

SECTION 1

Let's Get Started!

Demo: Let's See AWS in Action!



SECTION 2

Create a Free Tier Account on AWS

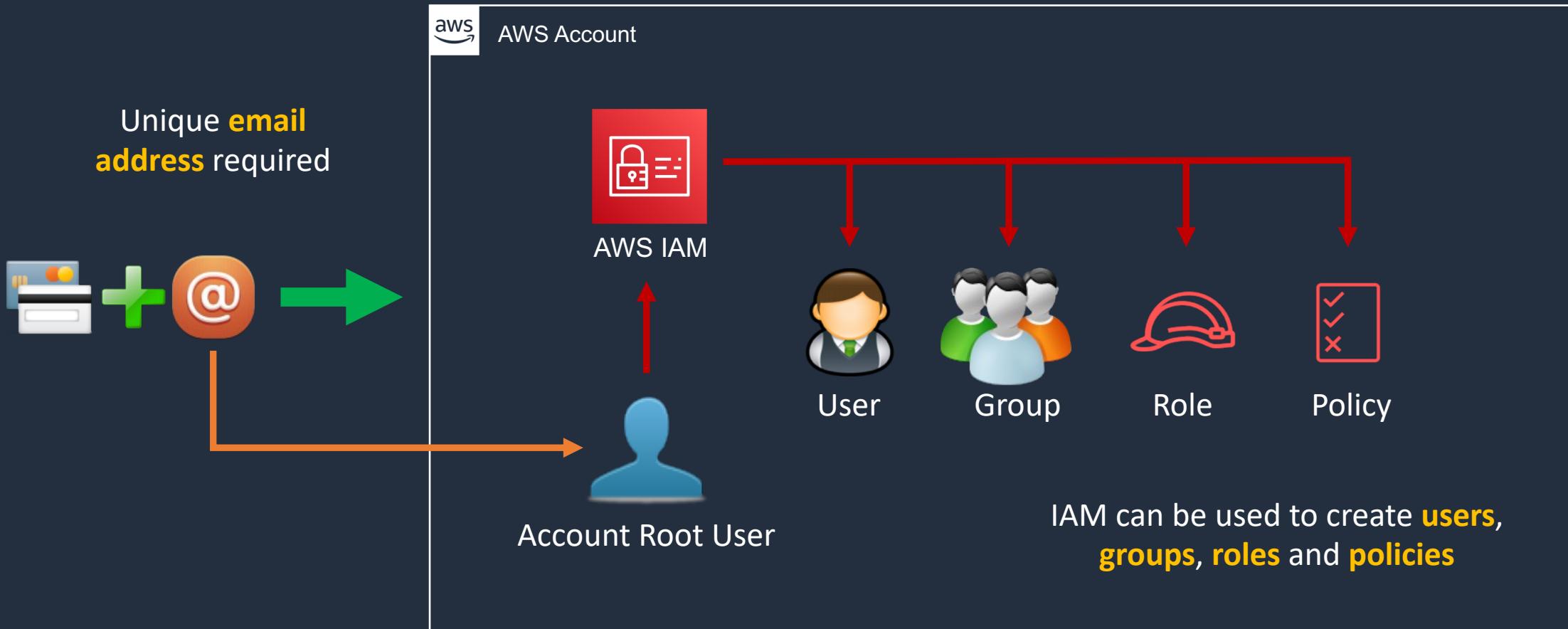
AWS Account Overview



AWS Account Overview



It's an IAM best practice to create **individual users** and to avoid using the **Root** account



The Root user has **full control** over the account

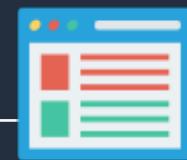
AWS Account Overview



Authentication: IAM principals authenticate to IAM using the console, API, or CLI

AWS Account

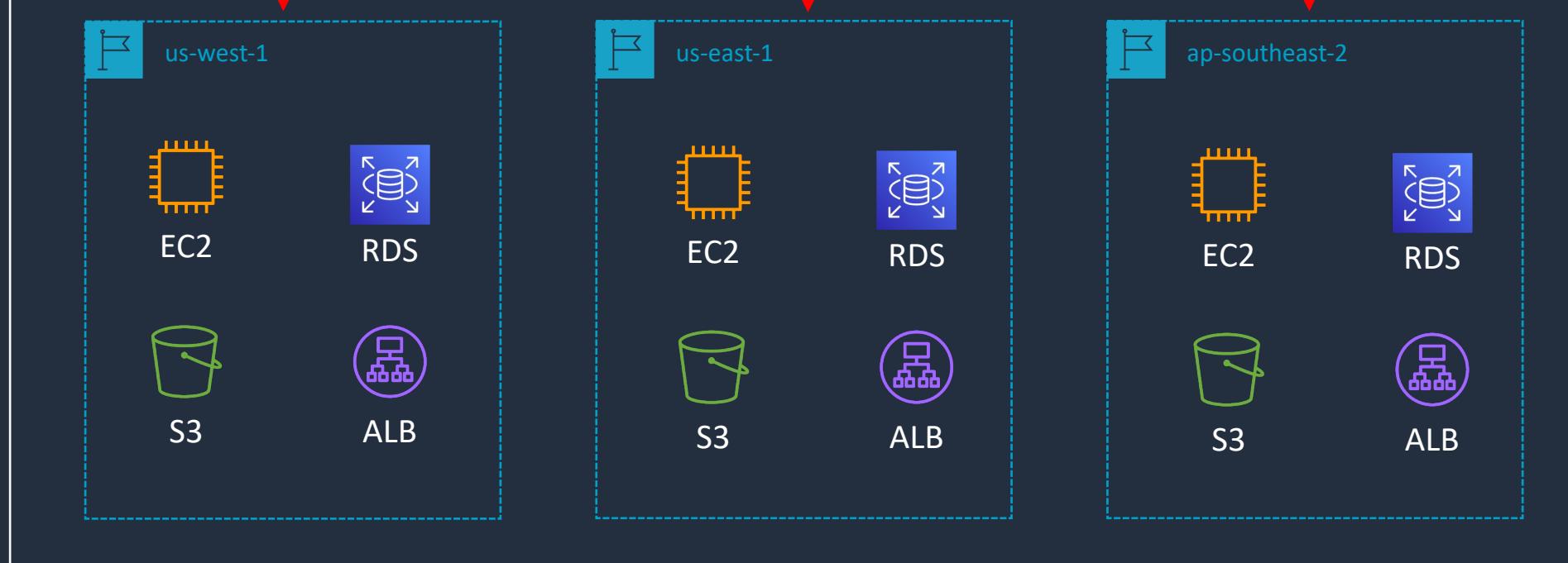
Authorization: IAM principals can then create resources across AWS Regions



AWS Management Console



AWS IAM



All AWS **identities** and **resources** are created within the AWS account

Create your AWS Free Tier Account



What you need...



Credit card for setting up the account and paying any bills



Unique email address for this account

john@gmail.com



Check if you can use a **dynamic alias** with an existing email address



john+ACCOUNT-ALIAS-1@gmail.com

john+ACCOUNT-ALIAS-2@gmail.com



AWS account name / alias



Phone to receive an **SMS** verification code

Configure Account and Create a Budget and Alarm



Account Configuration

- Configure **Account Alias**
- Enable access to billing for **IAM users**
- Update **billing preferences**
- Create a **budget and alarm**

Install Tools



SECTION 3

IT Fundamentals

Client - Server Computing

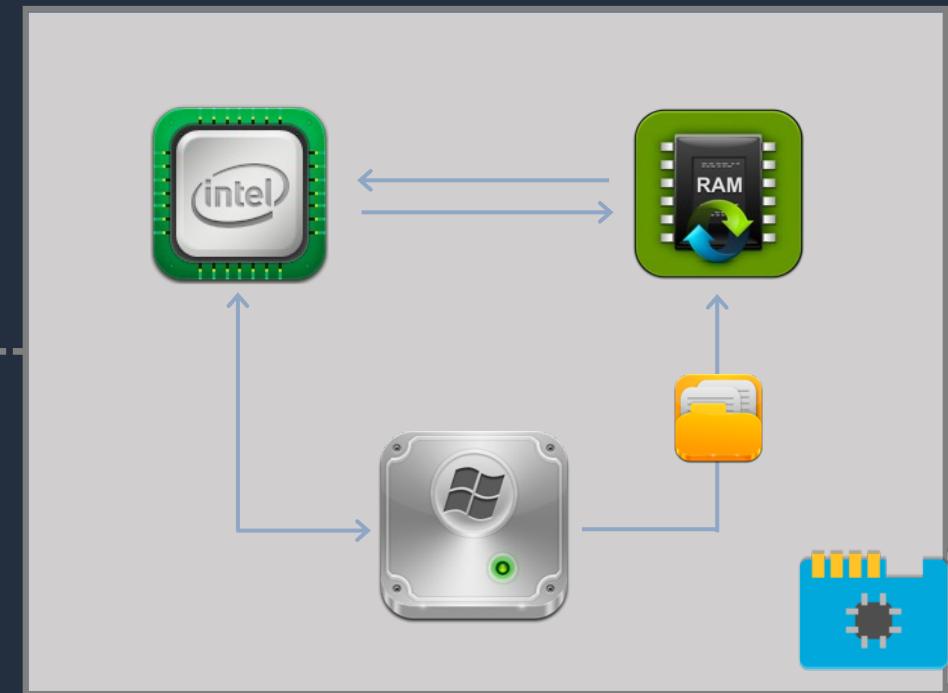




Basic Architecture of a Computer



Central Processing Unit (CPU)



RAM is **non-persistent** storage or “memory”

Network Interface Cards (NICs) connect computers to networks

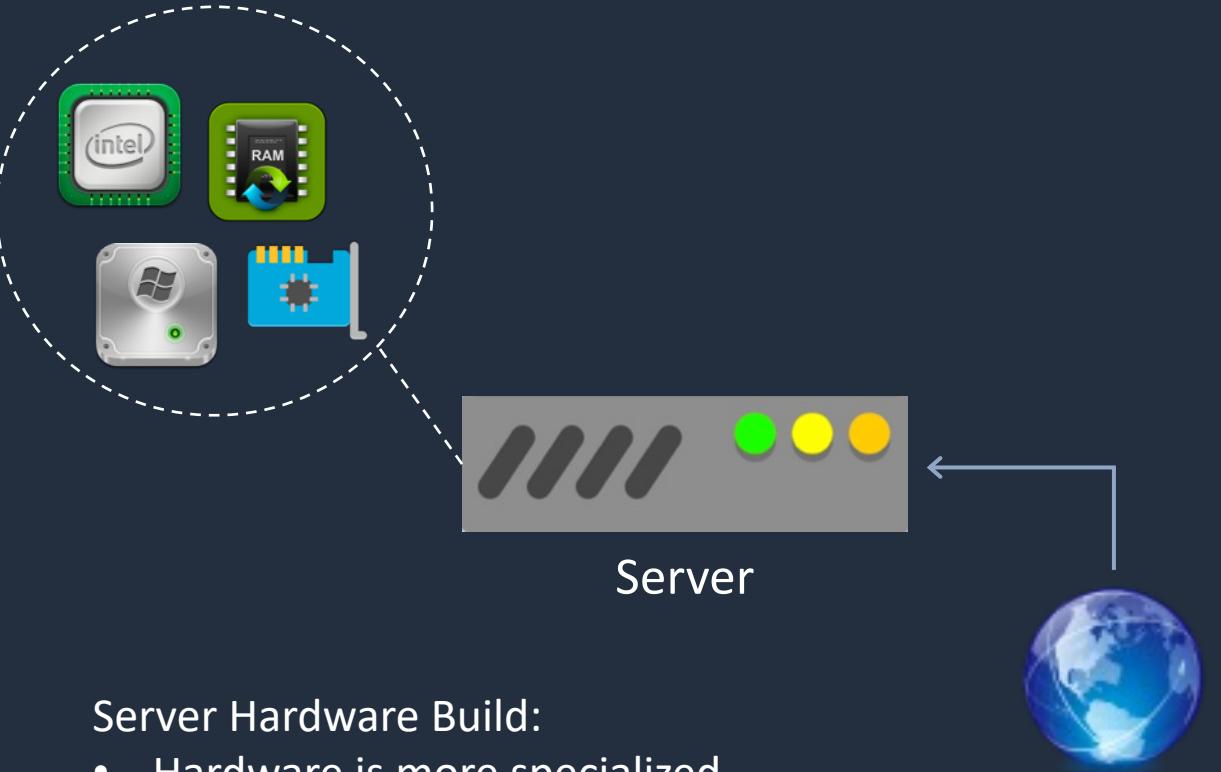
Hard Disk Drives (HDDs) offer **persistent** storage



Measurements for Components

Component	Unit of Measurement
Central Processing Unit (CPU)	Gigahertz (GHz)
Random Access Memory (RAM)	Gigabytes (GB)
Hard Disk Drive (HDD) / Solid State Drive (SSD)	Gigabytes (GB)
Network Interface Card (NIC)	Megabits per second (Mbps) Gigabits per second (Gbps)

Servers vs Desktops/Laptops



Server Hardware Build:

- Hardware is more specialized
- Much higher prices compared to desktops / laptops
- Includes redundancy



Servers can be used by many users over a network





Clients and Servers

Cloud Computing



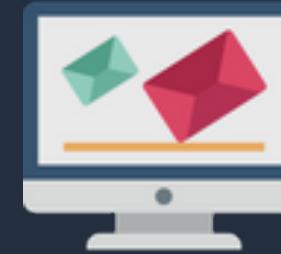
Servers

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

Client devices are connected via the **Internet**



Cloud Networking



Client Devices

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

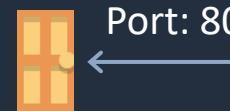


Connecting to Services

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



Web Server



Port: 80

Protocol: HTTP



File Server



Port: 445

Protocol: SMB



Email Server



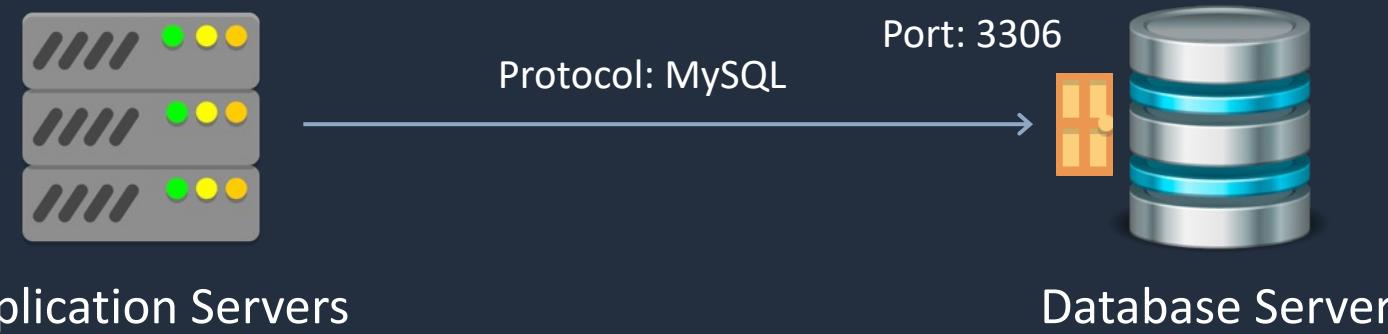
Port: 25

Protocol: SMTP





Server to Server Connectivity



Storage - Block vs File vs Object





Hard Drives



Hard Disk Drive (HDD)

- Also known as magnetic drives
- Older technology
- Much slower than SSD
- Much cheaper than SSD



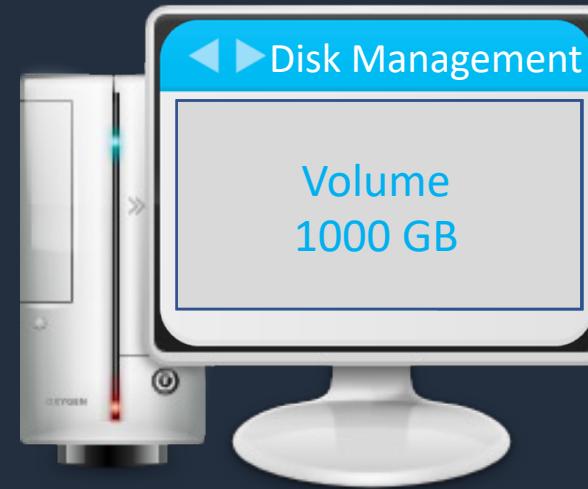
Solid State Drive (SSD)

- Uses flash memory
- Newer technology
- MUCH faster than HDD
- More expensive than HDD



Hard Drives

Hard drives are
block-based
storage systems

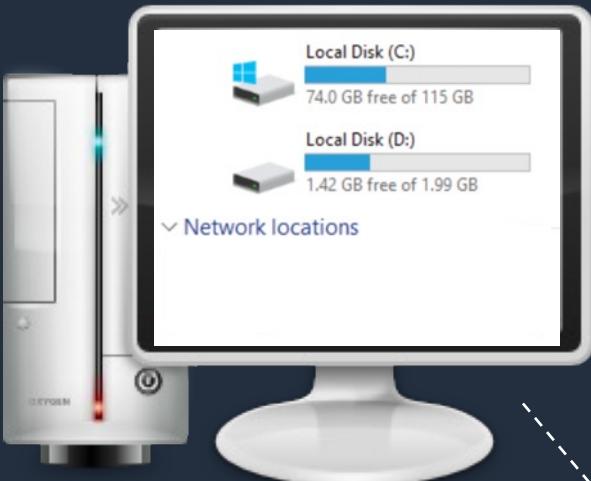


The Operating System
(OS) sees a volume. A
volume can be
partitioned and
formatted

Hard drives are block-based storage systems



Network Attached Storage



The Operating System (OS)
sees a filesystem that is
mapped to a local drive letter



Network Switch



Network Attached
Storage Server (NAS)



The NAS “shares”
filesystems over the
network

Object Storage Systems

User uploads **objects** using a web browser



The **HTTP protocol** is used with a **REST API** (e.g. GET, PUT, POST, SELECT, DELETE)

There is no hierarchy of objects in the container



Object Storage Container

Objects can be files, videos, images etc.

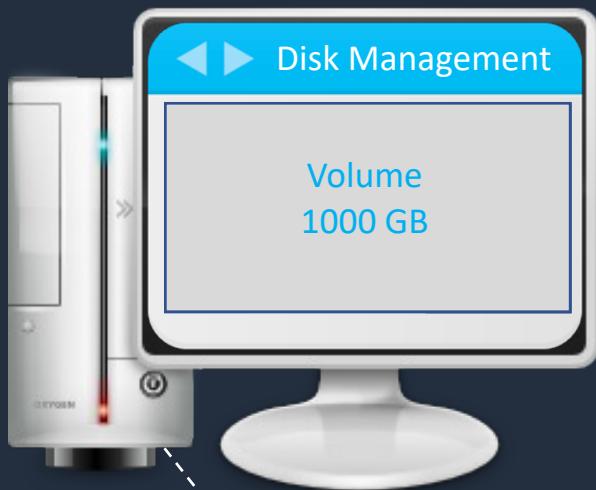
Block vs File vs Object Storage

The OS sees **volumes** that can be partitioned and formatted

A filesystem can be shared by many users

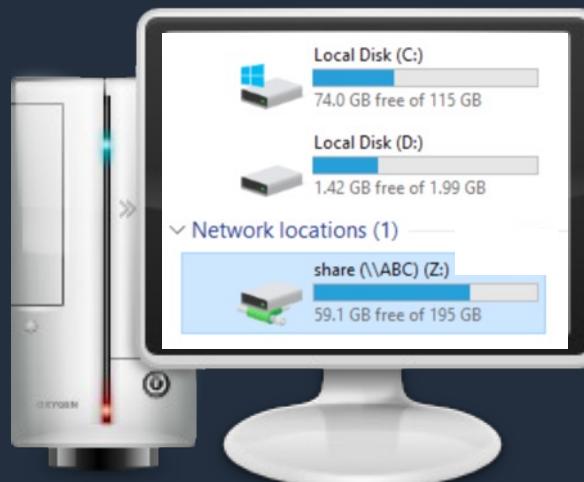
Massively scalable and low cost

Block Storage



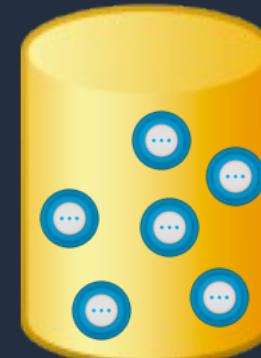
The OS reads/writes at the **block level**. Disks can be internal, or network attached

File Storage



A filesystem is “**mounted**” to the OS using a network share

Object Storage



Object Storage Container

There is **no hierarchy** of objects in the container

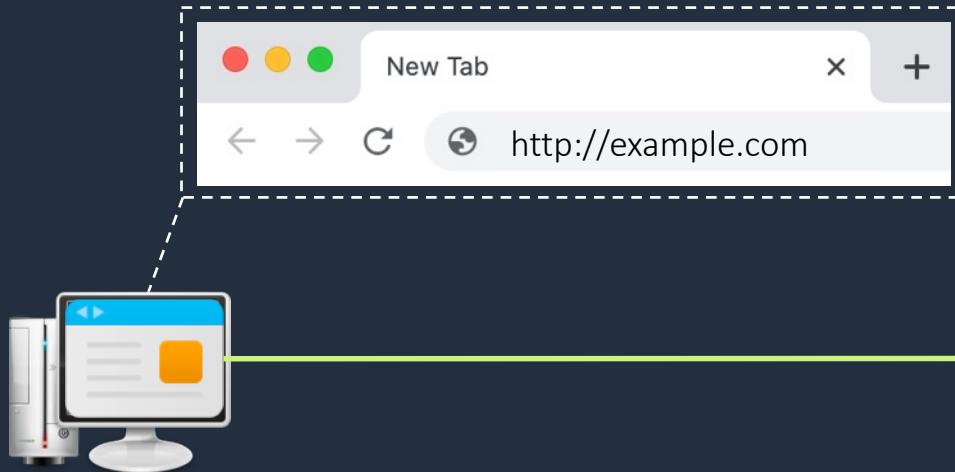
Uses a **REST API**

IP Addresses and DNS





What is an IPv4 address?

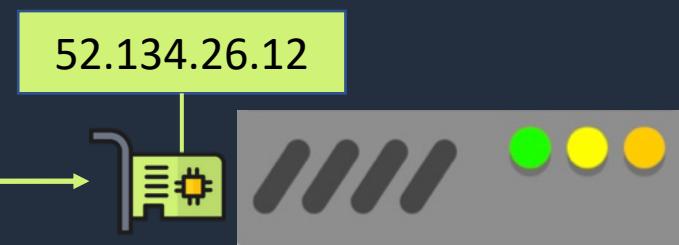


What's the **IP address** for example.com?



IP addresses are the addresses computers use to communicate

Connection is made using **IP address**



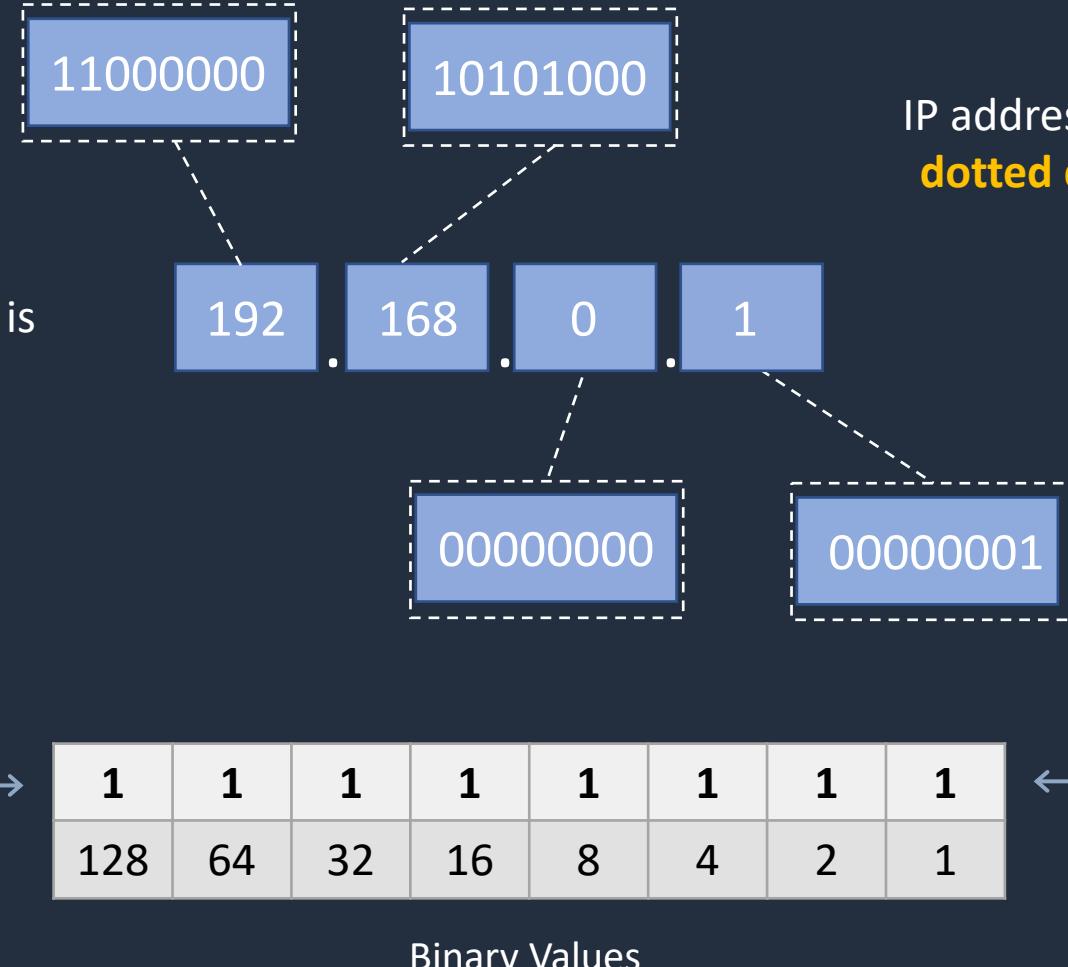
Name	Type	Value
example.com	A	52.134.26.12
test.com	A	137.10.47.51

DNS Zone File



Structure of an IPv4 Address

Each part of the address is a **binary octet**

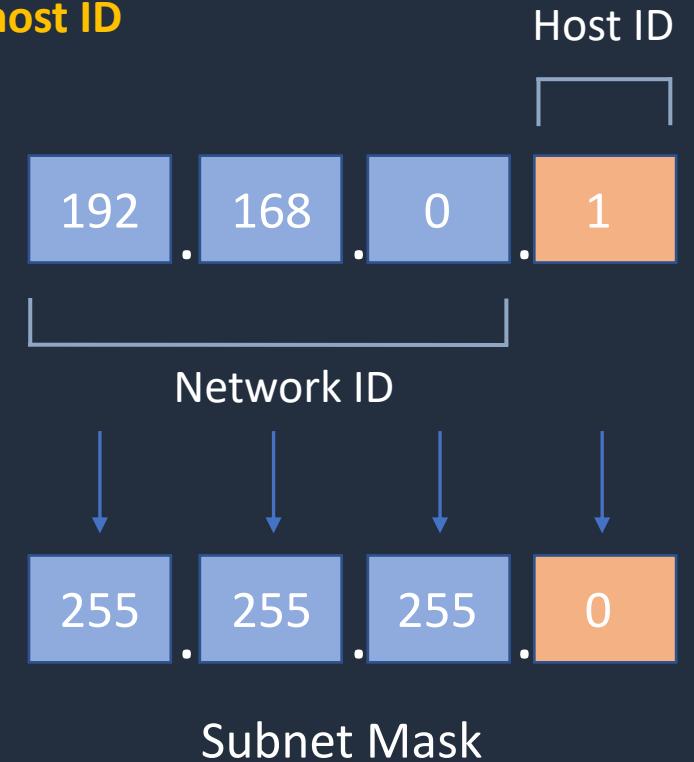


IP addresses are written in
dotted decimal notation



Networks and Hosts

An IPv4 address has a
network and **host ID**



The **subnet mask** is used to
define the **network** and **host
ID**



Networks and Hosts

Network

192	168	0	0
-----	-----	---	---

Subnet Mask

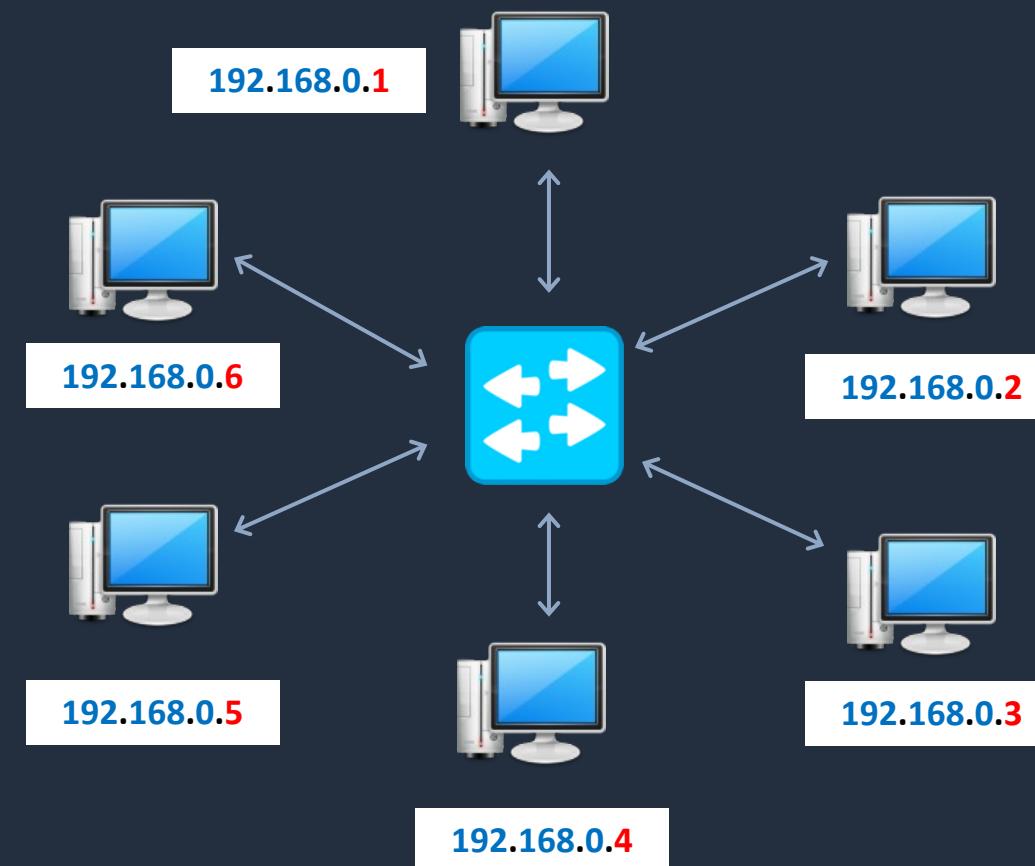
255	255	255	0
-----	-----	-----	---

24 bits

192.168.0.0/24

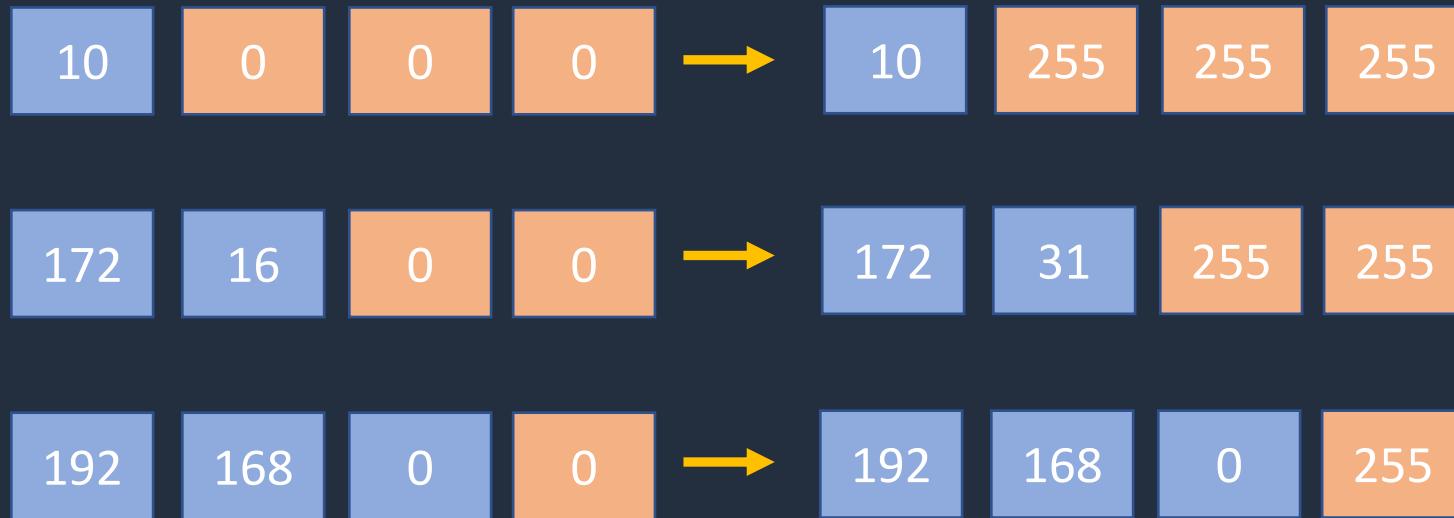
A **network** and **subnet mask** can also be written in this format

All computers share the same **network ID** and have a unique **host ID**



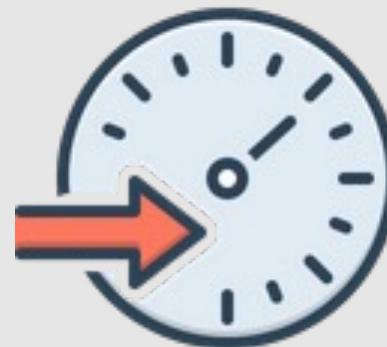


Private IP Address Ranges



These addresses are reserved
for **private use** according to
IETF RFC-1918

Bandwidth and Latency





Bandwidth and Latency

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

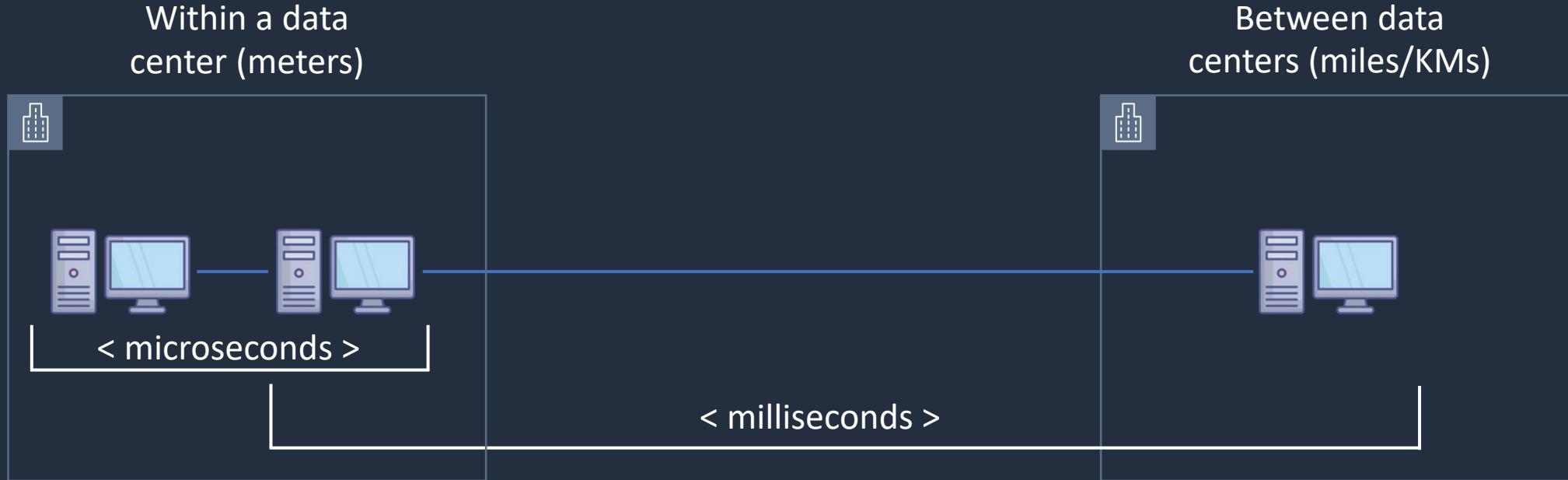


Bandwidth can be considered the **width** of the communication band

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

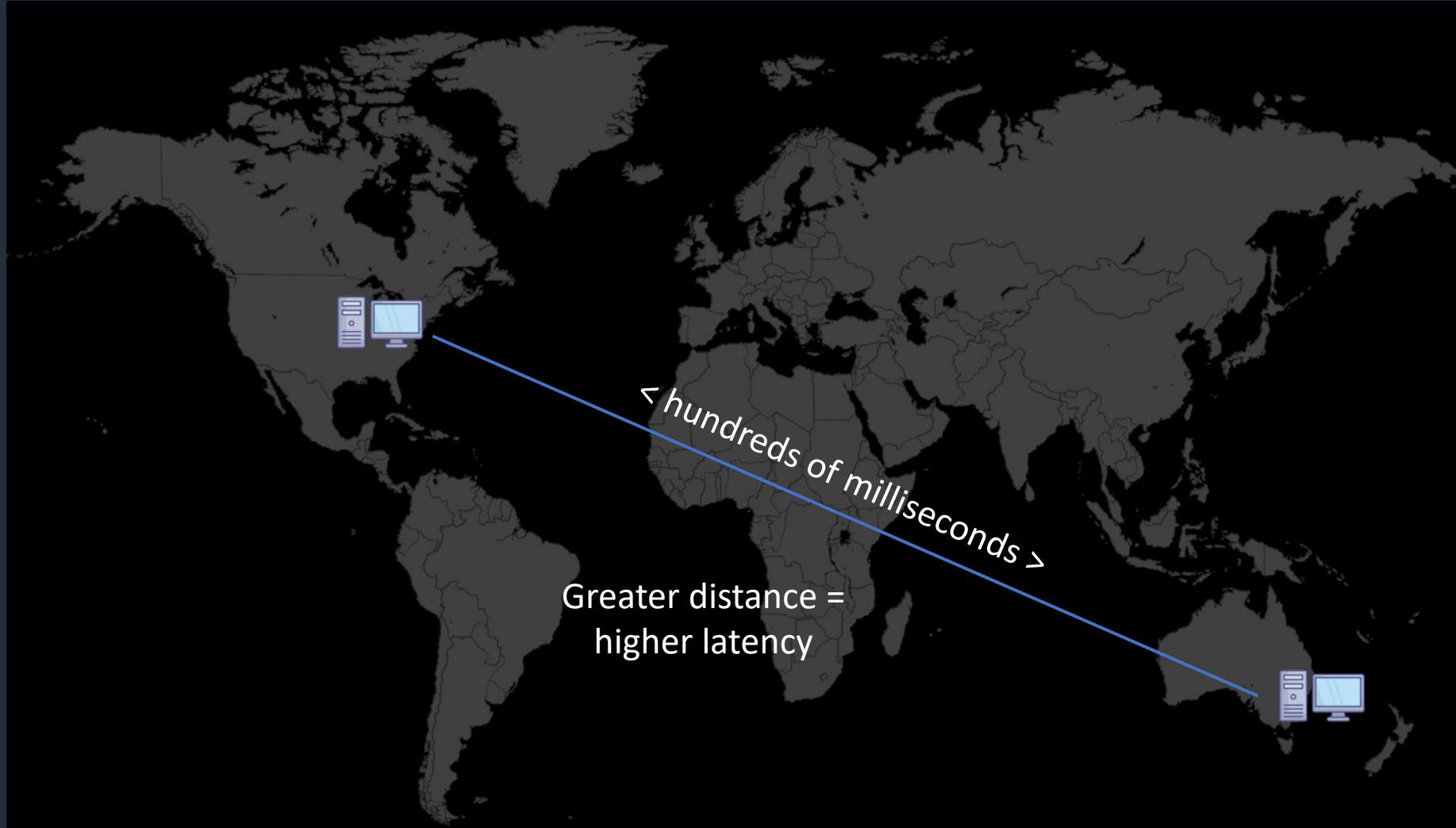


Bandwidth and Latency

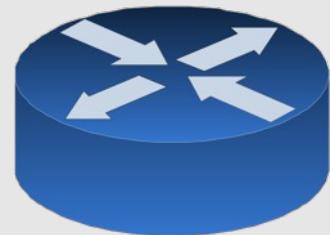




Bandwidth and Latency



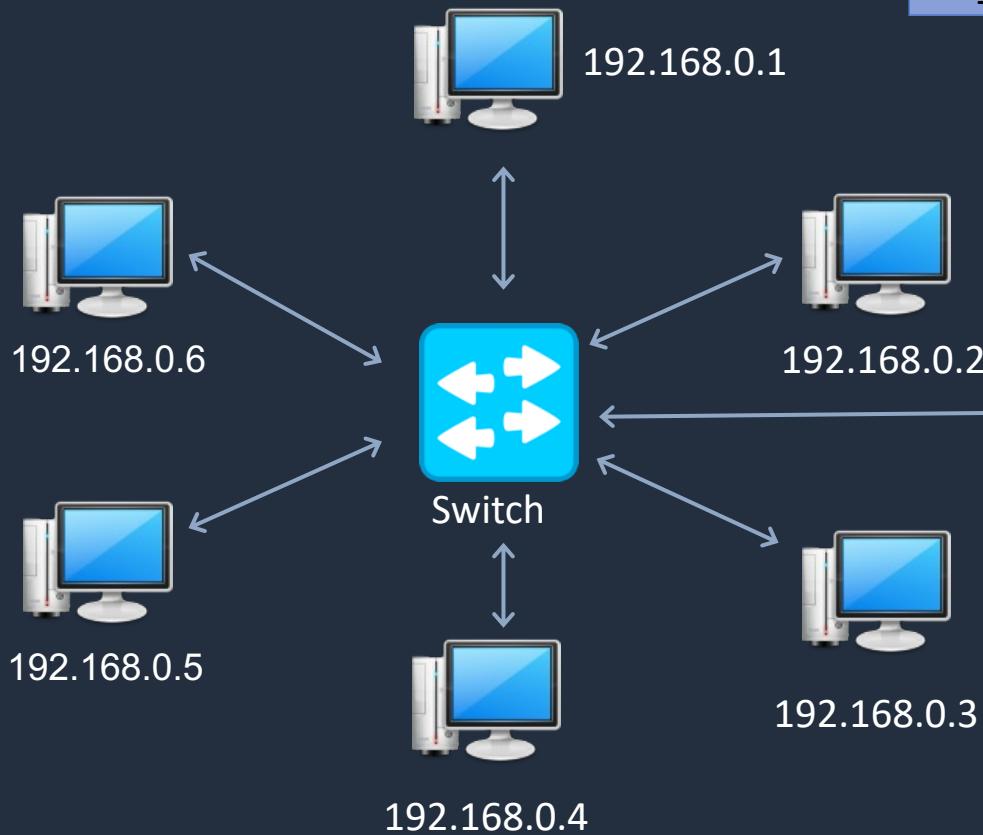
Networking – Routers, Switches and Firewalls





