Raj Saha

cloudwithraj.com

Cloud With Raj

linkedin.com/in/rajdeep-sa-at-aws/

## Instructor Bio:

Sr. Specialist SA – Containers/Serverless@aws

Bestselling Udemy/Pluralsight author

Tech Advisor of crypto startup

Public speaker and guest lecturer

Author of multiple official AWS blogs

YouTuber with 54K subscribers

Previously - Distinguished Cloud Architect @Verizon

*Opinions are my own*

# AWS Learning Path for Beginners

Raj Saha
cloudwithraj.com
Cloud With Raj

# No Comp Science Degree?

- I don’t have a Comp Sci degree
- Majority of the folks I work with in Cloud does NOT have comp sci degree
- Your one year+ salary is NOT impacted by your degree
  - Based on your performance

# Freshers Learning Path

AWS Concepts and Demos

↓

Hands on Projects in Popular Areas

↓

Stellar Resume and LinkedIn

↓

Interview Q/A Practice

Must Have

---

Git/GitHub

↓

Basic Linux

Highly Recommended

---

**Cloud Job**

# Newcomer Learning Path

AWS Concepts and Demos

Hands on Projects in Popular Areas

Stellar Resume and LinkedIn

Interview Q/A Practice

Git/GitHub

Basic Linux

**Must Have**

DevOps

Kubernetes

System Design

**Highly Recommended**

**Cloud Job**

# How About Certs and Coding?

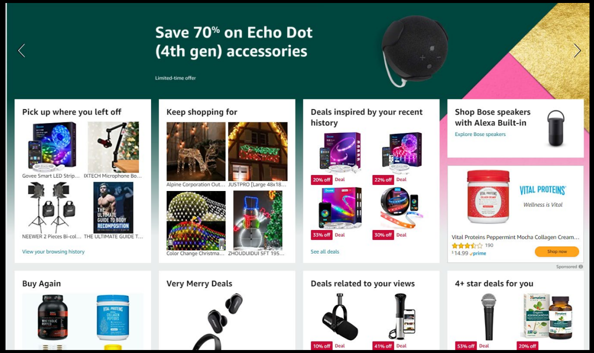
- Certs are good to get recruiter attention
  - AWS Associate Certs
  - Kubernetes Certs
  - Avoid super specific certs in the beginning

- Coding NOT a primary requirement
  - Can be picked up at job
  - No coding round in Cloud DevOps interview
  - Learn Python for greater scope

# Frontend Vs Backend – Role of AWS and What Should YOU Choose?

Raj Saha
cloudwithraj.com
Cloud With Raj

# Frontend Vs Backend




Shopping cart
Discount
Payment

```python
dynamodb = boto3.resource("dynamodb") # pylint: disable=invalid-name
table = dynamodb.Table(TABLE_NAME) # pylint: disable=invalid-name,no-member
logger = Logger() # pylint: disable=invalid-name
tracer = Tracer() # pylint: disable=invalid-name
metrics = Metrics(namespace="ecommerce.delivery", service="delivery")


@tracer.capture_method
def get_order(order_id: str) -> Optional[dict]:
    """
    Retrieve the order from ther Orders service
    """

    logger.debug({
        "message": "Retreiving order {}".format(order_id),
        "orderId": order_id
    })
```

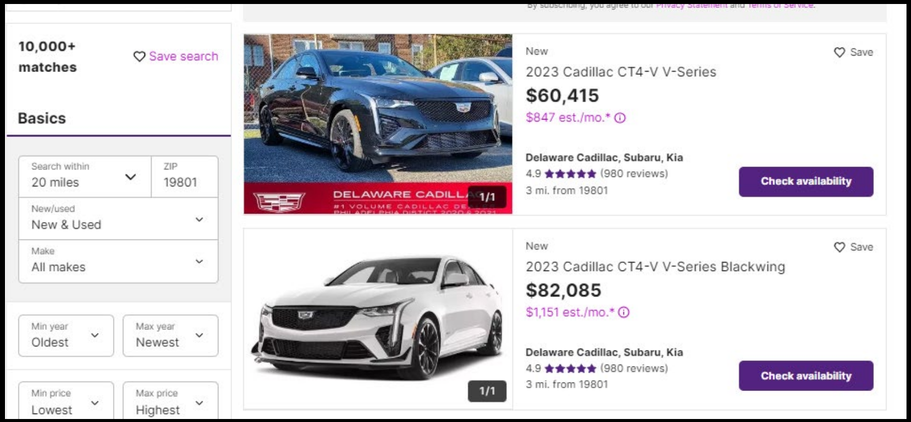
| CartID | PersonLogin | ProductID |
|---|---|---|
| 10000 | John.Wilson | 100 |
| 10000 | John.Wilson | 200 |
| 20000 | Tina.Smith | 300 |

Frontend (User Interface)

Backend (Business Logic)

Databases

# Frontend Vs Backend




Fetch car details based on user selection

```
dynamodb = boto3.resource("dynamodb") # pylint: disable=invalid-name
table = dynamodb.Table(TABLE_NAME) # pylint: disable=invalid-name,no-member
logger = Logger() # pylint: disable=invalid-name
tracer = Tracer() # pylint: disable=invalid-name
metrics = Metrics(namespace="ecommerce.delivery", service="delivery")


@tracer.capture_method
def get_order(order_id: str) -> Optional[dict]:
    """
    Retrieve the order from ther Orders service
    """

    logger.debug({
        "message": "Retreiving order {}".format(order_id),
        "orderId": order_id
    })
```

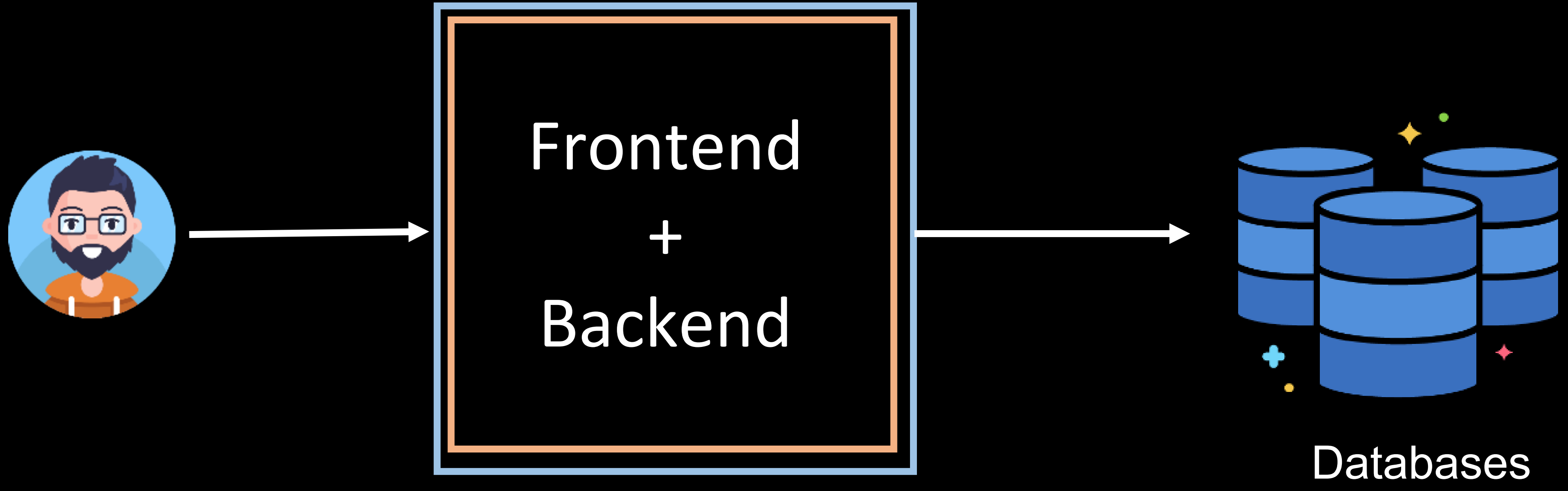
Frontend (User Interface)

Backend (Business Logic)

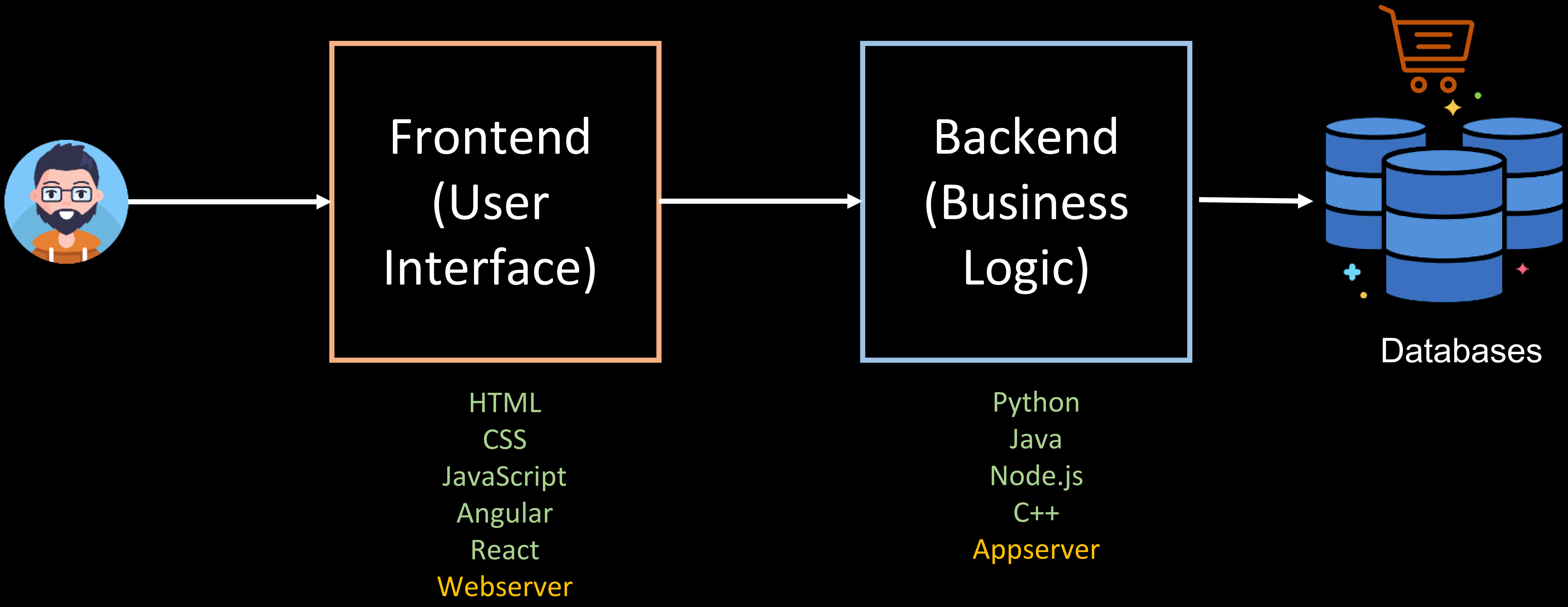
All car details

Databases

# Why Separate Frontend Vs Backend?



# Frontend Vs Backend



## Frontend

- HTML, CSS, JavaScript, Angular, React and more
- How things look and feel (Images, colors, style)
- Shorter feedback loop (Change and re-test)
- How much money?
  - Avg $105K/Year (indeed.com salary data)
  - More freelancing opportunity for app development
  - Ideal for visual person

## Backend

- Python, Java, Node.JS, C++ and more
- Business logic and interaction with core database
- Longer feedback loop (Simulate specific logic)
- How much money?
  - Avg $121K/Year (indeed.com salary data)
  - Data structure and algorithm, database interaction, analytics
  - Taught in Comp Sci (No formal degree required for job)

# How It Started – Data Center

Raj Saha
cloudwithraj.com
Cloud With Raj

# Modern World

# Modern World

- Computation requires CPU, Memory, and Storage
- In olden days – everything is done locally
    - Think of Netflix CDs, game CDs etc.

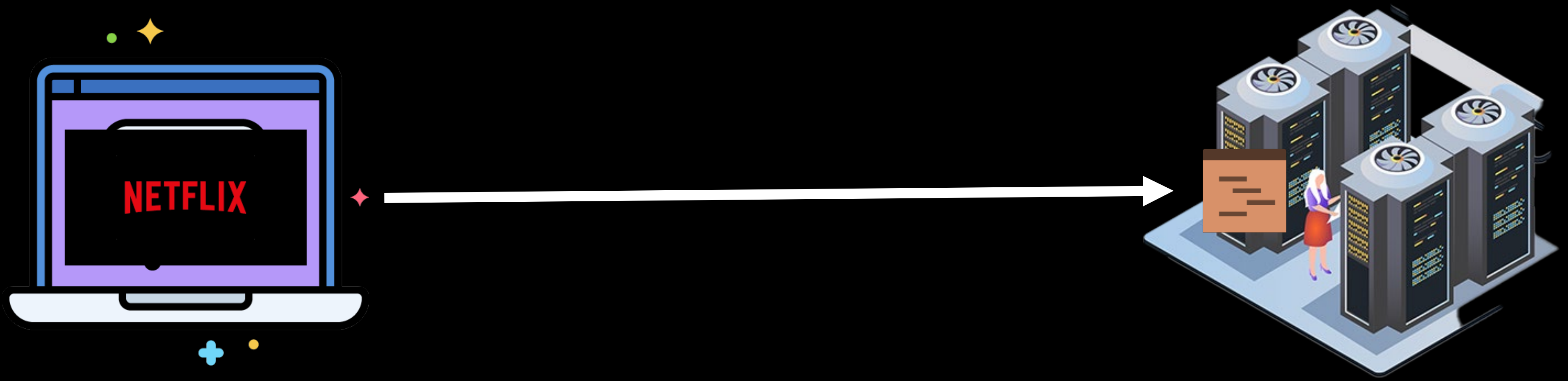
# Challenges with Local Computing


- Not every device is same
- Some applications requires lot of local resources
  - Can you save all the movies in your laptop?
- Updating local software is challenging
- Other use cases from enterprise perspective
  - Saving exa-bytes of data
  - Running high performance computing
  - Running analytics
  - and much more

# High Scale Applications


- You are accessing the application using a browser
- The application code is not running on your computer
- The physical computer(server) is running in a data center

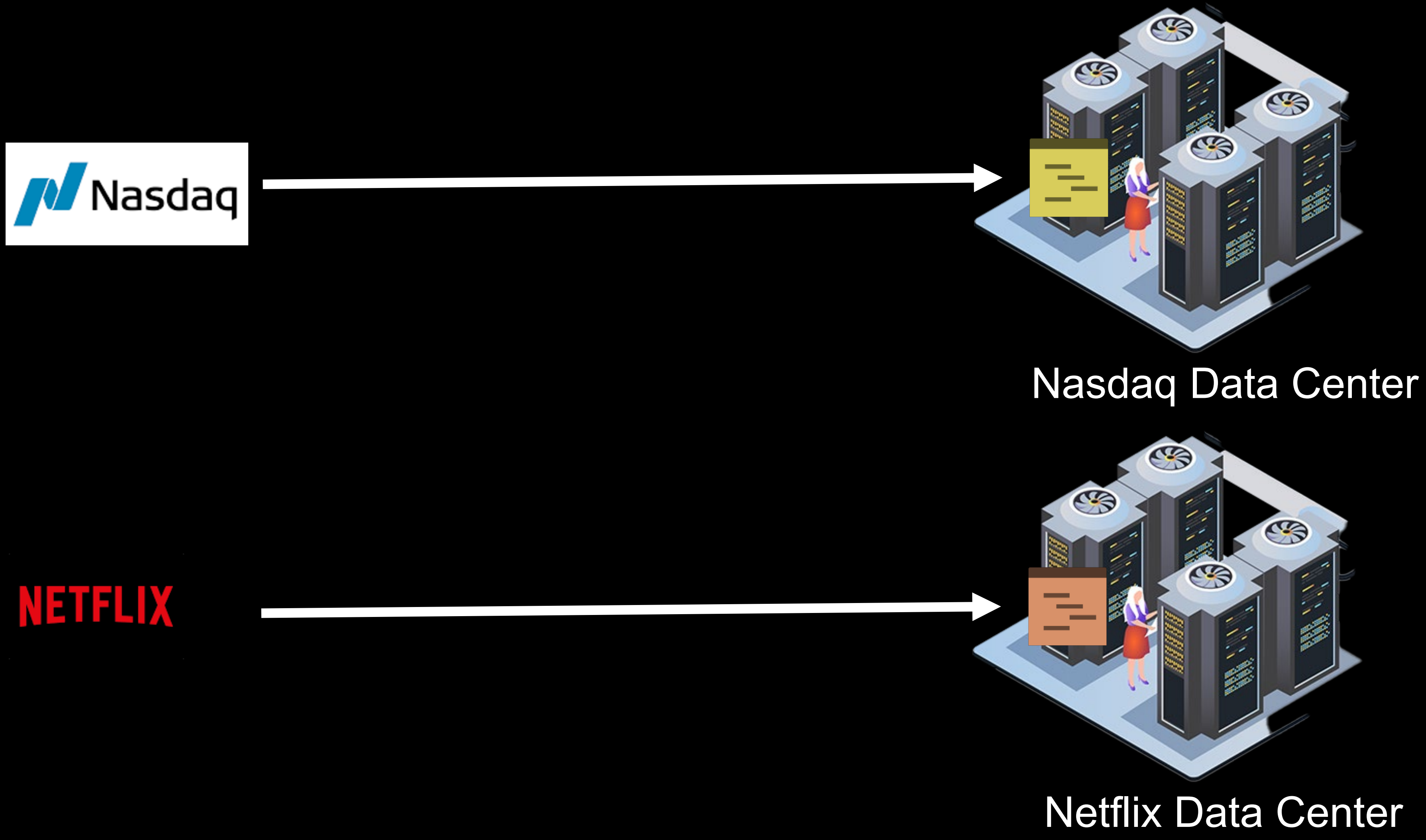
# High Scale Applications




Data Center

- Array of computers
- Known as servers

# Company Managed Data Center



# Data Center

Data Center

- Physical building
- Electricity, cooling, manpower

# Data Center Disadvantages


Raj Saha
cloudwithraj.com
Cloud With Raj

# Data Center

Data Center

- Physical building
- Electricity, cooling, manpower

# Data Center Challenges

Data Center

- Physical building
- Electricity, cooling, manpower

- Take months to procure building and servers
- Plan for max capacity
- Wasted capacity and money
- Additional overhead takes focus away from innovation
- Disaster in data center location

# Cloud Computing – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

# Company Managed Data Center

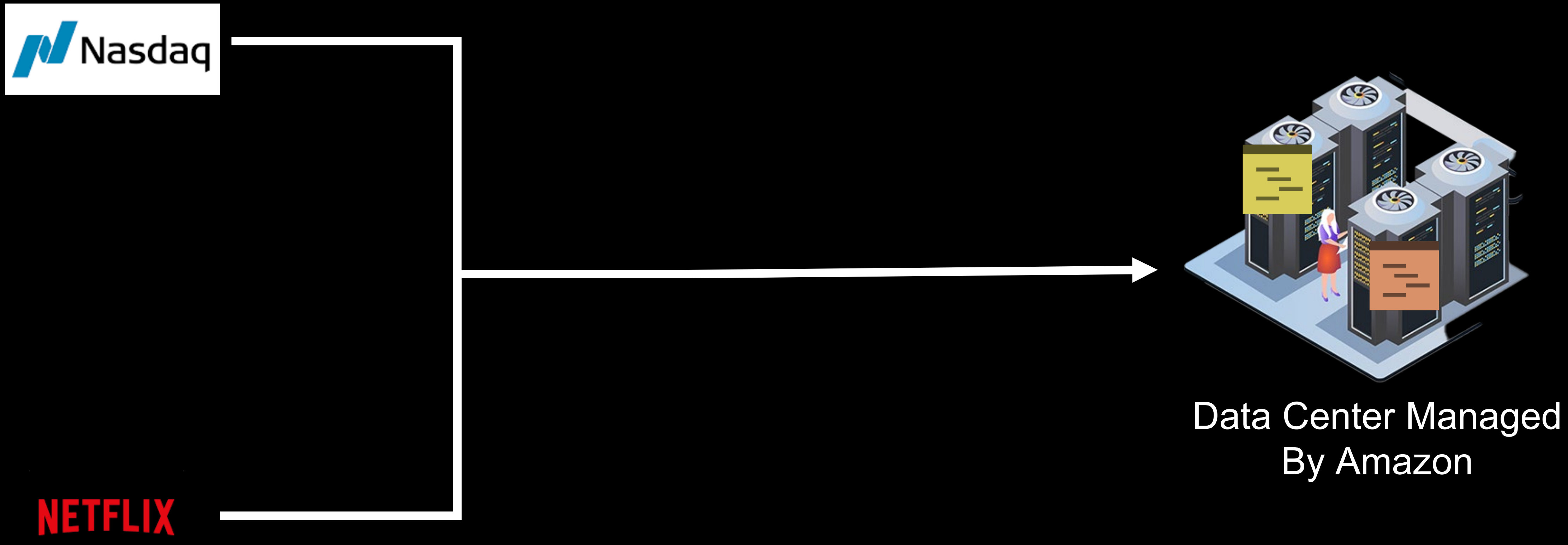
Nasdaq

Nasdaq Data Center

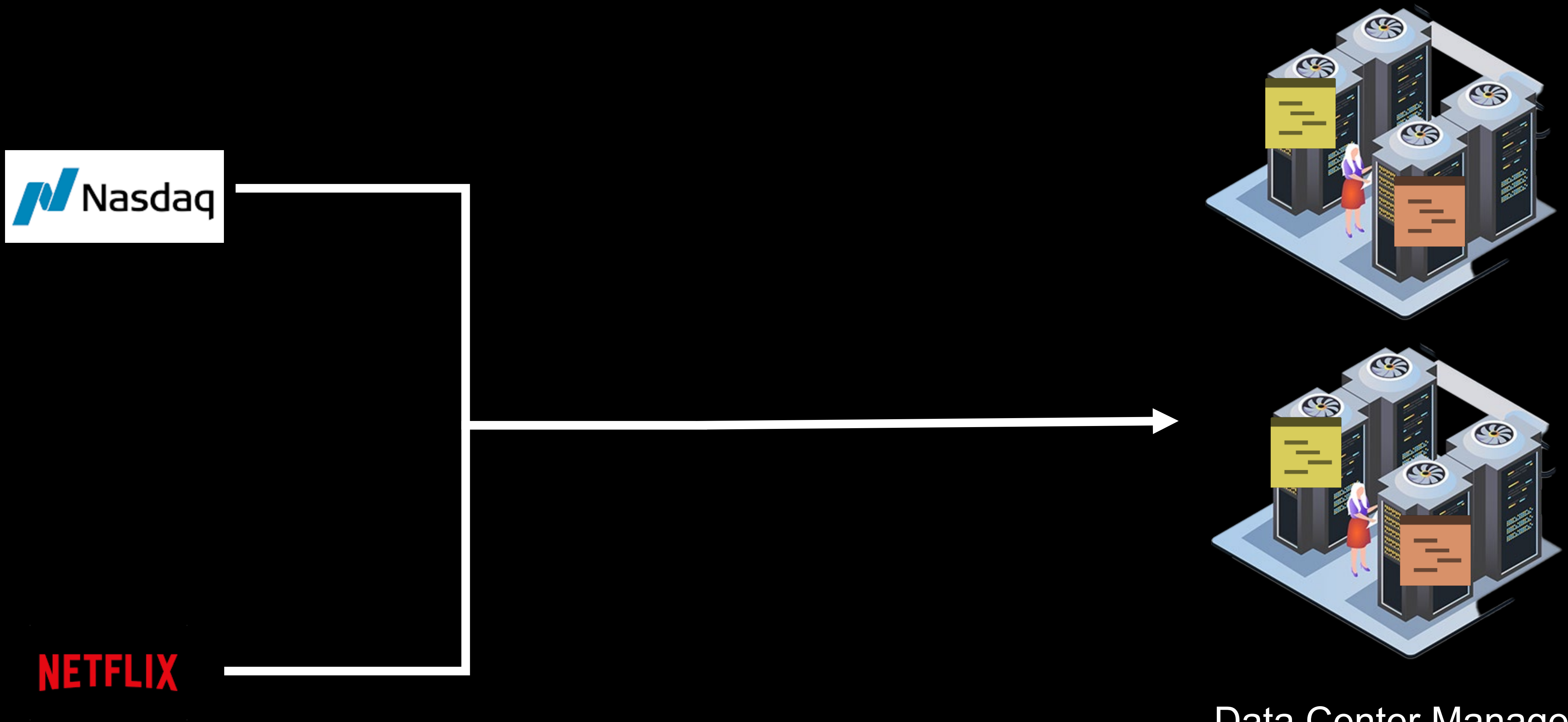
NETFLIX

Netflix Data Center

# Cloud Managed Data Center



# Cloud Managed Data Center




Data Center Managed
By Amazon

# Cloud Managed Data Center

aws

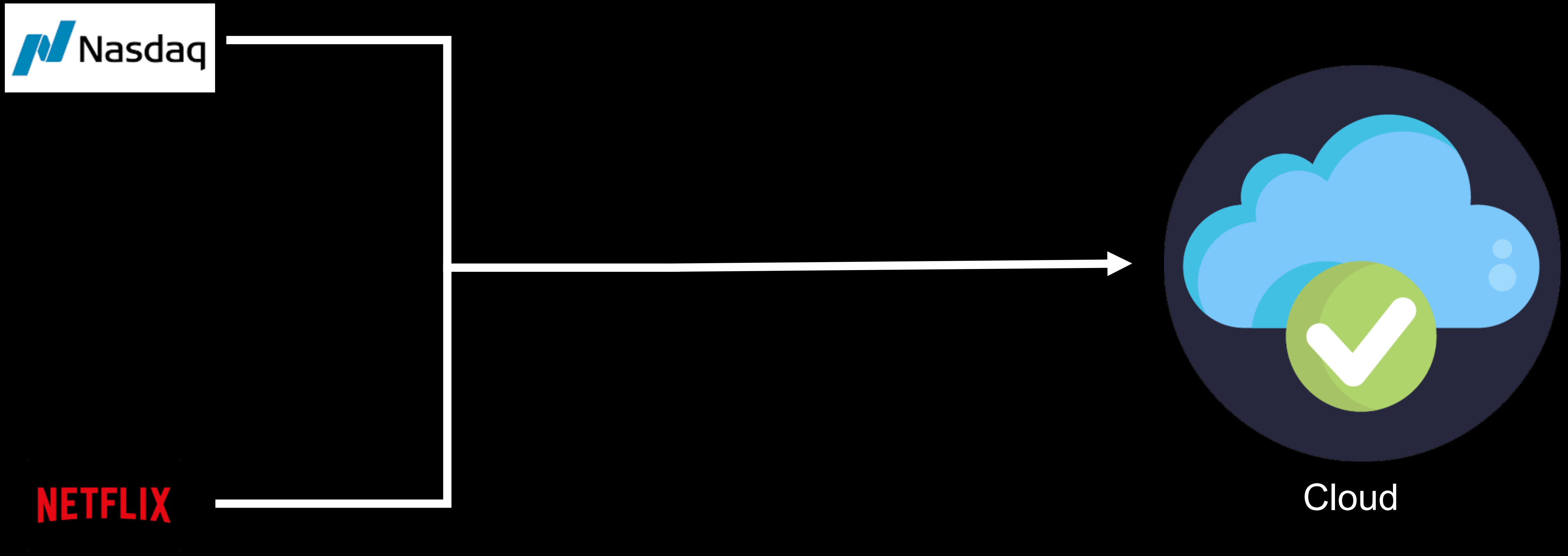
Azure

# Cloud Computing Definition

Memorize This:

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

# Cloud Managed Data Center



# Who is Using Cloud?


And thousands more…

# Cloud Computing Benefits

Raj Saha
cloudwithraj.com
Cloud With Raj

# Cloud Computing Benefits

- Agility
- Elasticity
- Cost savings
- Deploy globally in minutes

# Elastic

An elastic band stretches and shrinks based on need

# Elasticity

# Elasticity

NETFLIX

Big release

Get higher capacity

Normal time

Reduce capacity

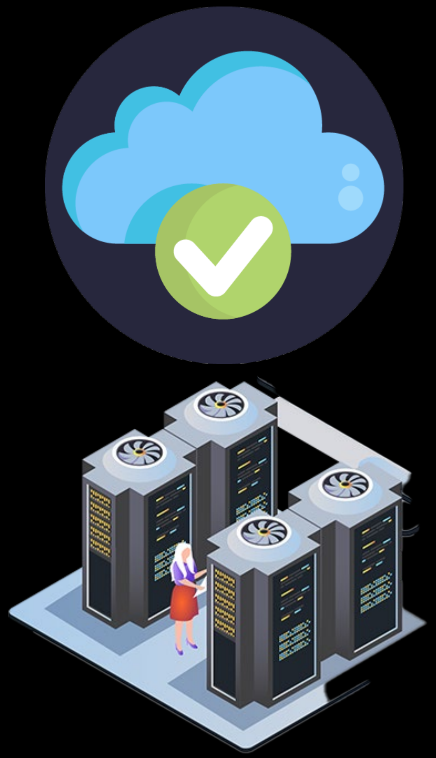
- Scale up(increase)/down(decrease)
- Scale out(increase)/in(decrease)

# Cloud Computing Deployment Models

Raj Saha
cloudwithraj.com
Cloud With Raj

# 3 Deployment Models

On-premises
(Self managed data center)



Hybrid

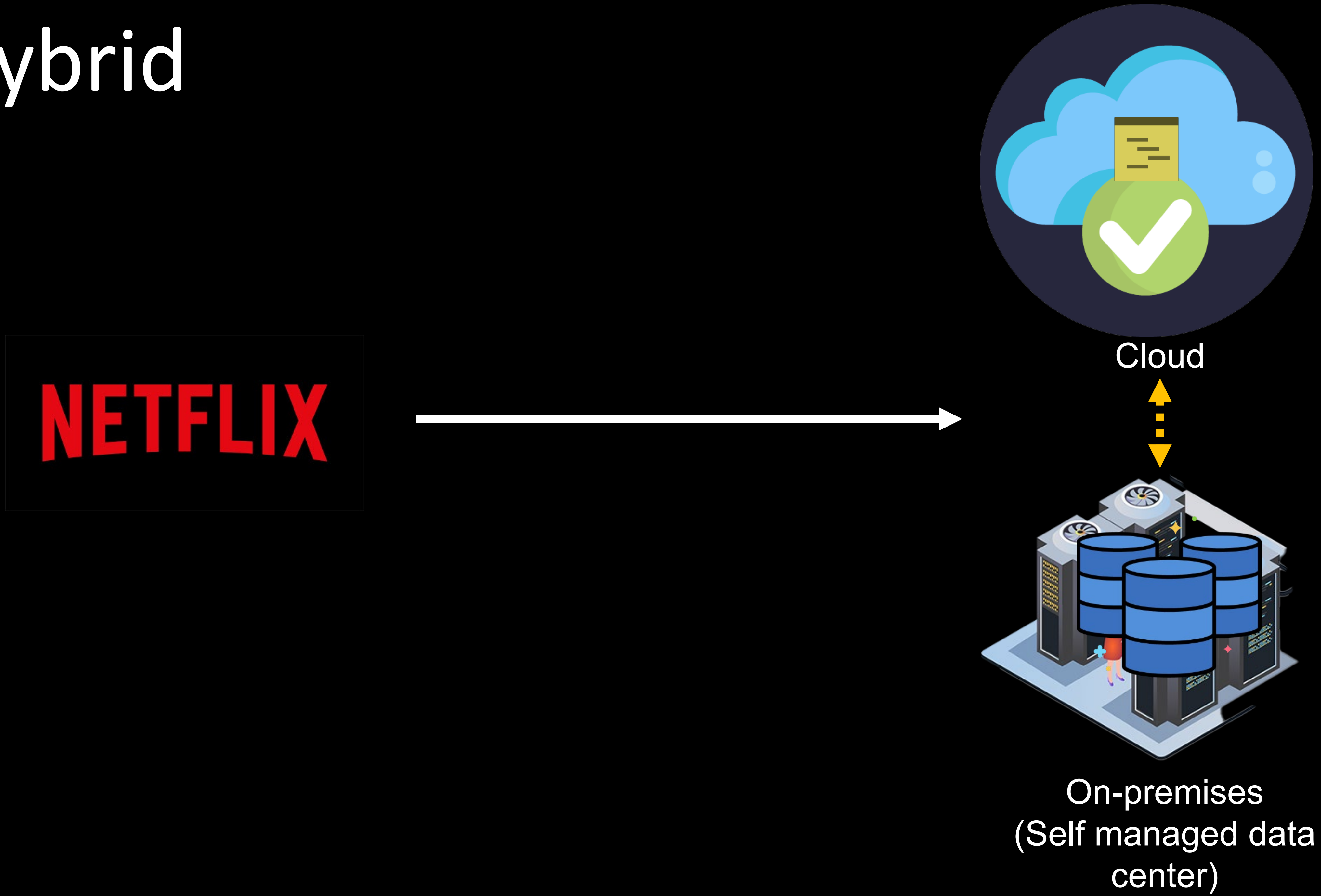
Cloud

# On-premises (Private Cloud)

NETFLIX

On-premises
(Self managed data
center)

# Hybrid



# Cloud

# Is AWS Secure?

Raj Saha
cloudwithraj.com
Cloud With Raj

# Yes, AWS is Secure

- You can implement your security practises of on prem to AWS
  - SSL
  - Encryption
  - Firewall
  - DDoS protection

- High security organizations like financial institutions and govt. workloads run on AWS
  - Fidelity, Finra, Fico, NASA, Dept of Defense (DoD), US Navy and many more

- AWS has the highest compliance
  - FedRAMP High, HIPAA, PCI, SOC and more
  - Have GovCloud, Secret, and Top Secret regions

- Largest ecosystem of security partners and solutions

- Economies of scale allows AWS to invest more on security than individual companies

# AWS Security Areas

## Identity and access management

AWS Identity and Access Management (IAM)

Amazon Cognito

AWS Directory Service

## Detection

Amazon GuardDuty

Amazon Inspector

Amazon CloudWatch

AWS CloudTrail

## Network and application protection

AWS Shield

AWS Network Firewall

AWS WAF

## Data protection

AWS Key Management Service (KMS)

AWS Secrets Manager

AWS Certificate Manager (ACM)

## Incident response

Amazon Detective

## Compliance

AWS Artifact

# Is AWS Expensive?

Raj Saha
cloudwithraj.com
Cloud With Raj

# Perception and Data

- AWS has reduced prices over 100 times since it's inception!
- AWS has multiple ways to optimize costs
  - Free tier
  - Reserved pricing mode
  - Compute savings plan
  - Spot pricing
  - Optimize applications
    - Auto scaling
    - Rightsize resources
    - AWS recommends insights
  - Billing forecasts and alarms
  - Numerous third party tools
- Customers – both conservative and startups adopted AWS

# Why Migrate to AWS?

Organizations who migrated to AWS from on-premises saw, on average:

| | | | | |
|---|---|---|---|---|
| **20%** infrastructure cost savings* | **66%** increase in administrator productivity* | **43%** lower time to market for new features* | **29%** increase in staff focus on innovation* | **45%** fewer security-related incidents* |

* "The Business Value of Migration to Amazon Web Services," The Hackett Group, January 2022

# AWS Cost Optimization Services

## AWS Native Services

AWS Cost & Usage Report

AWS Cost Explorer

AWS Budgets

Reserved Instance (RI) Reporting

Amazon CloudWatch Metrics Insights

AWS Well-Architected Tool

AWS Trusted Advisor

## Third Party Tools

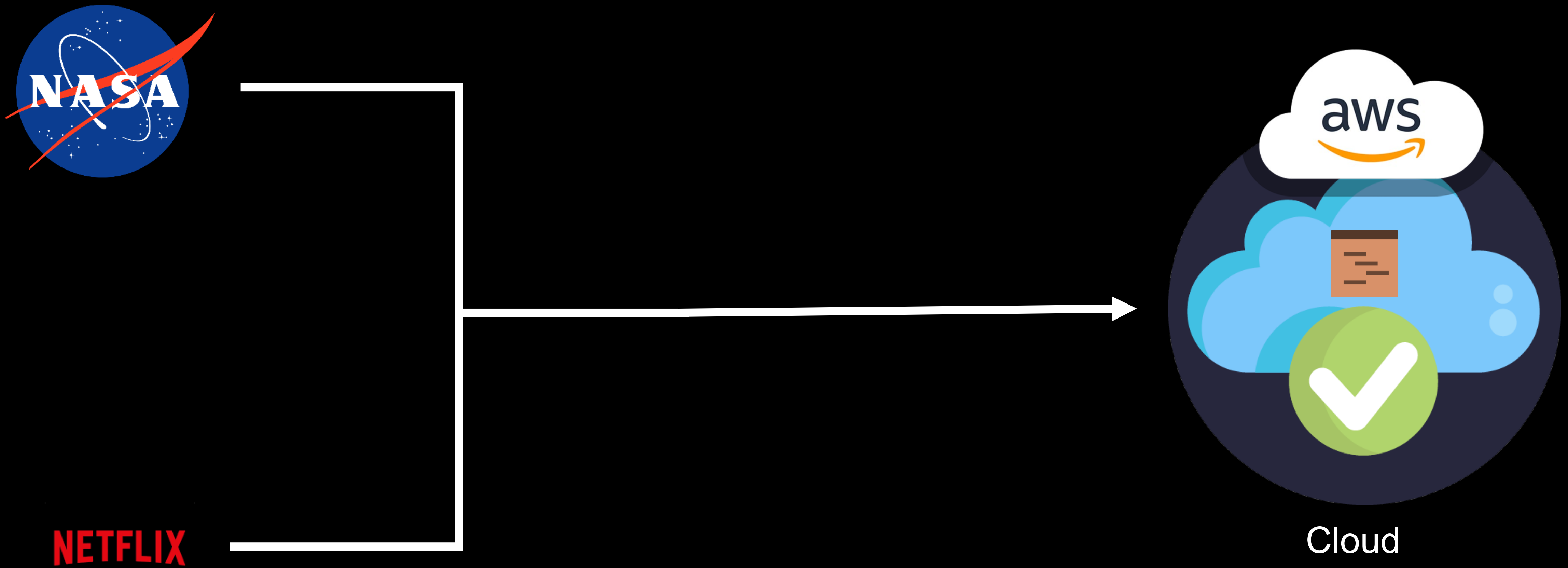
CloudHealth, Cloudability, CloudCheckr etc.

