CMD - WHOAMI

Lokesh Ava

Working in IT (MNC)

So what?

1) AWS Certified Solutions Architect Associate
2) AWS Certified Developer Associate
3) AWS Certified Sysops Administrator Associate

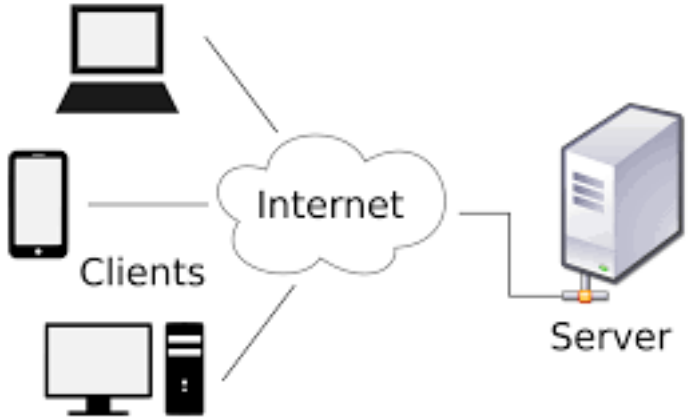
Also I'm an Instructor

Day – 1

- What we will learn in this course
- What is cloud
- Why we are moving into cloud
- Cloud Architecture
- Introduction to AWS
- List of services will cover in this course

What we will learn in this course

WHAT IS CLOUD



WHY WE ARE MOVING INTO CLOUD

Let's look at some of the most common reasons to use the cloud.

- File storage: You can store all types of information in the cloud, including files and email.

- File sharing: The cloud makes it easy to share files with several people at the same time.

- Backing up data: You can also use the cloud to protect your files.

Clients and Servers

Cloud Computing

Client devices are connected via the **Internet**

Servers

Cloud Networking

Client Devices

Client devices require connectivity via **wired**, **wireless** or **cellular** networks

Servers running in the cloud offer **services** which include the **application**, **processing** and **data storage**

Connecting to Services

The client application finds the server by **IP address**

Web Server

Port: 80 Protocol: HTTP

A **port** is like a door into the server

File Server

Port: 445 Protocol: SMB

SMB/CIFS is used by Microsoft file servers and clients

Port: 25 Protocol: SMTP

List of services will cover in this course

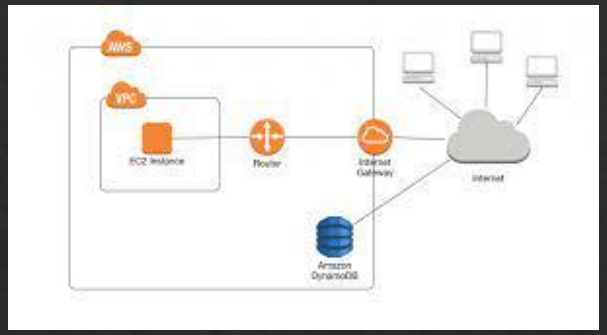
IAM

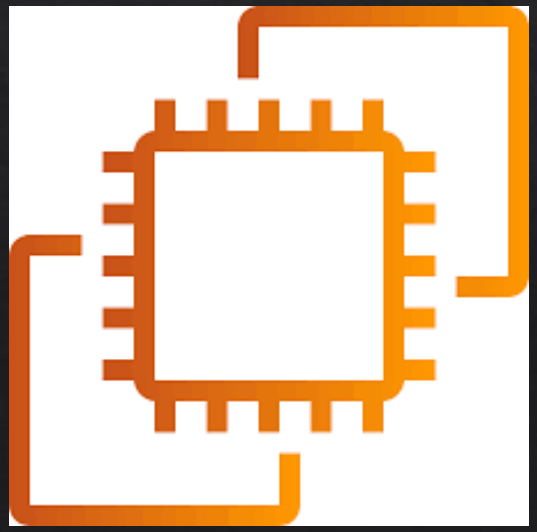
VPC

EC2

RDS

Hosting a Web Application in AWS





How IAM Works

IAM

IAM Principals must be **authenticated** to send requests (with a few exceptions)

AWS Account

AWS IAM

User | Role | Federated User | Application

Console

CLI

API

Request context:

- Actions / operations
- Resources
- Principal
- Environment data
- Resource data

A principal is a person or application that can make a request for an **action** or **operation** on an AWS resource

Identity-based policy | Resource-based policy

AWS determines whether to **authorize** the request (allow/deny)

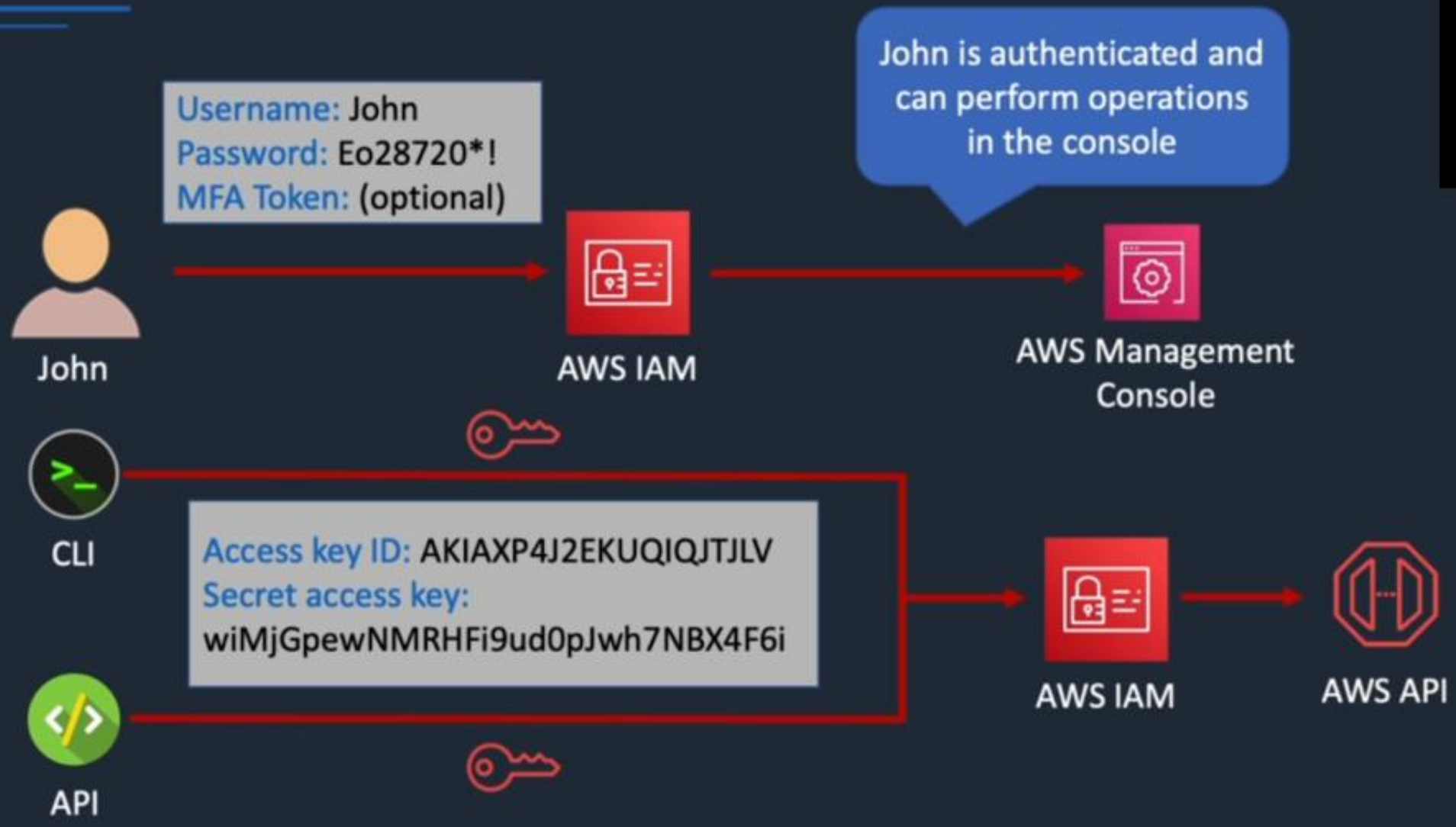
RunInstances → EC2

GetBucket → S3

CreateUser → IAM

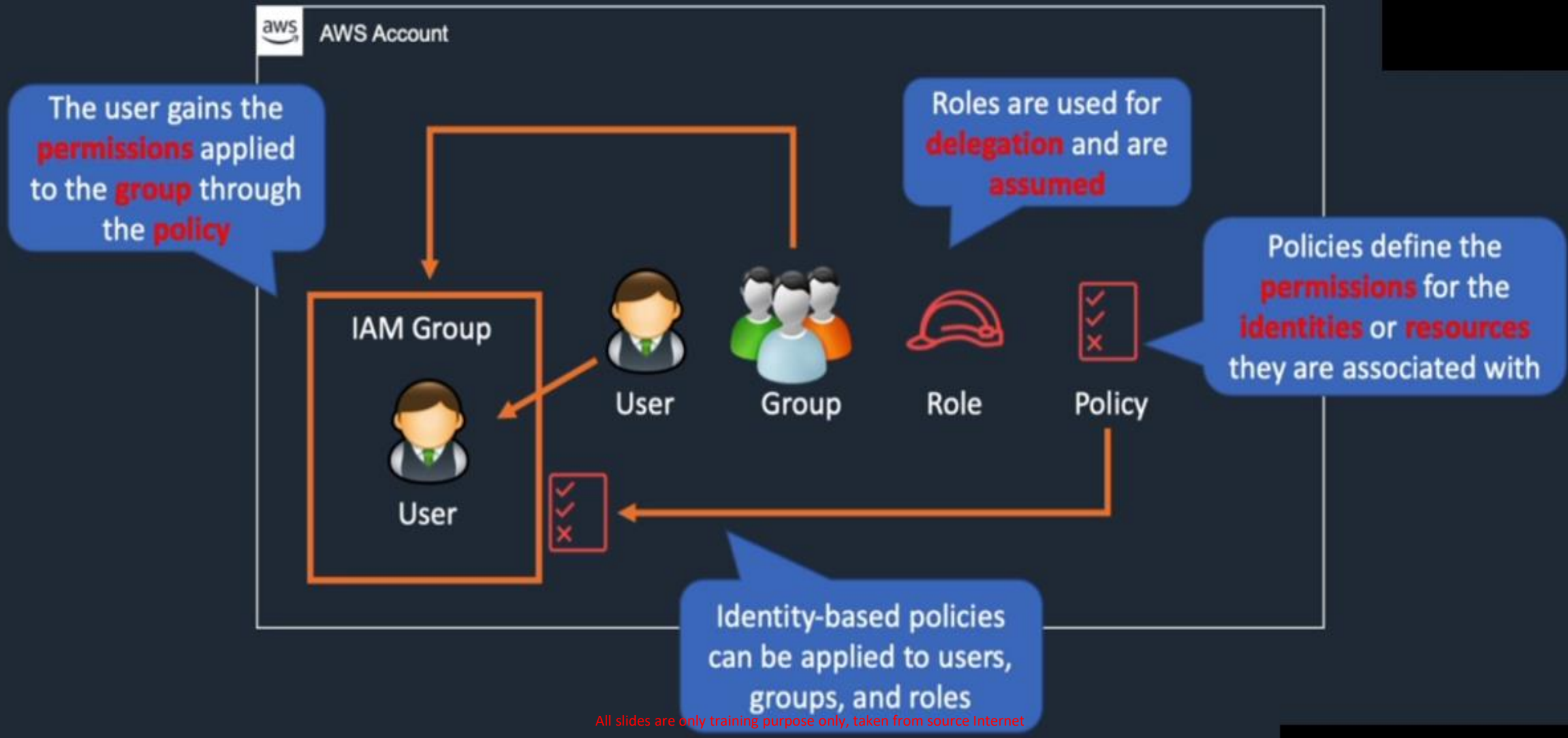
Actions are **authorized** on AWS **resources**

IAM Authentication Methods

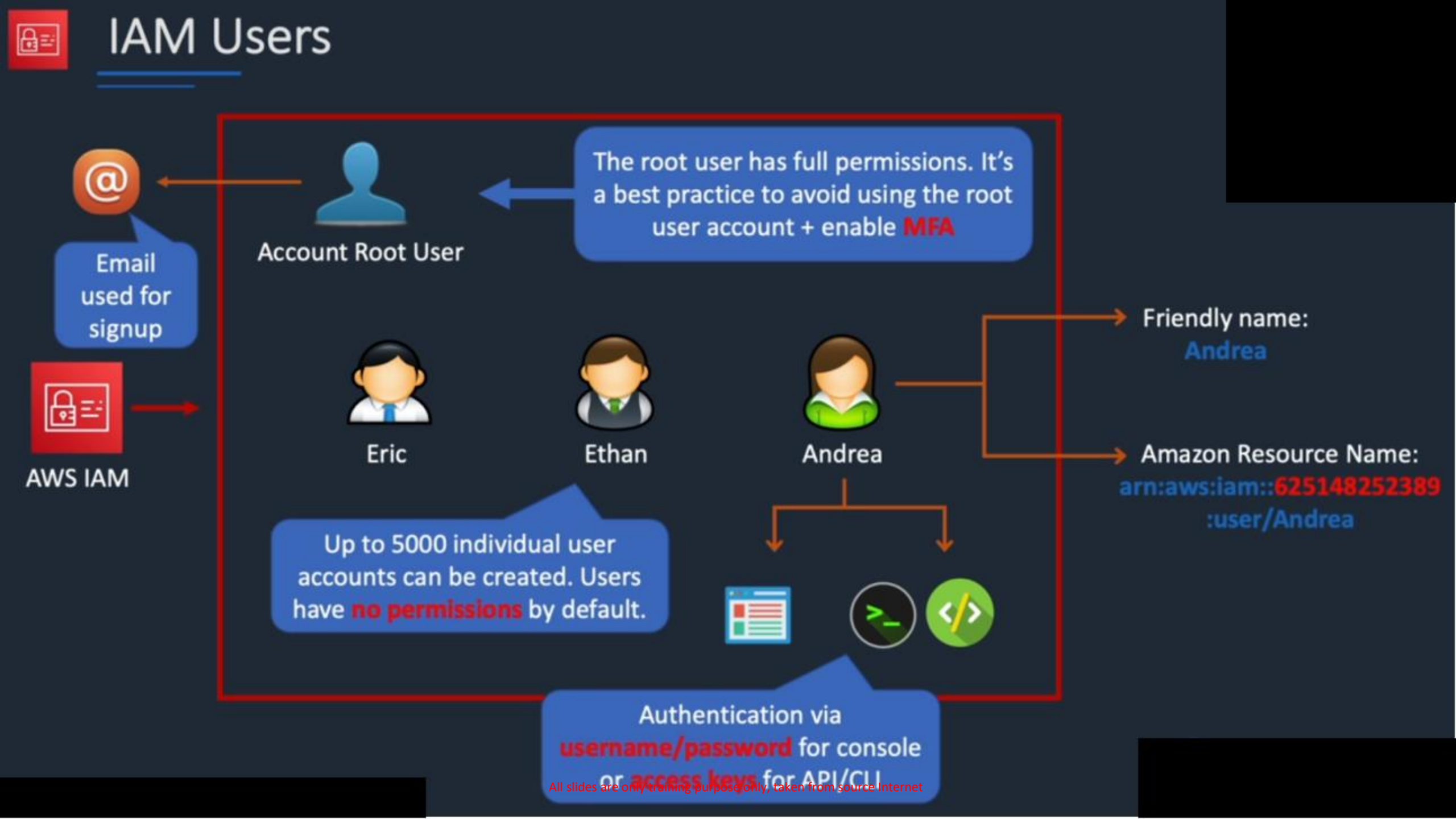


Users, Groups, Roles and Policies

IAM



IAM Users



IAM Policies

Policies are documents that define permissions and are written in JSON

AdministratorAccess

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

All permissions are implicitly denied by default

User

Group

Role

Identity-based policies can be applied to users, groups, and roles

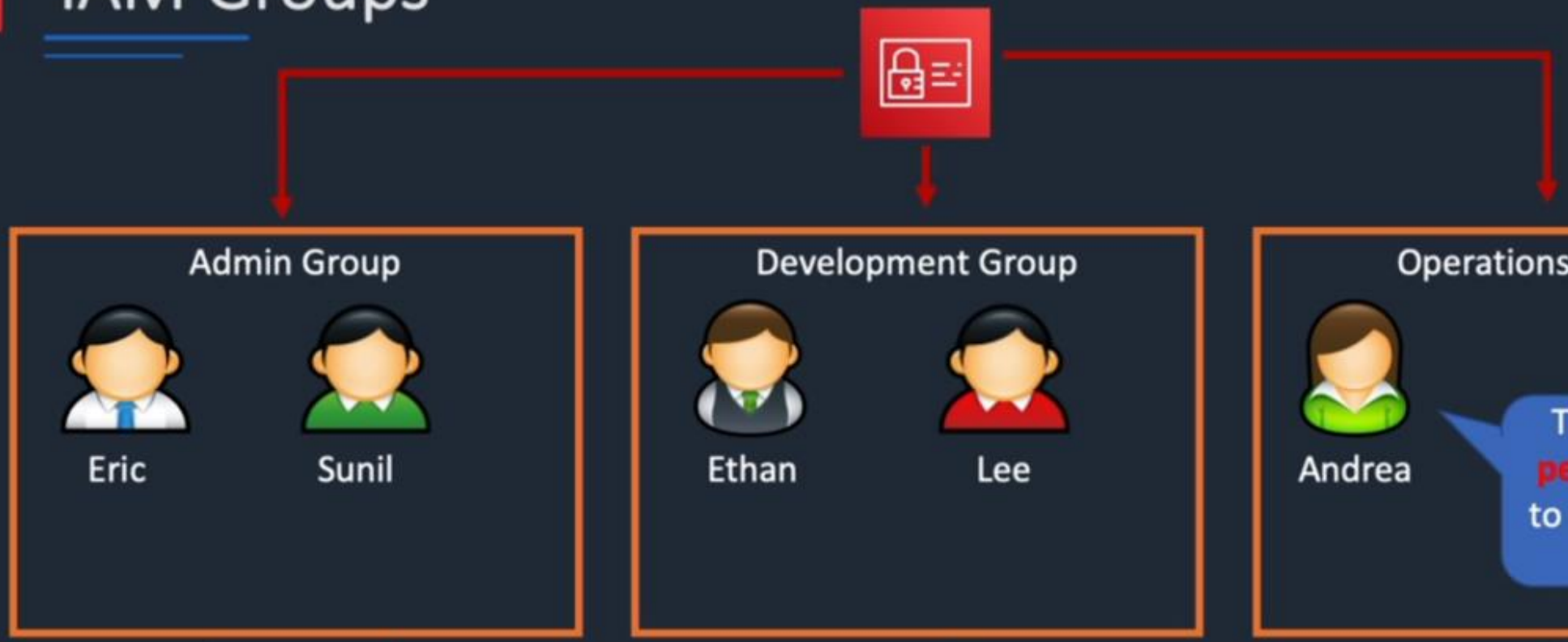
Bucket Policy

```
{
    "Version": "2012-10-17",
    "Id": "Policy1561964929358",
    "Statement":[
    {
        "Sid": "Stmt1561964454852",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::515148227241:user/Paul"
        },
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::dctcompany",
        "Condition": {
            "StringLike": {
                "s3:prefix": "Confidential/*"
            }
        }
    }
    ]
}
```

Resource-based policies apply to resources such as S3 buckets or DynamoDB tables

S3 Bucket

IAM Groups



Bandwidth and Latency

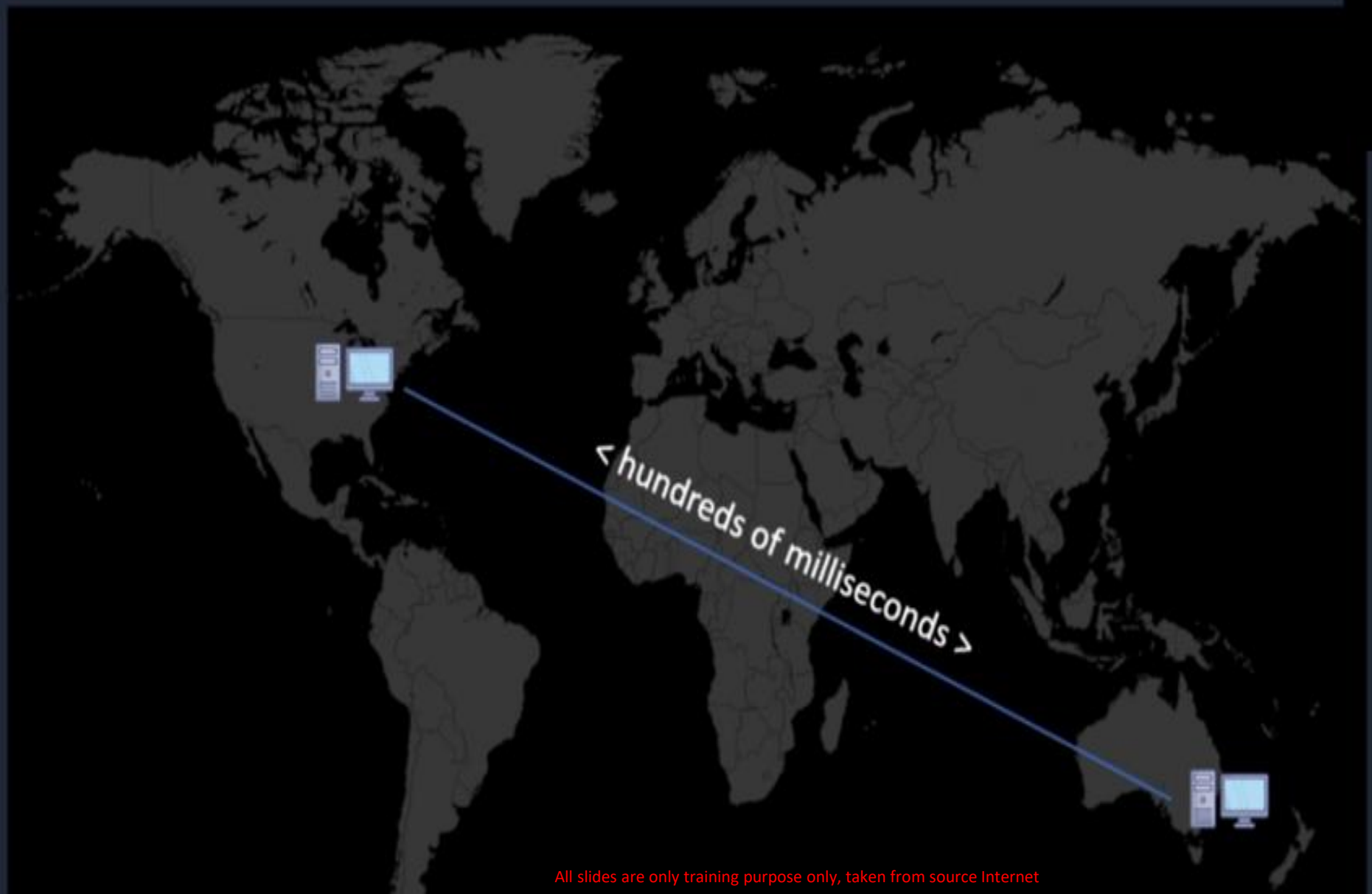
Bandwidth is the **rate** of data transfer for a **fixed period of time** measured in **Gbps**

Bandwidth can be considered the **width** of the communication band

Latency is the amount of time it takes to send data from one point to another measured in **microseconds** or **milliseconds**

