AWS for Solution Architects

Techlanders Solutions

Raman Khanna

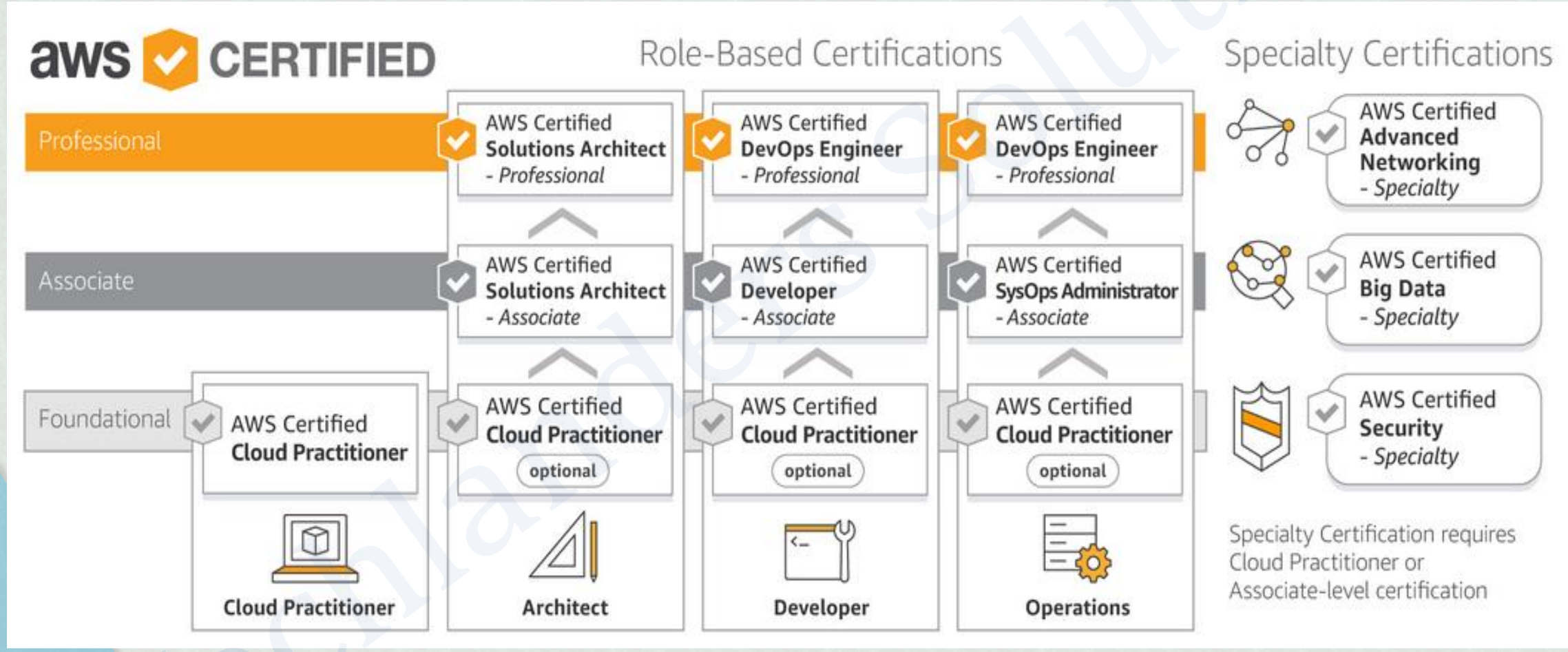
Introduction

Your Name

Background – Development / Infrastructure

Learnings on cloud/AWS

AWS Certification Roadmap



Introduction to Cloud Computing

What is Cloud?

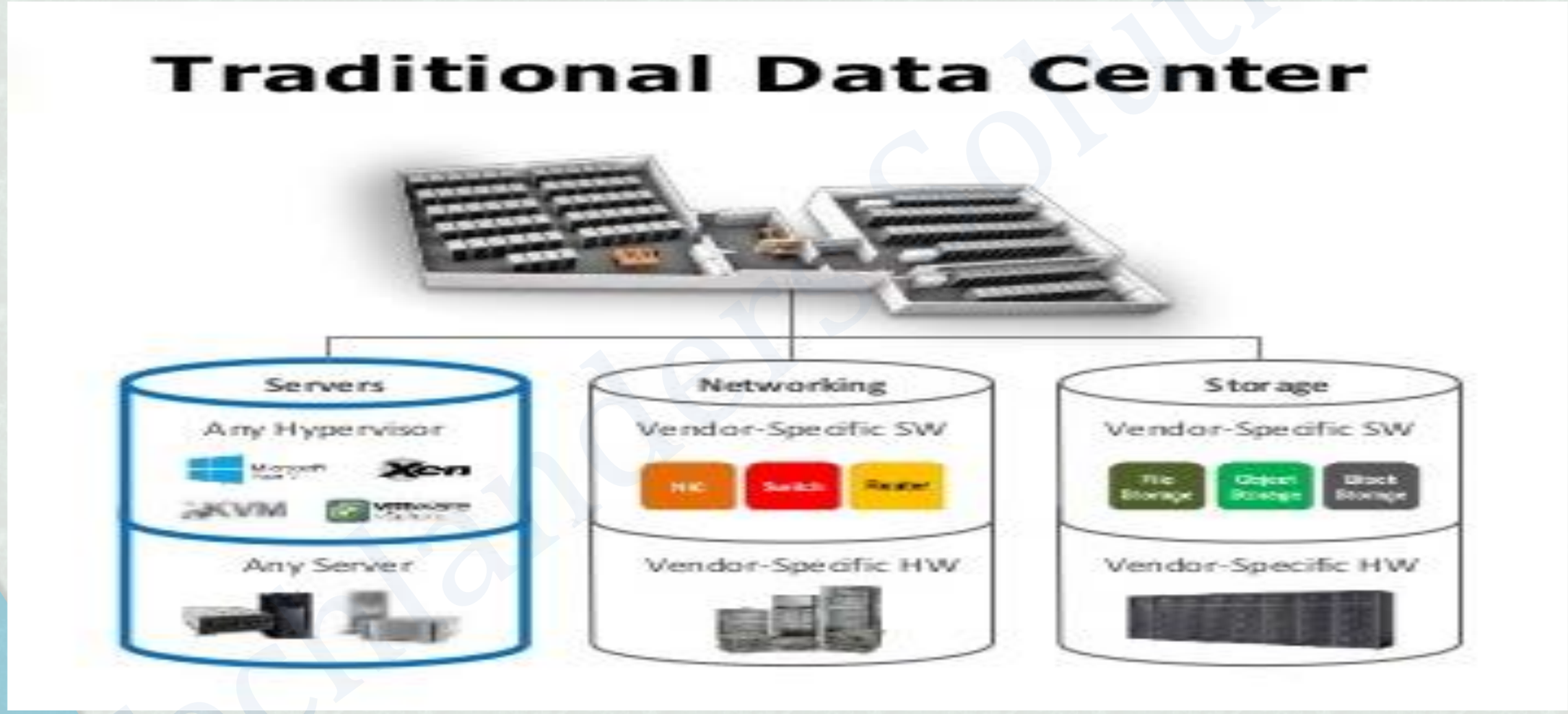
Introduction to Cloud Computing

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 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”

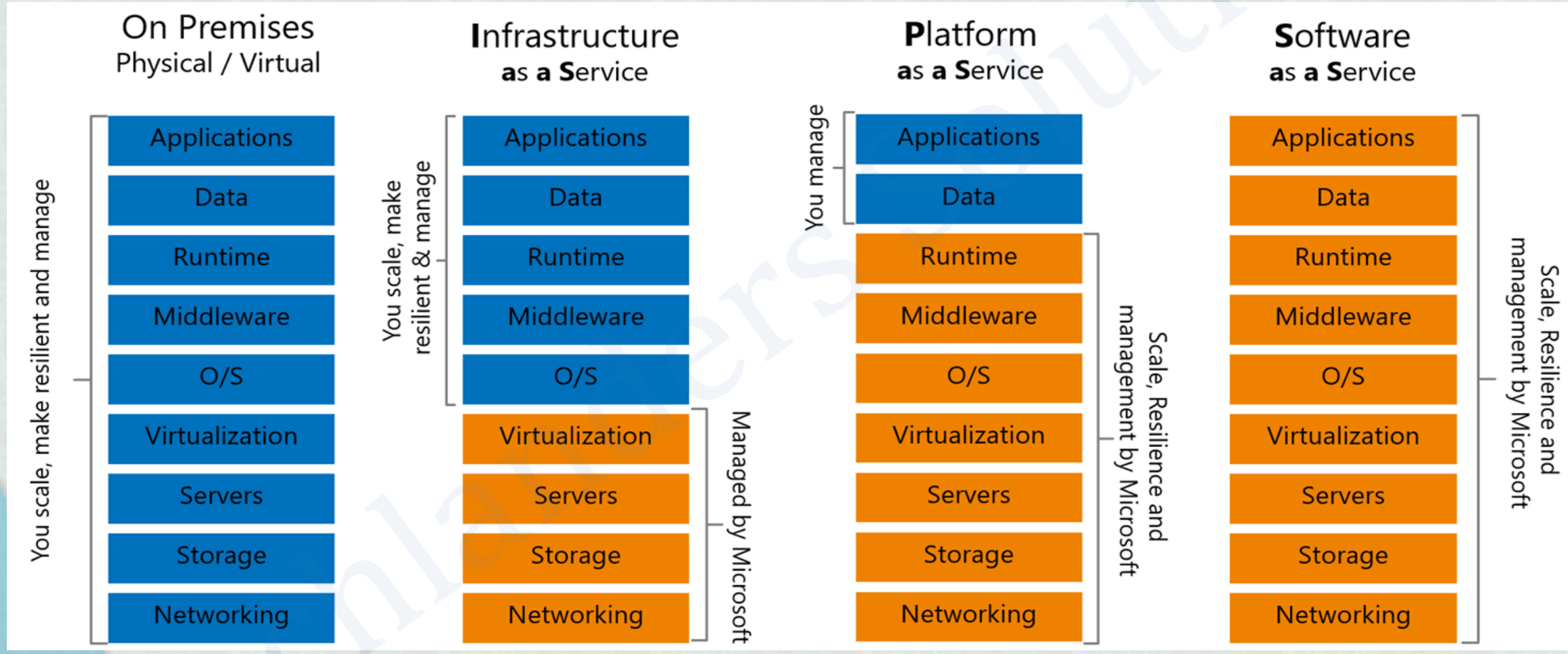
- No Dedicated Datacenter
- No Different Infrastructure Teams
- Higher/Faster Scalability
- Elasticity
- Pay per use model
- Option to adopt high availability
- Better performance
- Instant provisioning
- Optimized use of resources
- On demand scaling to any extent
- No to worry about capacity planning

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.

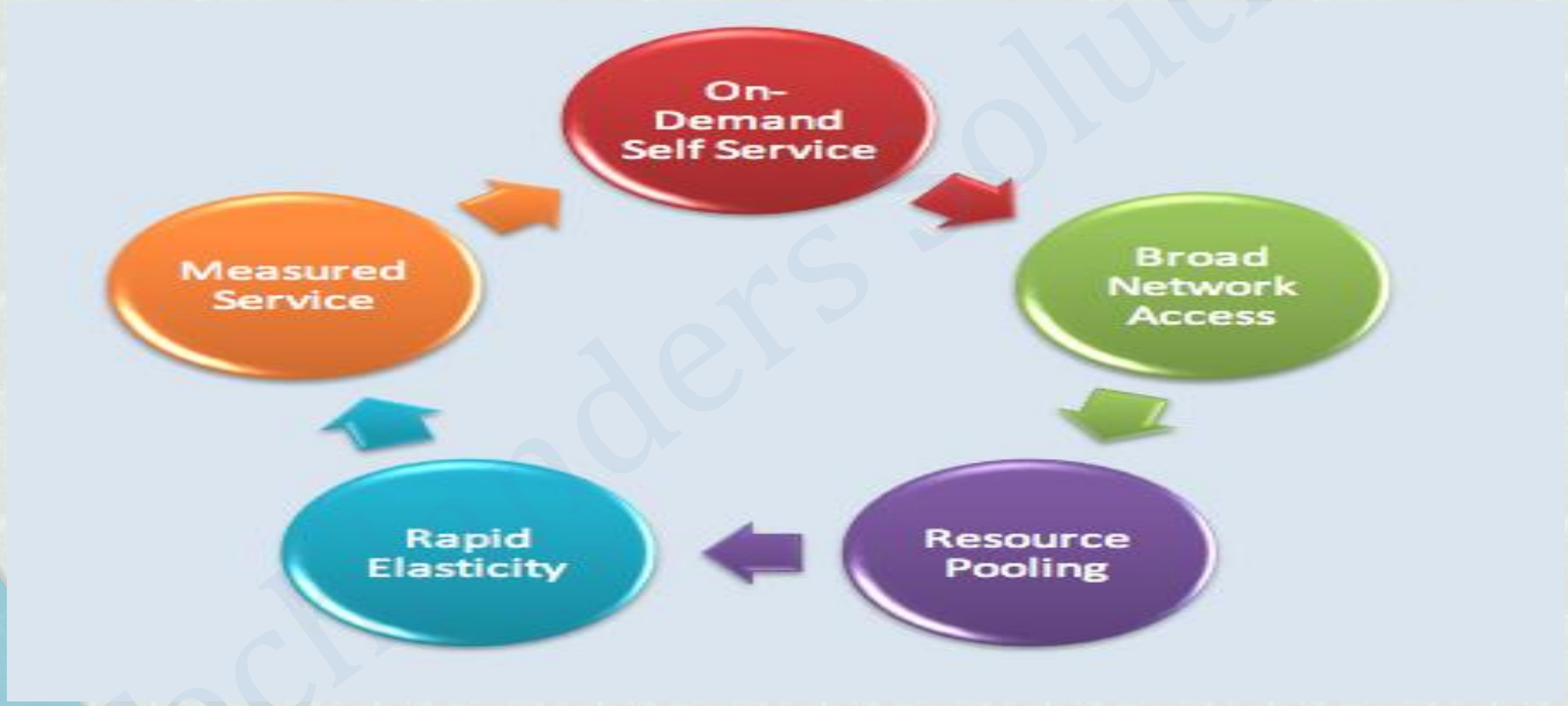
SaaS application are delivered as “one-to-many” model.