Kubecost (EKS Cost Optimization)

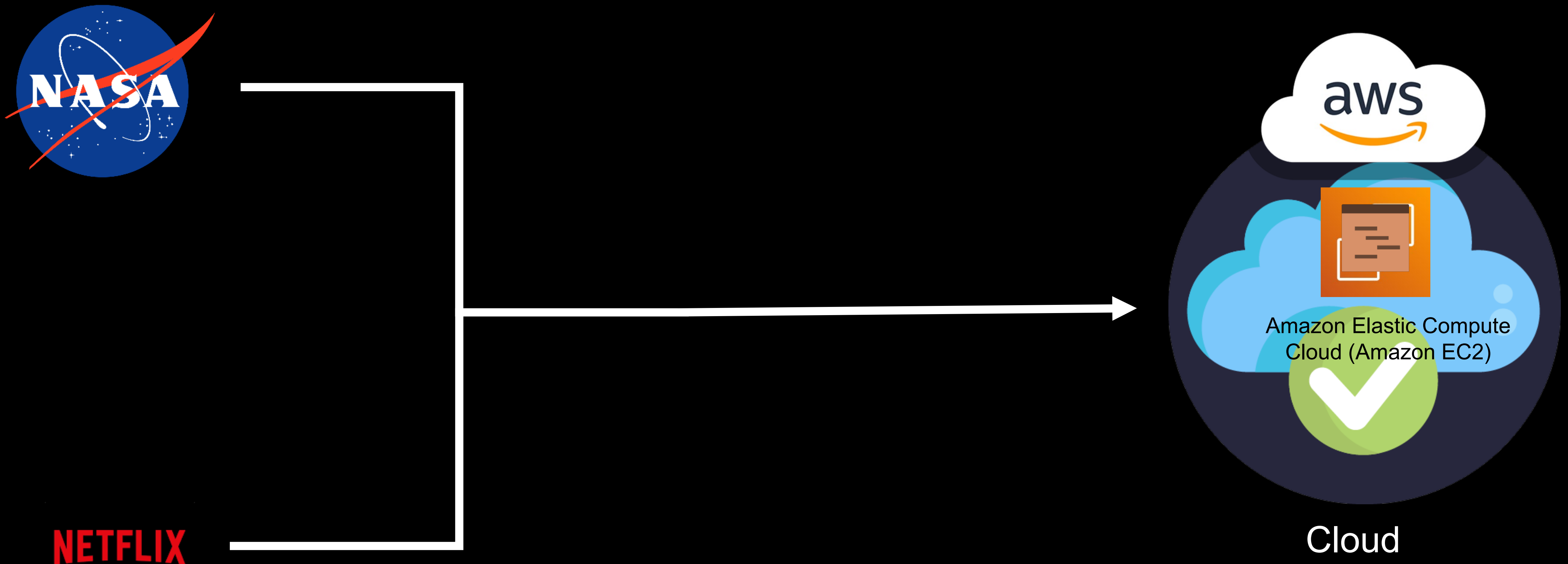
# Amazon EC2 – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

# Cloud Managed Data Center



# Cloud Managed Data Center



# Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2)

- Equivalent to “computer”/”server” in the cloud
- Provides compute, memory, storage for your application on the cloud
- Also known as instance
- Can be scaled (“Elastic”)
- Not all applications are same!
    - 500 different types of EC2s to choose from
    - Migrate from on-premises server to EC2

# Server VM Hypervisor

Raj Saha
cloudwithraj.com
Cloud With Raj

# Applications on Cloud

Give me 1 CPU

Give me 2 CPU

Give me 4 CPU

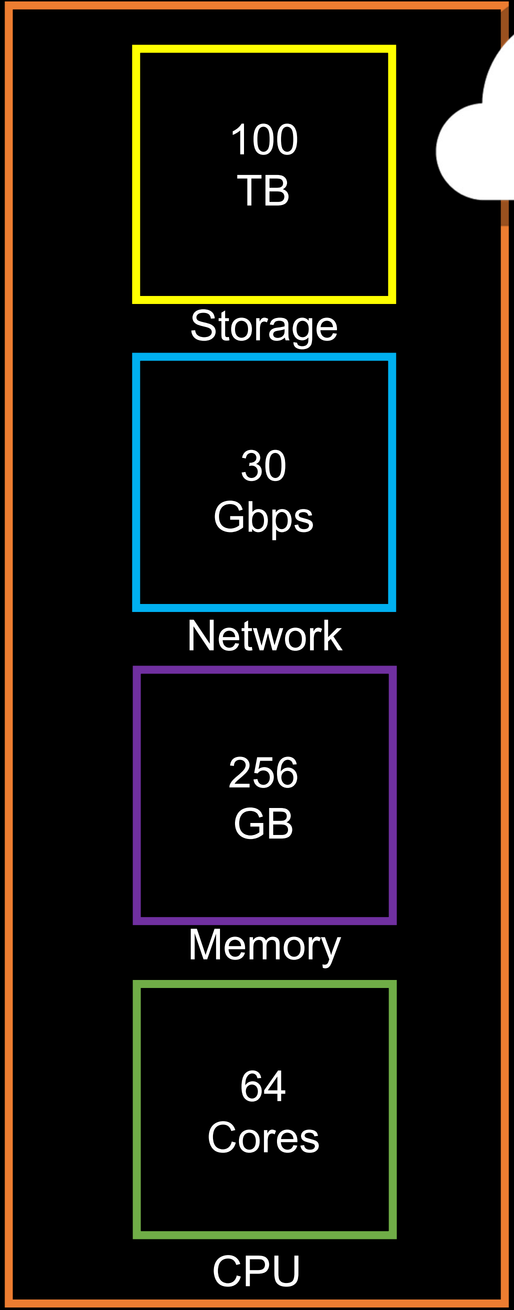

Data Center Managed by Cloud

- Array of computers
- Known as servers

# Inside the Datacenter

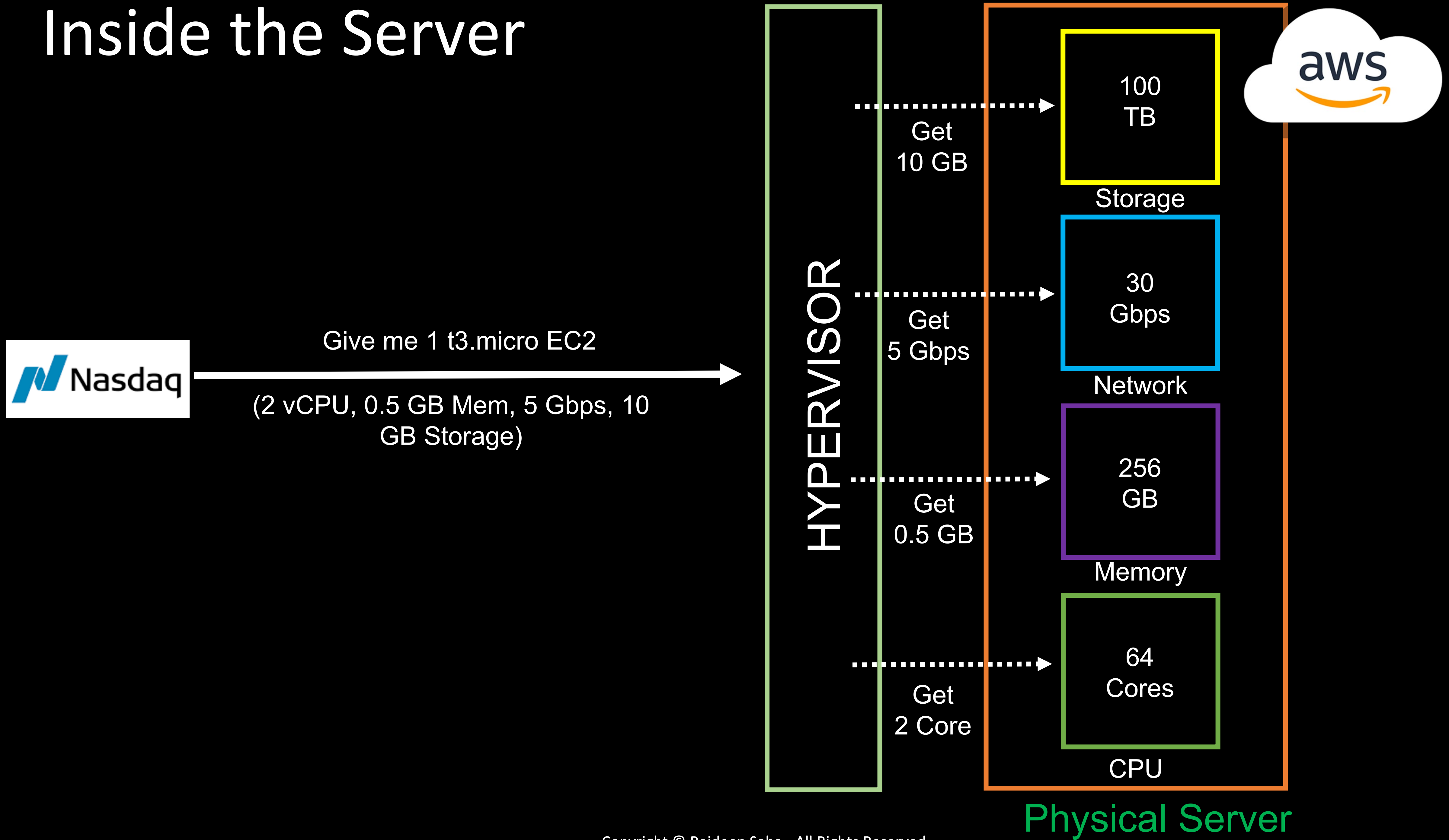
- They can't plug in a new server every time a small request comes
    - Not efficient
- Fill the data center with maximum capacity servers

# Inside the Datacenter

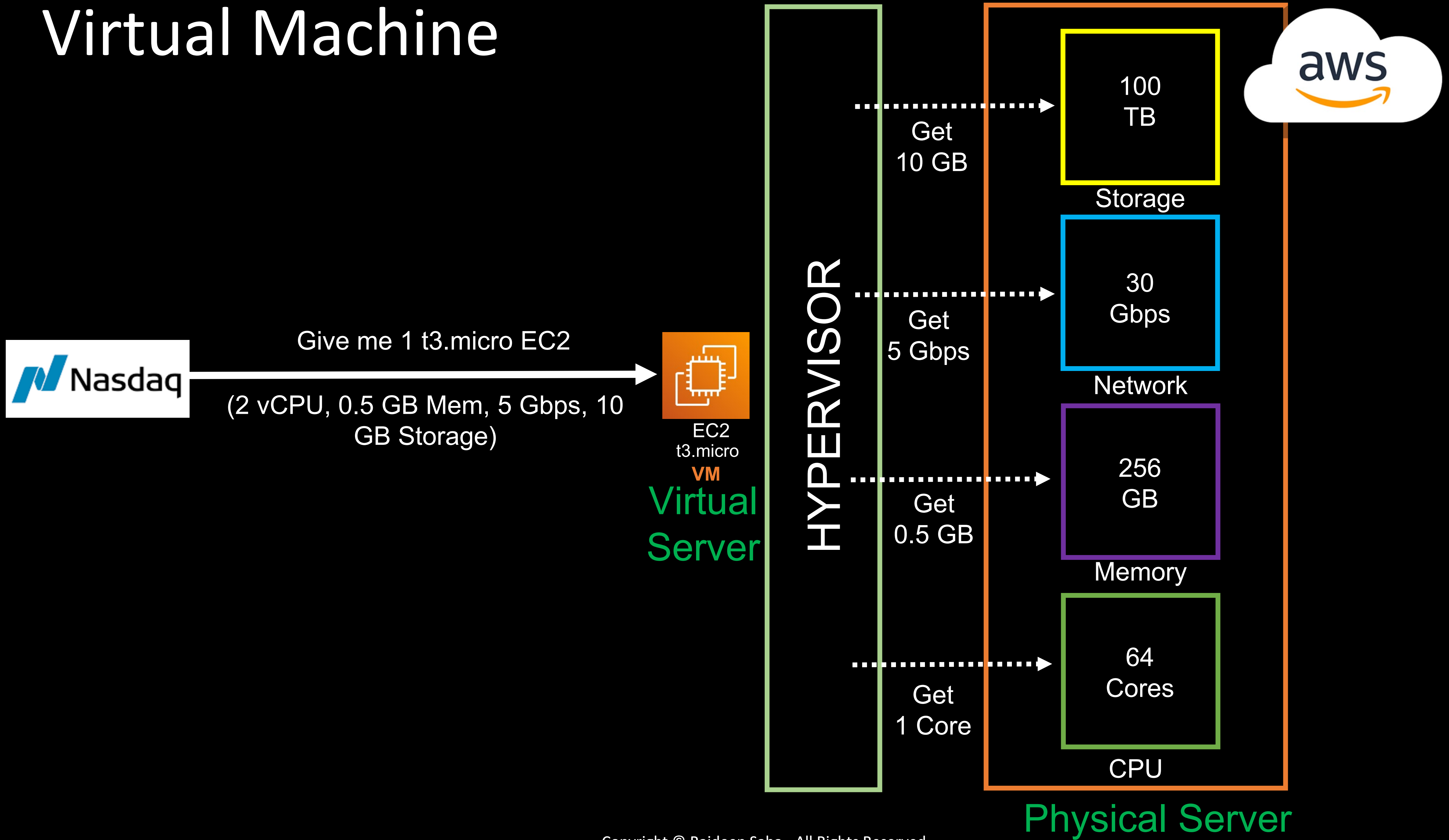



Physical Server (NOT EC2)

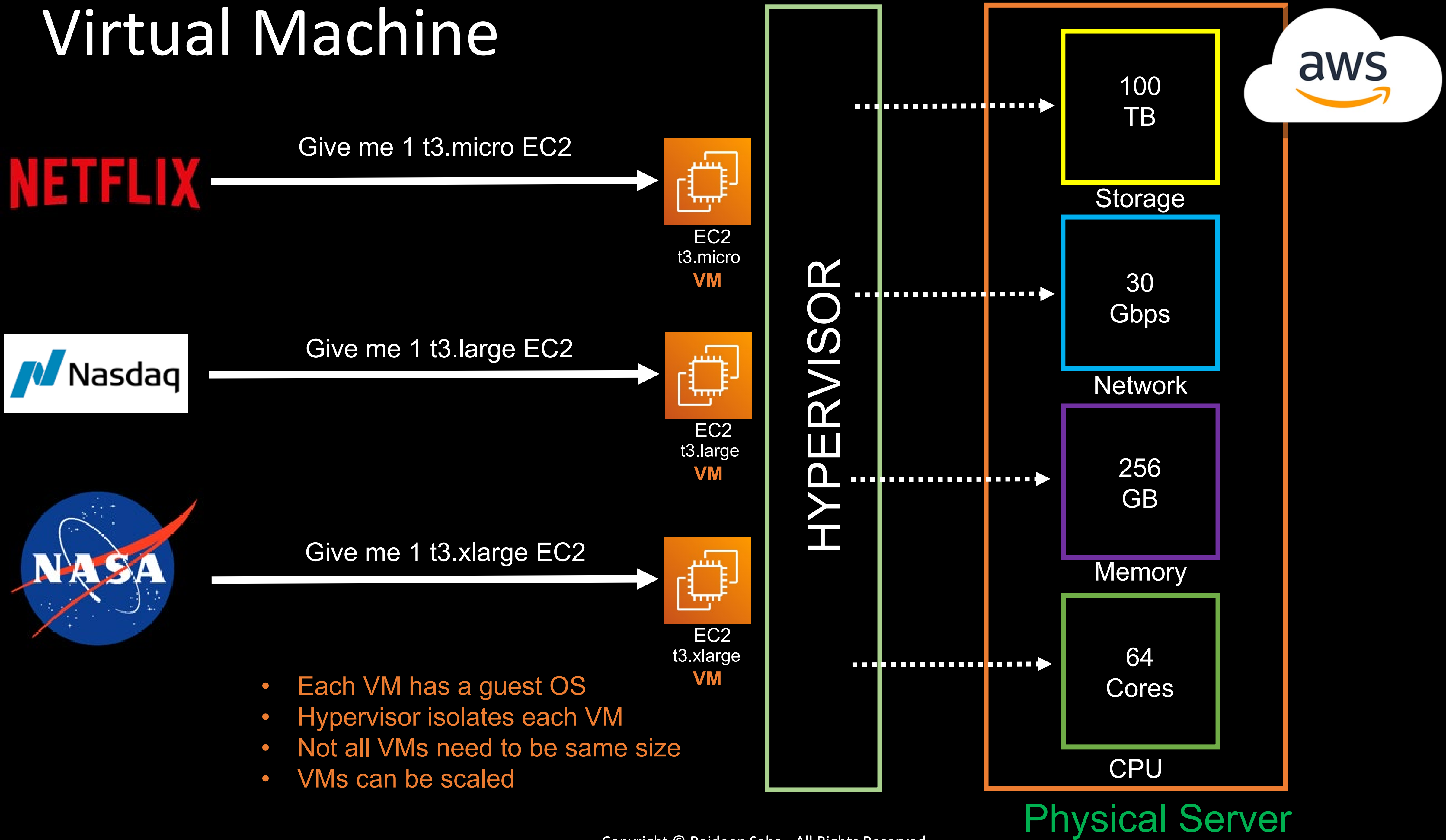
# Inside the Server



# Virtual Machine



# Virtual Machine




- Each VM has a guest OS
- Hypervisor isolates each VM
- Not all VMs need to be same size
- VMs can be scaled

# EC2 Types – Yes There's A LOT!!

Amazon Elastic Compute Cloud (Amazon EC2)

- General Purpose
- Compute Optimized
- Memory Optimized
- Accelerated Computing
- Storage Optimized

# EC2 Types – Yes There's A LOT!!

Amazon Elastic Compute Cloud
(Amazon EC2)

- General Purpose
  - Balance of compute, memory, networking
  - Diverse use cases – web servers, code repositories
- Compute Optimized
  - High performance processors
  - Batch processing, media transcoding, high performance computing (HPC), any compute intensive applications
- Memory Optimized
  - Process large datasets in memory
  - In memory database, caching
- Accelerated Computing
  - Hardware accelerators, co-processors, to perform functions
  - Graphics processing, data pattern matching
- Storage Optimized
  - High sequential read write to large data sets on local storage
  - Relational database, NoSQL databases, search engine, analytics workloads

# EC2 Types – Yes There's A LOT!!

Amazon Elastic Compute Cloud (Amazon EC2)

- General Purpose
  - Balance of compute, memory, networking
  - Diverse use cases – web servers, code repositories
  - Mac, T2/3/4, M5/6, A1

- Compute Optimized
  - High performance processors
  - Batch processing, media transcoding, high performance computing (HPC), any compute intensive applications
  - C5/6/7

- Memory Optimized
  - Process large datasets in memory
  - In memory database, caching
  - R4/5/6, X1/2

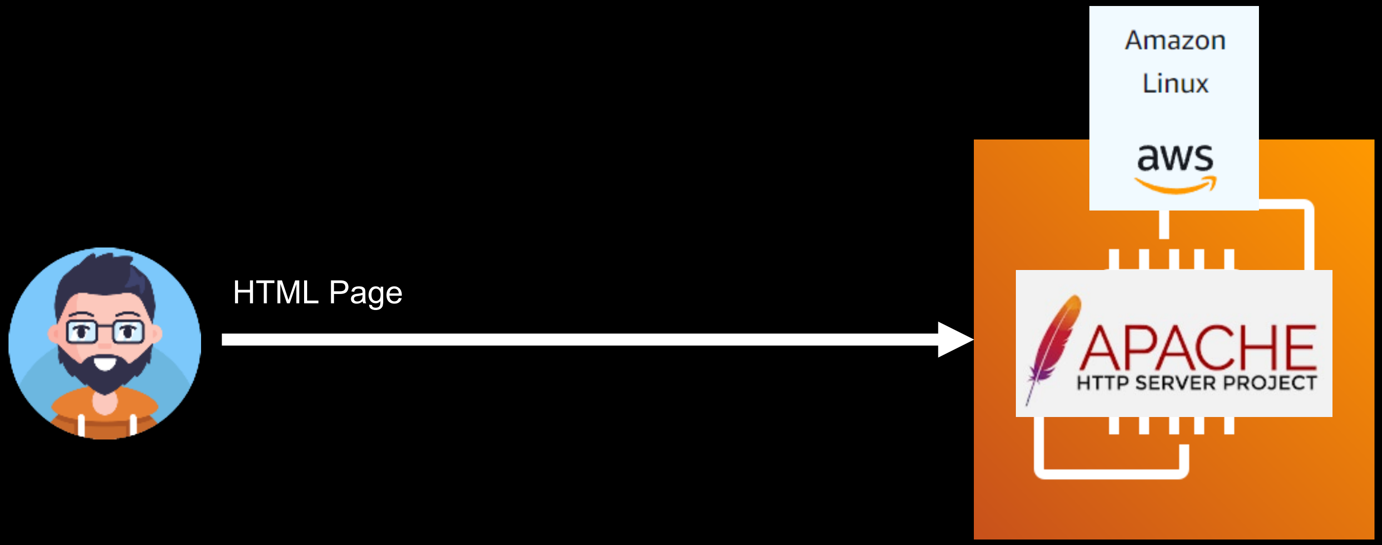
- Accelerated Computing
  - Hardware accelerators, co-processors, to perform functions
  - Graphics processing, data pattern matching
  - G3/4/5, P2/3/4

- Storage Optimized
  - High sequential read write to large data sets on local storage
  - Relational database, NoSQL databases, search engine, analytics workloads
  - I3/4, D2/3

# Demo – Our Very First EC2

Raj Saha
cloudwithraj.com
Cloud With Raj

# Demo - Amazon Elastic Compute Cloud (EC2)




Amazon Elastic Compute Cloud (Amazon EC2)

- Amazon Linux AMI
- Apache Webserver

# Scaling EC2 – Horizontal Vs Vertical

Raj Saha
cloudwithraj.com
Cloud With Raj

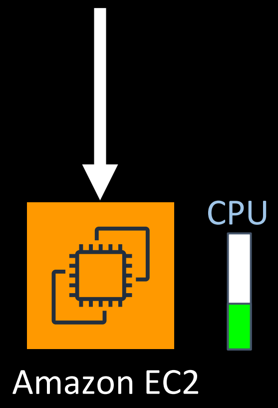
# Elastic?

Amazon Elastic Compute Cloud
(Amazon EC2)

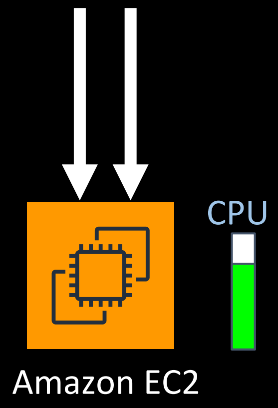
# Elastic?

An elastic band stretches and shrinks based on need

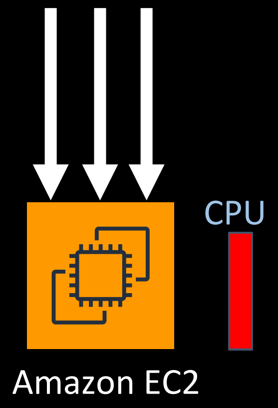
# App Running in EC2



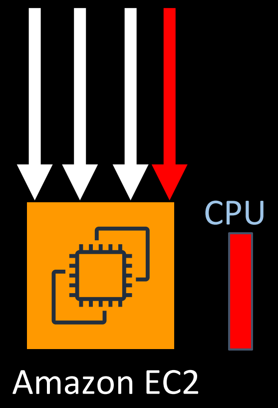
# App Running in EC2



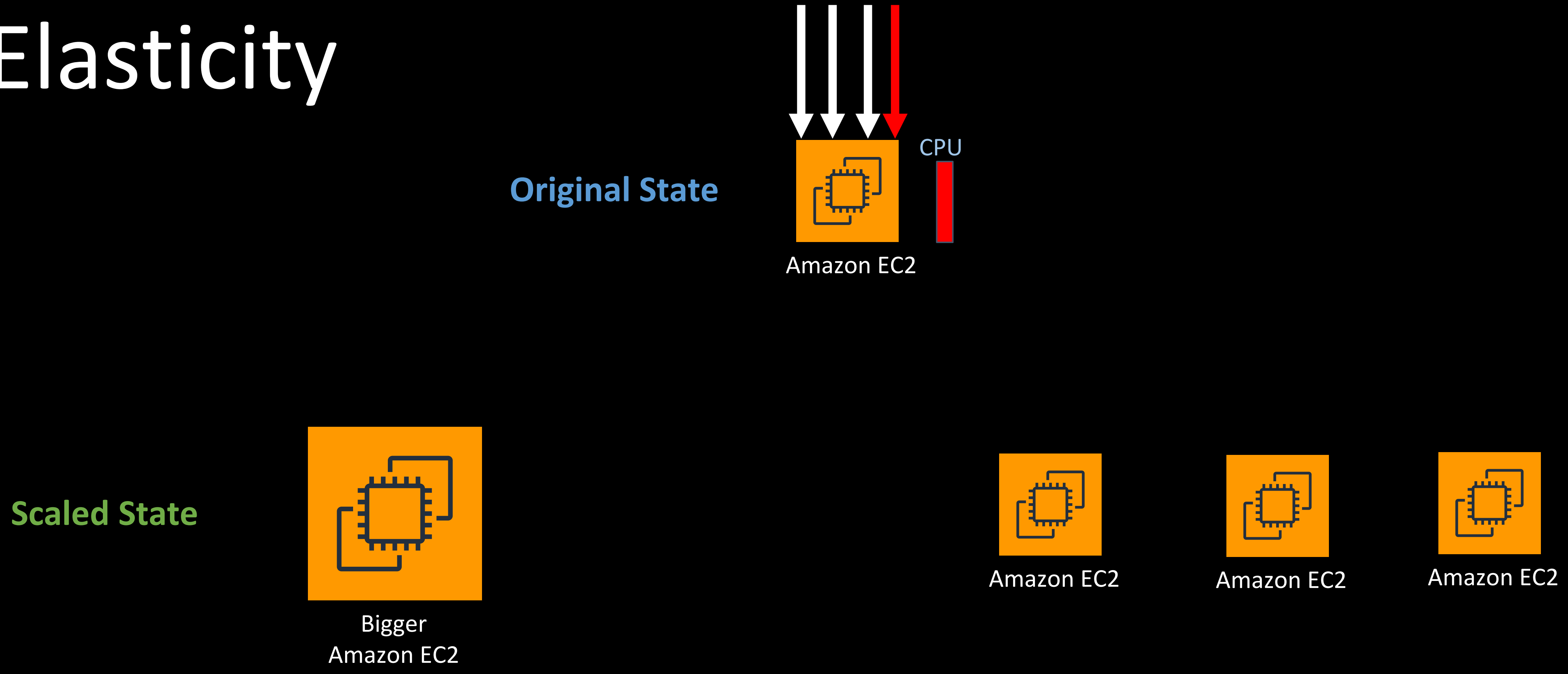
# App Running in EC2



# App Running in EC2



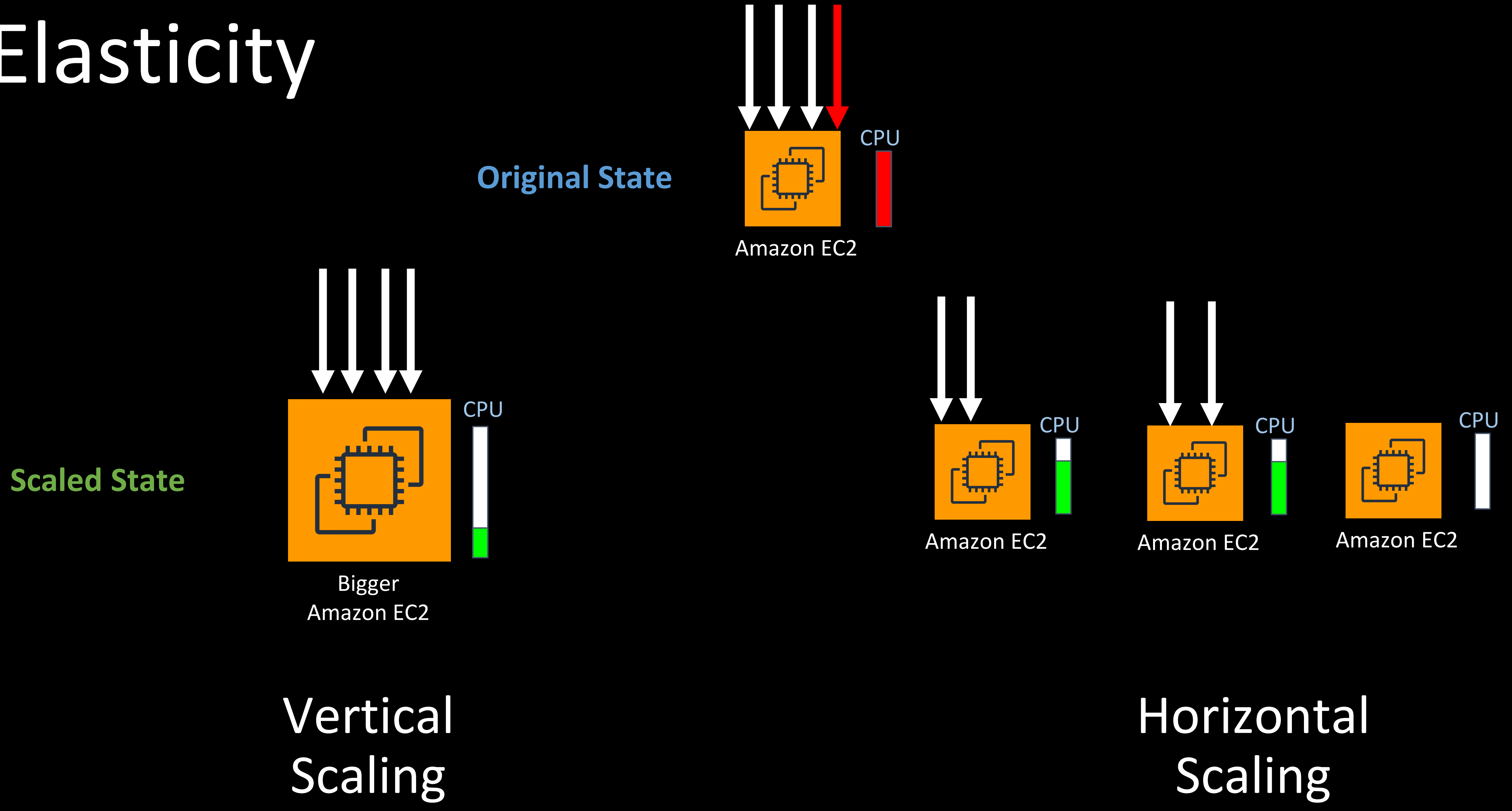
# Elasticity



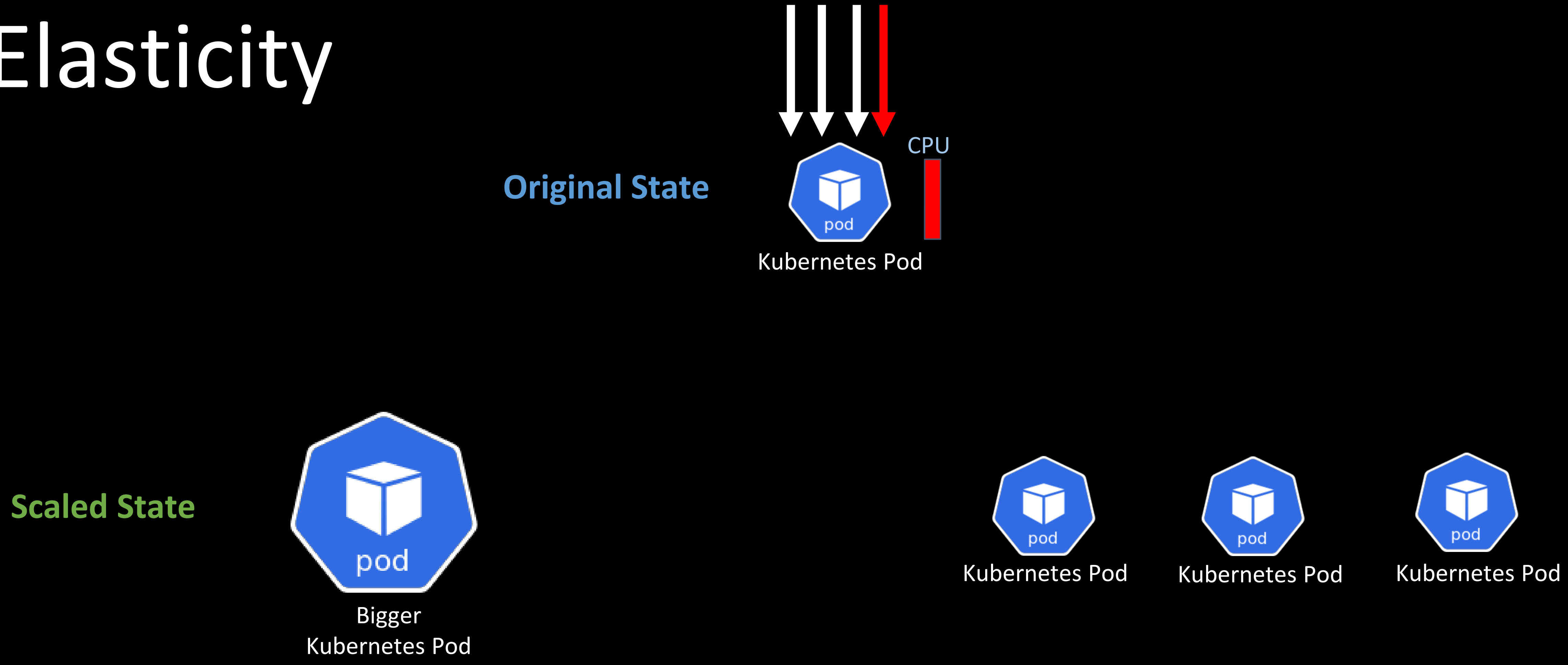

Vertical Scaling

Horizontal Scaling

# Elasticity


Original State
CPU
Amazon EC2
Scaled State
CPU
Bigger
Amazon EC2
CPU
Amazon EC2
CPU
Amazon EC2
CPU
Amazon EC2
Vertical
Scaling
Horizontal
Scaling

