

AWS for Solution Architects

Techlanders Solutions

Raman Khanna

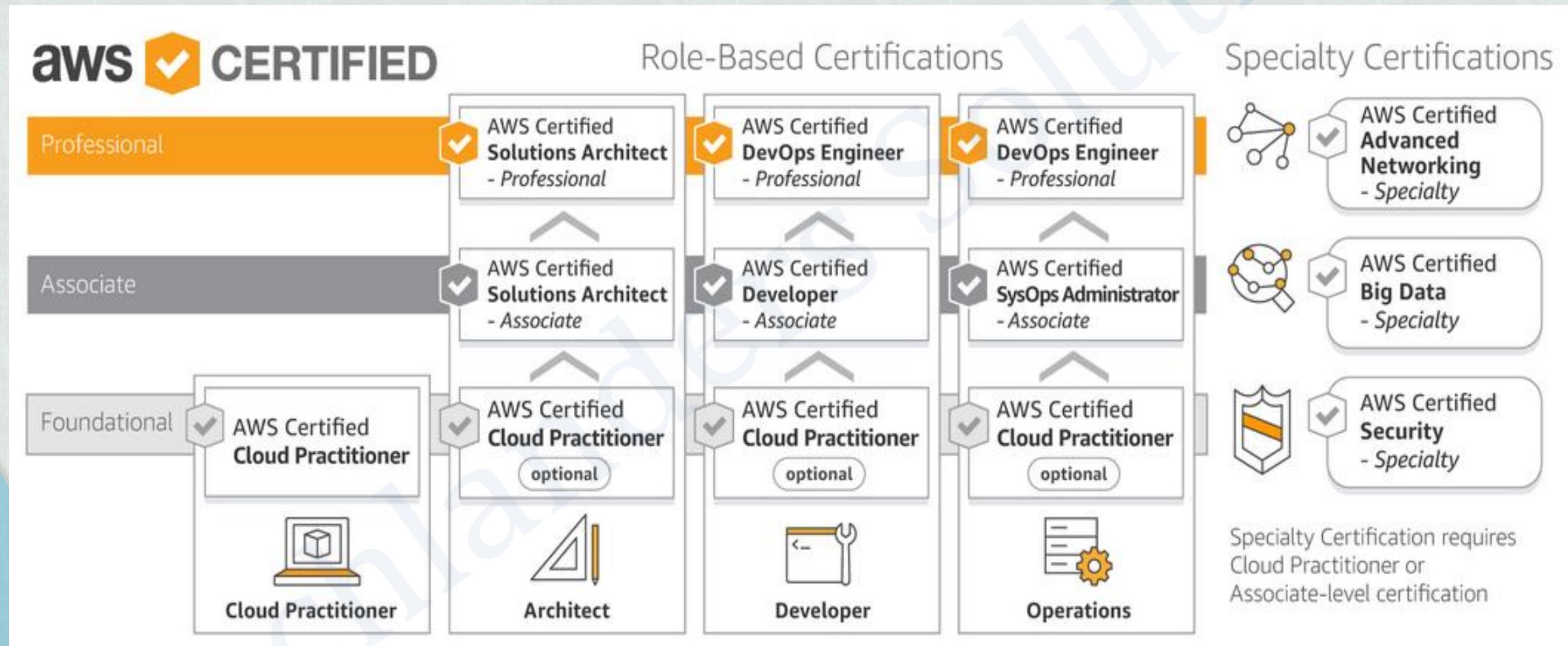
Introduction

Your Name

Background – Development / Infrastructure

Learnings on cloud/AWS

AWS Certification Roadmap



Introduction to Cloud Computing

What is Cloud?

Introduction to Cloud Computing

In simple words, Cloud computing is – Placing your data on someone else's datacenter, letting them manage underline hardware Infrastructure (optionally underline Database or applications too); while having your full control on the data, and accessing that data through Internet or dedicated network.

Cloud computing is a model for enabling **universal, on-demand** access to a **shared pool** of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be **rapidly provisioned** and **released** with **minimal management effort** on **Pay-per-use basis**.

Free available cloud Examples:

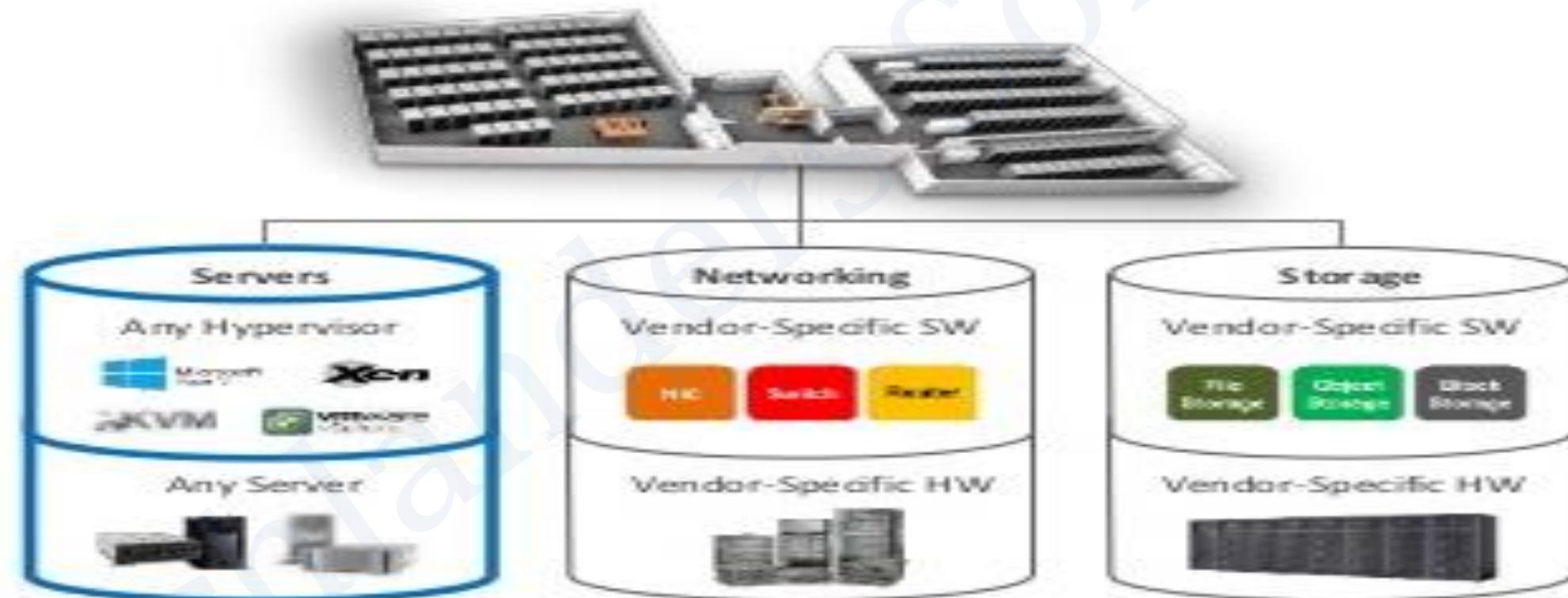
Gmail, IRCTC, WhatsApp/Facebook

Paid available cloud Examples:

AWS, Azure(Microsoft), Oracle Cloud

Traditional DataCenters

Traditional Data Center



Traditional DataCenters

- Main issues with Traditional IT Infrastructure.
 - Infrastructure is not a core business
 - Hard to Scale
 - Dedicated Infrastructure teams
 - Dedicated Datacenters
 - Dependency on vendors (servers, switches, cables etc.)
 - Underutilized Resources
 - High Cost
 - Difficult Capacity Planning
 - On-Spot demands were hard to manage
 - Provisioning resources was very time consuming

Why cloud?

- To overcome all of the discussed challenges, IT infrastructure domain drifted towards Service based model which is a real “cloud computing”
 - No Dedicated Datacenter
 - No Different Infrastructure Teams
 - Higher/Faster Scalability
 - Elasticity
 - Pay per use model
 - Option to adopt high availability
 - Better performance
 - Instant provisioning
 - Optimized use of resources
 - On demand scaling to any extent
 - No to worry about capacity planning

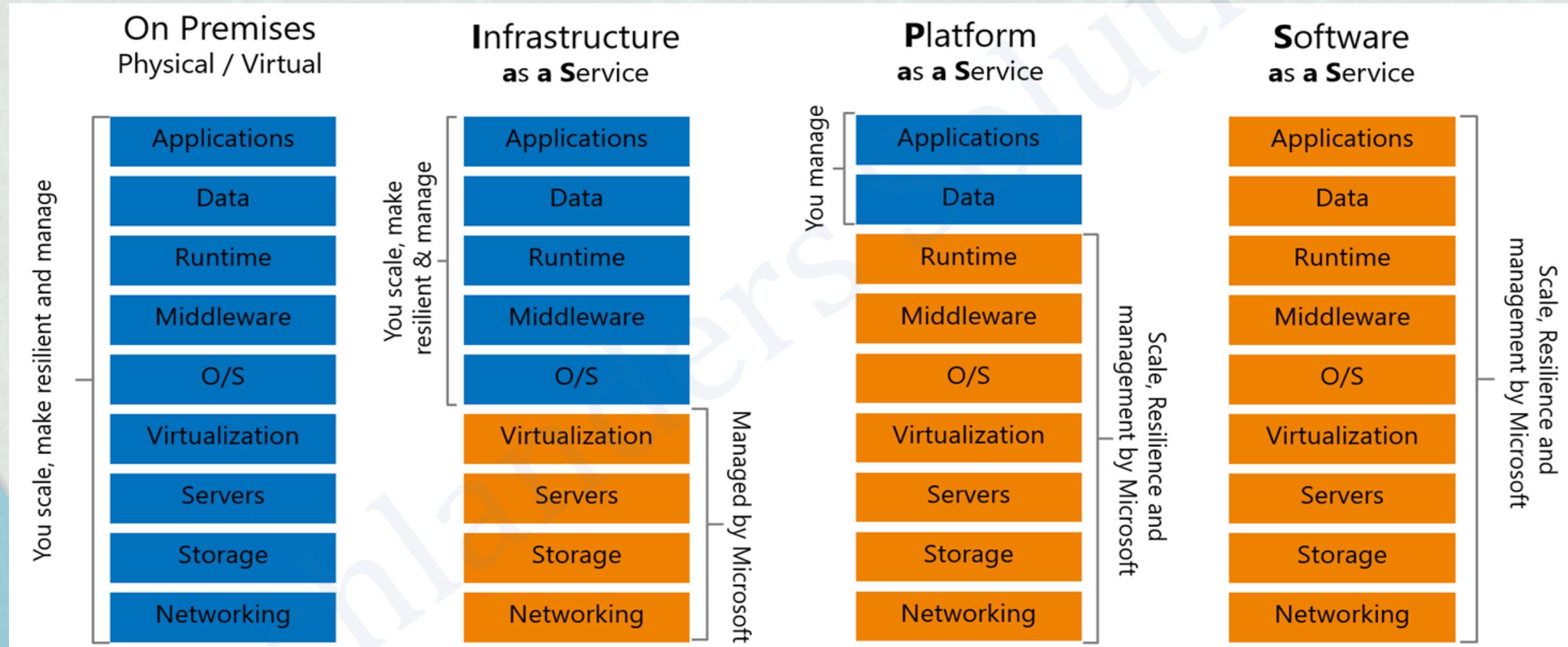
Cloud Advantages



Cloud Service Models

- There are three Cloud Computing Service Models:
 - Infrastructure as a Service (IaaS)
 - Platform as a Service (PaaS)
 - Software as a Service (SaaS)

Responsibility- Who owns What?



Responsibility- Who owns What?



Cloud Service Models - IaaS

IaaS is the most basic Cloud Service Model

It offers Underline Infrastructure for Compute, Storage and Networking

Infrastructure can be selected by customers as per their choice and Pay-per-use model.

- Examples: Bare metal servers, virtual Instances, Load balancers

IaaS - Benefits

- Drastic reduction in capital investment
- Easily Scalable
- Pay only for the used resources
- High Flexibility
- Reduced infrastructure support teams

Cloud Service Models - PaaS

Another service model, where cloud provider manages the OS & middleware part, along with IaaS

Provide capability to deploy applications on cloud infrastructure without managing underline Infra

Consumers are responsible for managing deployed applications and their environment specific configurations

- Examples: webservers and databases

PaaS - Benefits

- Includes all IaaS benefits
- No upfront licensing cost
- More reduction in Infrastructure support team
- Rapid time to market

Cloud Service Models - SaaS

SaaS deliver complete application to the consumers over the internet.

Consumers are not responsible for managing any application or underlying infrastructure.

SaaS application are delivered as “one-to-many” model.

- Examples: office365, Gmail, WhatsApp, JIRA, GIT, Service Now

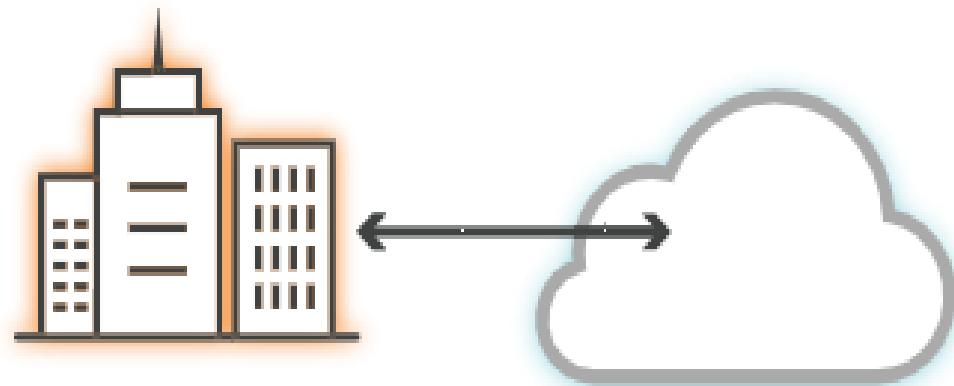
SaaS - Benefits

- Includes all discussed benefits which we get in PaaS
- Ability to access from anywhere
- Ability to access from multiple devices
- No installations and maintenance requirements
- No Application management/Licensing Required

Cloud Essentials Characteristics



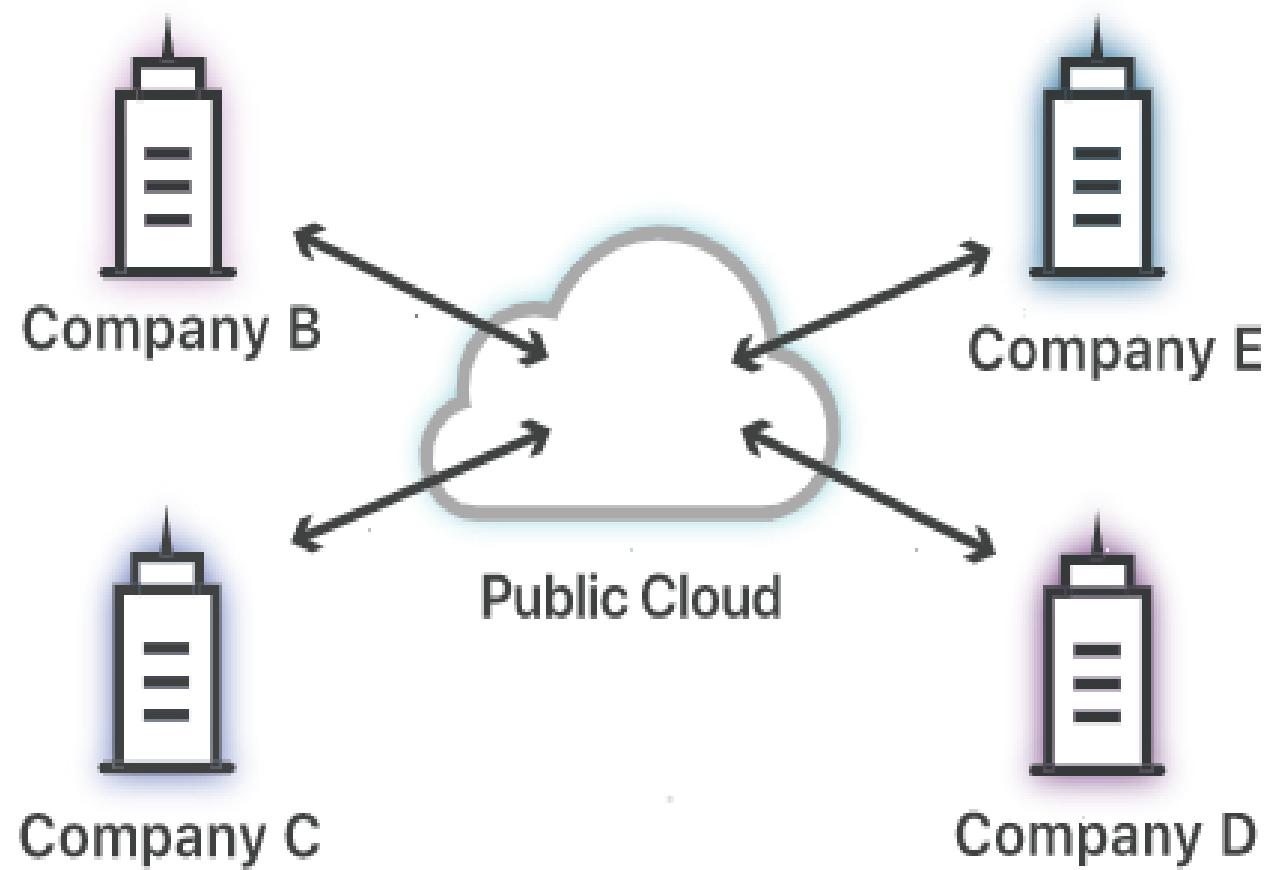
Private cloud



Company A

Company A's
private cloud

Public cloud shared
by multiple companies



Company C

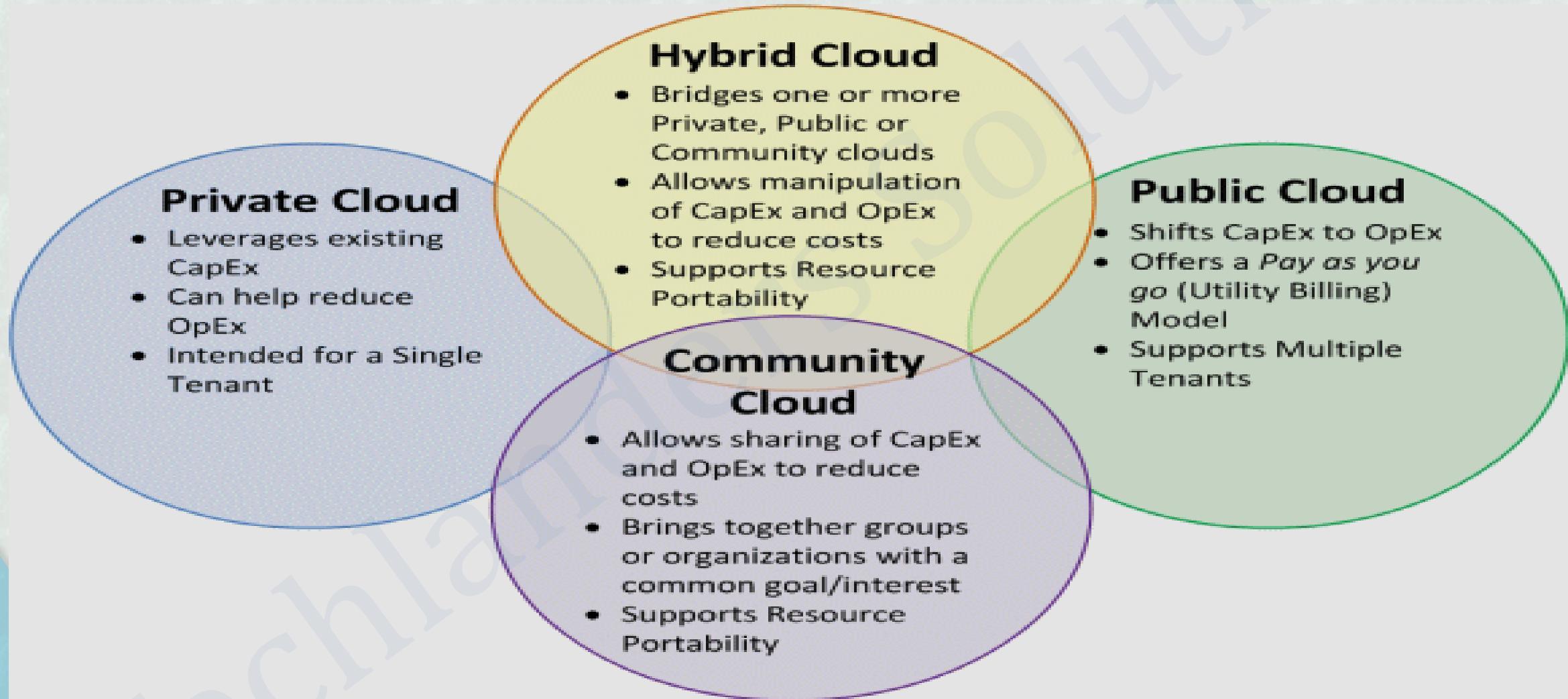
Company D

Company B

Company E

Public Cloud

Cloud Deployments Types



Cloud's 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

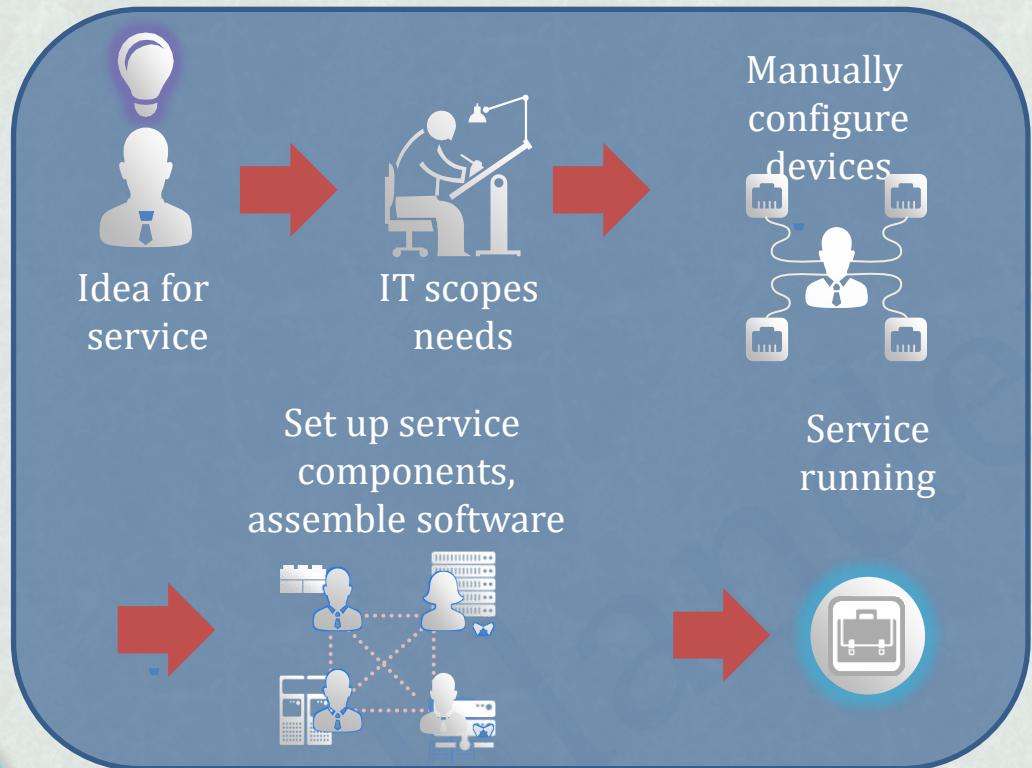


Predictable Bursting

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

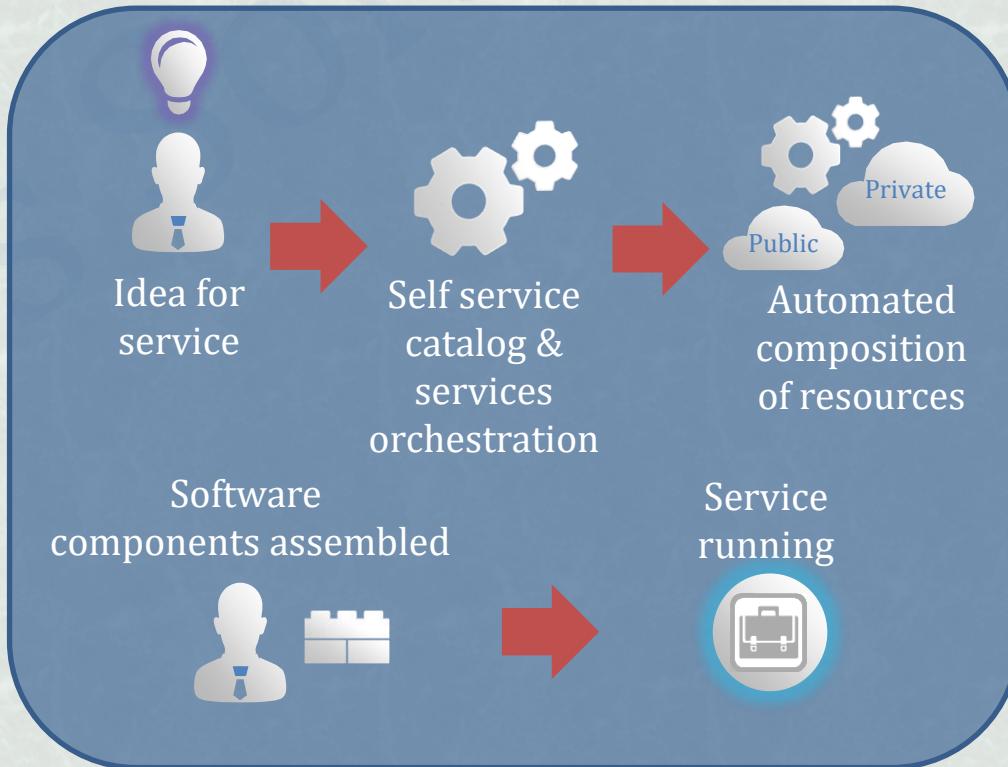
Business Impact of Cloud

Traditional Datacenter



Time to Provision New Service: Months

Cloud Infrastructure



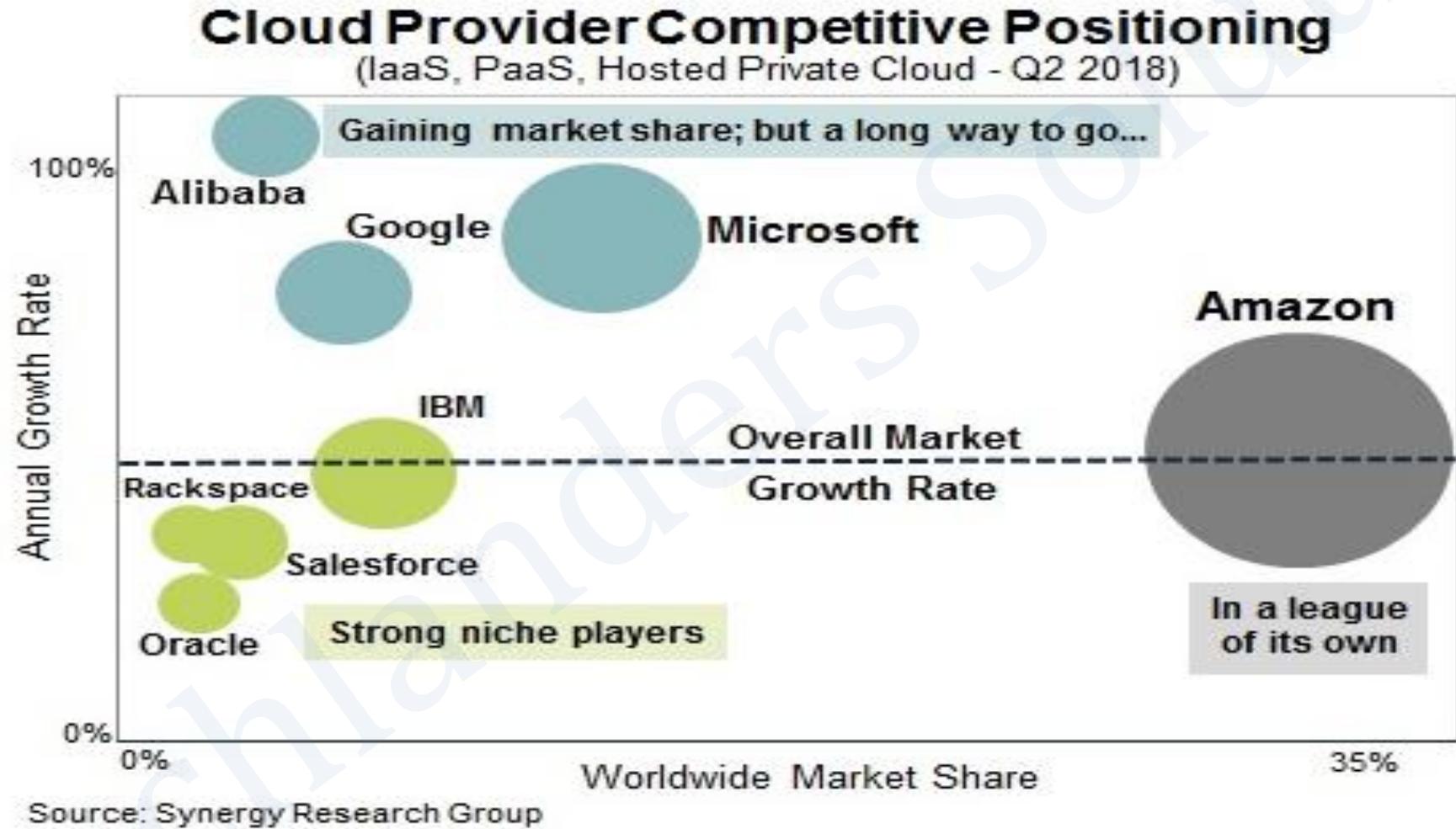
Time to Provision New Service: Minutes

Major Cloud Vendors

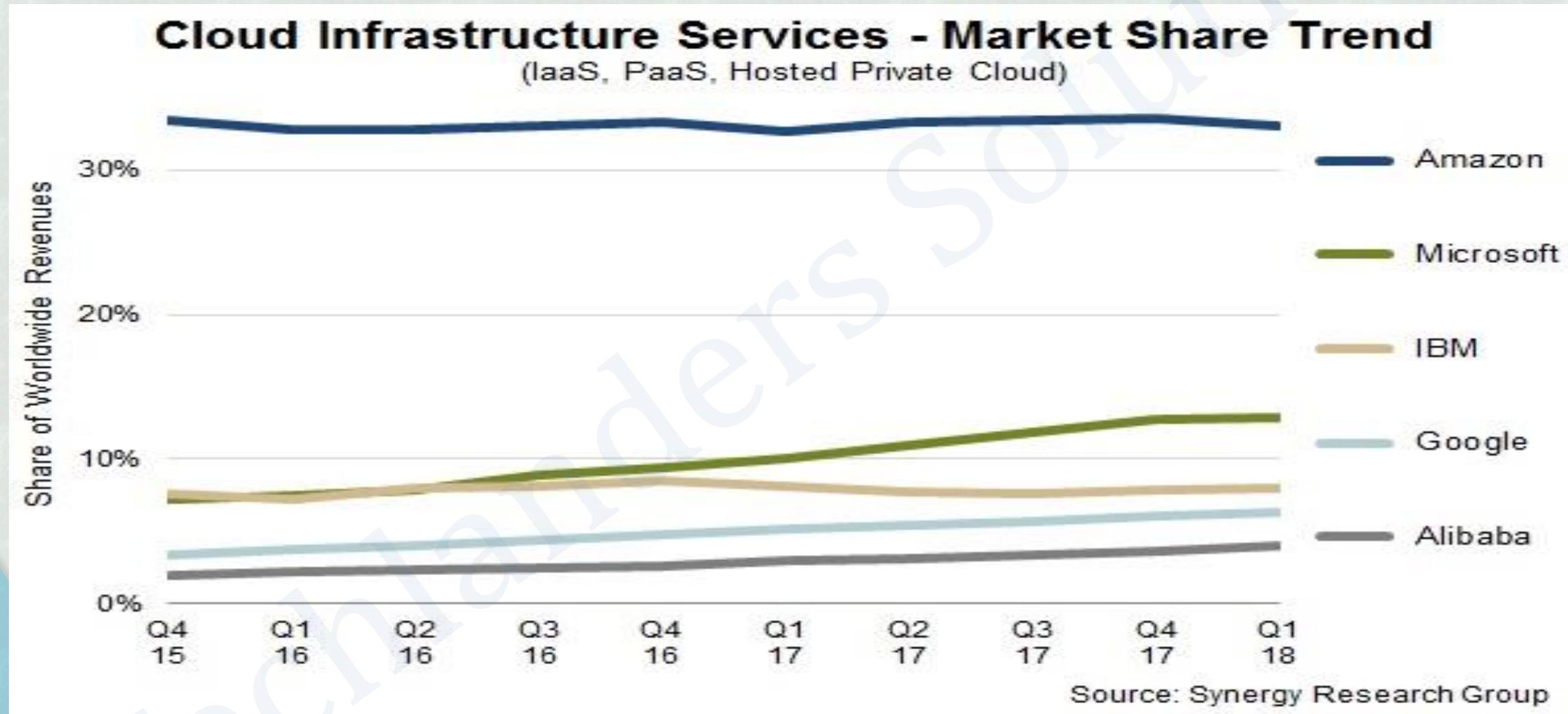
Top Cloud Computing Providers



Who stands where?