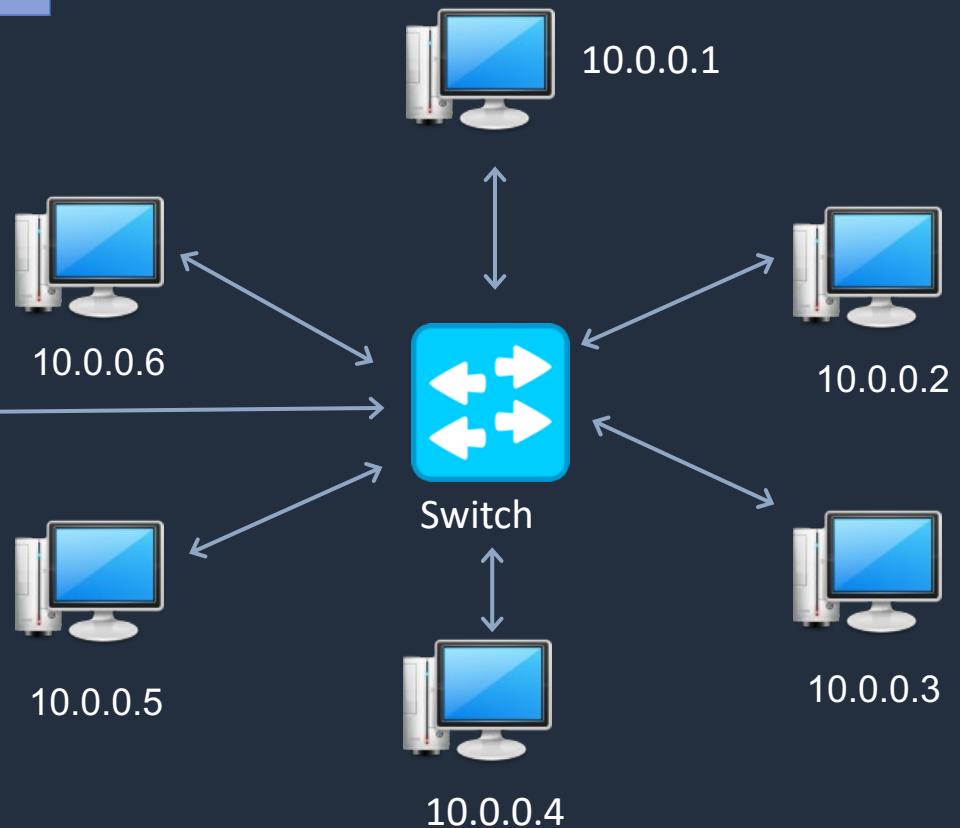
Routers and Switches

IP Subnet A: 192.168.0.0/24



Destination	Interface
192.168.0.0/24	eth0
10.0.0.0/24	eth1

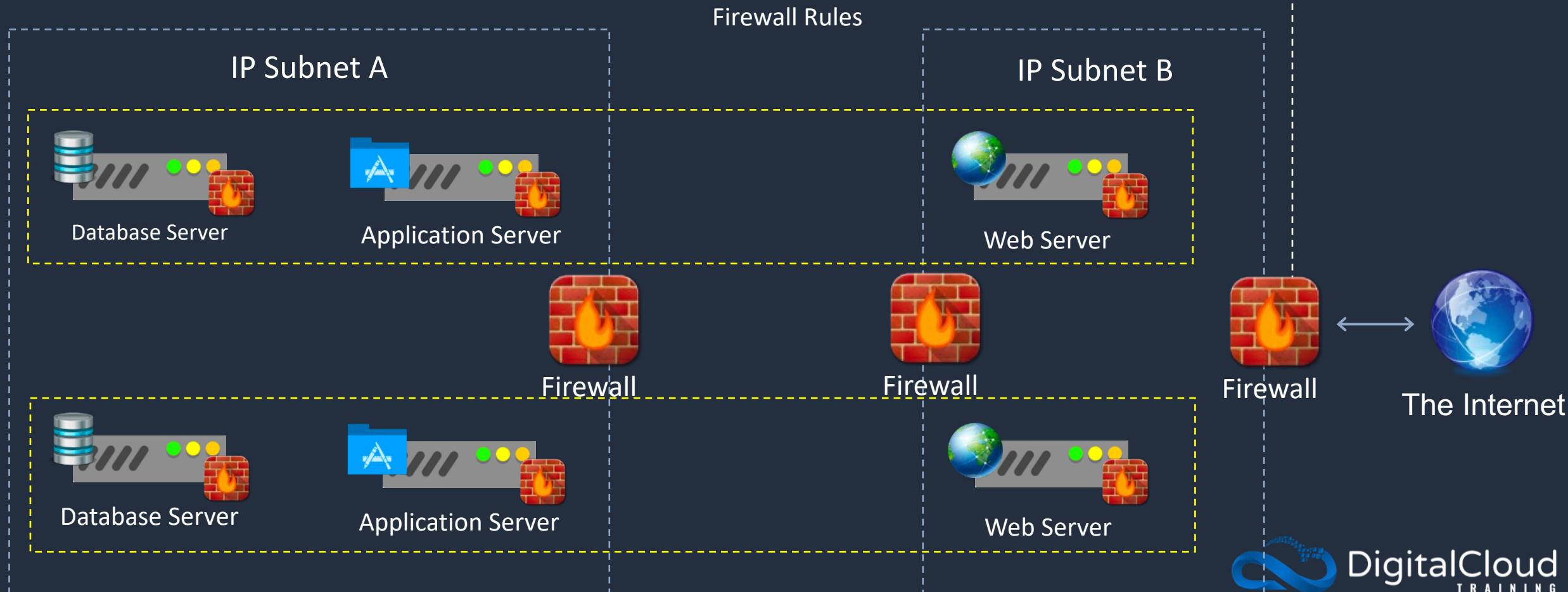
IP Subnet B: 10.0.0.0/24





Firewalls

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

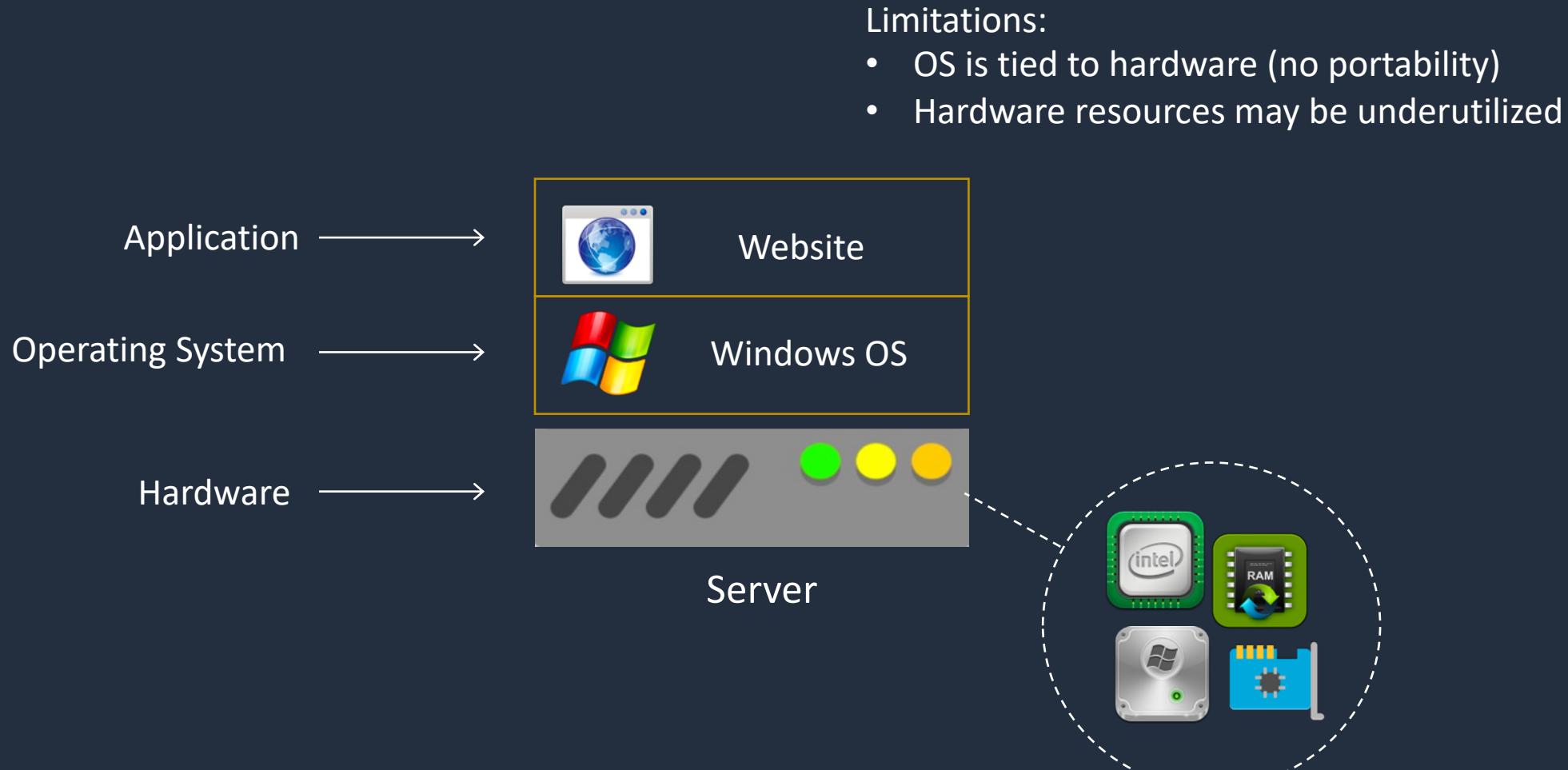


Server Virtualization and Containers





Server without Virtualization

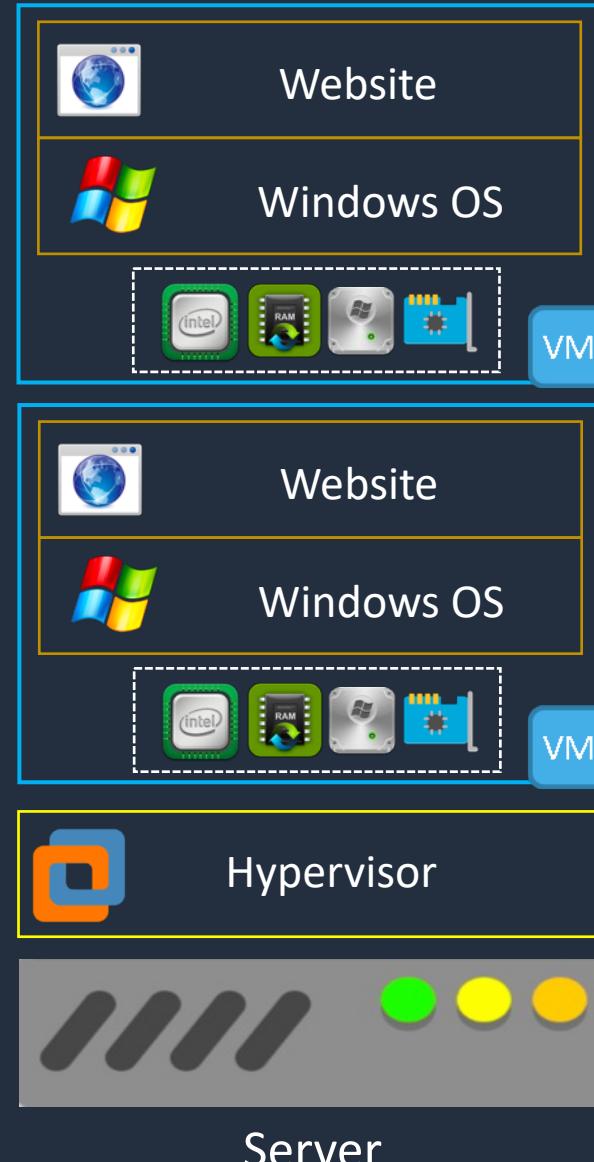




Server with Virtualization

This is known as a
virtual server or
"virtual machine"

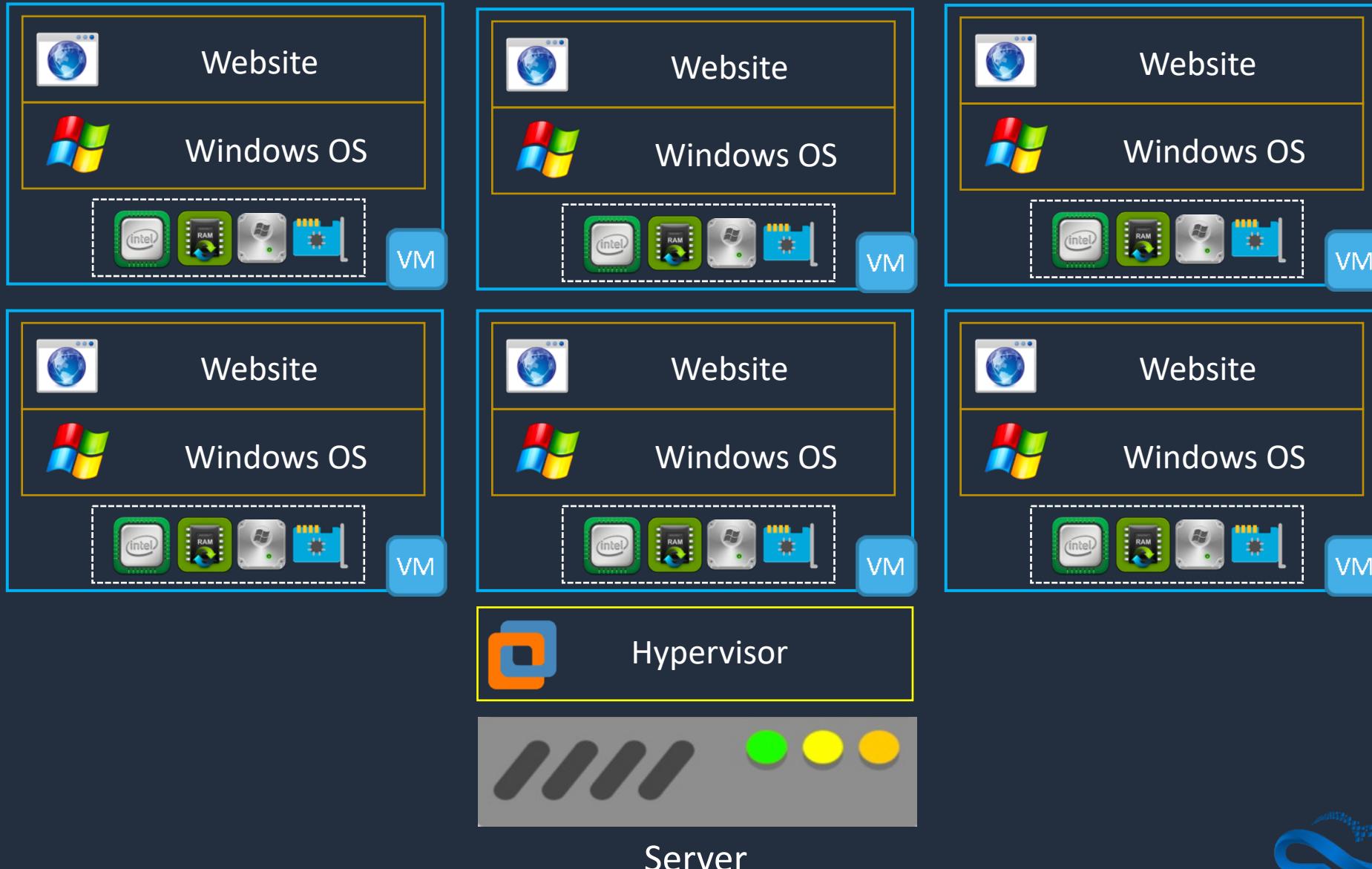
The **hypervisor** creates a
layer of abstraction



Many VMs can run on the
same **physical hardware**

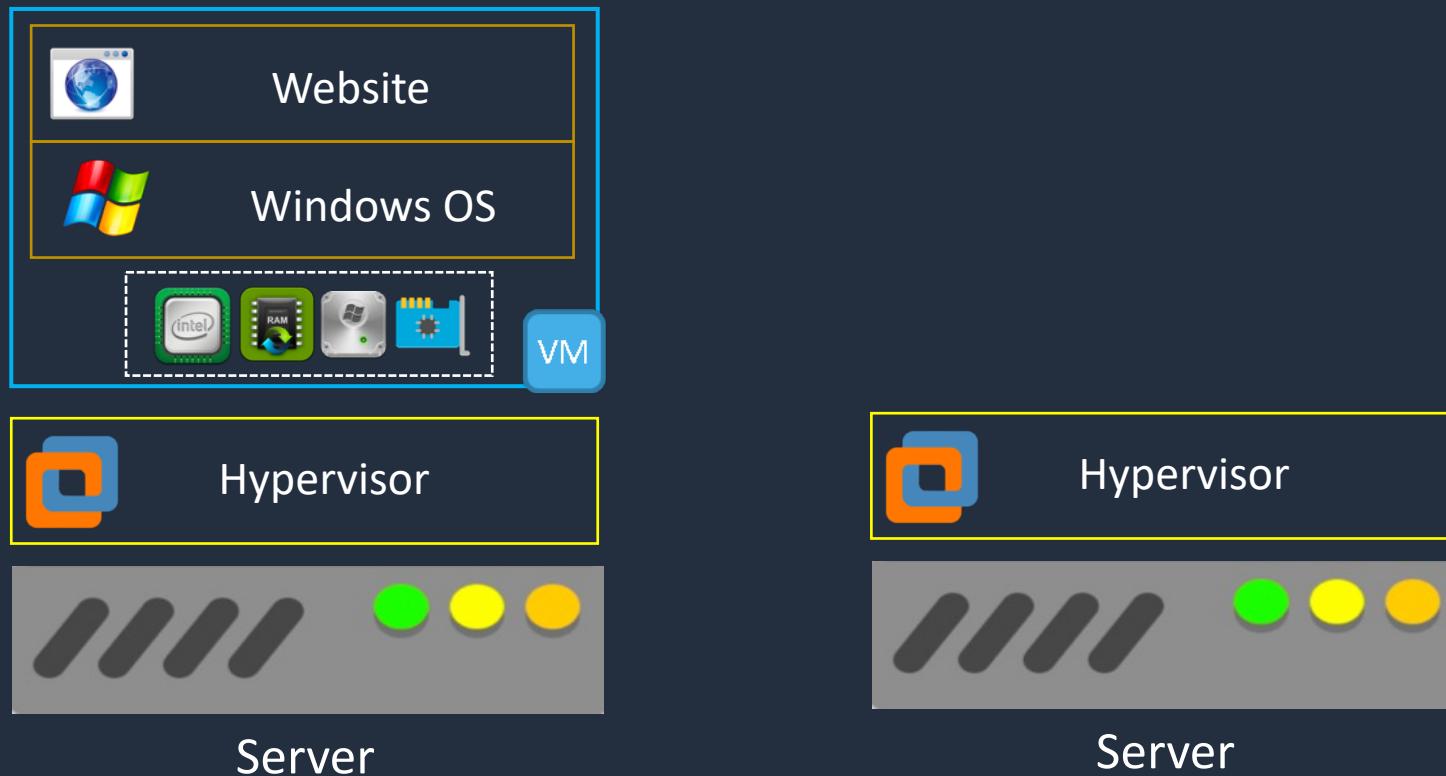
Virtual hardware is
presented to the OS

Server with Virtualization



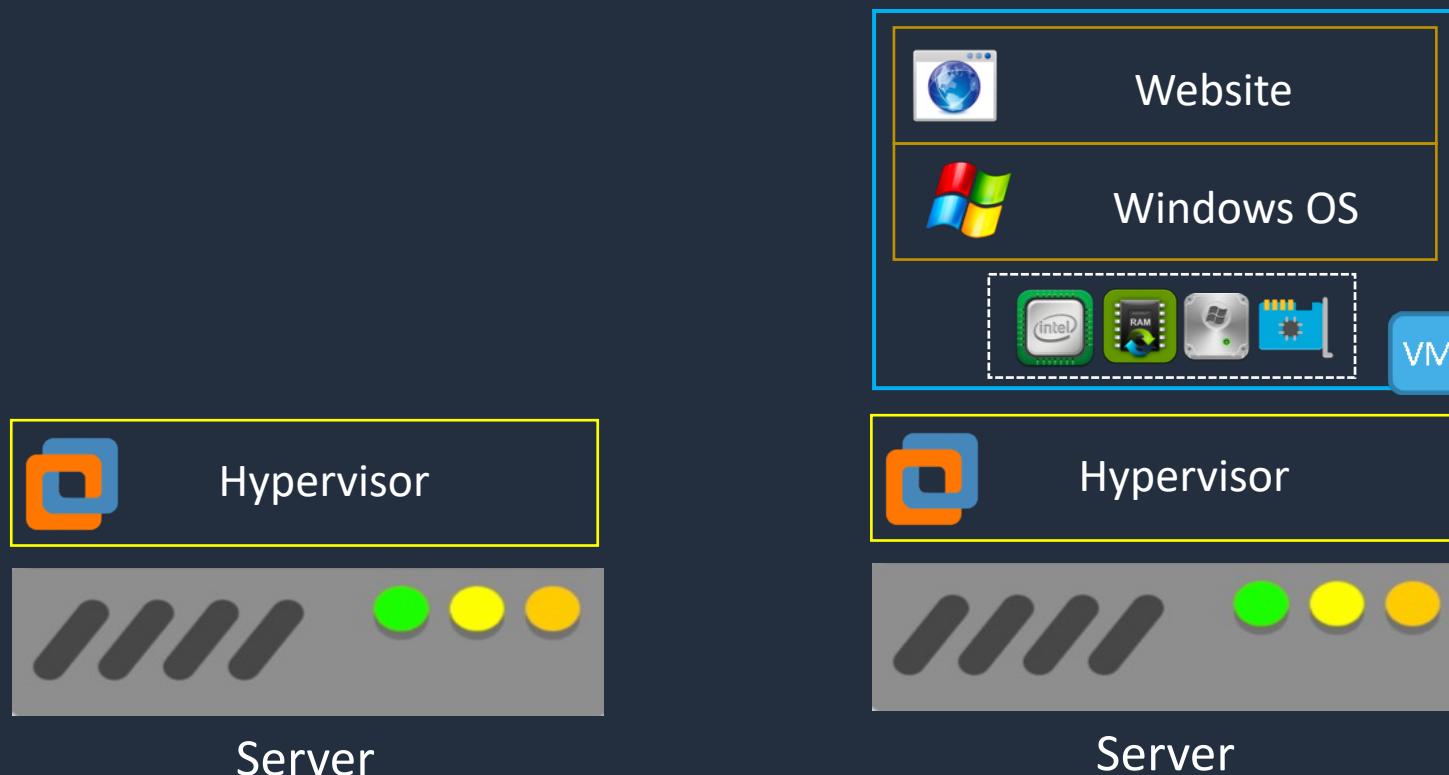


Server Virtualization: Portability





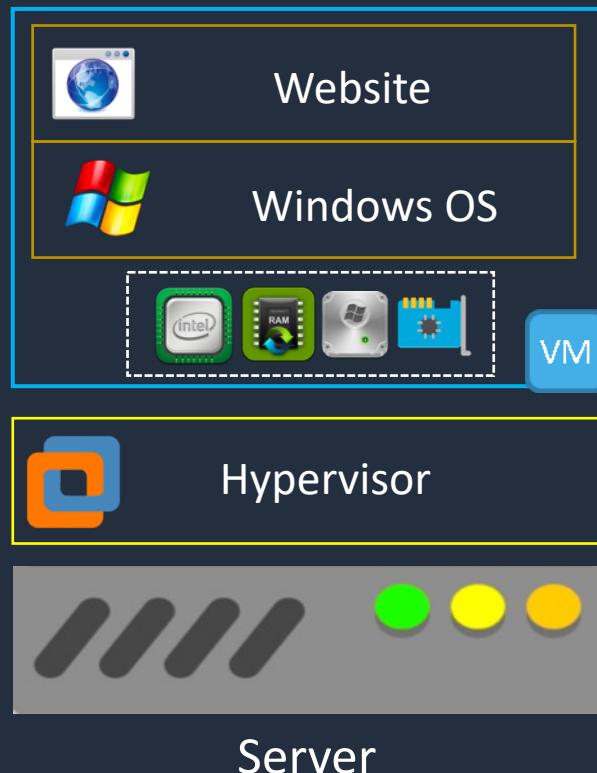
Server Virtualization: Portability





Docker Containers

Every VM needs an operating system
which uses significant resources





Docker Containers

Containers start up very **quickly**

A container includes all the code, settings, and dependencies for running the application



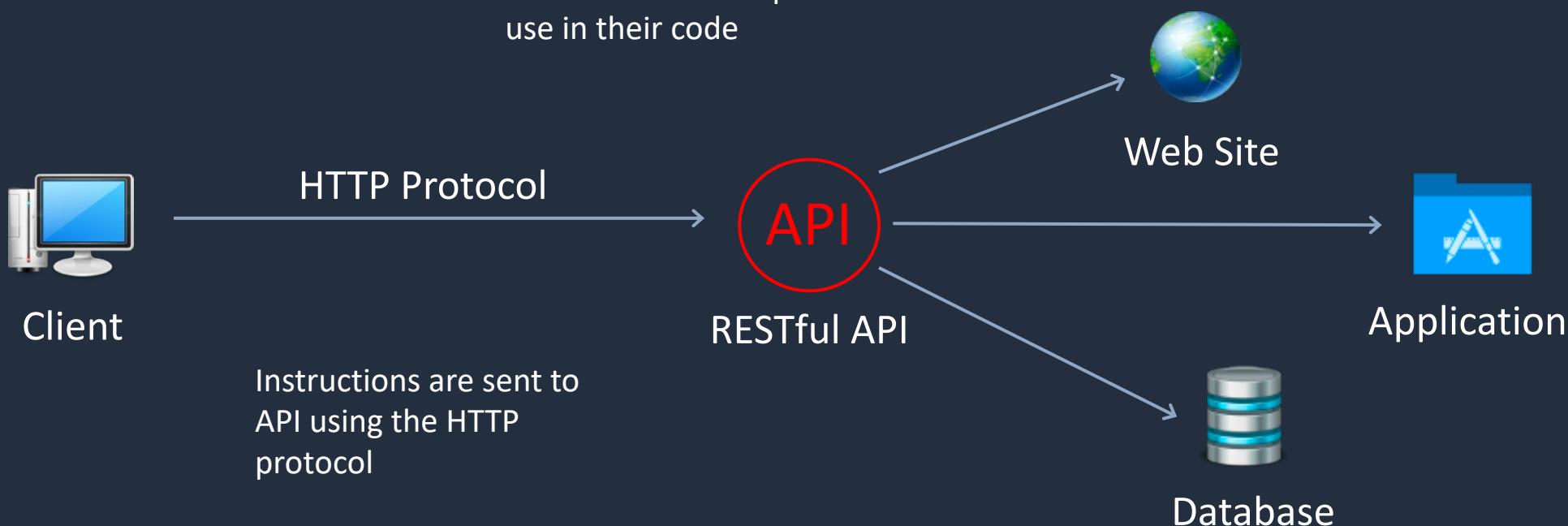
Containers are very **resource efficient**

Each container is **isolated** from other containers

Application Programming Interfaces (APIs)

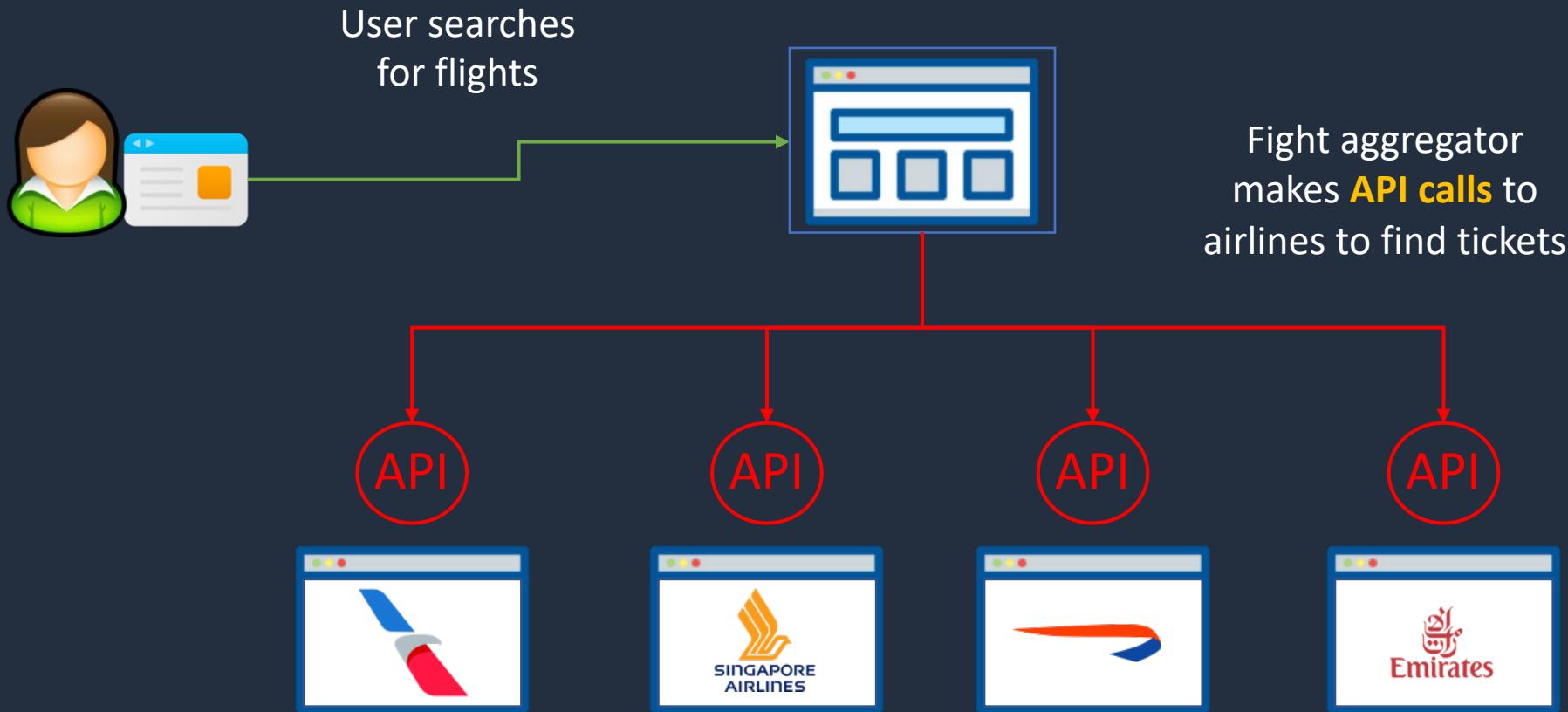


Application Programming Interfaces (APIs)



Flight Aggregator Example

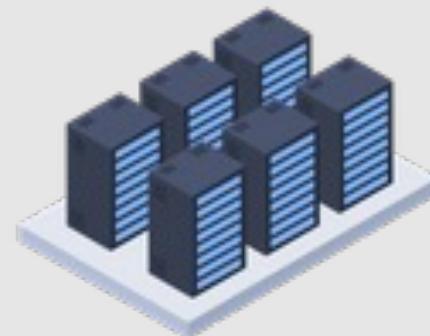
Flight aggregator such as
Momondo or Skyscanner



SECTION 4

Cloud Computing Concepts

Legacy IT / Traditional IT

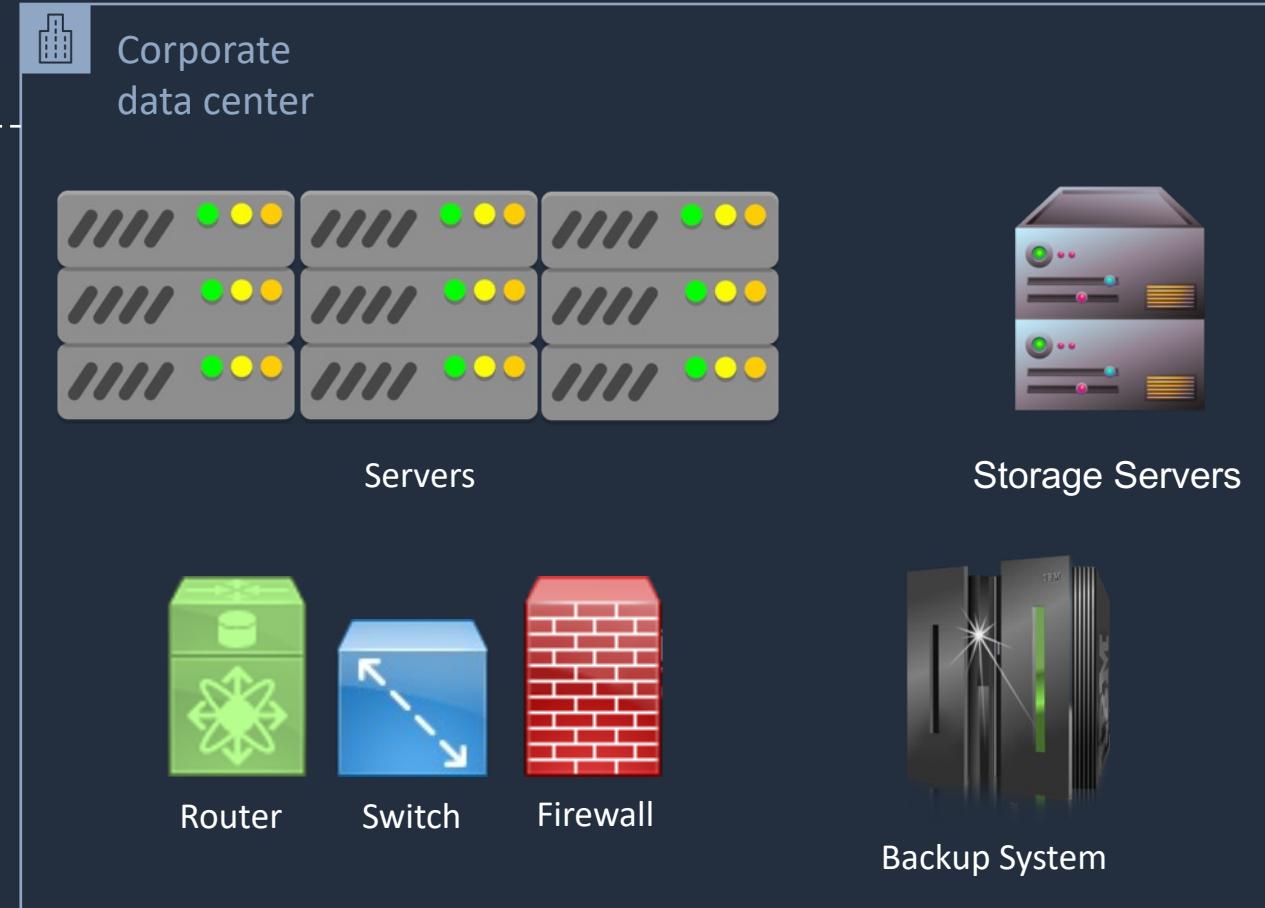




Traditional IT



This model is very **capital** intensive

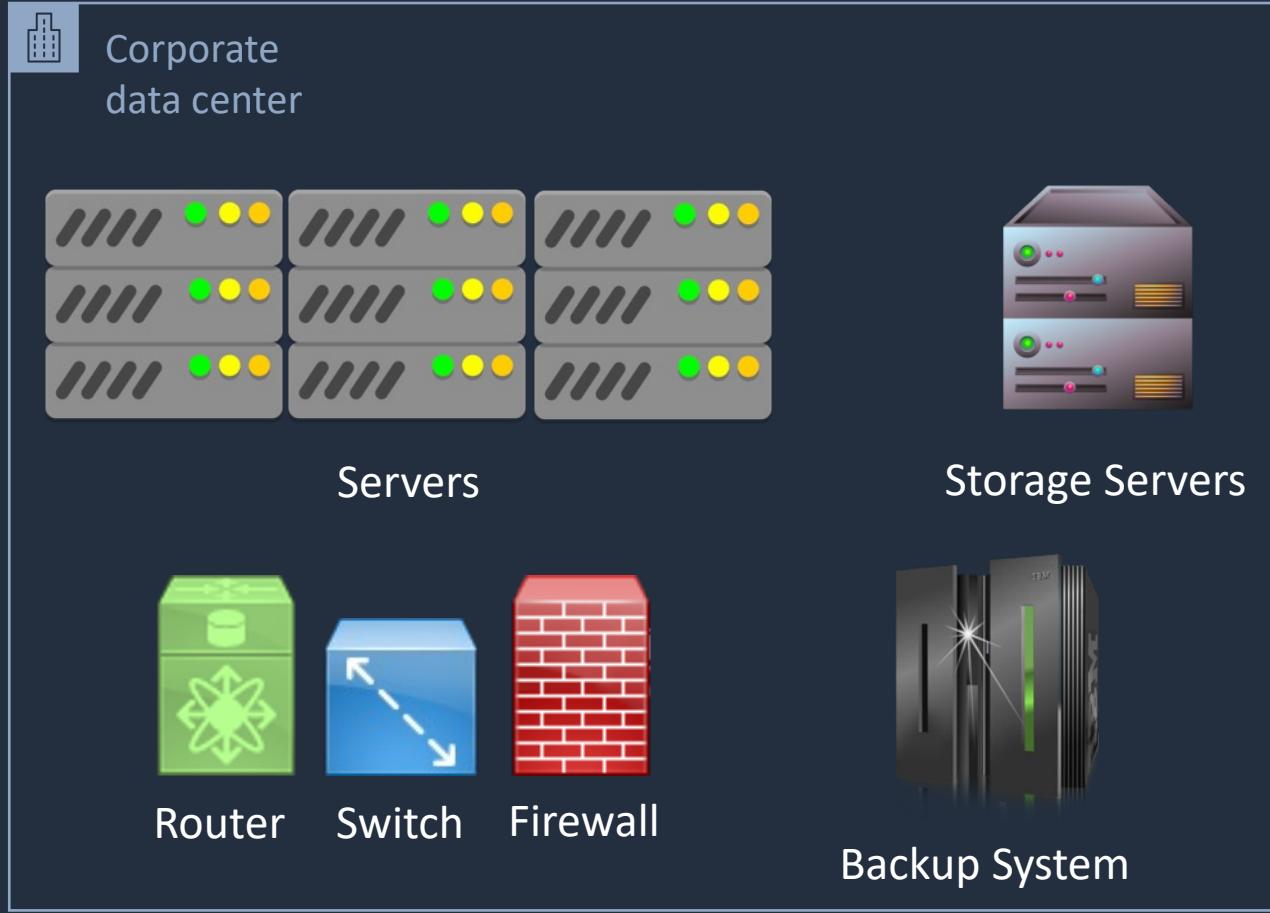


A company typically leases space in a data center, or may own the whole building

The IT equipment is **owned** by the company



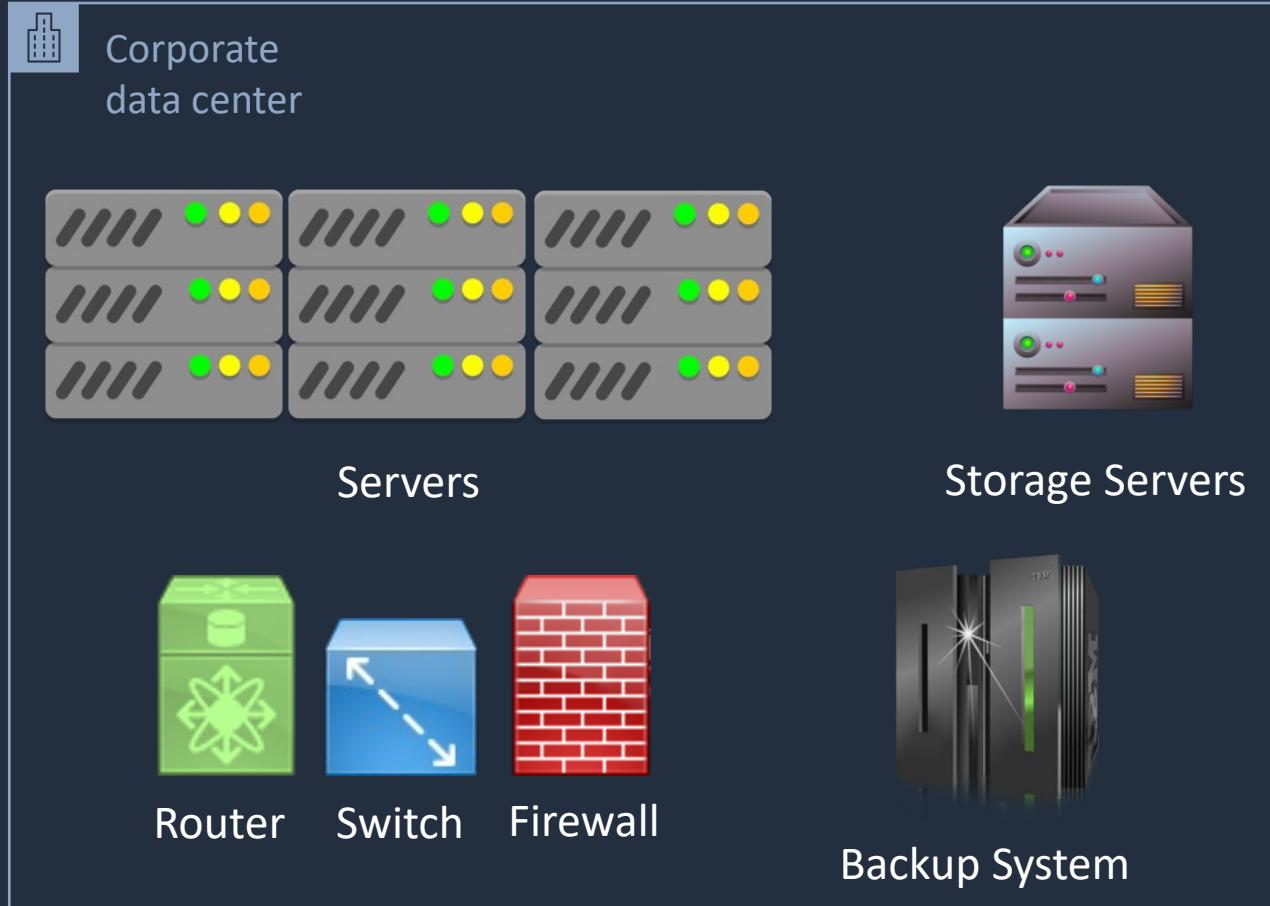
Traditional IT



IT staff must design,
build, operate, and
manage equipment



Traditional IT



Costs:

- Data center building
- Data center security
- Physical IT hardware
- Software licensing costs
- Maintenance contracts
- Power
- Internet connectivity
- Staff wages (design, build, operations, maintenance)

What is Cloud Computing?