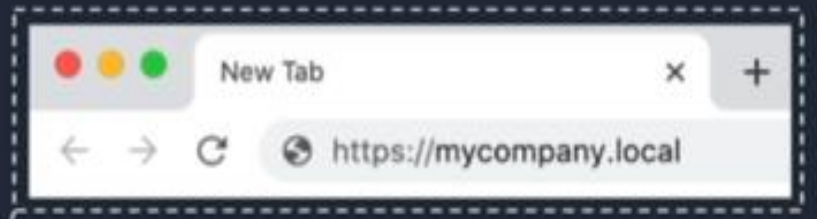
Bandwidth and Latency

Bandwidth and Latency



IP Addressing Basics

User enters website address in browser



Name	Type	Value
mycompany.local	A	192.168.0.1
emailserver.local	A	192.168.0.2

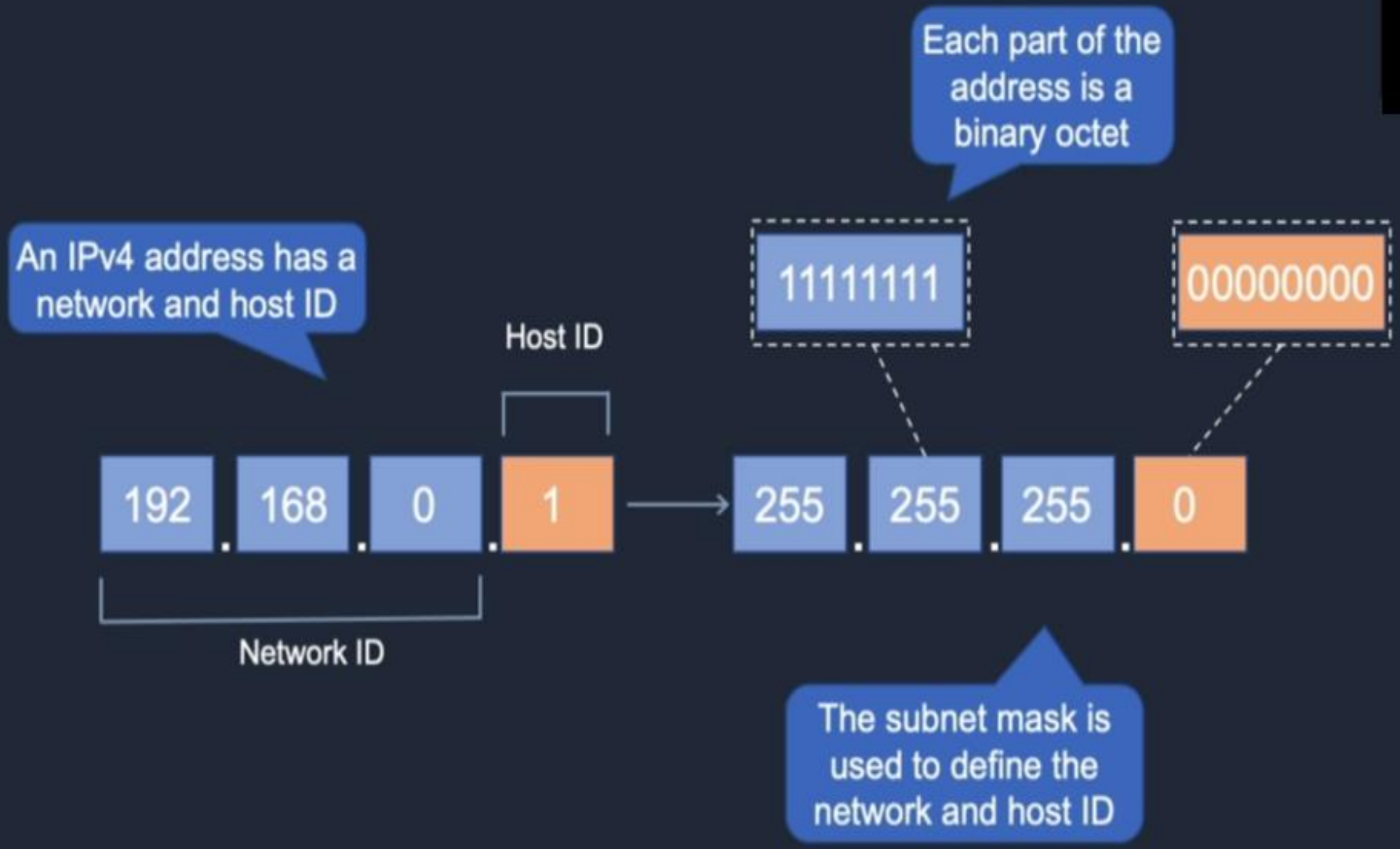
DNS Server

Domain name is resolved to the IP address of the webserver

Web Server

Computer connects to 192.168.0.1

IP Addressing Basics



IP Addressing Basics

A **network** and **subnet mask** can also be written in this format (**CIDR** notation)

= 192.168.0.0/24

IP Addressing Basics

Private IP Addresses

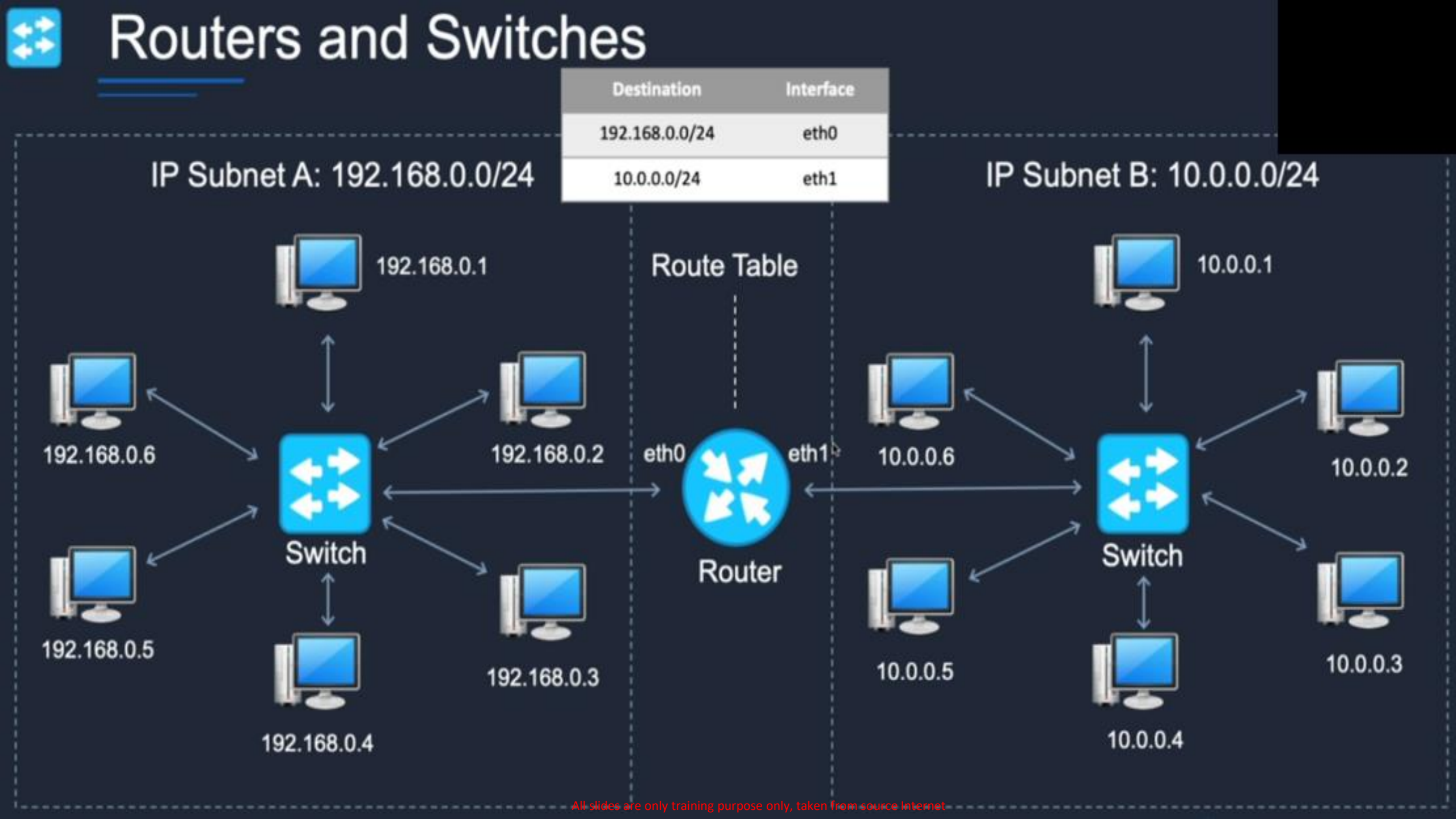
- There are several ranges of addresses reserved for **private usage** as defined in **RFC 1918**
- These are:

CIDR	First Address	Last Address
10.0.0.0/8	10.0.0.0	10.255.255.255
172.16.0.0/12	172.16.0.0	172.31.255.255
192.168.0.0/16	192.168.0.0	192.168.255.255

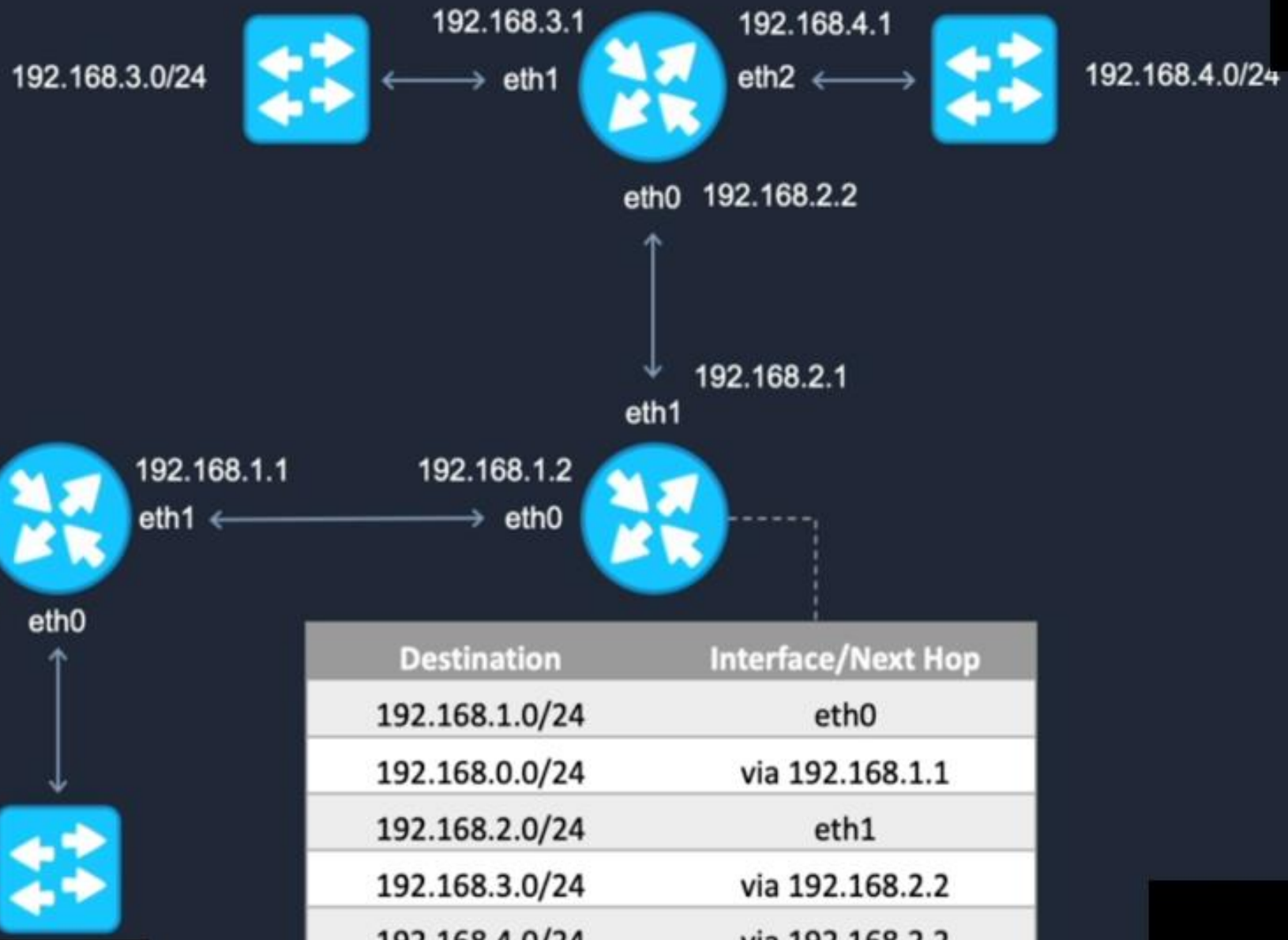
- Private addresses are **NOT** routable on the Internet

Routers and Switches

Destination	Interface
192.168.0.0/24	eth0
10.0.0.0/24	eth1

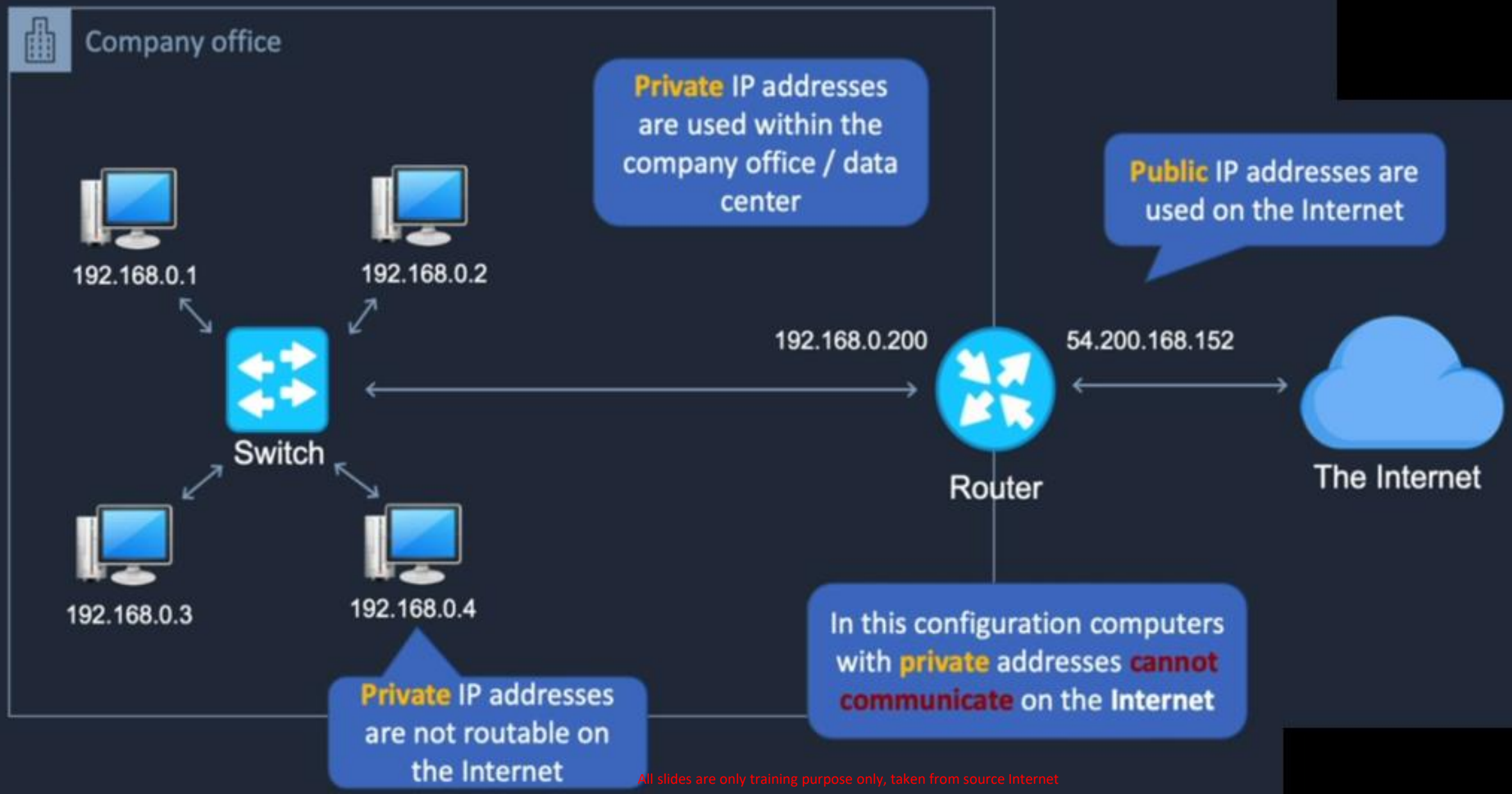


Route Tables

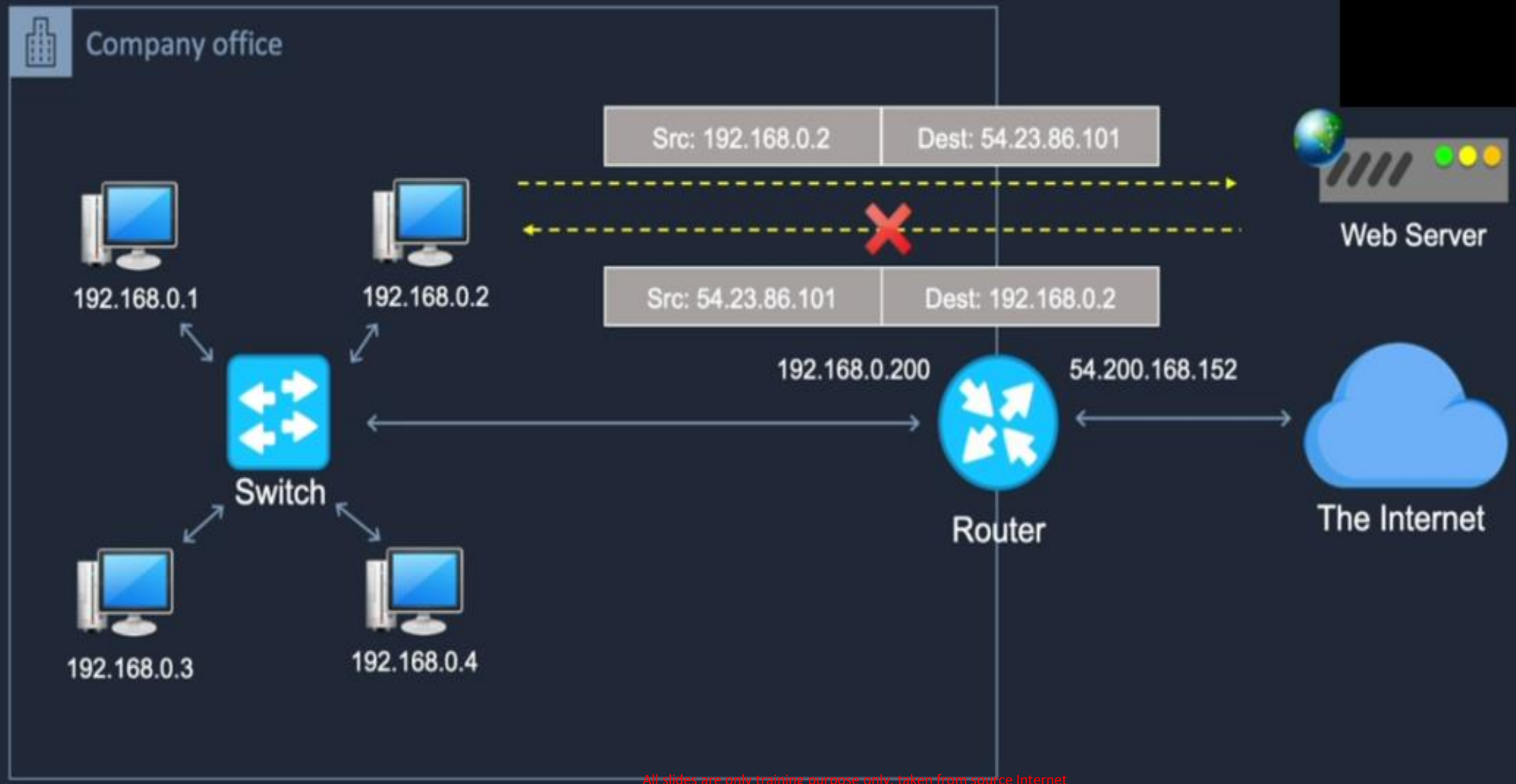


Destination	Interface/Next Hop
192.168.1.0/24	eth0
192.168.0.0/24	via 192.168.1.1
192.168.2.0/24	eth1
192.168.3.0/24	via 192.168.2.2
192.168.4.0/24	via 192.168.2.2

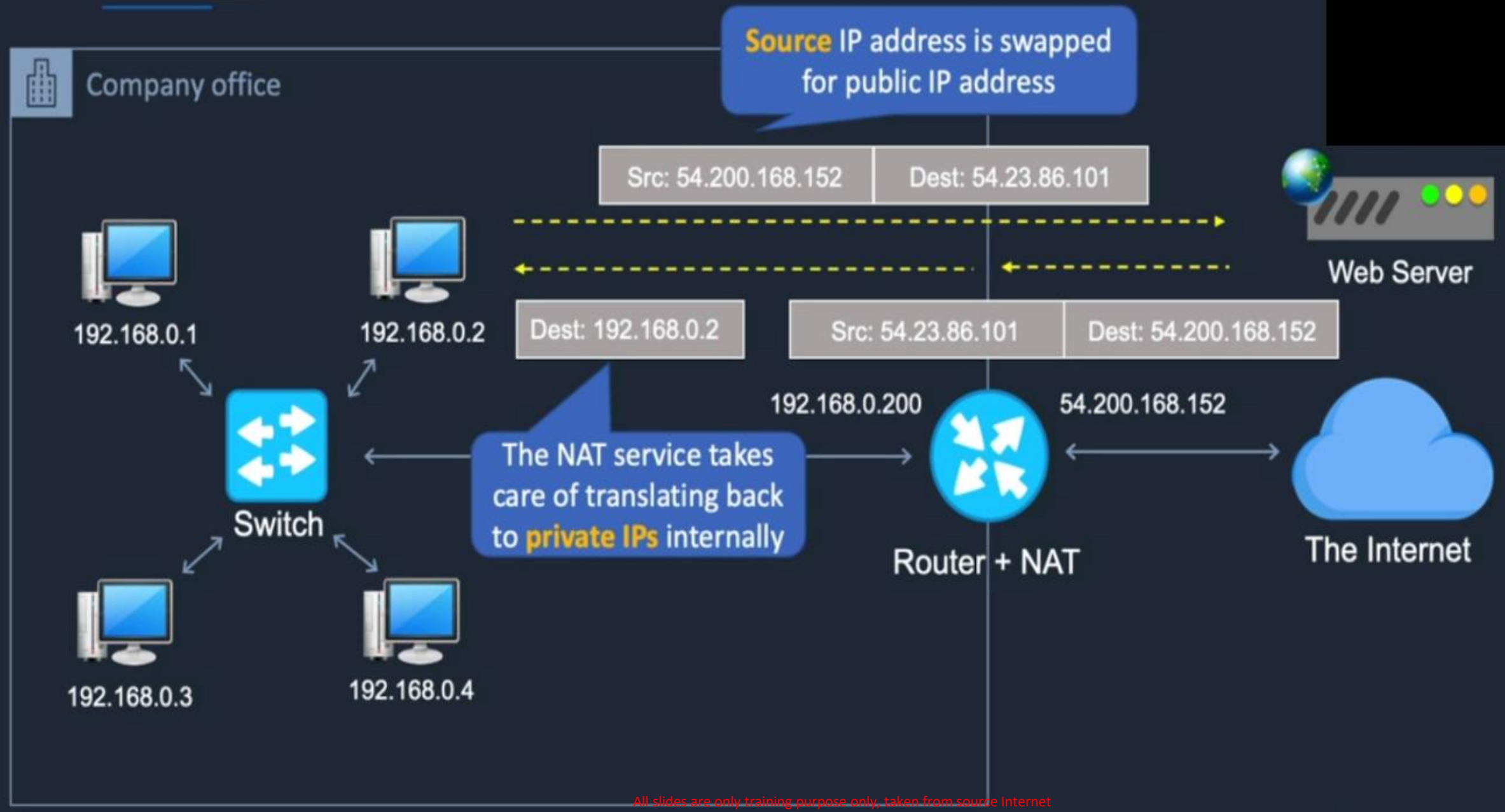
Without Network Address Translation (NAT)



Without Network Address Translation (NAT)