Who stands where?



Knowledge Checks

- Which Service Level (IaaS, PaaS, SaaS) provides you most control?
- What is Hybrid Cloud?
- Can two public clouds be connected?
- Connecting two public clouds, will be known as public cloud or Hybrid?
- Cloud provided Database, is a PaaS or SaaS?

AWS **(Amazon Cloud)**

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.

AWS Global Infrastructure

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.

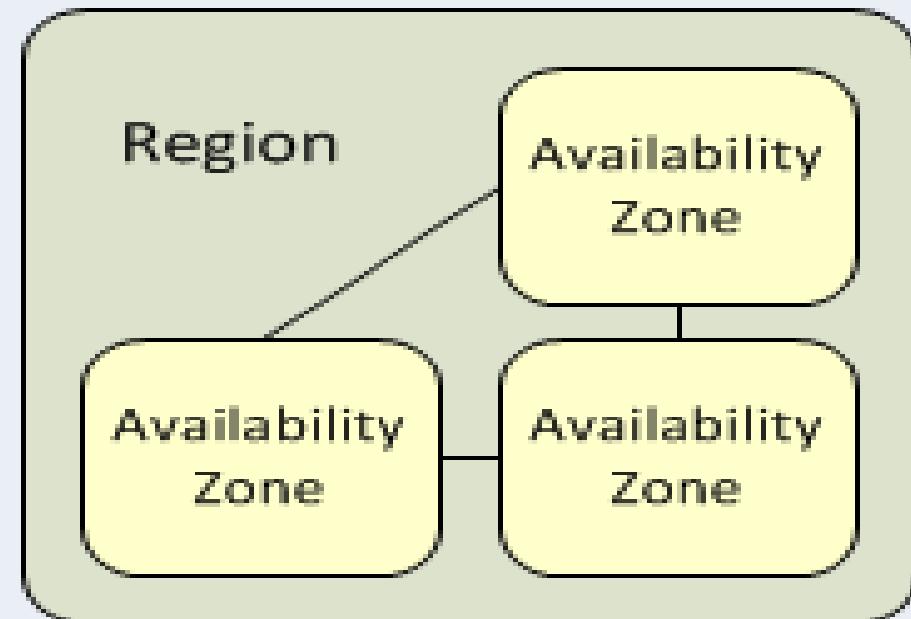
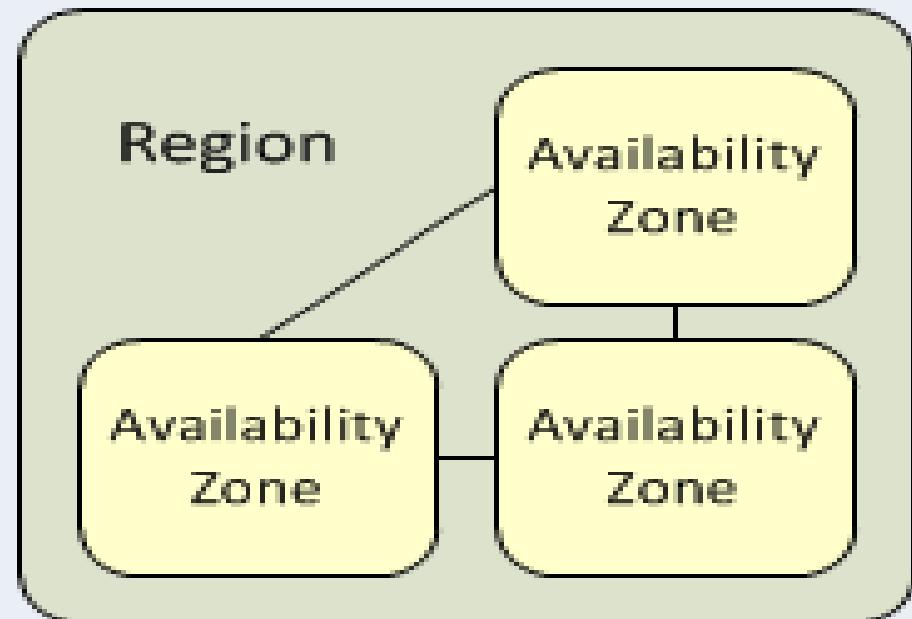
AWS Availability Zones

- AZ is a distinct location within a region
- Each zone is insulated (with low-latency links) from other to support single point of failures
- Each Region has minimum two AZ's
- Most of the services/resources are replicated across AZs for HA/DR purpose.

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

AWS Global Infrastructure

Amazon Web Services



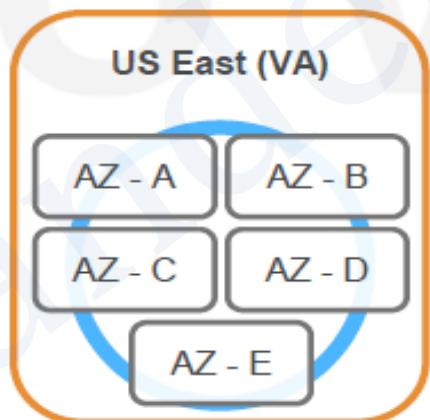
AWS Global Infrastructure

At least 2 AZs per region.

Examples:

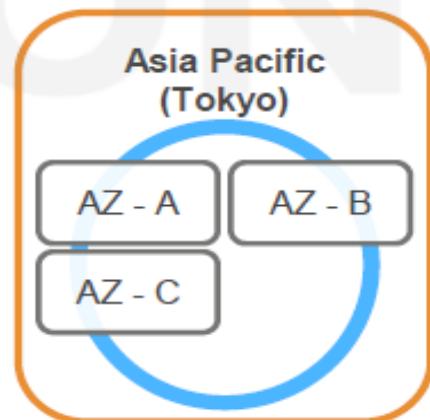
➤ US East (N. Virginia)

- us-east-1a
- us-east-1b
- us-east-1c
- us-east-1d
- us-east-1e



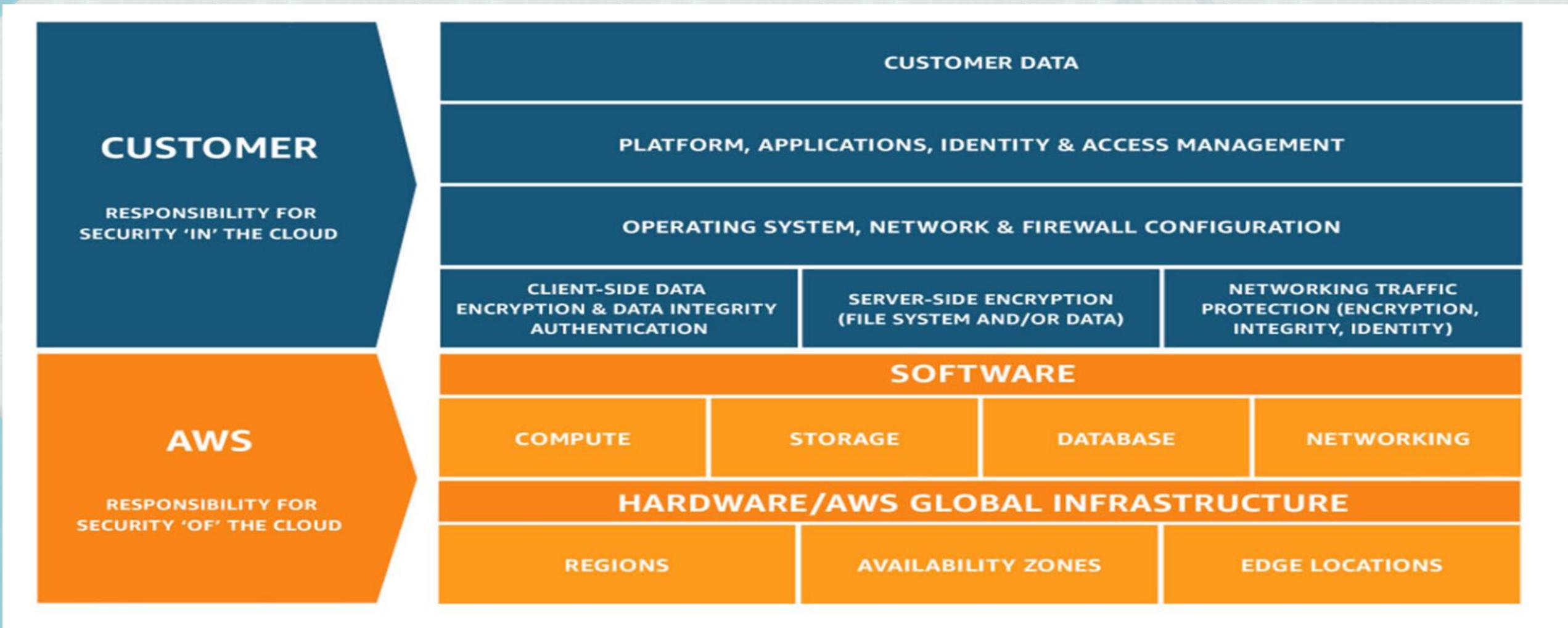
➤ Asia Pacific (Tokyo)

- ap-northeast-1a
- ap-northeast-1b
- ap-northeast-1c



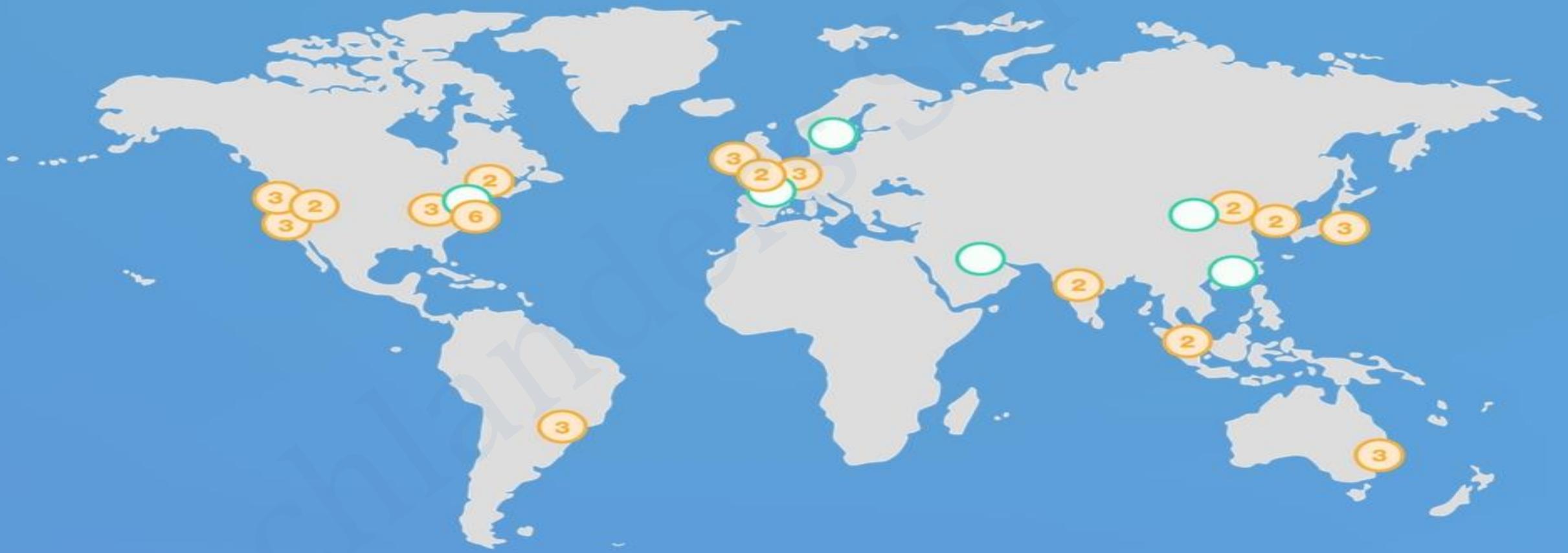
Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

Shared Responsibility



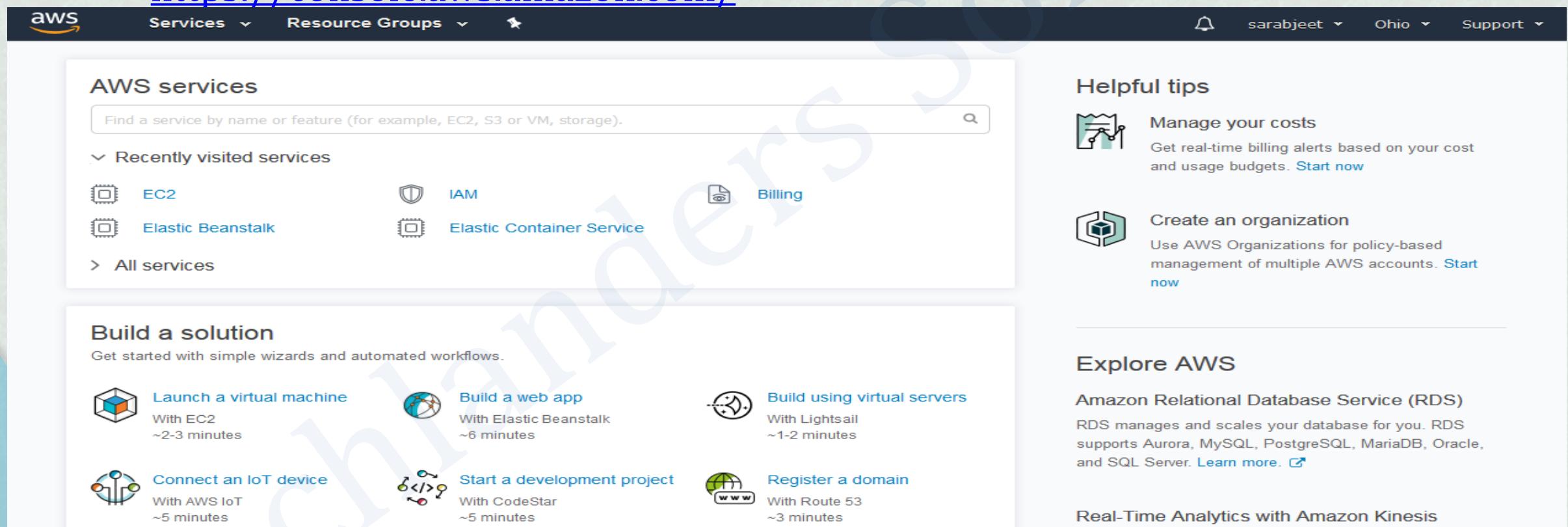
AWS Global Infrastructure

The AWS Cloud operates 44 Availability Zones within 16 geographic Regions around the world, with announced plans for 17 more Availability Zones and six more Regions in Bahrain, China, France, Hong Kong, Sweden, and a second AWS GovCloud Region in the US.



AWS Management Console

- Simple and intuitive web-based user interface.
 - <https://console.aws.amazon.com/>



The screenshot shows the AWS Management Console homepage. At the top, there's a navigation bar with the AWS logo, a search bar, and links for Services, Resource Groups, Notifications, User Profile (sarabjeet), Region (Ohio), and Support.

AWS services: A sidebar with a search bar and a list of recently visited services: EC2, IAM, Billing, Elastic Beanstalk, and Elastic Container Service. There's also a link to All services.

Build a solution: A section with six quick-start options:

- Launch a virtual machine (With EC2, ~2-3 minutes)
- Build a web app (With Elastic Beanstalk, ~6 minutes)
- Build using virtual servers (With Lightsail, ~1-2 minutes)
- Connect an IoT device (With AWS IoT, ~5 minutes)
- Start a development project (With CodeStar, ~5 minutes)
- Register a domain (With Route 53, ~3 minutes)

Helpful tips: A section with two items:

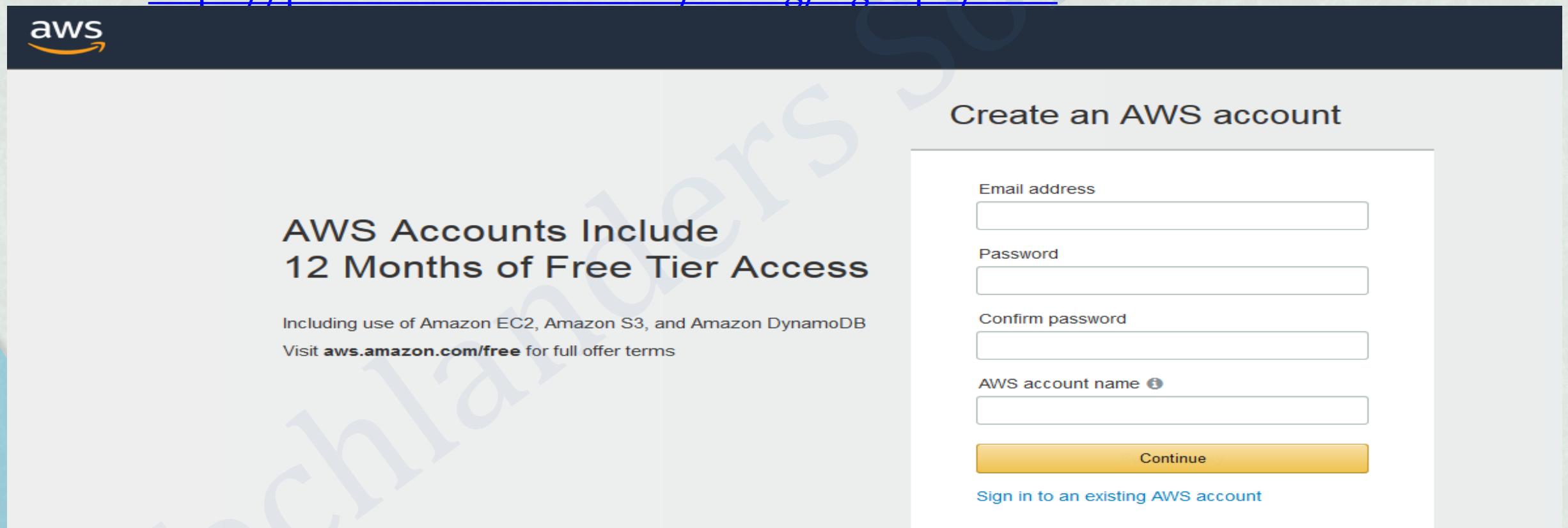
- Manage your costs**: Get real-time billing alerts based on your cost and usage budgets. [Start now](#)
- Create an organization**: Use AWS Organizations for policy-based management of multiple AWS accounts. [Start now](#)

Explore AWS: A section with two items:

- Amazon Relational Database Service (RDS)**: RDS manages and scales your database for you. RDS supports Aurora, MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. [Learn more](#).
- Real-Time Analytics with Amazon Kinesis**

LAB 1 : AWS Signup

- Create a new account at
 - <https://portal.aws.amazon.com/billing/signup#/start>



The screenshot shows the AWS Signup process. On the left, there's a promotional message about free tier access, and on the right is the account creation form.

Create an AWS account

Email address

Password

Confirm password

AWS account name

Continue

[Sign in to an existing AWS account](#)

AWS Accounts Include 12 Months of Free Tier Access

Including use of Amazon EC2, Amazon S3, and Amazon DynamoDB

Visit aws.amazon.com/free for full offer terms

AWS SIGNUP

creating the account

dhi Nagar

suite, unit, building, floor, etc

Province or region

e

net Services Pvt. Ltd. Customer

an India contact address are now required to
amazon Internet Service Private Ltd. (AISPL).
cal seller for AWS infrastructure services in

k here to indicate that you have read
gree to the terms of the AISPL
omer Agreement

Create Account and Continue 

Payment Information

We use your payment information to verify your identity and only for usage in excess of the [AWS Free Tier Limits](#). [We will not charge you for usage below the AWS Free Tier Limits](#). For more information, see the [frequently asked questions](#).



As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.

Credit/Debit card number

Expiration date

Cardholder's name

Select a Support Plan

AWS offers a selection of support plans to meet your needs. Choose the best aligns with your AWS usage. [Learn more](#)



Basic Plan

Free

- Included with all accounts
- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor
- 12-hour response time for nonproduction systems



Developer Plan

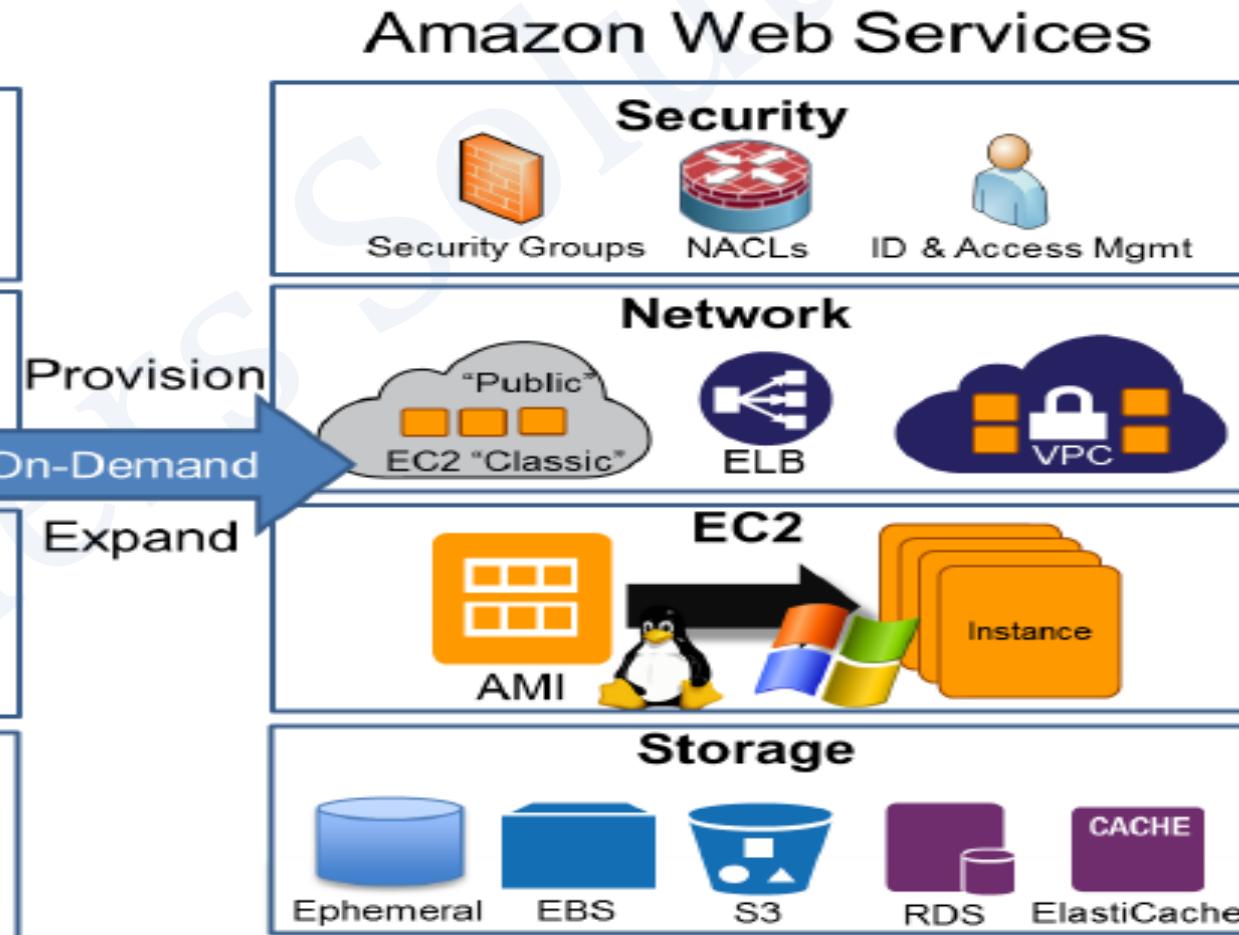
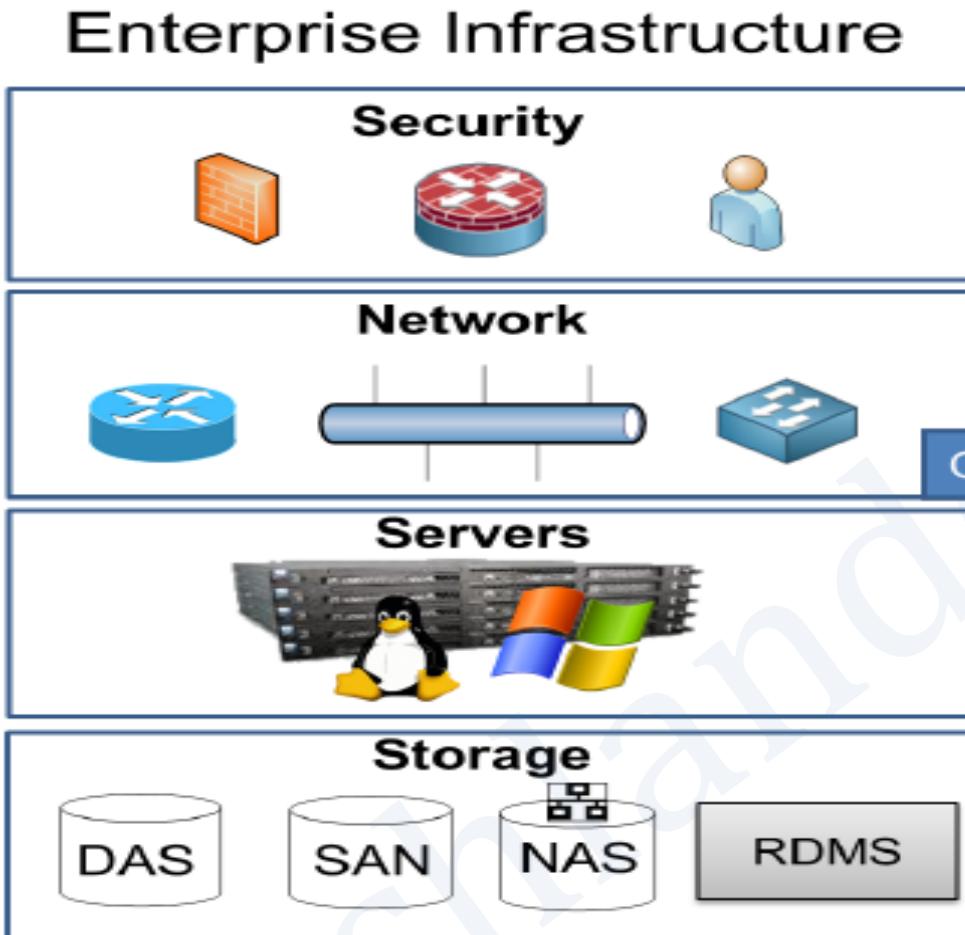
From \$29/month

- For early adoption, testing and development
- Email access to AWS Support during business hours
- 1 primary contact can open an unlimited number of support cases

Need Enterprise level support?

Contact your account manager for additional information on our enterprise support options.

AWS Core Infrastructure Services



AWS Security

- **Physical Security:**
- 24/7 trained security staff
- AWS data centers in nondescript and undisclosed facilities
- Two-factor authentication for authorized staff
- Authorization for data center access
- Multiple approval based change process

AWS Security

- **Hardware, Software, and Network :**
- Authentication and authorization in place
- RBAC based access control mechanism
- Firewall and other boundary devices
- Security at Server level, Application level and Network level
- AWS monitoring tools
- Services to log AWS resources access

AWS Security



Amazon Resource Names (ARNs)

Amazon Resource Names (ARNs) uniquely identify AWS resources.

We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, API calls etc.

ARN have a specific format:

arn:partition:service:region:account-id:resourcetype/resource

- IAM user name
 arn:aws:iam::123456789012:user/David
- IAM instance id:

arn:aws:ec2:region:account-id:dedicated-host/host_id

Eg. arn:aws:ec2:us-east-1:123456789012:dedicated-host/h-12345678

AWS Access Credentials

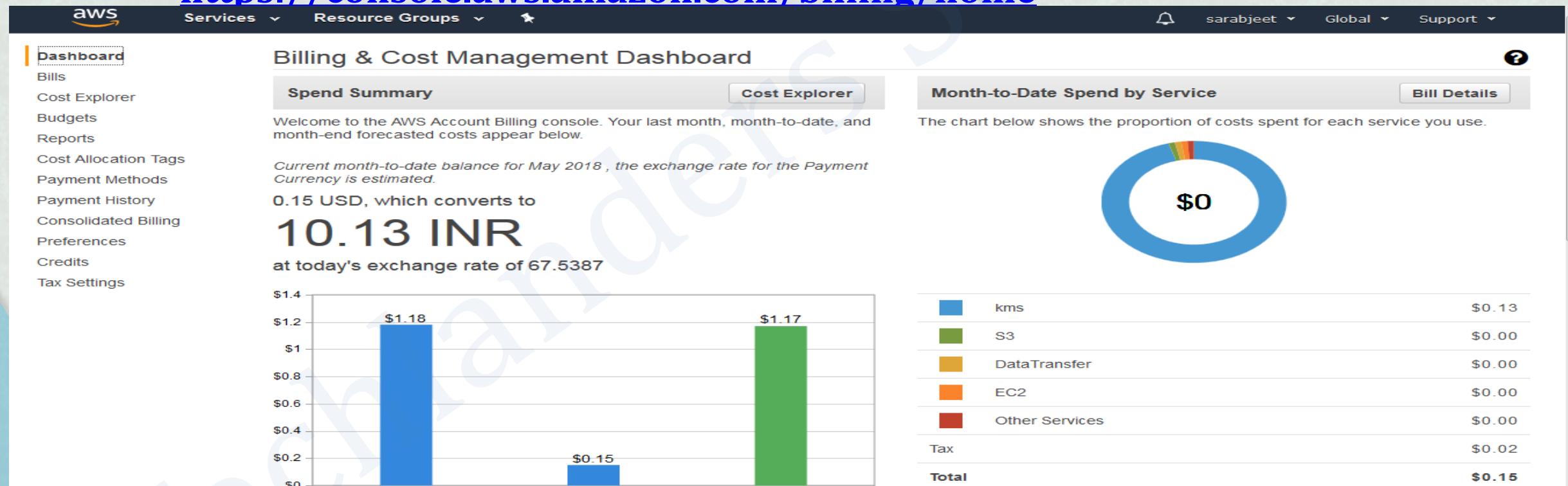
AWS resources can be access using several authentication methods:

- IAM User-id / Password
- Account ID/ AK (Access Key)/ SK (Security Key)
- Certificates
- Key pairs

AWS Billing

AWS Billings history, Previous payments, Current month cost, Budget fixing, Setting usage alarms etc, can be managed from AWS billing page :

<https://console.aws.amazon.com/billing/home>



AWS Pricing calculators

Simple Monthly Calculator:

You can Estimate your expected monthly bill using Simple Monthly Calculator.

<http://calculators.s3.amazonaws.com/index.html>

TCO Calculator:

You can Quickly compare the total cost of ownership (TCO) of your **on-premises infrastructure with a comparable AWS deployment** using TCO Calculator and estimate savings you can realize by moving to AWS. <https://awstcoccalculator.com/#>

Cost Explorer:

With Cost Explorer, you can track your actual account usage and bill, at any time using the billing portal. You can view data for up to the last 13 months, forecast how much you are likely to spend for the next three months, and get recommendations for what Reserved Instances to purchase.