What is Cloud Computing? Well-known examples

Non-Cloud Services:



Email Server



File Server



Customer Relationship
Management (CRM)

Cloud Services:



Gmail



Dropbox



Salesforce

You don't own or manage the infrastructure on which the service runs

Cloud services are offered on a subscription / consumption model

The service scales as demand changes

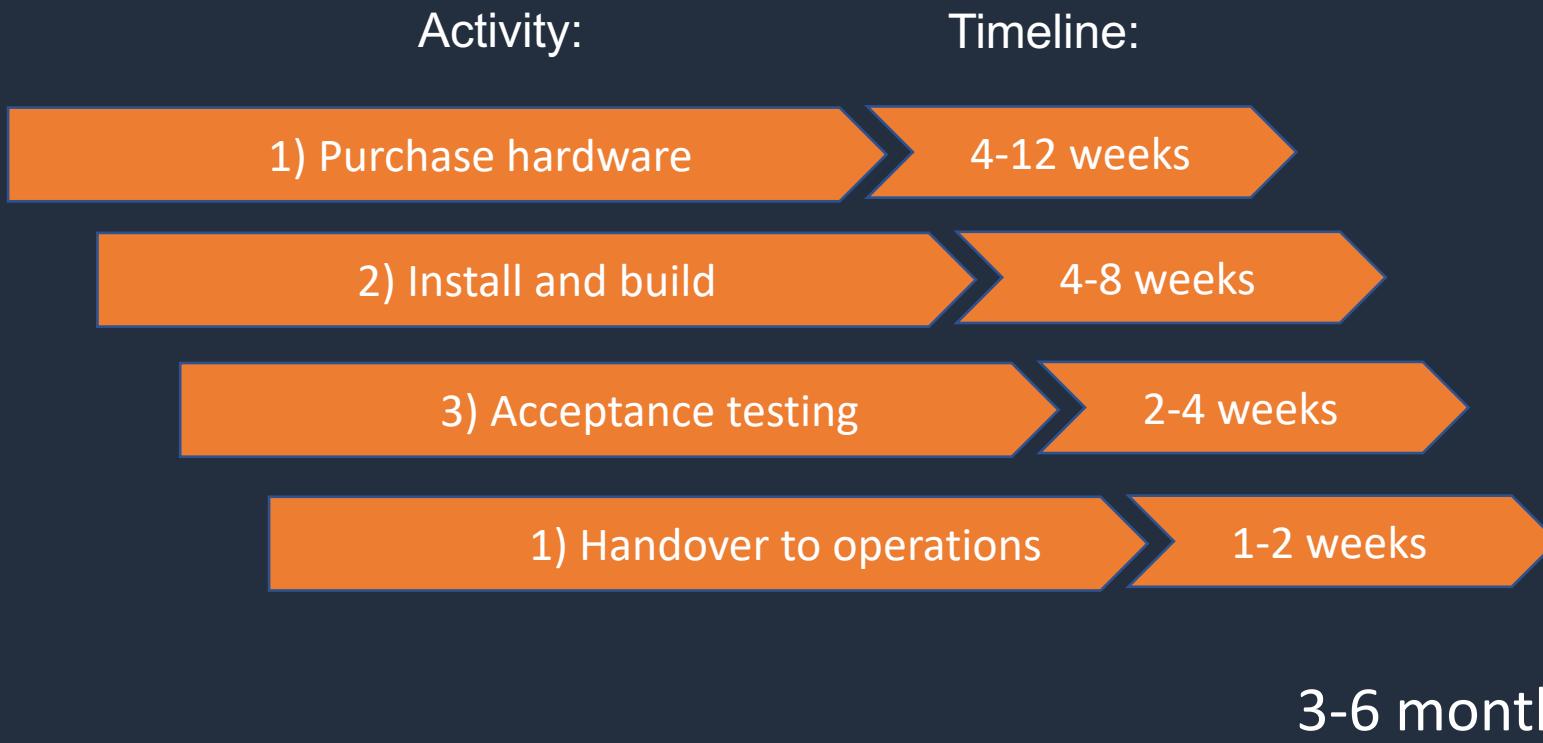


What is Cloud Computing? The Key Characteristics

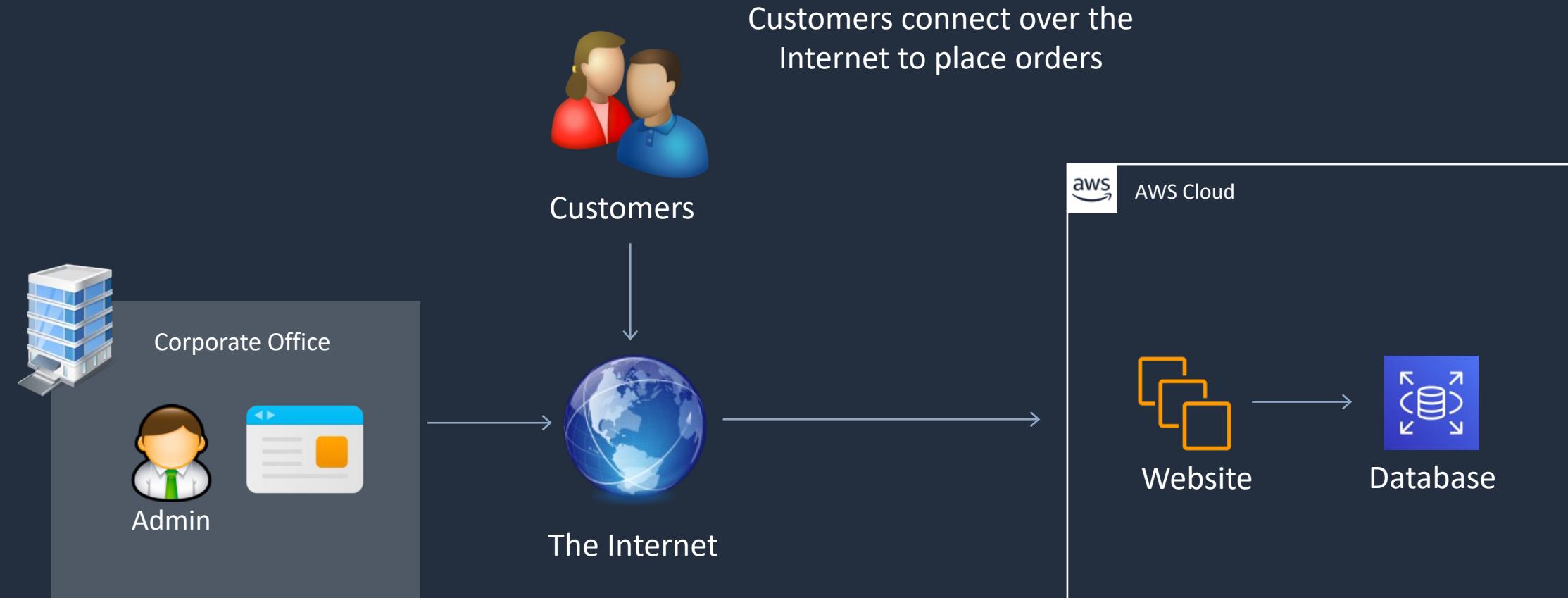
Name	Description
On-demand, self-service	A user can consume cloud resources, as needed, automatically, and without human interaction
Broad network access	Capabilities are available over the network using standard mechanisms. Can be the Internet or a Wide Area Network (WAN)
Resource pooling	The providers resources are pooled and serve multiple consumers using a multi-tenant model
Rapid elasticity	Capabilities can scale “elastically” based on demand
Measured service	Resource usage is monitored and metered



Deploying an eCommerce Website on-premises (aka the old way)



Deploying an eCommerce Website in the Cloud



The admin deploys an eCommerce website / database to AWS using the AWS Management Console



Launching Cloud Services: Management Console



Management Console

A web-based console accessed through a standard web browser

Compute EC2 Lightsail ↗ Lambda Batch Elastic Beanstalk Serverless Application Repository AWS Outposts EC2 Image Builder	Blockchain Amazon Managed Blockchain	Analytics Athena EMR CloudSearch Elasticsearch Service Kinesis QuickSight ↗ Data Pipeline AWS Data Exchange AWS Glue AWS Lake Formation MSK	Business Applications Alexa for Business Amazon Chime ↗ WorkMail Amazon Honeycode
Storage S3 EFS FSx S3 Glacier Storage Gateway AWS Backup	Satellite Ground Station	Quantum Technologies Amazon Braket ↗	End User Computing WorkSpaces AppStream 2.0 WorkDocs WorkLink
Database RDS DynamoDB ElastiCache Neptune Amazon Redshift Amazon QLDB	Management & Governance AWS Organizations CloudWatch AWS Auto Scaling CloudFormation CloudTrail Config OpsWorks Service Catalog Systems Manager AWS AppConfig Trusted Advisor Control Tower AWS License Manager AWS Well-Architected Tool	Security, Identity, & Compliance IAM Resource Access Manager Cognito Secrets Manager GuardDuty Inspector Amazon Macie AWS Single Sign-On Certificate Manager Key Management Service CloudHSM Directory Service	Internet Of Things IoT Core FreeRTOS IoT 1-Click IoT Analytics IoT Device Defender IoT Device Management IoT Events IoT Greengrass IoT SiteWise IoT Things Graph



Launching Cloud Services: Command Line



Command Line

This command launches a **virtual server** (EC2 instance) on Amazon Web Services



```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro
```



```
aws s3 ls s3://mys3databucket
```

This command lists the contents of a storage container (bucket) on **Amazon S3**



Launching Cloud Services: Software Development Kit



Code (SDK)

A developer writes code in an integrated development environment (IDE)

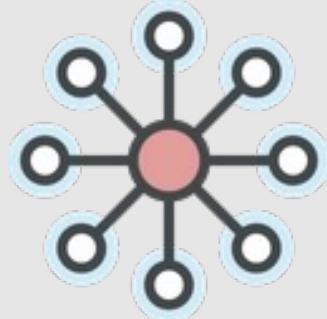
The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "MY-SAM-APP (WORKSPACE)".
- Editor:** The file "app.js" is open, containing the following JavaScript code for a Lambda function:

```
my-sam-app-nodejs > hello-world > JS app.js > lambdaHandler > exports.lambdaHandler
15  /*
16  */
Run Locally | Debug Locally | Configure
17 exports.lambdaHandler = async (event, context) => {
18   try {
19     // const ret = await axios(url);
20     response = {
21       'statusCode': 200,
22       'body': JSON.stringify({
23         message: 'hello world',
24         // location: ret.data.trim()
25       })
26     }
27   } catch (err) {
28     console.log(err);
29     return err;
30   }
31
32   return response
33 };
34
```
- Status Bar:** Shows "Ln 29, Col 1 (27 selected)" and other file metadata.

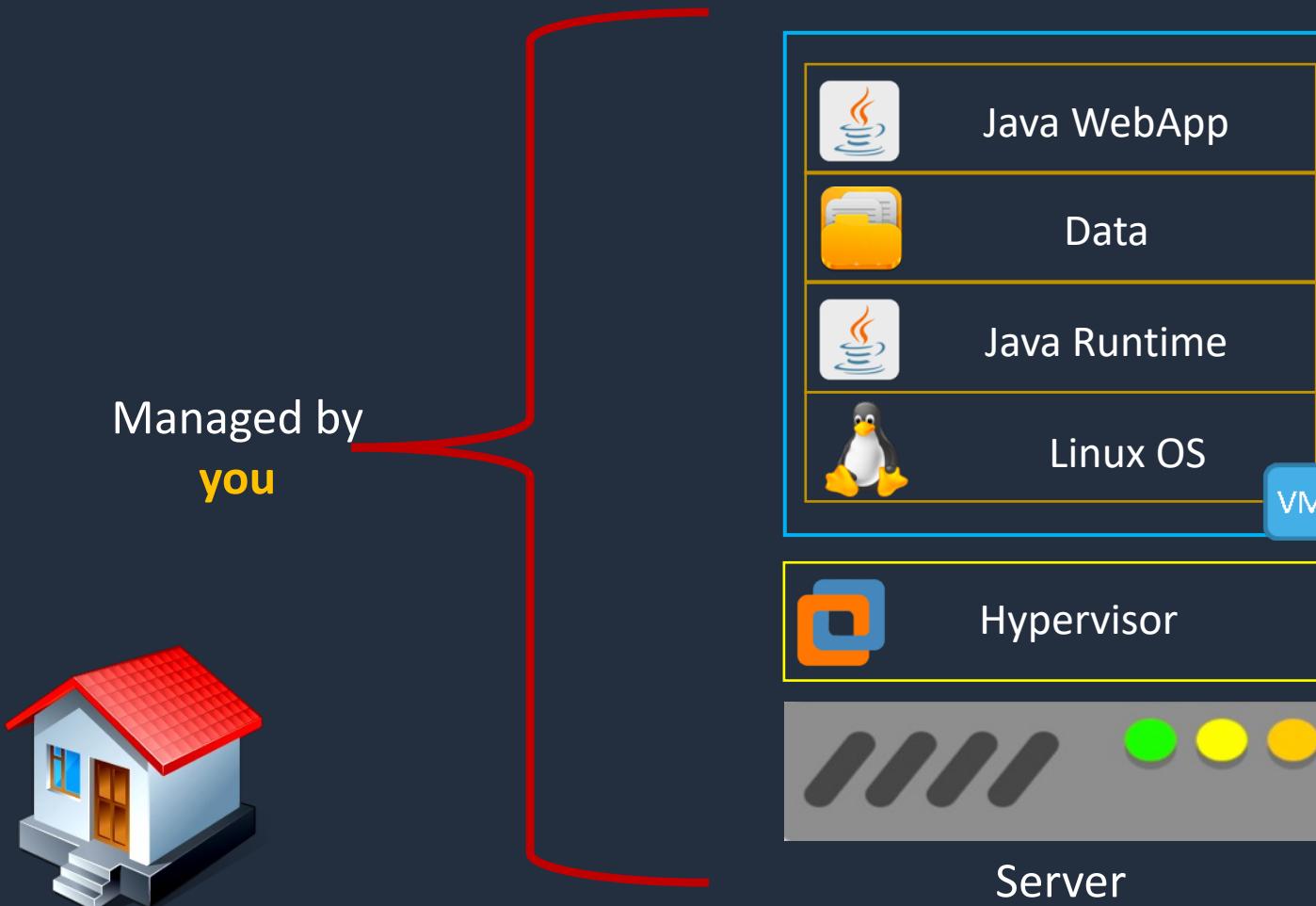
The code leverages the SDK to work with cloud services

Cloud Computing Service Models





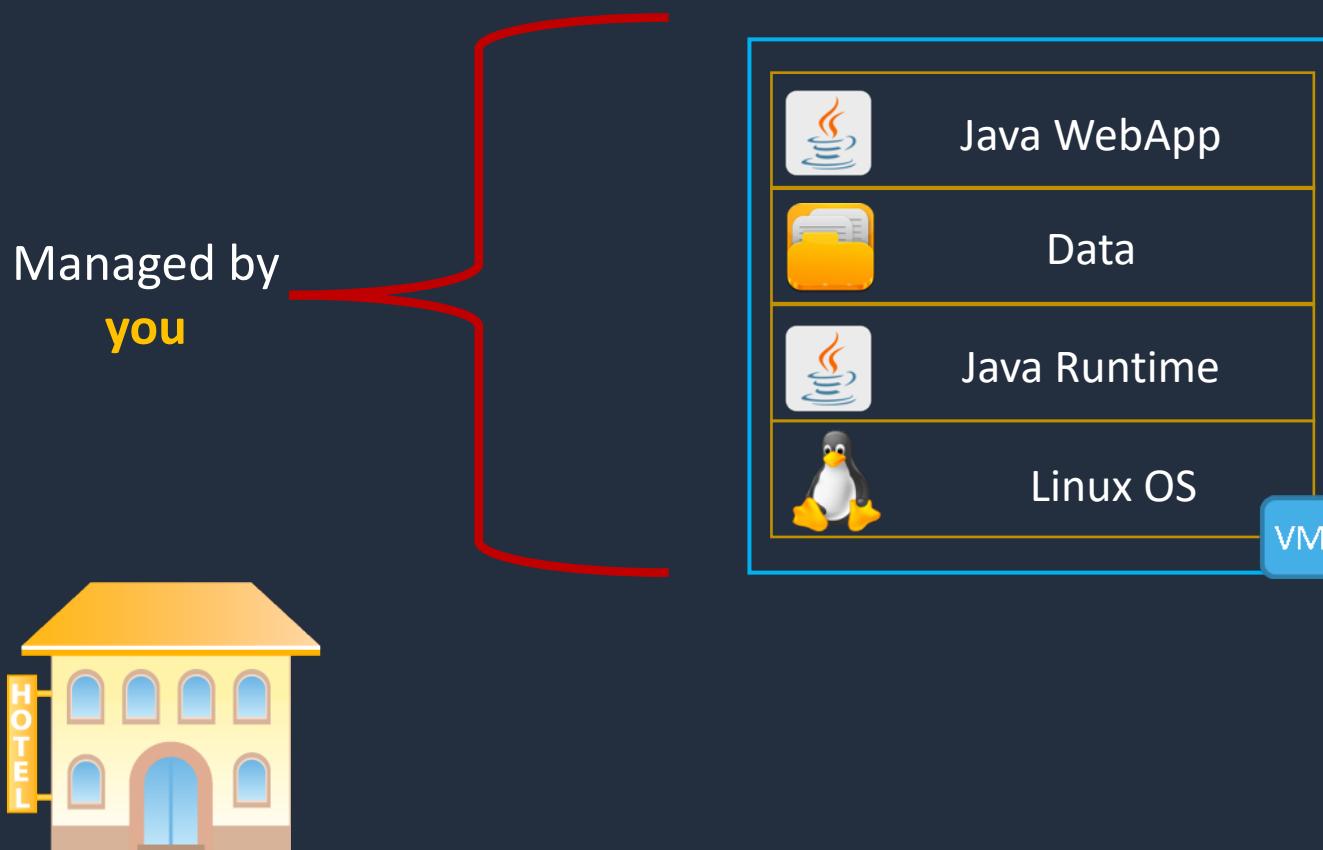
Cloud Service Models: Private Cloud



A **private cloud** must also include self-service, multi-tenancy, metering, and elasticity



Cloud Service Models: Infrastructure as a Service (IaaS)



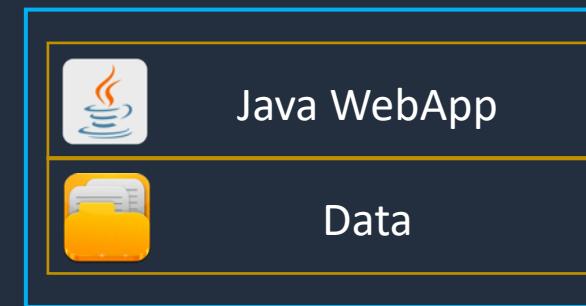
Examples:

- Amazon Elastic Compute Cloud (EC2)
- Azure Virtual Machines
- Google Compute Engine



Cloud Service Models: Platform as a Service (PaaS)

Managed by
you



Examples:

- AWS Elastic Beanstalk
- Azure WebApps
- Compute App Engine



Cloud Service Models: Software as a Service (SaaS)

Managed by
you



Java WebApp

Pure consumption
model

Examples:

- Google Apps
- Salesforce.com
- Zoom



Cloud Service Models: Comparison

Private Cloud



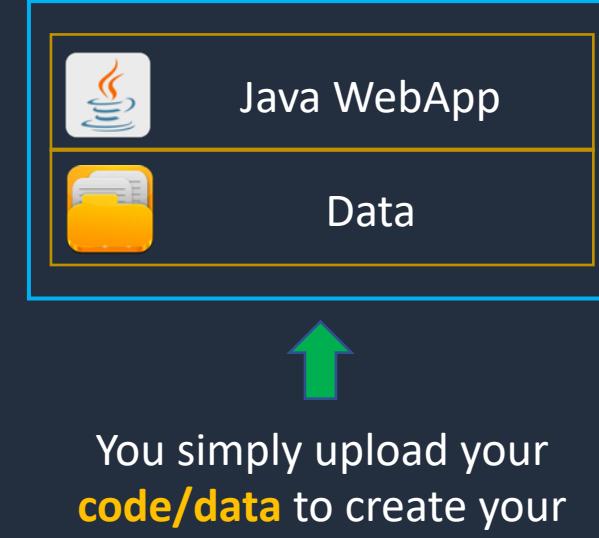
You manage **everything** -
greater responsibility +
greater control

IaaS



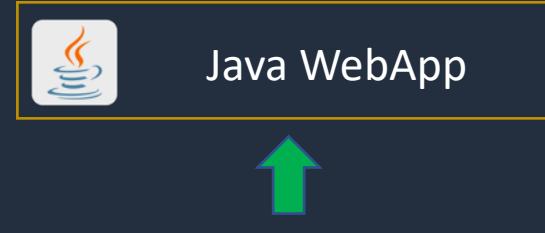
You manage from the
virtual server upwards

PaaS



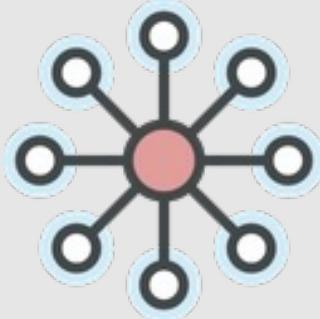
You simply upload your
code/data to create your
application

SaaS



You simply **consume** the
service - little responsibility
+ little control

Cloud Computing Deployment Models





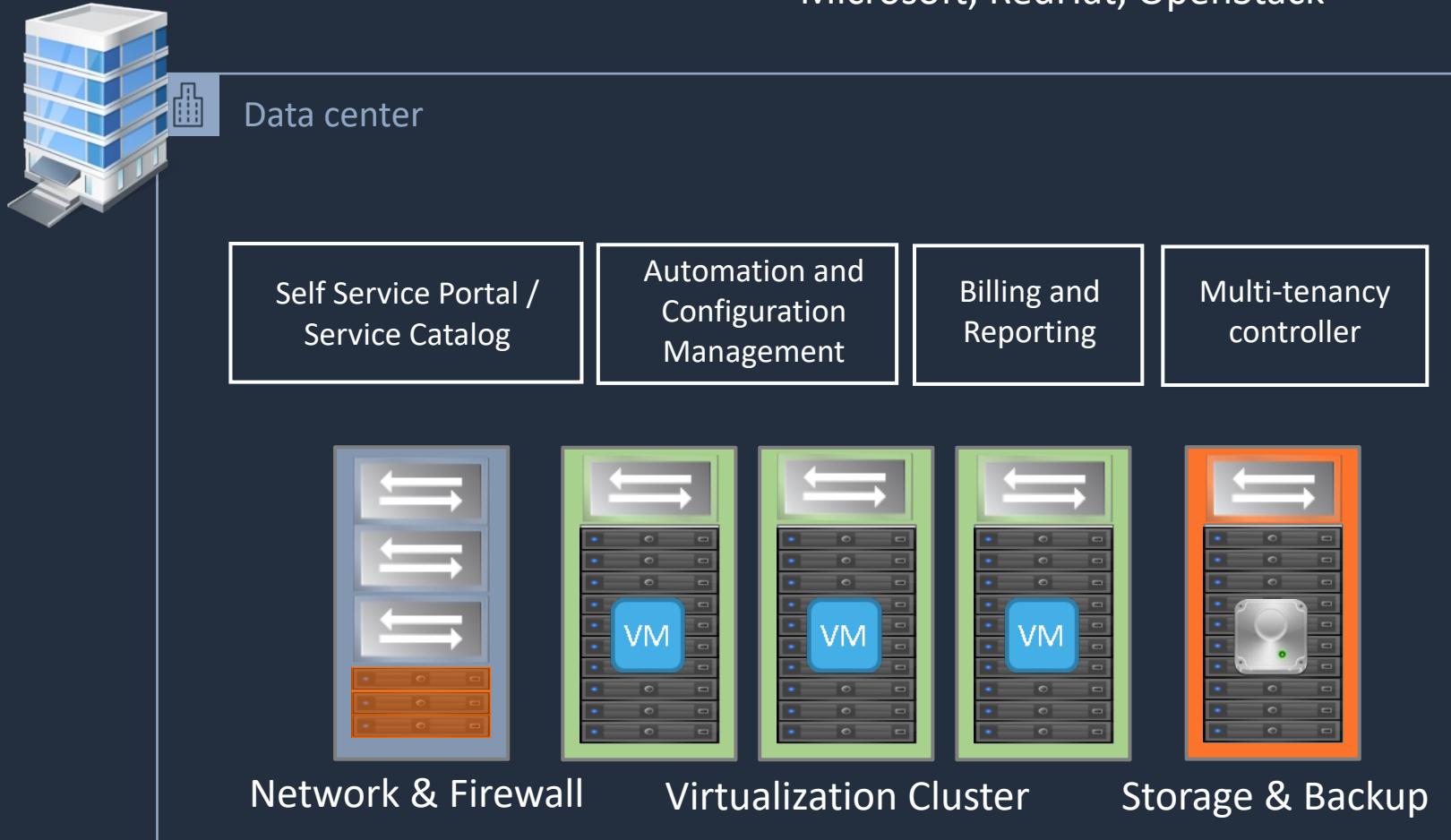
Cloud Computing Deployment Models

Name	Description	Examples
Private Cloud	An enterprise deploys their own infrastructure and applications into their own data center	VMware, Microsoft, RedHat, OpenStack
Public Cloud	The IT services that you consume are hosted and delivered from a third-party and accessed over the Internet	AWS, Microsoft Azure, Google Cloud Platform
Hybrid Cloud	A combination of on-premises, private cloud, and public cloud services are consumed	
Multicloud	Usage of two or more public clouds at a time, and possibly multiple private clouds	



Private Cloud

Examples are VMware,
Microsoft, RedHat, OpenStack



Cloud management
software layer

Benefits:

- Complete control of the entire stack
- Security – in a few cases, organizations may need to keep all or some of their applications and data in house

You **build** and **manage** the
cloud deployment

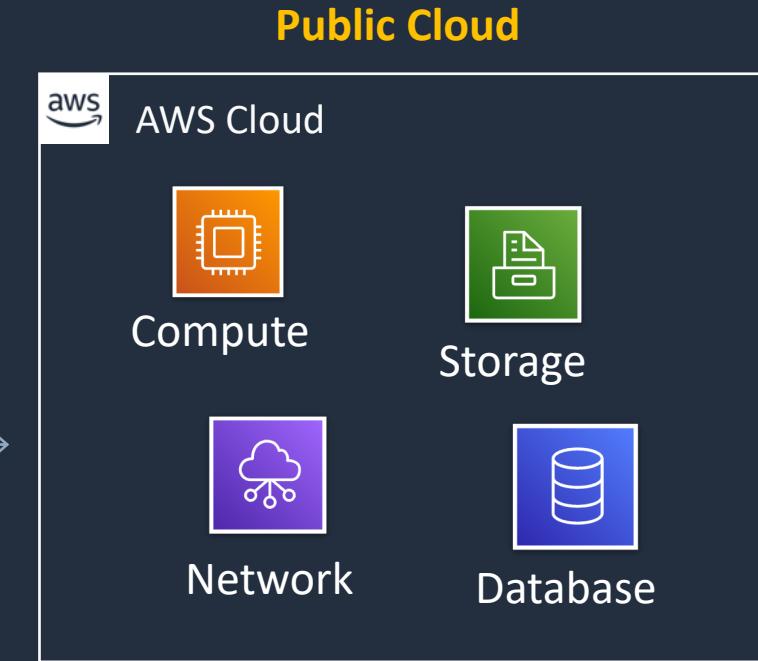


Cloud Deployment Models - Public Cloud

Examples are AWS, Microsoft Azure, Google Cloud Platform

Benefits:

- Variable expense, instead of capital expense
- Economies of scale
- Massive elasticity



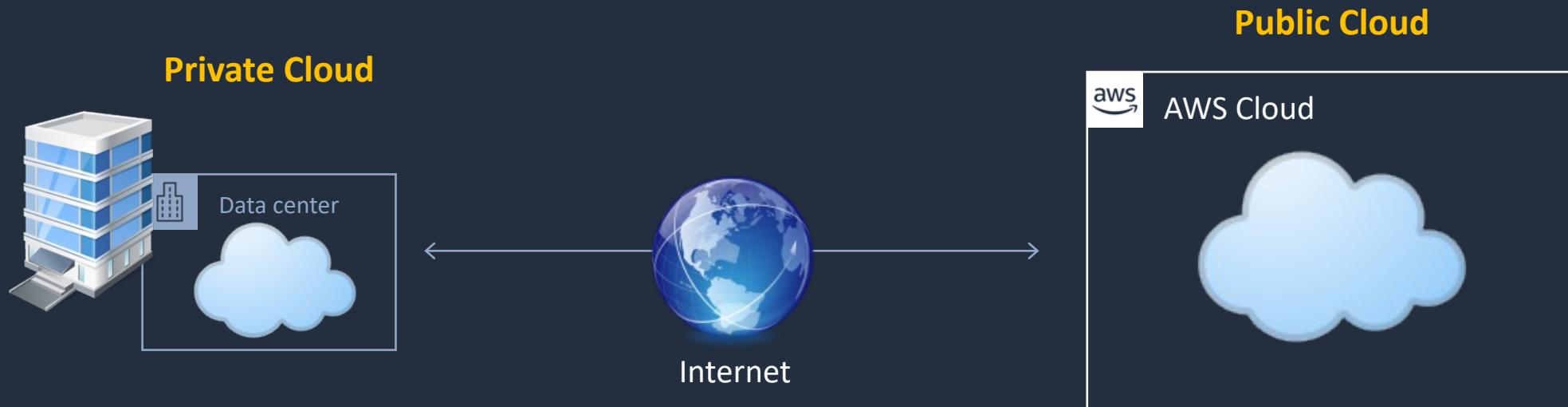
Connected using either the
Internet or a **private link**



Cloud Deployment Models - Hybrid Cloud

Benefits:

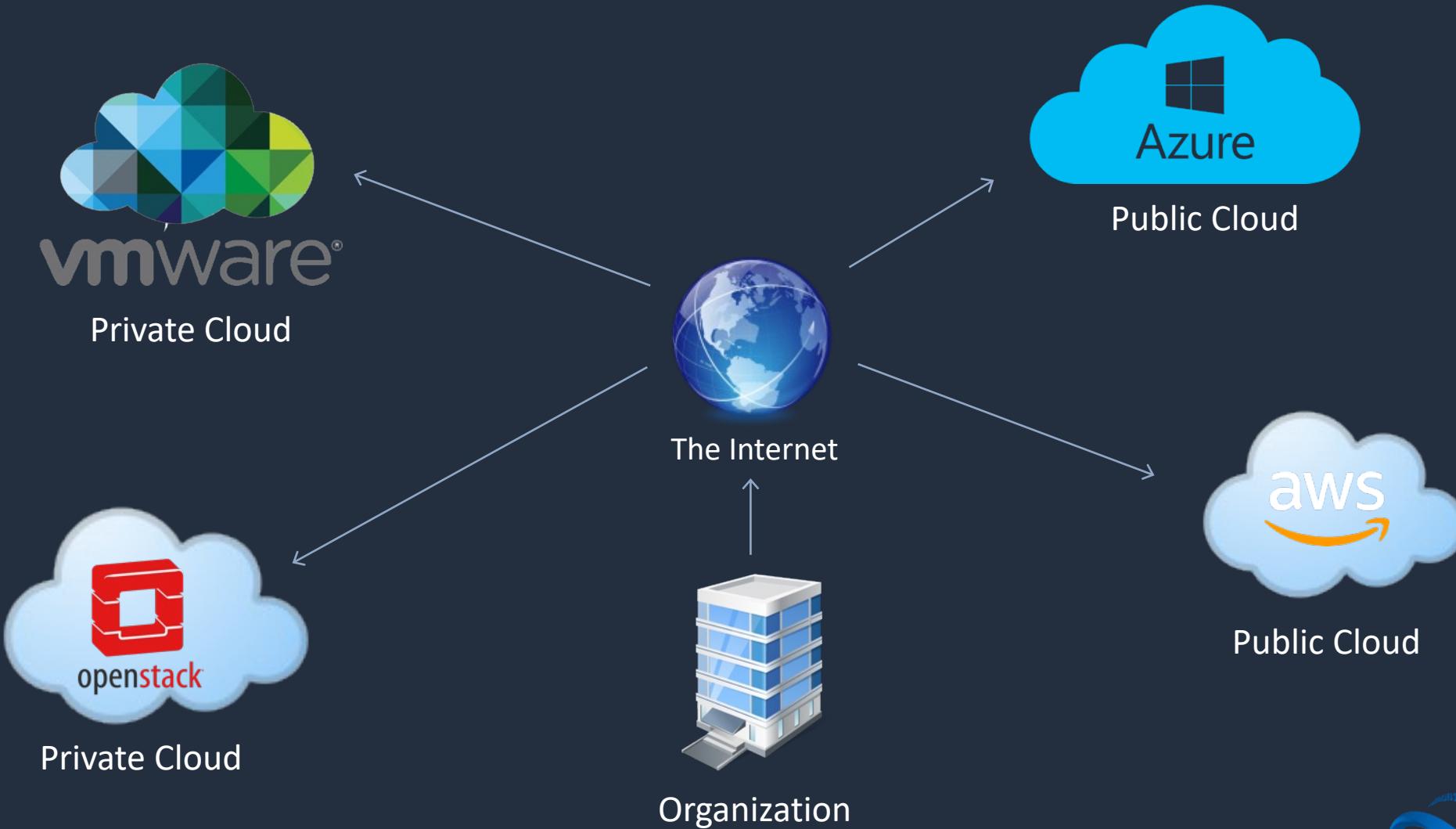
- Allows companies to keep the critical applications and sensitive data in a traditional data center environment or private cloud
- Take advantage of public cloud resources like SaaS, for the latest applications, and IaaS, for elastic virtual resources
- Facilitates portability of data, apps and services and more choices for deployment models



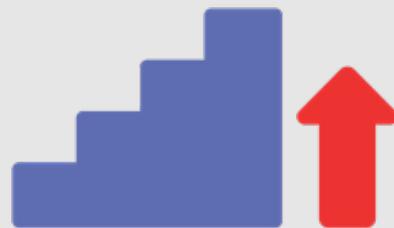
Connected using either the
Internet or a **private** link



Cloud Deployment Models - Multicloud



Scaling Up vs Scaling Out





Stateful vs Stateless Applications

Stateless:

No “state” is recorded about the user's session



Person check a news website

Netflix records what has been watched

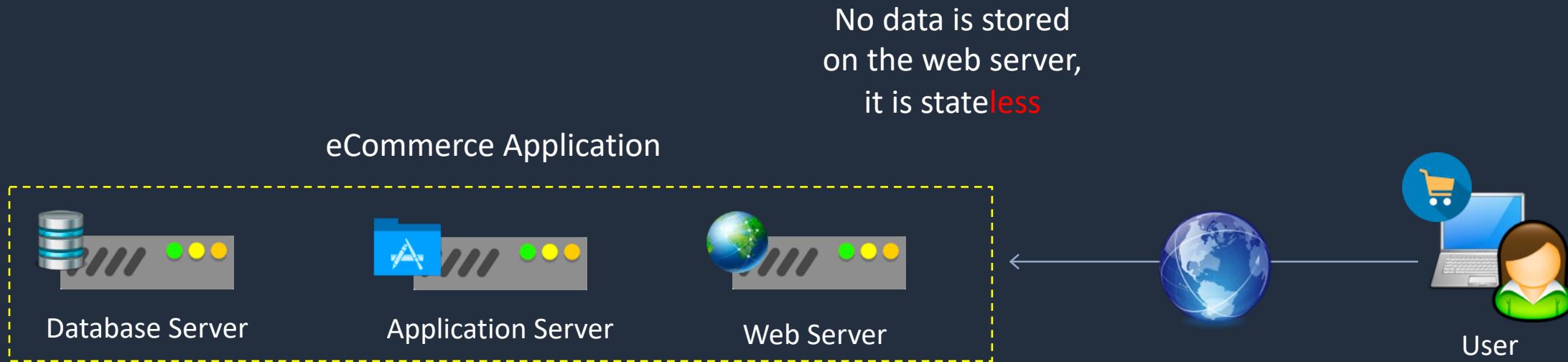
Stateful:



Person logs into Netflix



Stateful vs Stateless Applications

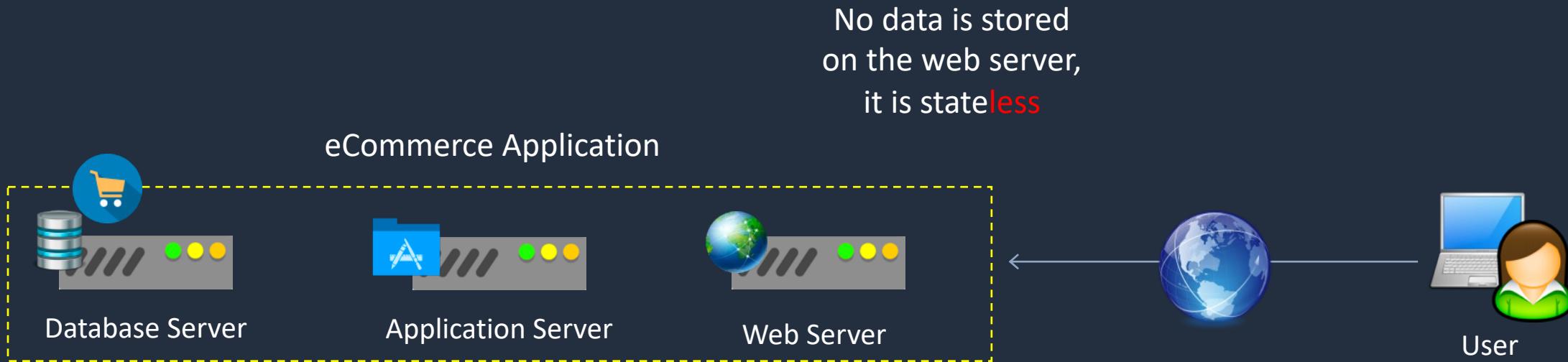


When the user purchases, the application layer processes the order and records the data in the database. This is **stateful**