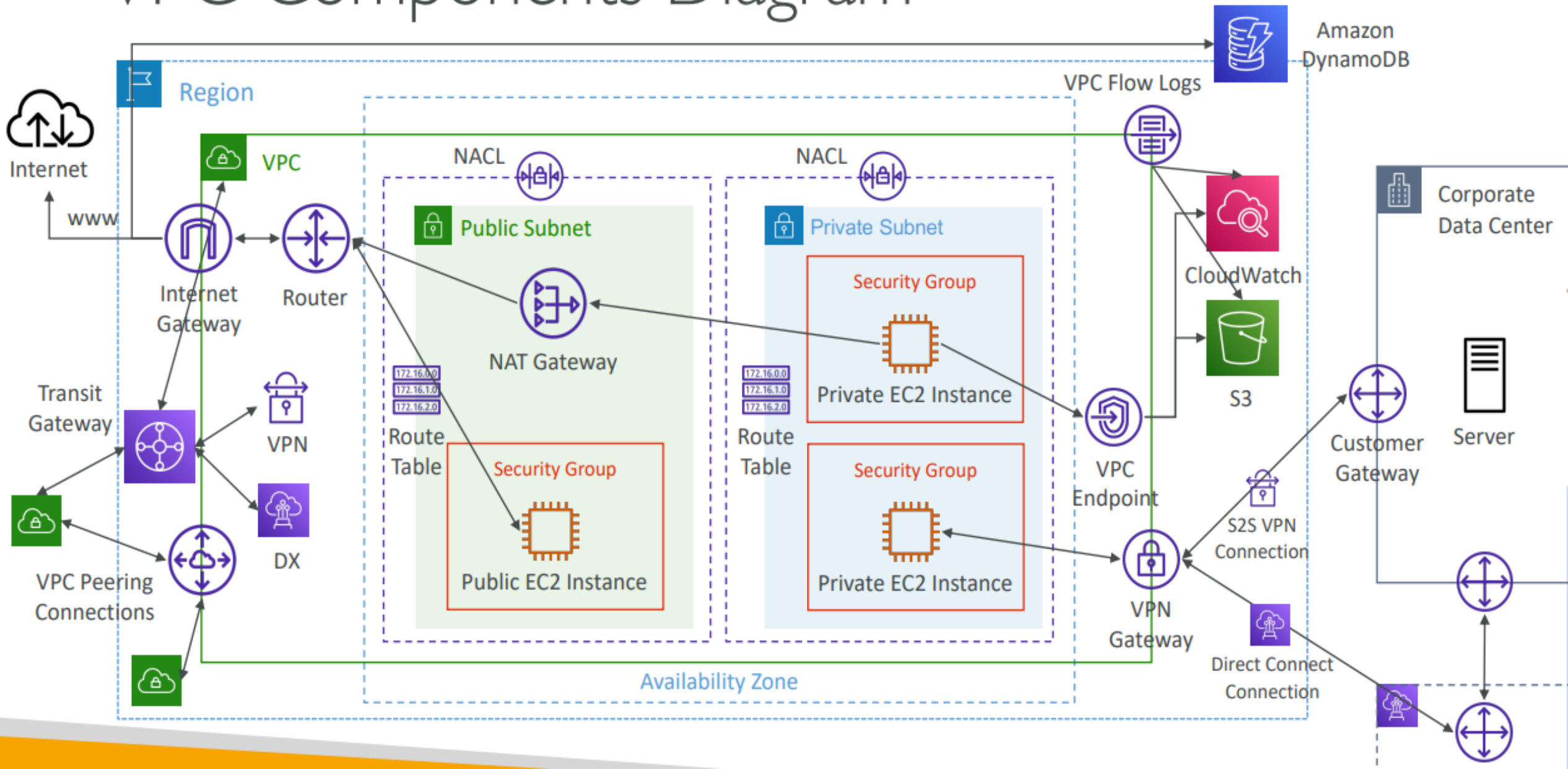
With Network Address Translation (NAT)



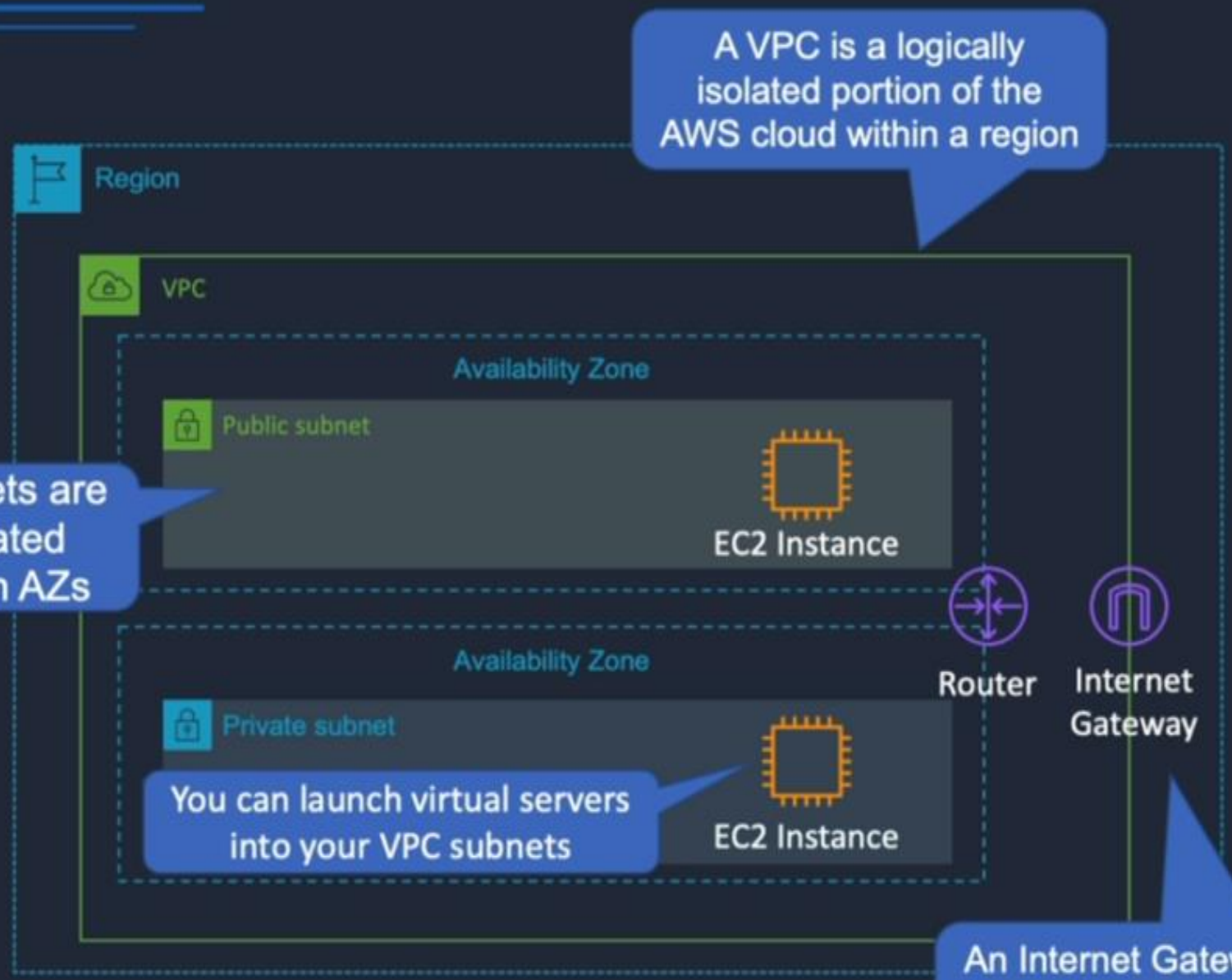
VPC

- No additional charge
- There are charges for some VPC components, such as NAT gateways, Reachability Analyzer, and traffic mirroring.
- Specify an IP address range for the VPC, add subnets, associate security groups, and configure route tables.
- A *route table* contains a set of rules, called routes, that are used to determine where network traffic from your VPC is directed.
- NAT maps multiple private IPv4 addresses to a single public IPv4 address.
- You can create a *VPC peering connection* between two VPCs that enables you to route traffic between them privately.

VPC Components Diagram



Amazon VPC Overview

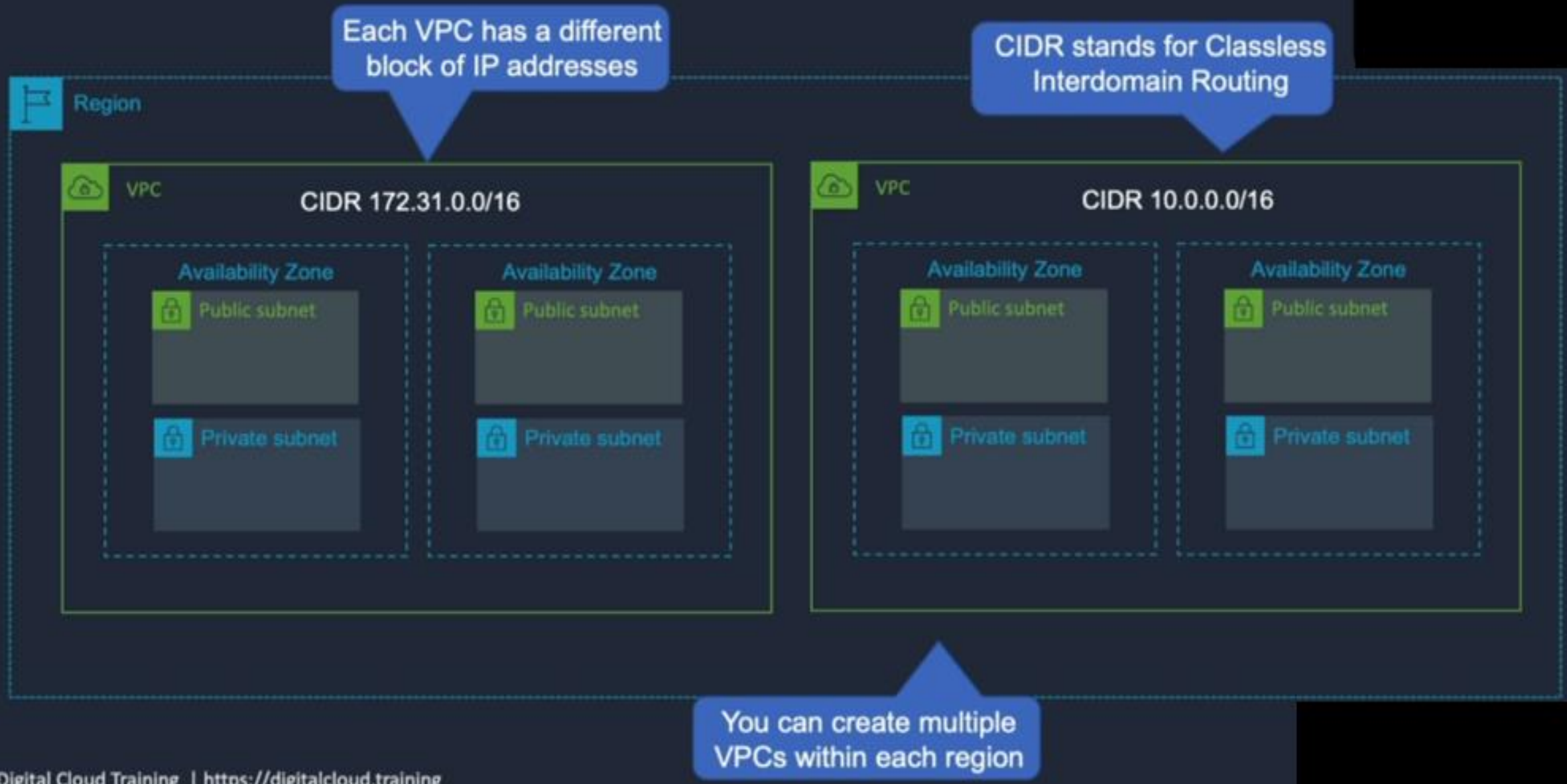


Main Route Table

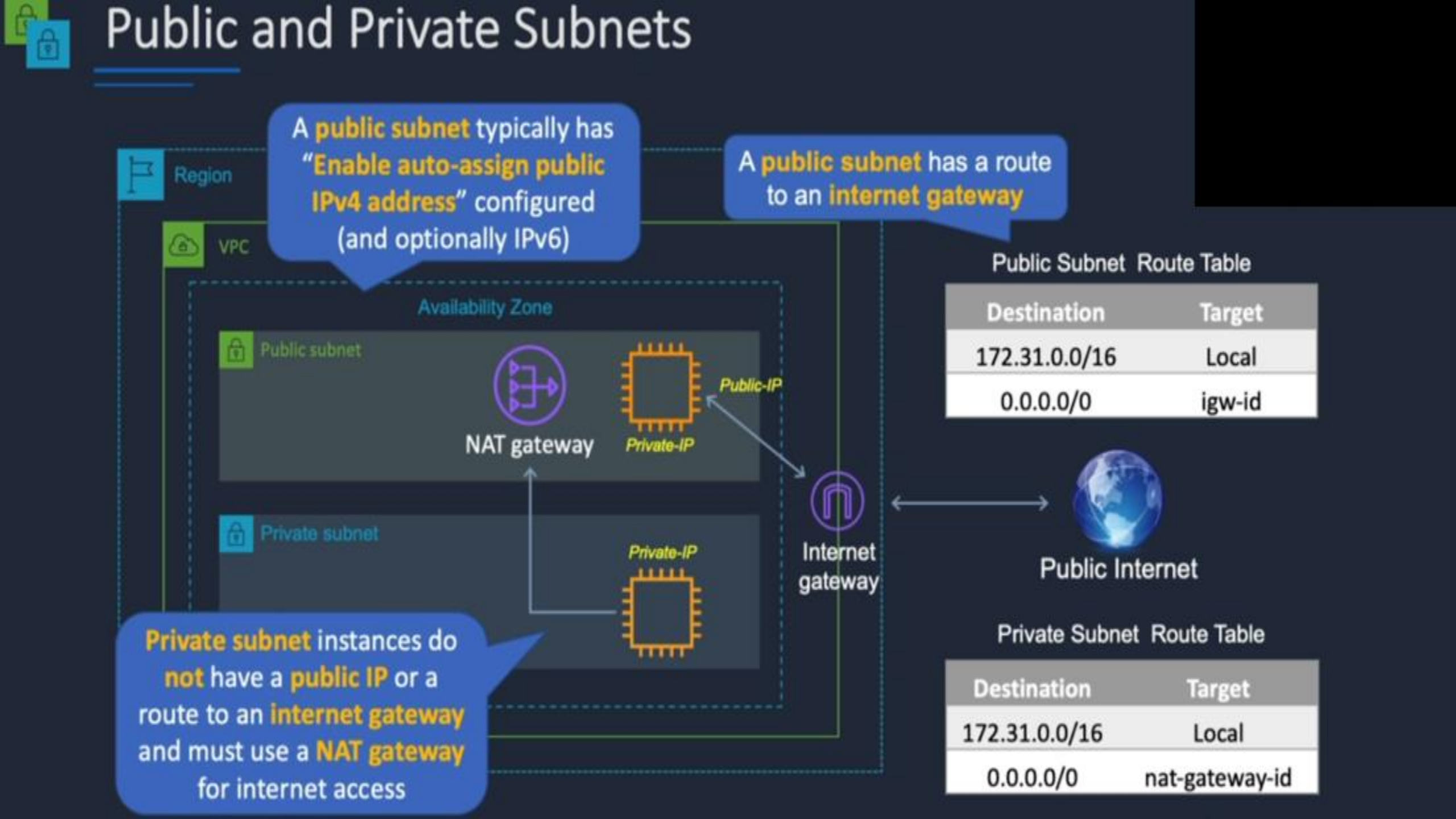
Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	igw-id

The route table is used to configure the VPC router

Amazon VPC Overview



Public and Private Subnets

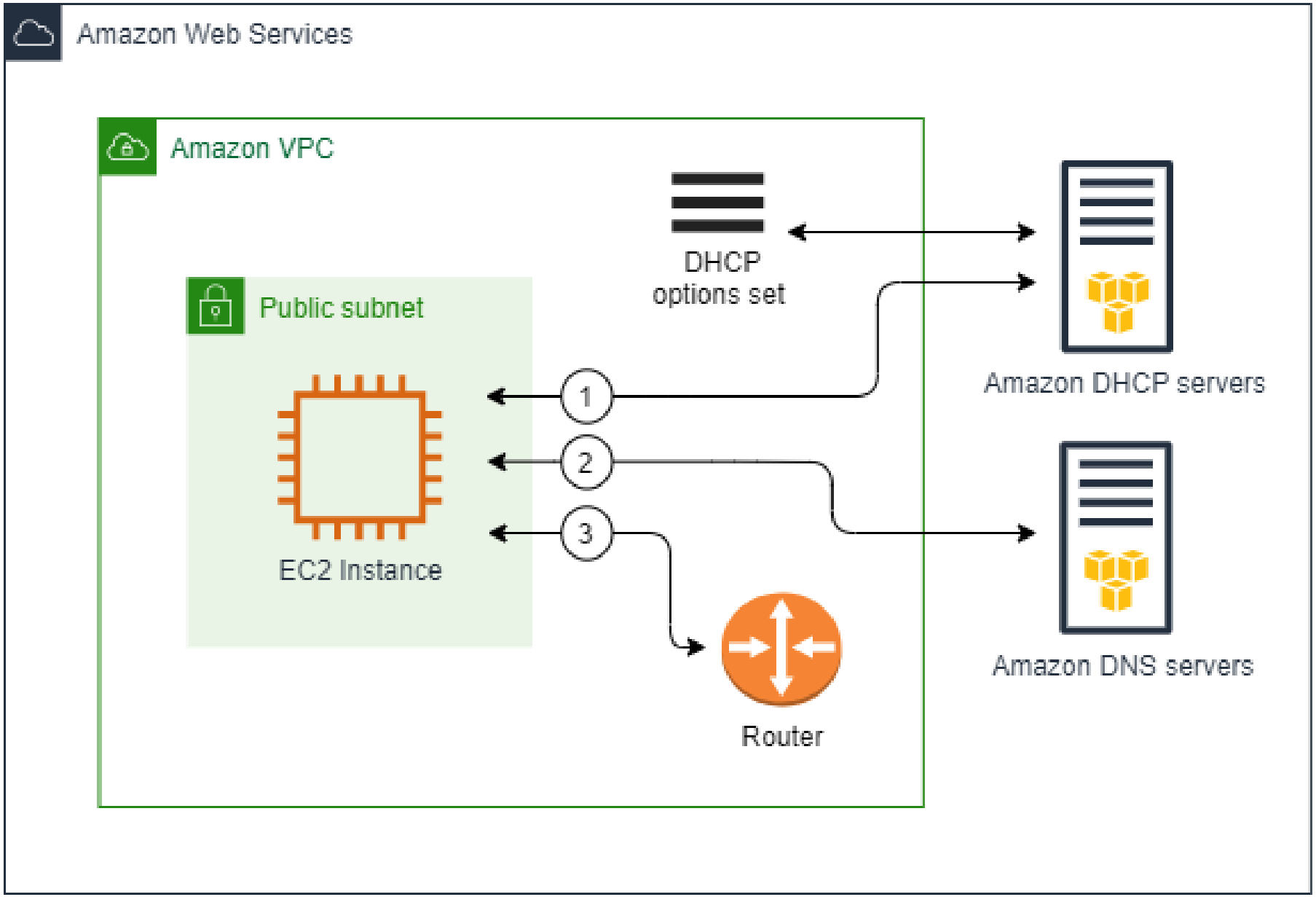


Internet Gateway (IGW)

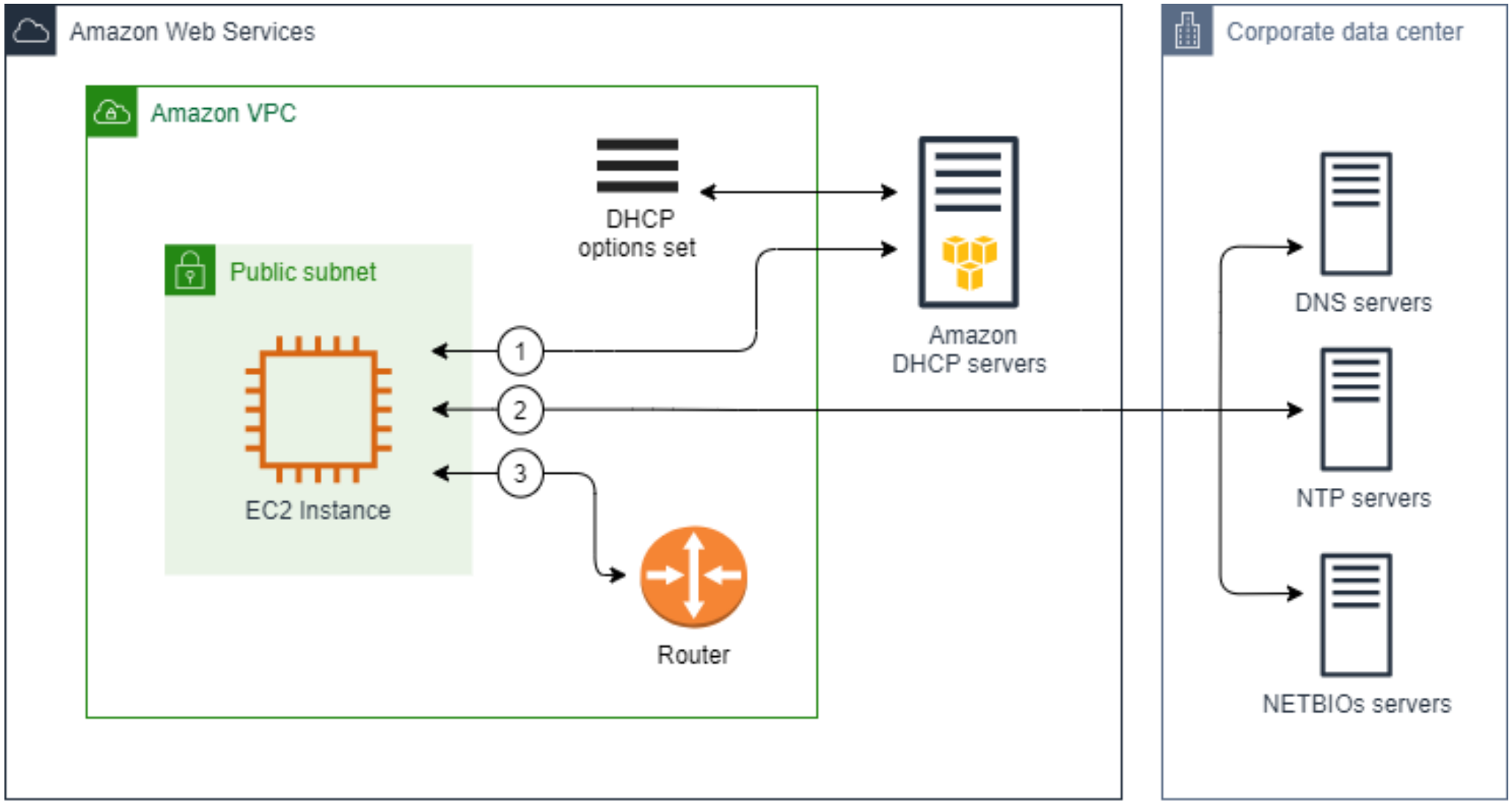
- Allows resources (e.g., EC2 instances) in a VPC connect to the Internet
- It scales horizontally and is highly available and redundant
- Must be created separately from a VPC
- One VPC can only be attached to one IGW and vice versa

- Internet Gateways on their own do not allow Internet access…
- Route tables must also be edited!

