AWS Cloud Bootcamp

Agenda (5 days)

- Cloud Native Systems
- History of AWS
- AWS Global Infrastructure
- Identity & Access Management (IAM)
- Elastic Compute Cloud (EC2)
- Virtual Private Cloud (VPC)
- Databases on AWS
  - DynamoDB
  - Aurora
  - ElastiCache
  - RDS
- S3 & Route53
- SQS & SNS
- Terraform & Gitlab Pipelines
- Serverless
  - AWS Lambda
  - API Gateway
- CloudWatch
- CloudTrail
- Key Management
- ECR & EKS Fargate
- Highly Available Architecture

CLOUD NATIVE APPLICATIONS

What is Cloud Native?

- Cloud-native refers to the way an application is built and deployed, specifically leveraging the benefits of the cloud. It implies that the apps exist in the public cloud, rather than within an on-premises datacenter.

- **Traditional development** focuses on the application first and where the infrastructure is hosted as second. Cloud native development is as much a philosophy as it is a set of tools. It's the notion that **we consider all factors from day one**. Through cloud native development we consider where the application is hosted and **what sort of services or technologies can be leveraged** from that platform (e.g. AWS).

Benefits

- Cloud native architectures take full advantage of **on-demand** delivery, **global** deployment, **elasticity**, and **higher-level** services.
- They enable huge improvements in –
  - developer productivity,
  - business agility,
  - scalability,
  - availability,
  - utilization,
  - cost savings.

Basics of cloud-native application architecture

- Cloud-native apps take advantage of cloud computing frameworks and their **loosely coupled** cloud services. Because not all services are on the same server, cloud-native application developers must create a network between machines using software-based architectures. The **services reside on different servers and they run in different locations**. This architecture enables applications to **scale out horizontally**.

- At the same time, these applications must be designed with redundancy. This allows the application to **withstand an equipment failure** and remap Internet Protocol (IP) addresses **automatically**.

What about Infrastructure?

Problems with traditional IT approach

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…)

Benefits of Public Cloud

Benefits of Public Cloud

- 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

Amazon Web Services **(AWS)** owns and maintains the network-connected hardware required for these application services, while you provision and use what you need.

Cloud Computing Deployment Models

Cloud

A cloud-based application is fully deployed in the cloud and all parts of the application run in the cloud. Applications in the cloud have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing. Cloud-based applications can be built on low-level infrastructure pieces or can use higher level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure.

Hybrid

A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure to extend, and grow, an organization's infrastructure into the cloud while connecting cloud resources to internal system. For more information on how AWS can help you with your hybrid deployment, please visit our hybrid page.

On-premises

Deploying resources on-premises, using virtualization and resource management tools, is sometimes called “private cloud”. On-premises deployment does not provide many of the benefits of cloud computing but is sometimes sought for its ability to provide dedicated resources. In most cases this deployment model is the same as legacy IT infrastructure while using application management and virtualization technologies to try and increase resource utilization.

Cloud Computing Models

Infrastructure as a Service (IaaS)

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

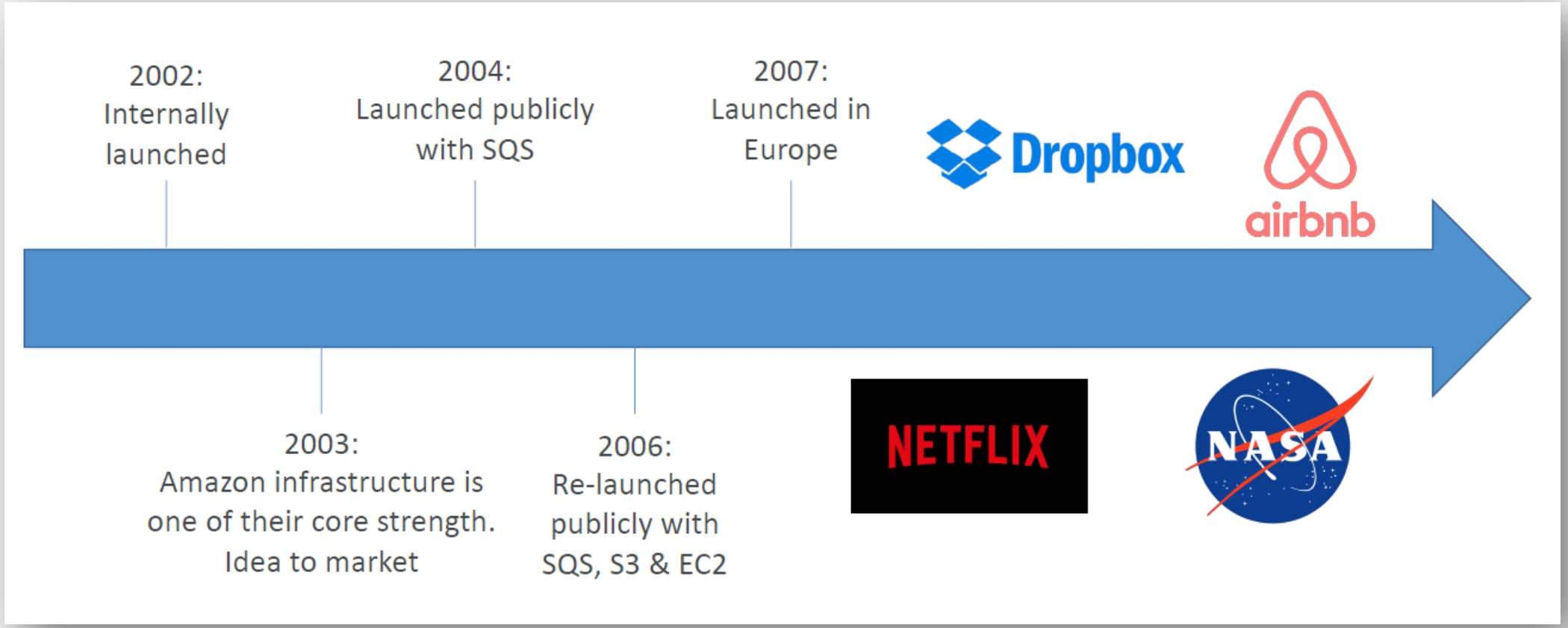
Platform as a Service (PaaS)

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Software as a Service (SaaS)

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.

AWS Cloud History



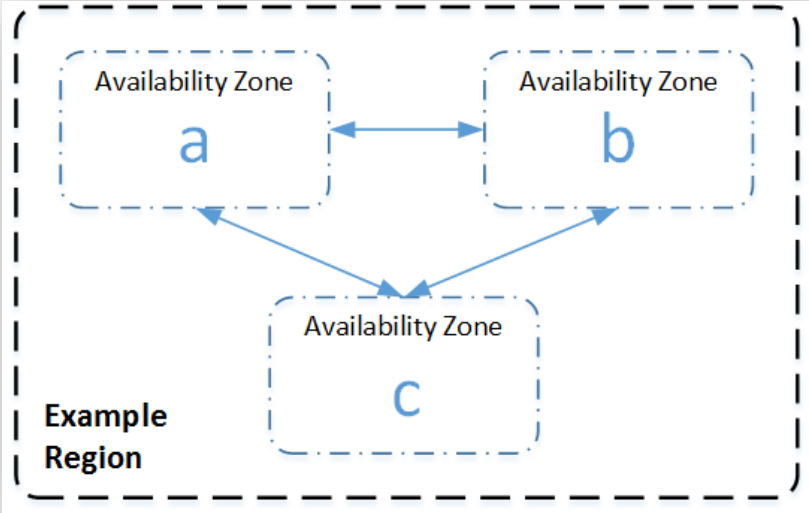
AWS Global Infrastructure Map

Regions & Availability Zones

Regions & Availability Zones

- **Regions**
  - Geographic locations (around the world)
  - Consists of **at least** 3 Availability Zones

- **Availability Zones**
  - Clusters of data centers
  - Isolated in their supplies from other AZ
  - Isolated from failures in other Availability Zones
  - AZs with in a region are connected via private network



Region:
- us-east-1

Availability Zone:
- us-east-1a
- us-east-1b
- us-east-1c

Accessing AWS

- AWS Management Console
  - GUI method via browser
  - Username & password
- AWS CLI
  - Install easily on your laptop / (or use **AWS CloudShell**)
  - Access Key ID & Secret Access Key
- AWS SDK
  - Many programming languages are supported
  - Access Key ID & Secret Access Key

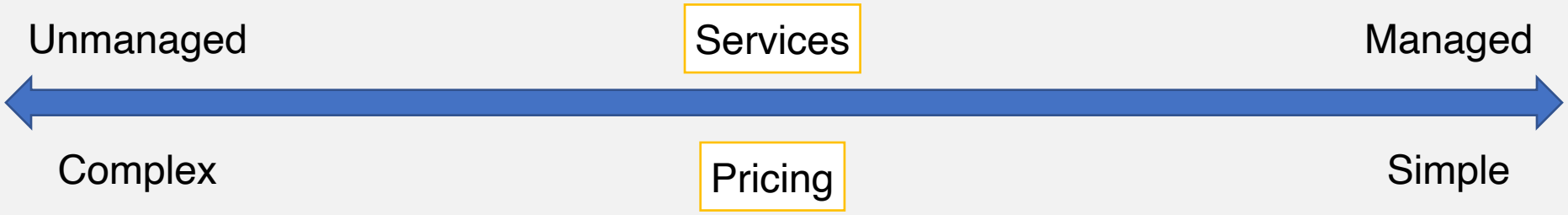
Scope of an AWS Service/Resource

- Global
  - Route53, IAM, CloudFront
- Regional
  - VPC, EFS, Load Balancer, DynamoDB, S3
- AZ based
  - Subnet, EC2, EBS, NAT Gateway, ENI

Data Transfer Considerations

- Data **IN** to AWS is **FREE**.
- Data **OUT** of AWS is **charged**.

The complexity of **Pricing on Cloud** depends on the type of service under consideration.



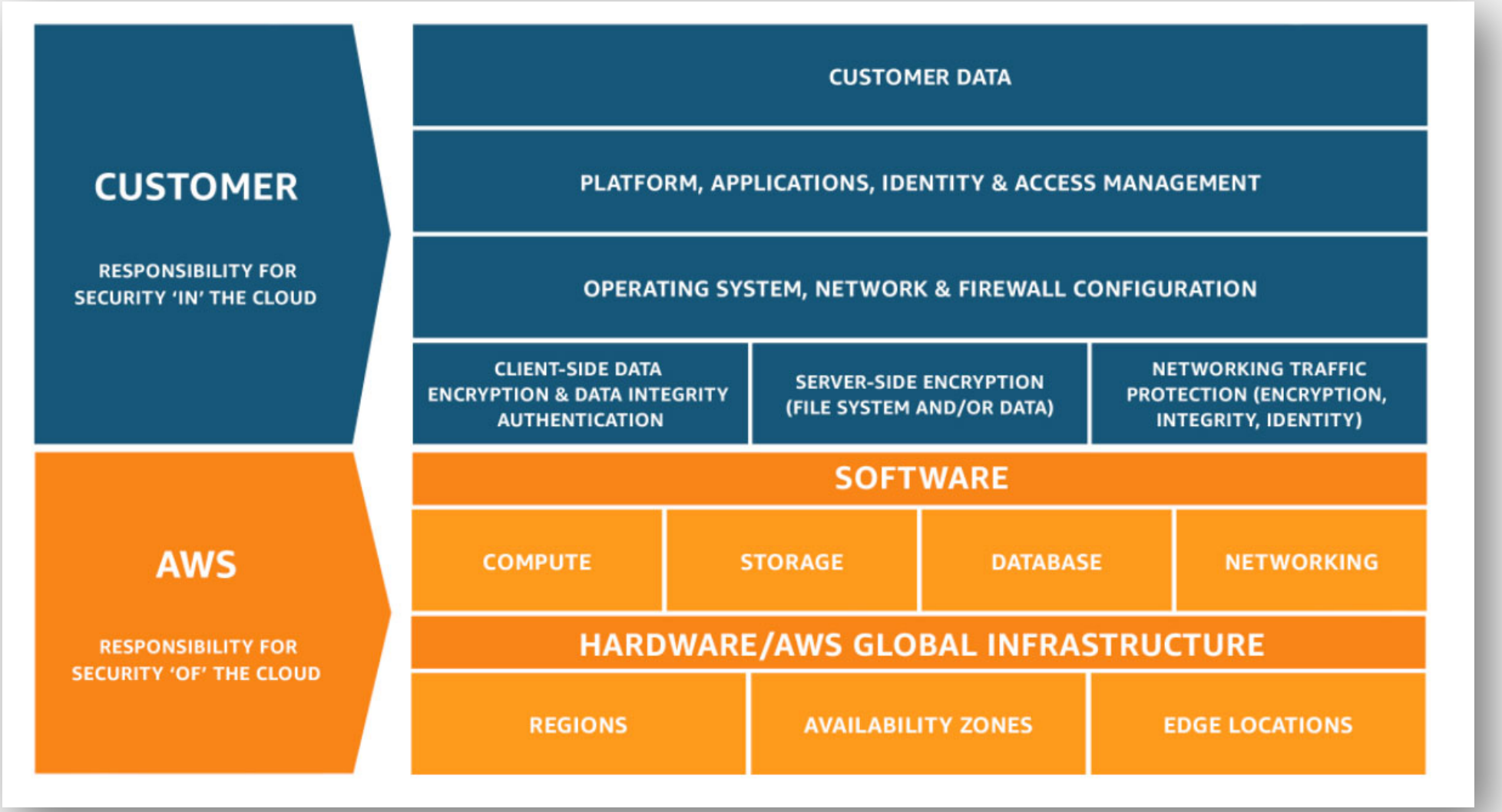
You need to **understand all the factors** around the pricing of a particular service.

Ensuring HA for an application

Ensuring HA for an application

- Any infrastructure component must be located in **at least 2 AZs**
  - Few AWS services do it automatically, for others you need to architect it
- For even higher Service Quality, **deploy in 2 regions**

Shared Responsibility Model



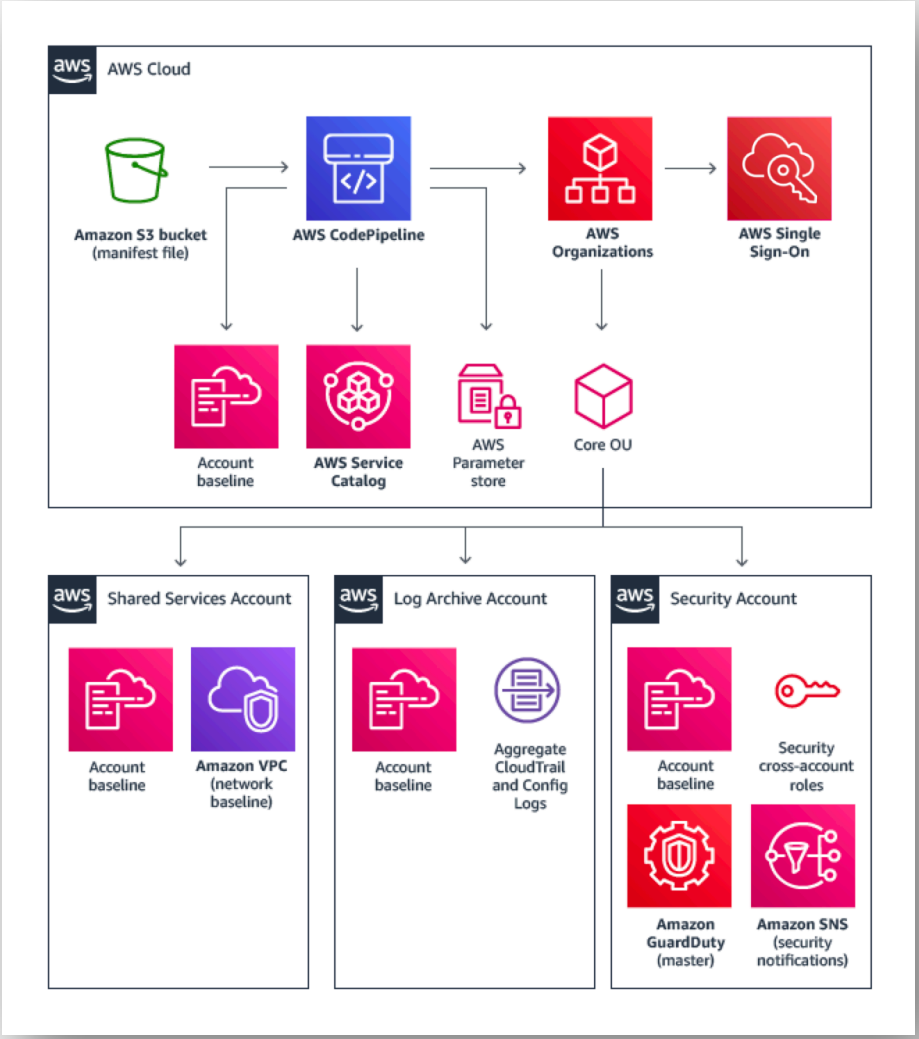
Questions / Quiz Time

IDENTITY & ACCESS MANAGEMENT

How many AWS accounts do you need?

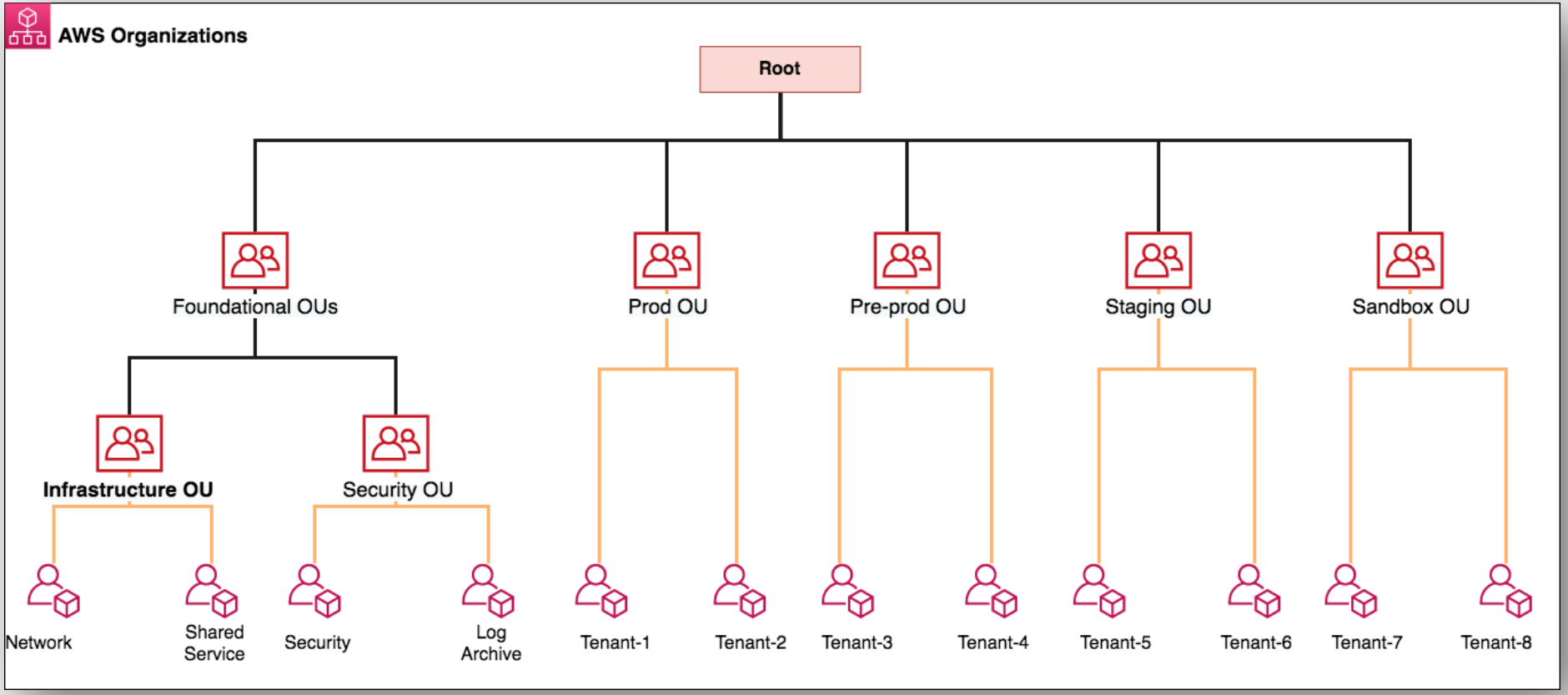
How many AWS accounts do you need?

- Typically more than one account is always required.
- First account is the Master Account (or Management Account). Bills of all the accounts get consolidated here. You get tiered discount as well.
- Other accounts (as per the best practices):
  - Log Archive Account
  - Security Account
  - Shared Services Account
  - Other accounts based on Business Unit, Workload, etc.



Managing multiple accounts - AWS Organizations

- SCPs (Service Control Policy) can be applied at any node.
- **Master/Management Account** remains **unaffected** by **SCPs**.



New AWS Account

- New AWS account creation process
    1. Independent account creation
    2. Under AWS Organizations
- Free Tier usage
- Why are Limits/Quota important?

IAM - Users & Groups

- **Root user** gets created by default. It shouldn't be used or shared (E.g. bu01@abc.com)
- Identity and Access Management (IAM) is a global service
- IAM Users are people within your organization, and can be grouped
- IAM Groups only contain users, not other groups
- Users may (or may not) belong to a group
- A user can belong to multiple groups

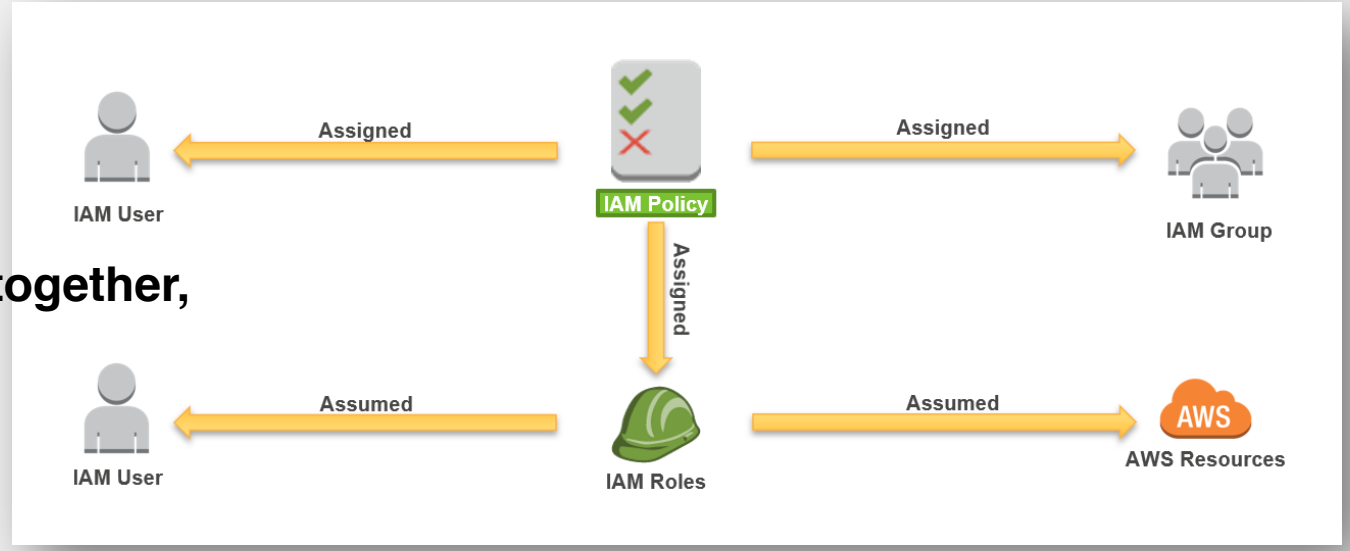
IAM Policies

There are following types of IAM policies:

- Inline Policies – these are written specific to a particular IAM entity.
- Managed Policies – these can be attached to multiple IAM entities.
  - Amazon Managed Policies
  - Customer Managed Policies

When two conflicting permissions come together,

DENY always WINS.



IAM - Roles

- Role is an IAM identity that you can create in your account and has specific permissions. An IAM role has some similarities to an IAM user. Roles and users are both AWS identities with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.
- Roles can be used by the following:
    - An IAM user in the same AWS account as the role
    - An IAM user in a different AWS account than the role
    - A web service offered by AWS such as Amazon EC2

Examples: EC2 Instance Roles, Roles for Lambda Function, Roles for CloudFormation

IAM Best Practices

- **Never** use Root User's access keys. Delete them.
- Create individual IAM users.
- Use groups to assign permissions to IAM users.
- Grant **least privilege** always.
- Configure a **strong** password policy.
- **Enable MFA** for all the IAM users.
- Use **roles** for applications that run on Amazon EC2 instances.
- **Delegate by using roles** instead of by sharing credentials.
- **Rotate** credentials regularly.
- Remove unnecessary/inactive users and credentials.
- Use policy conditions for extra security.