The cart items are stored in cookies on the computer

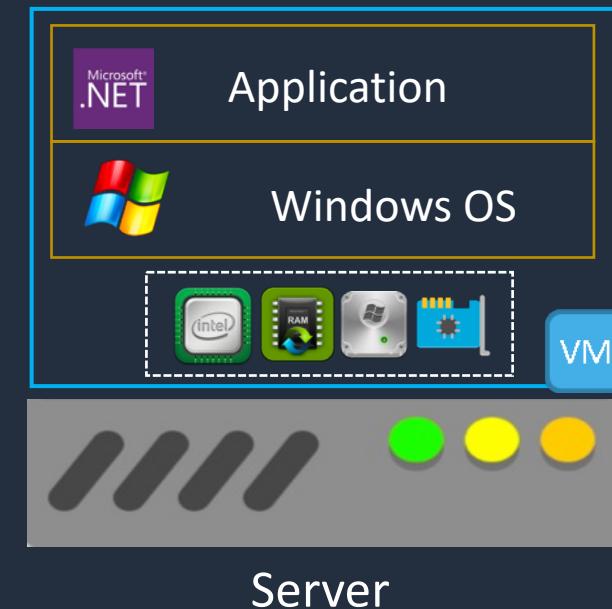


Stateful vs Stateless Applications





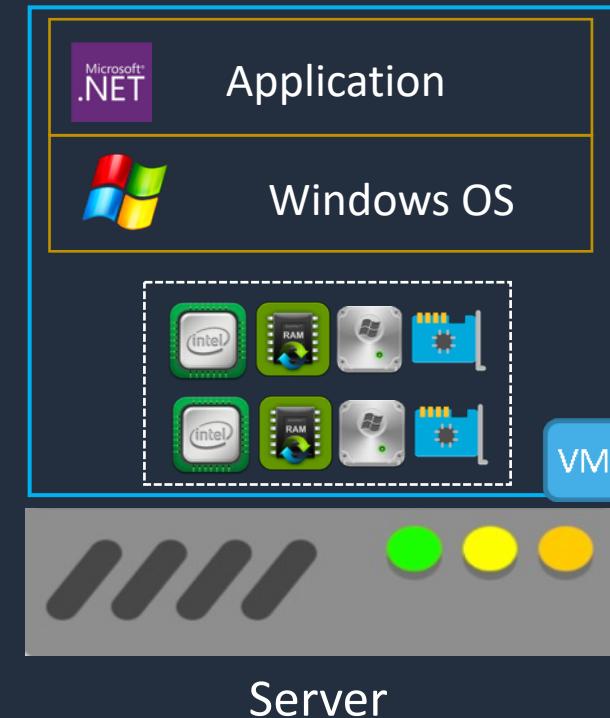
Scalability and Elasticity: Scaling Up





Scalability and Elasticity: Scaling Up

Scaling up means adding resources to the server

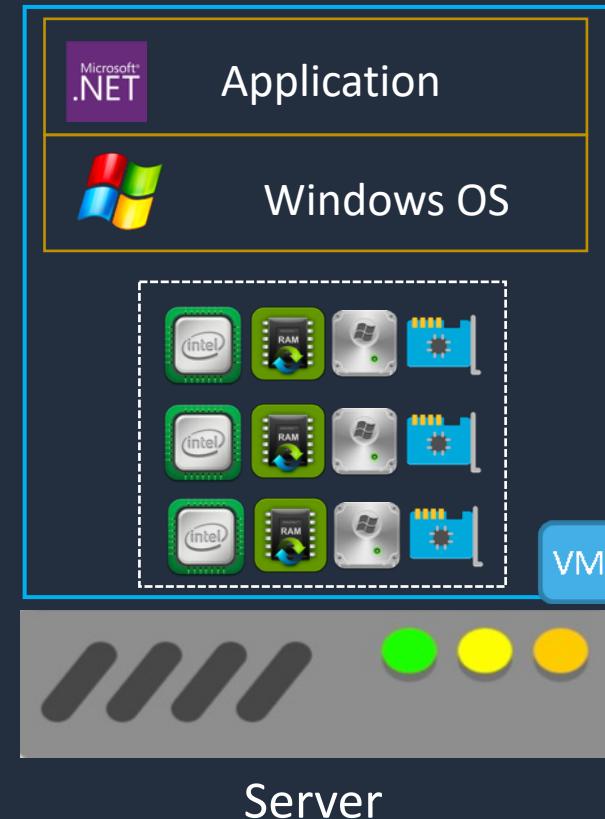


Server



Scalability and Elasticity: Scaling Up

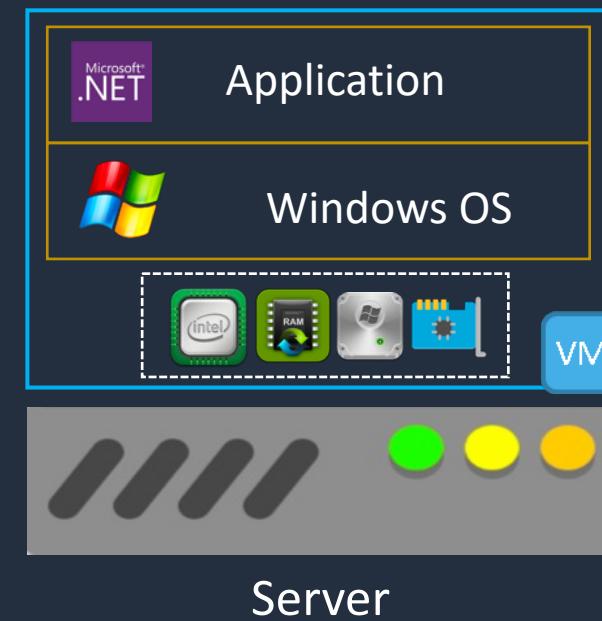
Scaling up means adding resources to the server



Server

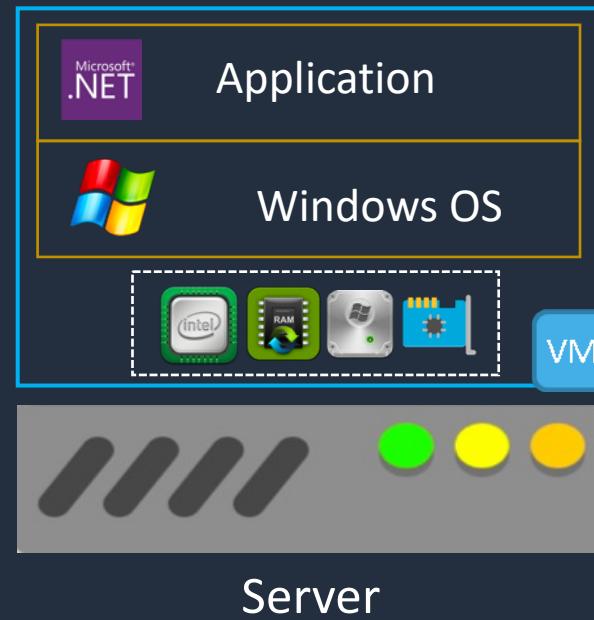


Scalability and Elasticity: Scaling Out





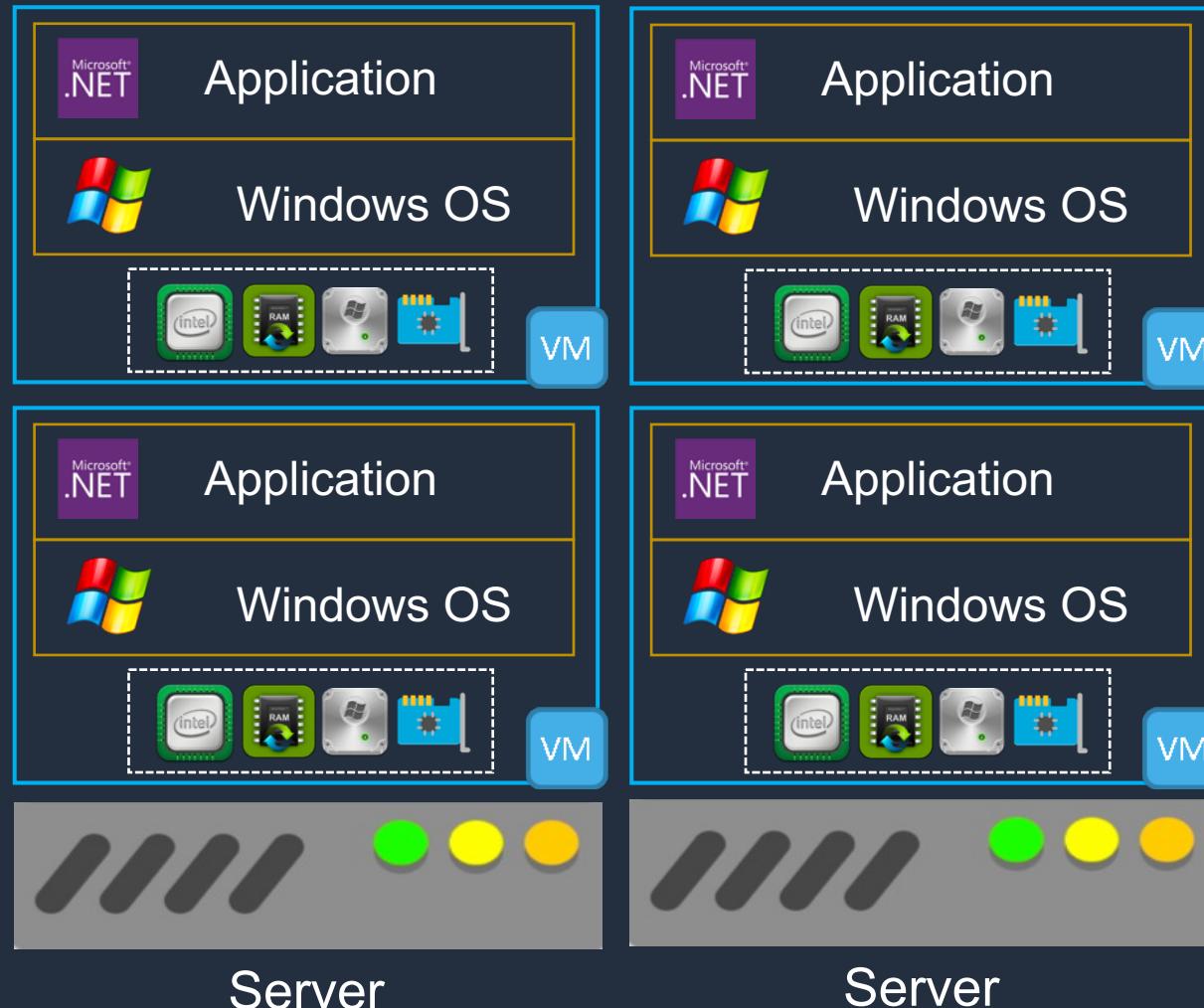
Scalability and Elasticity: Scaling Out



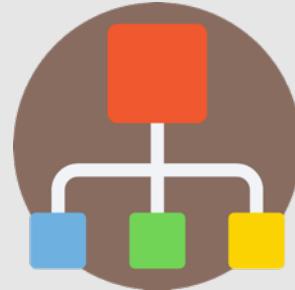


Scalability and Elasticity: Scaling Out

Scaling out means adding additional servers

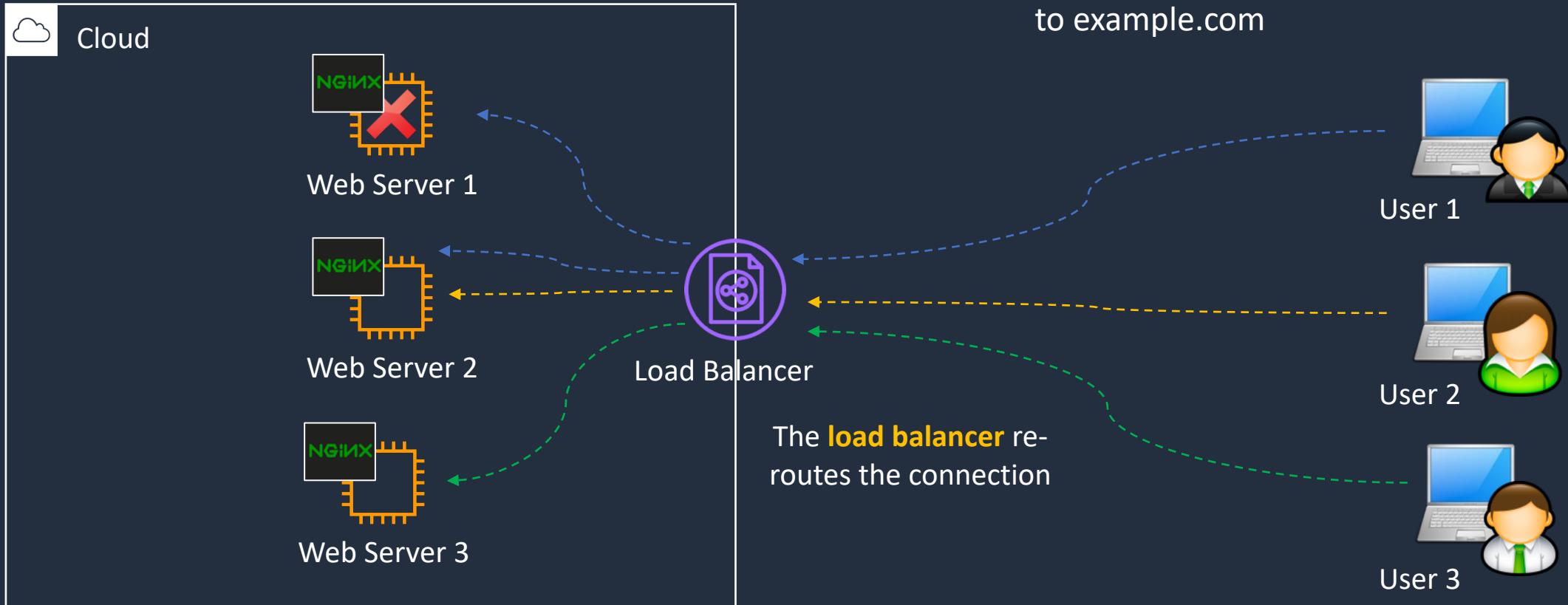


High Availability and Fault Tolerance





Load Balancing





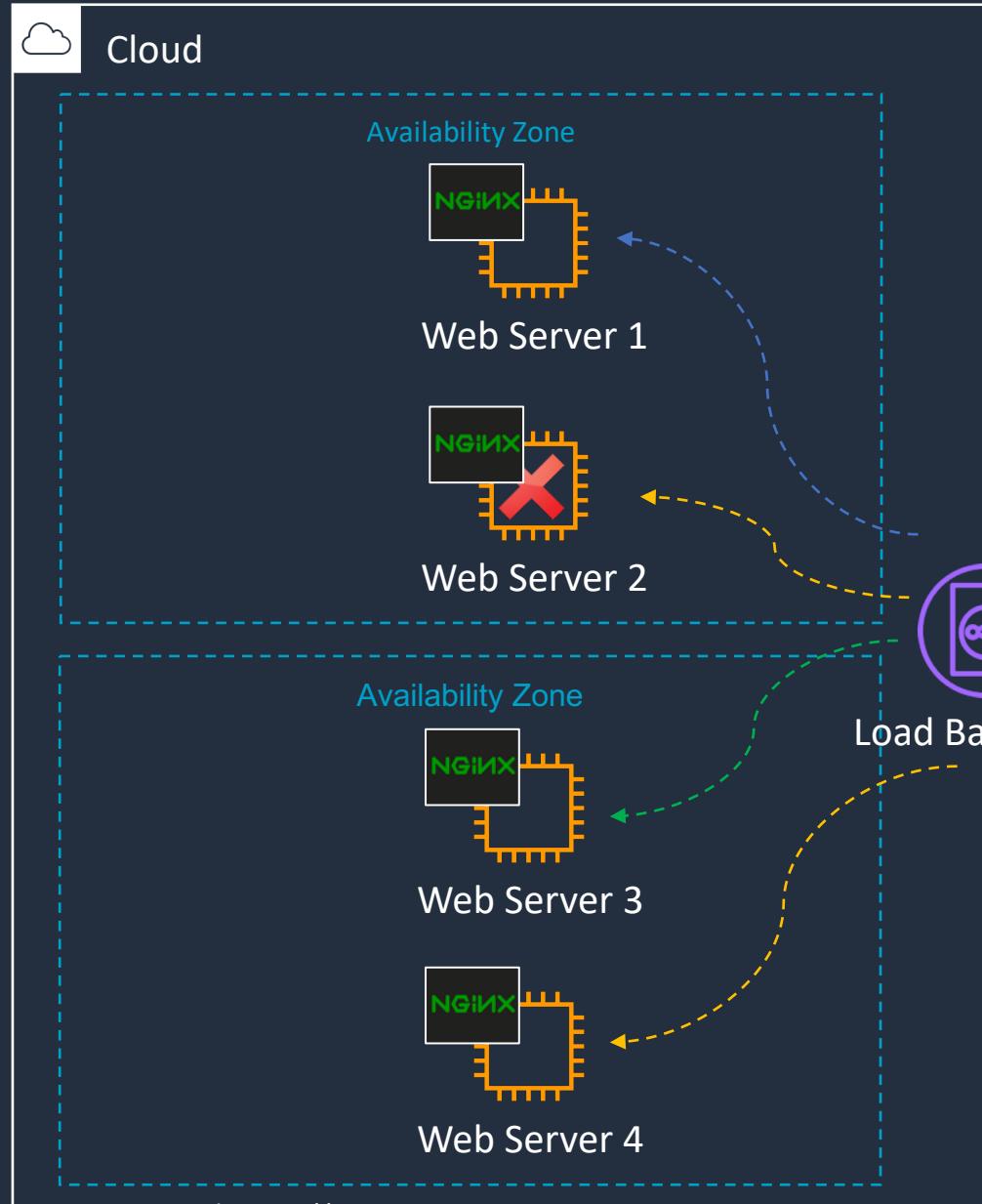
Fault Tolerance

Redundant components
allow the system to
continue to operate



The system may fail if there
is no built-in redundancy

High Availability and Fault Tolerance

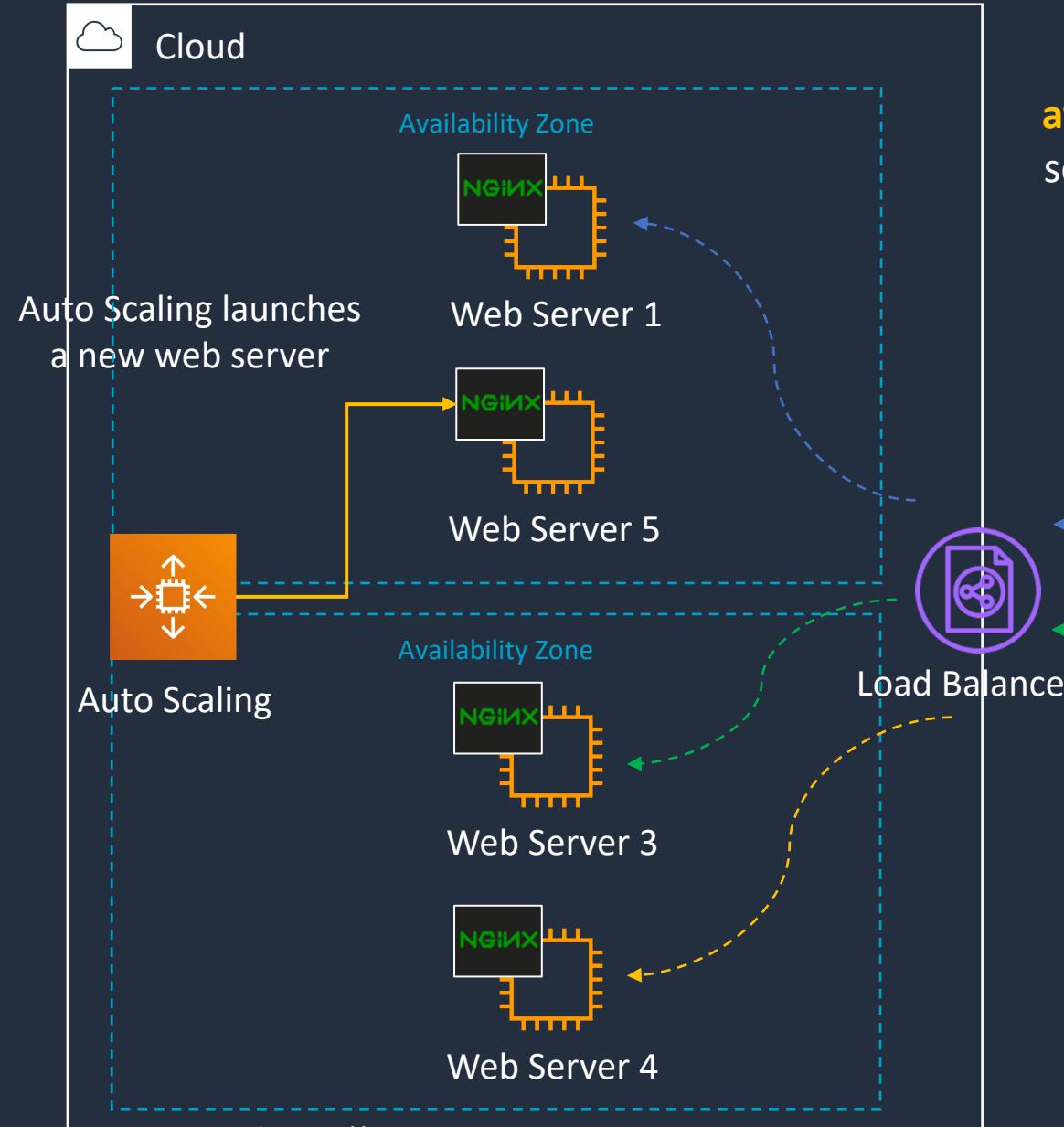


Think of an
availability zone as a
separate data center





High Availability and Fault Tolerance



Think of an **availability zone** as a separate data center



User 1



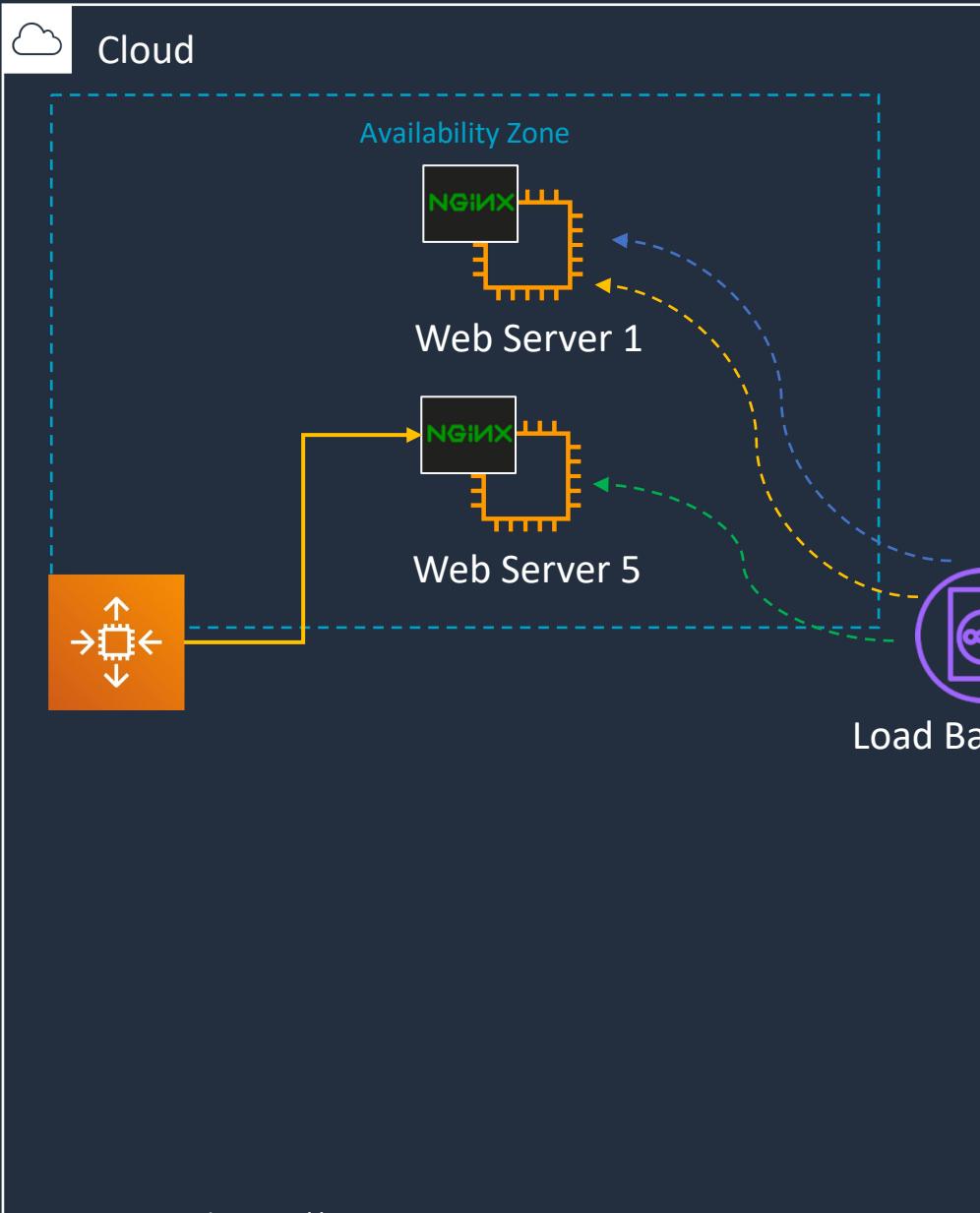
User 2



User 3



High Availability and Fault Tolerance



Think of an
availability zone as a
separate data center



User 1



User 2



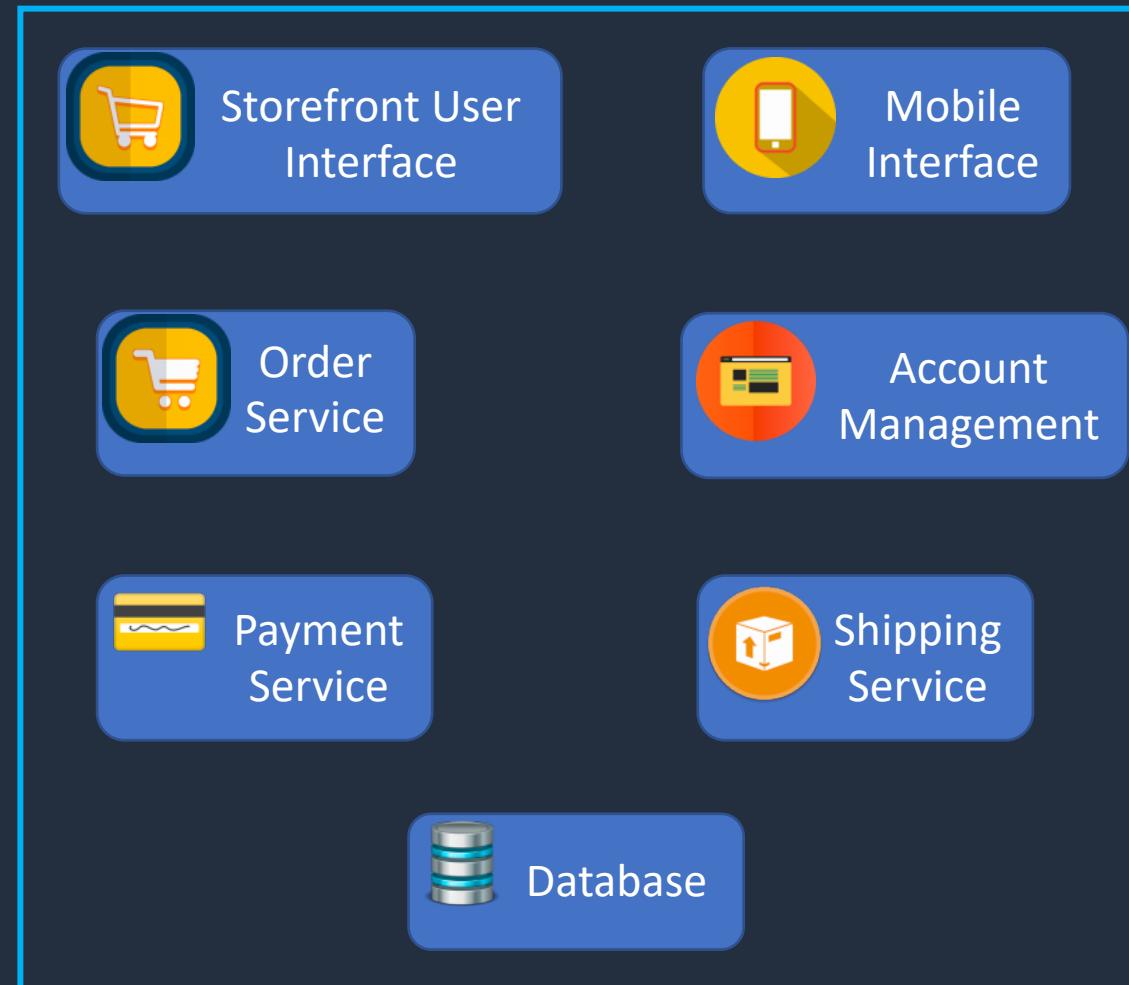
User 3

Monolithic and Microservices Architectures





Monolithic Application





Monolithic Application

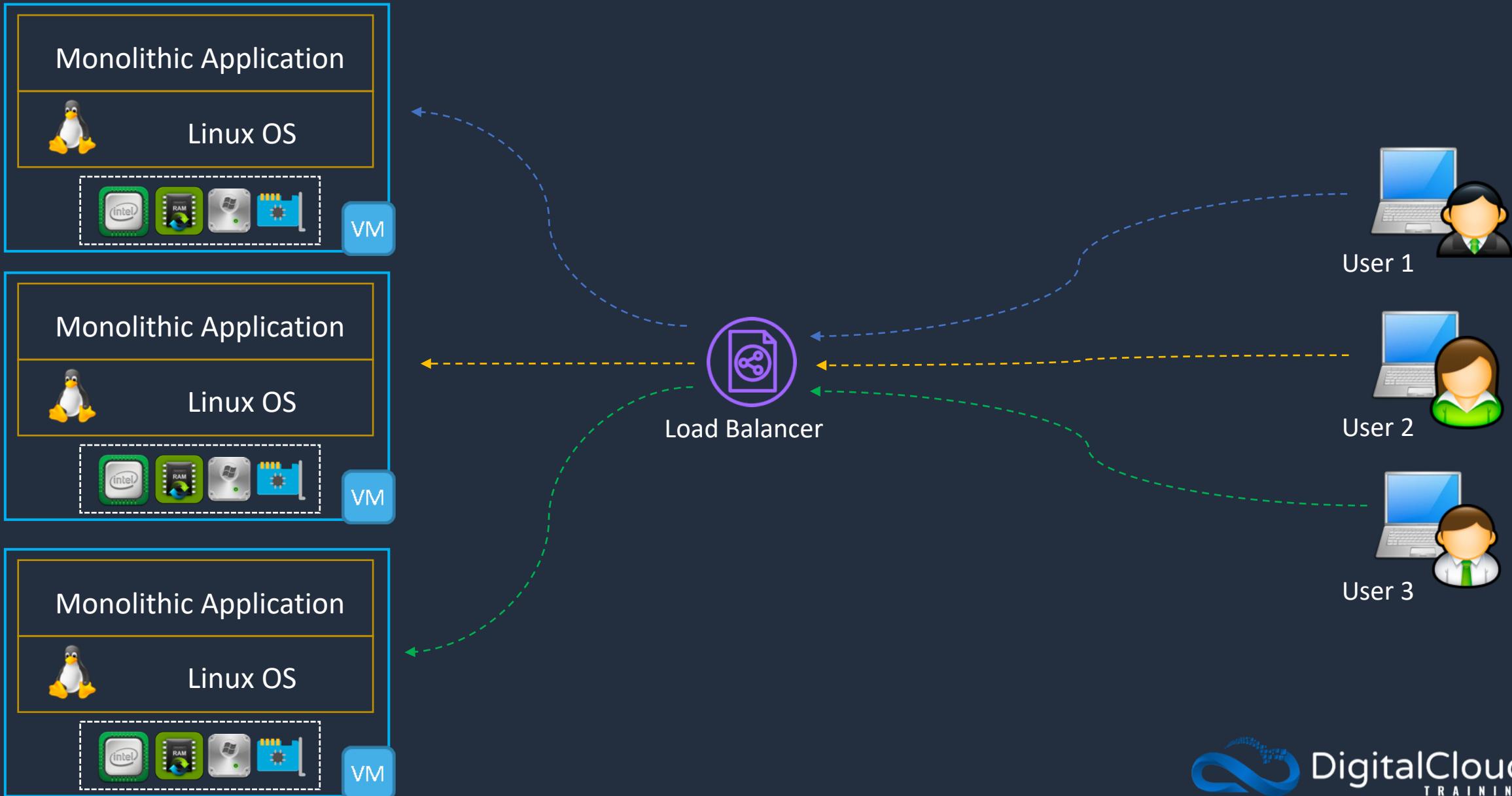
Updates to, or failures of, any single component can take down the whole application



The user interface, business logic, and data access layer are combined on a single platform



Monolithic Application





Monolithic Application

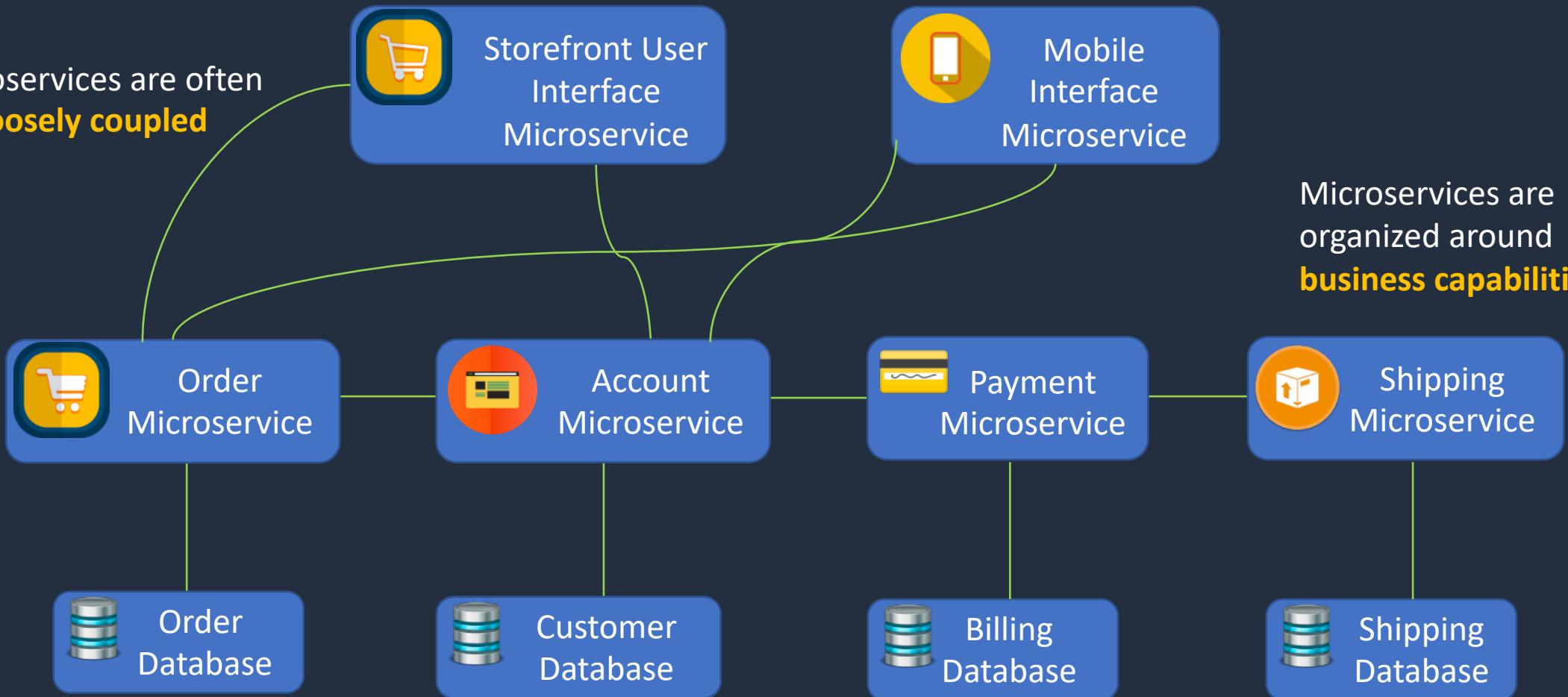




Microservices Architecture

A **microservice** is an independently deployable unit of code

Microservices are often
loosely coupled





Microservices using Docker Containers

Order
Microservice

Storefront User
Interface
Microservice

Shipping
Microservice



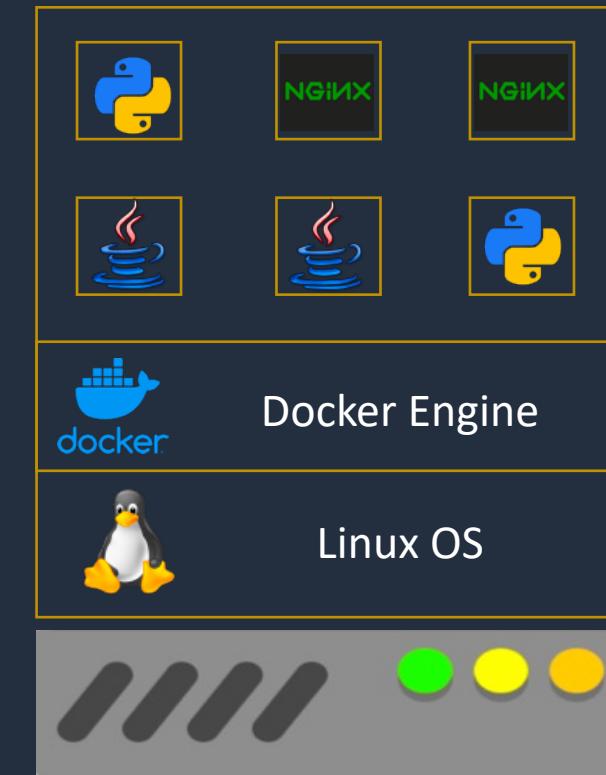
Server



Microservices using Docker Containers

Microservices can also be spread across hosts

Many instances of each microservice can run on each host





Microservices: Attributes and Benefits

Microservices Attribute	Microservices Benefit
Use of Application Programming Interfaces (APIs)	Easier integrations between application components; assists with loose coupling
Independently deployable blocks of code	Can be scaled and maintained independently
Business-oriented architecture	Development organized around business capabilities; teams may be cross-functional and services may be reused
Flexible use of technologies	Each microservice can be written using different technologies (e.g. programming languages)
Speed and agility	Fast to deploy and update. Easy to include high availability and fault tolerance for each microservice

SECTION 5

AWS Access Control and Networking

Amazon Web Services Overview





Amazon Web Services (AWS)

Charge for
services based
on usage

27 Regions
around the
world

Subsidiary
of Amazon

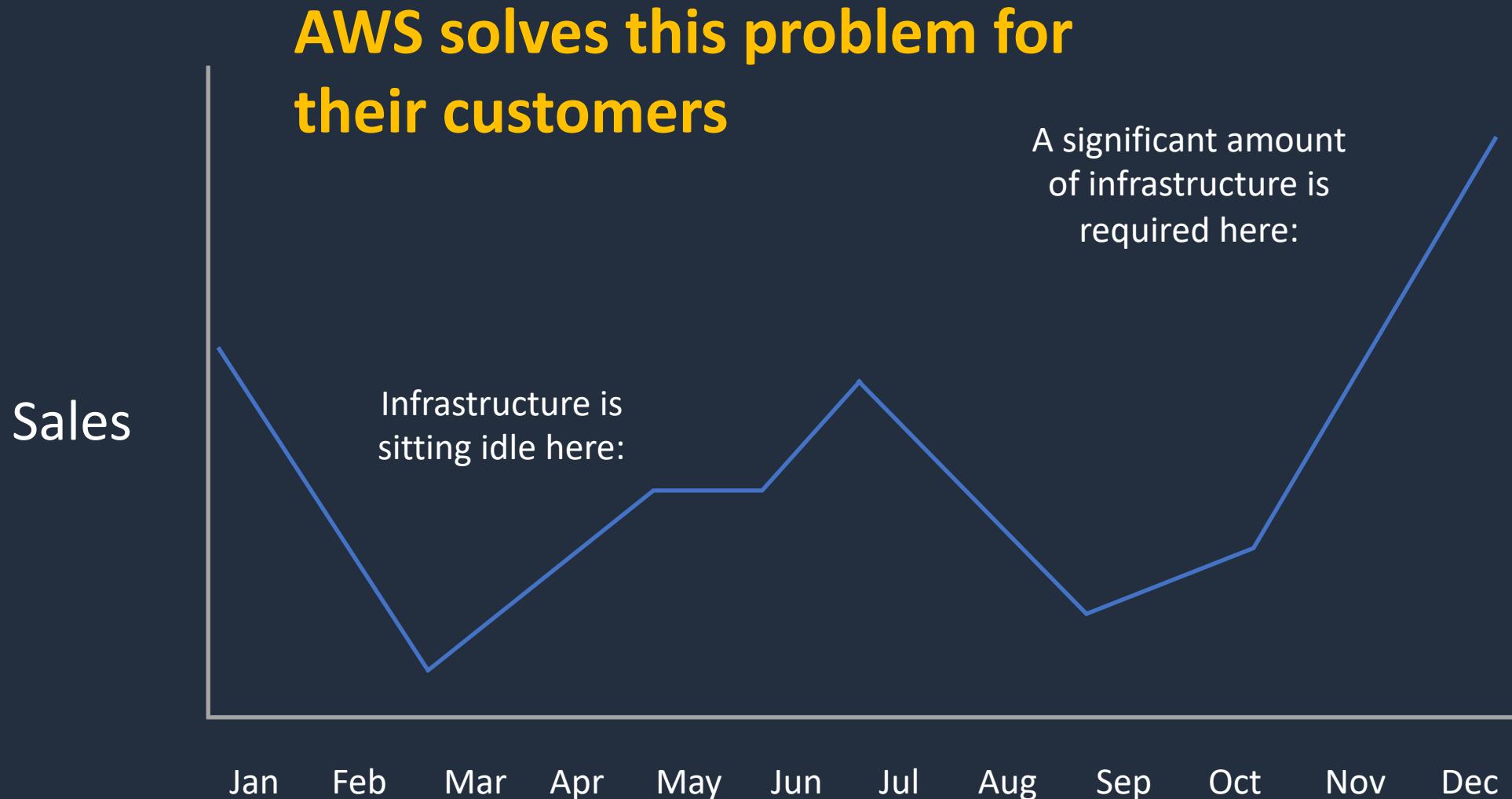


Hyperscale
Public Cloud
Provider

Services are
offered on-
demand



Overview of AWS





AWS Service Categories (a few examples)



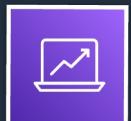
Compute



Storage



Database



Analytics



Machine Learning



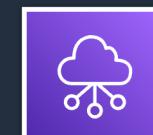
Many more categories
and over **200** services!



