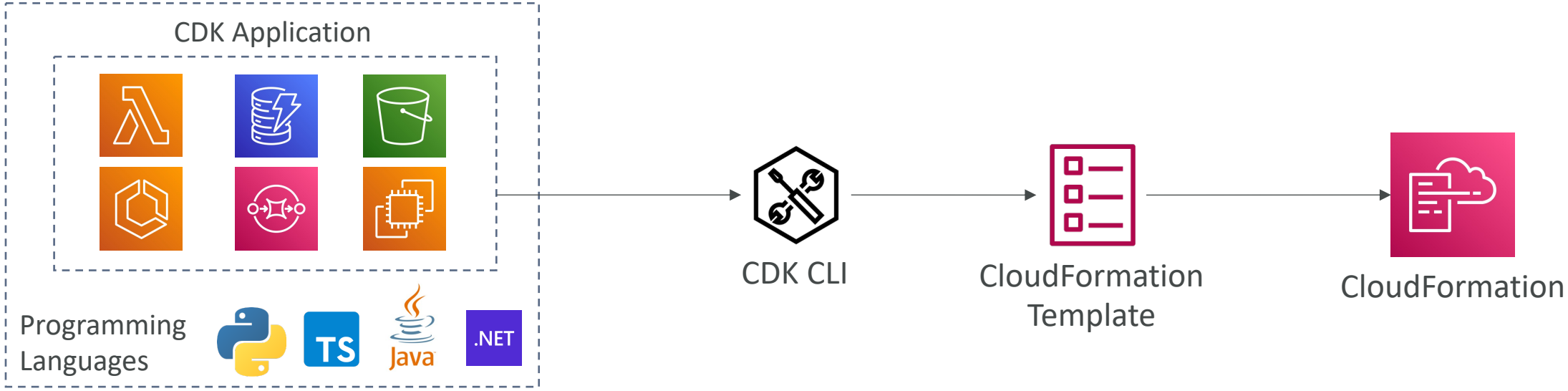
- Example: WordPress CloudFormation Stack

- We can see all the **resources**
- We can see the **relations** between the components



# AWS Cloud Development Kit (CDK)

- Define your cloud infrastructure using a familiar language:
  - JavaScript/TypeScript, Python, Java, and .NET
- The code is "compiled" into a CloudFormation template (JSON/YAML)
- You can therefore deploy infrastructure and application runtime code together
  - Great for Lambda functions
  - Great for Docker containers in ECS / EKS



# CDK Example

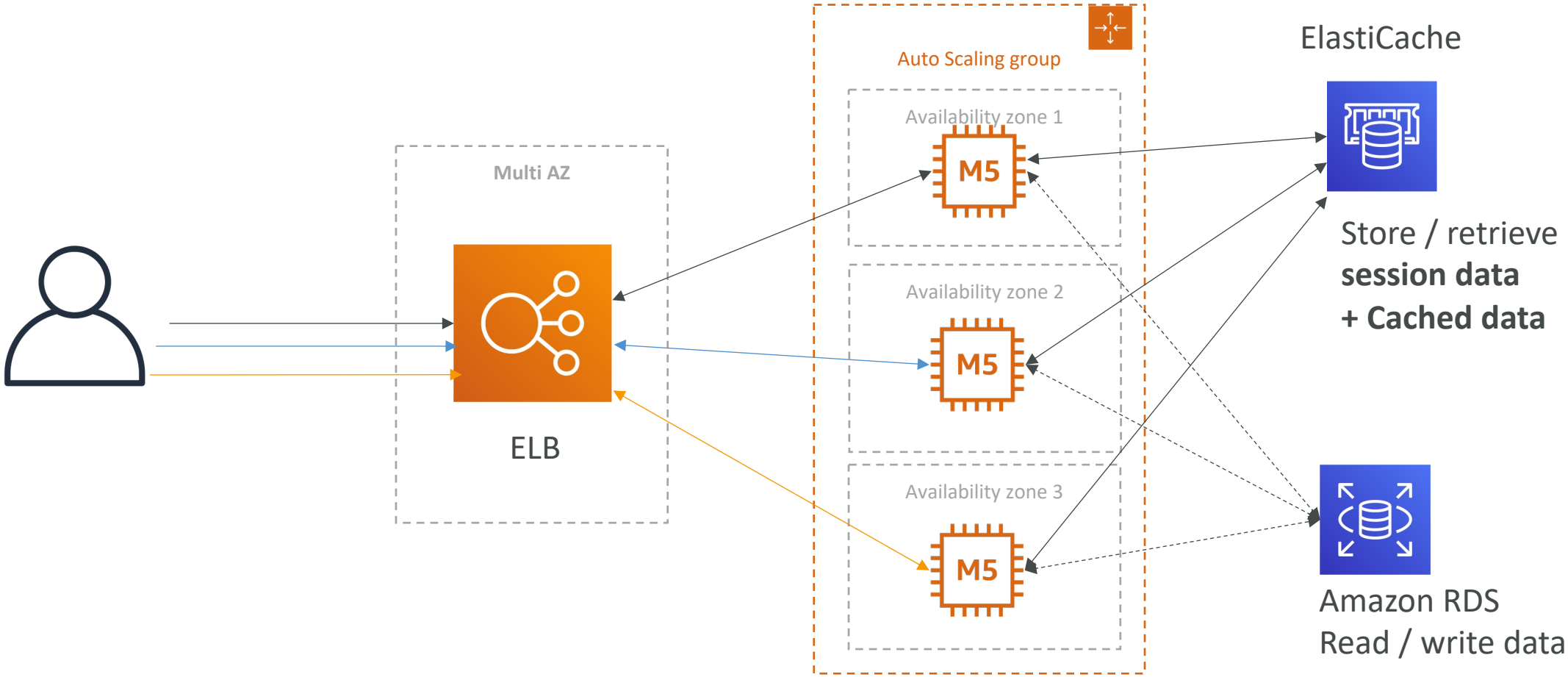
```
export class MyEcsConstructStack extends core.Stack {
  constructor(scope: core.App, id: string, props?: core.StackProps)
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, "MyVpc", {
      maxAzs: 3 // Default is all AZs in region
    });

    const cluster = new ecs.Cluster(this, "MyCluster", {
      vpc: vpc
    });

    // Create a load-balanced Fargate service and make it public
    new ecs_patterns.ApplicationLoadBalancedFargateService(this, "My
      cluster: cluster, // Required
      cpu: 512, // Default is 256
      desiredCount: 6, // Default is 1
      taskImageOptions: { image: ecs.ContainerImage.fromRegistry("am
      memoryLimitMiB: 2048, // Default is 512
      publicLoadBalancer: true // Default is false
    });
  }
}
```