<https://console.aws.amazon.com/billing/home#/costexplorer>

Resource Management Tools

- **AWS Management Console**
- AWS Console Mobile App (View resources)
- **AWS Command line interface**
- AWS Toolkit for PowerShell
- AWS-Shell

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

LAB 2 : Install AWS CLI

- Lets install an AWSCLI

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

- **aws --version**
- **aws help**
- **aws ec2 help / aws s3 help / aws <any-subcommand> 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:~#
```

LAB 2 : Install AWS CLI

- Watch Credentials and configurations in .aws directory under current directory being created/updated with aws configure:

```
C:\Users\gd>dir .aws
Volume in drive C is System
Volume Serial Number is 5205-3B4A
Directory of C:\Users\CQBJ1454\.aws
2018-02-11 19:13 <DIR> .
2018-02-11 19:13 <DIR> ..
2018-02-11 19:29       48 config
2018-05-05 09:03      119 credentials
                         2 File(s)    167 bytes
                         2 Dir(s)   452,272,128 bytes free
C:\Users\gd>
```

Knowledge Checks

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



AWS **IAM**

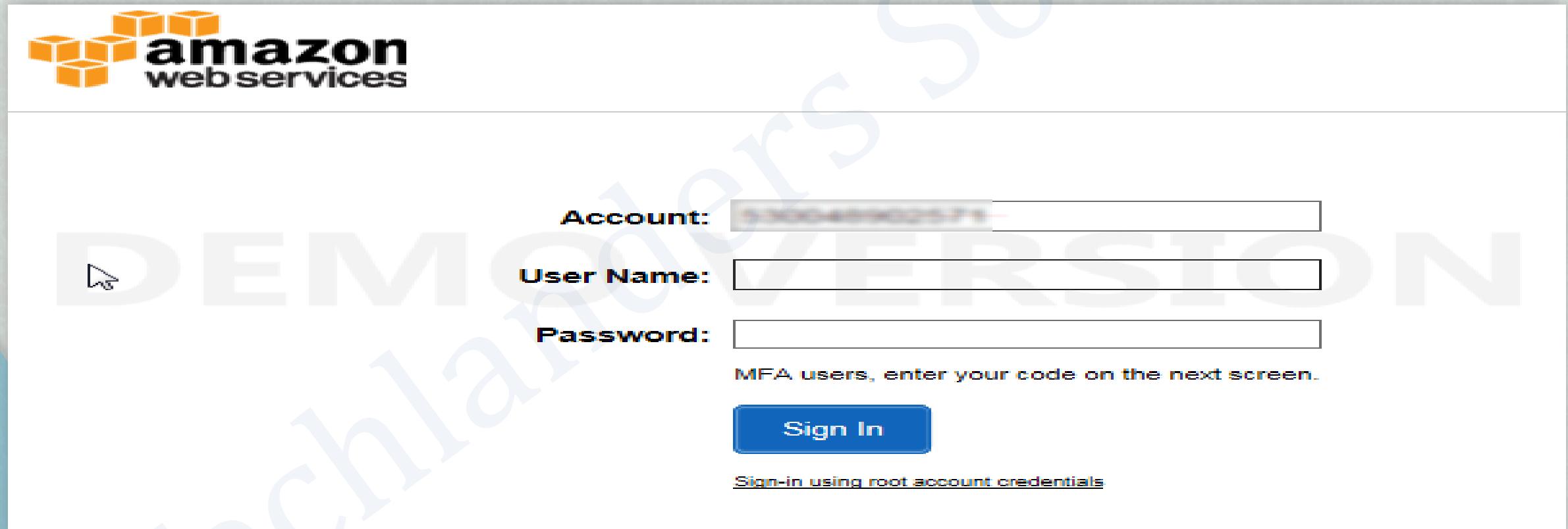
AWS Identity and Access Management

(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 - Users

- You can create IAM users to distribute your work among team/users.
- Different permissions can be given to IAM users



LAB 27 : IAM User

Create an IAM User

In this exercise, you will create an IAM user who can perform all administrative IAM functions. Then you will log in as that user so that you no longer need to use the root user login. Using the root user login only when explicitly required is a recommended security practice (along with adding MFA to your root user).

1. While logged in as the root user, create a new IAM user called user1.
2. Provide Access type -> Programmatic access and Management console access
3. Set a console password and click next
4. Logout from root and Use the customized sign-in link to sign in as Administrator
5. Try to login and check if you are able to access the s3 buckets in new user.
6. Were to able to see? What about other services?

IAM - Groups

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

▼ **Summary**

Group ARN:	arn:aws:iam::240950888888:group/read_group 
Users (in this group):	2
Path:	/
Creation Time:	2018-05-02 22:21 UTC+0530

Users Permissions Access Advisor

This view shows all users in this group: **2 Users**

User	Actions
 gagandeep	Remove User from Group
 Tejas	Remove User from Group

LAB 26 : IAM Group

Create an IAM Group

In this exercise, you will create a group for all IAM administrator users and assign the proper permissions to the new group. This will allow you to avoid assigning policies directly to a user later in these exercises.

1. Log in as the root user.
2. Go to **Security, Identity & Compliance → IAM → Groups**
3. Create an IAM group provide name Elevated_Users
4. Don't select any policy and click create.
5. Add user created to the group

IAM 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

LAB 28 : IAM Policies

Create an IAM Policy

In this exercise, you will create an IAM Policy, which will provide EC2 list and Read permissions to the policy holder

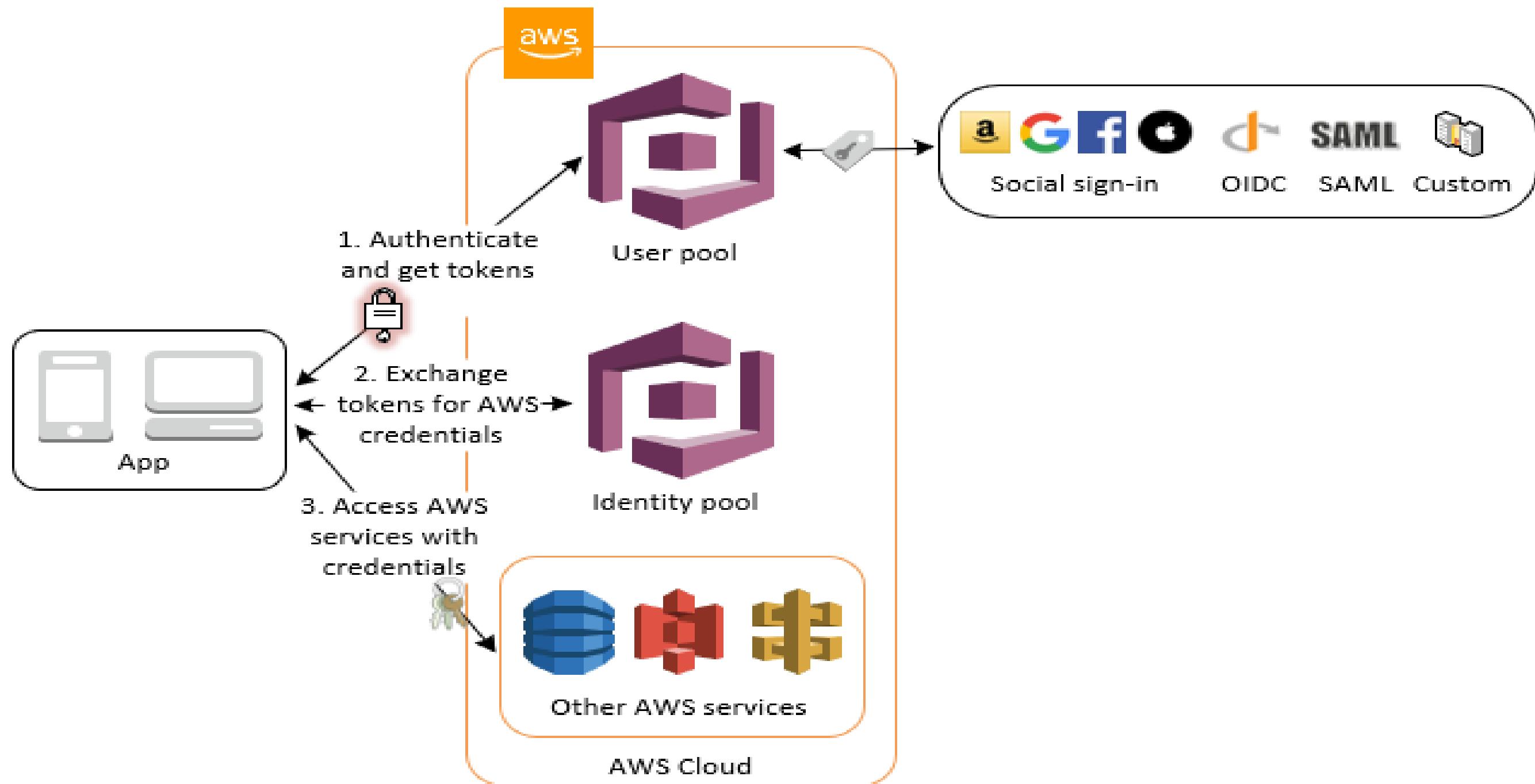
1. While logged in as the root user, Click on IAM → Policies.
2. Select S3Fullaccess policy or Click Create a policy
3. Select Service, Action, Resources.
4. Review Policy and click Create policy
5. View your policy in search bar.
6. Click on Action → Attach, and select the New user
7. Go to Users tab and see the new policy attached to user
8. Now try to view the s3 bucket features . Are you able to ?
9. Remove the policy .
10. Create a policy and then try to attach it to user.

AWS Identity and Access Management

(IAM)

- Delete AWS account (root) access keys.
- Create individual IAM users.
- Use groups to assign permissions to IAM users.
- Grant least privilege.
- Configure a strong password policy.
- Enable MFA for privileged users.

Cognito





AWS Compute Services

Virtual Machines

- What are the bare minimum infrastructure requirements for any application?
- Hardware (RAM, Processor, Processor type, HBA's, etc.)
- OS Disk & Data Disk
- OS Images
- Network
- Security (Firewall, Networking, Access Mechanism)
- Credentials

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.
- **Deploy across AWS Regions and Availability Zones for reliability/HA**
- Only X86 based OS supported. So platform specific OS are not supported.

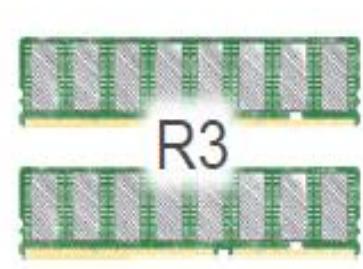
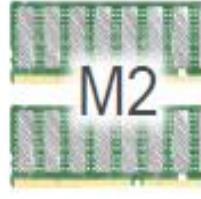
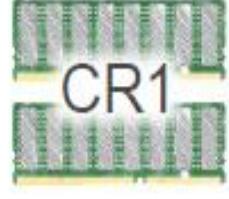
Instance Types (EC2)

- The instance type defines the different virtual hardware Models (sizes) supporting an Amazon EC2 instance.
- There are dozens of instance types available, varying in the following dimensions:
 - Virtual CPUs (vCPUs)
 - Memory
 - Storage (size and type)
 - Network performance
 - Graphical Processing
- Instance types are grouped into families based on the ratio of these values to each other.

Instance Types (EC2)

- EC2 instance types are optimized for different use cases and come in multiple sizes. This allows you to optimally scale resources to your workload requirements.
- AWS uses Intel® Xeon® processors for EC2 instances, providing customers with high performance and value.
- Consider the following when choosing your instances: Core count, memory size, storage size and type, network performance, and CPU technologies.
- Hurry Up and Go Idle - A larger compute instance can save you time and money, therefore paying more per hour for a shorter amount of time can be less expensive.

Instance Types with AWS

General purpose	Compute optimized	Storage and IO optimized	GPU enabled	Memory optimized
		 I2 HS1		
	 			 

EC2 Pricing Models

On-Demand Instances

Pay by the hour.

Reserved Instances

Purchase at significant discount.
Instances are always available.

1-year to 3-year terms.

Scheduled Instances

Purchase a 1-year RI for a recurring period of time.

Spot Instances

Highest bidder uses instance at a significant discount.
Spot blocks supported.

Dedicated Hosts

Physical host is fully dedicated to run your instances. Bring your per-socket, per-core, or per-VM software licenses to reduce cost.

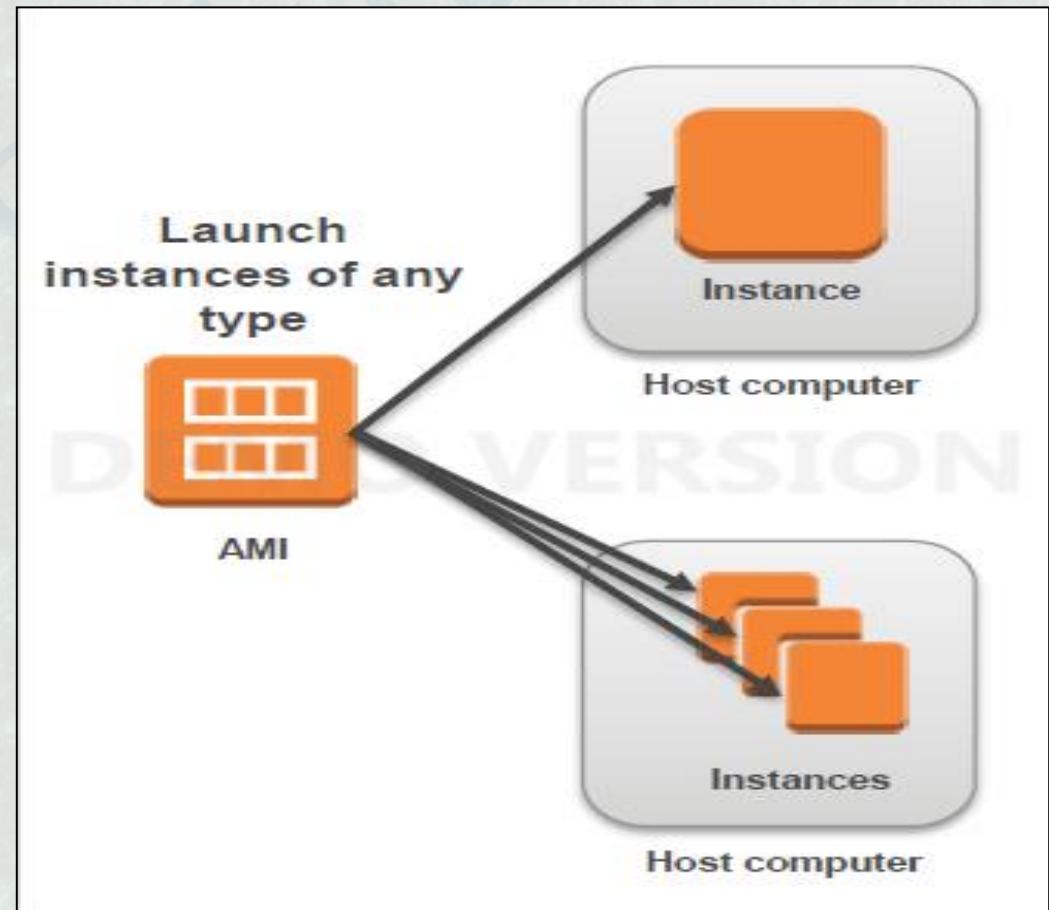
AMI's

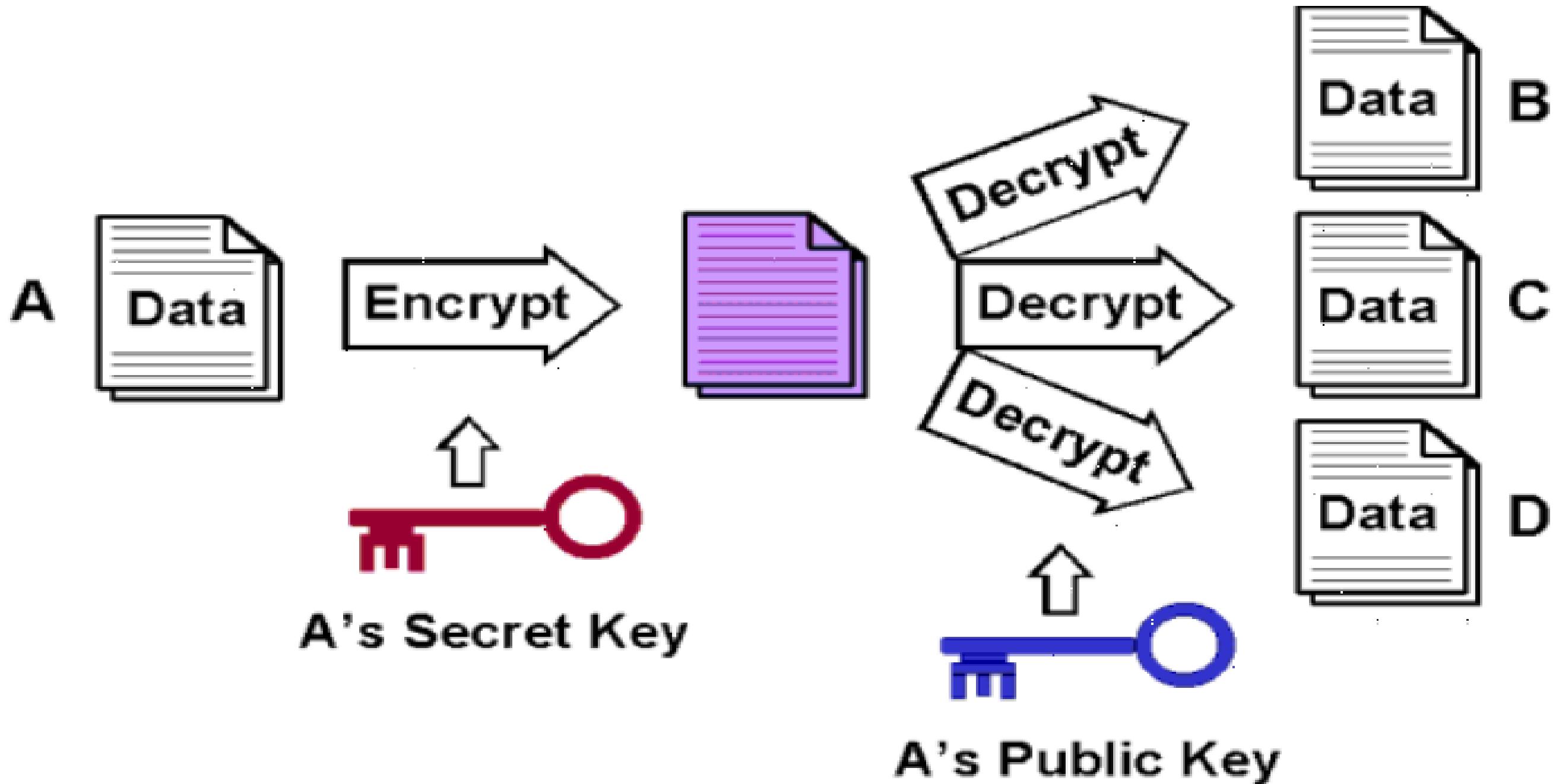
- Stands for **Amazon Machine Image**
- **AMI is a template for the root volume** for the instance (example: an OS image, a webserver, an application server etc.)
- It also includes the launch permissions that control which AWS accounts can use the AMI to launch instances.
- All AMIs are based on x86 OSs, either Linux or Windows.

Instances and AMI's

Select an AMI based on:

- Region
- Operating system
- Architecture (32-bit or 64-bit)
- Launch permissions
- Storage for the root device
- Virtualization Type





Credentials

- Used to access an Instance.
- Amazon EC2 uses public-key cryptography to encrypt and decrypt login information.
- Public-key cryptography uses a public key to encrypt a piece of data and an associated private key to decrypt the data.
- These two keys together are called a *key pair*.
- AWS stores the public key, and the private key is kept by the customer. The private key is essential to acquiring secure access to an instance for the first time.
- When launching a Windows instance, Amazon EC2 generates a random password for the local administrator account and encrypts the password using the public key.

Launching EC2 Instance

- Procedure to launch EC2 instance in minutes:
- Determine the AWS Region in which you want to launch the Amazon EC2 instance.
- Launch an Amazon EC2 instance from a pre-configured Amazon Machine Image (AMI).
- Choose an instance type based on CPU, memory, storage, and network requirements.
- Configure network, security groups, storage volume, tags, and key pair; or directly launch instance post selecting AMI and Instance type.

LAB 3 : Create an EC2 Linux Instance

Console Access: <https://console.aws.amazon.com/>

- Observe different available AMIs and Instance types
- Exercise:

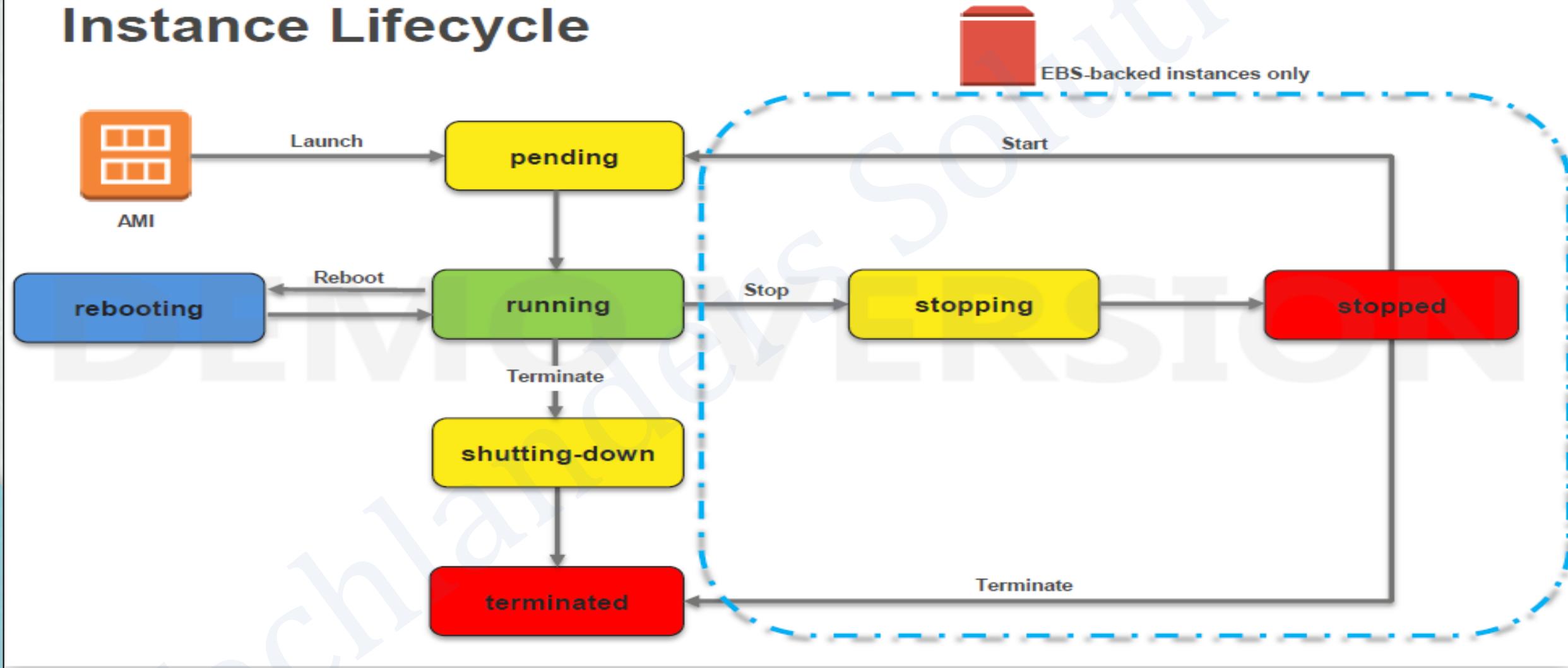
Create a Linux Operating System using mandatory configurations:

- Instance Type : **T2.Micro**
 - AMI : **Amazon Linux 2 AMI**
 - Credentials: **Create/download your own key pair and use same.**
-
- Observe the instance details and navigate through the diff tabs
 - Reboot, Stop, Start your server
 - Access your instance using above downloaded key pair using different platforms i.e , instance connect , using putty , cli.

Note: (For the first instance take the default parameters wherever inputs required)

Instance Lifecycle

Instance Lifecycle



Network part

Private IP: Alike traditional VMs/hosts, each EC2 instance must at least belong to one network (VPC in AWS) and must have at least one Private IP from that network/subnet.

Public IP: A launched instance may also have a public IP address assigned. This IP address is assigned from the addresses reserved by AWS and cannot be specified. This IP address is unique on the Internet, persists only while the instance is running, and cannot be transferred to another instance.

Private/Public Domain Name System (DNS): When you launch an instance, AWS creates one private and one Public DNS name that can be used to access the instance. This DNS name is generated automatically and cannot be specified by the customer. This DNS name persists only while the instance is running and cannot be transferred to another instance.

Elastic IP (EIP): An elastic IP address is an address unique on the Internet that you reserve independently and associate with an Amazon EC2 instance.

EC2 Security – Security Group

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.

Amazon Server Storage

Amazon EBS (Elastic Block Store)

- Data stored on an Amazon EBS volume can persist independently of the life of the instance.
- Persistent **block-level storage volumes** for use with Amazon EC2 instances.
- 99% used volume type in AWS.

Amazon EC2 Instance Store

- Like Swap volumes attached to your server for faster processing.
- Data stored on a local instance store persists only as long as the instance is alive.
- Storage is ephemeral.
- Older volume type, used exceptionally in some instances i.e DB,High processing systems.

AWS Tags

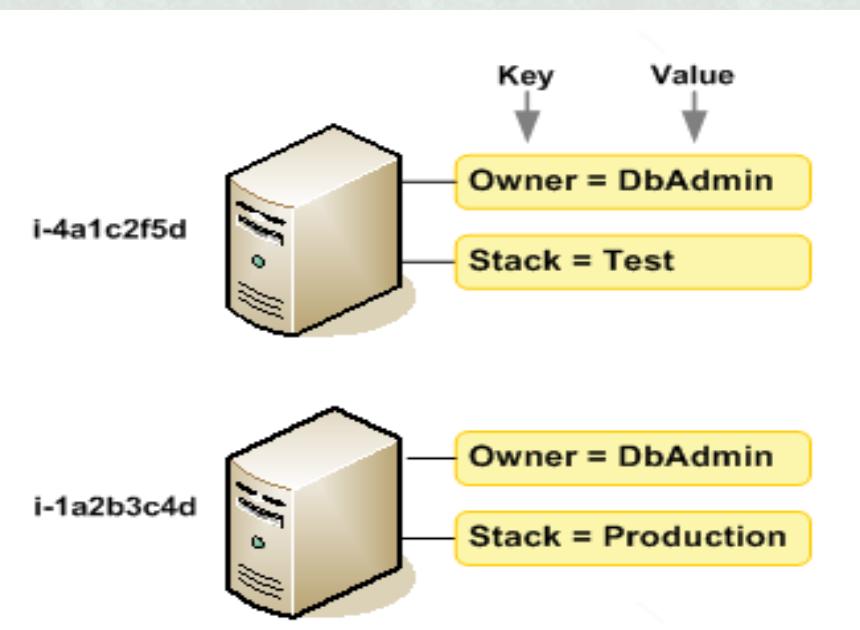
Tags helps you manage your instances, images, and other Amazon EC2 resources.

Its your own metadata to each resource in the form of *tags*.

Each tag consists of a *key* and an optional *value*, both of which you define.

It enable you to categorize your AWS resources in different ways, for example, by purpose, owner, or environment.

You can organize your billing information based on resources that have the same tag key values.



AWS Tags

- Maximum number of tags per resource—50
- Tag keys and values are case-sensitive.
- You can't terminate, stop, or delete a resource based solely on its tags; you must specify the resource identifier : ARN's
- Not all resources supports tags. Pls check the supported list from AWS website.

EC2 Metadata

AWS Instance Metadata –

Instance metadata is data/information about your instance.

An HTTP call to **http://169.254.169.254/latest/meta-data/** will return the top node of the instance metadata tree and will return the information about your instance e.g. Public IP, DNS name, OS type etc..

For Linux systems:

```
curl http://169.254.169.254/latest/meta-data/
```

Dynamic Instance Identity metadata:

```
http://169.254.169.254/latest/dynamic/instance-identity/
```

LAB 4 : Create Windows Instance

Exercise:

- Create a Windows Operating System using custom configurations:
 - Instance Type : **T2.Micro**
 - AMI : **Microsoft Windows Server 2019 Base**
 - Credentials: **Use existing key from Lab 3 or if not create a new one.**
 - Assign tags, **Key = Name, Value= Server-Name**
 - Assign tags, **Key = Owner, Value= Your name**
 - Add inbound port of **RDP**
- Access your instance using Administrator password, which will be retrieved using your downloaded key pair using mstsc or dns

LAB 5 : Work with IP/DNS

Exercise: Elastic ip may charge you if not associated with your instance.

- Use server created in previous exercise.
 - Case-1
 - Reboot and observe the Public IP (there will be no change)
 - Case-2
 - Stop the Instance and Start it back, **observe the Public IP**
- Que is how to make Public IP's Persistent ?
- Exercise: Create an EIP, Allocate the EIP to an Instance and perform both cases again.
- **NOTE : MAKE SURE THE ELASTIC IP IS CHARGEABLE IN FREE TIER WHEN :**

***ELASTIC IP IS NOT ASSOCIATED WITH ANY INSTANCE**

***INSTANCE ATTACHED WITH ELASTIC IP IS STOPPED .**

LAB 6 : Creating server using CLI

Check the details for all running instances using CLI

- aws ec2 describe-instances | grep -i instanceid

Creation of an AWS Instance using CLI:

- aws ec2 run-instances --image-id ami-033b95fb8079dc481 --instance-type t2.micro --key-name virginia-key
- aws ec2 describe-instances | grep -i instanceid
- aws ec2 stop-instances --instance-ids i-052af6a166198df10
- aws ec2 terminate-instances --instance-ids i-052af6a166198df10

LAB 8 : Working with User data

- User data is a feature of EC2, to provide build time commands to EC2 instance.
- Commands provided will run during instance startup.

Create one EC2 Linux instance and provide EC2 user data to run below scripts:

```
#!/bin/bash
sudo yum update -y;
sudo yum install -y httpd;
sudo systemctl start httpd;
sudo systemctl enable httpd;
sudo echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html
```

- If user-data is changed after the creation of server after stopping it then script will not run after modifying and restart as well.

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

LAB 29 : IAM Roles

In this exercise, you will create an IAM Role, which will allow EC2 instance to run codes/access other service on your behalf.

1. While logged in as the root user, Click on IAM → Roles.
2. Click Create Role
3. Select a service that will use this role to get access (select EC2 here).
4. Click “Next – Permissions”.
5. Select “AdministratorAccess”
6. Click on review & provide a Role name
7. Click on Create Role.
8. Connect to instance and using aws commands , you will be able to access s3 , ec2 and other resources .. You will see the resources :
9. aws ec2 describe-instances --region us-east-1 | grep -i instanceid
10. Try to access resources from cmd as well as ec2 instance connect , both will now not ask for access aws configure due to the role attached.



AWS

Storage Services



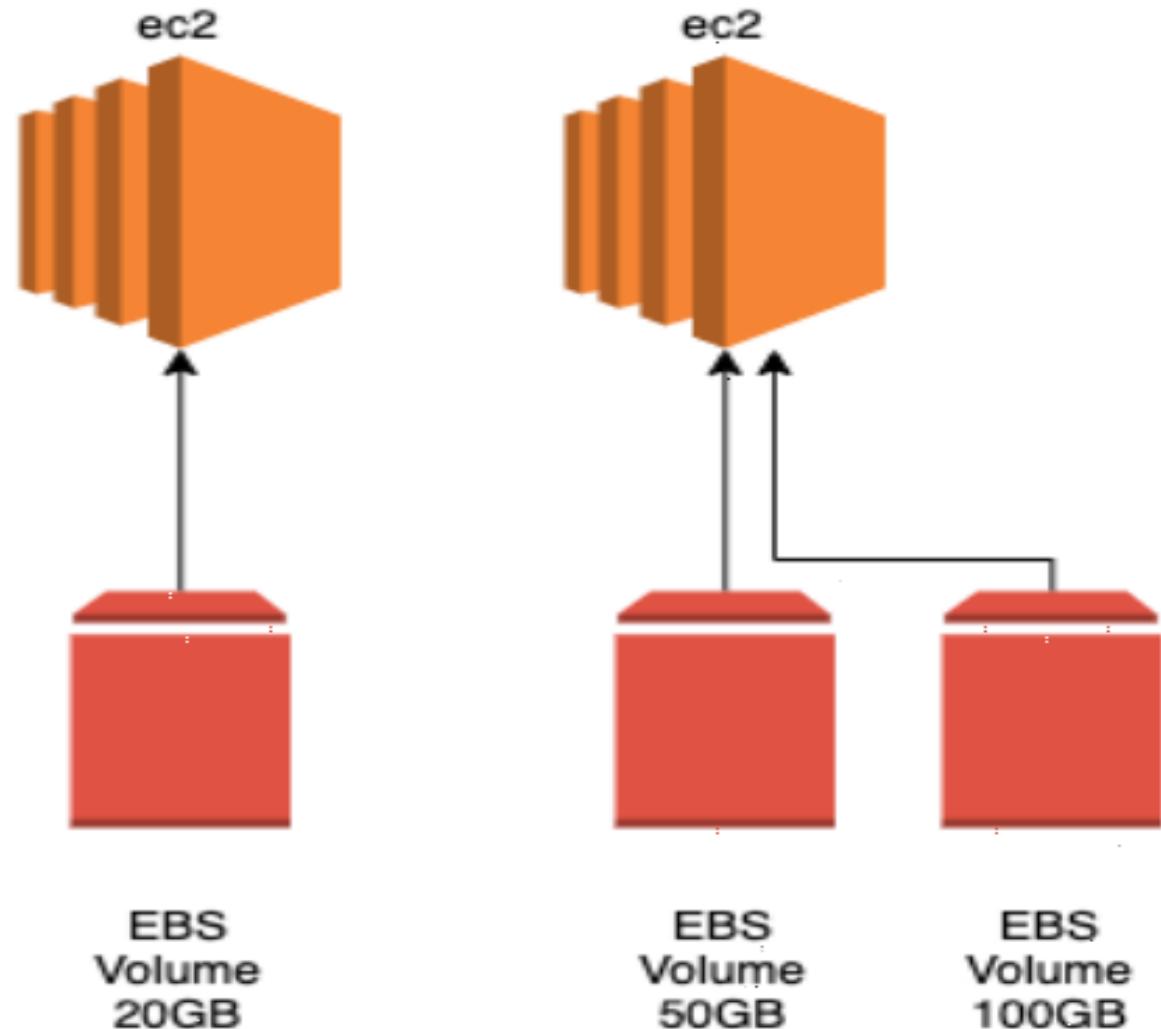
AWS

EBS Service

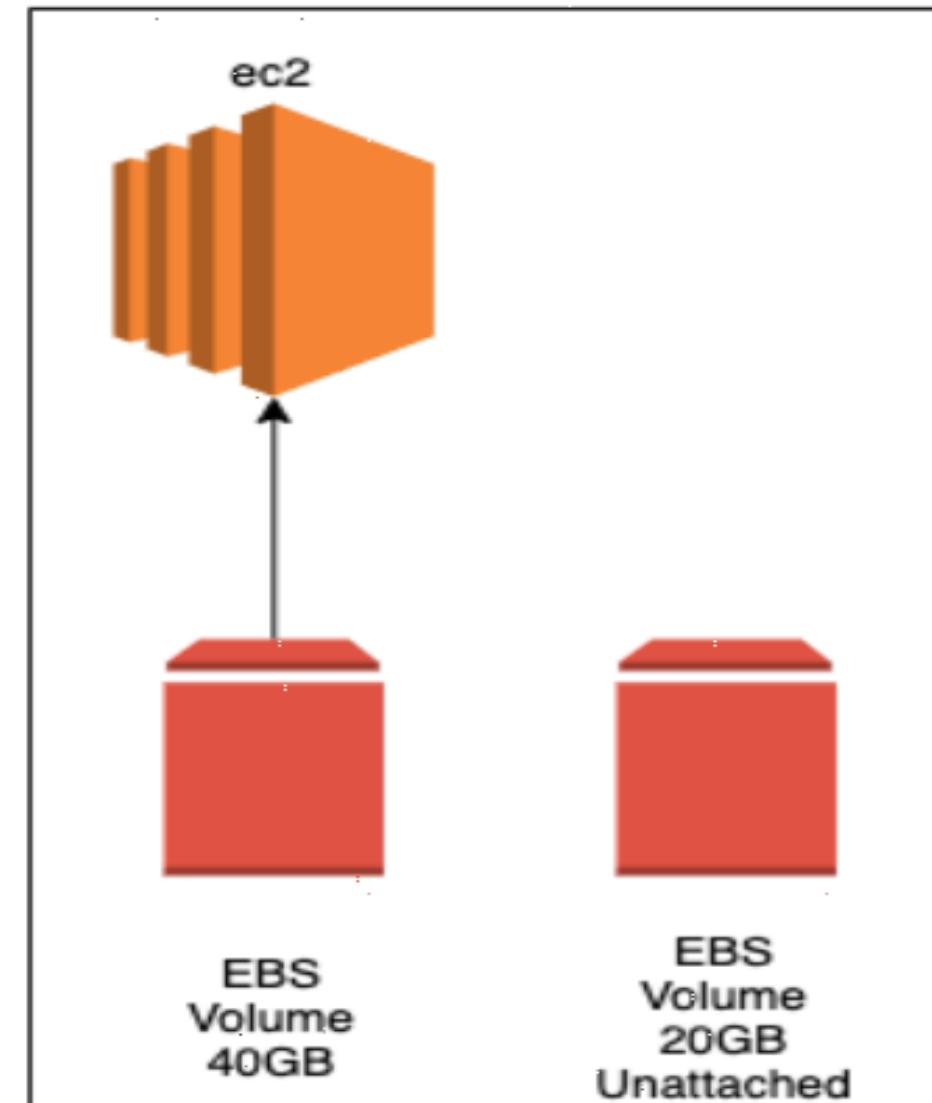
Storage Disk Addition

- Storage Addition, Filesystem Creation are very common tasks in day to day activity for any application.
- In traditional DC we have multiple dependencies on Hardware , Storage and OS teams. Hence a time consuming process.
- With AWS Cloud its no more a pain, we can add, delete EBS volume on the fly.
- Once EBS volume is attached, you can reclaim the storage in few clicks and save the cost of storage by deleting them when not needed.

us-west-1a



us-west-1b



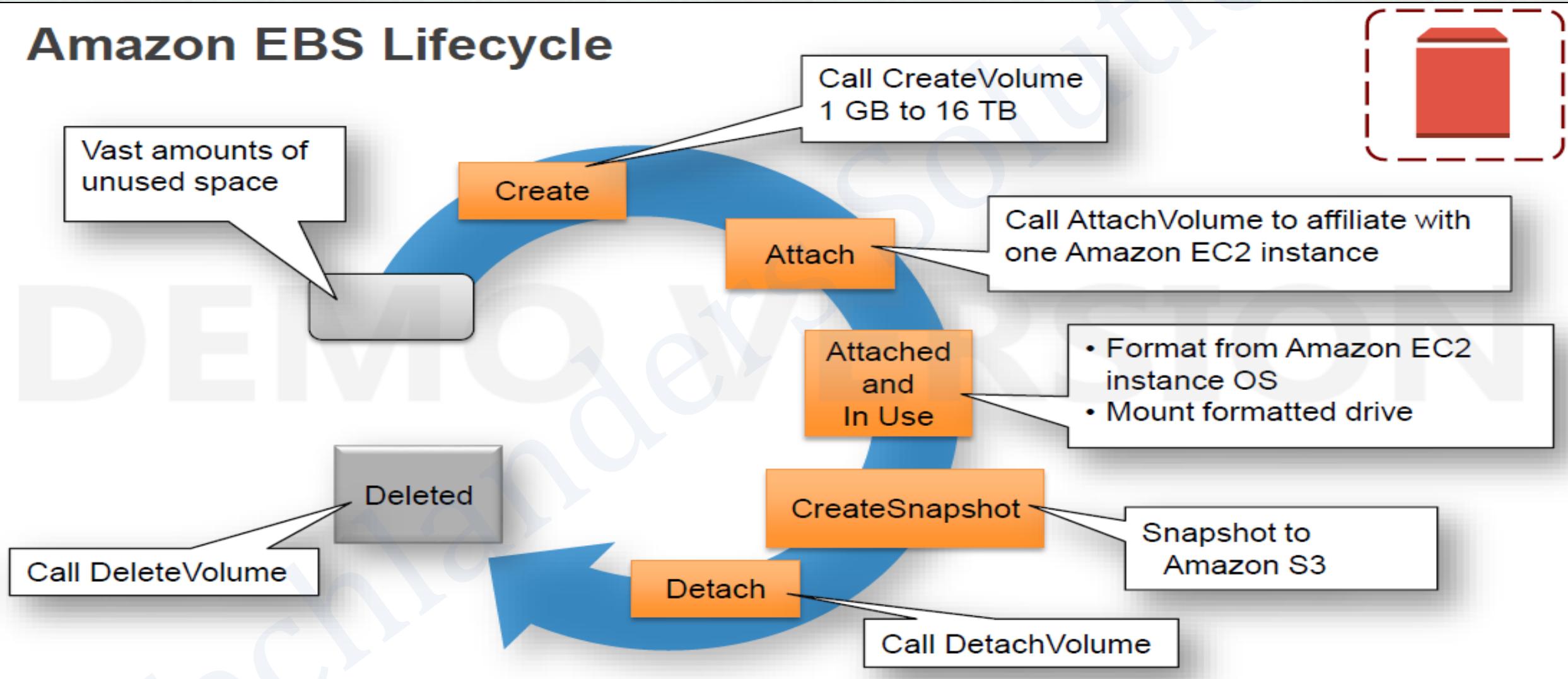
Amazon EBS Volumes

Amazon EBS

- EBS stands for ***Elastic Block Storage***
- **Persistent block level storage** volumes offering consistent and low-latency performance
- Automatically replicated within its Availability Zone
- Strictly bind to Availability Zone.

Amazon EBS Lifecycle

Amazon EBS Lifecycle



EBS Use Cases

- OS – Use for boot/root volume, secondary volumes
- Databases – Scales with your performance needs
- Enterprise applications – Provides reliable block storage to run mission-critical applications
- Business continuity – Minimize data loss and recovery time by regularly backing up using EBS Snapshots.
- Applications – Install and persist any application

EBS Volume Type Comparison

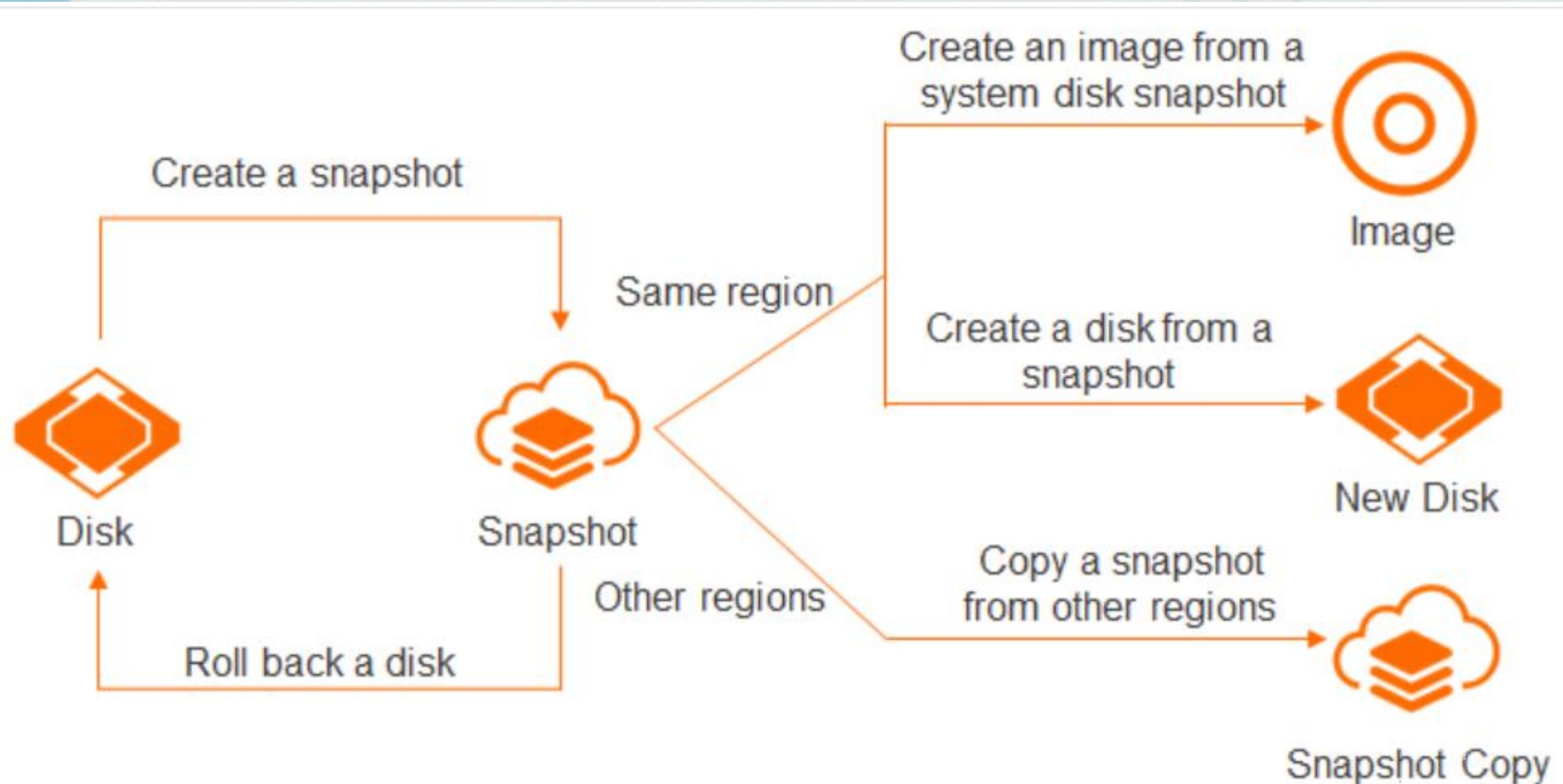
	Solid-State Drives (SSD)		Hard disk Drives (HDD)	
Volume Type	General Purpose SSD (gp2)*	Provisioned IOPS SSD (io1)	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Description	General purpose SSD volume that balances price and performance for a wide variety of workloads	Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads	Low cost HDD volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
Use Cases	<ul style="list-style-type: none"> Recommended for most workloads System boot volumes Virtual desktops Low-latency interactive apps Development and test environments 	<ul style="list-style-type: none"> Critical business applications that require sustained IOPS performance, or more than 10,000 IOPS or 160 MiB/s of throughput per volume Large database workloads, such as: <ul style="list-style-type: none"> MongoDB Cassandra Microsoft SQL Server MySQL PostgreSQL Oracle 	<ul style="list-style-type: none"> Streaming workloads requiring consistent, fast throughput at a low price Big data Data warehouses Log processing Cannot be a boot volume 	<ul style="list-style-type: none"> Throughput-oriented storage for large volumes of data that is infrequently accessed Scenarios where the lowest storage cost is important Cannot be a boot volume
API Name	gp2	io1	st1	sc1
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max. IOPS**/Volume	10,000	32,000***	500	250
Max. Throughput/Volume	160 MiB/s	500 MiB/s†	500 MiB/s	250 MiB/s
Max. IOPS/Instance	80,000	80,000	80,000	80,000
Max. Throughput/Instance††	1,750 MiB/s	1,750 MiB/s	1,750 MiB/s	1,750 MiB/s
Dominant Performance Attribute	IOPS	IOPS	MiB/s	MiB/s

LAB 9 : Working with Volumes

- Go to EC2 Instance created in → EBS → Volume → Create Volume
- Create a general purpose SSD Volume “Data-volume” of 1 GB
- While creating volume, check different combination of IOPS and Throughputs, by selecting different-2 Volumes types
- Attach volume to a server
- Try to create another 1 gb volume but in a different AZ 2 and try to attach it with instance in AZ 1.
- Detach volume from Server
- Delete the volumes.
- Make sure the volume and EC2 instance should be in same AZ to attach.

Snapshot

- Snapshots are Point-In-Time Copy of Volumes (not Servers)
- You can create snapshots from your existing volumes assigned on dedicated instance and use them on any other instance with same configuration/data.
- Snapshots can be taken for root/Data volumes and no reboot/unmount of volume required for same.
- Snapshots can be Private, can be Shared-with-limited-aws-account or can be public. Permission can be modified for same.
- Further, volumes or Images (Templates) can be created from Snapshots.
- You can copy the snapshot to other regions (charges will be applied).



LAB 10 : Working with Snapshots

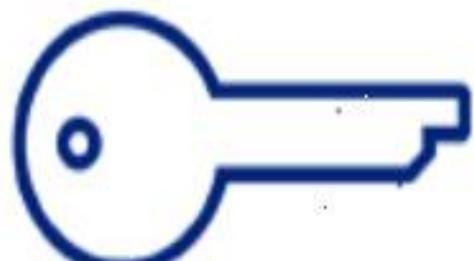
- Go to EC2 Instance in AZ 1 → EBS → Volumes → Create a data volume of 1 gb and attach it to the server in AZ1.
- Create a server in AZ2.
- Create a Snapshot of “Data-volume1” of server in AZ1
- Create “data-volume2” from snapshot in other AZ 2
- Attach “data-volume2” to server in AZ 2
- Delete the volume and delete the snapshot.

Encryption

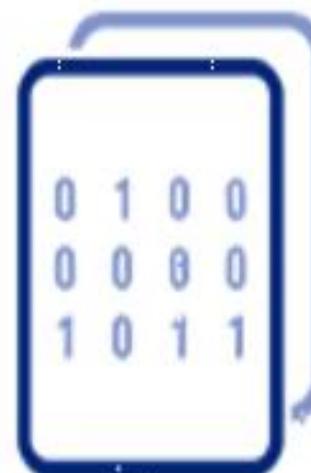
(used to protect sensitive information)



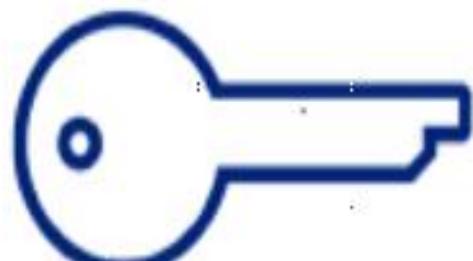
Plain text



Encryption



Encrypted text



Decryption



Plain text

EBS Encryption

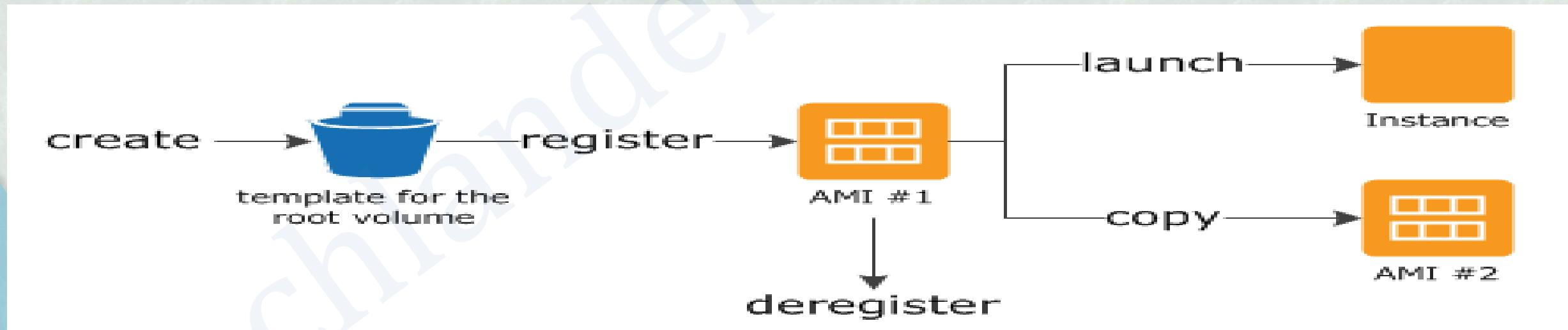
- EBS Encryption allows you to encrypt your data at rest or at flight.
- When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:
 - Data at rest inside the volume
 - All data moving between the volume and the instance
 - All snapshots created from the volume
 - All volumes created from those snapshots
- You can expect the same IOPS performance on encrypted volumes as on unencrypted volumes, with a minimal effect on latency.

EBS Encryption

- EBS encrypts your volume with a data key using the industry-standard AES-256 algorithm.
- There is no direct way to :
 - encrypt an existing unencrypted volume
 - remove encryption from an encrypted volume (However, you can migrate data between encrypted and unencrypted volumes.)
 - remove encryption from an encrypted snapshot.

AMI's

- **AMI is a template for the your server** (example: an OS image, a webserver, an application server etc.)
- AMIs are snapshots of your running system and AWS modify same to make it generic.
- In runtime too, you can create AMI from an instance, but that can be inconsistent.



AMI's

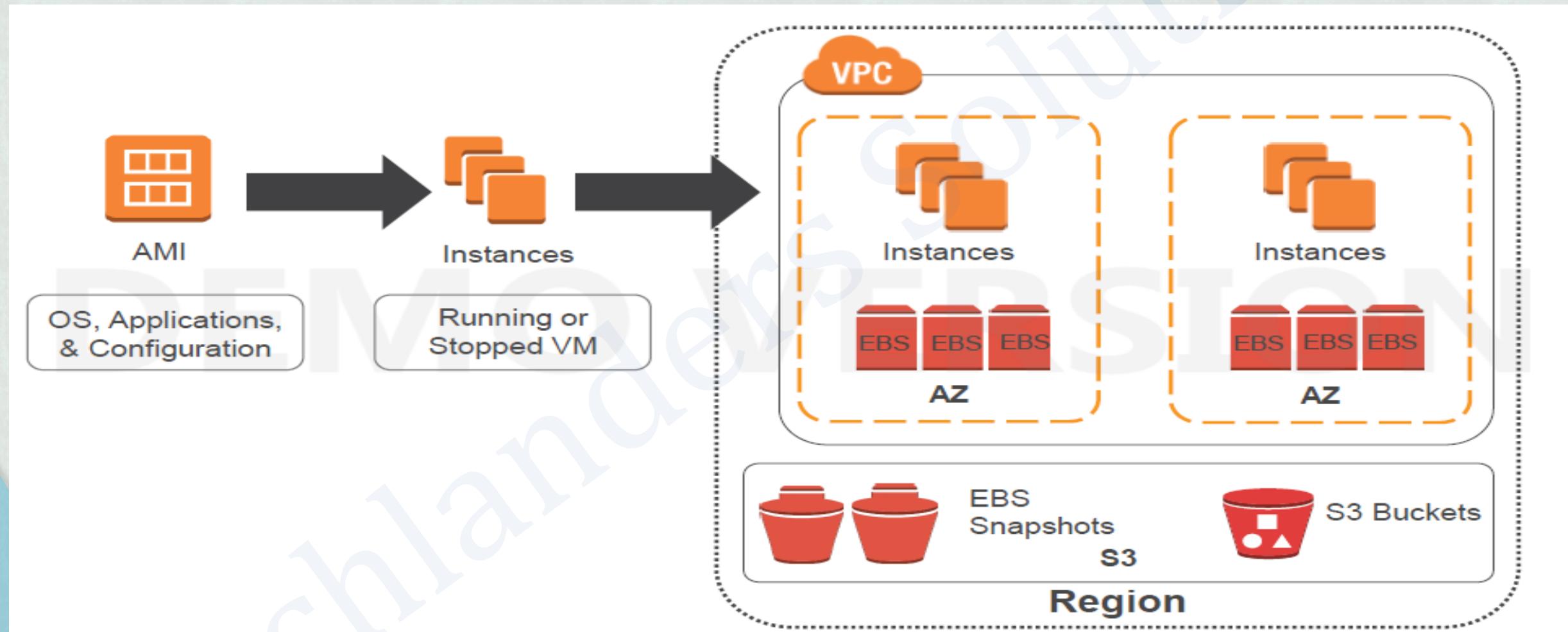
Generalizing an Image:

- Before others can use your Image, you should Generalize your AMI.
- Generalizing means removing few user specific configurations from that Image like:
 - Existing Users & their Credentials
 - Security Keys
 - User specific home directory
 - IP addresses & Hostnames
 - Event logs and volume specific settings

LAB 11 : Working with AMI

- Login to any EC2 instance and make some test file there.
- Now go to EC2 Instance → Select Server → Actions → Image → Create Image
- Type Image name and description
- Select whether Reboot or no reboot.
- Click on create Image
- Once finished, click on image and click launch instance
- Login to new server and check your test files there.

Amazon EC2 Instances



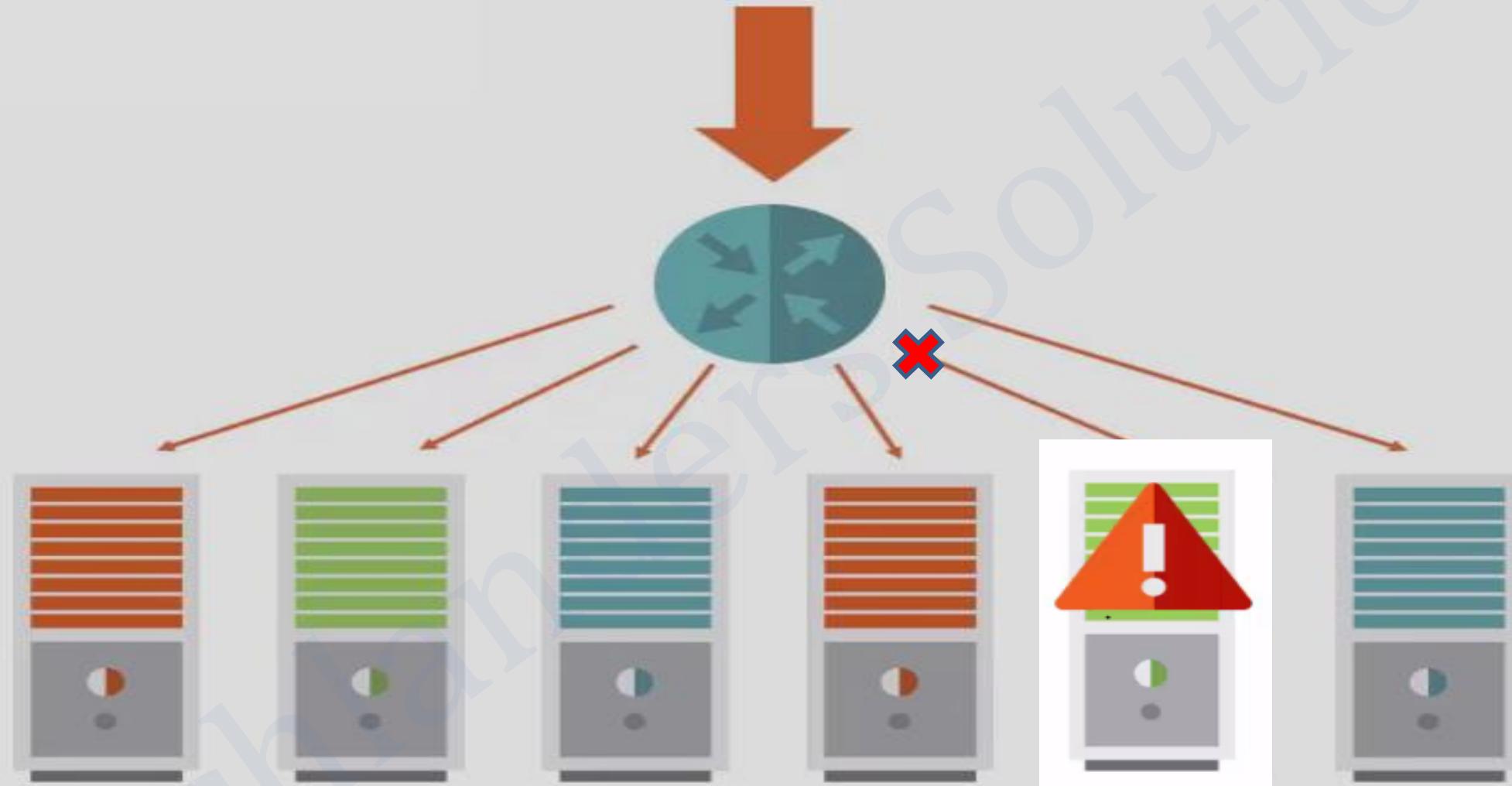


AWS
Load Balancer as a Service - ELB

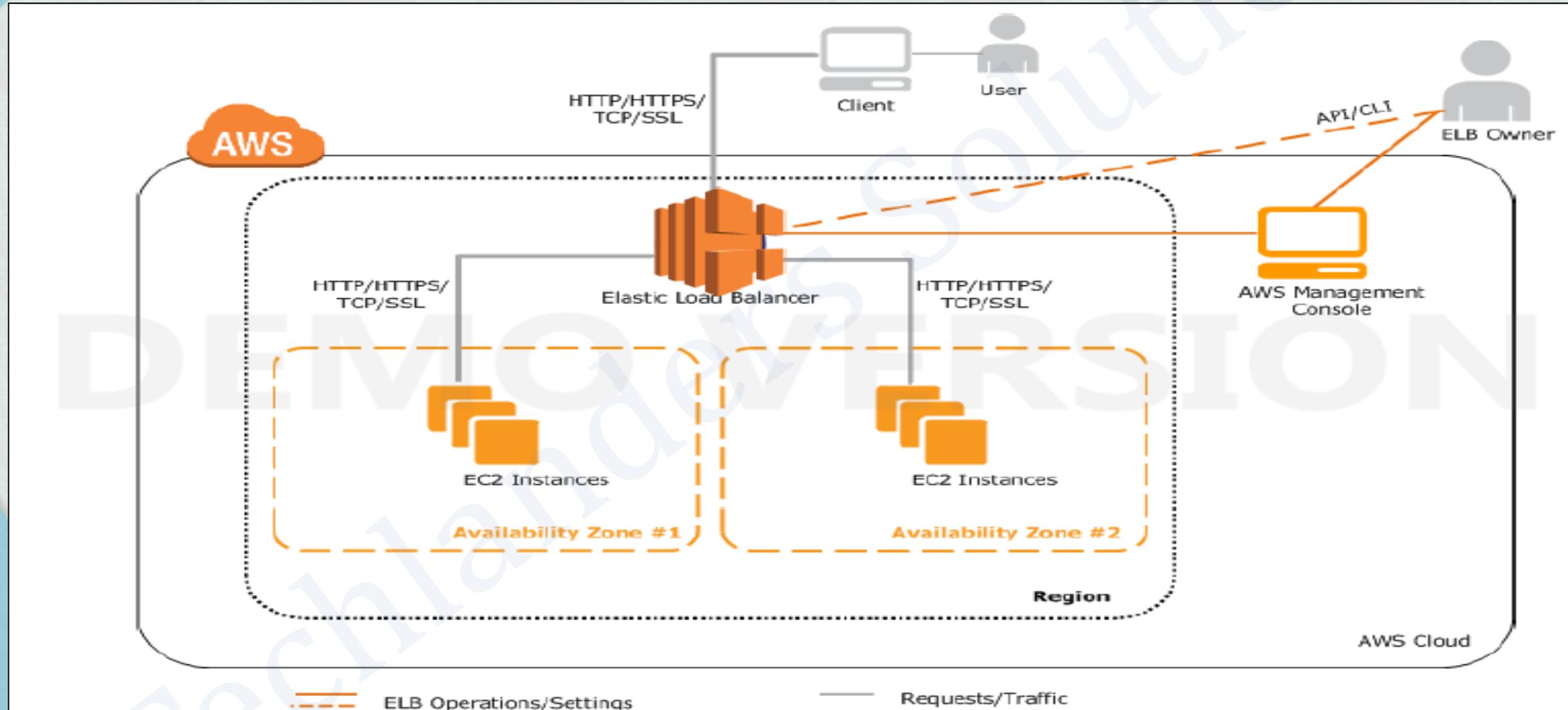
AWS Elastic Load Balancing

- **Distributes** traffic across multiple instances
- Supports **health checks** to detect unhealthy Amazon EC2 instances
- Supports the **routing and load balancing** of HTTP, HTTPS, and TCP traffic to Amazon EC2 instances
- Works within Region and across Availability zones, which means it can divert traffic to servers in different AZs, but within same Region.

Elastic Load Balancers

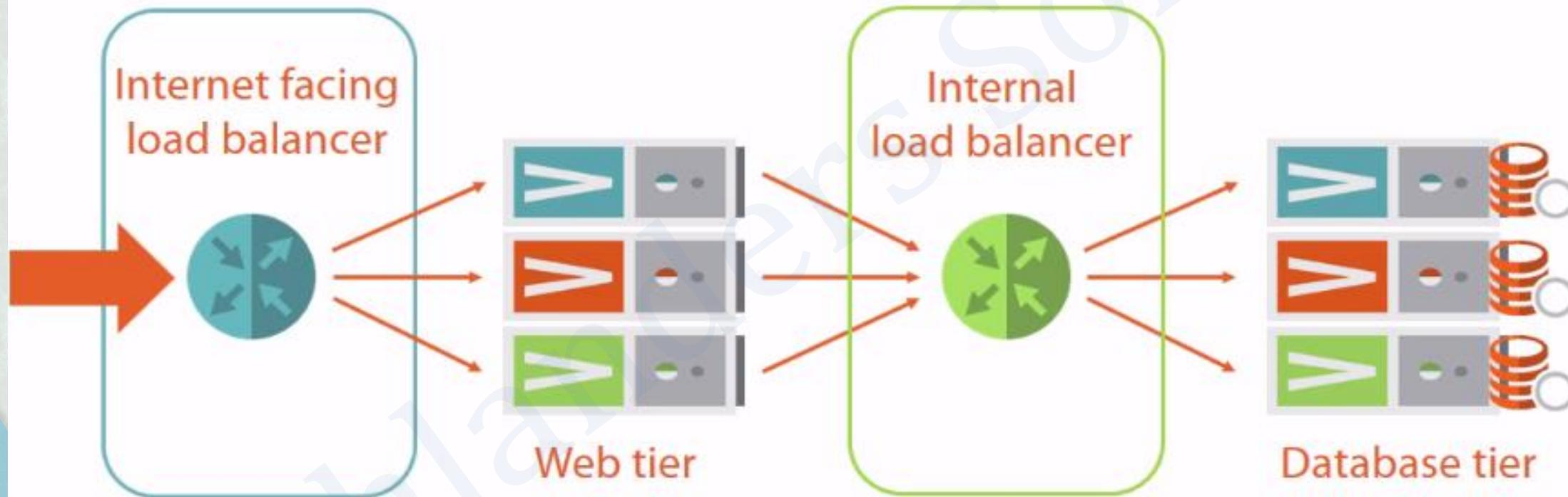


AWS Elastic Load Balancing



Load Balancer Types and Tiering

Tiered Applications & Load Balancers



ELB Terms

- **External/Internet Facing Load Balancer:** Receives Traffic from Open Internet. So carrying a public IP (or DNS name).
- **Internal Load Balancer:** Receives traffic from other VMs in same Network.
- **Frontend IP :** Public IP (Private IP in case of Internal LB) address for the incoming network traffic.
- **Target group :** Backend VM's Group which will receive actual traffic
- **Target:** Backend VM machines
- **Listener:** Port on ELB, which will accept request from end-client
- **Health checks:** Determines if a VM in the pool is healthy.

ELB Types

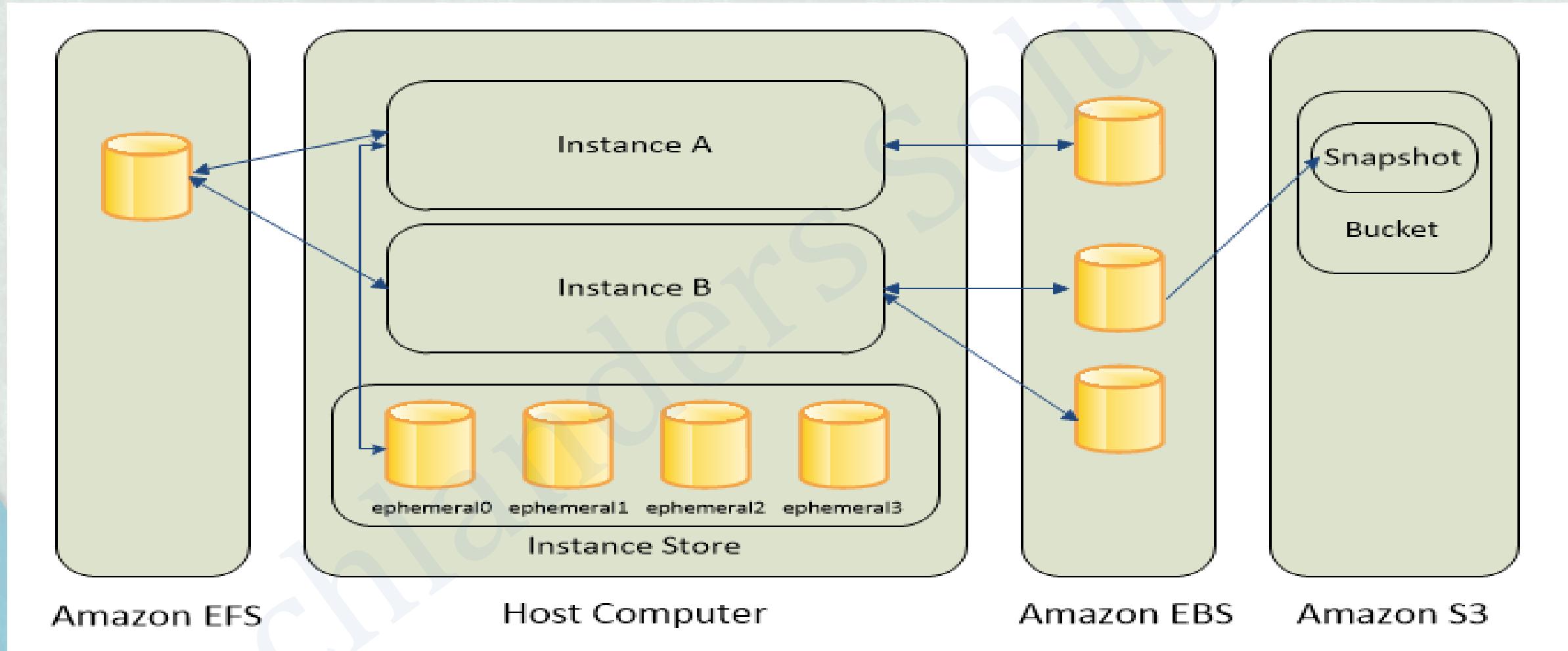
- **Application Load Balancer:**
 - An advance Load balancer which works on OSI Layer 7 (Application layer).
 - It contains flexible feature set (i.e. TLS termination, SSL handling etc.) for your web applications with HTTP and HTTPS traffic.
- **Network Load Balancer:**
 - Most commonly used Load balancer which works at any TCP port.
 - Works on Layer 4 (Transport layer).
 - LB with very high performance and capable of handling millions of requests per second while maintaining ultra-low latencies.
- **Classic Load Balancer:**

Old type of Load-balancer service which works with EC2-Classic network. Not being used any more.

LAB 12 : Working with ELB

- Create two AmazonLinux servers in different-2 Azs (us-east-1a &us-east1b)
- Make sure to add html script user-data for post-verifying on both servers.
- Create a security for alb---ec2 for access of traffic inside ec2 instances from our alb giving port 22 and 80 access to public for now and configure them on both servers us-east-1a and 1b while creating the instances.
- After creating both servers, create the application load balancer and while creating the LB , create a separate security group for ALB to give access to external users to our load balancer.
- Create a target group adding both instances to that target group.
- After adding a target group , click on create load balancer and wait for sometime.
- Now add user—ec2 securitygroup to alb—ec2instances security group to disallow the external users to access our internal instances behind the load balancers directly.

AWS Storage Services



AWS Auto Scaling

What do you understand by “Scaling”?

AWS Auto Scaling

What is the difference between Scale-up and Scale-out?

Scaling your Infra



AWS Auto Scaling

What is Auto-Scaling and Why required?

AWS Auto Scaling

- **Scale (out/in)** your Amazon EC2 capacity **automatically**
- Well-suited for applications that experience **variability in usage**
- Available at no additional charge
- Application instance Scale Out/In is done by service, based on the alerts or defined schedule.
- You can use LB service with Auto-scaling.
- Application should be able to handle scale-out and scale-in servers.
- Auto-scaling can span across-AZ, but remains within Region.

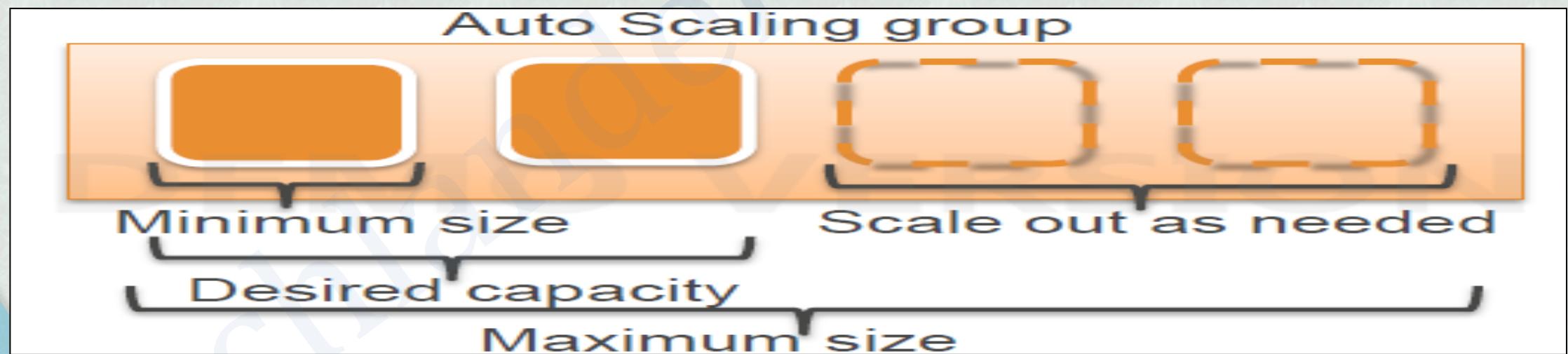
AWS Auto Scaling

Launch template : A launch configuration is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch template, you can specify:

- AMI ID
- Instance type
- Key pair
- Security groups
- Block device mapping
- User data

AWS AutoScaling

- **Auto Scaling Group:** Contain a collection of EC2 instances that share similar characteristics.
- Instances in an Auto Scaling group are treated as a logical grouping for the purpose of instance scaling and management.



Scaling conditions

Alert Based

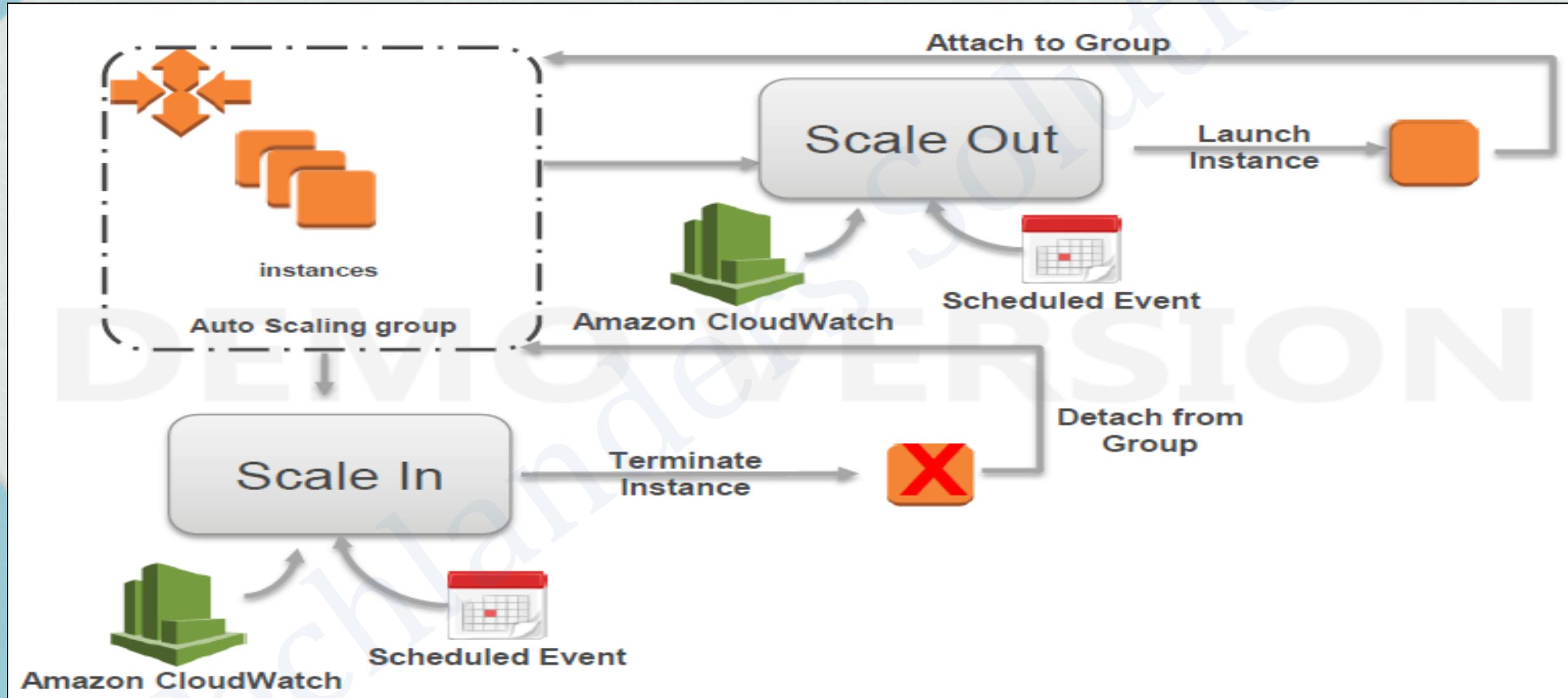
- Based on Health check alerts of Application instances
- Alerts can be based on CPU/Network/Disk IO etc.
- Both Scale-in and Scale out can be done based on Alerts
- Multiple Scale policies can be created and maintained

Scheduled Scaling

- Based on Scheduled Events
- Scaling can be configured to Auto scale once or re-occurrence basis.
- Scheduling can be done for years in advance
- Multiple Schedule policies can be created and maintained.

Note: Both Alert based and Scheduled scaling can co-exist simultaneously.

AWS AutoScaling



LAB 13 : Create AS Group

Console Access: <https://console.aws.amazon.com>

Go to aws console page

- From Services → EC2 → Auto Scaling → Launch temeplate
- Click on Create Launch template after adding required configurations like AMI, Configuration Details, Storage, Security Groups .
- Next step is to create an Autoscaling Group, click on create Auto scaling group from the launched template.
- Group Size is the key here (the number of instances group should have any time, this is also called desired capacity)
- Click next with the policies(optional)
- Click Create AutoScaling group and the instance will start automatically

LAB 14 : Work with AS Group

Console Access: <https://aws.amazon.com/>

Exercice1:

- **Add one more value in Auto scale group** and check one more instance will start getting created.
- Note: In this case we need to modify Desired/Minimum and Maximum attributes values for auto scaling group

Exercice2:

- **Deletion of Auto scale group**
- It will delete all of the instance also (created from Autoscale group).

AWS Cloud watch

- A **monitoring service** for AWS cloud resources and the applications you run on AWS
- **Visibility into** resource utilization, operational performance, and overall demand patterns
- **Custom application-specific** metrics of your own
- **Accessible** via AWS Management Console, APIs, SDK, or CLI
- **Monitor** other AWS resources
- View **graphics** and **statistics**
- Set **Alarms**

LAB 15 : SETTING AN ALARM

- Create a linux server in us-east-1 region.
- Create an alarm selecting cpu utilization as metric
- Add the instance id to associate with the alarm.
- Configure the properties of the alarm.
- Install stress package on the instance:

For amazon linux :