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

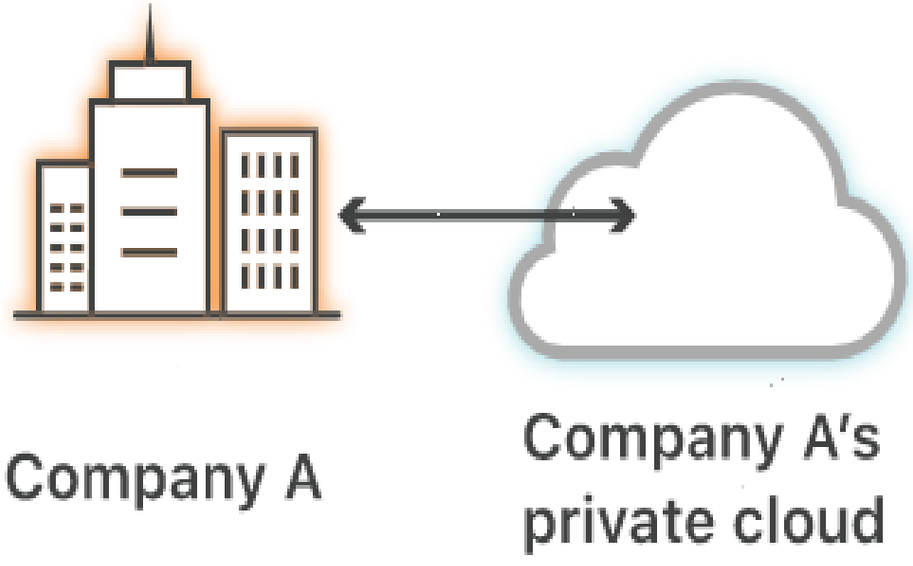
SaaS - Benefits

- 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

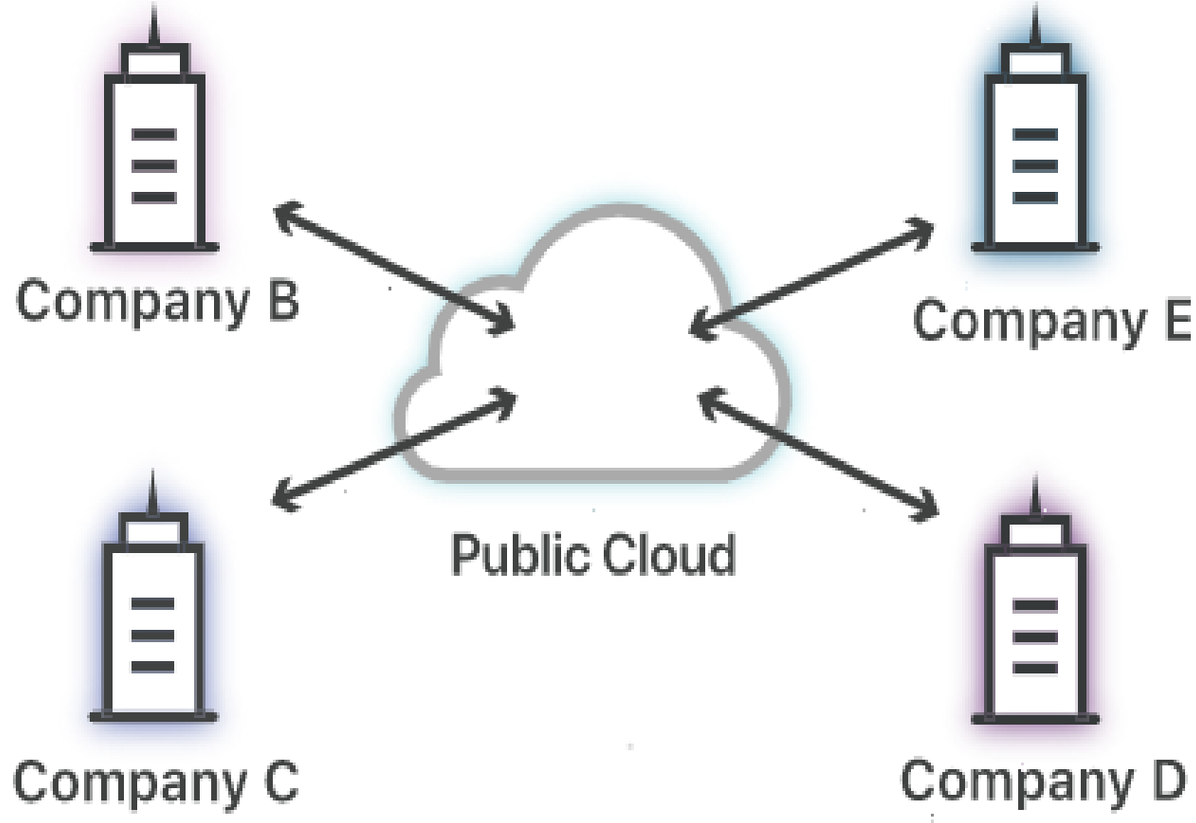
Cloud Essentials Characteristics



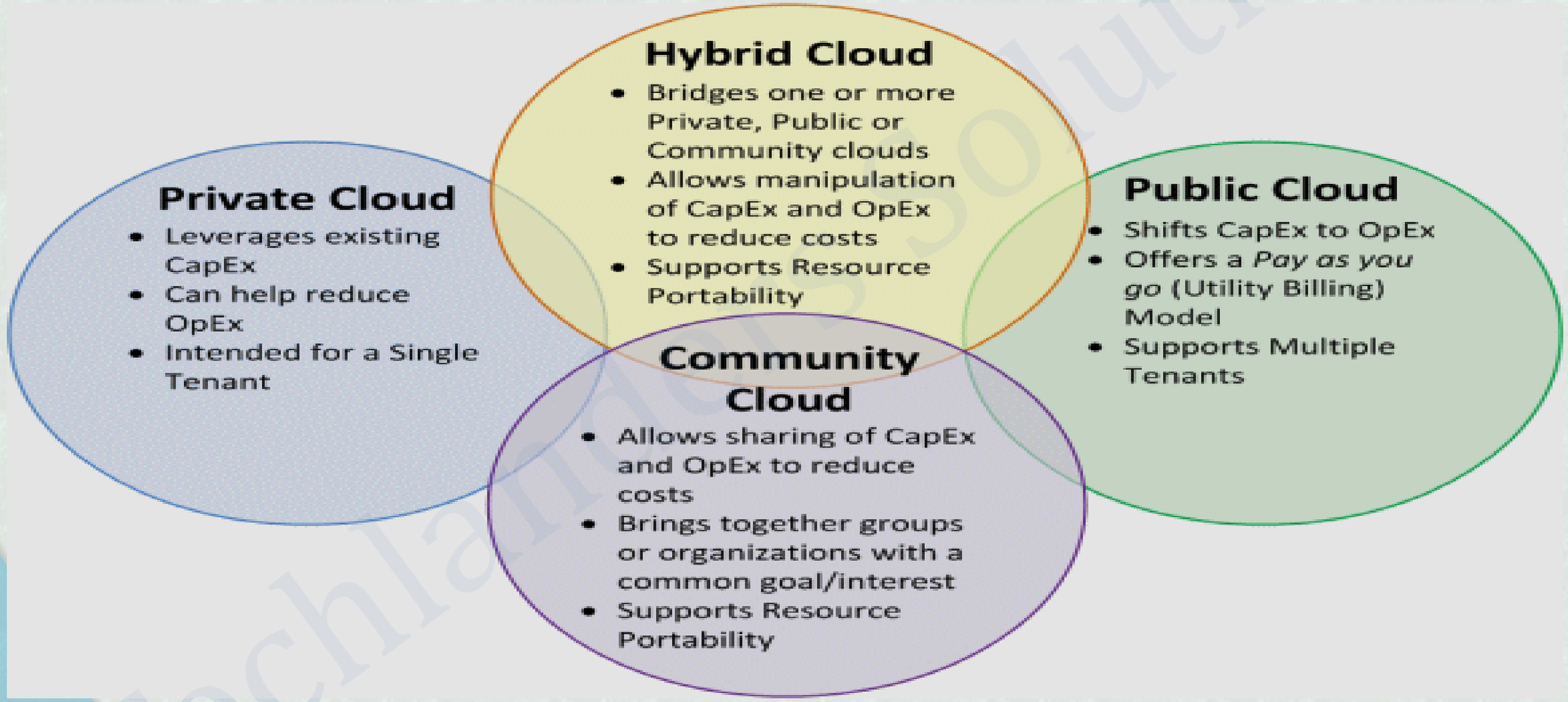
Private cloud



Public cloud shared by multiple companies



Cloud Deployments Types



Cloud's Major Use Cases

On and Off

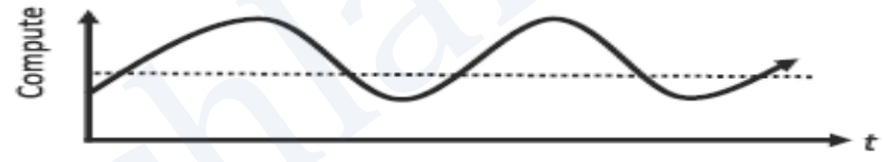
On and off workloads (e.g. batch job)
Over provisioned capacity is wasted
Time to market can be cumbersome

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

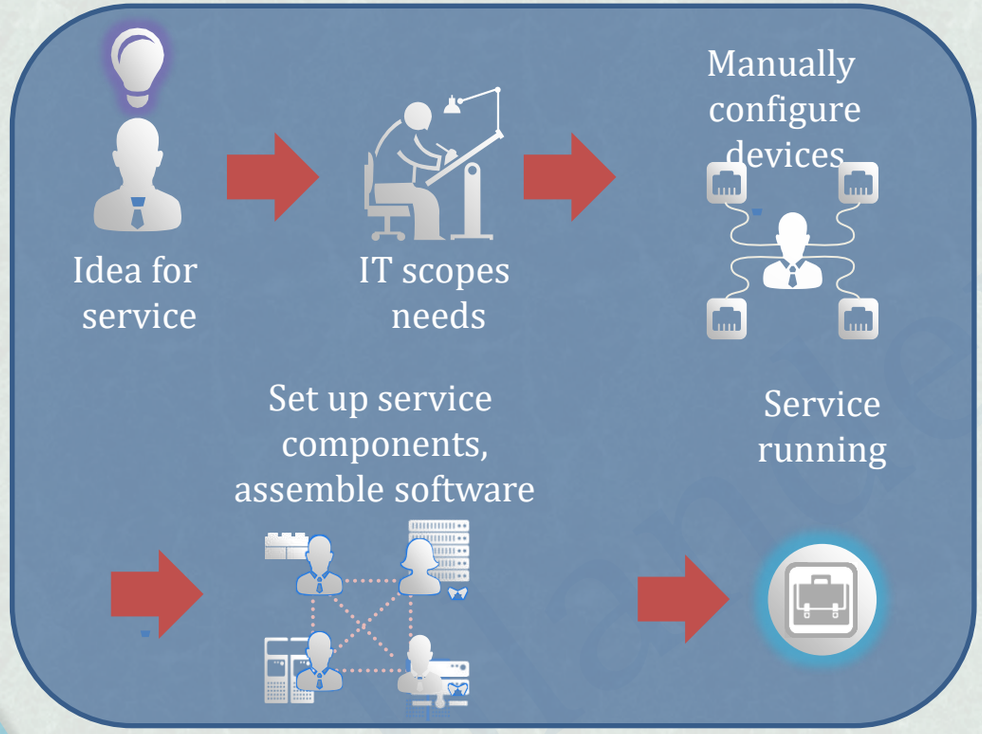


Predictable Bursting

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

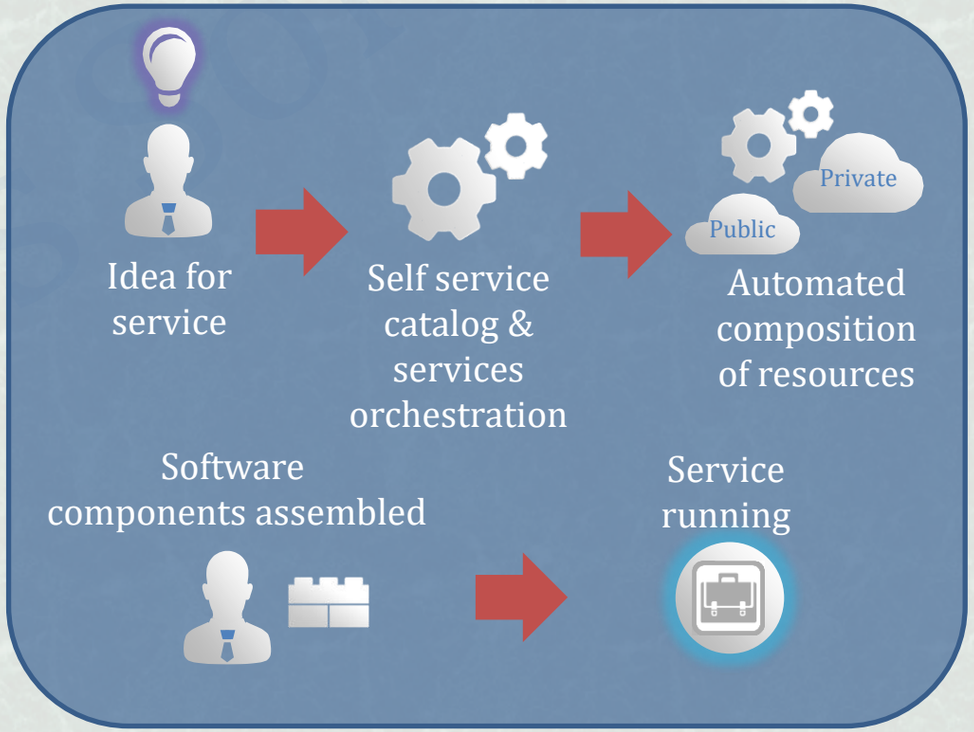
Business Impact of Cloud

Traditional Datacenter



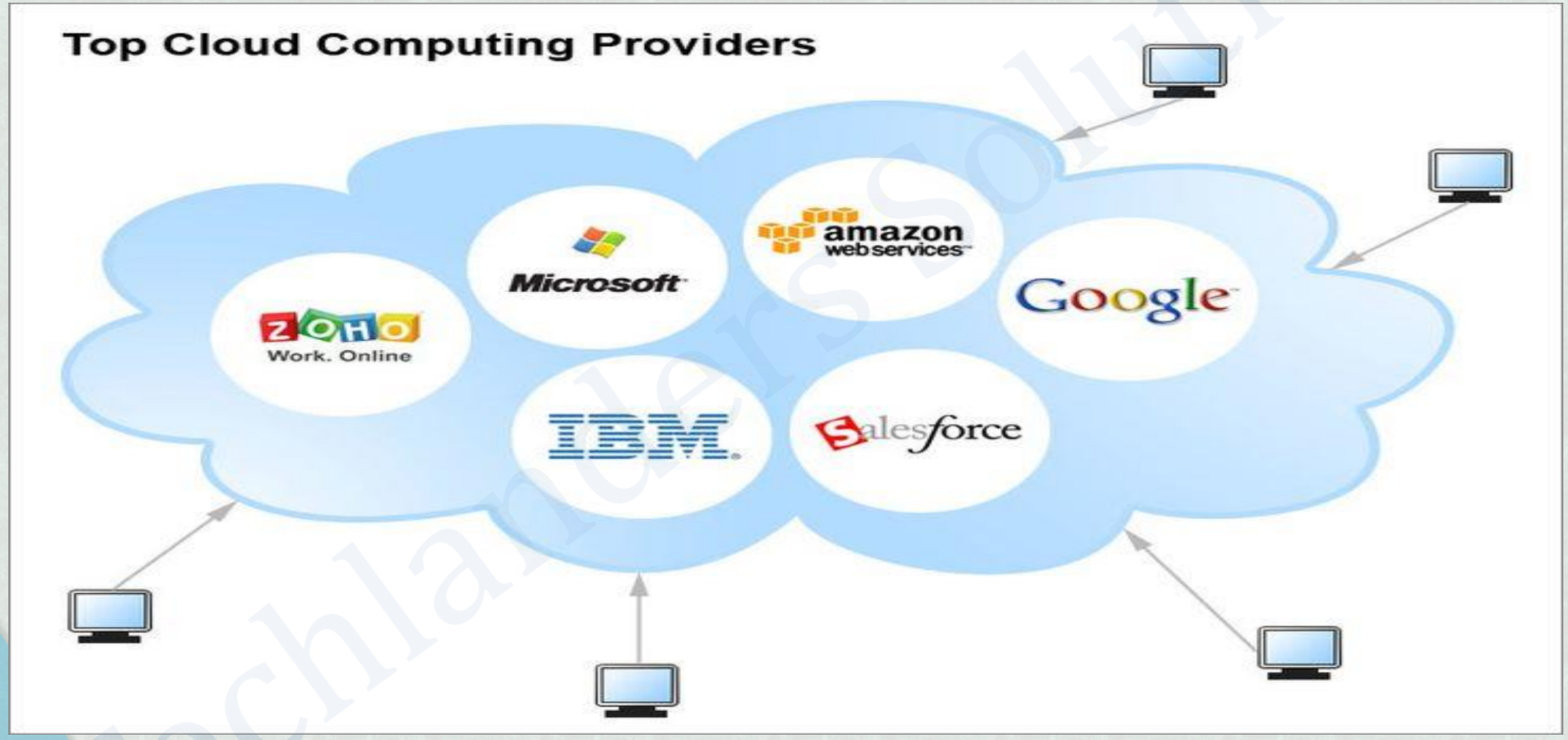
Time to Provision New Service: Months

Cloud Infrastructure

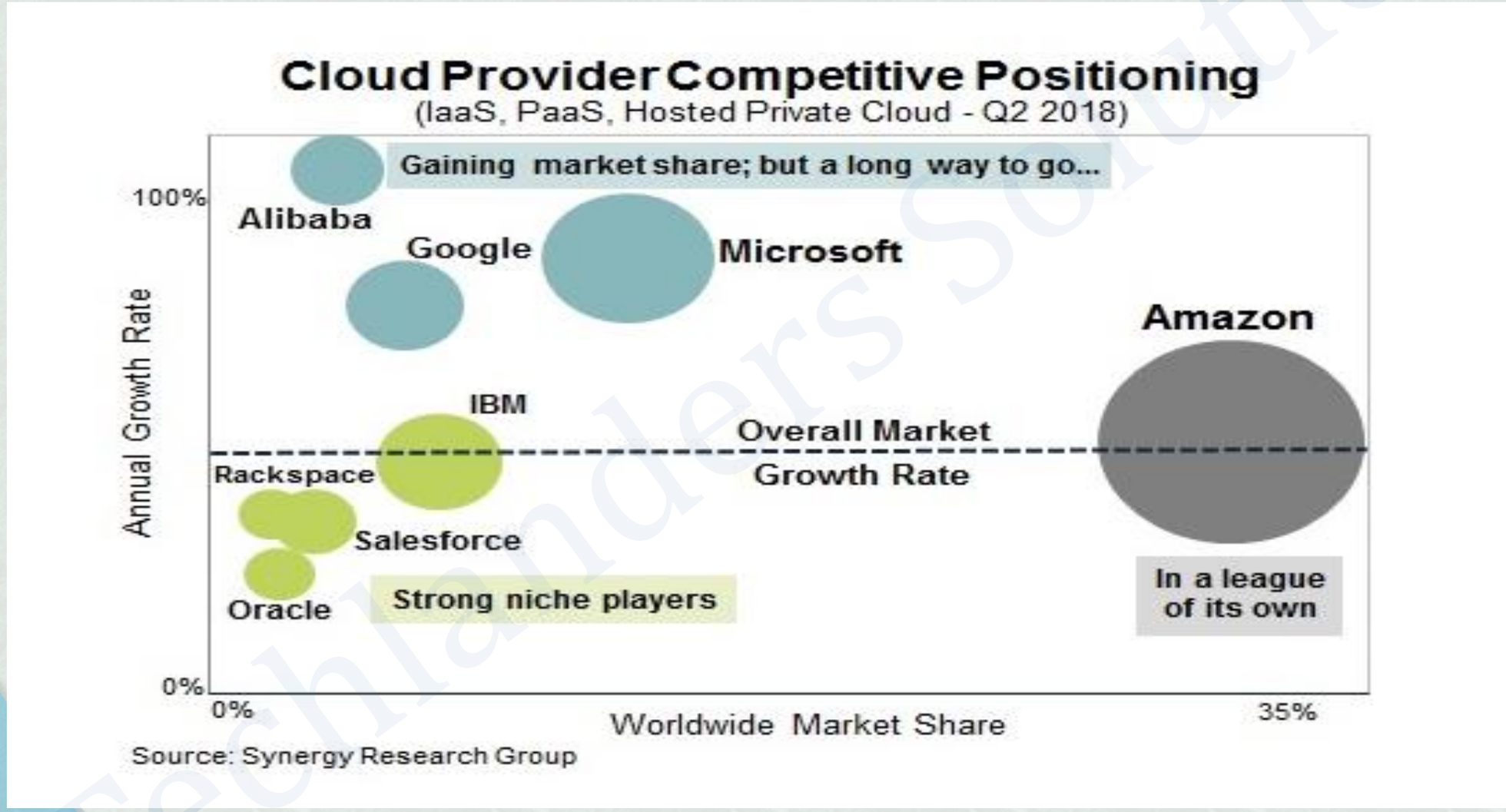


Time to Provision New Service: Minutes

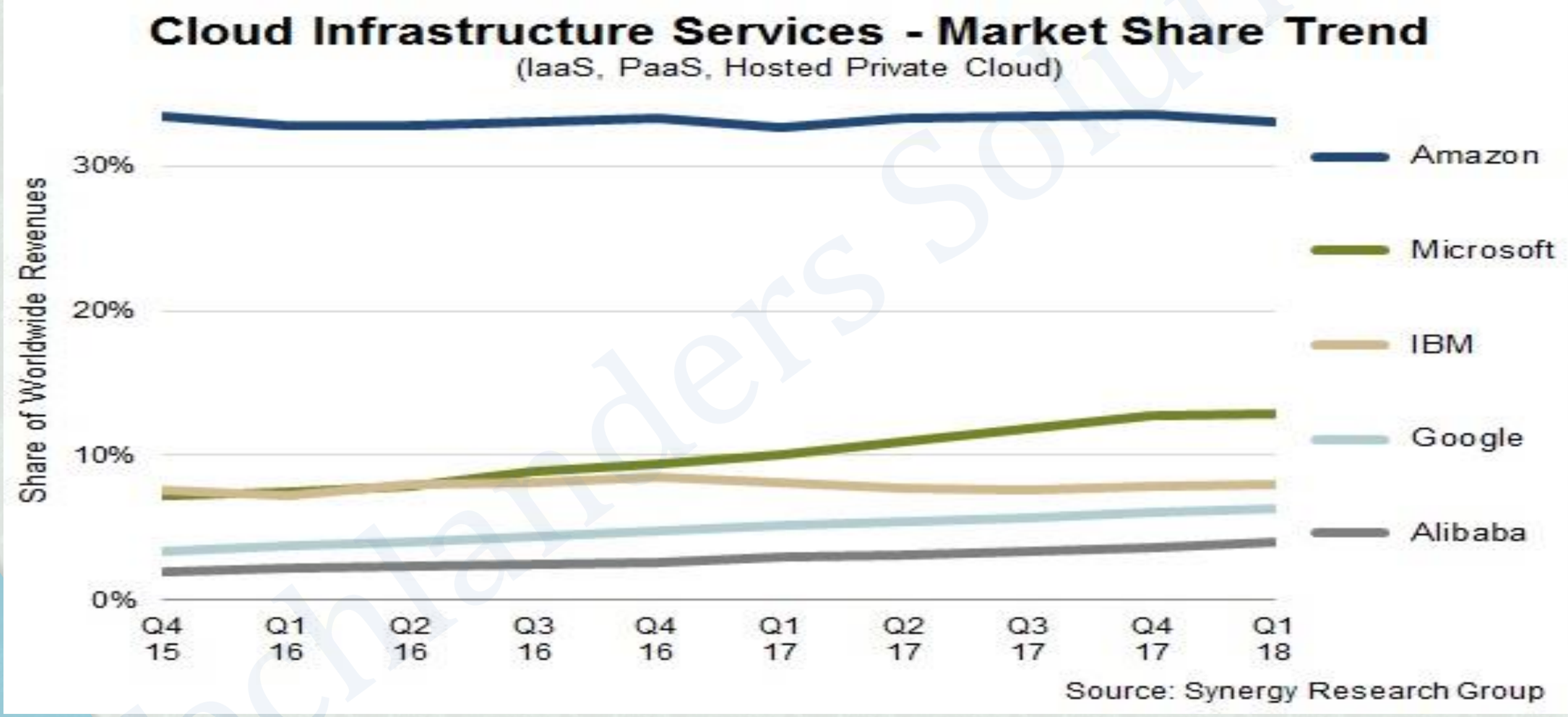
Major Cloud Vendors



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 know 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:

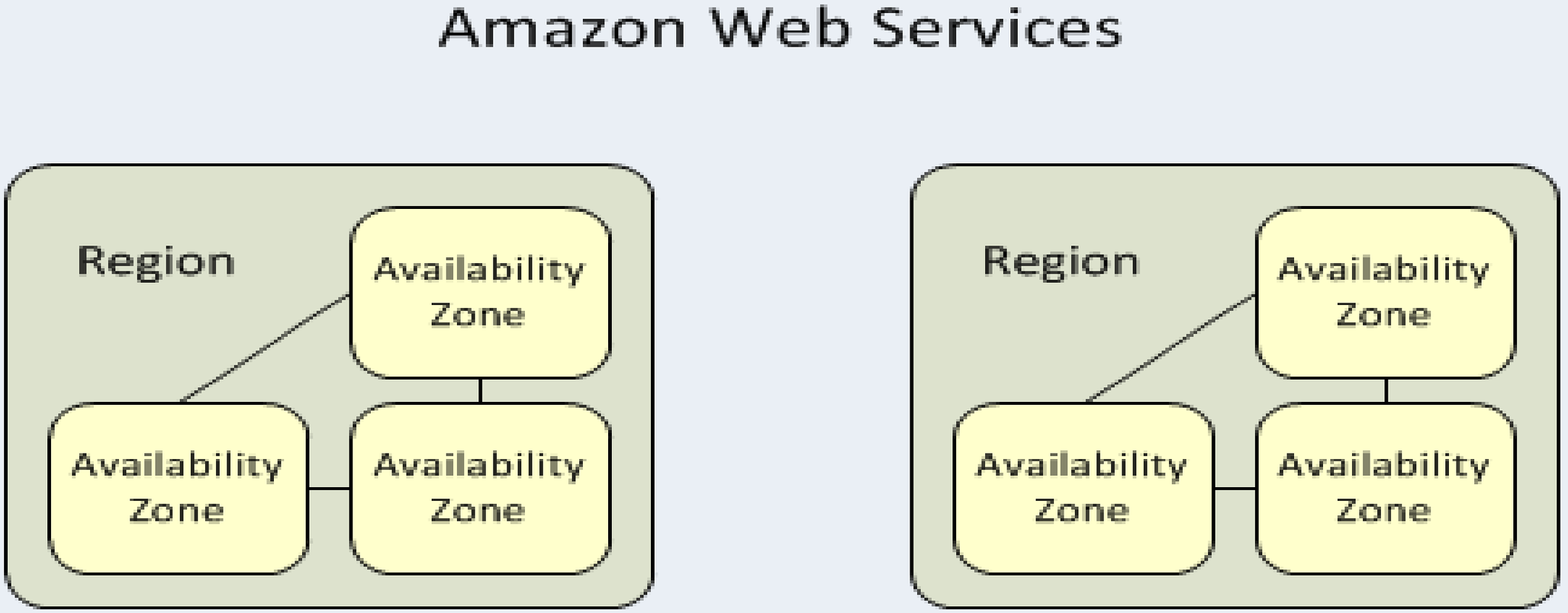
- Geographic Locations
- Consists of at least two Availability Zones(AZs)
- All of the regions are completely independent of each other with separate Power Sources, Cooling and Internet connectivity.

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.

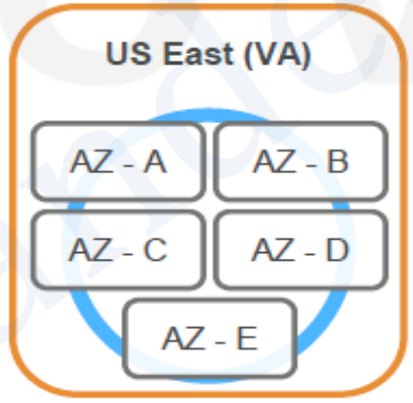
AWS Global Infrastructure

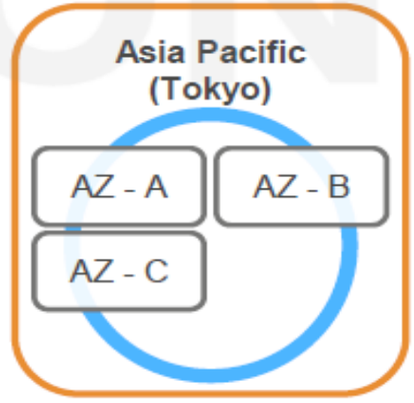


AWS Global Infrastructure

At least 2 AZs per region.

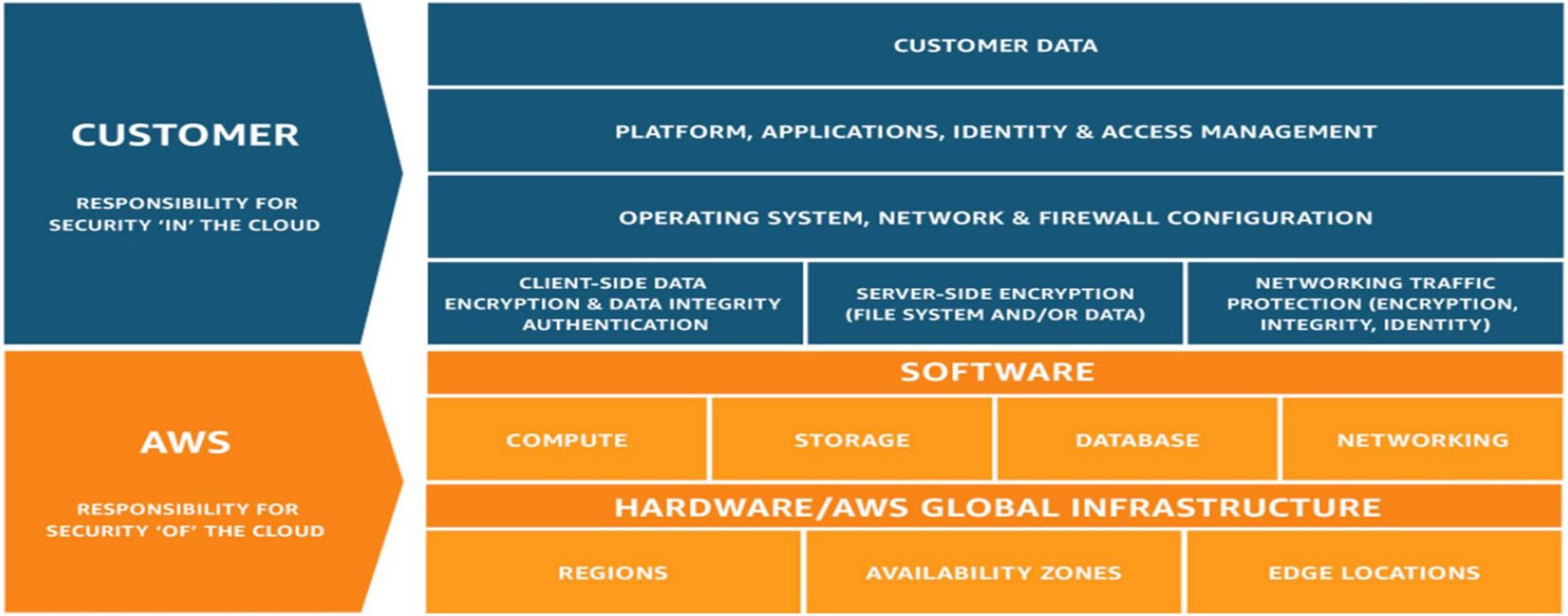
- Examples:
  - US East (N. Virginia)
    - us-east-1a
    - us-east-1b
    - us-east-1c
    - us-east-1d
    - us-east-1e
  - Asia Pacific (Tokyo)
    - ap-northeast-1a
    - ap-northeast-1b
    - ap-northeast-1c