AWS Account for the Labs

➢Please create **only those resources** as instructed during the course.

➢Kindly **delete all the resources** after a lab gets completed (or when instructed).

- Please **do not abuse this AWS environment** by creating any additional resources.
- Open the given **URL** and bookmark it in your browser for this training.
- Use the credentials & AWS account allocated to you.
- Feel free to ask me doubts while doing the labs. But, you should put your best efforts to explore as well.
- Login and change your password for further use. Please make a note of your password carefully!

Demo

1. Create an IAM role “**myEC2Role**”:
   - Allow this role to be assumed by EC2 service.
   - It should allow upload & download to S3 buckets having names starting with “**usaa**”.

2. Create an IAM user “**Steve**” who could do following:
   - View all resources in the account, except the details of EBS volumes.
   - Launch new EC2 instance, stop it, reboot it, but could not Terminate the EC2 instance with tag “Key: Environment”, “Value: Production”

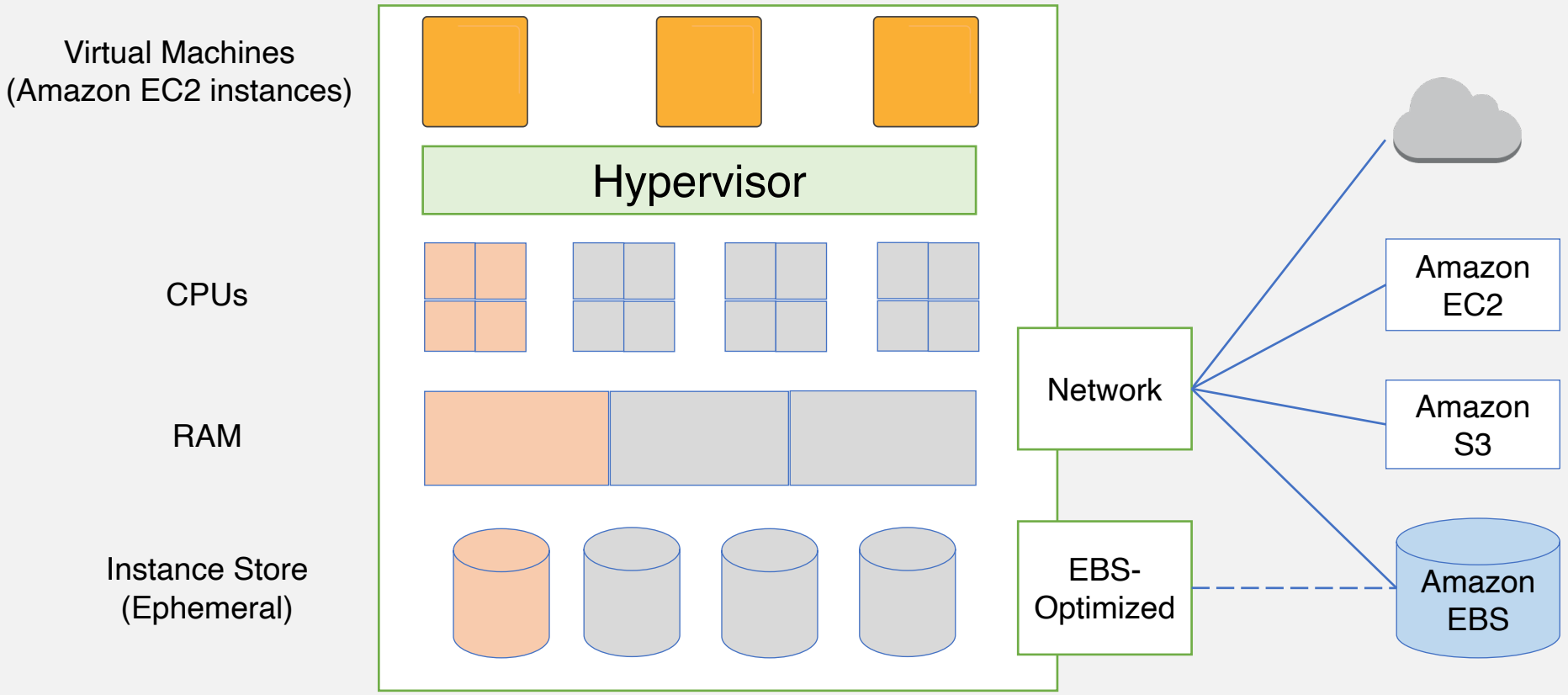
Questions / Quiz Time

ELASTIC COMPUTE CLOUD (EC2)

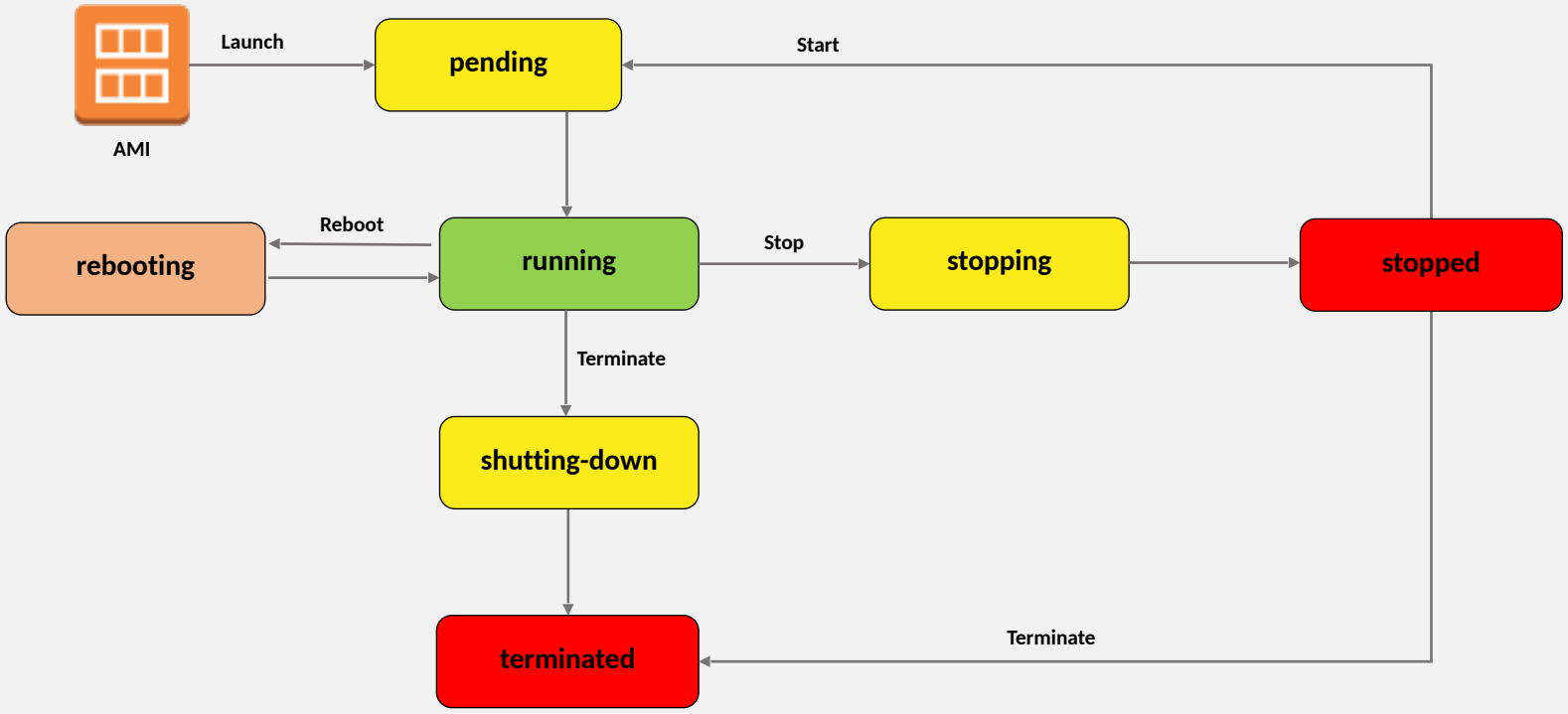
EC2 Basics

- **Operating System (OS):** RHEL, SLES, Ubuntu, Amazon Linux, Windows or Mac OS
- How much compute power & cores (CPU)
- How much random-access memory (RAM)
- How much storage space:
  - Network-attached (EBS)
  - Host (Instance Store)
- **Network card:** speed of the card, Public IP address
- **Firewall rules:** security group
- Bootstrap script (runs once during the launch): EC2 User Data

EC2 Architecture



EC2 Lifecycle



Reboot vs. Stop vs. Terminate

Characteristic	Reboot	Stop/Start (EBS-backed instances only)	Terminate
Host computer	The instance **stays on the same host computer**.	The instance runs on a **new host computer**.	
Public IP address	No change	**New address assigned**	
Elastic IP addresses (EIP)	EIP remains associated with the instance.	EIP remains associated with the instance.	EIP is **disassociated** from the instance.
Instance store volumes	Preserved	**Erased**	**Erased**
EBS volume	Preserved	Preserved	Boot volume is **deleted by default**.
Billing	Instance billing hour doesn't change.	You **stop incurring charges** as soon as state is changed to *stopping*.	You **stop incurring charges** as soon as state is changed to *shutting-down*.

EC2 Pricing

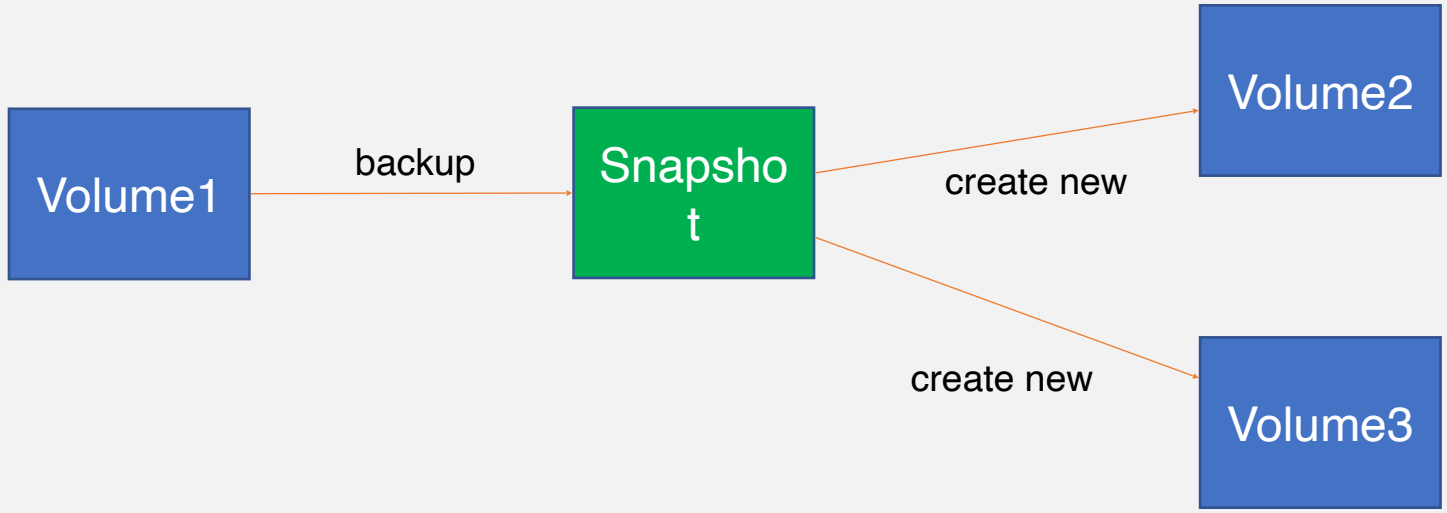
On-Demand	Reserved	Spot Instances
• No commitment • Pay by the hour • Any partial hour converted to full • A new billing cycle starts whenever an instance changes to “Running” state • A billing cycle ends when instance changes to “Stopping” state • Billing cycles don’t start at 9am, 10 am etc. • Pay per second (supported for few Operating Systems)	• Two terms available – 1 year or 3 years • 3 Payment options: - Full Upfront - Partial Upfront - No Upfront (not for 3 years term) • Lot of saving in comparison to On-Demand • Gives you Capacity Guarantee as well • You commit the usage for chosen term • You can re-sell on AWS if you choose not to use • Considered for full term	• Unused capacity at AWS is given in market for bidding • Look at pricing history and decide bid price • Instances are terminated with 2 minutes notice when market price goes above bid price • If terminated by AWS, last partial hour is free • Optionally, use Spot Block option with bid to block the instance (maximum 6 hours)

EBS Types

- **SSD**
  - General Purpose SSD (gp2 & gp3) – consistent IOPS, cheaper
  - Provisioned IOPS SSD (io1 & io2) – higher IOPS and costlier
- **HDD**
  - Throughput Optimized HDD (Higher throughput)
  - Cold HDD (lower throughput, cheaper)
  - These cannot act as the boot disk (OS volume)

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
- A regional resource and stored in Amazon managed S3
- Can copy snapshots across regions



Amazon Machine Image (AMI)

- An AMI includes the following:
  - One or more Amazon Elastic Block Store (Amazon EBS) snapshots.
  - Launch permissions that control which AWS accounts can use the AMI to launch instances.
  - A block device mapping that specifies the volumes to attach to the instance when it's launched.
- While creating an AMI, you add your own software, configuration, operating system, monitoring
- A custom AMI gives faster boot / configuration time because all your software is pre-packaged
- AMI is a regional resource. It can be copied across regions
- You can launch EC2 instances from:
  - A Public AMI: AWS provided
  - Your own AMI: you customize and maintain them yourself
  - An AWS Marketplace AMI: an AMI sold or provided by a 3rd party software vendor

Labs

1. Launch an EC2 instance (Amazon Linux, t2.micro). Connect to it via **EC2-connect** option from your browser. Install **Apache web server** on it. Access this public website from a browser. You may use your mobile to access it as well.
2. Stop this EC2 instance and then create an AMI of it. Launch another instance from the AMI and check if all your changes persist? (i.e. webserver installation).
3. After completing this, **terminate all** the EC2 instances. Also, deregister all the AMIs and delete the snapshots & volumes from your account.

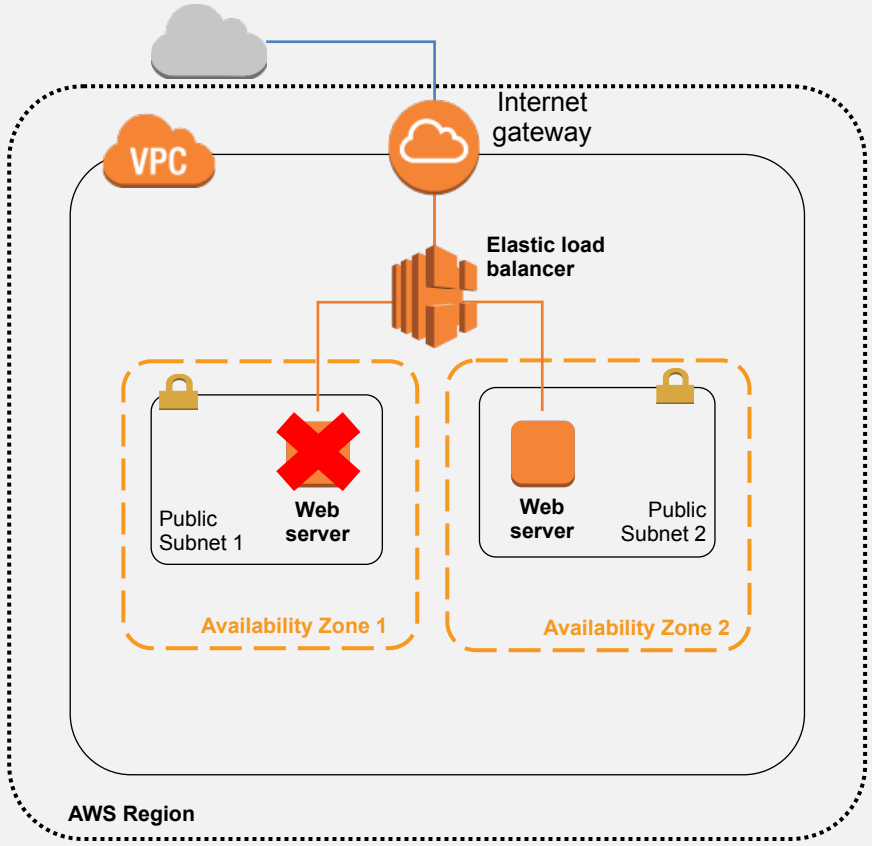
Questions / Quiz Time

How do you choose a region?

- Does the region meet your environment's data sovereignty and compliance requirements?
- How close is the region to your users or data centers?
- Does the region you're considering offer all of the services and features your environment might require?
- Are you choosing the most cost-effective region?

How Many Availability Zones Should I Use?

- Recommendation: Start with two Availability Zones per region.
  - Best practice: If resources in one Availability Zone are unreachable, your application shouldn't fail.
  - Most applications can support two Availability Zones.
  - Using more than two Availability Zones for HA (with in a region) is not usually cost-effective.



VIRTUAL PRIVATE CLOUD (VPC)

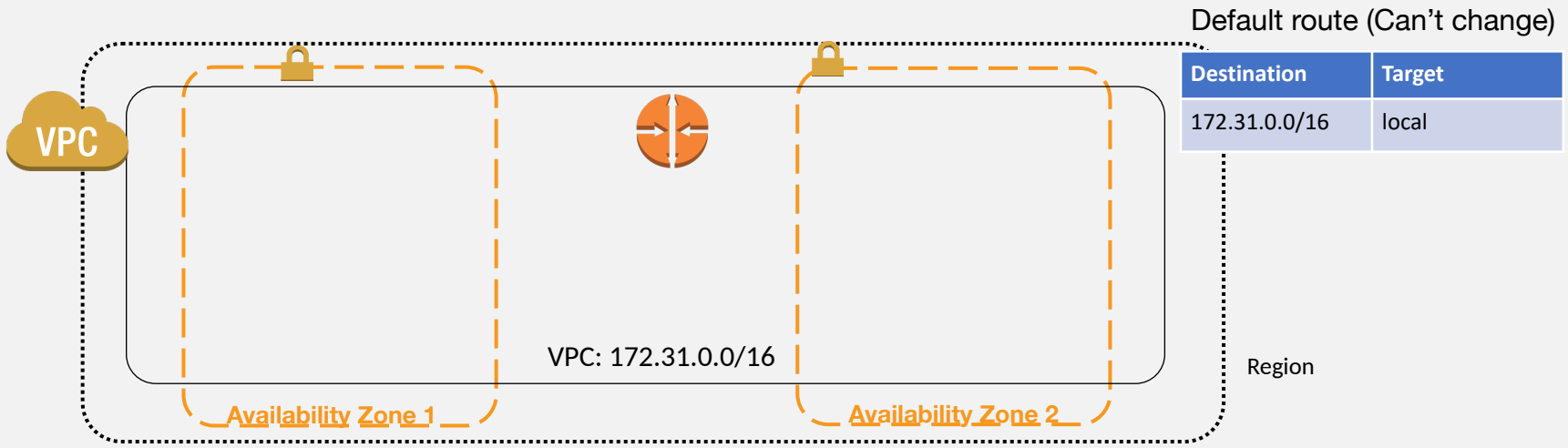
Amazon Virtual Private Cloud (VPC)

- Virtual network; isolated portion of AWS cloud that you design
  - Optional dedicated tenancy
  - Supports logical separation with subnets
  - Fine-grained security
- Private address ranges specified using Classless Inter-Domain Routing (CIDR) notation
- AWS VPCs can use CIDR ranges between /16 and /28.
- For every one step a CIDR range increases, the total number of IPs is cut in half:

CIDR / Total IPs						
/16	/17	/18	/19	/20	/21	/22
65,536	**32,768**	**16,384**	**8,192**	**4,096**	**2,048**	**1,024**
/23	/24	/25	/26	/27	/28	
512	**256**	**128**	**64**	**32**	**16**	

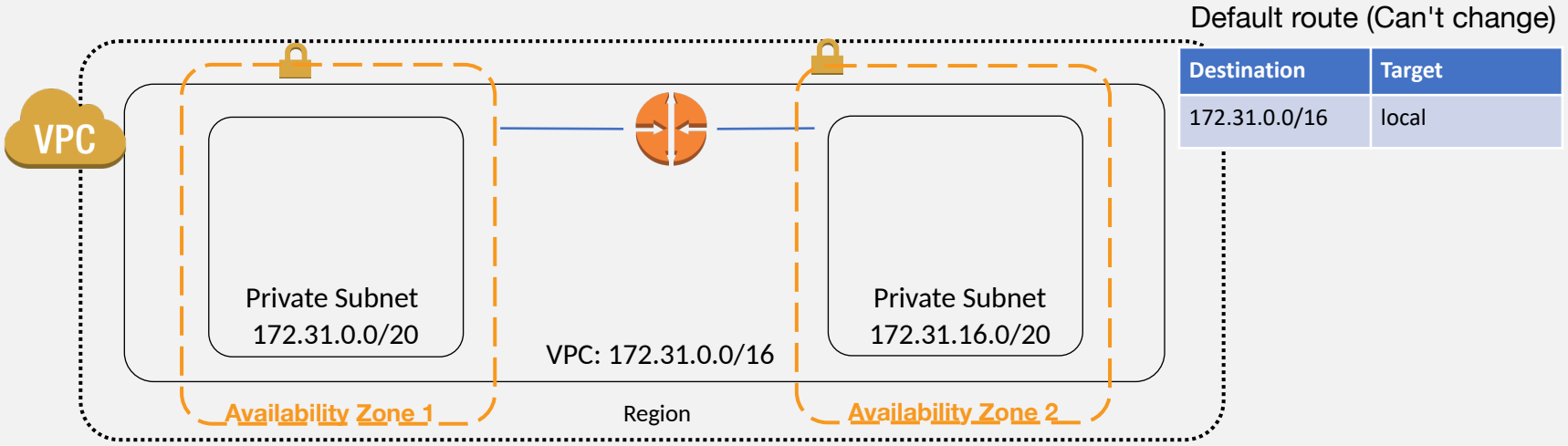
VPC

- VPCs can span across multiple Availability Zones within a region.
- VPCs have an implicit router and a default route table that routes local traffic within the VPC.
- VPCs are private networks until associated with an Internet gateway and a route table rule routing traffic through it.

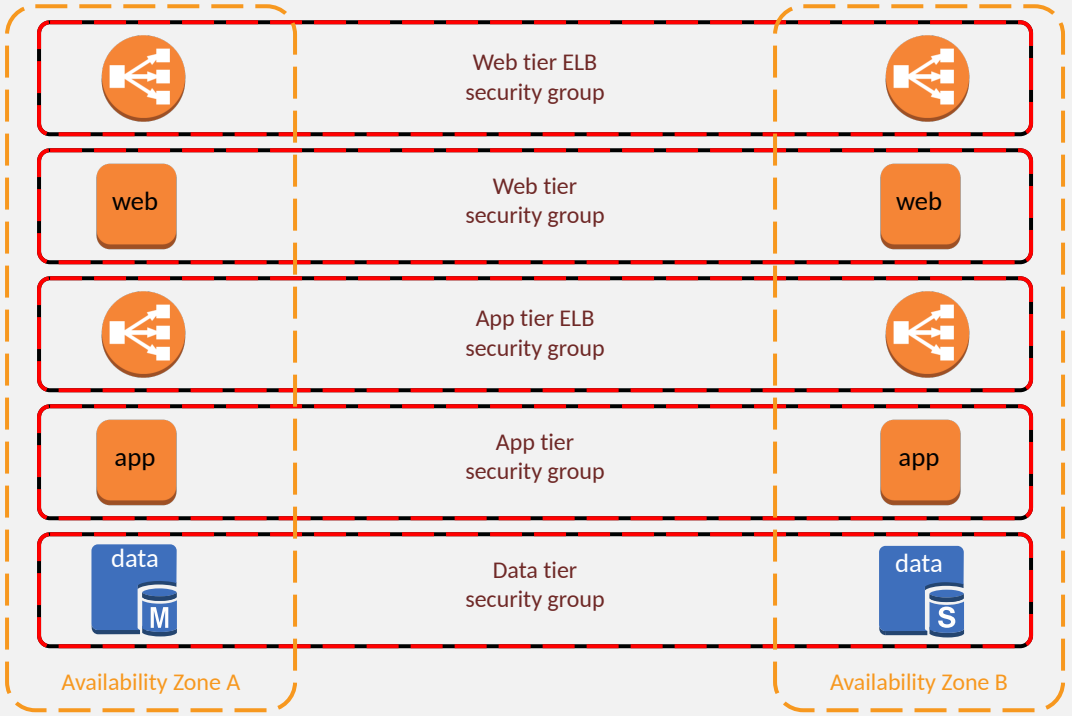


Subnets

- Subnets segment VPC address ranges even further.
- Subnets can exist within one and only one Availability Zone.
- Subnet CIDR blocks within a VPC must not overlap.
- Subnet inbound and outbound traffic can be restricted using NACLs.
- **Recommendation**: Allocate substantially more IPs for private subnets than for public subnets.