```
sudo amazon-linux-extras install epel -y  
sudo yum install stress -y  
stress --cpu 16
```

Note :As per the configuration select ec2 action as to STOP the instance , it will give an alarm and stop the instance.



AWS S3

Amazon S3

- Amazon S3 stands for **Simple Storage Services**.
- Amazon S3 is easy-to-use **object storage** with a **simple web service interface** that you can use to store and retrieve any amount of data from anywhere on the web.
- Natively always online, access via HTTP, It's a Storage services for the internet.
- **Highly scalable**, reliable, fast and durable.
- Amazon S3 also allows you to **pay only for the storage you actually use**, which eliminates the capacity planning and capacity constraints associated with traditional storage.

Amazon S3 Facts

- Able to store an **unlimited number of objects** in a bucket.
- Objects **up to 5 TB**; no bucket size limit.
- Designed for **99.99999999%** 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.

Use Cases in Production

- 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
- Disaster Recovery

Note: Pay only for what you use, No minimum fee, Estimate monthly bill using the **AWS Simple Monthly Calculator**

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.

LAB 17 : Create S3 Bucket

- Open GUI console <https://console.aws.amazon.com>
- Go to Services → Storage → S3
- Click Create bucket
- Enter Bucket name, select Region, Go default with properties, grant public access and finally create bucket
- Once the Bucket is created, check the properties you can enable in Bucket

Steps To make an object public

- To make an object public ::
- upload an object
- go to permissions
- set " Block all public access " to off
- change the object ownership in permisions as acl enabled
- edit Access control list (ACL) to read from public access
- select the object from the bucket
- select make public via acl

LAB 18 : Managing Objects

Upload, Make Public, and Delete Objects in Your Bucket

In this exercise, you will upload a new object to your bucket. You will then make this object public and view the object in your browser. You will then rename the object and finally delete it from the bucket.

1. Load your new bucket in the Amazon S3 console.
2. Select Upload, then Add Files.
3. Locate a file on your PC that you are okay with uploading to Amazon S3 and making public to the Internet. (We suggest using a non-personal image file for the purposes of this exercise.)
4. Select a suitable file, then Start Upload. You will see the status of your file in the Transfers section.
5. After your file is uploaded, the status should change to Done.
6. The file you uploaded is now stored as an Amazon S3 object and should be now listed in the contents of your bucket.
7. Try to give access to an object using ACL enabled(for other accounts).
8. Try to give read access by going to ACL in permissions.
9. Make sure to make the object public everytime while using ACL enabled policy
10. Finally delete the object.

S3 Bucket Properties

Amazon S3 Buckets have multiple properties, which makes them very efficient to use.

- Versioning (Versioning of Objects – SVN)
- Server Access Logging (Bucket access logs)
- Tags (to track your billing and differentiate your bucket)
- Object Level Logging (Object access logs, additional Cost)
- Encryption
- Static website hosting
- Notification triggering on S3 events
- Requester pay
- Storage Class (Object level)

S3 - Version Control

- Versioning enables you to keep multiple versions of an object in one bucket.
- It protects you from the consequences of unintended overwrites and deletions.
- Helps to roll-back or go on previous versions
- When you enable versioning on a bucket, objects already stored in the bucket are unchanged.
- Once we have enabled versioning, Amazon S3 assigns a unique version ID value for the object
- Versioning can be suspended and resumed at any stage.

LAB 19 : S3 Object Versioning

- Open GUI console <https://console.aws.amazon.com>
- Go to Services → Storage → S3
- Click Create bucket
- Enter Bucket name, select Region, Enable versioning and encryption under properties tab, grant public access and finally create bucket
- Try to upload some objects in S3 bucket. Try to re-upload the same name file with modified contents
- Check the multiple versions availability for object, under the Object's "Latest version" tab
- Delete the file to show delete marker (make sure to uncheck show versions tab before deleting)

S3 Storage Classes

- **Standard**
For frequently accessed data. Stores object data redundantly across multiple geographically separated Availability Zones
- **Standard-IA**
For infrequently accessed data. Stores object data redundantly across multiple geographically separated Availability Zones. Minimum 30-day retention period and minimum 128 KB object size.
- **One Zone-IA**
For infrequently accessed data. Stores object data in only one Availability Zone at a lower price than Standard-IA. Minimum 30-day retention period and minimum 128 KB object size.
- **Reduced redundancy**
For frequently accessed data. Stores noncritical, reproducible data at lower levels of redundancy than Standard.

Comparing AWS Cloud Storages

		File	Object	Block
		Amazon EFS	Amazon S3	Amazon EBS
Performance	Per-operation latency	Low, consistent	Low, for mixed request types, and integration with CloudFront	Lowest, consistent
	Throughput scale	Multiple GBs per second	Multiple GBs per second	Single GB per second
Characteristics	Data Availability/ Durability	Stored redundantly across multiple AZs	Stored redundantly across multiple AZs	Stored redundantly in a single AZ
	Access	One to thousands of EC2 instances or on-premises servers, from multiple AZs, concurrently	One to millions of connections over the web	Single EC2 instance in a single AZ
	Use Cases	Web serving and content management, enterprise applications, media and entertainment, home directories, database backups, developer tools, container storage, big data analytics	Web serving and content management, media and entertainment, backups, big data analytics, data lake	Boot volumes, transactional and NoSQL databases, data warehousing & ETL



AWS Backup Services

S3 Object/Data Lifecycle

Amazon **S3 Object Lifecycle Management** is roughly equivalent to automated *storage Tiering* in traditional IT storage infrastructures. In many cases, data has a natural lifecycle, starting out as “hot” (frequently accessed) data, moving to “warm” (less frequently accessed) data as it ages, and ending its life as “cold” (long-term backup or archive) data before eventual deletion.

For example, the lifecycle rules for backup data might be:

- Store backup data initially in Amazon S3 Standard.
- After 30 days, transition to Amazon Standard-IA.
- After 90 days, transition to Amazon Glacier(kind of backup storage).
- After 3 years, delete.

Amazon Glacier

- Long term low-cost archiving /backup service
- Optimal for infrequently accessed data
- Designed for 99.999999999% durability and 99.99% accessibility
- 3-5 hours retrieval time, so not a real time volume/storage .
- New! Three Retrieval options ranging from minutes to hours.
- Cheapest storage among all Storage classes
- Can store unlimited amount of virtually any kind of data, in any format
- Vault lock can be applied for Audit purpose. So, objects will become consistent

Amazon Glacier Factsheet

- Data is stored as **archives**. An archive can contain up to 40TB of data, and you can have an unlimited number of archives.
- Archives are stored in **vaults**. Max 1000 vaults per account.
- Each archive is assigned a unique archive ID at the time of creation.
- All archives are automatically encrypted, and **archives are immutable**—after an archive is created, it cannot be modified.
- You can put notifications, tags, permissions on Vault
- Vault lock can be created to become Audit-enabled. Once lock is in place, even account owner can't change the archives or archive policy.

LAB 21 : Create Data Lifecycle

- Open GUI console <https://console.aws.amazon.com>
- Go to Services → Storage → S3 → [Bucket]
- Under Bucket → Management → Add Lifecycle rule
- Enter Rule name and filter name(optional).
- Configure transaction for current/previous versions and add transition rules.
- Configure expiration of objects/data, along with incomplete uploads checks.
- Review the Lifecycle management policy. Don't save , just review the lifecycle

LAB : BUCKET POLICY

- Create a new bucket.
- Permission > Unblock all public access from bucket
- Permissions > Generate the bucket policy using policy generator.
- Copy the policy in permissions of bucket.
- Select the object url and try to access it.
- Bucket contents are now accessible using bucket policy.

LAB : S3 WEBSITE HOSTING

- To host a bucket as website :
- Go to properties > enable static website
- Add html filename saved in the bucket as object.
- Try accessing the website bucket url now.

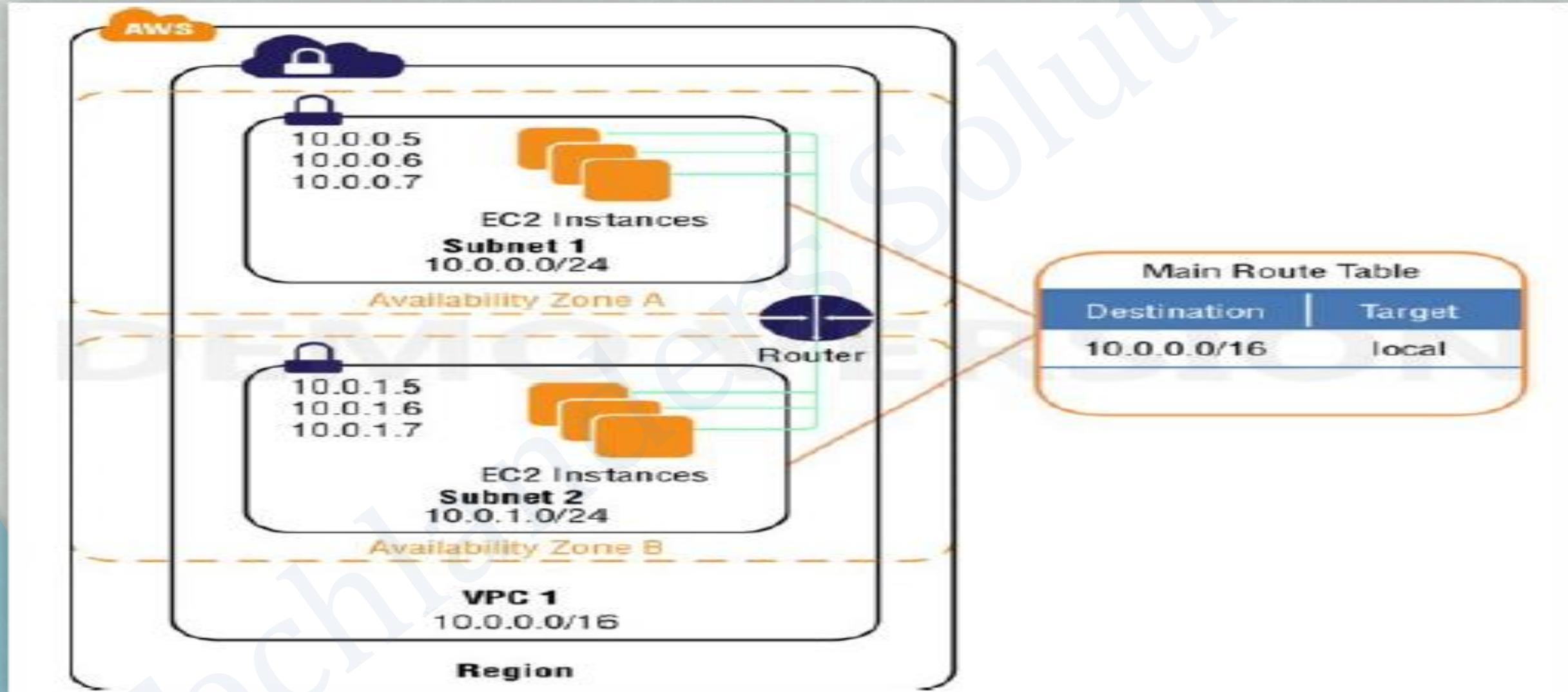


AWS **Network Services**

Amazon Networking (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).

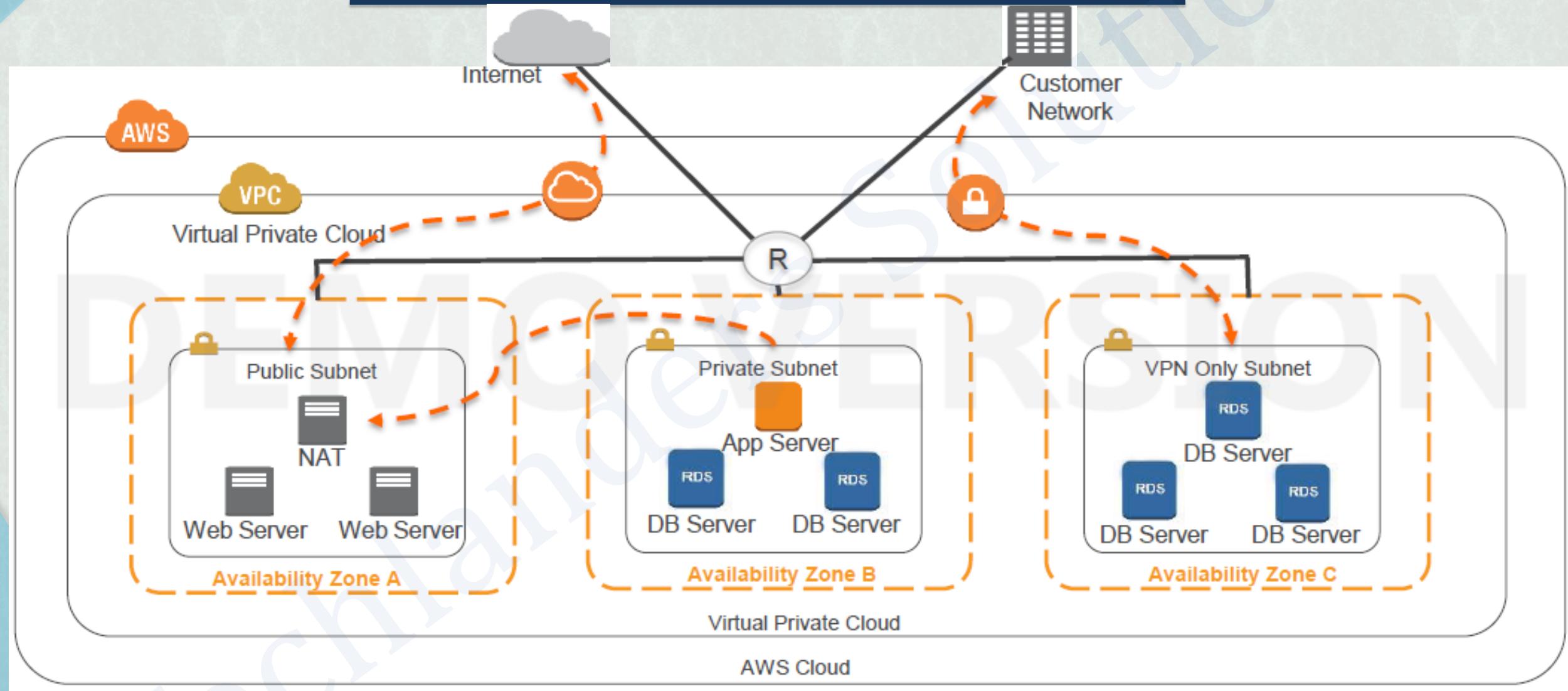
Amazon Networking (VPC)



Subnets

- A **subnet** defines a range of IP addresses in your VPC. A *subnet* is a segment of an Amazon VPC's IP address range where you can launch Amazon EC2 instances.
- AWS reserves the first four IP addresses and the last IP address of every subnet for internal networking purposes. For example, a subnet defined as a /28 has 16 available IP addresses; subtract the 5 IPs needed by AWS to yield 11 IP addresses for your use within the subnet.
- Each subnet must reside entirely within one Availability Zone and cannot span zones. You can add one or more subnets in each Availability Zone.
- Majorly three types of Subnets, based on usage: **Public**, **Private** and **VPN-only**.

VPC in Real Cloud World



VPC and Subnets

- You can launch AWS resources into a subnet that you select.
- A **private subnet** should be used for resources that won't be accessible over the Internet.