# Typical architecture: Web App 3-tier



# Developer problems on AWS

- Managing infrastructure
- Deploying Code
- Configuring all the databases, load balancers, etc
- Scaling concerns

- Most web apps have the same architecture (ALB + ASG)
- All the developers want is for their code to run!
- Possibly, consistently across different applications and environments

# AWS Elastic Beanstalk Overview

- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG, ELB, RDS, etc…
- But it's all in one view that's easy to make sense of!
- We still have full control over the configuration

- **Beanstalk = Platform as a Service (PaaS)**
- Beanstalk is free but you pay for the underlying instances

# Elastic Beanstalk

- Managed service
  - Instance configuration / OS is handled by Beanstalk
  - Deployment strategy is configurable but performed by Elastic Beanstalk
  - Capacity provisioning
  - Load balancing & auto-scaling
  - Application health-monitoring & responsiveness

- Just the application code is the responsibility of the developer

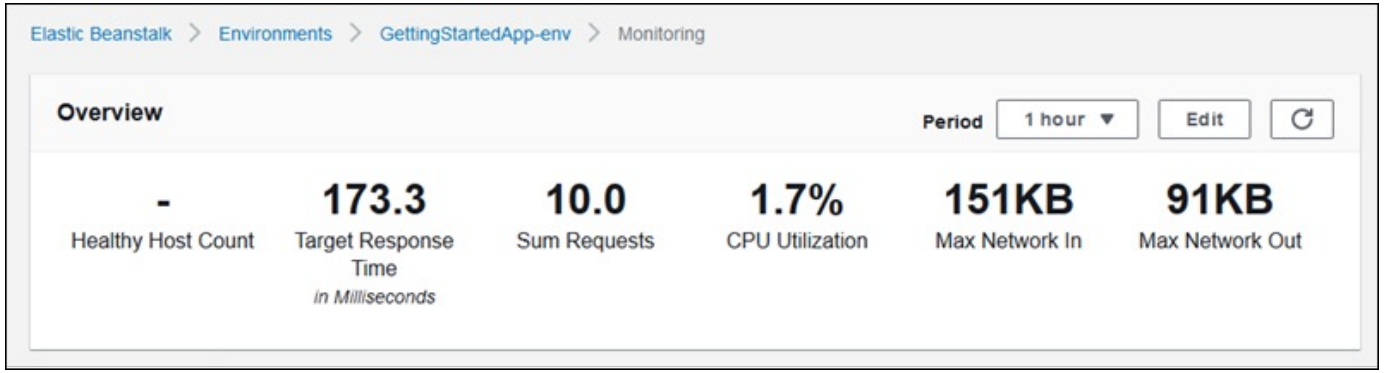
- Three architecture models:
  - Single Instance deployment: good for dev
  - LB + ASG: great for production or pre-production web applications
  - ASG only: great for non-web apps in production (workers, etc..)