If you use the default option set,

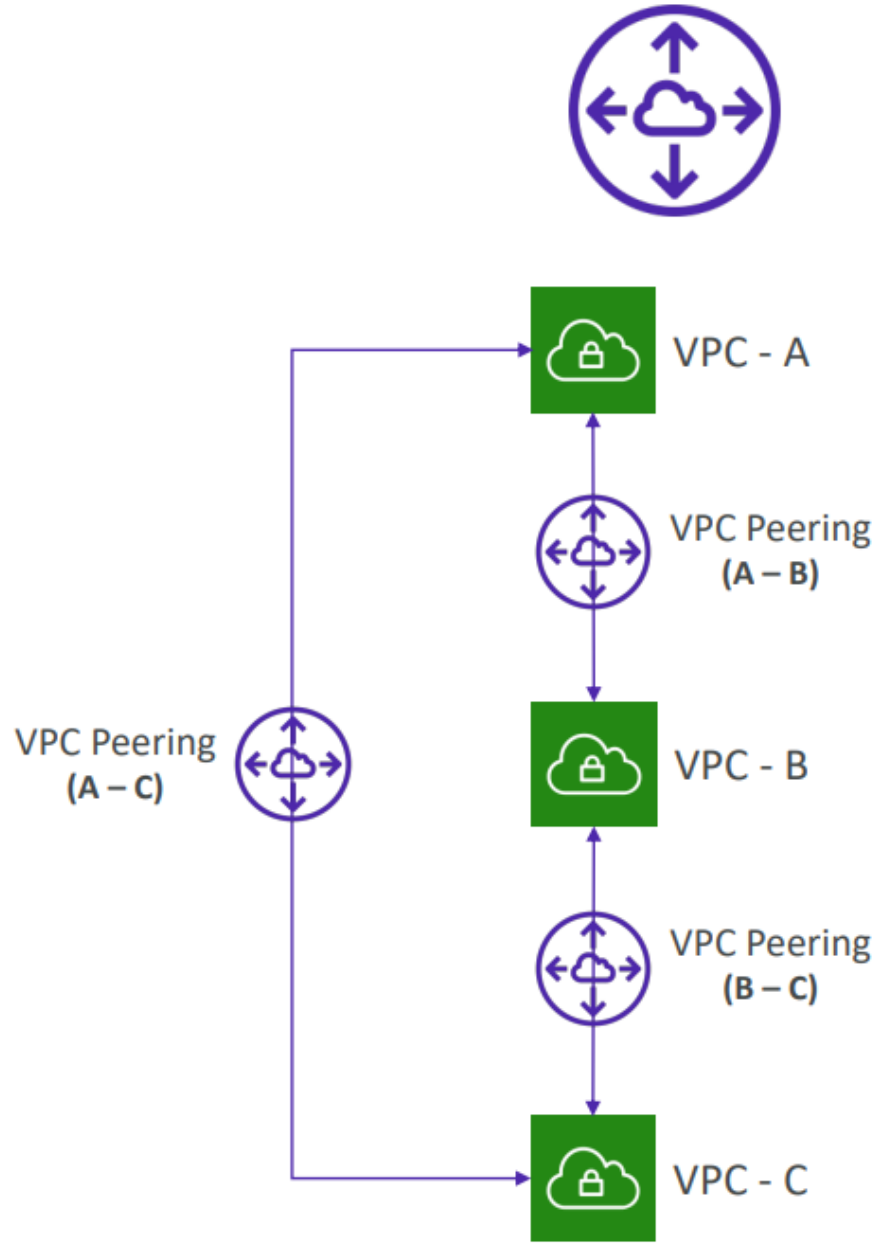


If you use a custom option set,



VPC Peering

- Privately connect two VPCs using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDRs
- VPC Peering connection is **NOT transitive** (must be established for each VPC that need to communicate with one another)
- **You must update route tables in each VPC's subnets to ensure EC2 instances can communicate with each other**



Firewalls

POLICY	PROTOCOL	PORT	DESTINATION	SOURCE
ALLOW	HTTP	80	INTERNAL	ANY
ALLOW	HTTPS	443	INTERNAL	ANY
DENY	ANY	ANY	INTERNAL	ANY

Firewall Rules

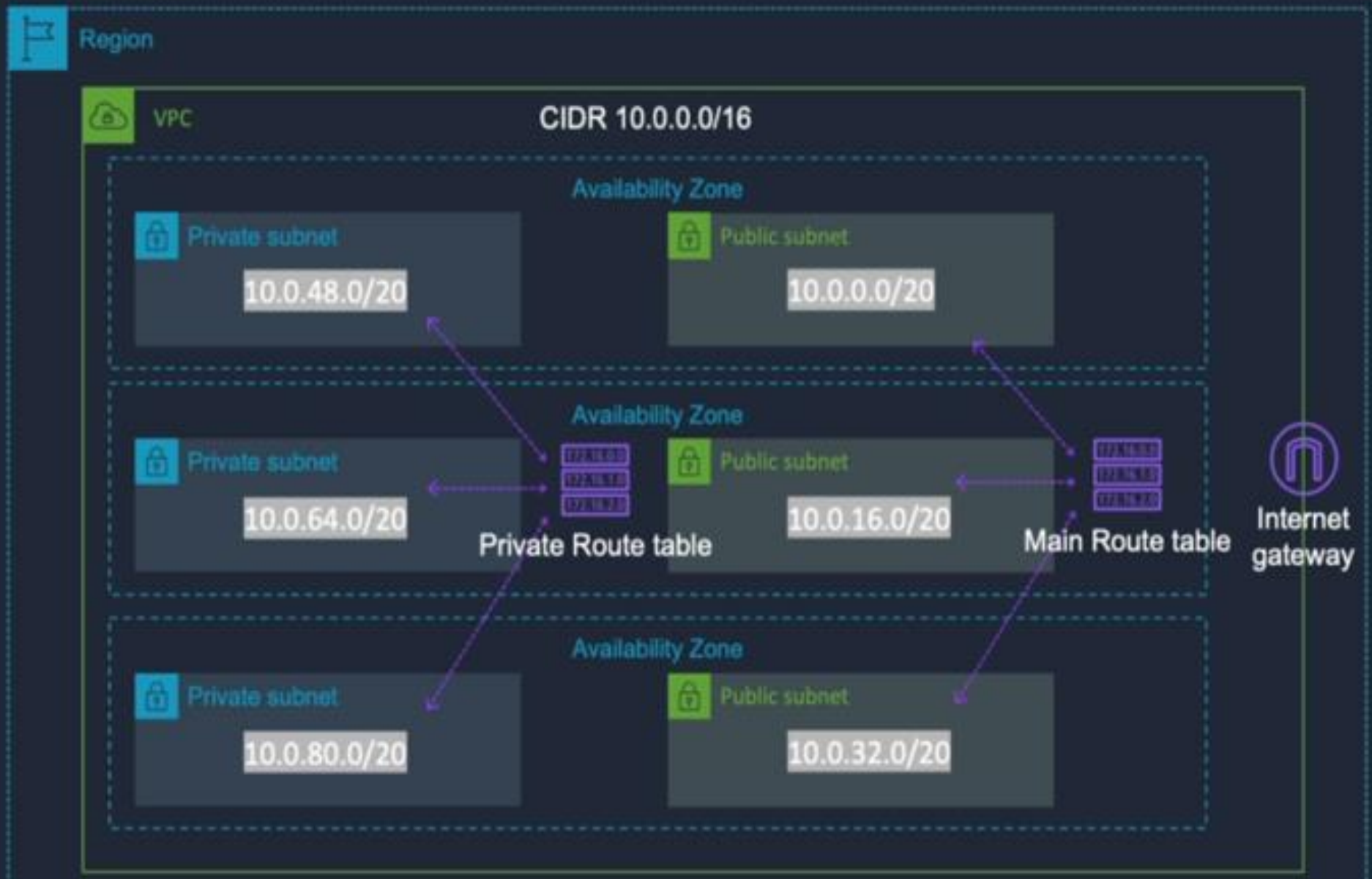
Stateful vs Stateless Firewalls

PROTOCOL	SOURCE IP	DESTINATION IP	SOURCE PORT	DESTINATION PORT
HTTP	10.1.1.1	10.2.1.10	65188	80
HTTP	10.2.1.10	10.1.1.1	80	65188

A stateful firewall allows the return traffic automatically

A stateless firewall checks for an allow rule for both connections

Create a Custom VPC



Main Route Table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

Private Route Table

Destination	Target
10.0.0.0/16	Local

What is EC2?

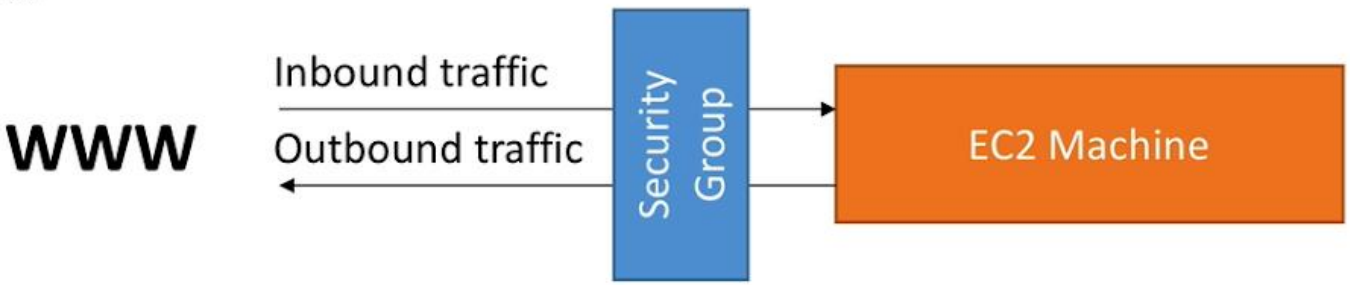
- EC2 is one of most popular of AWS offering
- It mainly consists in the capability of :
  - Renting virtual machines (EC2)
  - Storing data on virtual drives (EBS)
  - Distributing load across machines (ELB)
  - Scaling the services using an auto-scaling group (ASG)

Amazon EC2

- Knowing EC2 is fundamental to understand how the Cloud works

Introduction to Security Groups

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



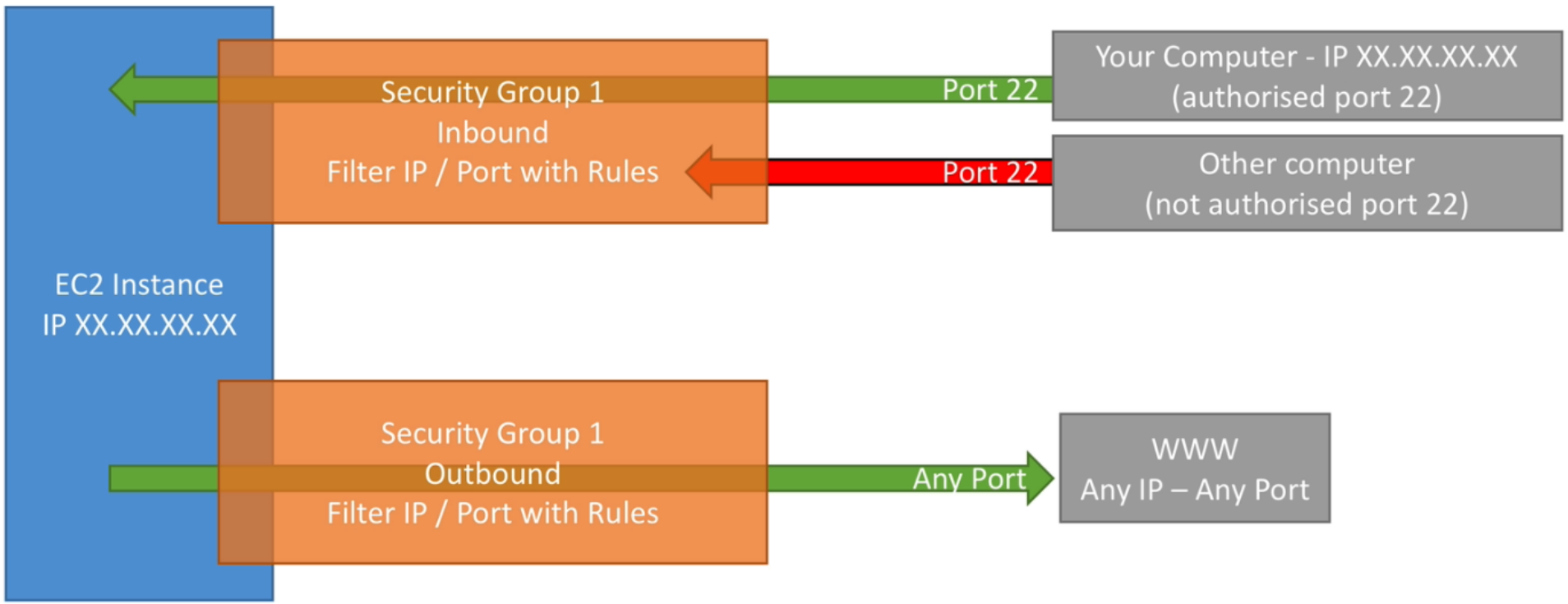
- It is the most fundamental skill to learn to troubleshoot networking issues
- In this lecture, we'll learn how to use them to allow or deny inbound and outbound ports

Security Groups Overview

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

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

Security Groups Diagram



What's an AMI?

- As we saw, AWS comes with base images such as:
  - Ubuntu
  - Fedora
  - RedHat
  - Windows
  - Etc…
- These images can be customised at runtime using EC2 User data

- But what if we could create our own image, ready to go?
- That's an AMI – an image to use to create our instances
- AMIs can be built for Linux or Windows machines

AMI Pricing

- AMIs live in Amazon S3, so you get charged for the actual space in takes in Amazon S3

- Amazon S3 pricing in US-EAST-1:
  - First 50 TB / month: $0.023 per GB
  - Next 450 TB / month: $0.022 per GB

- Overall it is quite inexpensive to store private AMIs.
- Make sure to remove the AMIs you don't use

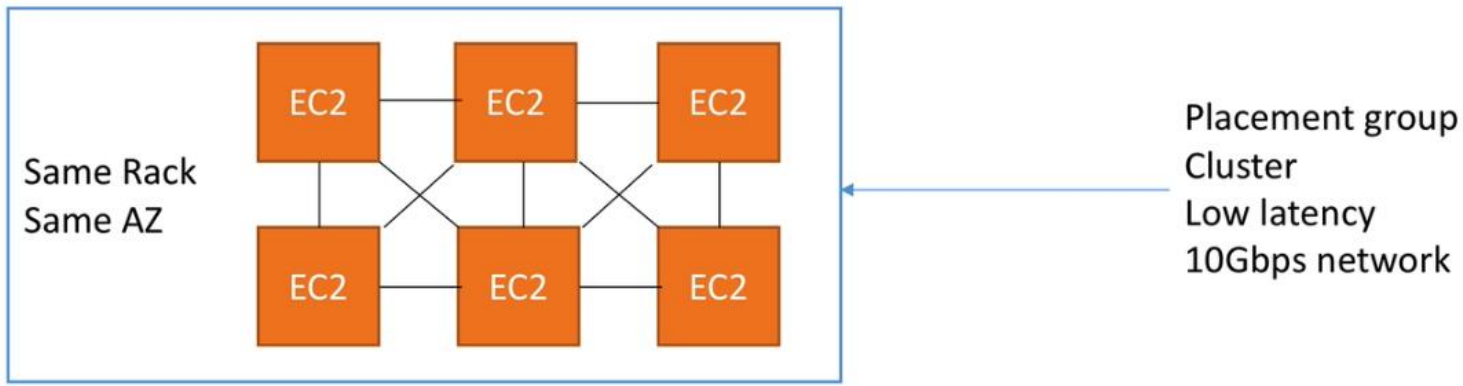
AMI Storage

- Your AMI take space and they live in Amazon S3
- Amazon S3 is a durable, cheap and resilient storage where most of your backups will live (but you won't see them in the S3 console)
- By default, your AMIs are private, and locked for your account / region
- You can also make your AMIs public and share them with other AWS accounts or sell them on the AMI Marketplace

EC2 User Data

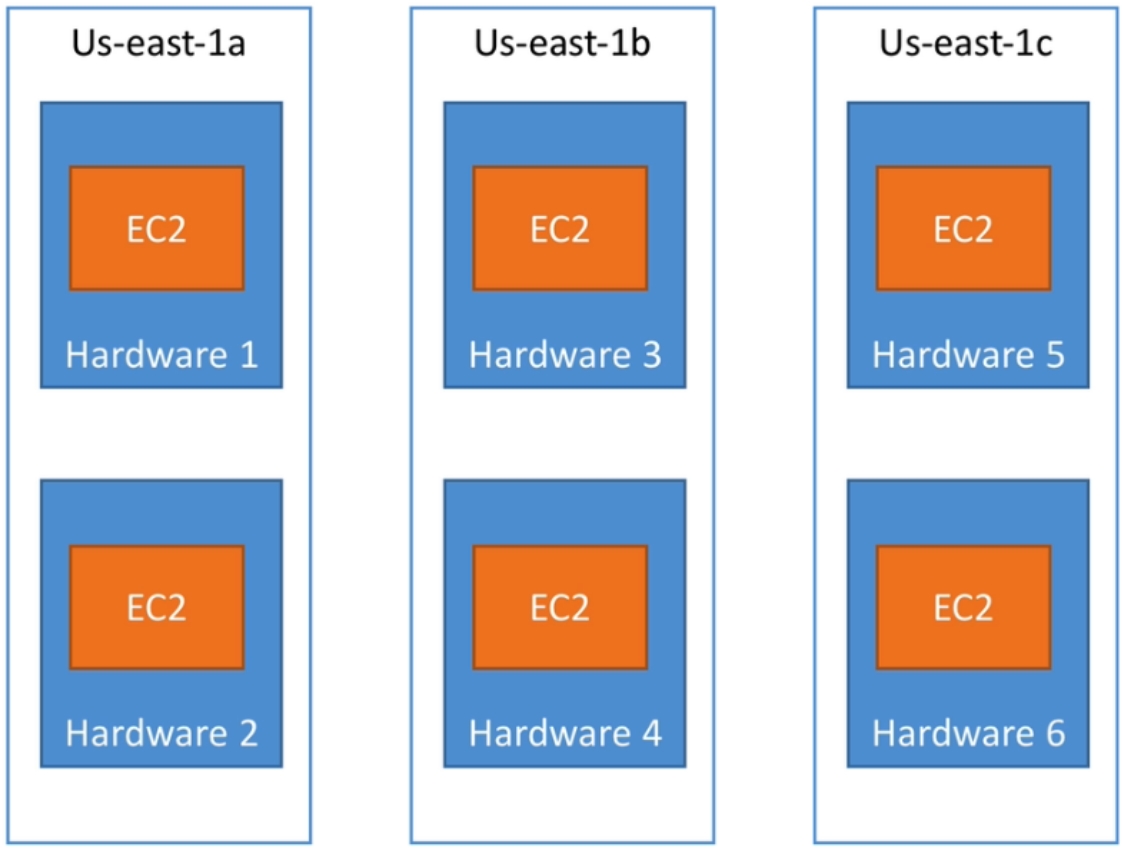
```
#!/bin/bash
sudo su
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html
```

Placement Groups
Cluster



- Pros: Great network (10 Gbps bandwidth between instances)
- Cons: If the rack fails, all instances fails at the same time
- Use case:
  - Big Data job that needs to complete fast
  - Application that needs extremely low latency and high network throughput

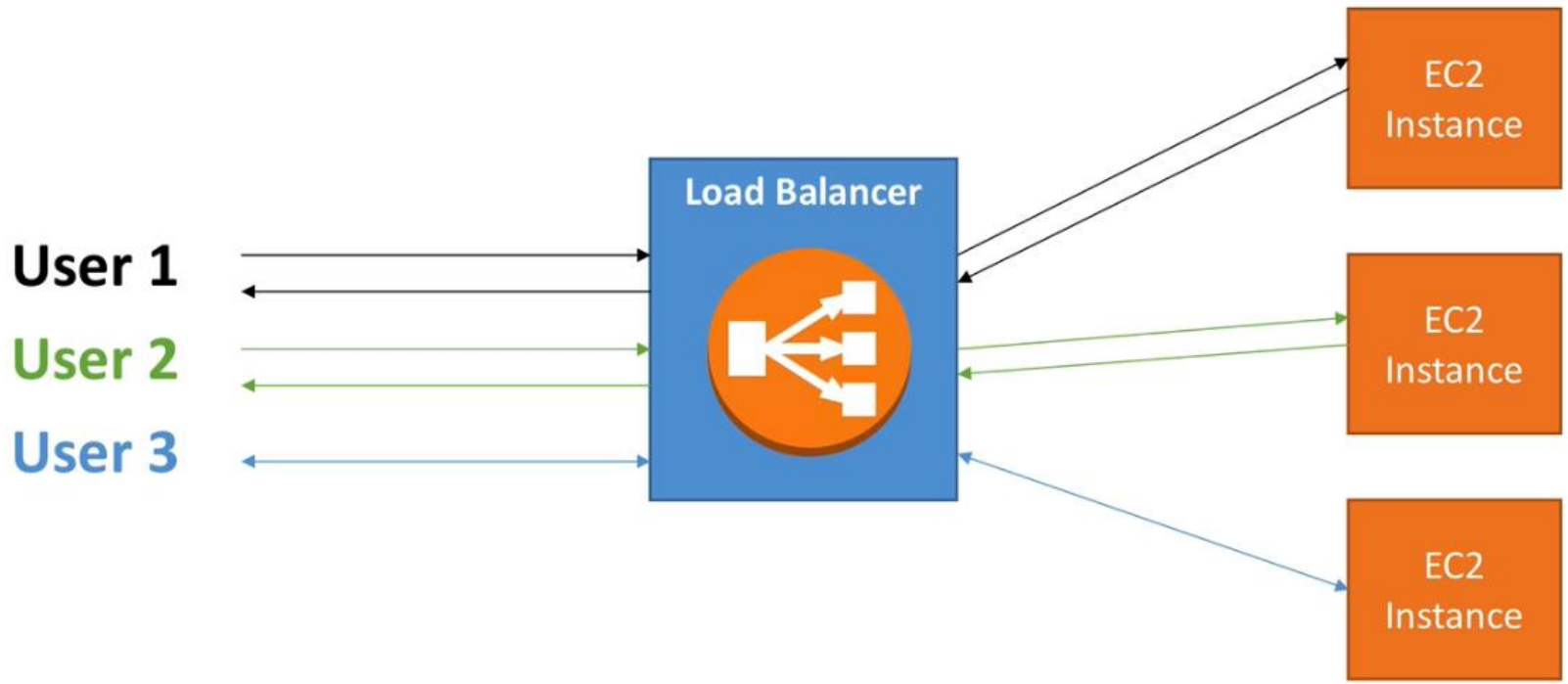
Placement Groups Spread



- Pros:
  - Can span across Availability Zones (AZ)
  - Reduced risk is simultaneous failure
  - EC2 Instances are on different physical hardware
- Cons:
  - Limited to 7 instances per AZ per placement group
- Use case:
  - Application that needs to maximize high availability
  - Cassandra Cluster, Kafka Cluster, Web application that is distributed

What is load balancing?