- A **public subnet** should be used for resources that will be accessed over the Internet.
- A **VPN only** subnet must talk only to Organization's VPN/services
- Each subnet must reside entirely within one Availability Zone and cannot span zones.

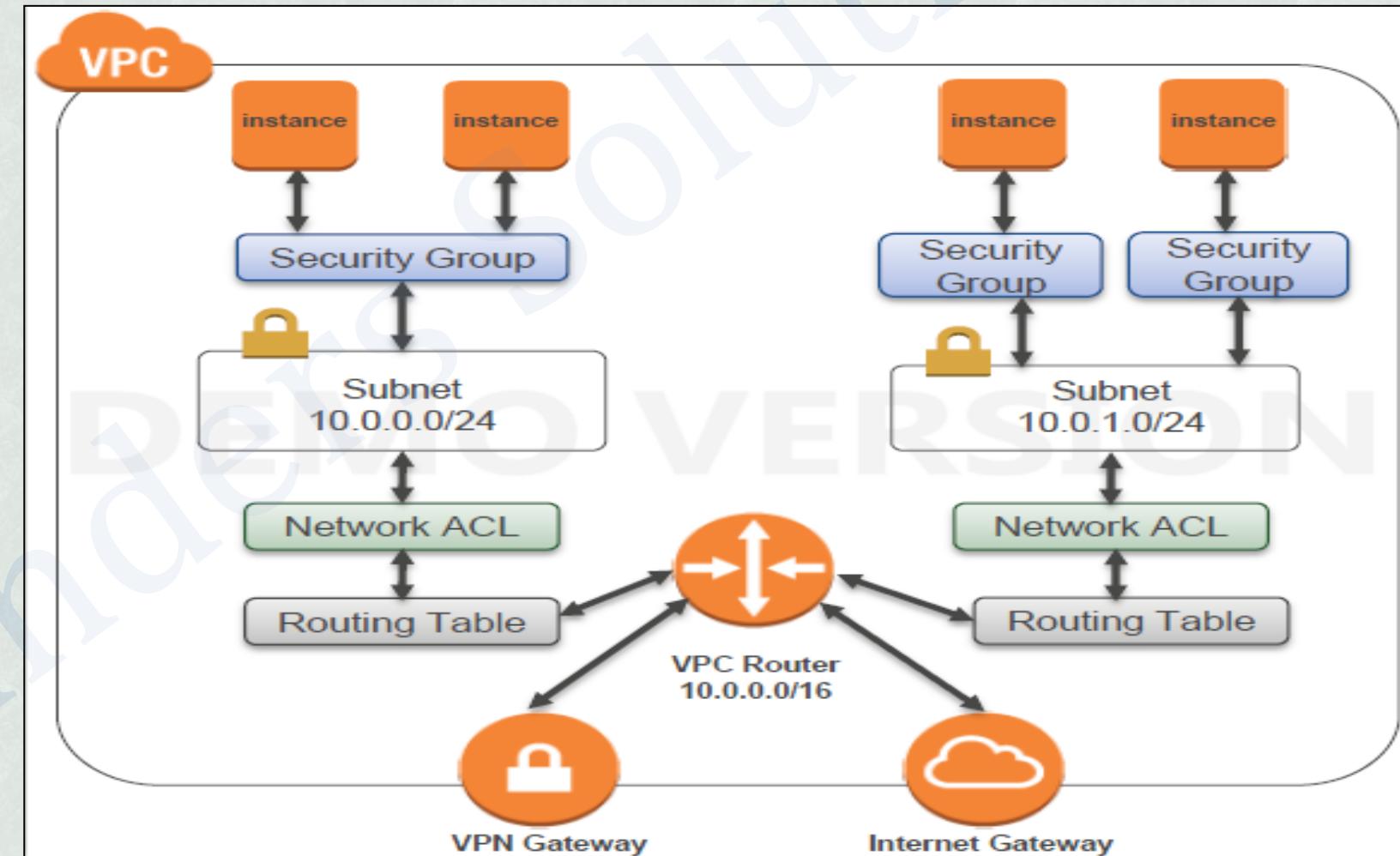
LAB 22 : Creating VPC

Create a Custom Amazon VPC

- Sign in to the AWS Management Console as an administrator or super user.
- Select the Amazon VPC icon to launch the Amazon VPC Dashboard.
- Create an Amazon VPC with a CIDR block equal to 10.0.0.0/16, a name tag of **My First VPC**, and default tenancy.
- You have created your first custom VPC.
- Now create a subnet under this VPC and launch a server in VPC

Security in VPC

- Security Group
- Network ACL
- Routing Tables

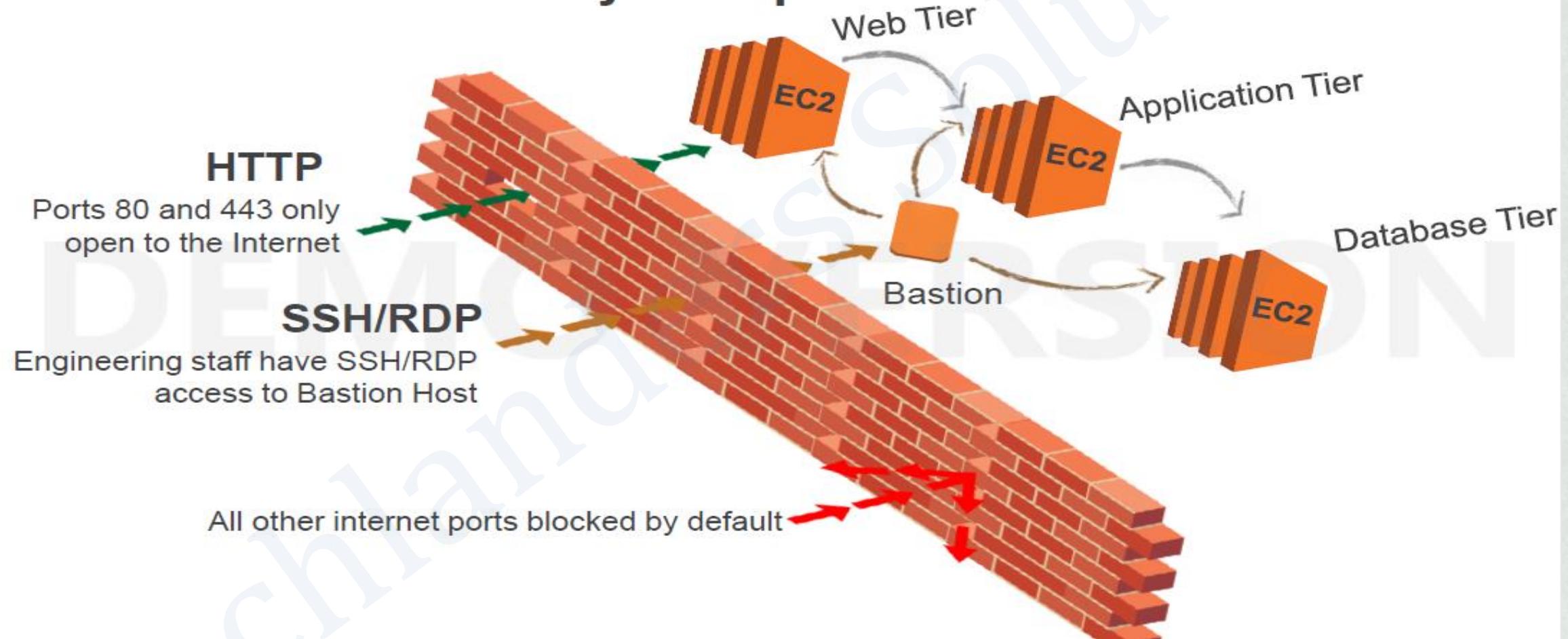


AWS Security

SSL Endpoints	Security Groups	VPC
Secure Transmission Establish secure communication sessions (HTTPS) using SSL/TLS.	Instance Firewalls Configure firewall rules for instances using Security Groups.	Network Control In your Virtual Private Cloud, create low-level networking constraints for resource access. Public and private subnets, NAT and VPN support.

Enhanced Security

AWS Multi-Tier Security Groups



LAB 23 : Working with Subnets

Working with subnets in your VPC

- Sign in to the AWS Management Console as an administrator or super user.
- Select your VPC under Amazon VPC Dashboard.
- Create two subnets under your VPC as:
 - Public-Subnet - 10.0.0.0/24 in us-east-1a
 - Private-Subnet – 10.0.1.0/24 in us-east-1b
- Create a public server under the public subnets (one in us-east-1a)
- While creating public server , make sure to enable auto assign public option during server creation.
- Make sure to add root password userdata while making the servers to bypass keypair encryption and access server by a password.
- and try to ping/ssh. Its working?

LAB 23 : Working with Subnets

- Create a route table named public route for giving vpc access to public server externally. In the route table , associate public subnet to the public route.
- Now Assign an Internet gateway to my-vpc and edit route in public route . Now check whether public server is accessible or not.
- Public subnet used for accessing the private subnet is called a **Bastion Host.**
- Note: Make sure to edit routes and add internet gateway to the public route.

LAB: NAT GATEWAY

Not in free tier

- Create a NAT gateway, select the public subnet to place the nat gateway and assign an elastic ip to it.
- Change the routes of private route table to associate it to the nat gateway.
- Try pinging or accessing internet on pvt instance now.

AWS VPC Peering

VPC Peering

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them.

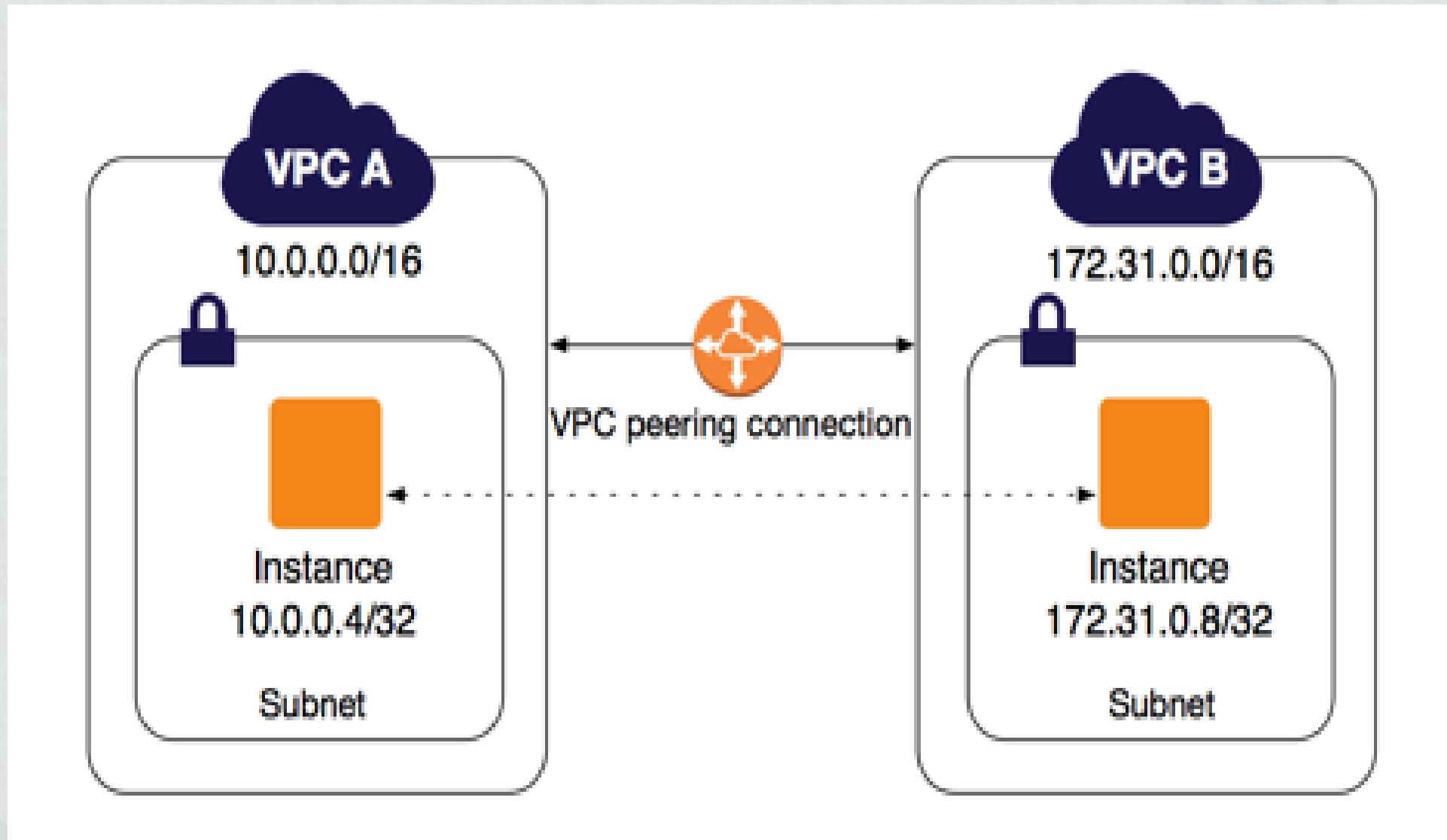
You can use private IPv4 addresses or IPv6 addresses to connect.

Instances in either VPC can communicate with each other as if they are within the same network.

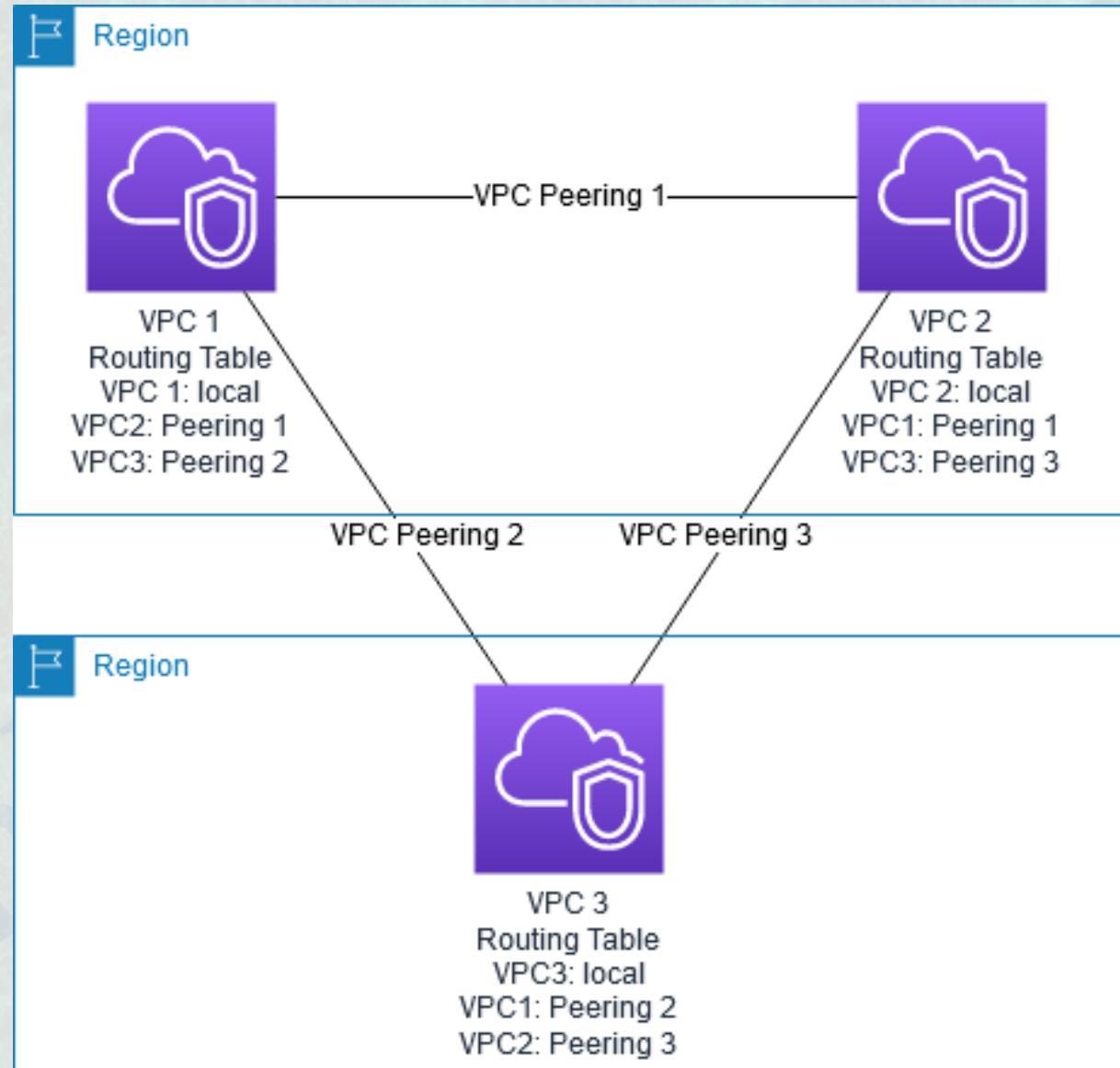
You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.

The VPCs can be in different regions (also known as an *inter-region* VPC peering connection). Its chargeable and incurs huge cost of VPNs and Traffic.

VPC Peering



VPC Peering



LAB: VPC Peering

Create VPC peering among two VPCs in same region

1. Create two VPCs in Mumbai Region and two subnets in each one
2. Create two servers in each VPC
3. Verify that servers within VPC are communicating with each other, while outside VPC, same is not pingable.
4. Create VPC peering by visiting “Peering connections” under VPC dashboard
5. Select both VPCs and create VPC connection. Approve the connection.
6. Add route to route tables & verify the traffic movement between these VPCs



AWS Monitoring Services

AWS Cloud watch

- A **monitoring service** for AWS cloud resources and the applications you run on AWS
- **Visibility into** resource utilization, operational performance, and overall demand patterns
- **Custom application-specific** metrics of your own
- **Accessible** via AWS Management Console, APIs, SDK, or CLI
- **Monitor** other AWS resources
- View **graphics and statistics**
- Set **Alarms**

LAB 24 : Cloud Watch

- Go through Logs and performance monitors in place for EC2 instances and observe cloud watch functioning.
- Create Alarms for CPU utilization

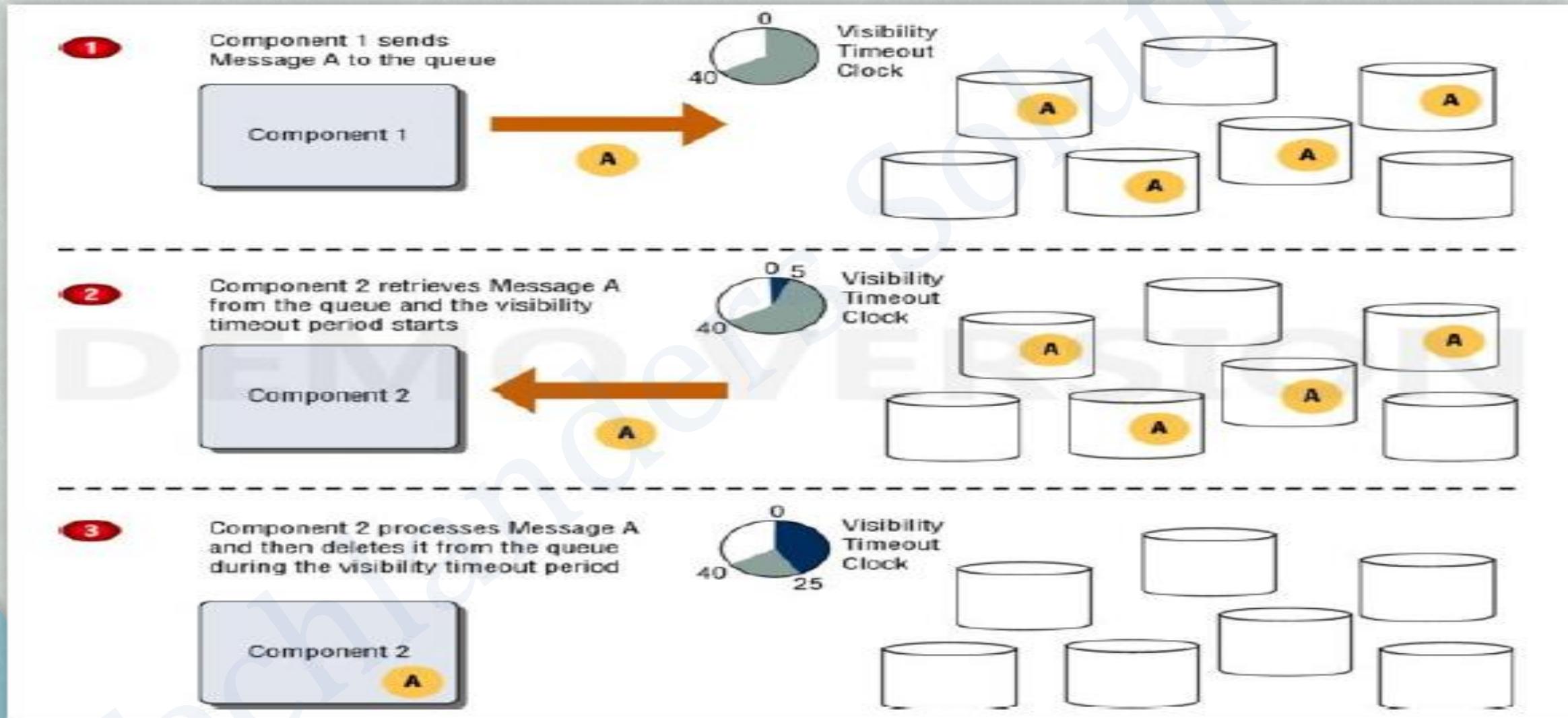
AWS SQS

- 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.

AWS SQS

- **Standard Queue:**
 - Unlimited Throughput
 - At-Least-Once Delivery, usually 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



LAB 38 : 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, 30 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.

LAB 39 : S3+SQS

We will be creating a system to trigger message in sqs queue as soon as an event occurs on S3.

- Create an sqs queue.
- Generate an SQS policy using policy generator to access sqs from any outside service.
- Go to s3 event notifications and set sqs object actions as an event for SQS service.
- Save it , then try to delete an object and notice there will be a message polled under SQS queue .

LAB 40 : SQS- DEAD LETTER QUEUE

- Create a queue named “deadqueue”
- Go to your previously made queue and select dead letter queue .
- Select a max message received limit.
- Try polling the messages again in the original queue and it will be working

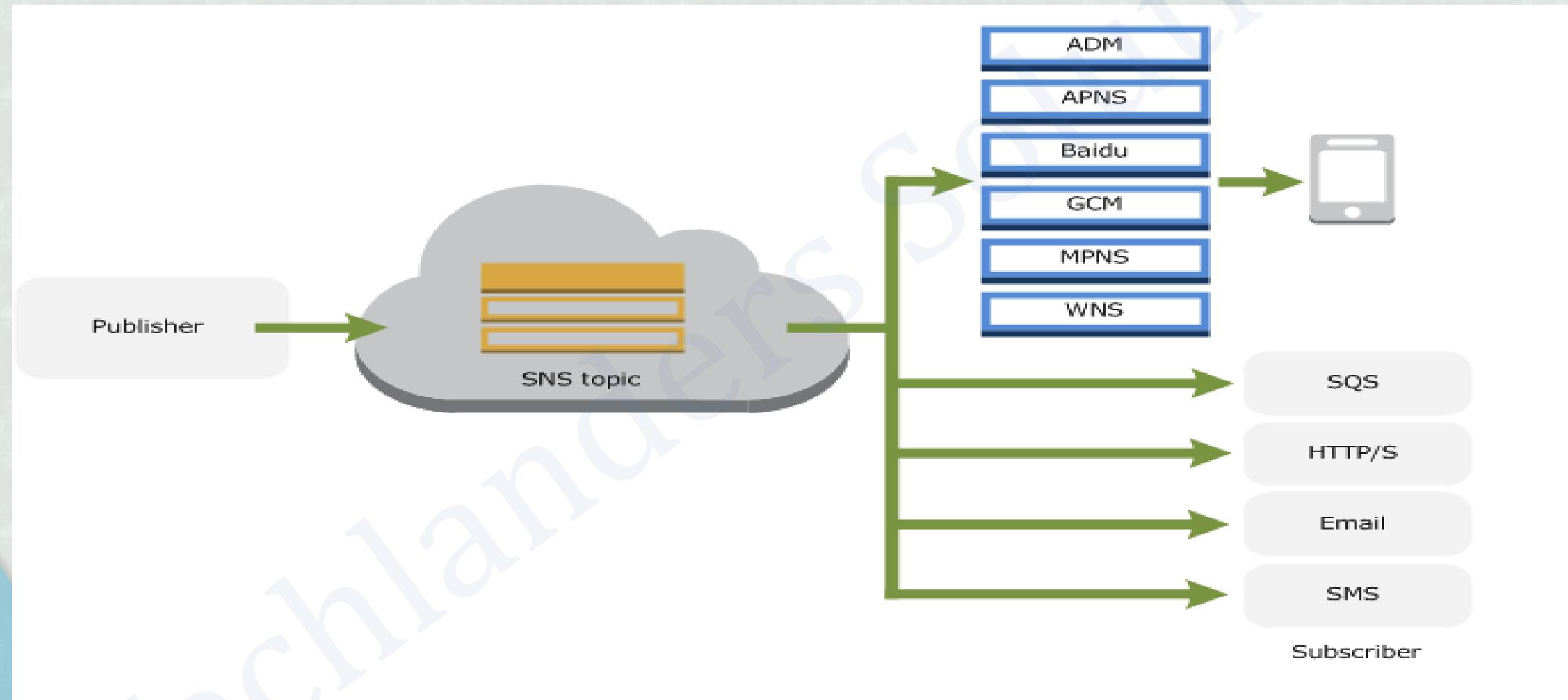
LAB 41: FIFO QUEUE

- Create a fifo queue named “demofifo”
- Type first message as “hello world1”
- Message group is : demo
- Message deduplication ID : 1
- Send four messages and observe the order of the messages changing deduplication to 2,3,4 as well.

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



LAB 42 : AWS SNS

Create an Amazon SNS Topic

In this exercise, you will create an Amazon SNS message.

1. Open a browser, and navigate to the AWS Management Console. Sign in to your AWS account.
2. Navigate to Mobile Services and then Amazon SNS to load the Amazon SNS dashboard.
3. Create a new topic, and use MyTopic for both the topic name and the display name.
4. Note that an Amazon Resource Name (ARN) is specified immediately.

LAB 43 : AWS SNS

Create a Subscription to Your Topic

In this exercise, you will create a subscription to the newly created topic using your email address. Then you confirm your email address.

1. In the Amazon SNS dashboard of the AWS Management Console, navigate to Topics.
2. Select the ARN that you just created. Create a Subscription with the protocol of Email, and enter your email address.
3. Create the Subscription.
4. The service sends a confirmation email to your email address. Before this subscription can go live, you need to click on the link in the email that AWS sent you to confirm your email address. Check your email, and confirm your address.

LAB 44 : AWS SNS

Publish to a Topic

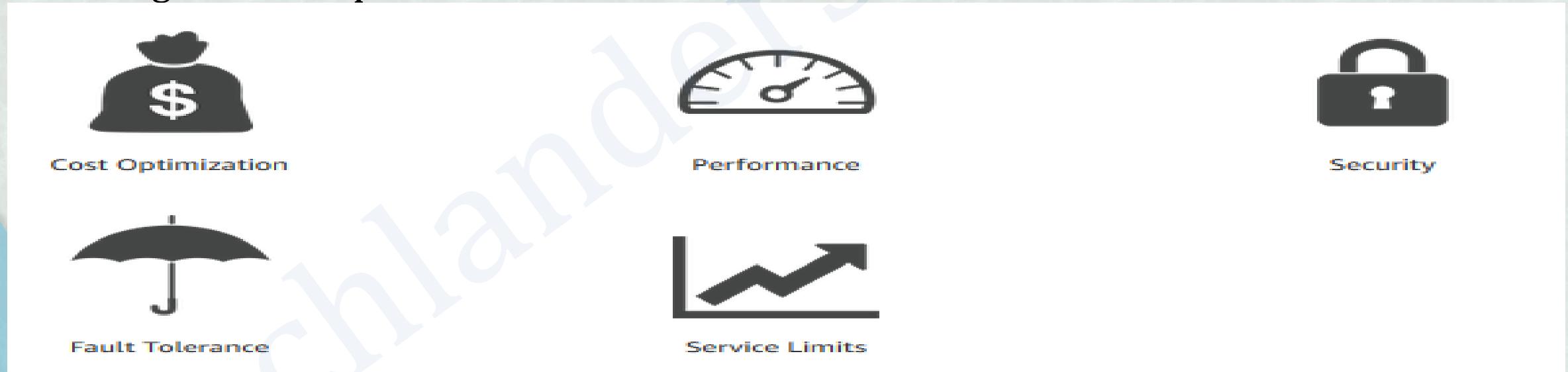
In this exercise, you will publish a message to your newly created topic.

1. In the Amazon SNS dashboard of the AWS Management Console, navigate to Topics.
2. Navigate to the ARN link for your newly created topic.
3. Update the subject with My Test Message, leave the message format to set to Raw, and use a Time to Live (TTL) field to 300.
4. Publish the message.
5. You should receive an email from your topic name with the subject that you specified. If you do not receive this email, check your junk folder.

AWS Trusted Advisor

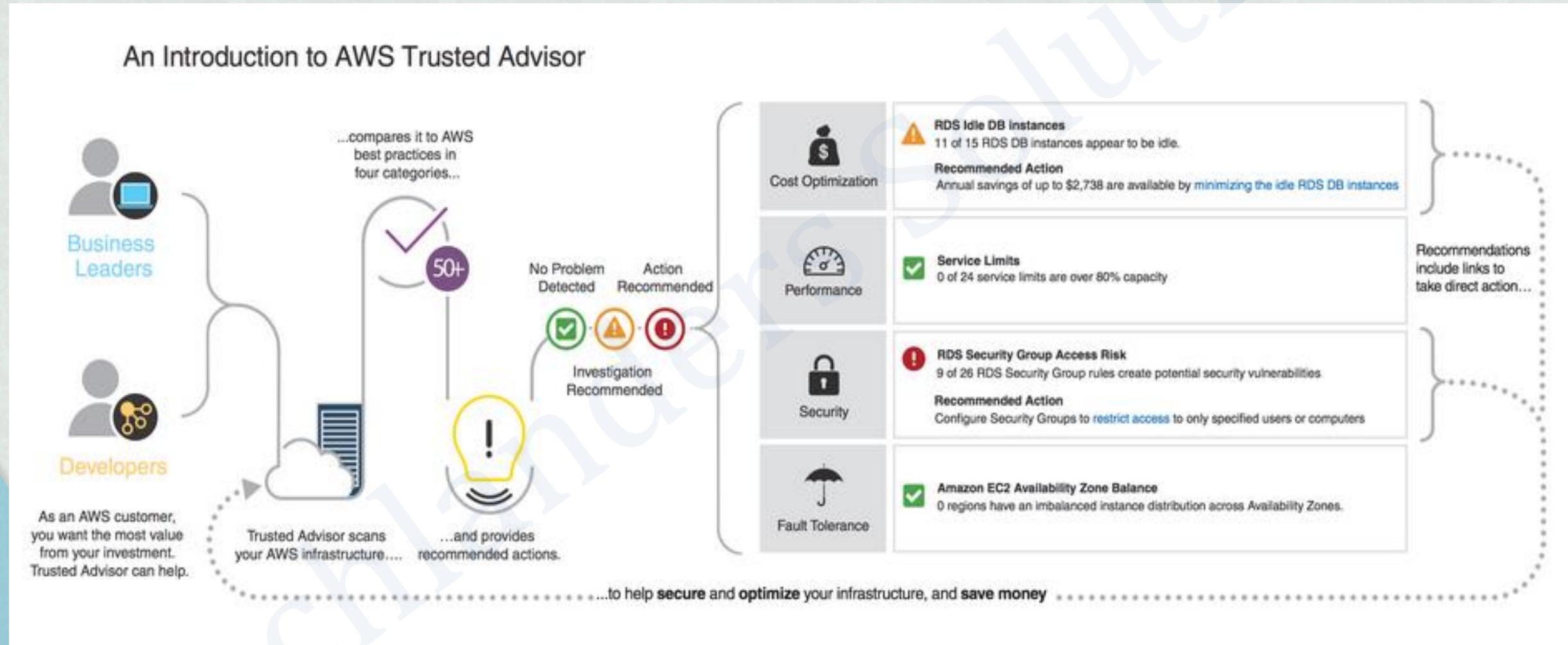
An online resource to help you reduce cost, increase performance, and improve security by optimizing your AWS environment.

Trusted Advisor provides real time guidance to help you provision your resources following AWS best practices.



AWS Trusted Advisor

An Introduction to AWS Trusted Advisor

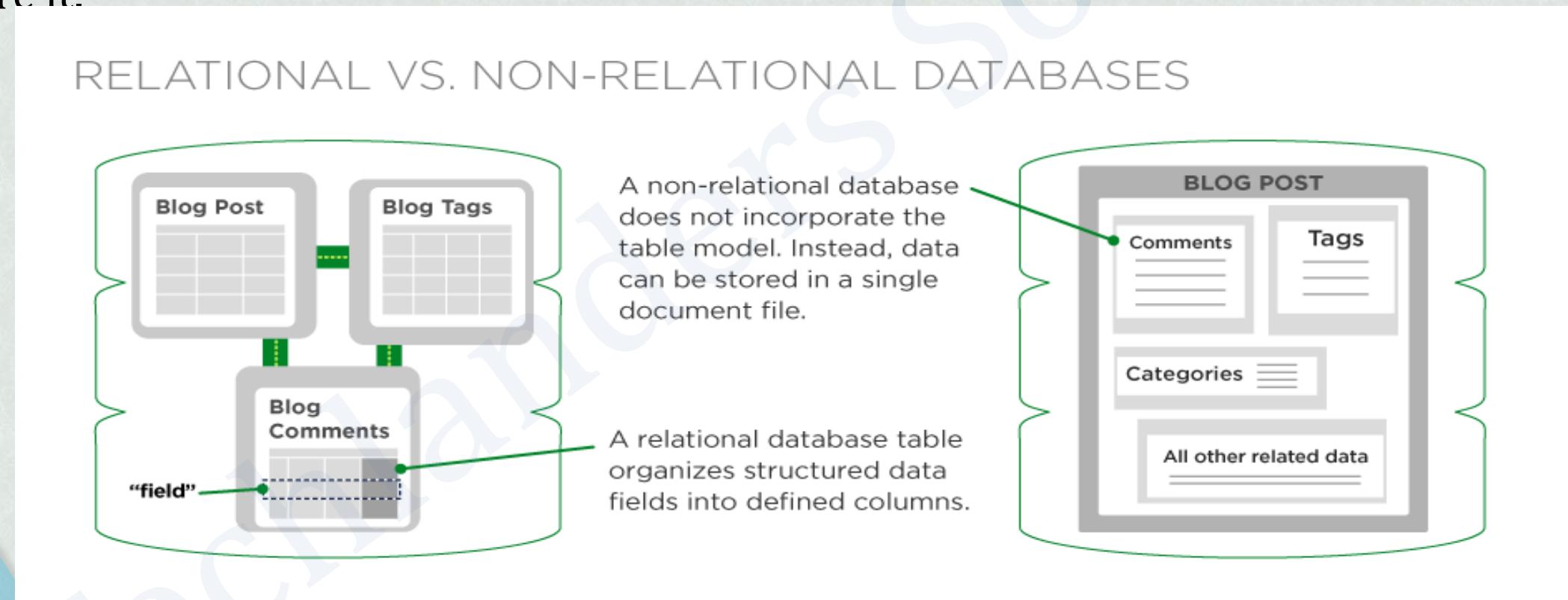




AWS Database as a Services

Database Types

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.



Relational databases

- Relational Databases are the oldest and widely used databases
- A Relational database consists of one or more **tables**, and a table consists of **columns** and **rows** similar to a spreadsheet. A database column contains a specific **attribute** of the record, such as a person's name, address, and telephone number. Each attribute is assigned a **data type** such as text, number, or date, and the database engine will reject invalid inputs.
- A Relational database can be categorized as either an Online Transaction Processing (OLTP) or Online Analytical Processing (OLAP) database system.
- Supports ACID (Atomicity, Consistency, Isolation, Durability) transactions
- **Example:** SQL database, Microsoft SQL Server, Oracle Database, IBM DB2, MySQL, Maria DB, Sybase, PostgreSQL

Non-Relational databases

- Recently came in lime light 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

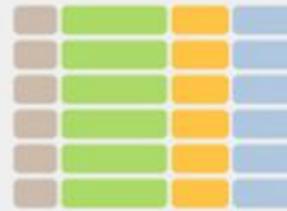
SQL vs NO-SQL

	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

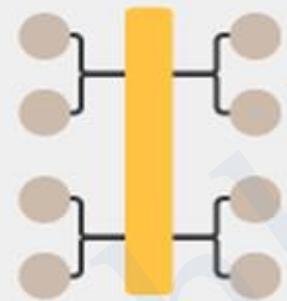
DB Sub-types

SQL Database

Relational

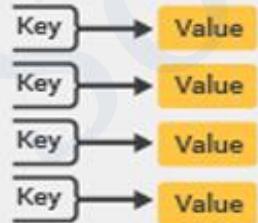


Analiticals (OLAP)



Non-SQL Database

Key-Value



Column-Family



Graph



Document



AWS Relational Databases

- 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**.

AWS Relational Databases

- With Amazon RDS, you can accelerate your development timelines and establish a consistent operating model for managing relational databases. For example, Amazon RDS makes it easy to replicate your data to increase availability, improve durability, or scale up or beyond a single database instance for read-heavy database workloads.
- With Amazon RDS, you can typically use the same tools to query, analyze, modify, and administer the database.
- The range of DB Instance classes extends from a db.t2.micro with 1 virtual CPU (vCPU) and 1 GB of memory, up to a db.r3.8xlarge with 32 vCPUs and 244 GB of memory. As your needs change over time, you can change the instance class and the balance of compute of memory, and Amazon RDS will migrate your data to a larger or smaller instance class.

