AWS Advanced

Gagandeep Singh

Introduction

Name

Total Experience

Background – Development / Infrastructure / Database / Network

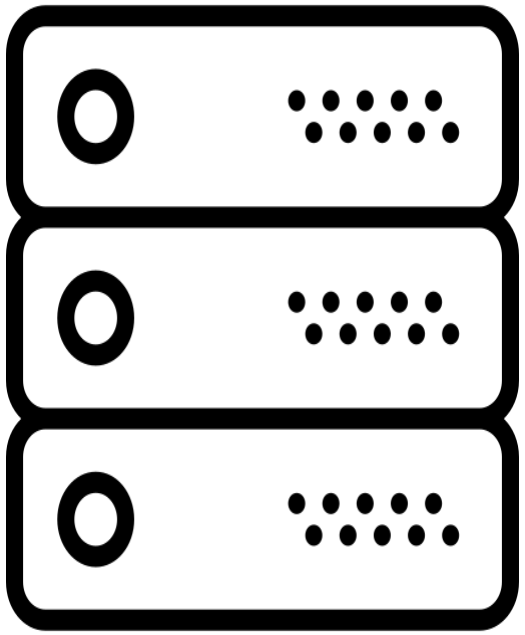
Experience on Cloud/AWS

Your expectations from this training

Cloud

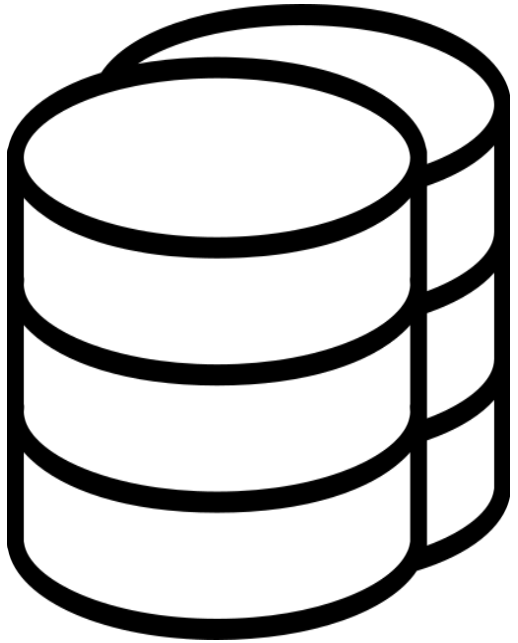
What is Cloud?

Traditional Datacenters



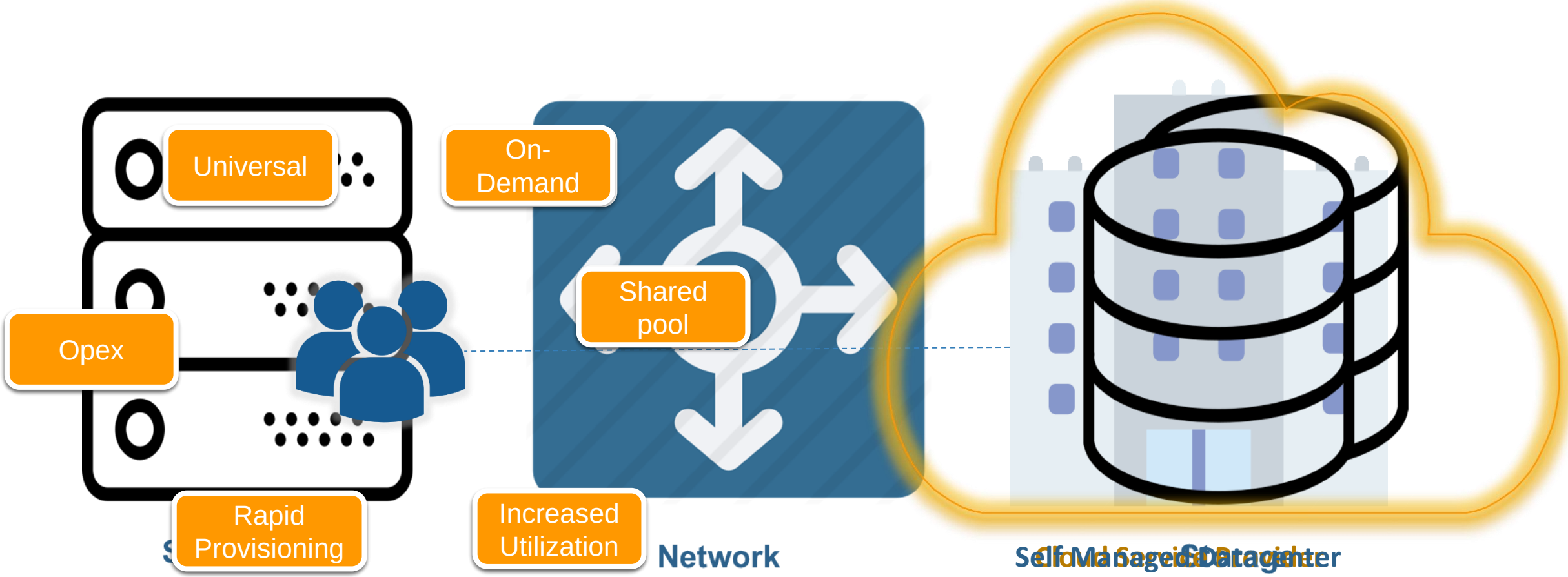
Servers

Network

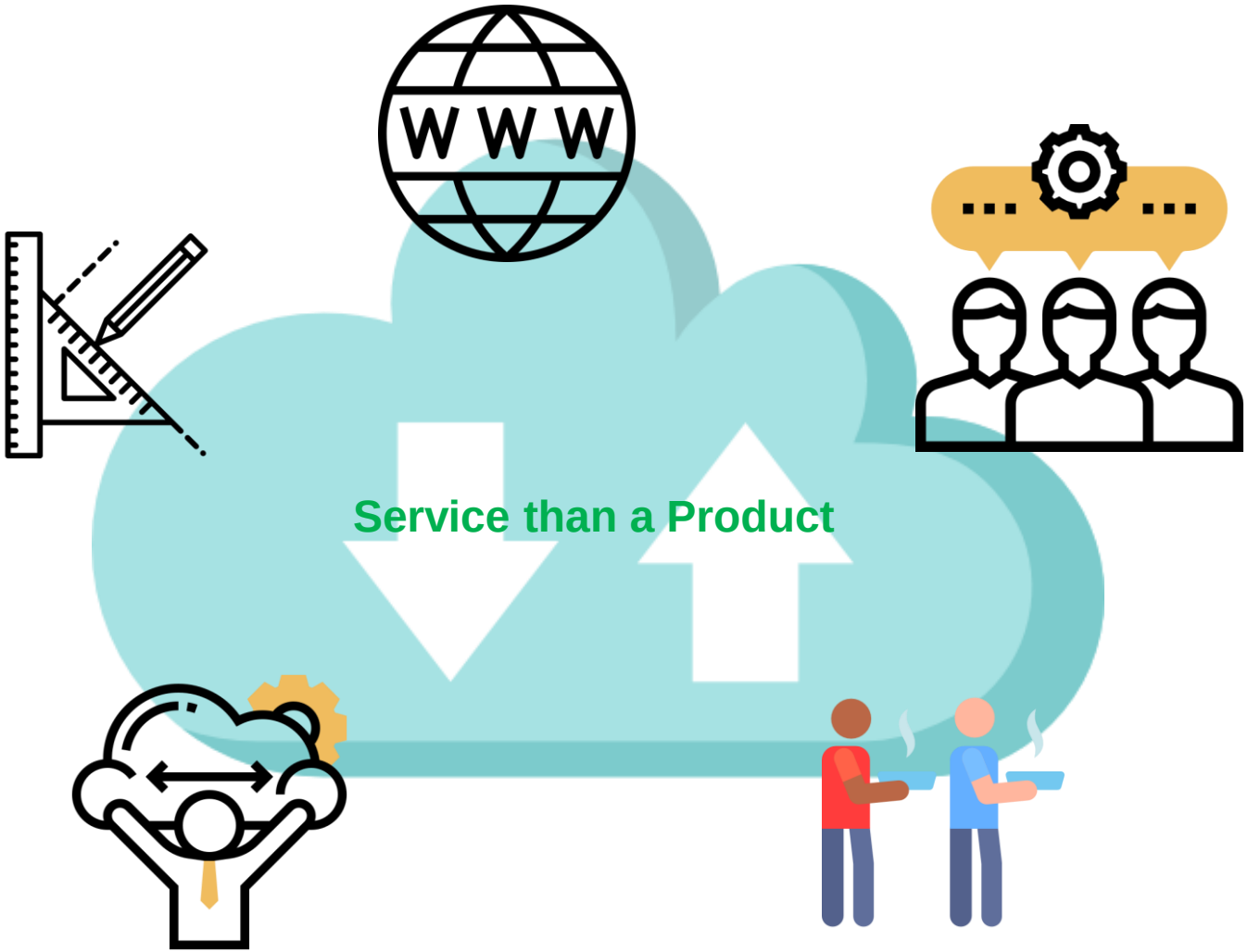


Storage

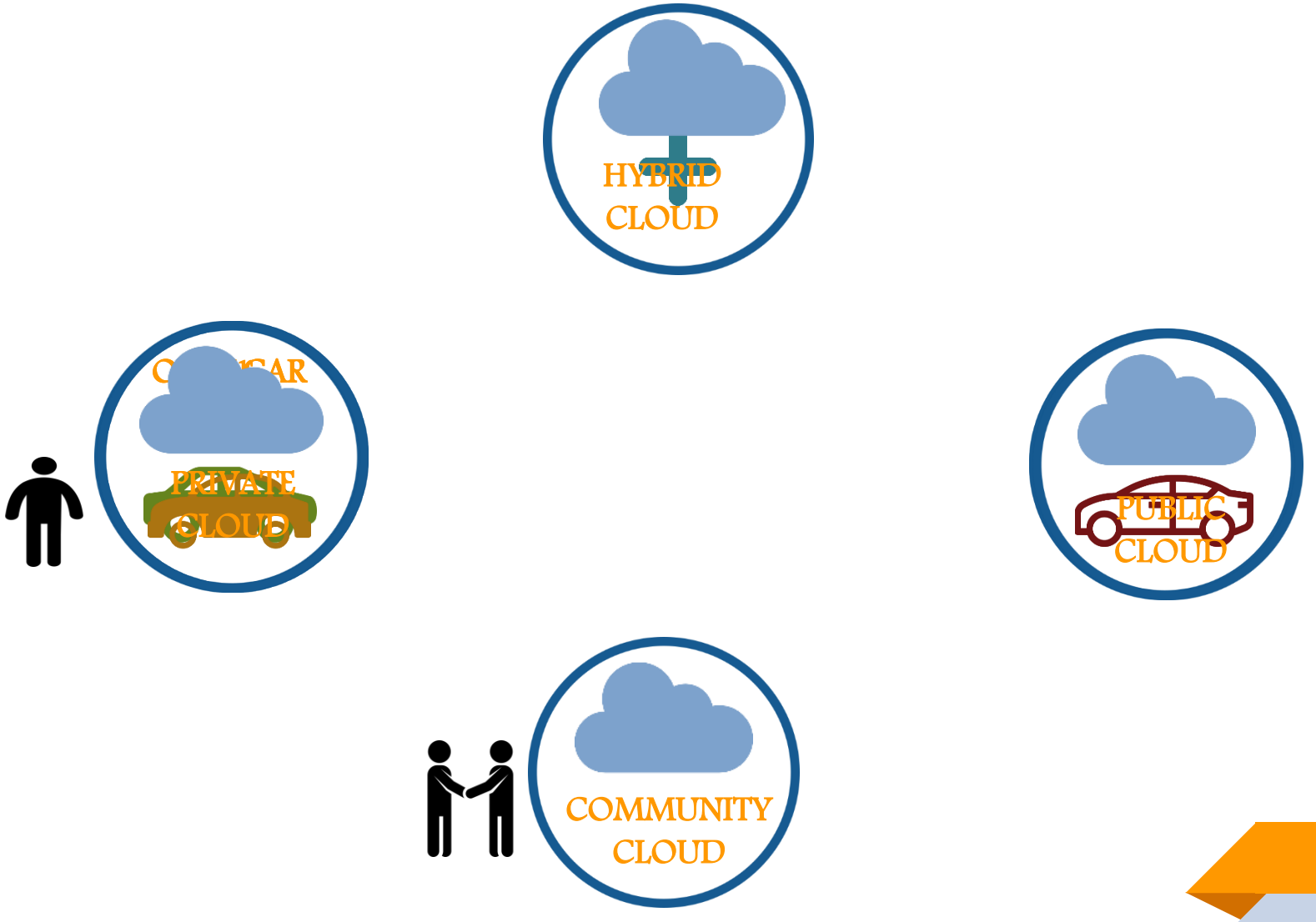
Traditional Data centers



Characteristics of Cloud

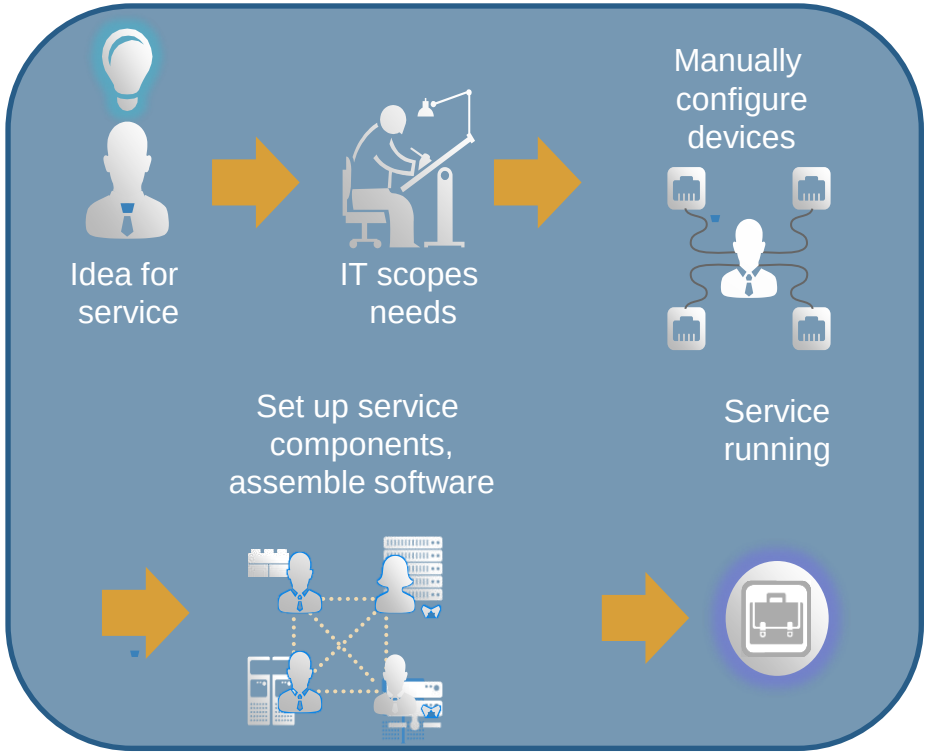


Cloud Deployment Types



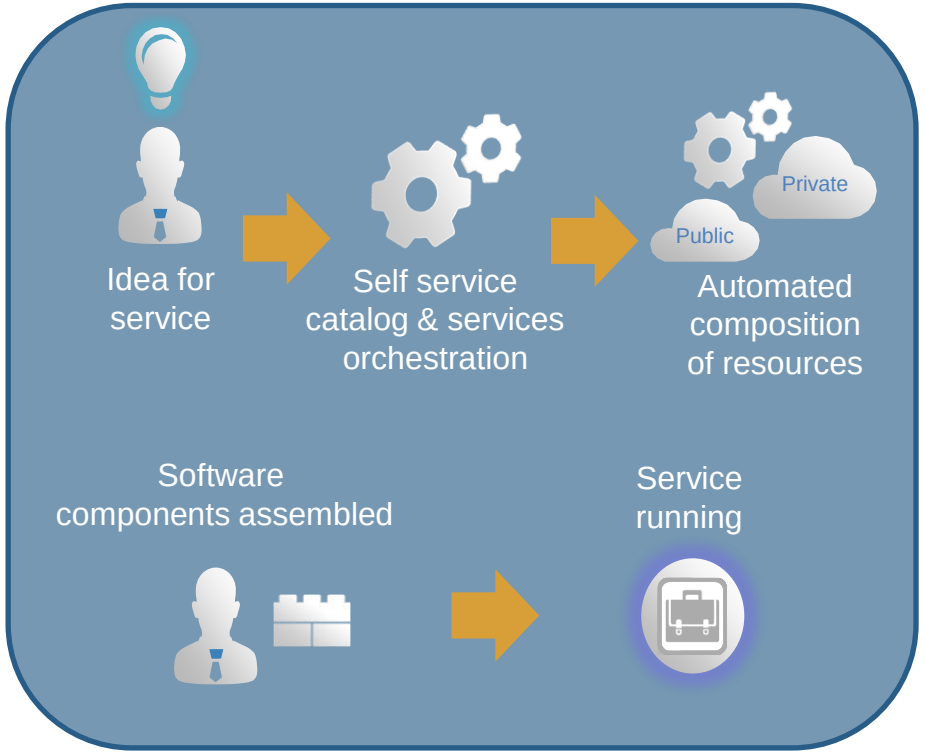
Business Impact

Traditional Datacenter



Time to Provision New Service: Months

Cloud Infrastructure



Time to Provision New Service: Minutes

Cloud Benefits

Cloud Major Use Cases

On and Off

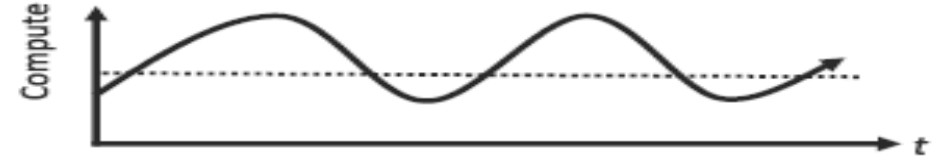
On and off workloads (e.g. batch job)
Over provisioned capacity is wasted
Time to market can be cumbersome

Growing Fast

Successful services needs to grow/scale
Keeping up with growth is a big IT challenge
Cannot provision hardware fast enough

Unpredictable Bursting

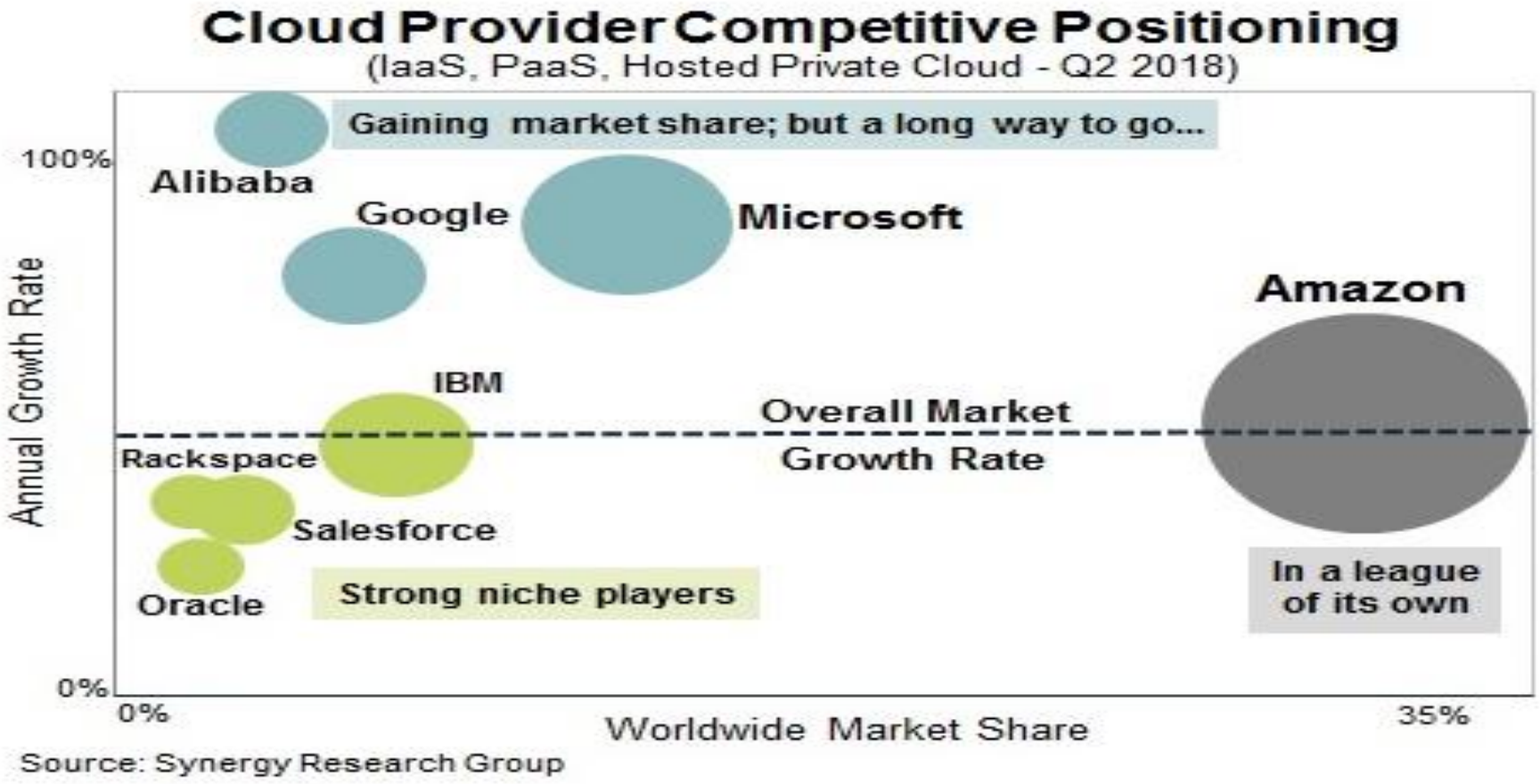
Unexpected/unplanned peak in demand
Sudden spike impacts performance
Cannot over provision for extreme cases



Predictable Bursting

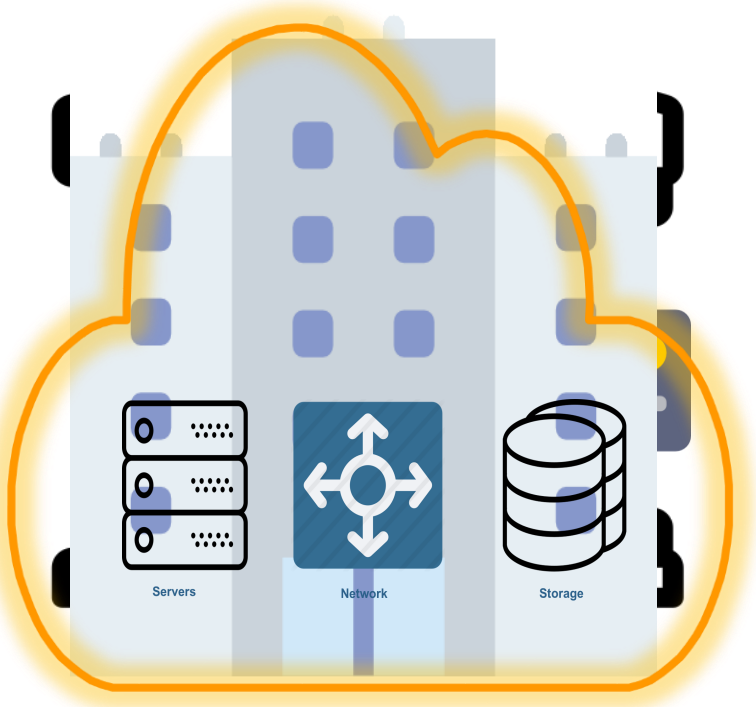
Services with micro seasonality trends
Peaks due to periodic increased demand
IT complexity and wasted capacity

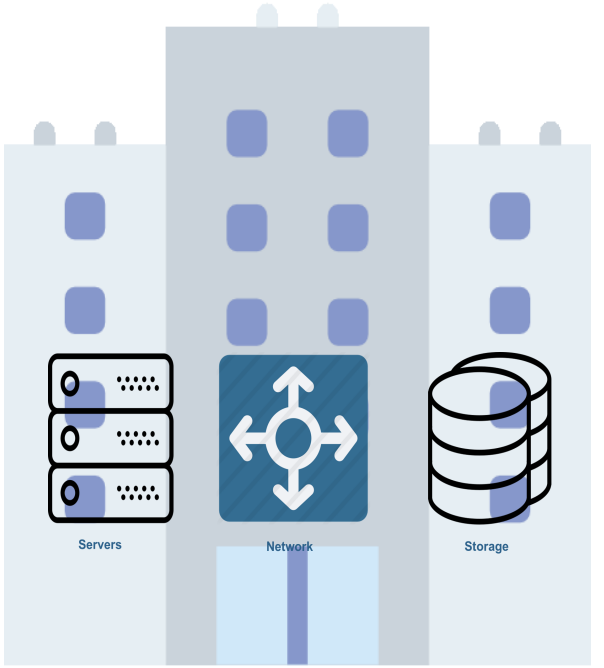
Cloud Players



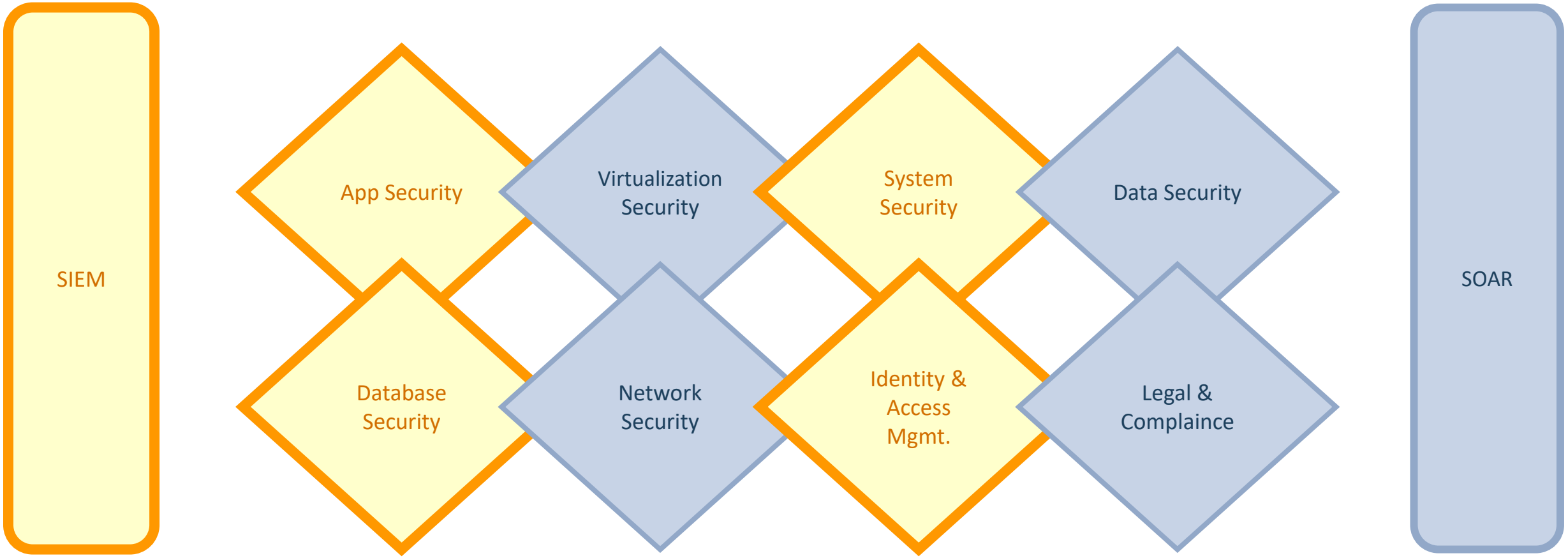
Security

Cloud Security





Security Aspects



AWS

Amazon Web Services

AWS (Amazon Web Services) is a group of web services (also known as cloud services) being provided by Amazon since 2006.

AWS provides huge list of services starting from basic IT infrastructure like CPU, Storage as a service, to advance services like Database as a service, Serverless applications, IOT, Machine Learning services etc..

Hundreds of instances can be build and use in few minutes as and when required, which saves ample amount of hardware cost for any organizations and make them efficient to focus on their core business areas.

Currently AWS is present and providing cloud services in more than 190 countries.

Well-known for IaaS, but now growing fast in PaaS and SaaS.

Why AWS?

Low Cost: AWS offers, pay as you go pricing. AWS models are usually cheapest among other service providers in the market.

Instant Elasticity: You need 1 server or 1000's of servers, AWS has a massive infrastructure at backend to serve almost any kind of infrastructure demands, with pay for what you use policy.

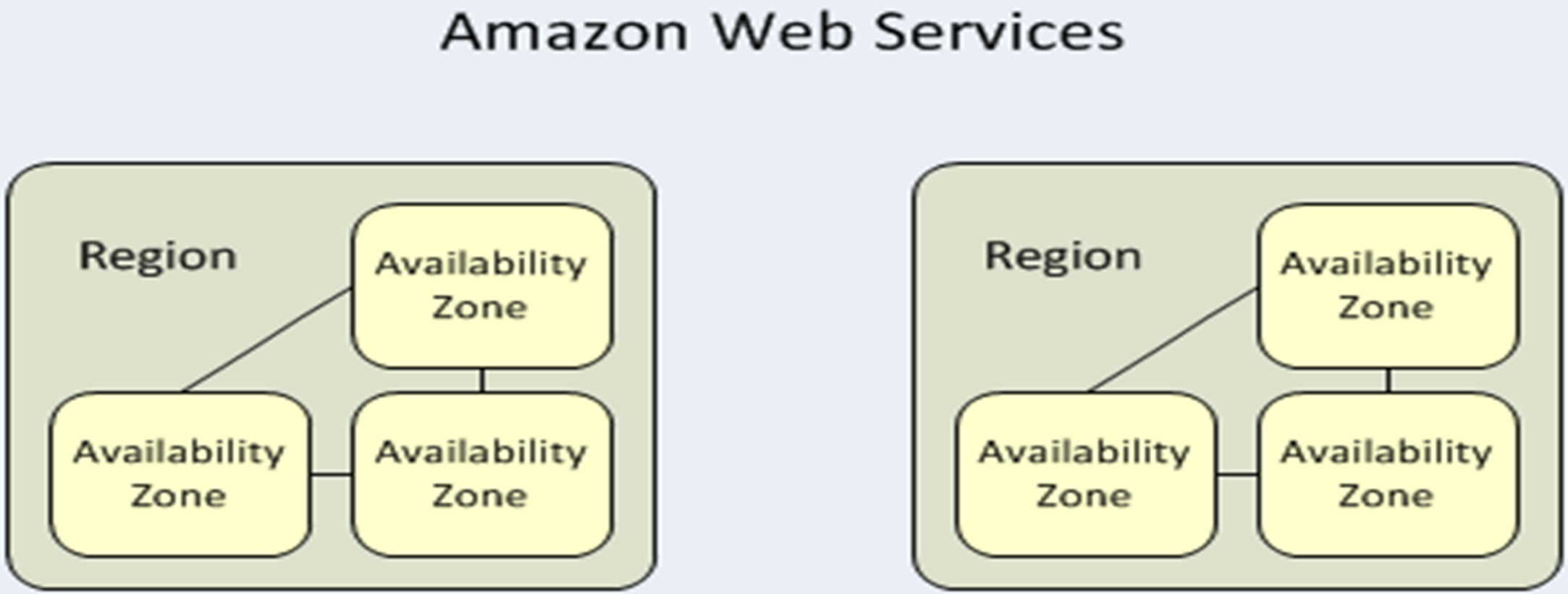
Scalability: Facing some resource issues, no problem within seconds you can scale up the resources and improve your application performance. This cannot be compared with traditional IT datacenters.

Multiple OS's: Choice and use any supported Operating systems.

Multiple Storage Options: Choice of high I/O storage, low cost storage. All is available in AWS, use and pay what you want to use with almost any scalability.

Secure: AWS is PCI DSS Level1, ISO 27001, FISMA Moderate, HIPAA, SAS 70 Type II passed. In-fact systems based on AWS are usually more secure than in-house IT infrastructure systems.

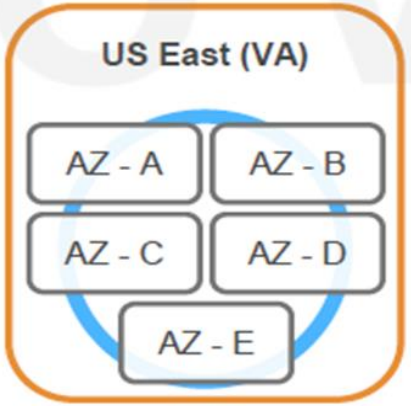
Amazon Web Services



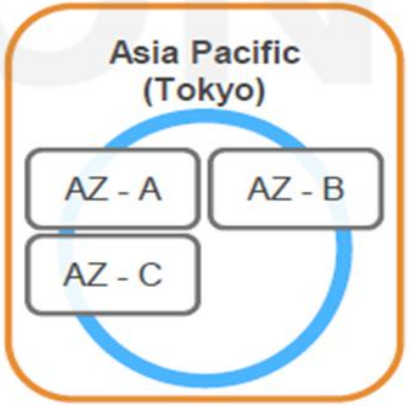
Amazon Web Services

At least 2 AZs per region.

- Examples:
  - US East (N. Virginia)
    - us-east-1a
    - us-east-1b
    - us-east-1c
    - us-east-1d
    - us-east-1e



  - Asia Pacific (Tokyo)
    - ap-northeast-1a
    - ap-northeast-1b
    - ap-northeast-1c



Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

Amazon Web Services

AWS Regions:

-Geographic Locations

-Consists of at least two Availability Zones(AZs)

-All of the regions are completely independent of each other with separate Power Sources, Cooling and Internet connectivity.

-This achieves the greatest possible fault tolerance and stability.

-There is a charge for data transfer between Regions.

-When you view your resources, you'll only see the resources tied to the Region you've specified.

-An AWS account provides multiple Regions so that you can launch Amazon EC2 instances in locations that meet your requirements. For example, you might want to launch instances in Europe to be closer to your European customers or to meet legal requirements.

-Resources aren't replicated across regions unless you do so specifically.

Amazon Web Services

AWS Availability Zones

-AZ is a distinct location within a region

-Each Availability Zone is isolated, but the Availability Zones in a Region are connected through low-latency links.

-Each Region has minimum two AZ’s

-Most of the services/resources are replicated across AZs for HA/DR purpose.

-While launching instance you should specify an Availability Zone if your new instances must be close to, or separated from, your running instances.