# Elasticity



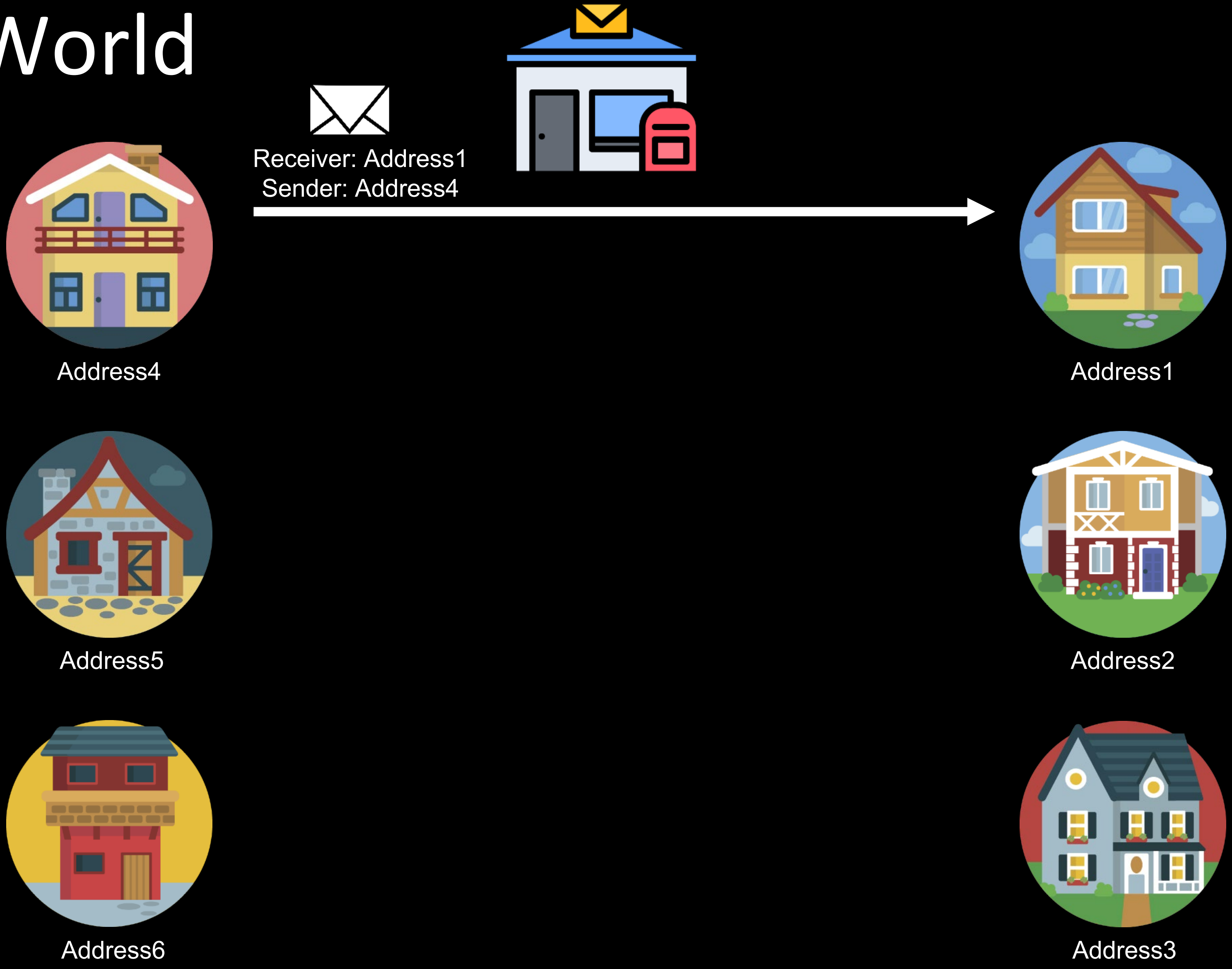

Vertical Scaling

Horizontal Scaling

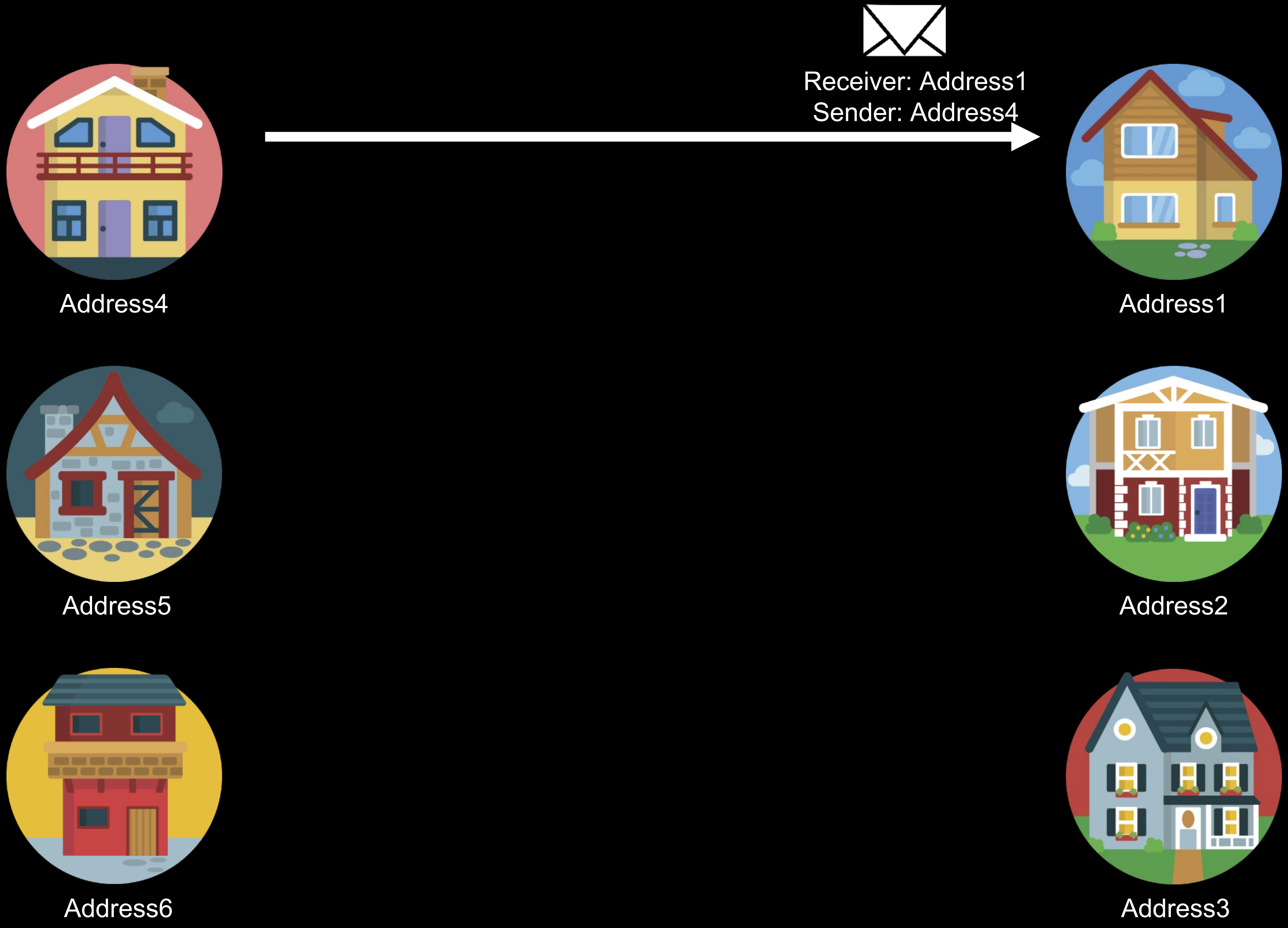
# IP Address Vs URL

Raj Saha
cloudwithraj.com
Cloud With Raj

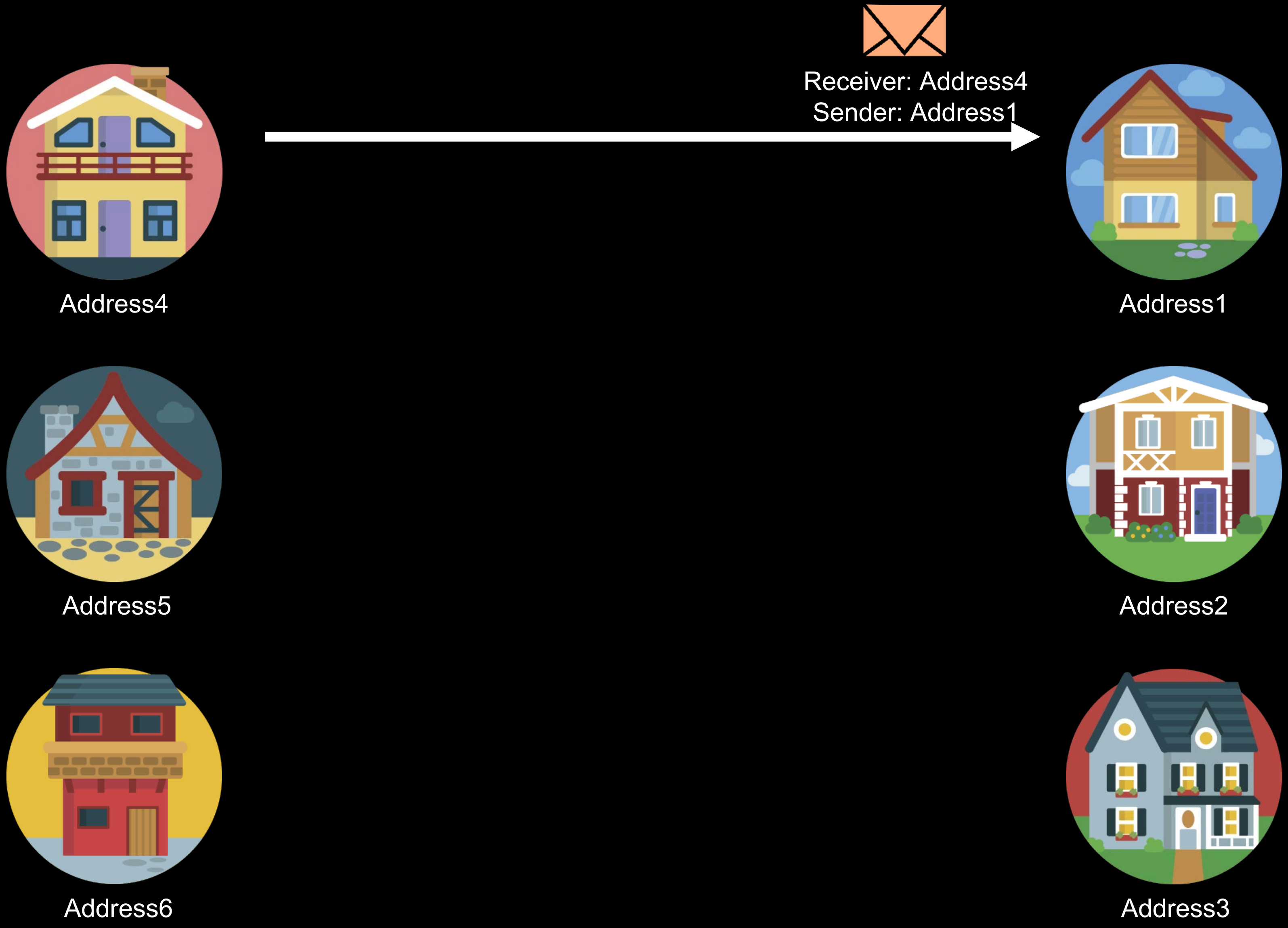
# In Real World



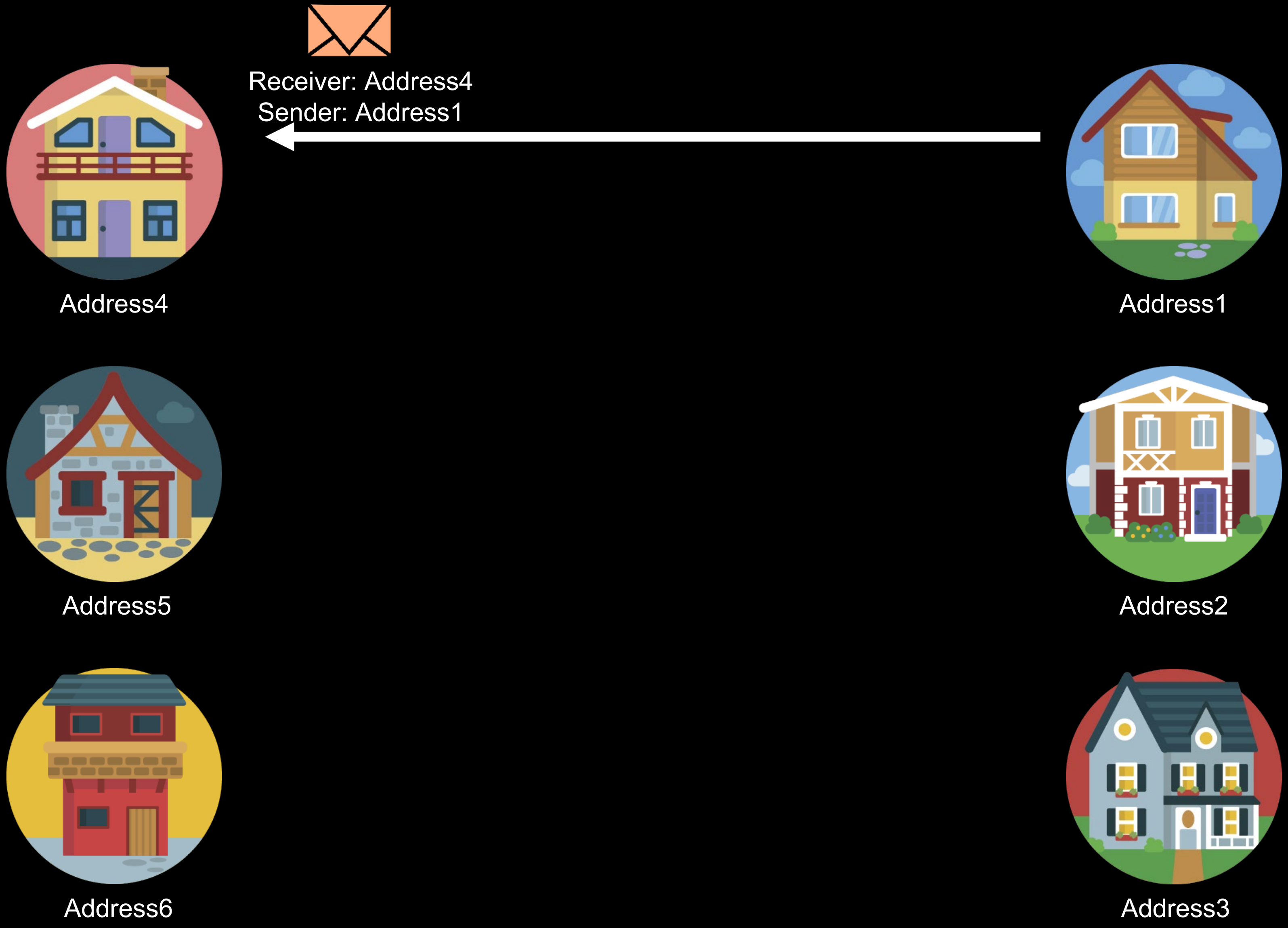
# In Real World



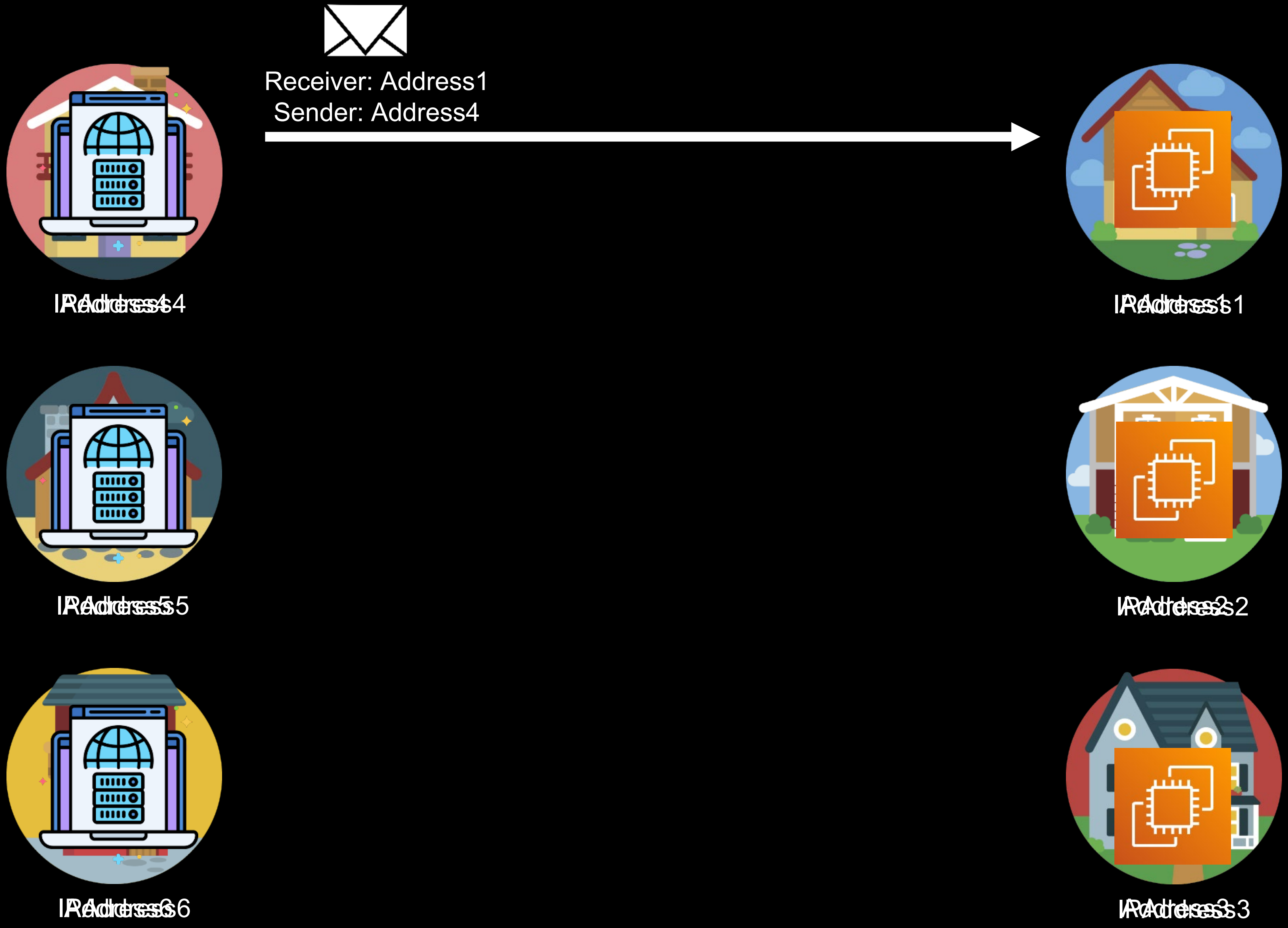
# In Real World



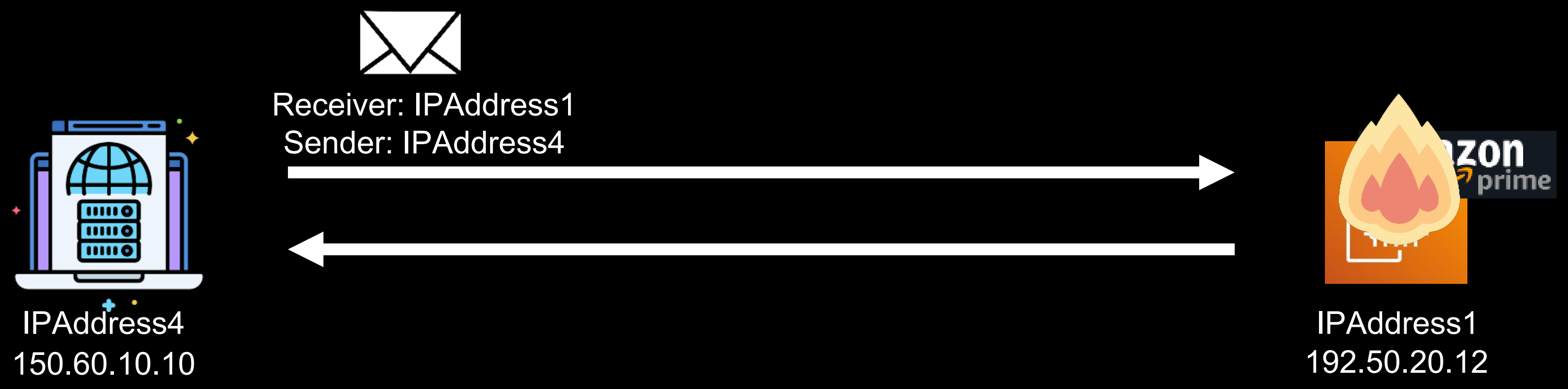
# In Real World



# In IT World



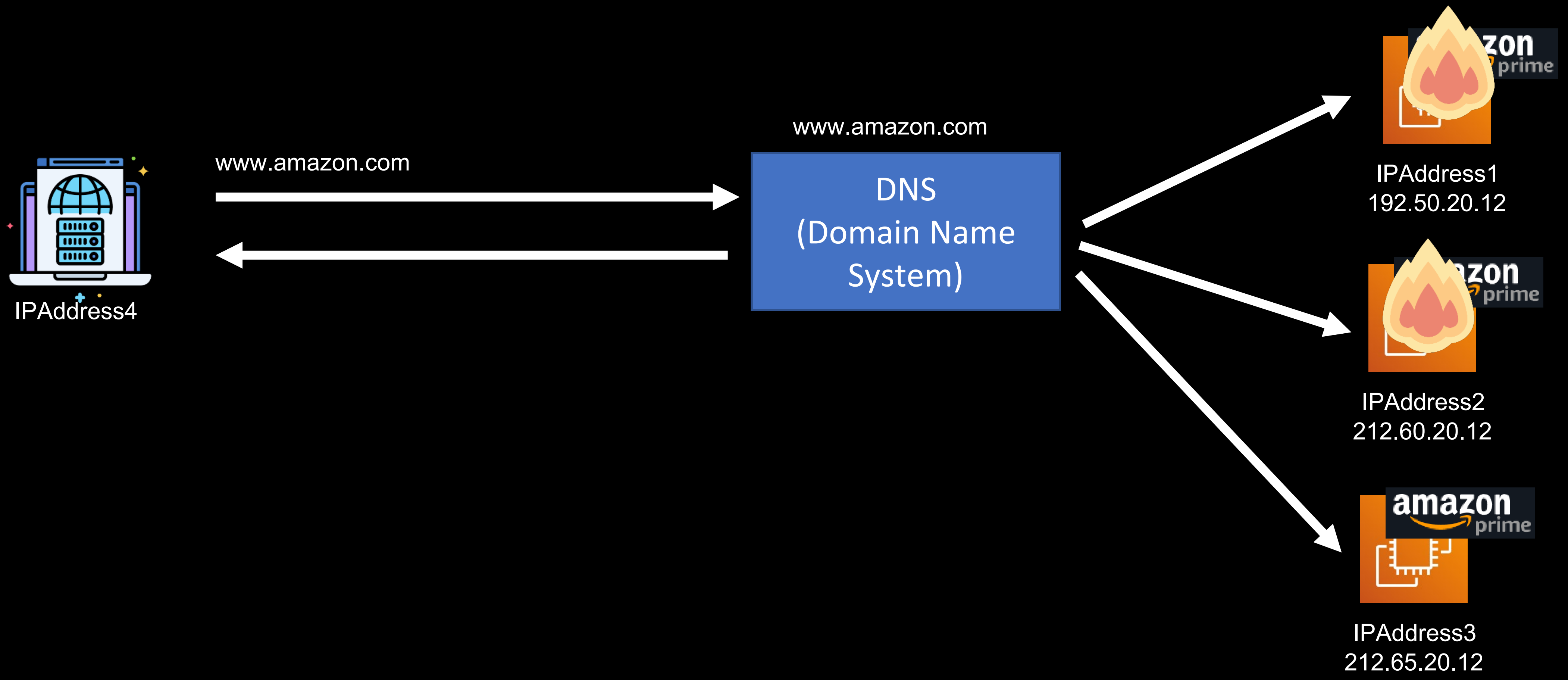
# In IT World


Receiver: IPAddress1
Sender: IPAddress4
IPAddress4
150.60.10.10
IPAddress1
192.50.20.12

# In IT World


Receiver: IPAddress1
Sender: IPAddress4
IPAddress4
150.60.10.10
IPAddress1
192.50.20.12
amazon prime
IPAddress2
212.60.20.12

# URL (Uniform Resource Location)



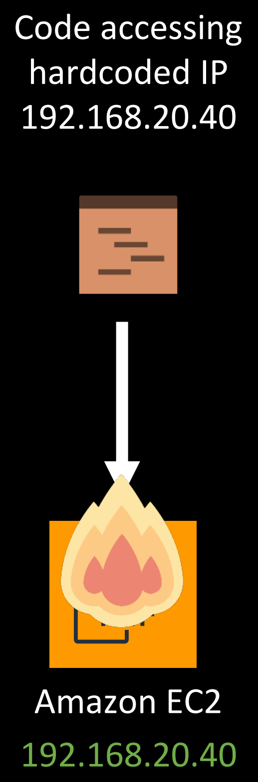
# URL (Uniform Resource Location)


- The address pointing to a unique resource in the web
- User friendly than dealing with IP address
- AWS DNS system is Amazon Route 53

# Intro to Elastic Load Balancer

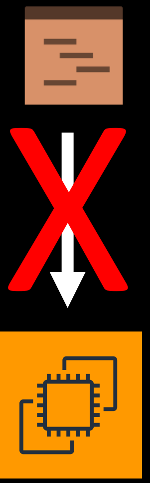
Raj Saha
cloudwithraj.com
Cloud With Raj

# Accessing Servers



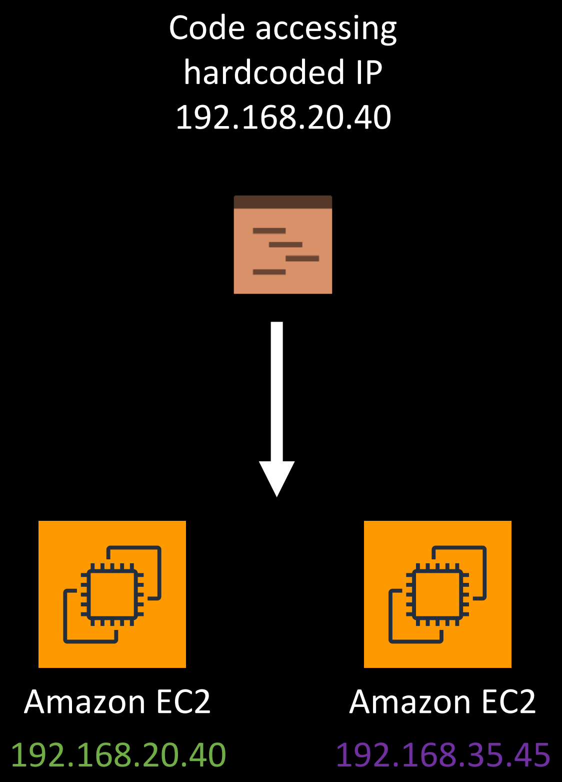
# Accessing Servers

Code accessing
hardcoded IP
192.168.20.40



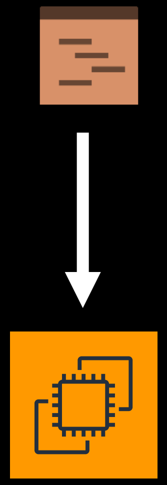
Amazon EC2
192.168.35.45

# Discovering Servers



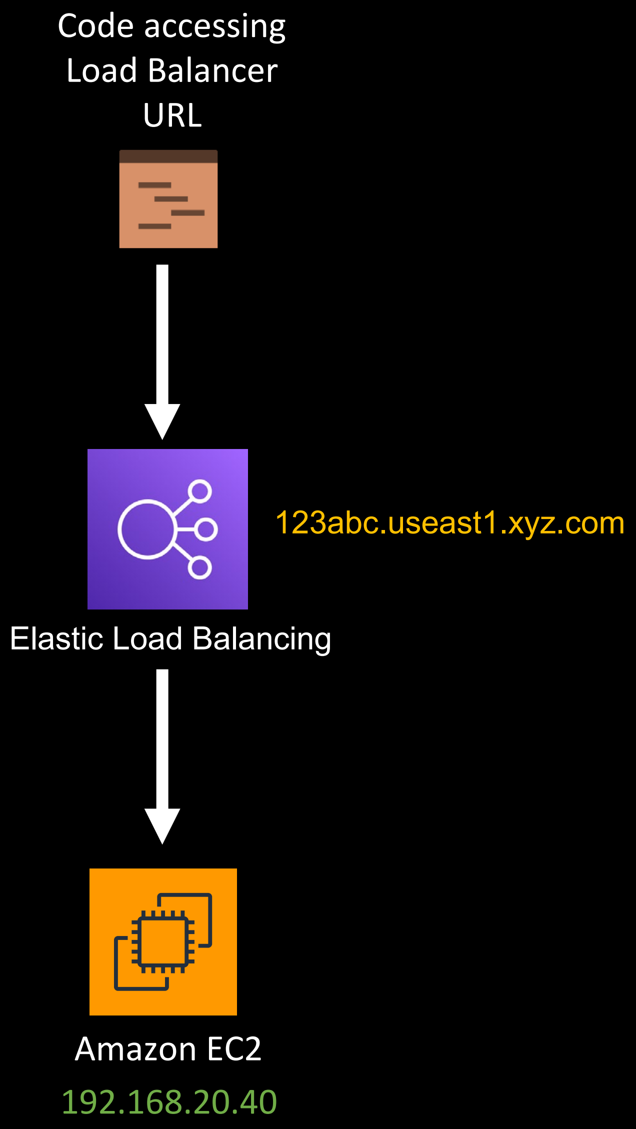
# Accessing Servers

Code accessing
hardcoded IP
192.168.20.40

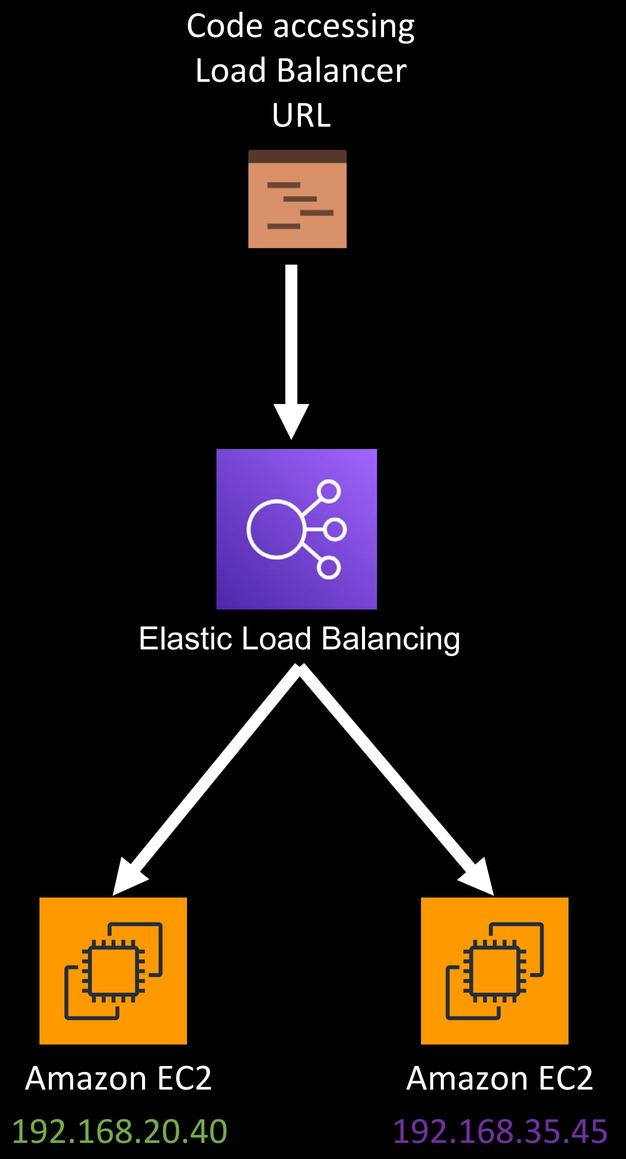


Amazon EC2
192.168.20.40

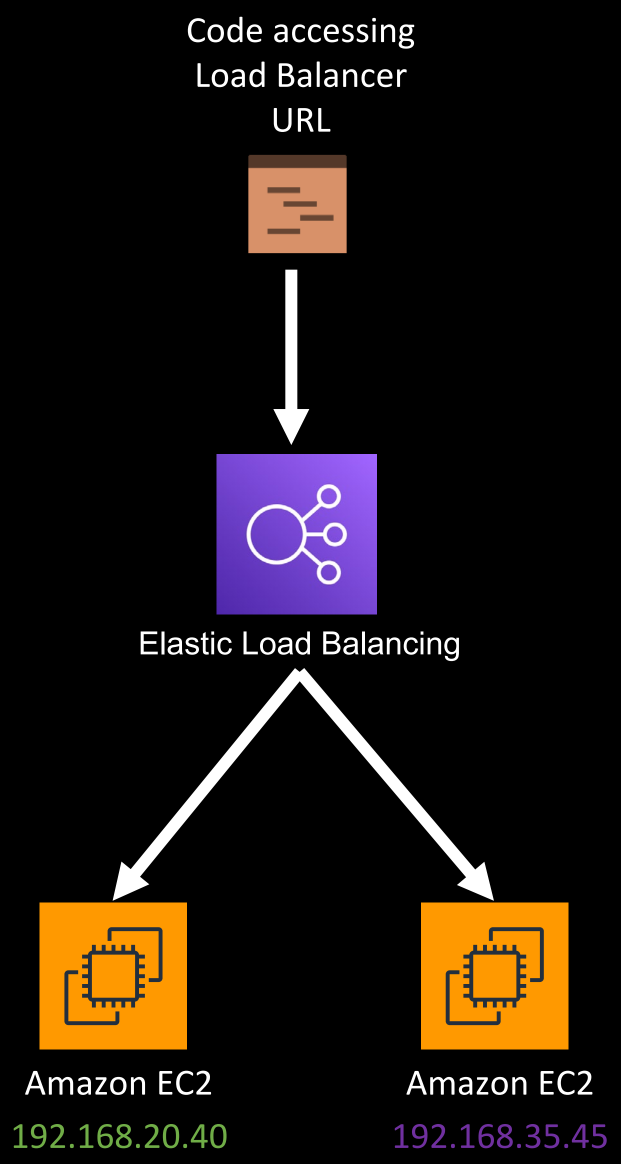
# Elastic Load Balancer



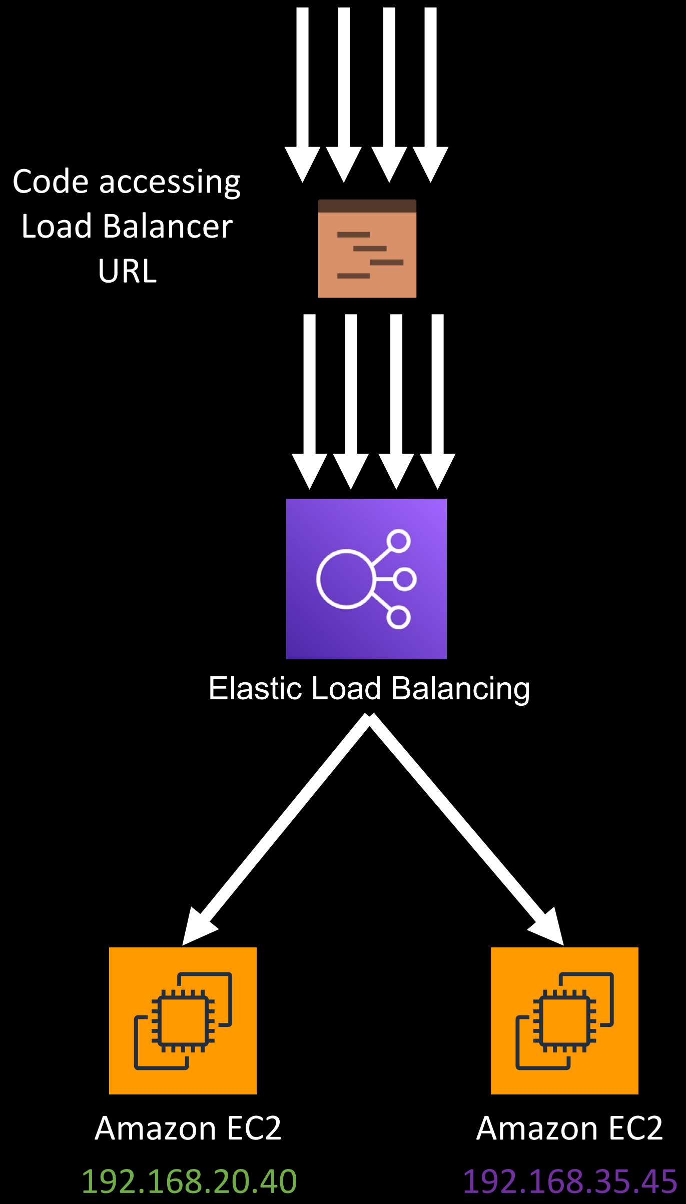
# Elastic Load Balancer



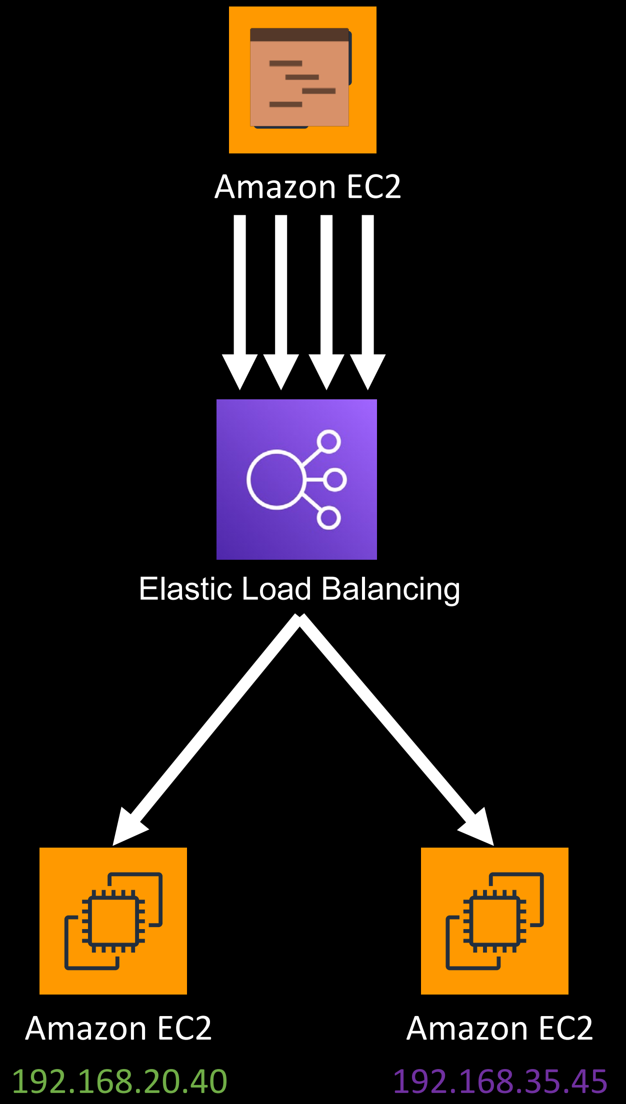
# Elastic Load Balancer



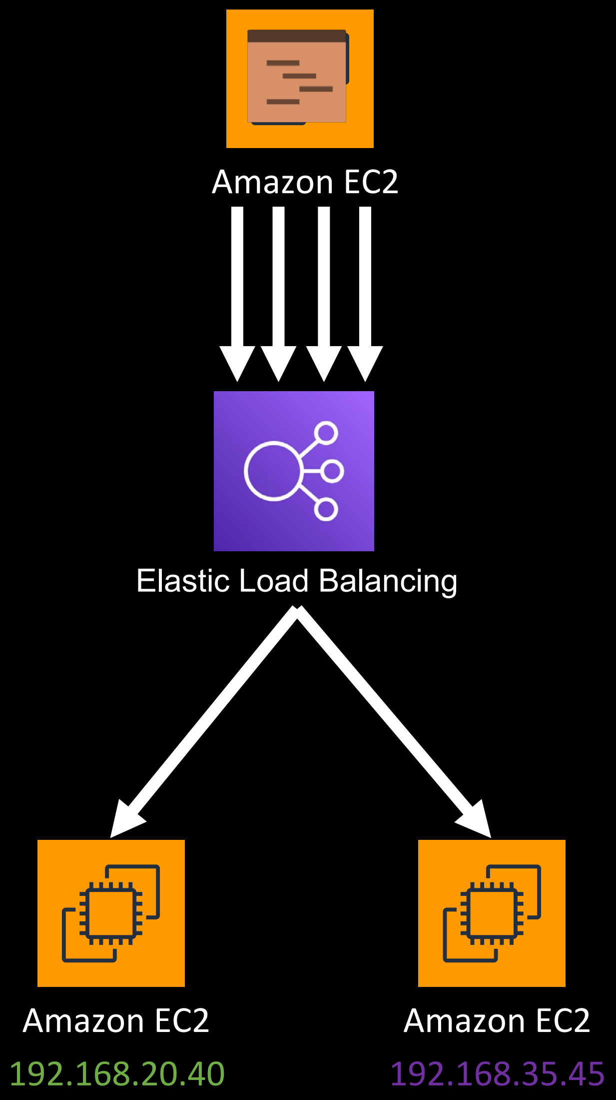
# Elastic Load Balancer



# Elastic Load Balancer



# Elastic Load Balancer

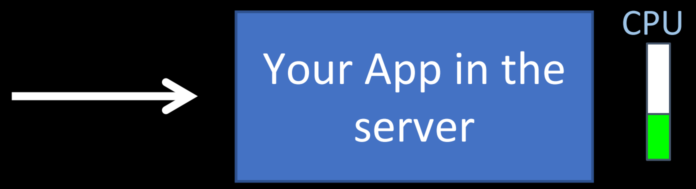



- Automatically distributes incoming traffic across multiple targets
- Automatically scales
- Automatically discovers targets
- ALB (Application Load Balancer) and NLB (Network Load Balancer)
- ALB has fixed URL, NLB has fixed IP (By Default)