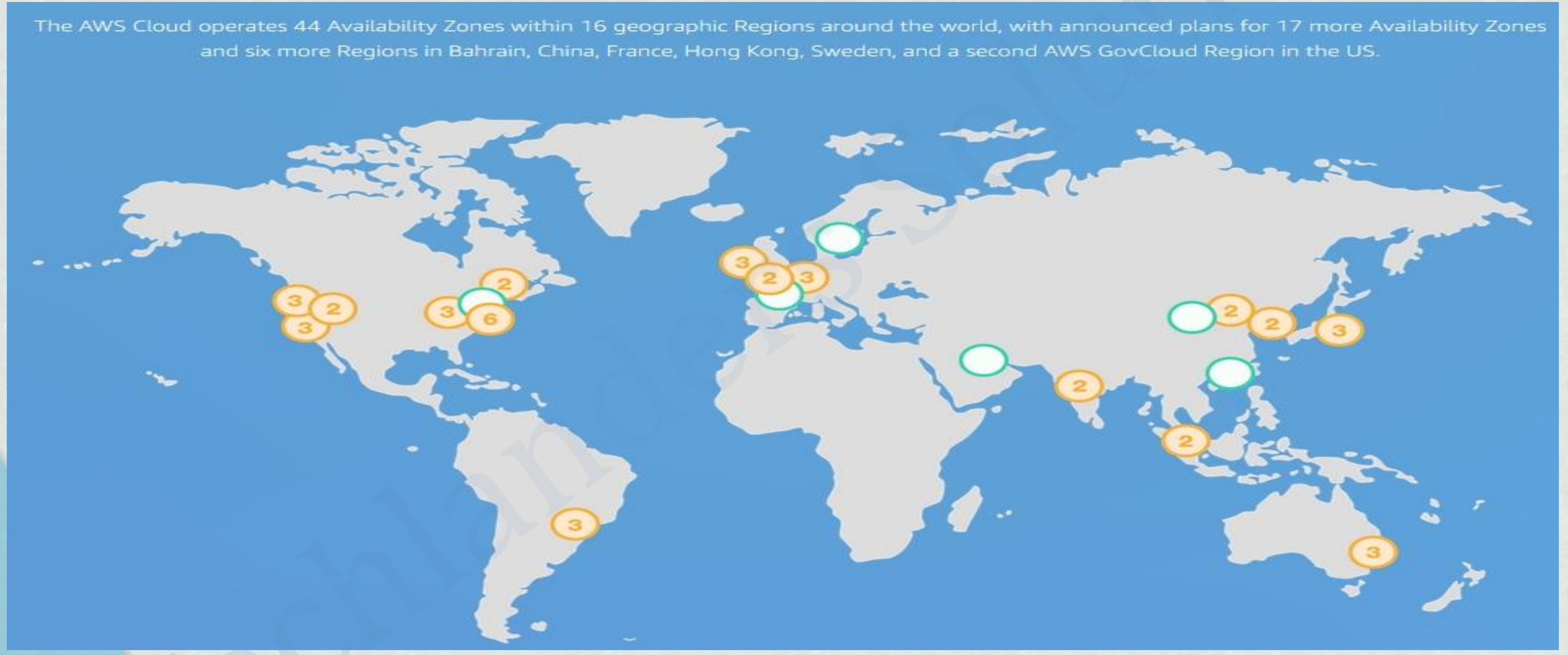


Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

Shared Responsibility

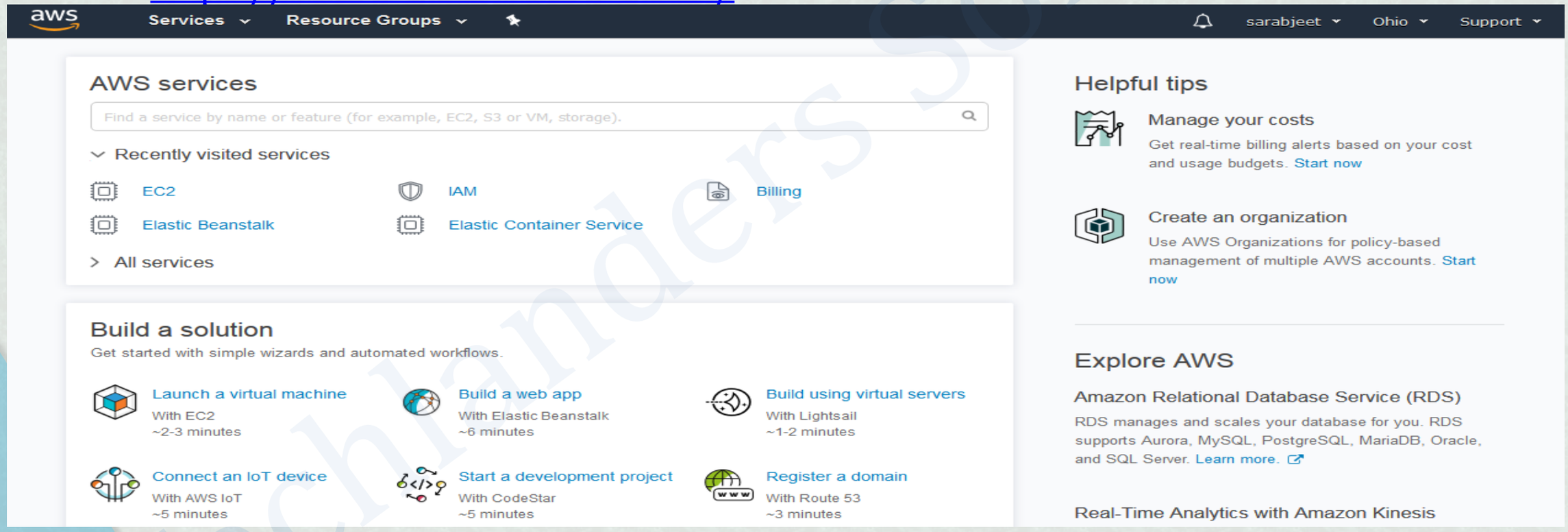


AWS Global Infrastructure



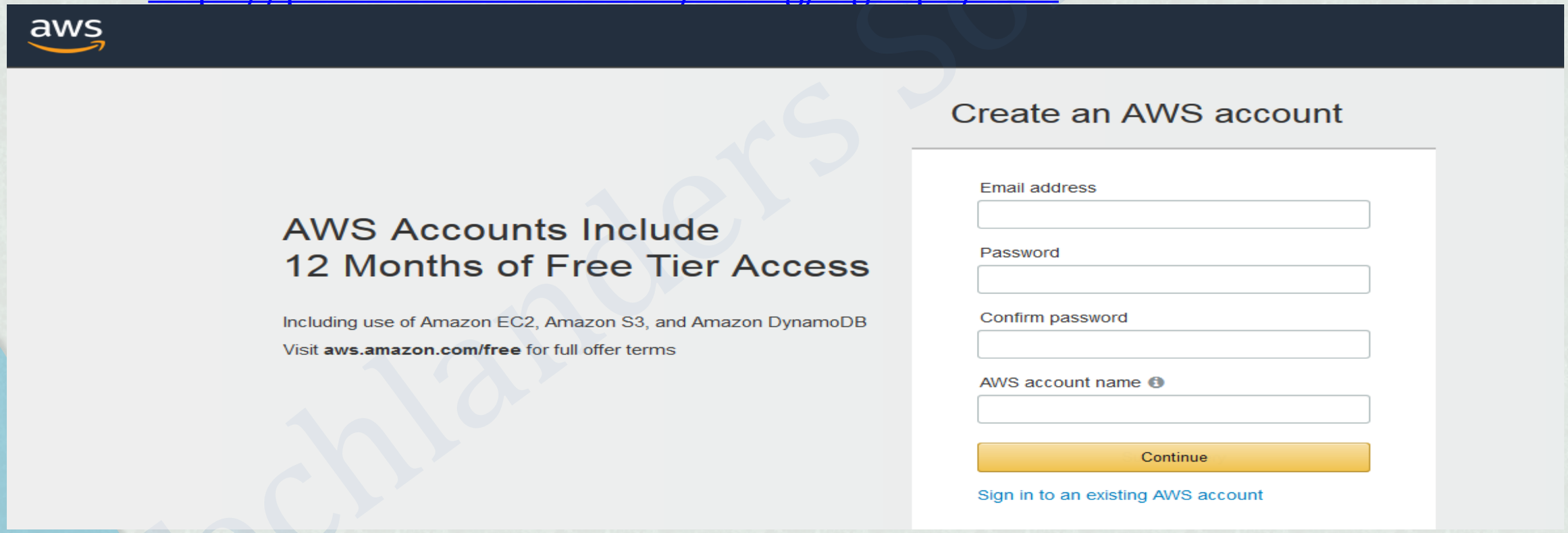
AWS Management Console

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

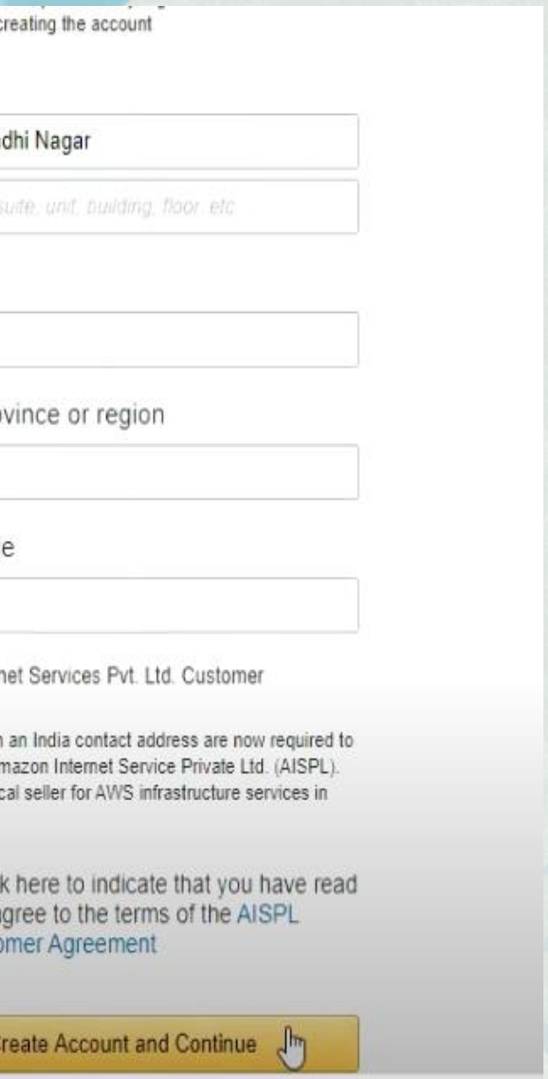


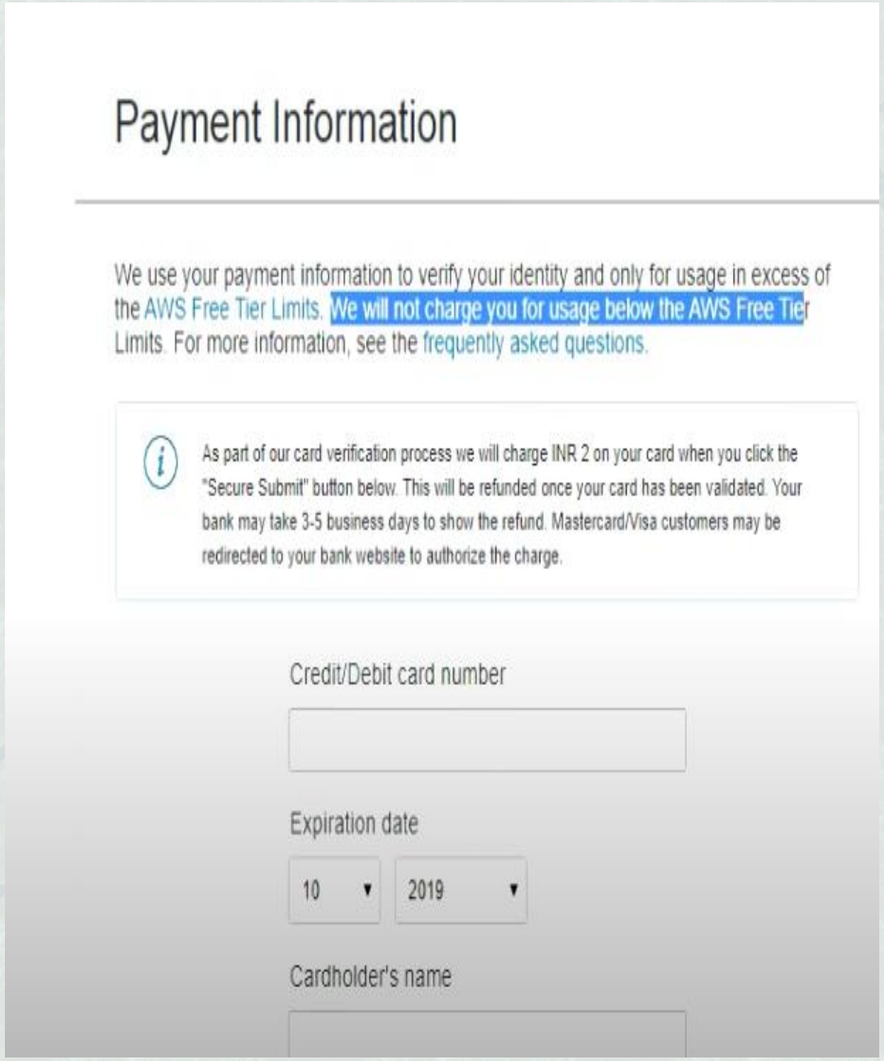
LAB 1 : AWS Signup

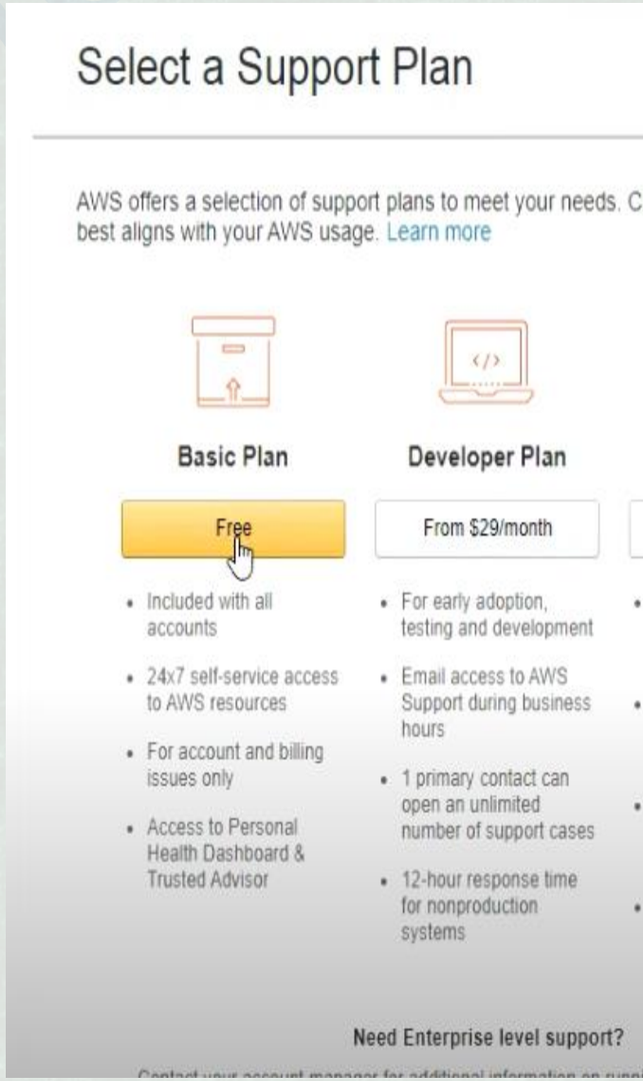
- Create a new account at
  - https://portal.aws.amazon.com/billing/signup#/start



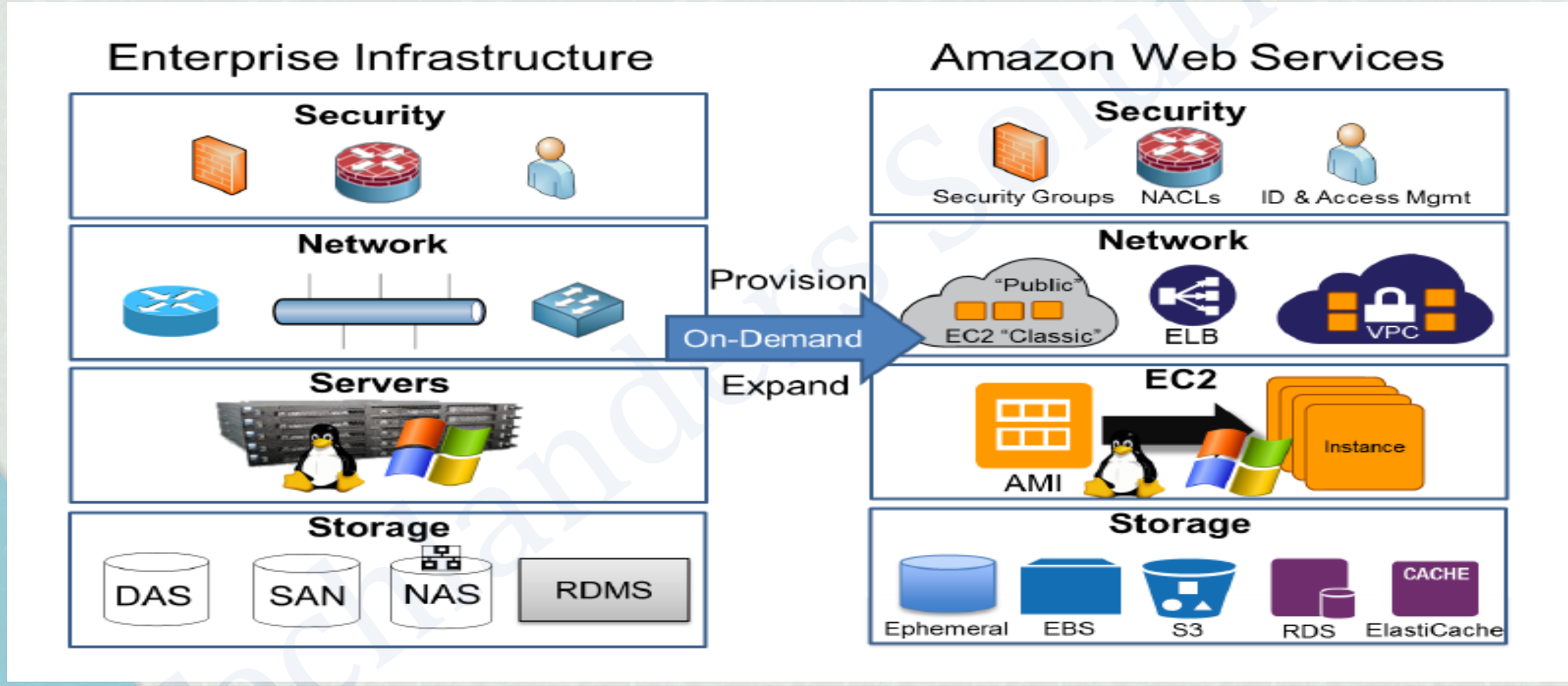
AWS SIGNUP







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

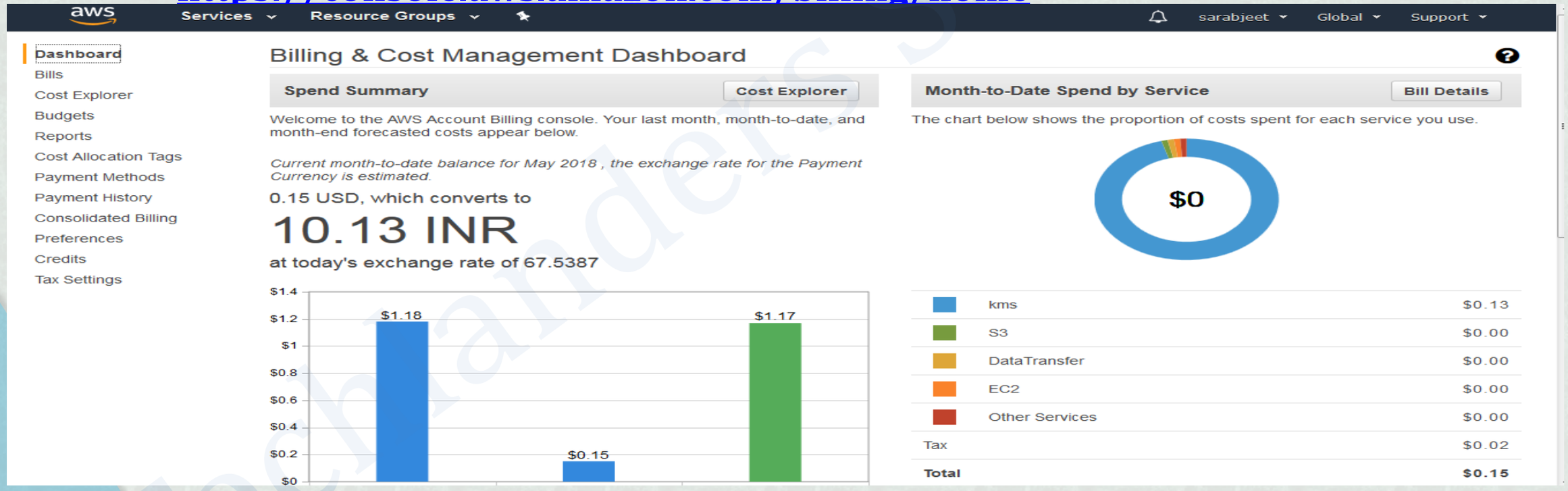
AWS resources can be access using several authentication methods:

- 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://calculator.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://awstcocalculator.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
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)

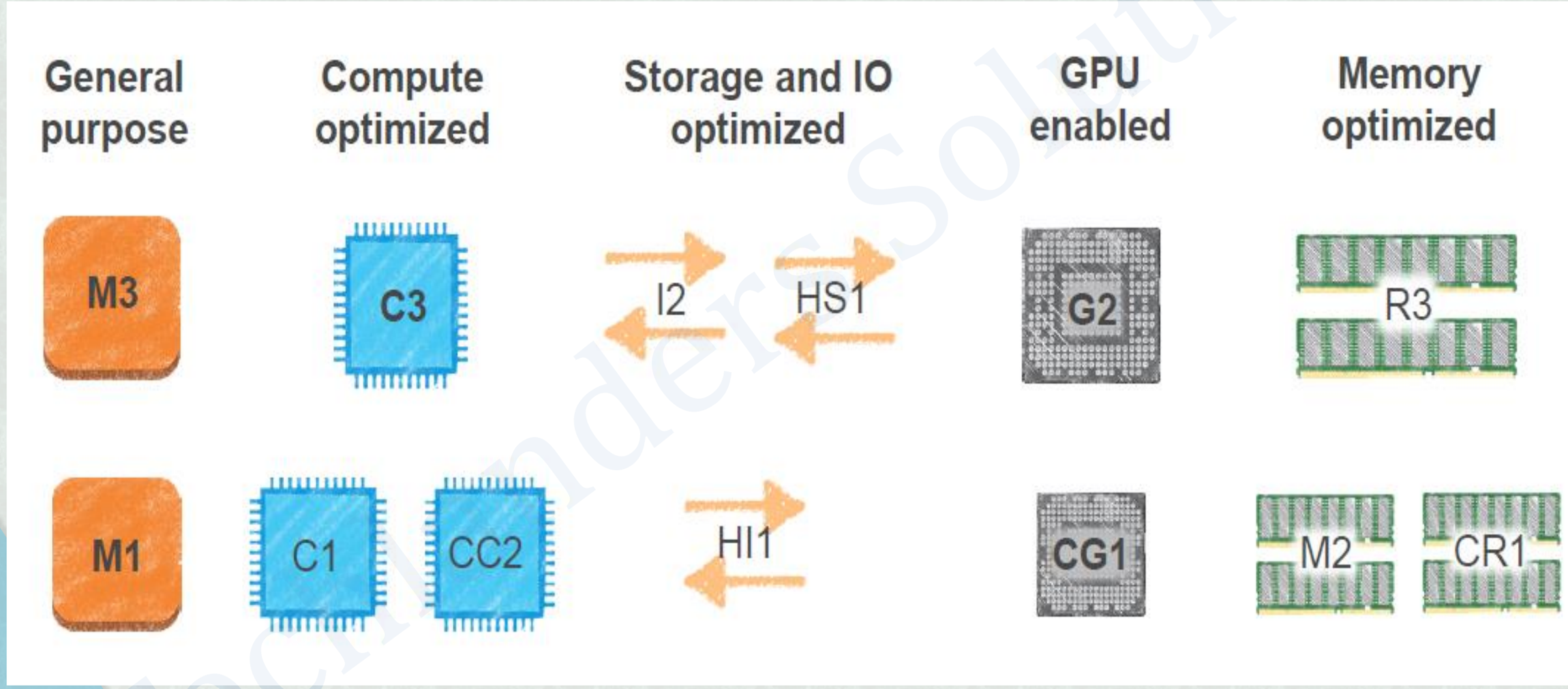
- EC2 instance types are optimized for different use cases and come in multiple sizes. This allows you to optimally scale resources to your workload requirements.

- AWS uses Intel® Xeon® processors for EC2 instances, providing customers with high performance and value.

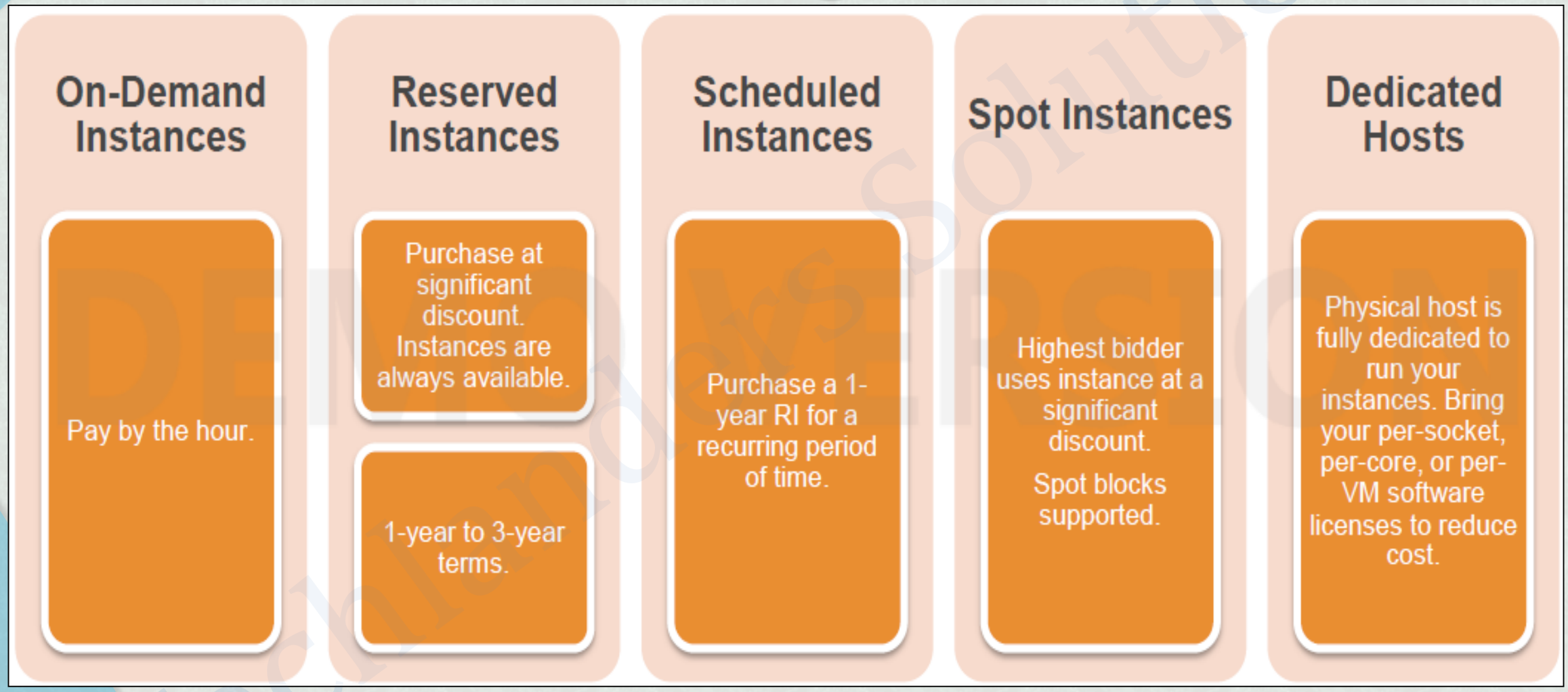
- Consider the following when choosing your instances: Core count, memory size, storage size and type, network performance, and CPU technologies.