Amazon Web Services

Current:

22 AWS Regions

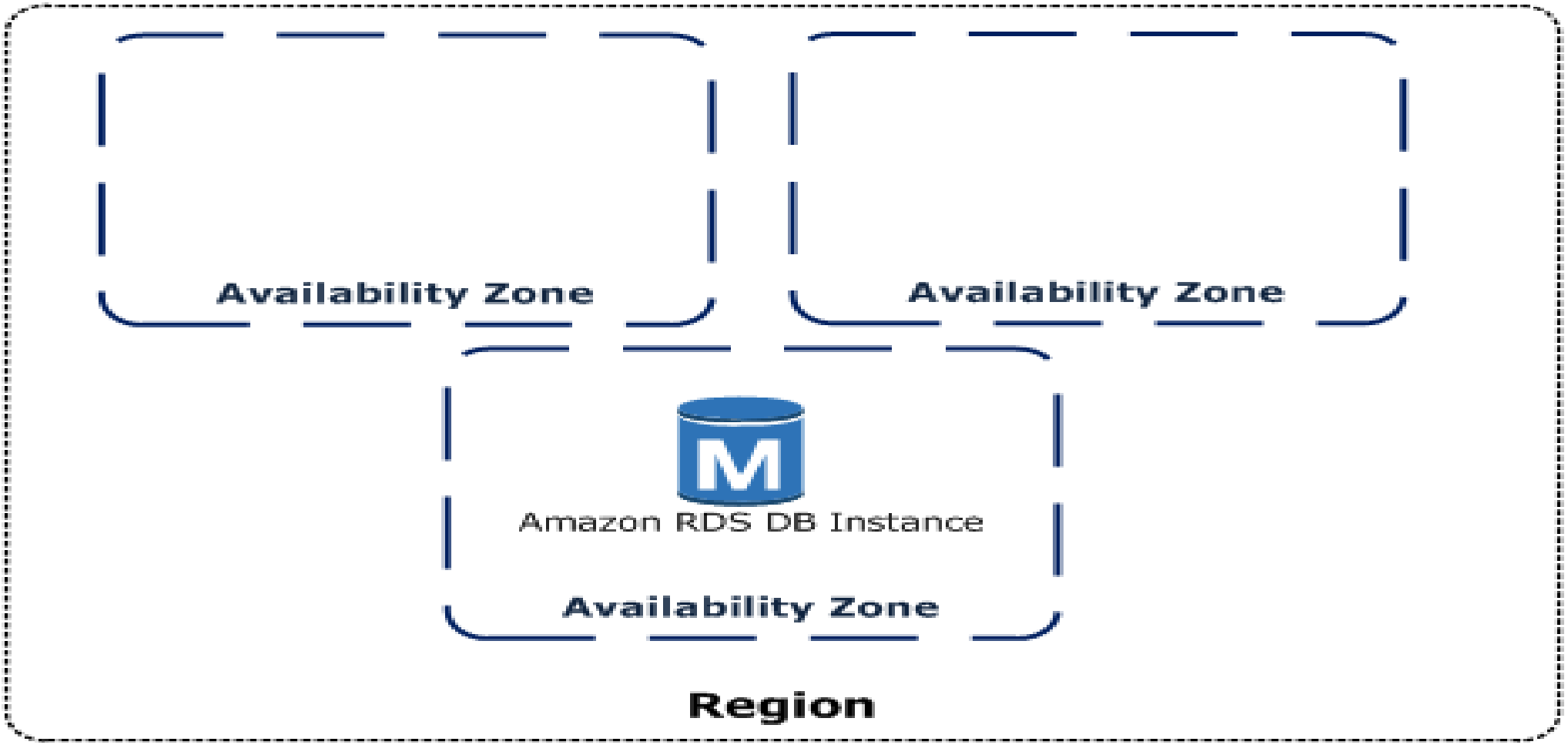
69 AZs

Upcoming:

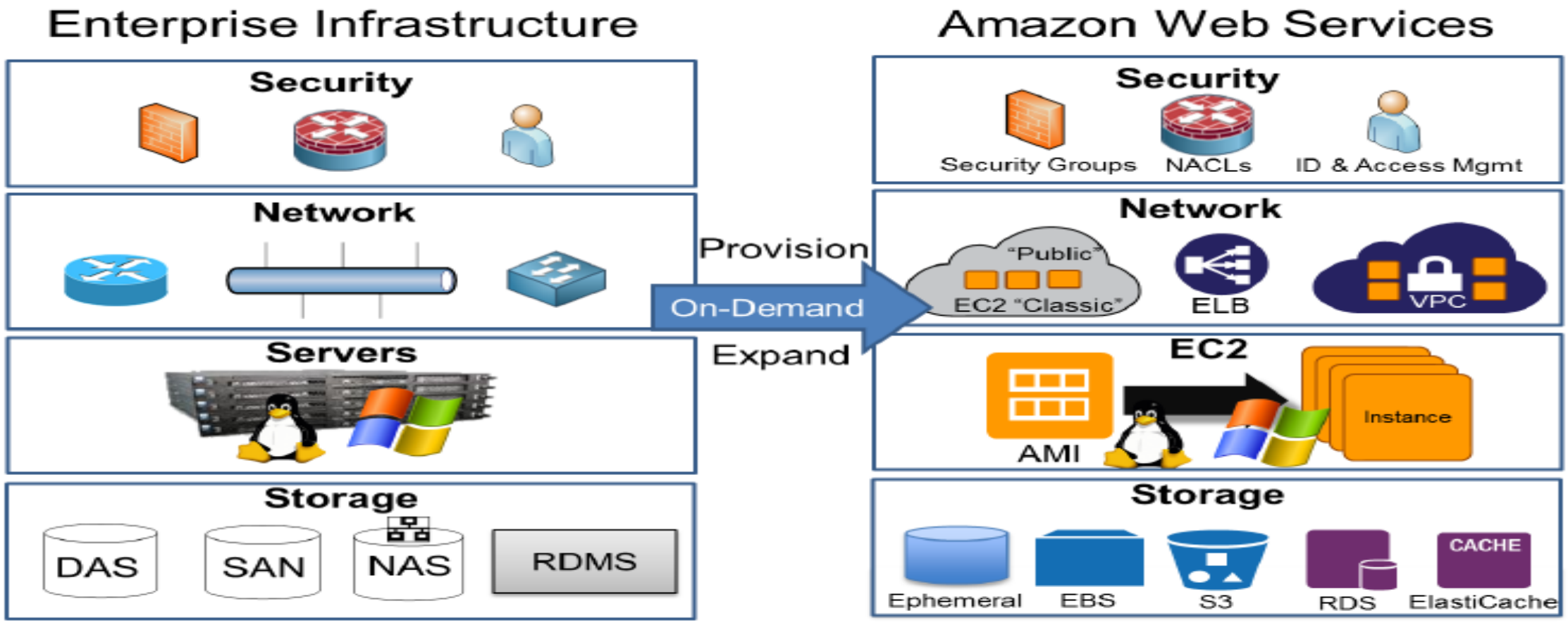
4 Regions

13 AZs

Amazon Web Services



AWS


Enterprise Infrastructure
Security
Network
Servers
Storage
DAS
SAN
NAS
RDMS
Provision
On-Demand
Expand
Amazon Web Services
Security
Security Groups
NACLs
ID & Access Mgmt
Network
"Public"
EC2 "Classic"
ELB
VPC
EC2
AMI
Instance
Storage
Ephemeral
EBS
S3
RDS
CACHE
ElastiCache

Knowledge Checks

- To Achieve HA in AWS, my Servers should be in different (Racks, Datacenters, Availability Zones, Regions)?

- To Achieve DR/High Durability, where should I have by Server backup (Different AZ or Different Regions)?

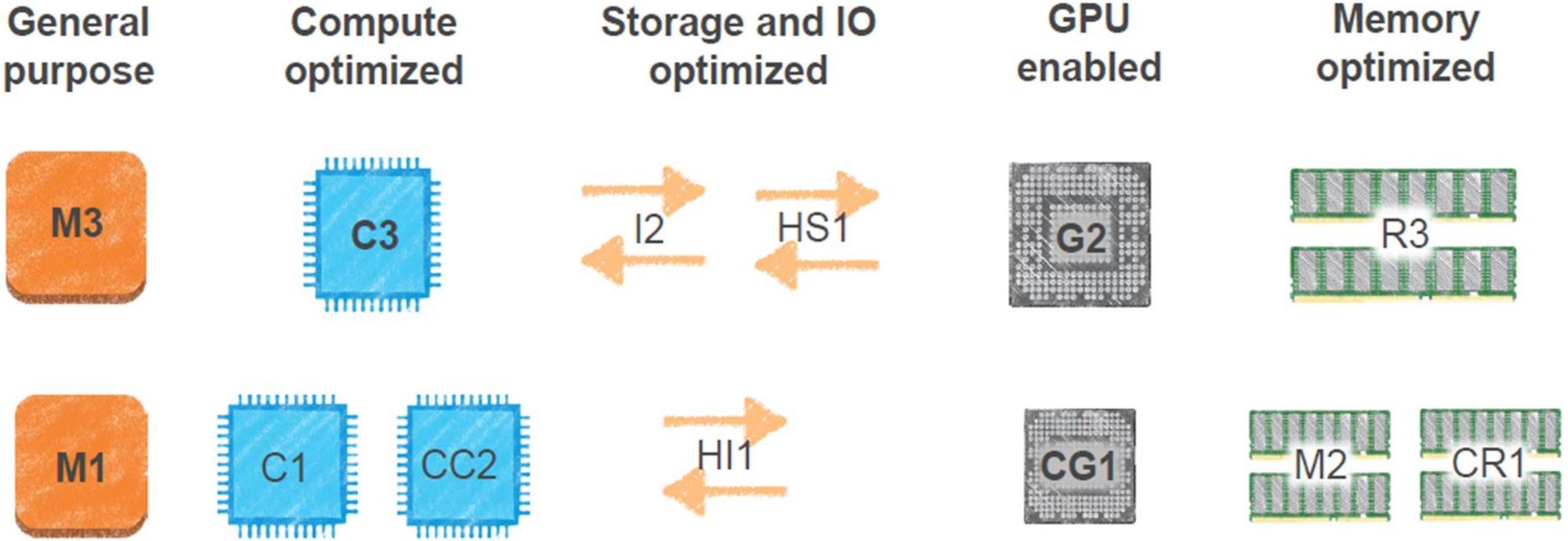
AWS
Compute Services

AWS Elastic Compute Cloud

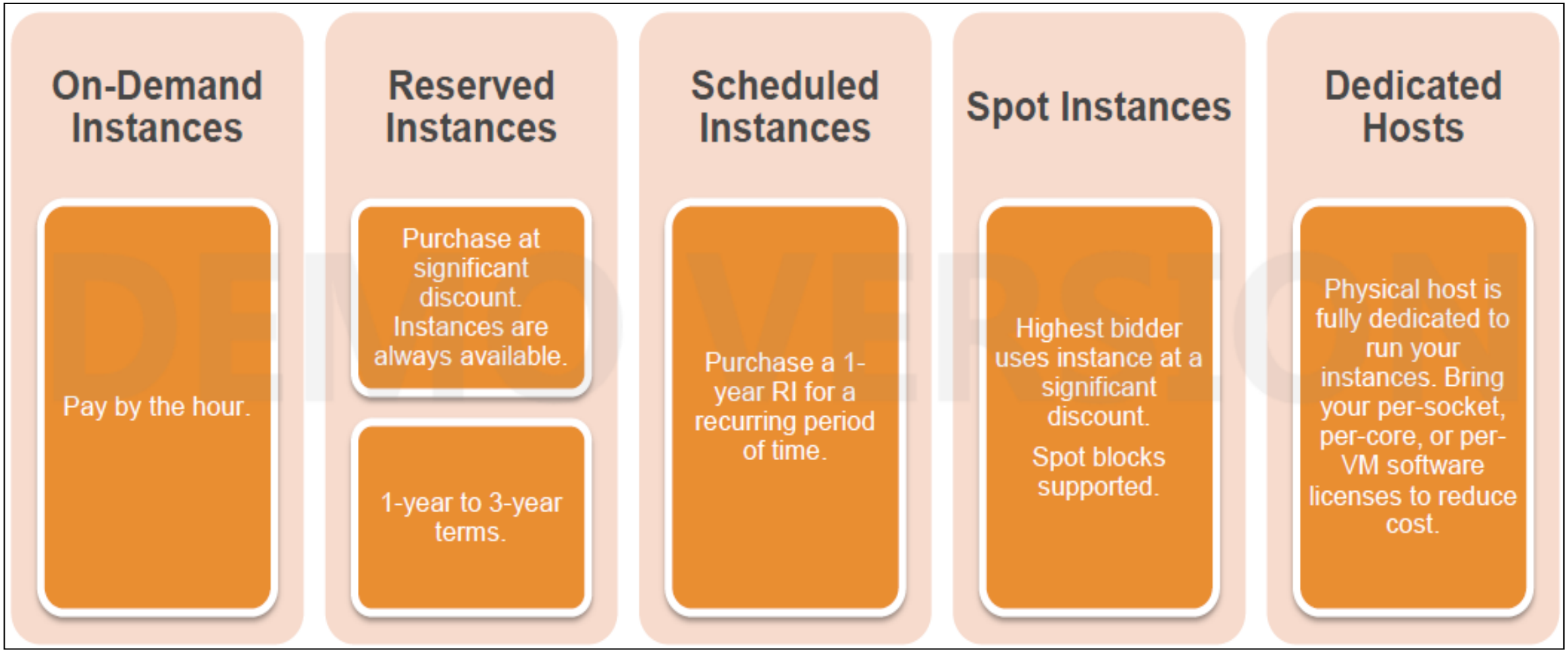
- Amazon EC2 stands for Elastic Compute Cloud, and is the Primary AWS web service.
- Provides Resizable compute capacity
- Reduces the time required to obtain and boot new server instances to minutes
- There are two key concepts to Launch instances in AWS:
  - Instance Type
  - AMI

- EC2 Facts:
  - Scale capacity as your computing requirements change
  - Pay only for capacity that you actually use
  - Choose Linux or Windows OS as per need. You have to Manage the OS and Security of same.
  - Deploy across AWS Regions and Availability Zones for reliability/HA

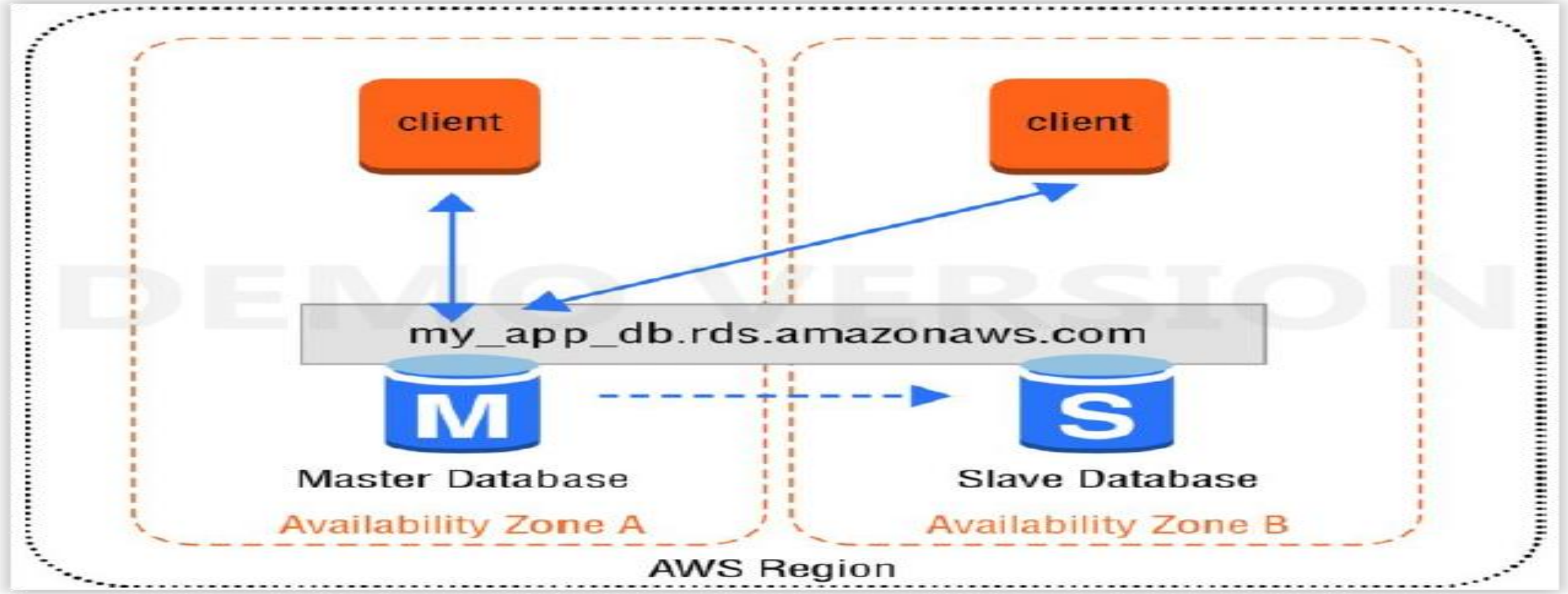
AWS EC2


General purpose
Compute optimized
Storage and IO optimized
GPU enabled
Memory optimized
M3
C3
I2
HS1
G2
R3
M1
C1
CC2
HI1
CG1
M2
CR1

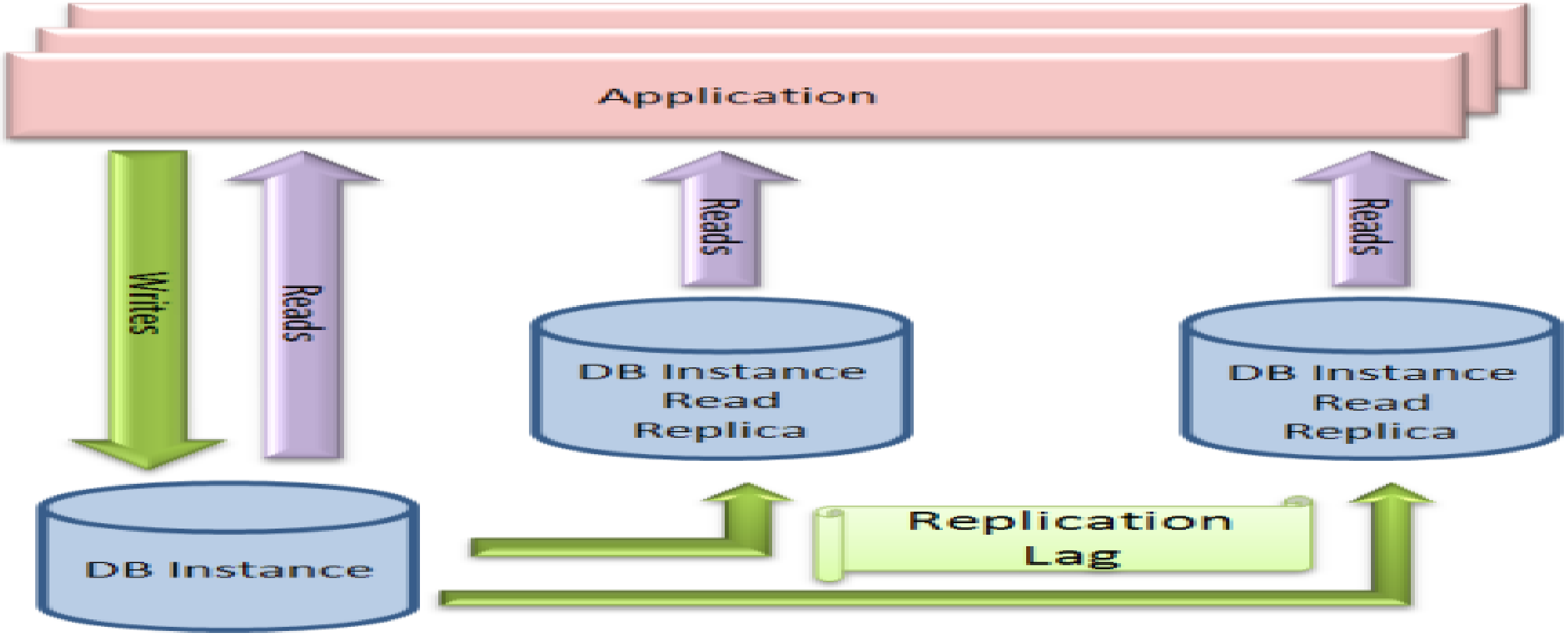
AWS EC2 Pricing



PaaS Example



PaaS Example



AWS CLI

Resource Management

- **AWS Management Console**
- AWS Console Mobile App (View resources)
- **AWS Command line interface**
- AWS Toolkit for PowerShell
- **AWS SDK (Software Development Kit's)**
- REST API/ SOAP / Query
- AWS-Shell (In preview)

AWS CLI

AWS CLI is a command-based utility to manage AWS resources

The primary distribution method for the AWS CLI on Linux, Windows, and macOS is pip, a package manager for Python that provides an easy way to install, upgrade, and remove Python packages and their dependencies

- http://docs.aws.amazon.com/cli/latest/userguide/installing.html

- **Requirements**
  - Python 2 version 2.6.5+ or Python 3 version 3.3+
  - Windows, Linux, macOS, or Unix
  - Pip package should be present (else install python-pip)

- Install AWSCLI: **pip install awscli --upgrade --user**
- For Windows, directly download the Windows installer from CLI webpage

AWS CLI

- Lets install an AWSCLI
  - https://aws.amazon.com/cli
- **aws --version**
- aws help
- aws ec2 help / aws s3 help / aws <anysubcommand> help
- Configure your default keys and region:

```
root@ip-172-31-28-145:~# aws configure
AWS Access Key ID [None]: ##########
AWS Secret Access Key [None]: ##############
Default region name [None]: us-west-2
Default output format [None]:
root@ip-172-31-28-145:~#
```

AWS CLI

Setting up credentials via env variable.

- export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
- export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
- export AWS_DEFAULT_REGION=us-west-2

LAB: AWS CLI

Check the details for all running instances using CLI

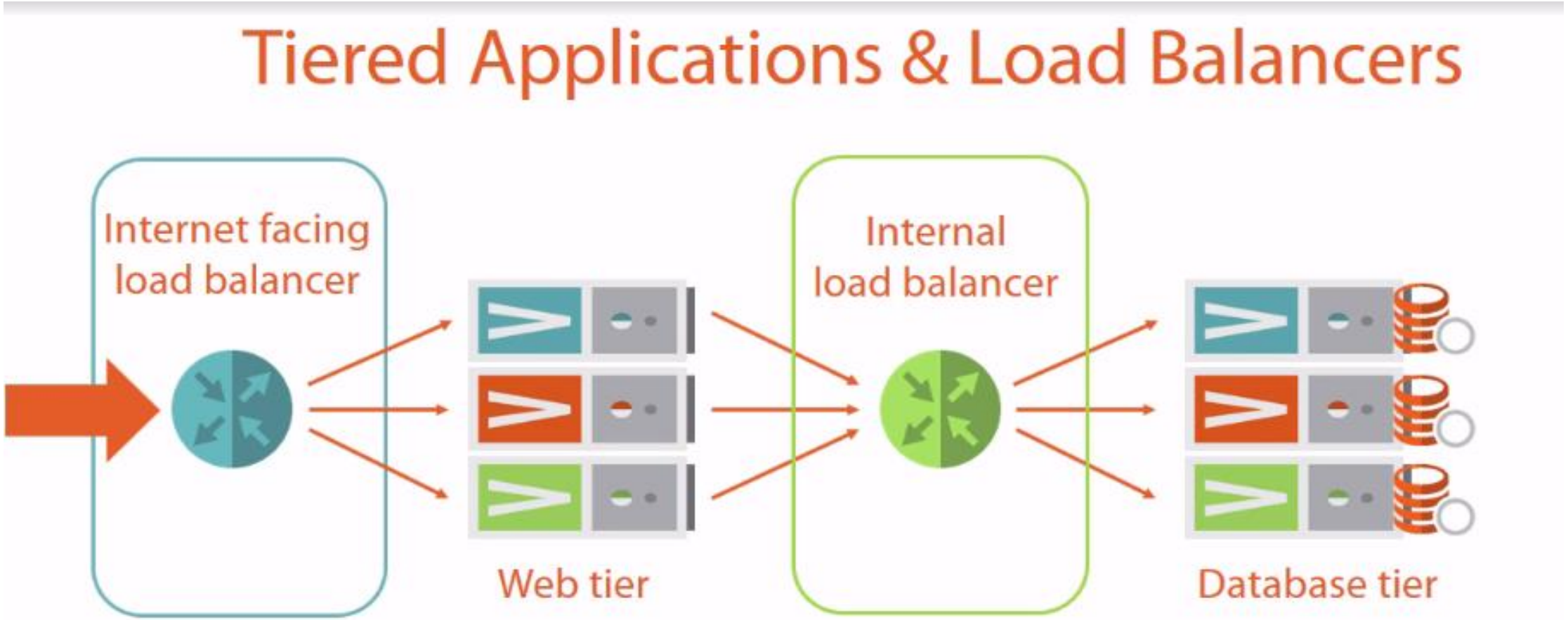
- aws ec2 describe-instances

Creation of an AWS Instance using CLI:

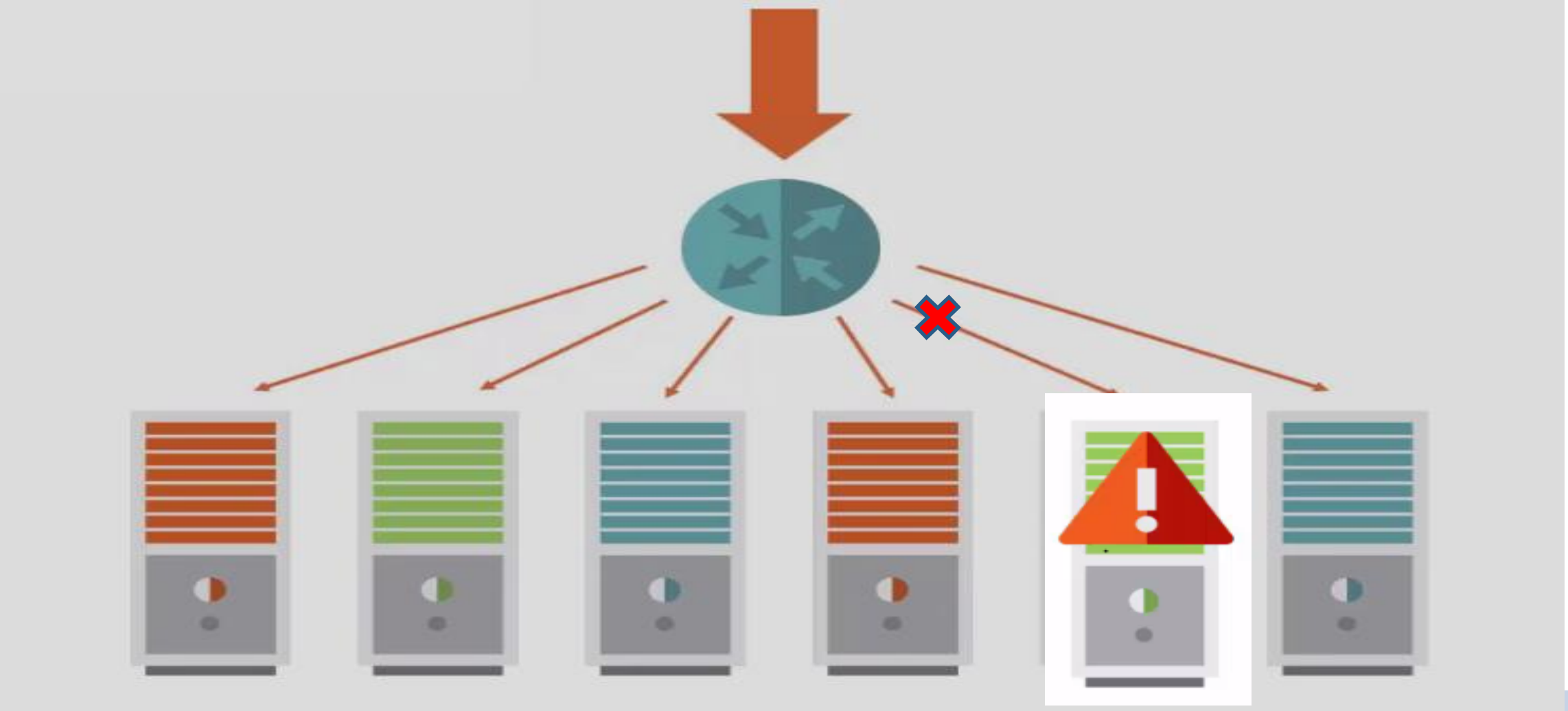
- aws ec2 run-instance help
- aws ec2 run-instances --image-id ami-76d6f519 --instance-type t2.micro --key-name test
- aws ec2 describe-instances
- aws ec2 stop-instances --instance-ids i-02e6b6c6c4dd3bbe0
- aws ec2 terminate-instances --instance-ids i-0297acea9e1b39a56

AWS ELB

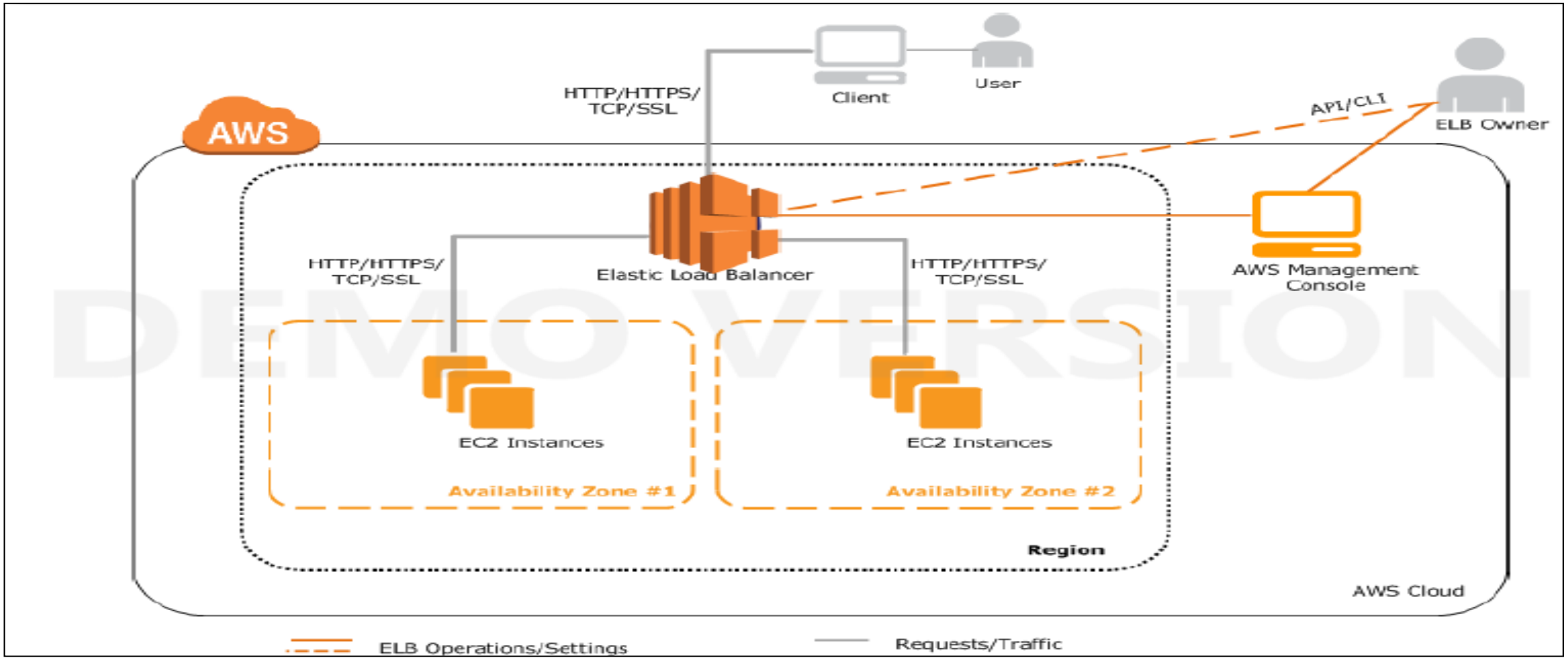
Load Balancer



Load Balancer

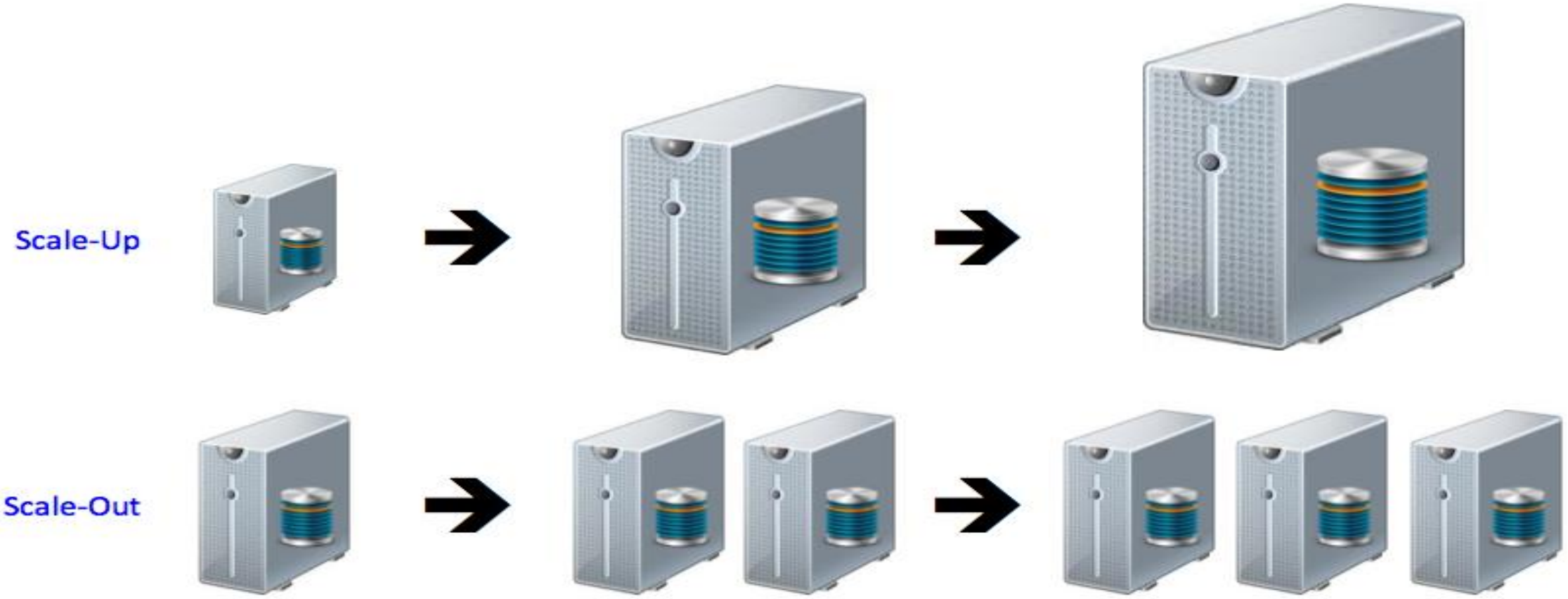


ELB

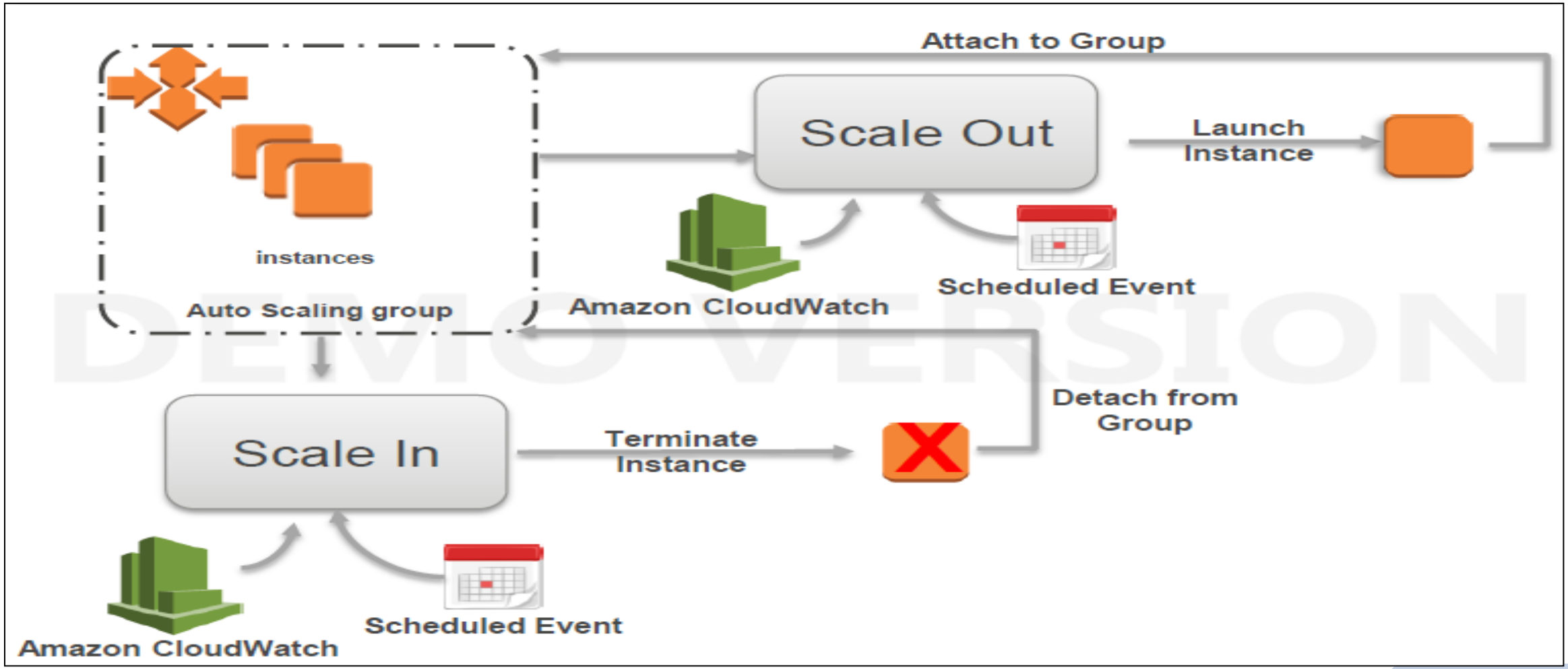


AWS AutoScaling

Auto Scaling

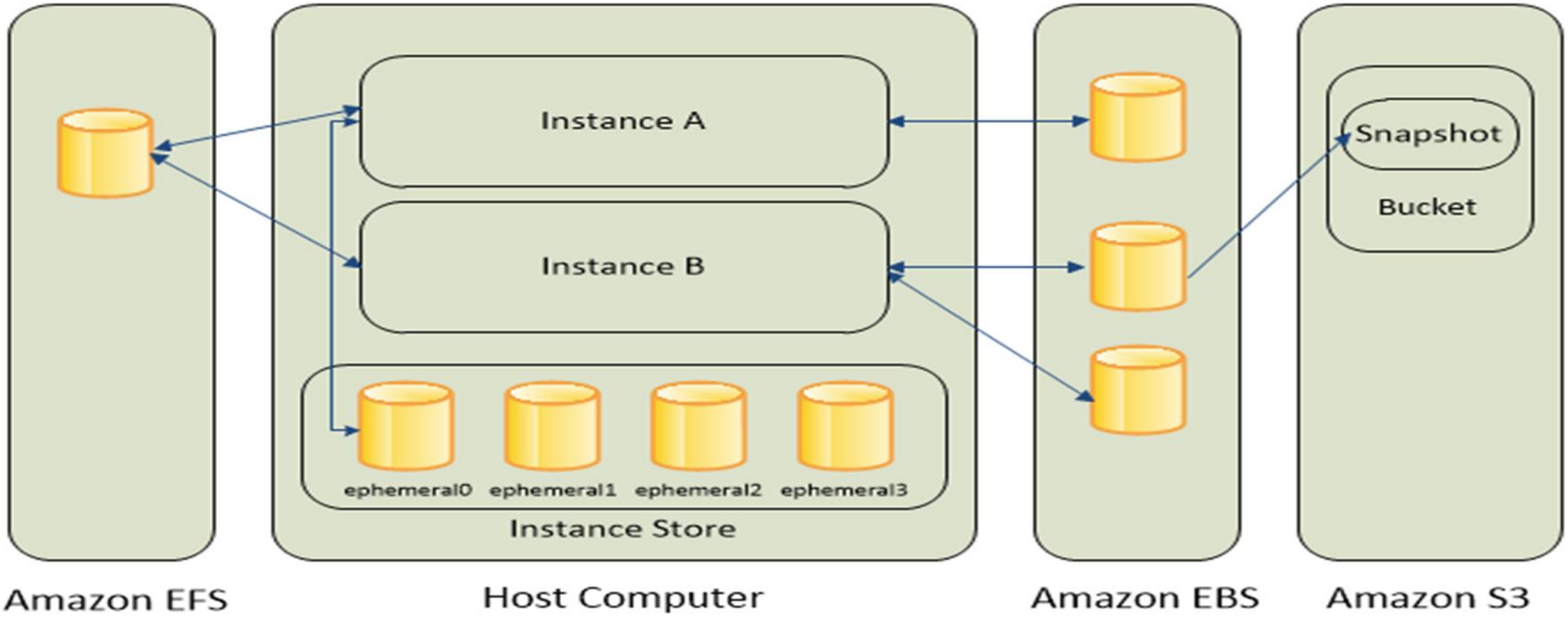


Auto Scaling



AWS Storage Services

Storage Services



S3 Storage Services

Able to store an unlimited number of objects in a bucket.