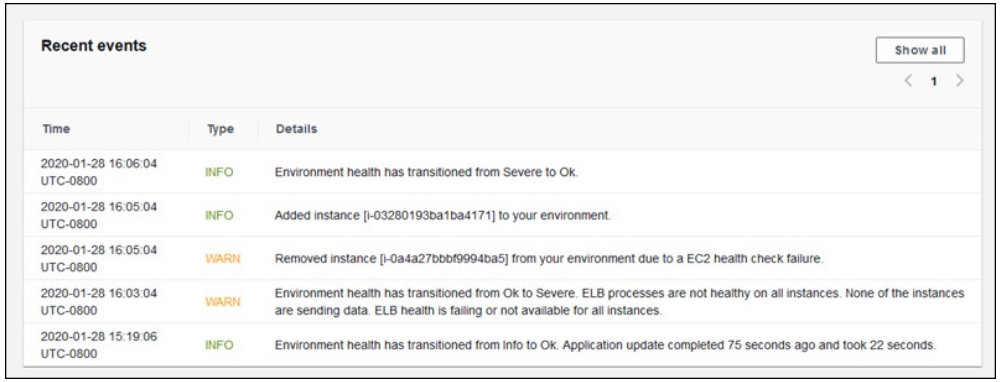
# Elastic Beanstalk

- Support for many platforms:
  - Go
  - Java SE
  - Java with Tomcat
  - .NET on Windows Server with IIS
  - Node.js
  - PHP
  - Python
  - Ruby
  - Packer Builder
  - Single Container Docker
  - Multi-Container Docker
  - Preconfigured Docker
- If not supported, you can write your custom platform (advanced)

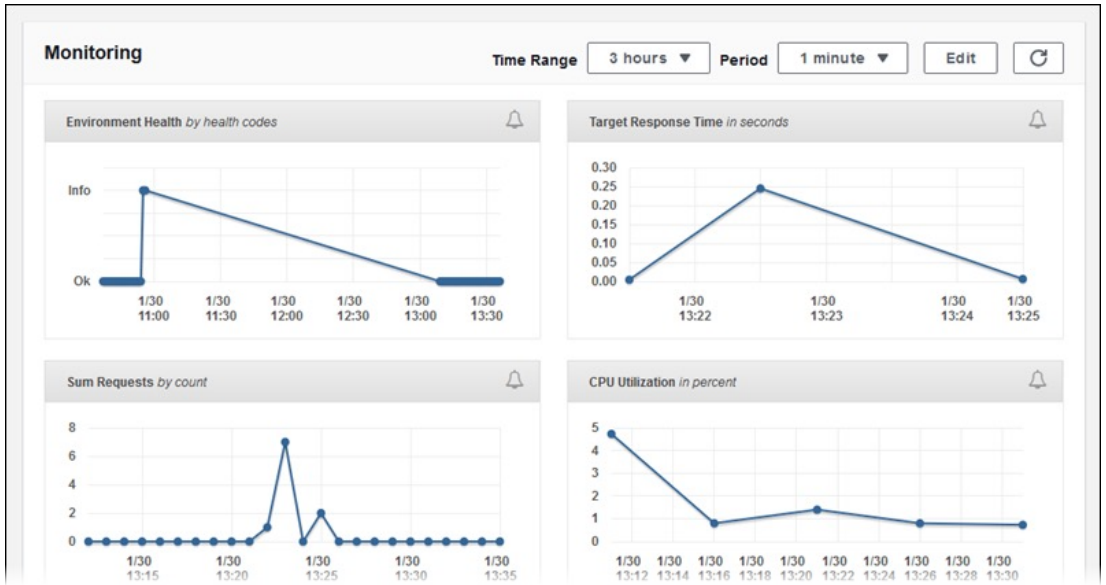
# Elastic Beanstalk – Health Monitoring

- Health agent pushes metrics to CloudWatch
- Checks for app health, publishes health events