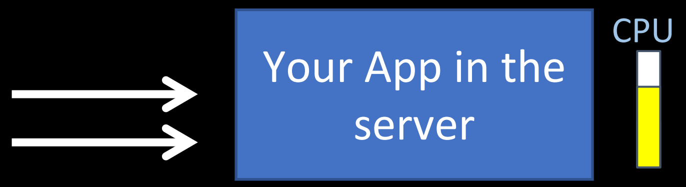
# Horizontal Vs Vertical Deep Dive

Raj Saha
cloudwithraj.com
Cloud With Raj

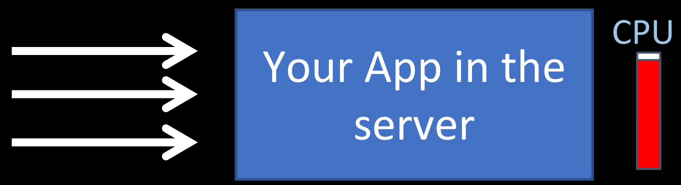
# Regular Application



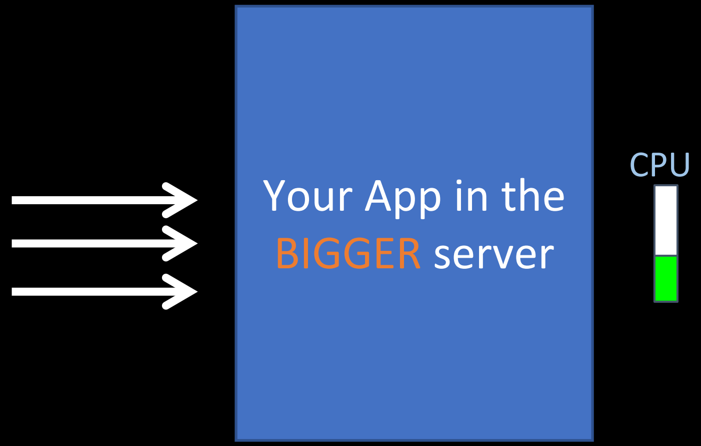
# Regular Application



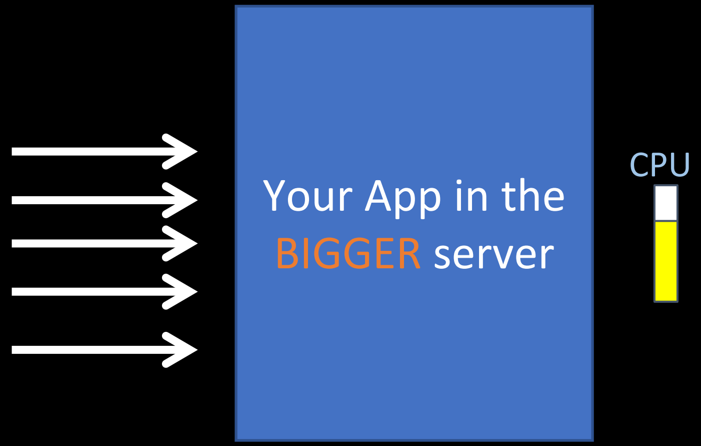
# Regular Application



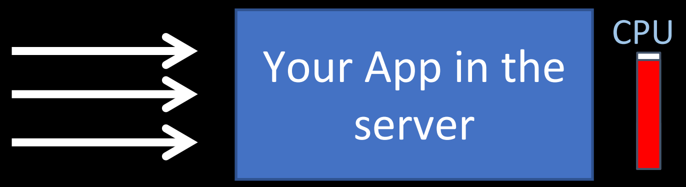
# Vertical Scaling



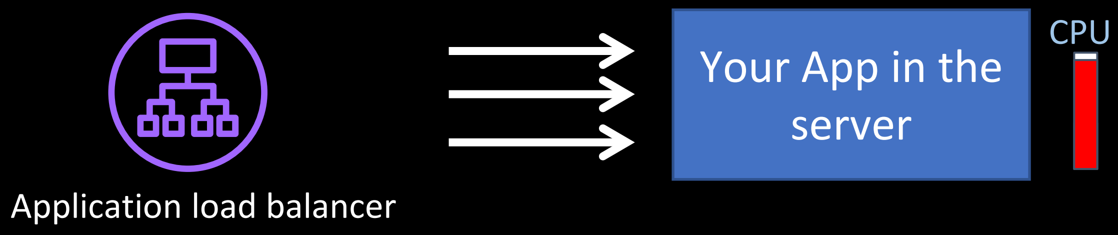
# Vertical Scaling



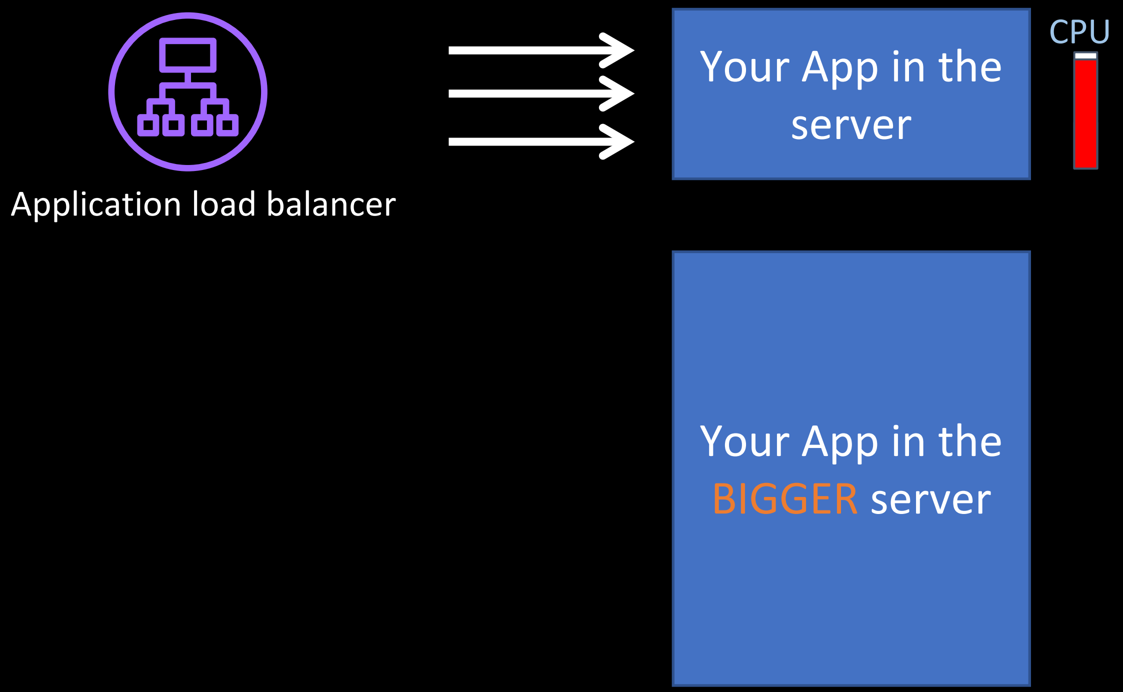
# Vertical Scaling Deep Dive



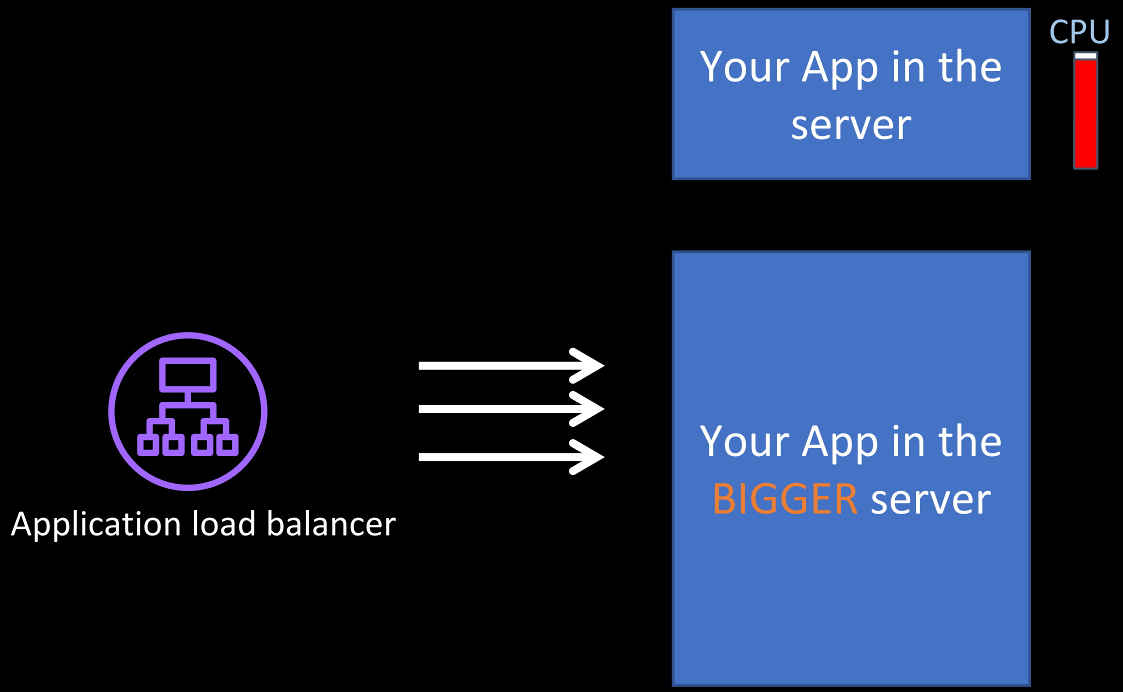
# Vertical Scaling Deep Dive



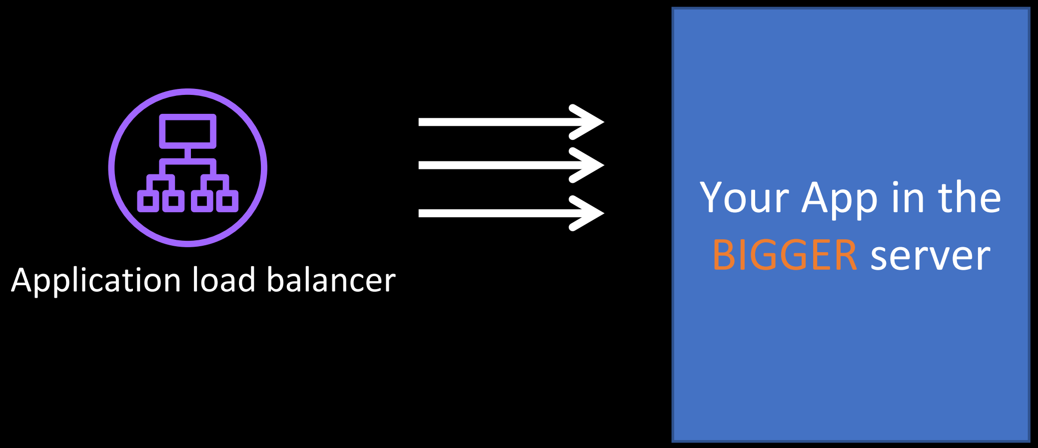
# Vertical Scaling Deep Dive



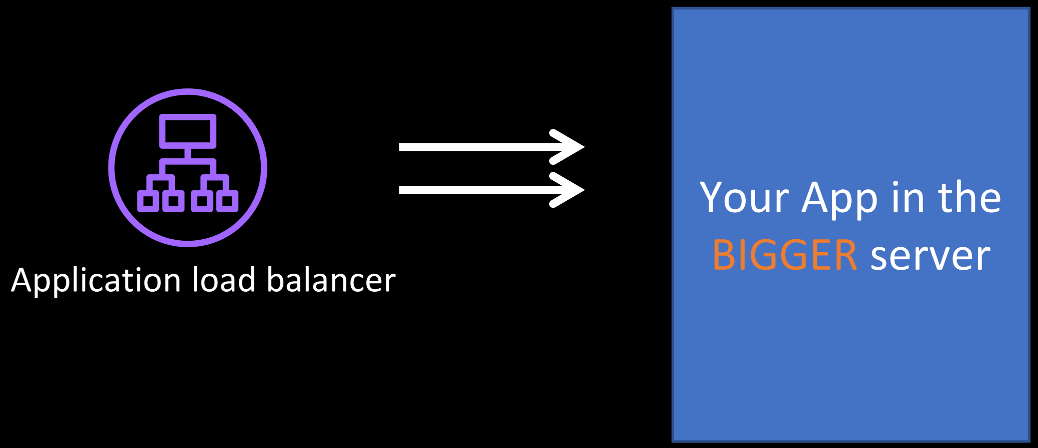
# Vertical Scaling Deep Dive



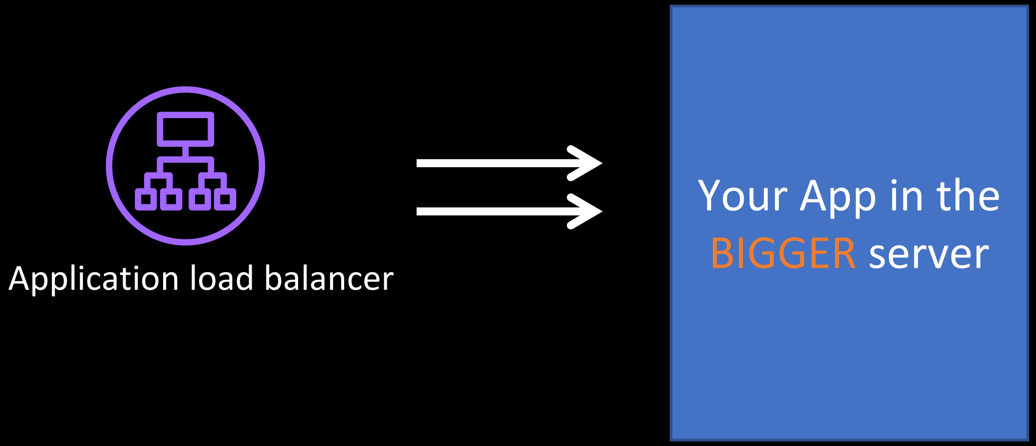
# Vertical Scaling Deep Dive



# Vertical Scaling Deep Dive

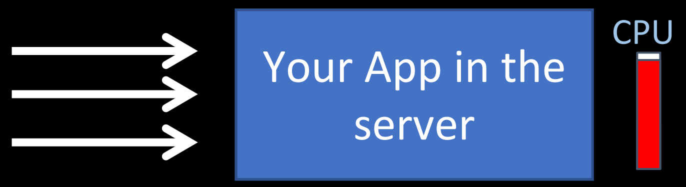


# Vertical Scaling Challenges

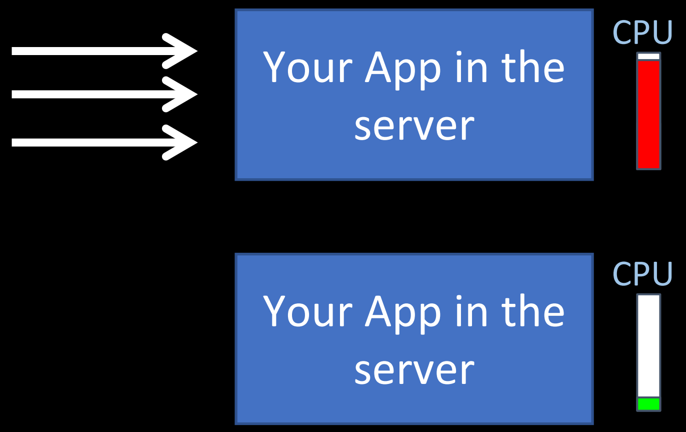



- Scaling up/down takes longer
- Chance of missing transactions during scaling cutover
- Limited scaling
- Expensive

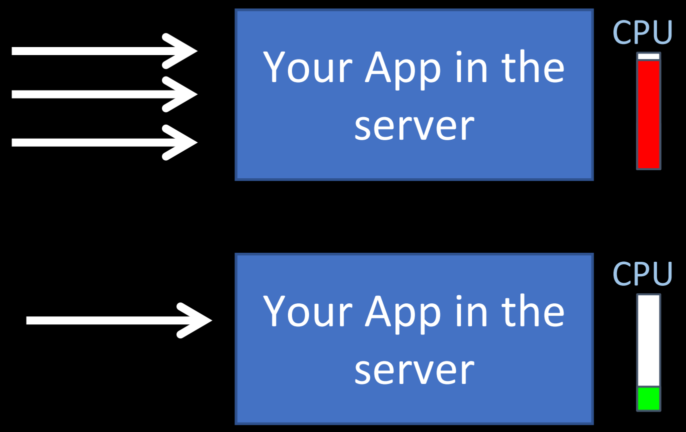
# Regular Application



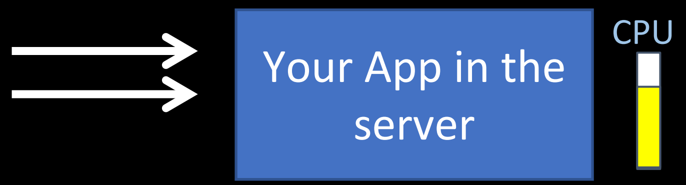
# Horizontal Scaling



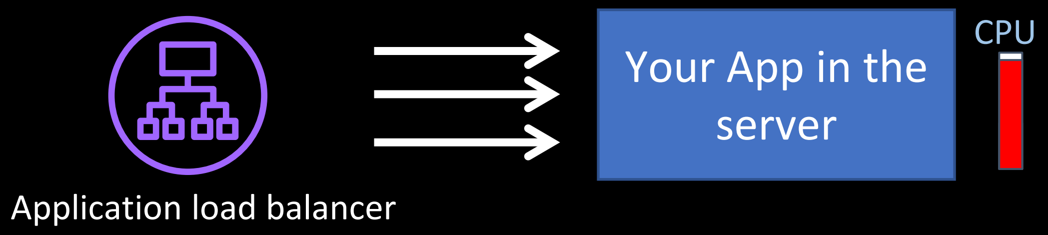
# Horizontal Scaling



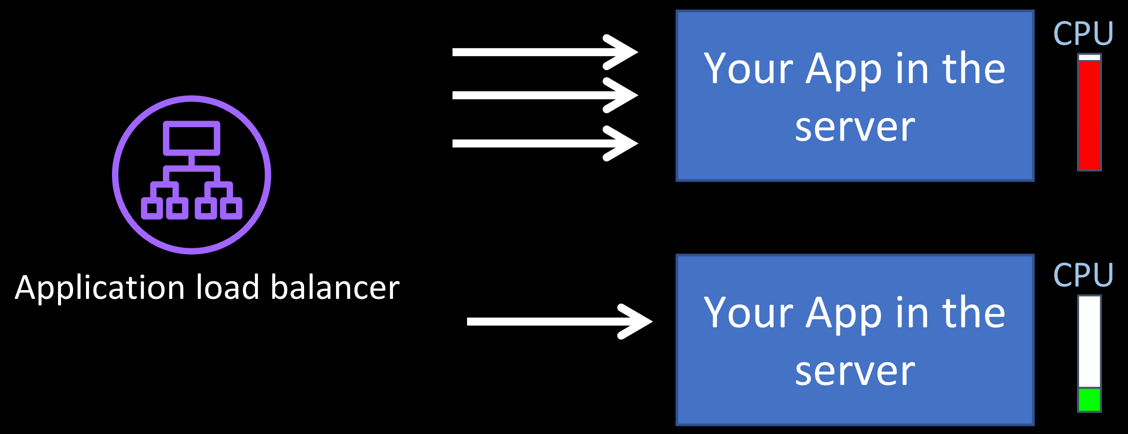
# Horizontal Scaling



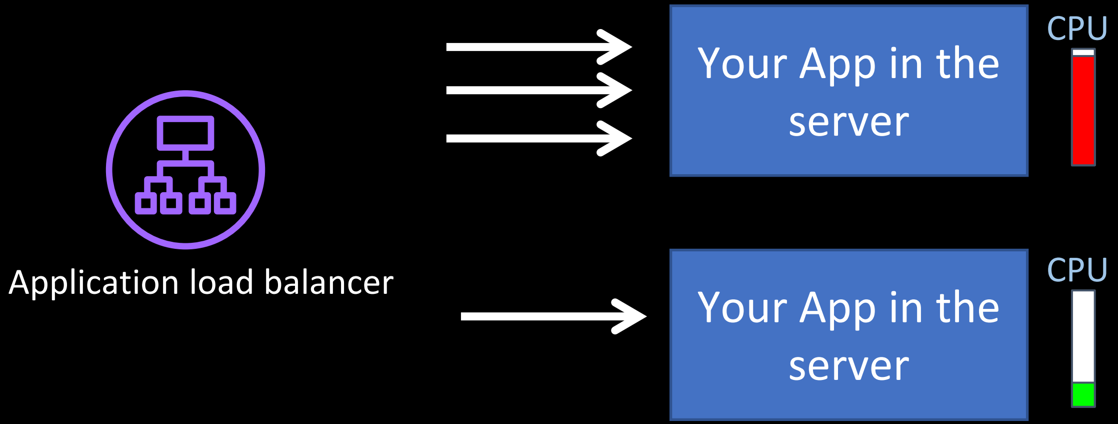
# Horizontal Scaling Deep Dive



# Horizontal Scaling Deep Dive



# Horizontal Scaling




- Scaling up/down faster
- Massively scalable
- Cost effective
- Legacy code needs to be refactored for horizontal scaling

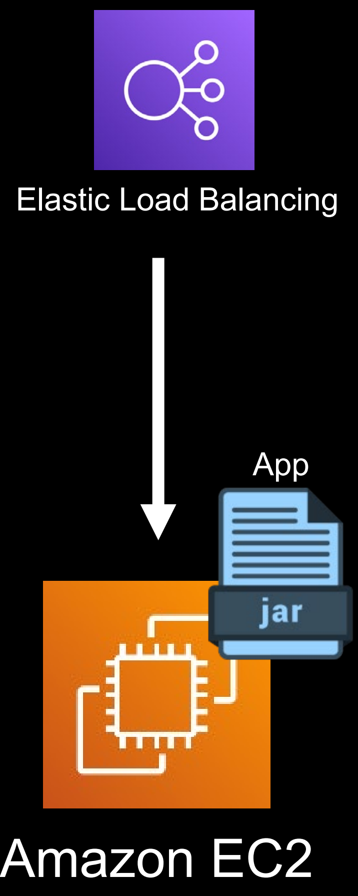
# Challenges of EC2 – Pathway to Lambda

Raj Saha
cloudwithraj.com
Cloud With Raj

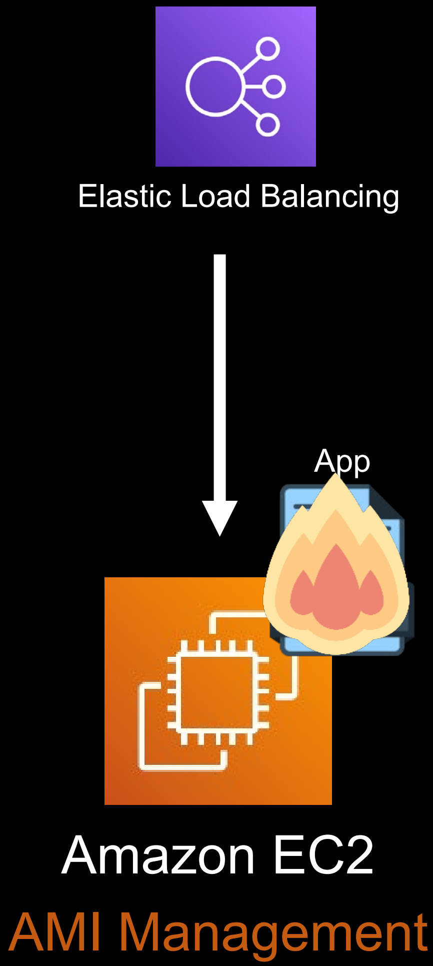
# AWS Lambda – What and Why

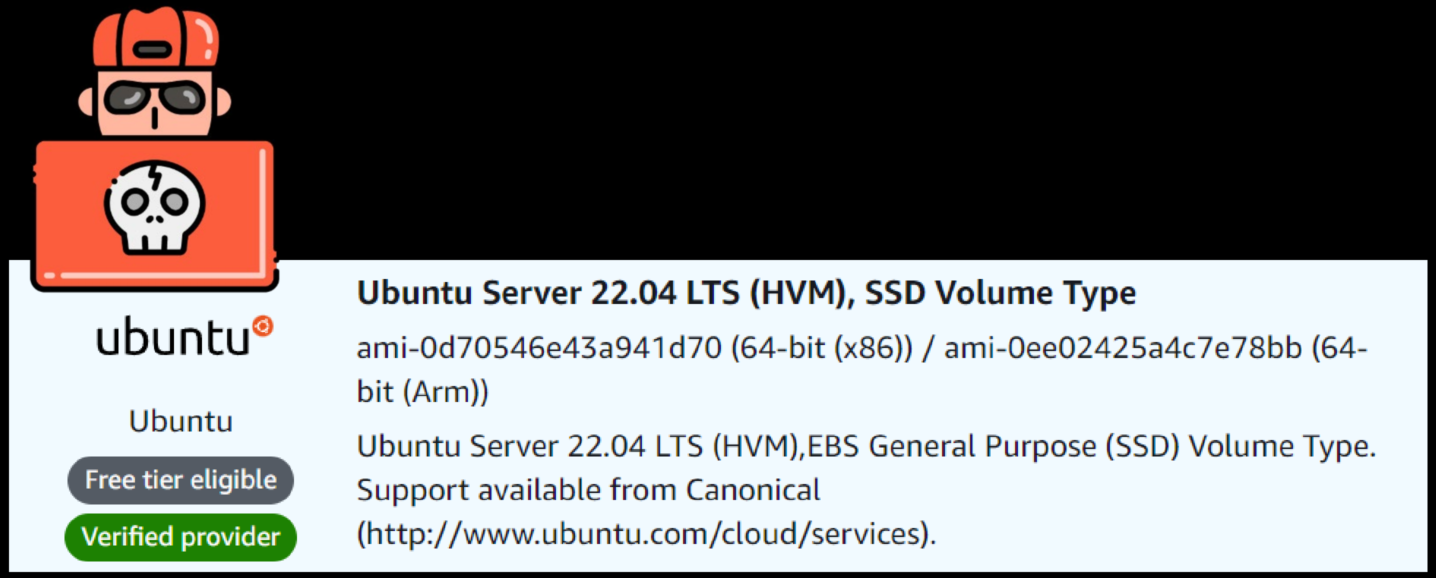
Raj Saha
cloudwithraj.com
Cloud With Raj

# Why Lambda?

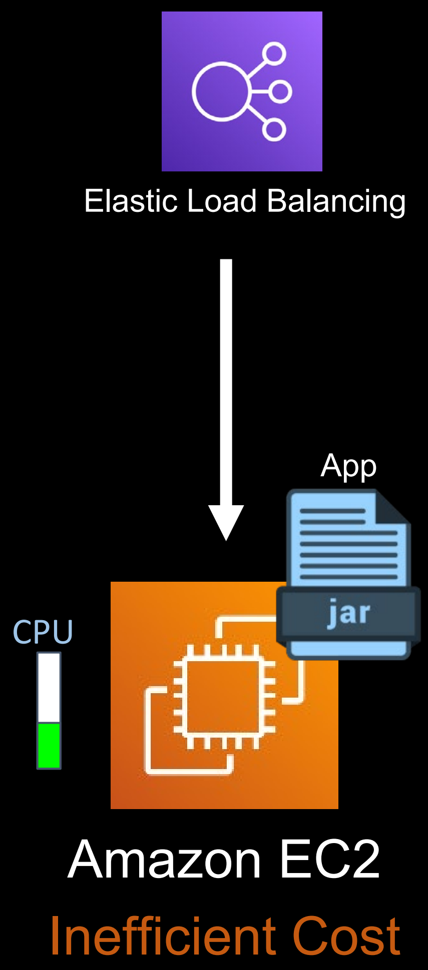


# EC2 Challenges – AMI Management

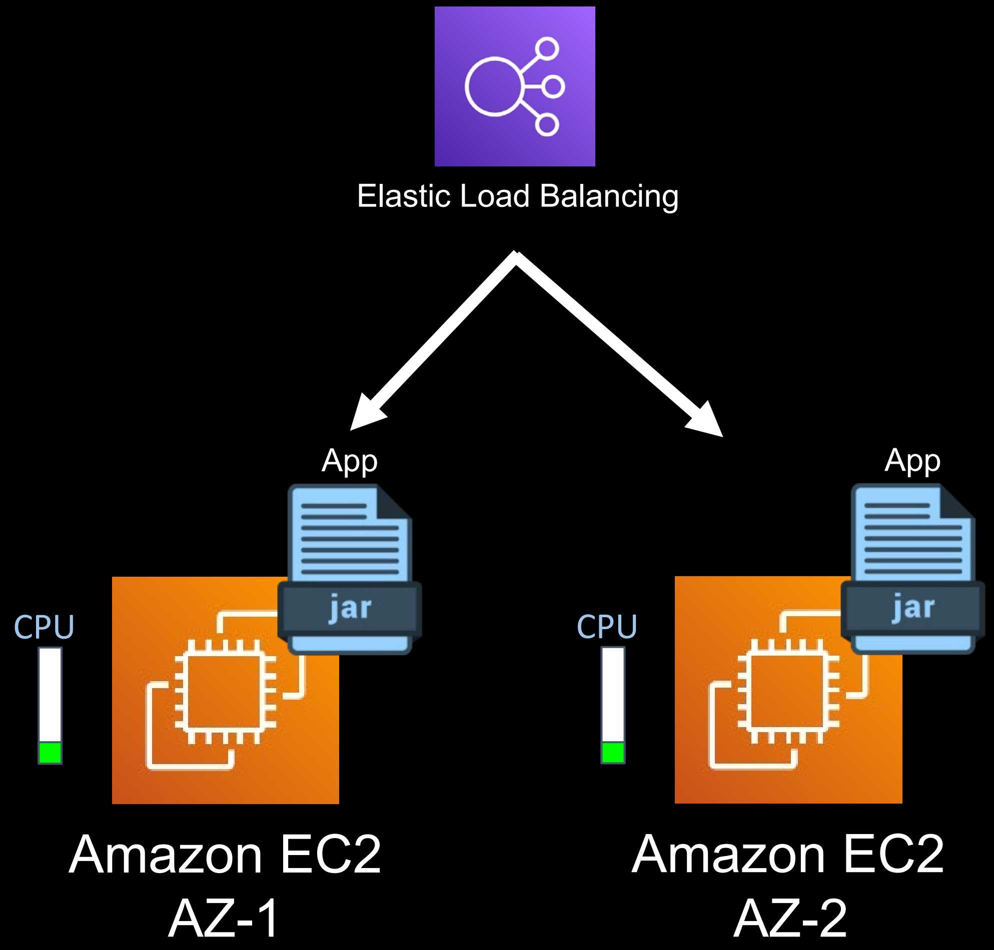


# EC2 Challenges – Cost Inefficiency



# EC2 Challenges – HA Overhead

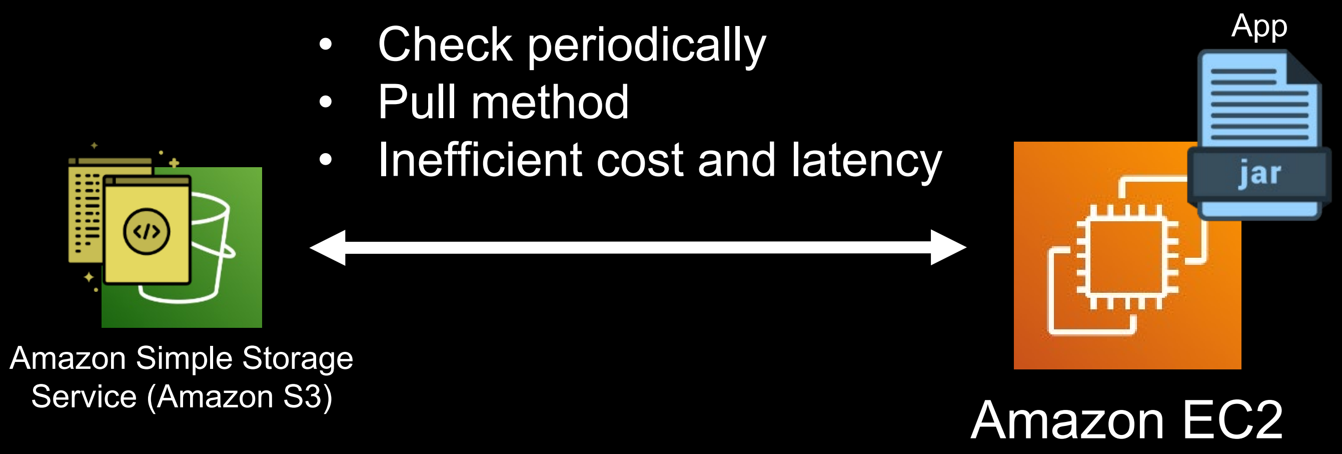


# Common Requirement

Amazon Simple Storage Service (Amazon S3)

Process file
(Run some code)

# Common Requirement



# That’s How Lambda Was Born!!

- Automatically triggered
- Run code
- Pay only when executed
- NO SERVER

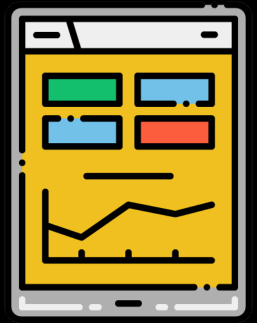
Amazon Simple Storage Service (Amazon S3)

AWS Lambda

AWS Lambda

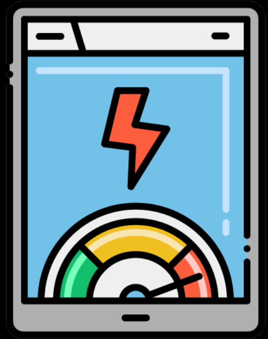
# Serverless

No servers to provision and manage



Automatically scales with usage

Never pay for idle



Automatically highly available

# Lambda Is NOT The Only Serverless Service

## COMPUTE

AWS Lambda

## DATA STORES

Amazon S3

Amazon Aurora Serverless

Amazon DynamoDB

## INTEGRATION

Amazon API Gateway

Amazon SQS

Amazon SNS

AWS Step Functions

AWS AppSync

AWS Lambda

- Run code without provisioning or managing servers
- Automatically scales without any scaling configuration
- Pay per use
- Highly available and fault tolerant
- Harder to visualize!
    - Demo will make it clear!

# AWS Lambda – Demo

Raj Saha
cloudwithraj.com
Cloud With Raj

# AWS Lambda – Pros and Cons

Raj Saha
cloudwithraj.com
Cloud With Raj

AWS Lambda

## Advantages

- Serverless features
  - No server!
  - Scales automatically
  - Pay per use
  - Inherently HA
- Integration with numerous AWS services
- Allows focusing on code, increases agility
- Many use cases and code samples available

## Challenges

- Max 15 min runtime
- Can’t select underlying infra for specialized workloads
- Code refactoring needed
- Cold start
  - Use provisioned concurrency (pay extra)
- Can’t be invoked from NLB

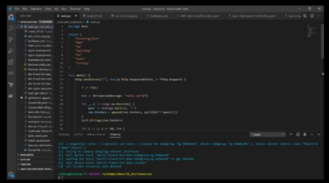
# Containers – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

# In The Beginning

Everything working great! I am genius!

## Environment: Dev



Code

Runtime Engine: Python 3.8

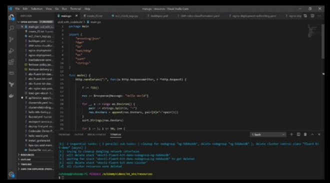
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Test



Code

Runtime Engine: Python 3.6

*import requests*
*import kitchen-sink*

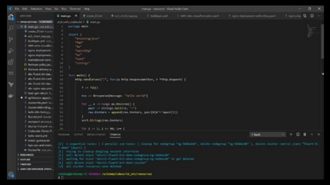
Dependencies

DNS Service Name
Database connection

Configuration

# In The Beginning


## Environment: Dev



Code

Runtime Engine: Python 3.8

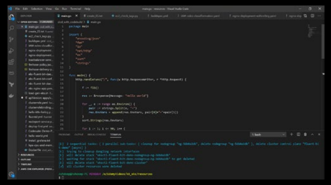
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Test



Code

Runtime Engine: Python 3.6

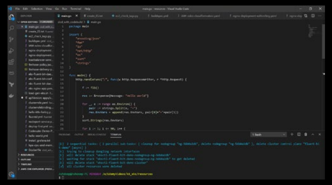
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Prod



Code

Runtime Engine: Python 2.7

*import requests*
*import kitchen-sink*

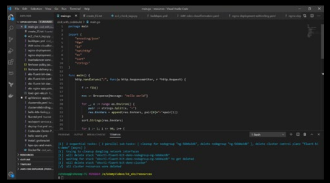
Dependencies

DNS Service Name
Database connection

Configuration

# In The Beginning


## Environment: Dev



Code

Runtime Engine: Python 3.8

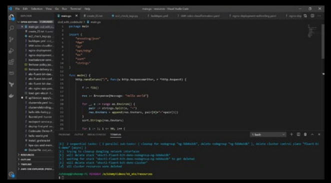
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Test



Code

Runtime Engine: Python 3.6

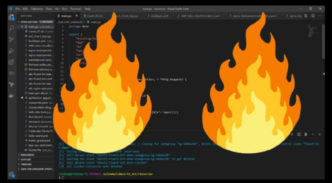
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Prod



Code

Runtime Engine: Python 2.7

*import requests*
*import kitchen-sink*

Dependencies

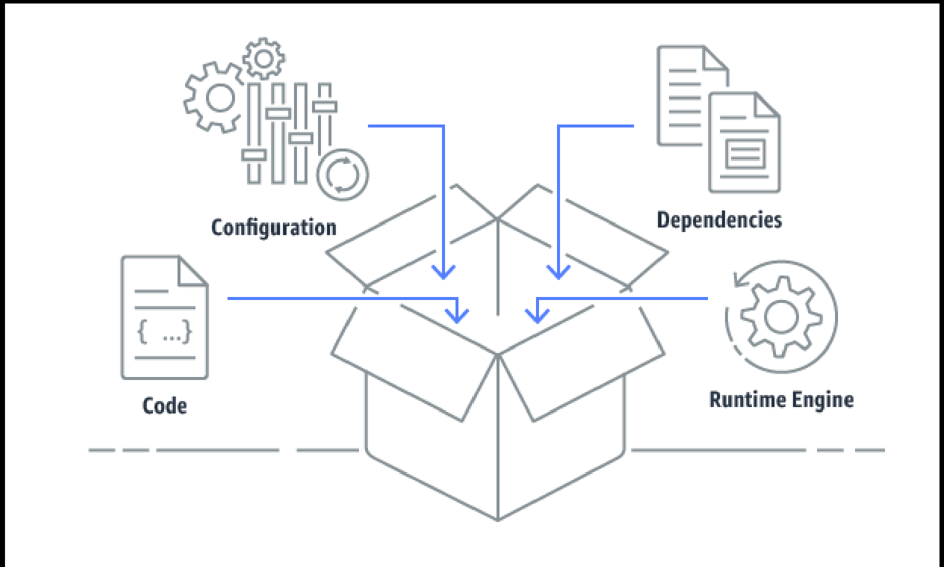
DNS Service Name
Database connection

Configuration


WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

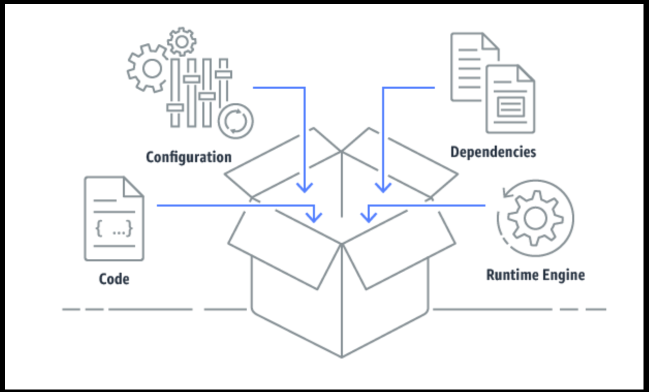

IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN
imgflip.com

# Container

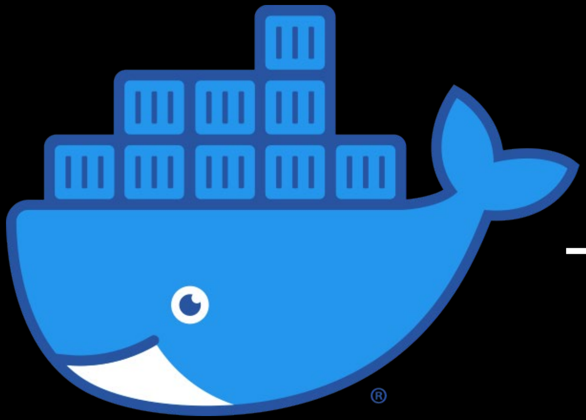



A container is an atomic, self contained package of software that includes everything it needs to run (code, runtime, libraries, packages, etc.)