Objects up to 5 TB; no bucket size limit.

Designed for 99.999999999% durability and 99.99% availability of objects over a given year.

HTTP/S endpoint to store and retrieve any amount of data, at any time, from anywhere on the web.

Highly scalable, reliable, fast, and inexpensive.

Optional server-side encryption using AWS or customer- managed provided client-side encryption.

Amazon S3 objects are automatically replicated on multiple devices in multiple facilities within a region.

S3 Use Cases

Backup and archive for on-premises or cloud data

Content, media, and software storage and distribution

Big data analytics

Static website hosting

Software Delivery

Store AMIs and Snapshots

Cloud-native mobile and Internet application hosting

Disaster Recovery

S3 Concepts

Buckets: A bucket is a container (web folder) for objects (files) stored in Amazon S3. Every Amazon S3 object is contained in a bucket. Buckets form the top-level namespace for Amazon S3 and bucket names are global.

AWS Regions: Amazon S3 bucket is created in a specific region that you choose. You control the location of your data; data in an Amazon S3 bucket is stored in that region unless you explicitly copy it to another bucket located in a different region.

Objects: Objects are the entities or files stored in Amazon S3 buckets. Objects can range in size from 0 bytes up to 5TB, and a single bucket can store an unlimited number of objects.

AWS VPC - Network Services

VPC

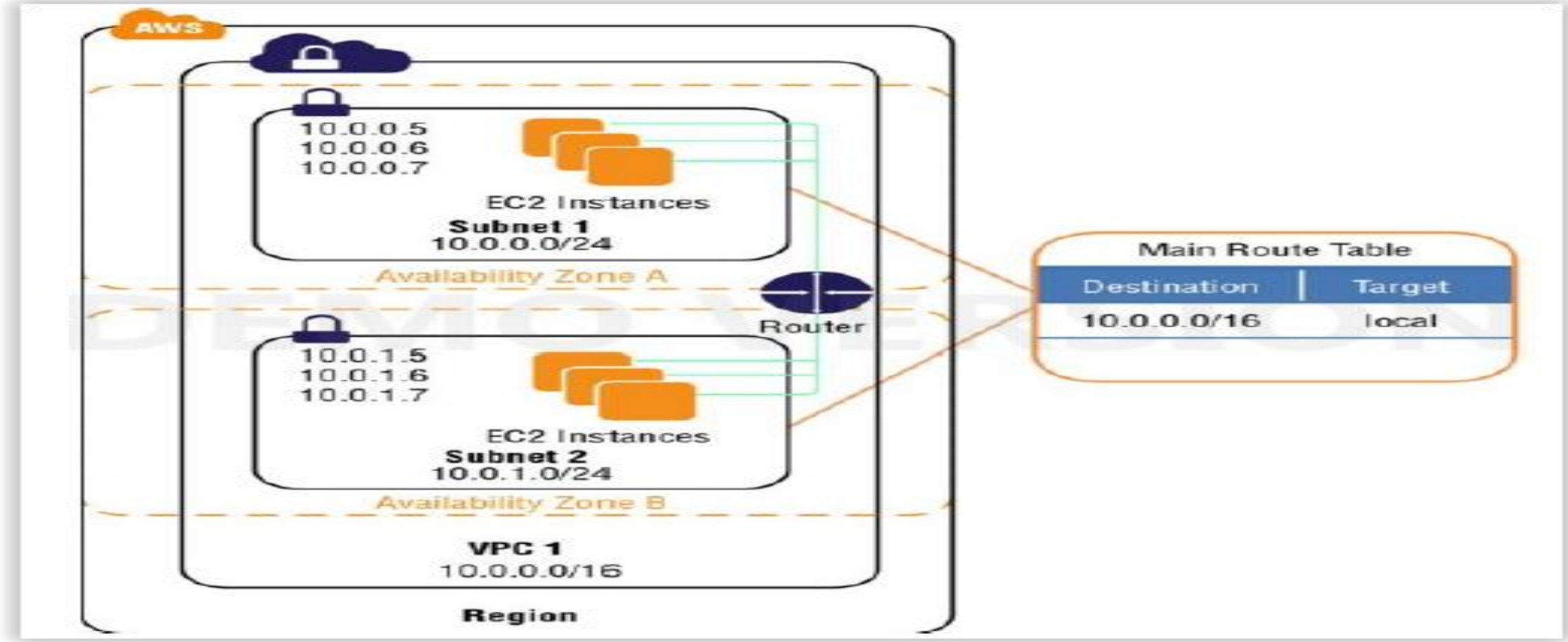
VPC stands for Virtual Private Cloud. Provision a **private, isolated virtual network** on the AWS cloud.

Amazon VPC is the networking layer for Amazon Elastic Compute Cloud (Amazon EC2), and it allows you to build your own virtual network within AWS.

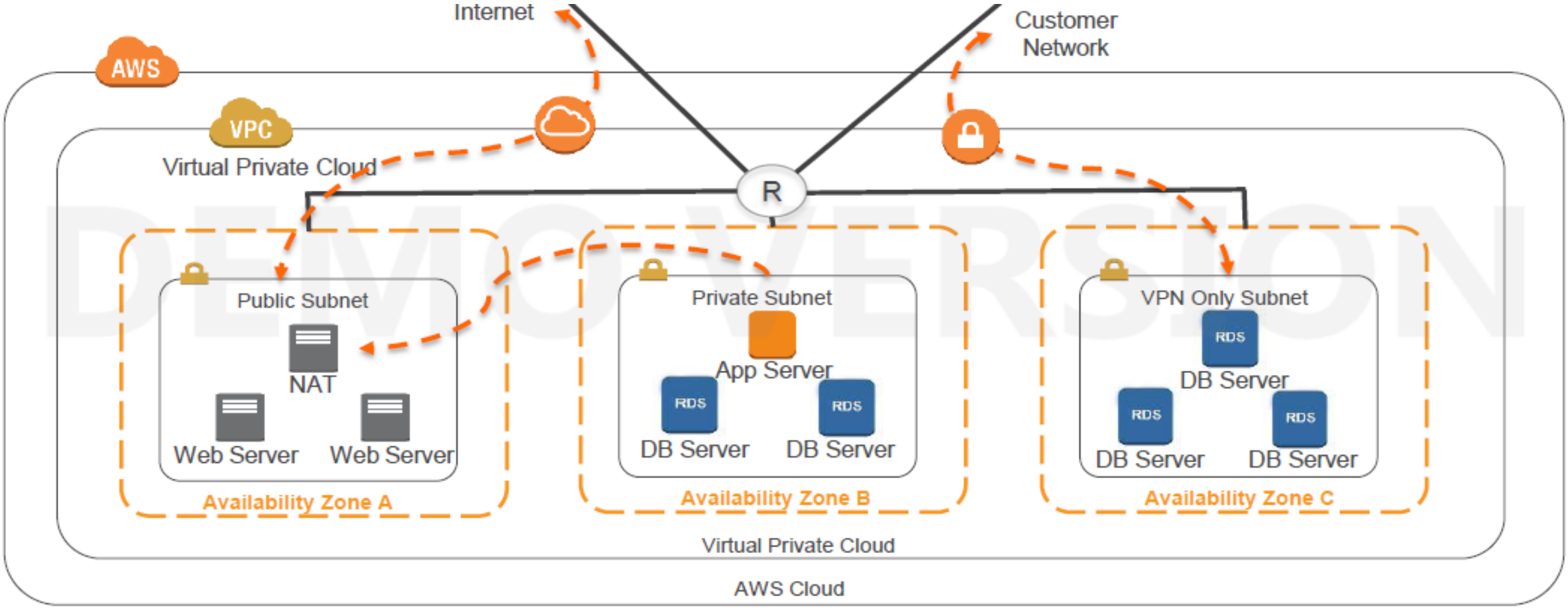
Within a region, you can create multiple Amazon VPCs, and each Amazon VPC is logically isolated.

Think of VPC as your Separate Datacenter with multiple VLANs (subnets can be treated like VLans here).

VPC


AWS
10.0.0.5
10.0.0.6
10.0.0.7
EC2 Instances
Subnet 1
10.0.0.0/24
Availability Zone A
Router
10.0.1.5
10.0.1.6
10.0.1.7
EC2 Instances
Subnet 2
10.0.1.0/24
Availability Zone B
VPC 1
10.0.0.0/16
Region
Main Route Table
Destination
Target
10.0.0.0/16
local

VPC/Subnets



Network Access Control List

A network access control list (ACL) is another layer of security that acts as a stateless firewall on a subnet level.

A network ACL is a numbered list of rules that AWS evaluates in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL.

Operates at the subnet level (Another layer of defense).

NACL is default allow.

EC2 Security Group

Security Group is a Virtual Firewall Protection.

AWS allows you to control traffic in and out of your instances through virtual firewalls called security groups.

Security groups allow you to control traffic based on port, protocol, and source(inbound)/destination(outbound).

Security groups are associated with instances when they are launched. Every instance must have at least one security group. Though they can have more.

A security group is default deny.

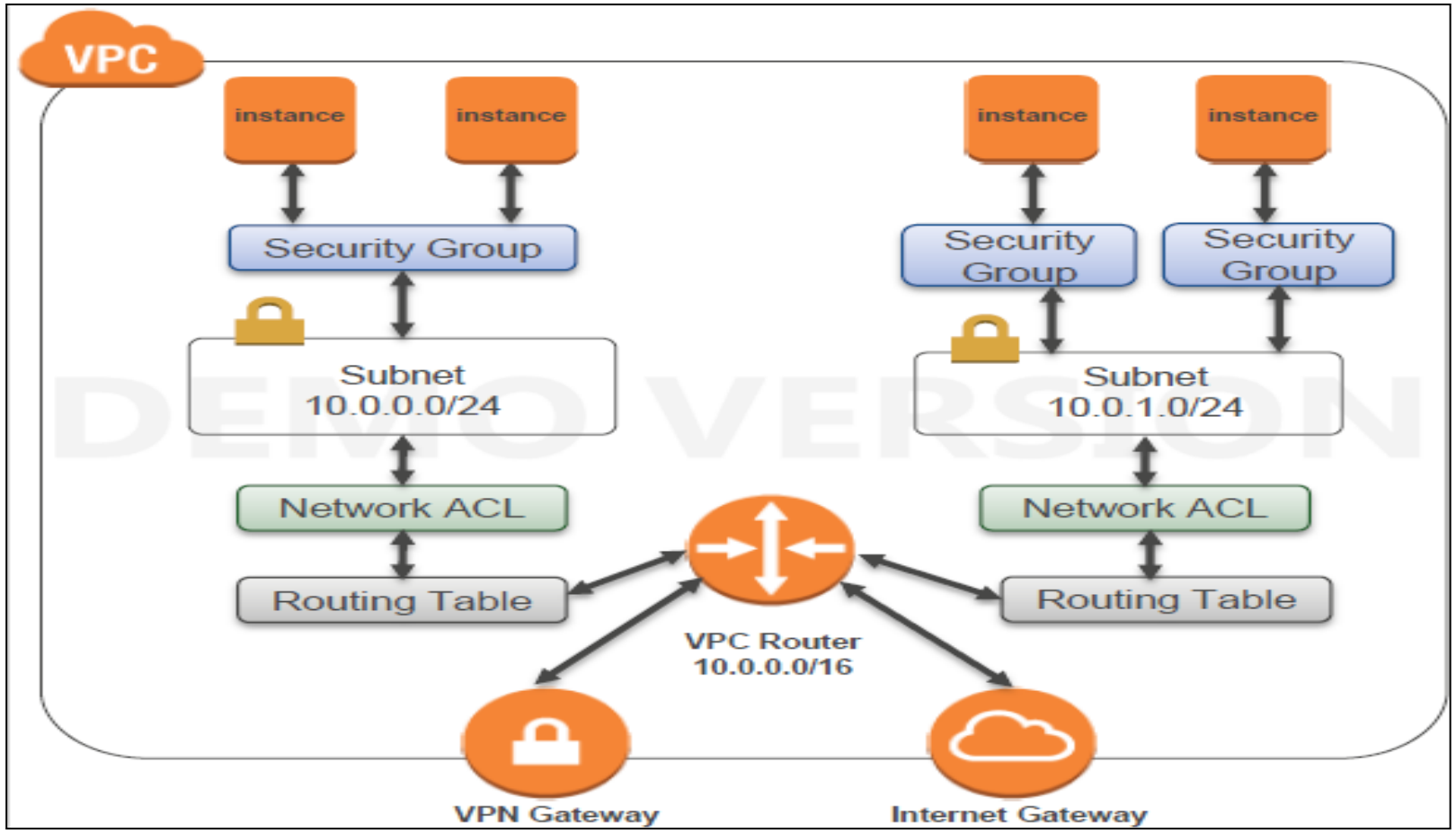
Security Layers

OS Firewall

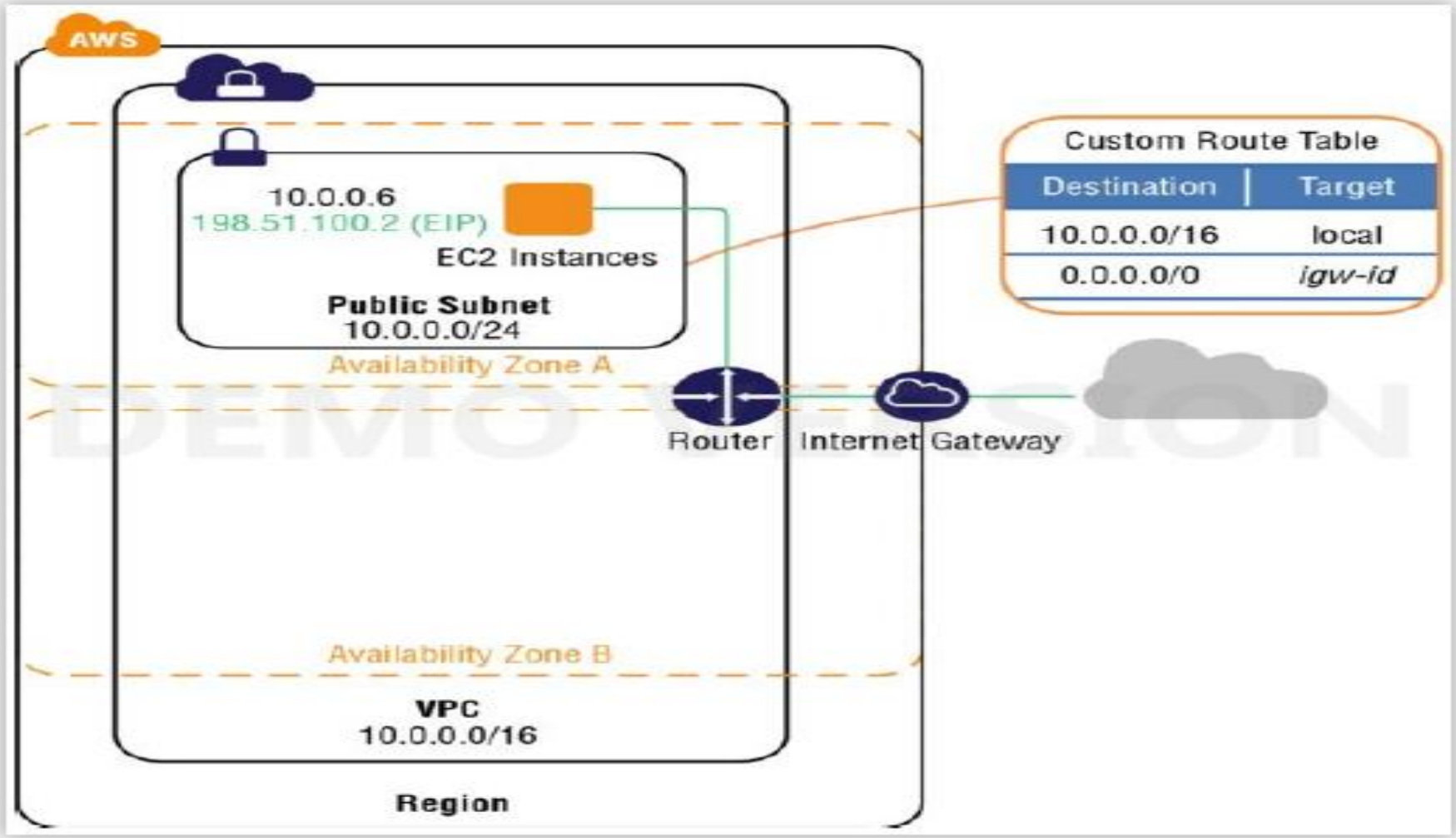
Security Group

Network ACL

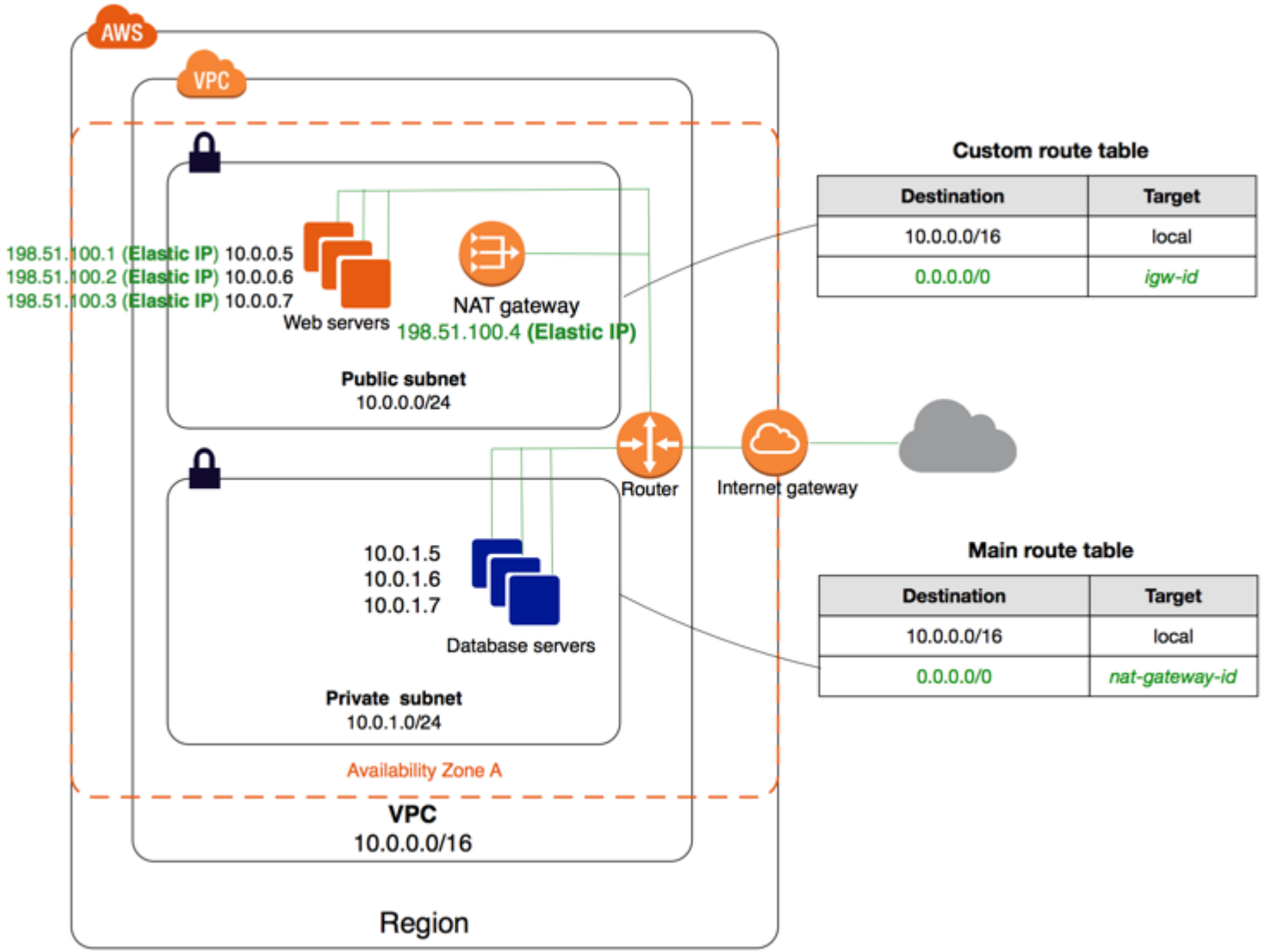
Routing Tables



Connectivity



NAT Gateway



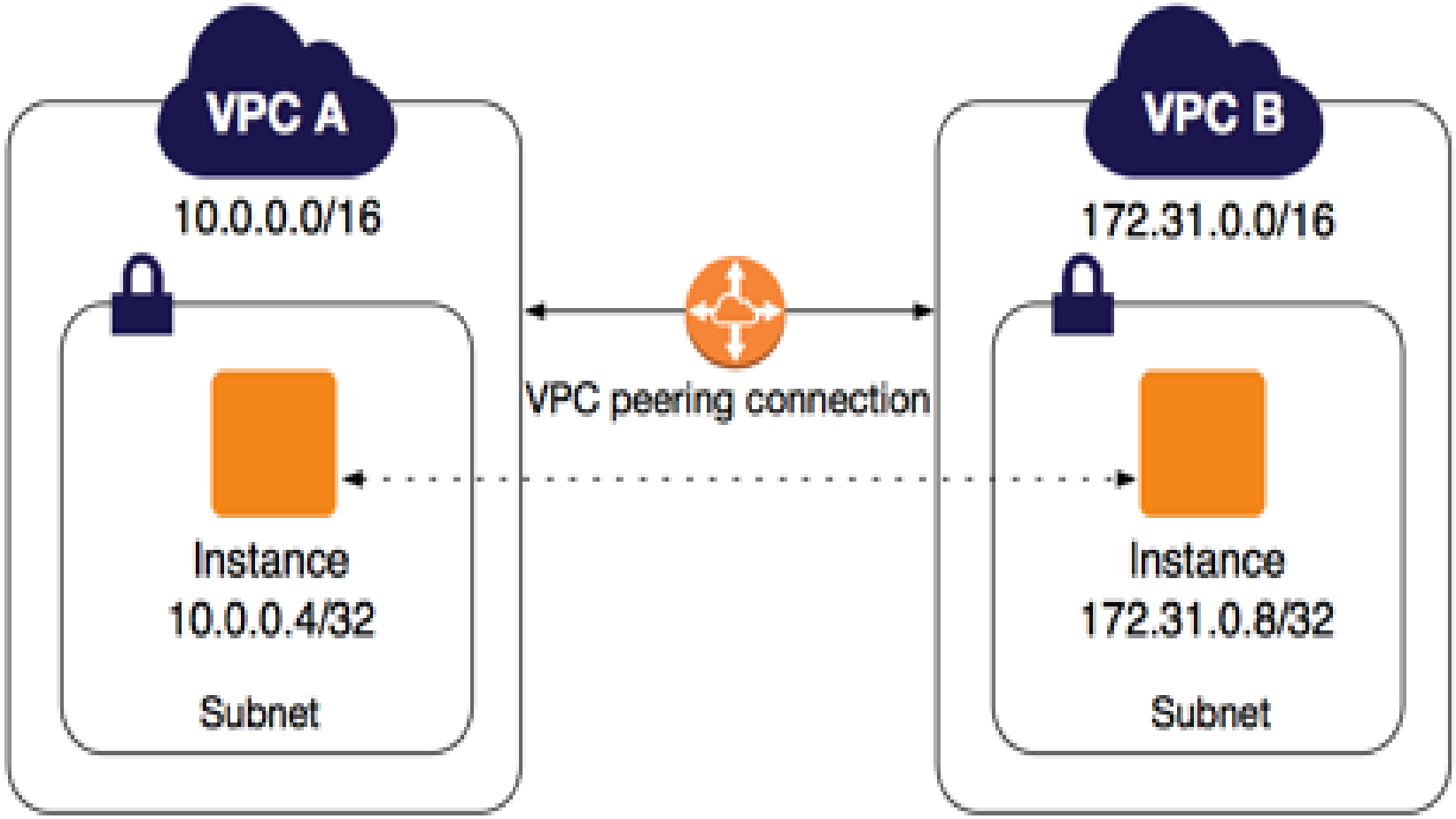
Custom route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*igw-id*

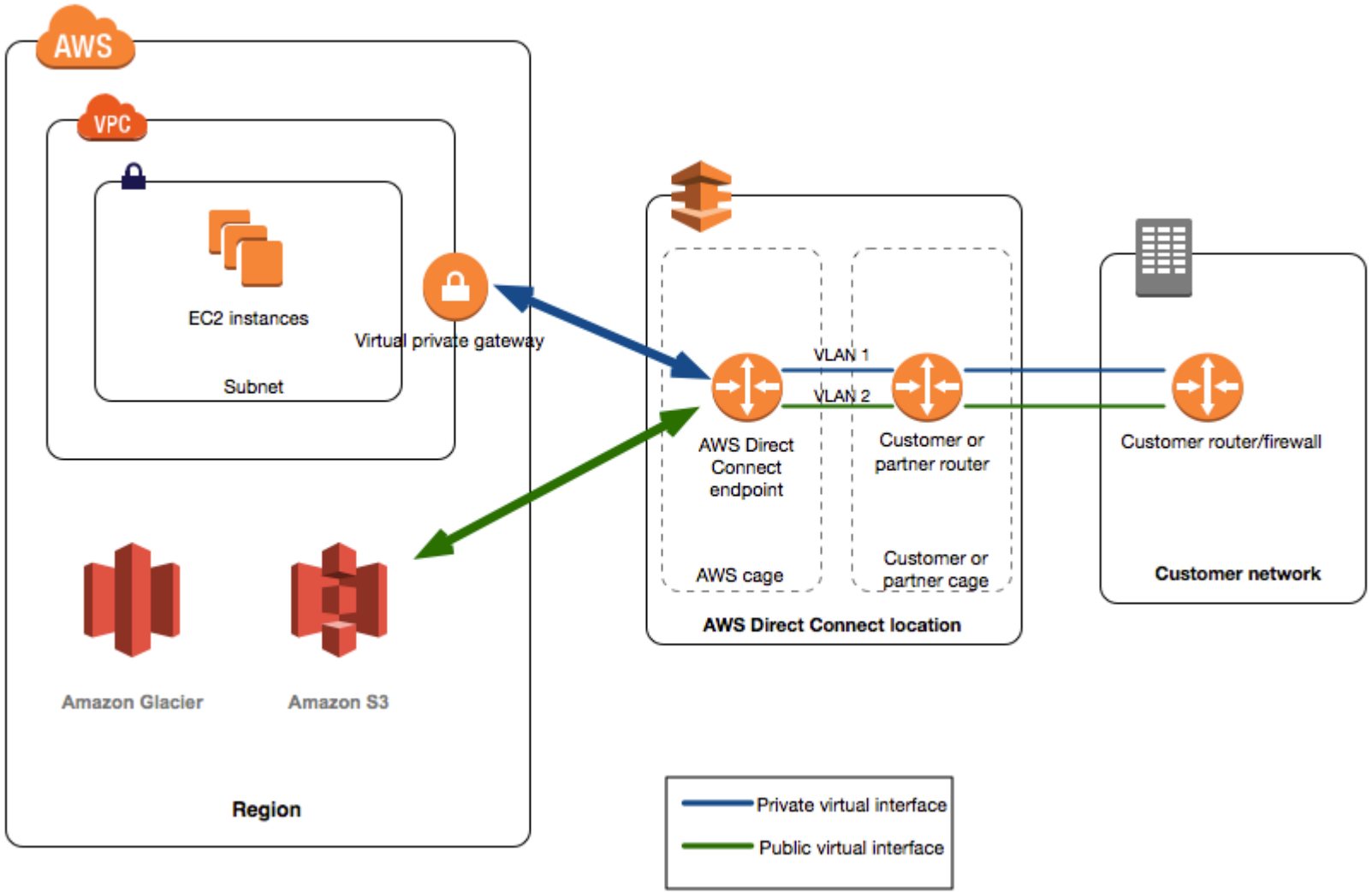
Main route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*nat-gateway-id*

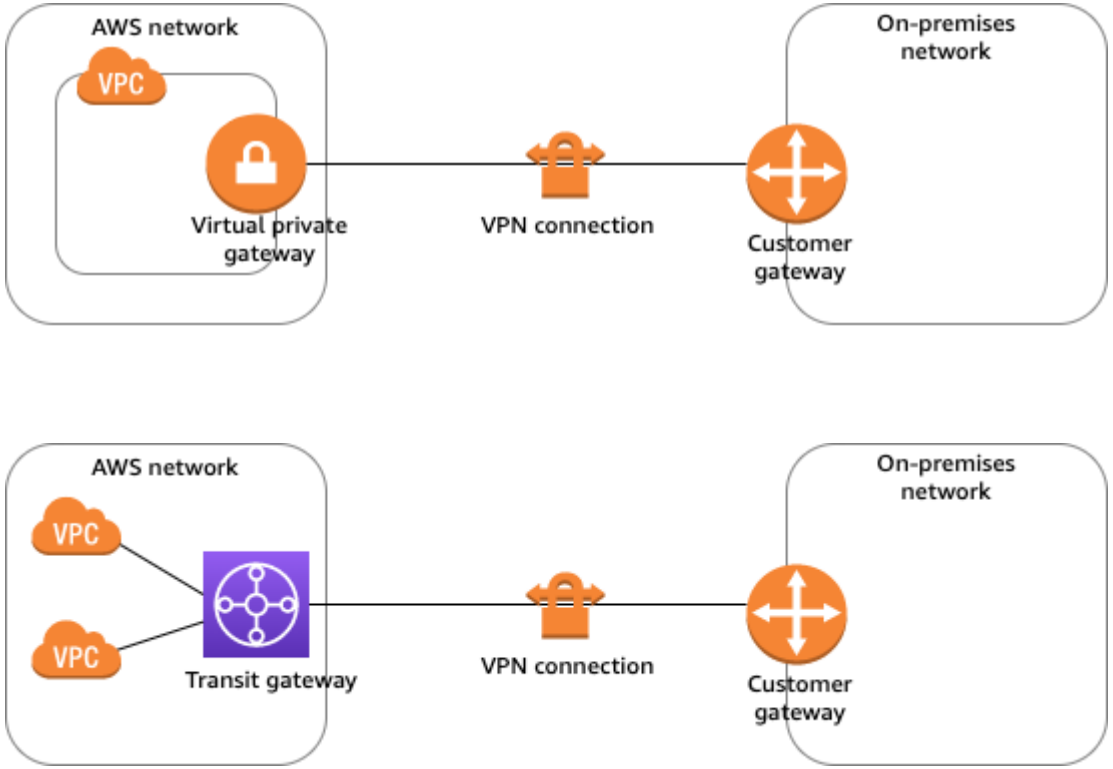
VPC Peering


VPC A
10.0.0.0/16
VPC B
172.31.0.0/16
VPC peering connection
Instance
10.0.0.4/32
Subnet
Instance
172.31.0.8/32
Subnet

AWS Direct Connect



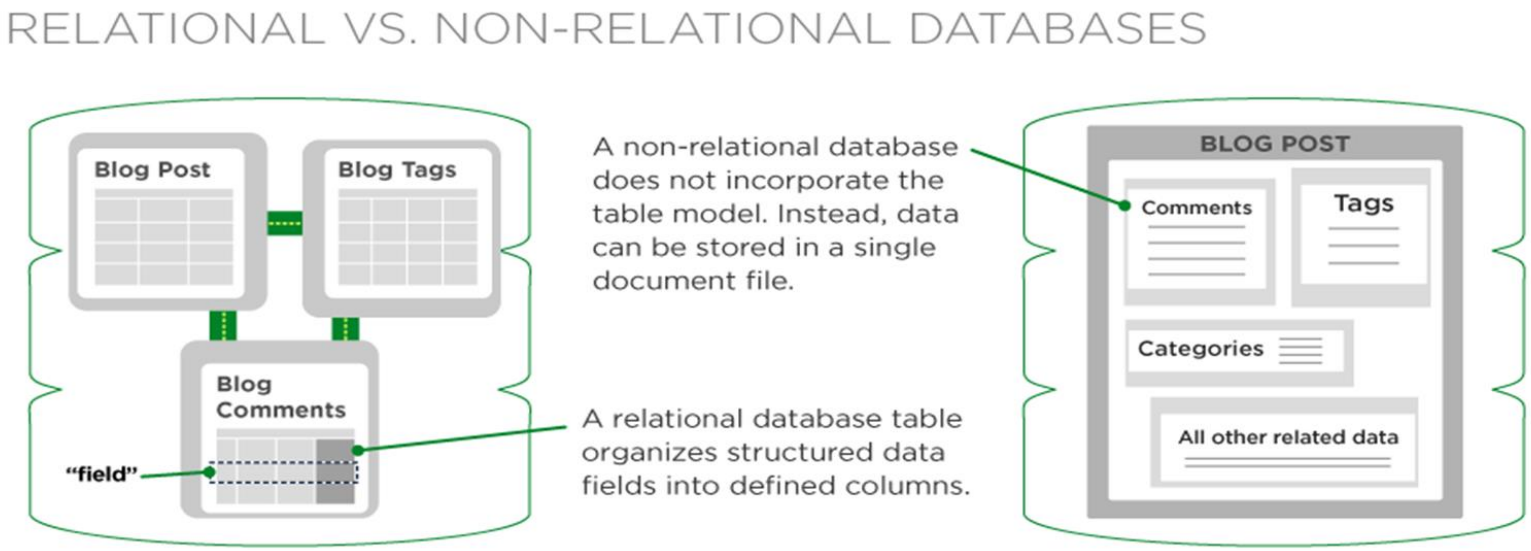
AWS VPN Connect



DB Design, Constraint & Enterprise Integration patterns

Database

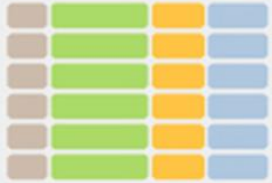
In the world of database technology, there are two main types of databases: **SQL** and **NoSQL** or in other terms **relational** databases and **non-relational** databases. The difference speaks to how they're built, the type of information they store, and how they store it.