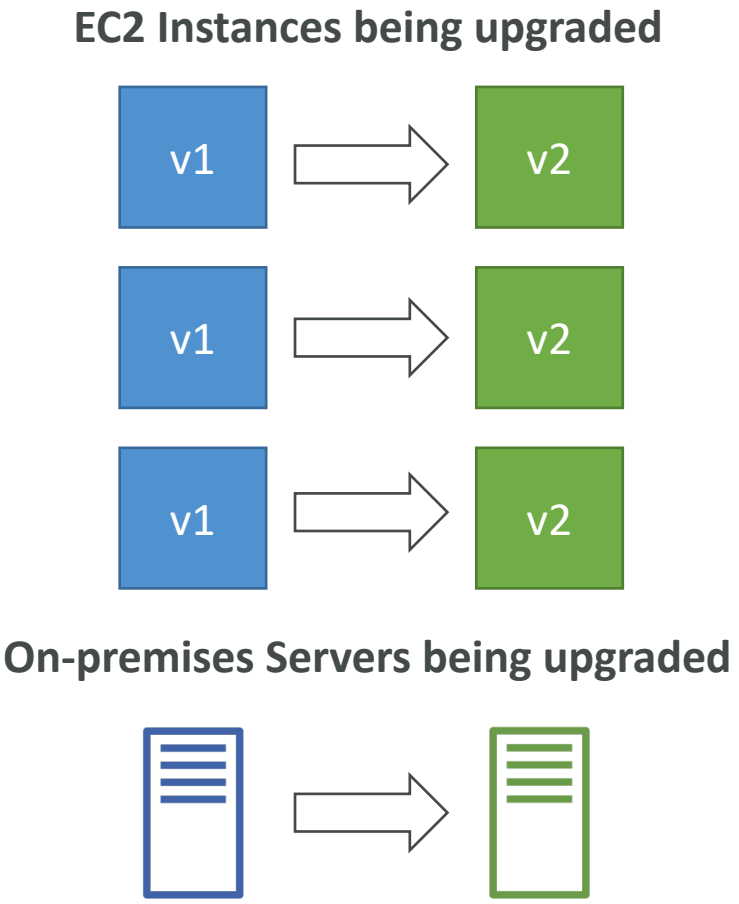


# AWS CodeDeploy

- We want to deploy our application **automatically**

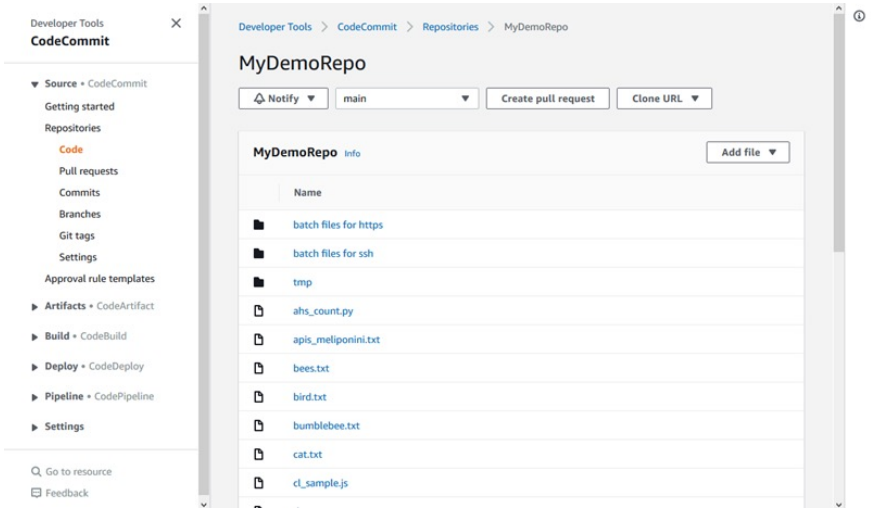
- Works with EC2 Instances
- Works with On-Premises Servers
- Hybrid service

- Servers / Instances must be provisioned and configured ahead of time with the CodeDeploy Agent



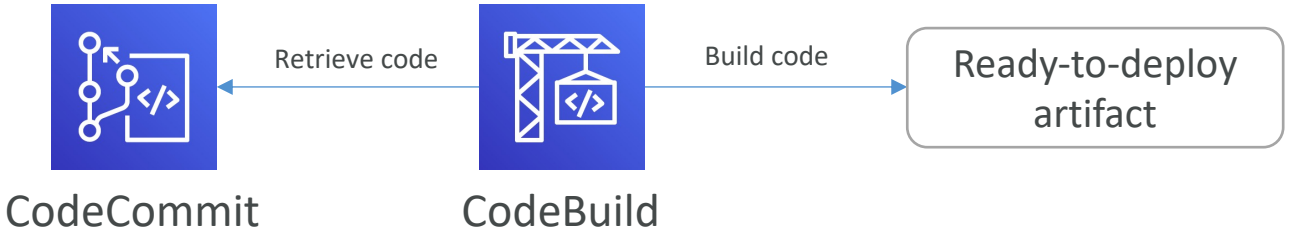
# AWS CodeCommit

- Before pushing the application code to servers, it needs to be stored somewhere
- Developers usually store **code in a repository, using the Git technology**
- A famous public offering is GitHub, AWS' competing product is **CodeCommit**
- CodeCommit:
  - Source-control service that **hosts Git-based repositories**
  - Makes it easy to **collaborate with others on code**
  - The code changes are automatically **versioned**
- Benefits:
  - Fully managed
  - Scalable & highly available
  - Private, Secured, Integrated with AWS