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



Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

Types of load balancer on AWS

- AWS has **3 kinds of Load Balancers**

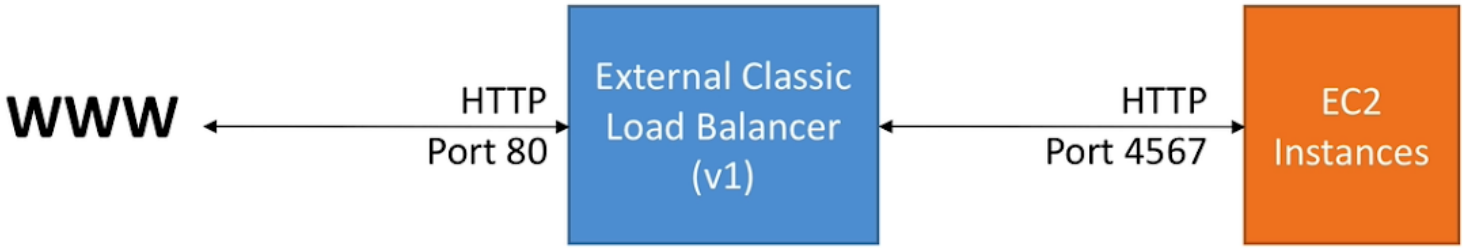
- Classic Load Balancer (v1 - old generation) - 2009
- Application Load Balancer (v2 - new generation) - 2016
- Network Load Balancer (v2 - new generation) - 2017
- Overall, it is recommended to use the newer / v2 generation load balancers as they provide more features

- You can setup **internal** (private) or **external** (public) ELBs

Classic Load Balancer (v1)

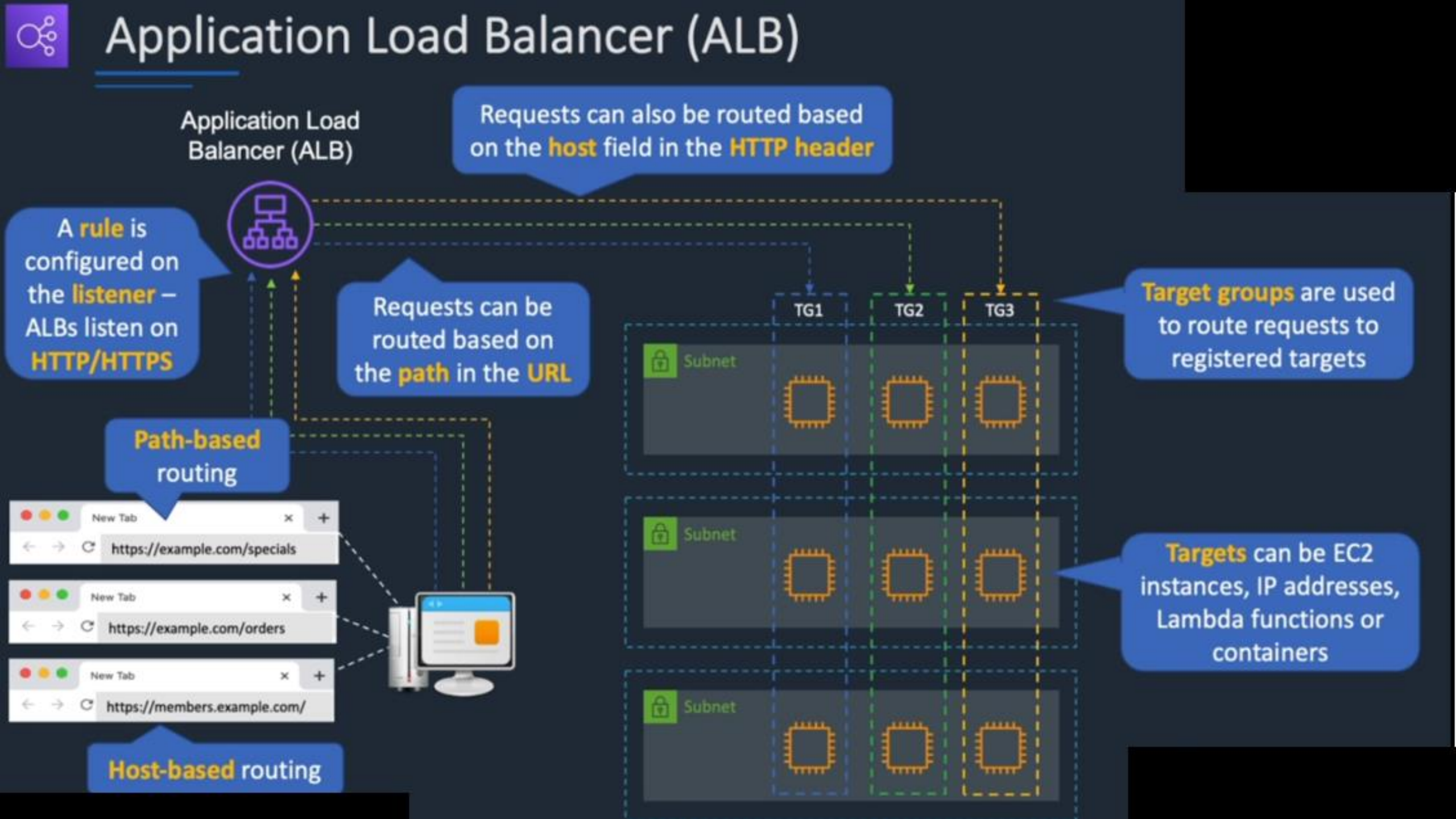
- **This hands on will cost you some money.**

- We are doing a classic load balancer hands on because:
  - many AWS users are still using these ones
  - they are great for learning the concept of load balancing

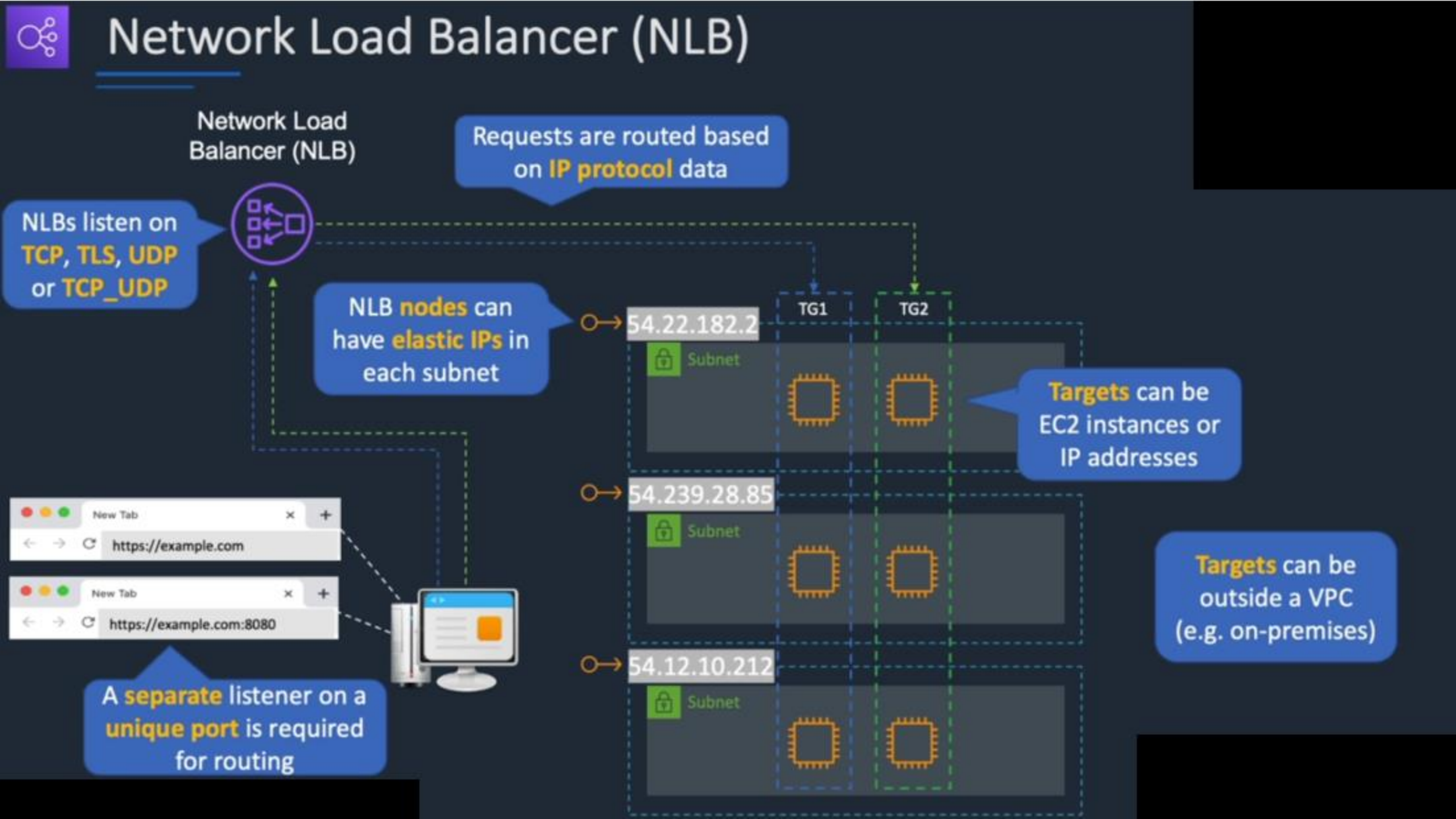


+ restrict traffic to ELB security group

Application Load Balancer (ALB)



Network Load Balancer (NLB)

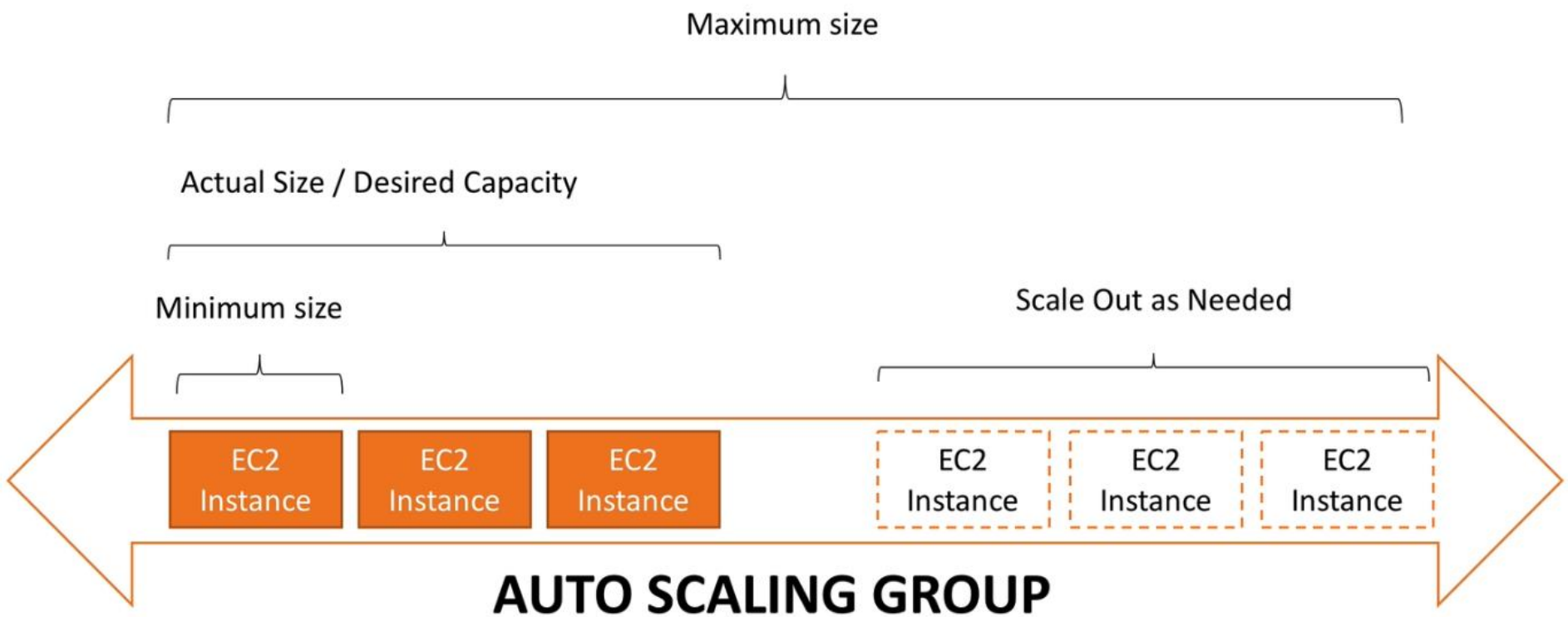


What's an Auto Scaling Group?

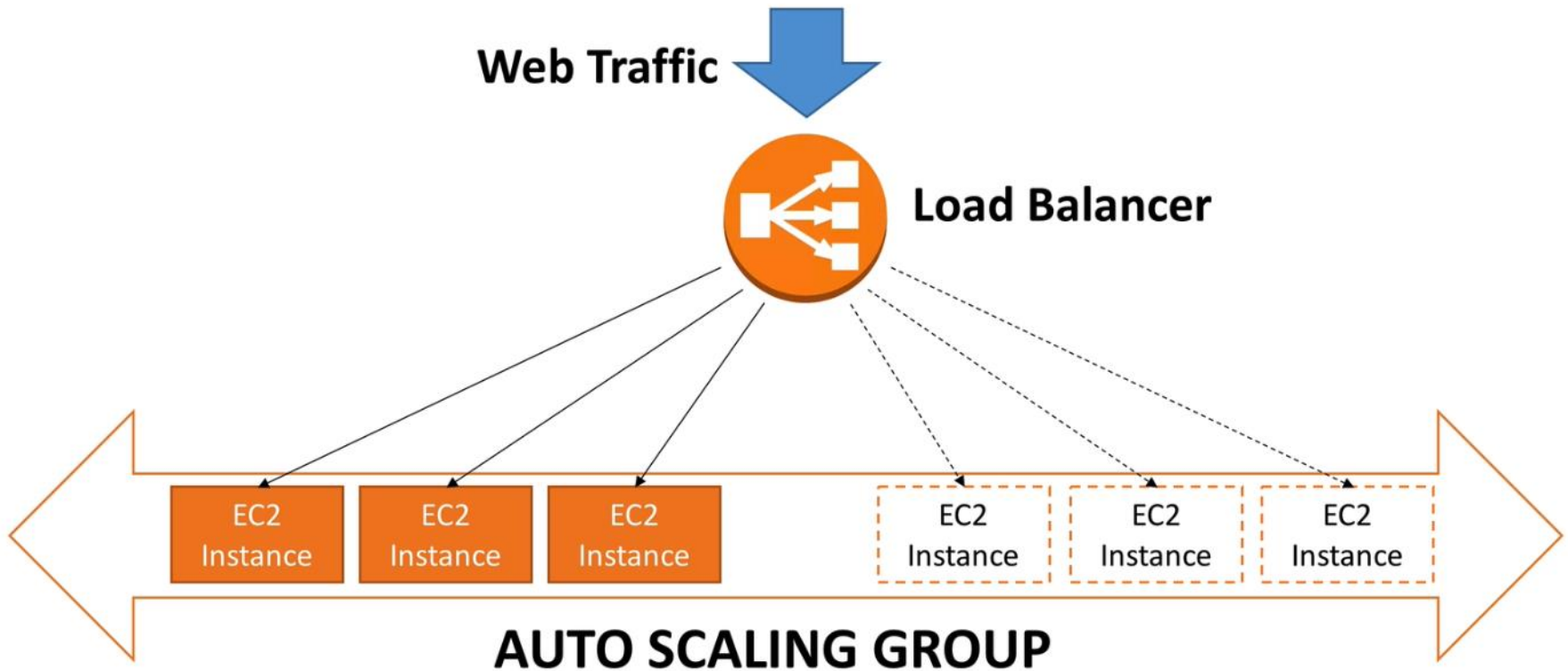
- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly

- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically Register new instances to a load balancer

Auto Scaling Group in AWS



Auto Scaling Group in AWS With Load Balancer



What's an EBS Volume?

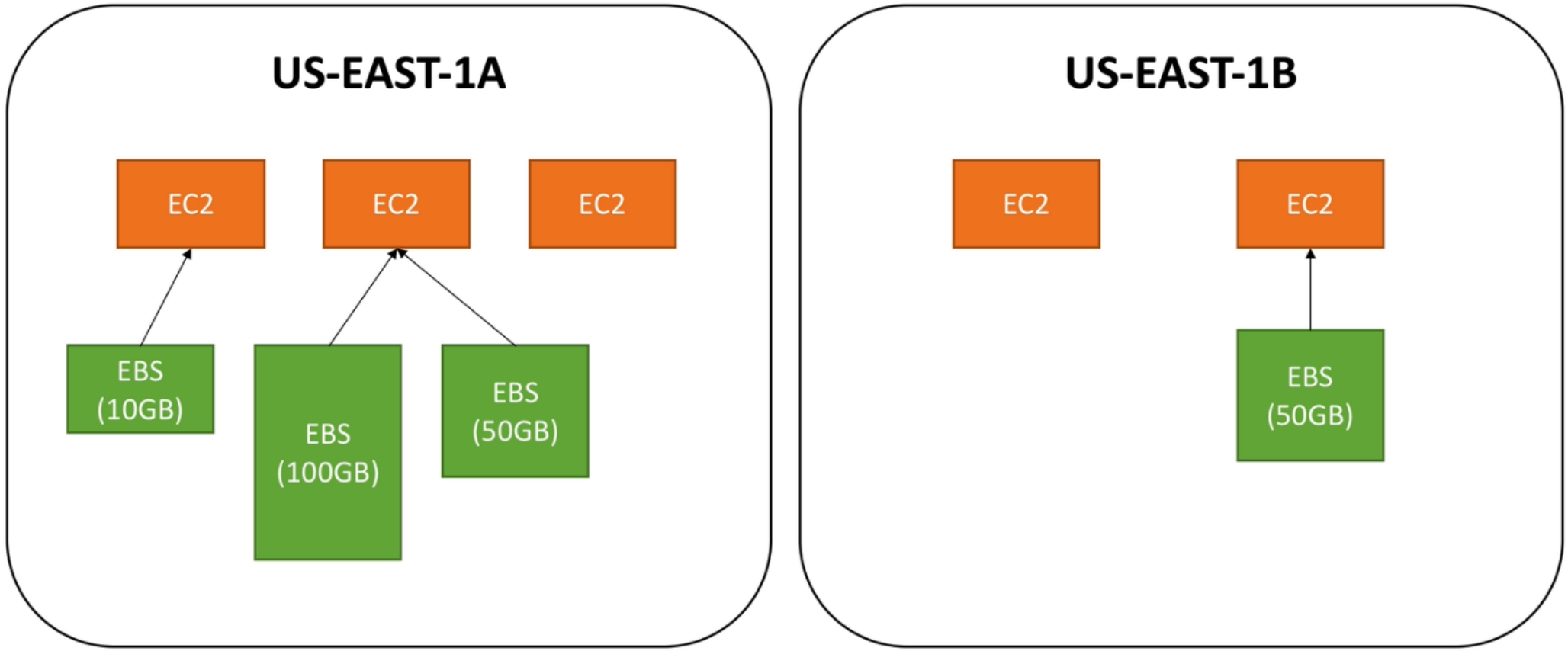
- An EC2 machine loses its root volume (main drive) when it is manually terminated.
- Unexpected terminations might happen from time to time (AWS would email you)
- Sometimes, you need a way to store your instance data somewhere

- An EBS (Elastic Block Store) Volume is a network drive you can attach to your instances while they run
- It allows your instances to persist data

EBS Volume

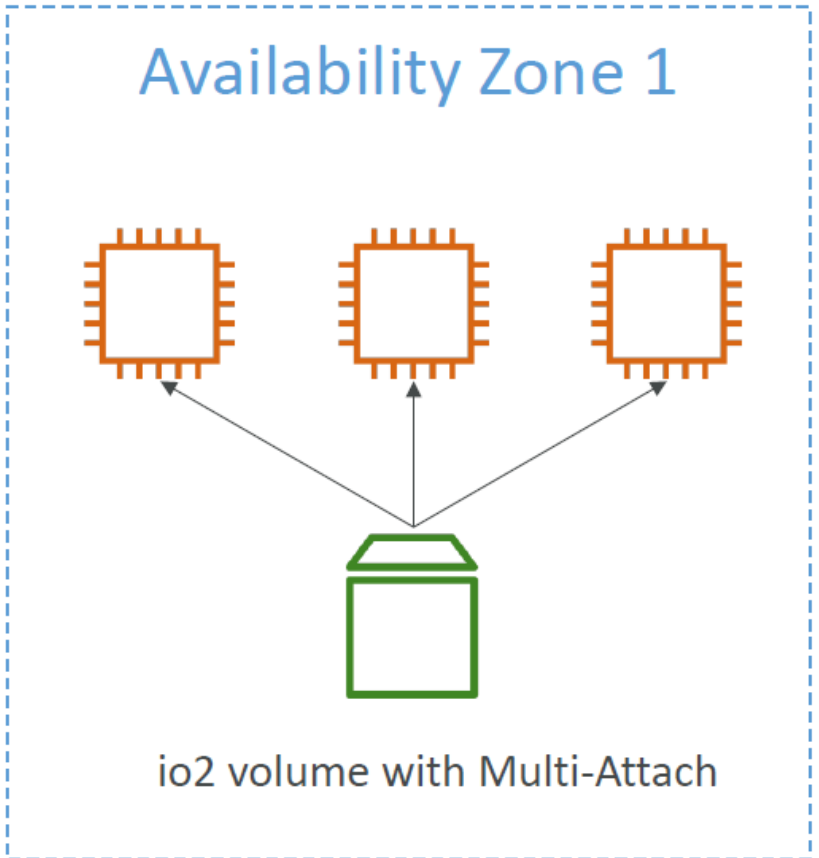
- It's a network drive (i.e. not a physical drive)
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
  - An EBS Volume in us-east-1a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

EBS Volume Example



EBS Multi-Attach – io1/io2 family

- Attach the same EBS volume to multiple EC2 instances in the same AZ
- Each instance has full read & write permissions to the high-performance volume
- Use case:
  - Achieve **higher application availability** in clustered Linux applications (ex: Teradata)
  - Applications must manage concurrent write operations
- Up to 16 EC2 Instances at a time
- Must use a file system that's cluster-aware (not XFS, EX4, etc…)



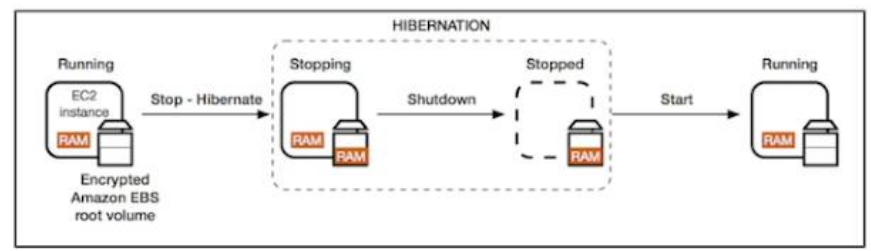
Elastic Network Interfaces (ENI)

- Logical component in a VPC that represents a **virtual network card**
- The ENI can have the following attributes:
  - Primary private IPv4, one or more secondary IPv4
  - One Elastic IP (IPv4) per private IPv4
  - One Public IPv4
  - One or more security groups
  - A MAC address
- You can create ENI independently and attach them on the fly (move them) on EC2 instances for failover
- Bound to a specific availability zone (AZ)

EC2 Hibernate

- Introducing **EC2 Hibernate**:
  - The in-memory (RAM) state is preserved
  - The instance boot is much faster!
    (the OS is not stopped / restarted)
  - Under the hood: the RAM state is written to a file in the root EBS volume
  - The root EBS volume must be encrypted

- **Use cases:**
  - long-running processing
  - saving the RAM state
  - services that take time to initialize



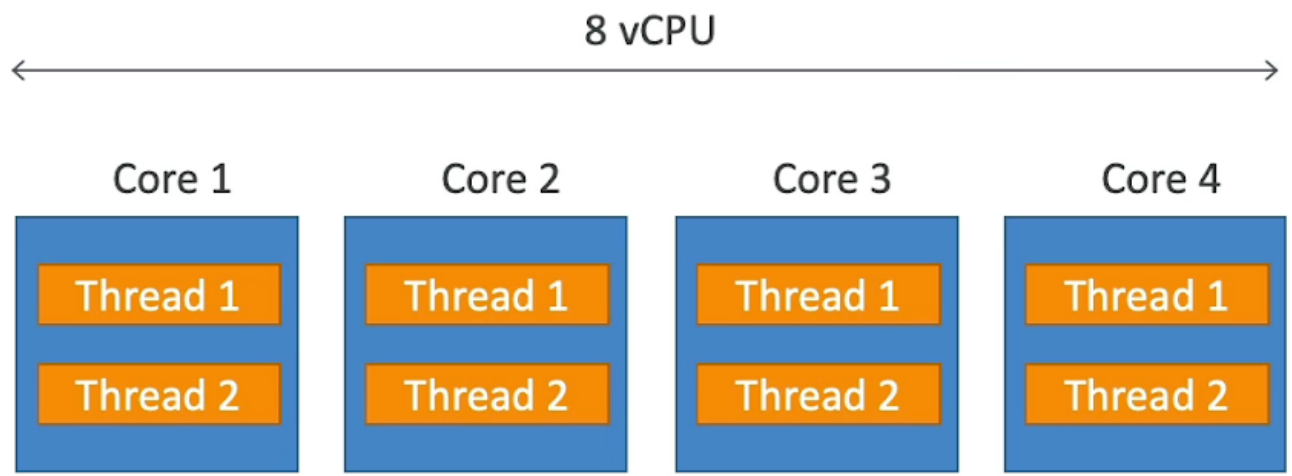
EC2 Hibernate

- We know we can stop, terminate instances
  - Stop: the data on disk (EBS) is kept intact in the next start
  - Terminate: any EBS volumes (root) also set-up to be destroyed is lost

- On start, the following happens:
  - First start: the OS boots & the EC2 User Data script is run
  - Following starts: the OS boots up
  - Then your application starts, caches get warmed up, and that can take time!