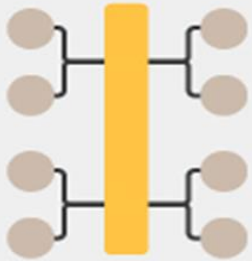
Database

SQL Database

Relational



Analiticals (OLAP)



Non-SQL Database

Key-Value

Column-Family

Graph

Document

Relational Database

- Relational Databases are the oldest and widely use databases
- A Relational database consists of one or more tables.
- A Relational database can be categorized as either an Online Transaction Processing (OLTP) or Online Analytical Processing (OLAP) database system.
- Support **ACID (Atomicity, Consistency, Isolation, Durability)** transactions
- Example: SQL database, Microsoft SQL Server, Oracle Database, IBM DB2, MY SQL, Maria DB, Sybase, PostgreSQL

OLTP vs OLAP

BASIS FOR COMPARISON	OLTP	OLAP
Basic	It is an online transactional system and manages database modification.	It is an online data retrieving and data analysis system.
Focus	Insert, Update, Delete information from the database.	Extract data for analyzing that helps in decision making.
Data	OLTP and its transactions are the original source of data.	Different OLTPs database becomes the source of data for OLAP.
Transaction	OLTP has short transactions.	OLAP has long transactions.
Time	The processing time of a transaction is comparatively less in OLTP.	The processing time of a transaction is comparatively more in OLAP.
Queries	Simpler queries.	Complex queries.
Integrity	OLTP database must maintain data integrity constraint.	OLAP database does not get frequently modified. Hence, data integrity is not affected.
Size	Small – usually in GBs	Large – usually in TBs

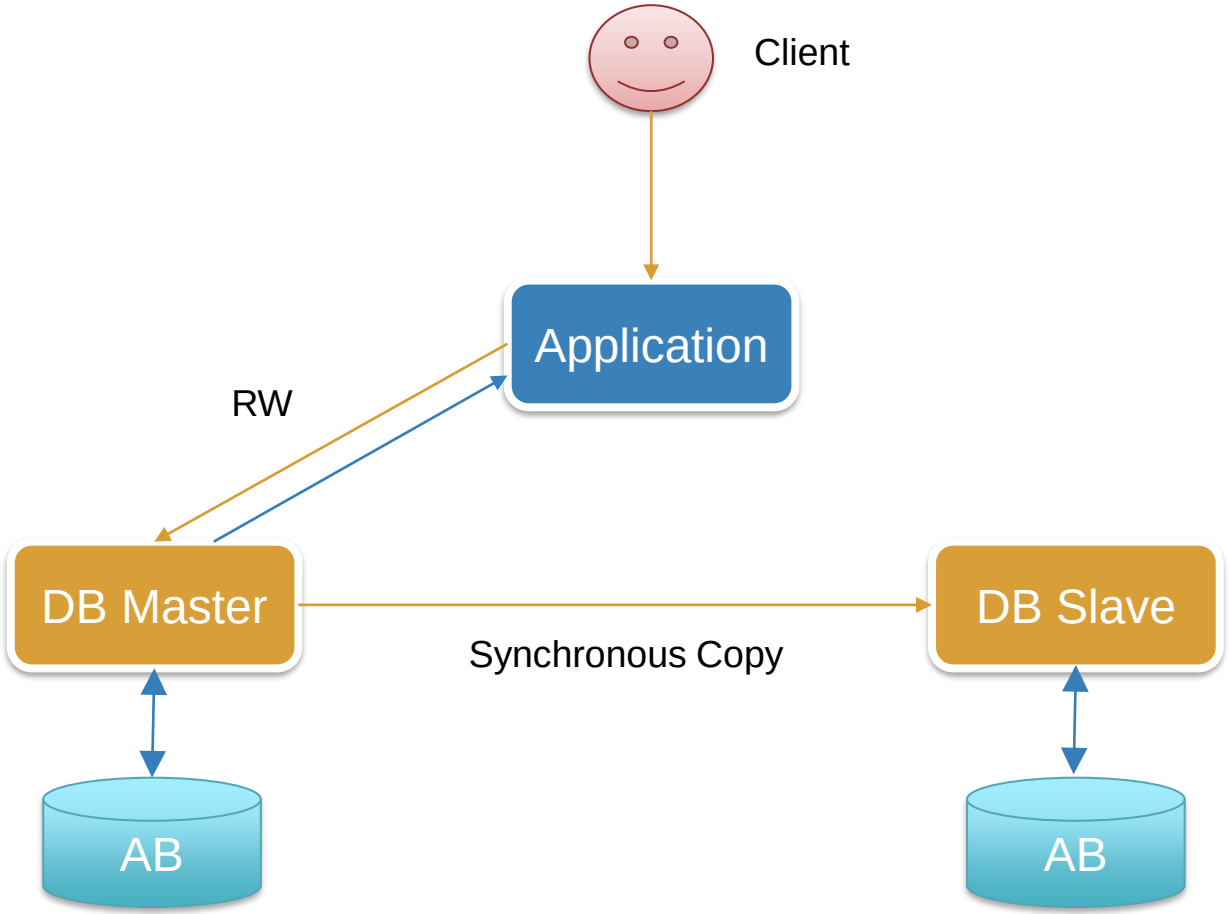
Non-Relational Database

- Recently came in limelight due to its unmatched benefits
- Good to use, If your data is unstructured and unpredicted. For example, requirements aren't clear at the outset, or you don't have a clearly defined schema .
- Overcome the limitations of SQL databases e.g., horizontal scaling, parallel query performance, replication etc.
- Multiple benefits over SQL, including **High concurrency, high volume random reads and writes, massive DB sizes**, **supporting unstructured** data, HA at low cost etc.
- Support **eventual consistency** instead of ACID transactions.
- **Example:** MongoDB, Cassandra, neo4j, Redis, Couchbase, Elasticsearch

Databases

	SQL	NoSQL
Performance	Low	High
Transactions	Atomic	Eventual Consistency
Consistency	Good	Poor
Reliability	Good	Poor
Supported DB Size	Mid-to-large size	Supports huge Database
Scalability	High (expensive)	High
Supported Datatype	Structured	Both Structured and Unstructured
Scaling method	Scale Up	Scale Out

Master Slave Architecture



AWS Relational Databases (PaaS)

Amazon **Relational Database Service (Amazon RDS)** is a service which makes it very easy to set up, operate, and scale relational databases in the cloud. With Amazon RDS you can launch one of many popular database engines in just a matter of few minutes.

Amazon RDS is a service that simplifies the setup, operations, and scaling of a relational database on AWS. With Amazon RDS, you can spend more time focusing on the application and the schema and let Amazon RDS offload common tasks like backups, patching, scaling, and replication.

Amazon Relational Database Service (Amazon RDS) significantly simplifies the setup and maintenance of OLTP and OLAP databases.

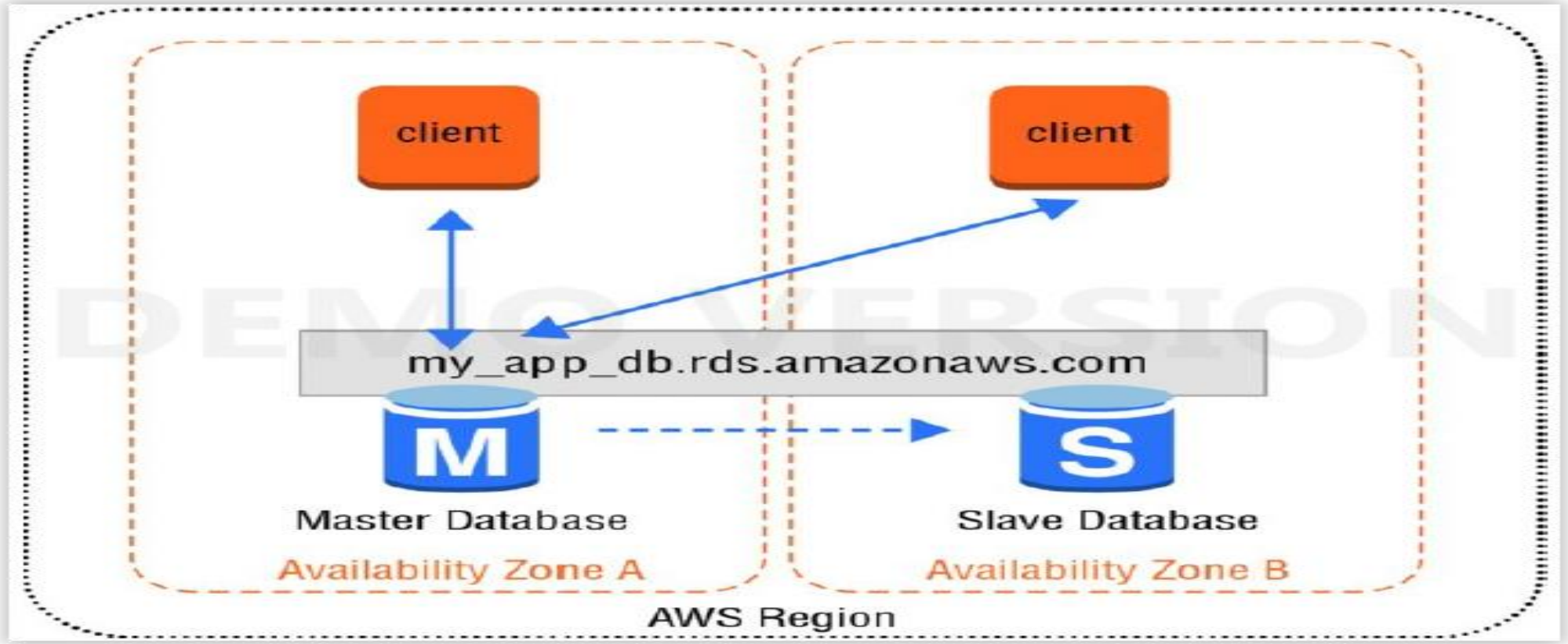
Amazon RDS provides support for six popular relational database engines: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MariaDB, and **Amazon Aurora**.

LAB: RDS

Create a MySQL Amazon RDS Instance

1. Log in to the AWS Management Console, and navigate to the Amazon RDS Console.
2. Launch a new Amazon RDS DB Instance, and select MySQL instance as the database engine.
3. Configure the DB Instance to use “Dev/Test – MySQL”. On next screen select General Purpose (SSD) storage – 20 Gb.
4. Set the DB Instance identifier and database name to **MySQL123**, and configure the master username and password.
5. Validate the configuration settings, and launch the DB Instance.
6. Return to the list of the Amazon RDS instances. You will see the status of your Amazon RDS database as Creating. It may take up to 20 minutes to create your new Amazon RDS instance.

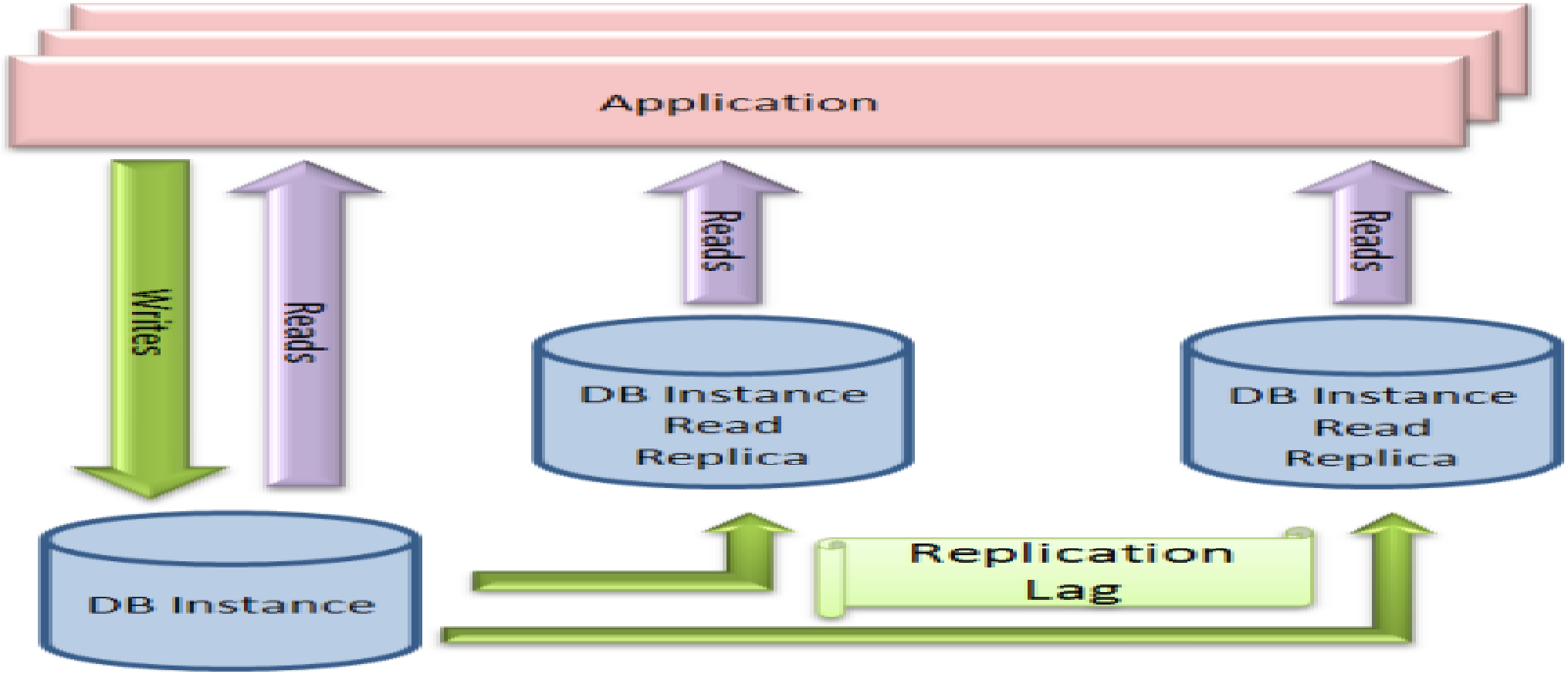
RDS Multi-AZ deployment



AWS RDS Read Replica

- Read Replicas are readable copies of your databases. Good to have same for read heavy databases.
- It can serve read traffic while the source DB instance is unavailable (for example, due to I/O suspension for backups or scheduled maintenance).
- **Business reporting** or data warehousing scenarios where you might want business reporting queries to run against a Read Replica, rather than your primary, production DB instance.
- You can use promote a Read Replica to a standalone instances as a **disaster recovery solution** if the source DB instance fails.
- Amazon RDS doesn't support circular replication.
- You can promote Read replicas to create Individual independent databases. It'll break the link between databases and will create this Read replica as individual database instance.
- Cross-region replication is possible.

AWS RDS Read Replica



Multi-AZ Deployment vs Read-Replica

Multi-AZ Deployments	Read Replicas
Synchronous replication – highly durable	Asynchronous replication – highly scalable
Only database engine on primary instance is active	All read replicas are accessible and can be used for read scaling
Automated backups are taken from standby	No backups configured by default
Always span two Availability Zones within a single Region	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Database engine version upgrades happen on primary	Database engine version upgrade is independent from source instance
Automatic failover to standby when a problem is detected	Can be manually promoted to a standalone database instance

Amazon Aurora

AWS Aurora

Amazon Aurora (Aurora) is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine. It provides increased reliability and performance over standard MySQL deployments.

Amazon Aurora can deliver up to five times the performance of MySQL and upto three time of PostgreSQL, without requiring changes to most of your existing web applications. You can use the same code, tools, and applications that you use with your existing MySQL databases with Amazon Aurora.

Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL and Amazon RDS for PostgreSQL applications to Aurora.

Amazon Aurora RDS Cluster, also support Autoscaling of DB instances.

The underlying storage grows automatically as needed, up to 64TB. It uses a single cluster volume which is spanned/replicated across AZ in one region and uses SSDs in backend.

AWS Aurora

Amazon Aurora (Aurora) is a fully managed, PaaS service. So all other RDS features are available in Aurora as well.
Four types of Endpoints available: Cluster endpoint, Reader endpoint, Custom endpoint & Instance endpoint.
Because of shared storage in backend, data will remain independent from the DB instances and failover, new DB instance creation etc. tasks are performed much faster.
Storage will be billed as per usage with "high water mark“.
Additional benefits like Storage Auto-Repair, Survivable Cache Warming, Crash Recovery, Replica Autoscaling etc.
Replication across AZ in less than 1ms. Replication across Regions in less than 1s.
Replica Auto scaling feature is available in Aurora.
Serverless feature available with Aurora

AWS Aurora

Amazon Aurora (Aurora) is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine. It provides increased reliability and performance over standard MySQL deployments.

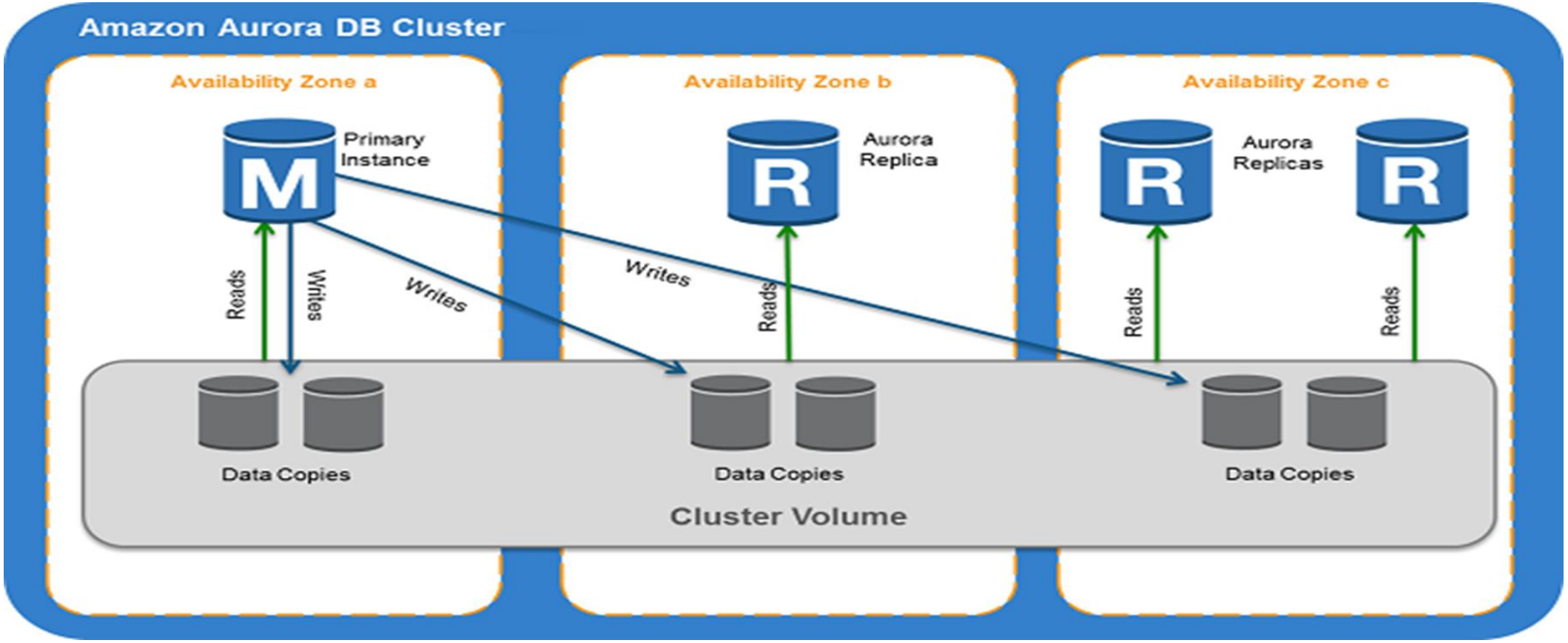
Amazon Aurora can deliver up to five times the performance of MySQL and upto three time of PostgreSQL, without requiring changes to most of your existing web applications. You can use the same code, tools, and applications that you use with your existing MySQL databases with Amazon Aurora.

Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL and Amazon RDS for PostgreSQL applications to Aurora.

Amazon Aurora RDS Cluster, also support Autoscaling of DB instances.

The underlying storage grows automatically as needed, up to 64TB. It uses a single cluster volume which is spanned/replicated across AZ in one region and uses SSDs in backend.

AWS Aurora


Amazon Aurora DB Cluster
Availability Zone a
Availability Zone b
Availability Zone c
M
Primary Instance
R
Aurora Replica
R
R
Aurora Replicas
Reads
Writes
Writes
Writes
Reads
Reads
Reads
Data Copies
Data Copies
Data Copies
Cluster Volume

RDS Limitations

Can not change some default DB settings i.e.:

- allow-suspicious-udfs for User defined anonymous functions
- basedir/data directory location for DB
- Handling core files/core-file settings (in case engine dies)

Default storage engines are limited and sometime only allowed InnoDB. Default it uses is innoDB.

Management of Log files

Own backup and monitoring tools can't be integrated.

Instance level access is un-available.

Mandatory upgrade once the DB version is unsupported by AWS.

Few DBs are available as BYOL, not all DBs and versions

Limited scope of OPTIONS available.

Migrating back from Aurora to MySQL/PostgreSQL can have implications and challenges.

AWS Redshift

Data Warehousing

Data Warehousing is used to extract data in periodic stages, or as they are generated, making it more efficient and simpler to process queries over data that actually came from different sources.

The raw data is turned into high-quality information to meet all enterprise reporting requirements and also for all levels of users.

The types of applications that are supported include OLAP (Online Analytical Processing), Data Mining and DSS (Decision Support System).

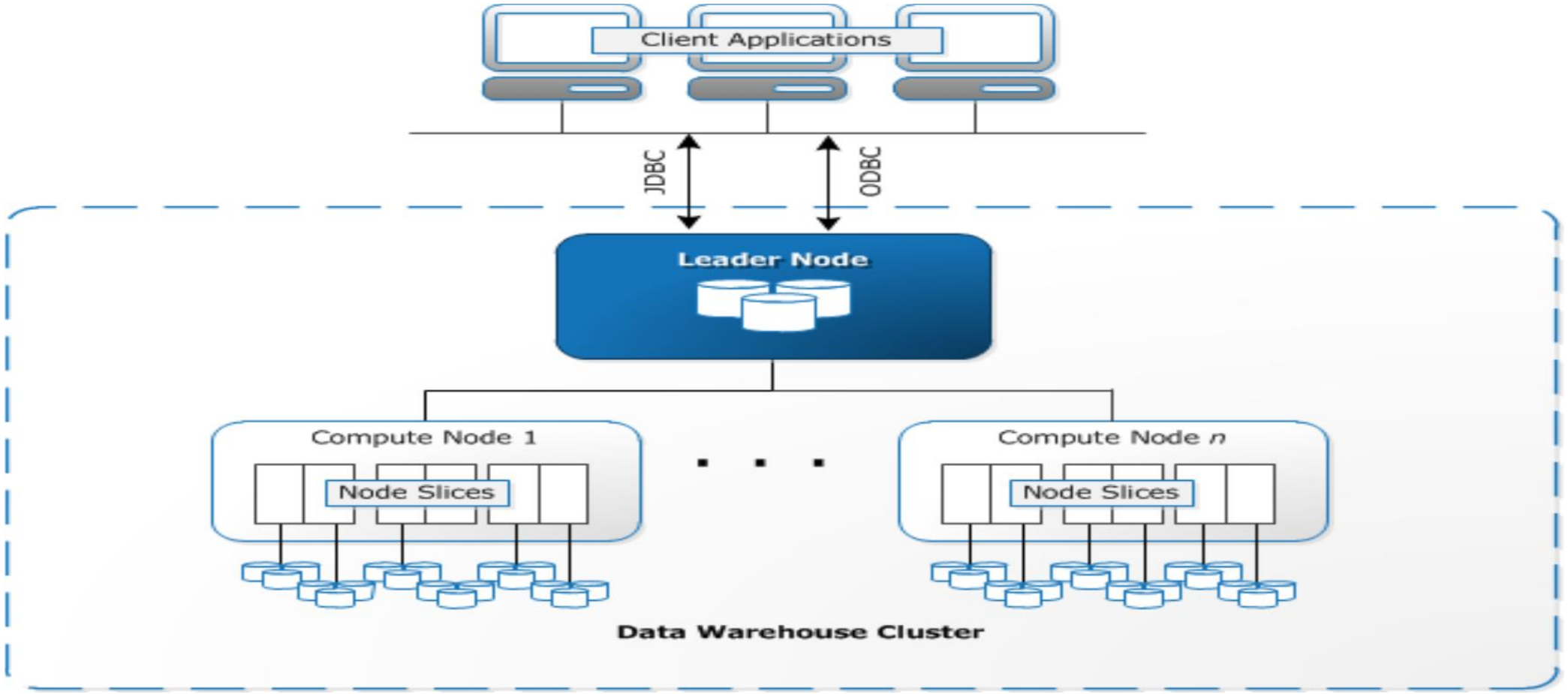
AWS Redshift

- Amazon Redshift is an Enterprise class, fast, powerful, Reliable, **fully managed, petabyte-scale data warehouse service** in the cloud.
- Released in February, 2013.
- Amazon Redshift is based on an **older version of PostgreSQL 8.0.2**, but have a number of important differences too.
- Amazon Redshift is a relational database **designed for OLAP scenarios**
- Optimized for high-performance analysis and reporting of very large datasets.
- Amazon Redshift gives you fast querying capabilities over structured data using standard SQL commands to support interactive querying over large datasets i.e. Data Mining and DSS (Decision Support System).

AWS Redshift

- Amazon Redshift manages the work needed to set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity to automating ongoing administrative tasks such as backups and patching (PaaS Service).
- Redshift helps you create OLAP DB cluster with multiple nodes, to store and process your Data.
- Amazing Redshift uses columnar storage technology in order to improve I/O efficiency and parallelized queries across multiple nodes and provide fast query performance.
- Amazon Redshift stores data in columns, using specialized data compression encodings for optimum memory usage and disk I/O.
- Redshift has an MPP (Massively Parallel Processing) architecture, distributing SQL operations and parallelizing techniques to take full advantage of all available resources.
- Distributed cluster for parallel processing.

AWS Redshift



AWS Redshift

Clusters and Nodes:

- The key component of an Amazon Redshift data warehouse is a cluster.
- A cluster is composed of a leader node and one or more compute nodes.
- The client application interacts directly only with the leader node, and the compute nodes are transparent to external applications.
- There will be minimum one Leader node and 1 Compute Node. (leader node is free of cost and automatically assigned based on number of Compute nodes)
- Redshift Node types are grouped into two categories: **Dense Compute and Dense Storage.**
- The Dense Compute node types support clusters up to 326TB using fast SSDs, while the Dense Storage nodes support clusters up to 2PB using large magnetic disks.
- 2-16 slices per compute node

AWS Redshift

Client applications

Amazon Redshift integrates with various data loading and ETL (extract, transform, and load) tools and business intelligence (BI) reporting, data mining, and analytics tools. Amazon Redshift is based on industry-standard PostgreSQL, so most existing SQL client applications will work with only minimal changes.

Connections

Amazon Redshift communicates with client applications by using industry-standard JDBC and ODBC drivers for PostgreSQL.

Clusters

A cluster is composed of one or more compute nodes. If a cluster is provisioned with two or more compute nodes, an additional leader node coordinates the compute nodes and handles external communication. Your client application interacts directly only with the leader node. The compute nodes are transparent to external applications.

AWS Redshift

Leader Node

The leader node manages communications with client programs and all communication with compute nodes. It parses and develops execution plans to carry out database operations, in particular, the series of steps necessary to obtain results for complex queries. Based on the execution plan, the leader node compiles code, distributes the compiled code to the compute nodes, and assigns a portion of the data to each compute node.

The leader node distributes SQL statements to the compute nodes only when a query references tables that are stored on the compute nodes. All other queries run exclusively on the leader node.

Compute Node

The compute nodes execute the compiled code and send intermediate results back to the leader node for final aggregation.

Each compute node has its own dedicated CPU, memory, and attached disk storage, which are determined by the node type.

AWS Redshift

Node slices

A compute node is partitioned into slices. Each slice is allocated a portion of the node's memory and disk space, where it processes a portion of the workload assigned to the node. The leader node manages distributing data to the slices and apportions the workload for any queries or other database operations to the slices. The slices then work in parallel to complete the operation.

The number of slices per node (typically 2-32) is determined by the node size of the cluster.

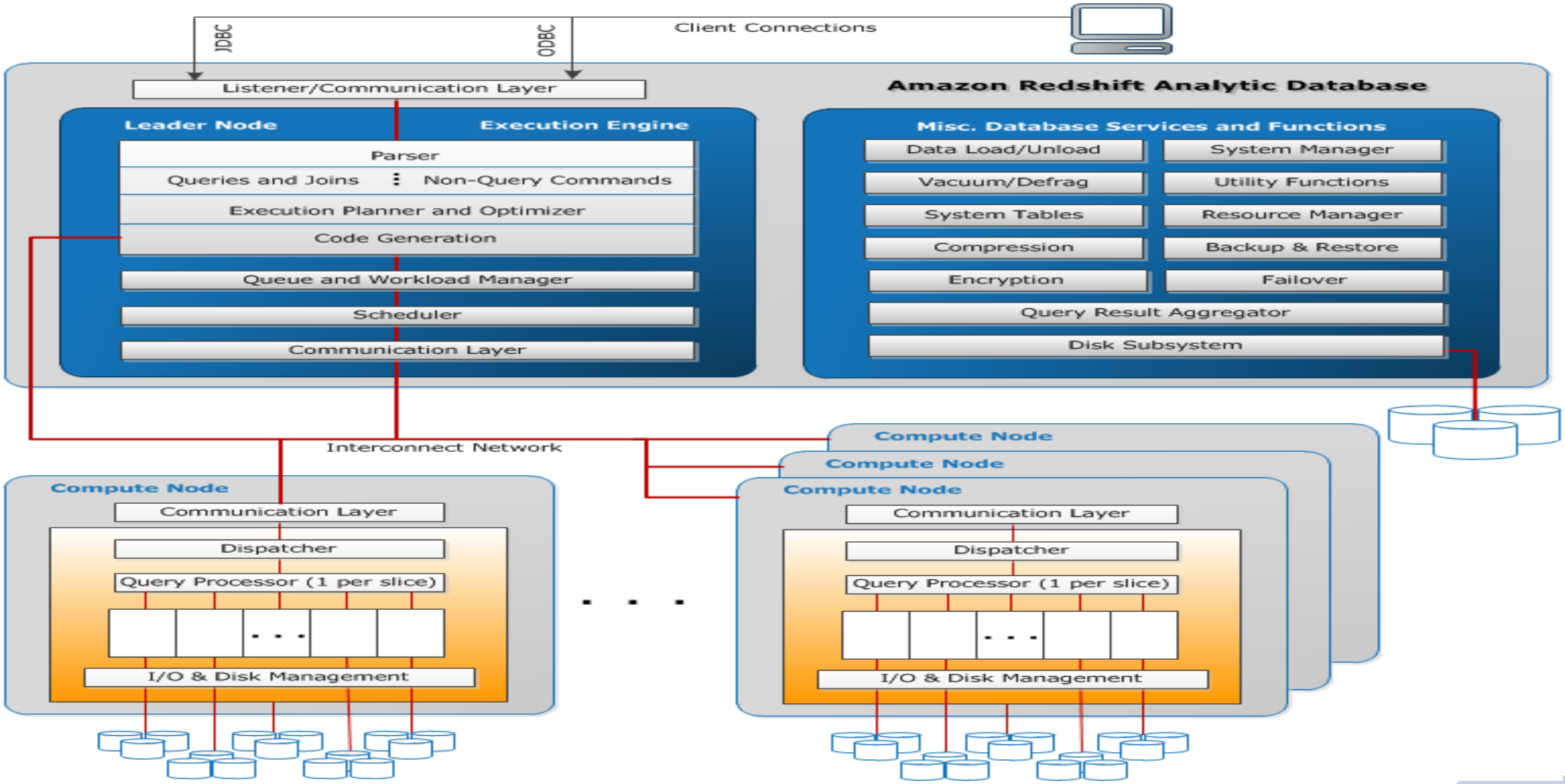
Internal network

Amazon Redshift takes advantage of high-bandwidth connections, close proximity, and custom communication protocols to provide private, very high-speed network communication between the leader node and compute nodes. The compute nodes run on a separate, isolated network that client applications never access directly.