# AWS CodeBuild

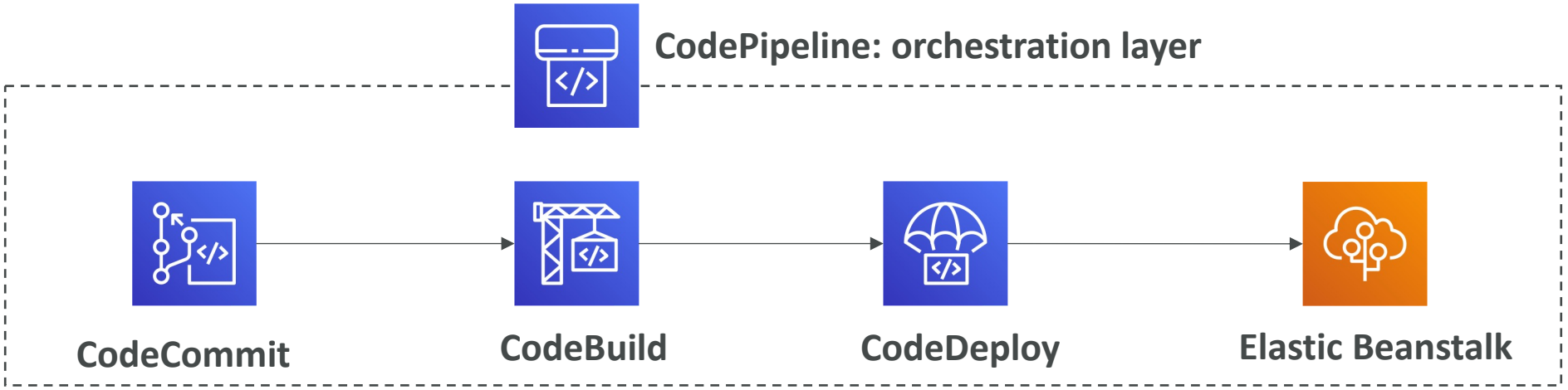
- Code building service in the cloud (name is obvious)
- Compiles source code, run tests, and produces packages that are ready to be deployed (by CodeDeploy for example)




- Benefits:
  - Fully managed, serverless
  - Continuously scalable & highly available
  - Secure
  - Pay-as-you-go pricing – only pay for the build time

# AWS CodePipeline

- Orchestrate the different steps to have the code automatically pushed to production
  - Code => Build => Test => Provision => Deploy
  - Basis for CICD (Continuous Integration & Continuous Delivery)
- Benefits:
  - Fully managed, compatible with CodeCommit, CodeBuild, CodeDeploy, Elastic Beanstalk, CloudFormation, GitHub, 3rd-party services (GitHub…) & custom plugins…
  - Fast delivery & rapid updates

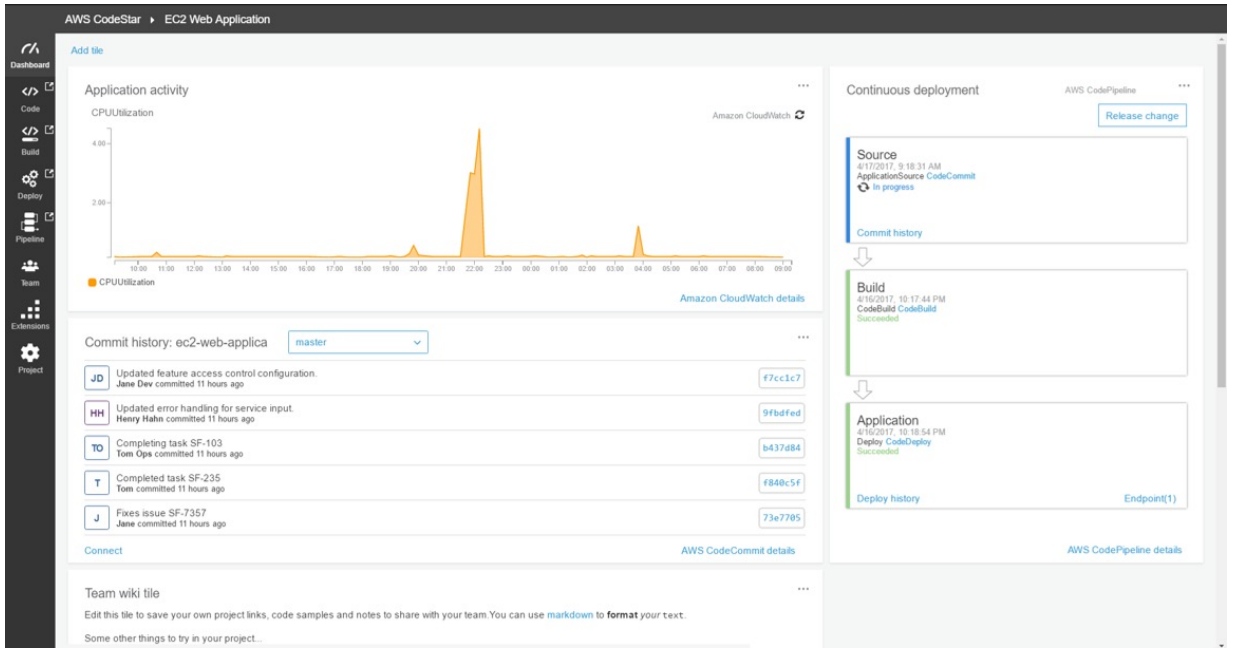


# AWS CodeArtifact

- Software packages depend on each other to be built (also called code dependencies), and new ones are created
- Storing and retrieving these dependencies is called **artifact management**
- Traditionally you need to setup your own artifact management system
- **CodeArtifact** is a secure, scalable, and cost-effective **artifact management** for software development
- Works with common dependency management tools such as Maven, Gradle, npm, yarn, twine, pip, and NuGet
- **Developers and CodeBuild can then retrieve dependencies straight from CodeArtifact**