- 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



EC2 Pricing Models



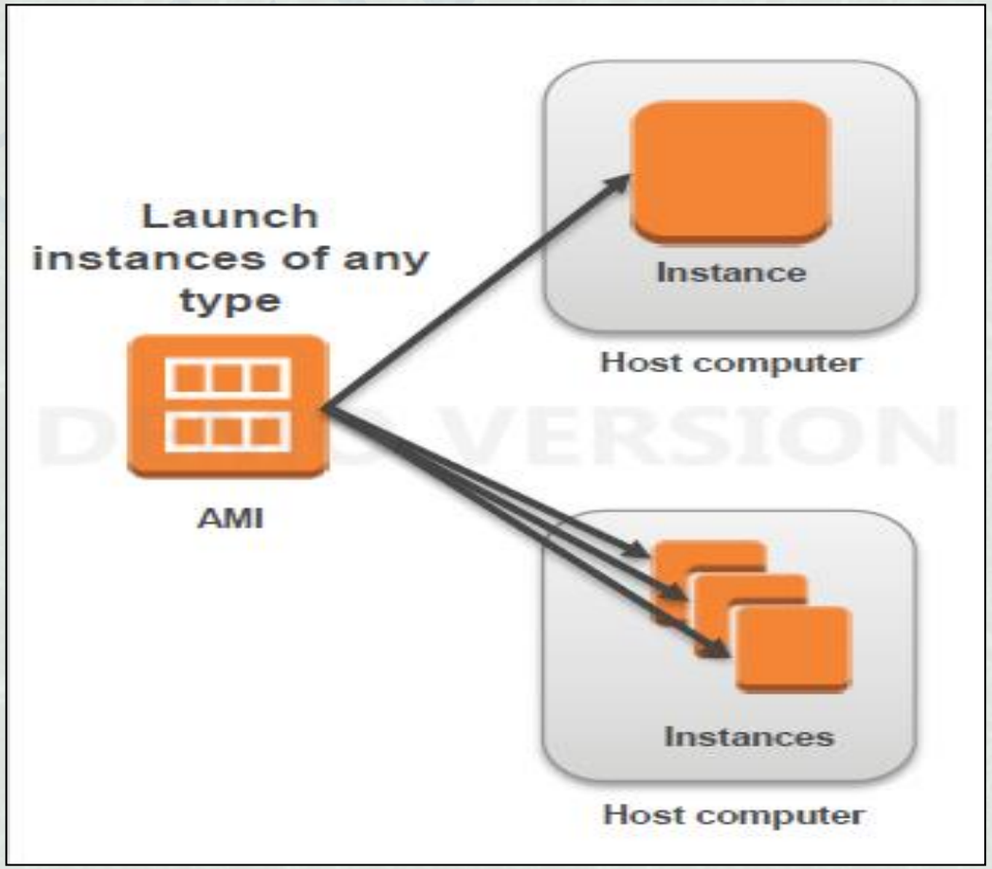
AMI's

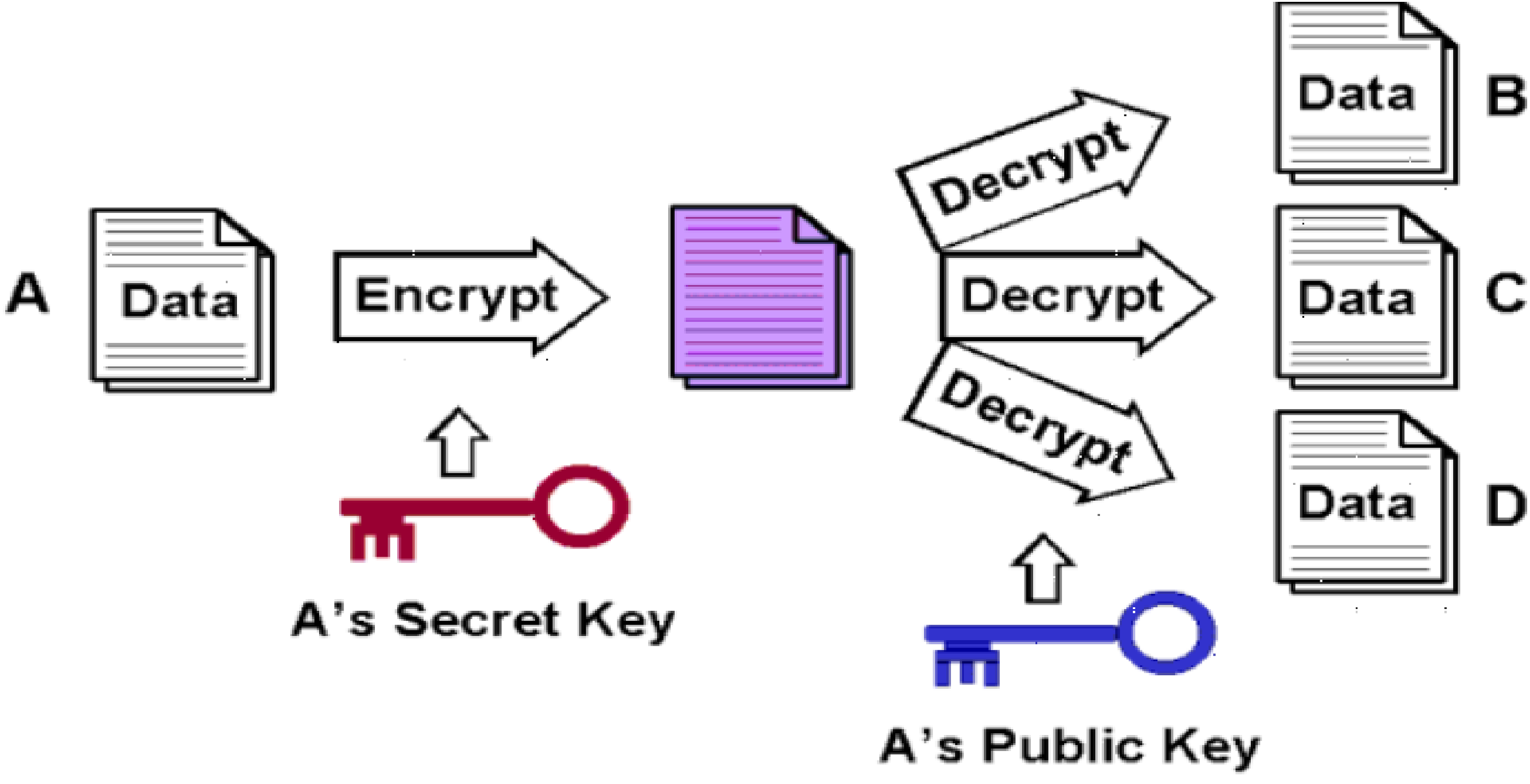
- Stands for **Amazon Machine Image**
- **AMI is a template for the root volume** for the instance (example: an OS image, a webserver, an application server etc.)
- It also includes the launch permissions that control which AWS accounts can use the AMI to launch instances.
- All AMIs are based on x86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




A
Data
Encrypt
A's Secret Key
Decrypt
Decrypt
Decrypt
A's Public Key
Data
B
Data
C
Data
D

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

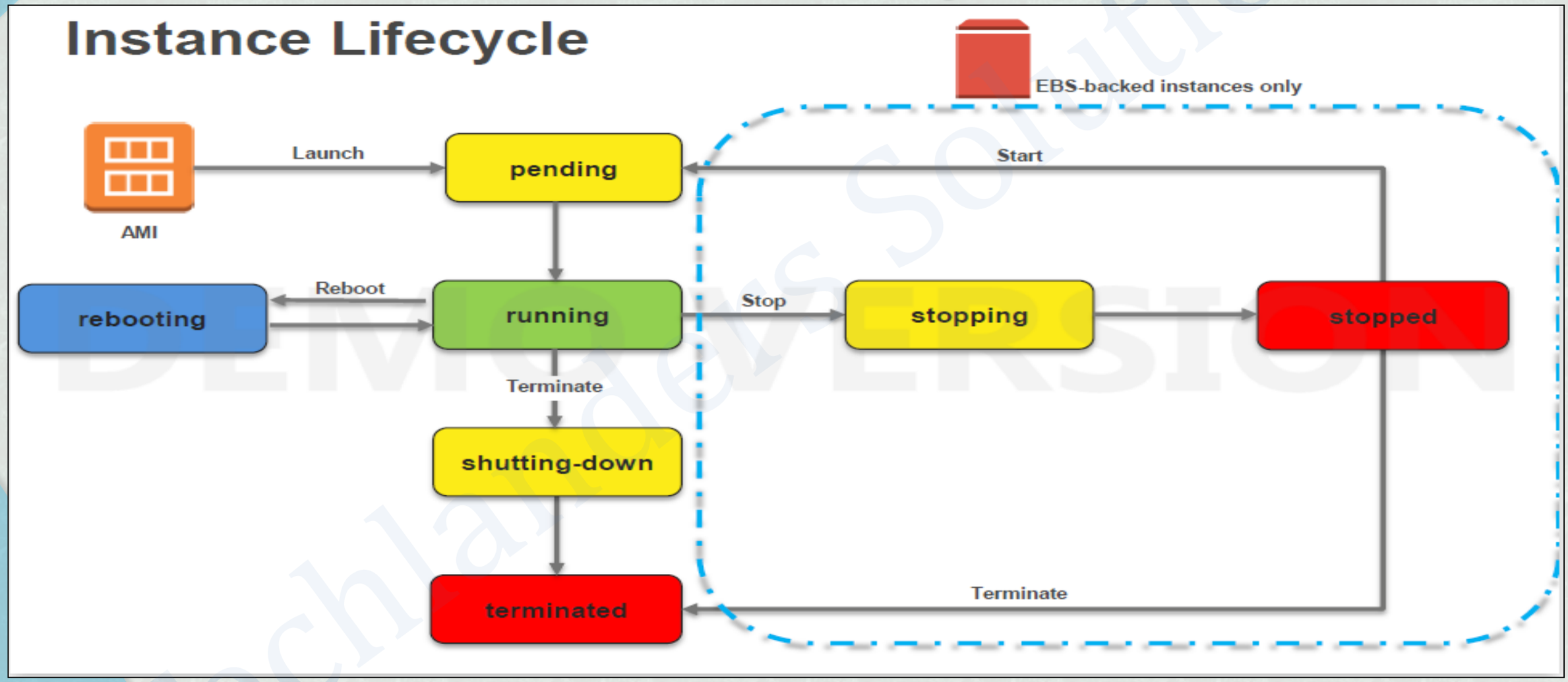
Console Access: https://console.aws.amazon.com/

- 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



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

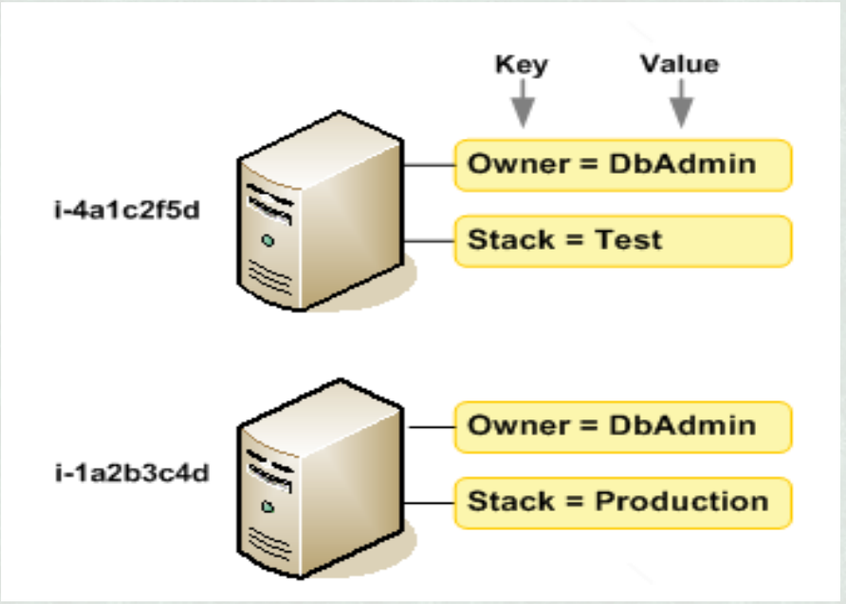
Tags helps you manage your instances, images, and other Amazon EC2 resources.

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 observer 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:

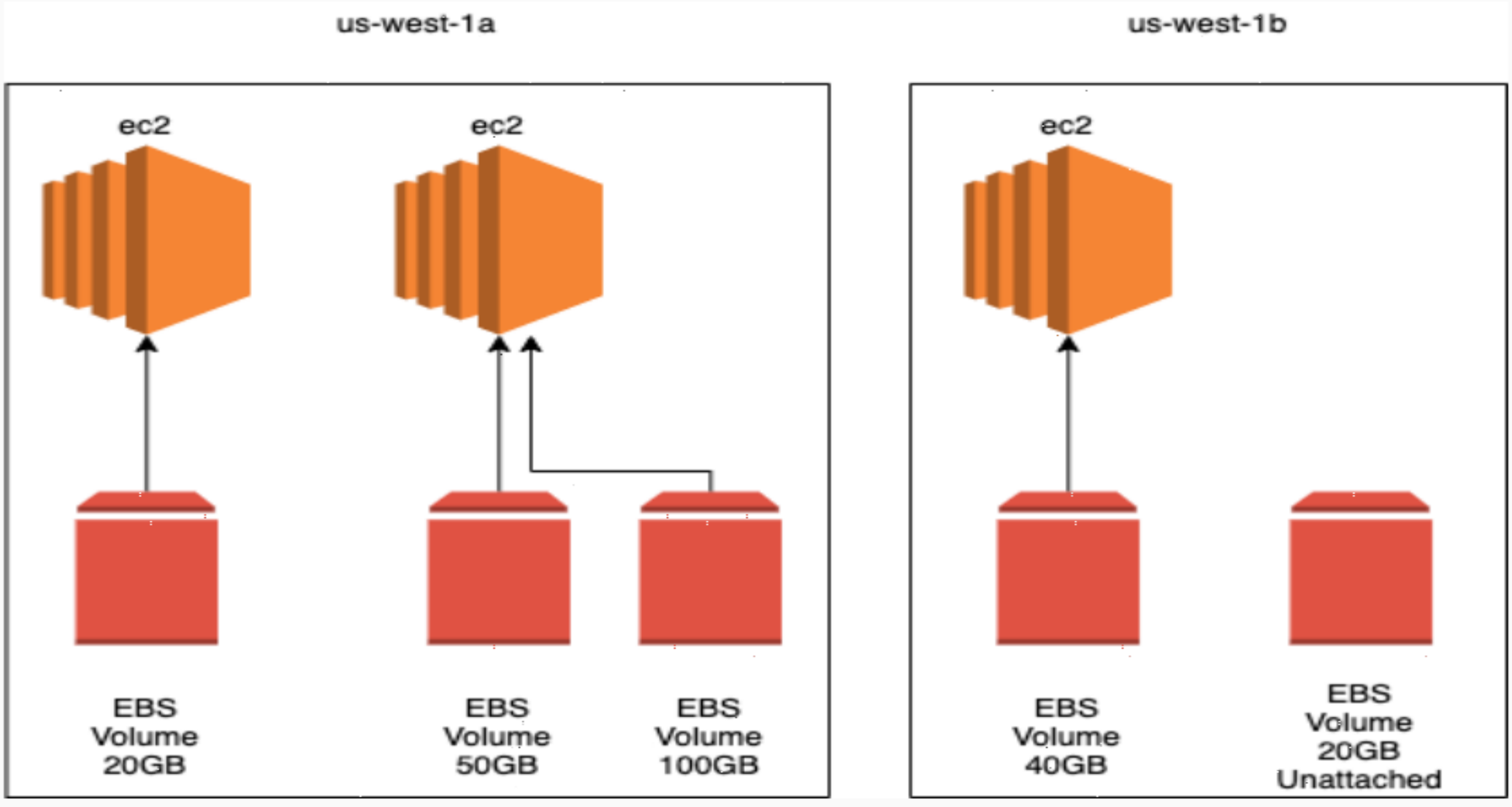
```
#!/bin/bash
sudo yum update -y;
sudo yum install -y httpd;
sudo systemctl start httpd;
sudo systemctl enable httpd;
sudo echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html
```

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

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.