Security Group Chaining



Inbound Rule
Allow TCP Port 443
Source: 0.0.0.0/0 (Any)

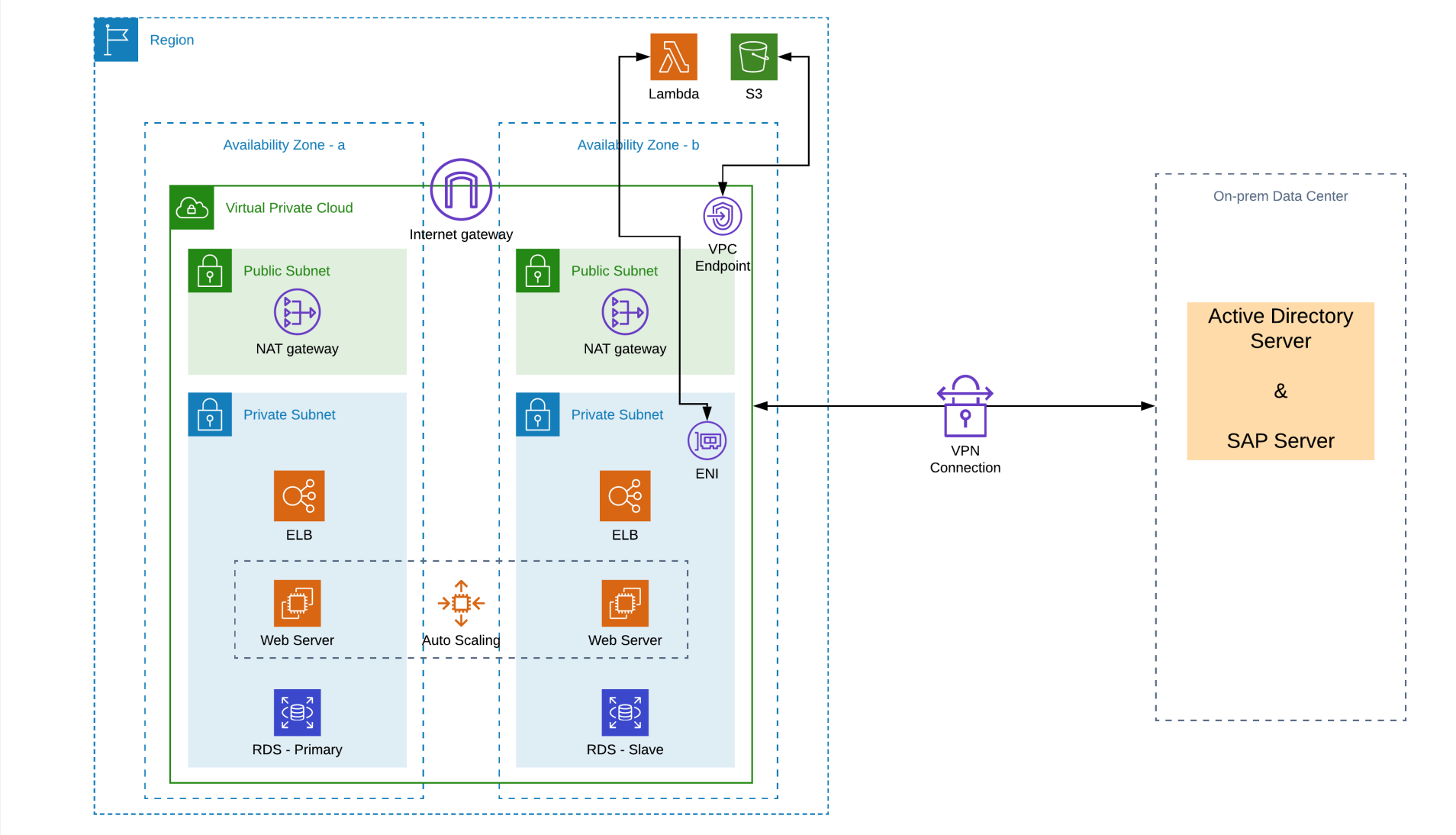
Inbound Rule
Allow TCP Port 80
Source: Web tier ELB

Inbound Rule
Allow TCP Port 8080
Source: Web tier

Inbound Rule
Allow TCP Port 8080
Source: App tier ELB

Inbound Rule
Allow TCP Port 3306
Source: App tier

VPC in action – A Production Setup



Questions / Quiz Time

Labs

- Create a new VPC with CIDR 10.0.0.0/26
- Divide this into 4 subnets of equal size across 2 AZ (e.g. a1, a2, b1, b2)
- Make a1 & b1 as Public subnets. a2 & b2 as Private Subnets.
- Create a NAT gateway in Public subnet and update Route table of Private subnet to use it.
- Launch a (t2.micro) **Linux** instance in Public subnet and another instance in Private subnet.
- Connect to Public & Private instances separately using EC2 Connect.
- Verify if the internet is accessible on both the instances!
- **Delete the NAT Gateway and then release the Elastic IP**. Check again if the internet is accessible on both the instances!
- Stop the Public Instance. Terminate the Private Instance.

DATABASES ON AWS

Unmanaged vs. Managed Services

Unmanaged:

Scaling, fault tolerance, and availability are managed by you.

Amazon Elastic Compute Cloud (EC2)

Managed:

Scaling, fault tolerance, and availability are typically built in to the service.

Amazon Relational Database Service (RDS)

Relational and Non-Relational Databases

	Relational	Non-Relational
Data Storage	Rows and Columns	Key-Value
Schemas	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scalability	Vertical	Horizontal

Relational

ISBN	Title	Author	Format
9182932465265	Cloud Computing Concepts	Wilson, Joe	Paperback
3142536475869	The Database Guru	Gomez, Maria	eBook

Non-Relational

```
{
        ISBN: 9182932465265,
        Title: “Cloud Computing Concepts”,
        Author: “Wilson, Joe”,
        Format: “Paperback”
}
```

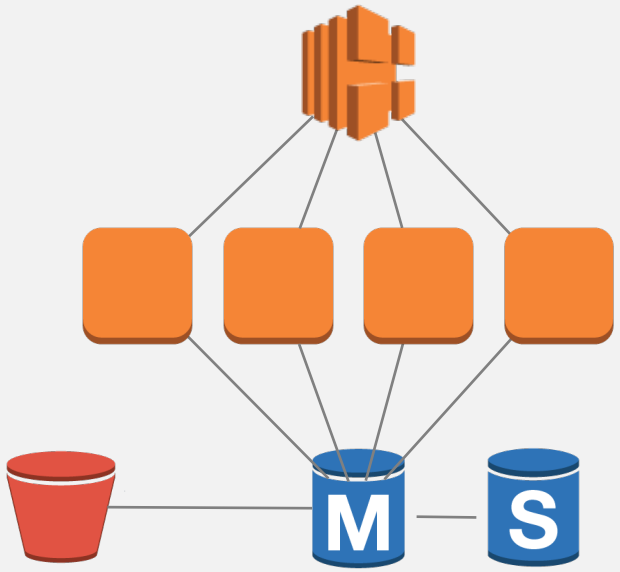
Different Database Services

- Amazon RDS, Aurora – Relational
- Amazon DynamoDB – NoSQL
- Amazon ElastiCache – In memory
- Amazon Redshift – Datawarehouse
- AWS Database Migration Service – For migrating data

RDS

- RDS could be provisioned as Single-AZ or Multi-AZ.
- With Multi-AZ operation, your database is **synchronously** replicated to an instance in another Availability Zone in the same AWS Region.
- **Failover** to the standby automatically occurs in case of master database failure. There will be delay for few seconds for sure.
- Planned maintenance is applied first to standby databases.

A Resilient, Durable Application Architecture



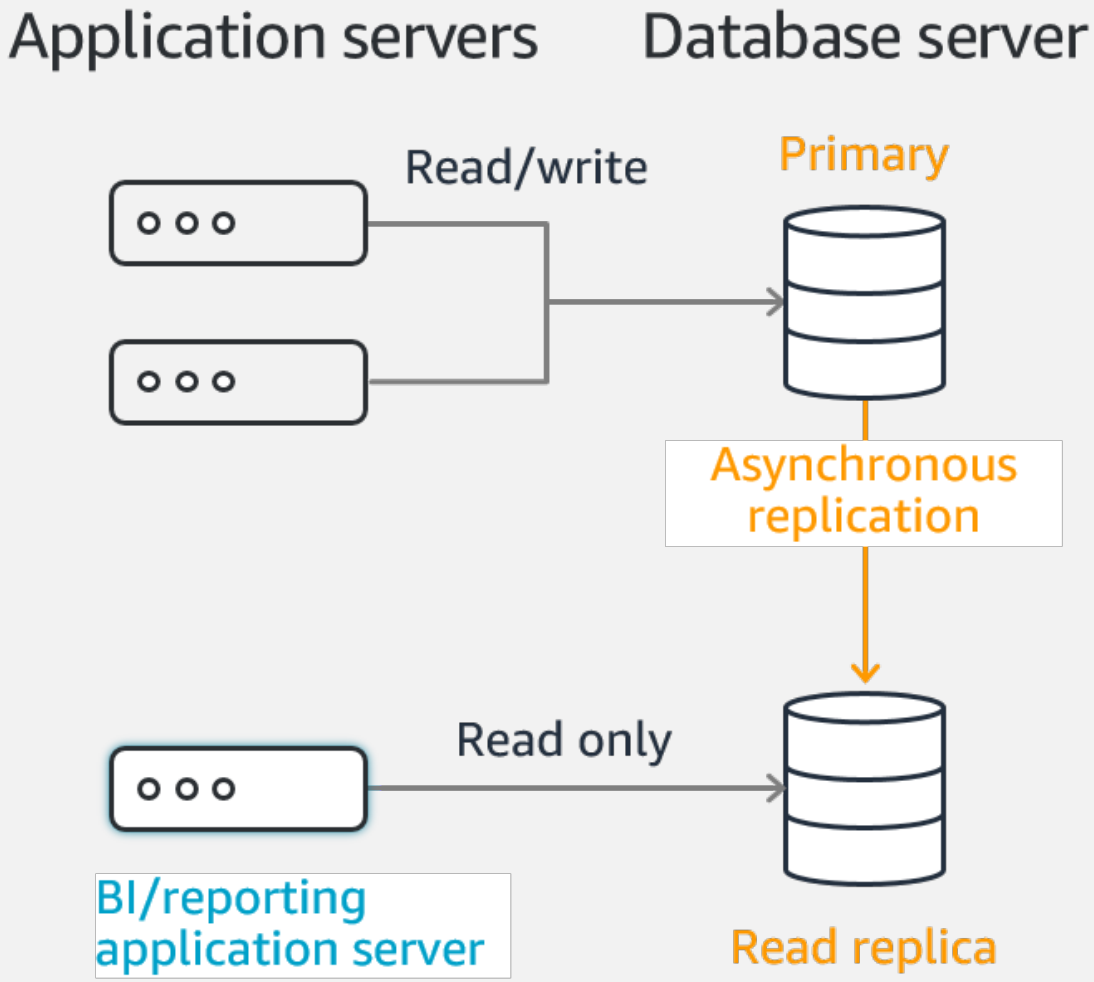
Elastic Load Balancing load balancer instance

Application, in Amazon EC2 instances

Amazon RDS database instances: Master and Multi-AZ standby

DB snapshots in Amazon S3

RDS Read Replica



RDS Backup & Restore

Automatic Backups:

- Restore your database to a point in time.
- Are enabled by default.
- Let you choose a retention period up to 35 days.

Manual Snapshots:

- Let you build a new database instance from a snapshot.
- Are initiated by the user.
- Persist until the user deletes them.
- Are stored in Amazon S3.

Restore Process:

- You can restore till current time minus ten minutes.
- New endpoint gets created for the restored DB.

RDS Optimizes Developer Productivity

- Let developers focus on innovation:
  - Query construction
  - Query optimization

Offload the operational burdens:

- ✓ Migration
- ✓ Backup and recovery
- ✓ Patching
- ✓ Software upgrades
- ✓ Storage upgrades
- ✓ Frequent server upgrades
- ✓ Hardware crash

Labs

- Launch a Multi-AZ RDS (**PostgreSQL**) instance in private subnets of your VPC. (type – db.t3.micro, Storage – GP SSD – 30GB)
- Once it is ready, connect it via the Public instance you had launched earlier.
- Check the private IP of the RDS instance serving you. Make a note of it.
  - You can use “ping –a <<RDS endpoint>>”
- Now, reboot your RDS **with failover option** and check the private IP of the RDS instance serving you. Make a note of this IP as well.
- Verify these in the ENI section (under EC2 Dashboard).
- Follow the instructions and use **psql** to access the RDS database.
  - Create a table and insert data into it.
- Use python code on your EC2 instance to access the data RDS table.

Amazon Aurora

Enterprise database at open source price

Delivered as a **managed** service

Speed and **availability** of high-end commercial databases

Simplicity and **cost-effectiveness** of open source databases

Drop-in **compatibility** with MySQL and PostgreSQL

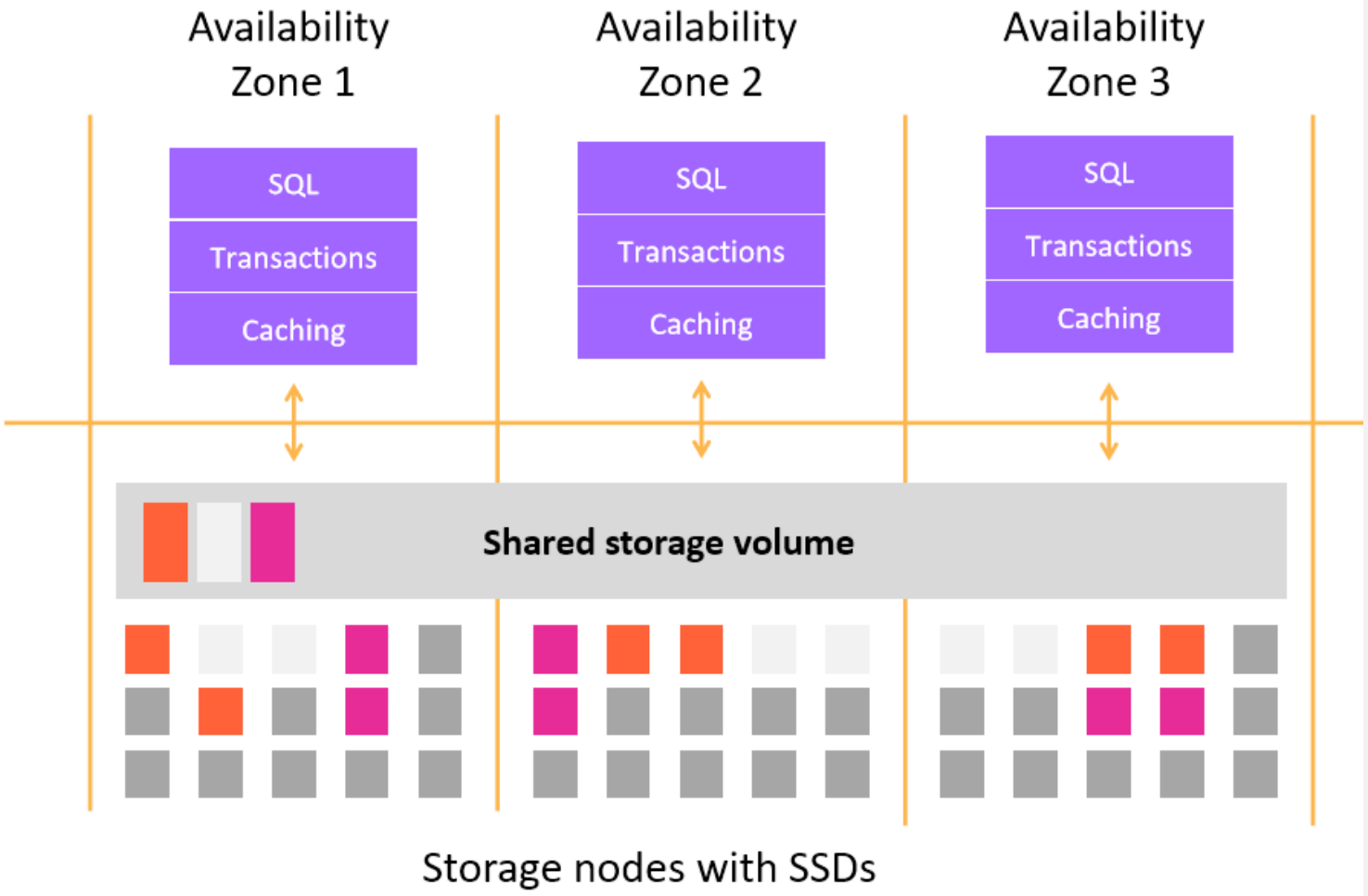
Simple **pay-as-you-go** pricing

Scale-out, distributed architecture

Purpose-built log-structured distributed storage system designed for databases

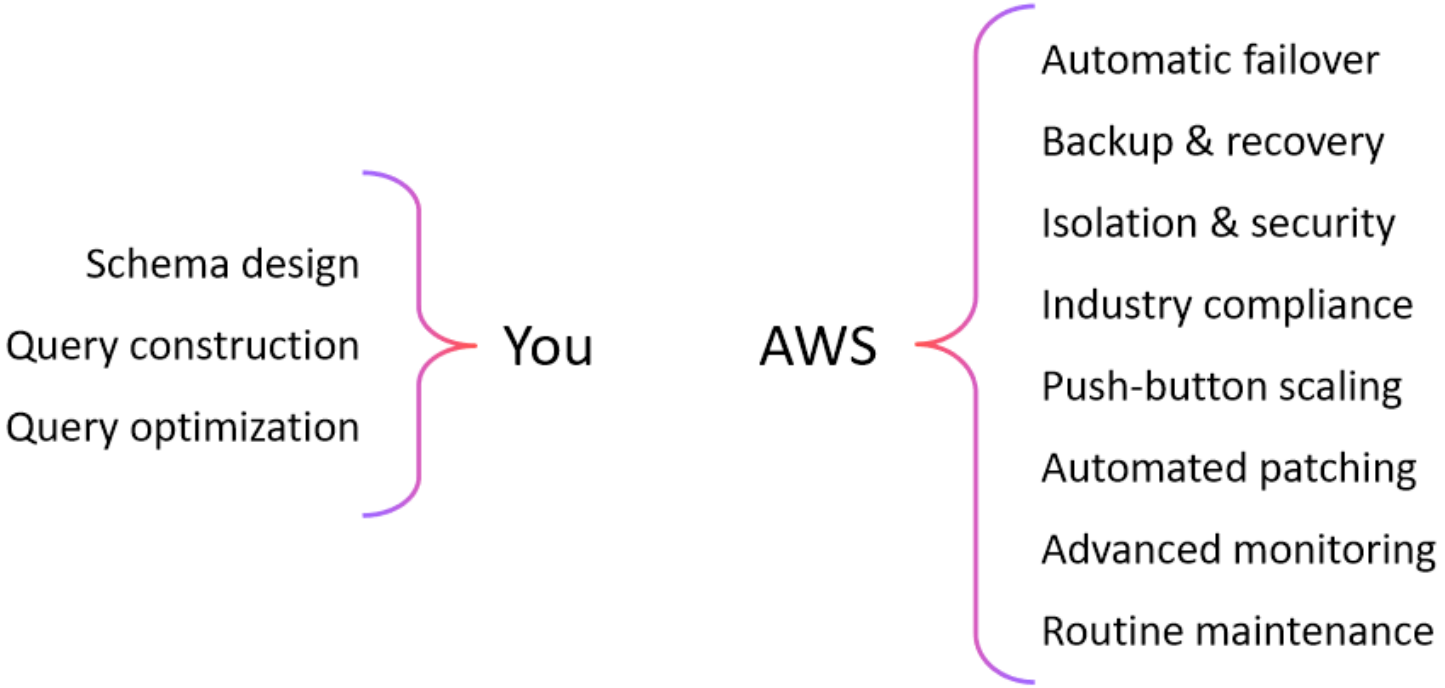
Storage volume is striped across hundreds of storage nodes distributed over three different Availability Zones

Six copies of data, two copies in each Availability Zone to protect against AZ+1 failures

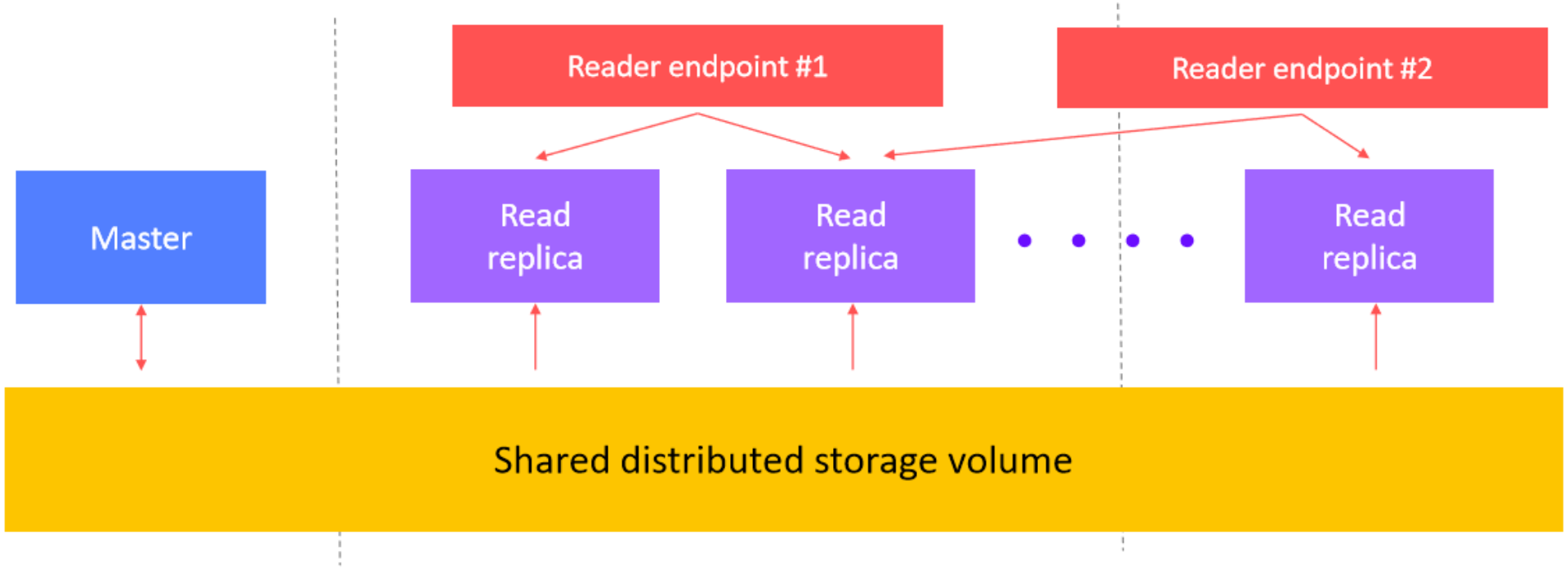


Automate administrative tasks

Takes care of your time-consuming database management tasks, freeing you to focus on your applications and business



Read replica and custom endpoint



Up to 15 promotable read replicas across multiple Availability Zones

Re-do log based replication leads to low replica lag—typically <10 ms

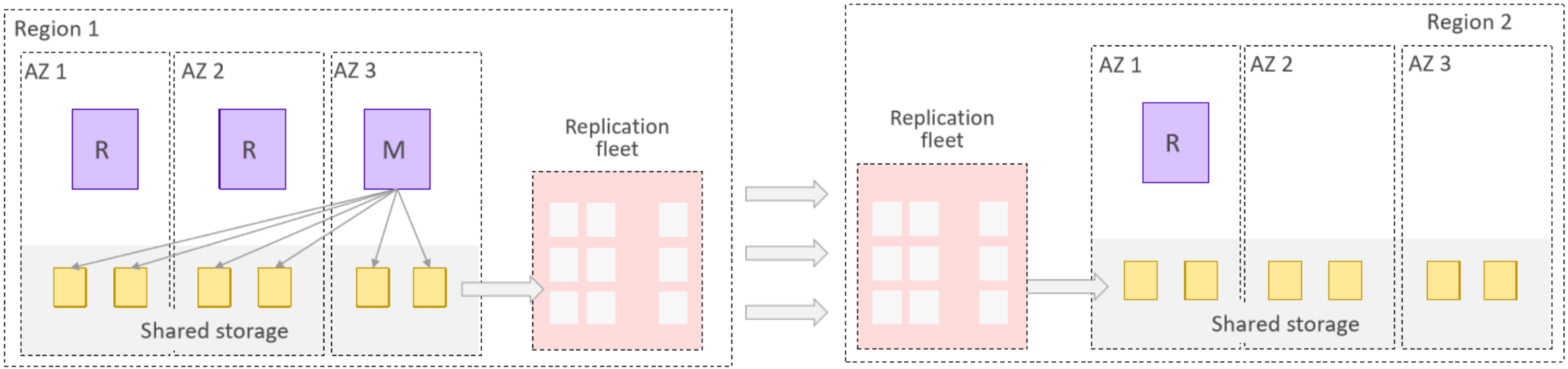
Custom reader endpoint with configurable failover order