Databases

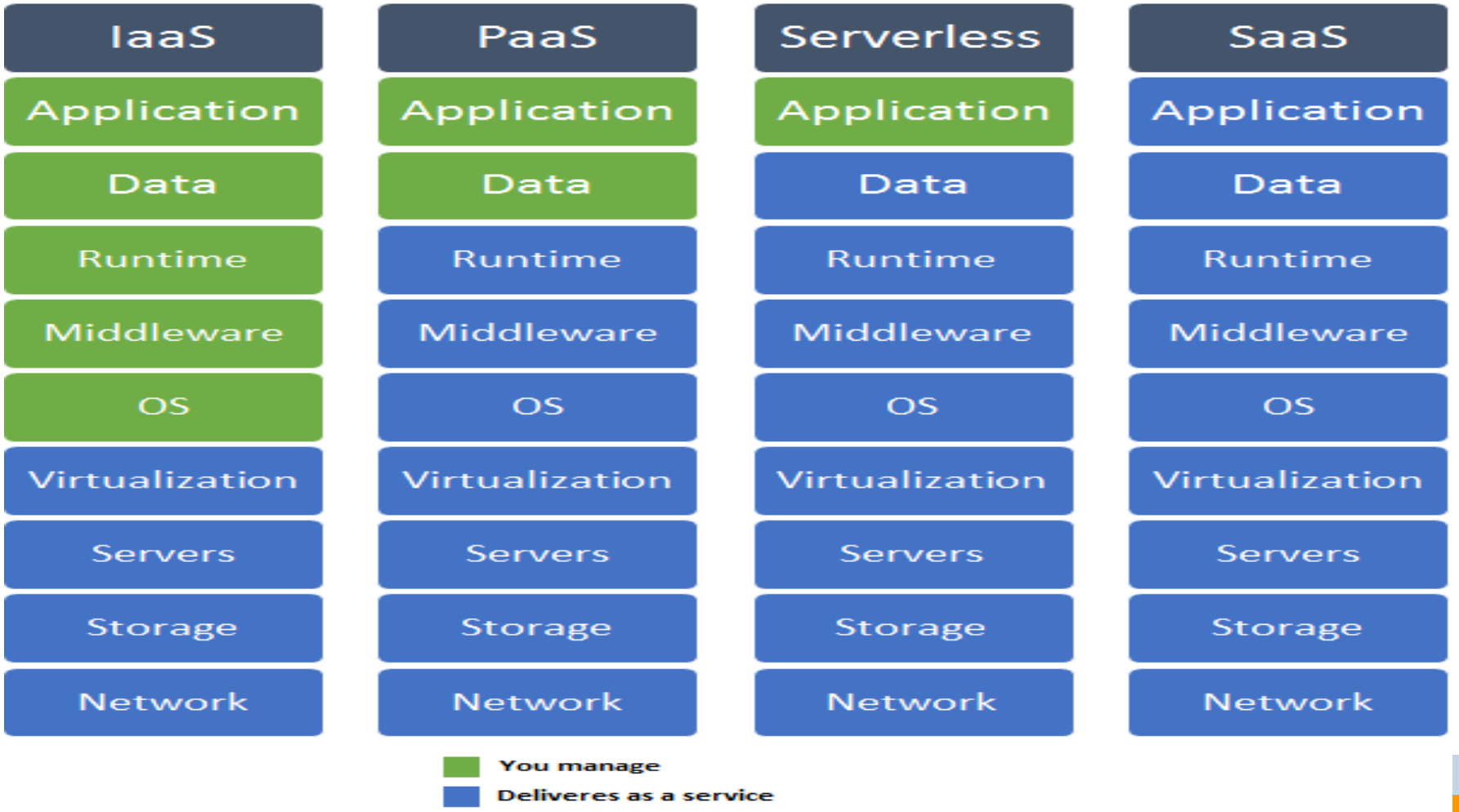
A cluster contains one or more databases. User data is stored on the compute nodes. Your SQL client communicates with the leader node, which in turn coordinates query execution with the compute nodes.

AWS Redshift

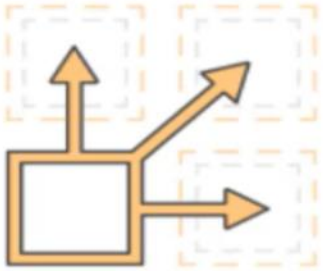


Serverless Services

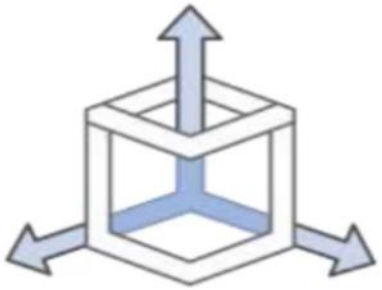
Serverless


IaaS
Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Network
PaaS
Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Network
Serverless
Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Network
SaaS
Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Network
You manage
Deliveres as a service

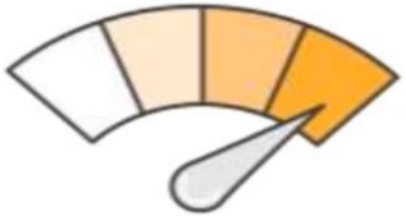
Serverless Benefits



No servers to provision or manage



Scales with usage



Never pay for idle

Availability and fault tolerance built in

DynamoDB

- Amazon DynamoDB is a **Serverless Fully managed NoSQL** database service that provides fast and low-latency performance that scales with ease.
- Amazon DynamoDB significantly simplifies the hardware provisioning, setup and configuration, replication, software patching, Encryption at rest and cluster scaling of NoSQL databases.
- Developers can create a table in Amazon DynamoDB and write an **unlimited number of items** with **consistent latency.**
- Amazon DynamoDB also provides automatic high-availability and durability protections by **replicating data across multiple Availability Zones** within an AWS Region.

DynamoDB

- Amazon DynamoDB automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent and fast performance.
- You can create on-demand backups as well as enable point-in-time recovery for your Amazon DynamoDB tables.
- Point-in-time recovery helps protect your Amazon DynamoDB tables from accidental write or delete operations. You can restore that table to any point in time during the last 35 days.
- DynamoDB allows you to delete expired items from tables automatically to help you reduce storage usage and the cost
- You can use APIs to connect to DynamoDB through your APP.

DynamoDB Benefits

- Single-digit millisecond response times at any scale.
- Virtually unlimited throughput and storage
- Consistent and fast performance.
- Automatic Global distribution of Data across Regions with Global tables.
- Microseconds latency achievable using DynamoDB Accelerator (DAX) -a fully managed in-memory cache.
- Serverless - no servers to provision, patch, or manage and no software to install, maintain, or operate.
- Automated scaling and Load distribution
- Built-in High Availability and fault tolerance
- Effective modes of costing - on-demand and provisioned capacity mode.
- Supports ACID transactions to enable you to build business-critical applications at scale.
- Security with Encryption and fine grained access control.
- SLA for guaranteed availability
- Automated backups and Point-in-time recovery feature.

DynamoDB Core Components

Tables, Items, and Attributes

The basic components of the Amazon DynamoDB data model include tables, items, and attributes.

A **Table** is a collection of items/data. Similar to other database systems, DynamoDB stores data in tables.

An **Item** is a group of attributes that is uniquely identifiable among all of the other items. Each item is a collection of one or more attributes Each item also has a primary key that uniquely identifies the item. here is no limit to the number of items you can store in a table.

An **Attribute** is a fundamental data element. Each Attribute in an item is a name/value pair. An attribute can be a single-valued or multivalued set. For example, an item in a People table contains attributes called PersonID, LastName, FirstName, and so on. DynamoDB supports nested attributes up to 32 levels deep.

LAB: Python with Boto3

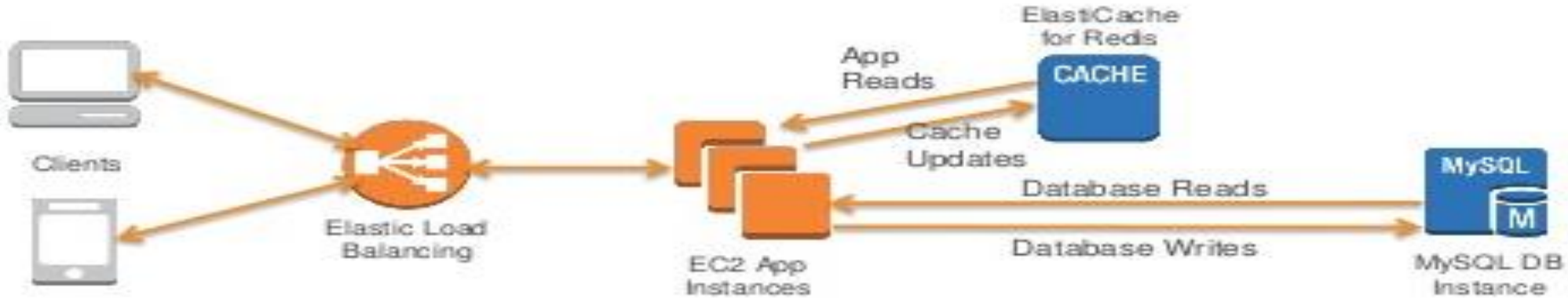
https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html

https://boto3.amazonaws.com/v1/documentation/api/latest/guide/dynamodb.html

AWS ElastiCache

AWS ElastiCache

- Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale distributed in-memory cache environments in the AWS Cloud.
- Removes the complexity associated with deploying, managing and Scaling a distributed cache environment
- ElastiCache works with both the Redis and Memcached engines.



AWS Redis Cache

- Redis is an open source, in-memory data structure store often used as a cache, database, or message broker.
- Automatic detection and recovery from cache node failures.
- Multi-AZ with automatic failover
- Supports partitioning your data across up to 15 shards
- Redis version 3.2.6 supports in-transit and at-rest encryption
- Integration with other AWS services such as Amazon EC2, Amazon CloudWatch, AWS CloudTrail, and Amazon SNS
- Existing applications that use Redis can use ElastiCache with almost no modification.

AWS Memcached

- Am In-memory key-value store service that can be used as a cache or a data store.
- Automatic detection and recovery from cache node failures.
- Multi-AZ with automatic failover
- Automatic discovery of nodes within a cluster
- Integration with other AWS services such as Amazon EC2, Amazon CloudWatch, AWS CloudTrail, and Amazon SNS
- Existing applications that use Memcached can use ElastiCache with almost no modification.

LAB 38

Create a ElastiCache Redis backed Cluster:

1. Open GUI console and Go to Services → Database → ElastiCache
2. Under Redis tab, click Create
3. Select Redis Cache Engine with Cluster mode enabled
4. Provide name and rest details
5. Select VPC/Subnet, Backup Schedule and Security group
6. Click Create
7. Open port 6379 in previously selected Security group
8. Try to connect Redis Cache cluster from any instance in same VPC/subnet using redis cache client :

   Redis-cli.exe –h hostname –p 6379

   Set name "value"

   Get Name

Delete the cluster when you have completed the exercise.

DB Services comparison

Amazon DynamoDB

- NoSQL database
- Fully managed
- Single-digit millisecond latency
- Massive and seamless scalability
- Low cost

Amazon RDS

- Relational databases
- Fully managed
- Predictable performance
- Simple and fast to scale
- Low cost, pay for what you use

Amazon Elasticache

- In-memory key-value store
- High-performance
- Memcached and Redis
- Fully managed, zero admin

Amazon Redshift

- Relational data warehouse
- Massively parallel; petabyte scale
- Fully managed
- HDD and SSD platforms
- $1,000/TB/year; starts at $ 0.25/hour

DB on Cloud

Best Practices & Constraints:

Use DB Paas (DBaaS)

- When you want to save License & Support cost (Windows + DB License).
- For new deployments and to deal with complex installation & management of DB HA.
- To avoid overhead of management issues alike, DB patching, upgrade, security fixes, failures, scaling, Backup, Mirroring, logs etc.
- When you want to go complaint instantly for your newly deployed DBs
- When your application is on cloud.

Use DB on IaaS

- If you want to utilize your existing licenses, as PaaS don’t always offers you BYOL for DB & OS.
- In case you need access to DB libraries or need all DB features
- When OS level access is required
- IaaS is the only choice for legacy and un-supported DB versions.

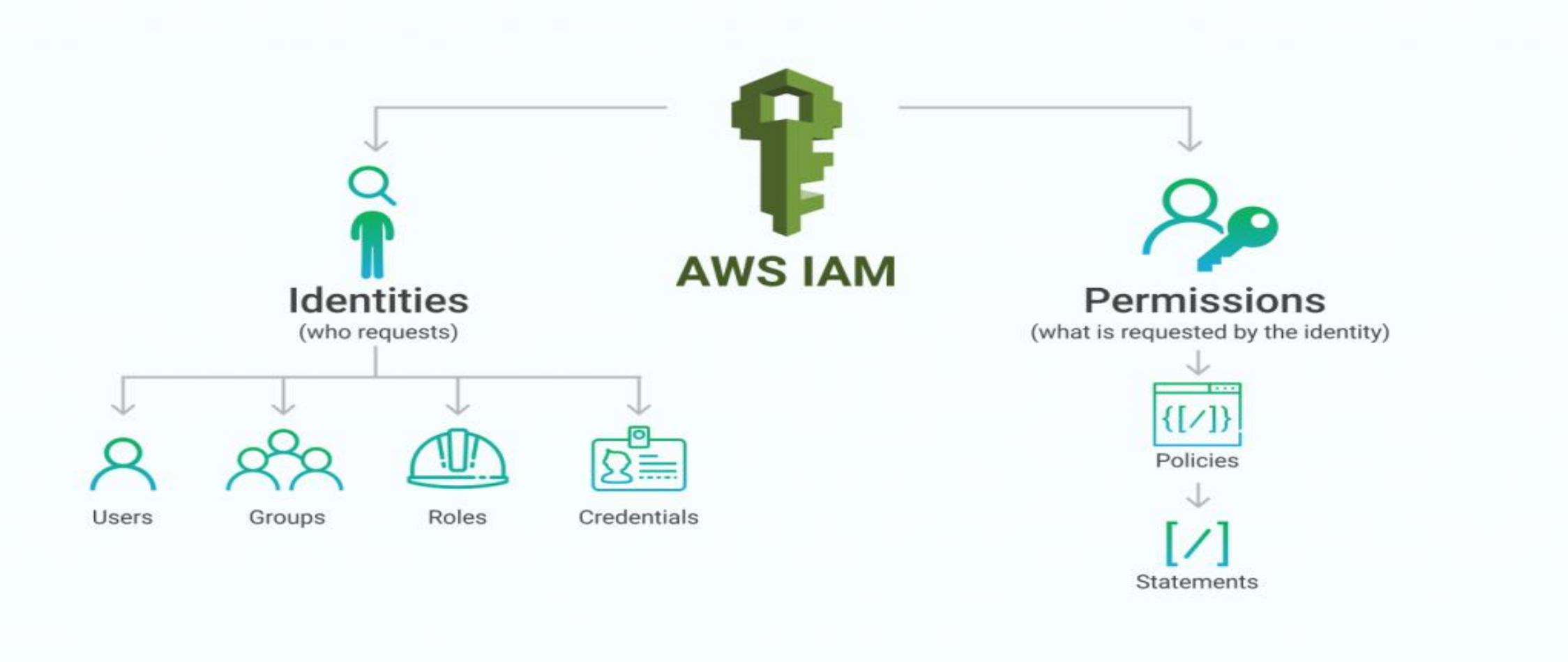
AWS IAM

IAM

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources for your users. You use IAM to control who can use your AWS resources (authentication) and what resources they can use and in what ways (authorization).

- ✓ Shared access to your AWS account
- ✓ Granular permissions
- ✓ Secure access to AWS resources for applications that run on Amazon EC2
- ✓ Integrated with many AWS services
- ✓ Free to use

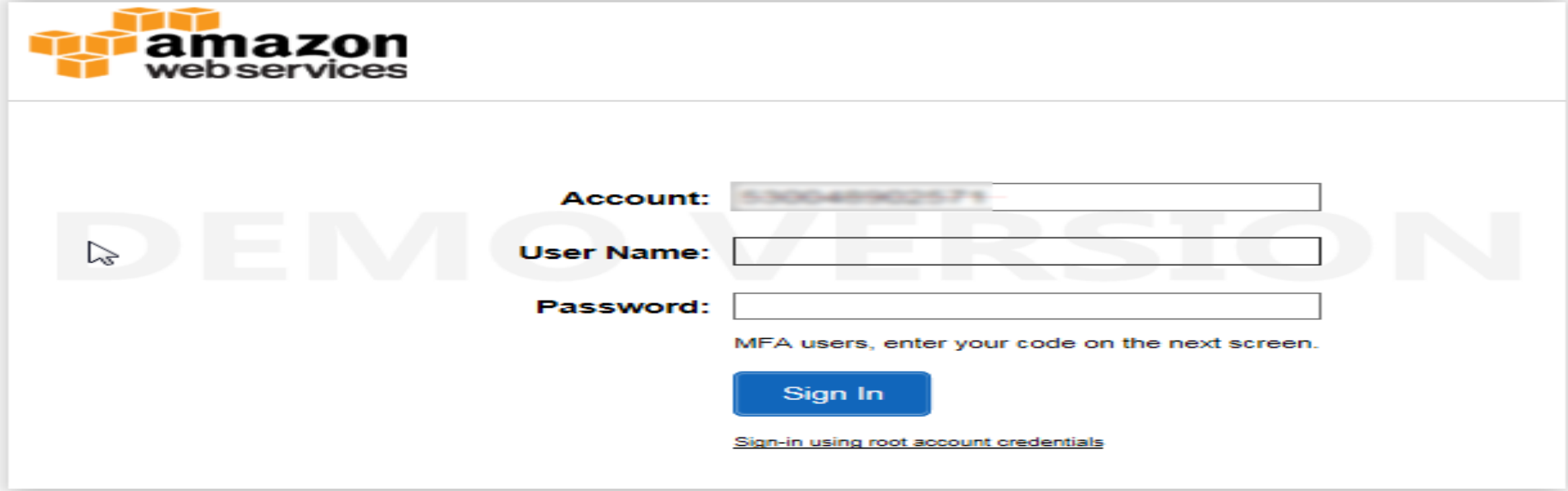
IAM


AWS IAM
Identities
(who requests)
Users
Groups
Roles
Credentials
Permissions
(what is requested by the identity)
Policies
Statements

IAM Users

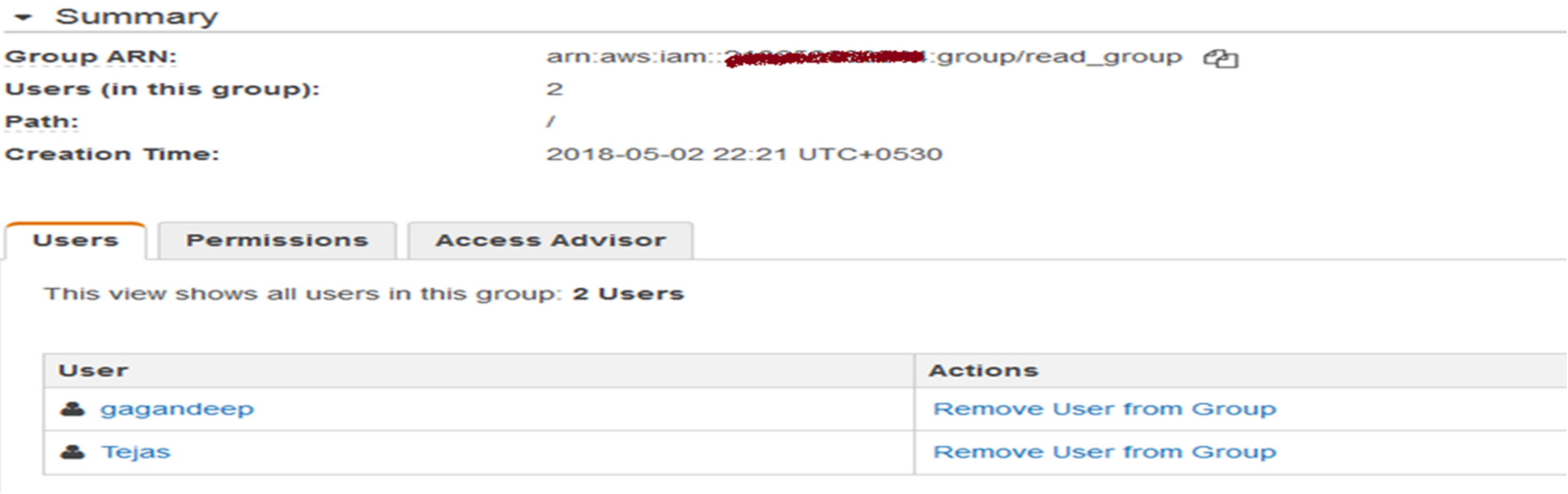
You can create IAM users to distribute your work among team/users.

Different permissions can be given to IAM users



IAM Groups

Users are part of IAM groups
Common policies can be applied on groups



IAM Roles

IAM roles are a secure way to grant permissions to entities that you trust.

- ✓ IAM user in another account
- ✓ Application code running on an EC2 instance that needs to perform actions on AWS resources
- ✓ An AWS service that needs to act on resources in your account to provide its features
- ✓ Users from a corporate directory who use identity federation with SAML

Identity Policies

Policies are the way to define permissions/authorization. Policies can be assigned to Users, Groups & Roles.

Contains 4 parts:

- ✓ Service
- ✓ Action
- ✓ Resources
- ✓ Request Conditions

A policy is a document (written in the Access Policy Language) that acts as a container for one or more statements.

AWS KMS
(Key Management Services)

KMS

AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data.

AWS Key Management Service is integrated with other AWS services including Amazon EBS, Amazon S3, Amazon Redshift, and others to make it simple to encrypt your data with encryption keys that you manage.

by default AWS KMS generates the key material for that CMK. But you can create a CMK without key material and then import your own key material into that CMK, a feature often known as "bring your own key" (BYOK).

You can Enable, Disable the key and even do schedule deletion (no Immediate deletion of keys permitted).

KMS Benefits

Fully managed

Centralized key management

Integrated with AWS services

Encryption for all your applications (using SDKs)

Built-in auditing

No commitment for usage

Secure with FIPS 140-2 validated hardware security modules

KMS uses highly durable storage and a resilient architecture to ensure that your keys are always available and are never lost.

You can set usage policies that determine which users can use your keys and what actions they can perform. All requests to use these keys are logged in AWS CloudTrail so that you can track who used which key, how and when.

KMS

Earlier AWS was supporting only Symmetric Keys But since 2019 Nov, its started providing support for Asymmetric keys too.

Symmetric (default): A 256-bit single encryption key which is used for both encryption and decryption.

Asymmetric: A public and private key pair that can be used for encrypt/decrypt or sign/verify operations

If your use case requires encryption outside of AWS by users who cannot call AWS KMS, asymmetric CMKs are a good choice. However, if you are creating a CMK to encrypt the data that you store or manage in an AWS service, use a symmetric CMK.

Compare the keys services and operations at:

https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-compare.html

LAB: KMS

Managing key under KMS

Go to **IAM → encryption keys**

1. Click on create key
2. Provide name and click on advance tab to select the key origin. For now, keep it KMS only.
3. Select Admins, who can manage the Key
4. Select users, who can use this key to encrypt & decrypt their data
5. Preview and launch the key creation
6. Try to create a new volume and check in Volume encryption, whether new key is popping up or not.
7. Try to enable & disable this key.
8. Do schedule key deletion and set a period of 7 days to get the key deleted.

AWS IAM Advanced

CLI Credentials Priority

- The CLI will look for credentials in this order
  1. Command line options – --region, --output, and --profile
  2. Environment variables – AWS_ACCESS_KEY_ID,AWS_SECRET_ACCESS_KEY, and AWS_SESSION_TOKEN
  3. CLI credentials file –aws configure
  ~/.aws/credentials on Linux / Mac & C:\Users\user\.aws\credentials on Windows
  4. CLI configuration file – aws configure
  ~/.aws/config on Linux / macOS & C:\Users\USERNAME\.aws\config on Windows
  5. Container credentials – for ECS tasks
  6. Instance profile credentials – for EC2 Instance Profiles

- Note: CLI actually uses Python BOTO3 SDK in backend.

SDK Credentials Priority

1. Environment variables –

AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY

2. Java system properties – aws.accessKeyId and aws.secretKey

3. The default credential profiles file – ex at: ~/.aws/credentials, shared by

many SDK

4. Amazon ECS container credentials – for ECS containers

5. Instance profile credentials– used on EC2 instances

Credentials Best practice

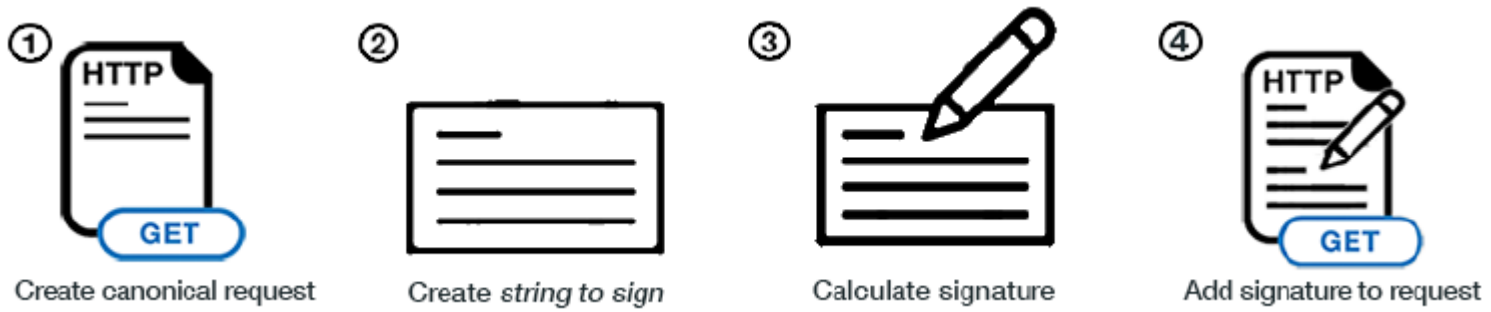
Overall, NEVER EVER STORE AWS CREDENTIALS IN YOUR CODE

- Best practice is for credentials to be inherited from the credentials chain
- If using working within AWS, use IAM Roles
- => EC2 Instances Roles for EC2 Instances
- => ECS Roles for ECS tasks
- => Lambda Roles for Lambda functions
- If working outside of AWS, use environment variables

Signing AWS Request

When you call the AWS HTTP API, you sign the request so that AWS can identify you, using your AWS credentials (access key & secret key)

- If you use the SDK or CLI, the HTTP requests are signed for you
- You should sign an AWS HTTP request using Signature v4 (SigV4)



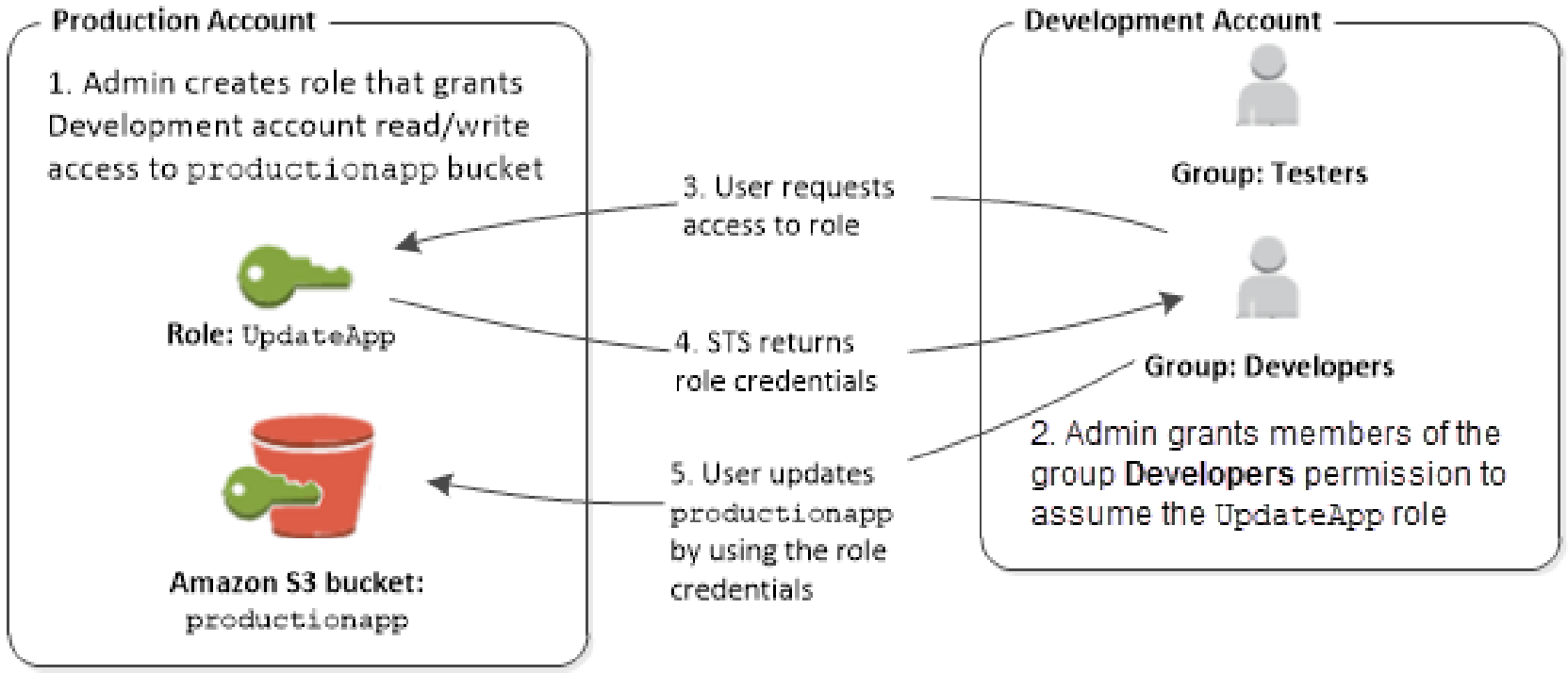
STS

AWS Security Token Service (AWS STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

Allows to grant limited and temporary access to AWS resources (up to 1 hour).

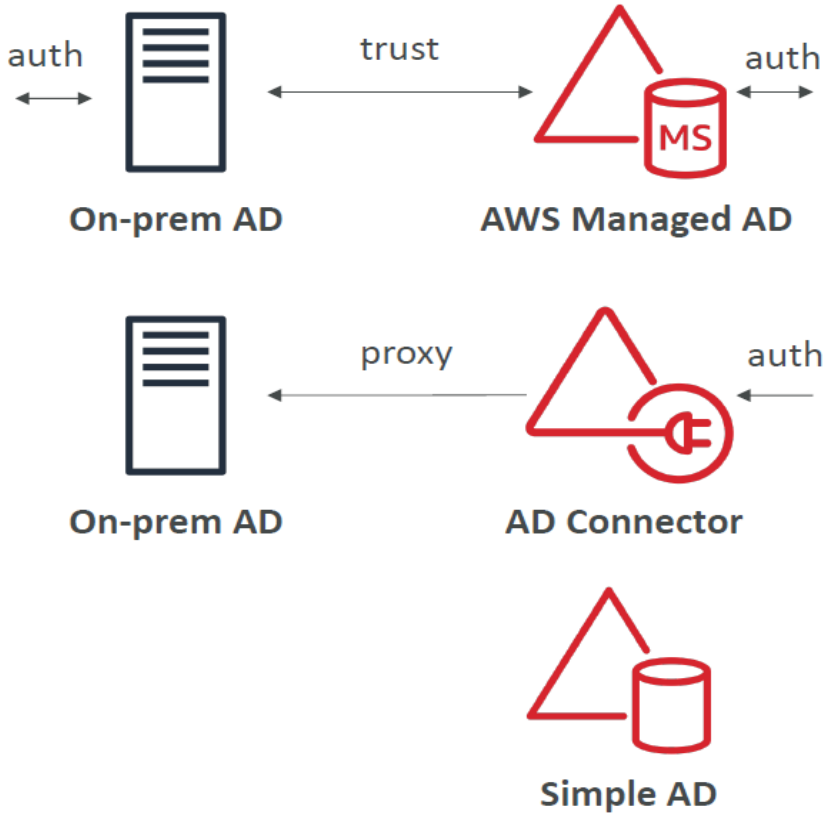
- AssumeRole: Assume roles within your account or cross account
- AssumeRoleWithSAML: return credentials for users logged with SAML
- AssumeRoleWithWebIdentity
- GetSessionToken: for MFA, from a user or AWS account root user
- GetFederationToken: obtain temporary creds for a federated user
- GetCallerIdentity: return details about the IAM user or role used in the API call
- DecodeAuthorizationMessage: decode error message when an AWS API is denied

Cross-Account Permission



Active Directory on AWS

- AWS Managed Microsoft AD
  - Create your own AD in AWS, manage users locally, supports MFA
  - Establish "trust" connections with your on-premise AD
- AD Connector
  - Directory Gateway (proxy) to redirect to on-premise AD
  - Users are managed on the on-premise AD
- Simple AD
  - AD-compatible managed directory on AWS
  - Cannot be joined with on-premise AD



AWS Cognito

AWS Cognito

We want to give our users an identity so that they can interact with our application.