# Docker Image Vs Container

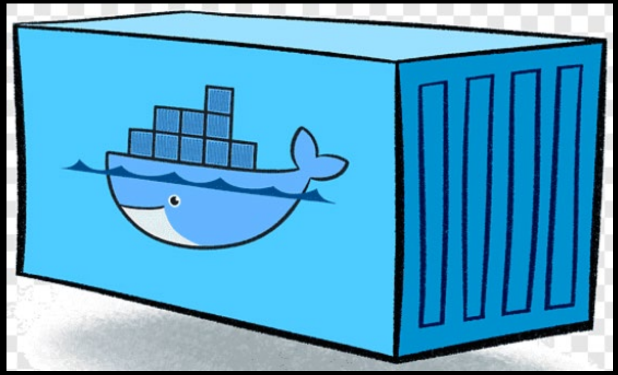



Build



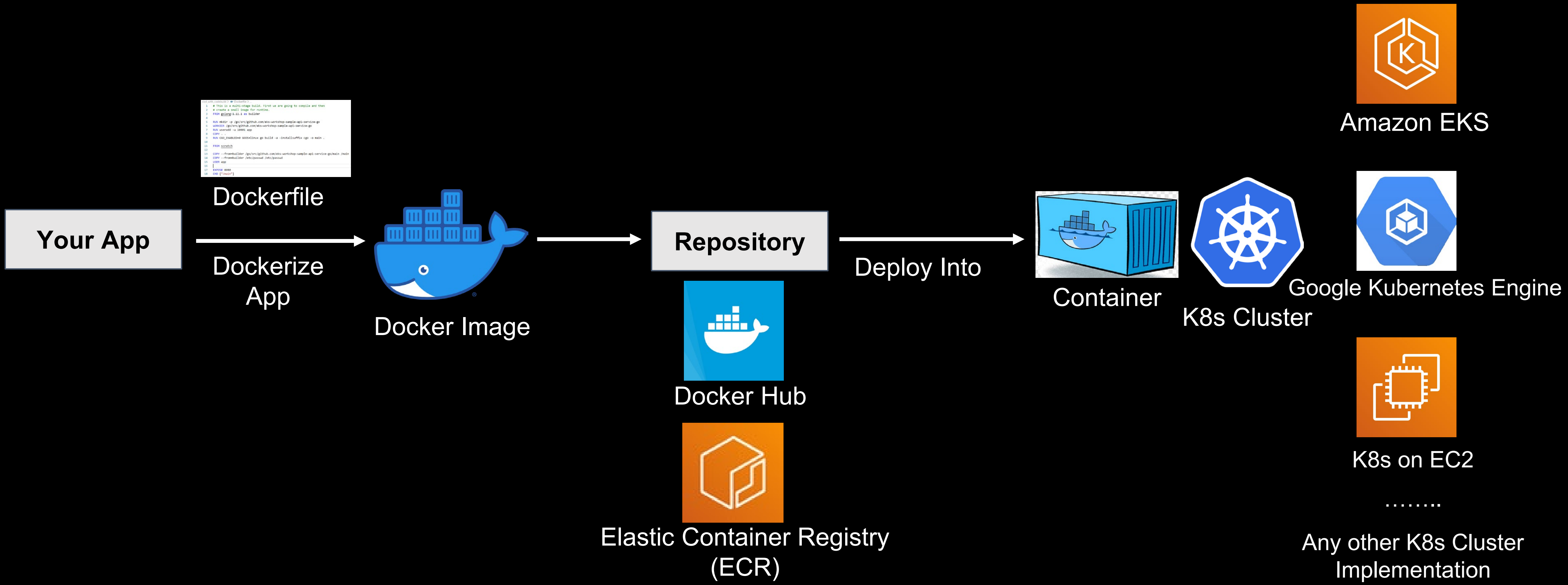
Docker Image
Container Image

Run

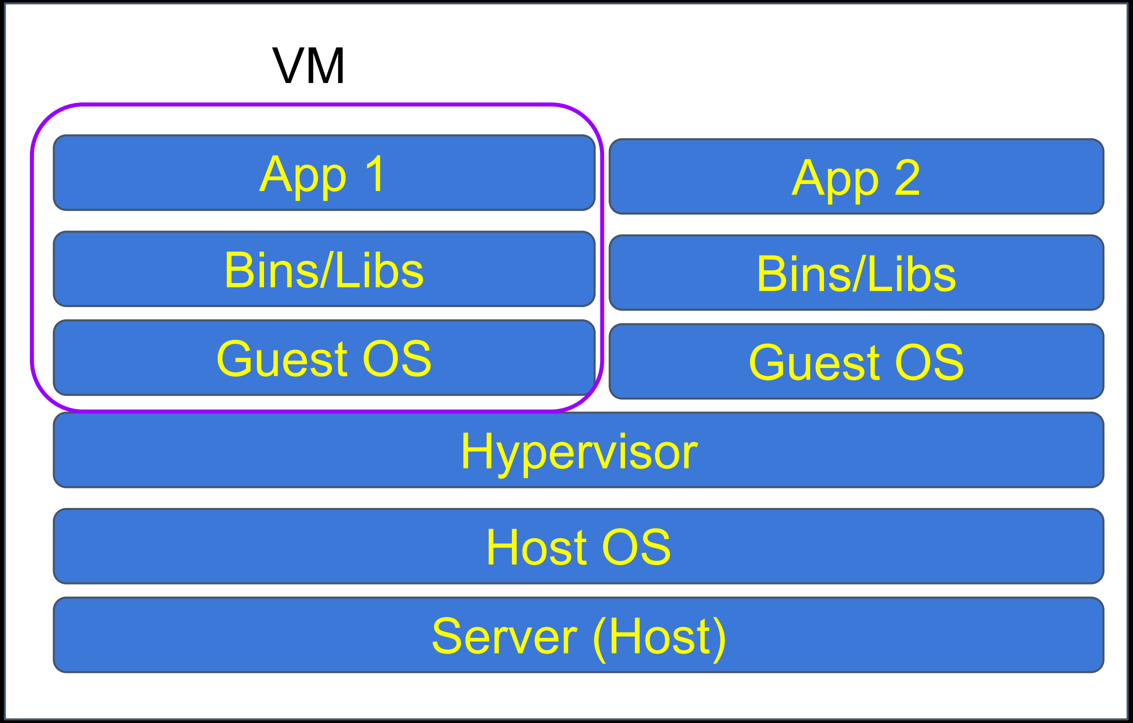


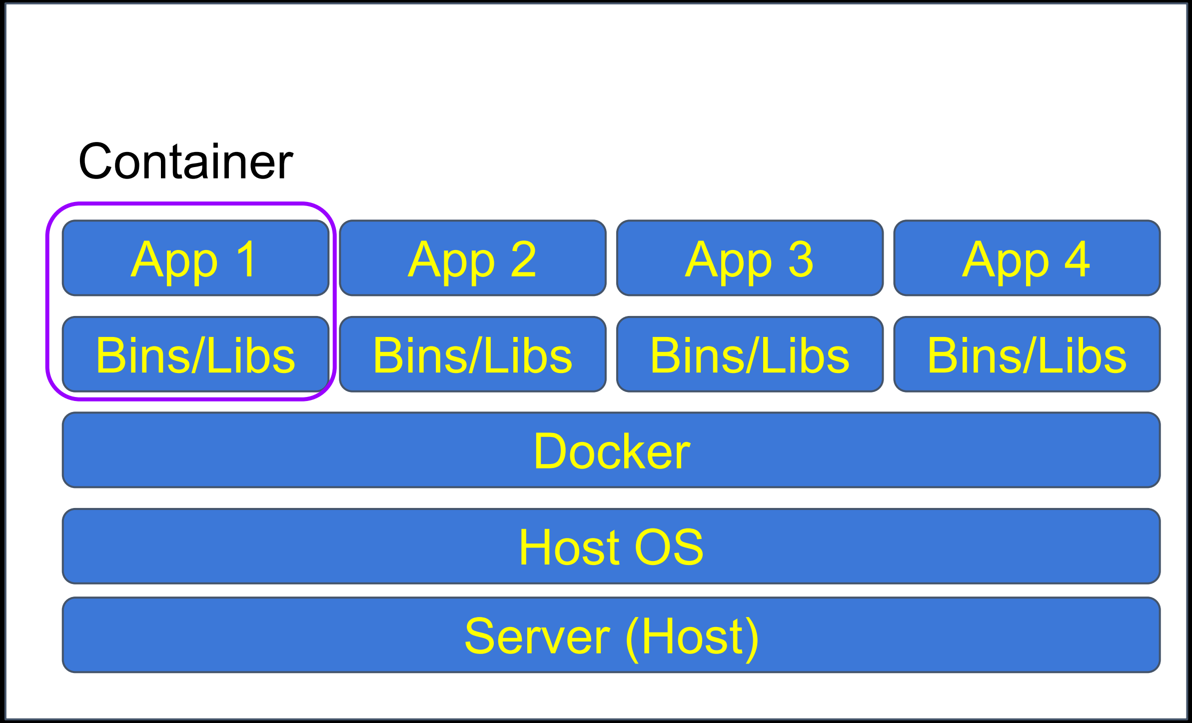
Container

# The Big Picture

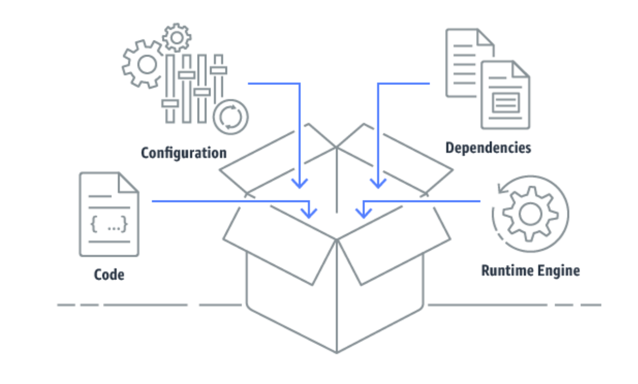


# Virtual Machine Vs Container



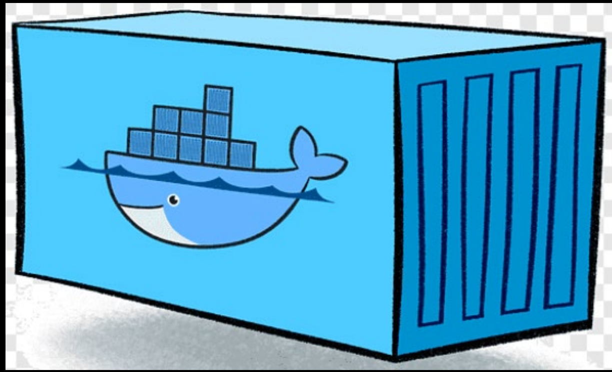
# Advantages




Build →

Docker Image

Run →

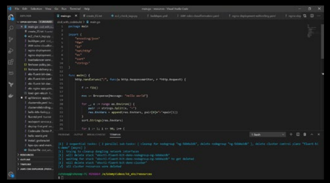


Container

Runs reliably in any environment

# Going Back to Our Sad Developer


## Environment: Dev



Code

Runtime Engine: Python 3.8

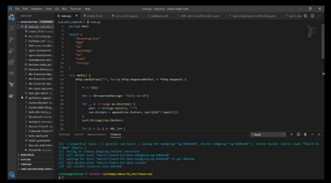
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Test



Code

Runtime Engine: Python 3.6

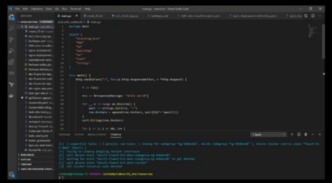
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

## Environment: Prod



Code

Runtime Engine: Python 2.7

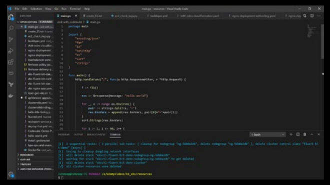
*import requests*
*import kitchen-sink*

Dependencies

DNS Service Name
Database connection

Configuration

# Going Back to Our Sad Developer

Code

Runtime Engine: Python 3.8

*import requests*
*import kitchen-sink*

Dependencies

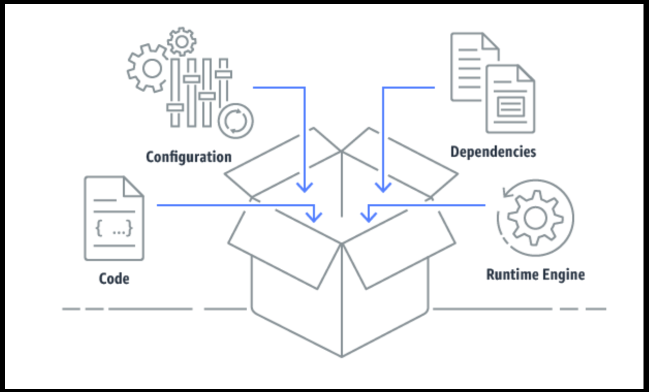
DNS Service Name
Database connection

Configuration

Docker Image

Runs seamlessly in Dev, Test, Prod

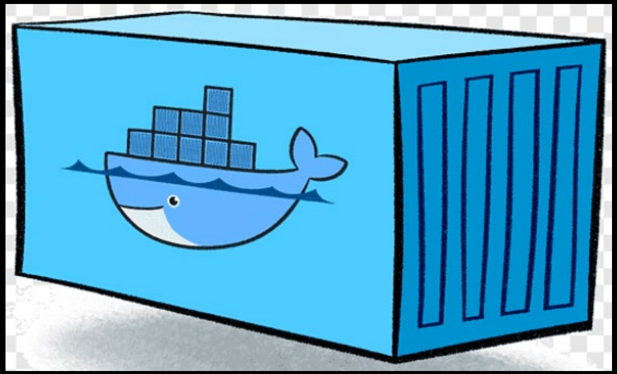
# Advantages




Build

Docker Image

Run



Container

Runs reliably in any environment

Better resource utilization

App isolation

Speed

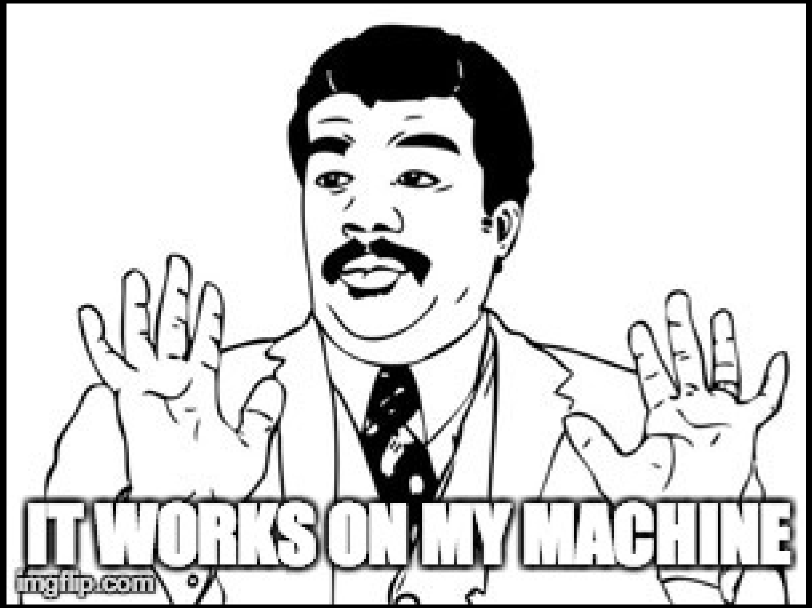
Container Orchestration is SOLVED!

# Container Orchestrator

Raj Saha
cloudwithraj.com
Cloud With Raj

# What is Docker/Container?

- Docker packages software into standardized units called containers that have everything your software needs to run including libraries, code and runtime
- Lets you quickly deploy and scale applications into any environment



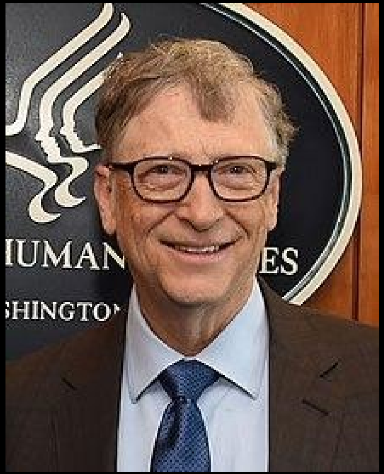
# What is Container Orchestrator?

# How Does Docker Work?

# Tasks Associated with Containers

- Deployment of Containers
- Redundancy and availability of Containers
- Scaling up or down of Containers
- Load Balancing
- Health Monitoring of Containers and Hosts
- Service Discovery
- And More...

# Container Orchestrator



# Say Hello to Container Orchestrators

- Docker Swarm
- Apache Mesos
- Cattle, Nomad, Empire
- AWS ECS (Elastic Container Service)
- Kubernetes
    - EKS (Elastic Container Service for Kubernetes)
- AWS Fargate

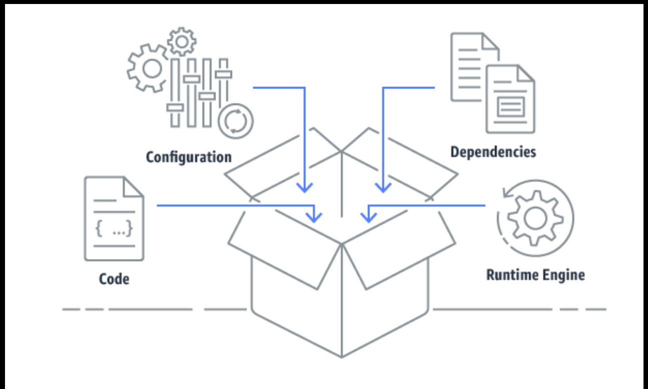
# Kubernetes – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

# What is Kubernetes?

(The most popular Container Orchestrator)

# Working Backwards

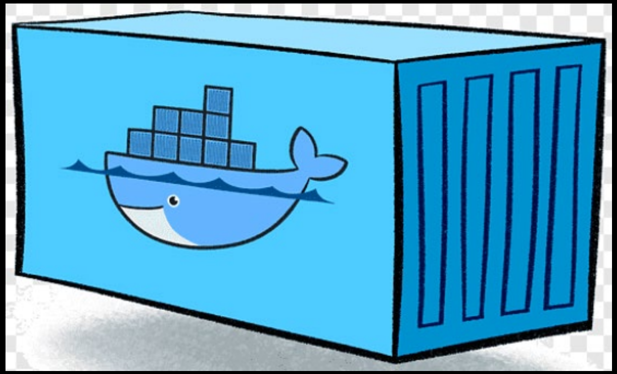



Your Application

Build

Docker Image
Container Image

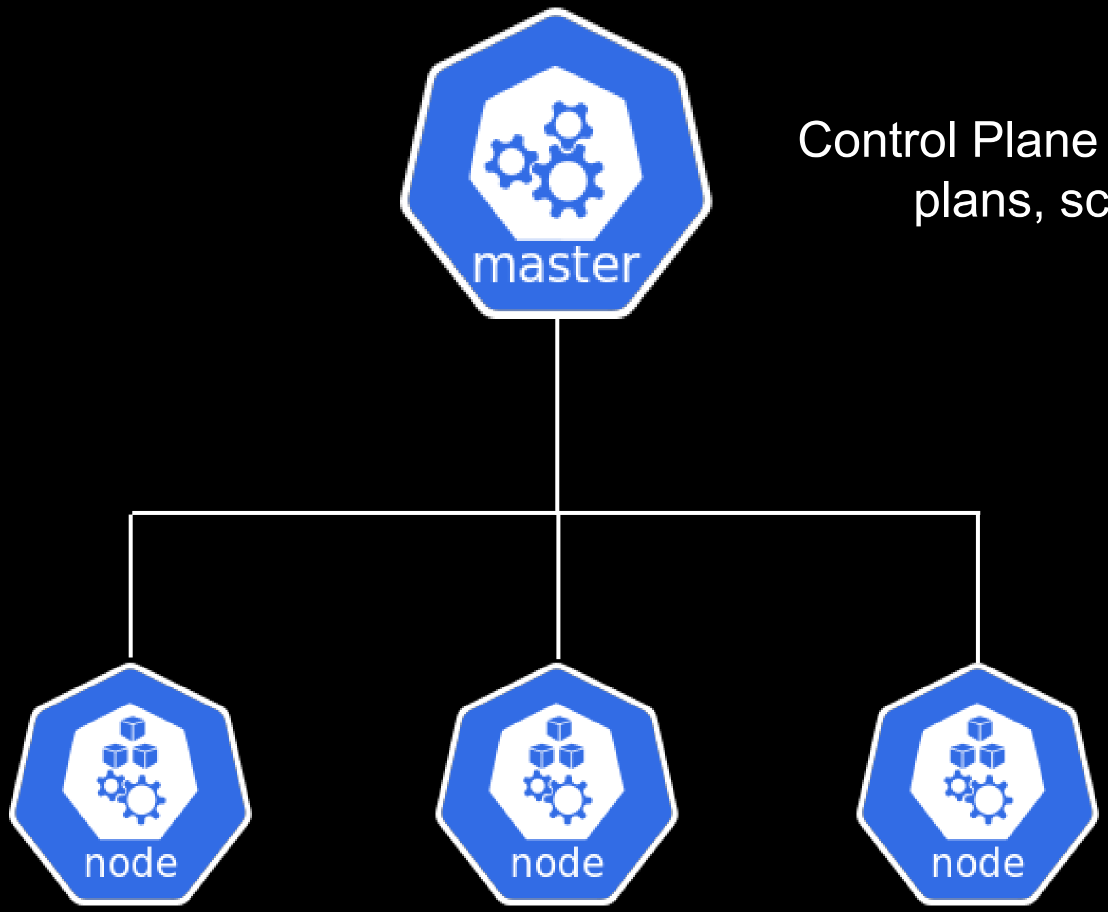
Run



Container

# Kubernetes Architecture

# Who Manages Nodes?




Control Plane manages, monitors, plans, schedules nodes

Worker Nodes host Containers

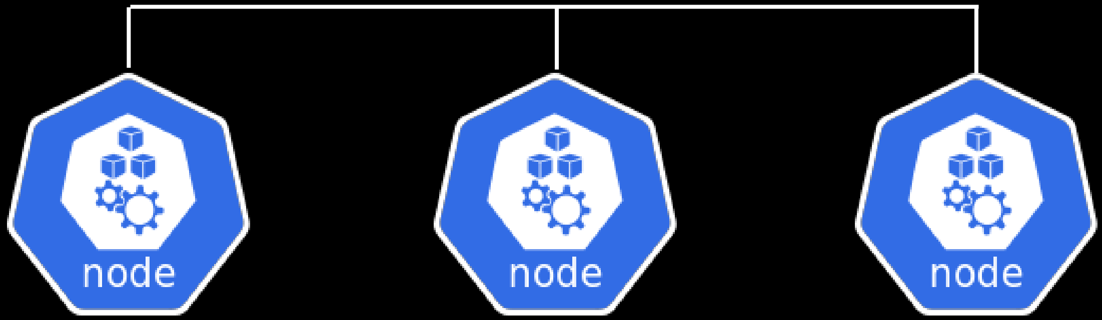
# Control Plane Components


Control Plane manages, monitors, plans, schedules nodes


Key Value Store for critical cluster info

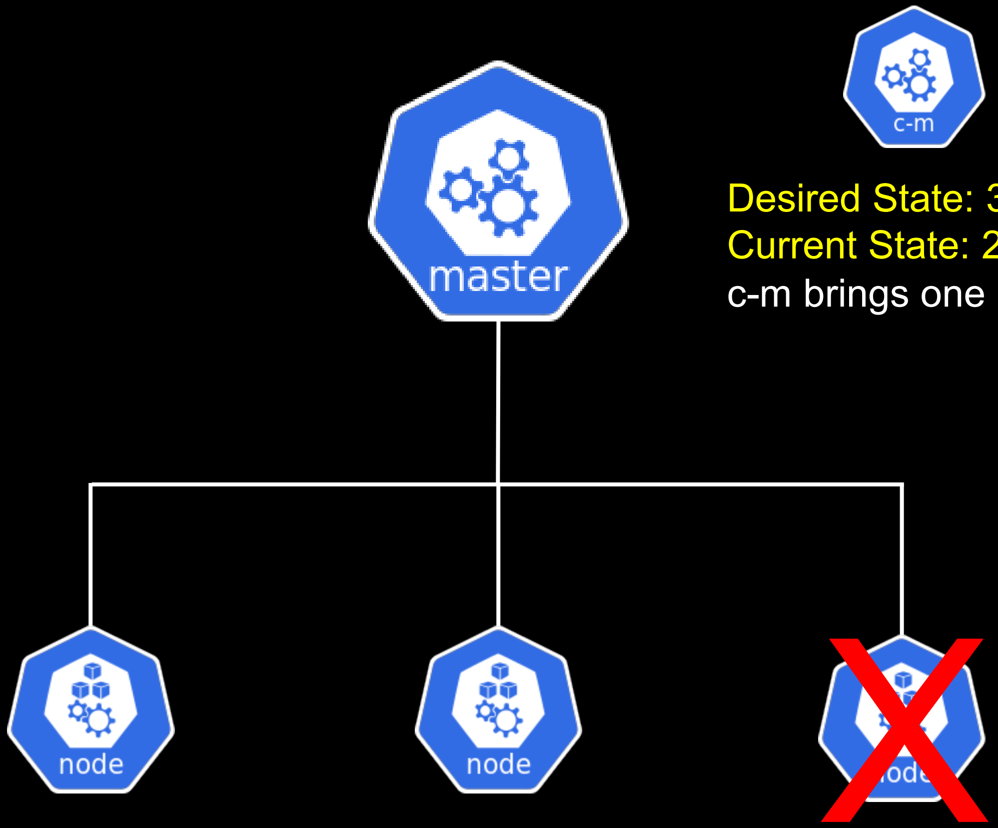

Ensures proper state of cluster components


Puts containers to proper nodes



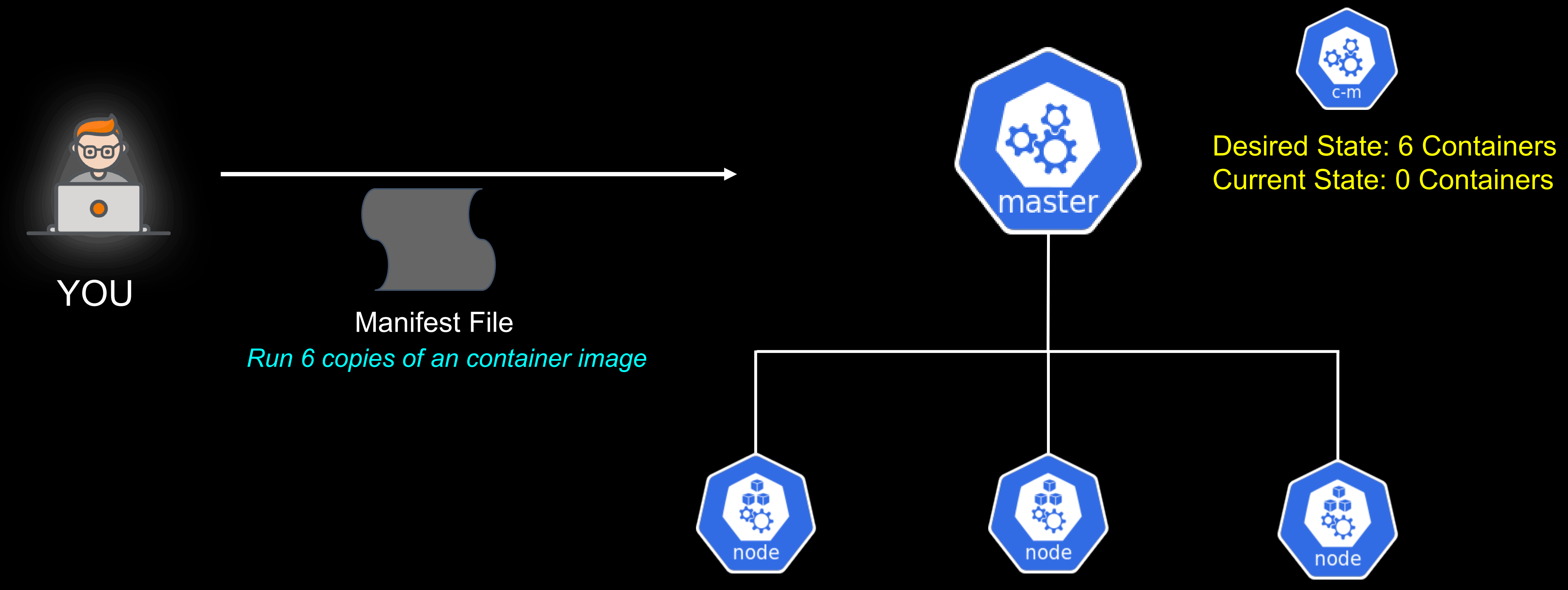

Worker Nodes host Containers

# Kubernetes Cluster State

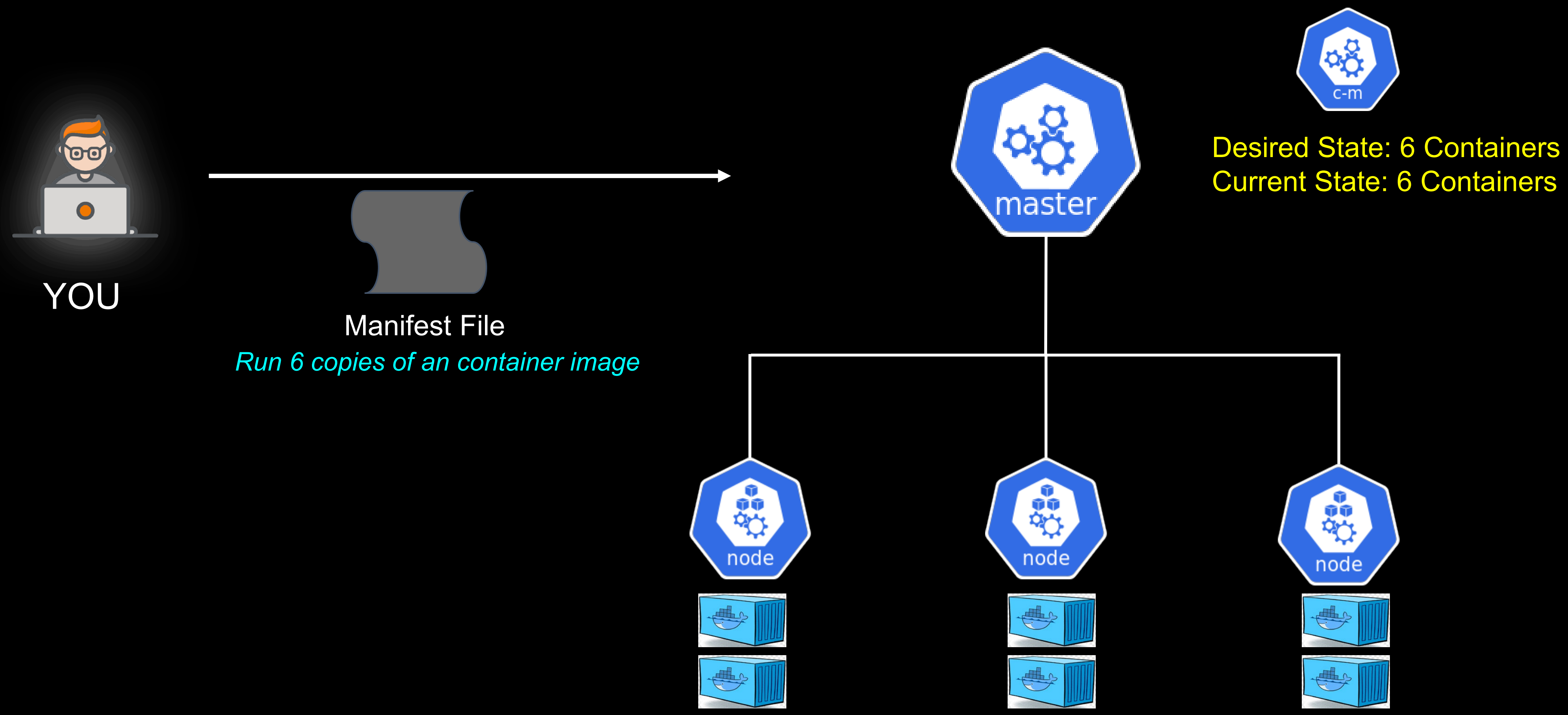



Desired State: 3 Nodes
Current State: 2 Nodes
c-m brings one node up so current state = desired state

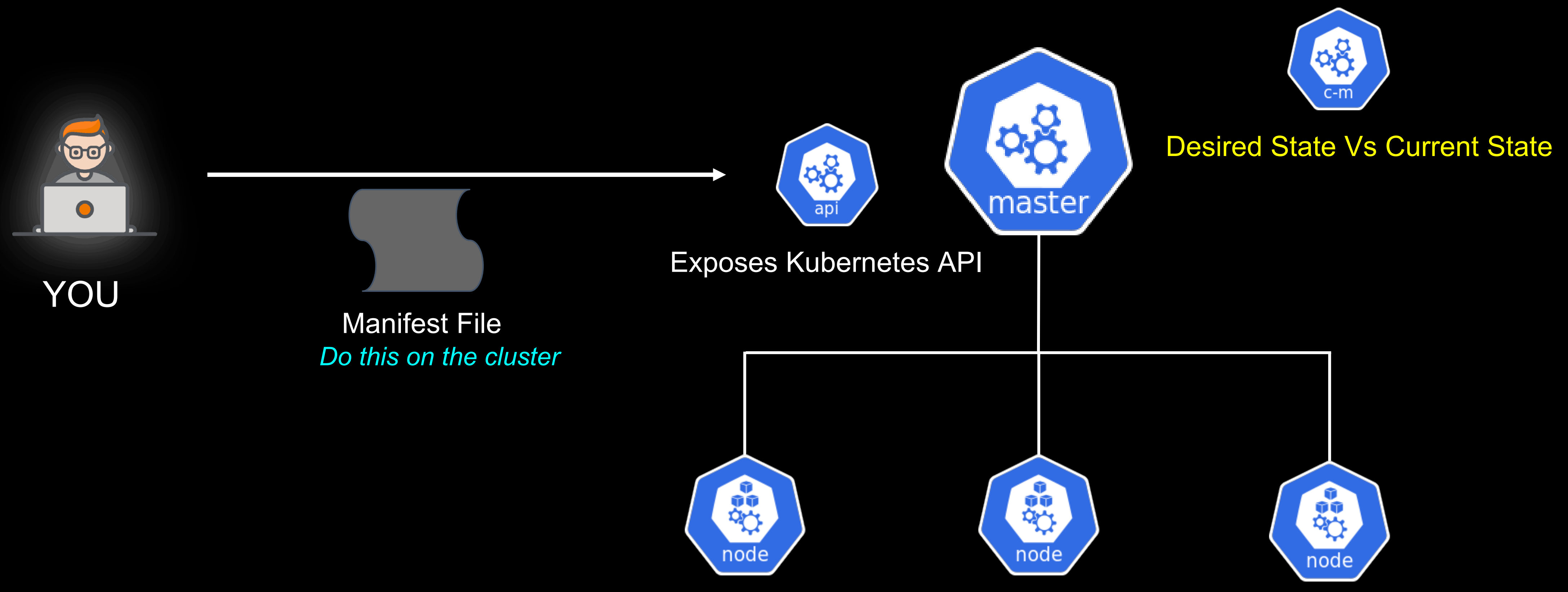
# Who Specifies State?