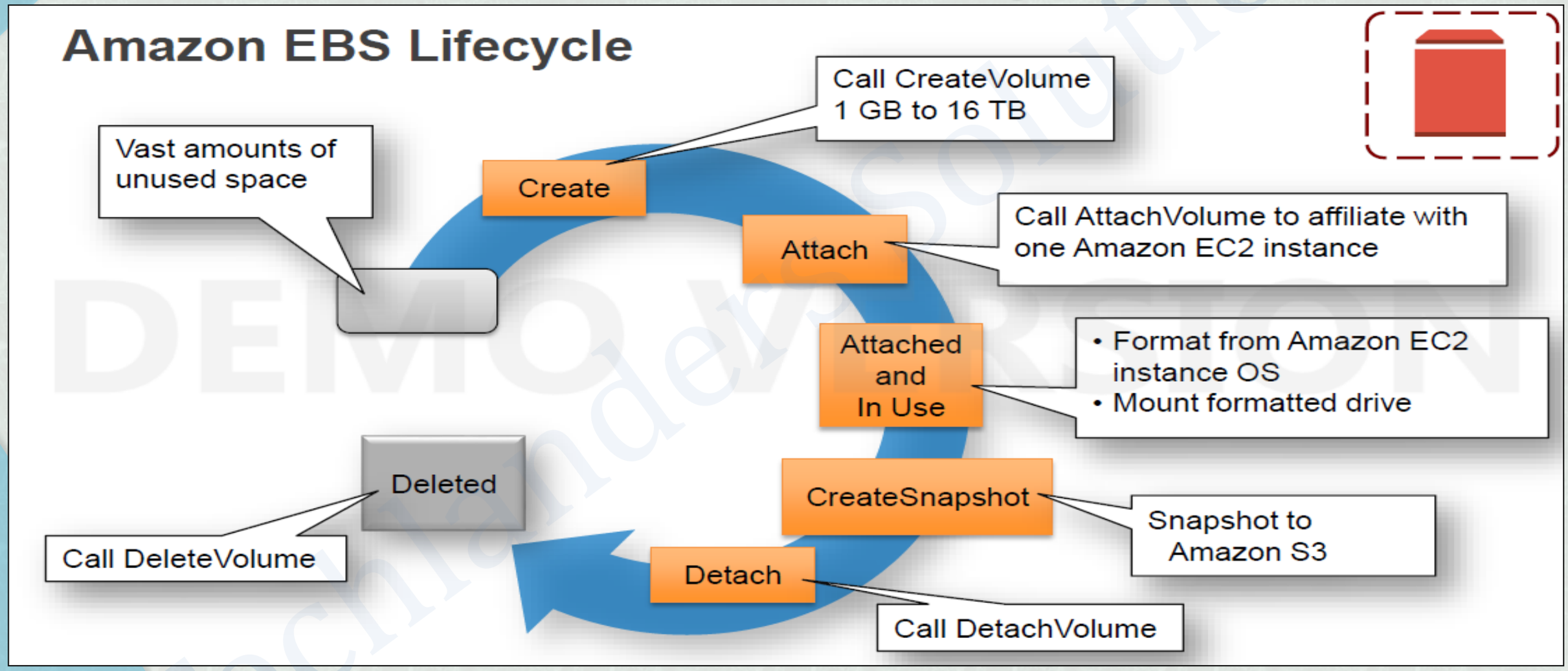


Amazon EBS Volumes

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



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	• Recommended for most workloads
	• System boot volumes
	• Virtual desktops
	• Low-latency interactive apps
	• Development and test environments
		• 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:
		  ◦ MongoDB
		  ◦ Cassandra
		  ◦ Microsoft SQL Server
		  ◦ MySQL
		  ◦ PostgreSQL
		  ◦ Oracle
			• Streaming workloads requiring consistent, fast throughput at a low price
			• Big data
			• Data warehouses
			• Log processing
			• Cannot be a boot volume
				• 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

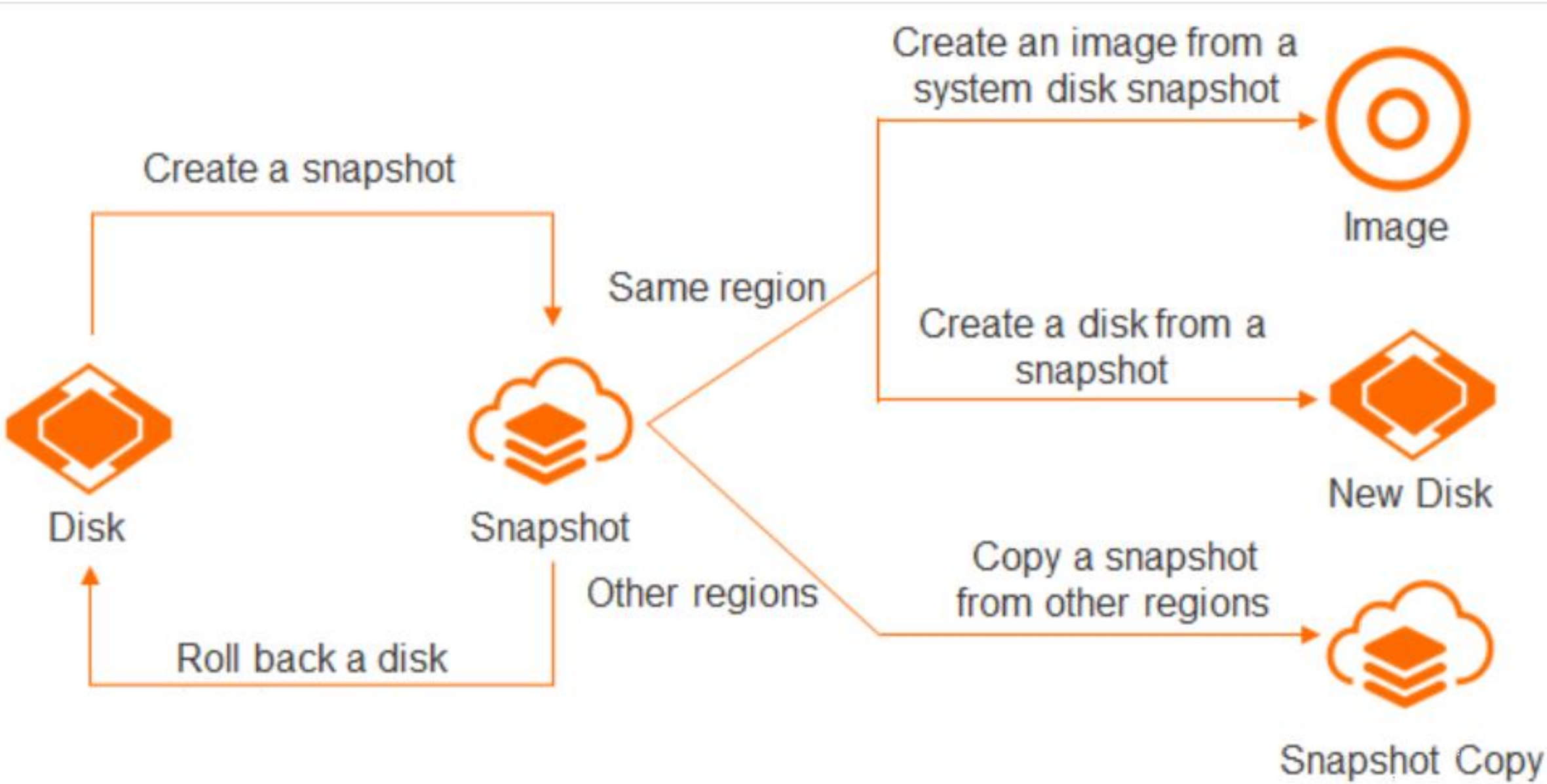
API Name	gp2	io1	st1	sc1
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max. IOPS/Volume**	10,000	32,000***	500	250
Max. Throughput/Volume	160 MiB/s	500 MiB/s†	500 MiB/s	250 MiB/s
Max. IOPS/Instance	80,000	80,000	80,000	80,000
Max. Throughput/Instance††	1,750 MiB/s	1,750 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).

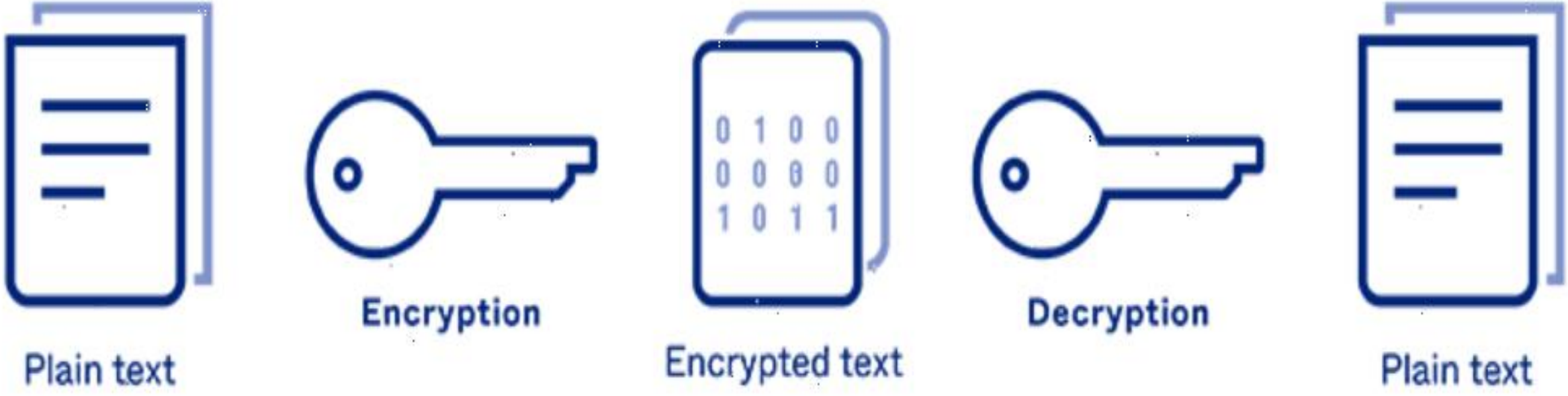

Create an image from a system disk snapshot
Image
Create a snapshot
Same region
Create a disk from a snapshot
New Disk
Disk
Snapshot
Copy a snapshot from other regions
Other regions
Roll back a disk
Snapshot Copy

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)



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.

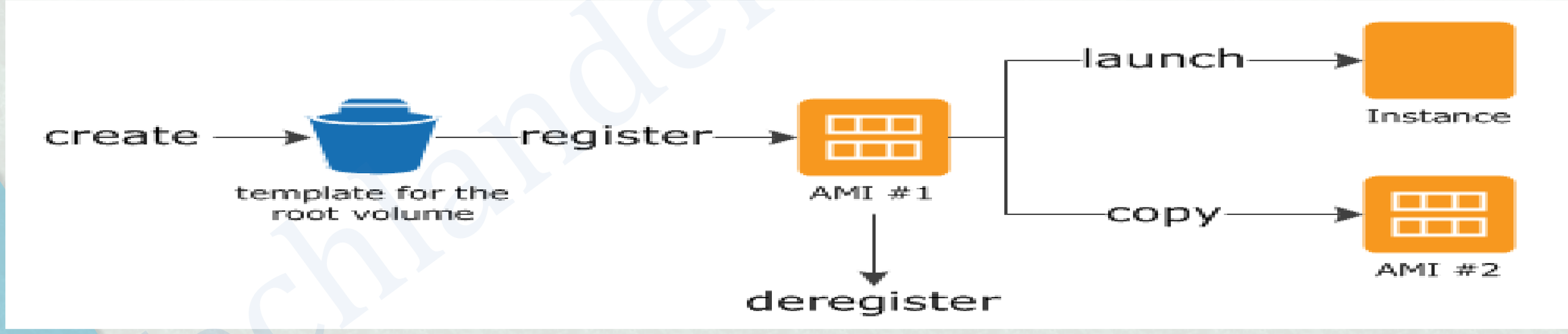
EBS Encryption

- 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

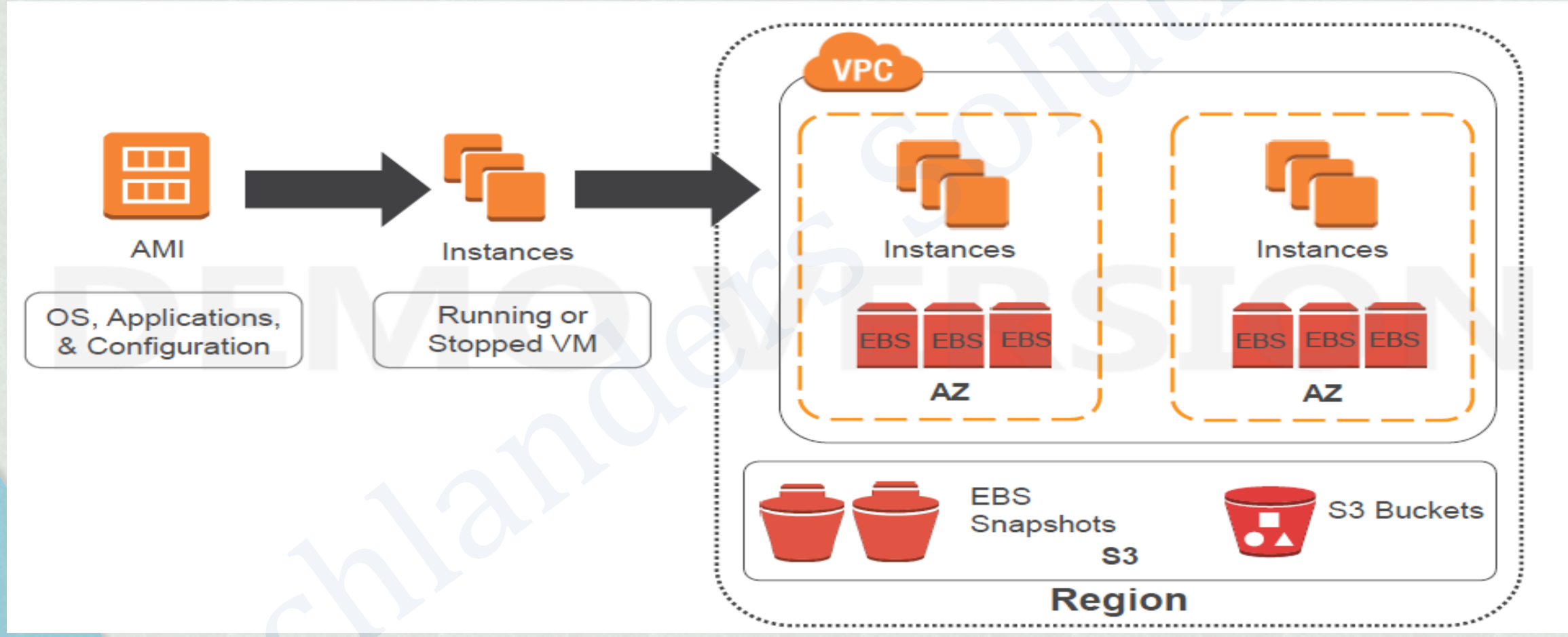
Generalizing an Image:

- 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



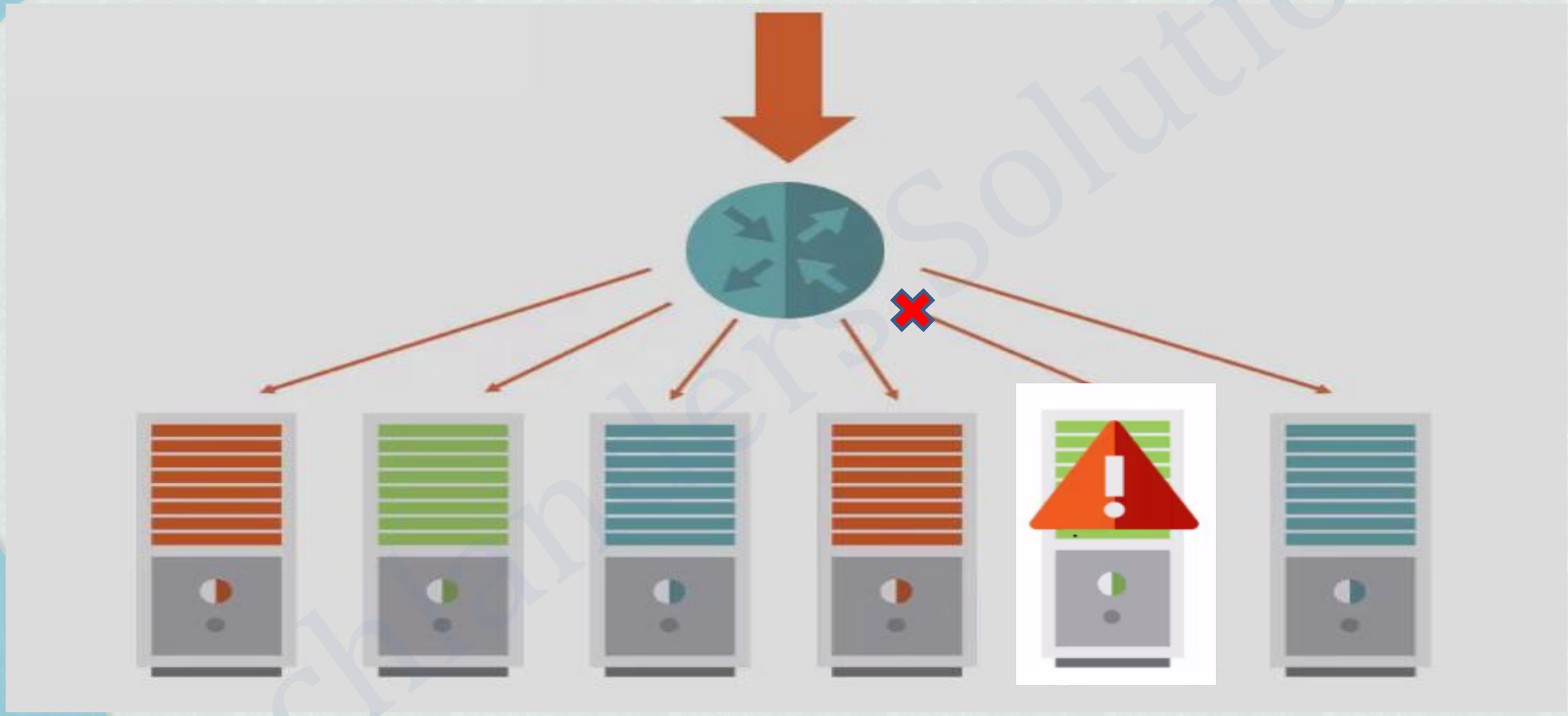
AWS

Load Balancer as a Service - ELB

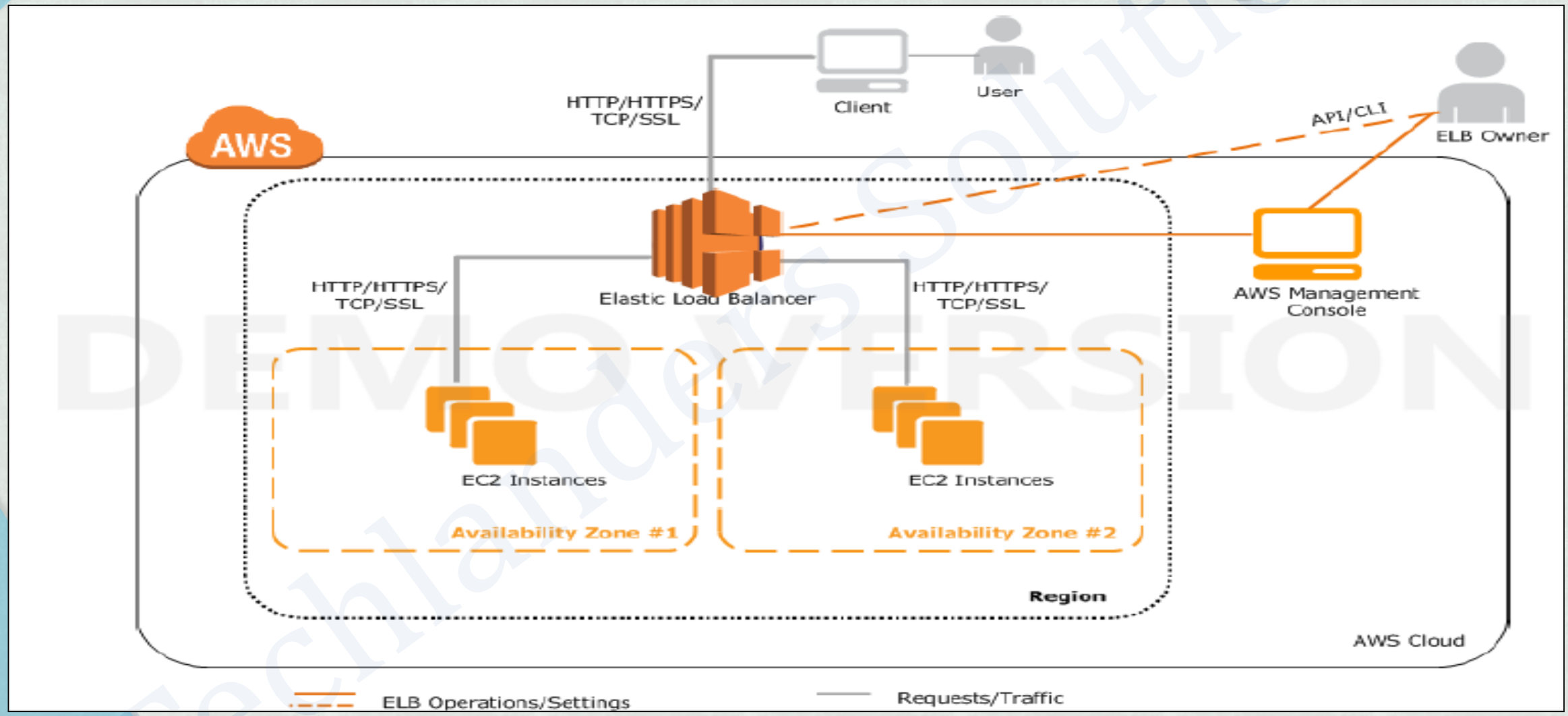
AWS Elastic Load Balancing

- **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.