Global replication: Logical

Faster disaster recovery and enhanced data locality

- Promote read replica to a master
  for faster recovery
  in case of a disaster
- Bring data close to your customer's
  applications
  in different regions
- Promote to a master
  for easy migration

Global replication: Physical

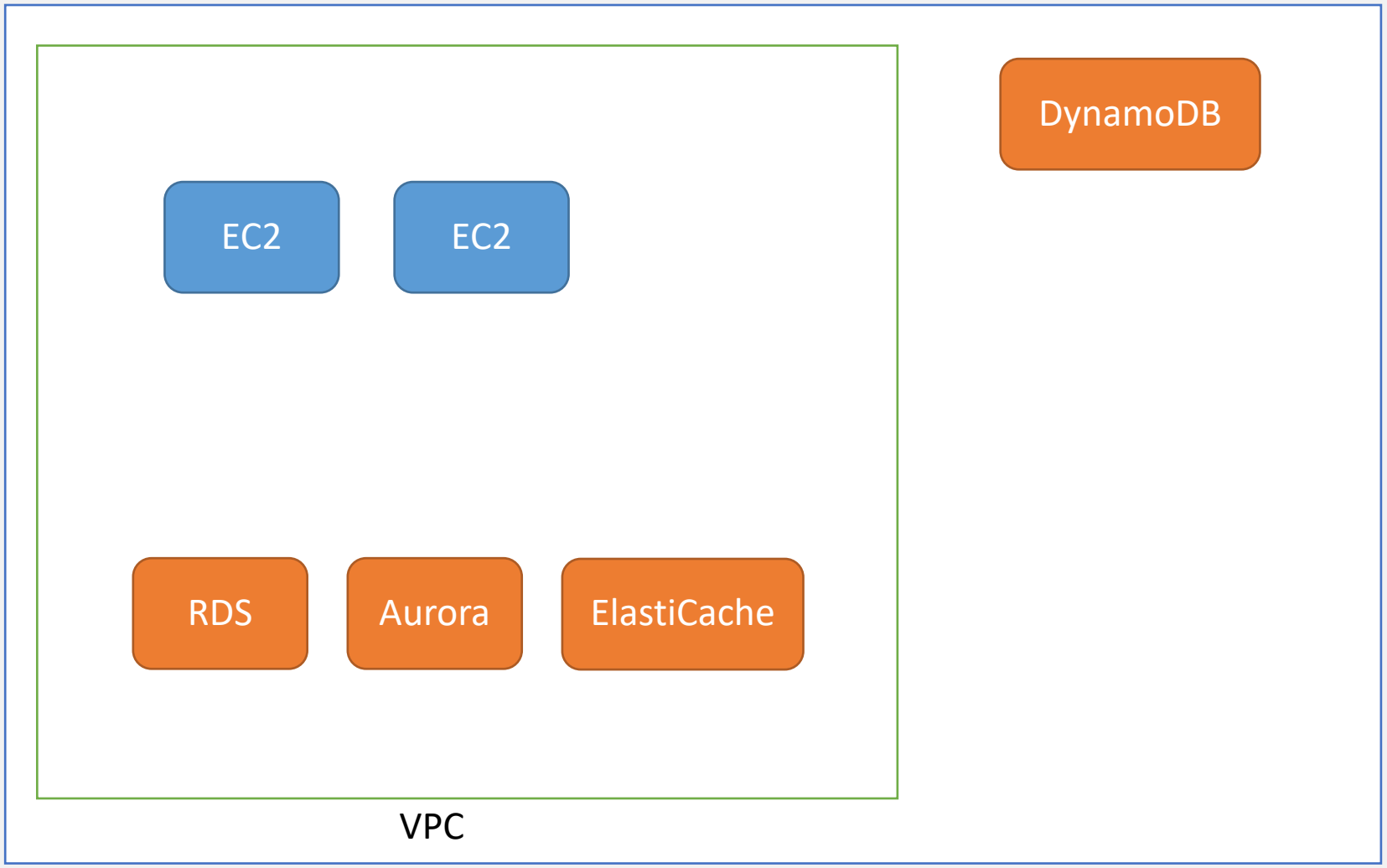


High throughput: Up to 150K writes/sec.—negligible performance impact

Low replica lag: <1 sec. cross-country replica lag under heavy load

Fast recovery: <1 min. to accept full read/write workloads after region failure

Connectivity of Databases on AWS

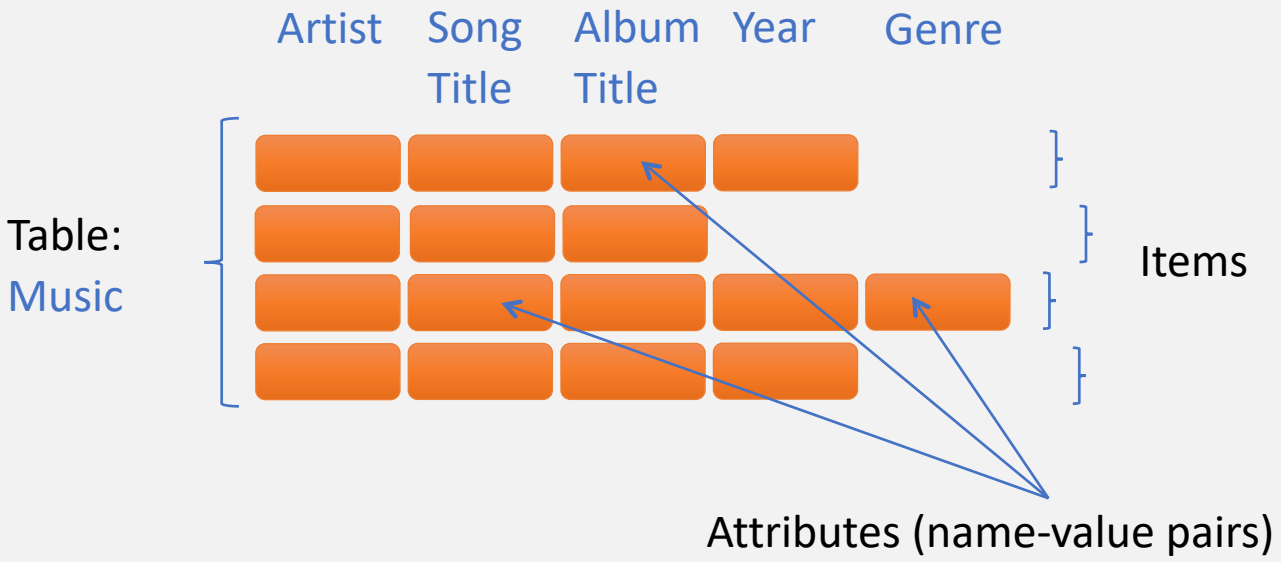


Inside and outside of VPC

DynamoDB

- Allows you to store any amount of data with **no limits**.
- Provides fast, predictable performance using **SSDs**.
- Allows you to easily provision and change the **request capacity** needed for each table.
- Is a **fully managed, NoSQL** database service.
- No throughput limits.
- **Single-digit** milliseconds latency.
- Regional resource with data stored in multiple AZs.

DynamoDB Data Model



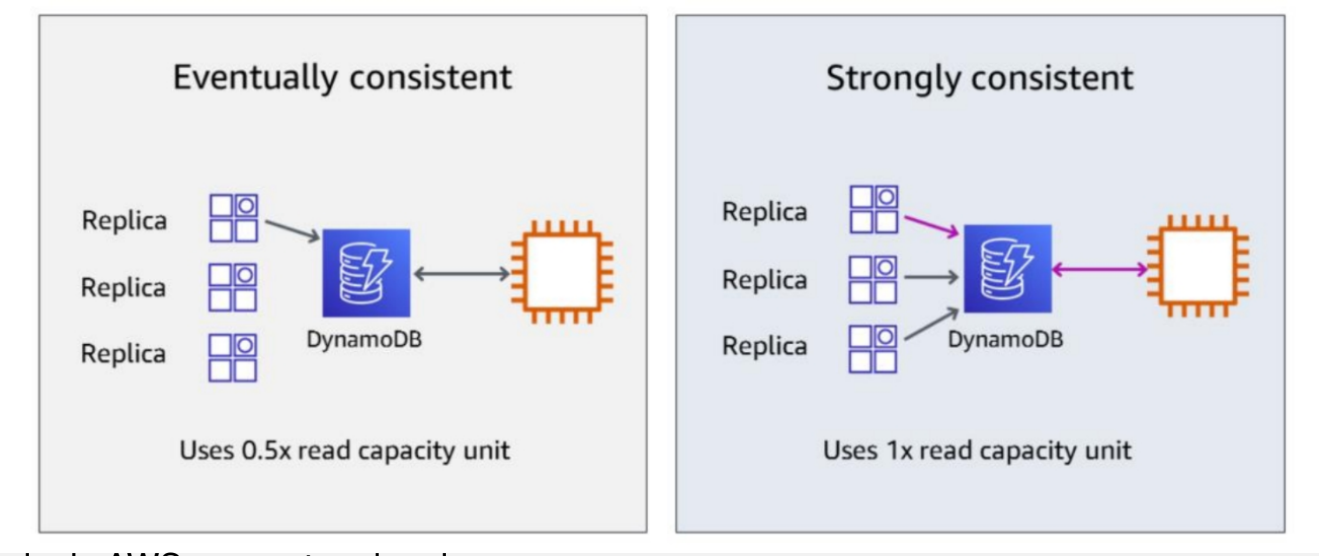
Primary Keys

Table: Music
Partition Key: Artist
Sort Key: Song Title

(DynamoDB maintains a sorted index for both keys)
You should choose a Partition Key that avoids Hot Partitions.

Consistency Options

- **Read Consistency:** strong consistency, eventual consistency, and transactional
- **Write Consistency:** standard and transactional
- **Strong Consistency**
  - The most up-to-date data
  - Must be requested explicitly
- **Eventual Consistency**
  - May or may not reflect the latest copy of data
  - Default consistency for all operations
  - 50% cheaper than strong consistency
- **Transactional Reads and Writes**
  - For ACID support across one or more tables within a single AWS account and region
  - 2x the cost of strongly consistent reads
  - 2x the cost of standard writes



DynamoDB Throughput

- **On-demand capacity mode**
  - With on-demand capacity mode, DynamoDB charges you for the data reads and writes your application performs on your tables. You do not need to specify how much read and write throughput you expect your application to perform because DynamoDB instantly accommodates your workloads as they ramp up or down.
- **Provisioned capacity mode**
  - With provisioned capacity mode, you specify the number of reads and writes per second that you expect your application to require. You can use auto scaling to automatically adjust your table’s capacity based on the specified utilization rate to ensure application performance while reducing costs.

Supported Operations

- **Query:**
  - Query a table using the partition key and an optional sort key filter.
  - If the table has a secondary index, **query the index** using its key.
  - It is the **most efficient way to retrieve items** from a table or secondary index.
- **Scan:**
  - You can scan a table or secondary index.
  - Scan reads every item – **slower than querying**.
- You can use conditional expressions in both Query and Scan operations.

DynamoDB LSI (Local Secondary Index)

- Has same partition/hash key attribute as the primary index of the table
- Has different sort/range key than the primary index of the table
- Must have a sort/range key (=composite key)
- Indexed items must be ≤ 10 GB
- Can only be created at the time of creating the table and cannot be deleted later
- Can only query single partition (specified by hash key)
- Supports eventual / strong / transactional consistency
- Consumes provisioned throughput of the base table

user_id	game_id	game_ts	result	duration
12broiu45	1234	“2018-03-15T17:43:08”	win	45
12broiu45	3456	“2018-06-20T19:02:32”	lose	33
34oiusd21	4567	“2018-02-11T-04:11:31	lose	45

Primary Key
(user_id + game_id)

LSI Examples
- user_id + game_ts
- user_id + result

DynamoDB GSI (Global Secondary Index)

- Can have same or different partition/hash key than the table's primary index
- Can have same or different sort/range key than the table's primary index
- Can omit sort/range key (=simple and composite)
- No size restrictions for indexed items

- Can be created or deleted any time
- Can query across partitions (over entire table)
- Supports **only eventual** consistency
- Has its own provisioned throughput
- Can only query projected attributes (attributes included in the index)

user_id	game_id	game_ts	result	duration
12broiu45	1234	"2020-03-15T17:43:08"	win	45
12broiu45	3456	"2020-06-20T19:02:32"	lose	33
34oiusd21	4567	"2020-02-11T-04:11:31	lose	45

Primary Key
(user_id + game_id)

GSI Examples

- user_id
- game_id
- user_id + result
- game_ts + game_id
- game_ts + duration

When to choose which index?

Local Secondary Indexes

- When application needs same partition key as the table
- When you need to avoid additional costs
- When application needs strongly consistent index reads

Global Secondary Indexes

- When application needs different or same partition key as the table
- When application needs finer throughput control
- When application only needs eventually consistent index reads

Labs

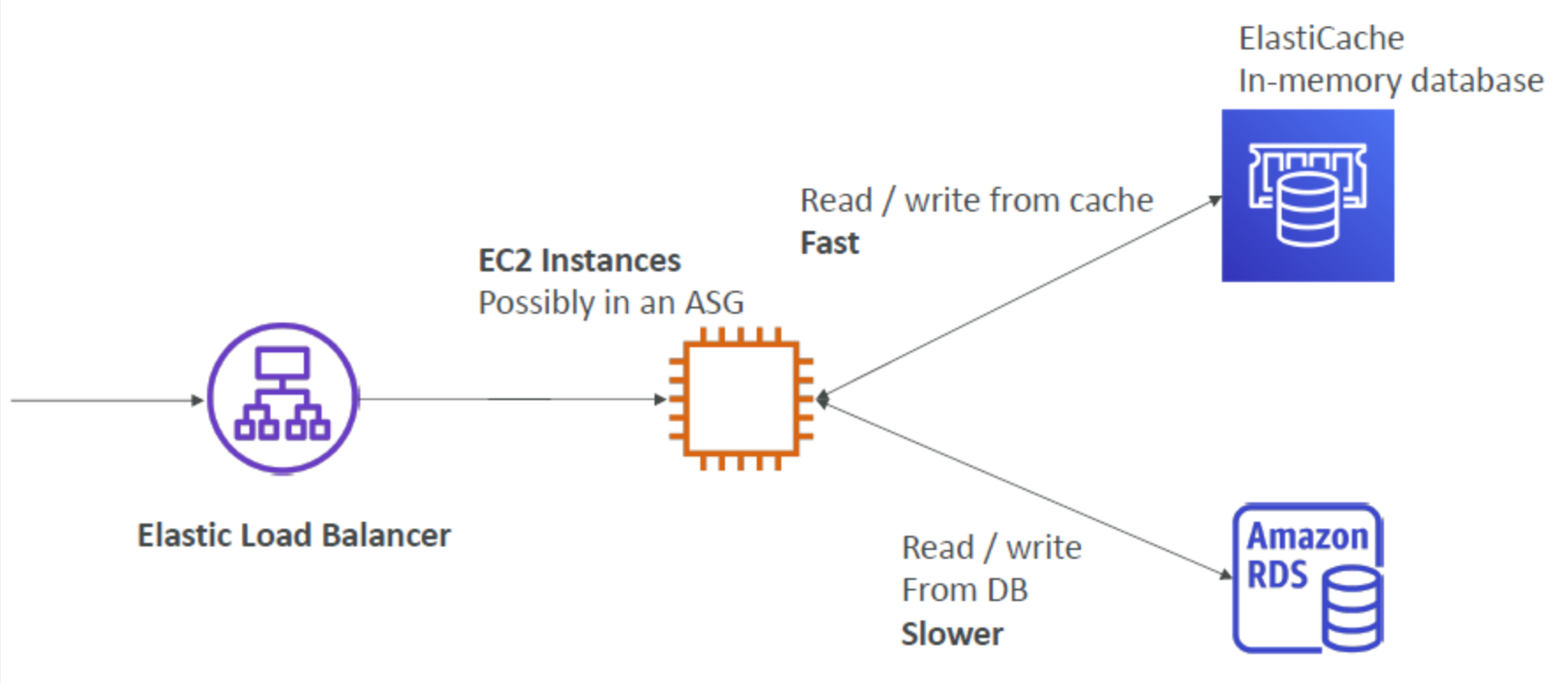
- Create an **IAM role** having permissions to work with DynamoDB.
- Allocate this role to your EC2 instance.
- Use python code from your EC2 instance to:
  - Create DynamoDB table
  - Insert data into this table
- Verify the inserted data from Management Console (DynamoDB Dashboard).

Amazon ElastiCache

- 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

Amazon ElastiCache Example



Amazon ElastiCache - Overview

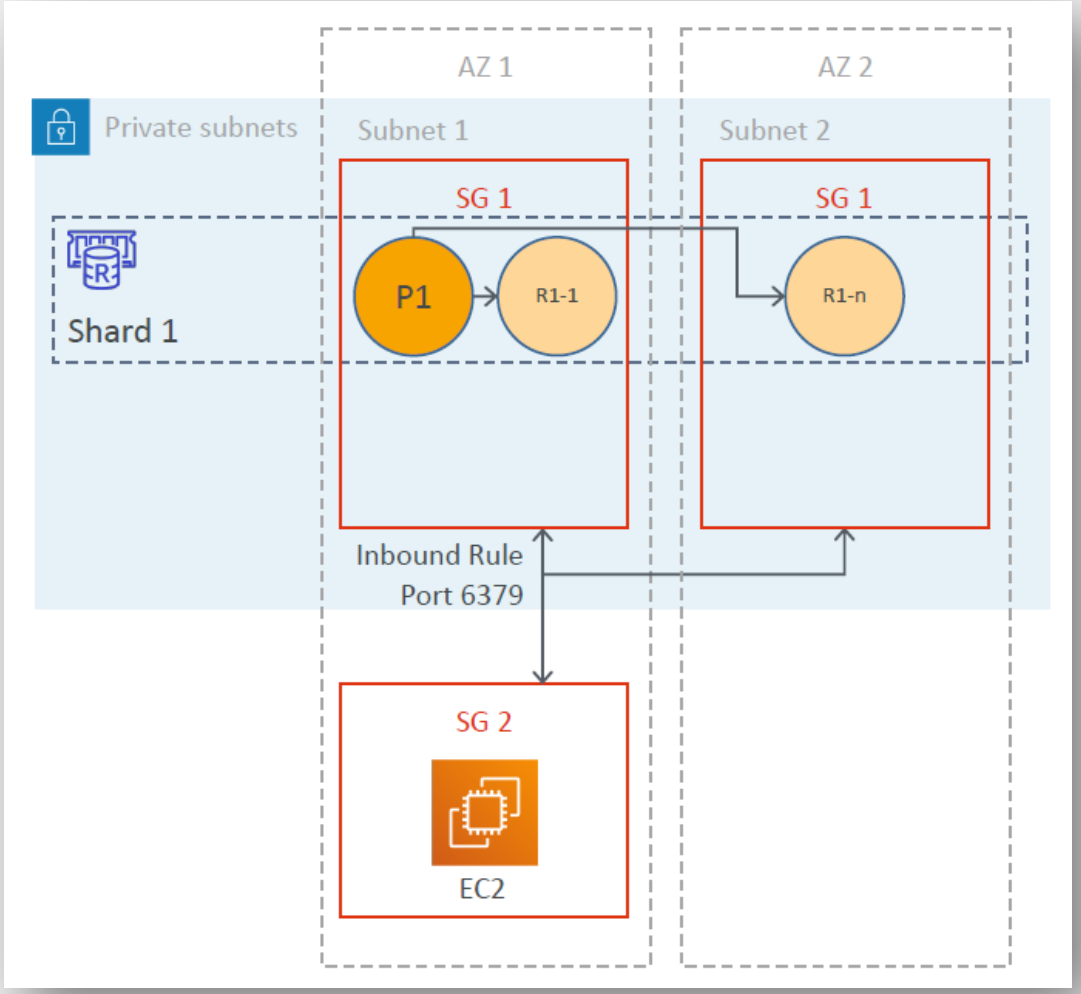
- **Fully-managed in-memory** data store (caching service, to boost DB read performance)
- It is a **remote caching service**, or a side cache i.e. separate dedicated caching instance
- Provides rapid access to data across distributed nodes
- Two flavors (both are open-source key-value stores)
  - Amazon ElastiCache for Redis
  - Amazon ElastiCache for Memcached
- **Sub-millisecond latency** for real-time applications
- Redis supports complex data types, snapshots, replication, encryption, transactions, pub/sub messaging, transactional Lua scripting, and support for geospatial data
- Multithreaded architecture support with Memcached
- Redis is suitable for complex applications including message queues, session caching, leaderboards etc.
- Memcached suitable for relatively simple applications like static website caching

Caching Strategies

- Lazy loading
- Write-through
- Adding TTL

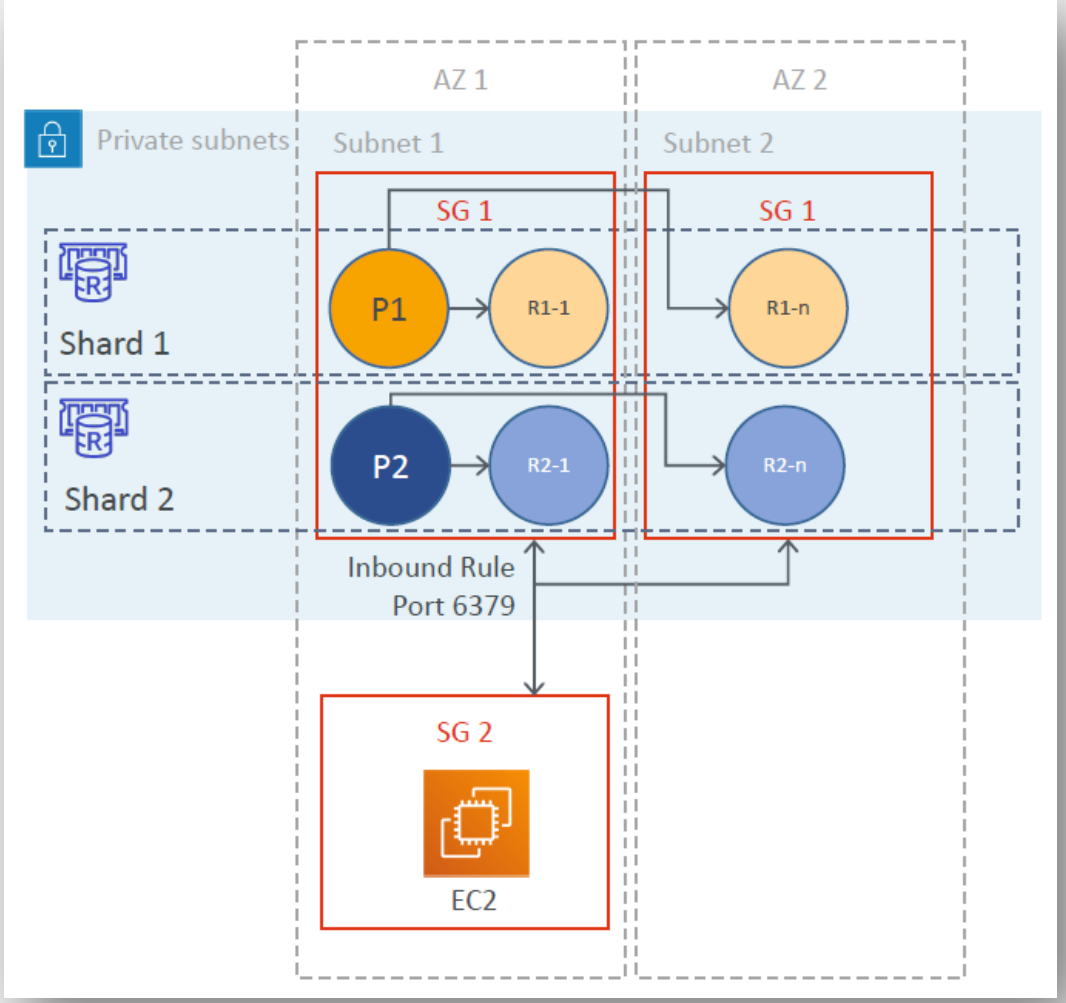
Redis – Cluster Mode Disabled

- Redis clusters are generally placed in private subnets
- Accessed from EC2 instance placed in a public (or private) subnet in a VPC
- Cluster mode disabled – single shard
- A shard has a primary node and 0-5 replicas
- A shard with replicas is also called as a replication group
- Replicas can be deployed as Multi-AZ
- Multi-AZ replicas support **Auto-Failover capability**
- Single reader endpoint (auto updates replica endpoint changes)



Redis – Cluster Mode Enabled

- Cluster mode enabled – multiple shards
- Data is **distributed across the available shards**
- A shard has a primary node and 0-5 replicas
- Multi-AZ replicas support Auto-Failover capability
- Max 90 nodes per cluster (90 shards w/no replicas to 15 shards w/ 5 replicas each)
- Minimum 3 shards recommended for HA
- Use nitro system-based node types for higher performance (e.g. M5 / R5 etc.)