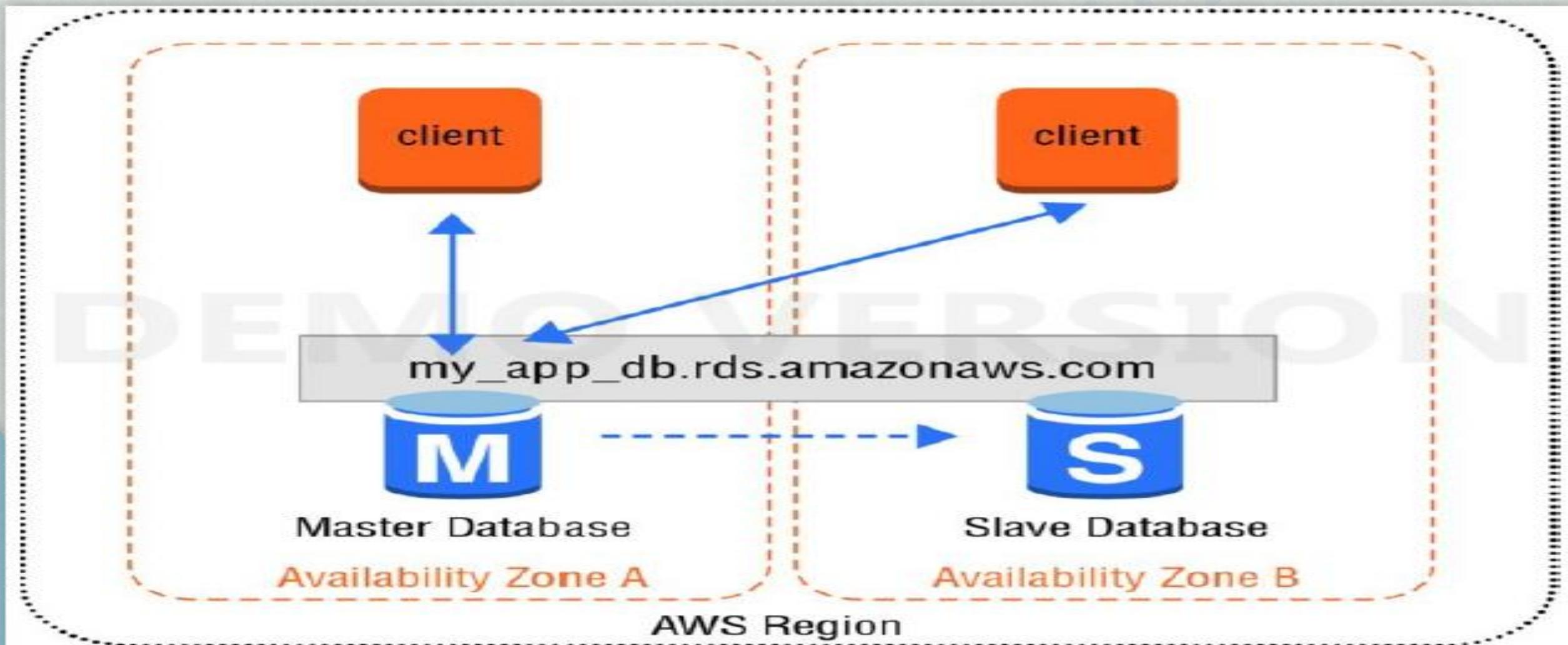
AWS Relational Databases

- You can also choose to run nearly any database engine using Windows or Linux Amazon Elastic Compute Cloud (Amazon EC2) instances and manage the installation and administration yourself.
- As RDS is a managed Service (PaaS), Amazon doesn't provide shell access for RDS DB instances.
- You can have automated backups performed when you need them, or manually create your own backup snapshot.

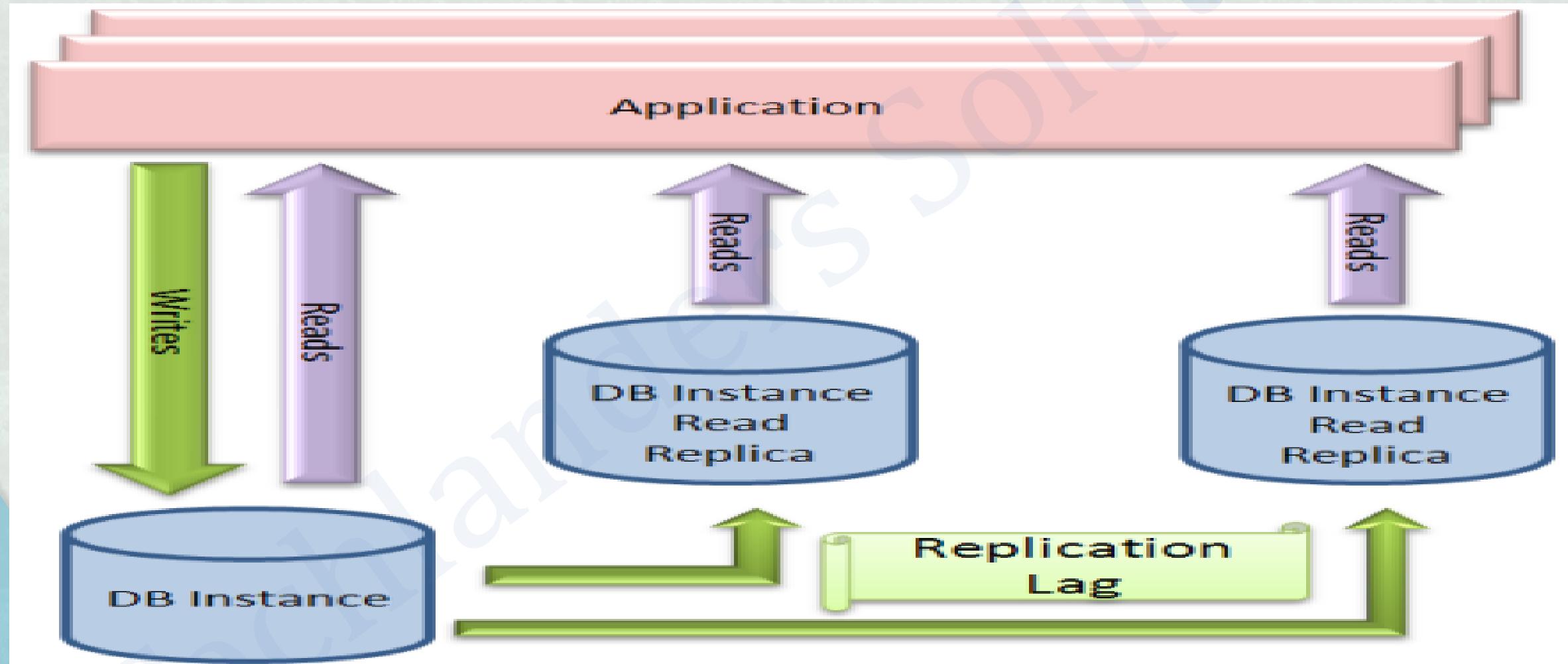
Licensing:

- AWS offers two licensing models: **License Included and Bring Your Own License (BYOL)**.
- License Included In the License Included model, the license is held by AWS and is included in the Amazon RDS instance price.
- Bring Your Own License (BYOL) In the BYOL model, you provide your own license.

AWS RDS HA with multi AZ



AWS RDS Read Replica



LAB 31 : Creating RDS Instance

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 Community Edition instance as the database engine.
3. Configure the DB Instance to use Multi-AZ and General Purpose (SSD) storage.

Warning: This is not eligible for AWS Free Tier; you will incur a small charge by provisioning this instance.

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.

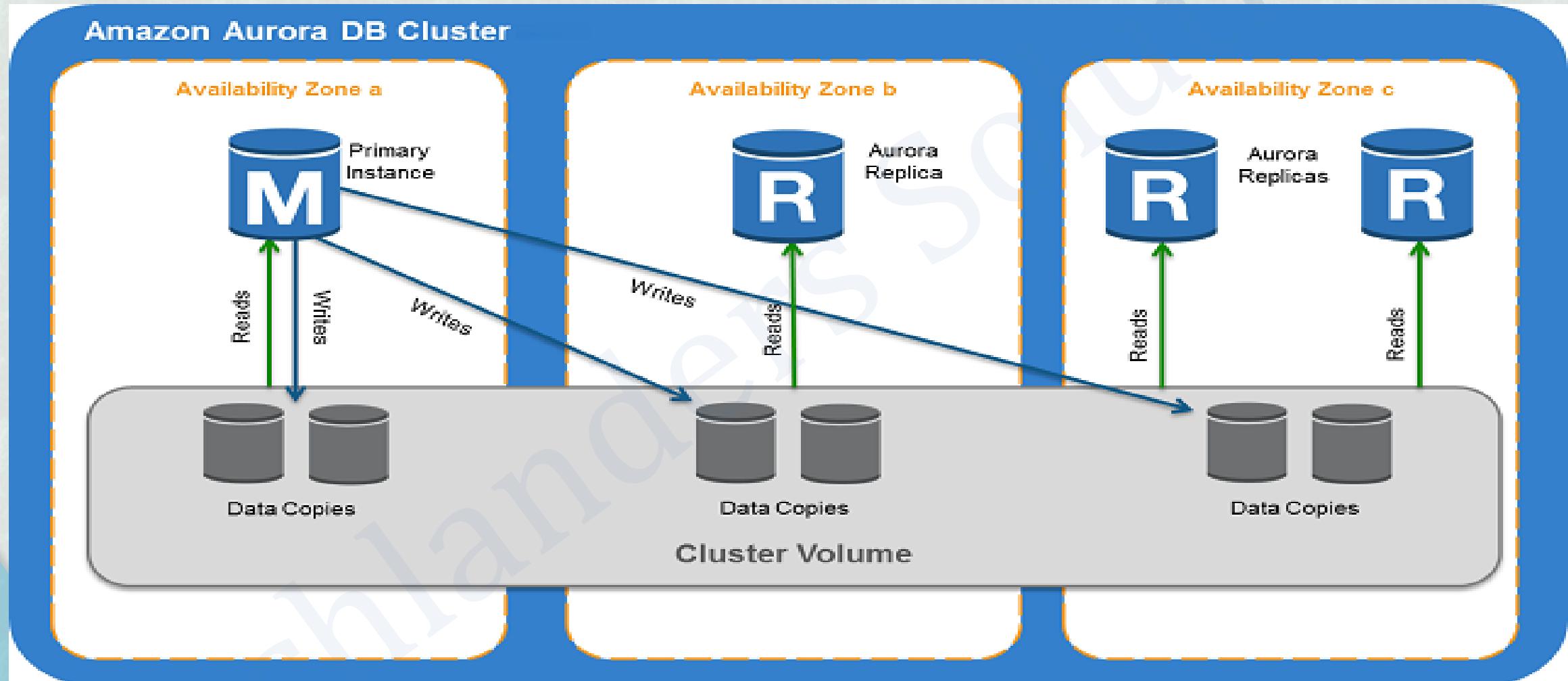
Multi-AZ deployment vs Read Replica

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

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 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.

AWS Aurora DB Cluster





AWS **Redshift**

Amazon Redshift

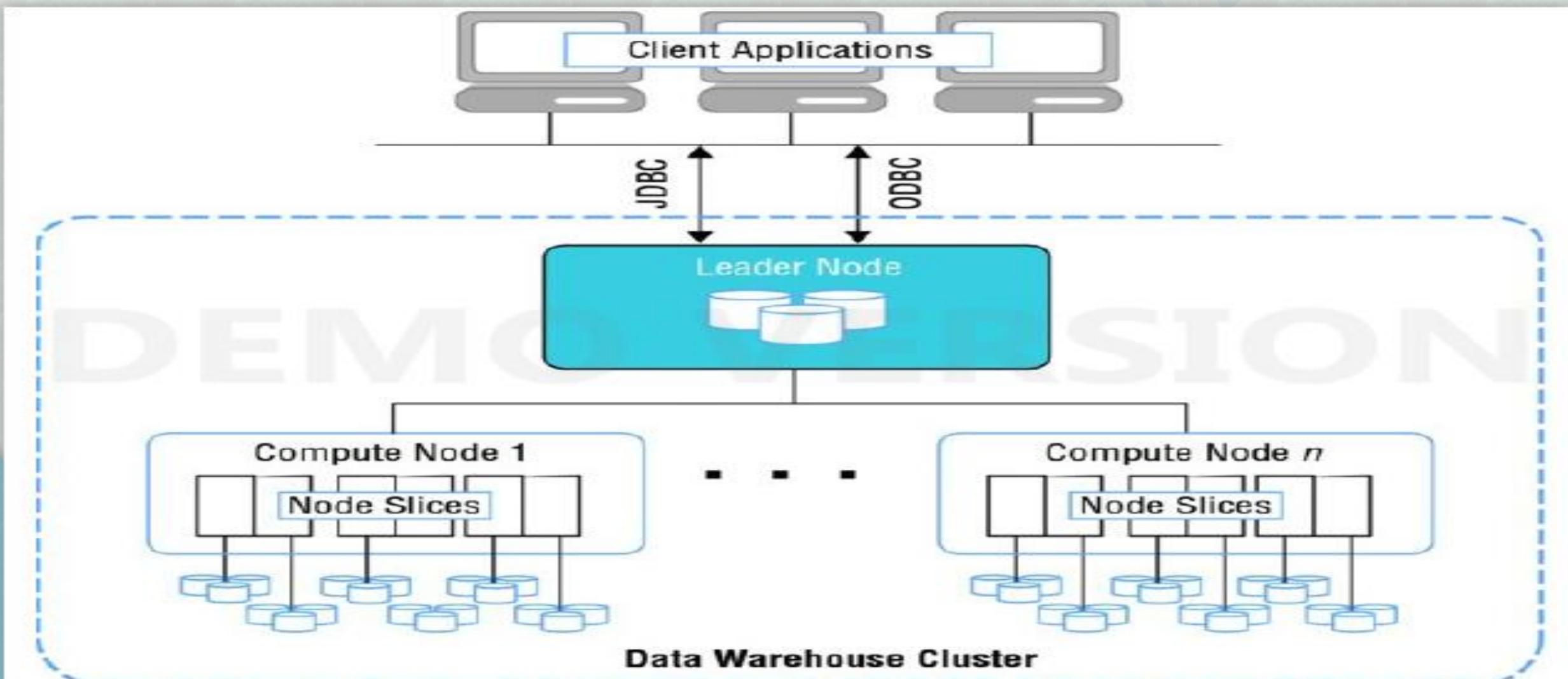
- Amazon Redshift is a fast, powerful, fully managed, petabyte-scale data warehouse service in the cloud.
- Amazon Redshift is a relational database designed for OLAP scenarios and optimized for high-performance analysis and reporting of very large datasets.
- Amazon Redshift gives you fast querying capabilities over structured data using standard SQL commands to support interactive querying over large datasets.
- 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.

Amazon 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.
- Amazon Redshift currently has support for six different node types and each has a different mix of CPU, memory, and storage.
- The six 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.

Amazon Redshift



Amazon Redshift

Amazon Redshift cluster architecture:

- Each cluster contains one or more databases. User data for each table is distributed across the compute nodes. Your application or SQL client communicates with Amazon Redshift using standard JDBC or ODBC connections with the leader node, which in turn coordinates query execution with the compute nodes. Your application does not interact directly with the compute nodes.
- The disk storage for a compute node is divided into a number of slices. The number of slices per node depends on the node size of the cluster and typically varies between 2 and 16. The nodes all participate in parallel query execution.
- Amazon Redshift allows you to resize a cluster to add storage and compute capacity over time as your needs evolve. Whenever you perform a resize operation, Amazon Redshift will create a new cluster and migrate data from the old cluster to the new one. During a resize operation, the database will become read-only until the operation is finished.



AWS DynamoDB

Amazon DynamoDB

- Amazon DynamoDB is a **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, 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.

Amazon DynamoDB

Data Model:

- The basic components of the Amazon DynamoDB data model include tables, items, and attributes.
- A table is a collection of items and each item is a collection of one or more attributes.
- Each item also has a primary key that uniquely identifies the item.
- Each attribute in an item is a name/value pair. An attribute can be a single-valued or multivalued set.
- For example, a book item can have title and authors attributes. Each book has one title but can have many authors.

Amazon DynamoDB

Data Model:

Table	Item	Primary Key	Attribute	
	Item	Primary Key	Attribute	
	Item	Primary Key	Attribute	Attribute
	Item	Primary Key	Attribute	Attribute

LAB 35 : AWS DynamoDB

Read and Write from a DynamoDB Table:

In this exercise, you will create an Amazon DynamoDB table and then read and write to it using the AWS Management Console.

1. Log in to the AWS Management Console, and view the Amazon DynamoDB console.
2. Create a new table named **User Profile** with a partition key of userID of type String.
3. After the table has been created, view the list of items in the table.
4. Using the Amazon DynamoDB console, create and save a new item in the table. Set the userID to U01, and append another String attribute called **name** with a value of **[yourname]**.
5. Perform a scan on the table to retrieve the new item.

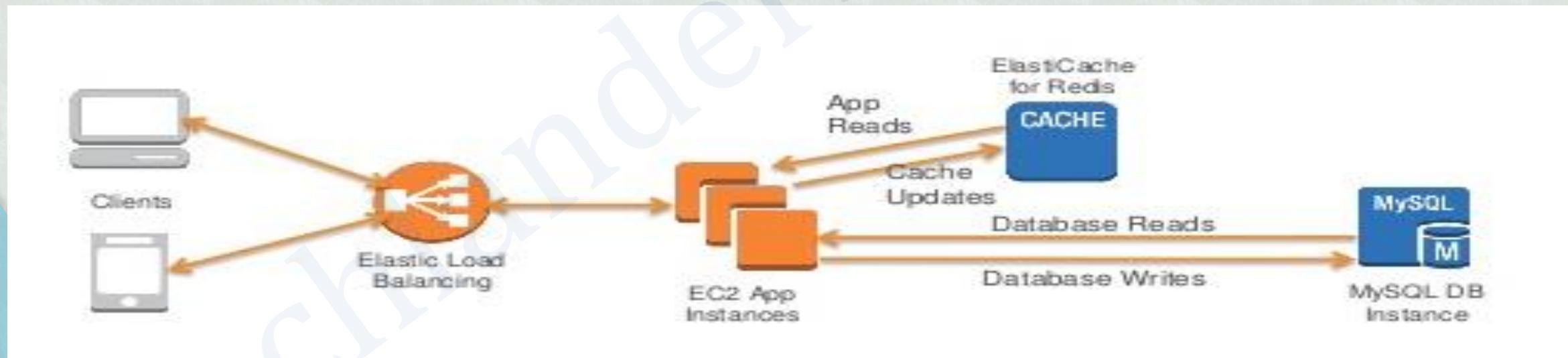
You have now created a simple Amazon DynamoDB table, put a new item, and retrieved it using Scan. Delete the DynamoDB table.



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



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

- An 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.

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

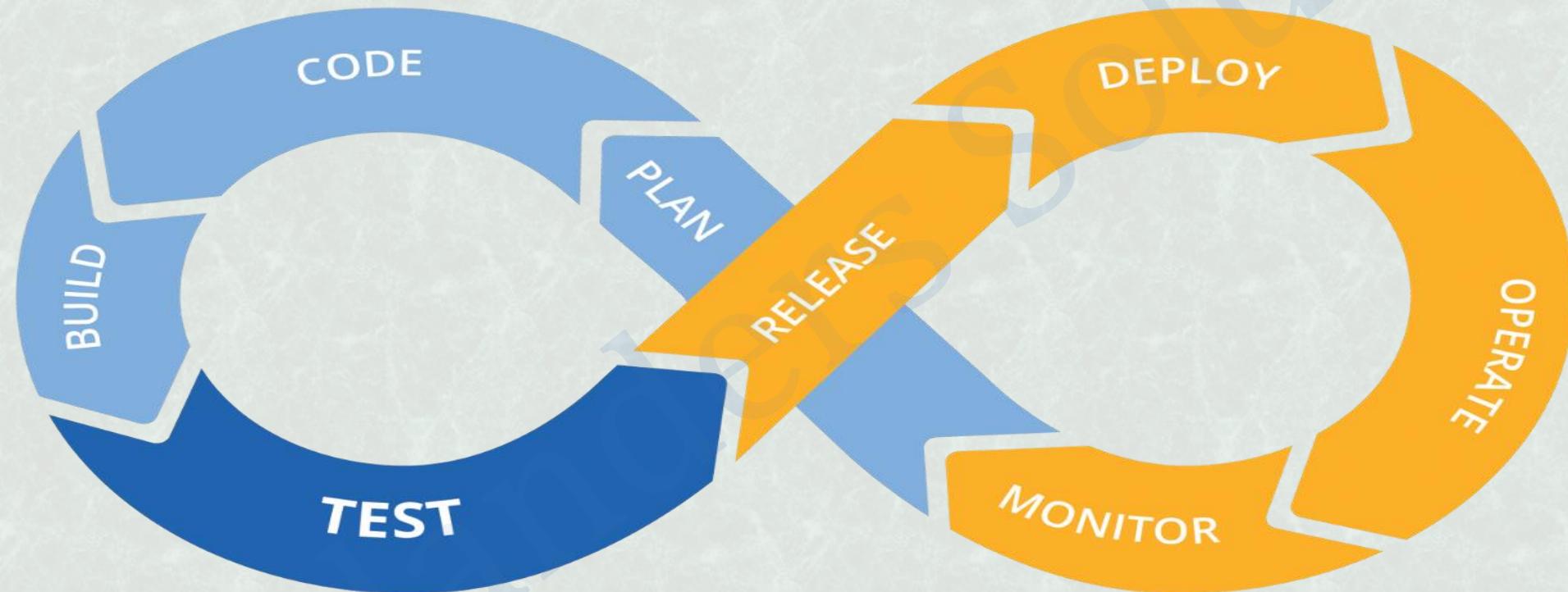
- If you want to utilize your existing licenses, as PaaS don't always offers you BYOL for DB & OS.
- In case you need access to DB libraries or need all DB features
- When OS level access is required
- IaaS is the only choice for legacy and un-supported DB versions.



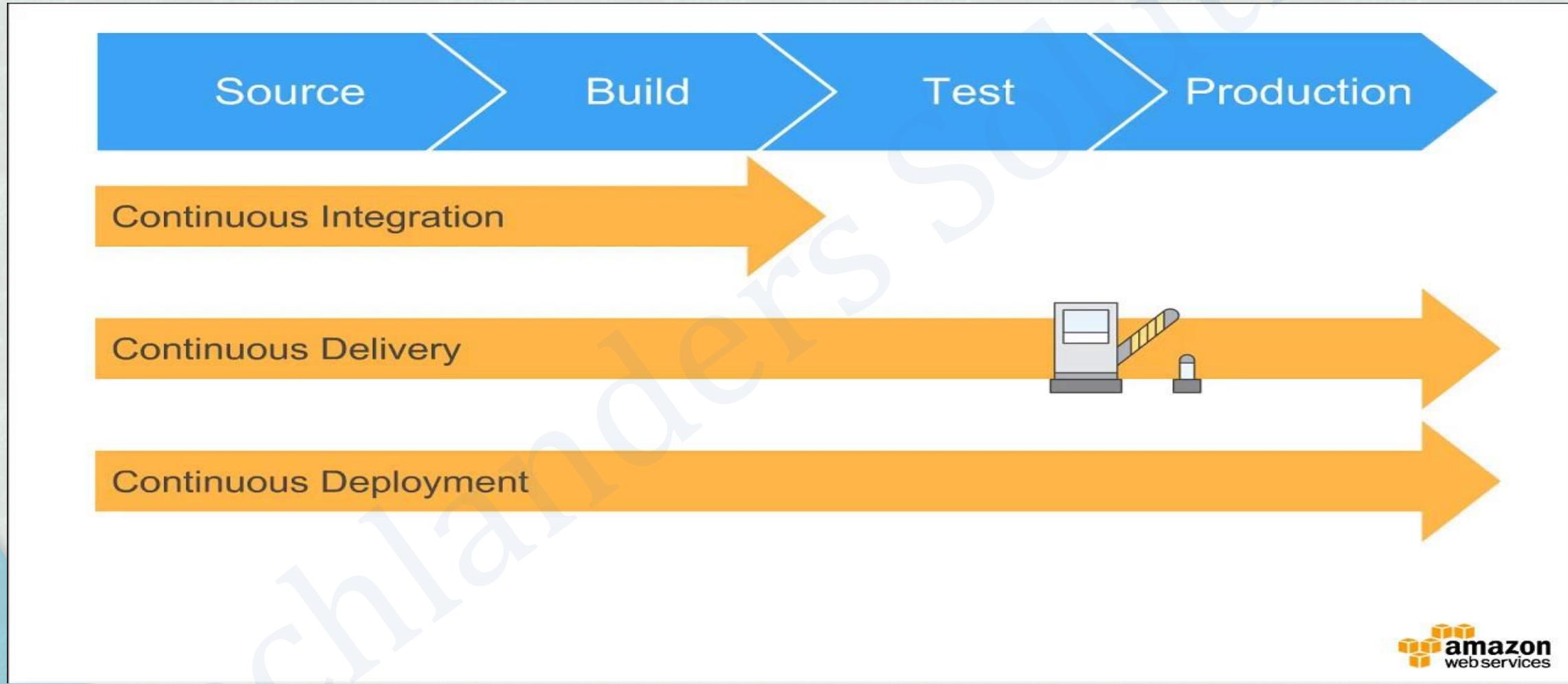
AWS

APP Services

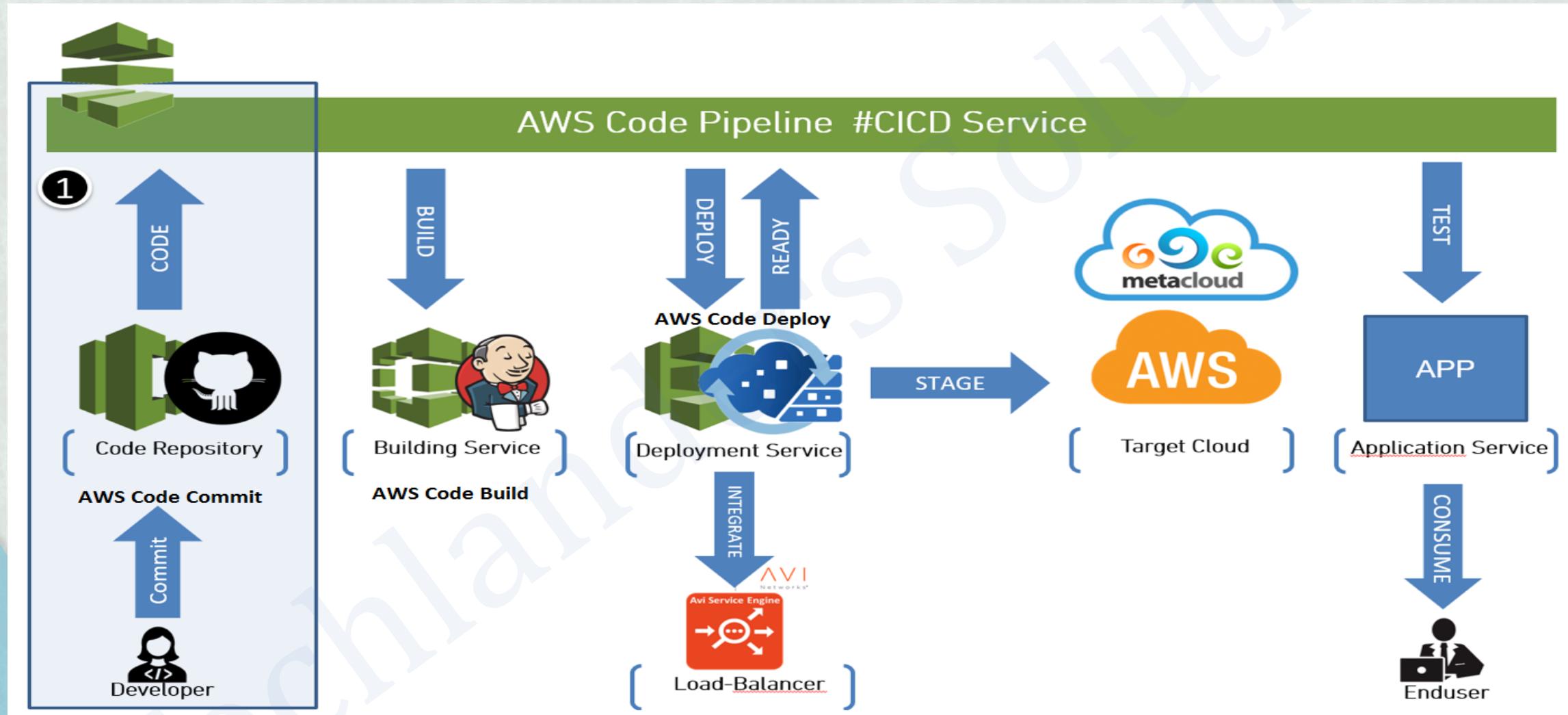
CI/CD pipeline overview



CI/CD pipeline overview



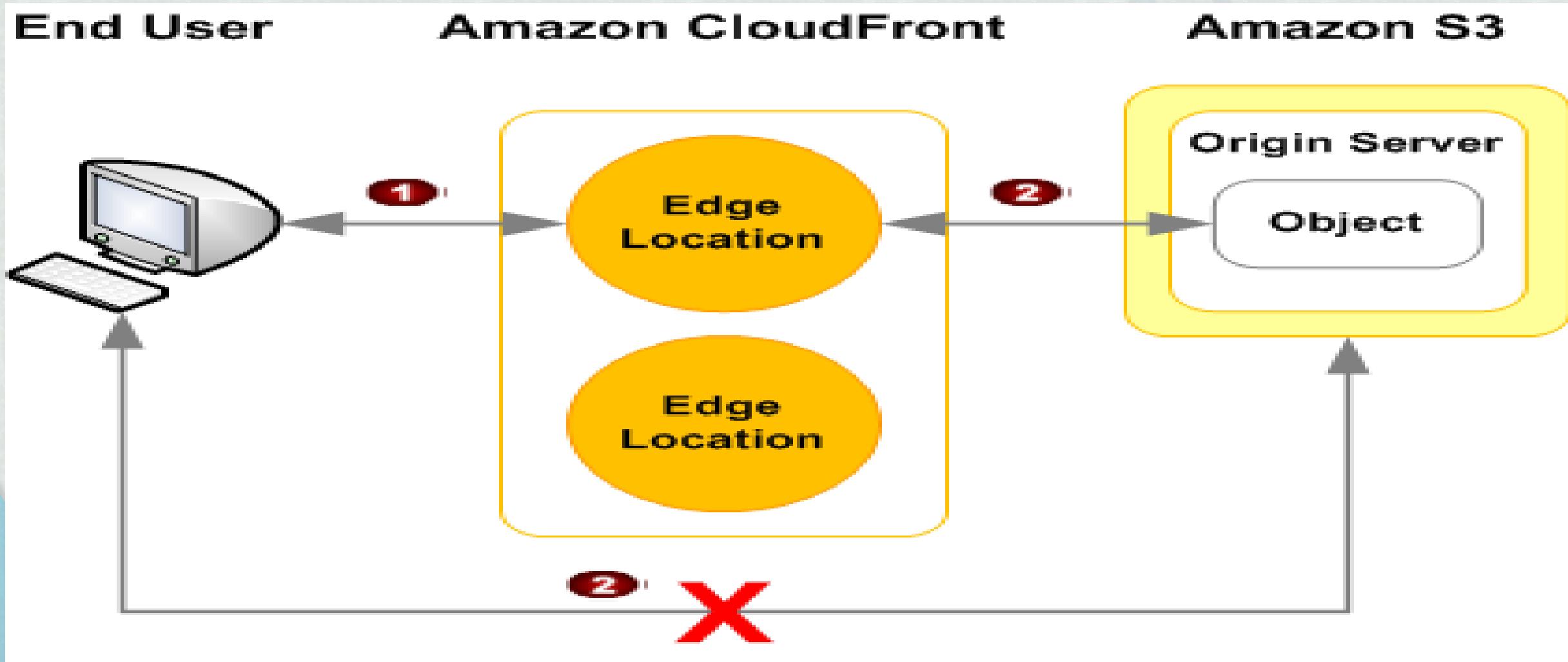
CI/CD pipeline overview



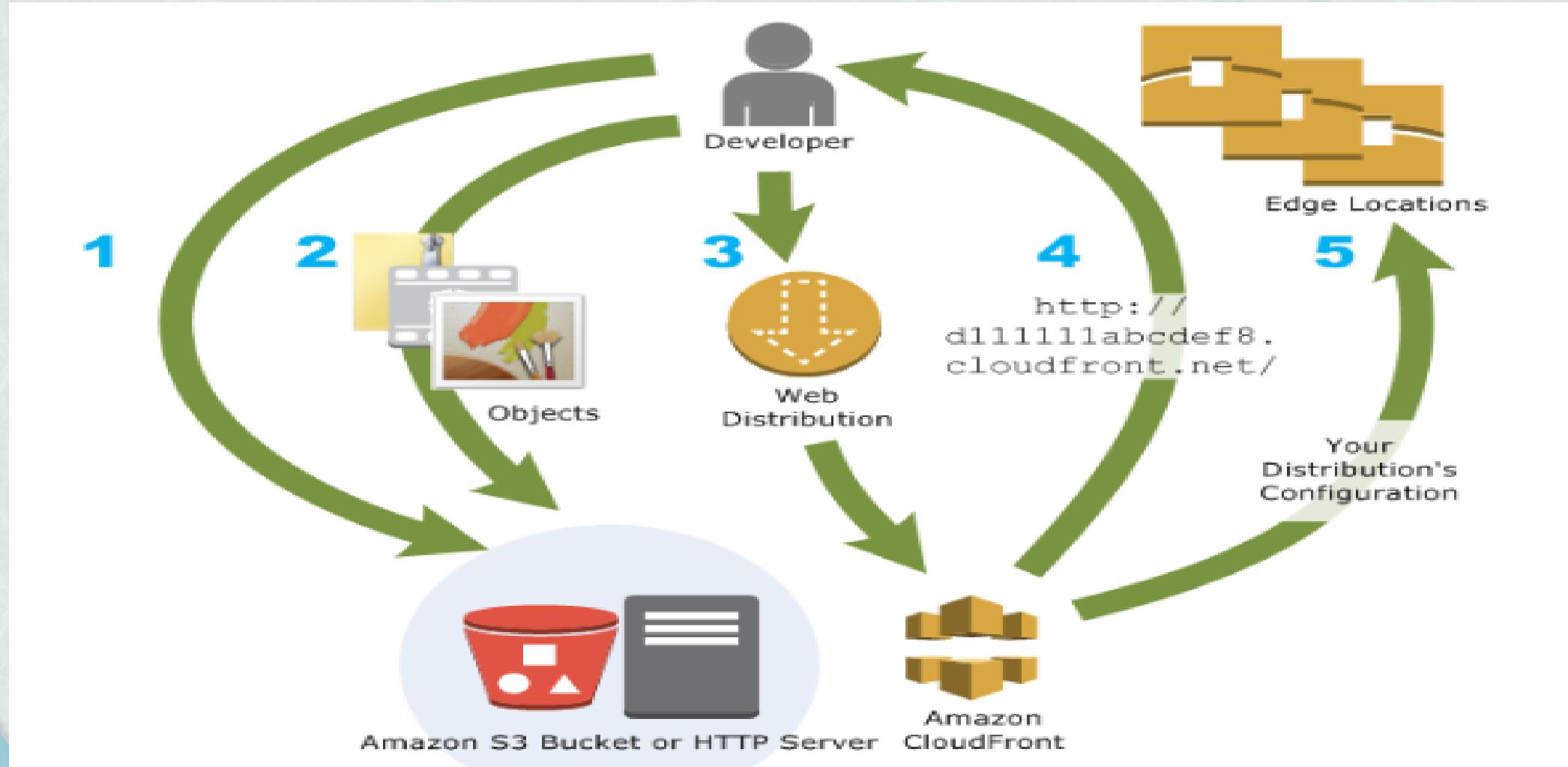
AWS CloudFront

- Amazon CloudFront is a global content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to your viewers with low latency and high transfer speeds.
- It is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users.
- CloudFront delivers your content through a worldwide network of data centers called **edge locations**.
- When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.
- Amazon CloudFront supports multiple protocols (audio and video – both live and on demand) for media streaming

AWS CloudFront



AWS CloudFront

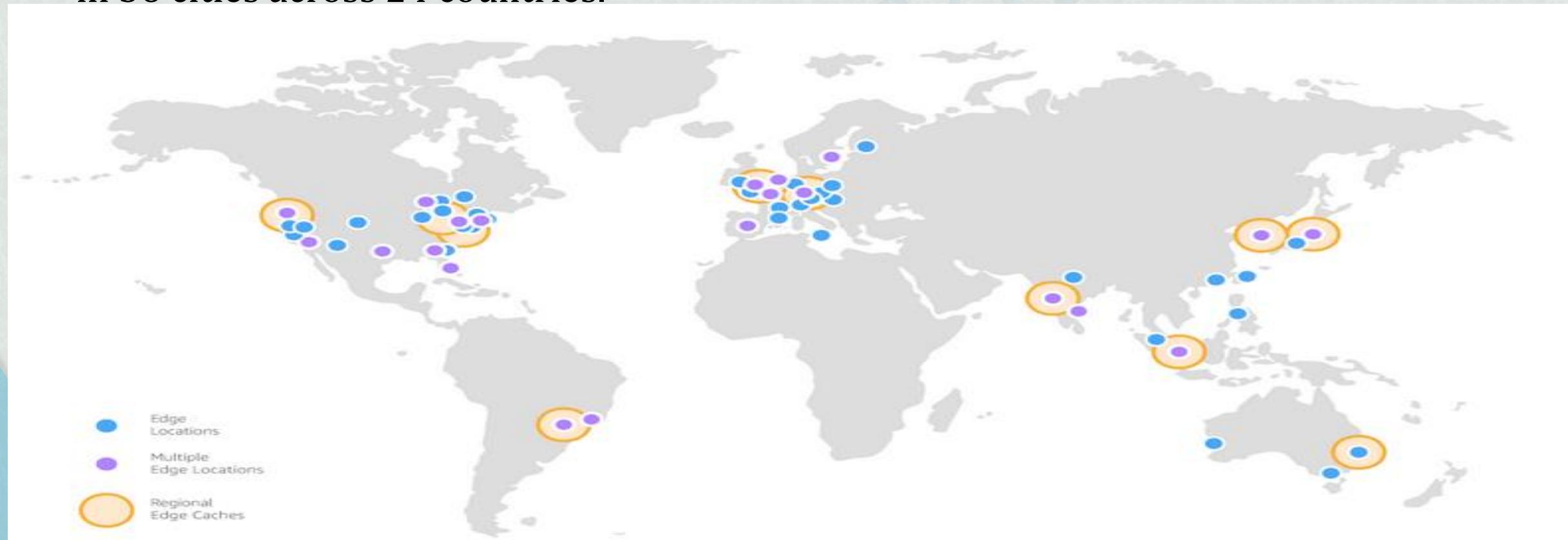


AWS CloudFront

- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.
- If the content is not in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server.
- Next requests will be served from Edge location in above case.
- Users will not directly communicate with the end host/webserver
- Due to caching to your near location, other users of your area will be served faster through already cached data.

AWS CloudFront

- To deliver content to end users with lower latency, Amazon CloudFront uses a global network of 116 Points of Presence (105 Edge Locations and 11 Regional Edge Caches) in 56 cities across 24 countries.



AWS CloudFront

Benefits/Usage:

- Website access is faster for all global/specify users. So requirement of creating dedicated setup in that region is no more needed.
- Website/Web-Servers will be handled centrally. Management of single website is easy then creating multiple clusters/copies in different regions.
- Top referrers, monitoring, cache statics etc. are being generated by CloudFront. Customers can use same to analyze the end-user behaviors and trend.
- RTMP distribution can be used to speed up distribution of your streaming media files using Adobe Flash Media Server's RTMP protocol.

LAB 46 : CloudFront

Create a CloudFront distribution:

In this exercise, you will create an AWS CloudFront distribution

1. Log in to the AWS Management Console and click on CloudFront → “Create Distribution”
2. Select “Get Started”, under Web.
3. Select your bucket name against “Origin Domain Name” and “/index.html” under the path field
4. Go through rest settings and check the help tags to understand the values it support.
5. For first time, you can leave all other fields as it is.
6. Click on create distribution.
7. Wait for sometime and post creation, access the website through CloudFront URL.

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



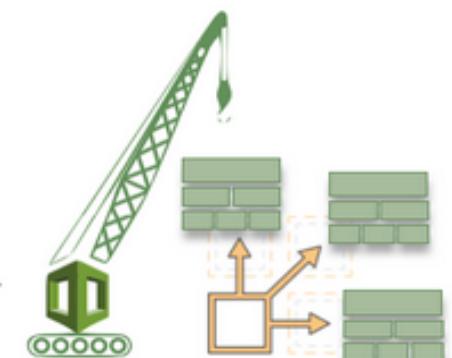
Code your infrastructure from scratch with the CloudFormation template language, in either YAML or JSON format, or start from many available sample templates



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.
- You 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

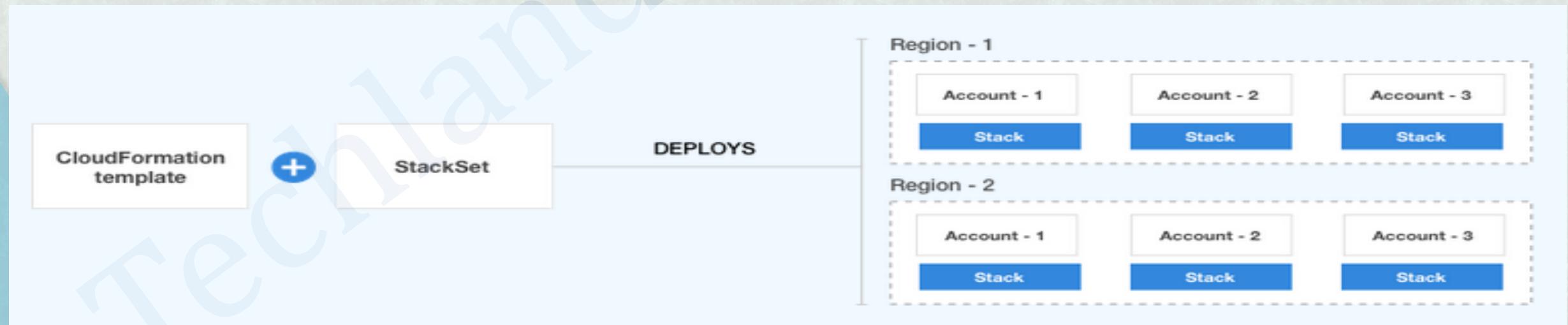
Change Sets:

- 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

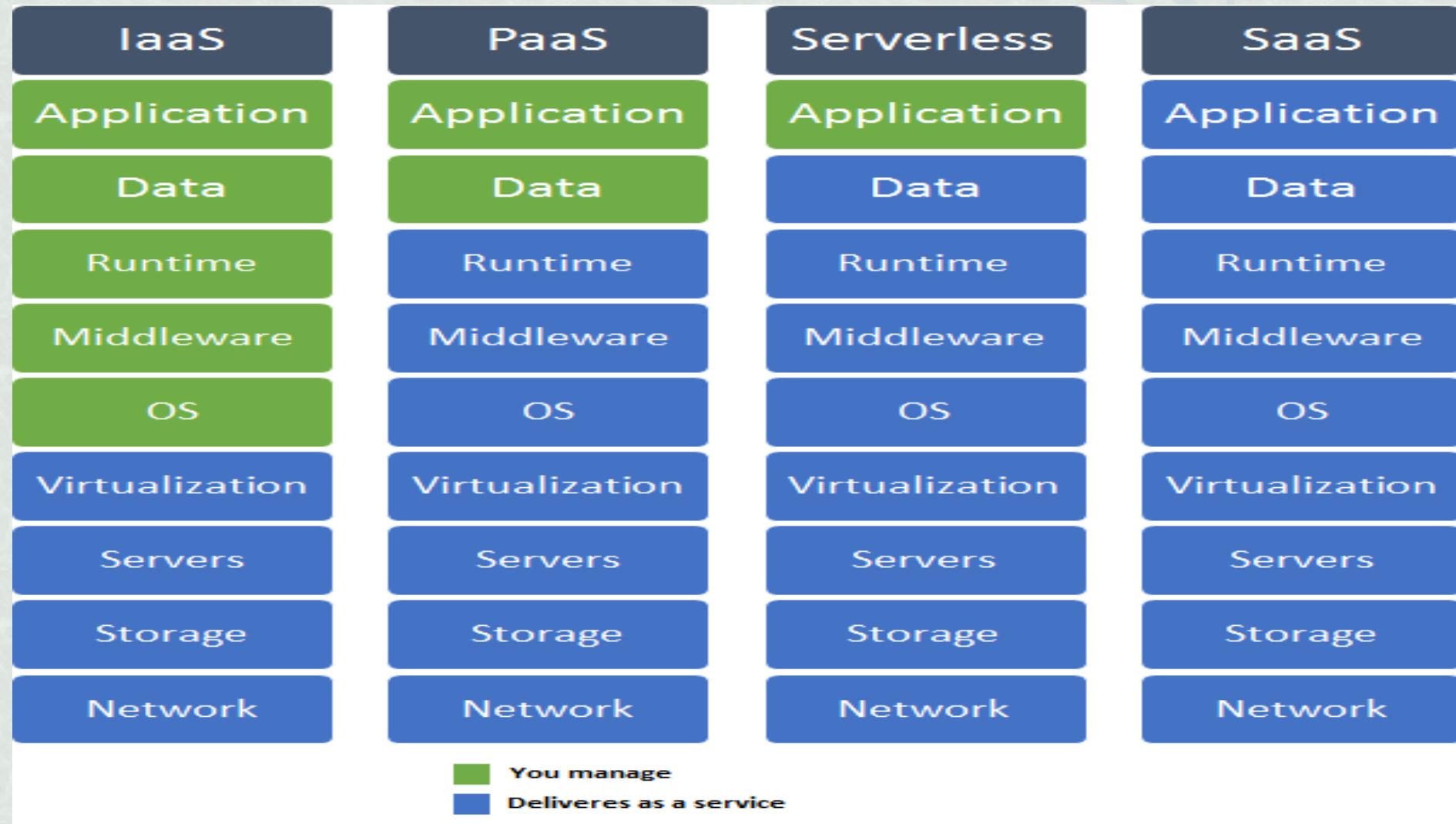
StackSets:

- A StackSet is a container for AWS CloudFormation stacks that lets you provision stacks across **AWS accounts and regions** by using a single AWS CloudFormation template.
- Using an administrator account, you define and manage an AWS CloudFormation template, and use the template as the basis for provisioning stacks into selected target accounts across specified regions.
- A stack set is a regional resource. If you create a stack set in one region, you cannot see it or change it in other regions.



AWS Serverless Platform

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

AWS Serverless Platform

Serverless Applications:

Compute → AWS Lambda

API Proxy → API gateway

Storage → Amazon S3

Data Stores → DynamoDB

Messaging → SQS, SNS

Orchestration → AWS Step Functions

Analytics → Kinesis, Athena

Devops Tools → CI/CD pipeline

ServerLess Computing - 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

Serverless Computing - 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.

LAB 47 : First LAMBDA function

- 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

First LAMBDA function

Code entry type

Edit code inline