Cognito User Pools:

- Sign in functionality for app users
- Integrate with API Gateway & Application Load Balancer

Cognito Identity Pools (Federated Identity):

- Provide AWS credentials to users so they can access AWS resources directly
- Integrate with Cognito User Pools as an identity provider

Cognito Sync:

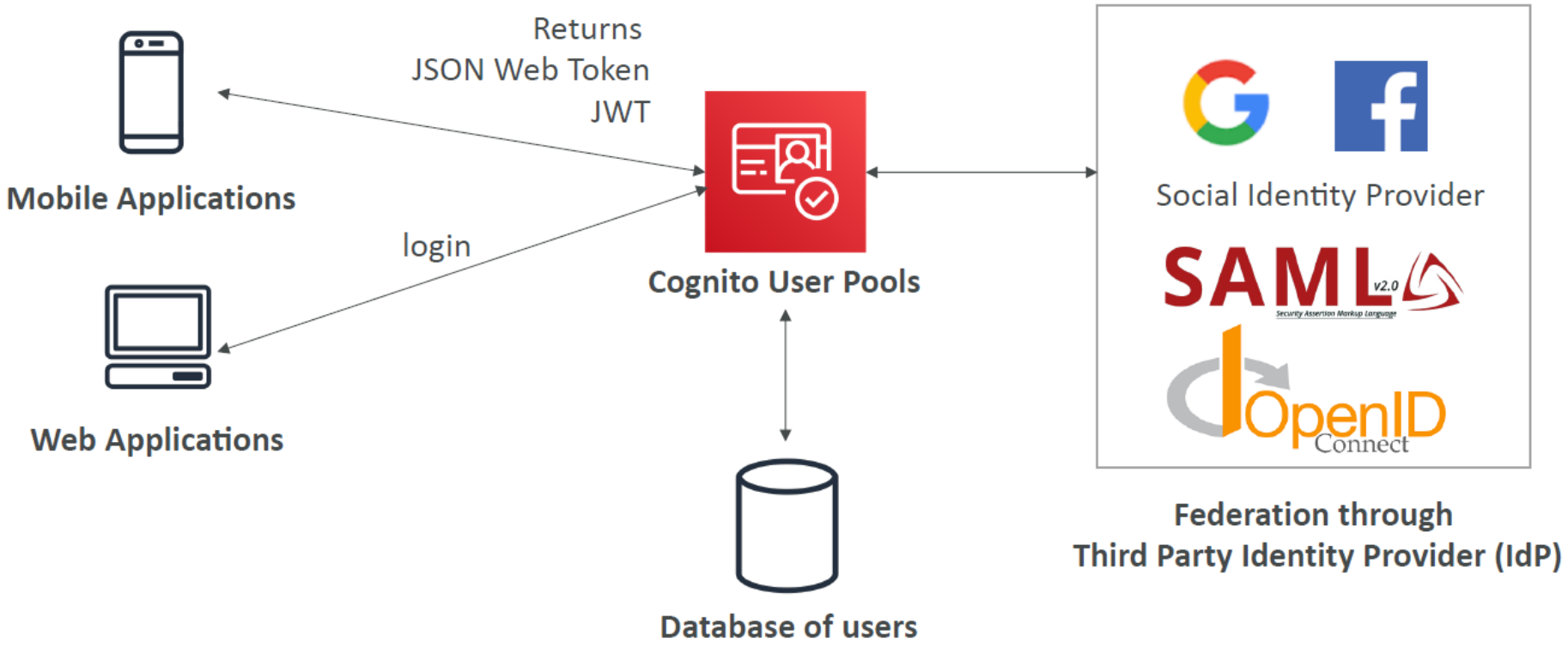
- Synchronize data from device to Cognito.
- Is deprecated and replaced by AppSync
- Cognito vs IAM: “hundreds of users”, ”mobile users”, “authenticate with SAML”

AWS Cognito

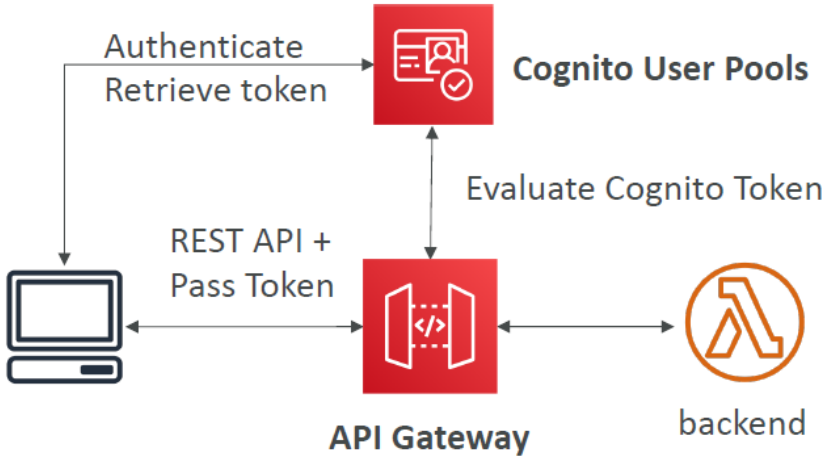
Cognito User Pools (CUP) – User Features

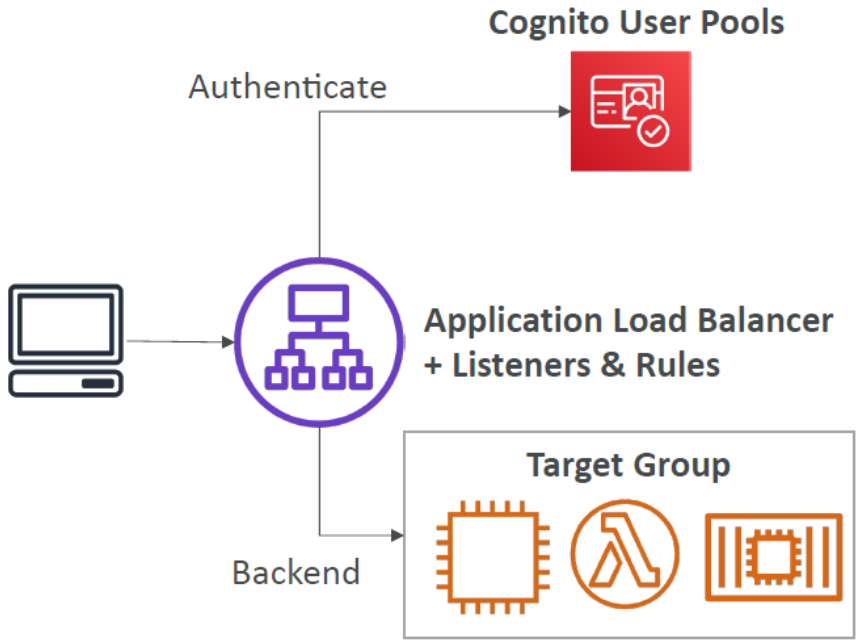
- Create a serverless database of user for your web & mobile apps
- Simple login: Username (or email) / password combination
- Password reset
- Email & Phone Number Verification
- Multi-factor authentication (MFA)
- Federated Identities: users from Facebook, Google, SAML…
- Feature: block users if their credentials are compromised elsewhere
- Login sends back a JSON Web Token (JWT)

AWS Cognito Usage



User Pool Integrations





UserPool Lambda Trigger

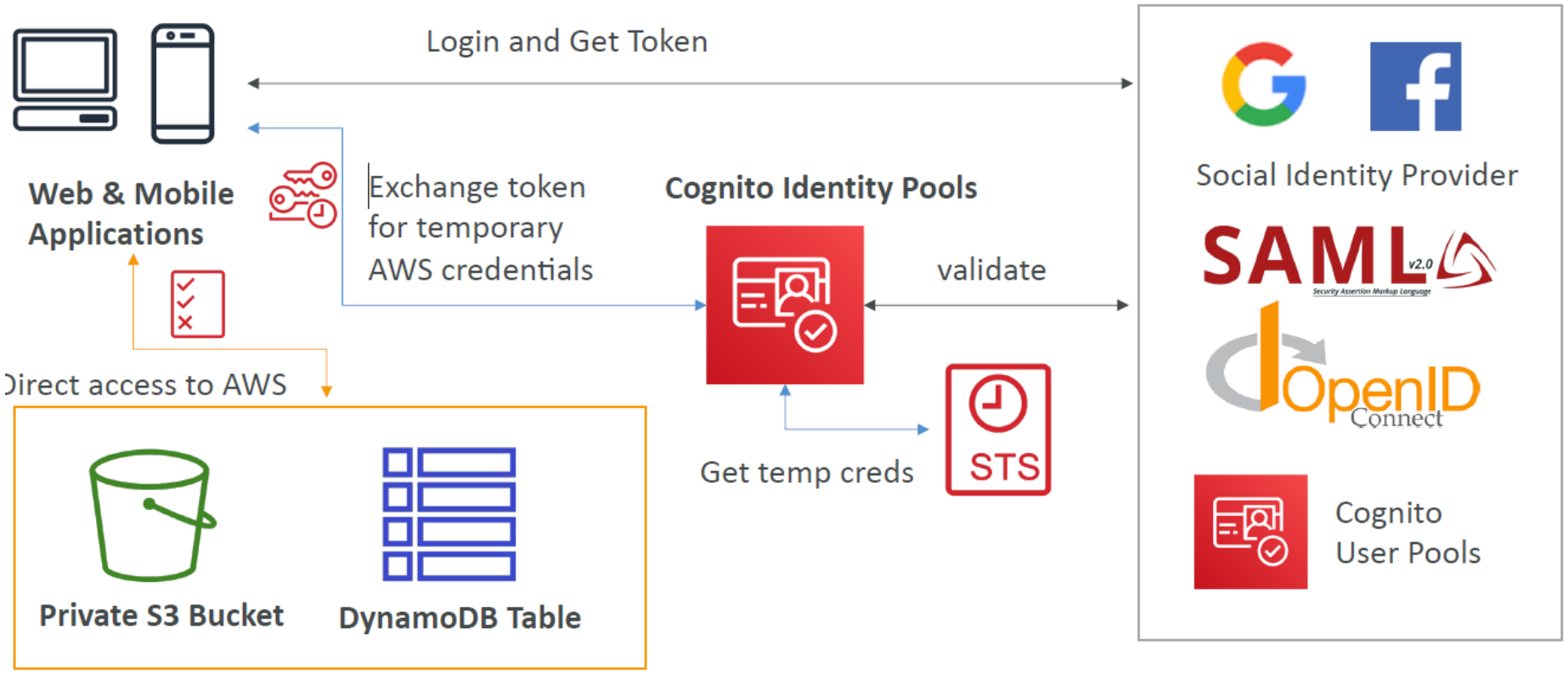
User Pool Flow	Operation	Description
Authentication Events	Pre Authentication Lambda Trigger	Custom validation to accept or deny the sign-in request
	Post Authentication Lambda Trigger	Event logging for custom analytics
	Pre Token Generation Lambda Trigger	Augment or suppress token claims
Sign-Up	Pre Sign-up Lambda Trigger	Custom validation to accept or deny the sign-up request
	Post Confirmation Lambda Trigger	Custom welcome messages or event logging for custom analytics
	Migrate User Lambda Trigger	Migrate a user from an existing user directory to user pools
Messages	Custom Message Lambda Trigger	Advanced customization and localization of messages
Token Creation	Pre Token Generation Lambda Trigger	Add or remove attributes in Id tokens

Cognito Identity Pool

Get identities for “users” so they obtain temporary AWS credentials

- Your identity pool (e.g identity source) can include:
- Public Providers (Login with Amazon, Facebook, Google, Apple)
- Users in an Amazon Cognito user pool
- OpenID Connect Providers & SAML Identity Providers
- Developer Authenticated Identities (custom login server)
- Cognito Identity Pools allow for unauthenticated (guest) access
- Users can then access AWS services directly or through API Gateway
- The IAM policies applied to the credentials are defined in Cognito
- They can be customized based on the user_id for fine grained control

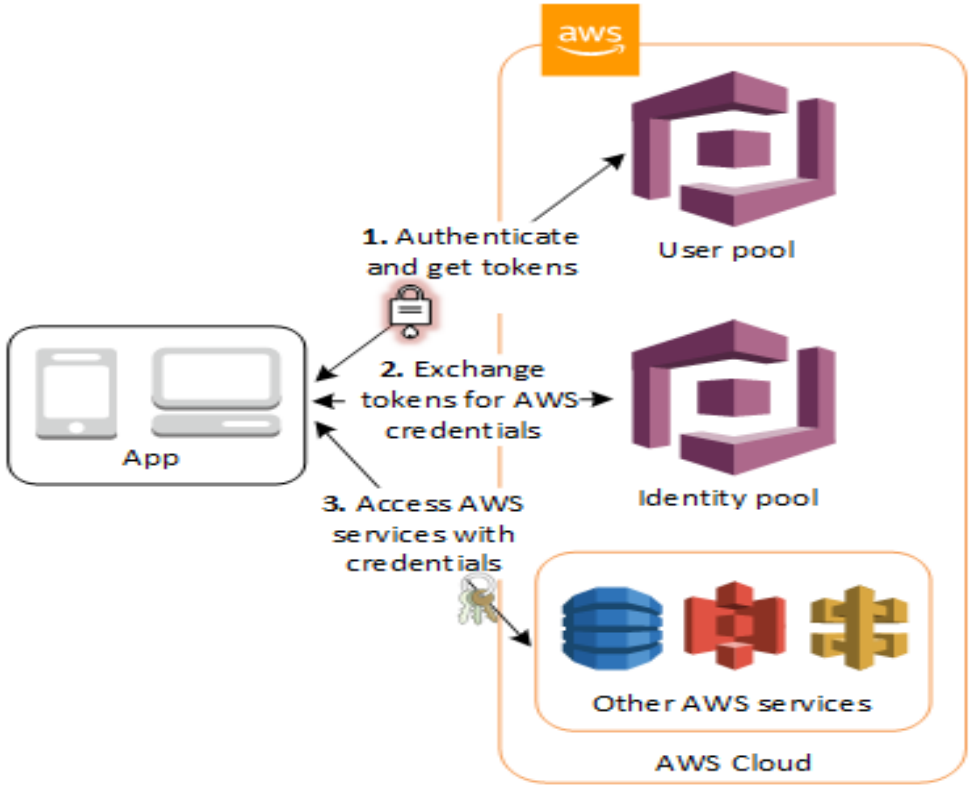
Identity Pool



Cognito

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple.

The two main components of Amazon Cognito are **user pools** and **identity pools**. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.



Cognito

https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-example-photo-album.html

Elastic Beanstalk

AWS Elastic BeanStalk

An easy-to-use service for deploying and managing web applications and services without worrying about the infrastructure that runs those applications.

Support App/Services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

Just upload the code. Rest Beanstalk will take care.

You retain full control over the AWS resources powering your application and can access the underlying resources at any time.

Free of cost service.

AWS Elastic BeanStalk

You can have a set of resources created and visible to you all the time under one umbrella.

Almost all resources which are required for your application, can be visible and manageable from EB dash board itself.

Central Logging and health monitoring.

A dedicated CLI for managing Elastic Beanstalk environment- EB CLI.

Keep track of Application codes and activities been done on project environment.

In case of any issues with Server/underline infra, it'll recreate additional particular infrastructure piece to maintain the good state.

AWS Elastic BeanStalk

You can have your production environment cloned and that can be your dev/test environment.

With one click you can switch between these environments. So your development/preprod environment can be your production environment in one click and in just a matter of few seconds.

Only your traffic needs to be migrated through previous environment to new env, so no downtime for end users.

Due to multiple environments, you can keep your previous environment as it is for few days and then upgrade the same to latest version.

Due to this your application will remain always available and 99.9% uptime can be achievable.

AWS SQS

AWS SQS

Amazon SQS (Simple Queue Service) is a fast, reliable, scalable, and fully managed message queuing service.

An Amazon SQS queue is basically a buffer between the application components that receive data and those components that process the data in your system. If your processing servers cannot process the work fast enough (perhaps due to a spike in traffic), the work is queued so that the processing servers can get to it when they are ready. This means that work is not lost due to insufficient resources.

Amazon SQS ensures delivery of each message at least once.

Amazon SQS is engineered to be highly available and to deliver messages reliably and efficiently.

Amazon SQS supports ***Signature Version 4***, a protocol for authenticating inbound **API requests**. Applications with existing Message brokers -AMQP, MQTT, OpenWire, STOMP and JMS(API based), can be migrated to SQS.

AWS SQS

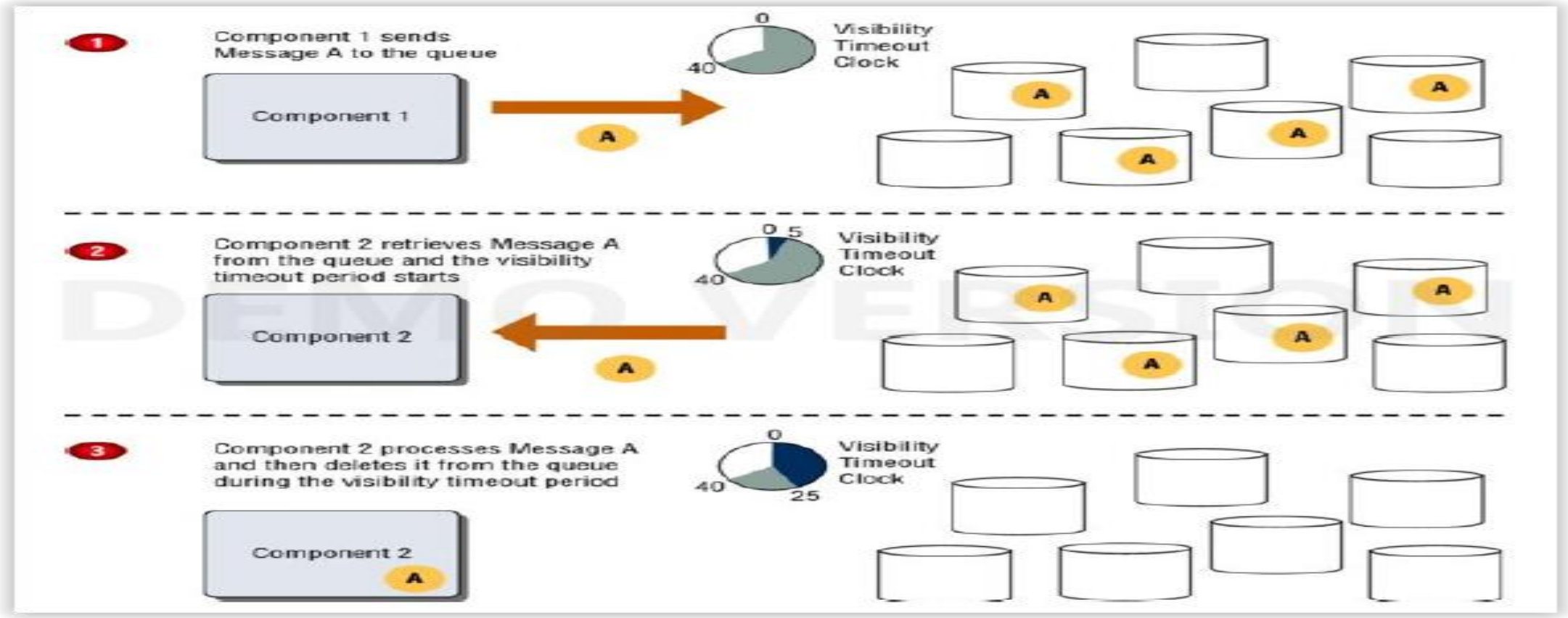
Standard Queue:

Unlimited Throughput
At-Least-Once Delivery, occasionally delivered more than once
Best-Effort Ordering
Suitable for applications, where ordering doesn’t matter & high throughput required

FIFO Queue:

High Throughput (By Default 300 operations per second, can be increased)
Strictly First-In-First-out Delivery
Exactly-Once Processing
Suitable for applications where Ordering must be in sequence and to be exactly once.

AWS SQS


1
Component 1 sends Message A to the queue
Component 1
A
0
40
Visibility Timeout Clock
2
Component 2 retrieves Message A from the queue and the visibility timeout period starts
Component 2
A
0 5
40
Visibility Timeout Clock
3
Component 2 processes Message A and then deletes it from the queue during the visibility timeout period
Component 2
A
0
40
25
Visibility Timeout Clock

Lab: AWS SQS

Create Queue

1.In the AWS Management Console, navigate to Application Services and then to Amazon SQS to load the Amazon SQS dashboard.

2. Create a new standard queue with **input** as the queue name, 60 seconds for the default visibility timeout, and 5 minutes for the message retention period. Leave the remaining default values for this exercise.

3. Create the queue.

AWS SNS

AWS SNS

Amazon SNS (Simple Notification Service) is a web service for mobile and enterprise messaging that enables you to set up, operate, and send notifications.

You can use Amazon SNS to publish messages in Http, Https, Emails, SMS, SQS, or Lambda protocols.

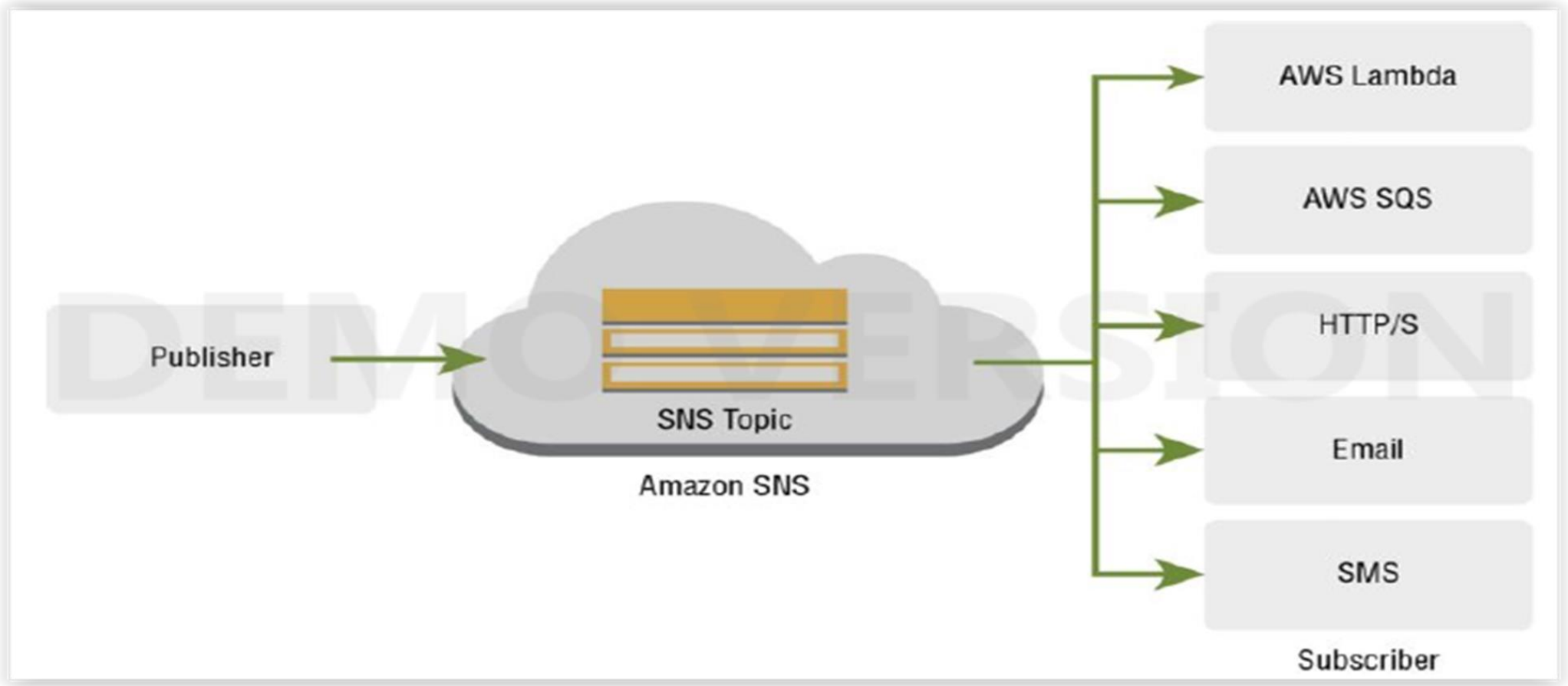
Amazon SNS consists of two types of clients: publishers and subscribers (sometimes known as producers and consumers).

Publishers communicate to subscribers asynchronously by sending a message to a topic.

A topic is simply a logical access point/communication channel that contains a list of subscribers and the methods used to communicate to them.

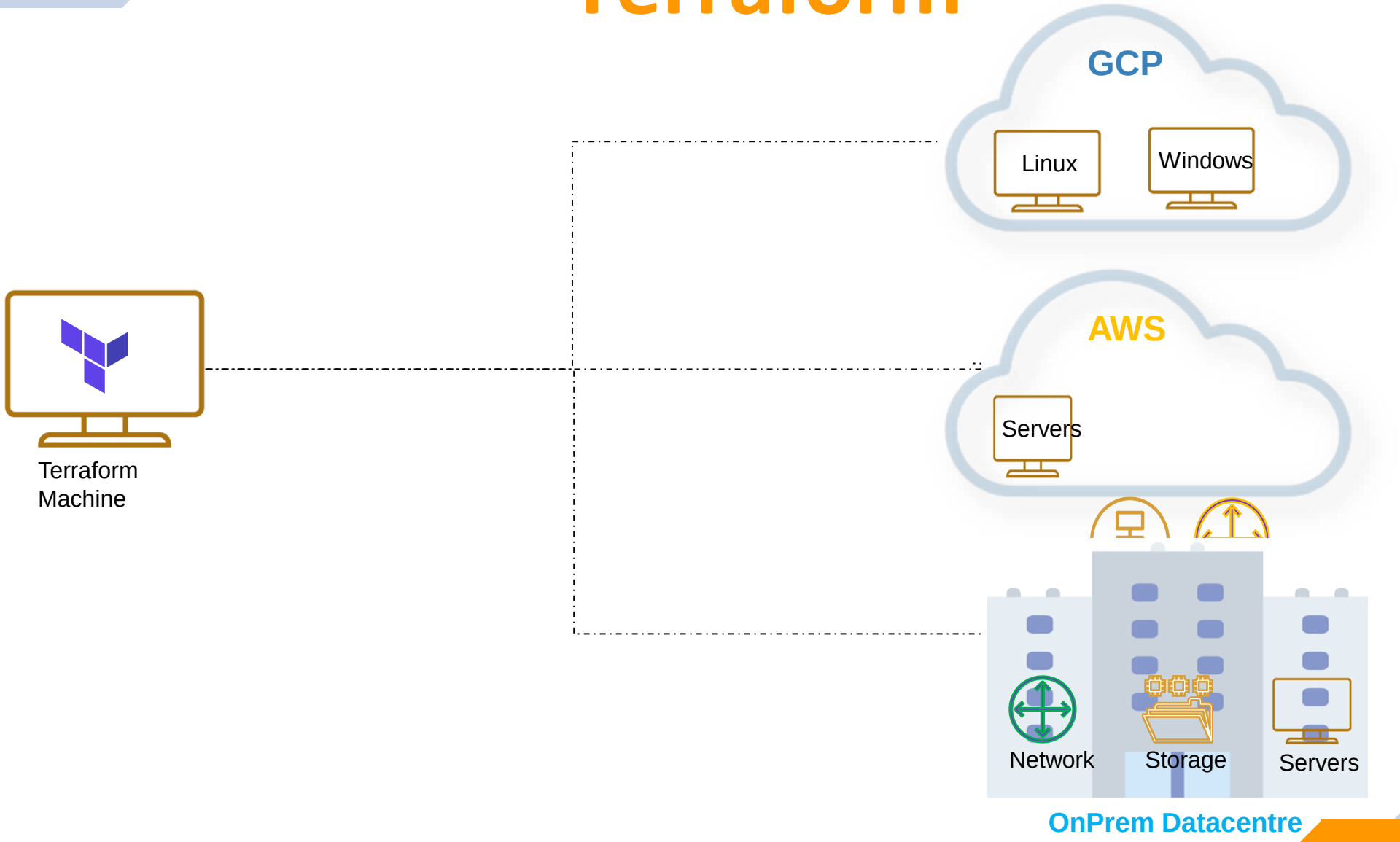
When you send a message to a topic, it is automatically forwarded to each subscriber of that topic using the communication method configured for that subscriber.

AWS SNS



What is Orchestration?

Terraform


GCP
Linux
Windows
AWS
Servers
Terraform
Machine
Network
Storage
Servers
OnPrem Datacentre

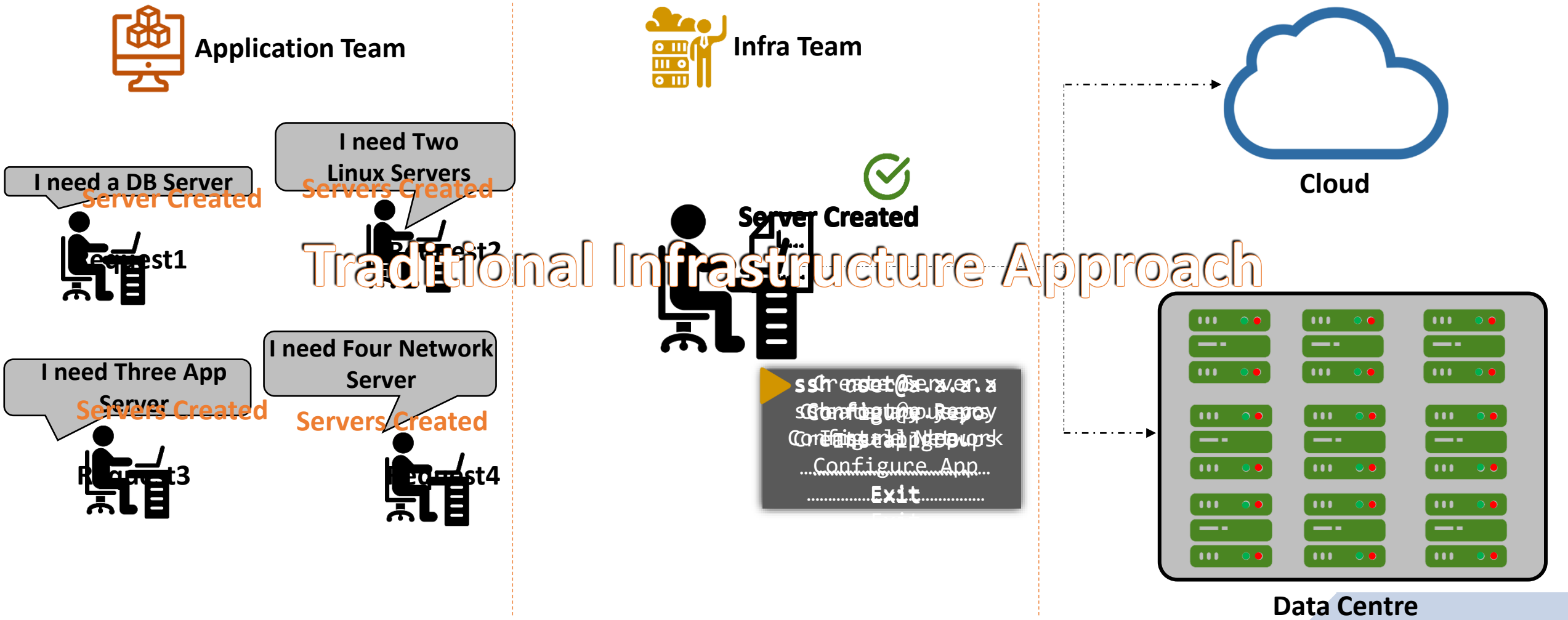
GUI vs CLI vs IAC

- **GUI (Graphical User Interface)**
  - ✓ Best for end user experience
  - ✓ Easy management
  - ✓ Bad for Automation
  - ✓ Not helpful for Administrators

- **CLI (Command Line Interface)**
  - Best for Admin Experience
  - Easy management for Admin level tasks
  - Bad for end user experience
  - Bad for maintaining desired state and consistency

- **IaC (Infrastructure as Code)**
  - Best for Admin Experience
  - Easy management for Admin tasks
  - Easy to understand for end users too
  - Can easily maintain consistency and desired state
  - Infrastructure is written in files, so can be versioned

Why DevOps IaC



Traditional Infrastructure Approach

IAC (Infrastructure as Code)

Application Team

I need DB Server

Server Created

Request1

I need Two Linux Servers

Servers Created

Request2

need Three Linux Servers

Servers Created

Request3

I need Four Linux Servers

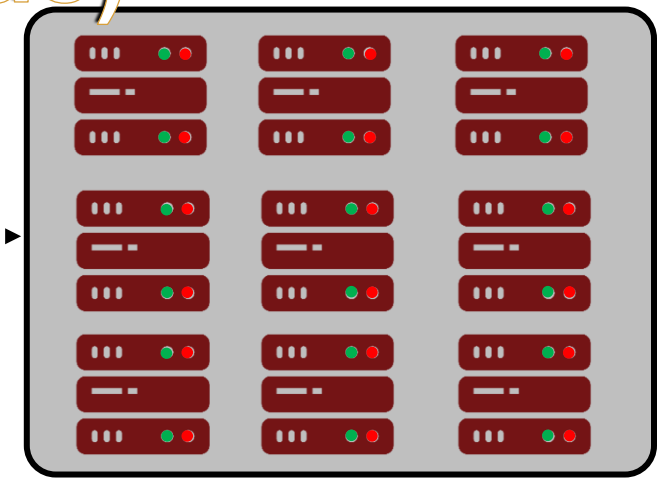
Servers Created

Request4

Infra Team

```
File is: main.tf
provider "aws" {
  region = "us-east-1"
}
resource "aws_instance" “requestfour" {
  count = "4"
  ami = "ami-03d5c68bab01f3496"
  instance_type = "t2.micro"
  tags = {
    Name = "DevOpsInAction"
  }
}
output "myawsserver" {
  value =
"${aws_instance.myawsserver.public_ip}"
}
```

Cloud



Data Centre

IaC is Managing Infrastructure in files rather than manually configuring resources in a user interface

AWS
Cloud Formation

AWS CloudFormation

- CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly.
- It enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).
- A JSON/YAML based simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all regions and accounts.
- No additional charge. You pay only for the AWS resources needed to run your applications.
- A true Infrastructure as Code (IAC).

AWS CloudFormation

Benefits:

- Model/Template your entire infrastructure in a text file.
- Automate and deploy : Provisions your resources in a safe, repeatable manner, allowing you to build and rebuild your infrastructure and applications, without having to perform manual actions.
- Automated Roll-backs in case of errors
- Infrastructure as Code
- Versioning of your Infrastructure
- Quickly Replicate Your Infrastructure
- Tested & validated code. So less prone to errors.

AWS CloudFormation

How it works

Code your infrastructure from scratch with the CloudFormation template language, in either YAML or JSON format, or start from many available sample templates

Check out your template code locally, or upload it into an S3 bucket

Use AWS CloudFormation via the browser console, command line tools or APIs to create a stack based on your template code

AWS CloudFormation provisions and configures the stacks and resources you specified on your template

AWS CloudFormation

Usage:

- Quickly, Reliably, Error free & Easy deployment of your infrastructure.
- Easy replication for your infrastructure i.e. replicating your production stack to development or test environment.
- Your can tier down or create infrastructure whenever you want.
- Also beneficial, in case you need to deploy similar architecture across your clients.
- Can share your infra code/setup with client/community/partners.
- Previously created, tested & validated templates will help to reduce TTM for new projects.

AWS CloudFormation Concepts

Templates:

- An AWS CloudFormation template is a JSON or YAML formatted text file.
- CloudFormation uses these templates as blueprints for building your AWS resources. Same like we get blueprints created by architects before building our home.
- You can define multiple resources in template like EC2 instance, LB etc. & their properties
- You can add input parameters whose values are specified when you create an AWS CloudFormation stack.