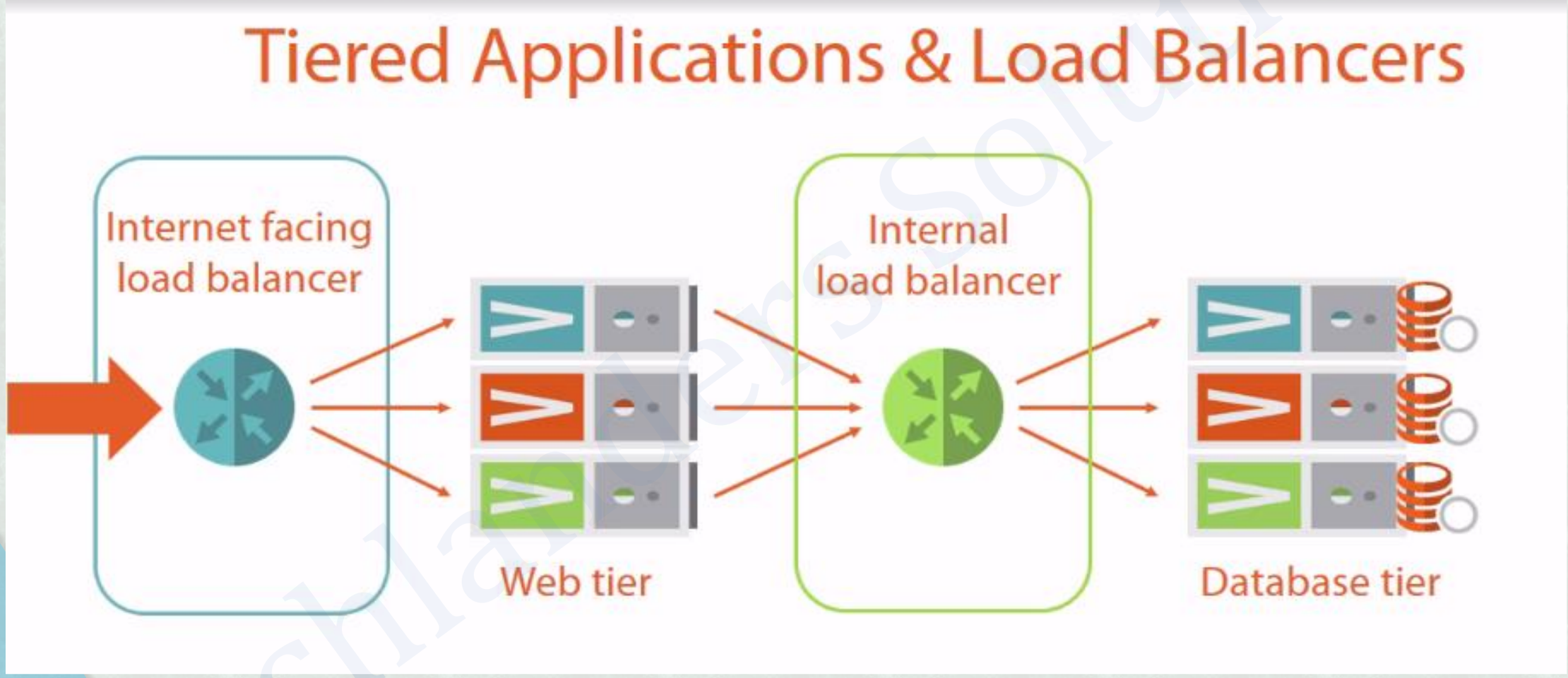
Elastic Load Balancers



AWS Elastic Load Balancing



Load Balancer Types and Tiering



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.

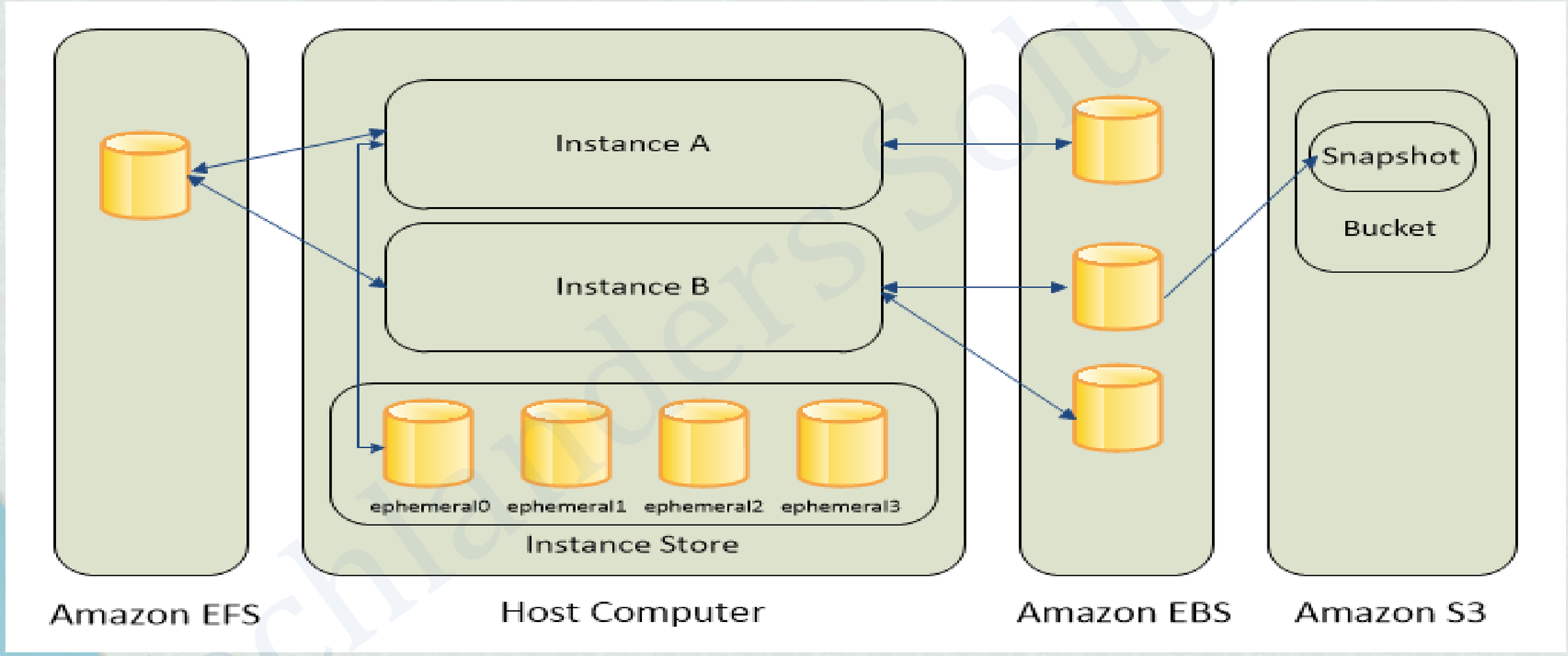
- **Classic Load Balancer:**

  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

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

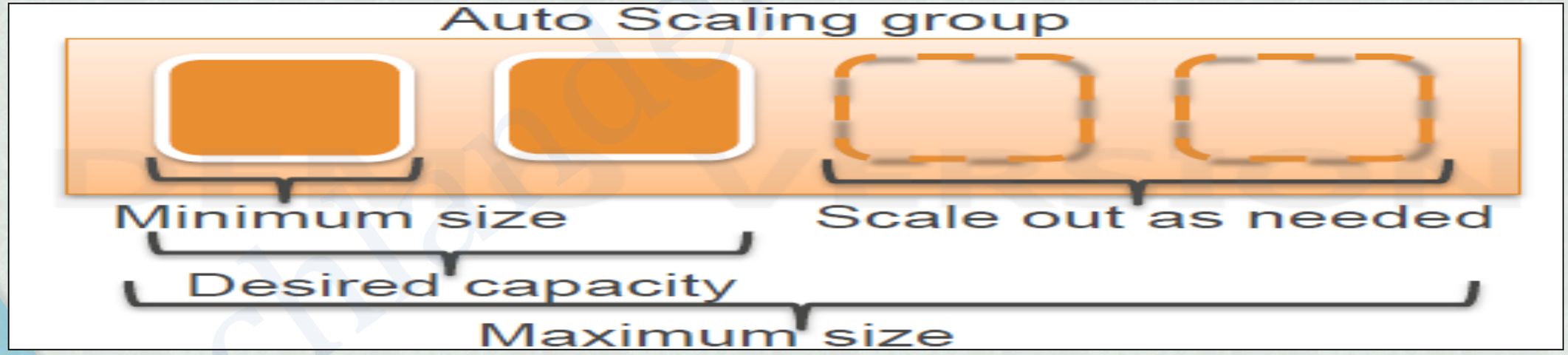
Launch template : A launch configuration is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch template, you can specify:

- 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

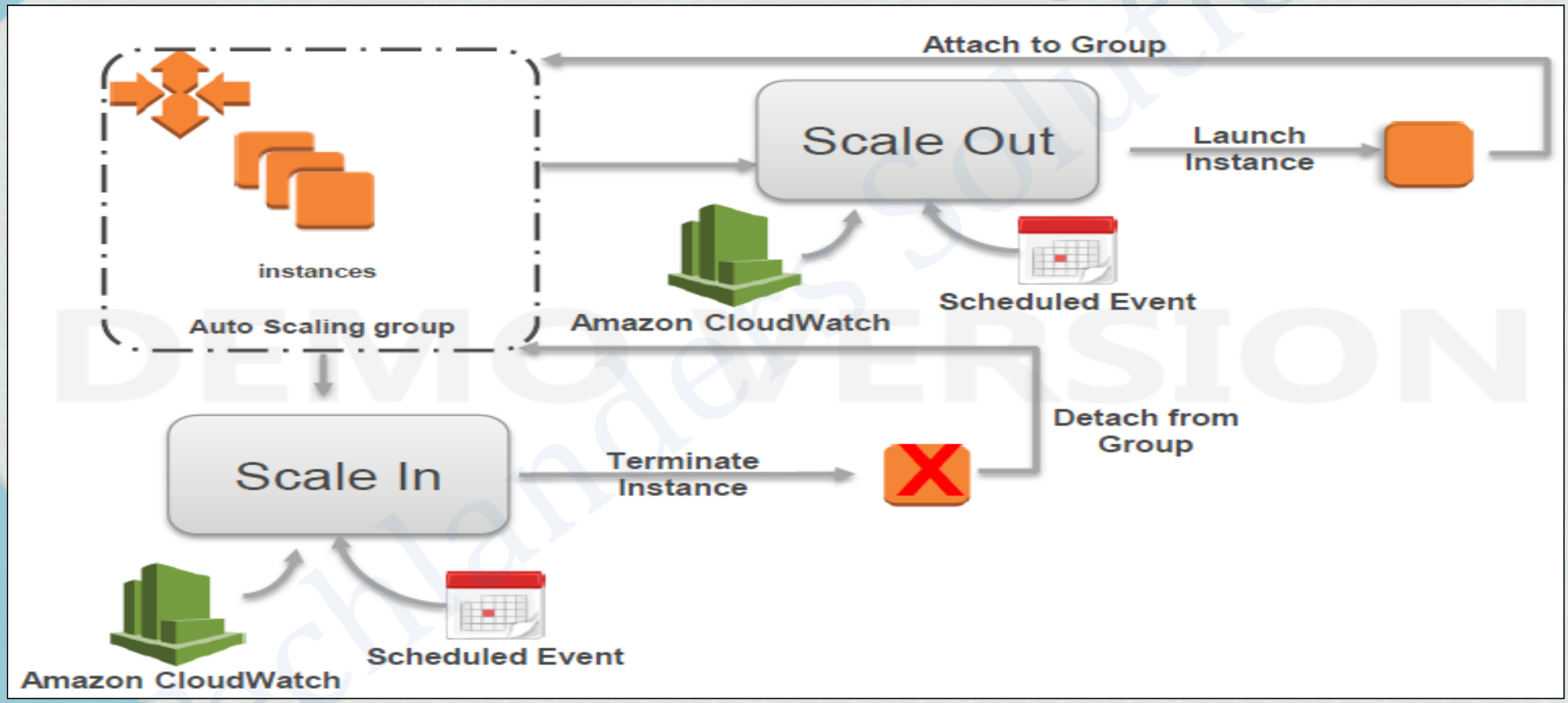
- Based on Health check alerts of Application instances
- Alerts can be based on CPU/Network/Disk IO etc.
- Both Scale-in and Scale out can be done based on Alerts
- Multiple Scale policies can be created and maintained

Scheduled Scaling

- Based on Scheduled Events
- Scaling can be configured to Auto scale once or re-occurance 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/

Execrice1:

- **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

Execrice2:

- **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 **S**imple **S**torage **S**ervices.
- 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 **99.999999999%** durability and **99.99%** availability of objects over a given year.
- **HTTP/S** endpoint to store and retrieve any amount of data, at any time, from anywhere on the web.
- Highly scalable, reliable, fast, and inexpensive.
- Optional server-side **encryption** using AWS or customer- managed provided client-side encryption.
- Amazon S3 objects are automatically replicated on multiple devices in multiple facilities within a region.

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 and try to access it.
- Bucket contents are now accessible using bucket policy.

LAB : S3 WEBSITE HOSTING

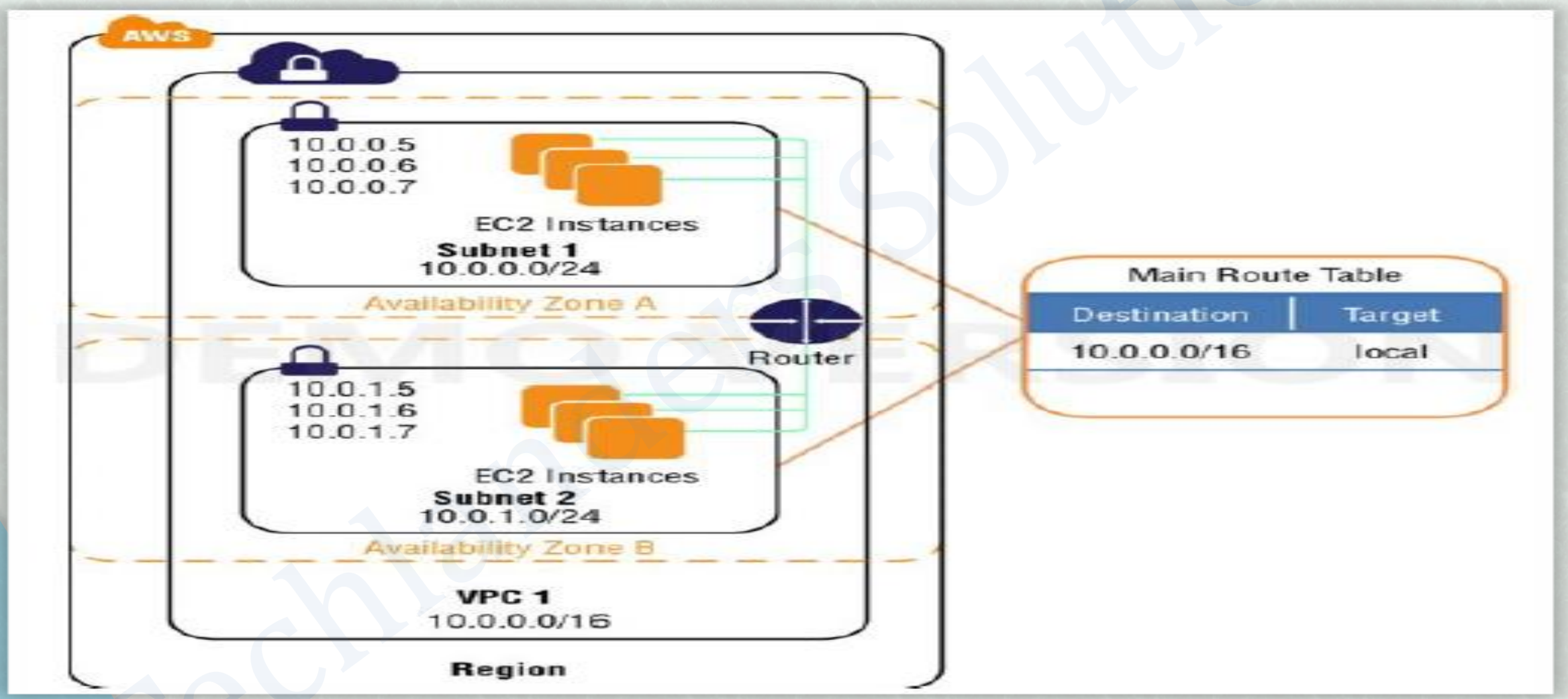
- To host a bucket as website :
- Go to properties > enable static website
- Add html filename saved in the bucket as object.
- Try accessing the website bucket url now.