Internet of Things



Media Services



Networking



End User Computing



Fundamentals of AWS Pricing

Compute



Amount of resources such as CPU and RAM and duration

Storage



Quantity of data stored

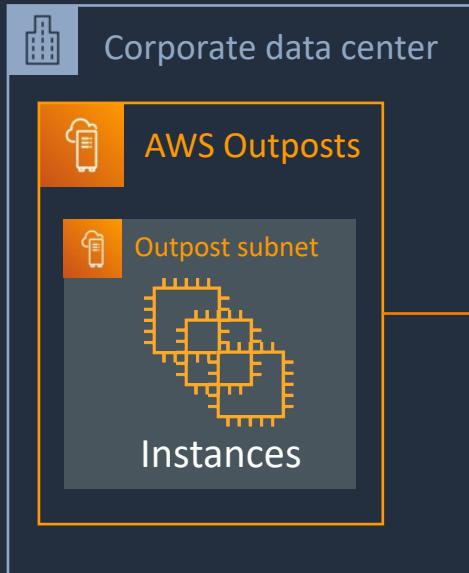
Outbound Data Transfer



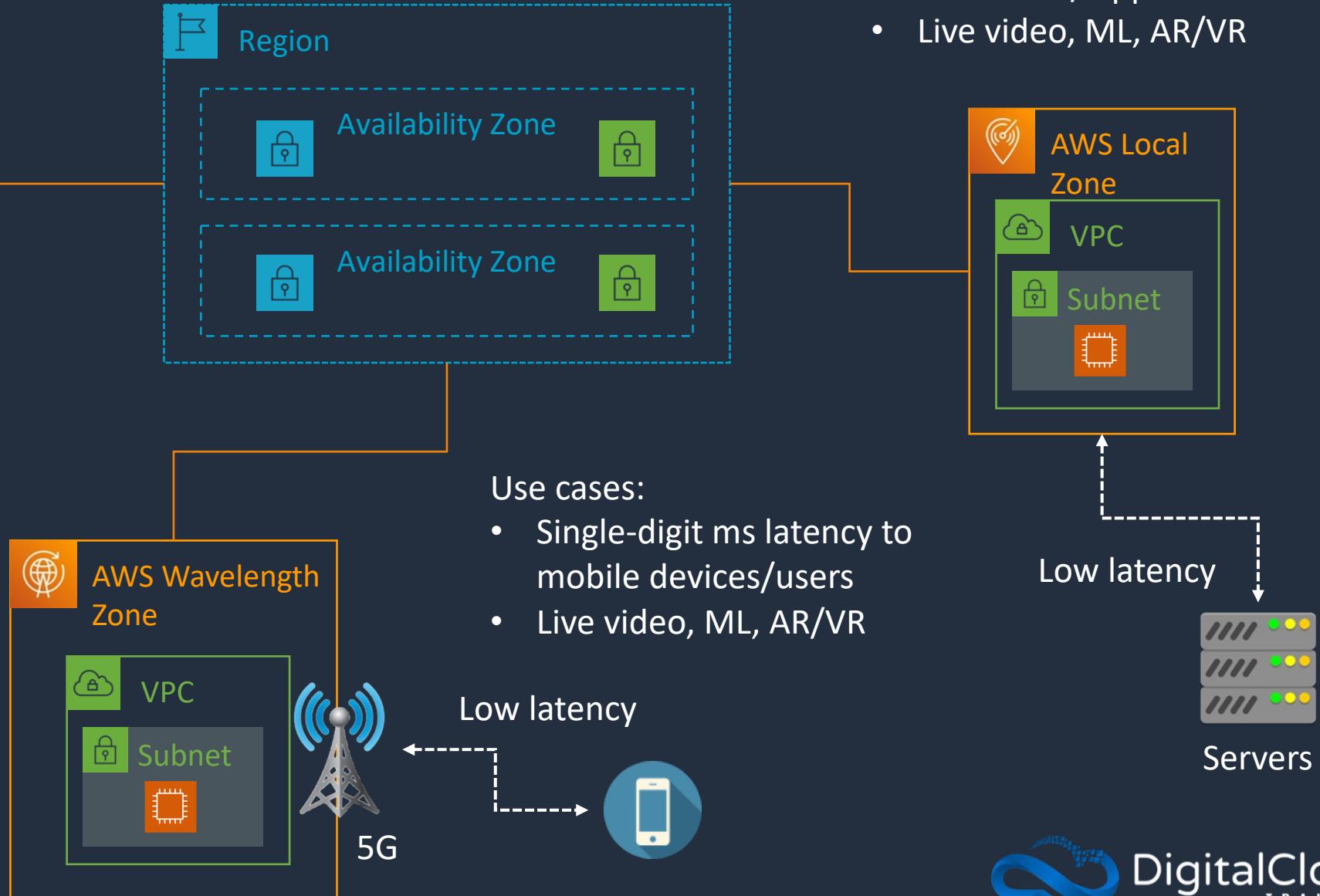
Quantity of data that is transferred out from all services



AWS Global Infrastructure



- Use cases:
- Use AWS services on-premises
 - Use AWS APIs on-premises



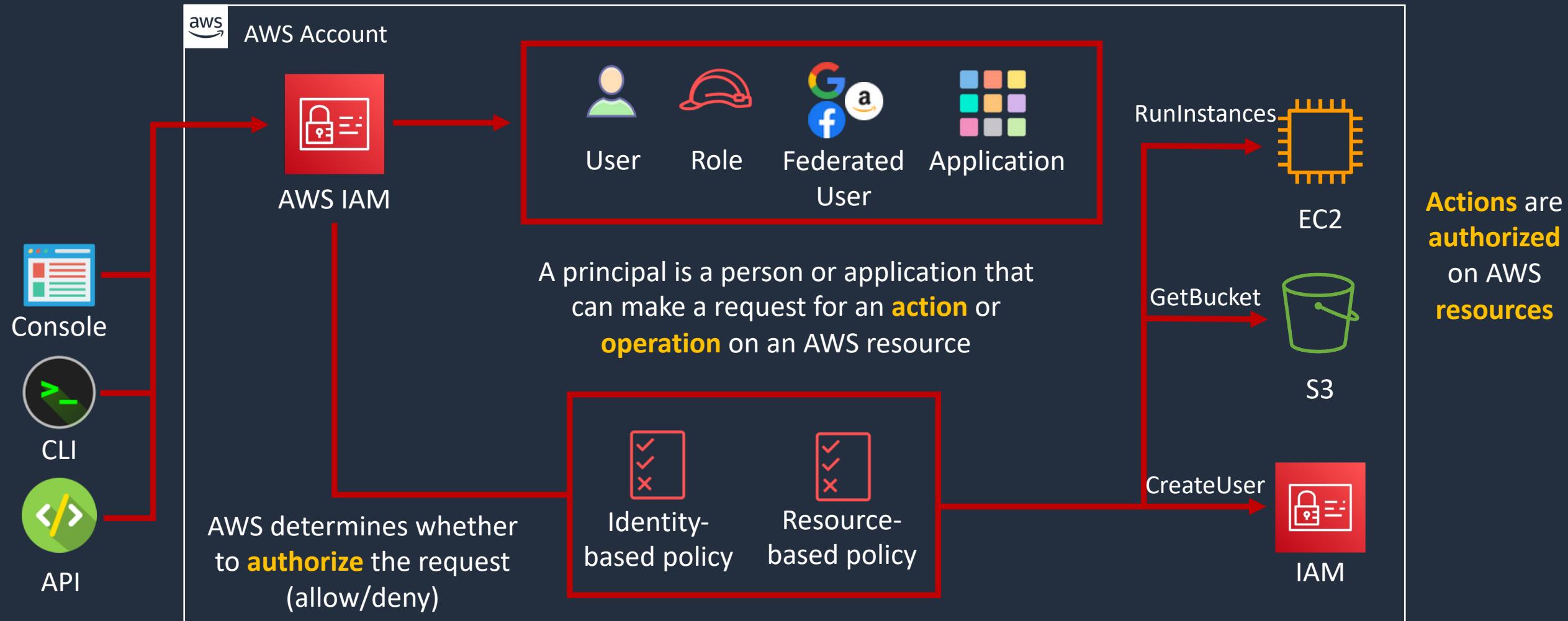
IAM Overview





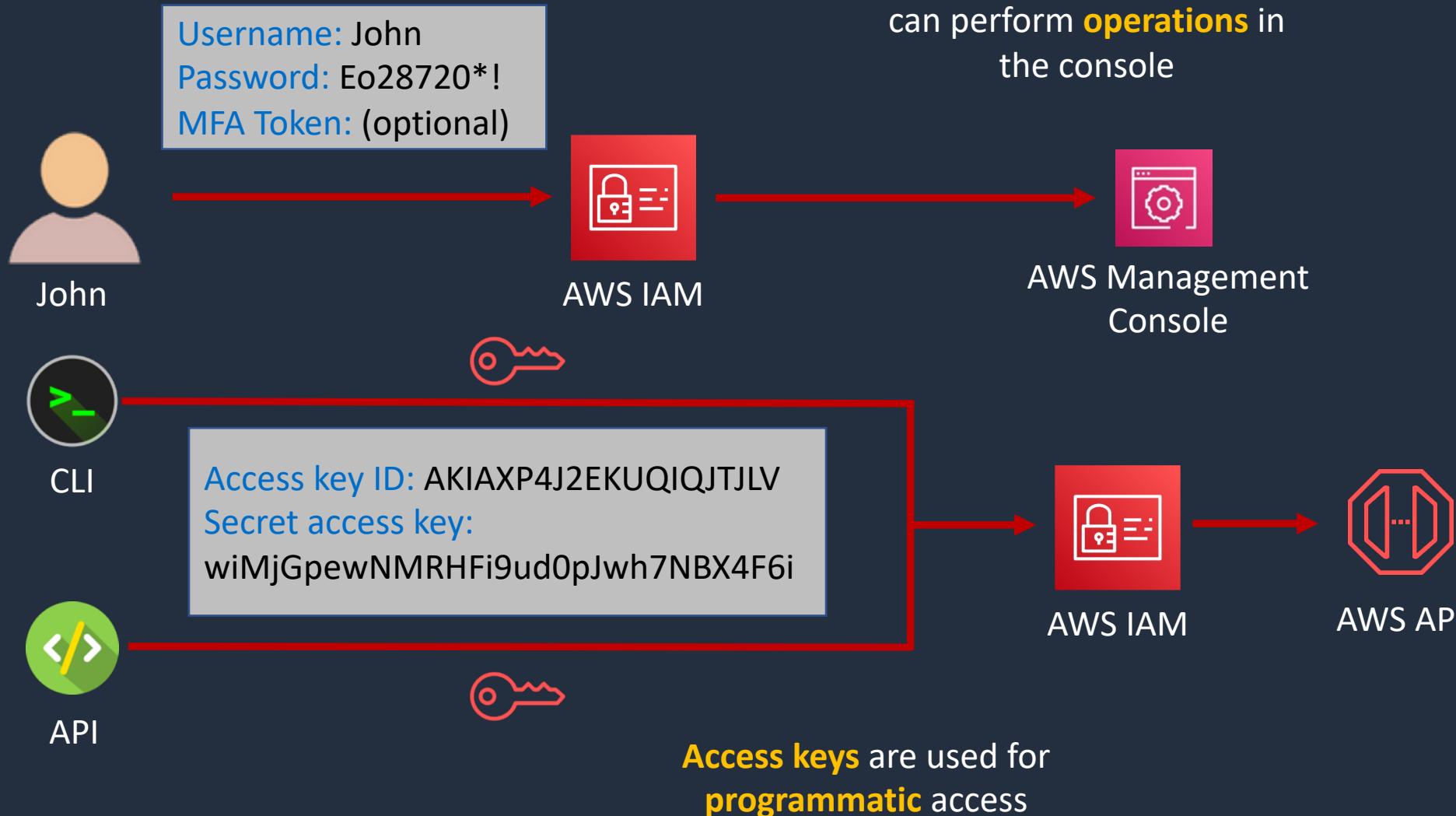
AWS Identity and Access Management (IAM)

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





IAM Authentication Methods



Create an IAM User and Group



Amazon Virtual Private Cloud (VPC)

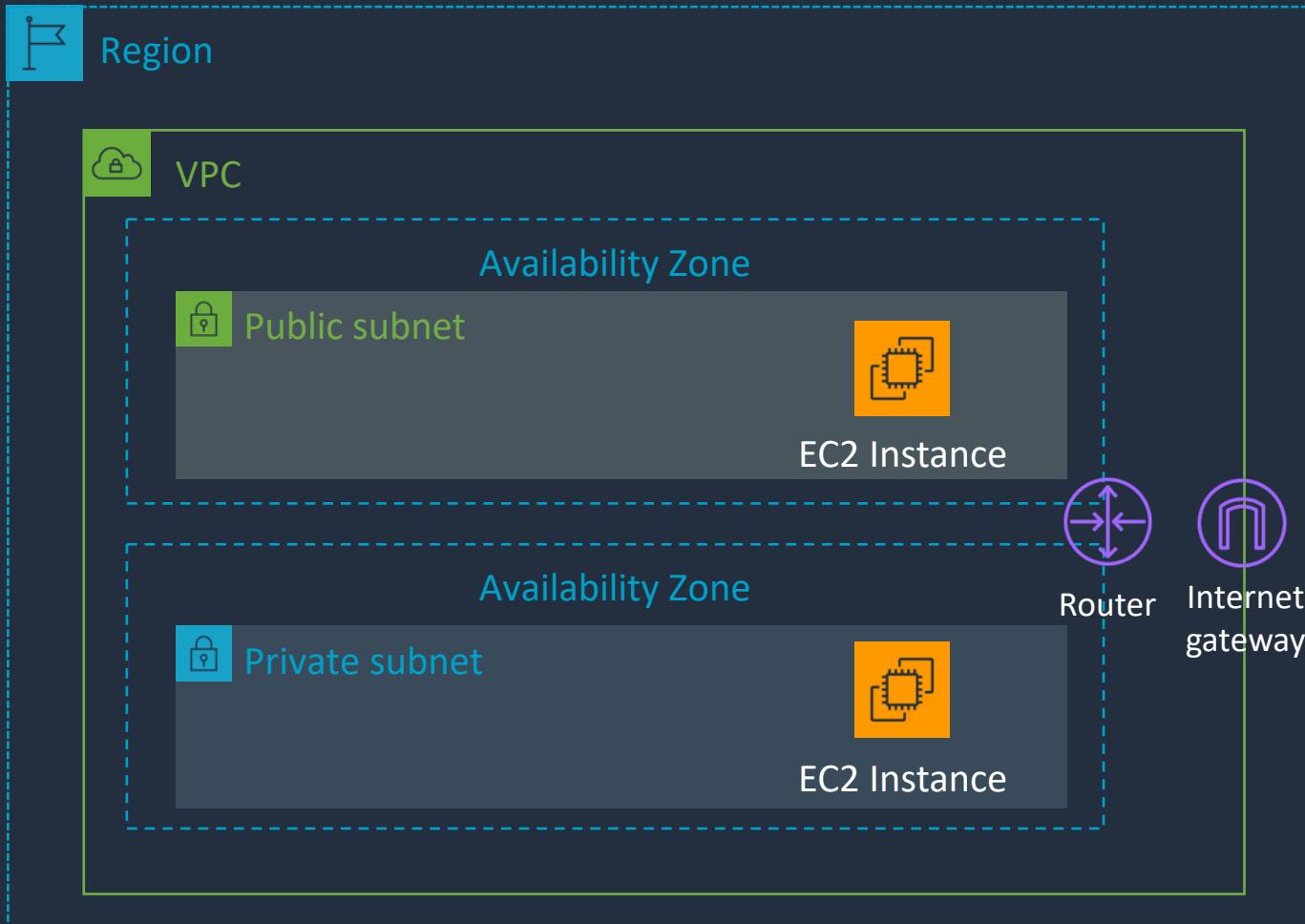




Amazon VPC

A **VPC** is a logically isolated portion of the AWS cloud within a region

Subnets are created within **AZs**



You can launch **EC2 instances** into your VPC subnets

Main Route Table

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

The **route table** is used to configure the VPC router

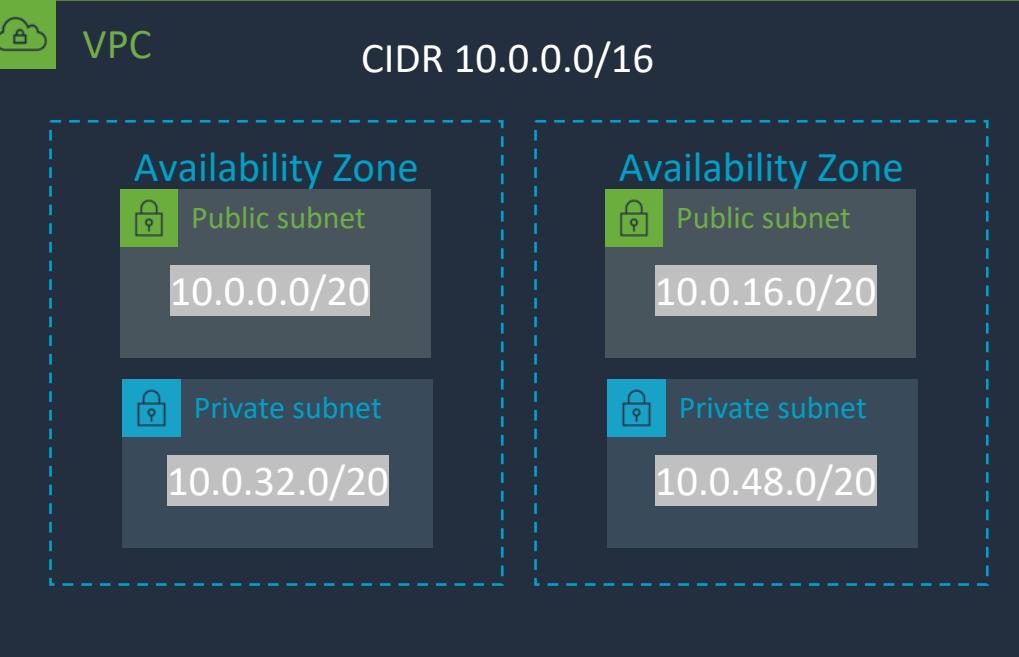
An **Internet Gateway** is used to connect to the Internet



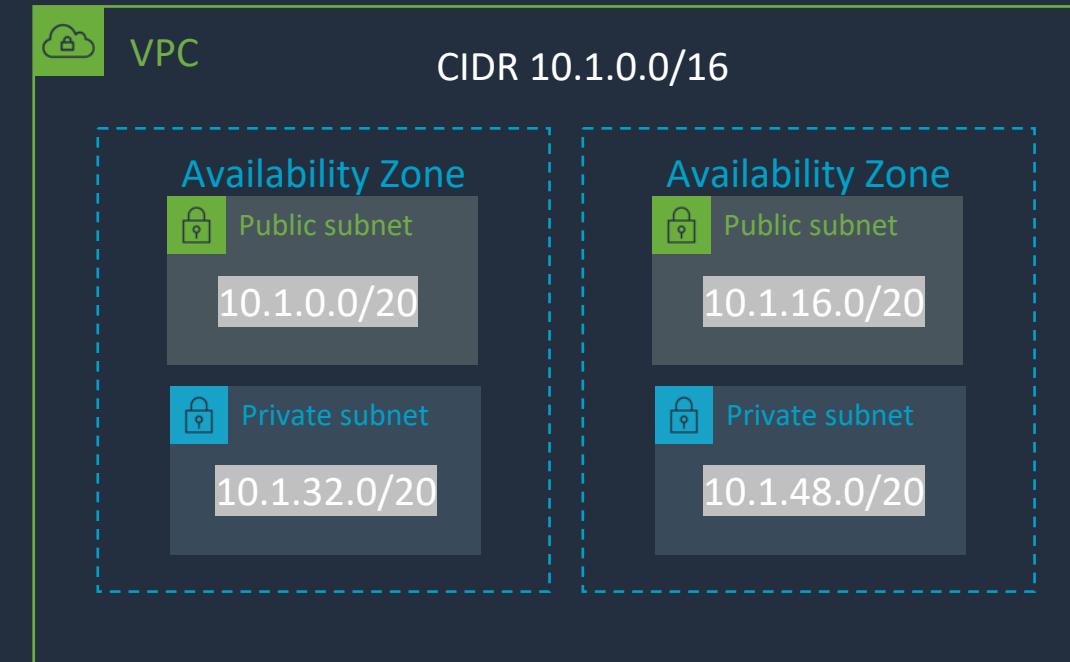
Amazon VPC

Each **VPC** has a different block of IP addresses

CIDR stands for Classless Interdomain Routing



Each subnet has a block of **IP addresses** from the CIDR block

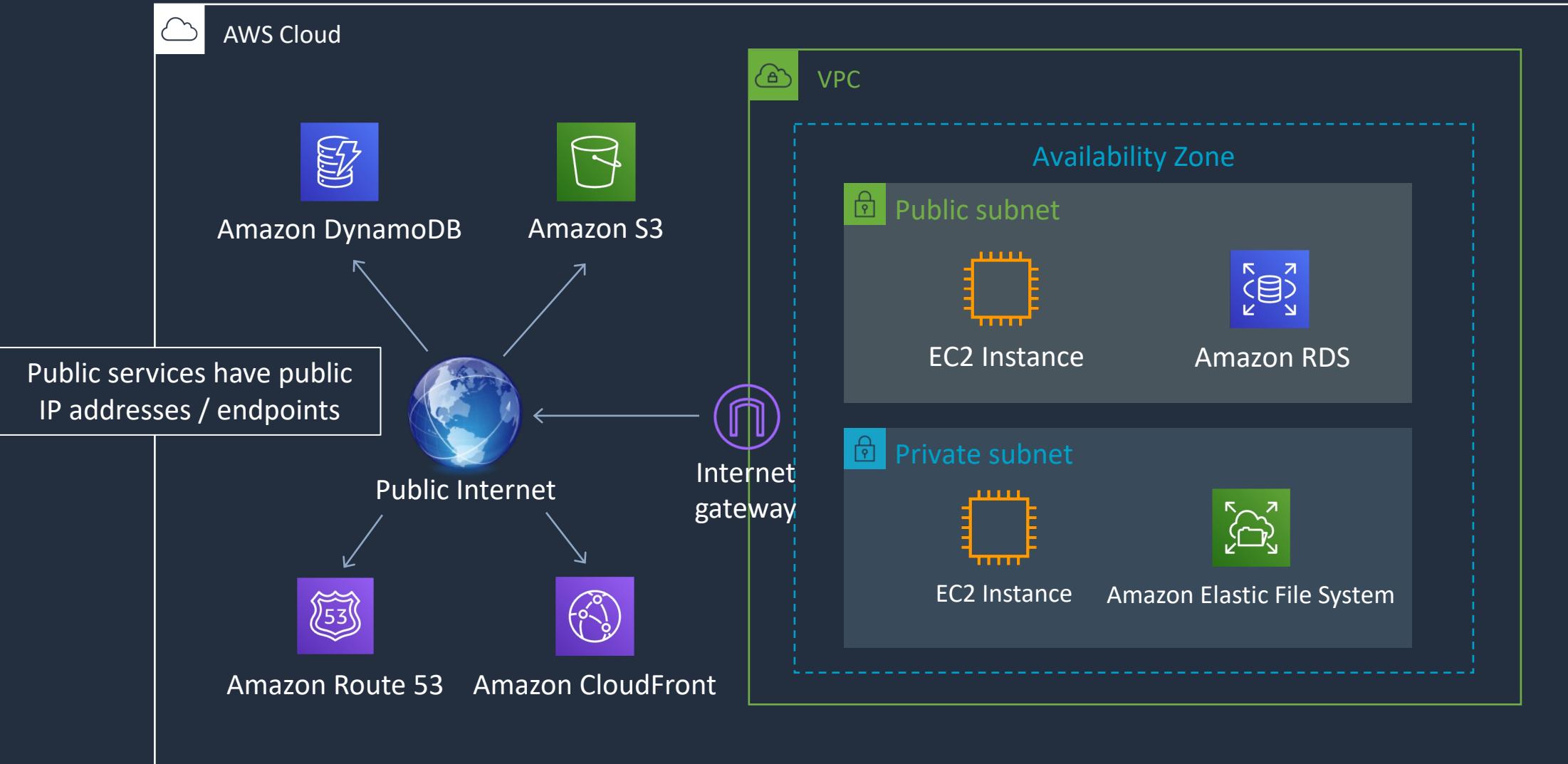


You can create **multiple VPCs** within each region



AWS Public and Private Services

Private services can have public IP addresses but exist within the VPC

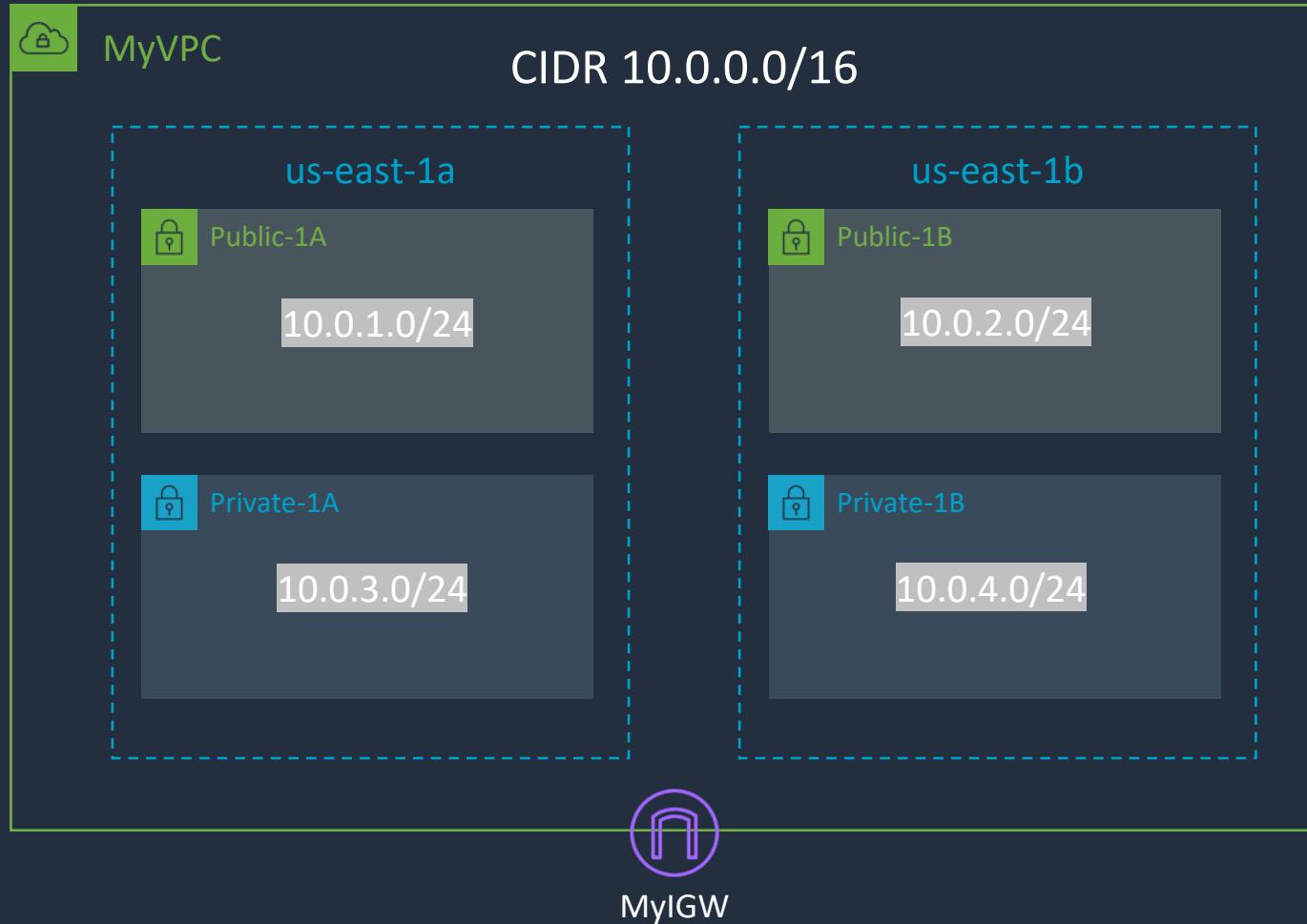


Create a Custom VPC





Custom VPC



Main Route Table

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

Private-RT Route Table

Destination	Target
10.0.0.0/16	Local

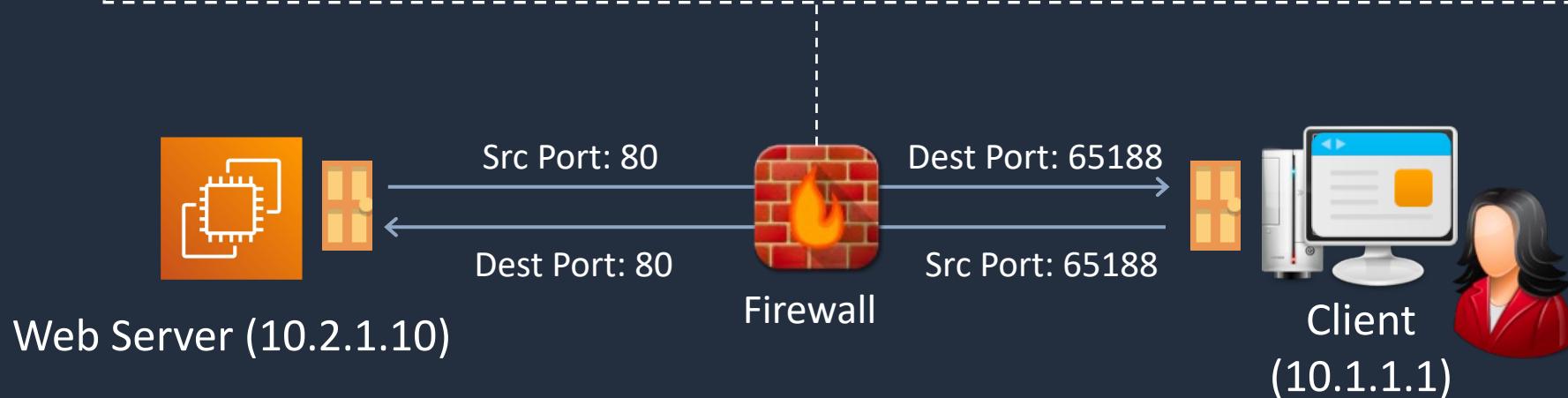
Security Groups and Network ACLs





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

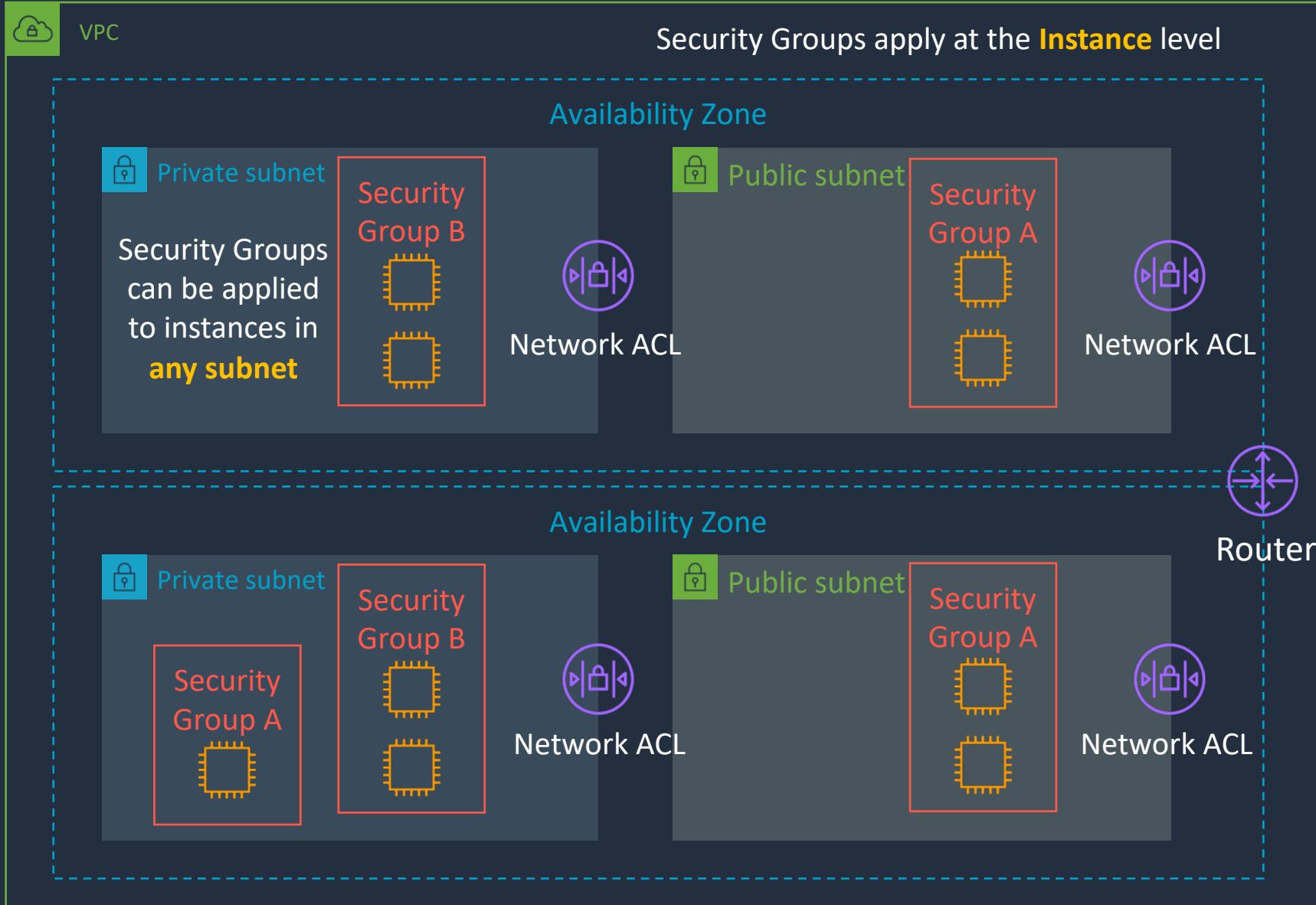


A **stateful** firewall allows the return traffic automatically

A **stateless** firewall checks for an allow rule for **both** connections



Security Groups and Network ACLs





Security Groups & Network Access Control Lists (NACLs)

Security Group	Network ACL
Operates at the instance (interface) level	Operates at the subnet level
Supports allow rules only	Supports allow and deny rules
Stateful	Stateless
Evaluates all rules	Processes rules in order
Applies to an instance only if associated with a group	Automatically applies to all instances in the subnets its associated with

Create a Security Group



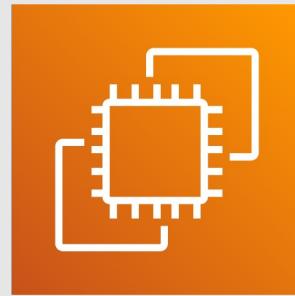
Configure the AWS CLI



SECTION 6

Amazon EC2, Auto Scaling, and Load Balancing

Amazon EC2 Overview



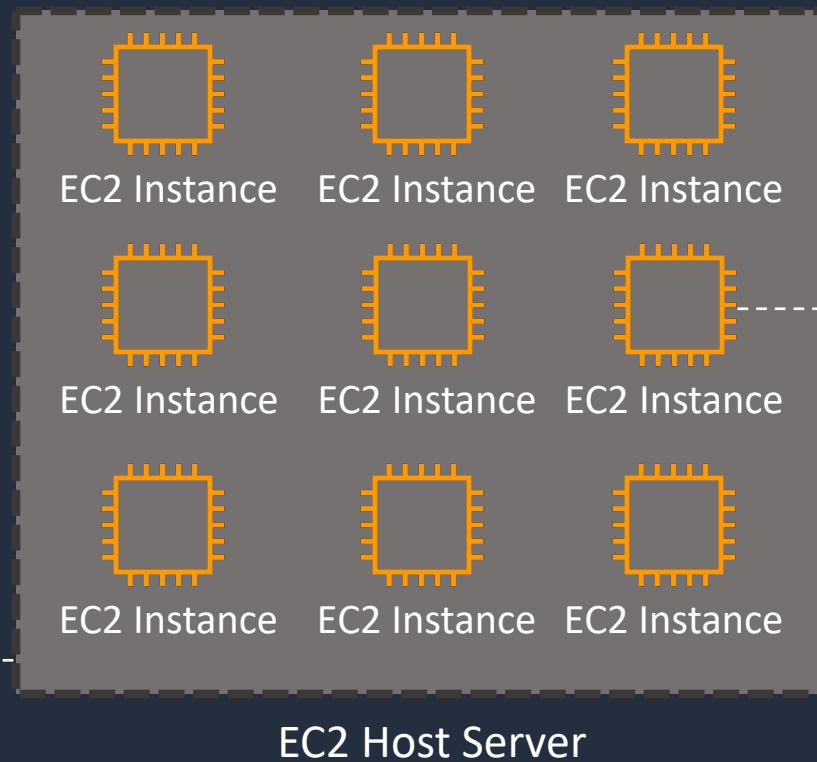
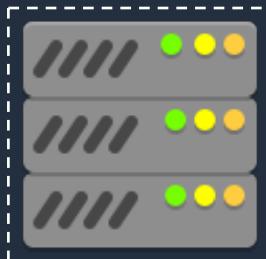


Amazon Elastic Compute Cloud (EC2)

EC2 instances run
Windows, Linux, or
MacOS

An **EC2 instance** is a
virtual server

EC2 hosts are
managed by AWS



A selection of **instance types**
come with varying combinations
of CPU, memory, storage and
networking

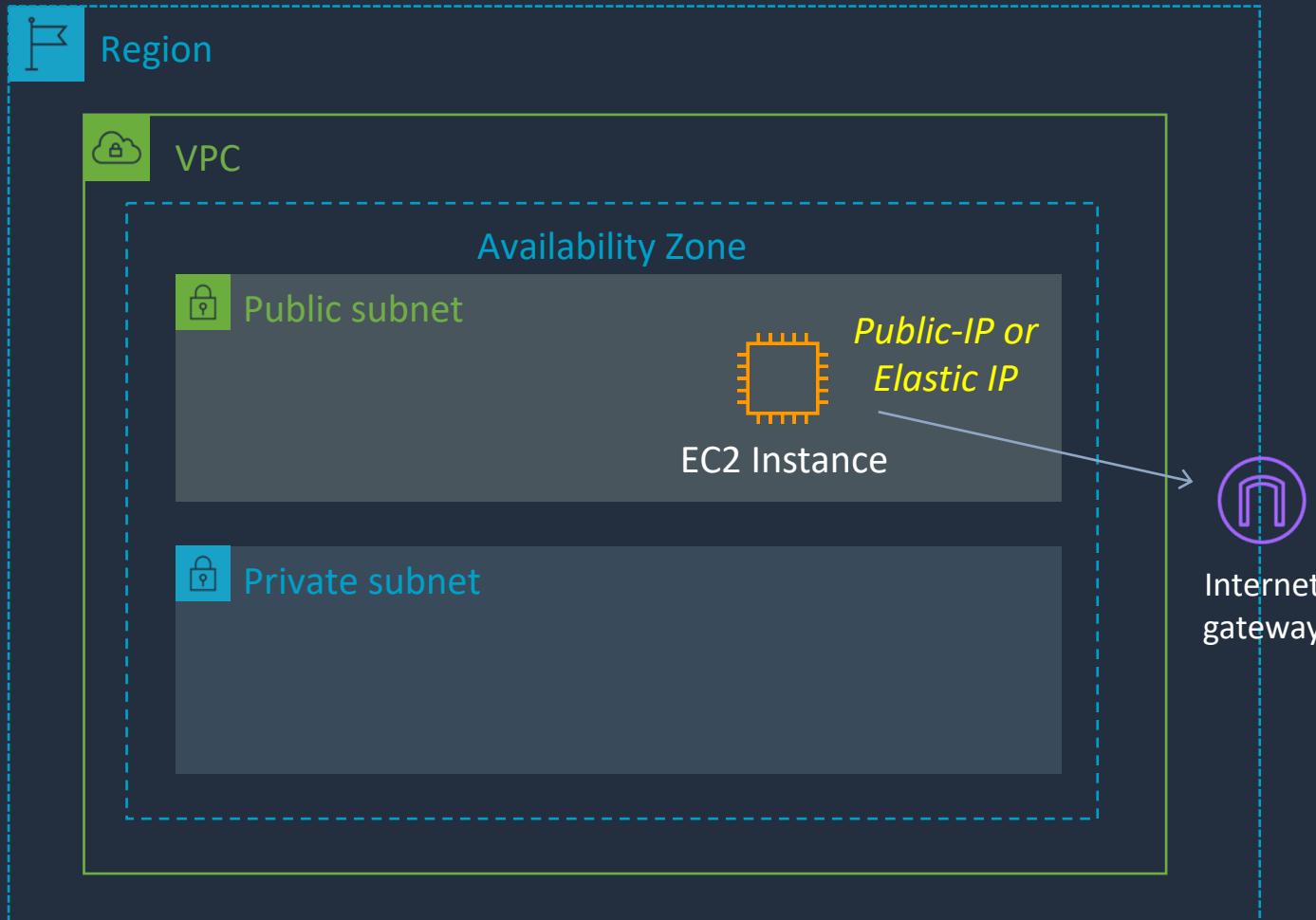


Public, Private, and Elastic IP addresses

Type	Description
Public IP address	<p>Lost when the instance is stopped</p> <p>Used in Public Subnets</p> <p>No charge</p> <p>Associated with a private IP address on the instance</p> <p>Cannot be moved between instances</p>
Private IP address	<p>Retained when the instance is stopped</p> <p>Used in Public and Private Subnets</p>
Elastic IP address	<p>Static Public IP address</p> <p>You are charged if not used</p> <p>Associated with a private IP address on the instance</p> <p>Can be moved between instances and Elastic Network Adapters</p>