AWS CloudFormation Concepts

Templates: To create a simple EC2 instance

```
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Resources:
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-2f726546"
      InstanceType: t1.micro
      KeyName: testkey
      BlockDeviceMappings:
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

AWS CloudFormation Concepts

Change Sets:

- Before making changes to your stack resources, you can generate a change set, which is summary of your proposed changes.
- Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.
- For example, if you change the name of an Amazon RDS database instance, AWS CloudFormation will create a new database and delete the old one. You will lose the data in the old database unless you've already backed it up. If you generate a change set, you will see that your change will cause your database to be replaced, and you will be able to plan accordingly before you update your stack.

AWS CloudFormation Concepts

Stacks:

- When you use AWS CloudFormation, you manage related resources as a single unit called a stack.
- You create, update, and delete a collection of resources by creating, updating, and deleting stacks.
- Resources in a stack are defined by the stack's AWS CloudFormation template
- You can work with stacks by using the AWS CloudFormation console, API, or AWS CLI.

LAB 51

Create a CloudFormation stack

In this exercise, you will create a Wordpress blog using cloudformation:

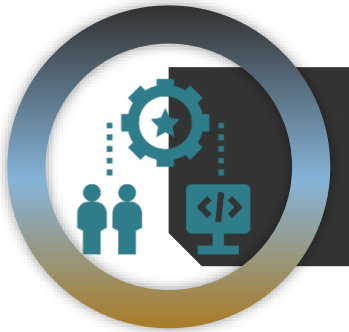
1. Log in to the AWS Management Console and click on CloudFormation→ “Create Stack”
2. From “Choose a template”, select “WordPress blog”.
3. Click next.
4. Provide stack name and rest parameter values
5. Review and create stack
6. Click on stack name and check the different tabs and values.
7. Under EC2 section, check the instance being created for this stack

Terraform

Terraform is an easy-to-use IT Orchestration & Automation Software for System Administrators & DevOps Engineers.

- It is the infrastructure as code offering from Hashicorp.
- It is a tool for building, changing, and managing infrastructure in a safe, repeatable way.
- Configuration language called the HashiCorp Configuration Language (HCL) is used to configure the Infrastructure.
- Compatible with almost all major public and private Cloud service provider

Terraform



Infrastructure as code (IAC)

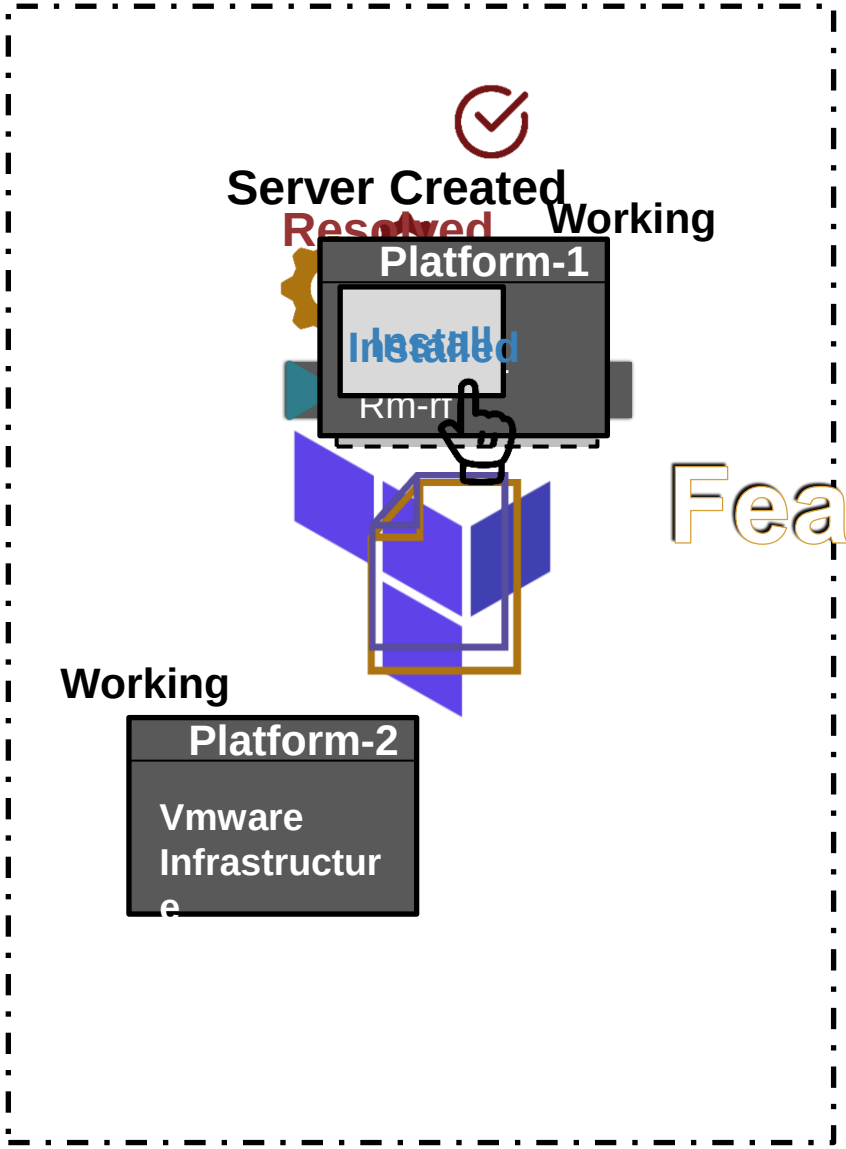
July 2014, HashiCorp

What is Terraform?

Opensource / Enterprise

HCL (Hashicorp Configuration Language)

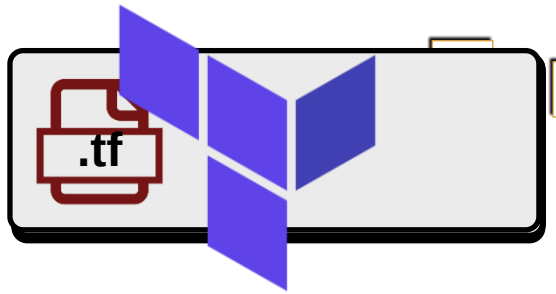
Terraform



Feature & Advantages

- Easy Installation
- Declarative in Nature
- Idempotent
- Intelligent Dependency Resolver
- Platform Agnostic
- Simple and easy to use

Terraform



Terraform Terminologies

Providers

Variables

Resources

Provisioners

DataSources

Outputs

Modules

File extension
.tf

Terraform

main.tf

```
provider "aws" {
  region = "us-east-1"
}
```

Provider Block

```
resource "aws_instance" "myserver" {
  ami = "ami-030ff268bd7b4e8b5"
  instance_type = "t2.micro"
  tags = {
    Name = "DevOpsInAction"
  }
}
```

Resource Block

```
output "myserveroutputs" {
  description = "Display Servers Public IP"
  value = "${aws_instance.myserver.public_ip}"
}
```

Output Block

Terraform File (Sample Code)

Developer's tools

DevOps - CICD

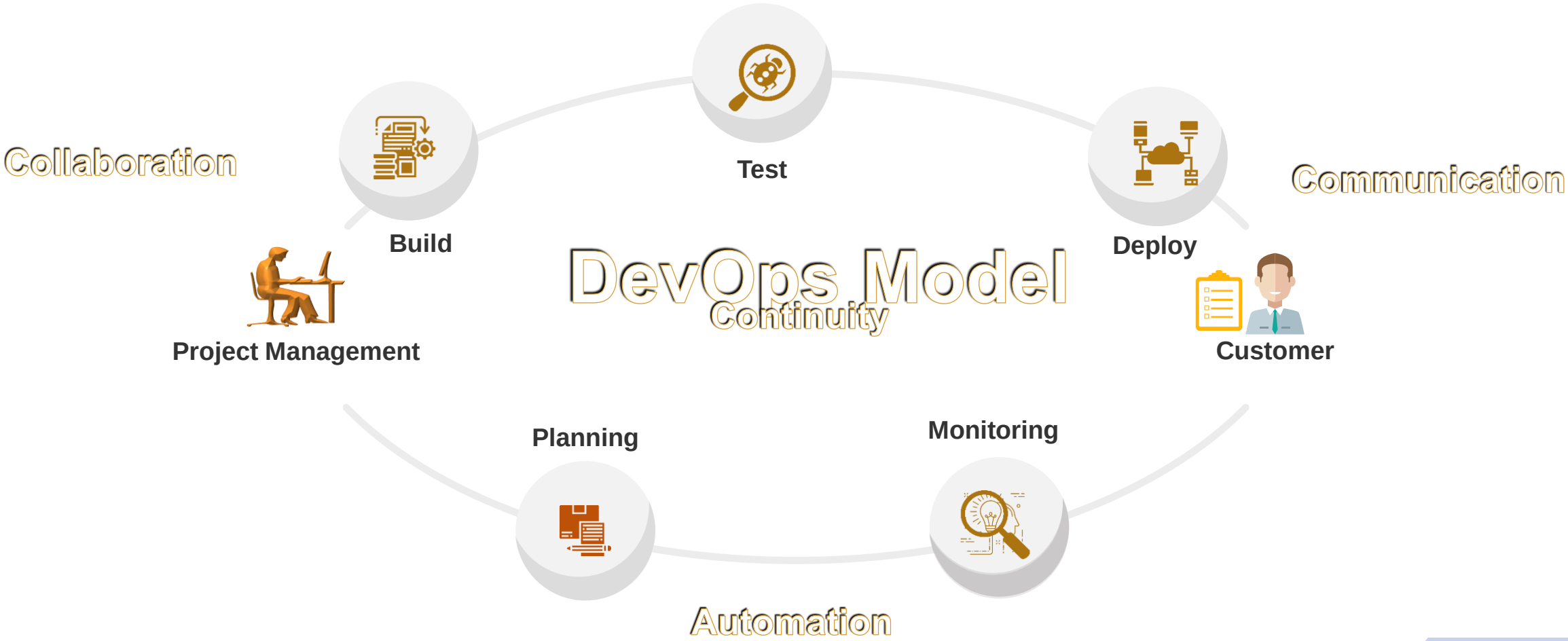
CI & CD stands for Continuous Integration and Continuous Delivery/Deployment.

CI & CD are the best practices for Software development and deployment

They enable frequent software changes to be applied while maintaining your system and application stability

AWS, Netflix, Google, facebook, Microsoft have reached to the top of this approach to release code, successfully applying thousands of changes per day without any impact

Note: AWS did 50 million deployment per year which is 1 deployment per second


Collaboration
Build
Test
Deploy
Communication
Project Management
DevOps Model
Continuity
Customer
Planning
Monitoring
Automation

Continuous Feedback

Continuous Improvement

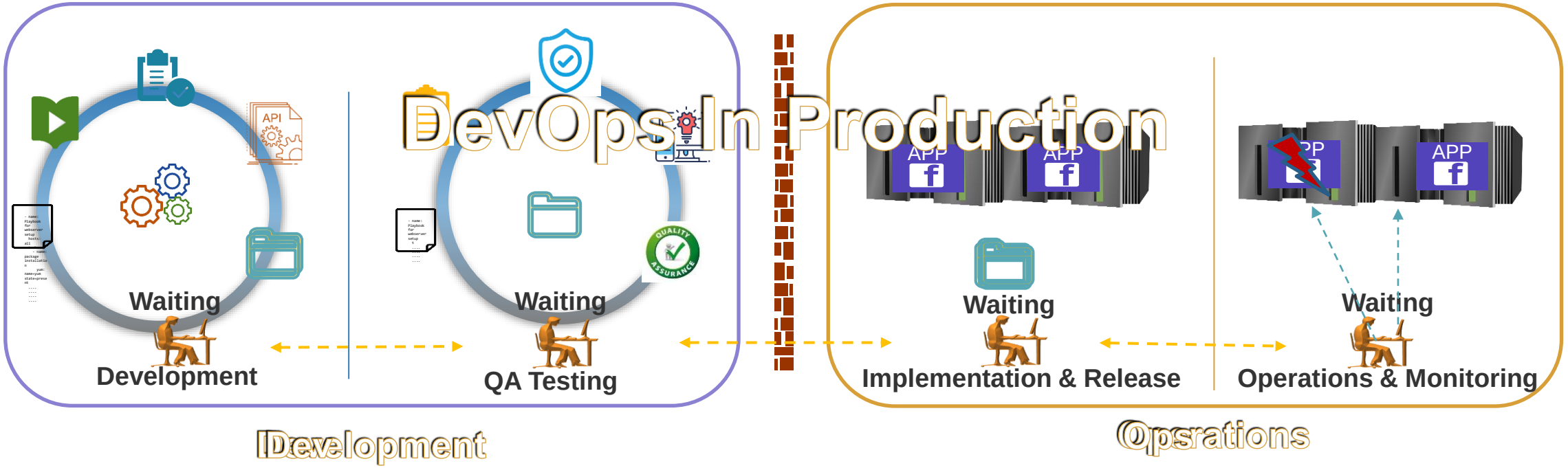
Continuous Planning

Continuous Integration

Continuous Delivery

Continuous Deployment

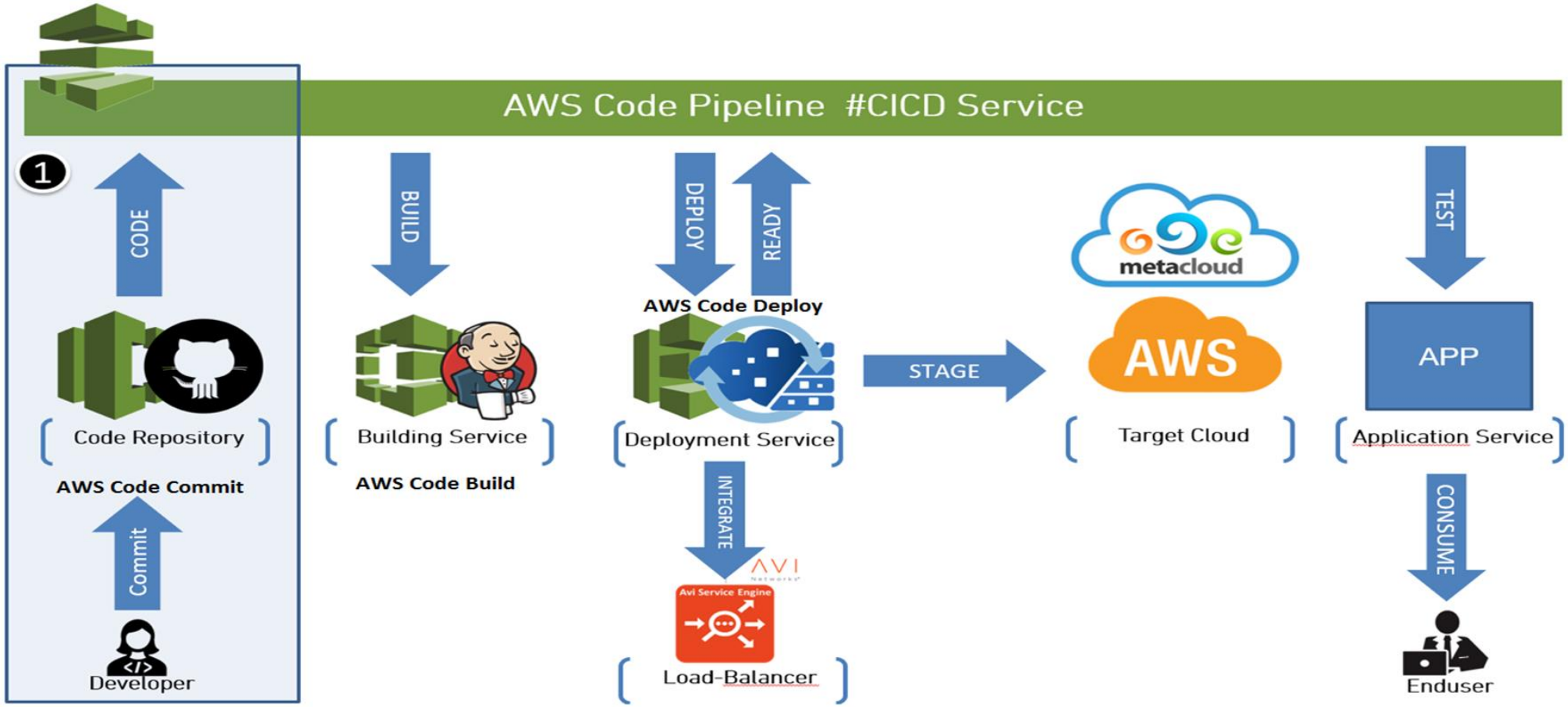
Continuous Monitoring



Development

Operations

DevOps - CICD


AWS Code Pipeline #CICD Service
1
CODE
Code Repository
AWS Code Commit
Commit
Developer
BUILD
Building Service
AWS Code Build
DEPLOY
READY
AWS Code Deploy
Deployment Service
INTEGRATE
AVI
Avi Service Engine
Load-Balancer
STAGE
metacloud
AWS
Target Cloud
TEST
APP
Application Service
CONSUME
Enduser

DevOps - CICD

CI & CD stands for Continuous Integration and Continuous Delivery/Deployment.

CI & CD are the best practices for Software development and deployment

They enable frequent software changes to be applied while maintaining your system and application stability

AWS, Netflix, Google, facebook, Microsoft have reached to the top of this approach to release code, successfully applying thousands of changes per day without any impact

Note: AWS did 50 million deployment per year which is 1 deployment per second

Code Commit

Version Control Service for Team Software development

Fully managed, Secure, highly scalable & durable, Source control service

Highly Secure - Private Git Repository, Encryption in place

Store anything – Code, files, Binaries (no limit on file/repo size & type)

Compatible with Existing Git-based tools.

Encrypted at rest as well as in transit. SSH/HTTPS protocols to connect repo.

Redundancy across Availability zones

Integrated with other AWS Services tightly. You can put notifications/ trigger lambda function/events, based on code upload/pull.

Lab: Code Commit

Create a CodeCommit Repository:
In this exercise, you will create a CodeCommit Repository through GUI console and will be accessing this Repo through Windows/Linux system.
1.Log in to the AWS Management Console and click on “Create repository” under AWS CodeCommit dashboard.
2.Provide a name and description to repo and set notifications in next screen.
3.Click create and wait until it finishes
4.Install Git (1.7.9 or later supported) without the Git Credential Manager utility.
5.Install the AWS CLI.

Lab: Code Commit

Create a CodeCommit Repository:

Generate ssh/https credentials in IAM from root account for that user.
Assign below policies to your IAM user:
AWSCodeCommitFullAccess
IAMSelfManageServiceSpecificCredentials
IAMReadOnlyAccess
Steps to clone your repo:
git config --global credential.helper "!aws codecommit credential-helper $@"
git config --global credential.UseHttpPath true
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/[Your Repo]
Use git push/pull to work with Repo. Or use eclipse to modify repo.

Lab: Code Commit

Working with CodeCommit Repository:

1. Generate ssh/https credentials in IAM from root account for that user.
2. Use git push/pull to work with Repo. Or use eclipse to modify repo.
3. Few git commands to test/start:

   git clone https://git-codecommit.us-east2.amazonaws.com/v1/repos/Demo-Repo

   Echo “test” > testfile

   git add testfile

   git commit –am “added testfile”

   git push origin master

4. To upload to another branch:

   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/Repo -b dev

   git push origin dev

5. Pull in the new branch (**git fetch origin**).

   (**git branch --all** displays a list of all branches for the repository).

6. Switch to the new branch (**git checkout MyNewBranch**).
7. **git status** or **git branch** for status
8. View the list of commits in the branch (**git log**).

Now you are good to Manage the GIT repo

Code Build

Fully managed build service in the cloud.

Compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.

It provides prepackaged build environments for the most popular programming languages and build tools such as Apache Maven, Gradle, and more.

You can also customize build environments to use your own build tools

Scales automatically to meet peak build requests.

You can run AWS CodeBuild by using the AWS CodeBuild, SDK, CLI or AWS CodePipeline console.

You are charged based only on the number of minutes it takes to complete your build.

Code Build

Benefits over traditional Build tools:

Fully Managed Build Service

Continuous Scaling

Pay as You Go

Customized build environments

Enables Continuous Integration and Delivery

Secure (Encryption in place)

IAM integrated – Access Management enabled

Code Deploy

- Fully managed Deployment service in the cloud.
- AWS CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, or Serverless Lambda functions.
- Scaling can be done to any extent
- No additional charge for code deployments to Amazon EC2 or AWS Lambda. But $0.02 per on-premises instance update.
- AWS Code Deploy makes it easier for you to:

  Rapidly release new features.

  Avoid downtime during application deployment.

  Handle the complexity of updating your applications, without many of the risks associated with error prone manual deployments.

Code Deploy

Benefits of Code Deploy:

Manage Server and Serverless applications.

Automated deployments across dev, test and prod environment

Minimize downtime with in-place deployment & blue green approach

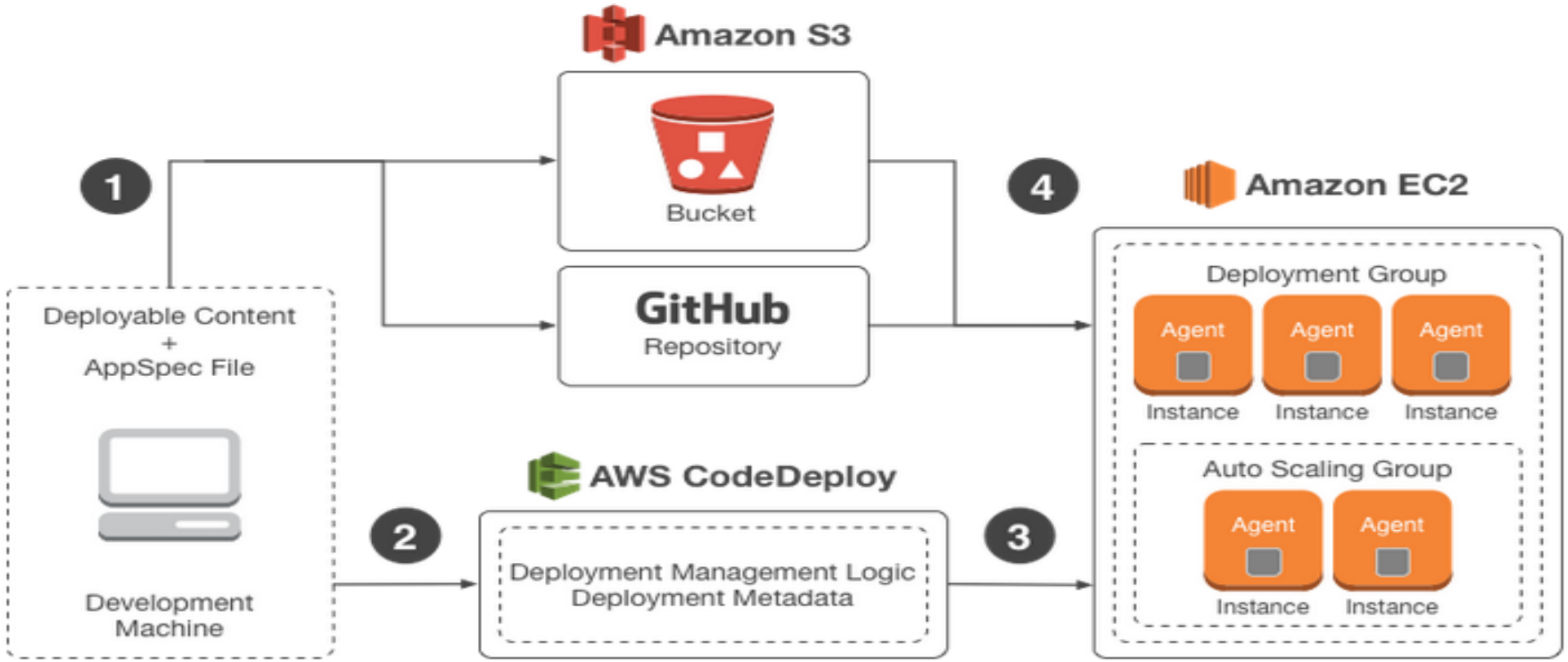
Stop and roll back facility

Centralized control.

Easy to adopt.

Scales along with your infra

AWS Code Deploy Execution Flow



AWS Code deploy

Blue/Green Deployment model

- CodeDeploy service will create equal number of standby servers and deploy application new version on same.
- Traffic on new systems can be migrated partially or all at once.
- Once new application is live and validated, older can be destroyed.
- Only Ec2 and lambda support Blue/Green Model.

In-Place Deployment model

- The application on each instance in the deployment group is stopped, the latest application revision is installed & started.
- Only EC2/On-Premises compute platform can use in-place deployments.

AWS Code Pipeline

AWS CodePipeline is a **Continuous delivery service**

Being used to automate, visualize and channelize the software releases

AWS CodePipeline automates the steps required to release your software changes continuously.

Establish a consistent release process

Speed up delivery while improving quality

AWS CodePipeline costs $1 per active pipeline per month. To encourage experimentation, pipelines are free for the first 30 days after creation.

View pipeline history and progress

AWS Code Pipeline Tutorial

https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-simple-s3.html

AWS Codestar

Service to create & manage software development project on AWS

Start new software projects on AWS in minutes using templates for web applications, web services and more, for various project types and programming languages.

AWS CodeStar takes care of the setup, all your project resources are created & configured to work together.

Integrated user management for developers access

Visualize, operate, and collaborate on your projects in one place

Good for creating Generic applications.

AWS Codestar Tutorial

Create a CodeStar Project:

In this exercise, you will create a CodeStar Project and will be managing same through console:
Log in to the AWS Management Console and click on “Create Project” under AWS CodeStar dashboard.
Select Application category, Programming Language & AWS Services which you want to use, on left hand side.
Click on the shortlisted template which you want to run
Provide basic details & select repository source
Click Create Project on next screen.
Wait for 30 minute and watch all resources being created/deployed on your behalf from template
Delete the project

AWS Pipeline Labs

https://aws.amazon.com/blogs/devops/continuous-deployment-to-kubernetes-using-aws-codepipeline-aws-codecommit-aws-codebuild-amazon-ecr-and-aws-lambda/

https://aws.amazon.com/blogs/devops/ci-cd-on-amazon-eks-using-aws-codecommit-aws-codepipeline-aws-codebuild-and-fluxcd/

Serverless Computing

Serverless Computing

Build and run applications without thinking about servers

Serverless applications don't require you to provision, scale, and manage any servers

Everything required to run and scale your application with high availability is handled for you.

Serverless applications have three main benefits.

No server management

Flexible scaling

Automated high availability

Lambda

AWS Lambda – Run code without thinking about server

Compute Service: AWS Lambda is a server-less compute service.

Amazon's Infrastructure: It runs your code in response to events and automatically manages the compute resources for you.

Function as a Service: Automatically run code in response to events (like s3, dynamodb, API gateway, http call etc) which is called triggers in Lambda.

Autoscaling and DR: Runs the code in high-availability compute infrastructure with automatic scaling capabilities.

Supported by: Java, Python, NodeJS and few more stacks

Lambda

Your code, is a function in AWS LAMBDA

Forgot about the servers, its all handled by AWS and you just need to pay for the time your code is running.

Functions should be stateless, but you have the flexibility to use S3, DynamoDB (other AWS services) to store stateful functions.

AWS LAMBDA "Serverless Service", In reality "Still there are servers!" but managed by AWS with Autoscaling enabled by default.

Pricing: Free tier is available for testing
with 1M requests
3M seconds of compute time

You are charged for every 100ms your code executes and the number of times your code is triggered. No usage means No cost to pay.

Lambda

EC2:

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

Lambda

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

Lab

Go to Lambda console

Select hello-world blueprint

Give Function name and select Node.js

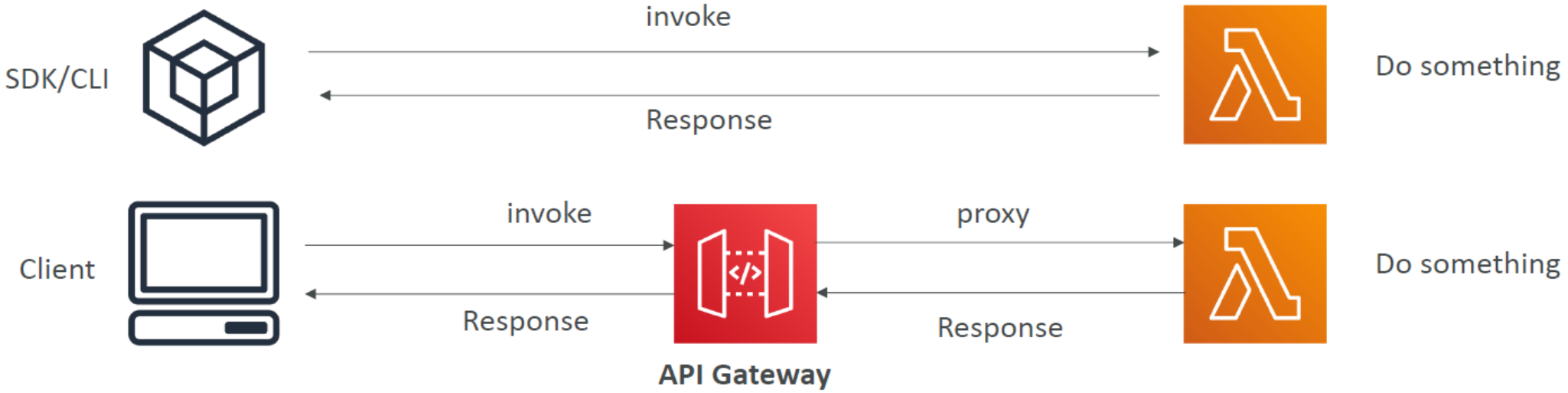
Select Role (make it simple for the first time)

Create Function, Save and Run with key values

Lambda Facts

Synchronous Invocation:

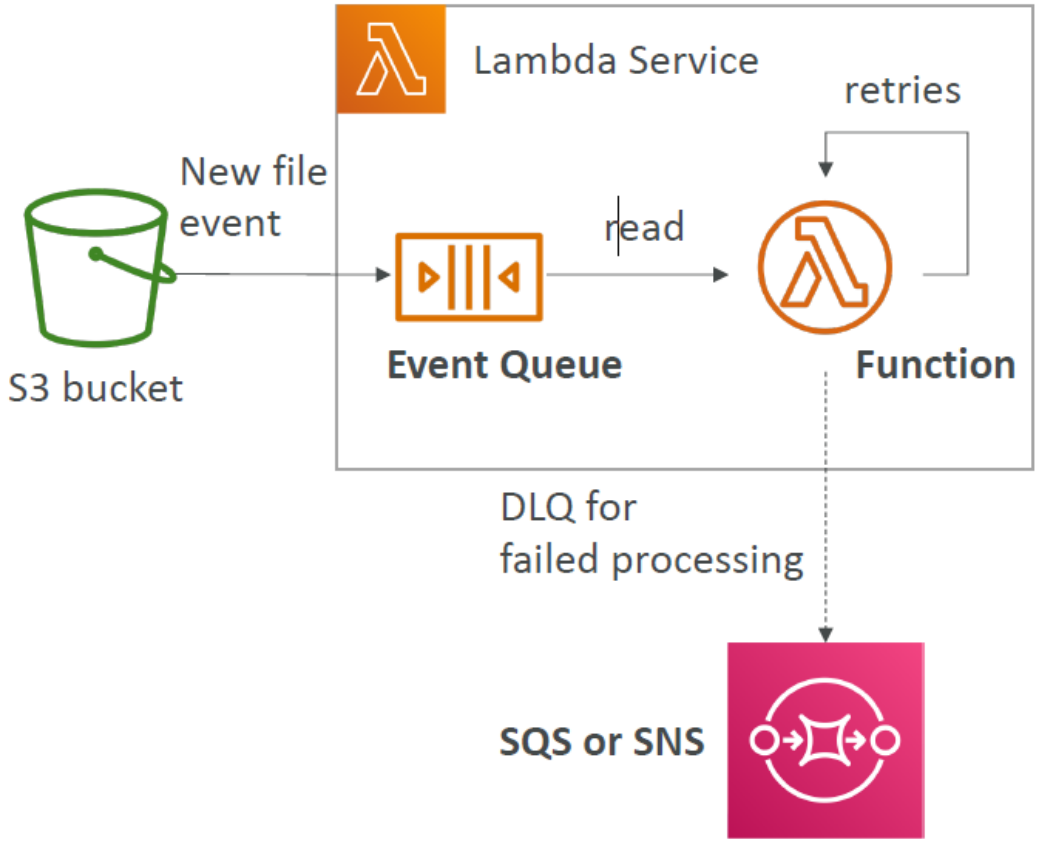
Synchronous: CLI, SDK, API Gateway, Application Load Balancer
- Results is returned right away
- Error handling must happen client side (retries, exponential backoff, etc…)



Lambda Facts

Asynchronous Invocation:

S3, SNS, CloudWatch Events…
- The events are placed in an Event Queue
- Lambda attempts to retry on errors
- 3 tries total
- 1 minute wait after 1st , then 2 minutes wait
- Make sure the processing is idempotent (in case of retries)
- If the function is retried, you will see duplicate logs entries in CloudWatch Logs
- Can define a DLQ (dead-letter queue) – SNS or SQS – for failed processing (need correct IAM permissions)
- Asynchronous invocations allow you to speed up the processing if you don't need to wait for the result (ex: you need 1000 files processed)



Lambda Facts

Throttle/Concurrency:

Throttling is concurrent execution limit control. You can throttle your function both the account level and the function level. Use throttling:
In case indefinite scaling in backend is not desired
to have a controlled cost factor
to regulate how long it takes you to process a batch of events
to match function limit with a downstream/backend resource i.e. sql concurrent session limits.
set the concurrency to 0, to stop all invocations (in case require same to hold execution temporarily)
Handle it with care. If you are reserving the concurrency for one function, rest function will be limited to use the concurrent executions.

Note: By default, 1000 concurrent executions allowed per account, out of which 100 concurrent executions must remain unreserved.

Lambda Facts

Versioning of your Function code can be done.

You can export your function to work locally or save/share with others.

You can allocate/limit memory to your function (Cost factor associated).

Function timeout can be set to avoid cost, due to code errors.

AWS Lambda automatically retry (twice) failed executions for asynchronous (event based) invocations.

AWS Lambda directs events that cannot be processed to the specified Amazon SNS topic or Amazon SQS queue. Functions that don't specify a DLQ will discard events after they have exhausted their retries.

Cloud trail can be created for Lambda functions invocations logging.

Lambda Facts

Test events can be configured for Lambda functions, to trigger its execution.

These events can be an input from Alexa, CloudFront, Code commit, SNS etc.. Refer test events for latest list of supported events.

Traffic Shifting Using Aliases:
You can create aliases to shift your traffic to two different versions simultaneously.
You can define the traffic percentage weightage for each version.
You can watch the result-executed-version in cloud watch logs.
Mainly used to check the new version in production, before fully integrating same.

Invocation errors, Throttled invocations, DLQ errors etc. can be monitored through the monitoring tab. Same should be regularly checked.

Lab: Lambda Function

```
'use strict';
console.log('Loading function');
exports.handler = (event, context, callback) => {

        let min = 0;
        let max = 10;

        let RandomNumberGenerator = Math.floor(Math.random() * max) + min;