Redis Replication - Comparison

Cluster Mode Disabled

- 1 shard
- 0-5 replicas
- If 0 replicas, primary failure = total data loss
- Multi-AZ supported
- Partitioning is not supported
- If primary load is read-heavy, you can scale the cluster (though up to 5 replicas max)

Cluster Mode Enabled

- Up to 90 shards
- 0-5 replicas per shard
- If 0 replicas, primary failure = total data loss in that shard
- Multi-AZ required
- Supports partitioning
- Good for write-heavy nodes (you get additional write endpoints, one per shard)

Redis Security - Auth and Access Control

Authentication into the cache

• Redis AUTH – server can authenticate the clients (requires SSL/TLS enabled)

IAM

• IAM policies can be used for AWS API level security (create cache, update cache etc.)

• ElastiCache doesn't support IAM permissions for actions within ElastiCache (which clients can access what)

Enabling Redis AUTH

```
aws elasticache
modify-replication-group
--replication-group-id redidgroup
--auth-token This-is-the-password
--auth-token-update-strategy ROTATE
--apply-immediately n
```

Connecting with Redis AUTH

```
src/redis-cli -h <endpoint> -p 6379
-a This-is-the-password
```

Redis or Memcached

Redis

- Sub-millisecond latency
- Supports complex data types (sorted sets, hashes, bitmaps, hyperlog, geospatial index)
- Multi AZ with Auto-Failover, supports sharding
- Read Replicas for scalability and HA
- Data Durability using AOF persistence
- Backup and restore features

Memcached

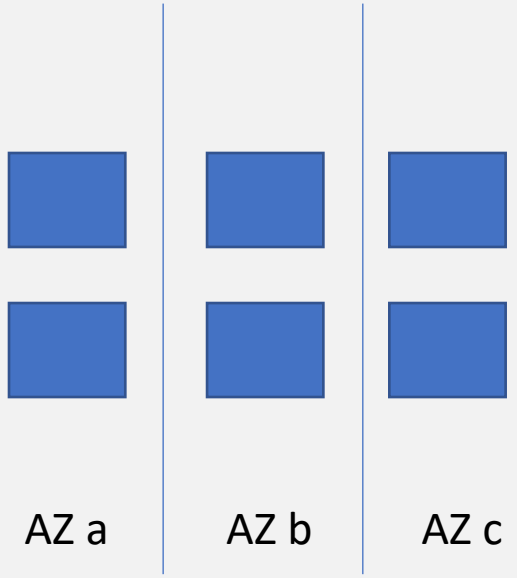
- Sub-millisecond latency
- Support only simple data types (string, objects)
- Multi-node for sharding
- Non persistent
- No backup and restore
- Multi-threaded architecture

Labs

- Create required security group for ElastiCache cluster.
- Create a subnet group for ElastiCache cluster (with private subnets).
- Create a Multi-AZ ElastiCache Redis cluster in the private subnets of your VPC.
  - Choose cache.t2.micro
  - Multi-AZ
  - Number of Replicas: 1
- Check its connectivity using Python from the EC2 instance in the same VPC.

Storage Services

- Overview of S3, EBS, EFS
- Pricing of S3 (different storage classes), EBS, EFS
- Right use-cases for S3, EBS, EFS



Storage Services

S3

- Gives 11 9s of reliability
- Uses global namespace
- S3 bucket is a regional resource.
- You interact via AWS APIs
- WORM usage

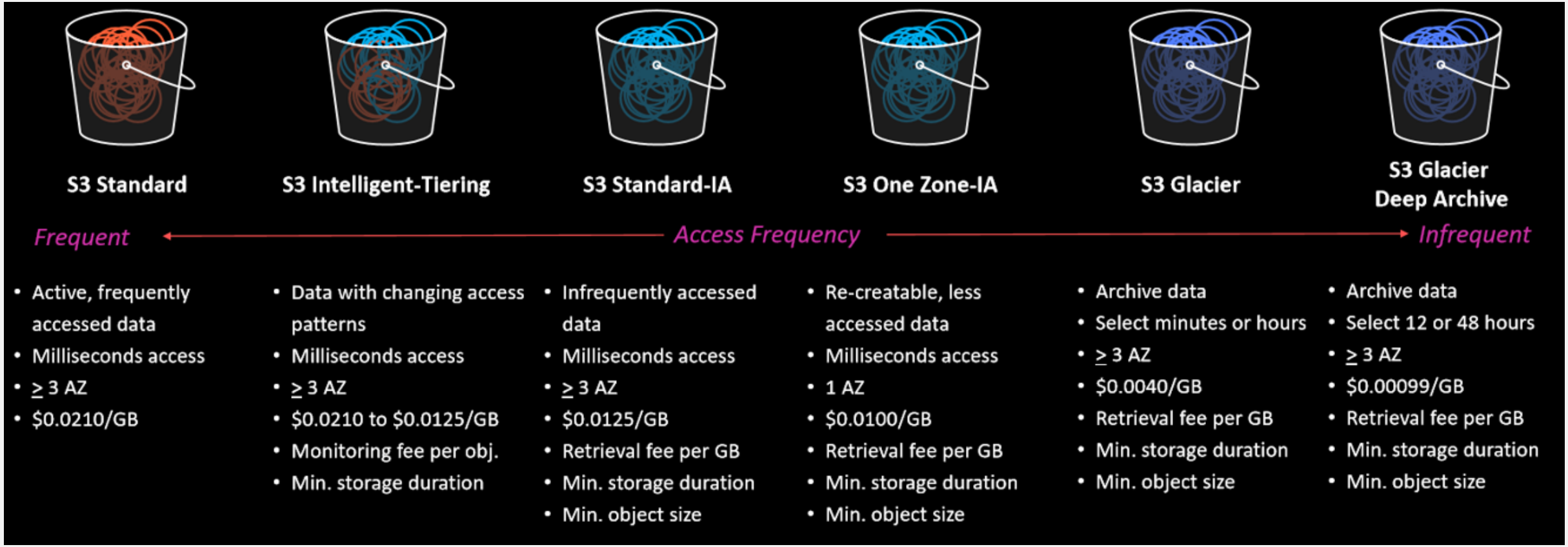
EBS

- Block based storage (disk)
- Used with EC2
- Read / write via OS level operation
- It is located in an AZ
- OS & DB usage

EFS

- It can be mounted to multiple EC2 instances
- Spans across AZs in a region
- Read/write possible from multiple instances

S3 Storage Classes



Labs

- Activate **CloudShell**
- Execute “aws configure” & “aws sts get-caller-identity”
- Create new buckets and execute the following commands using AWS CLI https://docs.aws.amazon.com/cli/latest/reference/s3/
- Delete all the S3 buckets at the end.

Route53

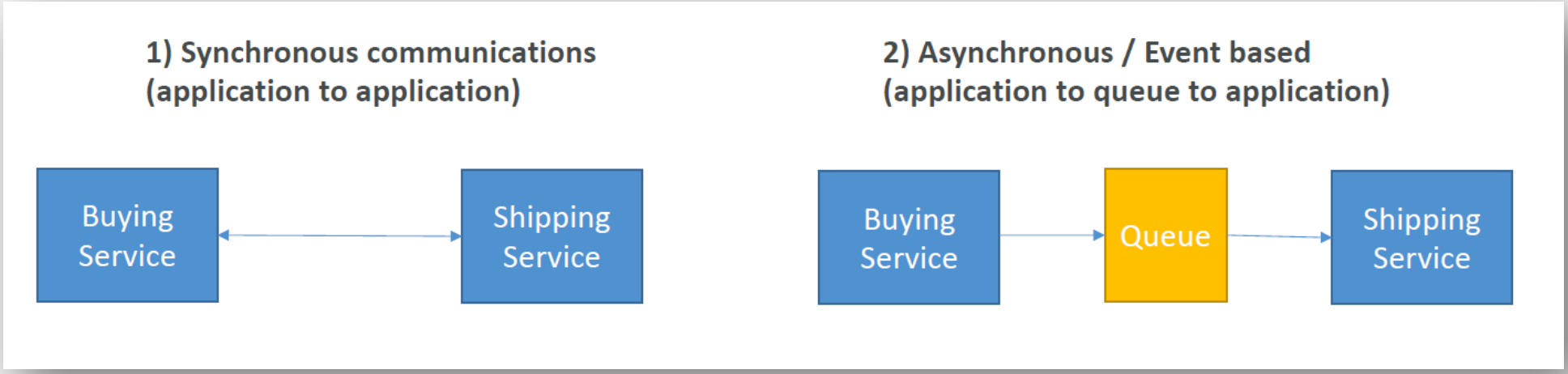
- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through URLs.
- In AWS, the most common records are:
  - A: hostname to IPv4
  - AAAA: hostname to IPv6
  - CNAME: hostname to hostname
  - Alias: hostname to AWS resource.

Route53 Routing Policies

- Simple
- Weighted
- Latency
- Failover
- Geolocation

Decoupled Systems

- When we start deploying multiple applications, they will inevitably need to communicate with one another
- There are two patterns of application communication



Simple Queue Service (SQS)

- Fully managed service, used to decouple applications
- Unlimited throughput, unlimited number of messages in queue
- Default retention of messages: 4 days, maximum of 14 days
- Low latency (<10 ms on publish and receive)
- Limitation of 256KB per message sent
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)

SQS – Producing Messages

- Produced to SQS using the SDK (**SendMessage** API)
- The message is persisted in SQS until a consumer deletes it
- Message retention: default 4 days, up to 14 days

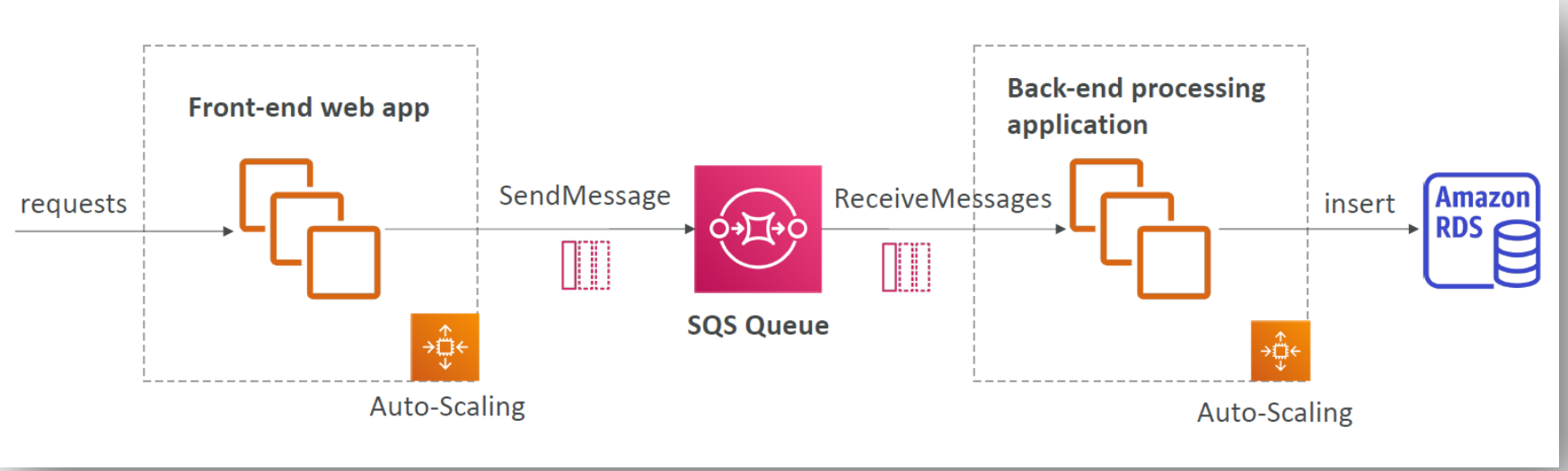
Example: send an order to be processed

- Order id
- Customer id
- Any attributes you want

SQS standard: unlimited throughput

SQS – Consuming Messages

- Consumers (running on EC2 instances, servers, or AWS Lambda)…
- Poll SQS for messages (receive up to 10 messages at a time)
- Process the messages (example: insert the message into an RDS database)
- Delete the messages using the **DeleteMessage** API



SQS – Security

Encryption:

- In-flight encryption using HTTPS API
- At-rest encryption using KMS keys
- Client-side encryption if the client wants to perform encryption/decryption itself

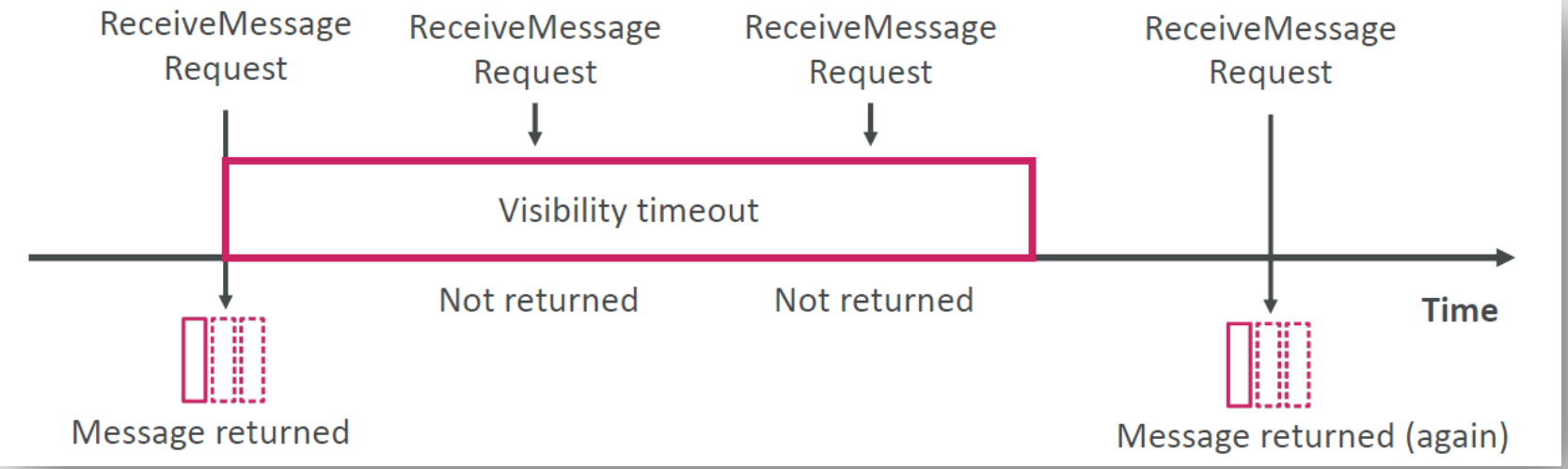
Access Controls: IAM policies to regulate access to the SQS API

SQS Access Policies (similar to S3 bucket policies)

- Useful for cross-account access to SQS queues
- Useful for allowing other services (SNS, S3, etc.) to write to an SQS queue

SQS – Message Visibility Timeout

- After a message is polled by a consumer, it becomes **invisible** to other consumers
- By default, the “message visibility timeout” is **30 seconds**
- That means the message has 30 seconds to be processed
- After the message visibility timeout is over, the message is “**visible**” in SQS
- If a message is not processed within the visibility timeout, it will be processed twice
- A consumer could call the **ChangeMessageVisibility** API to get more time



SQS – Dead Letter Queue

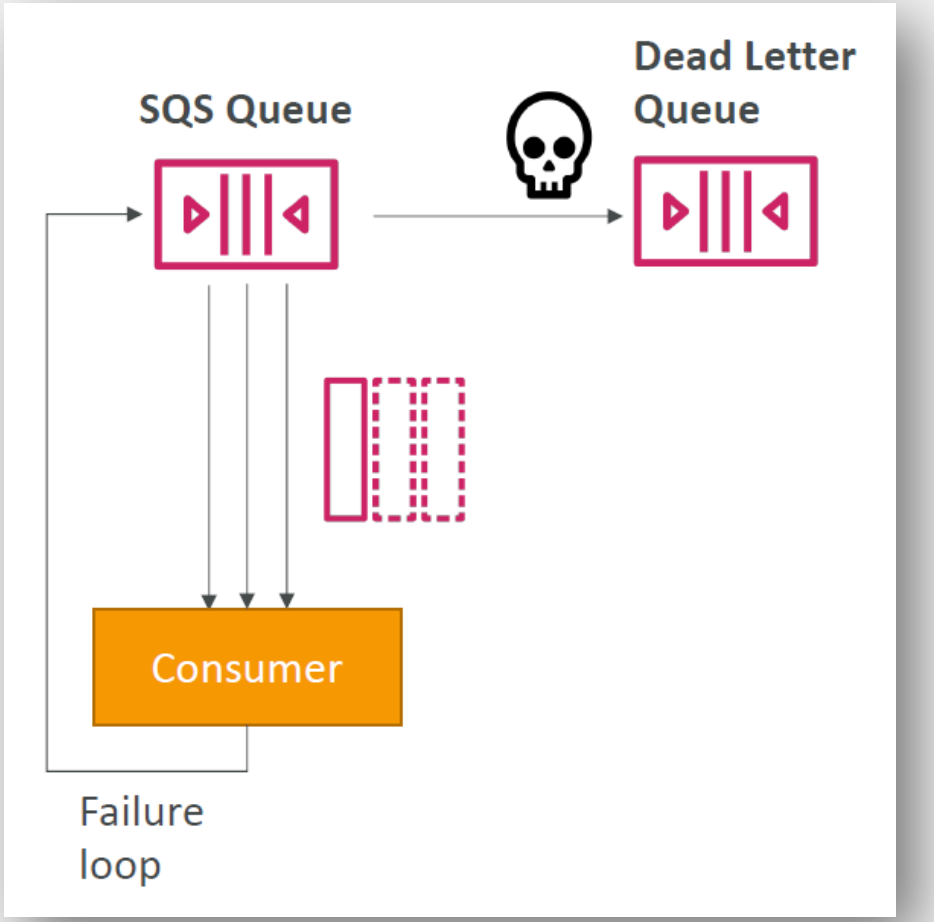
• If a consumer fails to process a message within the Visibility Timeout, the message goes back to the queue!

• We can set a threshold of how many times a message can go back to the queue

• After the **MaximumReceives** threshold is exceeded, the message goes into a dead letter queue (DLQ)

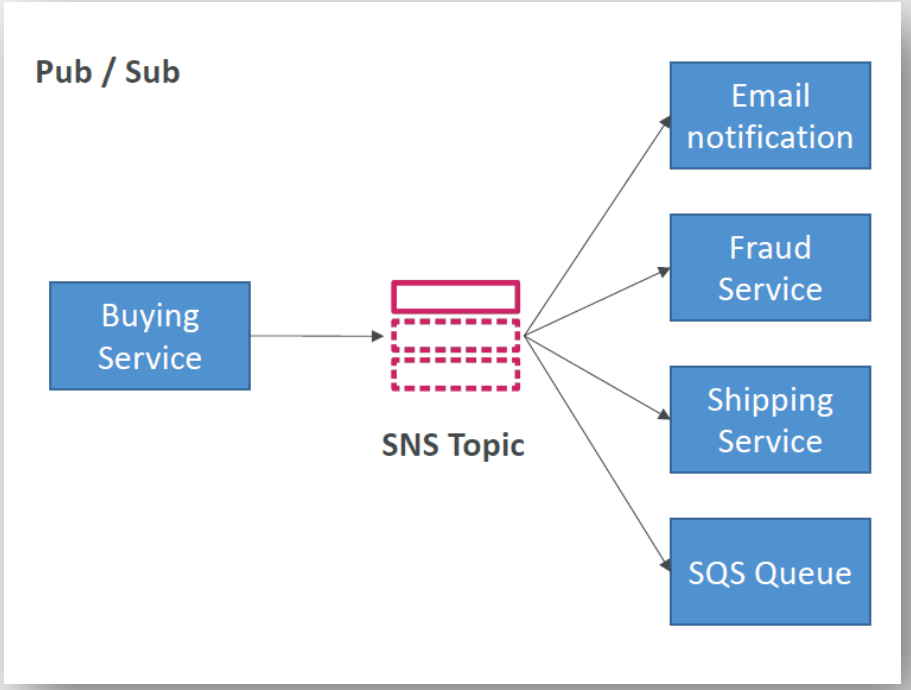
• Useful for debugging!

• Make sure to process the messages in the DLQ before they expire. Good to set a retention of 14 days in the DLQ



Simple Notification Service (SNS)

- The “event producer” only sends message to one SNS topic
- As many “event receivers” (subscriptions) as we want to listen to the SNS topic notifications
- Each subscriber to the topic will get all the messages
- Up to 10,000,000 subscriptions per topic
- 100,000 topics limit
- Subscribers can be:
  - SQS
  - HTTP / HTTPS
  - Lambda
  - Emails
  - SMS messages
  - Mobile Notifications



SNS Integration (examples)

- Many AWS services can send data directly to SNS for notifications
- CloudWatch (for alarms)
- Auto Scaling Groups notifications
- Amazon S3 (on bucket events)
- CloudFormation (upon state changes => failed to build, etc.)