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

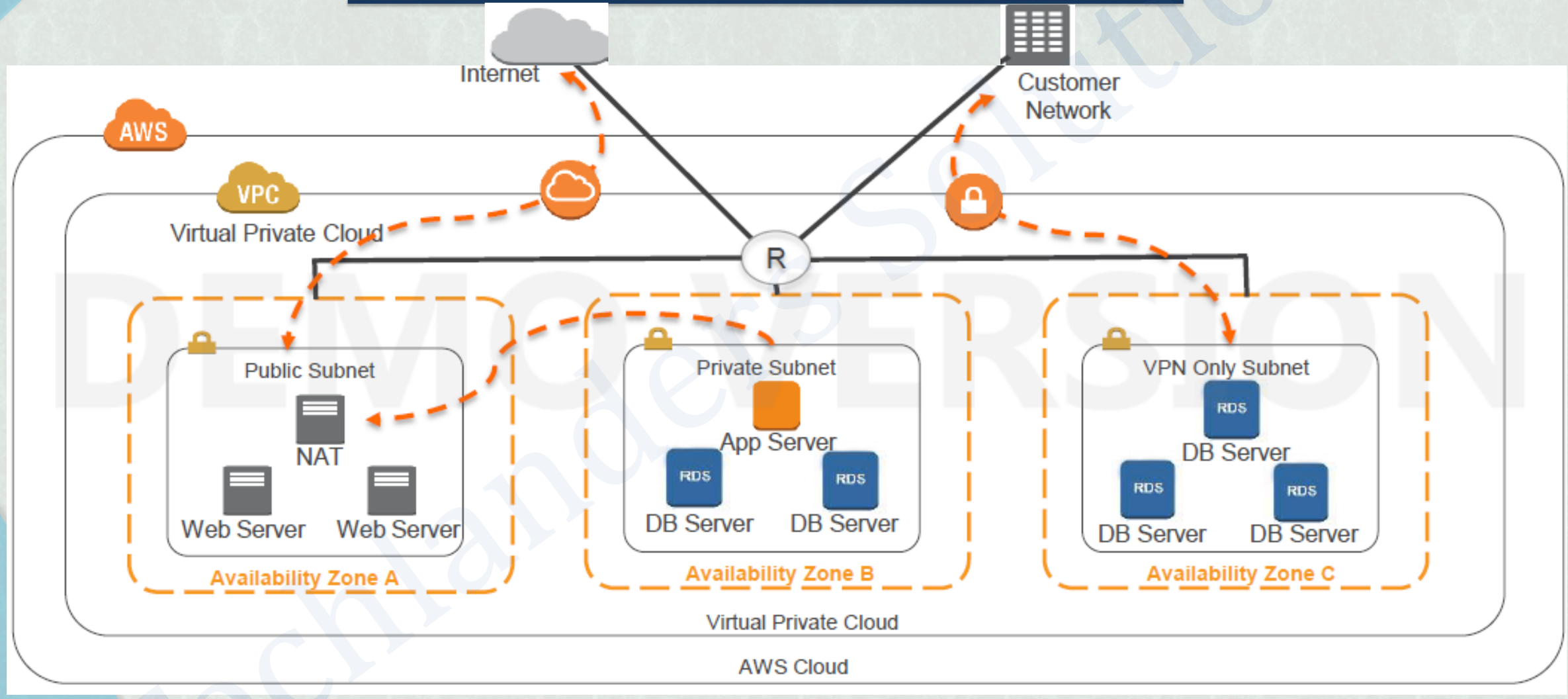
- A **subnet** defines a range of IP addresses in your VPC. A *subnet* is a segment of an Amazon VPC's IP address range where you can launch Amazon EC2 instances.

- 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.

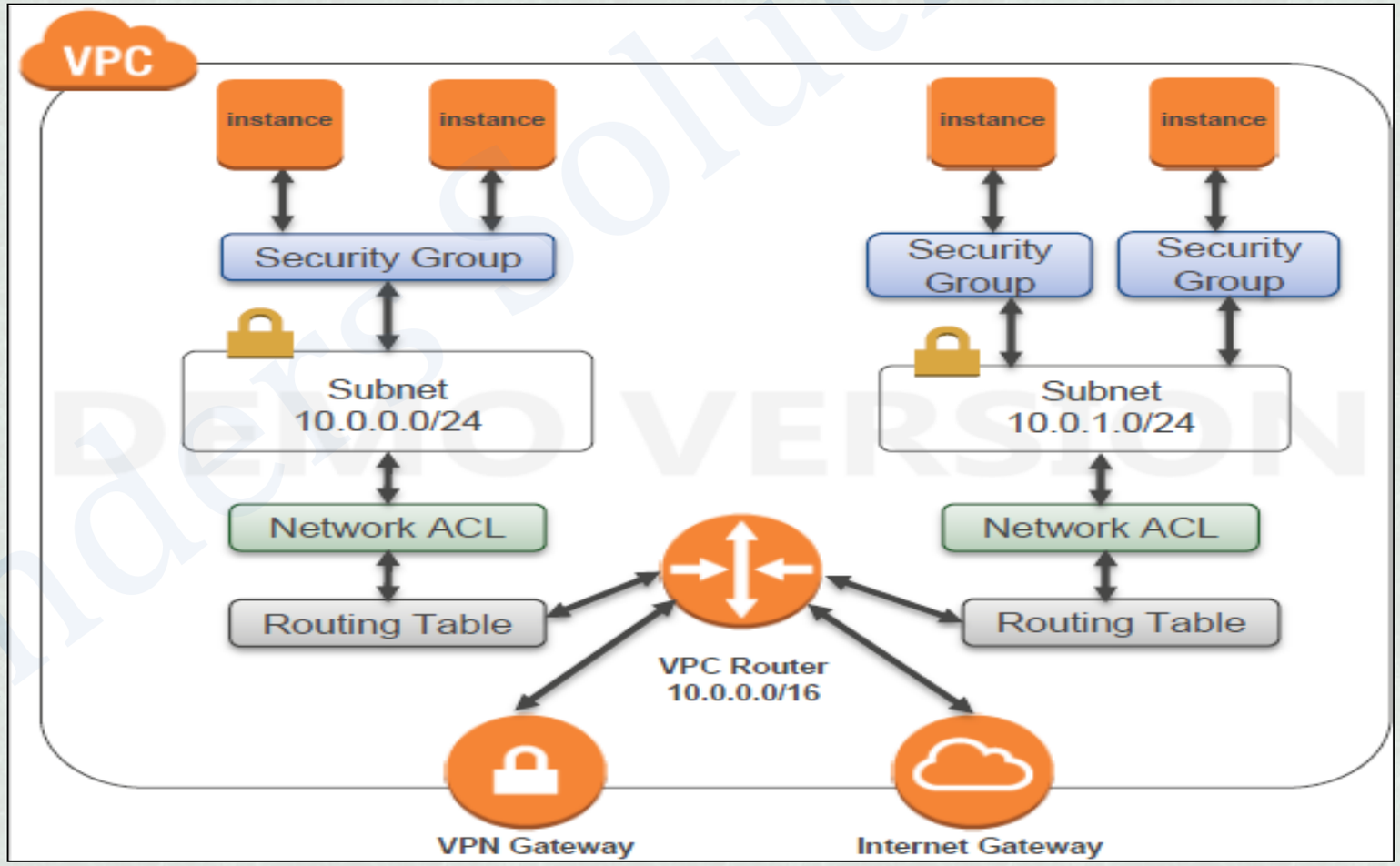
LAB 22 : Creating VPC

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

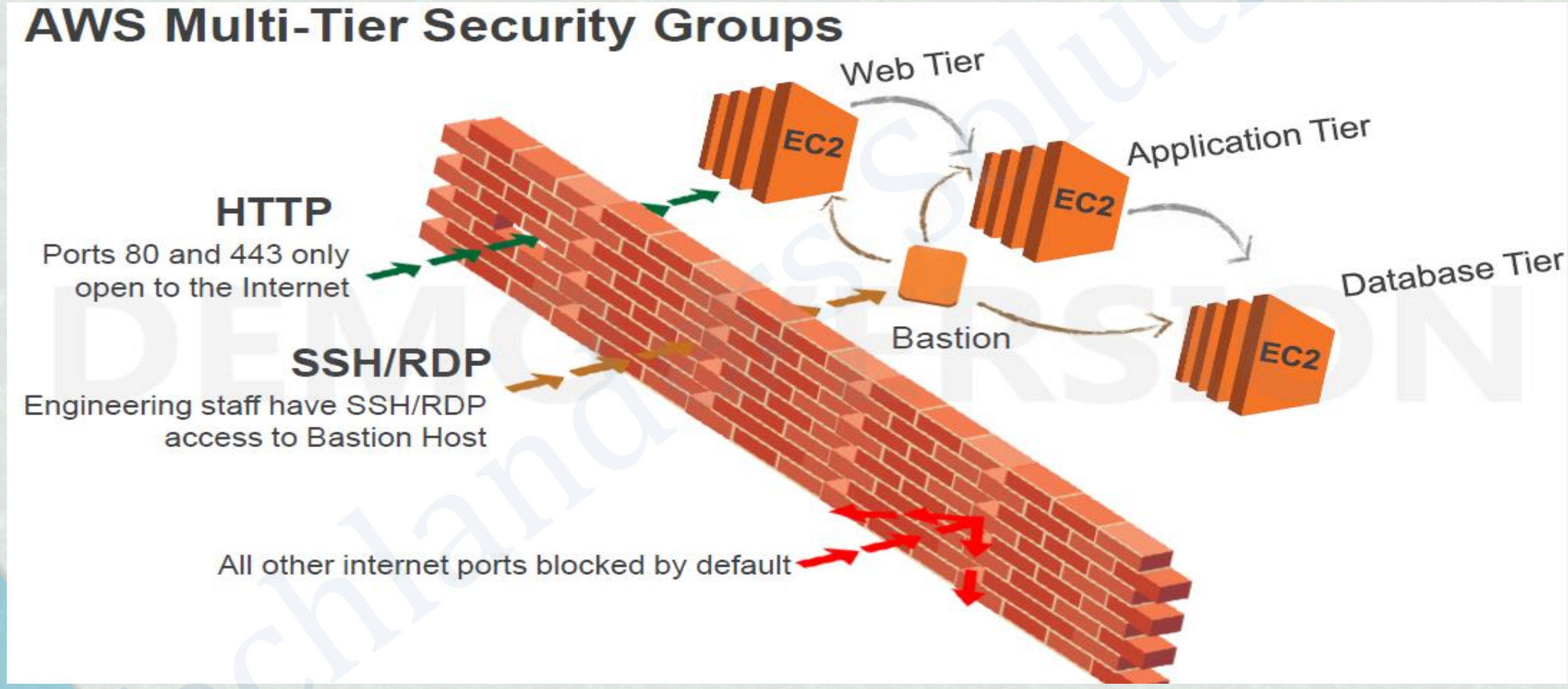
- Security Group
- Network ACL
- Routing Tables



AWS Security

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

Enhanced Security



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.

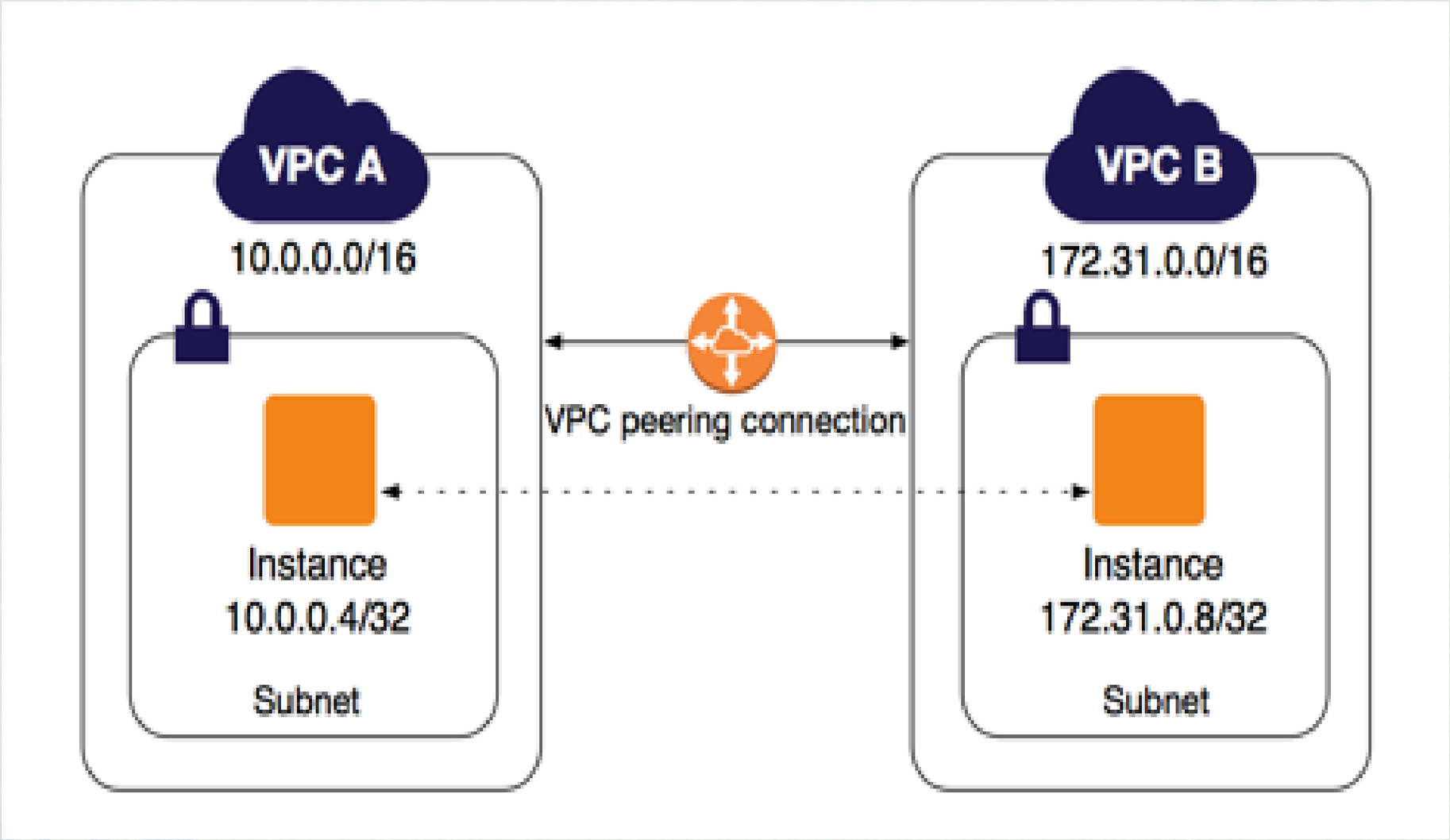
You can use private IPv4 addresses or IPv6 addresses to connect.

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

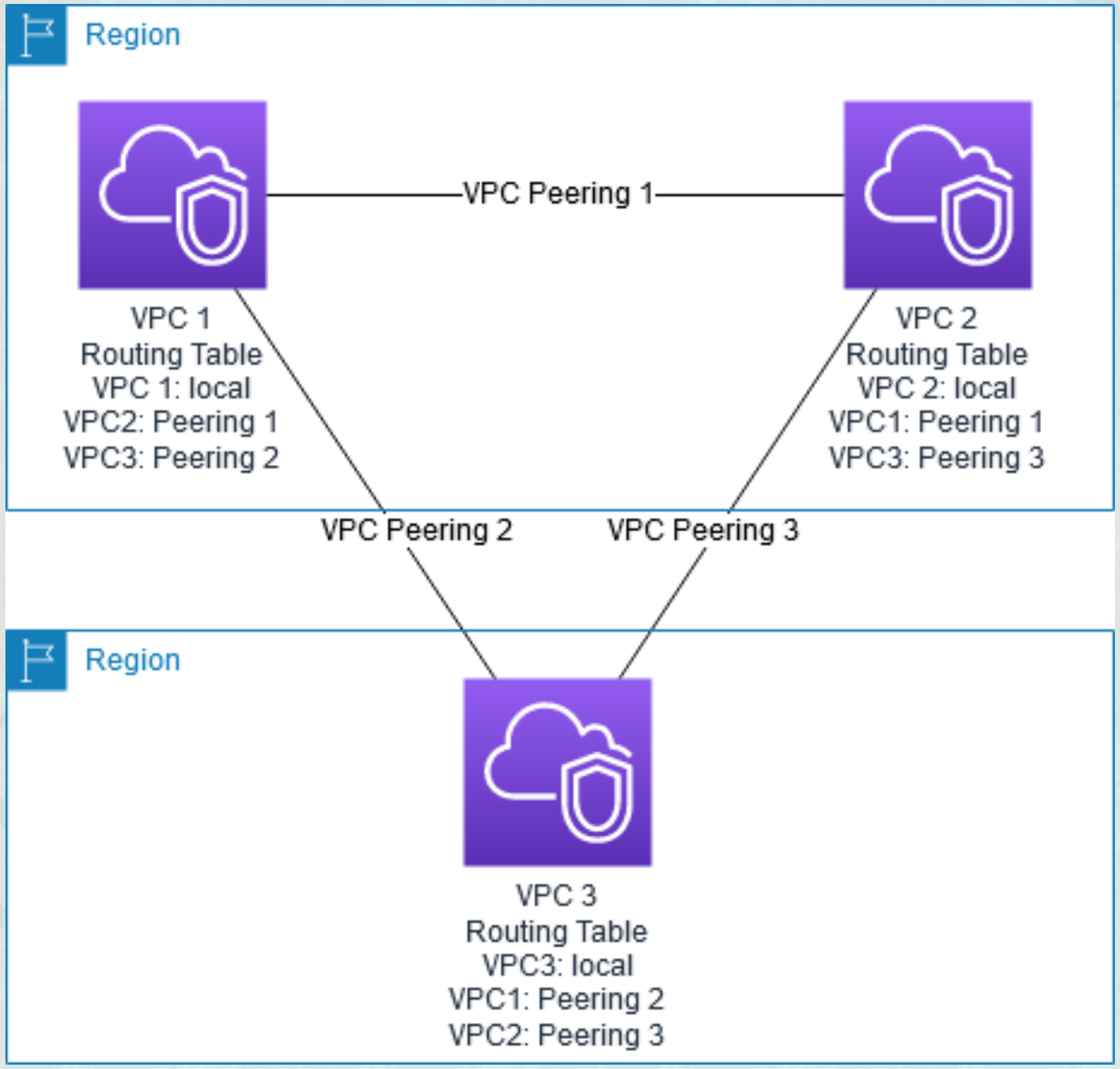
You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.

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

VPC Peering


Region
VPC Peering 1
VPC 1
Routing Table
VPC 1: local
VPC2: Peering 1
VPC3: Peering 2
VPC 2
Routing Table
VPC 2: local
VPC1: Peering 1
VPC3: Peering 3
VPC Peering 2
VPC Peering 3
Region
VPC 3
Routing Table
VPC3: local
VPC1: Peering 2
VPC2: Peering 3

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