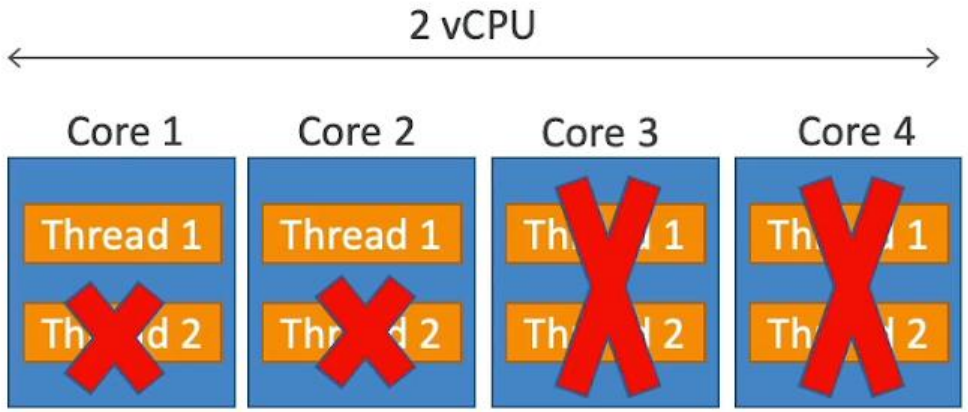
EC2 – Understanding vCPU

- Multiple threads can run on one CPU (multithreading)
- Each thread is represented as a virtual CPU (vCPU)
- Example: m5.2xlarge
  - 4 CPU
  - 2 threads per CPU
  - => 8 vCPU in total

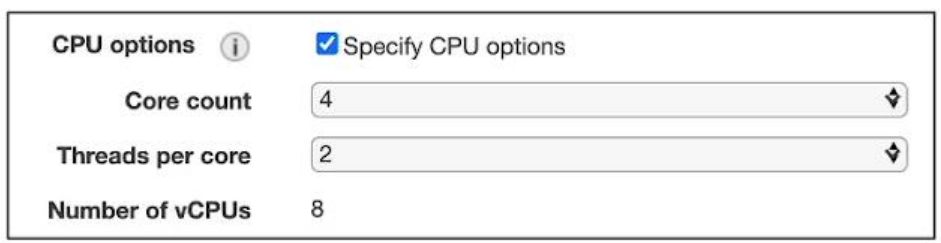


EC2 – Optimizing CPU options

- EC2 instances come with a combination of RAM and vCPU
- But in some cases, you may want to change the vCPU options:
  - **# of CPU cores:** you can decrease it (helpful if you need high RAM and low number of CPU) – to decrease licensing costs
  - **# of threads per core:** disable multithreading to have 1 thread per CPU – helpful for high performance computing (HPC) workloads
- Only specified during instance launch



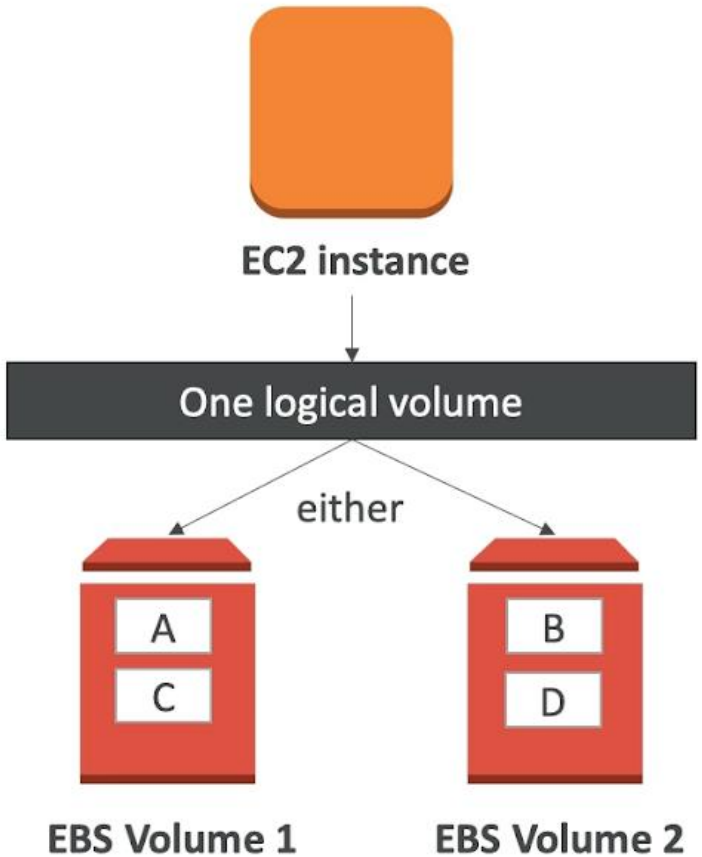
Instance type	Default vCPUs	Default CPU cores	Default threads per core	Valid CPU cores	Valid threads per core
r4.2xlarge	8	4	2	1, 2, 3, 4	1, 2



EBS RAID Options

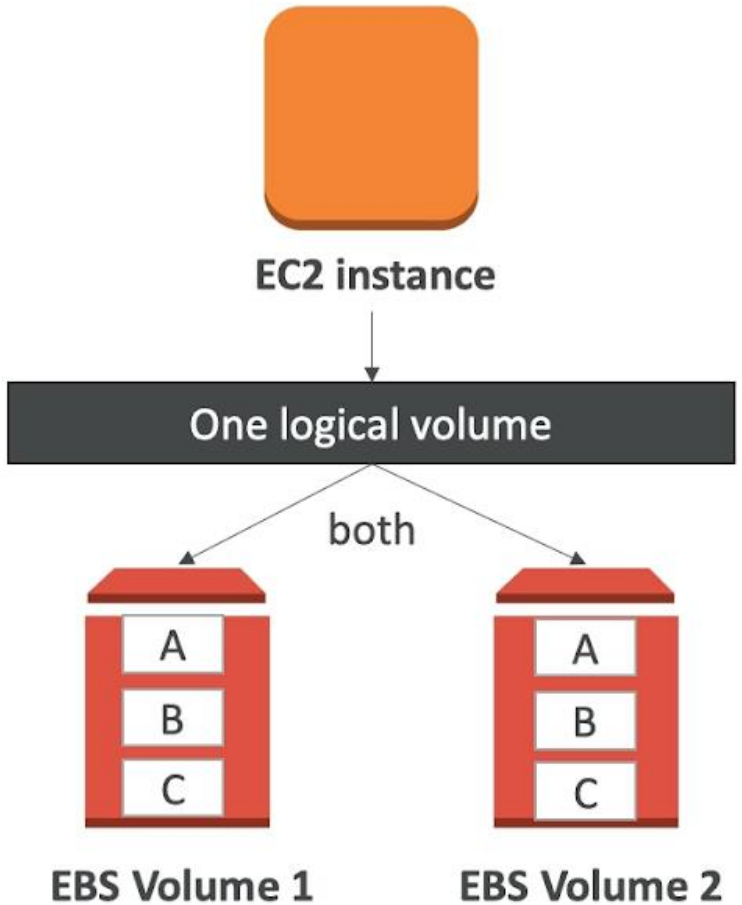
- EBS is already redundant storage (replicated within an AZ)
- But what if you want to increase IOPS to say 100 000 IOPS?
- What if you want to mirror your EBS volumes?
- You would mount volumes in parallel in RAID settings!
- RAID is possible as long as your OS supports it
- Some RAID options are:
  - RAID 0
  - RAID 1
  - RAID 5 (not recommended for EBS – see documentation)
  - RAID 6 (not recommended for EBS – see documentation)
- We'll explore RAID 0 and RAID 1

RAID 0 (increase performance)



- Combining 2 or more volumes and getting the total disk space and I/O
- But one disk fails, all the data is failed
- Use cases would be:
  - An application that needs a lot of IOPS and doesn't need fault-tolerance
  - A database that has replication already built-in
- Using this, we can have a very big disk with a lot of IOPS
- For example
  - two 500 GiB Amazon EBS io1 volumes with 4,000 provisioned IOPS each will create a…
  - 1000 GiB RAID 0 array with an available bandwidth of 8,000 IOPS and 1,000 MB/s of throughput

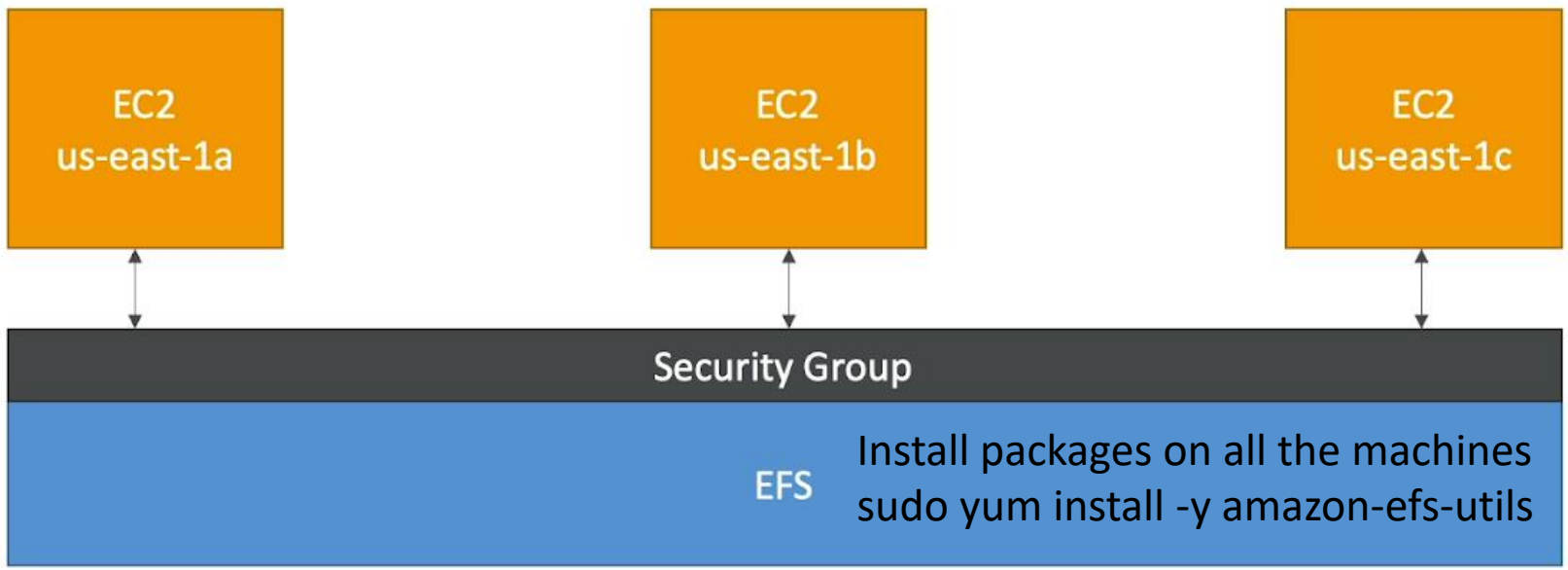
RAID 1 (increase fault tolerance)



- RAID 1 = Mirroring a volume to another
- If one disk fails, our logical volume is still working
- We have to send the data to two EBS volume at the same time (2x network)
- Use case:
  - Application that need increase volume fault tolerance
  - Application where you need to service disks
- For example:
  - two 500 GiB Amazon EBS io1 volumes with 4,000 provisioned IOPS each will create a…
  - 500 GiB RAID 1 array with an available bandwidth of 4,000 IOPS and 500 MB/s of throughput

EFS – Elastic File System

- Managed NFS (network file system) that can be mounted on many EC2
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use



EFS – Elastic File System

- Use cases: content management, web serving, data sharing, Wordpress
- Uses NFSv4.1 protocol
- Uses security group to control access to EFS
- Compatible with Linux based AMI (not Windows)
- Encryption at rest using KMS

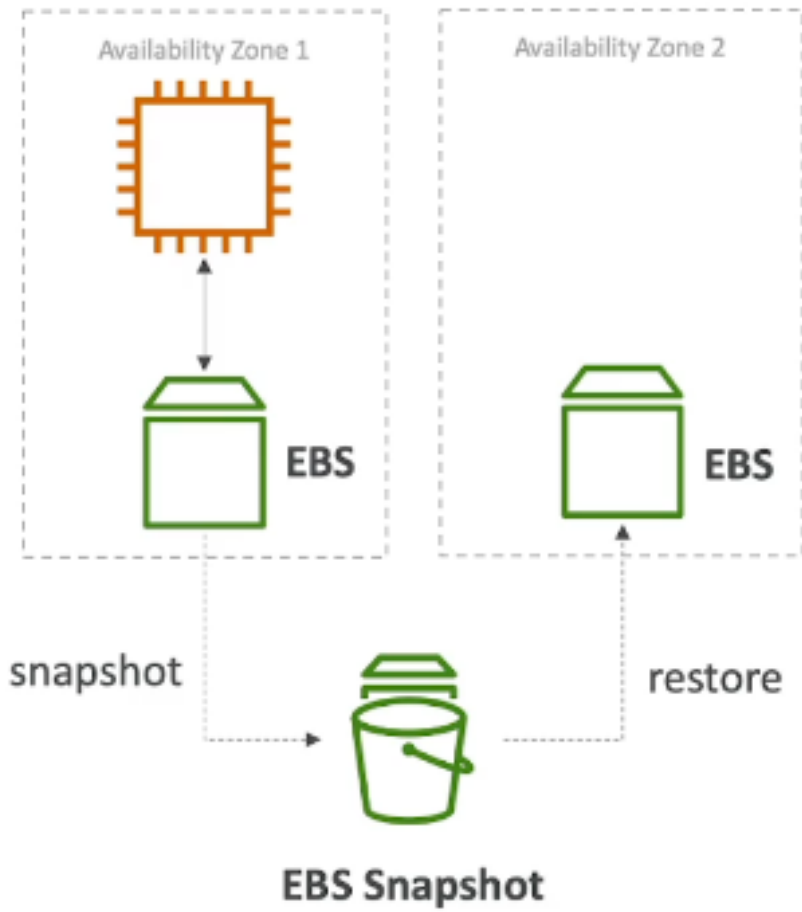
- POSIX file system (~Linux) that has a standard file API

EFS – Performance & Storage Classes

- EFS Scale
  - 1000s of concurrent NFS clients, 10 GB+ /s throughput
  - Grow to Petabyte-scale network file system, automatically
- Performance mode (set at EFS creation time)
  - General purpose (default): latency-sensitive use cases (web server, CMS, etc…)
  - Max I/O – higher latency, throughput, highly parallel (big data, media processing)
- Throughput mode
  - Bursting (1 TB = 50MiB/s + burst of up to 100MiB/s)
  - Provisioned: set your throughput regardless of storage size, ex: 1 GiB/s for 1 TB storage
- Storage Tiers (lifecycle management feature – move file after N days)
  - Standard: for frequently accessed files
  - Infrequent access (EFS-IA): cost to retrieve files, lower price to store

EBS vs EFS – Elastic Block Storage

- EBS volumes…
  - can be attached to only one instance at a time
  - are locked at the Availability Zone (AZ) level
  - gp2: IO increases if the disk size increases
  - io1: can increase IO independently
- To migrate an EBS volume across AZ
  - Take a snapshot
  - Restore the snapshot to another AZ
  - EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)

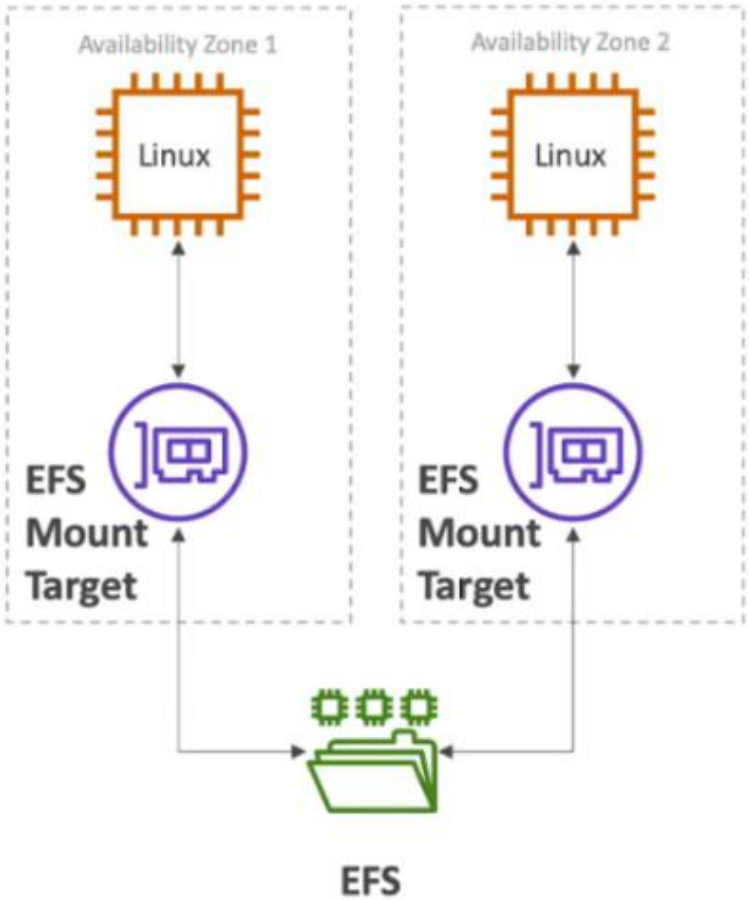


EBS vs EFS – Elastic File System

- Mounting 100s of instances across AZ
- EFS share website files (WordPress)
- Only for Linux Instances (POSIX)

- EFS has a higher price point than EBS
- Can leverage EFS-IA for cost savings

- Remember: EFS vs EBS vs Instance Store



Mounting the EBS volume

List the volumes
lsblk
Check if the volume has any data
sudo file -s /dev/xvdf
Format the volume as ext4
sudo mkfs -t ext4 /dev/xvdf
To check the UUID of the drive
sudo blkid
Create a directory
sudo mkdir /myexternalvolume
Mount our EBS to our directory
sudo mount /dev/xvdf /myexternalvolume

Go to the directory and verify size and free space
cd /myexternalvolume
echo “hello” > hello.txt
df -h .
sudo touch hello.txt
ls
After reboot the fs needs to mount again instead, so we need to run this commands
cat /etc/fstab
sudo cp /etc/fstab /etc/fstab.bak
sudo yum install -y nano
sudo nano /etc/fstab
add -> /dev/xvdf /myexternalvolume ext4 defaults,nofail 0 0
cat /etc/fstab
sudo reboot -h
lsblk
cd /myexternalvolume
ls
sudo mount -a

Resizing the EBS volume
sudo resize2fs /dev/xvdf

Use of fstab -
Your Linux system's filesystem table, aka fstab , is a configuration table designed to ease the burden of mounting and unmounting file systems to a machine. It is a set of rules used to control how different filesystems are treated each time they are introduced to a system. Consider USB drives, for example.

EC2 User Data

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

Custom loop script

```
#!/bin/bash

while true
do
   curl http://dns/ip
done
```

To run

```
chmod +x test.sh
./script_name.sh
```

```
curl
nslookup
dig
```

Steps to mount the Instances the EFS
Install packages on all the machines
sudo yum install -y amazon-efs-utils
Create a directory “efs” on all the machines
mkdir efs
Mount all the machines to that folder
sudo mount -t efs -o tls fs-0c0edb08bd07c9bd8:/ efs
Now all the machines are mounted to the same directory in the NFS
Note - to access the NFS we need to route the traffic from ec2 machine to NFS (SG-to-SG reference)

Hi-Tech Institution

The Ec2 Instance not able to do the RDP/SSH

1. **Sec group Port – 3389/22**
2. **Instance Current status – Running or not**
3. **Ec2 instance Status checks – 2/2**
    1. **1/2 - N/w, Antivirus, EBS, Nw interface => Stop -> Start**
    2. **0/2 – Need check with AWS | Backend H/w issue => Stop -> Start**
4. **Need to check instance screenshot from Ec2 console**
    1. **If it shows any BSOD => Restore from latest backup**
5. **System logs from Ec2 console**
6. **Cloud Trail -> Any recent changes to that instance**

Hi-Tech Institution

The Ec2 Instance not able to do the RDP/SSH

1. Any rule in place on ACL
2. Private Vs Public Subnet
3. VPC -> Has IGW or not
4. VPC -> Has RT entries or not
5. General Human error
6. Stop -> Start
7. Restore from latest snapshot backup

Hi-Tech Institution

Troubleshoot your Application Load Balancers

1. Load Balancer backend instances status Inservice/Healthy

2. If Load Balancer backend instances status is Out of service/Unhealthy

3. Ec2 instance Status checks – 2/2 Checks Passed
   i. 1/2 - N/w, Antivirus, EBS, Nw interface
   ii. 0/2 – Need check with AWS | Backend H/w issue

AWS troubleshooting | Ec2 instance connection timed out error | AWS realtime issues | video-1

59 views • 3 hours ago

Hi-Tech Institution

Hi-Tech Institution - Phone/Whats app: 7092909192 All Technical Training, all over the world. Cloud Computing, All Cloud ..

New

Hi-Tech Institution

Troubleshoot your Application Load Balancers

4. Check Security Group and ACL for required ports is opened or Not

5. Load balancer health check configuration need to be verified

6. Try to restart the Application service inside of Backend Ec2 instances

7. Check for the ALB Forwarding rules configured properly

Hi-Tech Institution

Troubleshoot your Application Load Balancers

7. Private Vs Public Subnet

8. VPC -> Has IGW or not

9. Cloud Trail -> Any recent changes to that instance

10. Stop -> Start

AWS RDS Overview

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

Advantage over using RDS versus deploying DB on EC2

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

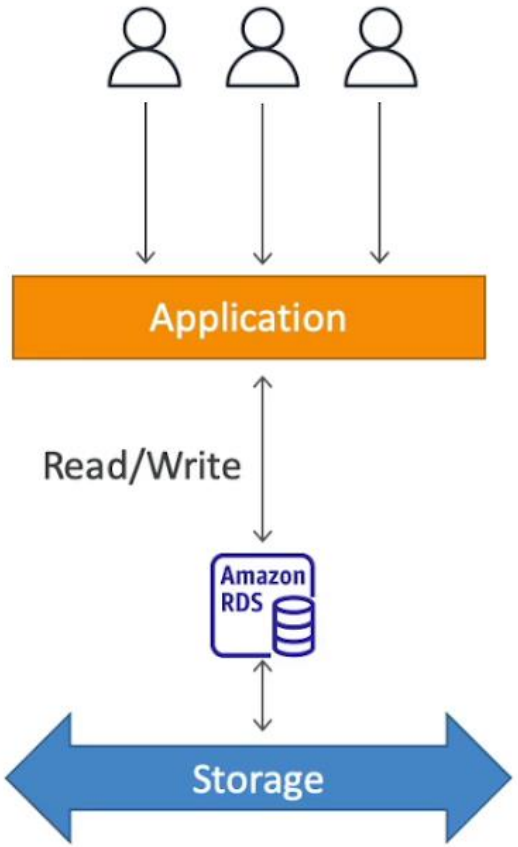
RDS Backups

- Backups are automatically enabled in RDS
- Automated backups:
  - Daily full backup of the database (during the maintenance window)
  - Transaction logs are backed-up by RDS every 5 minutes
  - => ability to restore to any point in time (from oldest backup to 5 minutes ago)
  - 7 days retention (can be increased to 35 days)

- DB Snapshots:
  - Manually triggered by the user
  - Retention of backup for as long as you want

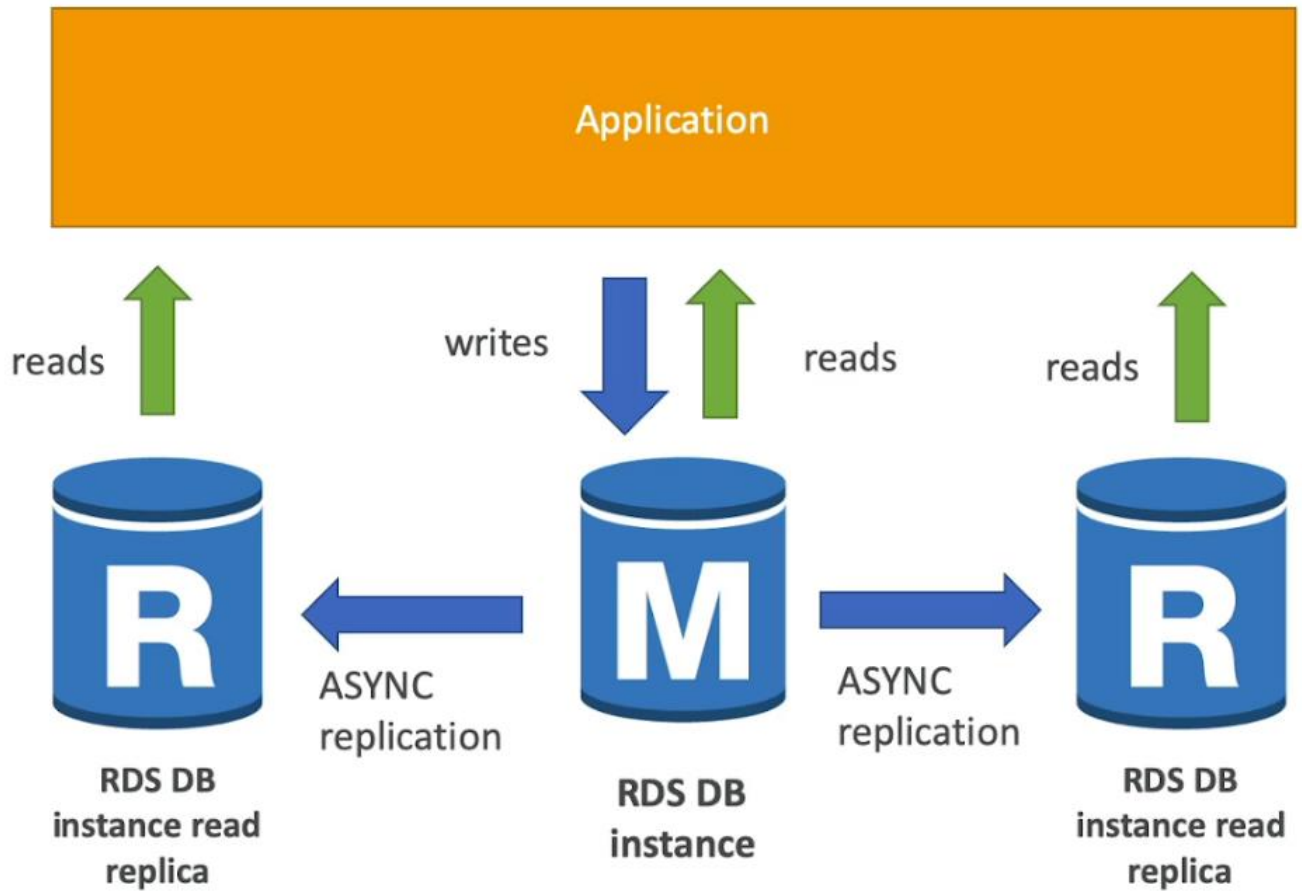
RDS – Storage Auto Scaling

- Helps you increase storage on your RDS DB instance dynamically
- When RDS detects you are running out of free database storage, it scales automatically
- Avoid manually scaling your database storage
- You have to set **Maximum Storage Threshold** (maximum limit for DB storage)
- Automatically modify storage if:
  - Free storage is less than 10% of allocated storage
  - Low-storage lasts at least 5 minutes
  - 6 hours have passed since last modification
- Useful for applications with **unpredictable workloads**
- Supports all RDS database engines (MariaDB, MySQL, PostgreSQL, SQL Server, Oracle)