# Who Specifies State?



# Gateway to Control Plane



# Kubernetes Architecture

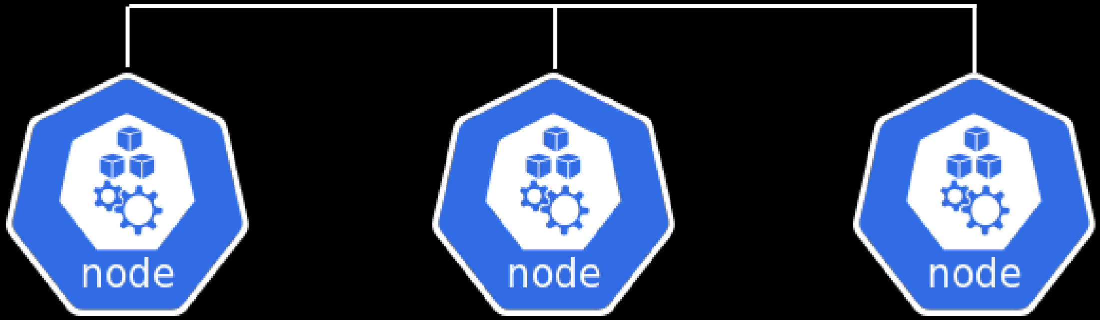

Control Plane manages, monitors, plans, schedules nodes


Key Value Store for critical cluster info


Ensures proper state of cluster components


Puts containers to proper nodes


Exposes Kubernetes API

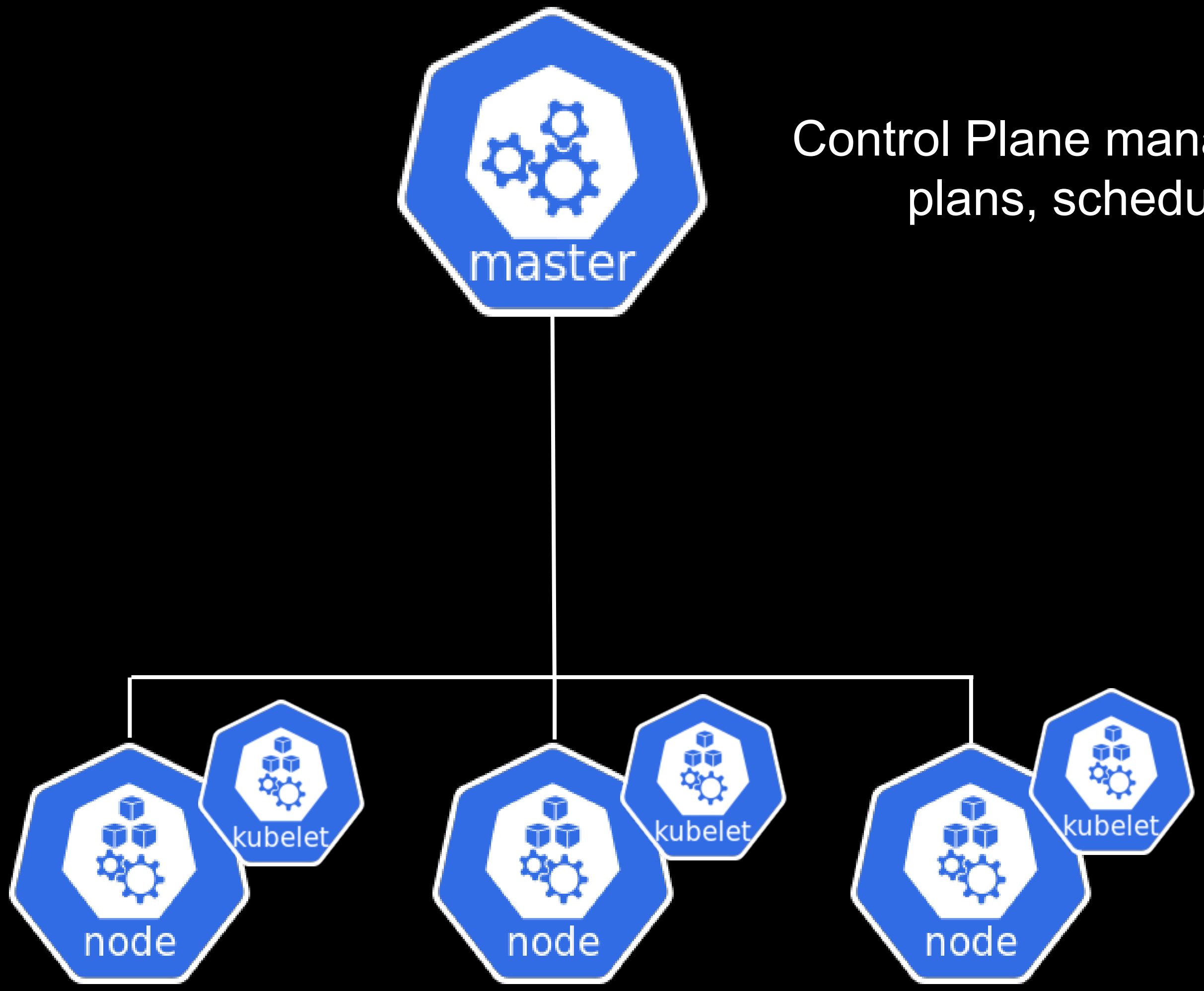



Worker Nodes host Containers

# What's In Node?

Container Runtime Engine
Docker, Containerd, CRI-O, frakti

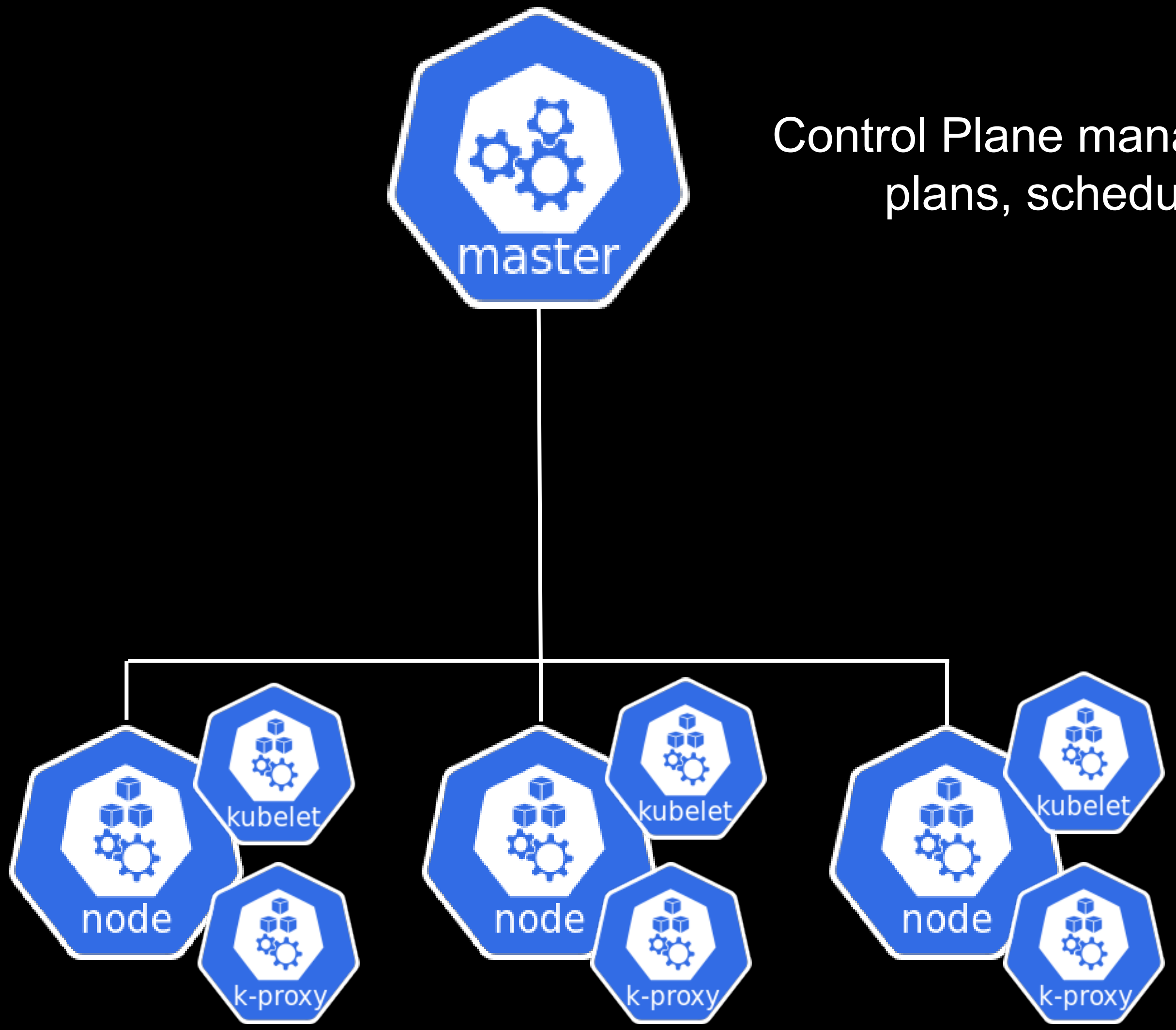
# Control Plane - Node Communication




Control Plane manages, monitors, plans, schedules nodes

Status of the node and containers to Control Plane

# Container-Container Communication

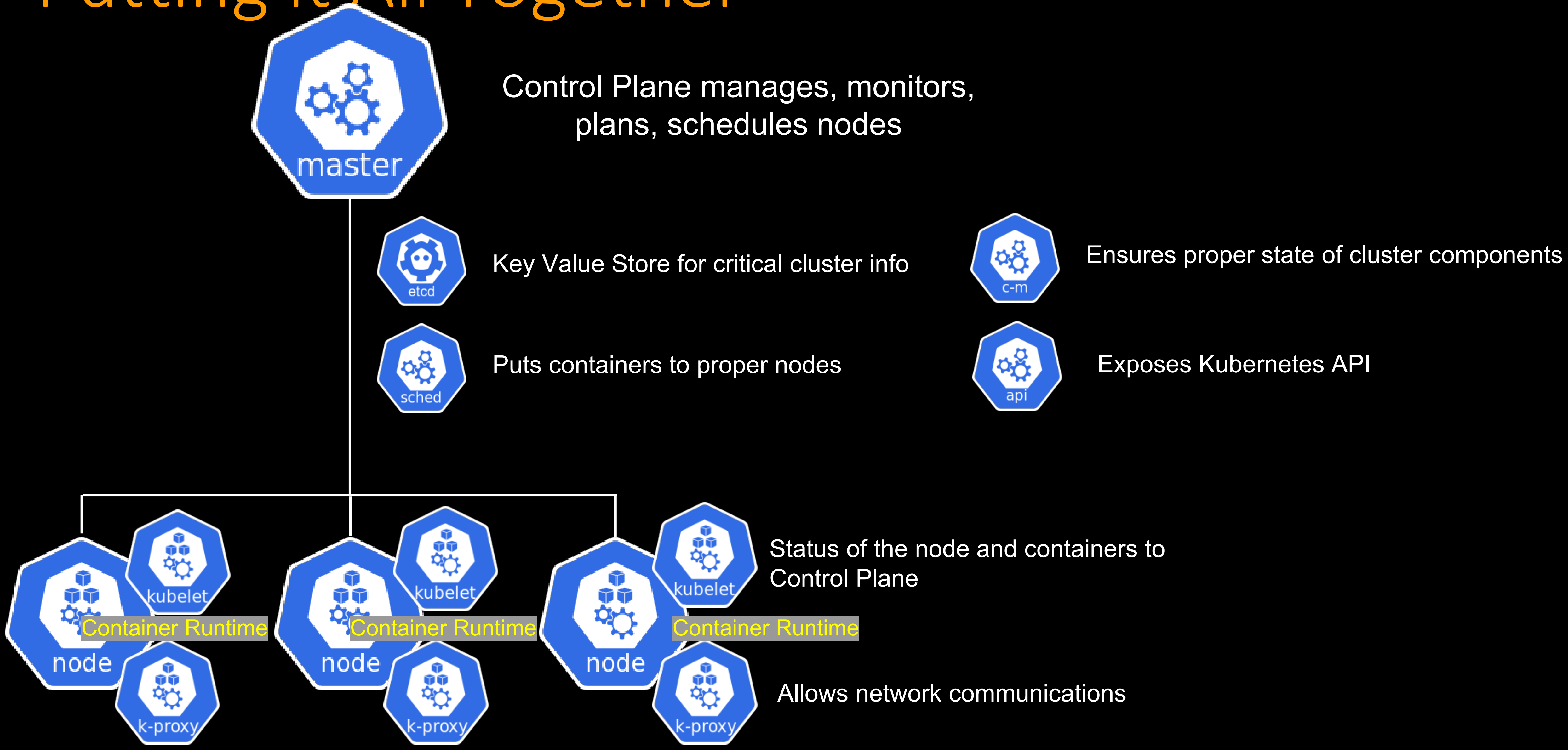



Control Plane manages, monitors, plans, schedules nodes

Status of the node and containers to Control Plane

Allows network communications

# Putting it All Together



# K8s?!

Kubernetes

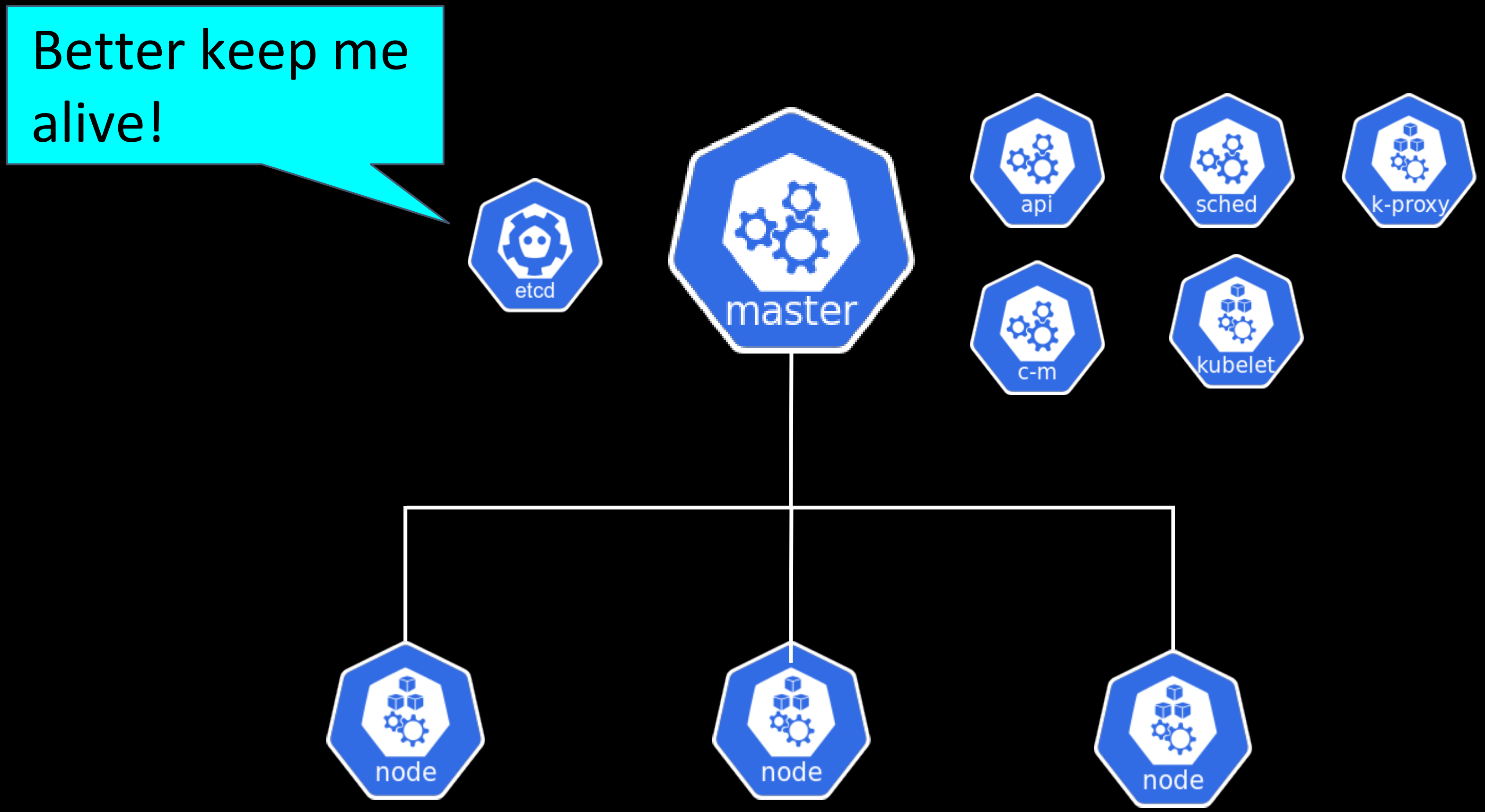
8 Letters

K8s

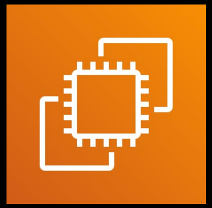
# Amazon EKS and ECS – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

# Kubernetes Architecture



# Kubernetes Control Plane - Self Managed

Amazon EC2

- Need to make Control Plane Highly Available
  - Maintain multiple EC2 in multiple AZ
- Scale Control Plane if needed
- Keep etcd up and running
- Overhead of managing EC2s
  - AMI Rehydration
  - Security Patching
  - Replace failed EC2s
  - Orchestration for Kubernetes Version Upgrade

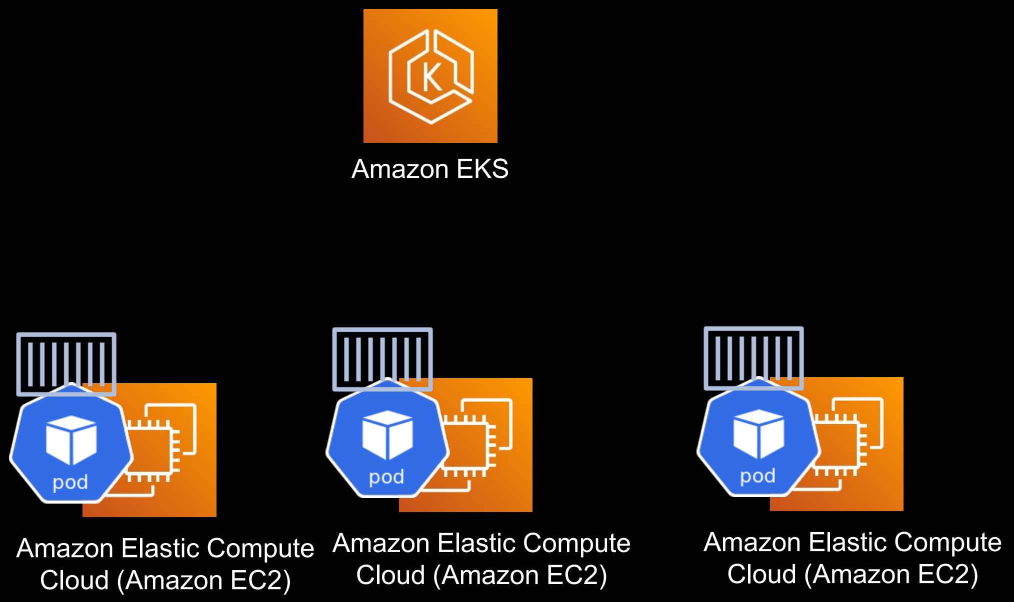
# Kubernetes Control Plane - AWS Managed

Amazon Elastic
Kubernetes Service

AWS Manages Kubernetes Control Plane

- AWS maintains High Availability - Multiple EC2s in Multiple AZs
- AWS Detects and Replaces Unhealthy Control Plane Instances
- AWS Scales Control Plane
- AWS Maintains etcd
- Provides Automated Version Upgrade and Patching
- Supports Native and Upstream Kubernetes
- Integrated with AWS Ecosystem
- Costs 10 cents/hour + worker plane cost

# Amazon EKS Data Plane



# Kubernetes on AWS

63%

of Kubernetes workloads run on AWS today

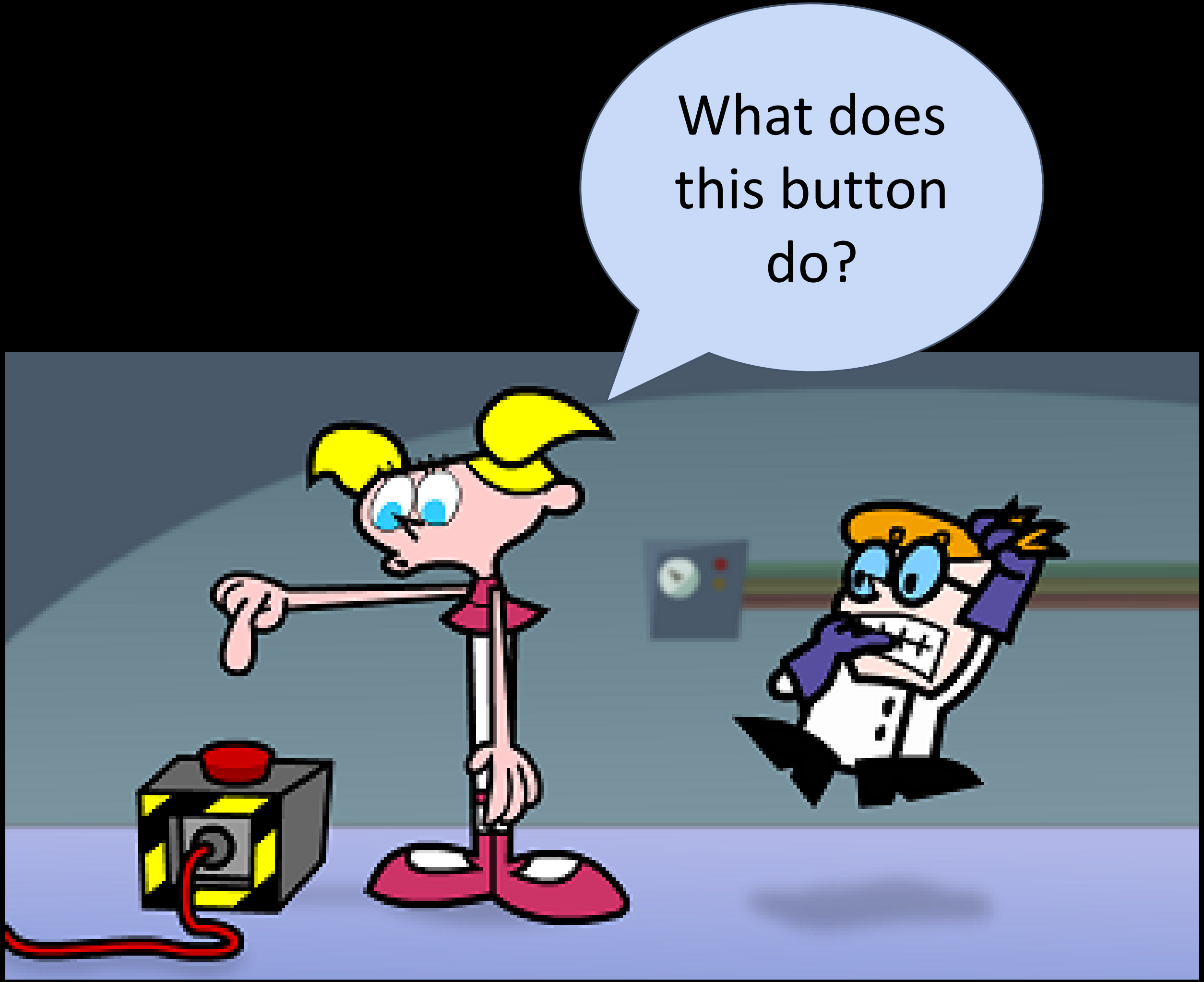
- CNCF

# Amazon EKS Customers

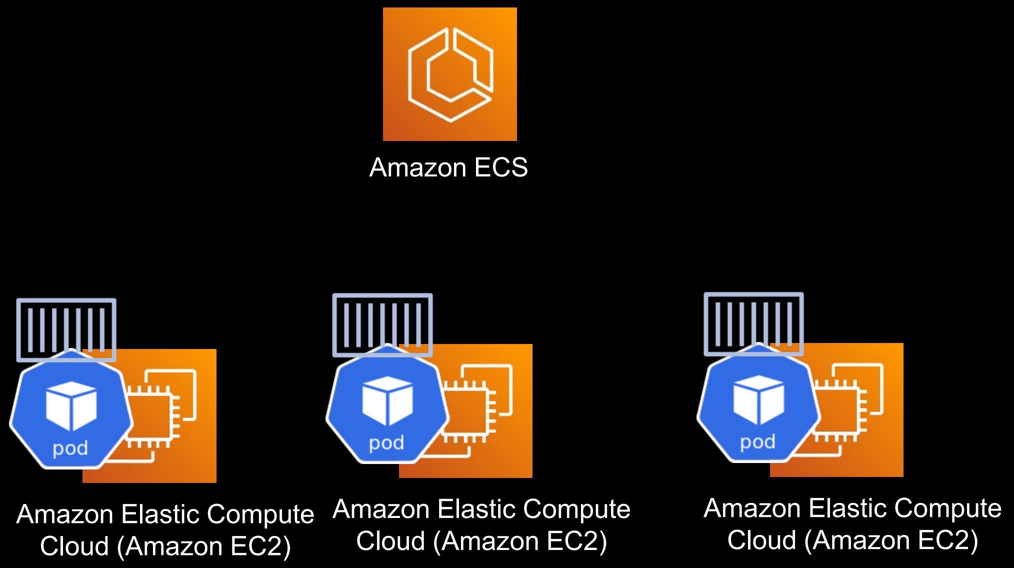
And Many More..

Source: https://aws.amazon.com/eks/customers/

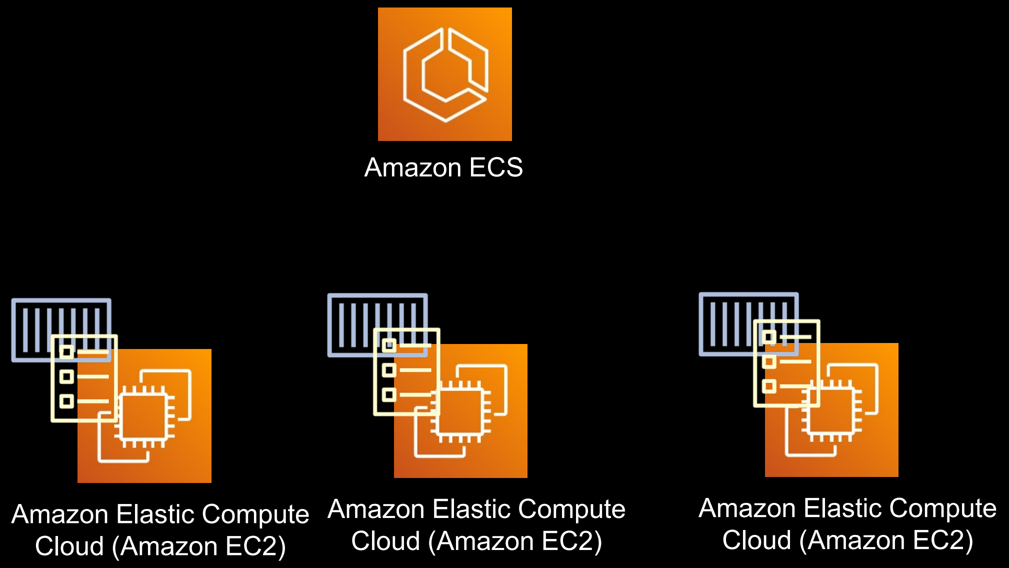
# Kubernetes is complex!



# Amazon ECS (Elastic Container Service)



# Amazon ECS (Elastic Container Service)

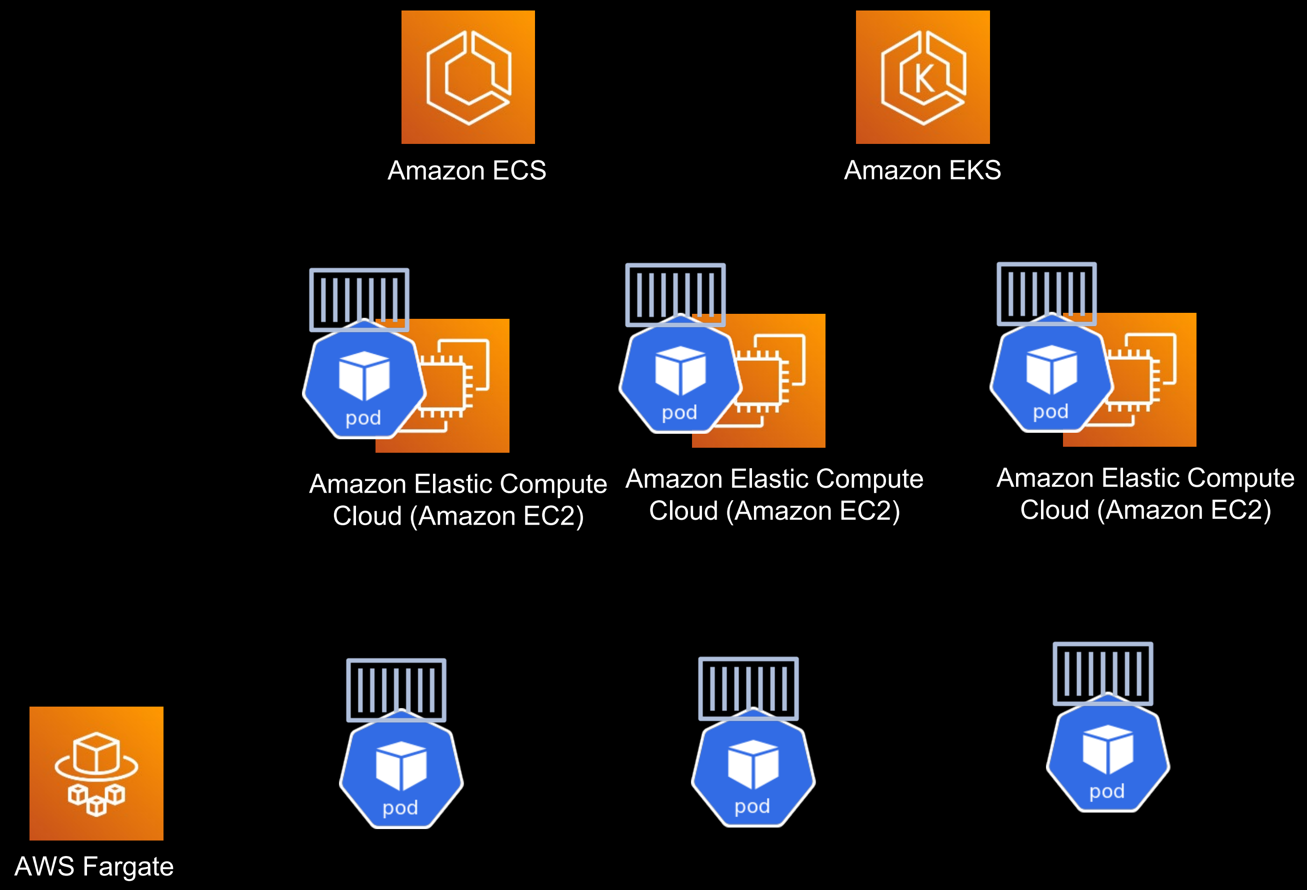


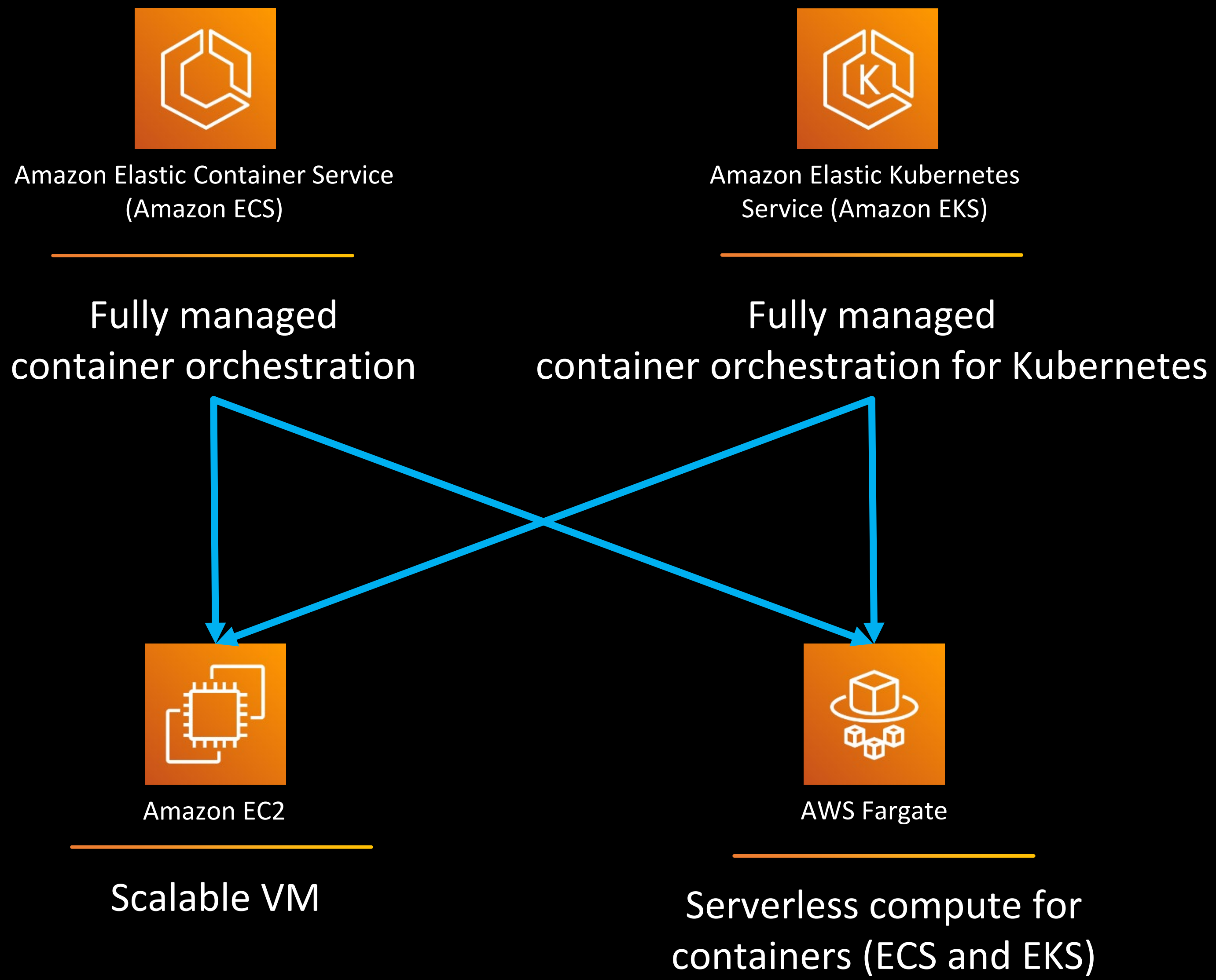
# Amazon ECS (Elastic Container Service)

Amazon
ECS

- AWS’s native container orchestrator
- Leverage existing AWS knowledge to run your container
  - ASG for scaling, IAM/KMS for security, CloudWatch for observability etc.
  - Can integrate with some open-source products
- Many large enterprises are using ECS
  - Affirm, Autodesk, Volkswagen etc.
- ECS control plane is free
- Anecdote from customer conversation and interview tip

# AWS Fargate


Amazon ECS
Amazon EKS
pod
Amazon Elastic Compute Cloud (Amazon EC2)
pod
Amazon Elastic Compute Cloud (Amazon EC2)
pod
Amazon Elastic Compute Cloud (Amazon EC2)
AWS Fargate
pod
pod
pod


Amazon Elastic Container Service (Amazon ECS)
Fully managed container orchestration
Amazon Elastic Kubernetes Service (Amazon EKS)
Fully managed container orchestration for Kubernetes
Amazon EC2
Scalable VM
AWS Fargate
Serverless compute for containers (ECS and EKS)

# In summary

### AWS Fargate

**Serverless**

- No servers to manage
- Pay for resources only when used
- Fargate supports both EKS and ECS

### Amazon ECS

**Powerful simplicity**

- For container-based workloads on AWS
- Opinionated solution for containers
- Reduced time to build and deploy
- Fewer decisions needed

### Amazon EKS

**Open flexibility**

- If you are invested in Kubernetes
- Vibrant ecosystem and community
- Consistent open-source APIs
- Consider resource upskill time

# Do Freshers Need to Learn Kubernetes?

Raj Saha
cloudwithraj.com
Cloud With Raj

# Kubernetes Areas

- Kubernetes Fundamentals
- Observability
- Security
- Networking
- Managed Kubernetes Service from cloud providers
- DevOps
- Service Mesh and API Gateways
- Day 2 Ops

Kubernetes Fundamentals

↓

Observability

↓

Security

↓

Networking

↓

Managed Kubernetes Service from cloud providers

↓

DevOps

↓

Service Mesh and API Gateways

↓

Day 2 Ops

# How About Docker?

- No need to learn separately
- I learnt ‘as needed’ basis while learning Kubernetes

# DO YOU NEED TO KNOW ALL OF THESE!!??

NO

# How About Tools?

- Not all tools are important
- I will call out important tools
- Some tools are useless for interviews

# Kubernetes Fundamentals - Basic

- What is container?
- What is container orchestrator?
- Dockerfile
    - Multi-stage build
    - Docker container runtime
- What is Kubernetes?
- Pods
- Deployment
- Replicaset
- Services
- Daemonset
- Write podspec file
- Namespace
- Popular Kubectl commands

# Kubernetes Fundamentals - Advanced

- Ingress
- Service vs Ingress
- Scaling
  - Metrics server
  - Horizontal Pod Autoscaler (HPA)
  - Cluster Autoscaler (CA)
  - Vertical Pod Autoscaler (VPA)
  - Overprovisioner
  - Custom metrics (Keda)
- Availability
  - Taints and tolerations
  - Liveness, readiness, and startup probes
- Cost optimization
  - Kubecost
- Statefulsets
- If using EKS (AWS)
  - Managed node group
  - Fargate
  - Karpenter

# Kubernetes Observability

- Observability
  - Logs
    - Control plane logs
    - Container logs
      - fluentbit, fluentd
      - Container insights (AWS)
  - Metrics
    - Control plane metrics
    - Data plane metrics
      - Prometheus, Grafana
      - Container insights (AWS)
  - Traces
    - Container
      - X-Ray (AWS)

# Kubernetes Security

- Role Based Access Control (RBAC)
- Service Account
- Role and Clusterrole
- Rolebinding and Clusterrolebinding
- Kubeconfig, Configmap, Secrets
- How different app and users can have different access?
- AuthN/Z with API Gateway and Ingress
- Open Policy Agent (OPA)
- Security best practices
  - Infrastructure, image, pod, network, incident response
  - kube-bench, sysdig/twistlock, inspector (if AWS)
- If using EKS (AWS)
  - IAM Role for Service Account (IRSA)
  - Pod Security Group
  - aws-iam-authenticator

# Kubernetes Networking

- How IP addresses are allocated
- CNI Plugins
  - AWS VPC CNI (If EKS)
- Network policy
  - Calico
- IPv6
  - Why?
  - How does it work with IPv4
  - How is it different than adding secondary subnet in IPv4
- Core DNS
- Ingress
- Private cluster (Public + Private, Private)

# Managed Kubernetes Service from Cloud Providers

- Pick one!
  - EKS (AWS)
  - AKS (Azure)
  - GKE (GCP)
- Advantages of managed K8s service
  - Managed control plane
  - Integration with other cloud services
    - Storage, workflow, security
  - For EKS (AWS)
    - Managed node group, Spot, VPC-CNI, managed addons, IPv6 support, prefix support, AWS controller, container insights, managed Prometheus & Grafana, secure AMI, free ECR scanning, eksctl etc.
    - EBS/EFS/FSx integration, Step Function, IAM, Inspector & Security Hub
- Gaps of managed K8s service
  - Check open roadmap

# Kubernetes DevOps

- DevOps flow to deploy into K8s
- Container repo
  - DockerHub, ECR if AWS
- GitOps
  - ArgoCD, Flux (Pick one)
- Implementing into a cluster using one DevOps tool
  - Jenkins
  - Jenkins + GitOps
- Blue green, canary deployment (Service mesh)
- Operator and controllers (Only theory for rare interview)
- Cluster API (CAPI)

# Service Mesh and API Gateways

- Service mesh architecture
- Service mesh benefits
  - Encryption
- How to throttle, circuit break, retry
- Service mesh auth with third party
- Scale using service mesh metrics (Keda)
- Blue green, canary deployment (Service mesh)
- Service mesh products – Istio, Appmesh (pick one)
- API Gateway
  - Separate service to K8s
  - API Gateway running on K8s
    - Gloo, Kong, Mulesoft (pick one)

# Kubernetes Day2 Ops

- Patching
- Hardening AMI
- Upgrading K8s without downtime
- Deploying application without downtime
- Backup K8s cluster
- Common Day2 challenges
  - Running out of IP
  - Private cluster errors
  - AZ goes down

# IaaS, PaaS, SaaS (Cloud Computing Models)

Raj Saha
cloudwithraj.com
Cloud With Raj

# IaaS (Infrastructure as a Service)

Closest model to managing on-premises data center without the overhead of managing the building

# IaaS (Infrastructure as a Service)

- Provides with computing infrastructure
- Physical, virtual machines, block and file based storage, load balancers etc.
- Basic building block of IT
- Example – Amazon EC2, S3, EBS etc.
- You are in charge of resource procurement, configuring OS, software maintenance, patching

# PaaS (Platform as a Service)

- Provides with computing platform
- Abstraction on top of IaaS
- AWS manages the underlying infrastructure and operating systems
- Example – AWS Lambda, Fargate, App Runner, Elastic Beanstalk etc.
- You can focus more on deploying and managing your application

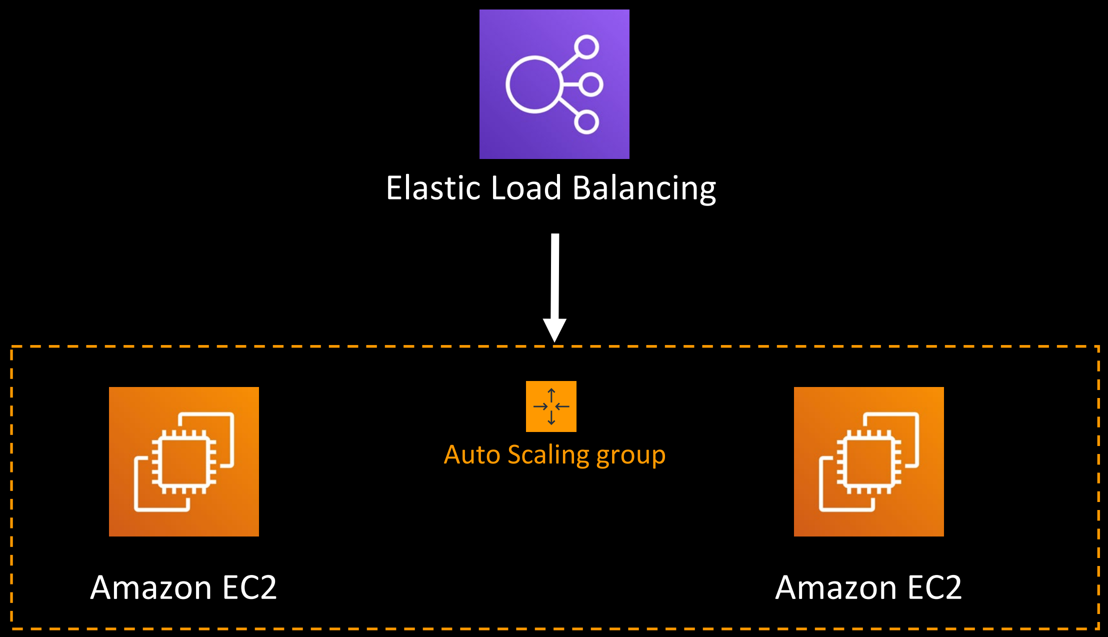
# SaaS (Software as a Service)

- Provides with platform software
- Abstraction on top of PaaS
- You access application software or on-demand software, and just use it
- Example – CRM, Email, Office 365, G-Suite, Amazon Chime etc.
- You do not manage the service, infrastructure, patching, scaling etc.
- AWS is chosen to build SaaS by the providers

# Amazon Elastic Beanstalk

Raj Saha
cloudwithraj.com
Cloud With Raj

# Creating a Traditional Web App



# Using Console

Creating VM and other infra is easy from console

I can do everything
from console!