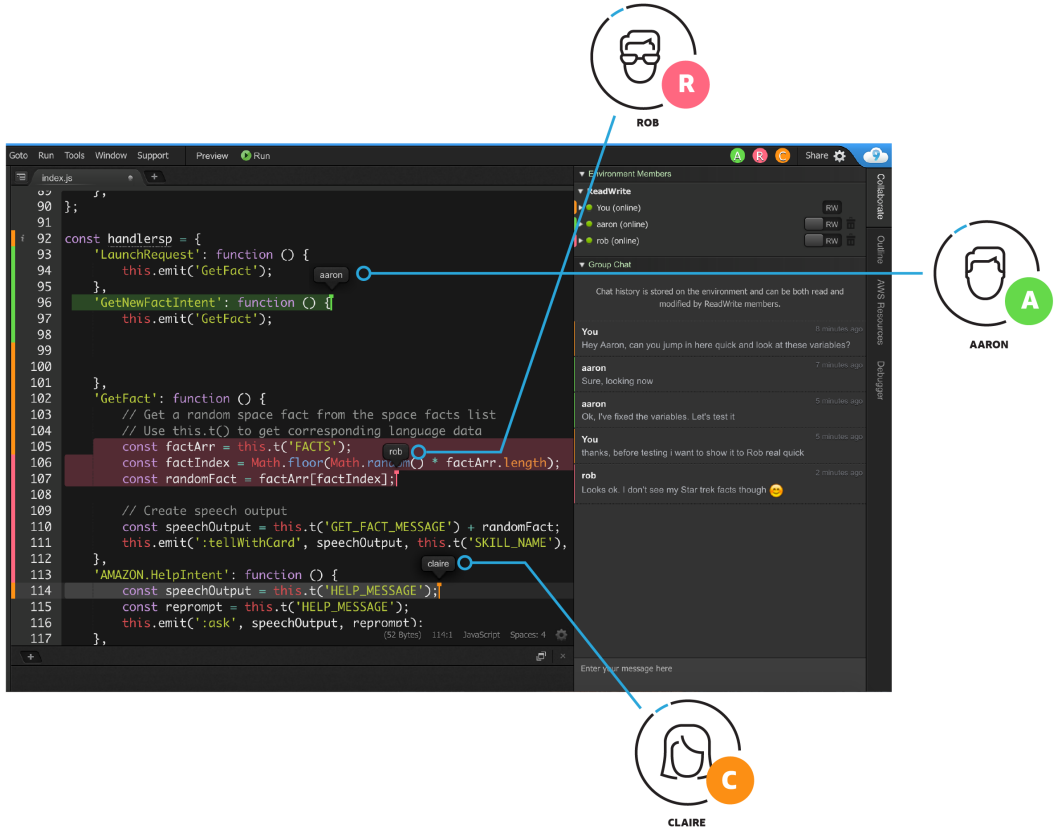
# AWS CodeStar

- **Unified UI** to easily manage software development activities **in one place**
- "Quick way" to get started to correctly set-up CodeCommit, CodePipeline, CodeBuild, CodeDeploy, Elastic Beanstalk, EC2, etc…
- Can edit the code "in-the-cloud" using **AWS Cloud9**



# AWS Cloud9

- AWS Cloud9 is a cloud IDE (Integrated Development Environment) for writing, running and debugging code
- "Classic" IDE (like IntelliJ, Visual Studio Code…) are downloaded on a computer before being used
- A cloud IDE can be used within a web browser, meaning you can work on your projects from your office, home, or anywhere with internet with no setup necessary
- AWS Cloud9 also allows for code collaboration in real-time (pair programming)