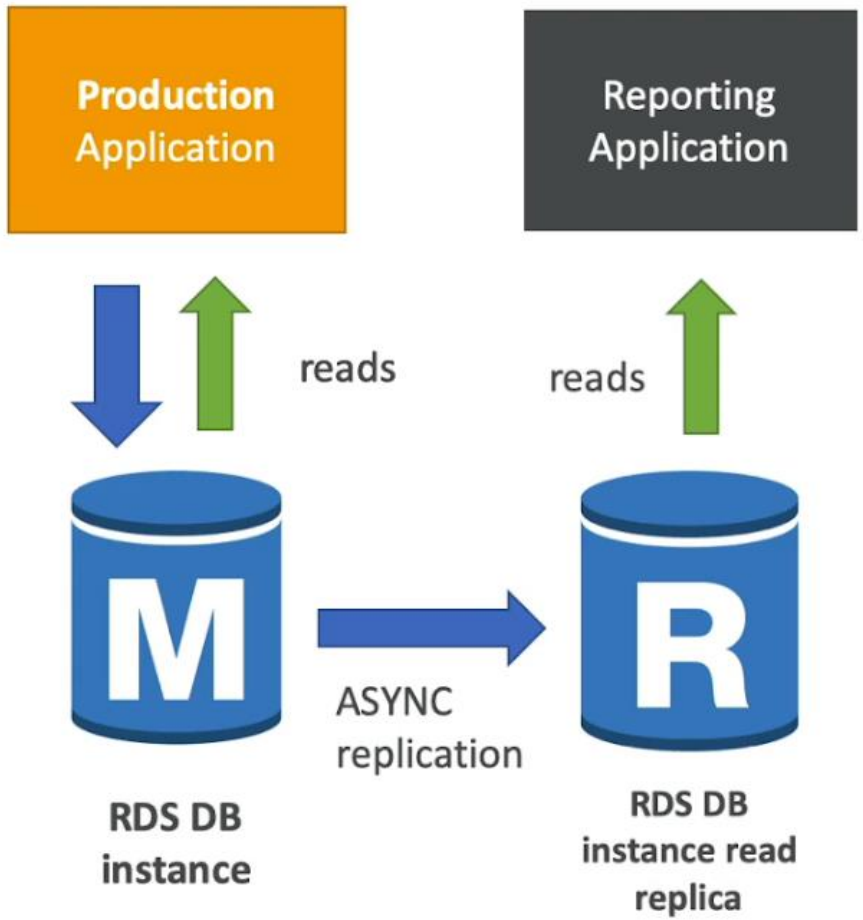
RDS Read Replicas for read scalability

- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is ASYNC, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



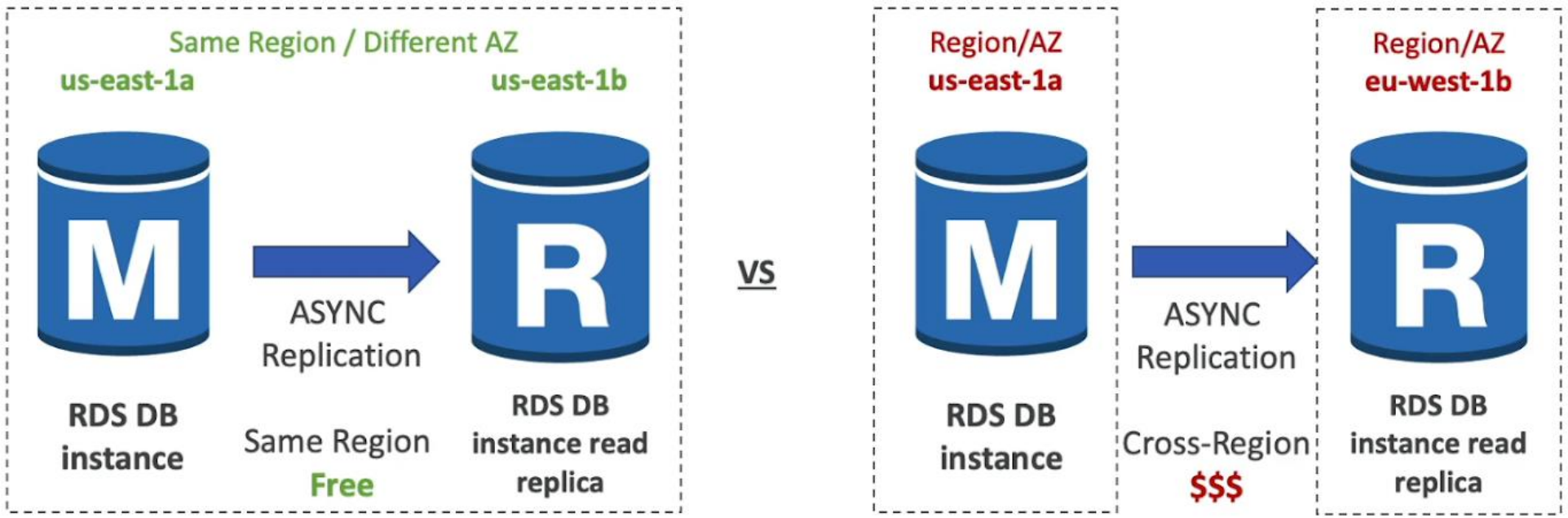
RDS Read Replicas – Use Cases

- You have a production database that is taking on normal load
- You want to run a reporting application to run some analytics
- You create a Read Replica to run the new workload there
- The production application is unaffected
- Read replicas are used for SELECT (=read) only kind of statements (not INSERT, UPDATE, DELETE)



RDS Read Replicas – Network Cost

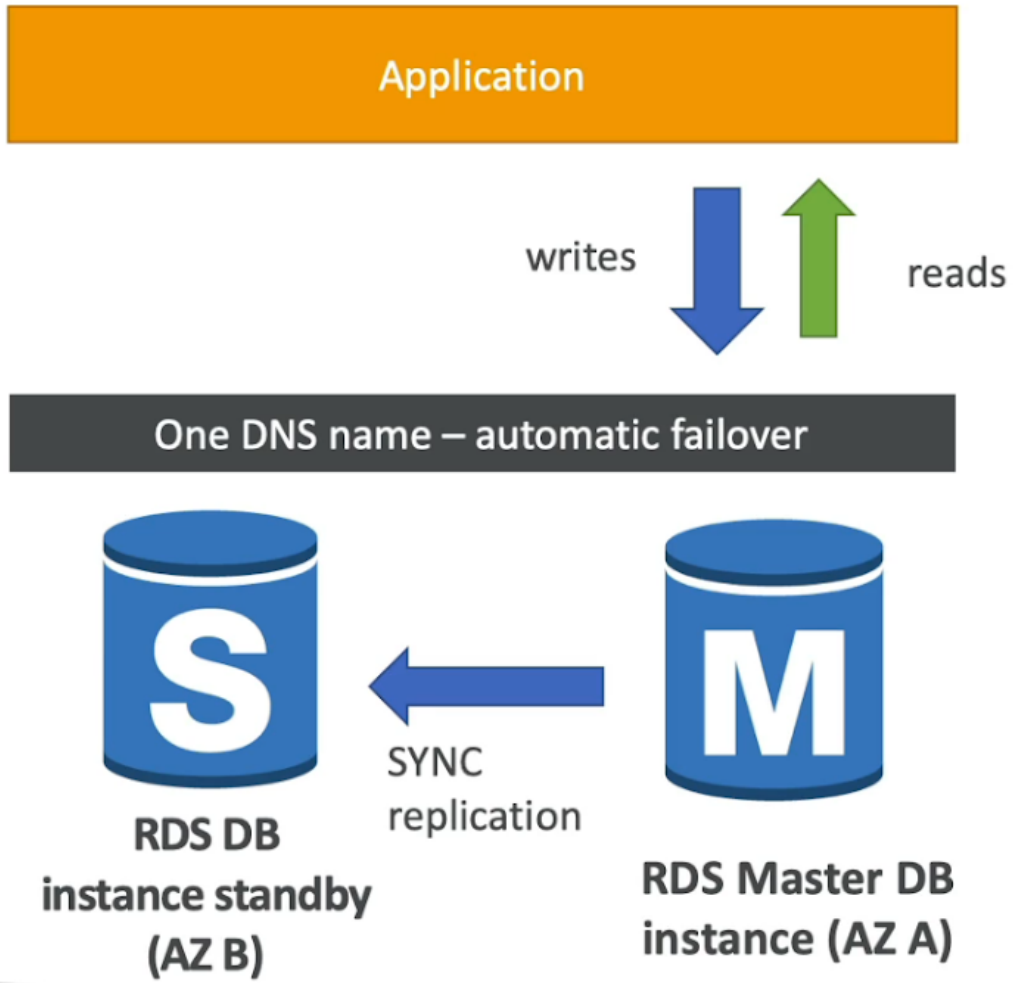
- In AWS there's a network cost when data goes from one AZ to another
- For RDS Read Replicas within the same region, you don't pay that fee



RDS Multi AZ (Disaster Recovery)

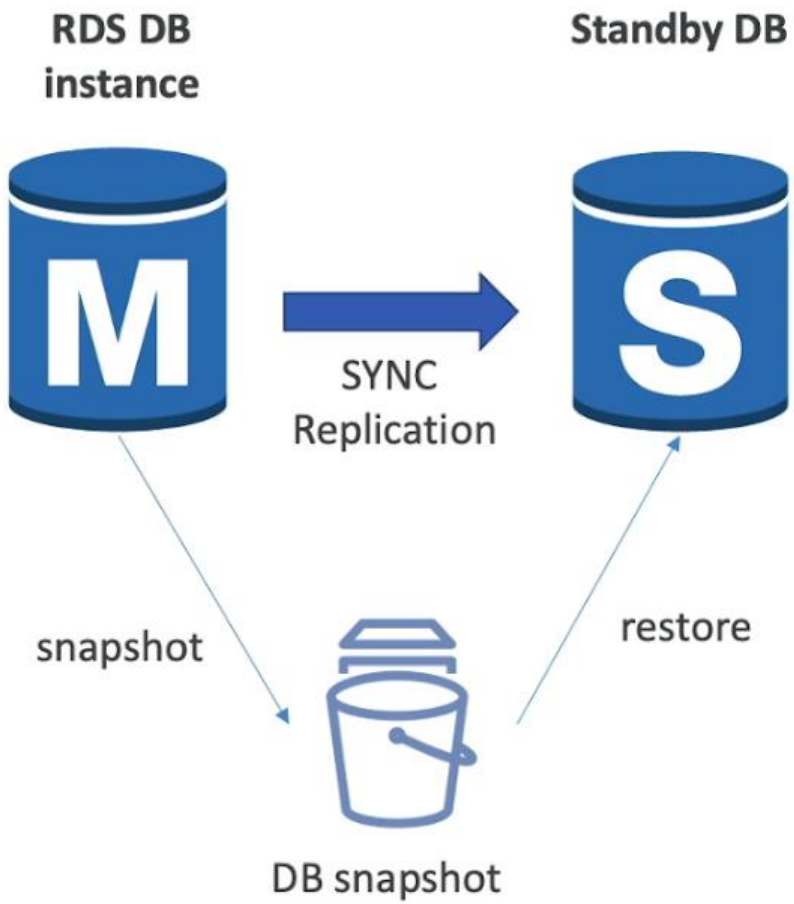
- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling

- Note: The Read Replicas be setup as Multi AZ for Disaster Recovery (DR)



RDS – From Single-AZ to Multi-AZ

- Zero downtime operation (no need to stop the DB)
- Just click on "modify" for the database
- The following happens internally:
  - A snapshot is taken
  - A new DB is restored from the snapshot in a new AZ
  - Synchronization is established between the two databases



RDS Security - Encryption

- At rest encryption
  - Possibility to encrypt the master & read replicas with AWS KMS - AES-256 encryption
  - Encryption has to be defined at launch time
  - **If the master is not encrypted, the read replicas cannot be encrypted**
  - Transparent Data Encryption (TDE) available for Oracle and SQL Server

- In-flight encryption
  - SSL certificates to encrypt data to RDS in flight
  - Provide SSL options with trust certificate when connecting to database
  - To enforce SSL:
    - **PostgreSQL:** rds.force_ssl=1 in the AWS RDS Console (Parameter Groups)
    - **MySQL:** Within the DB:
      GRANT USAGE ON *.* TO 'mysqluser'@'%' **REQUIRE SSL;**

RDS Encryption Operations

- Encrypting RDS backups
  - Snapshots of un-encrypted RDS databases are un-encrypted
  - Snapshots of encrypted RDS databases are encrypted
  - Can copy a snapshot into an encrypted one

- To encrypt an un-encrypted RDS database:
  - Create a snapshot of the un-encrypted database
  - Copy the snapshot and enable encryption for the snapshot
  - Restore the database from the encrypted snapshot
  - Migrate applications to the new database, and delete the old database

RDS Security – Network & IAM

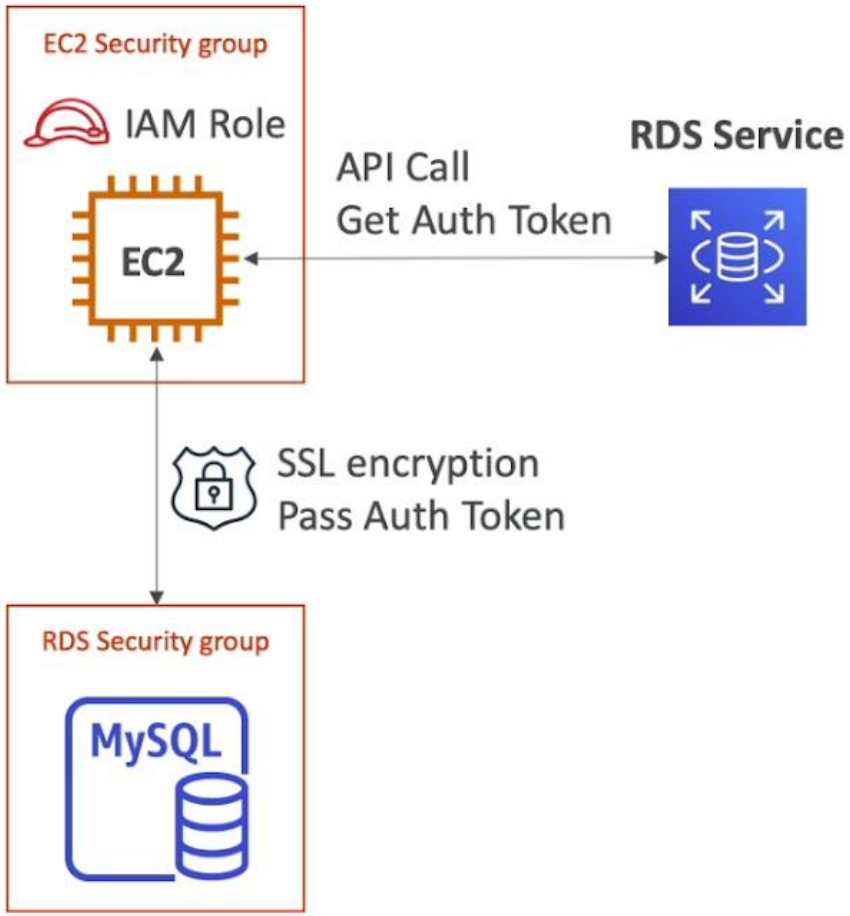
- Network Security
  - RDS databases are usually deployed within a private subnet, not in a public one
  - RDS security works by leveraging security groups (the same concept as for EC2 instances) – it controls which IP / security group can **communicate** with RDS

- Access Management
  - IAM policies help control who can **manage** AWS RDS (through the RDS API)
  - Traditional Username and Password can be used to **login into** the database
  - IAM-based authentication can be used to login into RDS MySQL & PostgreSQL

RDS - IAM Authentication

- IAM database authentication works with **MySQL and PostgreSQL**
- You don't need a password, just an authentication token obtained through IAM & RDS API calls
- Auth token has a lifetime of 15 minutes

- Benefits:
  - Network in/out must be encrypted using SSL
  - IAM to centrally manage users instead of DB
  - Can leverage IAM Roles and EC2 Instance profiles for easy integration



RDS Security – Summary

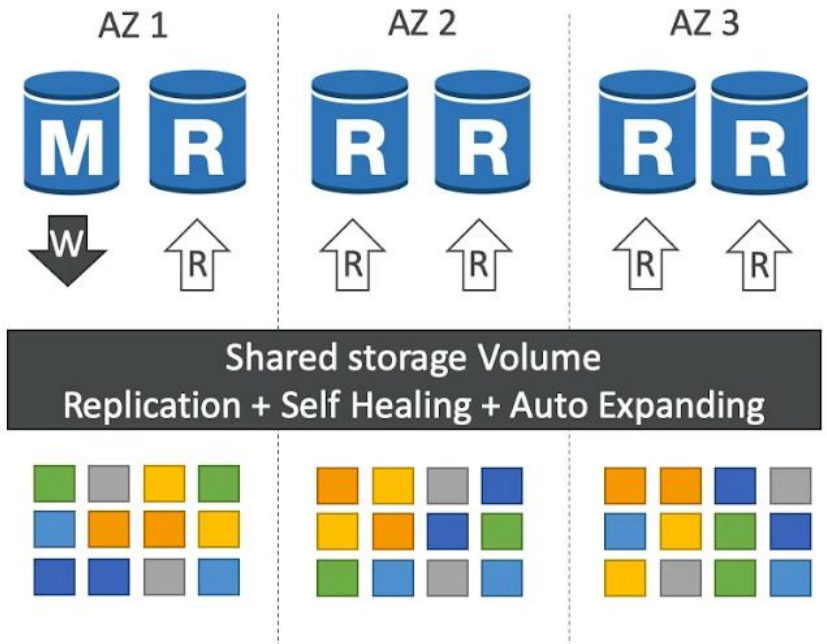
- Encryption at rest:
  - Is done only when you first create the DB instance
  - or: unencrypted DB => snapshot => copy snapshot as encrypted => create DB from snapshot
- Your responsibility:
  - Check the ports / IP / security group inbound rules in DB's SG
  - In-database user creation and permissions or manage through IAM
  - Creating a database with or without public access
  - Ensure parameter groups or DB is configured to only allow SSL connections
- AWS responsibility:
  - No SSH access
  - No manual DB patching
  - No manual OS patching
  - No way to audit the underlying instance

Amazon Aurora

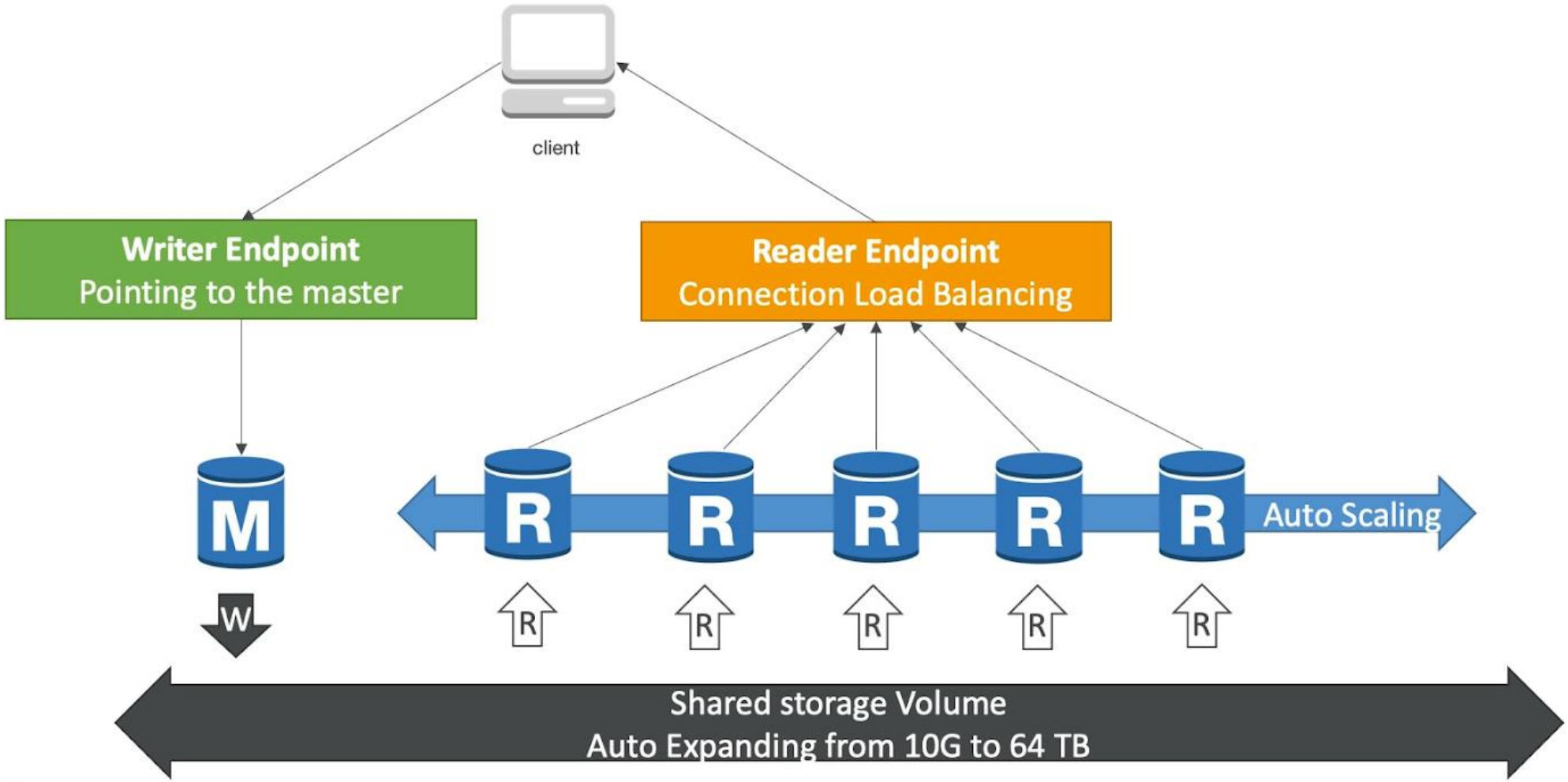
- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64 TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It's HA native.
- Aurora costs more than RDS (20% more) – but is more efficient

Aurora High Availability and Read Scaling

- 6 copies of your data across 3 AZ:
  - 4 copies out of 6 needed for writes
  - 3 copies out of 6 need for reads
  - Self healing with peer-to-peer replication
  - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds
- Master + up to 15 Aurora Read Replicas serve reads
- Support for Cross Region Replication