        callback(null, RandomNumberGenerator);
};
```

Note: Lambda Function Name and name mentioned in the code should be same (like RandomNumberGenerator) is my Lambda Function name and the same used in the code.

Lab: Lambda - Snapshot

Open Lambda console and create a new python function.

Paste below code from S3 to Lambda

https://s3.us-east-2.amazonaws.com/newgde/Lambda-Snapshot-creation.txt

Now, create test case and run the lambda function.

Check the snapshot being created for servers with Tagging “Environment” type as “production”.

Lab: Lambda – S3 Trigger

Lets create a simple get function with S3 and understand how Lambda works with S3 events.

Go to Lambda Console
Create a new function with Python 2.7 and S3-get-object-python
Configure a trigger, select “yourbucket”, event type all Objected Created
Prefix images/, object type and Enable trigger
Create a function with any name
Handler lambda_function.lambda_handler
Create new role from template
Role name could be any relevant name
Create function with s3 object read only permissions.

API Gateway

API Gateway


REST API
PROXY REQUESTS
CRUD
Client
API Gateway
Lambda
DynamoDB

API Gateway

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

API Gateway Integrations

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…

AWS Service

- Expose any AWS API through the API Gateway?
- Example: start an AWS Step Function workflow, post a message to SQS
- Why? Add authentication, deploy publicly, rate control…

API Gateway Integrations

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

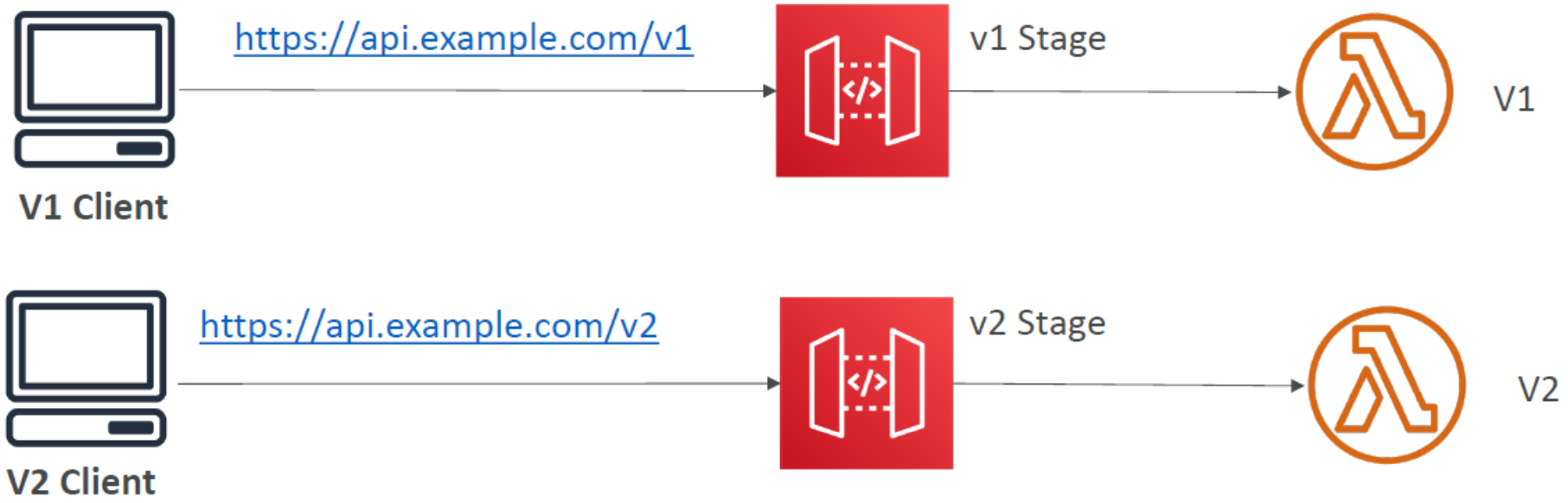
Regional:

- 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

API Gateway Integrations



API Gateway Stage Variables

Stage variables are like environment variables for API Gateway

Use them to change often changing configuration values

They can be used in:

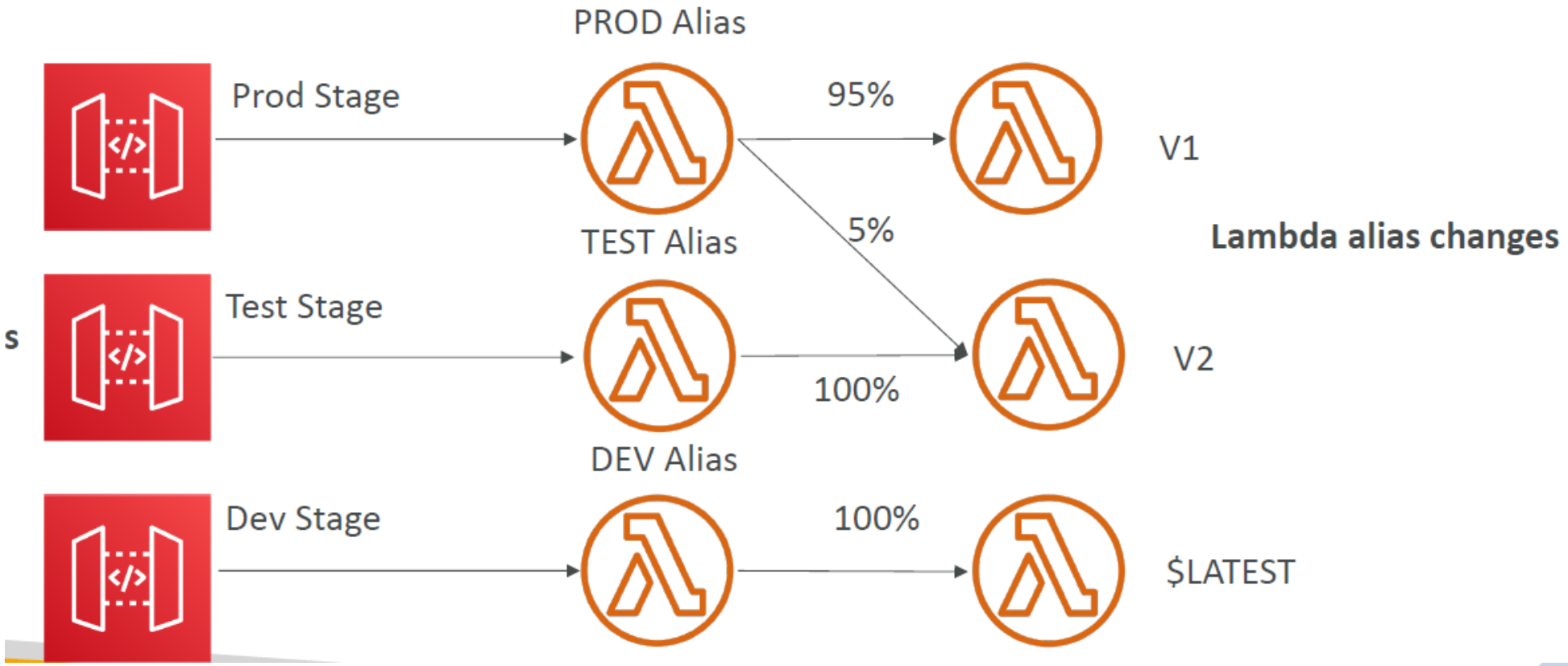
- Lambda function ARN
- HTTP Endpoint
- Parameter mapping templates

Use cases:

- Configure HTTP endpoints your stages talk to (dev, test, prod…)
- Pass configuration parameters to AWS Lambda through mapping templates

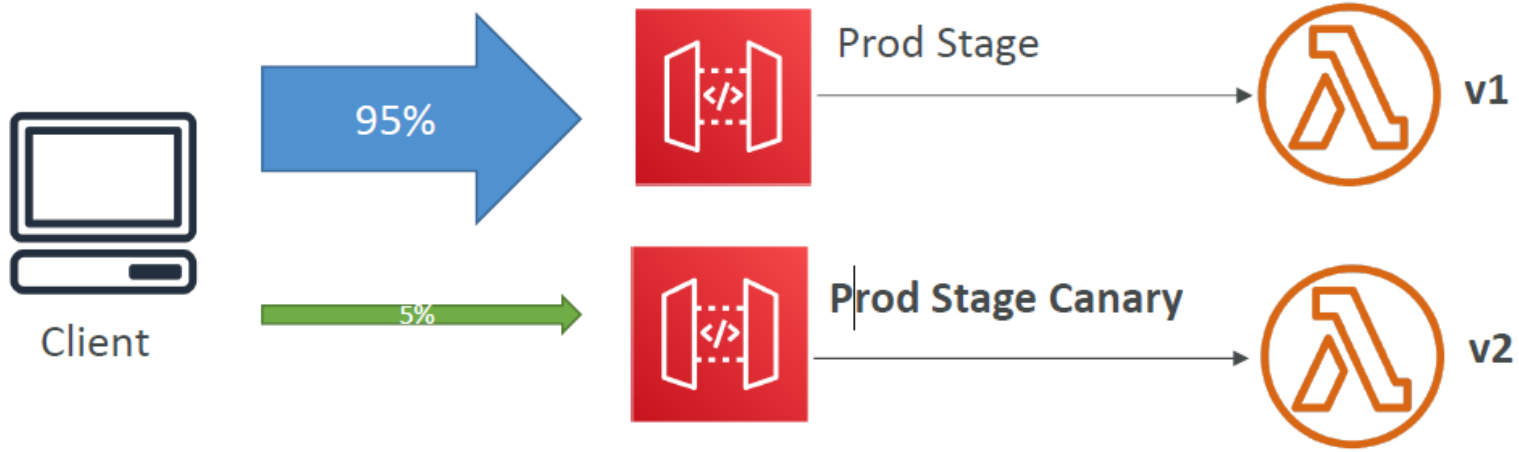
Stage variables are passed to the ”context” object in AWS Lambda

Using stages and Alias



Canary Deployment

- Possibility to enable canary deployments for any stage (usually prod)
- Choose the % of traffic the canary channel receives



- Metrics & Logs are separate (for better monitoring)
- Possibility to override stage variables for canary
- This is blue / green deployment with AWS Lambda & API Gateway

Mapping Templates

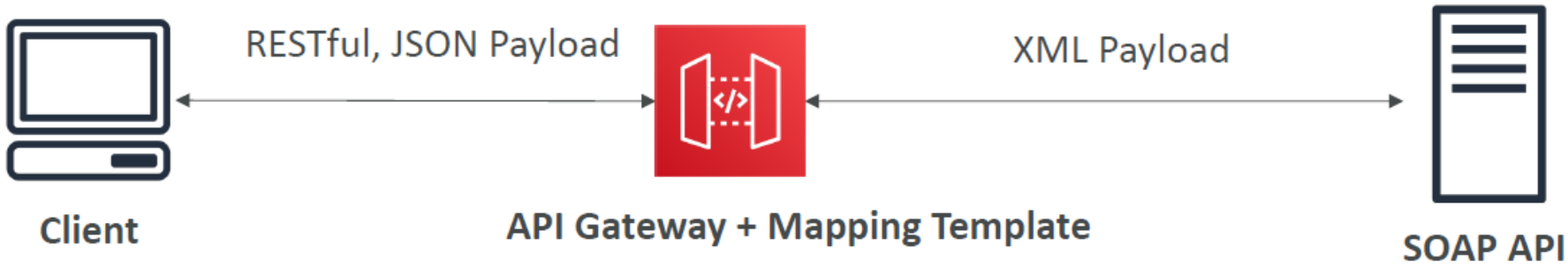
- Mapping templates can be used to modify request / responses
- Rename / Modify query string parameters
- Modify body content
- Add headers
- Uses Velocity Template Language (VTL): for loop, if etc…
- Filter output results (remove unnecessary data)

Mapping Example: JSON to XML with SOAP

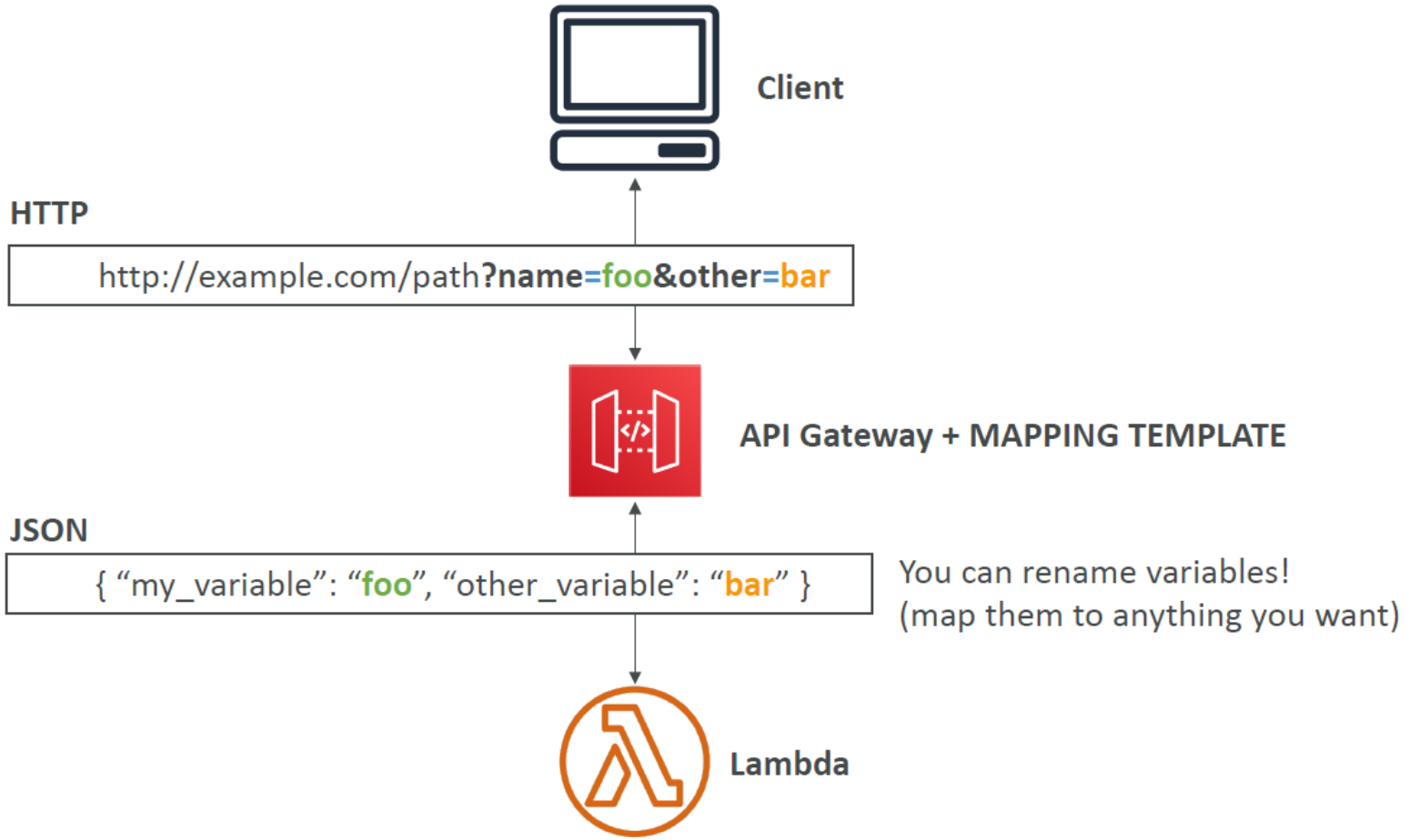
SOAP API are XML based, whereas REST API are JSON based

In this case, API Gateway should:

- Extract data from the request: either path, payload or header
- Build SOAP message based on request data (mapping template)
- Call SOAP service and receive XML response
- Transform XML response to desired format (like JSON), and respond to the user



Mapping Example: JSON to XML with SOAP



AWS API Gateway Swagger / Open API spec

Common way of defining REST APIs, using API definition as code

- Import existing Swagger / OpenAPI 3.0 spec to API Gateway
  - Method
  - Method Request
  - Integration Request
  - Method Response
  - + AWS extensions for API gateway and setup every single option
- Can export current API as Swagger / OpenAPI spec
- Swagger can be written in YAML or JSON
- Using Swagger we can generate SDK for our applications

AWS API Gateway Swagger / Open API spec

Caching reduces the number of calls made to the backend

- Default TTL (time to live) is 300 seconds

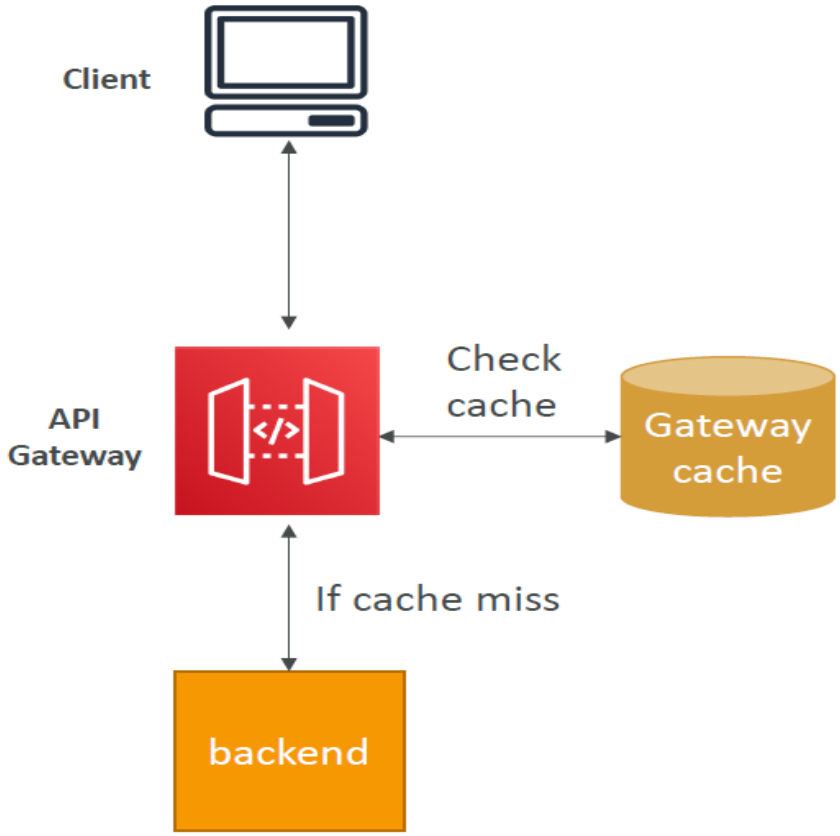
(min: 0s, max: 3600s)

- Caches are defined per stage
- Possible to override cache settings per

method

- Cache encryption option
- Cache capacity between 0.5GB to 237GB
- Cache is expensive, makes sense in

production, may not make sense in dev / test



API Gateway – Usage Plans & API Keys

If you want to make an API available as an offering ($) to your customers

- Usage Plan:
  - who can access one or more deployed API stages and methods
  - how much and how fast they can access them
  - uses API keys to identify API clients and meter access
  - configure throttling limits and quota limits that are enforced on individual client
- API Keys:
  - alphanumeric string values to distribute to your customers
  - Ex: WBjHxNtoAb4WPKBC7cGm64CBibIb24b4jt8jJHo9
  - Can use with usage plans to control access
  - Throttling limits are applied to the API keys
  - Quotas limits is the overall number of maximum requests

API Gateway – Metrics

- Metrics are by stage, Possibility to enable detailed metrics
- **CacheHitCount** & **CacheMissCount**: efficiency of the cache
- **Count**: The total number API requests in a given period.
- **IntegrationLatency**: The time between when API Gateway relays a request to the backend and when it receives a response from the backend.
- **Latency**: The time between when API Gateway receives a request from a client and when it returns a response to the client. The latency includes the integration latency and other API Gateway overhead.
- **4XXError** (client-side) & **5XXError** (server-side)

API Gateway – Security

IAM:

- Great for users / roles already within your AWS account, + resource policy for cross account
- Handle authentication + authorization
- Leverages Signature v4

Custom Authorizer:

- Great for 3rd party tokens
- Very flexible in terms of what IAM policy is returned
- Handle Authentication verification + Authorization in the Lambda function
- Pay per Lambda invocation, results are cached

Cognito User Pool:

- You manage your own user pool (can be backed by Facebook, Google login etc…)
- No need to write any custom code
- Must implement authorization in the backend

AWS Auditing

Cloud Trail

A monitoring service for AWS Account Activity

You can use Cloud Trail to view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure.

It logs all activities made through anywhere i.e AWS Management Console, AWS Command Line Interface, AWS SDKs and APIs.

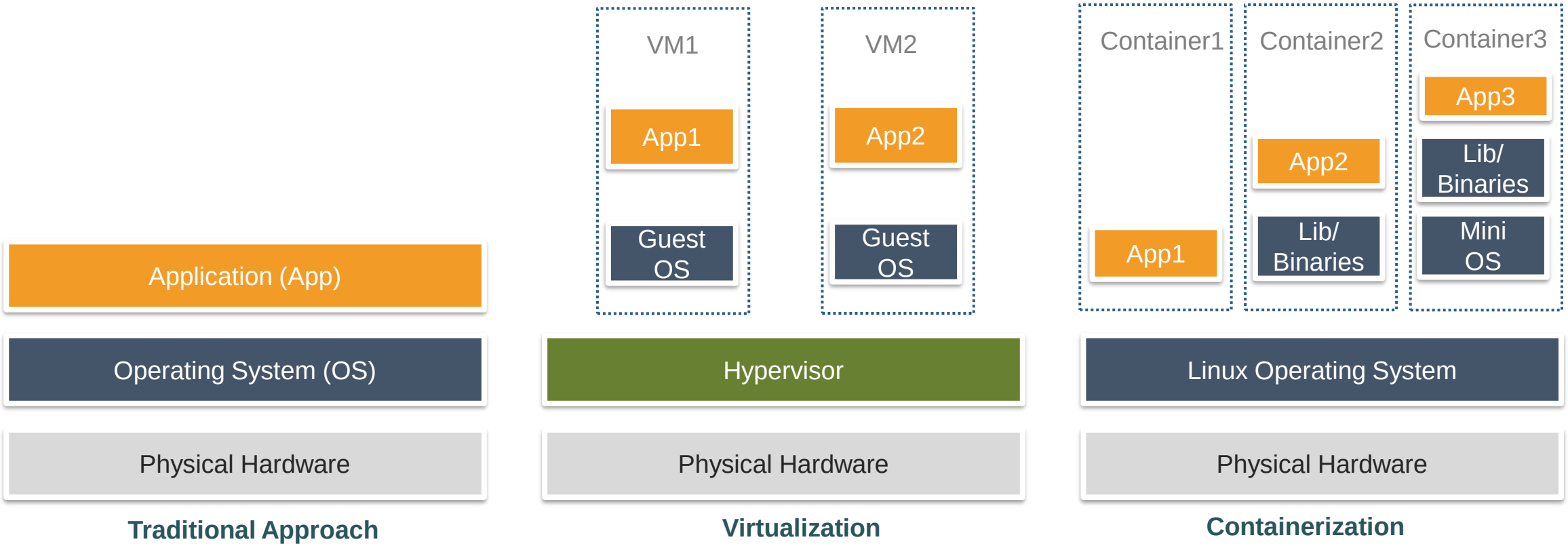
By default Maintains data for last 90 days. You can download the trail data every three months to have backup.

You can Create a trail to retain a record of specific events (As longer as you want)

Default 90 days logging is free, creating trail will be chargeable.

Containers on AWS

Containers


Application (App)
Operating System (OS)
Physical Hardware
Traditional Approach
VM1
App1
Guest OS
VM2
App2
Guest OS
Hypervisor
Physical Hardware
Virtualization
Container1
App1
Container2
App2
Lib/ Binaries
Container3
App3
Lib/ Binaries
Mini OS
Linux Operating System
Physical Hardware
Containerization

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

Containers Facts

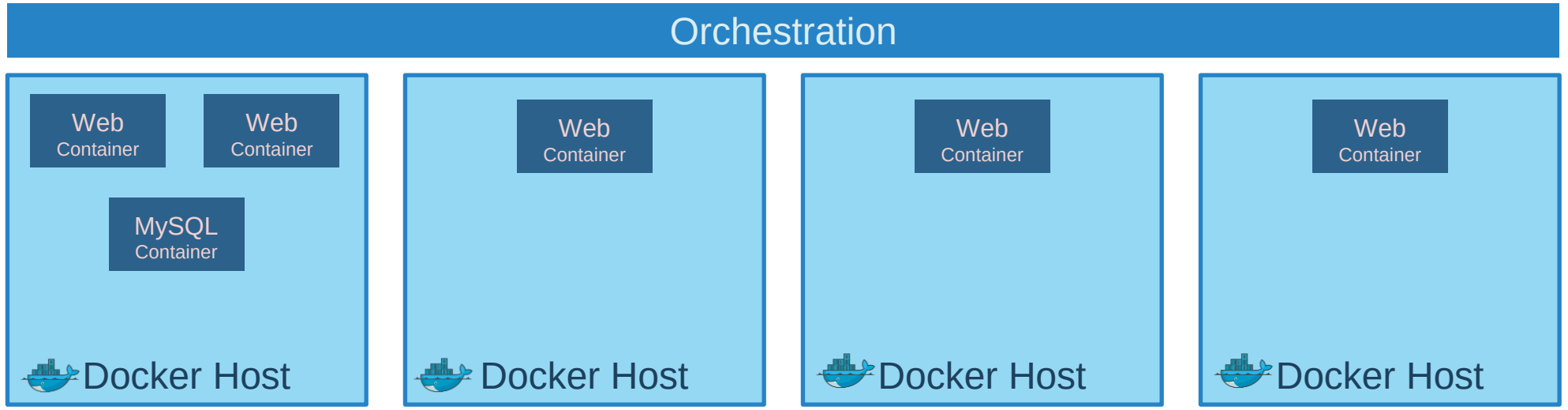
A Docker container is a standardized unit of software development, containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc. Containers are created from a read-only template called an *image*.

Images are typically built from a Dockerfile, a plain text file that specifies all of the components that are included in the container.

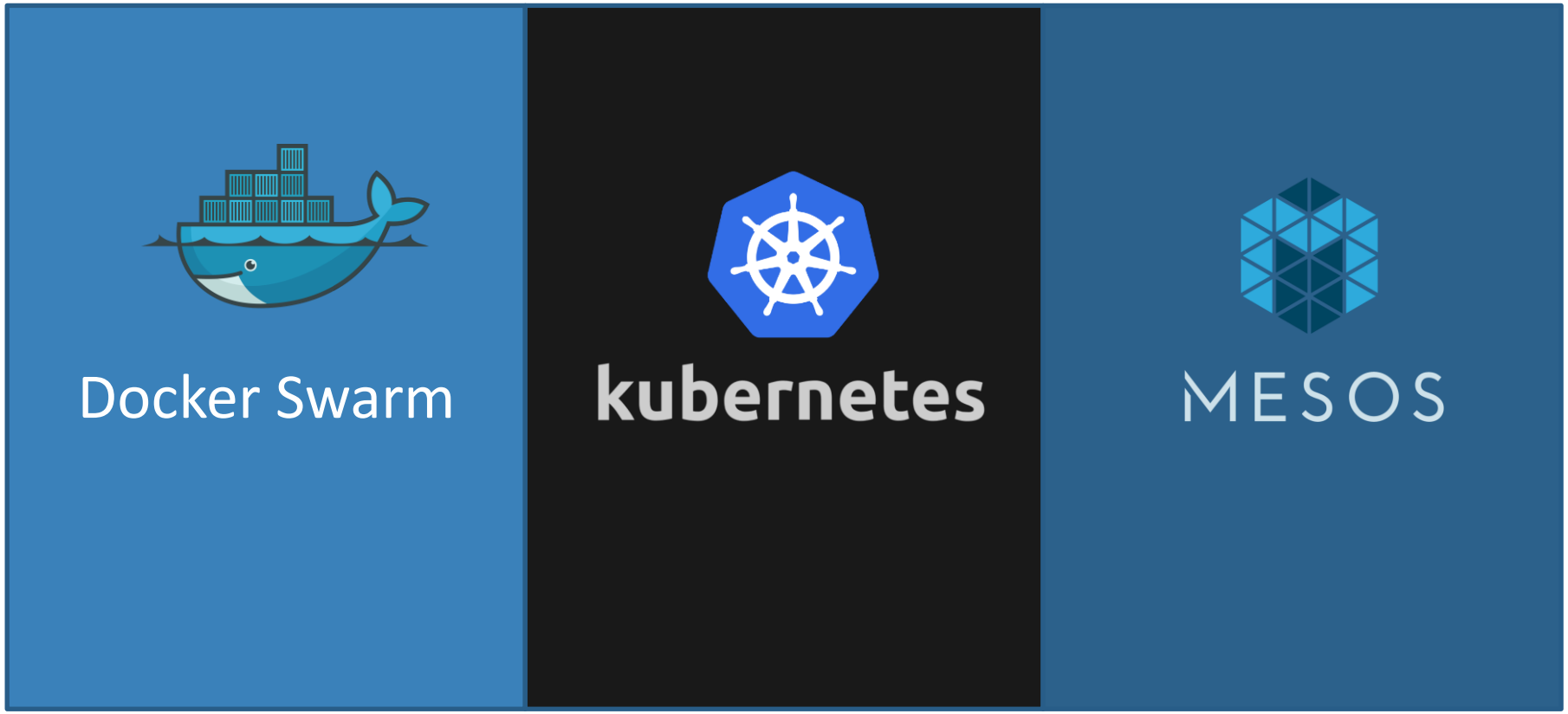
Docker images are stored in Docker Repositories

- 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)

Container Orchestration



Orchestration Technologies



Containers on AWS

To manage containers, we need a container management platform

Three choices on AWS:

- ECS: Amazon's own platform
- Fargate: Amazon's own Serverless platform
- EKS: Amazon's managed Kubernetes (open source)

Elastic Containers Service

Amazon Elastic Container Service (Amazon ECS) is the Amazon Web Service you use to run Docker applications on a scalable cluster.

Amazon ECS makes it easy to deploy, manage, and scale Docker containers running applications, services, and batch processes

Amazon ECS places containers across your cluster based on your resource needs and is integrated with familiar features like Elastic Load Balancing, EC2 security groups, EBS volumes and IAM roles

Eliminates the need for you to install, operate, and scale your own cluster management infrastructure.

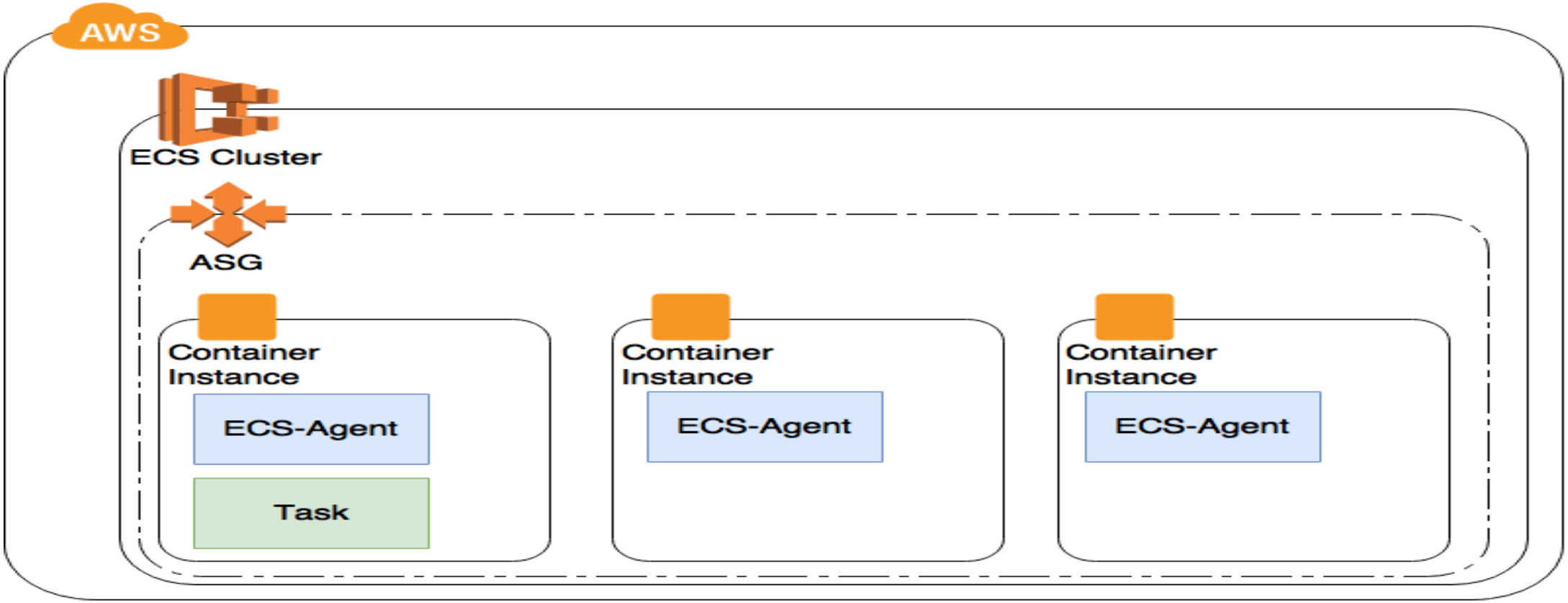
Existing ECS Cluster is Blox (Open source project for container management) based cluster.

To deploy applications on Amazon ECS, your application components must be architected to run in *containers*.

Elastic Containers Service

- ✓ No Need to create/manage the cluster environment
- ✓ Autoscaling, Clustering is automatically managed in backend
- ✓ Containers can run on any node in cluster (Across AZ too)
- ✓ Application run here as tasks (similar to pods in Kubernities)
- ✓ Cluster instances are EC2 in backend.
- ✓ ECS is using Autoscaling in Backend (ELB can be configured)

Elastic Containers Service



AWS ECS Fargate

- AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of EC2 instances.
- When you run your tasks and services with the Fargate launch type, you package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application.
- You pay for the amount of vCPU and memory resources that your containerized application requests.
- Requires Network mode to be awsvpc.
- Fargate supports only Docker Public Repo or AWS Repo. No outside private Repo is allowed for now.

Kubernetes

The Kubernetes project was started by Google in 2014.

Kubernetes builds upon a decade and a half of experience that Google has with running production workloads at scale.

Kubernetes can run on a range of platforms, from your laptop, to VMs on a cloud provider, to rack of bare metal servers.

Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure.

portable: with all public, private, hybrid, community cloud

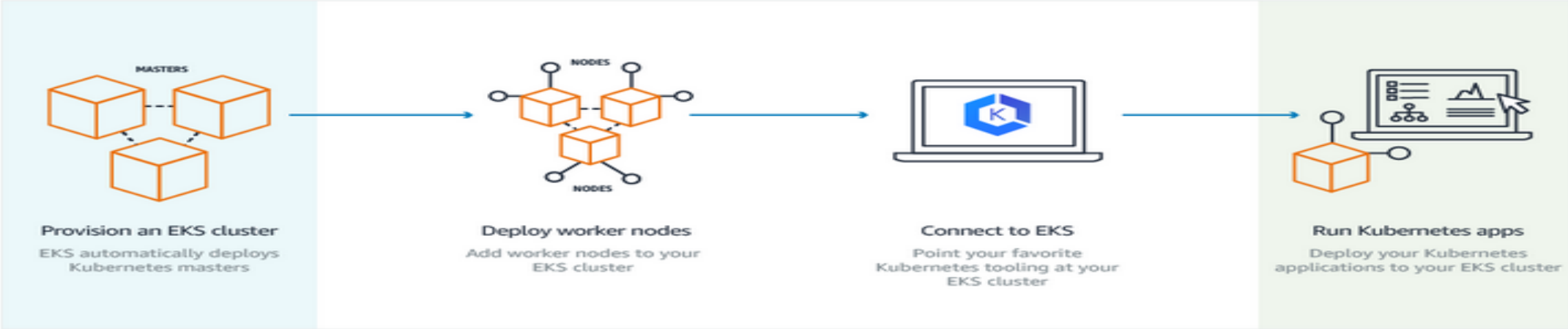
self-healing: auto-placement, auto-restart, auto-replication, auto-scaling

AWS EKS

- Amazon Elastic Container Service for Kubernetes (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to install and operate your own Kubernetes clusters.
- With Amazon EKS you get a highly-available, and secure Kubernetes control plane without needing to worry about provisioning, upgrades, or patching.
- It is certified Kubernetes conformant so you can use all existing plugins and tooling from the Kubernetes community.
- Any application running on any standard Kubernetes environment is fully compatible.

AWS EKS

- No Masters to manage, no upgrades to take care of.
- Secure by Default
- Conformant and Compatible
- Allow Hybrid Container Deployments
- Application Migration



AWS ECR

- So far we've been using Docker images from Docker Hub (public)
- ECR is a private Docker image repository
- Access is controlled through IAM (permission errors => policy)

- AWS CLI v2 login command (newer, may also be asked at the exam)
- aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin 1234567890.dkr.ecr.eu-west-1.amazonaws.com

- Docker Push & Pull:
- docker push 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
- docker pull 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest

Container Security

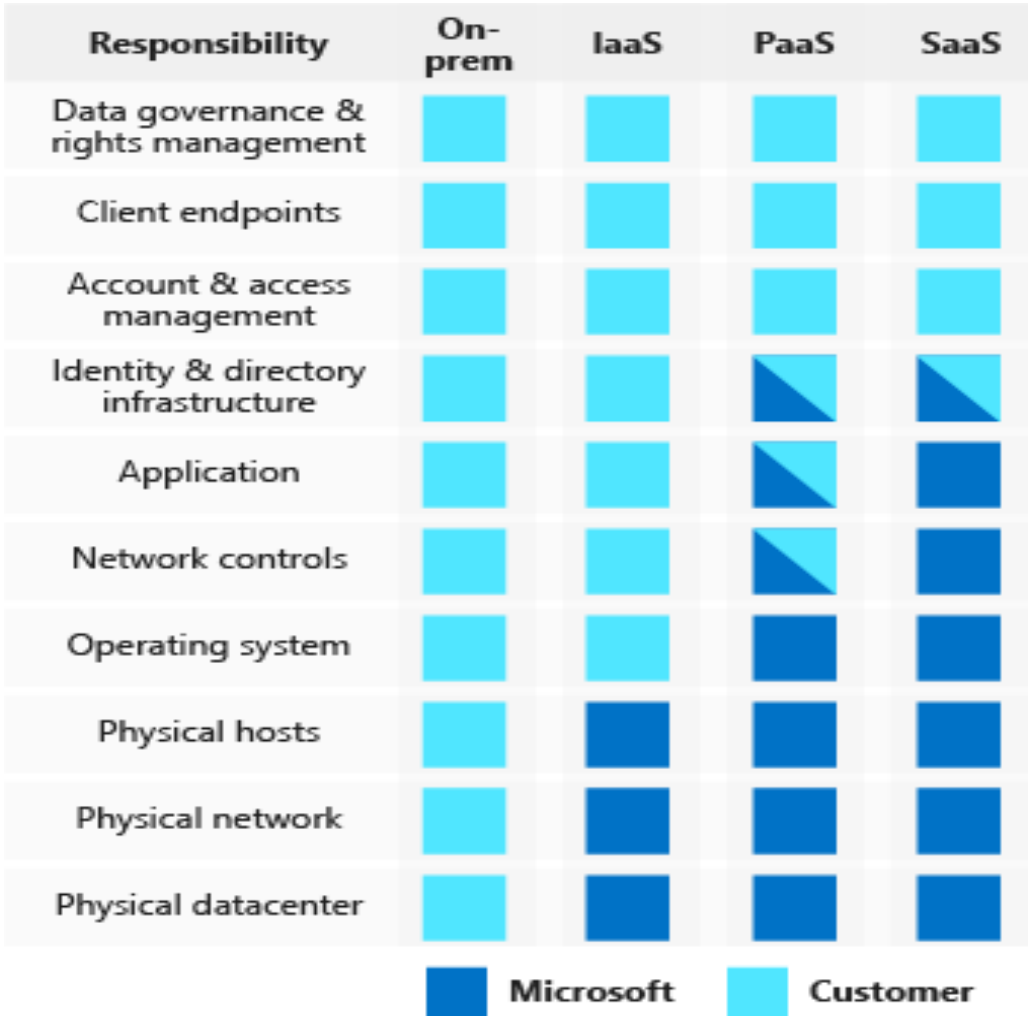
- Use Authorized Images only
- Have your outside firewall and Security group rules very rigid and strong
- Have a proper control over your Network using Service Mesh like Isitio
- Container Compatible Monitoring tools like Prometheus
- Image Scanning and Vulnerability Management
- Enable Logging and keep your logs outside of your container hosts

Further Security Concepts

Security Concepts

Encryption & Key Management: The ever green and most trusted method of security in cloud computing. Encryption can be at Rest or at Motion.
Access Control through Identity and Access Management.
Data & Media Sanitization – Data erasure, Formatting, Disk deletion are not the full-proof method of sanitization. Physical deletion is not possible in cloud. So the alternate left is **cryptographic eraser**.
Network Security: With the help of VPC, Subnetting, NACL & Security groups & Egress monitoring.
Application Security: With WAF & Physical Firewalls.
Virtualization security and hardening at different layers.
OS based security hardening
Data Security: Masking, Obfuscation, Anonymization, Tokenization

Security Responsibility



Regardless of the deployment type, Client always retain responsibility for the following:

- Data
- Endpoints
- Accounts
- Access management

Regardless of the deployment type, Azure always retain responsibility for the following:

- Physical Host
- Physical Network
- Physical dataCenter

https://forms.gle/YT1LkEYm2Vtw9aGT9

THANK YOU