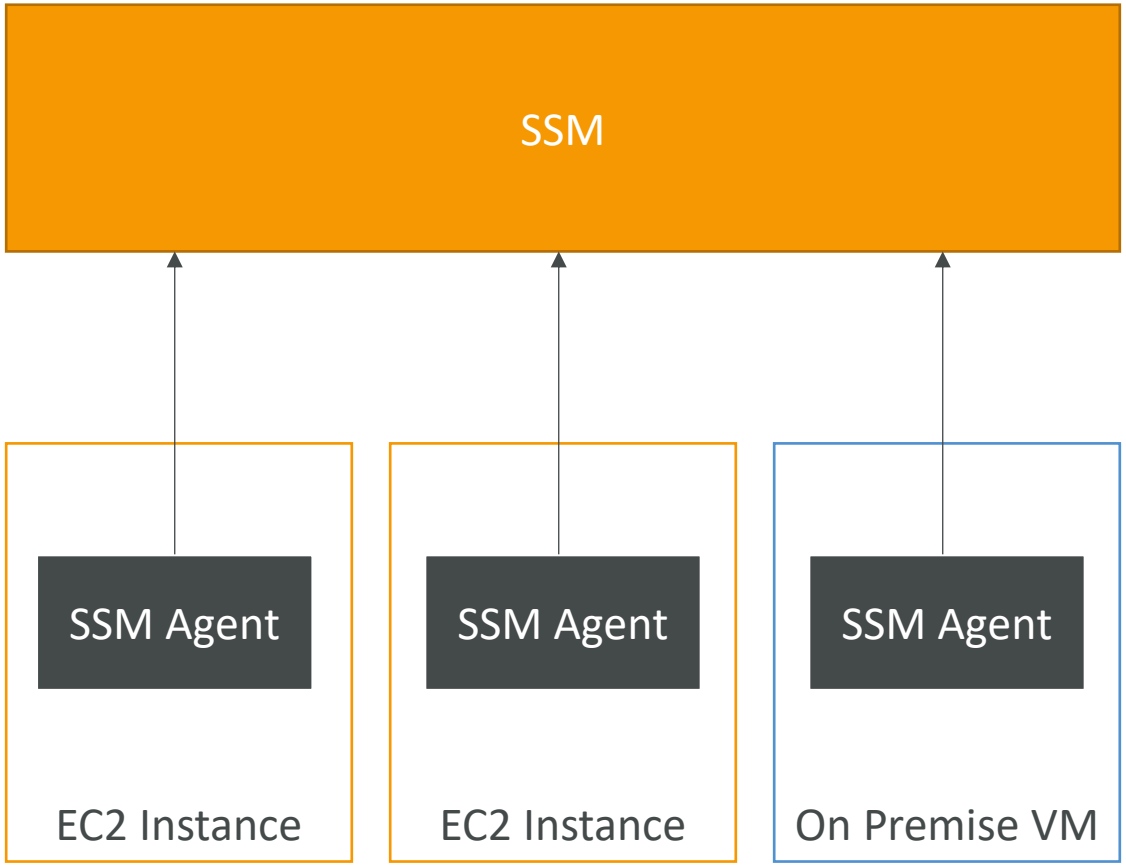


# AWS Systems Manager (SSM)

- Helps you manage your **EC2 and On-Premises** systems at scale
- Another **Hybrid** AWS service
- Get operational insights about the state of your infrastructure
- Suite of 10+ products
- Most important features are:
  - **Patching automation for enhanced compliance**
  - **Run commands across an entire fleet of servers**
  - Store parameter configuration with the SSM Parameter Store
- Works for both Windows and Linux OS

# How Systems Manager works

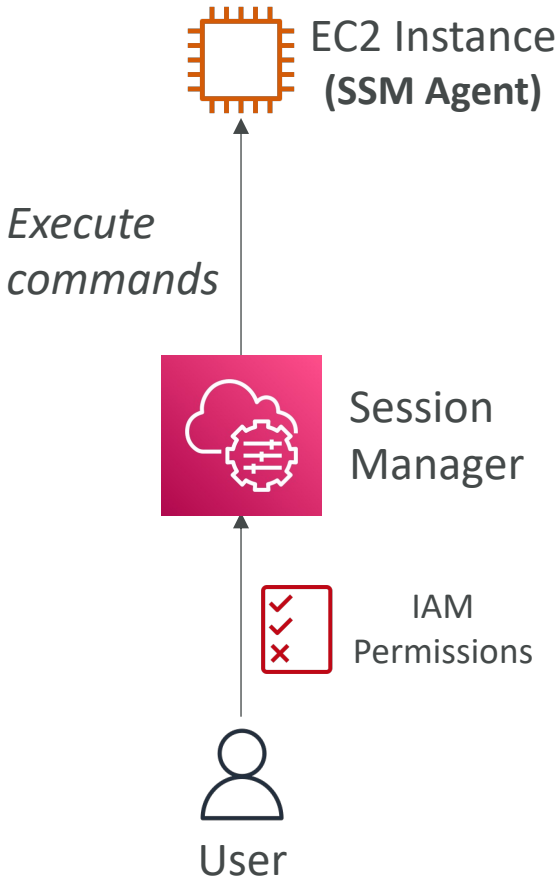
- We need to install the SSM agent onto the systems we control
- Installed by default on Amazon Linux AMI & some Ubuntu AMI
- If an instance can't be controlled with SSM, it's probably an issue with the SSM agent!
- Thanks to the SSM agent, we can **run commands**, **patch** & **configure** our servers



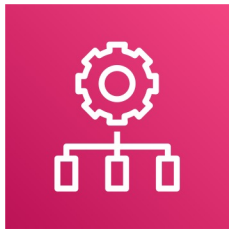
# Systems Manager – SSM Session Manager

- Allows you to start a secure shell on your EC2 and on-premises servers

- No SSH access, bastion hosts, or SSH keys needed
- No port 22 needed (better security)
- Supports Linux, macOS, and Windows
- Send session log data to S3 or CloudWatch Logs