SNS – How to publish

- Topic Publish (using the SDK)
  - Create a topic
  - Create a subscription (or many)
  - Publish to the topic
- Direct Publish (for mobile apps SDK)
  - Create a platform application
  - Create a platform endpoint
  - Publish to the platform endpoint

SNS - Security

Encryption:

- In-flight encryption using HTTPS API
- At-rest encryption using KMS keys
- Client-side encryption if the client wants to perform encryption/decryption itself

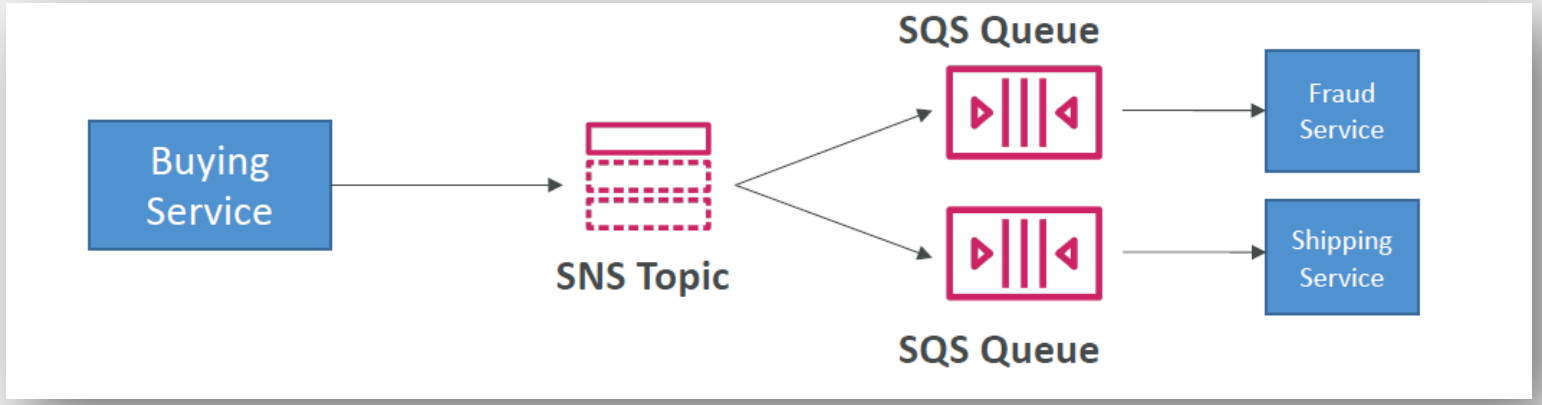
Access Controls: IAM policies to regulate access to the SNS API

SNS Access Policies (similar to S3 bucket policies)

- Useful for cross-account access to SNS topics
- Useful for allowing other services (S3 etc.) to write to an SNS topic

SNS + SQS (Fan Out)

- Push once in SNS, receive in all SQS queues that are subscribers
- Fully decoupled, no data loss
- SQS allows for: data persistence, delayed processing and retries of work
- Ability to add more SQS subscribers over time
- Make sure your SQS queue access policy allows for SNS to write



Monitoring

- CloudWatch Metrics
- CloudWatch Events
- CloudWatch Logs
- Use of CloudWatch with other AWS services and on-premises

AWS CloudWatch Metrics

- CloudWatch provides metrics for every services in AWS
- Metric is a variable to monitor (CPUUtilization, NetworkIn…)
- Metrics belong to namespaces
- Dimension is an attribute of a metric (instance id, environment, etc.)
- Up to 10 dimensions per metric
- Metrics have timestamps
- Can create CloudWatch dashboards of metrics
- Integrated with most of the AWS services
- EC2 Standard & Detailed Monitoring
- Standard & Custom Metrics

CloudWatch Dashboards

- Great way to setup custom dashboards for quick access to key metrics and alarms
- **Dashboards are global**
- **Dashboards can include graphs from different AWS accounts and regions**
- You can change the time zone & time range of the dashboards
- You can setup automatic refresh (10s, 1m, 2m, 5m, 15m)
- Dashboards can be shared with people who don't have an AWS account (public, email address, 3rd party SSO provider through Amazon Cognito)

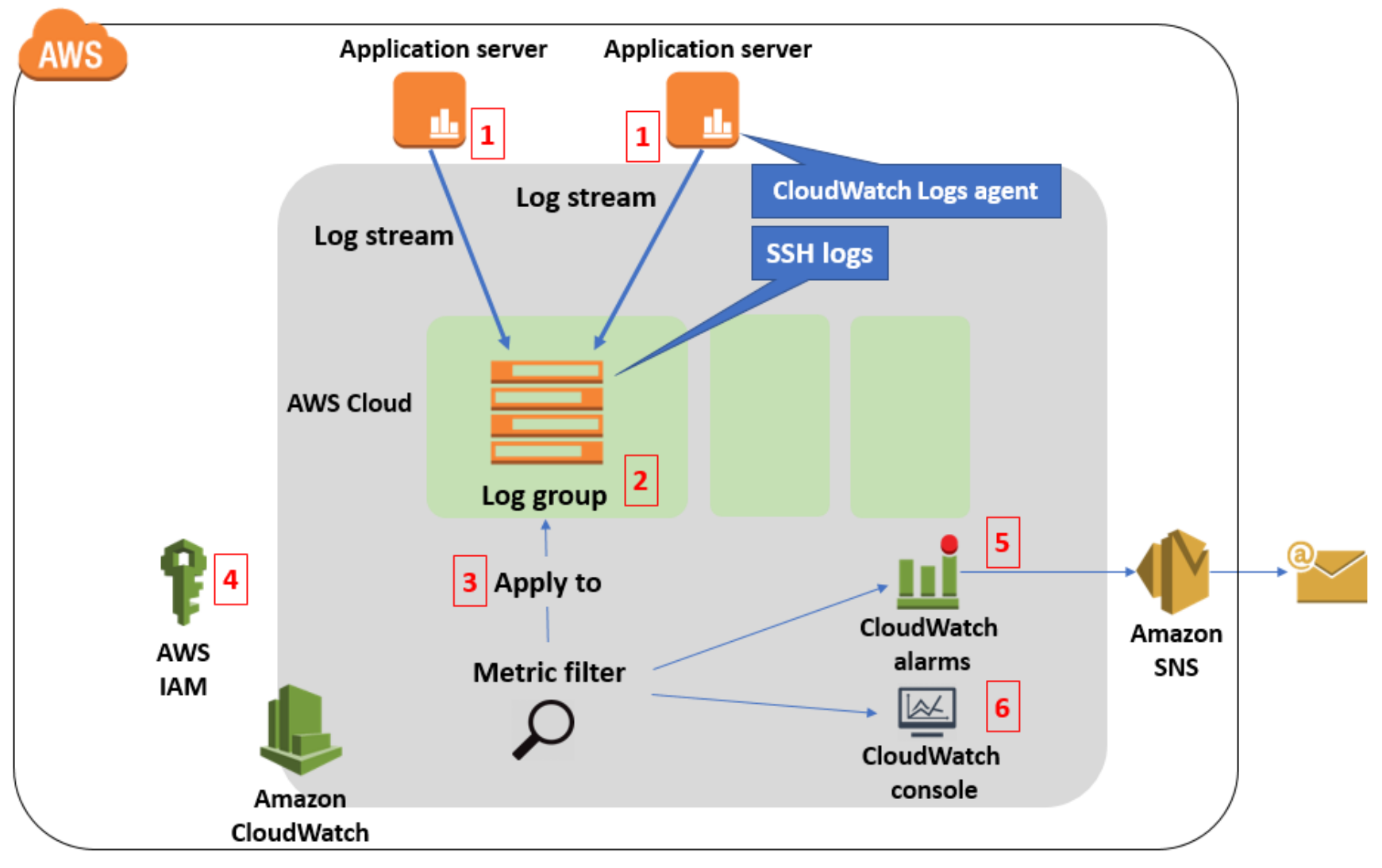
Pricing:

- 3 dashboards (up to 50 metrics) for free
- $3/dashboard/month afterwards

CloudWatch Logs

- Applications can send logs to CloudWatch using the AWS SDK
- CloudWatch **can collect log from**:
  - Elastic Beanstalk: collection of logs from application
  - ECS: collection from containers
  - AWS Lambda: collection from function logs
  - VPC Flow Logs: VPC specific logs
  - API Gateway
  - CloudTrail based on filter
  - CloudWatch log agents: for example on EC2 machines
  - Route53: Log DNS queries
- CloudWatch Logs **can be sent to**:
  - Batch exporter to S3 for archival
  - Stream to ElasticSearch cluster for further analytics

CloudWatch Logs



CloudWatch Logs Metric Filter

CloudWatch Logs can use filter expressions

- For example, find a specific IP inside of a log
- Metric filters can be used to trigger alarms

CloudWatch Unified Agent

- Collect additional system-level metrics such as RAM, processes, etc.
- Collect logs to send to CloudWatch Logs

CloudWatch Events

Event Pattern: Intercept events from AWS services (Sources)

- Example sources: EC2 Instance Start, CodeBuild Failure, S3, Trusted Advisor
- Can intercept any API call with CloudTrail integration

Schedule or Cron (example: create an event every 4 hours)

A JSON payload is created from the event and passed to a target

- Compute: Lambda, Batch, ECS task
- Integration: SQS, SNS, Kinesis Data Streams, Kinesis Data Firehose
- Orchestration: Step Functions, CodePipeline, CodeBuild
- Maintenance: SSM, EC2 Actions

SERVERLESS

Serverless approach

- Serverless is a new paradigm in which the developers don't have to manage servers anymore
- They just deploy code in the form of functions
- Serverless **does not mean there are no servers**, it means you just don't manage / provision / see them

- Serverless was made popular by AWS Lambda but now it also includes anything that's managed: "databases, messaging, storage, etc.". E.g. DynamoDB, S3, etc.

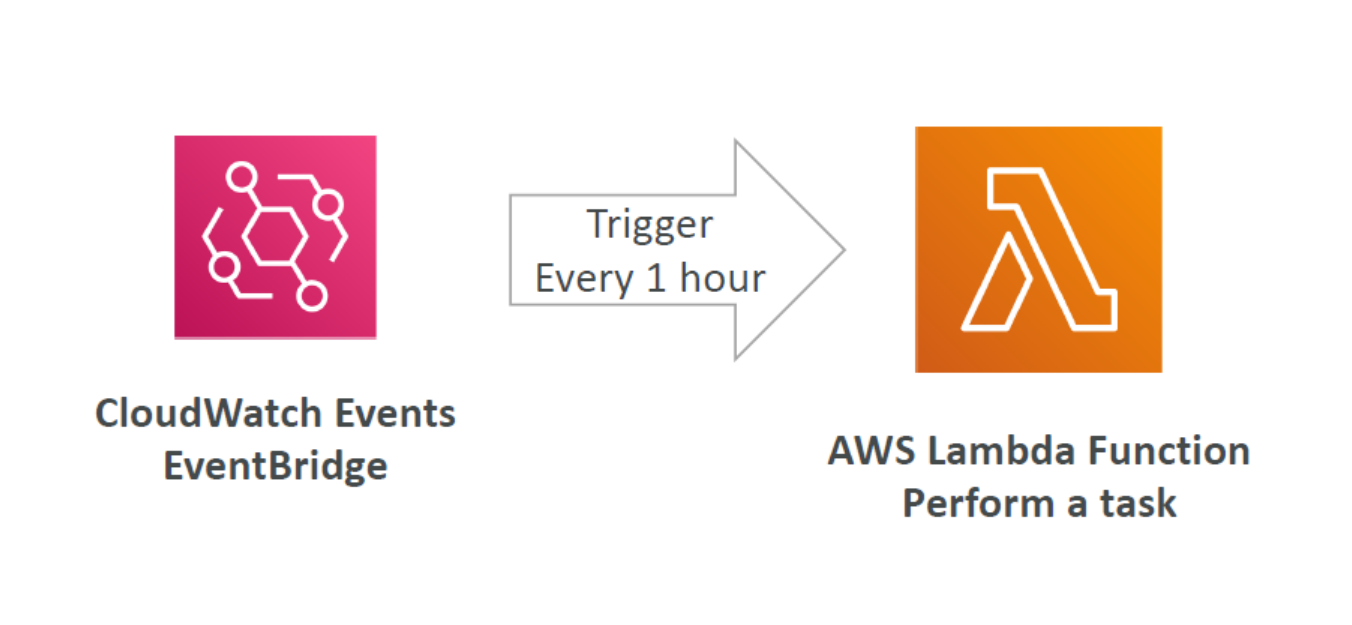
AWS Lambda Overview

- No servers to maintain - serverless
- Supports code in many languages - C#, Java, Python, node.js, Go, etc.
- Offers many levels of resources:
  - 128 MB of memory and the lowest CPU power, to
  - 10 GB of memory and the highest CPU power
- A function can run up to 15 minutes when invoked
- Pricing depends on
  - resource level chosen
  - time of execution (rounded off to nearest 1ms)
  - number of invocation requests
- Saves money as compute is used **only when required** (no resources are blocked/running all the time)
- Supports IAM role & VPC connectivity

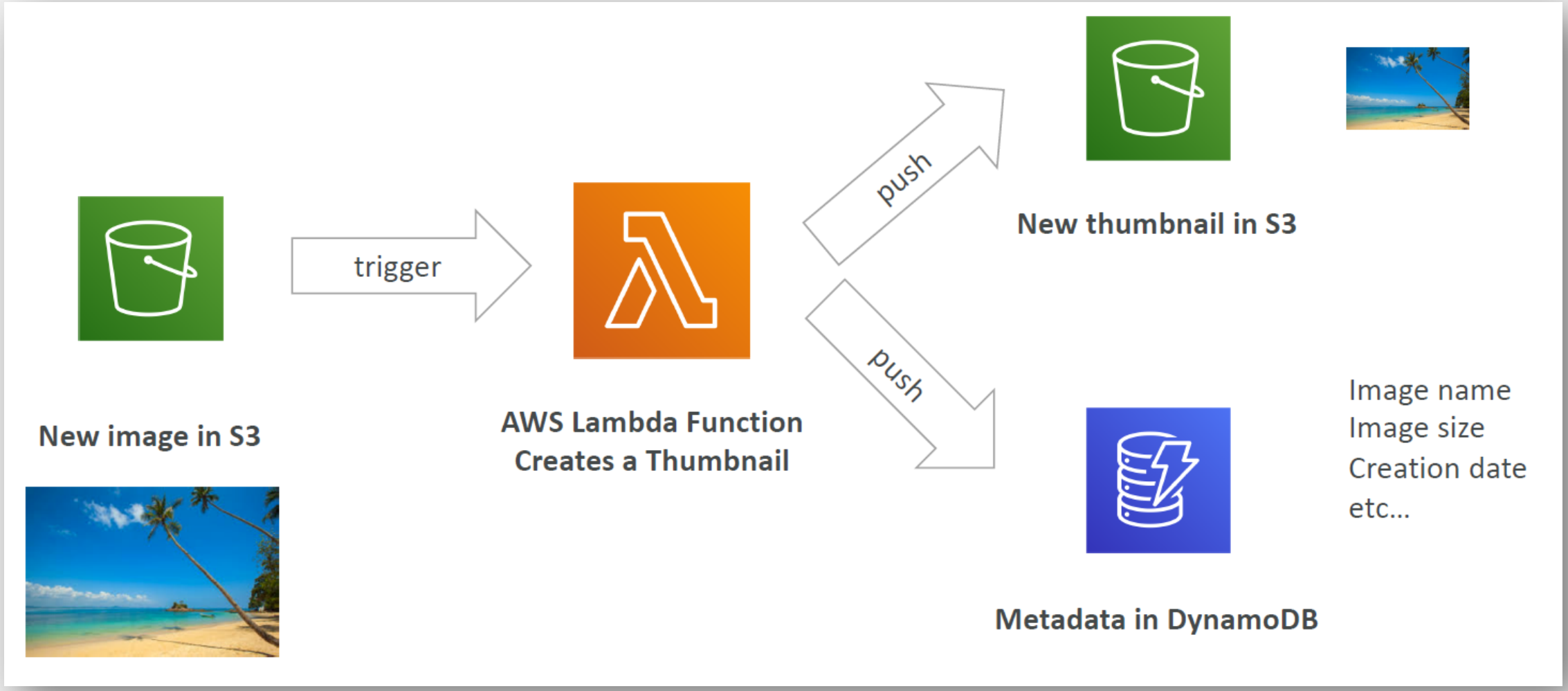
AWS Lambda Overview

- Lambda **could be triggered** as result of events happening from **many AWS services** like S3, DynamoDB, CloudWatch, SNS, etc.
- You can test Lambda in Free Tier account, you get enough credits
- When should you consider Lambda instead of EC2:
  - For stand-alone (stateless) code execution, which gets executed and then stores the result somewhere (Database, S3)
  - When you don’t want to maintain servers (OS update, security, scalability etc.)

Example: Serverless CRON Job



Example: Serverless Thumbnail creation



AWS Lambda Limits

Execution:

- Memory allocation: 128 MB – 10GB (64 MB increments)
- Maximum execution time: 900 seconds (15 minutes)
- Environment variables (4 KB)
- Disk capacity in the “function container” (in /tmp): 512 MB
- Concurrency executions: 1000 (can be increased)

Deployment:

- Lambda function deployment size (compressed .zip): 50 MB
- Size of uncompressed deployment (code + dependencies): 250 MB
- Can use the /tmp directory to load other files at startup
- Size of environment variables: 4 KB

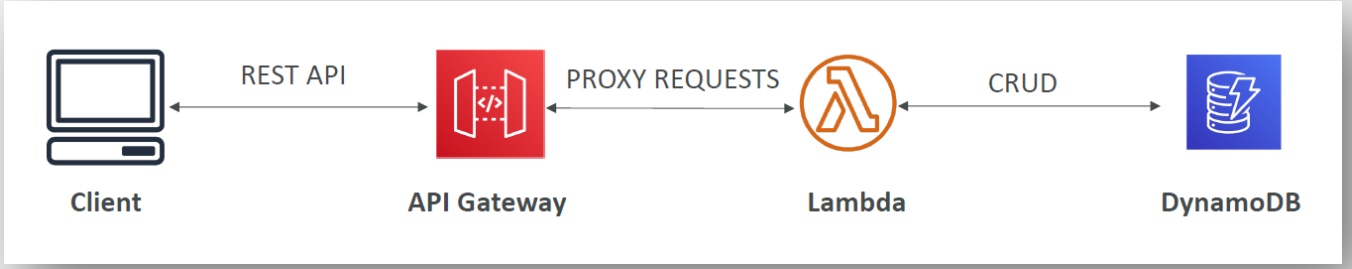
AWS Lambda Layers

- A Lambda layer is a **.zip file archive that can contain additional code or data**. A layer can contain libraries, a custom runtime, data, or configuration files.
- Layers promote **code sharing and separation of responsibilities** so that you can iterate faster on writing business logic.
- When you create a layer, you must bundle all its content into **a .zip file archive**. You upload the .zip file archive to your layer from Amazon Simple Storage Service (Amazon S3) or your local machine. Lambda **extracts the layer contents into the /opt directory** when setting up the execution environment for the function.

  - Layers are private to your account by default but you can optionally share with other AWS accounts or make a layer public.
  - You can include up to five layers per function, which count towards the standard Lambda deployment size limits.

API Gateway – Building Serverless API

- AWS Lambda + API Gateway: No infrastructure to manage
- Cache API responses
- Support for the WebSocket Protocol
- Handles API versioning (v1, v2…)
- Handles different environments (dev, test, prod…)
- Handles security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Integration with CloudWatch Metrics & CloudWatch Logs



API Gateway – Integrations High Level

Lambda Function

- Invoke Lambda function
- Easy way to expose REST API backed by AWS Lambda

HTTP

- Expose HTTP endpoints in the backend
- Example: internal HTTP API on premise, Application Load Balancer
- Why? Add rate limiting, caching, user authentications, API keys, etc.

API Gateway - Endpoint Types

Edge-Optimized (default): For global clients

- Requests are routed through the CloudFront Edge locations (improves latency)
- The API Gateway still lives in only one region

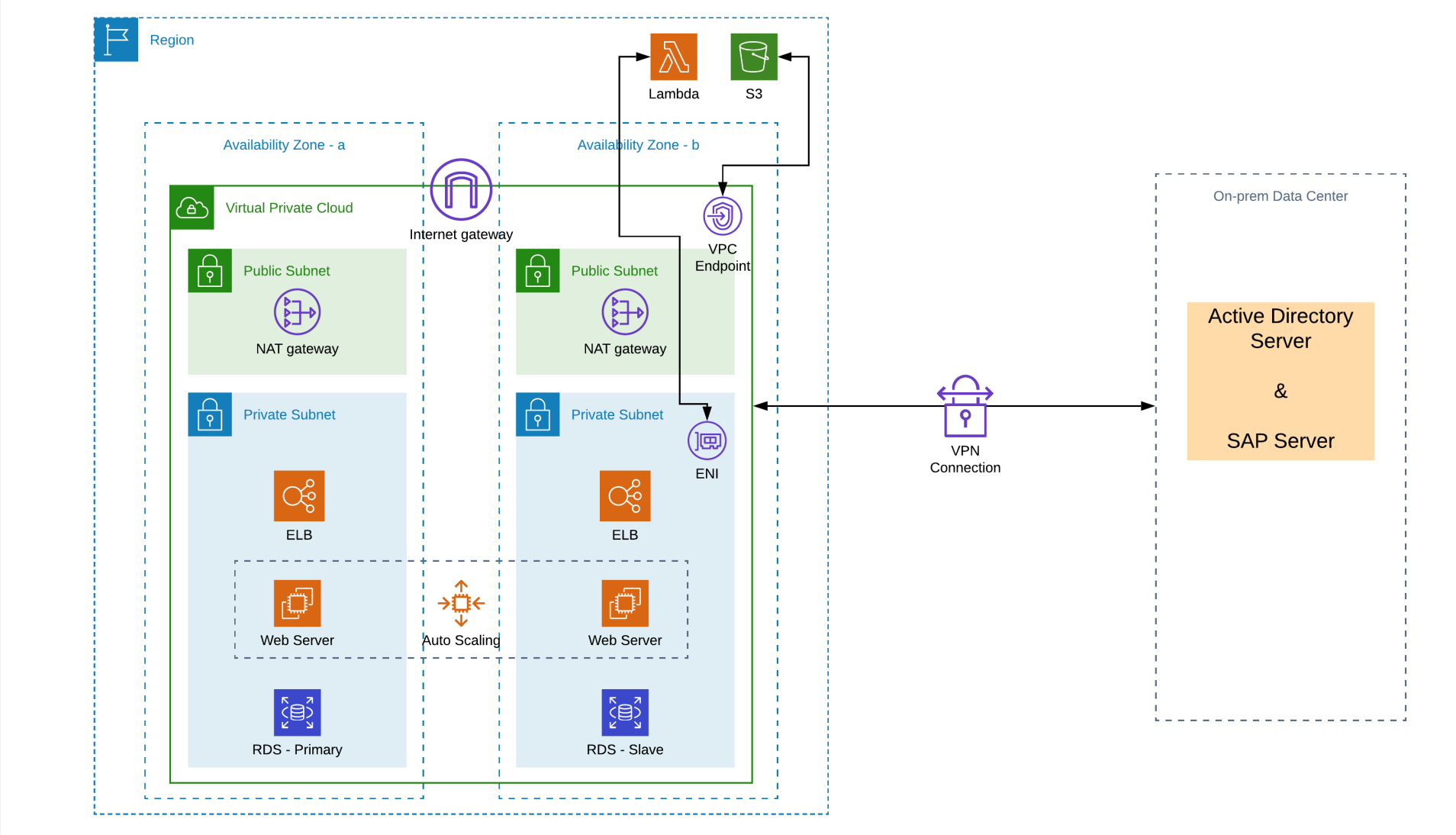
Regional:

- For clients within the same region
- Could manually combine with CloudFront (more control over the caching strategies and the distribution)

Private:

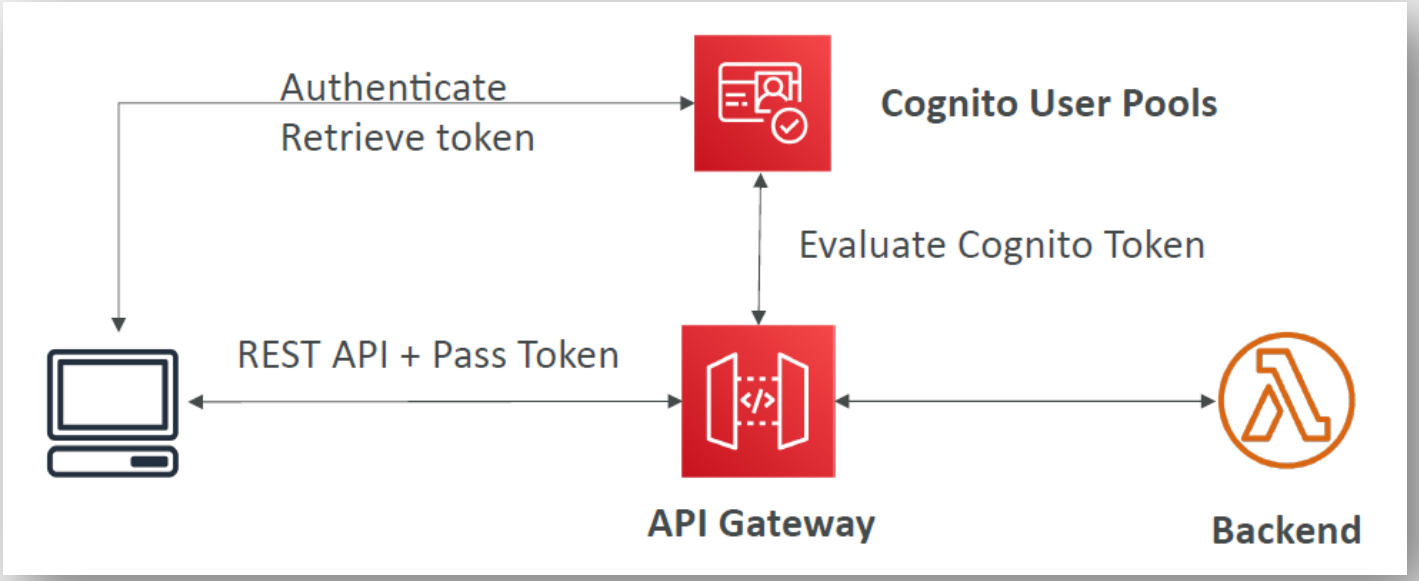
- Can only be accessed from your VPC using an interface VPC endpoint (ENI)
- Use a resource policy to define access

VPC in action – A Production Setup



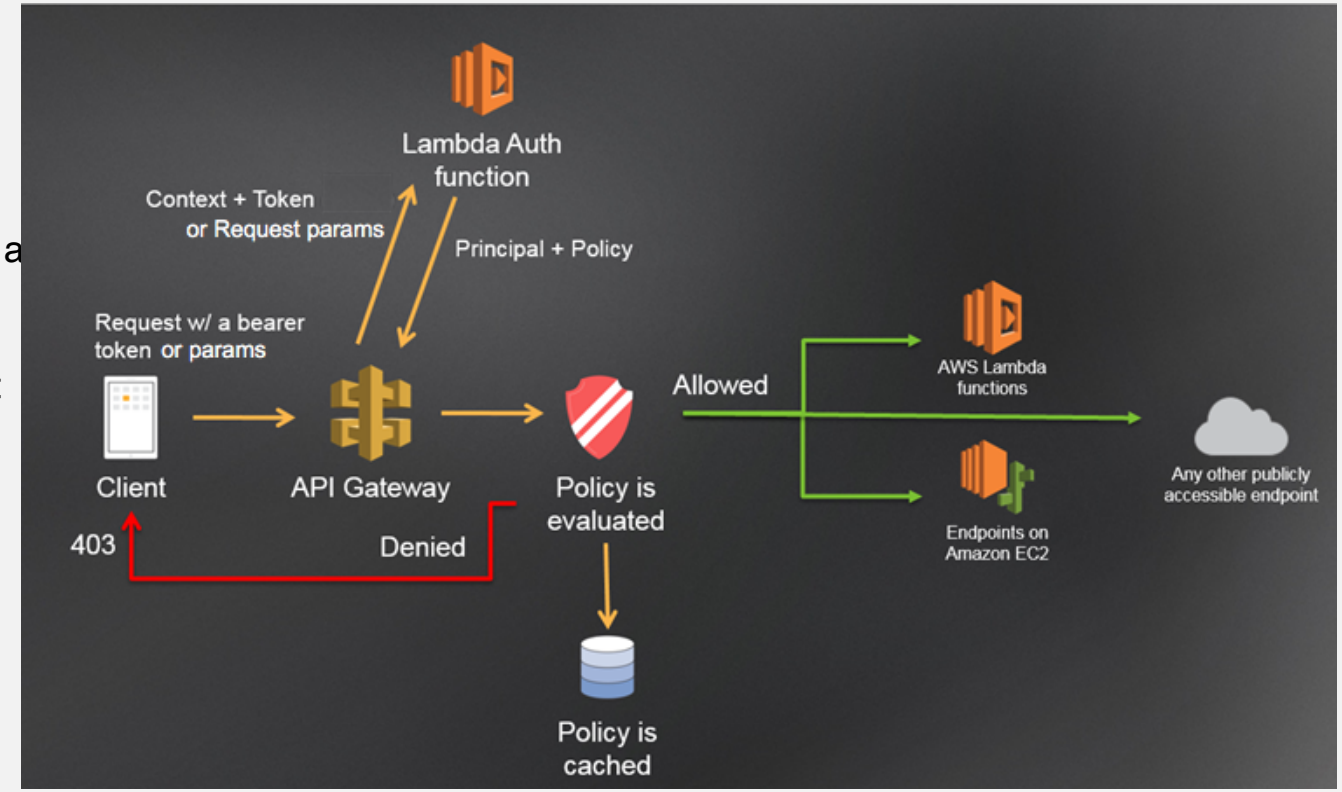
API Gateway – Security

- **Cognito** fully manages user lifecycle, token expires automatically
- API gateway verifies identity automatically from AWS Cognito
- No custom implementation required
- This handles the authentication part. Authorization is handled at the level of Lambda (or backend code).