Now you are tasked to build multiple environments

# Provisioning Webserver

Apache Tomcat - Java WebServer

# Provisioning Webserver with Cfn

- CloudFormation spins off the EC2s, Auto Scaling Group etc.
- You can SSH and install software in EC2s
- Changing instance types difficult
- Very manual and prone to error
- Super boring to do these for 100 servers

# Elastic Beanstalk to The Rescue!

AWS Elastic Beanstalk

- Deploy, monitor and scale quickly and easily
  - Choose from available blueprints, such as Tomcat, to spin up infra
  - Point your Java WAR file from git
  - Supports containers (Docker on EC2/ECS)
- No need to worry about nitty gritty stuff
  - Did I do VPC correctly?
  - Did I use the right security group with the EC2s?
  - Did I create load balancer properly?
  - Ohh no, I messed up my Auto Scaling Group and my CloudFormation failed!
- Elastic Beanstalk uses CloudFormation behind the scenes

# Demo – Elastic Beanstalk


Raj Saha
cloudwithraj.com
Cloud With Raj

# AWS Marketplace

Raj Saha
cloudwithraj.com
Cloud With Raj

# AWS Marketplace Benefits

AWS Marketplace

- Use your favorite 3rd party software with your AWS application
  - Includes Professional Services
- Consolidated billing with AWS
  - Charges appear on your AWS bill
- Flexible consumption and contract models
  - Free trial, hourly, monthly, annual, multi-year, and consumption based models
- Easy deployment
  - Get started in one click
- Secure
  - QA and curation done by AWS

# Microservices

# The monolith

"…a **single-tiered software application** in which the user interface and data access code are combined into a **single program** from a **single platform**."

- Wikipedia

# Monolith is not the bad guy!

Pros:

- At first…
  - Simple
  - No over-engineering
- Single code base
- Resource efficient at small scale

Cons:

Modularity is hard to enforce as app grows

Scaling is a challenge

All or nothing deployment

Long release cycles

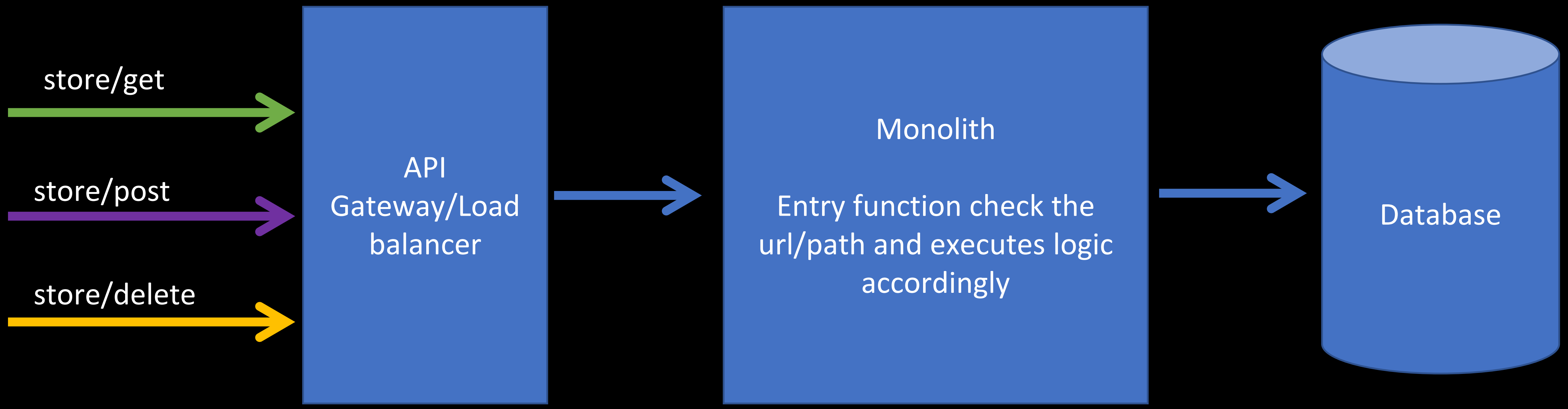
Slow to react to customer demand

# Can you use API with Monolith?

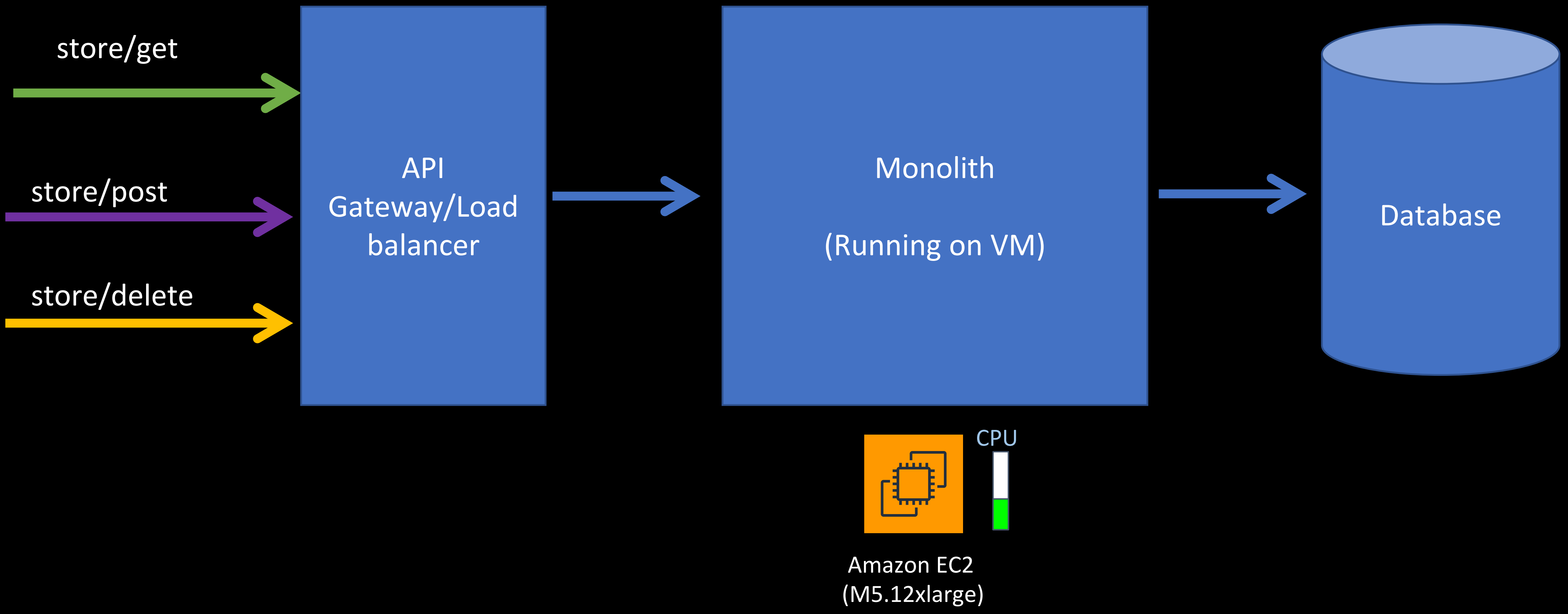
# Absolutely

APIs does NOT equal microservices

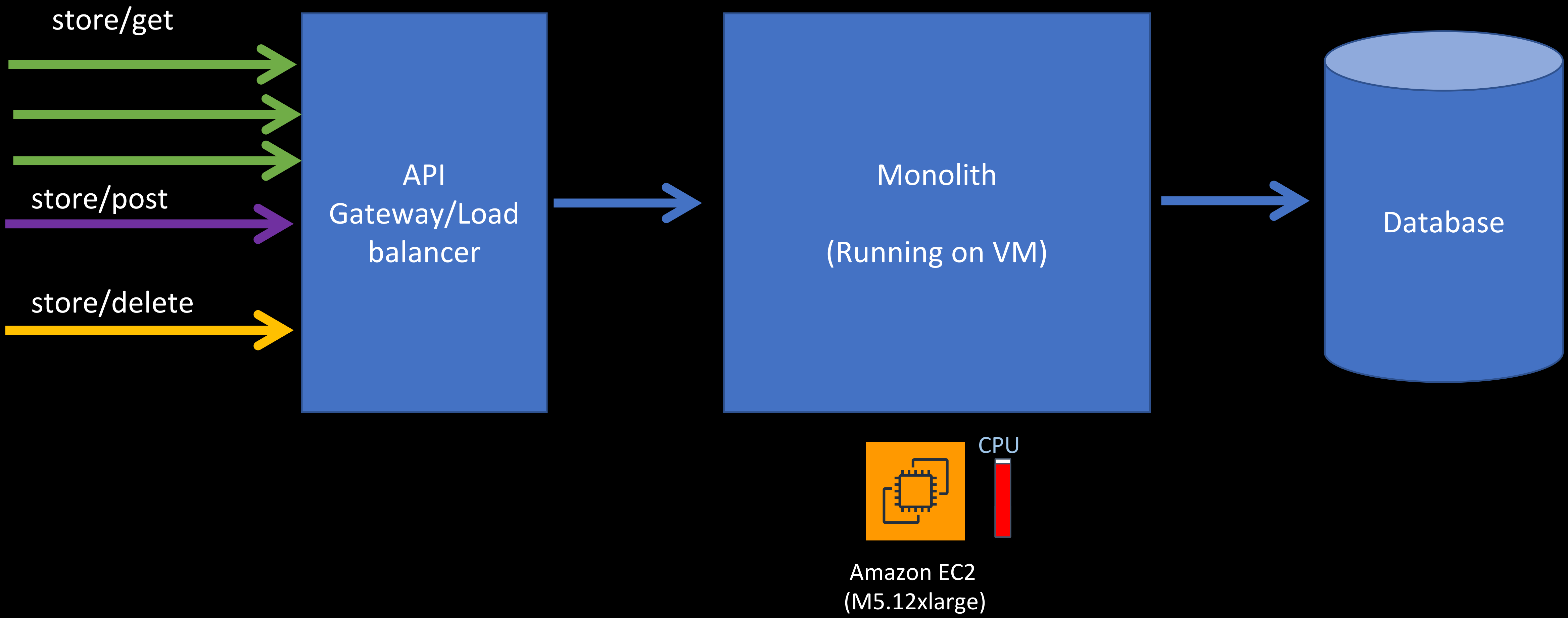
# APIs in Monolith



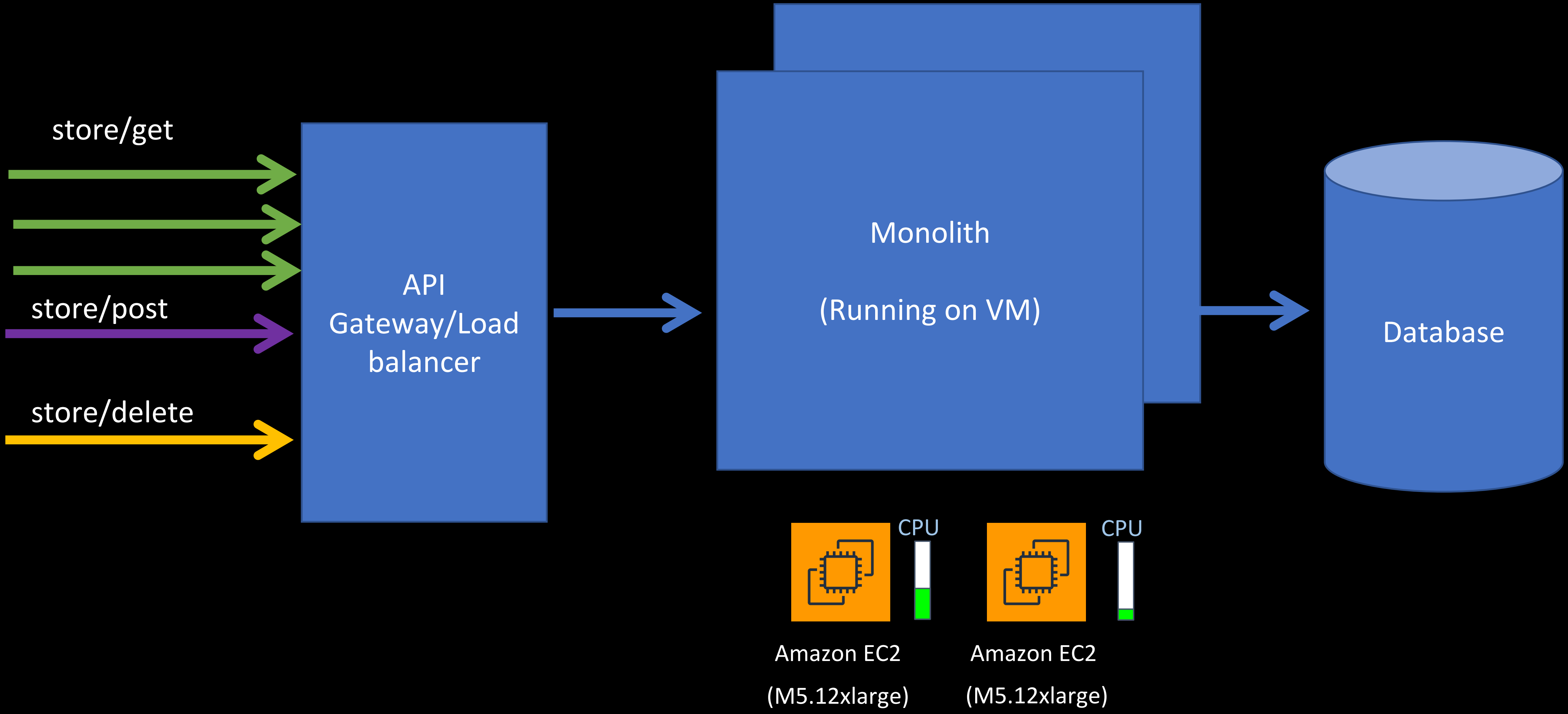
# Issue of Scaling



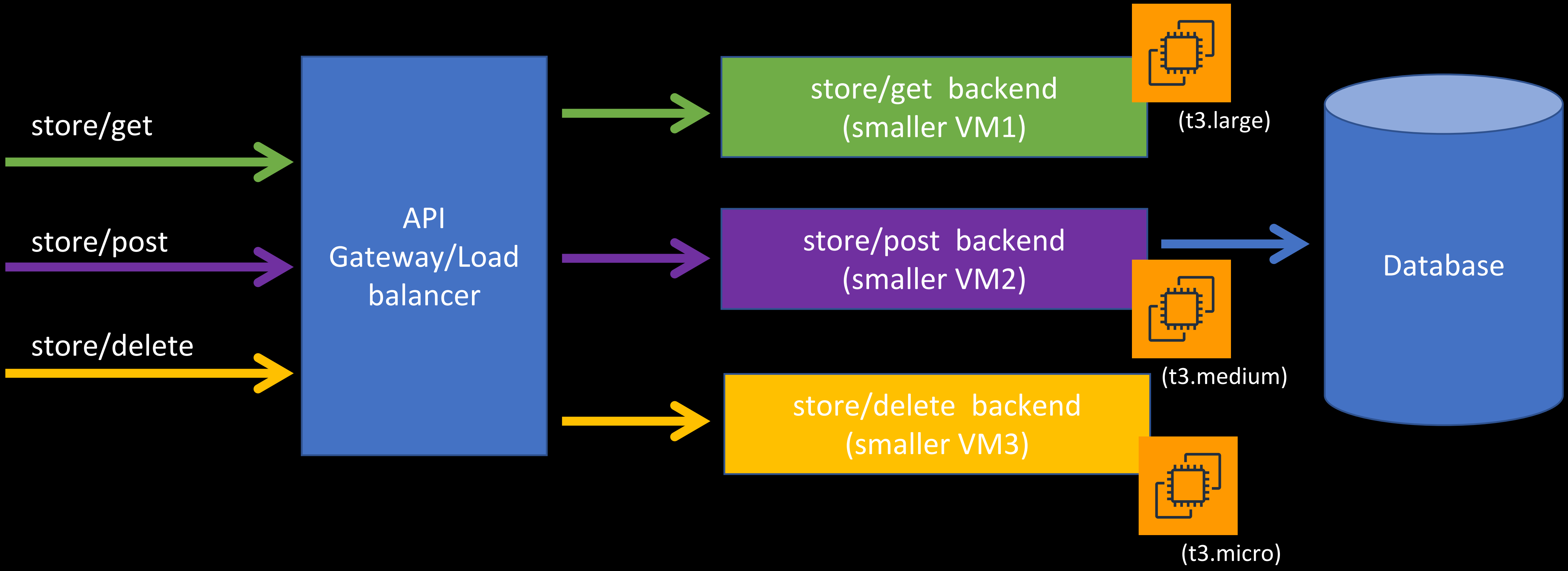
# Issue of Scaling



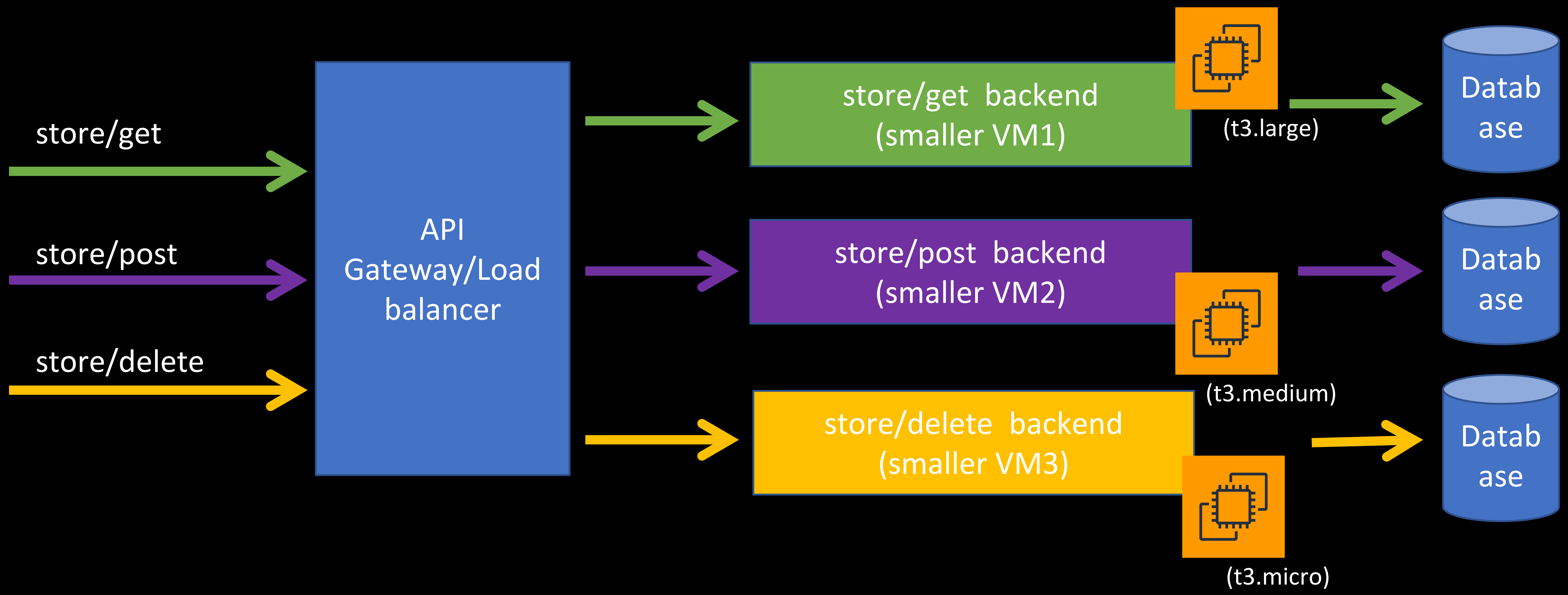
# Entire Monolith Need to Scale



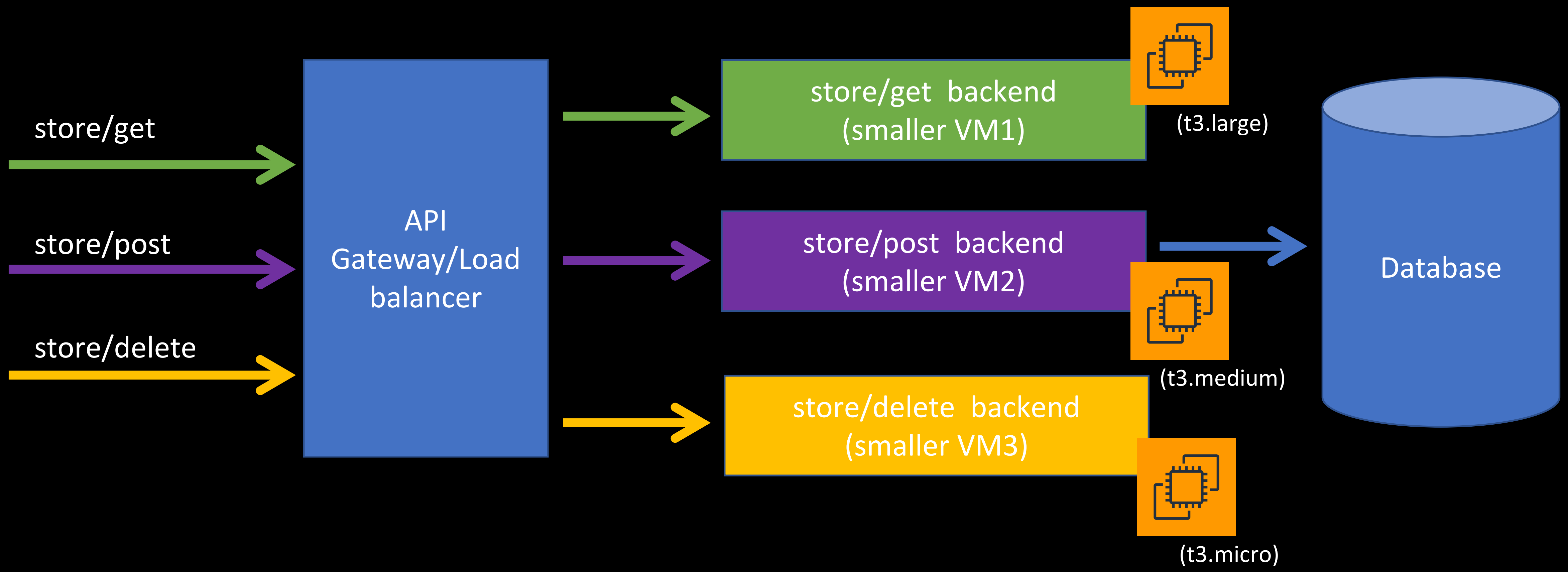
# APIs in Microservice



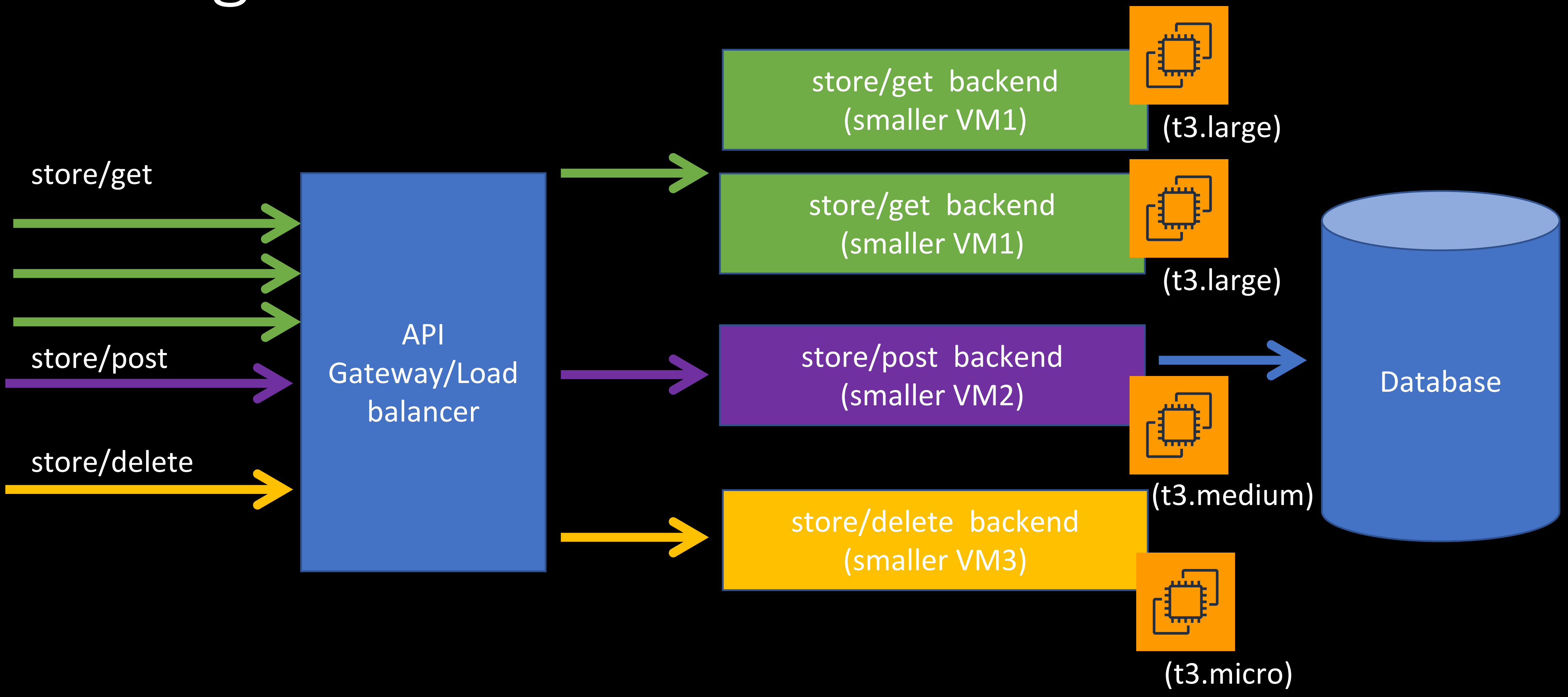
# APIs in Microservice



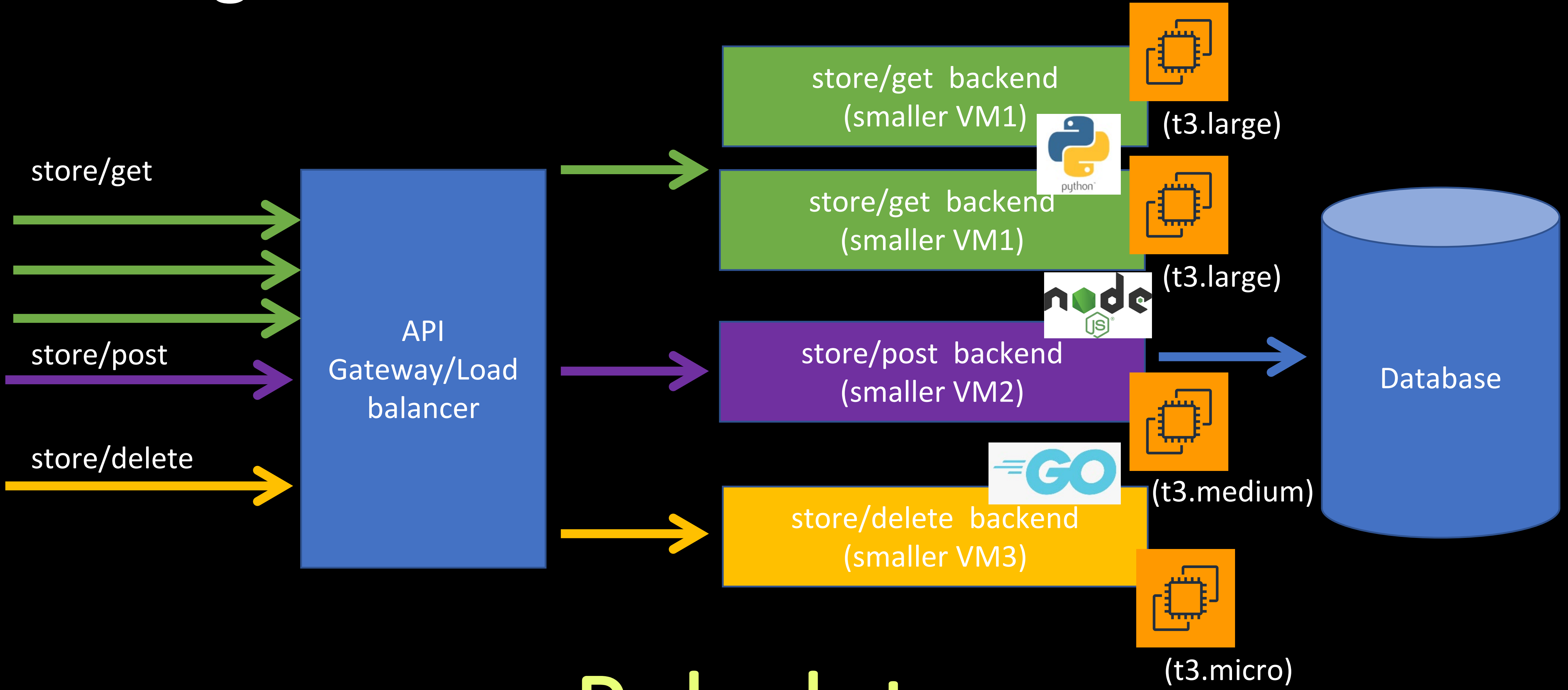
# APIs in Microservice



# Scaling APIs in Microservice



# Scaling APIs in Microservice




Polyglot

# Characteristics of microservice architectures

- Independent
  - Scaling
  - Governance
  - Deployment
  - Testing
  - Functionality

Important - Not required to follow every characteristic

# AWS Networking

# AWS Region and Availability Zones

Raj Saha
cloudwithraj.com
Cloud With Raj

# Cloud Managed Data Center


Data Center Managed
By Amazon

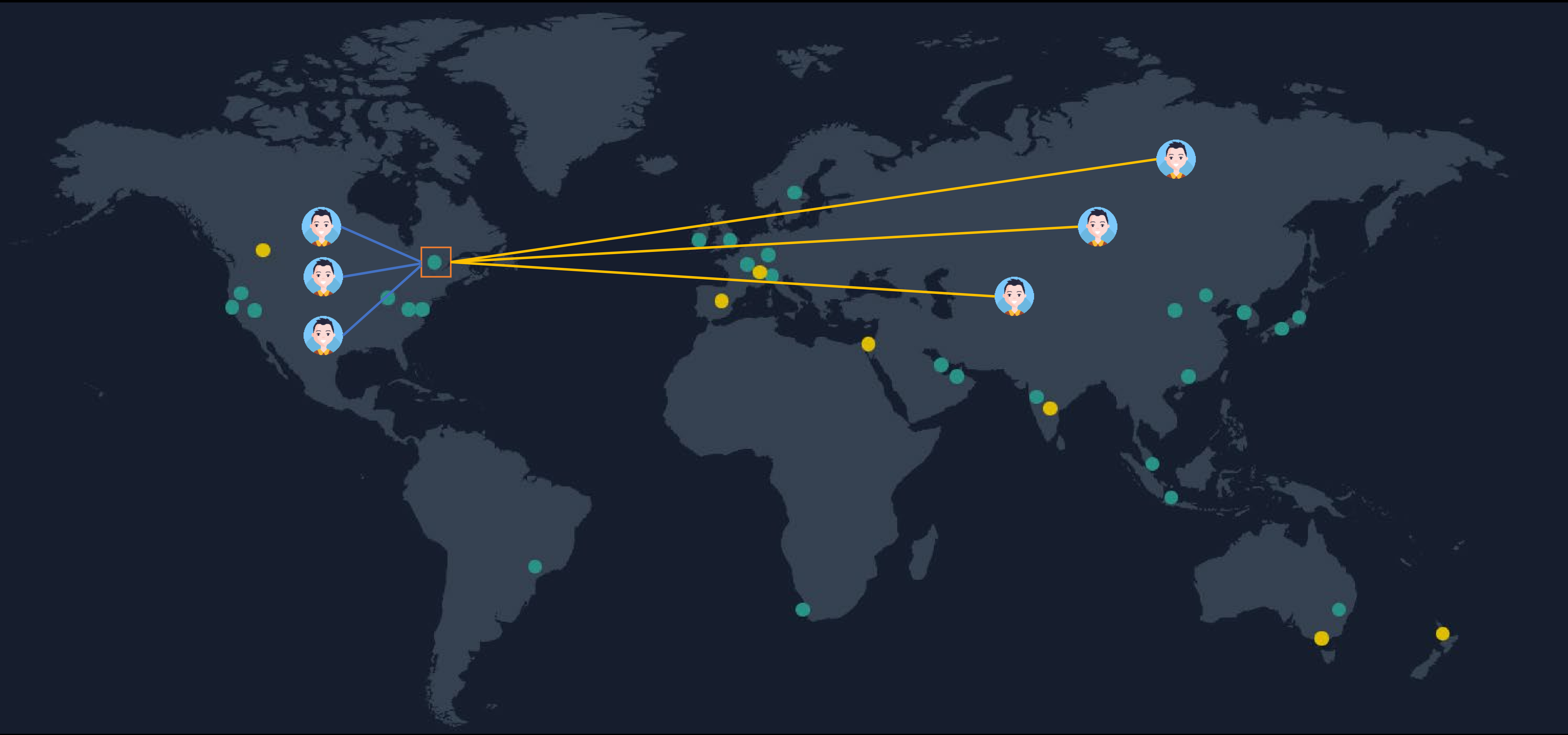
# AWS Region and Availability Zone

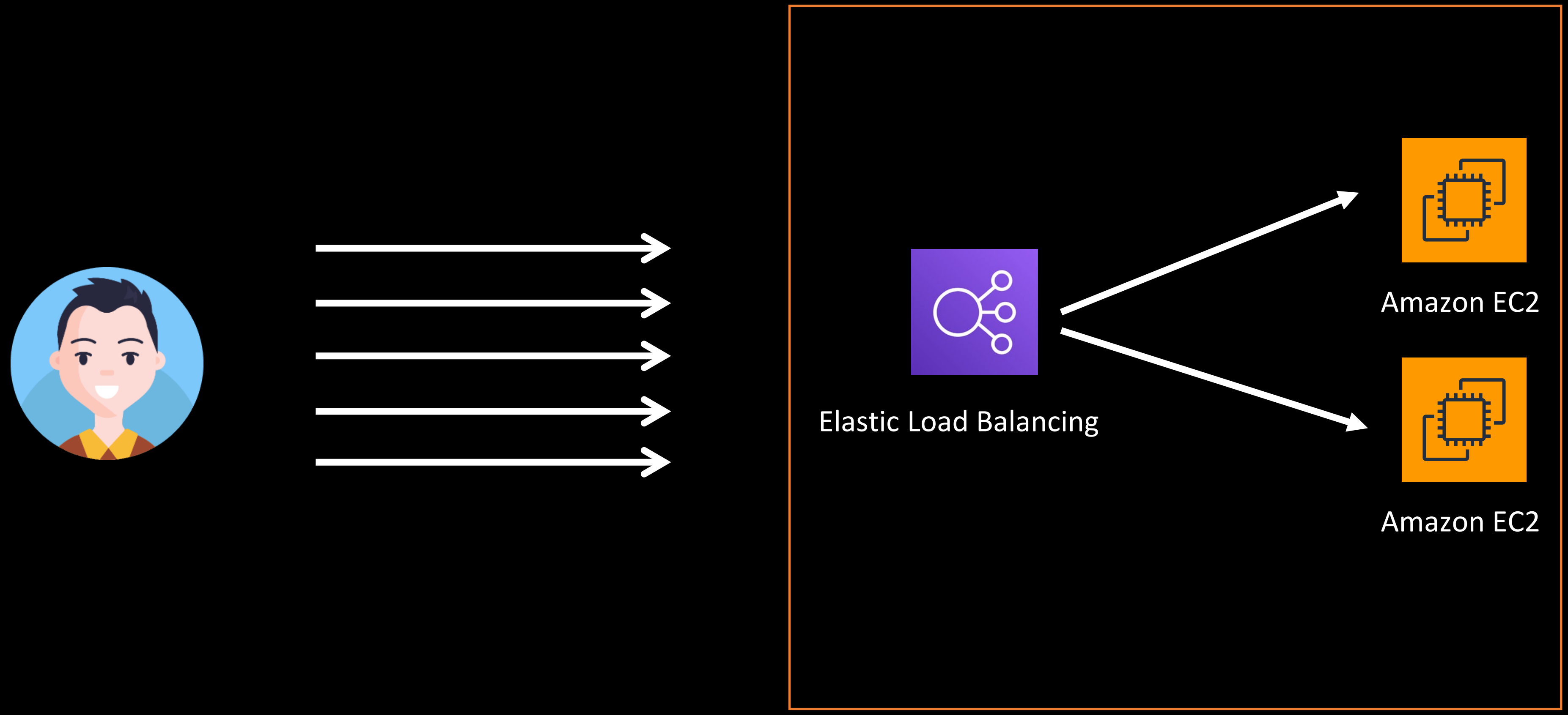
- Region - Physical location around the world where AWS cluster data centers
- Each region consists of multiple Availability Zone (AZ)
- AZ is a group of logical data centers
- Each AZ has independent power, cooling, and physical security
- All AZs are interconnected with high-bandwidth, low-latency networking
  - Traffic between AZs is encrypted
  - AZs are separated but within 100km(60 miles)
  - Other clouds call one region as one datacenter