Public Subnets



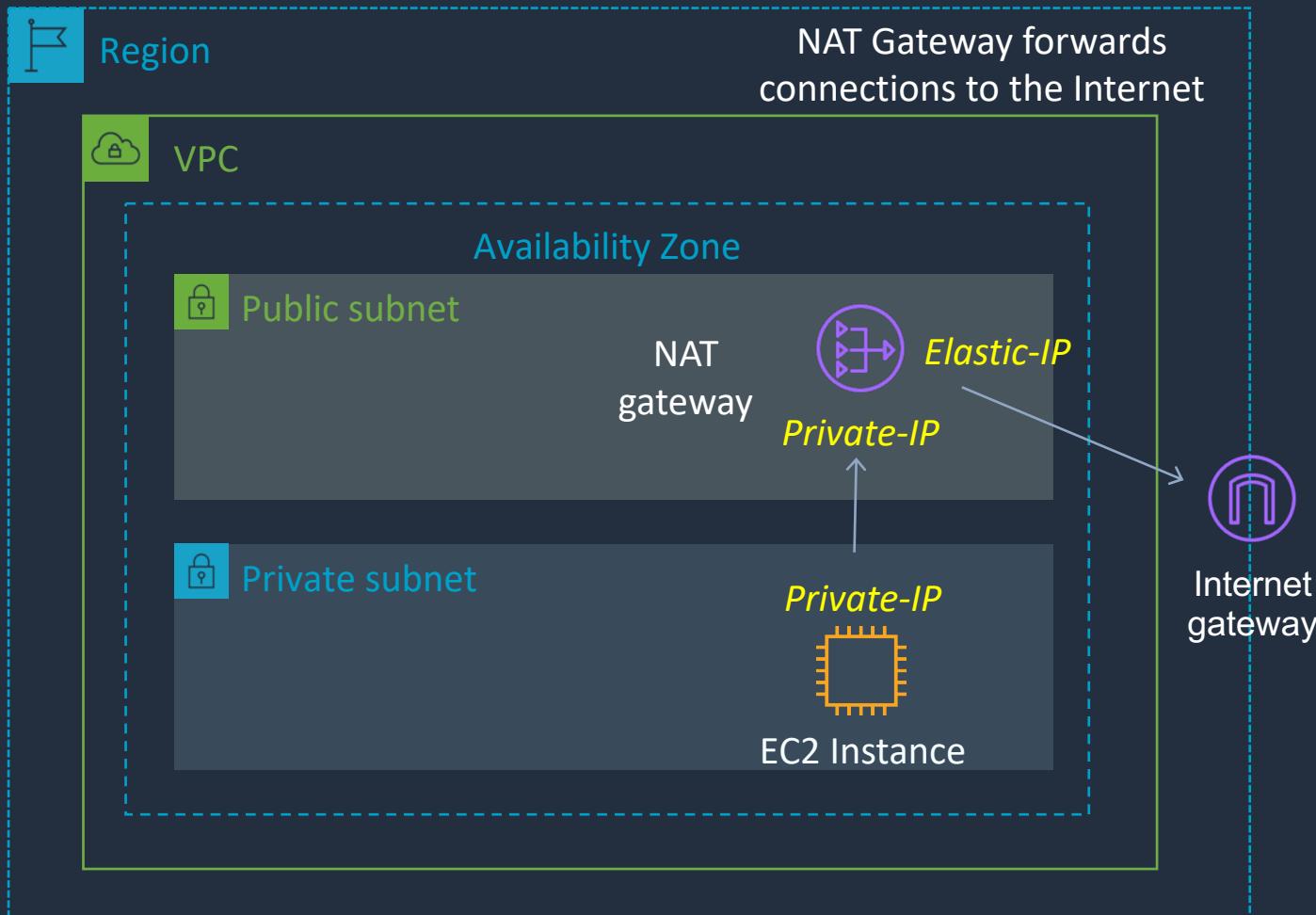
Public Subnet Route Table

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



Private Subnets

NAT = Network Address Translation



Public Subnet Route Table

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

Private Subnet Route Table

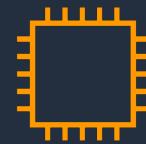
Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	nat-gateway-id

Launching Amazon EC2 Instances





Launching an EC2 Instance



Select an **instance type**

An **AMI** defines the configuration of the instance



Amazon Machine Image (AMI)

A **snapshot** is a point-in-time backup of an instance



EBS Snapshot



Linux



Microsoft Windows

You can customize your instance and create a **custom AMI**



Customized AMI

Family	Type	vCPUs	Memory (GiB)
General purpose	t2.micro	1	1
Compute optimized	c5n.large	2	5.25
Memory optimized	r5ad.large	2	16
Storage optimized	d2.xlarge	4	30.5
GPU instances	g2.2xlarge	8	15

The **instance type** defines the hardware profile (and cost)

Connecting to Amazon EC2

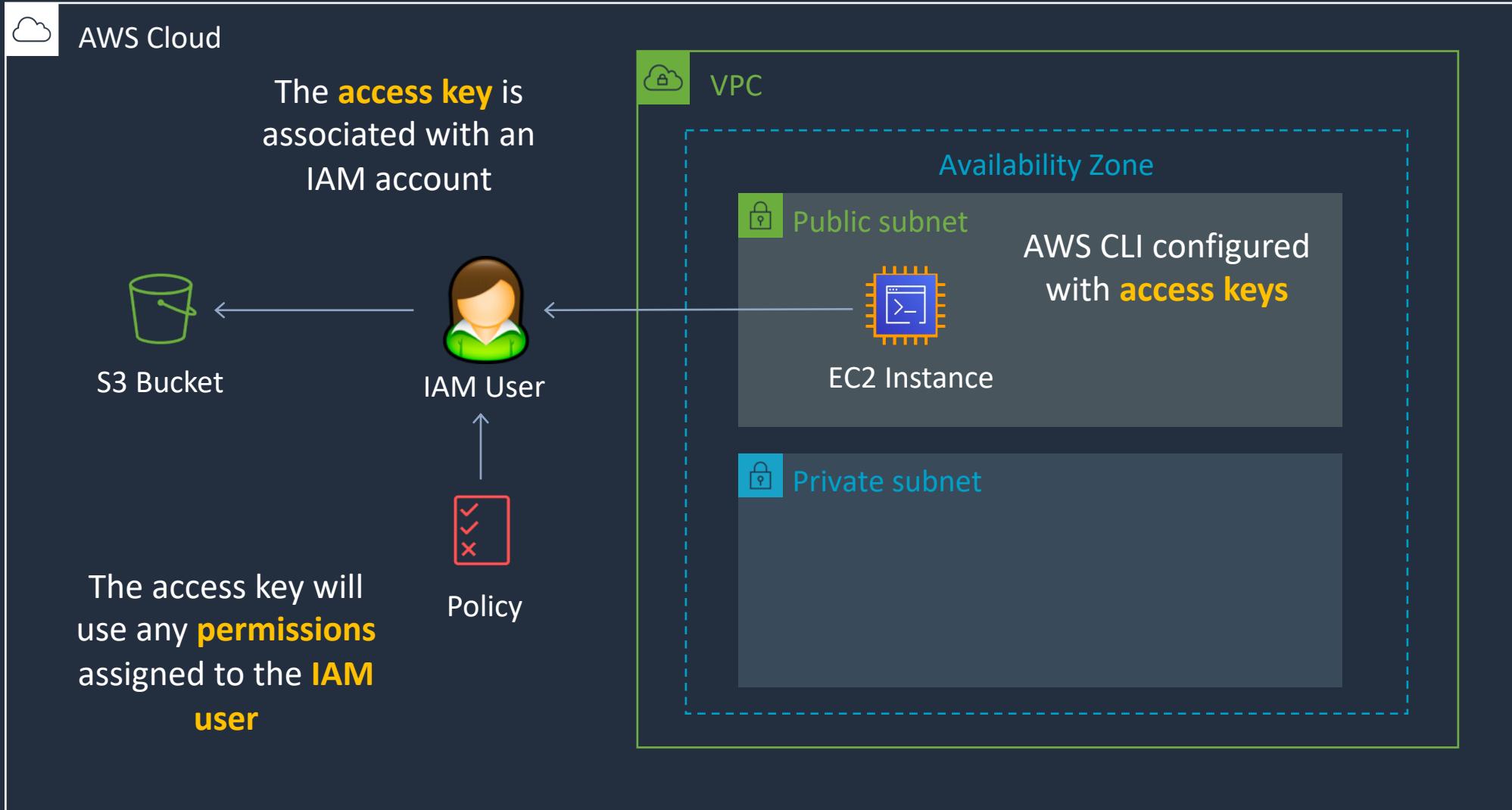


Access Keys and IAM Roles with EC2



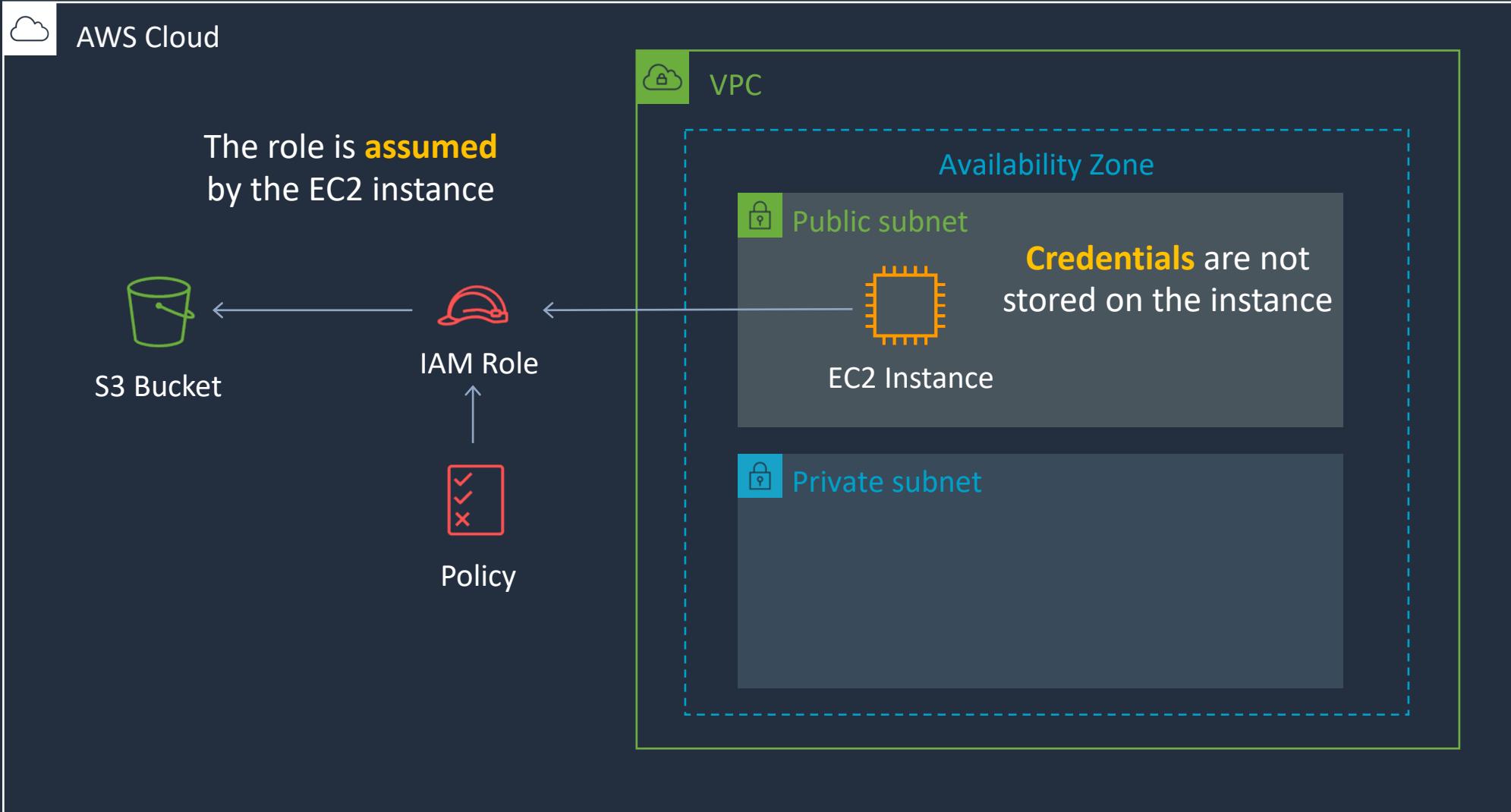


Using Access Keys with Amazon EC2





Using Roles with Amazon EC2



Practice with Access Keys and IAM Roles



Create a Website with User Data





Amazon EC2 User Data

The code is run when the instance starts for the **first time**

AWS Management Console

Batch and **PowerShell** scripts can be run on Windows

The screenshot shows the 'User data' configuration in the AWS Management Console. It includes options for 'As text' (selected), 'As file', and 'Input is already base64 encoded'. The user data script is displayed in a text area:

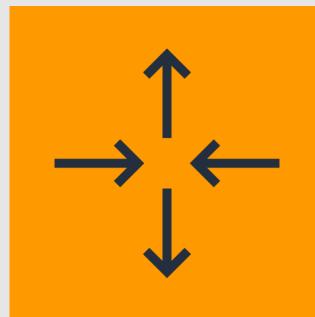
```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
```

Limited to
16 KB



EC2 Instance with a **web service** is launched

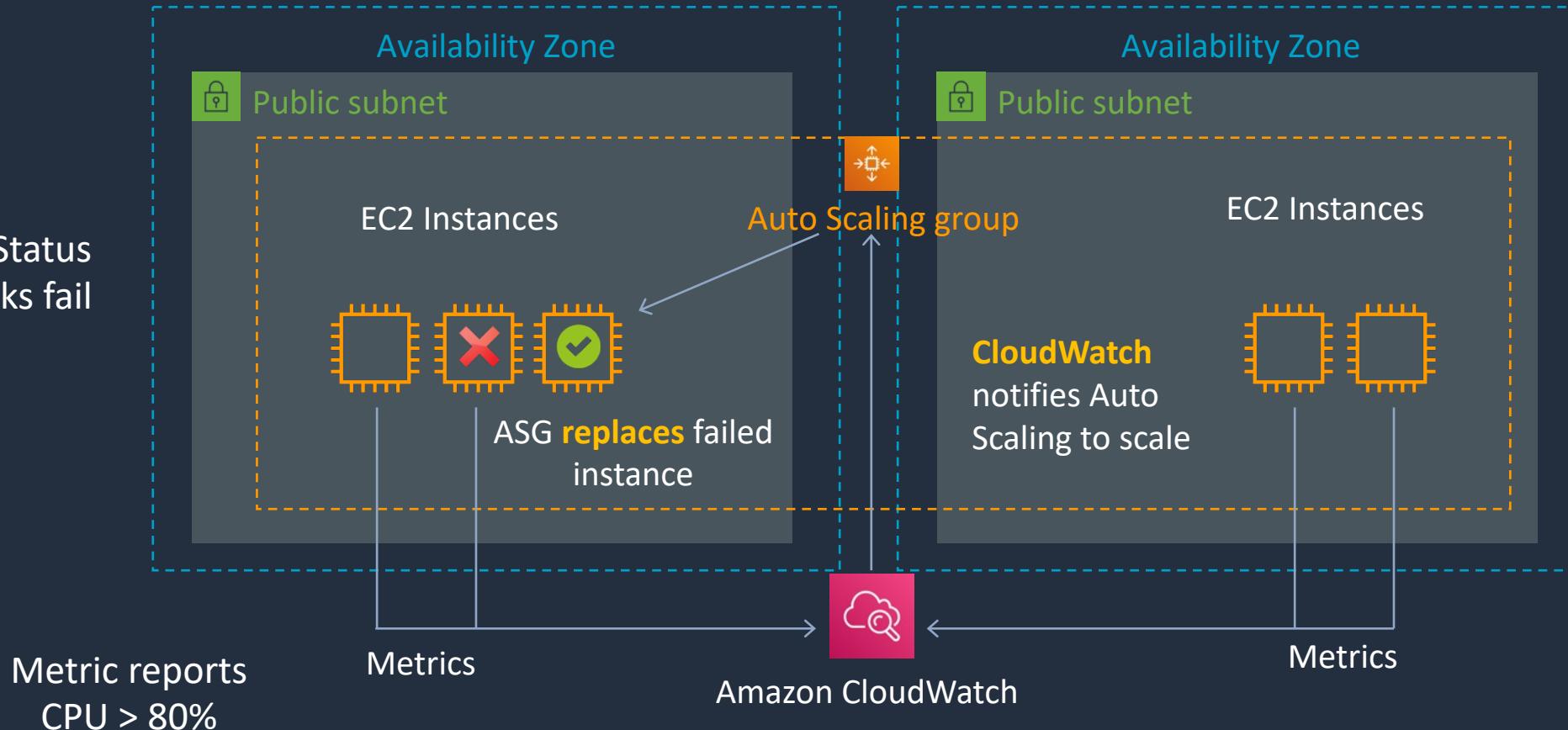
Amazon EC2 Auto Scaling



Amazon EC2 Auto Scaling

EC2 Status
Checks fail

Auto Scaling launches
extra instance



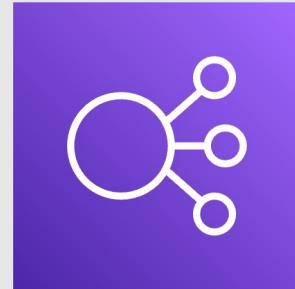
Create an Auto Scaling Group



Create a Scaling Policy



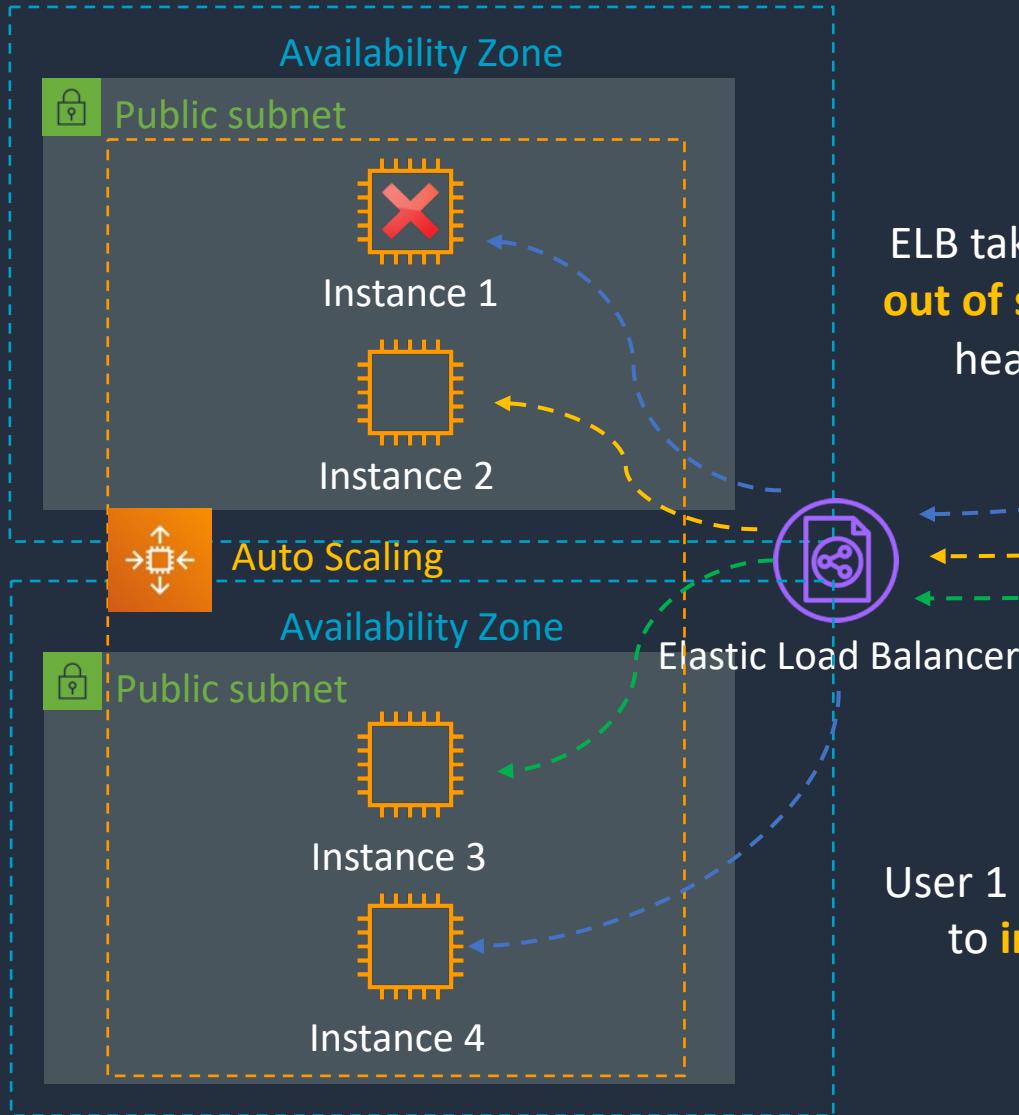
Amazon Elastic Load Balancing





Amazon Elastic Load Balancing

EC2 Auto Scaling
terminates
instance 1



ELB takes instance 1
out of service (failed
health check)

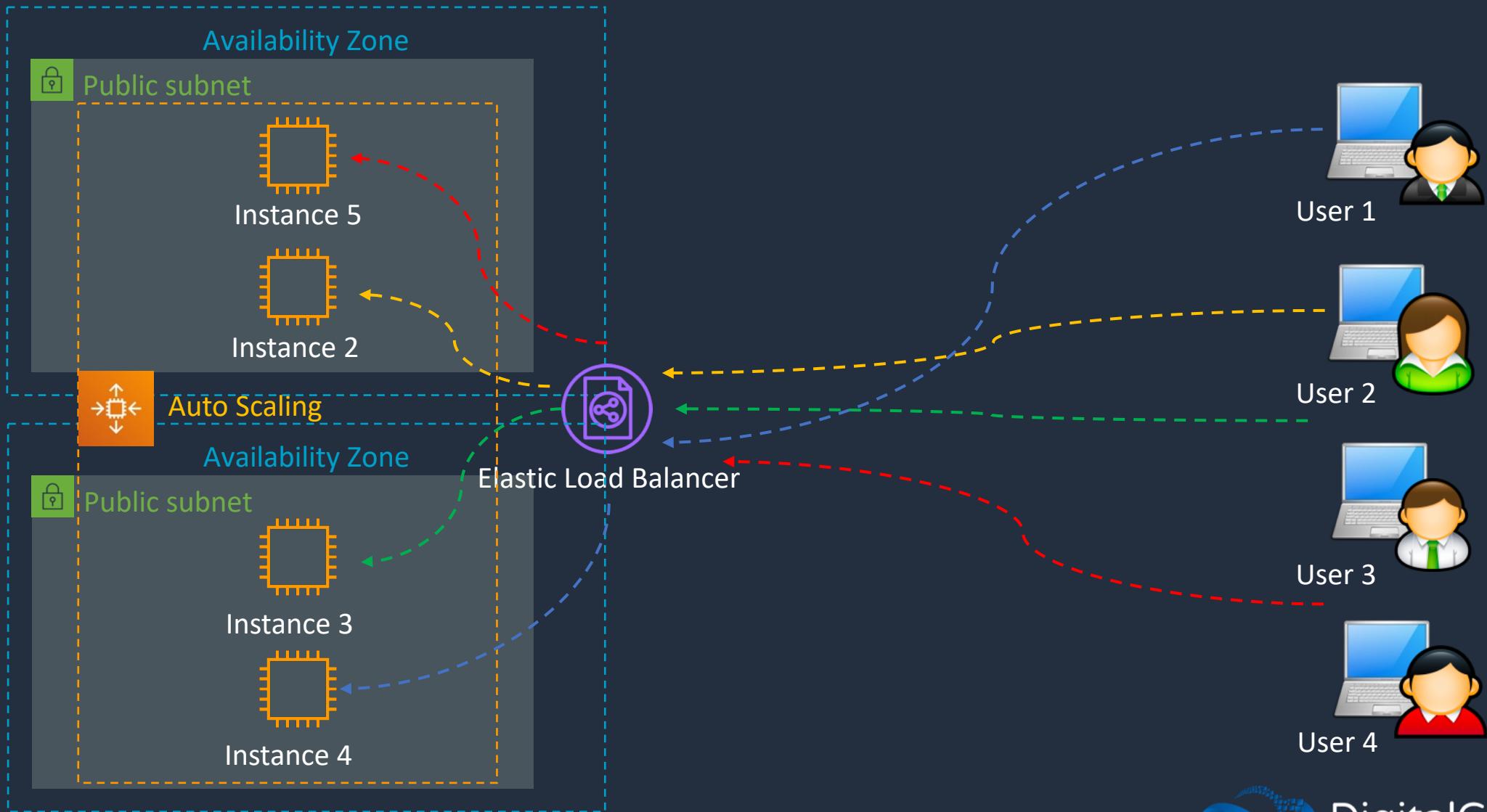
User 1 is connected
to **instance 4**





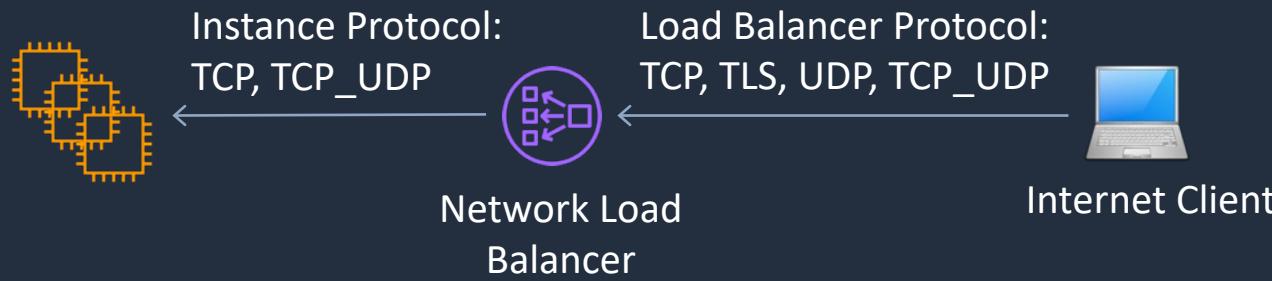
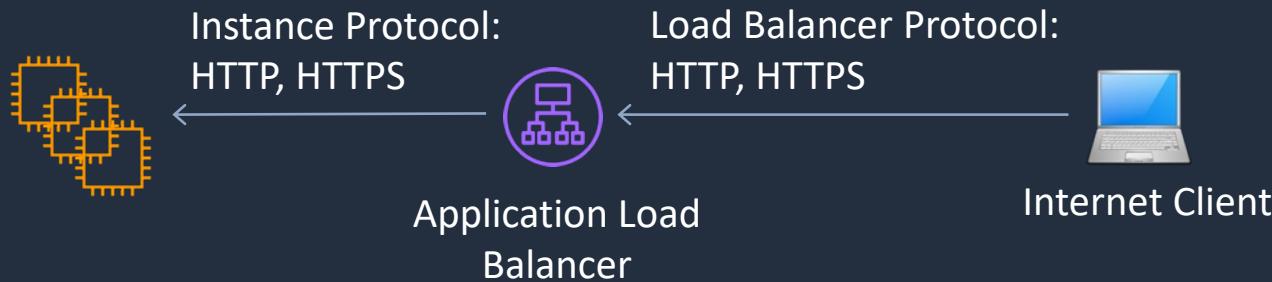
Amazon Elastic Load Balancing

EC2 Auto Scaling
launches
instance 5





Types of Elastic Load Balancer (ELB)



Application Load Balancer

- Operates at the request level
- Routes based on the content of the request (layer 7)
- Supports path-based routing, host-based routing, query string parameter-based routing, and source IP address-based routing
- Supports instances, IP addresses, Lambda functions and containers as targets

Network Load Balancer

- Operates at the connection level
- Routes connections based on IP protocol data (layer 4)
- Offers ultra high performance, low latency and TLS offloading at scale
- Can have a static IP / Elastic IP
- Supports UDP and static IP addresses as targets



ELB Use Cases

Application Load Balancer

- Web applications with L7 routing (HTTP/HTTPS)
- Microservices architectures (e.g. Docker containers)
- Lambda targets

Network Load Balancer

- TCP and UDP based applications
- Ultra-low latency
- Static IP addresses
- VPC endpoint services

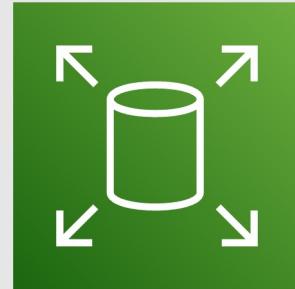
Create an Application Load Balancer



SECTION 7

AWS Storage Services

Amazon EBS and Instance Stores





Amazon Elastic Block Store (EBS)

EBS volumes exist within an **Availability Zone**

Availability Zone



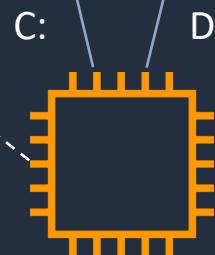
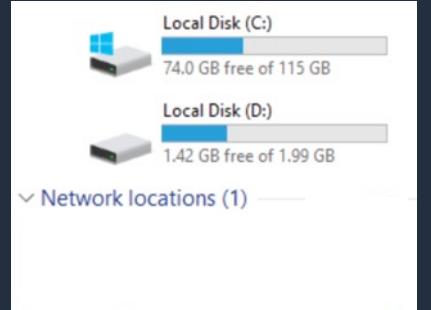
EBS Volume



EBS Volume

The volume is automatically replicated **within the AZ**

The volume is **attached** over a network



EC2 Instance



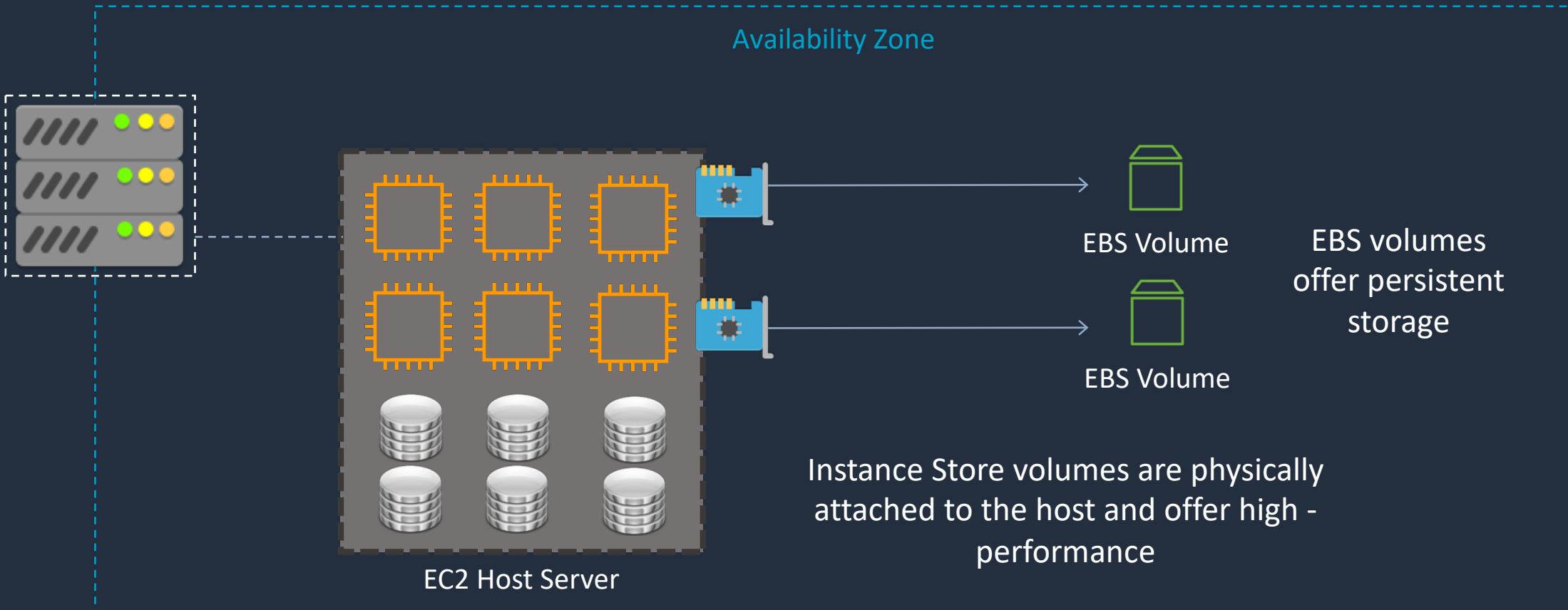
Amazon EBS Volume Types

	General Purpose SSD		Provisioned IOPS SSD		
Volume type	gp3	gp2	io2 Block Express ‡	io2	io1
Durability	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.999% durability (0.001% annual failure rate)		99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
Use cases	<ul style="list-style-type: none">Low-latency interactive appsDevelopment and test environments		Workloads that require sub-millisecond latency, and sustained IOPS performance or more than 64,000 IOPS or 1,000 MiB/s of throughput		<ul style="list-style-type: none">Workloads that require sustained IOPS performance or more than 16,000 IOPSI/O-intensive database workloads
Volume size	1 GiB - 16 TiB		4 GiB - 64 TiB	4 GiB - 16 TiB	
Max IOPS per volume (16 KiB I/O)	16,000		256,000	64,000 †	
Max throughput per volume	1,000 MiB/s	250 MiB/s *	4,000 MiB/s	1,000 MiB/s †	
Amazon EBS Multi-attach	Not supported		Not supported	Supported	
Boot volume	Supported				



Amazon EBS vs Instance Store

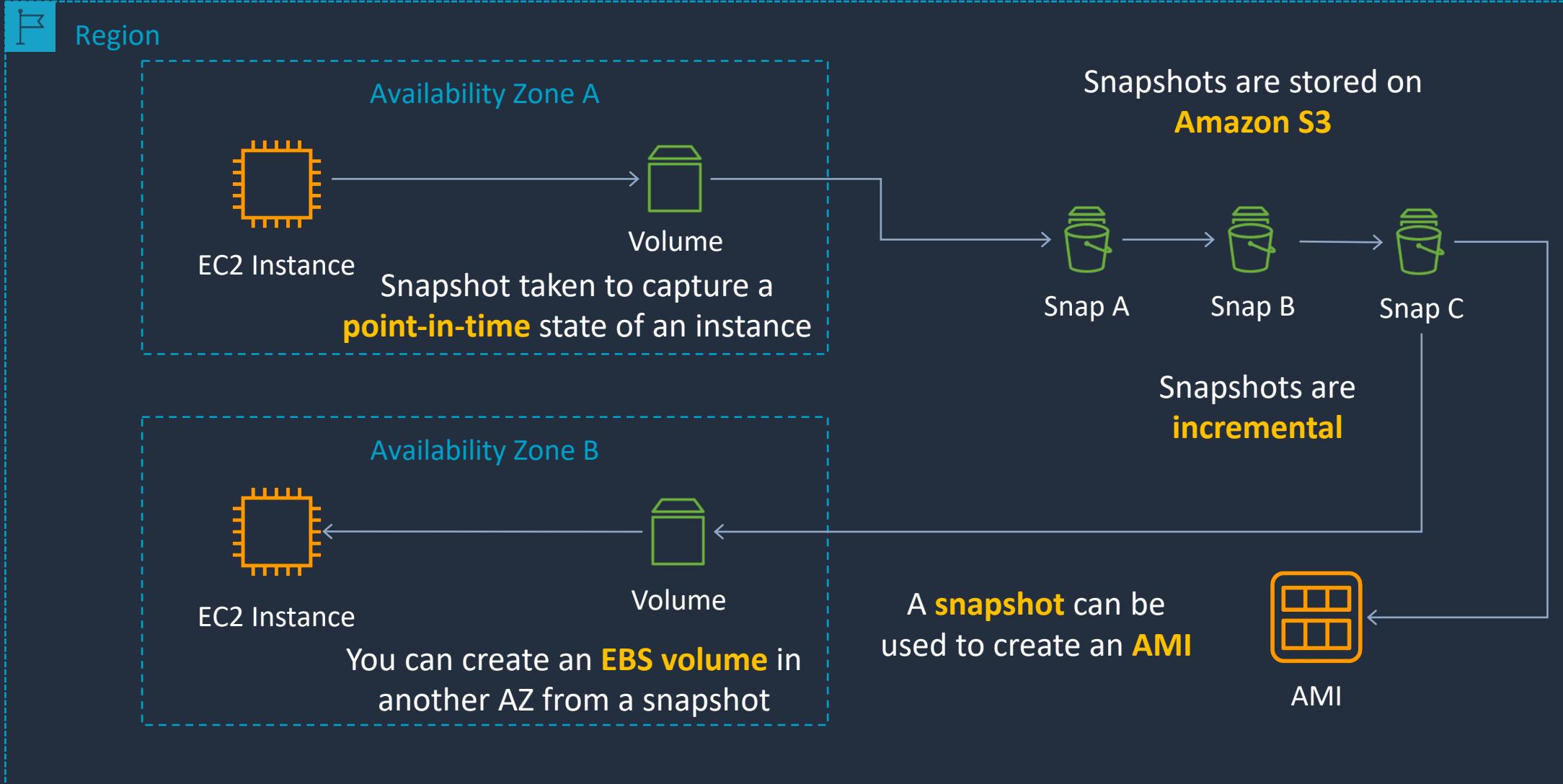
EBS volumes are **attached** over the **network**



Instance Store volumes are **ephemeral** (non-persistent)



Amazon EBS Snapshots



Create and Attach an EBS Volume



EBS Snapshots and AMIs

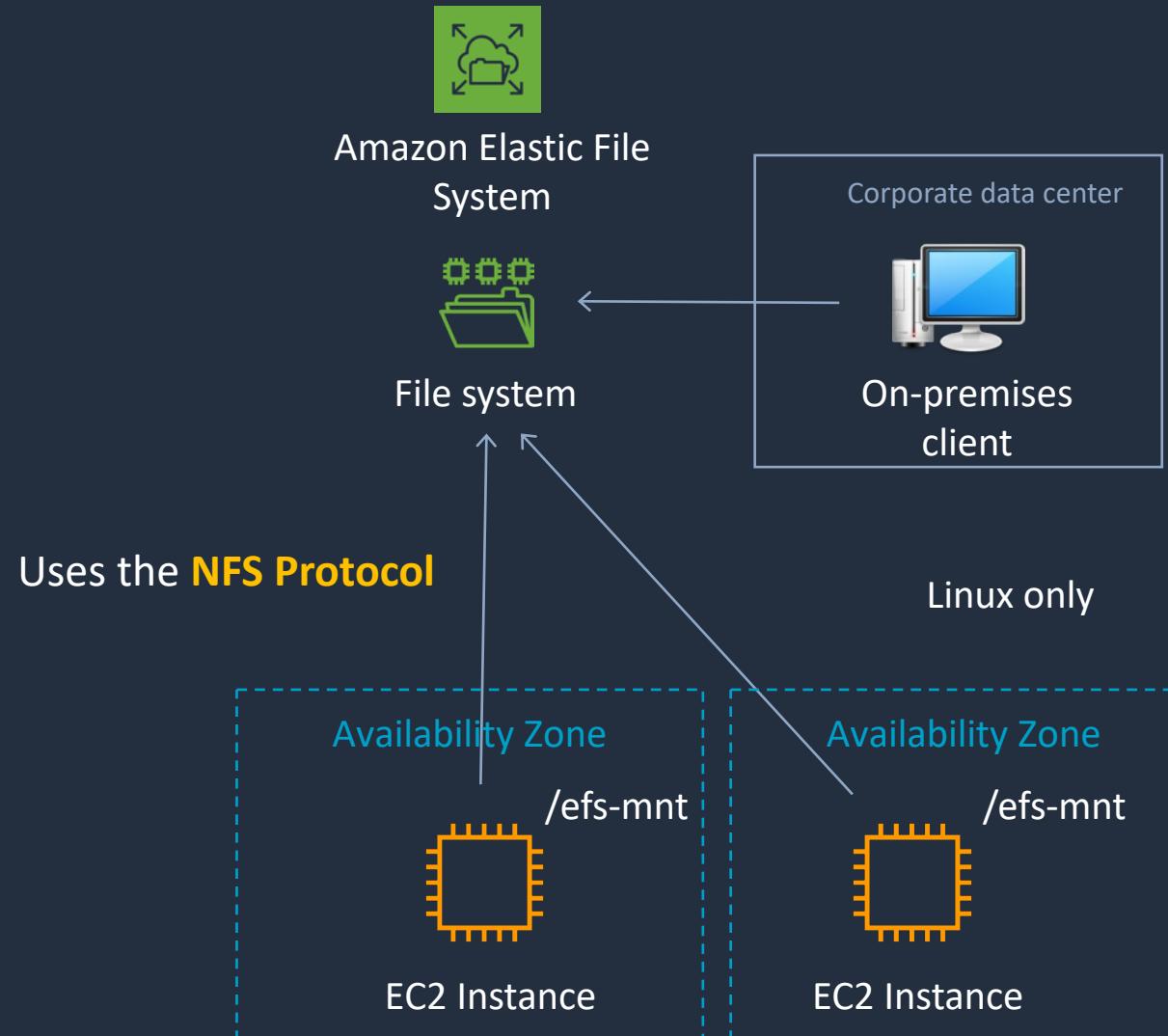


Amazon Elastic File System (EFS)