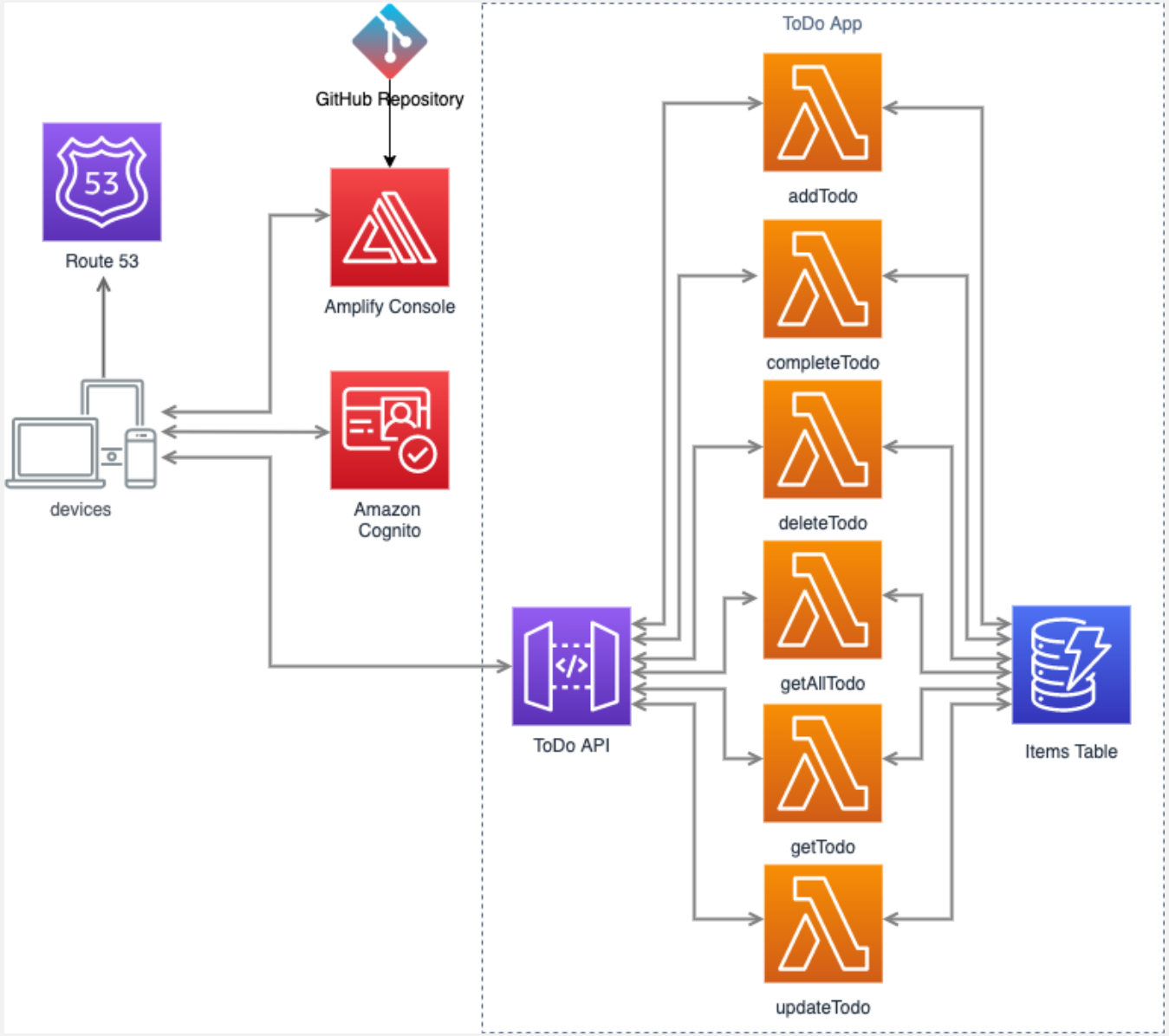


API Gateway – Security

- A **Lambda authorizer** (formerly known as a custom authorizer) is an API Gateway feature that uses a Lambda function to control access to your API.
- A Lambda authorizer is useful if you want to implement a custom authorization scheme that uses a bearer token authentication strategy such as OAuth or SAML, or that uses request parameters to determine the caller's identity.
- When a client makes a request to one of your API's methods, API Gateway calls your Lambda authorizer, which takes the caller's identity as input and returns an IAM policy as output.

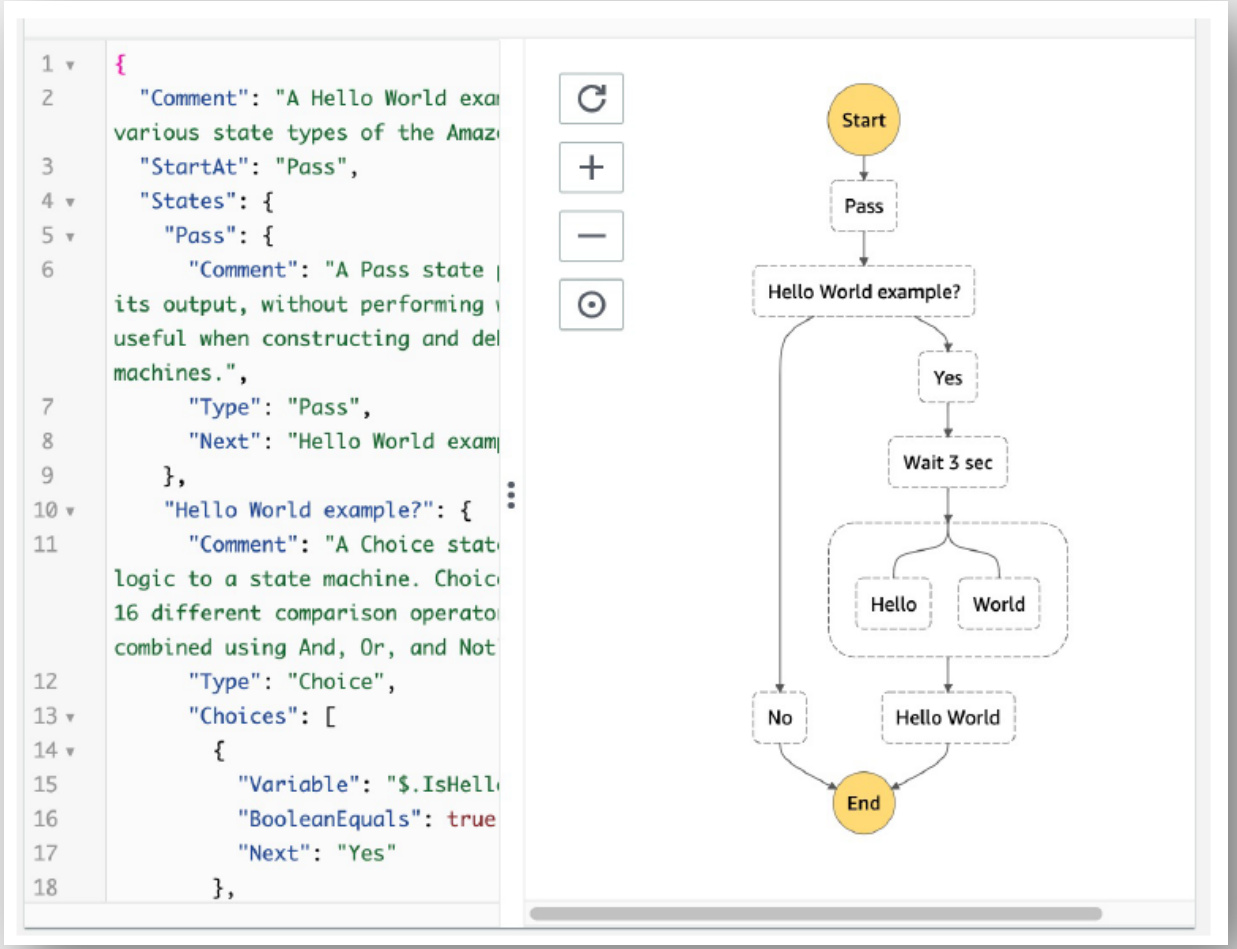


A Serverless Webapp Architecture



AWS Step Functions

- Model your workflows as state machines (one per workflow)
  - Order fulfillment, Data processing
  - Web applications, Any workflow
- Written in JSON
- Visualization of the workflow and the execution of the workflow, as well as history
- Start workflow with SDK call, API Gateway, CloudWatch Event
- Has in-built error handling and retry features



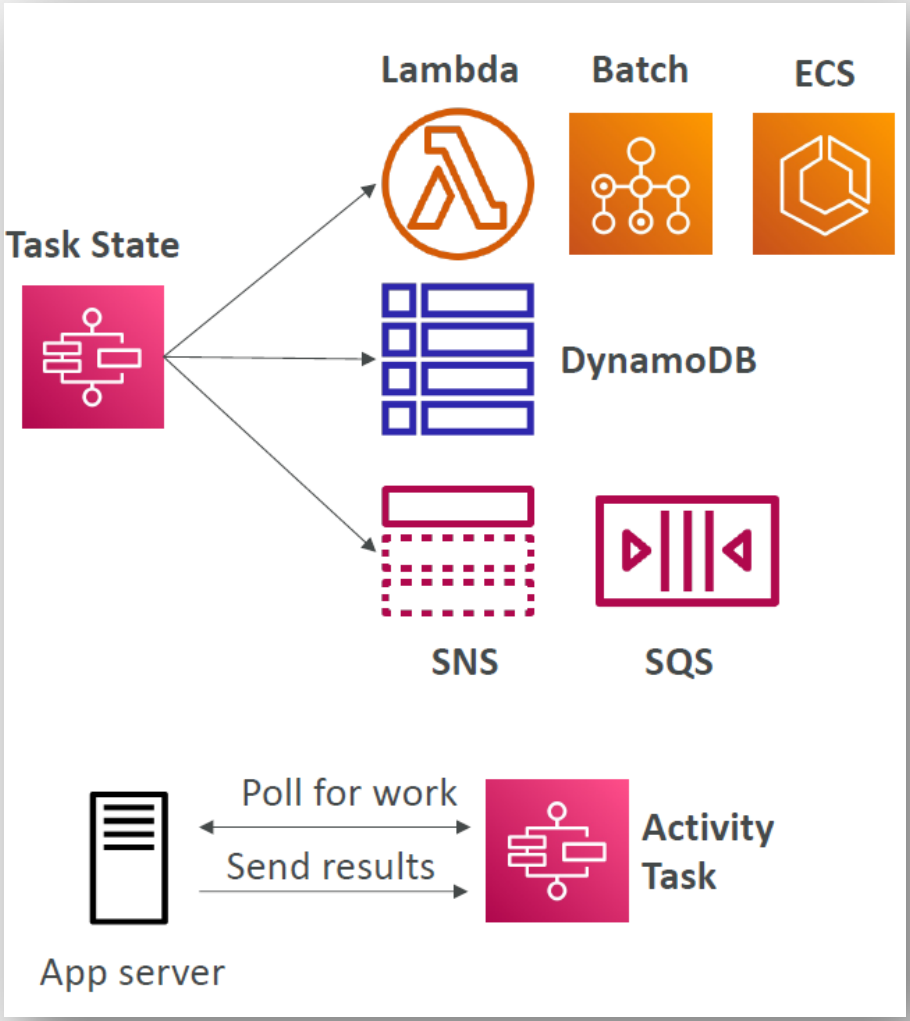
Step Function – Task States

Invoke an AWS service

- Can invoke a Lambda function
- Run an AWS Batch job
- Run an ECS task and wait for it to complete
- Insert an item from DynamoDB
- Publish message to SNS, SQS
- Launch another Step Function workflow…

Run an Activity

- EC2, Amazon ECS, on-premises
- Activities poll the Step functions for work
- Activities send results back to Step Functions



AWS Step Functions

- **Choice State -** Test for a condition to send to a branch (or default branch)
- **Fail or Succeed State** - Stop execution with failure or success
- **Pass State -** Simply pass its input to its output or inject some fixed data, without performing work.
- **Wait State -** Provide a delay for a certain amount of time or until a specified time/date.
- **Map State -** Dynamically iterate steps.
- **Parallel State -** Begin parallel branches of execution.

Labs

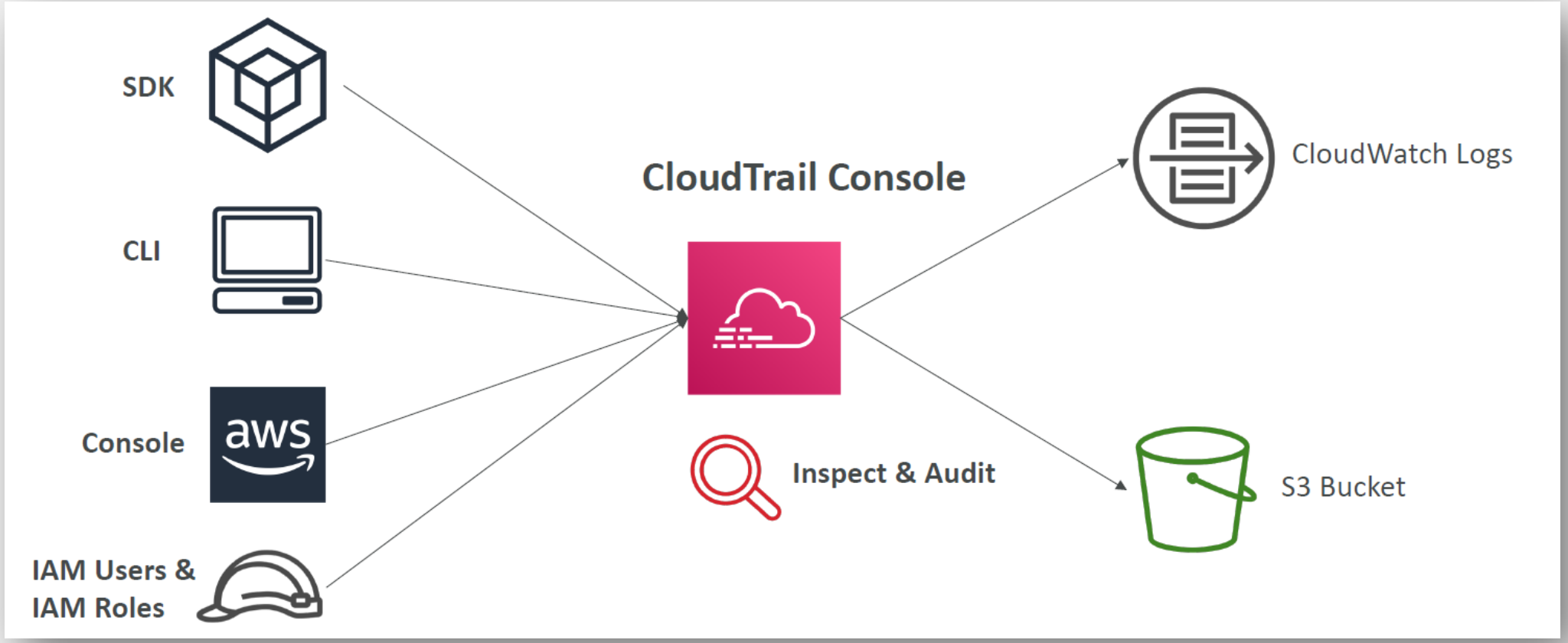
1. Setup a complete serverless web-application using Lambda, DynamoDB, API Gateway, etc.

2. Create a Lambda function to create snapshots of EBS volumes of EC2 instances.
   - Please use the code provided and review the same.
   - You will have to create an IAM role for your Lambda function. Allocate this IAM role to the Lambda Function.
   - Set the timeout for Lambda Function to 1 minute.
   - Execute the Lambda Function and then verify the result by looking at the EBS Snapshots.
   - Delete all the snapshots at the end of the lab.

AWS CloudTrail

- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs or S3
- A trail can be applied to All Regions (default) or a single Region.
- If a resource is deleted in AWS, investigate CloudTrail first!

AWS CloudTrail – Information Flow



AWS CloudTrail Events

Management Events:

- Operations that are performed on resources in your AWS account
- Examples:
  - Configuring security (IAM AttachRolePolicy)
  - Configuring rules for routing data (Amazon EC2 CreateSubnet)
  - Setting up logging (AWS CloudTrail CreateTrail)
- By default, trails are configured to log management events.
- Can separate Read Events (that don't modify resources) from Write Events (that may modify resources)

Data Events:

- By default, data events are not logged (because of high volume operations)
- Amazon S3 object-level activity (ex: GetObject, DeleteObject, PutObject): can separate Read and Write Events
- AWS Lambda function execution activity (the Invoke API)

AWS Secrets Manager

- A service meant for storing secrets
- Capability to force **rotation of secrets** every X days
- Automate generation of secrets on rotation (uses Lambda – custom code to interact with the respective service)
- Integration with Amazon RDS (MySQL, PostgreSQL, Aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration

AWS Key Management Service (KMS)

- AWS KMS makes it easy for you to create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications.

- AWS KMS is integrated with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs.

- Three types of Customer Master Keys (CMK):
  - AWS Managed Service Default CMK: free
  - User Keys created in KMS: $1 / month
  - User Keys imported (must be 256-bit symmetric key): $1 / month

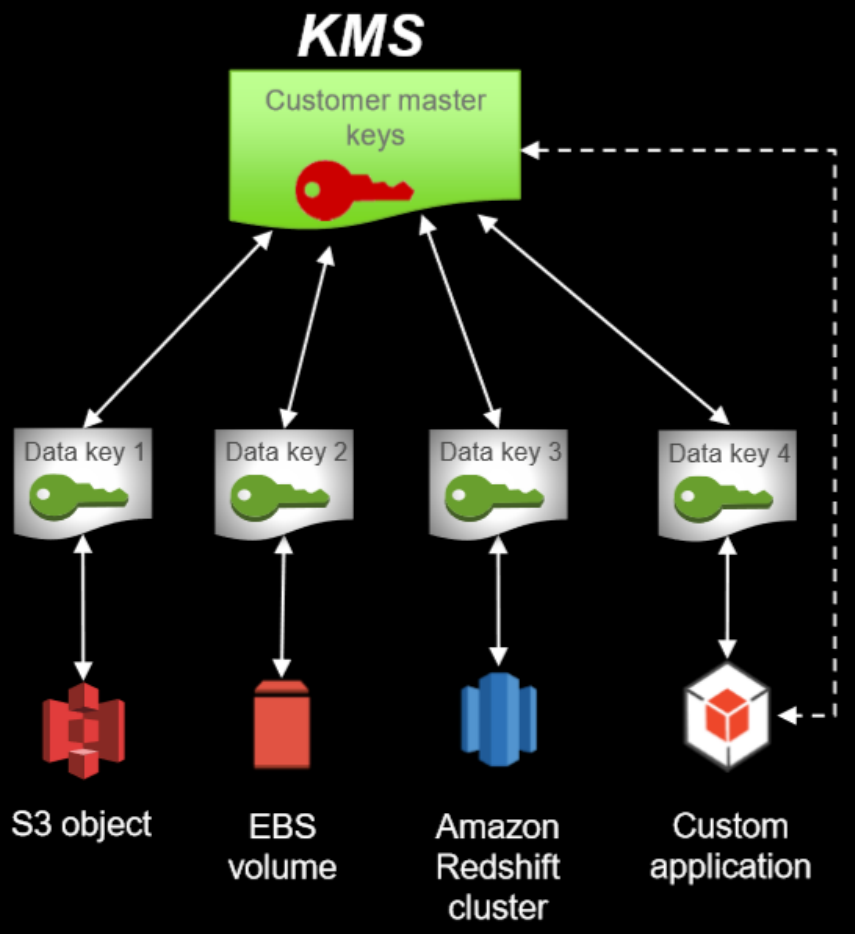
Client Integration with KMS

Two-tiered key hierarchy using envelope encryption

- Unique data key encrypts customer data
- KMS master keys encrypt data keys

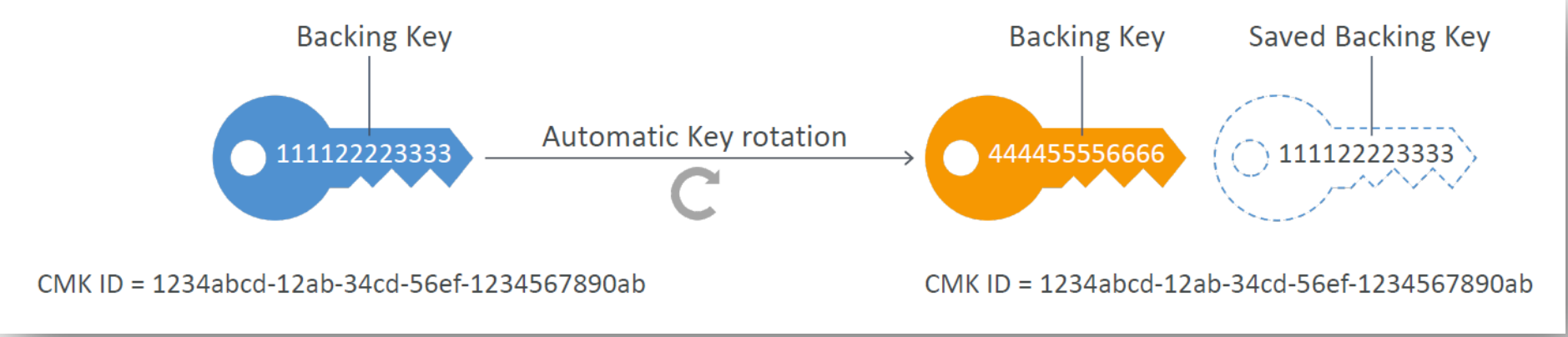
Benefits

- Limits risk of compromised data key
- Better performance for encrypting large data
- Easier to manage small number of master keys than millions of data keys
- Centralized access and audit of key activity



KMS Automatic Key Rotation

- For Customer-managed CMK (not AWS managed CMK)
- If enabled: automatic key rotation happens every 1 year
- Previous key is kept active so you can decrypt old data
- New Key has the same CMK ID (only the backing key is changed)