# AWS OpsWorks

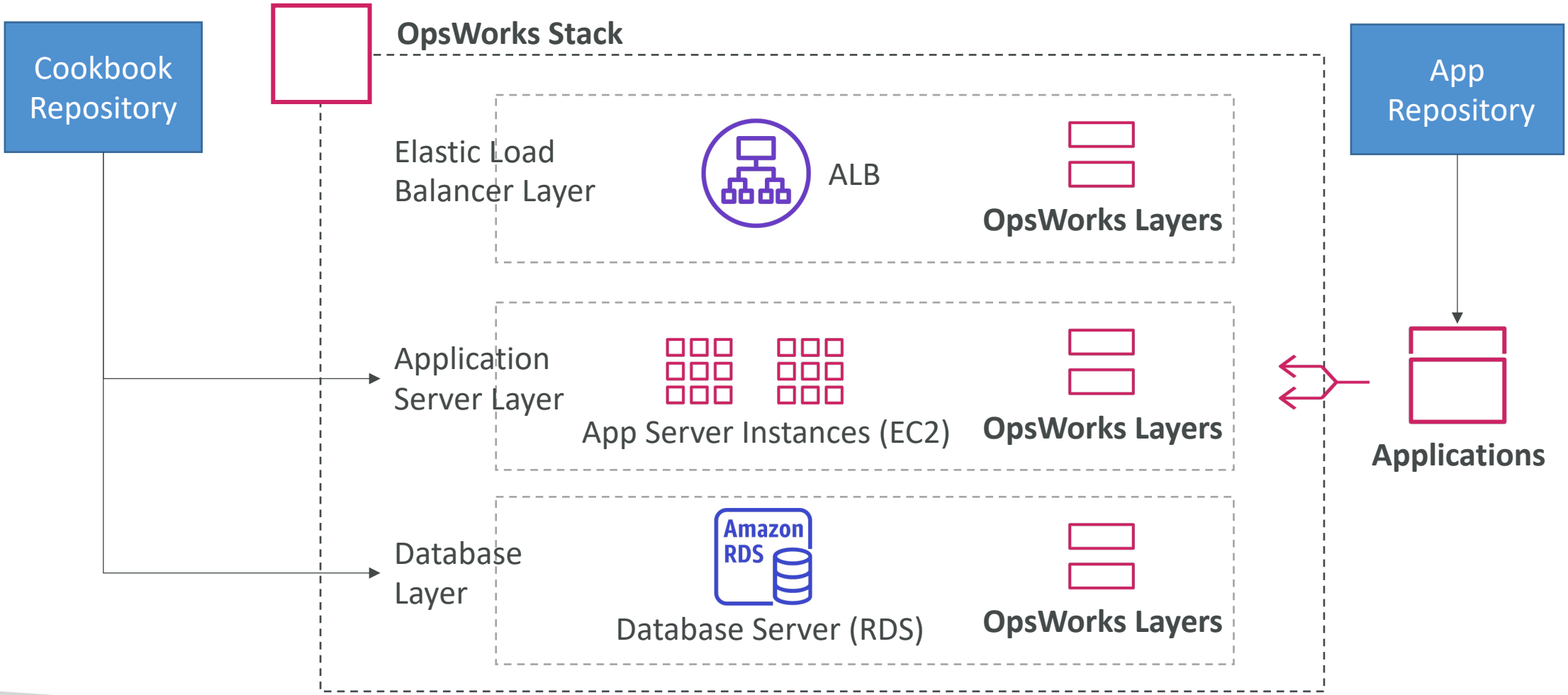


- Chef & Puppet help you perform server configuration automatically, or repetitive actions
- They work great with EC2 & On-Premises VM
- AWS OpsWorks = Managed Chef & Puppet
- It's an alternative to AWS SSM
- Only provision **standard AWS resources**:
  - EC2 Instances, Databases, Load Balancers, EBS volumes…



- In the exam: Chef or Puppet needed => AWS OpsWorks

# OpsWorks Architecture



# Deployment - Summary

- **CloudFormation:** (AWS only)
  - Infrastructure as Code, works with almost all of AWS resources
  - Repeat across Regions & Accounts
- **Beanstalk:** (AWS only)
  - Platform as a Service (PaaS), limited to certain programming languages or Docker
  - Deploy code consistently with a known architecture: ex, ALB + EC2 + RDS
- **CodeDeploy** (hybrid): deploy & upgrade any application onto servers
- **Systems Manager** (hybrid): patch, configure and run commands at scale
- **OpsWorks** (hybrid): managed Chef and Puppet in AWS

# Developer Services - Summary

- **CodeCommit**: Store code in private git repository (version controlled)
- **CodeBuild**: Build & test code in AWS
- **CodeDeploy**: Deploy code onto servers
- **CodePipeline**: Orchestration of pipeline (from code to build to deploy)
- **CodeArtifact**: Store software packages / dependencies on AWS
- **CodeStar**: Unified view for allowing developers to do CICD and code
- **Cloud9**: Cloud IDE (Integrated Development Environment) with collab
- **AWS CDK**: Define your cloud infrastructure using a programming language

# Global Infrastructure Section