Aurora DB Cluster



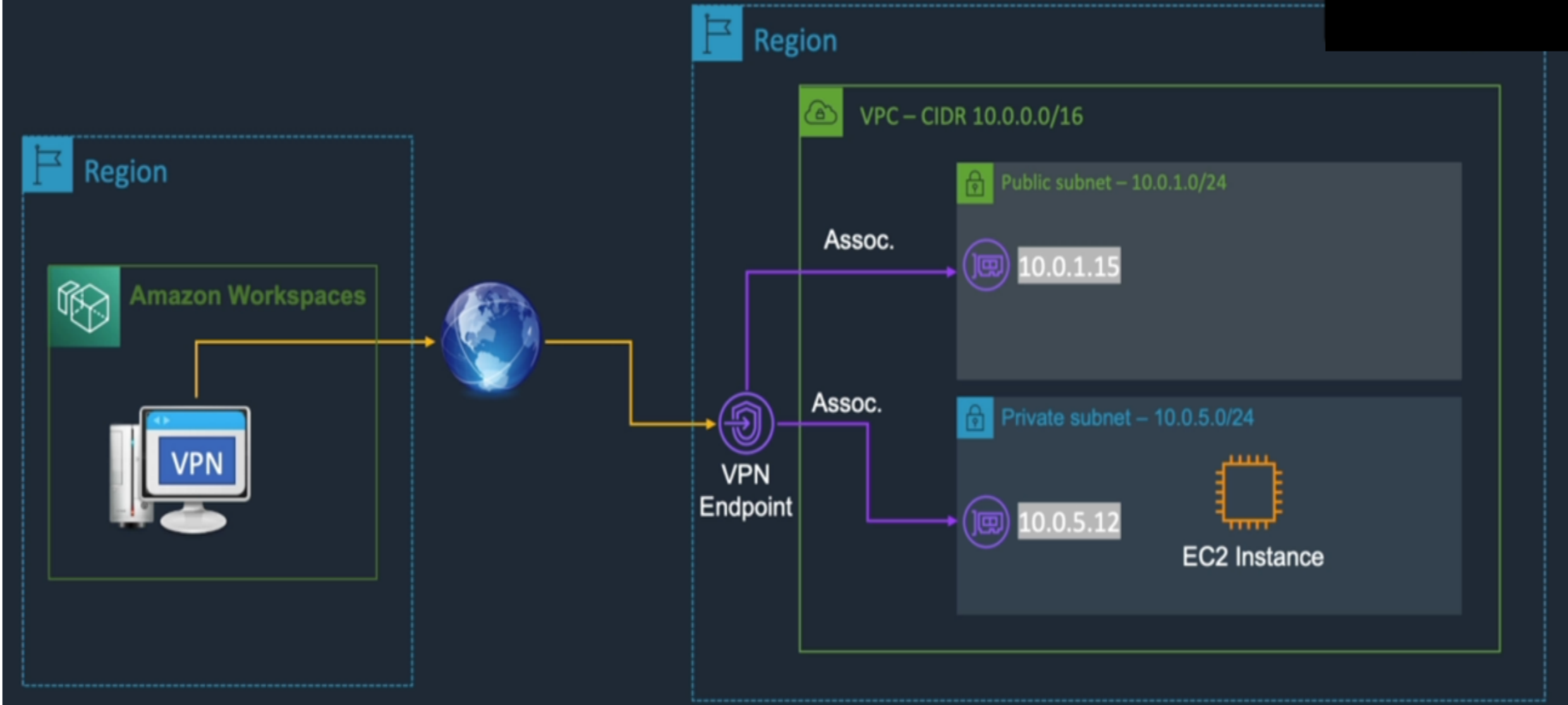
Features of Aurora

- Automatic fail-over
- Backup and Recovery
- Isolation and security
- Industry compliance
- Push-button scaling
- Automated Patching with Zero Downtime
- Advanced Monitoring
- Routine Maintenance
- Backtrack: restore data at any point of time without using backups

Aurora Security

- Similar to RDS because uses the same engines
- Encryption at rest using KMS
- Automated backups, snapshots and replicas are also encrypted
- Encryption in flight using SSL (same process as MySQL or Postgres)
- Possibility to authenticate using IAM token (same method as RDS)
- You are responsible for protecting the instance with security groups
- You can't SSH

AWS Client VPN – Hands-On



Section introduction

- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage
- It's widely popular and deserves its own section

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

Amazon S3 Overview - Buckets

- Amazon S3 allows people to store objects (files) in "buckets" (directories)
- Buckets must have a **globally unique name**
- Buckets are defined at the region level
- Naming convention
  - No uppercase
  - No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number

Amazon S3 Overview - Objects

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

Amazon S3 Overview – Objects (continued)

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

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

Amazon S3 - Versioning

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

S3 Bucket Policies

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

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

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

Bucket settings for Block Public Access

- Block public access to buckets and objects granted through
  - *new* access control lists (ACLs)
  - *any* access control lists (ACLs)
  - *new* public bucket or access point policies
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

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

Hands on to the above mentioned option how to access the objects from the public URL of the bucket object

IAM S3 Allow Policy For Single Bucket:

```
{    "Version": "2012-10-17",    "Statement": [        {
"Effect": "Allow",            "Action": [
"s3:GetBucketLocation",                "s3:ListAllMyBuckets"            ],
"Resource": "arn:aws:s3:::*"        },        {            "Effect":
"Allow",            "Action": "s3:*",            "Resource": [
"arn:aws:s3:::lokeshava",                "arn:aws:s3:::lokeshava/*"
]        }    ]}
```

S3 Deny Policy For A Single User:

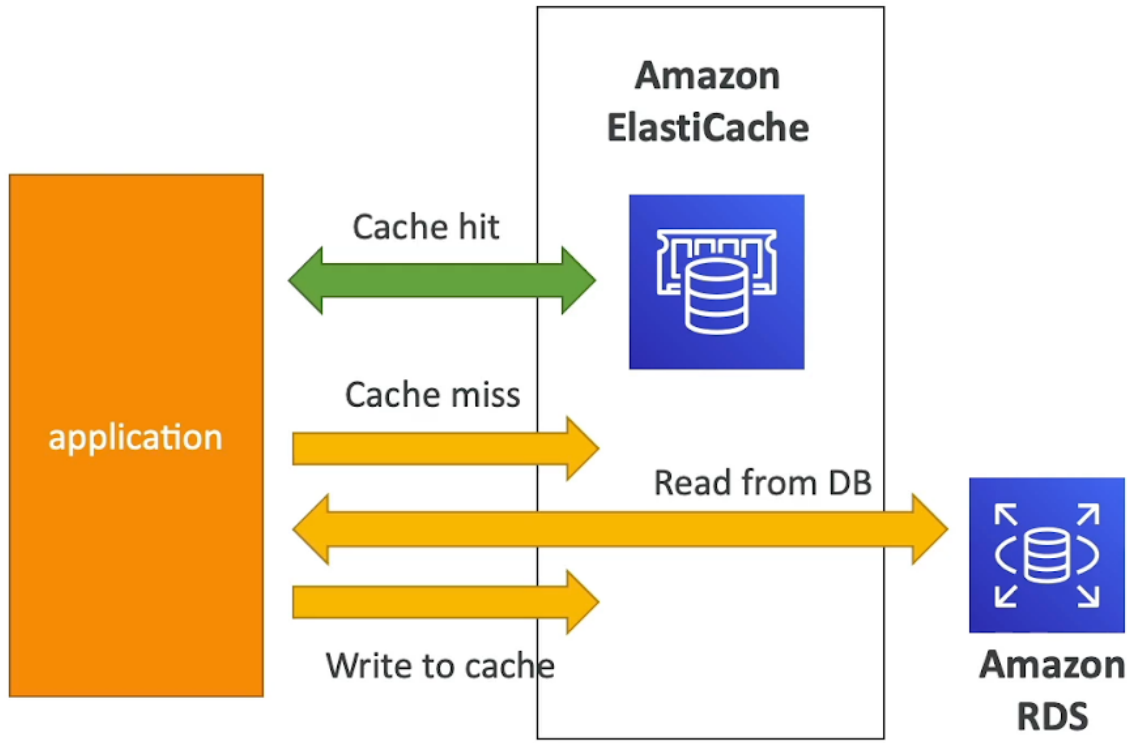
```
{
    "Version": "2012-10-17",
    "Id": "DenyAccessToUser",
    "Statement": [
        {
            "Sid": "DenyUserAccess",
            "Effect": "Deny",
            "Principal": {
                "AWS":
"arn:aws:iam::241597361541:user/USER1"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::lokeshava",
                "arn:aws:s3:::lokeshava/*"
            ]
        }
    ]
}
```

Amazon ElastiCache Overview

- The same way RDS is to get managed Relational Databases…
- ElastiCache is to get managed Redis or Memcached
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Helps make your application stateless
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups
- Using ElastiCache involves heavy application code changes

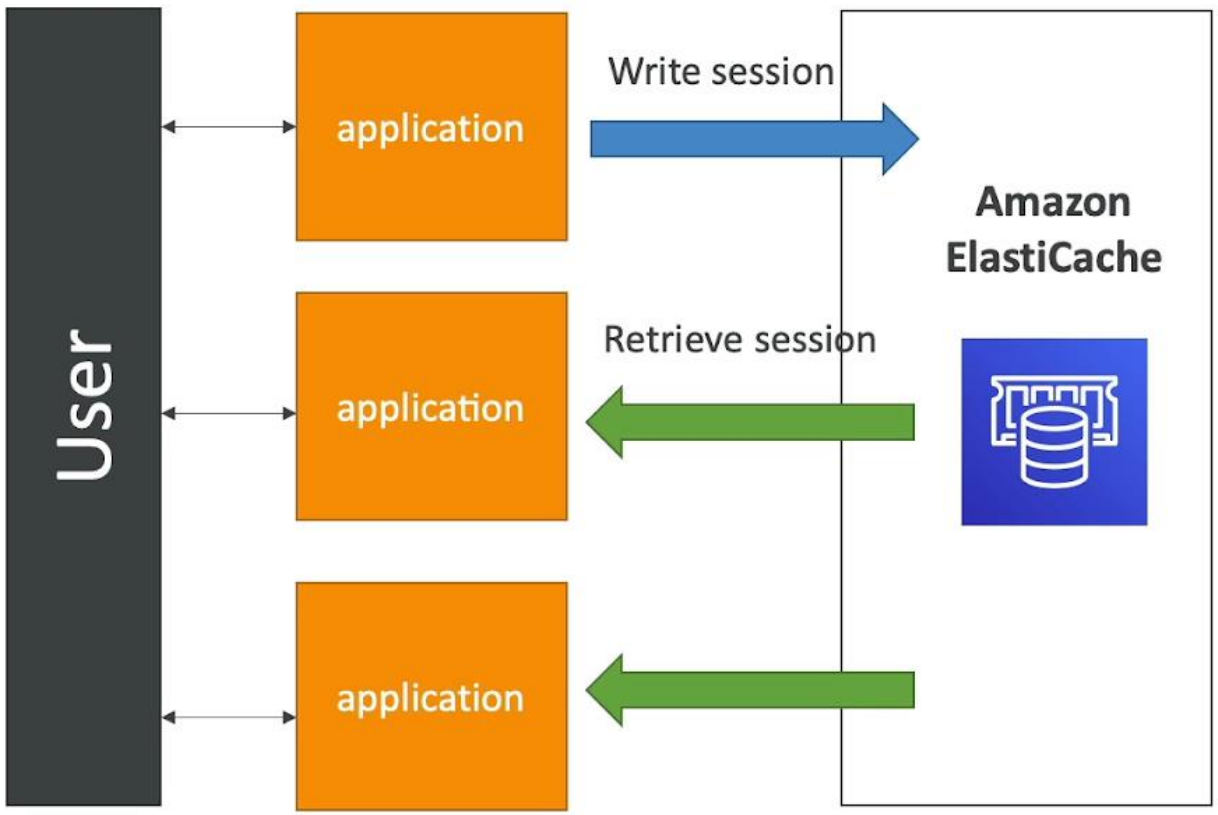
ElastiCache
Solution Architecture - DB Cache

- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.



ElastiCache
Solution Architecture – User Session Store

- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



ElastiCache – Redis vs Memcached

REDIS

- **Multi AZ** with Auto-Failover
- **Read Replicas** to scale reads and have **high availability**
- Data Durability using AOF persistence
- **Backup and restore features**

MEMCACHED

- Multi-node for partitioning of data (sharding)
- **No high availability (replication)**
- **Non persistent**
- **No backup and restore**
- Multi-threaded architecture

AWS CloudFormation

Managing your infrastructure as code

Infrastructure as Code

- Currently, we have been doing a lot of manual work
- All this manual work will be very tough to reproduce:
  - In another region
  - in another AWS account
  - Within the same region if everything was deleted

- Wouldn't it be great, if all our infrastructure was… code?
- That code would be deployed and create / update / delete our infrastructure

What is CloudFormation

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

How CloudFormation Works

- Templates have to be uploaded in S3 and then referenced in CloudFormation
- To update a template, we can't edit previous ones. We have to re-upload a new version of the template to AWS
- Stacks are identified by a name
- Deleting a stack deletes every single artifact that was created by CloudFormation.

Deploying CloudFormation templates

- Manual way:
  - Editing templates in the CloudFormation Designer
  - Using the console to input parameters, etc

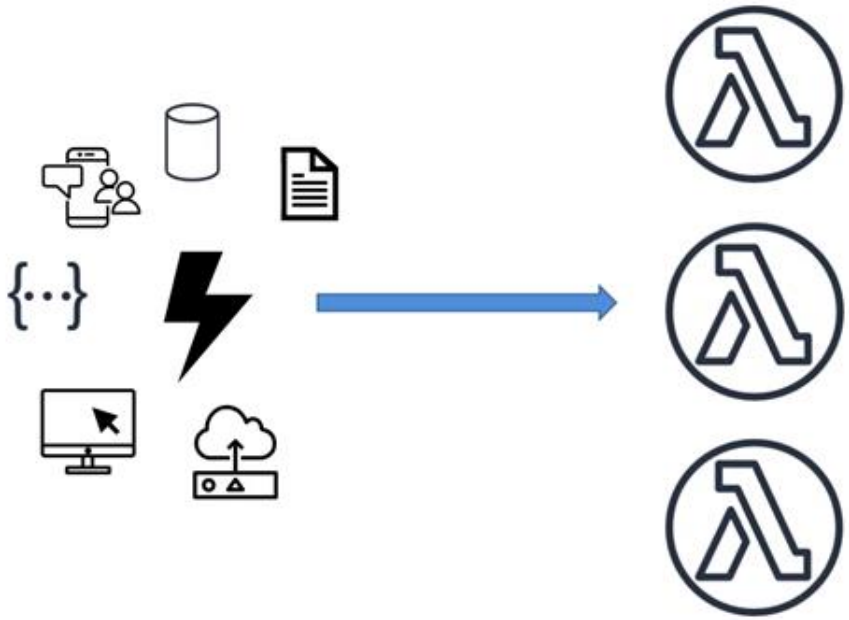
- Automated way:
  - Editing templates in a YAML file
  - Using the AWS CLI (Command Line Interface) to deploy the templates
  - Recommended way when you fully want to automate your flow

Serverless Computing – A 30,000 ft. View

Traditional Architecture

Traditional Architecture

How Serverless Works?



Additional Serverless Services in AWS


Amazon S3


Amazon SNS


Amazon SQS


AWS Step Functions


Amazon Kinesis


Amazon Athena


AWS X-Ray


Amazon CloudWatch


Amazon Cognito


AWS Tools and SDKs

AWS Lambda

It's a serverless world

What's serverless?

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

Why AWS Lambda

Amazon EC2

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

Amazon Lambda

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

Benefits of AWS Lambda

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

AWS Lambda language support

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

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

AWS Lambda Integrations
Main ones

API Gateway

Kinesis

DynamoDB

S3

CloudFront

CloudWatch Events
EventBridge

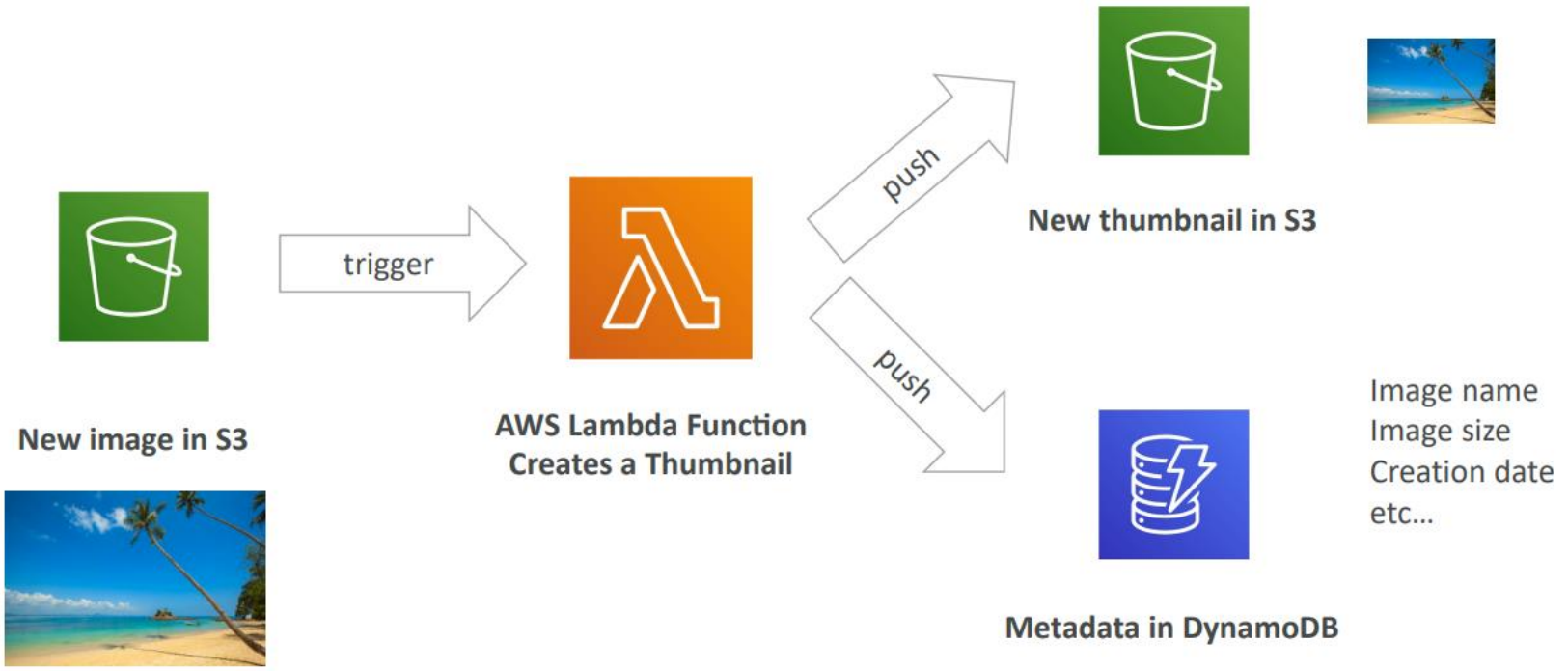
CloudWatch Logs

SNS

SQS

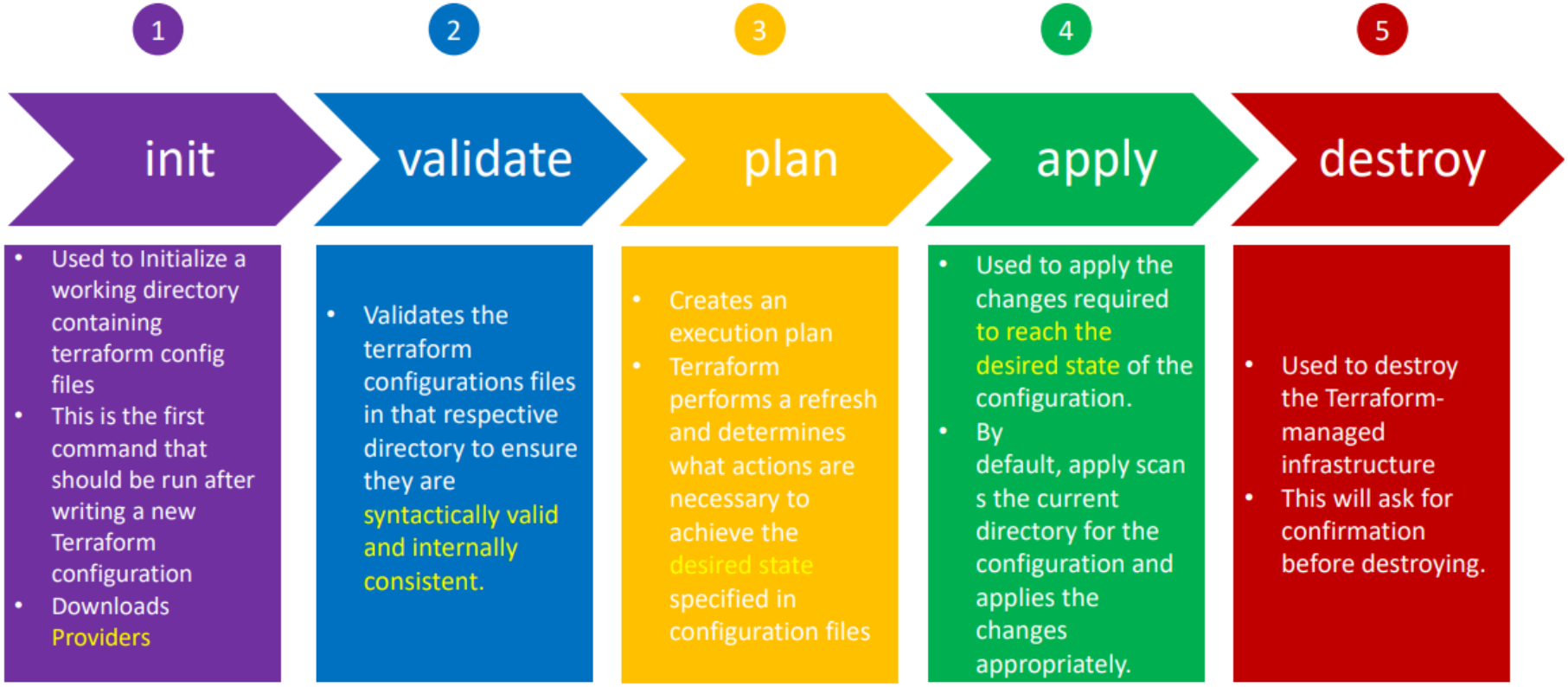
Cognito

Example: Serverless Thumbnail creation

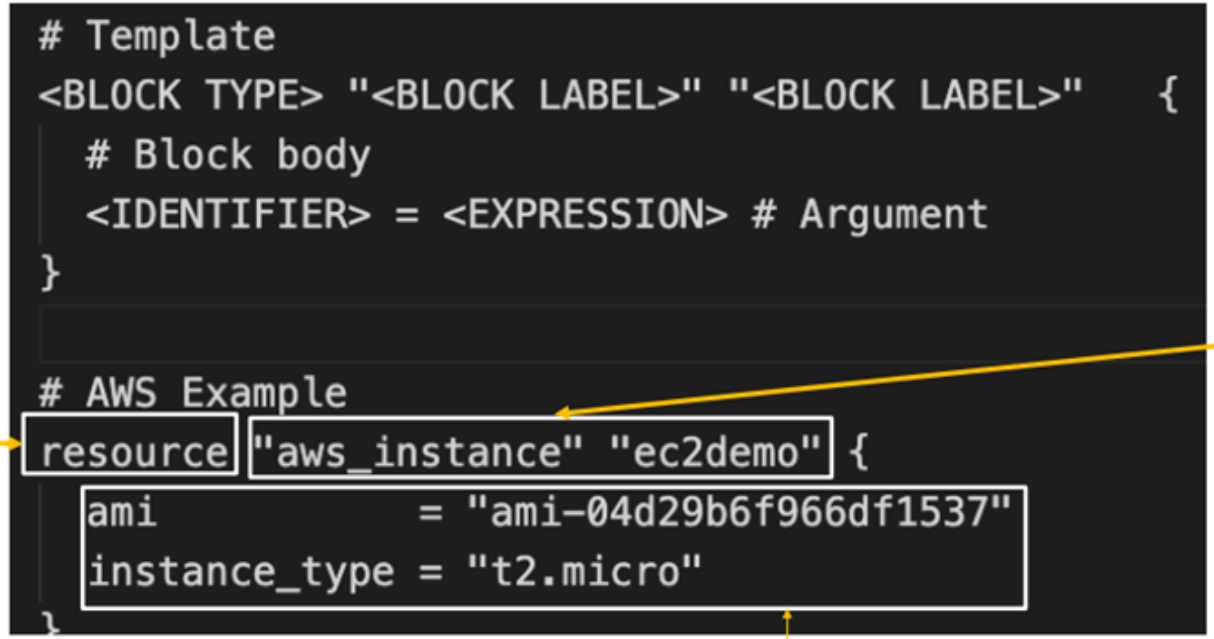


Example: Serverless CRON Job

Terraform Workflow



Terraform Language Basics – Configuration Syntax



Block Labels

Block Type

Top Level & Block inside Blocks

Based on Block Type block labels will be 1 or 2
Example:
Resource – 2 labels
Variables – 1 label

Top Level Blocks: resource, provider

Block Inside Block: provisioners, resource specific blocks like tags

Arguments

Terraform Language Basics – Configuration Syntax