Infrastructure as Code

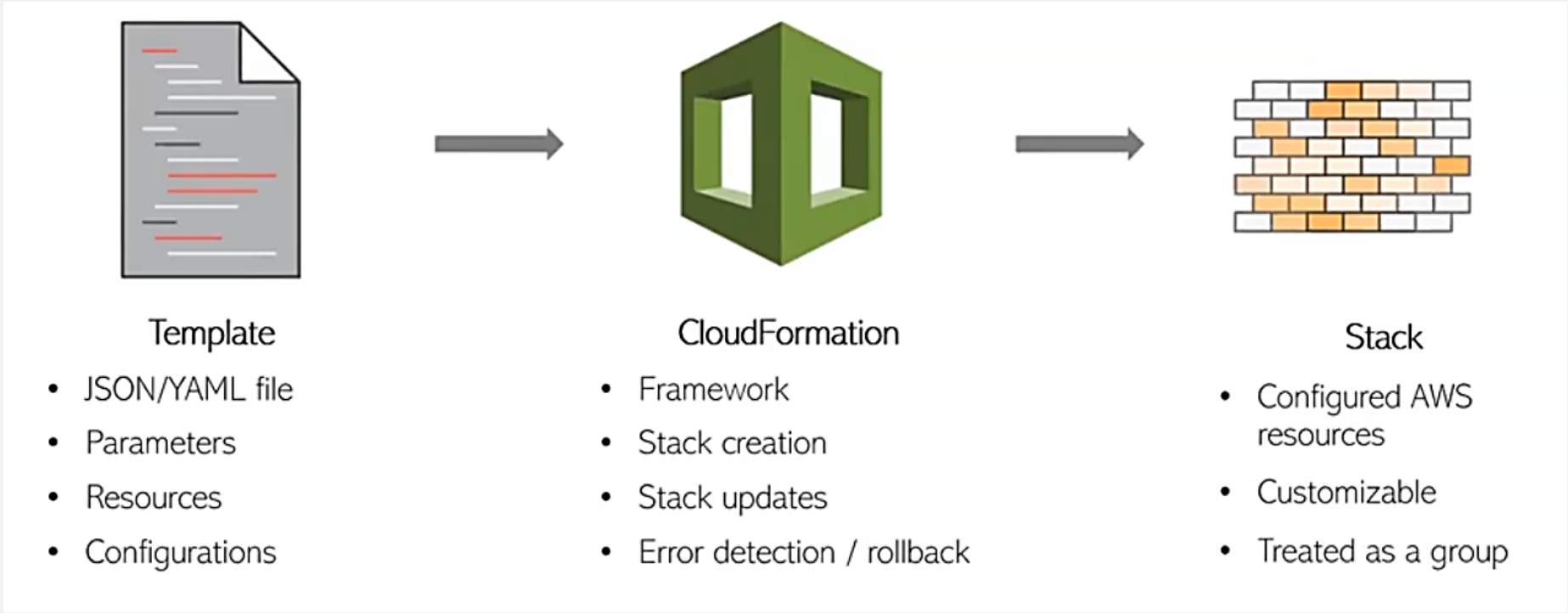
Infrastructure as Code

- IaC is a practice in which traditional infrastructure management techniques are **supplemented by or replaced** with **code-based** tools and **software development** techniques.
- Cloud gives you the option to interact with infrastructure via code (simple API calls).
- What you want >> Resource
- The way you want to customize it >> Attributes

How could it help my Organization

- Save time & bring standards (reduce manual intervention)
- Version controlled and hence ability to go back
- Foundation step to enable self-service in your organization

The Flow


Template
JSON/YAML file
Parameters
Resources
Configurations
CloudFormation
Framework
Stack creation
Stack updates
Error detection / rollback
Stack
Configured AWS resources
Customizable
Treated as a group

Terraform vs CloudFormation

	AWS CloudFormation	Terraform
Native to a Cloud Provider	Y	N
Declarative/programming	Declarative	Declarative
UI support to enable self-service	Y	N
RBAC support	Y	Y (in premium version)
Support Cost	N	Y (in premium version)
DevOps Support	Y	Y
Integration with CMP	N (depends)	Y

Configuration Management Options

- Tools like Chef, Ansible, Puppet
- AWS Systems Manager
- EC2 UserData
- Managing Golden Image

Infra as Code + Pipeline

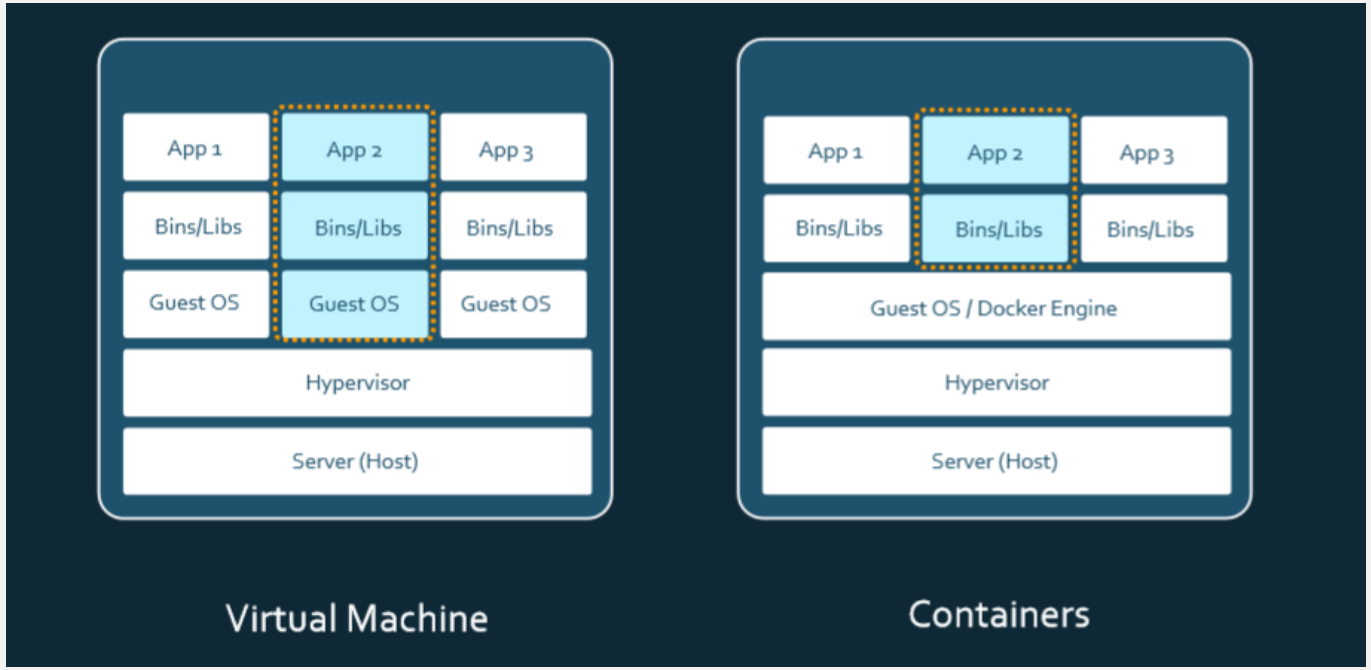
- Pipelines are the top-level component of continuous integration, delivery, and deployment.
- Pipelines comprise:
  - Jobs, which define what to do. For example, jobs that compile or test code.
  - Stages, which define when to run the jobs.
- Gitlab Pipelines are used to trigger Terraform code
- There are multiple stages in the pipeline.
- There is a manual approval stage in this.
- There are terraform templates for the approved building blocks in your organization.
- User can choose the relevant version of code from the repository and modify it to provide the parameter values.

Labs

- AWS resource deployment using Terraform Code.
- Configure terraform in the Public EC2 instance and execute the steps.
- Verify the created resources in AWS Management Console.

Containers

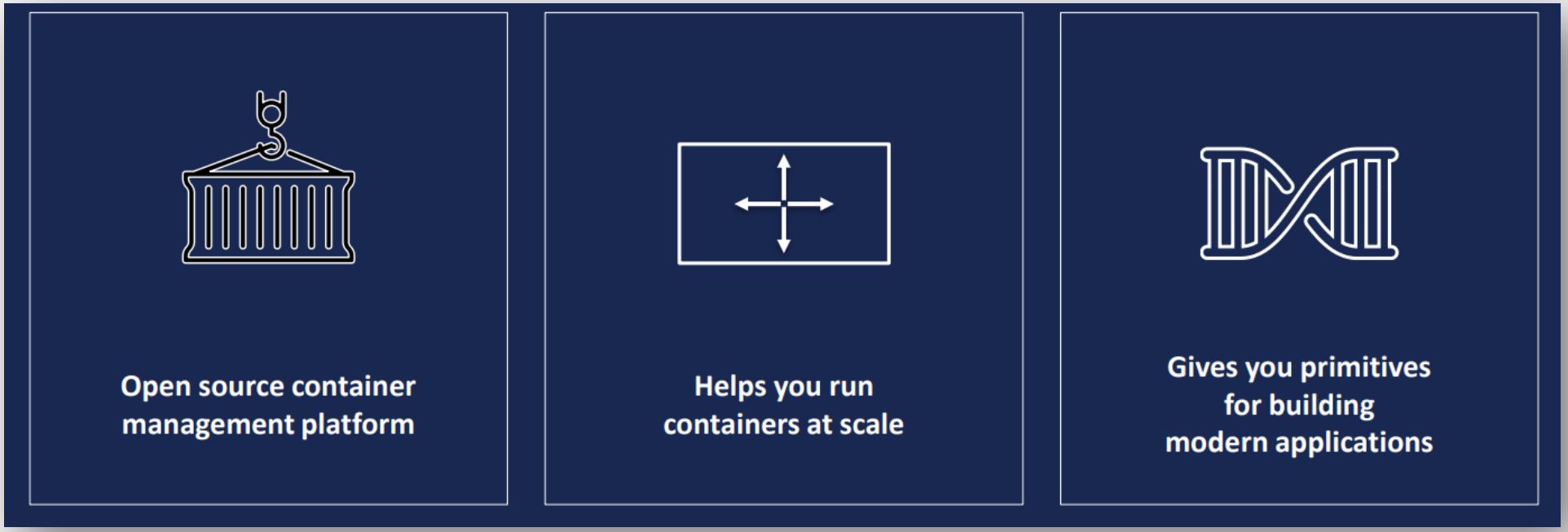
VMs vs Containers



Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight because they don't need the extra load of a hypervisor, but run directly within the host machine's kernel. This means you can run more containers on a given hardware combination than if you were using virtual machines. You can even run Docker containers within host machines that are actually virtual machines!

What is Kubernetes?

- Built on over a decade of experience and best practices
- Utilizes declarative configuration and automation
- Draws upon a large ecosystem of tools, services, support



EKS Overview

- Amazon EKS = Amazon Elastic Kubernetes Service
- It is a way to launch **managed Kubernetes clusters on AWS**
- EKS supports EC2 if you want to deploy worker nodes or Fargate to deploy serverless containers
- **Use case**: if your company is already using Kubernetes on-premises or in another cloud, and wants to migrate to AWS using Kubernetes
- **Kubernetes is cloud-agnostic** (can be used in any cloud – AWS, Azure, GCP, etc.)
- Can utilize images from Docker Hub or ECR.

EKS – Managed Kubernetes Setup

Managed K8s control plane — highly available master and etcd

Bring your own worker nodes, like ECS

Core tenets

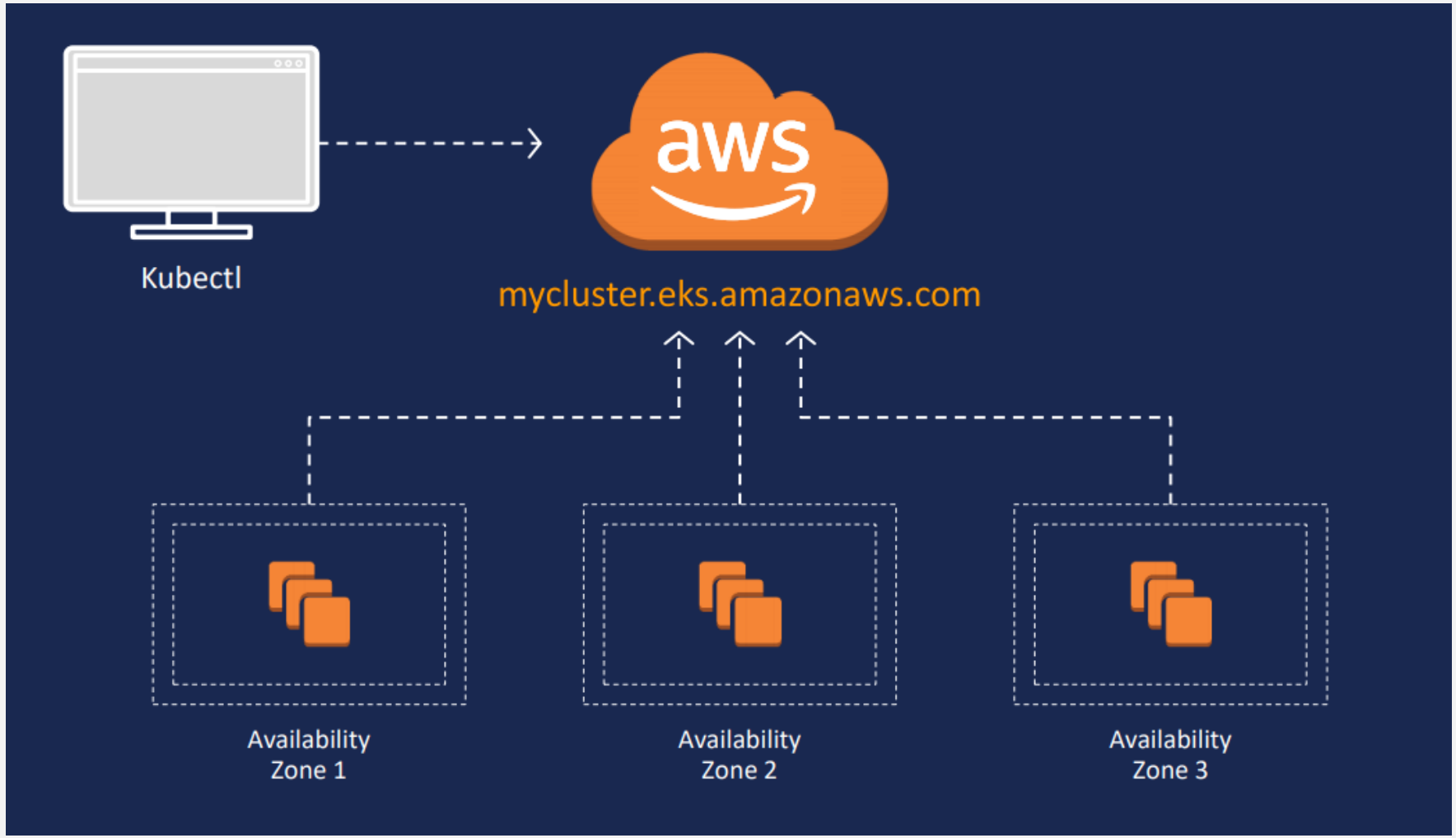
- Platform for enterprises to run production-grade workloads
- Provides a native and upstream Kubernetes experience – Kubernetes certified
- Not forced to use additional AWS services, but offer seamless integration
- Actively contributes to the Kubernetes project

APIs

```
aws eks create-cluster \
  --cluster-name <> \
  --desired-master-version <> \
  --role-arn <>
```

Otherwise, you can use **eksctl** to create EKS cluster. It has certain advantages.

High Level EKS Setup



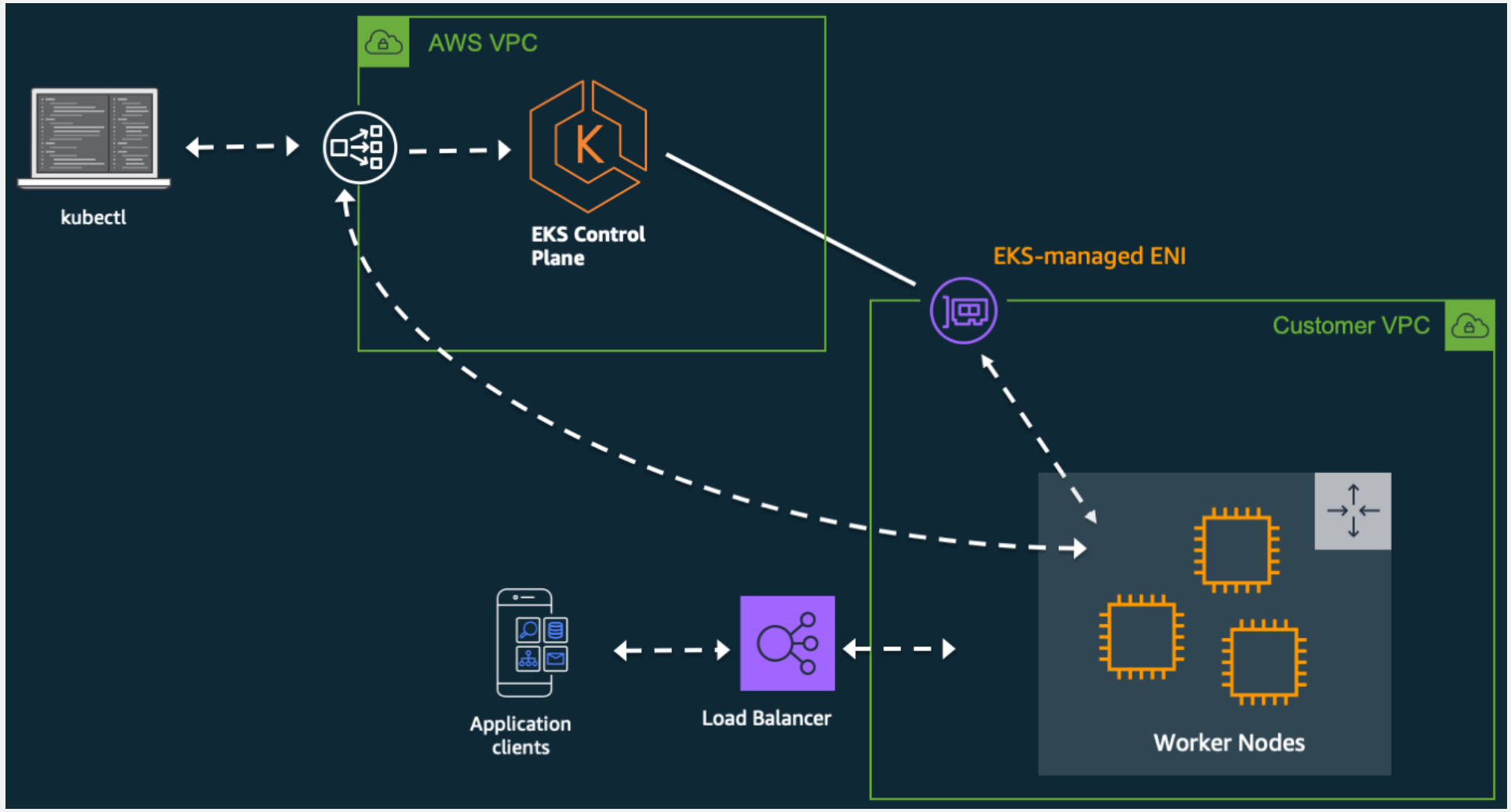
Kubernetes Nodes

- The machines that make up a Kubernetes cluster are called **nodes**.
- Nodes in a Kubernetes cluster may be **physical**, or **virtual**.

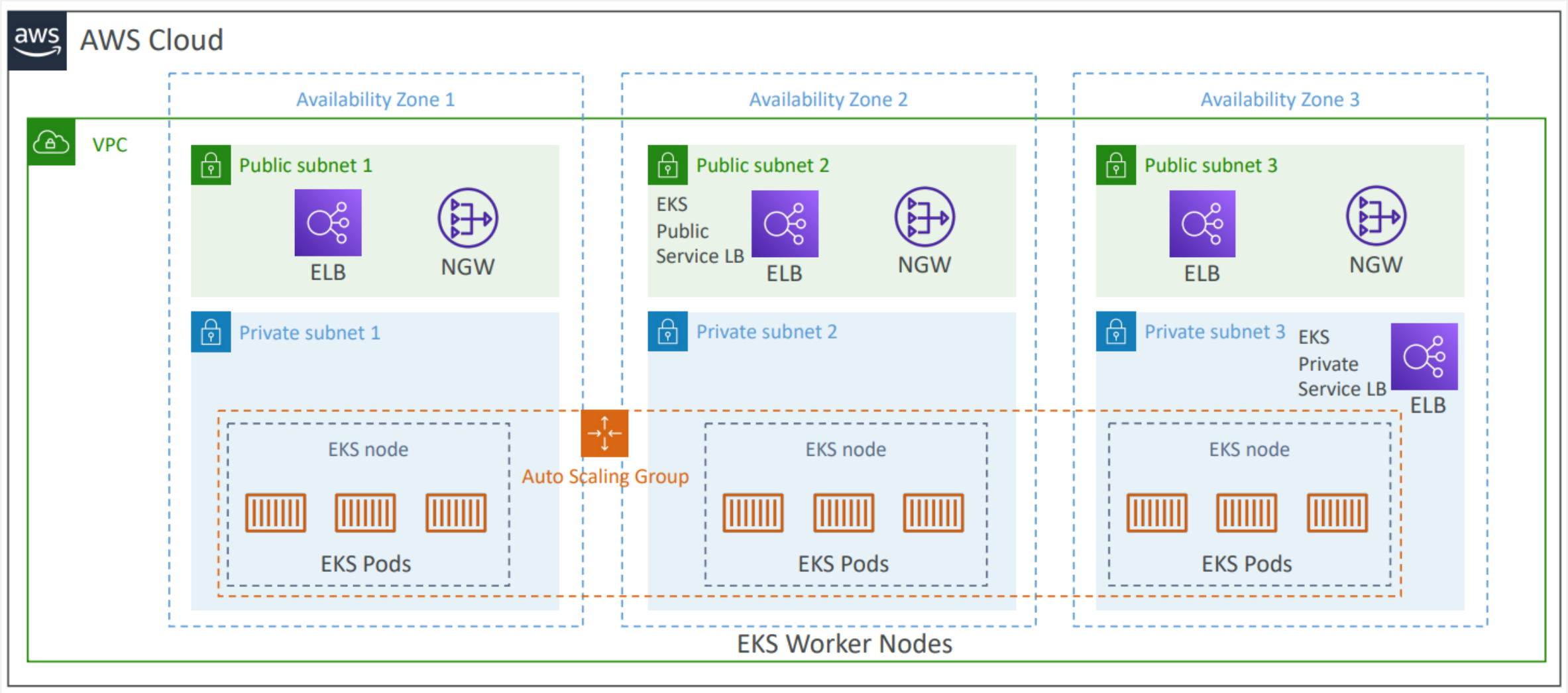
There are two types of nodes:

1. A **Control-plane-node** type, which makes up the Control Plane, acts as the "brains" of the cluster.
2. A **Worker-node** type, which makes up the Data Plane, runs the actual container images (via pods).

Architecture for CP and Worker Communication



EKS Diagram


AWS Cloud
Availability Zone 1
Availability Zone 2
Availability Zone 3
VPC
Public subnet 1
ELB
NGW
Public subnet 2
EKS
Public
Service LB
ELB
NGW
Public subnet 3
ELB
NGW
Private subnet 1
Private subnet 2
Private subnet 3
EKS
Private
Service LB
ELB
EKS node
EKS Pods
Auto Scaling Group
EKS node
EKS Pods
EKS node
EKS Pods
EKS Worker Nodes

EKS Pricing

- **For EC2 Model:**
  - You pay $0.10 per hour for each Amazon EKS cluster that you create.
  - Additionally, the number of EC2 instances you create.

- **For Fargate Model:**
  - You pay $0.10 per hour for each Amazon EKS cluster that you create.
  - Additionally, you pay based on the vCPU and Memory chosen.

EKS Fargate

- AWS Fargate is a technology that provides **on-demand, right-sized compute capacity** for containers.
- With AWS Fargate, you **don't have to provision**, configure, or scale groups of **virtual machines** on your own to run containers.
- You also **don't need to choose server types**, decide when to scale your node groups, or optimize cluster packing.
- You can control **which pods start on Fargate** and how they run with Fargate profiles.
- Fargate profiles are defined as part of your Amazon EKS cluster.

Specifications with Fargate

vCPU value	Memory value
.25 vCPU	0.5 GB, 1 GB, 2 GB
.5 vCPU	1 GB, 2 GB, 3 GB, 4 GB
1 vCPU	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB
2 vCPU	Between 4 GB and 16 GB in 1-GB increments
4 vCPU	Between 8 GB and 30 GB in 1-GB increments

- If you do not specify a vCPU and memory combination, then the **smallest available combination** is used (.25 vCPU and 0.5 GB memory).
- The **additional memory reserved for the Kubernetes components (256 MB)** can cause a Fargate task with more vCPUs than requested to be provisioned. For example, a request for 1 vCPU and 8 GB memory will have 256 MB added to its memory request, and will provision a Fargate task with 2 vCPUs and 9 GB memory, **since no task with 1 vCPU and 9 GB memory is available.**

Labs

- On your Public EC2 instance, configure AWS CLI with your credentials (generate it from IAM > CloudLearner).
- Install eksctl & kubectl
- Create an EKS Fargate cluster using eksctl.
- Create pods and target them to the correct Fargate profile.
- Access an AWS service from the pod.
- Delete the EKS cluster.