```
1 'use strict';
2
3 console.log('Loading function');
4
5 + exports.handler = (event, context, callback) => {
6     //console.log('Received event:', JSON.stringify(event, null, 2));
7     console.log('value1 =', event.key1);
8     console.log('value2 =', event.key2);
9     console.log('value3 =', event.key3);
10    callback(null, event.key1); // Echo back the first key value
11    //callback('Something went wrong');
12 }
```

Execution result: succeeded (logs)

The area below shows the result returned by your function execution. [Learn more about returning results from your function.](#)

```
"hello"
```

First LAMBDA function

- Execution Result
- Summary
- Log output

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 48 : LAMBDA function

Random Number Generator

```
'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 48 : LAMBDA function

Random Number Generator

Code entry type

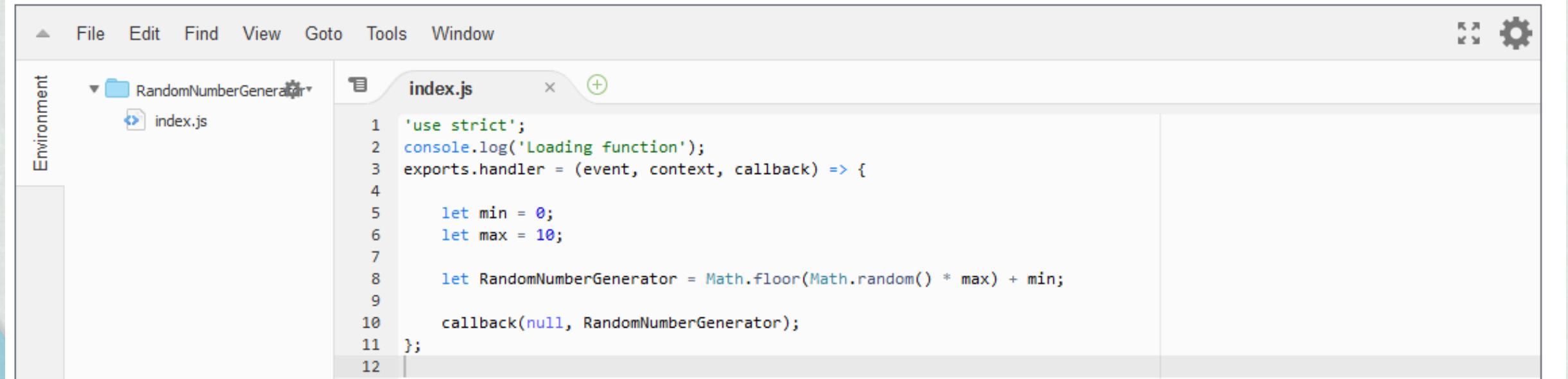
Runtime

Handler Info

Edit code inline ▾

Node.js 6.10 ▾

index.handler



The screenshot shows a code editor interface with the following details:

- Code entry type:** Edit code inline
- Runtime:** Node.js 6.10
- Handler Info:** index.handler

The code editor displays the `index.js` file under the `RandomNumberGenerator` folder. The code is as follows:

```
'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);
};
```

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

3



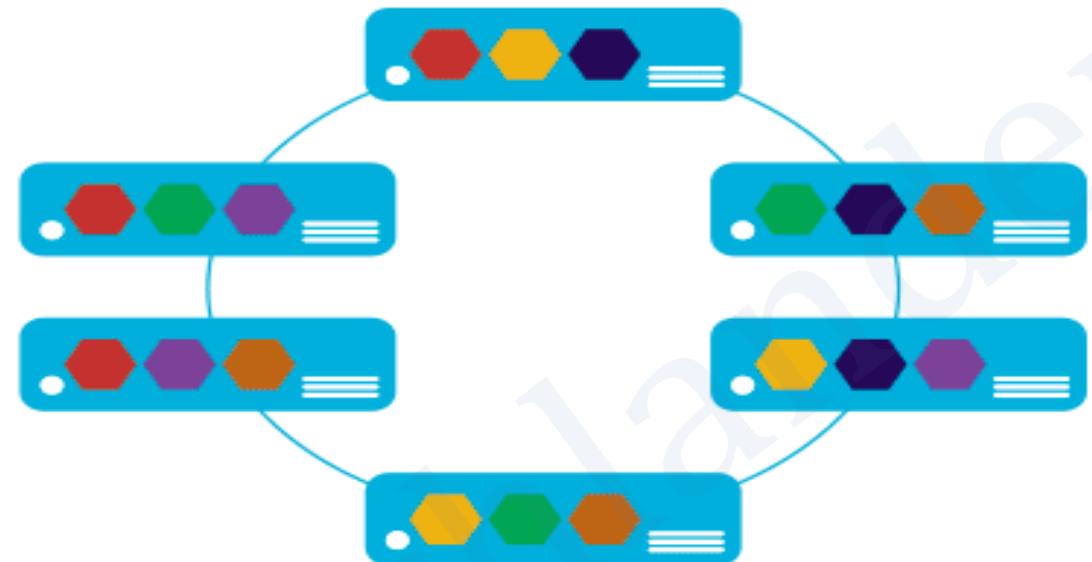
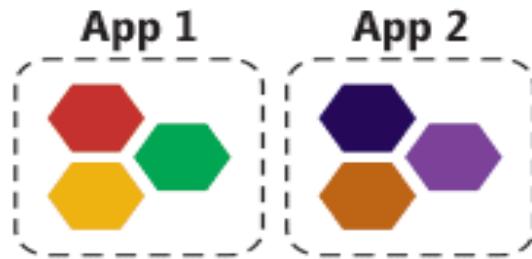
Containers

Micro Services vs Monolithic

Microservices Approach

A microservice approach segregates functionality into small autonomous services.

And scales out by **deploying independently** and replicating these services across servers/VMs/containers.

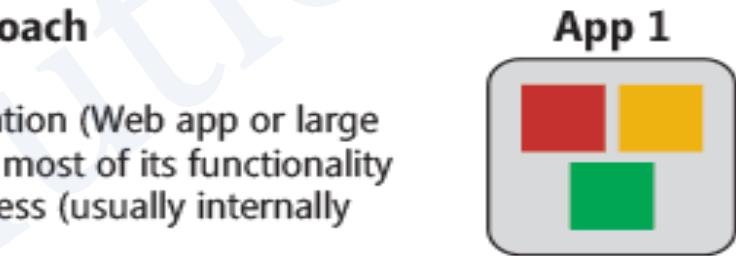


VS.

Traditional Approach

A traditional application (Web app or large service) usually has most of its functionality within a single process (usually internally layered, though).

And scales by cloning the whole app on multiple servers/VMs/containers.



AWS Elastic Container 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.

Containers Facts

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

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*.

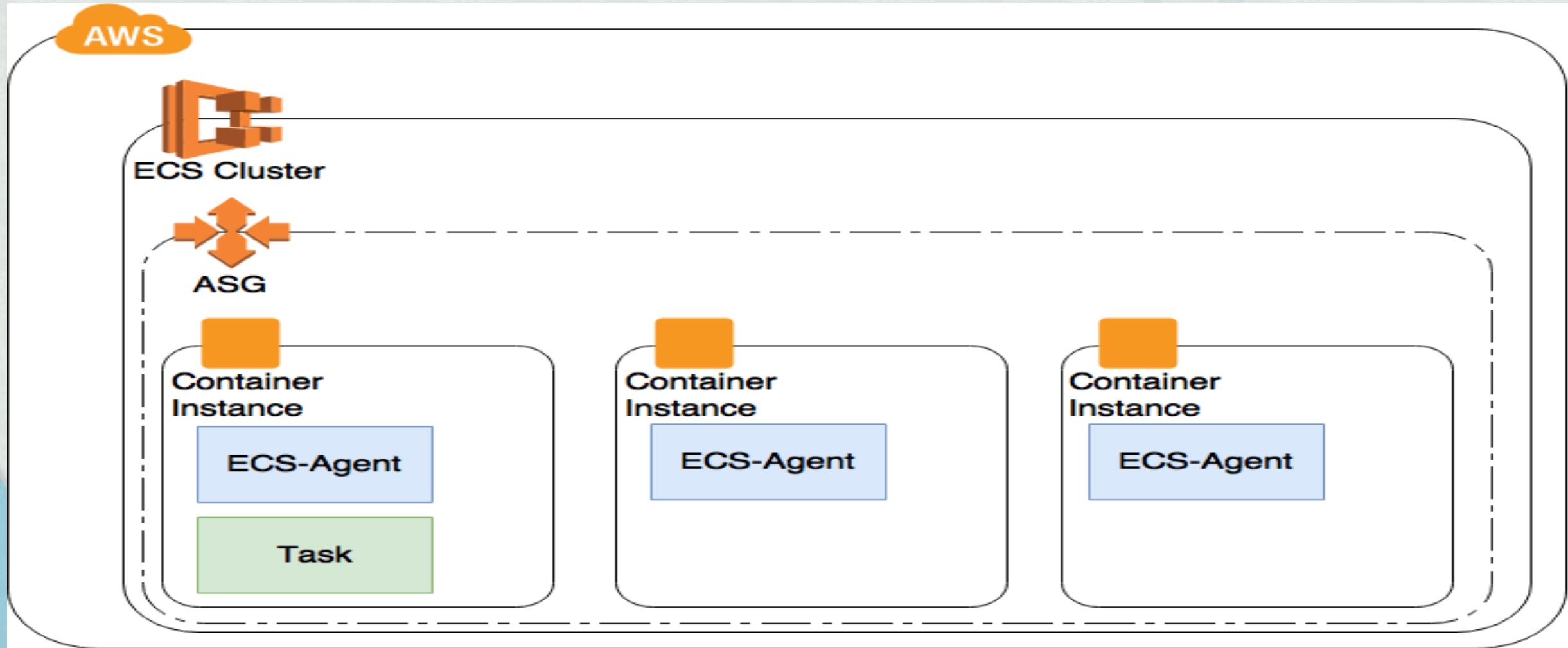
Images are typically built from a Dockerfile, a plain text file that specifies all of the components that are included in the container.

AWS Elastic Container Service

- **Facts**

- 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 Kubernetes)
- Cluster instances are EC2 in backend.
- ECS is using Autoscaling in Backend (ELB can be configured)

AWS Elastic Container Service



ECS Facts

Containers

- Containers are Light weighted VMs with shared Kernel. Container definition includes, Image name, Port mappings, health checks, Env etc.

Task Definitions

- A *task* is the instantiation of a task definition within a cluster. To prepare your application to run on Amazon ECS, you create a *task definition*. The task definition is a text file, in JSON format, that describes executions roles, one or more containers, up to a maximum of ten, that form your application. It can be thought of as a blueprint for your application.

Services

- Amazon ECS allows you to run and maintain a specified number (the "desired count") of instances of a task definition simultaneously in an Amazon ECS cluster. This is called a service. If any of your tasks should fail or stop for any reason, the Amazon ECS service scheduler launches another instance of your task definition to replace it and maintain the desired count of tasks in the service.

ECS Facts

Cluster

- Logical grouping of resources/Backend instances, which run your Blox cluster behind AS and LB.
- If you use the Fargate launch type with tasks within your cluster, Amazon ECS manages your cluster resources. If you use the EC2 launch type, then your clusters will be a group of container instances you manage.

Container Agent

- The *container agent* runs on each infrastructure resource within an Amazon ECS cluster. It sends information about the resource's current running tasks and resource utilization to Amazon ECS, and starts and stops tasks whenever it receives a request from Amazon ECS.

Amazon Elastic Container Registry

- Amazon ECR is a managed AWS Docker registry service that is secure, scalable, and reliable. Amazon ECR supports private Docker repositories with resource-based permissions using IAM so that specific users or EC2 instances can access repositories and images.

ECR (Elastic Container Registry)

Amazon Elastic Container Registry (ECR) is a fully-managed container registry that makes it easy for developers to store, manage, and deploy container images

- Fully managed
- Highly available
- Secure
- integrates with Amazon ECS and the Docker CLI,

LAB 49 : ECS

Create a ECS Cluster:

In this exercise, you will create an AWS ECS Cluster:

1. Log in to the AWS Management Console and click on ECS → “Create Cluster”
2. Select “EC2 Linux + Networking” under cluster template
3. Select **on demand instances**,
instance type **T2-micro**,
number of instance -**2**,
keypair,
VPC and subnets
4. Click create.
5. Your ECS cluster is ready in few minutes

LAB 50 : ECS

Create a ECS task definition:

In this exercise, you will create an Task Definition:

1. Log in to the AWS Management Console and click on ECS → “Create task definition”
2. Provide “Task Definition Name” under cluster template
3. Configure task role,
 - Network type Bridge,
 - Memory 300, cpu 1 vcpu,
4. Add container → Provide container details
5. Click create.
6. Your task definition is ready

LAB 58 : ECS

Check containers:

In this exercise, you will create ECS Services

- Login to Servers and check your containers, running on it docker version
 - Docker ps
 - docker pull centos
 - docker images
 - Docker stop [instance-id]
 - docker exec -it edbd7c16d6ff bash
 - Ctrl + p +q //to exit from container
- Observe that post stopping the container, your cluster service is starting same again.

AWS Fargate

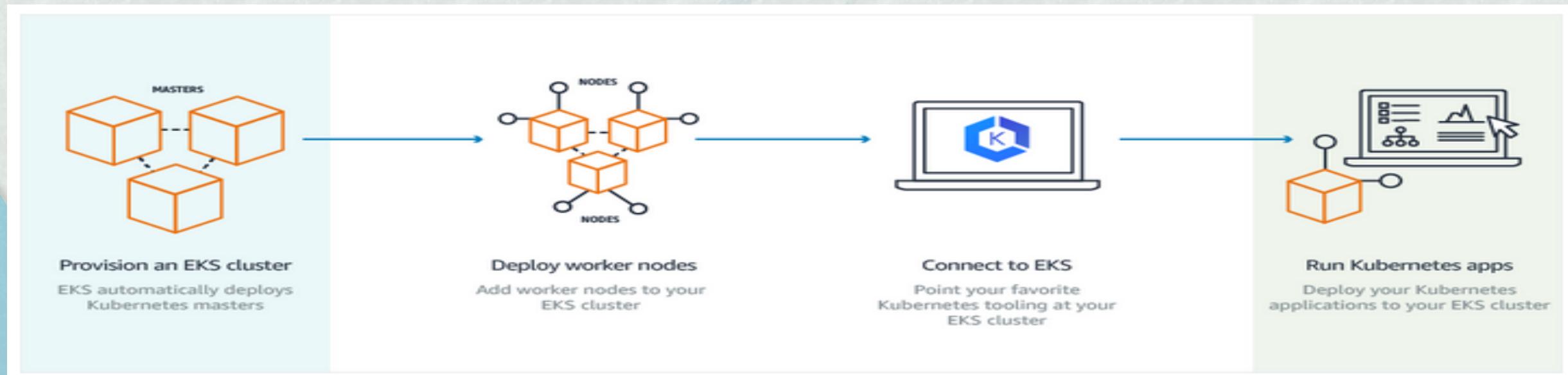
- AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of EC2 instances.
- When you run your tasks and services with the Fargate launch type, you package your application in containers, specify the CPU and memory requirements, define 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.
- Currently available in us-east-1, us-east-2, eu-west-1, us-west-2 Regions.

AWS ECS for Kubernetes

- 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 ECS for Kubernetes

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





Common Practical Approaches

Development and test scenarios

Flexibility for Innovation

Load testing and Live Site swapping

Managed Services

Hybrid scenarios

Burst to the cloud

Latest technology within minutes

AWS as DR

Questions & Answers



THANK YOU

<http://blog.Techlanders.com/>

For any queries or questions, please contact:

support@Techlanders.com

raman.khanna@Techlanders.com