Amazon Elastic File System (EFS)



Create an Amazon EFS Filesystem





Mounting EFS File Systems

```
mkdir ~/efs-mount-point
```

Install NFS client

```
sudo yum -y install nfs-utils
```

Mount using the NFS client

```
sudo mount -t nfs4 -o  
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport  
fs-0187058e25d360ce2.efs.us-east-1.amazonaws.com:/ ~/efs-mount-point
```

Install EFS utils

```
sudo yum install -y amazon-efs-utils
```

Mount using the EFS mount helper:

```
sudo mount -t efs -o tls fs-0187058e25d360ce2.efs.us-east-1.amazonaws.com:/  
~/efs-mount-point
```

Amazon Simple Storage Service (S3)





Amazon Simple Storage Service (S3)



S3 Bucket



A **bucket** is a container for objects

An **object** is a file you upload

You can store millions of **objects** in a **bucket**



Accessing objects in a bucket:

`https://bucket.s3.aws-region.amazonaws.com/key`
`https://s3.aws-region.amazonaws.com/bucket/key`

The **HTTP protocol** is used with a **REST API** (e.g. GET, PUT, POST, SELECT, DELETE)



Amazon Simple Storage Service (S3)



Bucket

A **bucket** is a container for objects

<http://bucket.s3.aws-region.amazonaws.com>

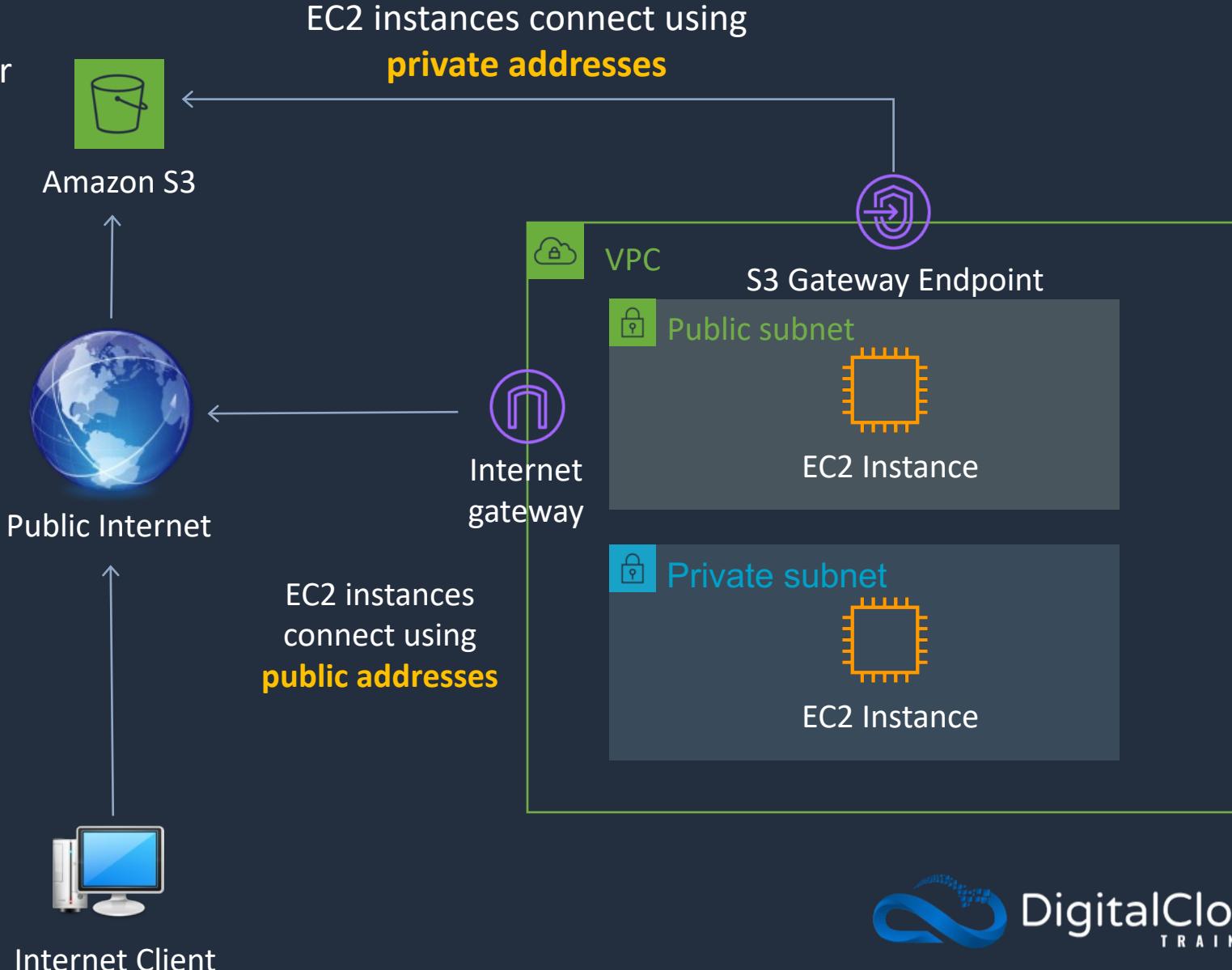
<http://s3.aws-region.amazonaws.com/bucket>



Object

An object consists of:

- Key (name of objects)
- Version ID
- Value (actual data)
- Metadata
- Subresources
- Access control information





File Storage vs Object Storage

File Share



- Data stored in directories
- A hierarchy of directories can be formed
- File systems are mounted to an operating system
- Function like local storage
- Network connection is maintained
- Example is Amazon EFS

Object Store



[http://*bucket*.s3.*aws-region*.amazonaws.com](http://bucket.s3.amazonaws.com)

- Data stored in buckets
- Flat namespace (no hierarchy)
- Hierarchy can be mimicked with prefixes
- Accessed by **REST API** and cannot be mounted
- Network connection is completed after each request
- Example is Amazon S3

Working with S3 Buckets and Objects



Create an S3 Static Website



SECTION 8

AWS Database Services

Amazon Relational Database Service (RDS)





Amazon RDS

RDS runs on EC2 instances, so you choose an **instance type**

RDS is a managed, **relational database**

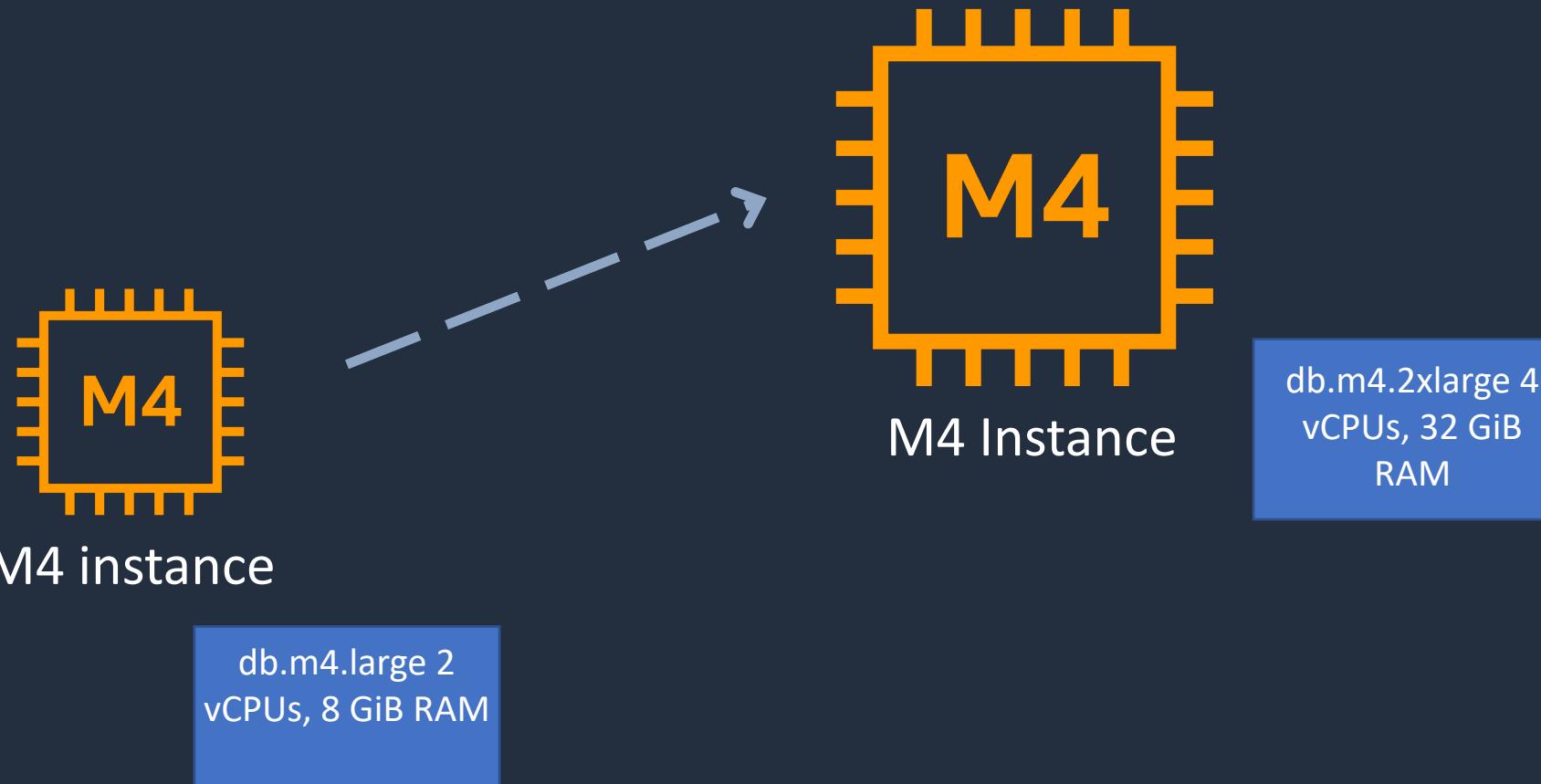


RDS supports the following database engines:

- Amazon Aurora
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL



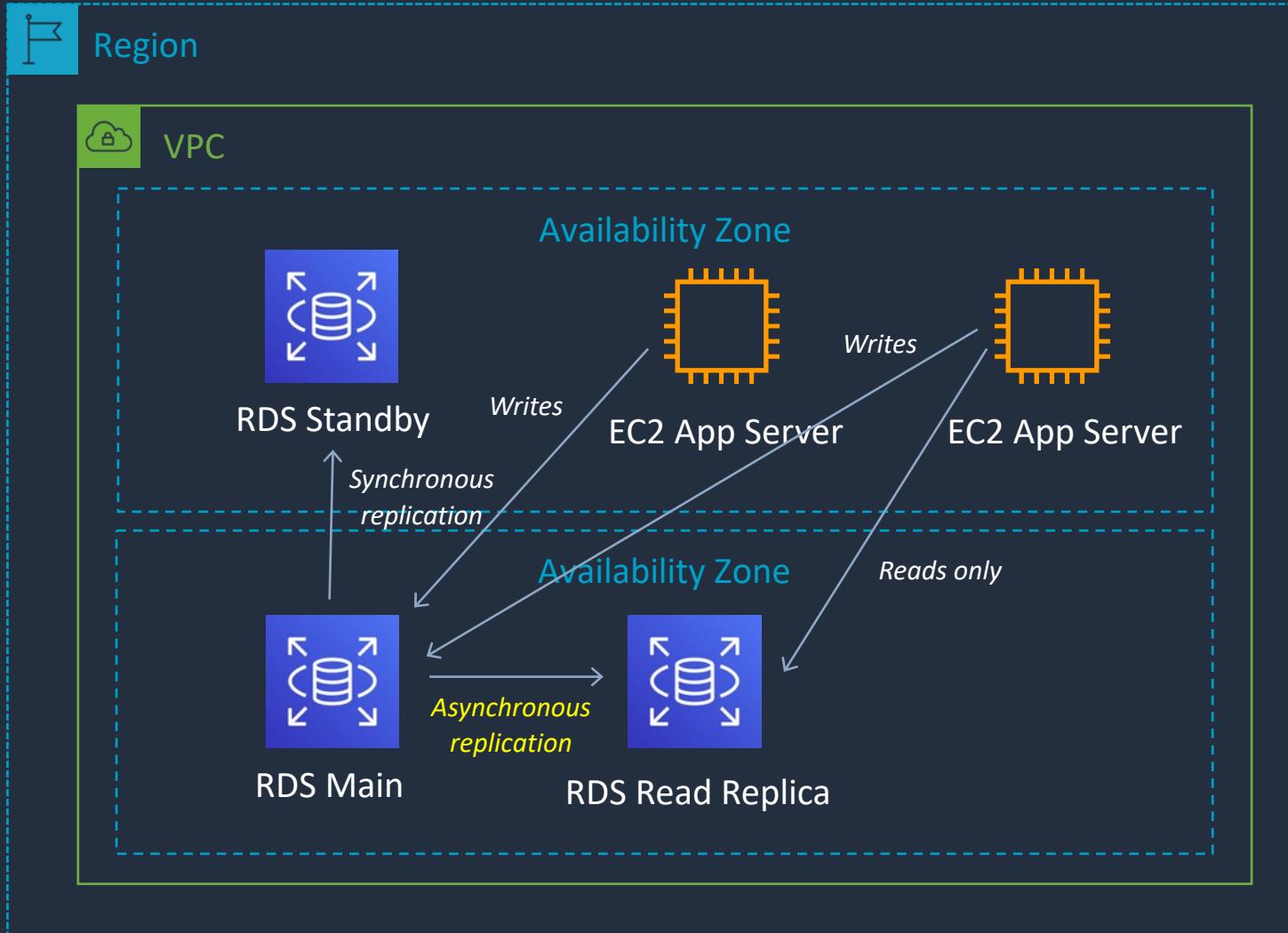
Amazon RDS – Scaling up (vertically)





Amazon RDS – Disaster Recovery (DR) and Scaling Out (horizontally)

Multi-AZ creates a passive standby.
Primarily used for **disaster recovery**



Application servers can read from the read replica and write to the master

Read Replicas are used for scaling database queries (reads)

Create an Amazon RDS Database



Add an Amazon RDS Read Replica



Amazon DynamoDB





Amazon DynamoDB

Fully managed service. You create tables on an existing database

Offers seamless, horizontal, **scaling**



DynamoDB Table

DynamoDB is a **NoSQL**, key-value type of database

Data is replicated across **multiple AZs** within a region



Amazon DynamoDB

- DynamoDB is made up of:
 - Tables
 - Items
 - Attributes

userid	orderid	book	price	date
user001	1000092	ISBN100..	9.99	2020.04..
user002	1000102	ISBN100..	24.99	2020.03..
user003	1000168	ISBN2X0..	12.50	2020.04..

Create an Amazon DynamoDB Table



SECTION 9

Automation and DevOps on AWS

Infrastructure as Code with AWS CloudFormation





AWS CloudFormation

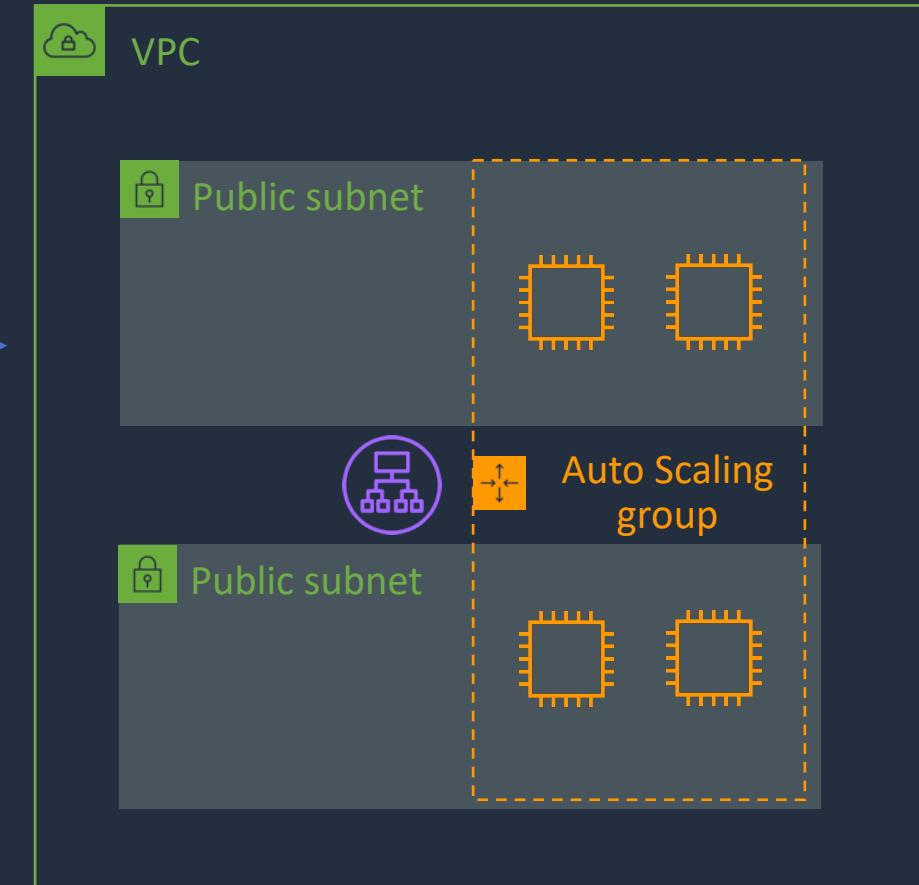
Infrastructure patterns are defined in a **template** file using **code**



AWS CloudFormation

```
1 "AWSTemplateFormatVersion" : "2010-09-09",
2
3 "Description" : "AWS CloudFormation Sample Template WordPress_Multi_AZ: WordPress is web
4
5 "Parameters" : {
6   "VpcId" : {
7     "Type" : "AWS::EC2::VPC::Id",
8     "Description" : "VpcId of your existing Virtual Private Cloud (VPC)",
9     "ConstraintDescription" : "must be the VPC Id of an existing Virtual Private Cloud."
10 },
11
12 "Subnets" : {
13   "Type" : "List<AWS::EC2::Subnet::Id>",
14   "Description" : "The list of SubnetIds in your Virtual Private Cloud (VPC)",
15   "ConstraintDescription" : "must be a list of at least two existing subnets associated
16 },
```

CloudFormation **builds** your infrastructure according to the **template**





AWS CloudFormation

Component	Description
Templates	The JSON or YAML text file that contains the instructions for building out the AWS environment
Stacks	The entire environment described by the template and created, updated, and deleted as a single unit
StackSets	AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation
Change Sets	A summary of proposed changes to your stack that will allow you to see how those changes might impact your existing resources before implementing them

Creating and Updating Stacks



Deploy a VPC Using CloudFormation



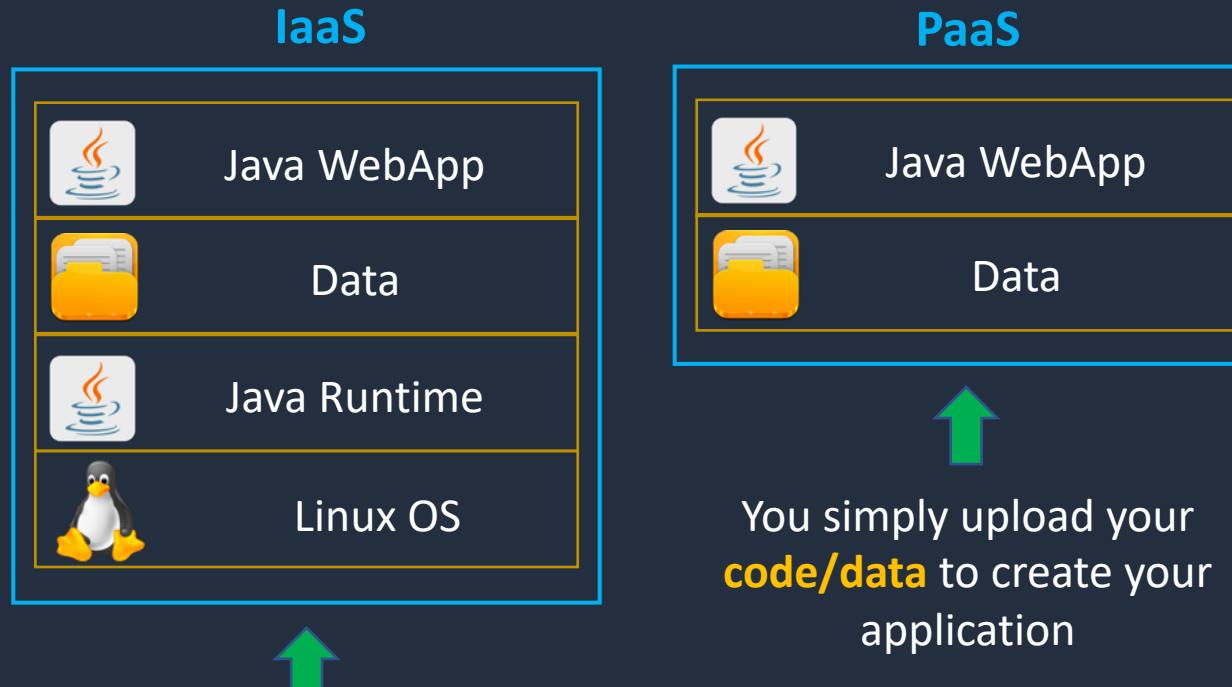
Platform as a Service with AWS Elastic Beanstalk





Cloud Service Models: Comparison

Example is
Amazon EC2

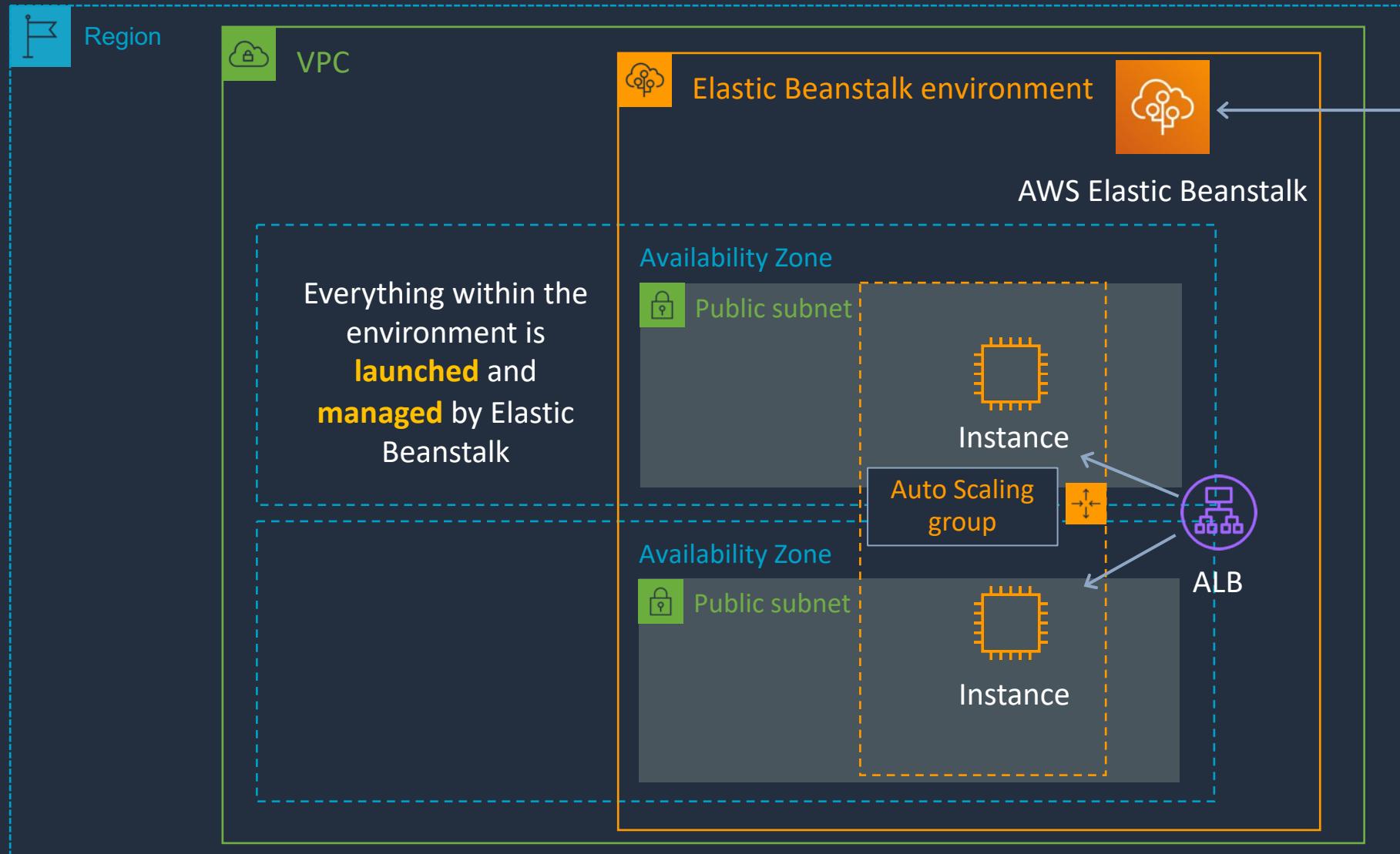


Example is **AWS Elastic Beanstalk**

You simply upload your
code/data to create your
application



AWS Elastic Beanstalk



Upload **source code** in ZIP file

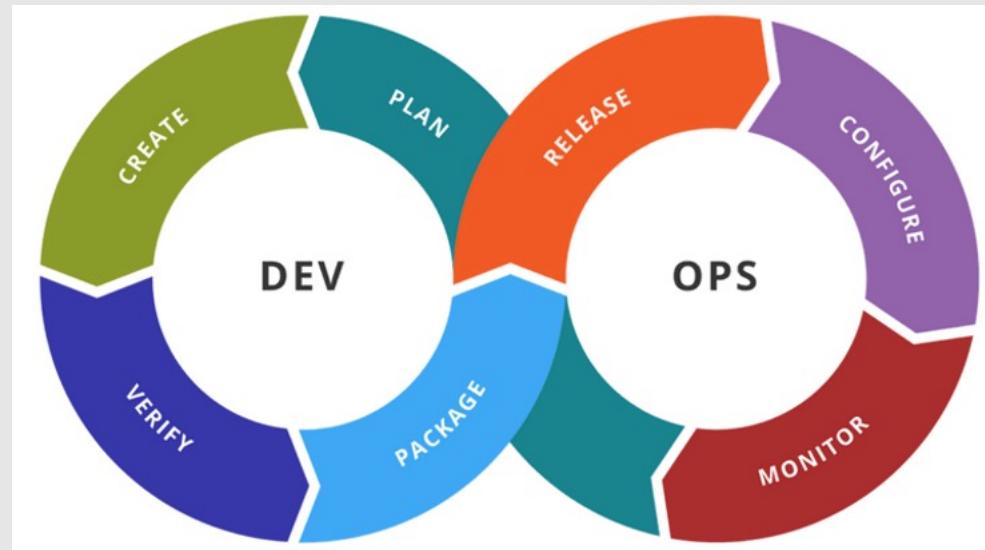


Developer Client

Create an Elastic Beanstalk Application

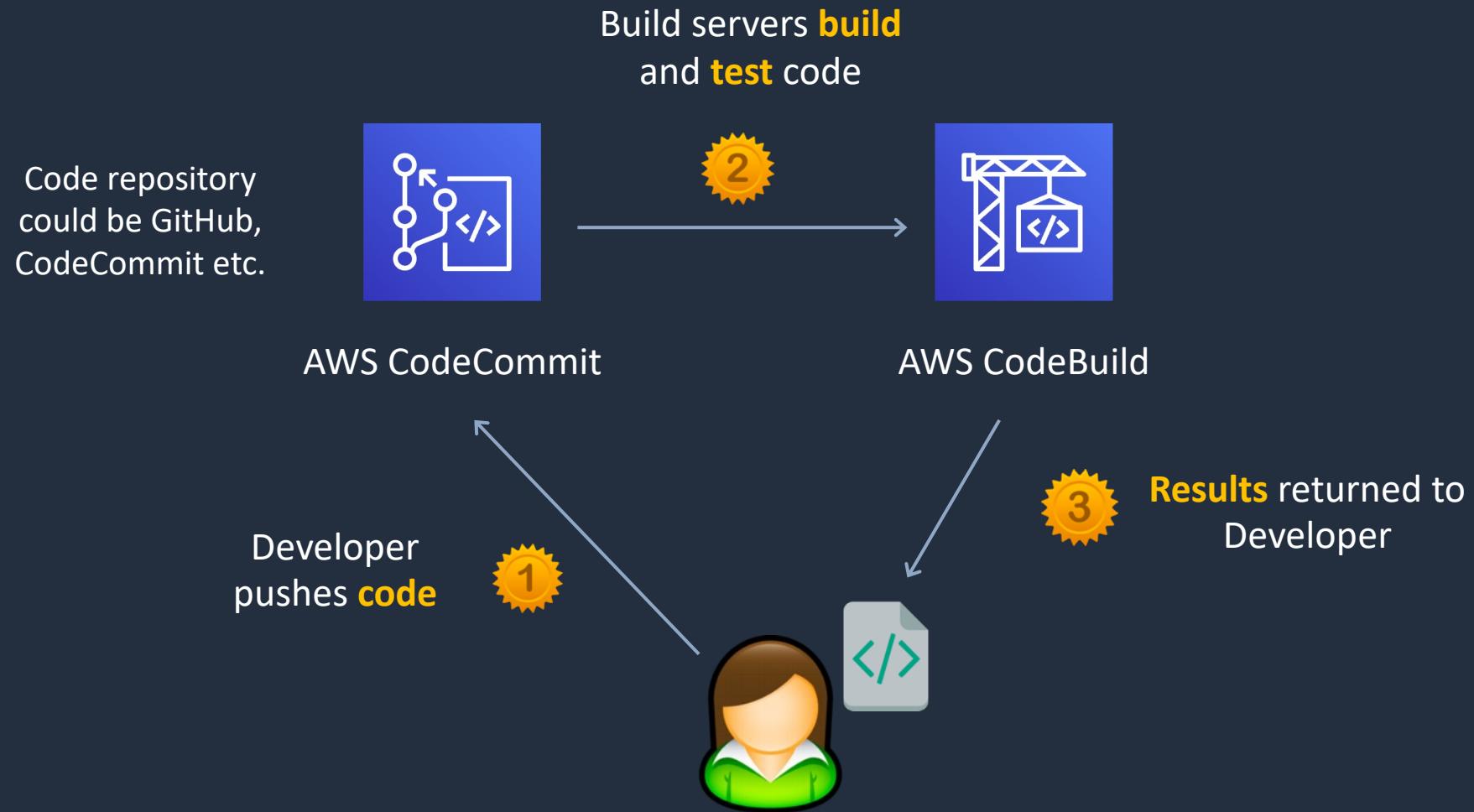


Continuous Integration and Continuous Delivery (CI/CD)



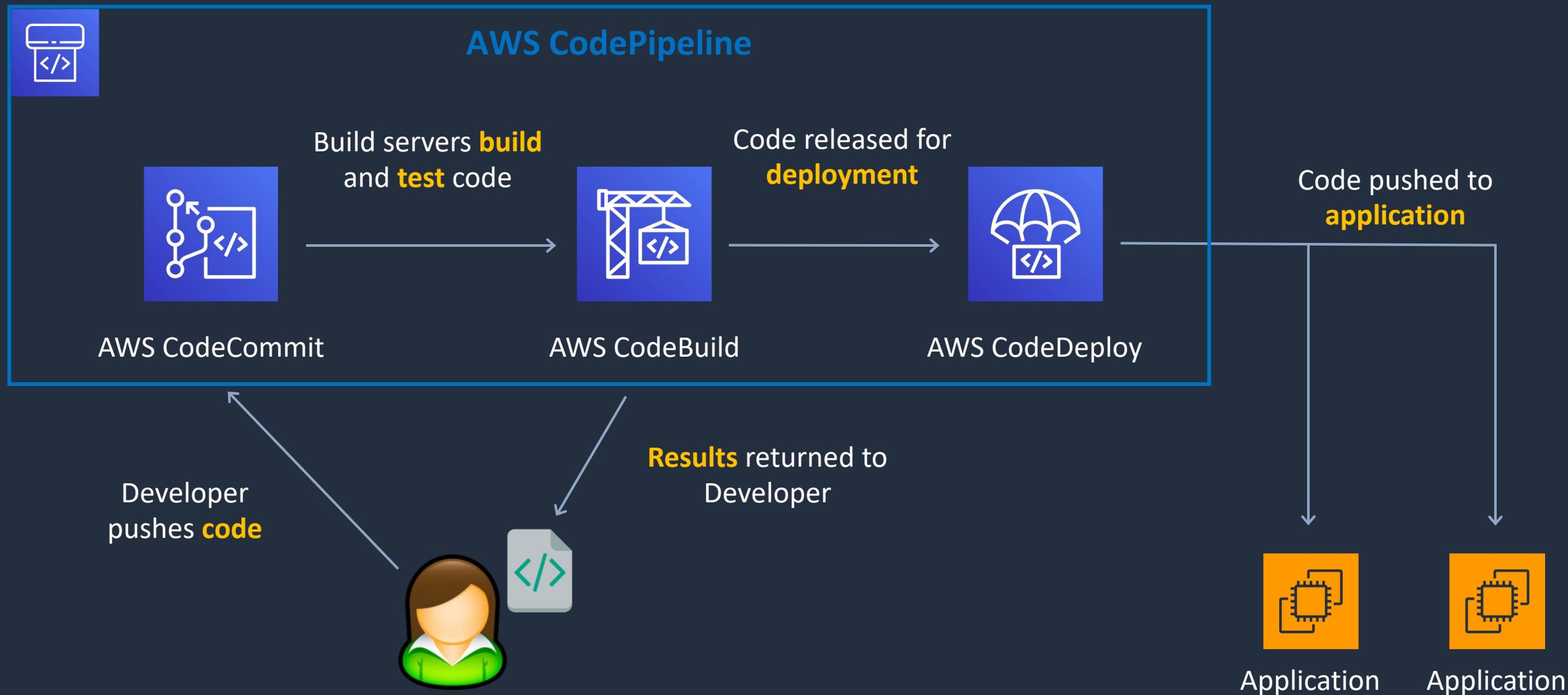


Continuous Integration





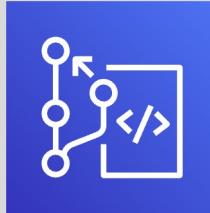
Continuous Integration & Continuous Delivery





Continuous Integration and Continuous Delivery

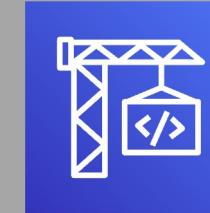
CODE



AWS CodeCommit

BUILD & TEST

AWS CodePipeline



AWS CodeBuild

DEPLOY



AWS CodeDeploy



Create an AWS CodeCommit Repository

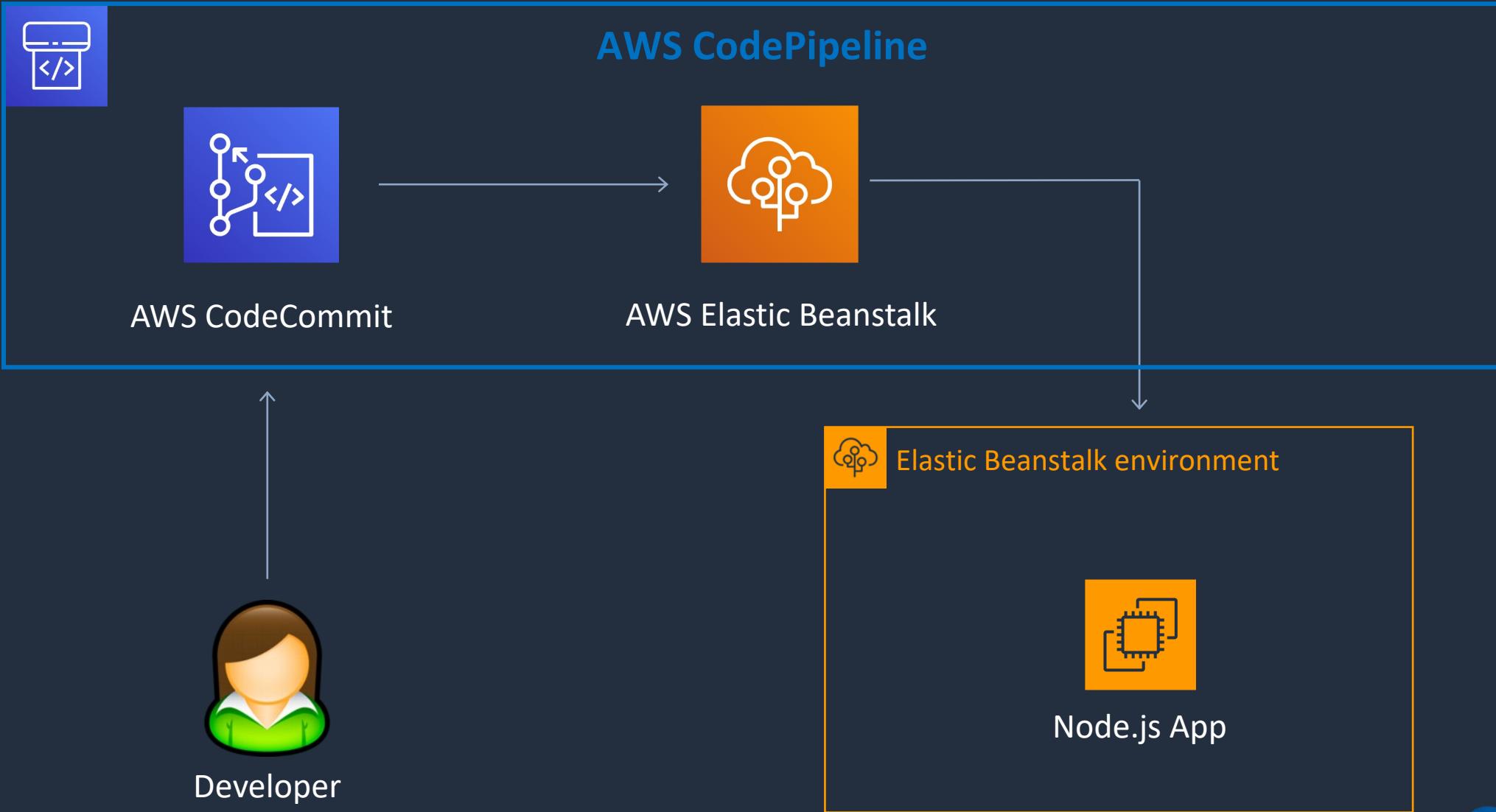


AWS CodePipeline with AWS Elastic Beanstalk





AWS CodePipeline with Elastic Beanstalk



SECTION 10

DNS, Caching, and Performance Optimization

Amazon Route 53 DNS

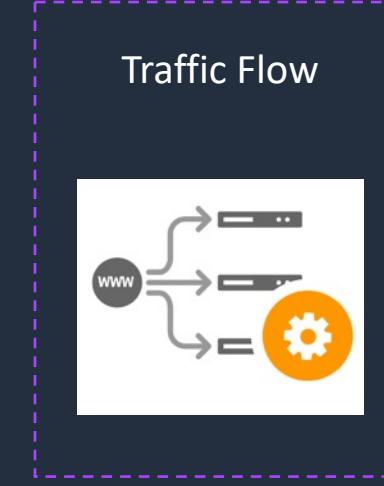
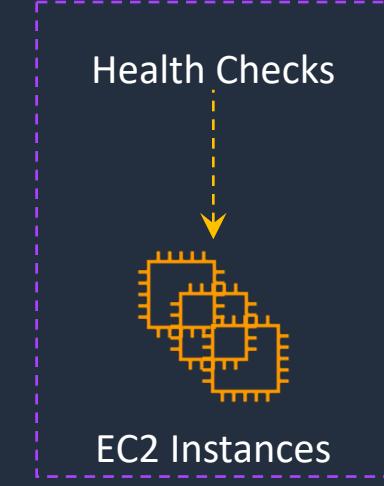




Amazon Route 53

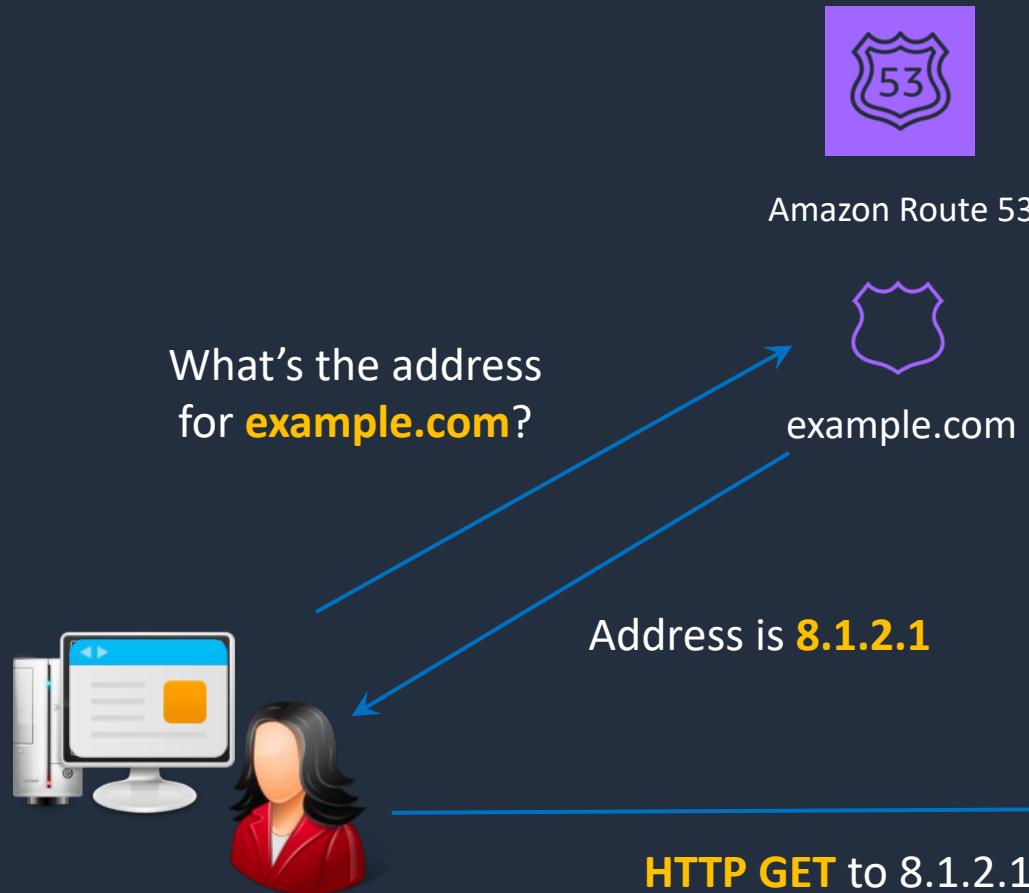


Amazon Route 53

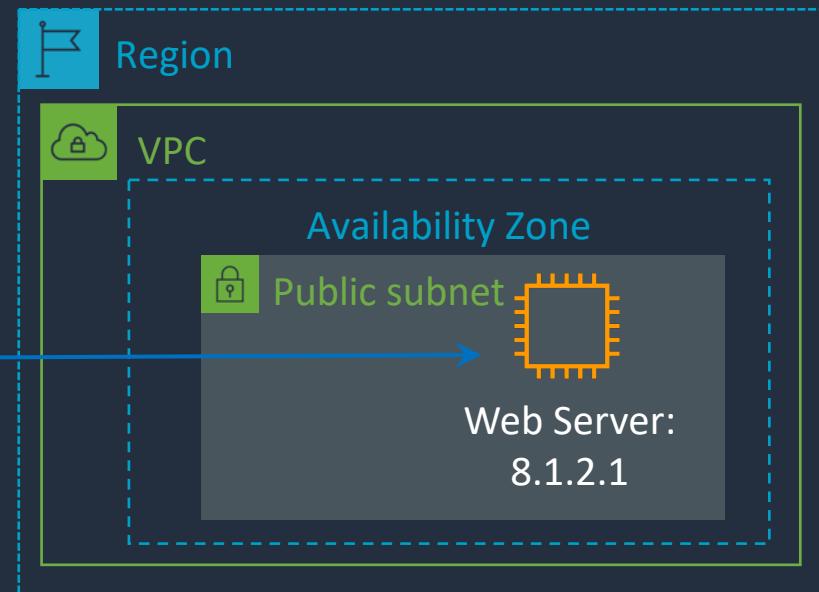




DNS Resolution with Route 53



A **hosted zone** represents a set of records belonging to a domain





Route 53 DNS Record Types

Routing Policy	What it does
Simple	Simple DNS response providing the IP address associated with a name
Failover	If primary is down (based on health checks), routes to secondary destination
Geolocation	Uses geographic location you're in (e.g. Europe) to route you to the closest region
Geoproximity	Routes you to the closest region within a geographic area
Latency	Directs you based on the lowest latency route to resources
Multivalue answer	Returns several IP addresses and functions as a basic load balancer
Weighted	Uses the relative weights assigned to resources to determine which to route to

Register a Domain using Route 53



Amazon CloudFront





Amazon CloudFront



Create an Amazon CloudFront Distribution

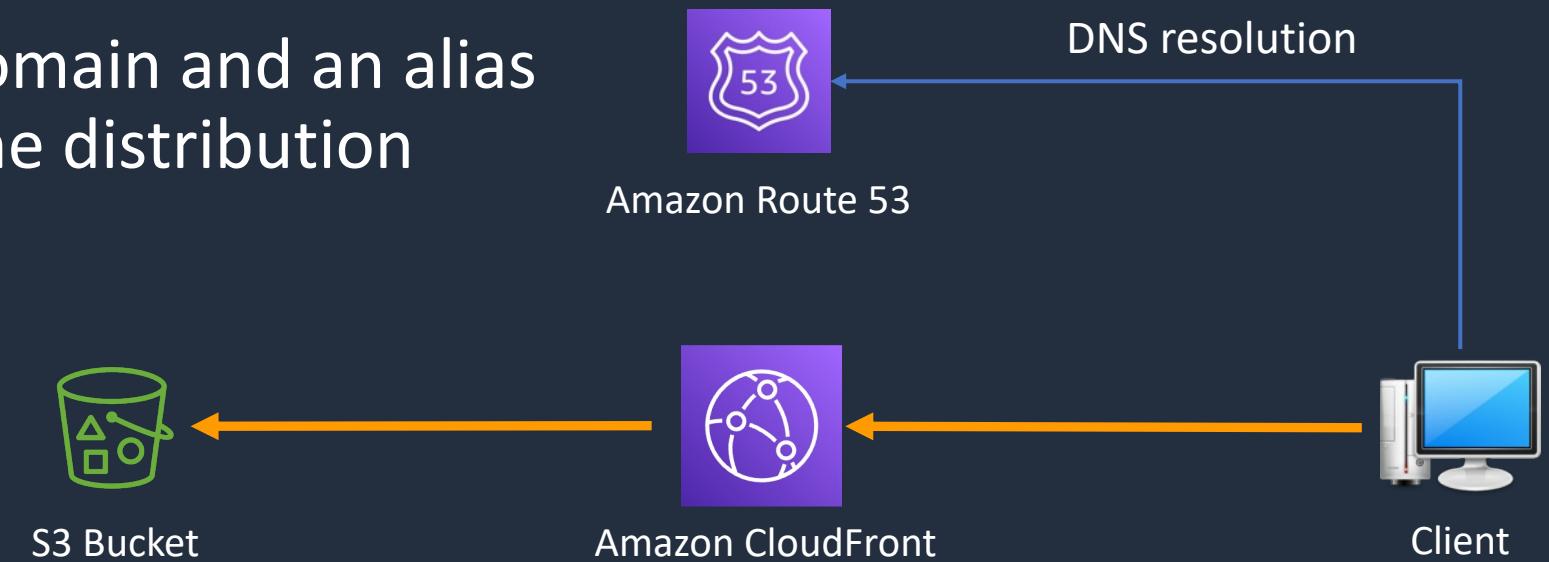




Amazon CloudFront Distribution

We will create:

- Amazon S3 bucket
- AWS Certificate Manager SSL/TLS certificate
- Amazon CloudFront distribution
- Amazon Route 53 domain and an alias record pointing to the distribution



AWS Global Accelerator



AWS Global Accelerator



Users in US

Resolve dctlabs.com



Answer:
51.45.2.12
53.58.31.89

Amazon Route 53



Edge location



User traffic ingresses using
the closest **Edge Location**

Addresses:
51.45.2.12
53.58.31.89

Static **anycast**
IP addresses

Requests are
routed to the
optimal endpoint

AWS Global Network

Users are **redirected**
to another **endpoint**



us-east-1



Traffic traverses
the **AWS global
network**



ap-southeast-2

SECTION 11

Containers and Serverless Computing

Docker Containers on Amazon ECS





Amazon ECS

ECS **Services** are used to maintain a **desired count** of tasks

An ECS **Task** is created from a **Task Definition**

Task Definition

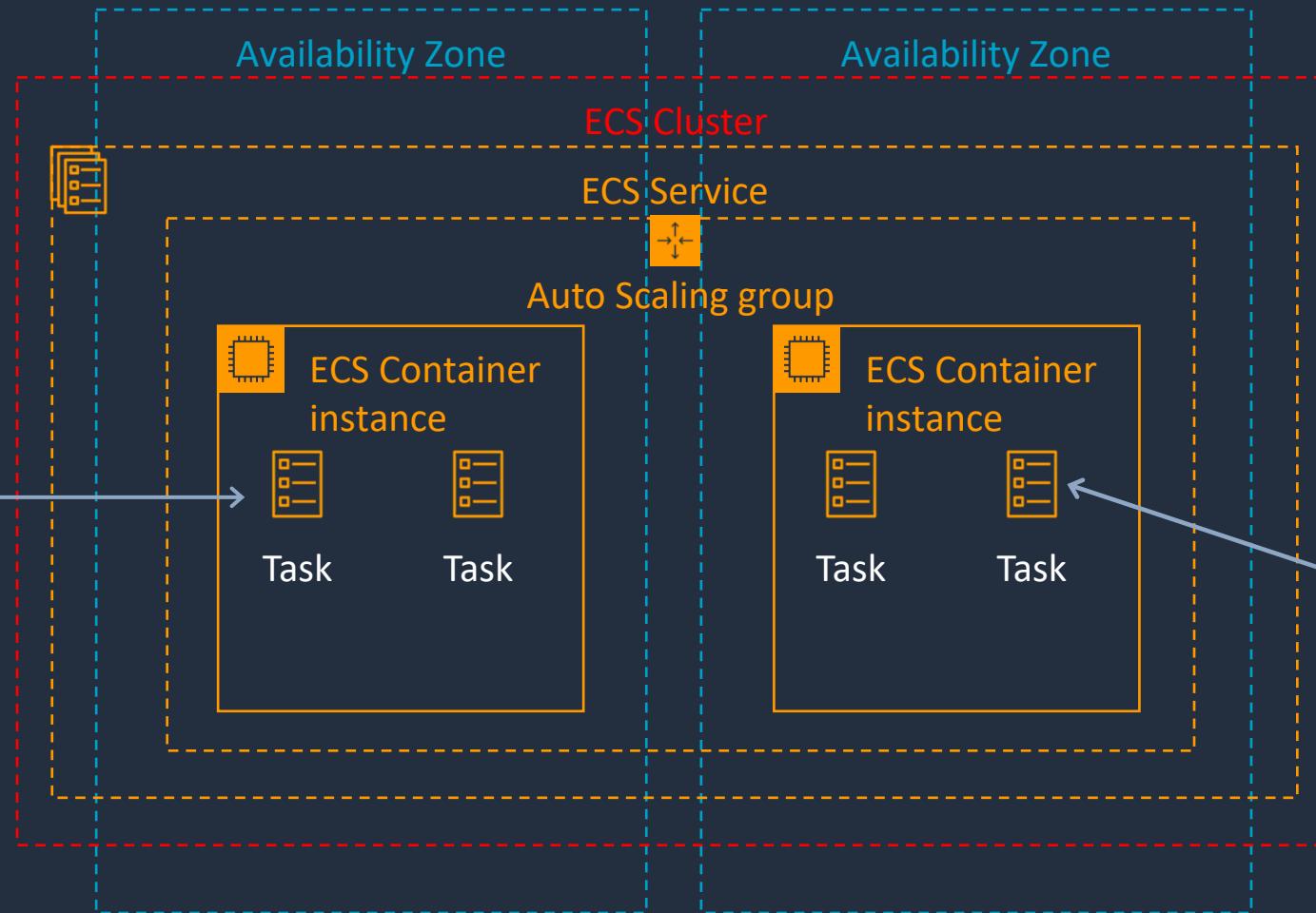
```
{
  "containerDefinitions": [
    {
      "name": "wordpress",
      "links": [
        "mysql"
      ],
      "image": "wordpress",
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "memory": 500,
      "cpu": 10
    }
  ]
}
```

An ECS **Task** is a running Docker container



Amazon Elastic Container Service

An Amazon **ECS Cluster** is a logical grouping of **tasks or services**



Docker **images** can be stored in **Amazon ECR**



Amazon Elastic Container Registry





Amazon ECS Components

Elastic Container Service (ECS)	Description
Cluster	Logical grouping of tasks or services
Container instance	EC2 instance running the the ECS agent
Task Definition	Blueprint that describes how a docker container should launch
Task	A running container using settings in a Task Definition
Service	Defines long running tasks – can control task count with Auto Scaling and attach an ELB

Create an AWS Fargate Cluster

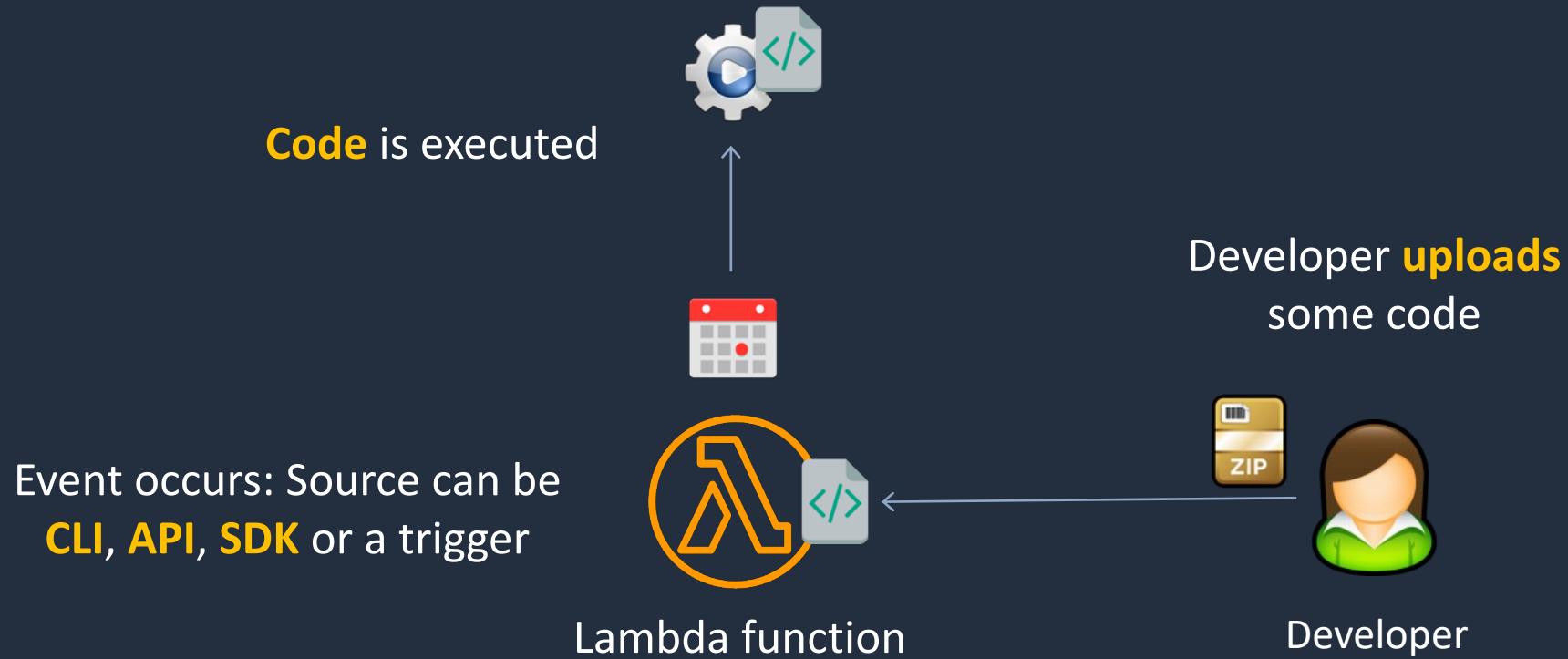


Serverless with AWS Lambda





AWS Lambda



Create an AWS Lambda Function

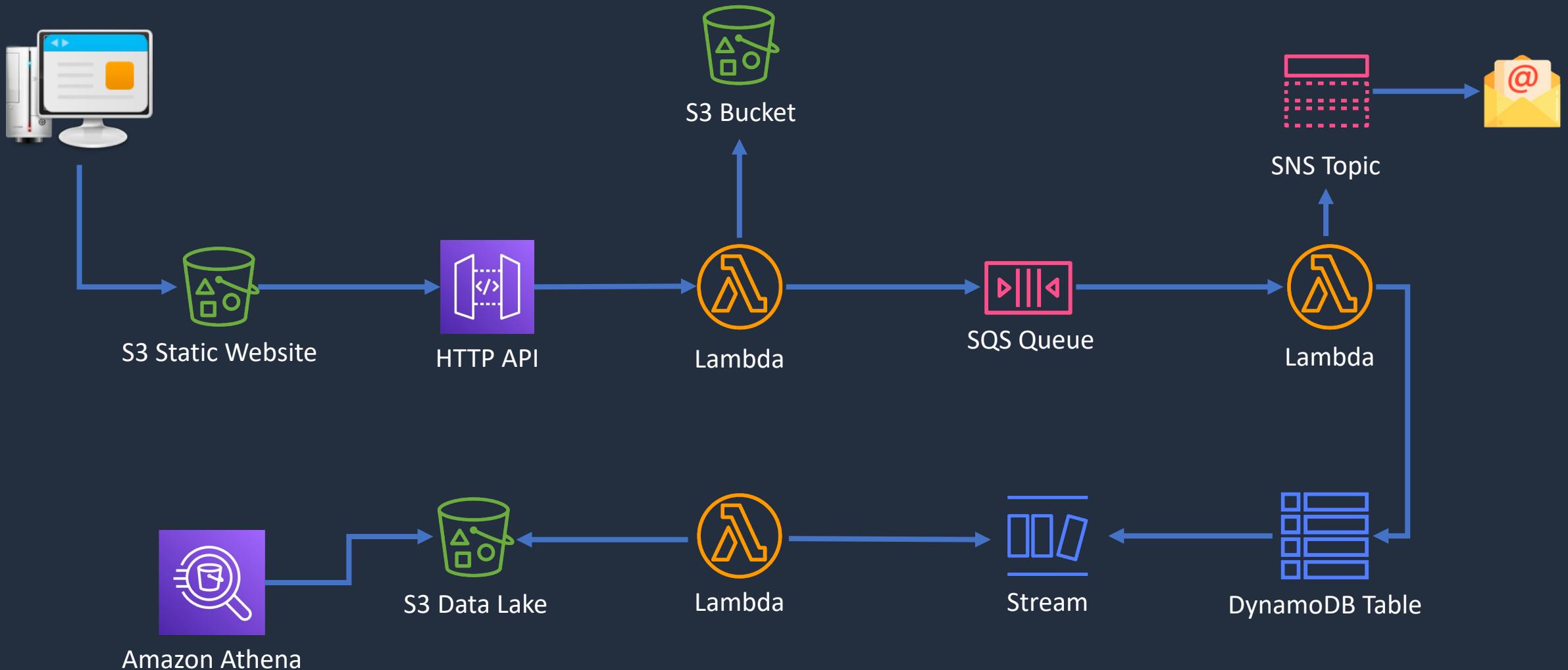


Application Integration Services



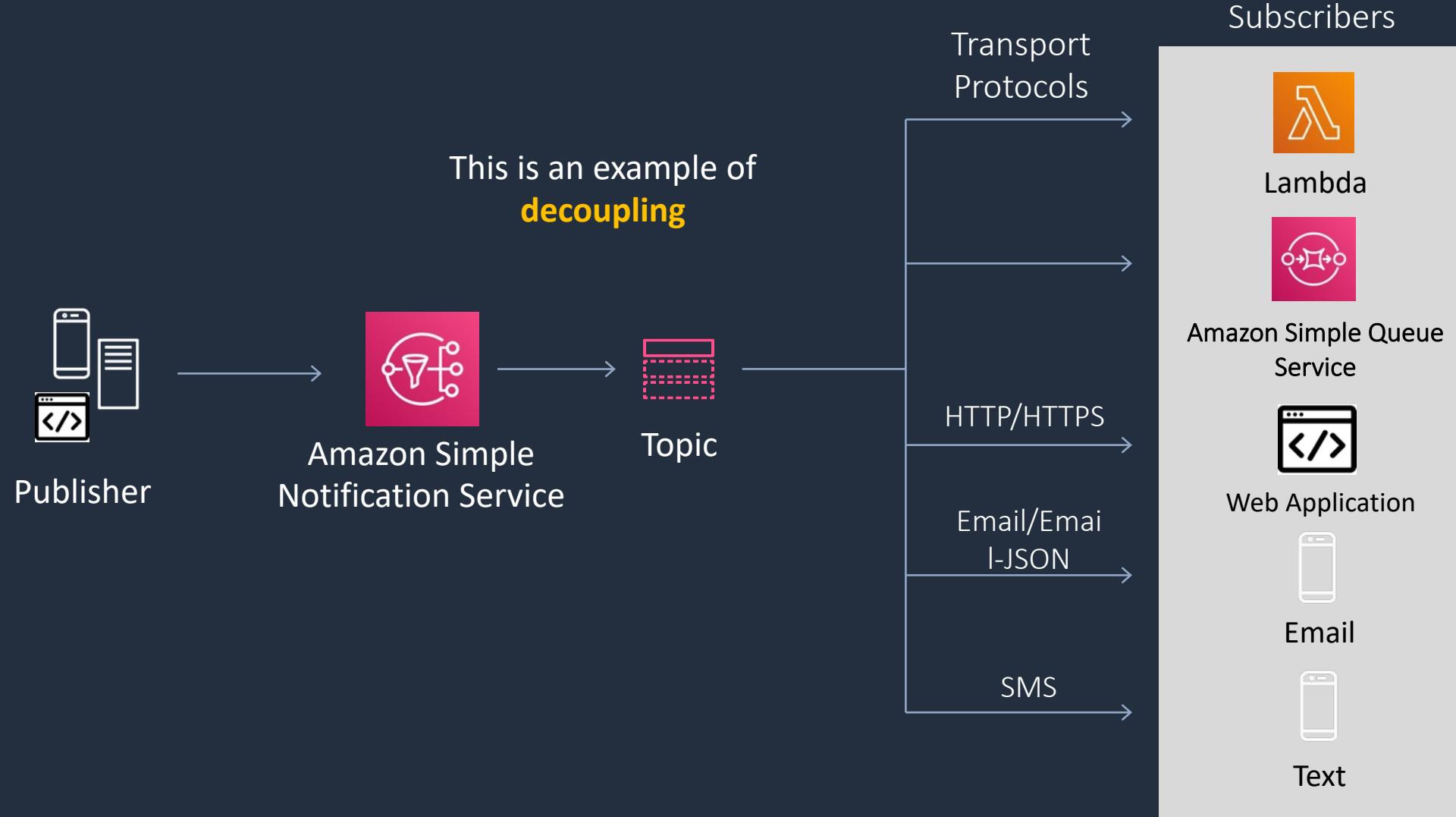


Serverless Services and Event-Driven Architecture



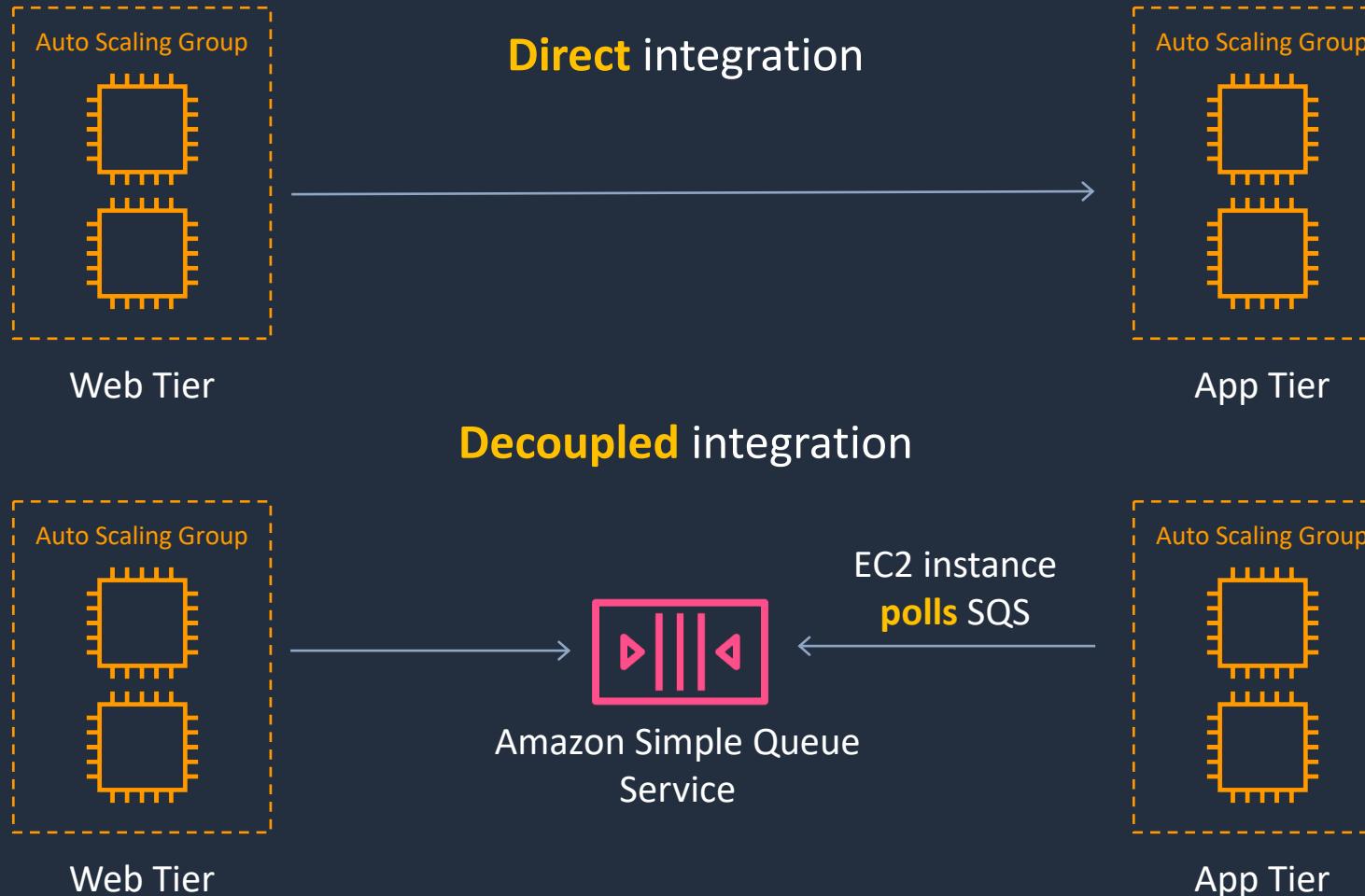


Amazon Simple Notification Service (SNS)





Amazon Simple Queue Service (SQS)

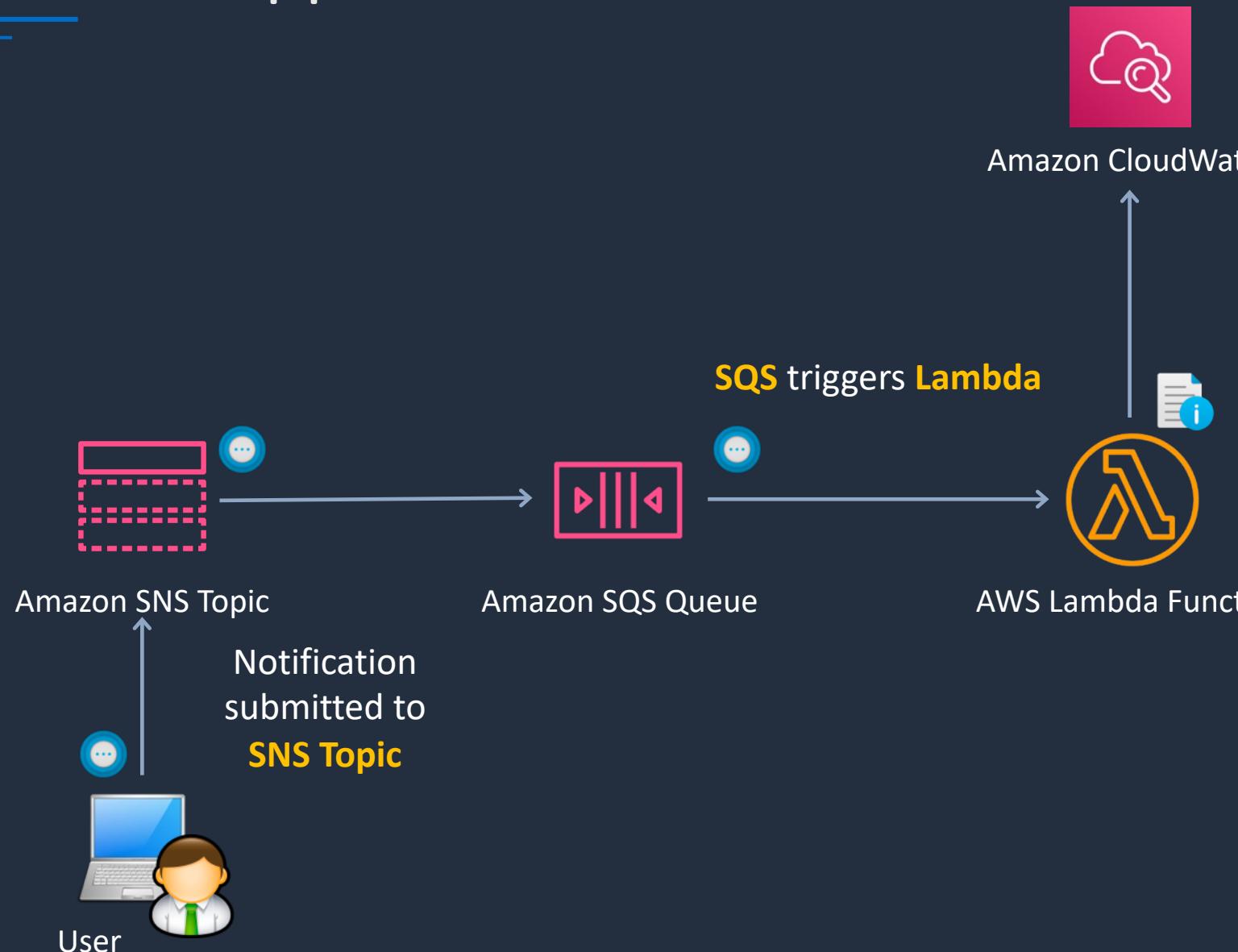


Create a Serverless Application





Serverless Application



Amazon EventBridge

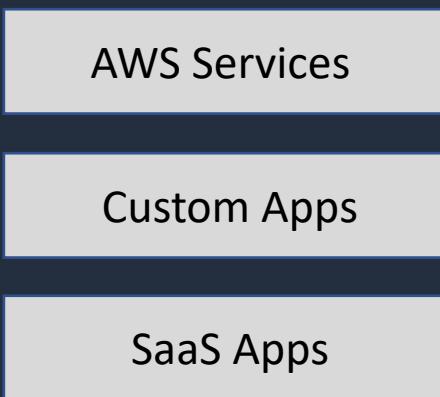




Amazon EventBridge

EventBridge used to be known as **CloudWatch Events**

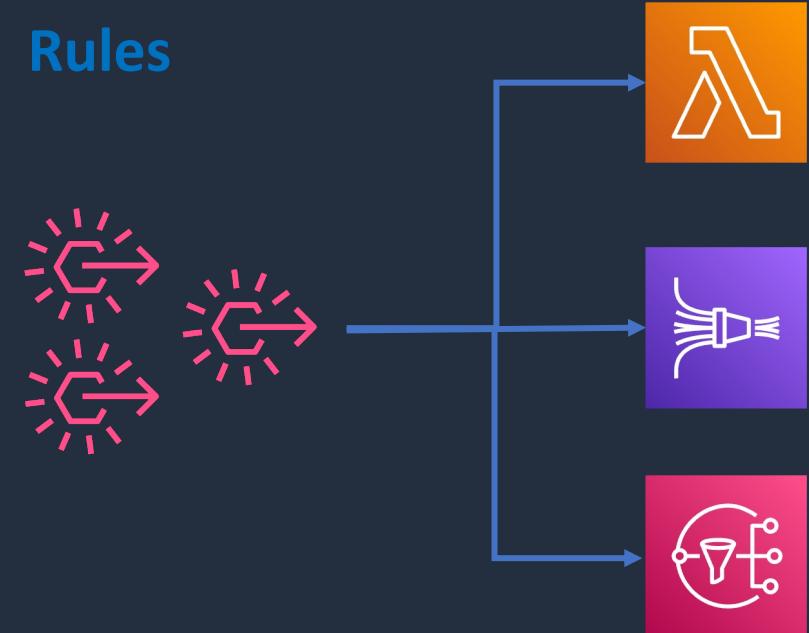
Event Sources



Events

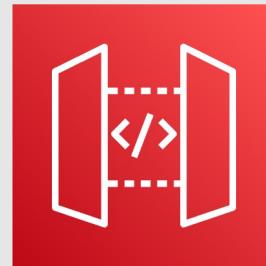
EventBridge
event bus

Rules



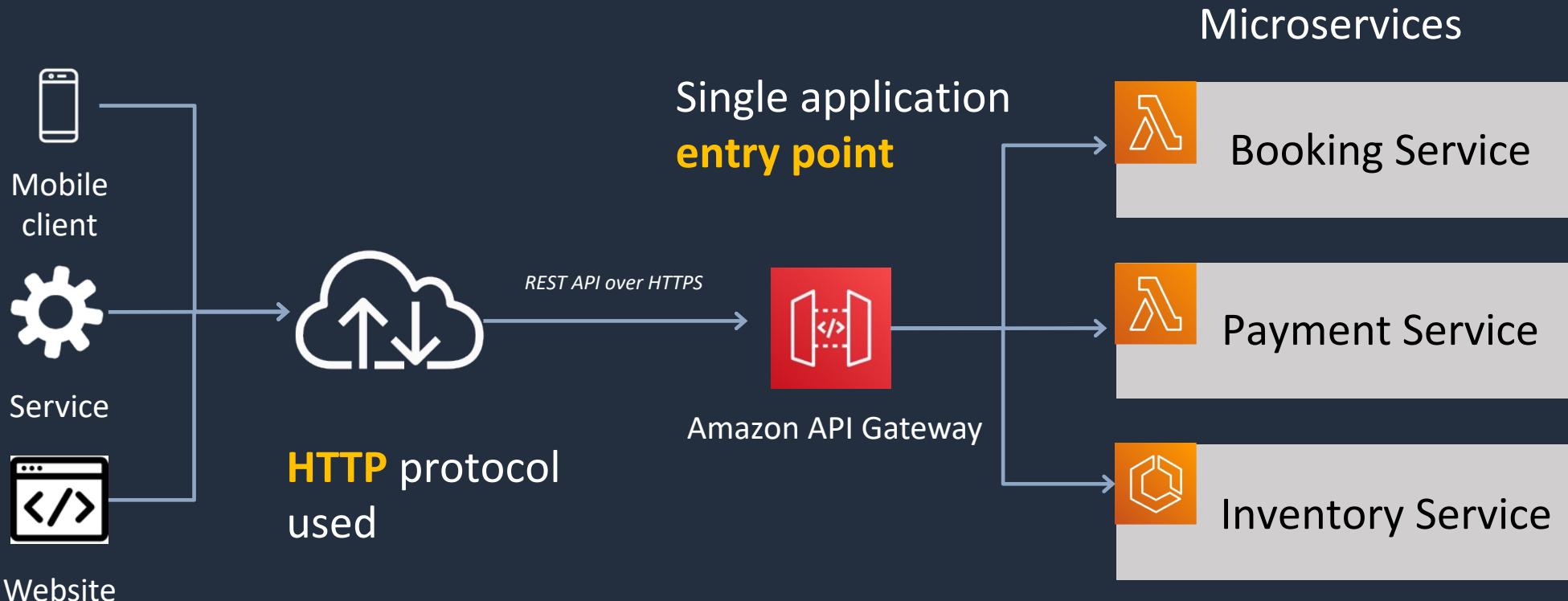
Targets

Amazon API Gateway

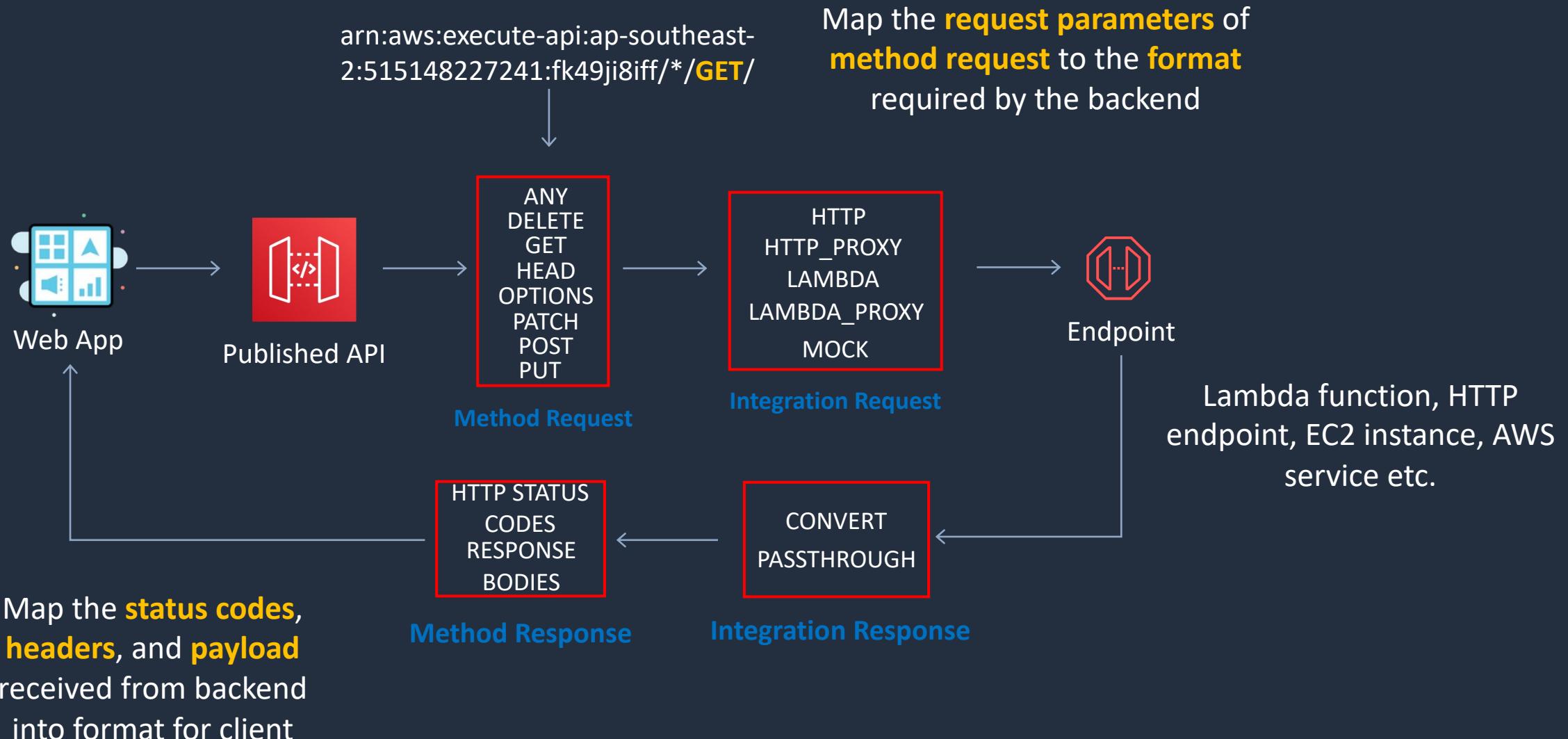




REST API with Amazon API Gateway



Structure of a REST API



Create a REST API with Lambda Proxy Integration



SECTION 13

Get Certified on AWS

Get Certified on AWS



Why work in cloud computing?

1. Job demand
2. Globally relevant skills
3. Rewarding career paths
4. Great salaries

Why get AWS certified?

1. Demonstrate skills to employers
2. Differentiate yourself
3. Gain knowledge
4. Develop practical skills



AWS Certification

New to the industry

FOUNDATIONAL

Six months of fundamental AWS Cloud and industry knowledge



ASSOCIATE

One year of experience solving problems and implementing solutions using the AWS Cloud



Some experience

PROFESSIONAL

Two years of experience designing, operating, and troubleshooting solutions using the AWS Cloud



Highly experienced

SPECIALTY

Technical AWS Cloud experience in the Specialty domain as specified in the exam guide



Specialized in an IT role

ON-DEMAND TRAINING COURSES



Prepare for your next AWS certification with video courses & practice exams

HANDS-ON CHALLENGE LABS



Build hands-on cloud skills in a secure sandbox environment

LIVE AWS BOOTCAMPS



Get ready for your next cloud job with real-world projects in a virtual classroom