# https://aws.amazon.com/about-aws/global-infrastructure/

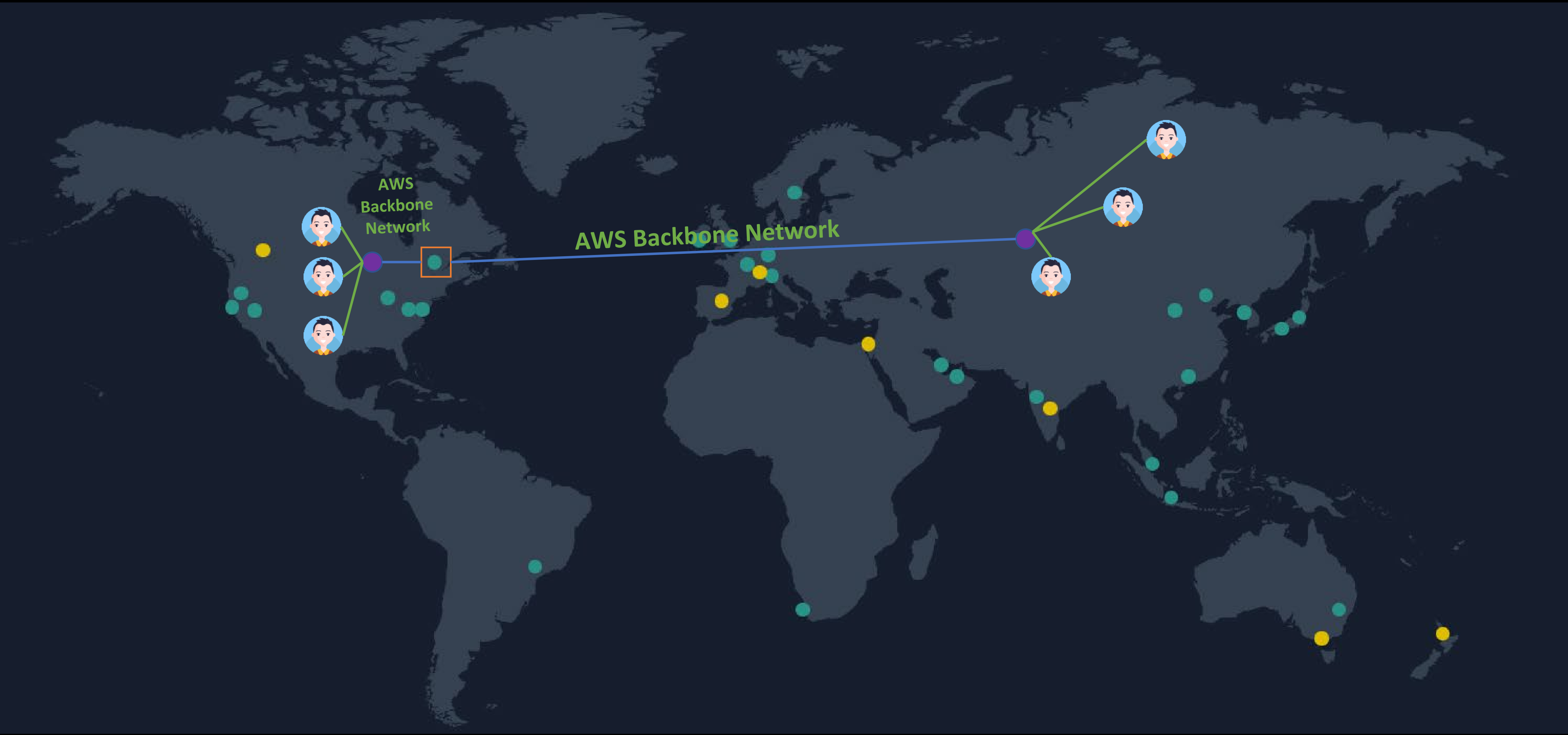
# Edge Location and CloudFront – What and Why

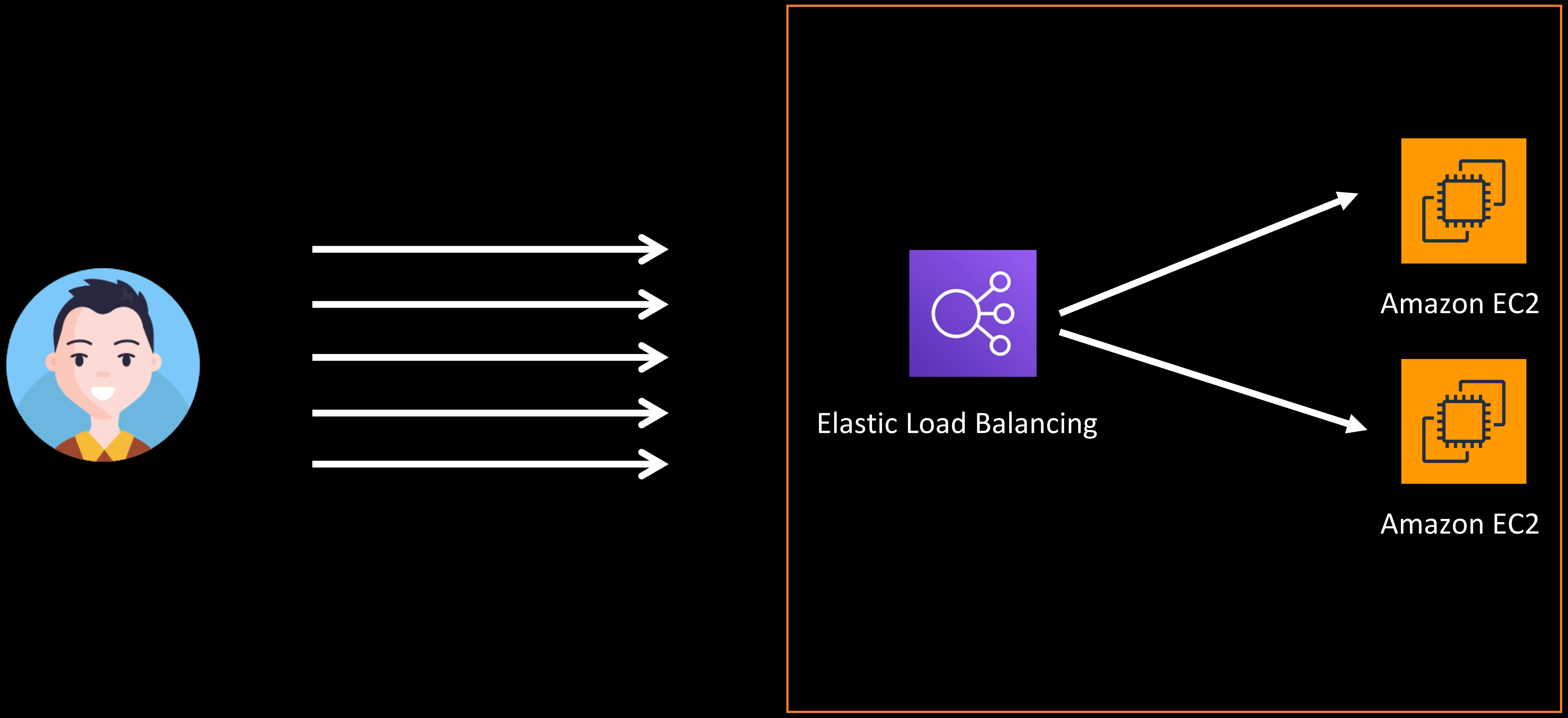
Raj Saha
cloudwithraj.com
Cloud With Raj



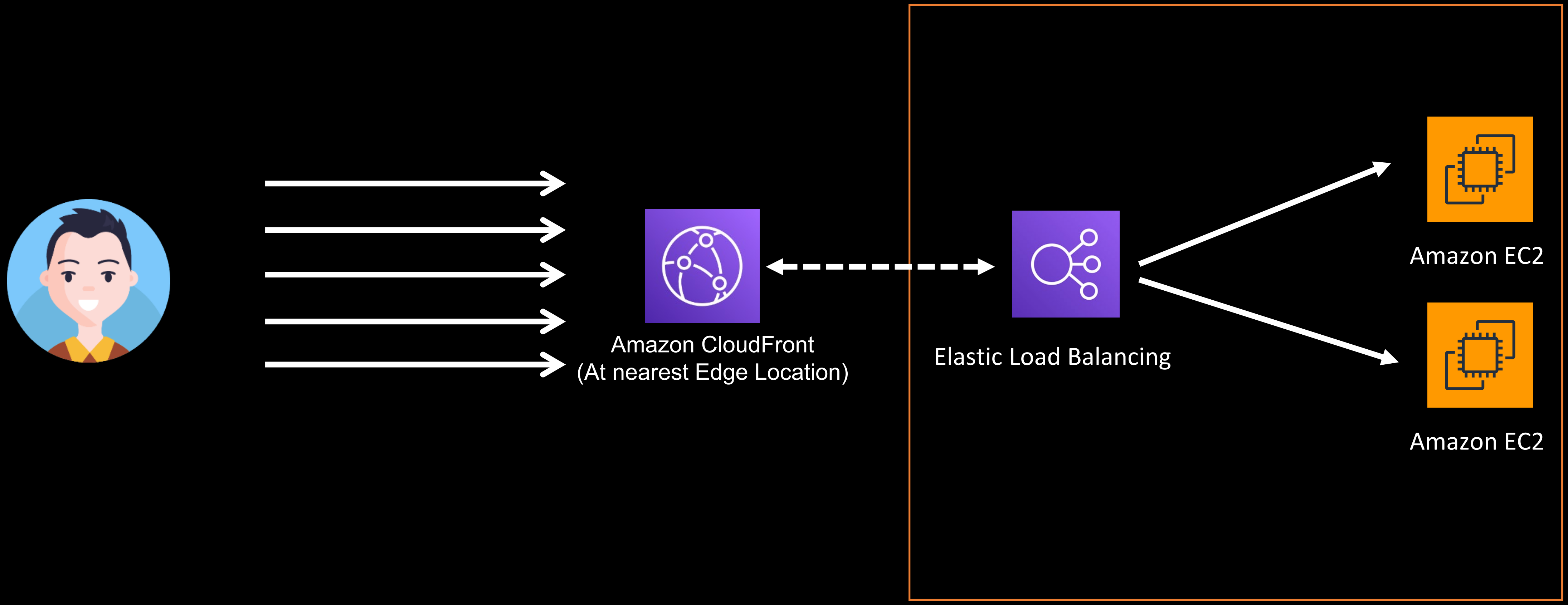



- Latency of responses from the application to the end user
- Compute is expensive to scale


AWS
Backbone
Network
AWS Backbone Network




- Latency of responses from the application to the end user
- Compute is expensive to scale




- Lower latency for popular content
- Less scaling stress on the actual application
- Also known as AWS CDN (Content Delivery Network)

Amazon CloudFront

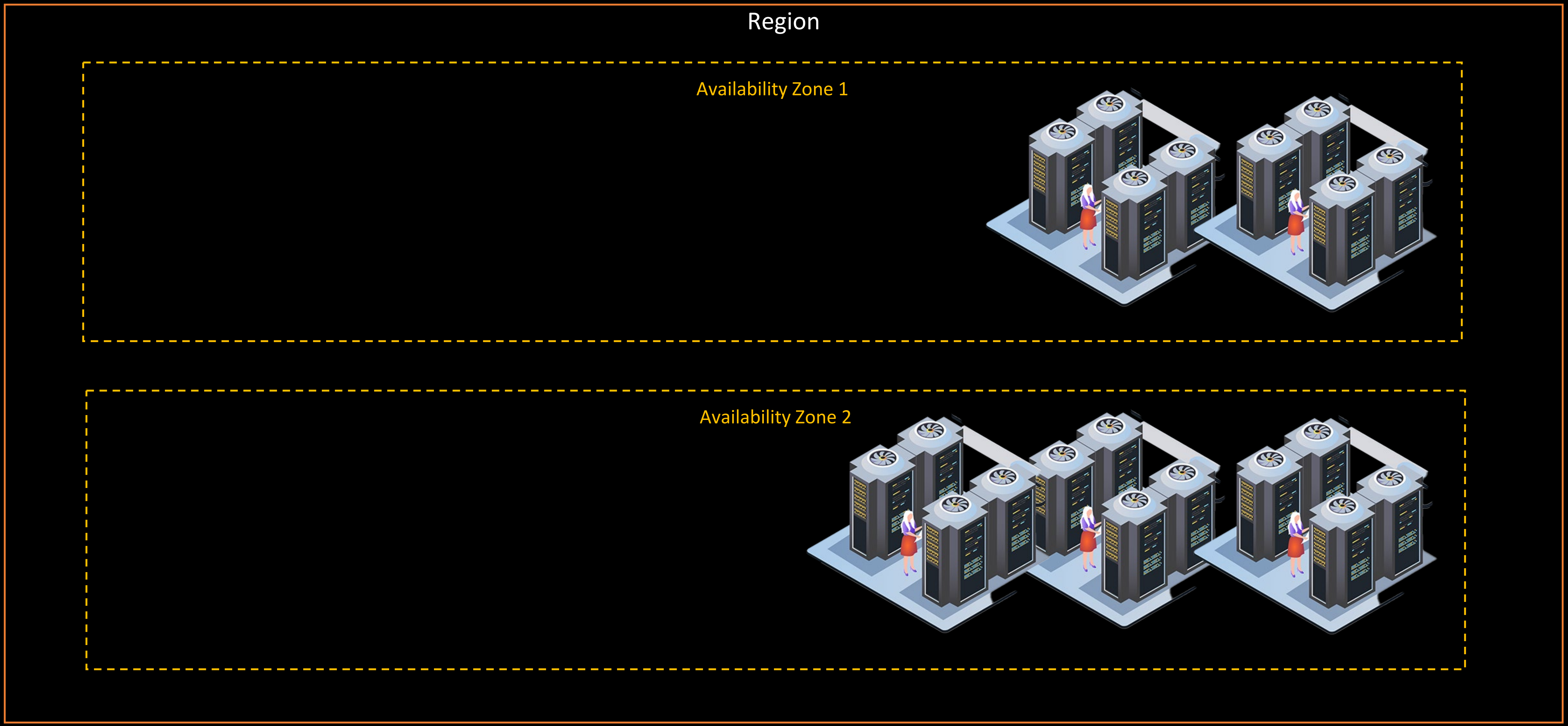
- Lower latency for popular content
- Less scaling stress on the actual application
- Also known as AWS CDN (Content Delivery Network)
- Connected to the AWS regions through AWS network backbone
- Improved security
    - Traffic encryption
    - Access controls
    - Defend against DDoS using AWS Shield Standard
- Equivalent third party products – Akamai, Cloudfare

# VPC (Virtual Private Cloud)

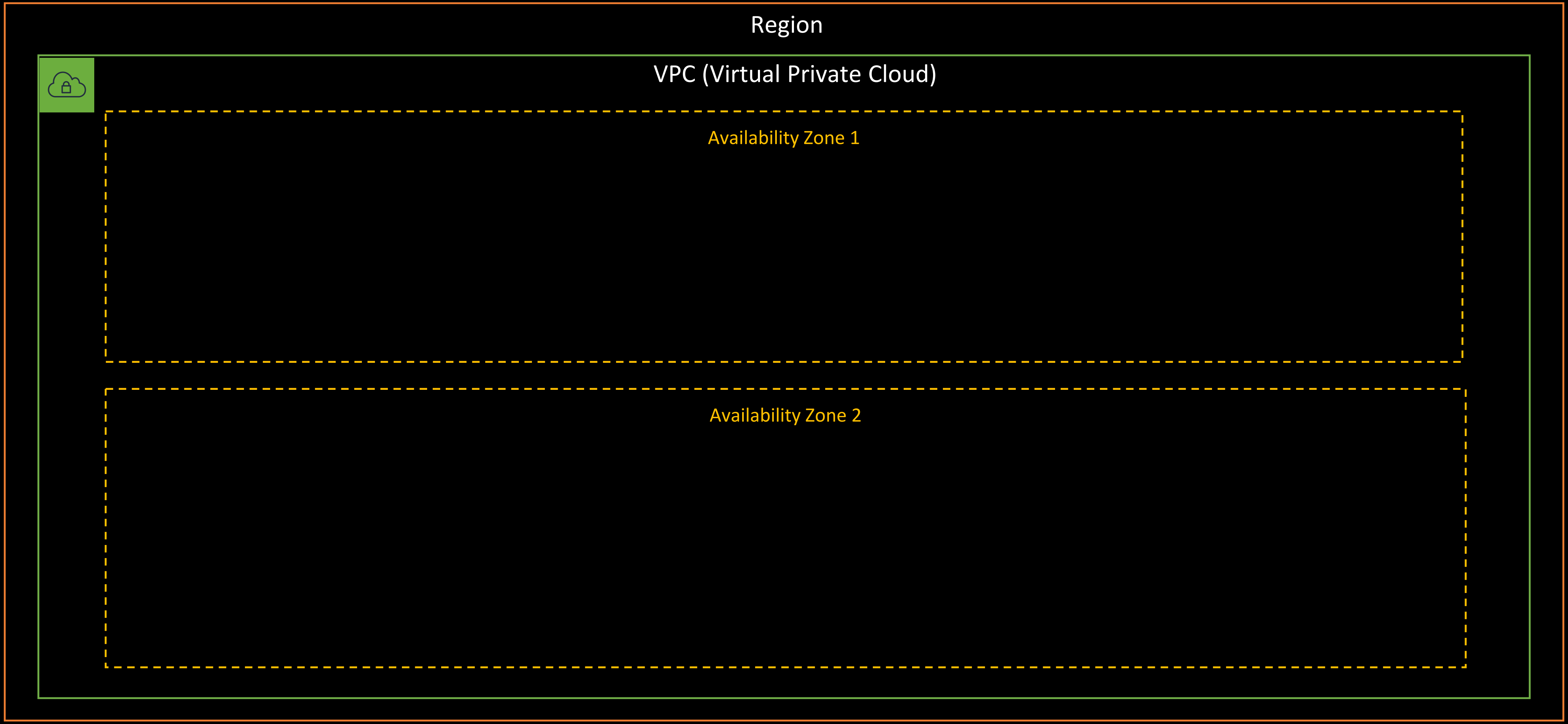
Raj Saha
cloudwithraj.com
Cloud With Raj

# AWS Regions

# AWS

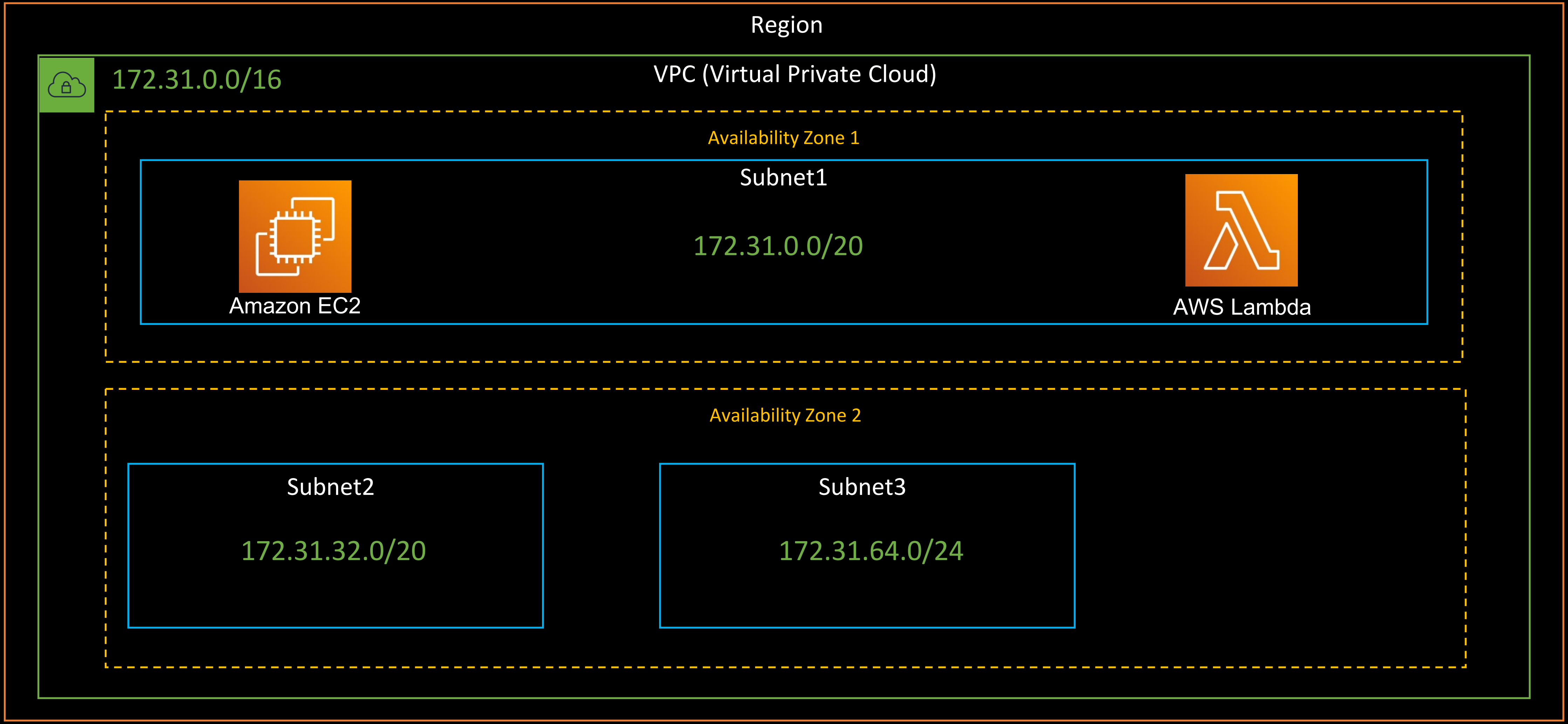


# VPC (Virtual Private Cloud)

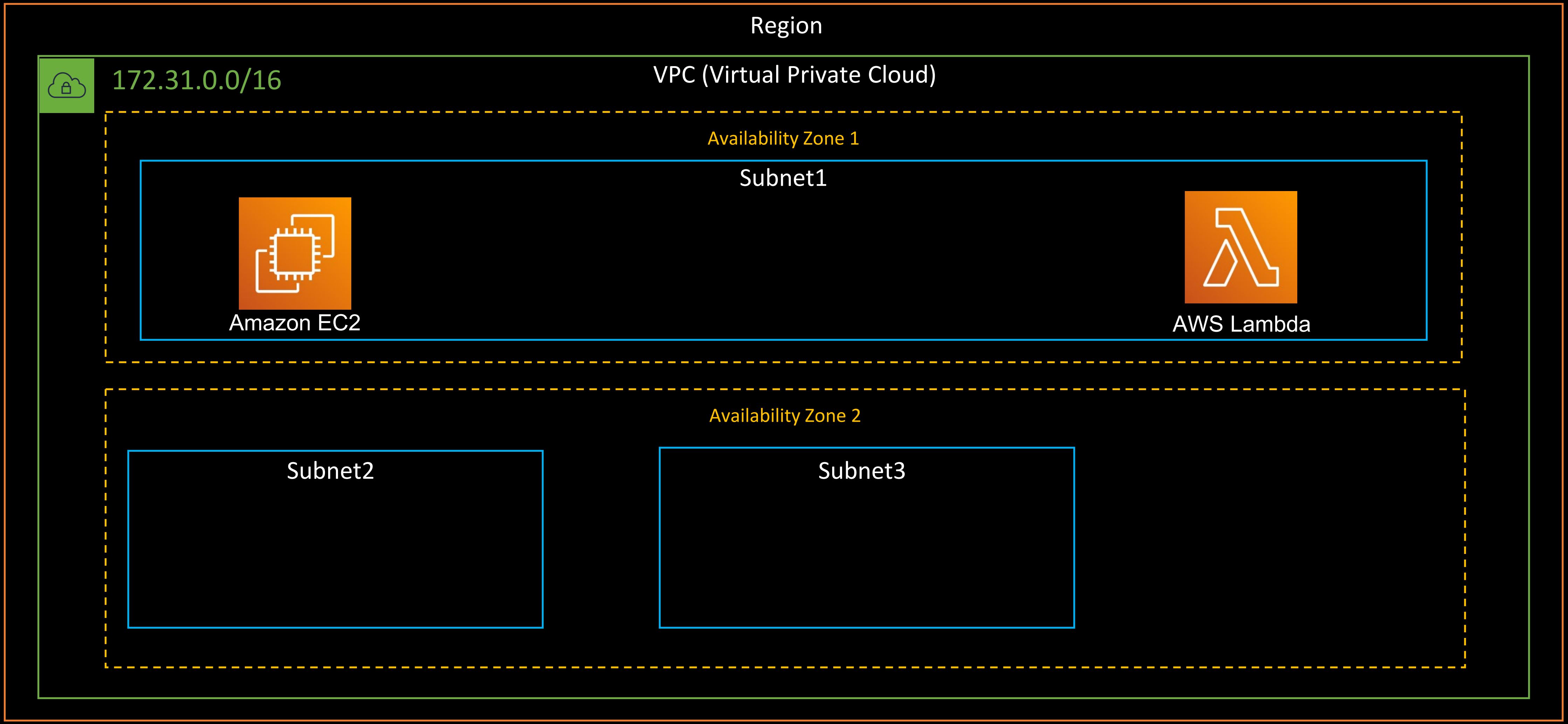


# VPC (Virtual Private Cloud)

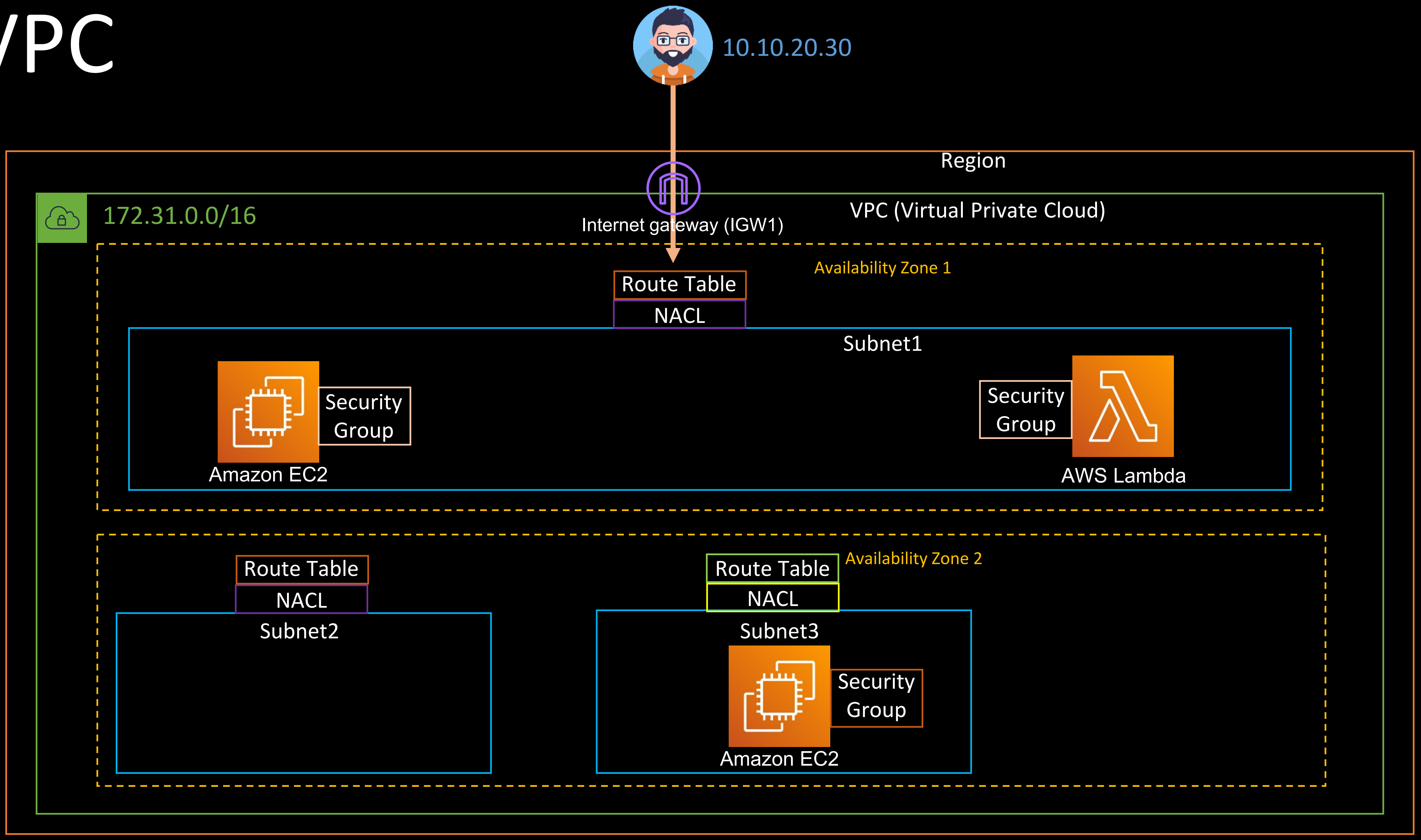
# VPC (Virtual Private Cloud)



# VPC


Region
172.31.0.0/16
VPC (Virtual Private Cloud)
Availability Zone 1
Subnet1
Amazon EC2
AWS Lambda
Availability Zone 2
Subnet2
Subnet3

# VPC

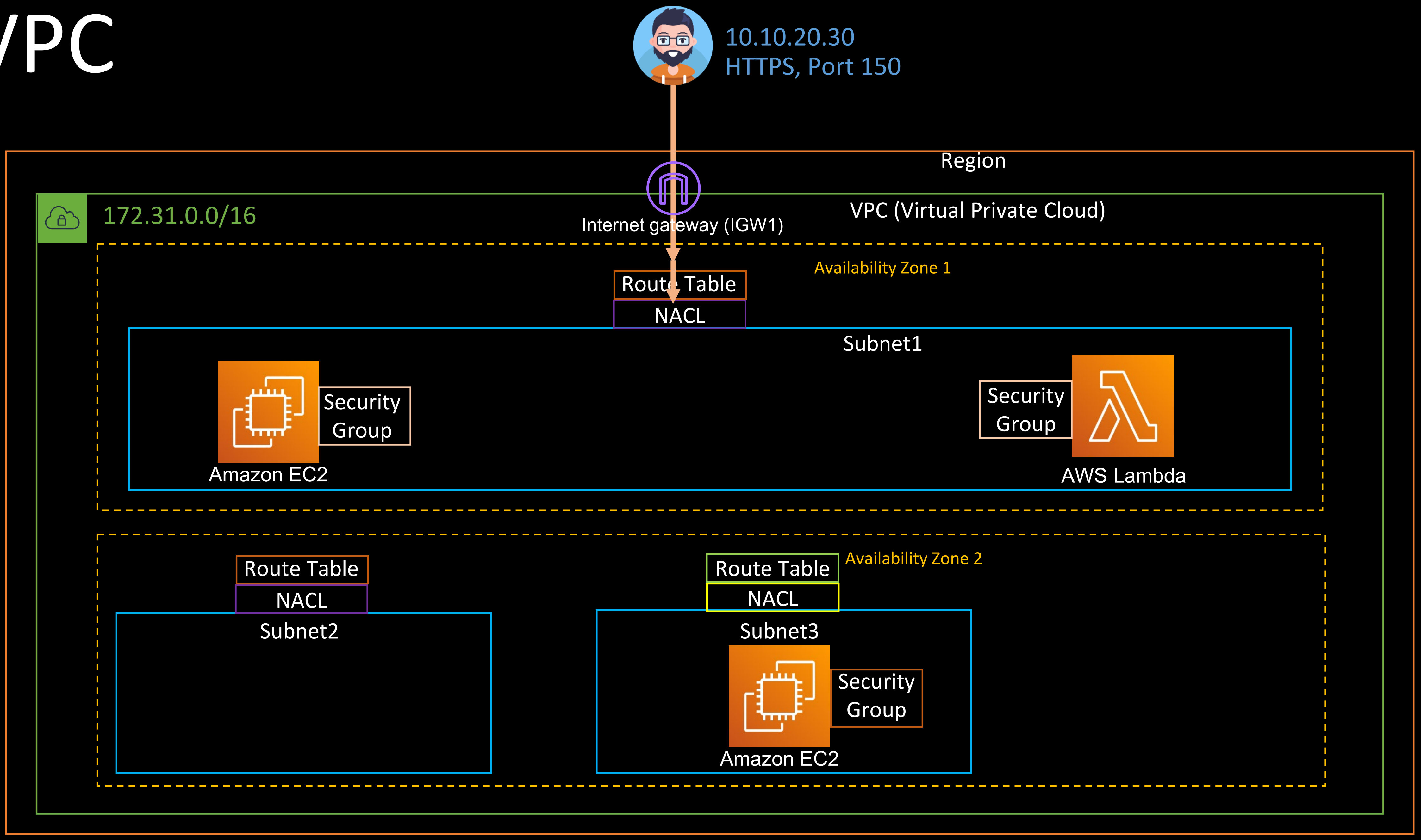

10.10.20.30
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
Security Group
Amazon EC2
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
Route Table
NACL
Subnet3
Security Group
Amazon EC2

# Route Table

Route Table attached to subnet1

| Destination | Target |
|---|---|
| 0.0.0.0/0 | IGW1 |
| 172.31.0.0/16 | local |

# VPC


10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
Security Group
Amazon EC2
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
Route Table
NACL
Subnet3
Security Group
Amazon EC2

# NACL (Network Access Control List)

10.10.20.30
HTTPS, Port 150

NACL attached to subnet1

## Inbound rules

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 110 | HTTPS (443) | TCP (6) | 100-300 | 10.10.0.0/16 | Allow |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

## Outbound rules

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | All traffic | All | All | 0.0.0.0/0 | Allow |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

NACL are stateless, inbound and outbound traffic evaluated

# NACL (Network Access Control List)

10.10.20.30
HTTPS, Port 150

NACL attached to subnet1

## Inbound rules

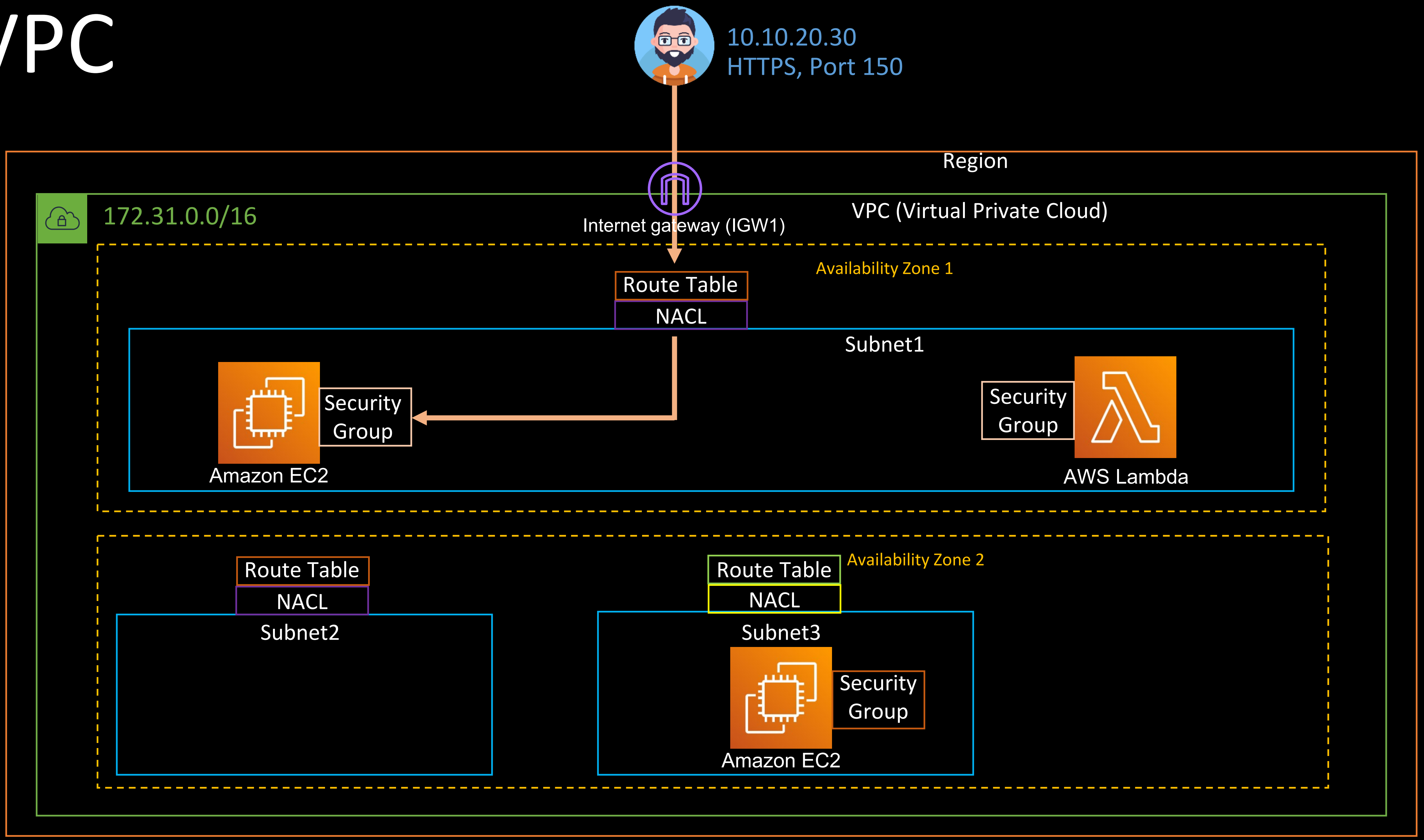
| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | All traffic | All | All | 0.0.0.0/0 | Allow |
| 110 | HTTPS (443) | TCP (6) | 100-300 | 10.10.0.0/16 | Deny |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

## Outbound rules

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | All traffic | All | All | 0.0.0.0/0 | Allow |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

NACL are stateless, inbound and outbound traffic evaluated

# VPC


10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
Security Group
Amazon EC2
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
Route Table
NACL
Subnet3
Security Group
Amazon EC2

# Security Group

Security
Group

Inbound rules

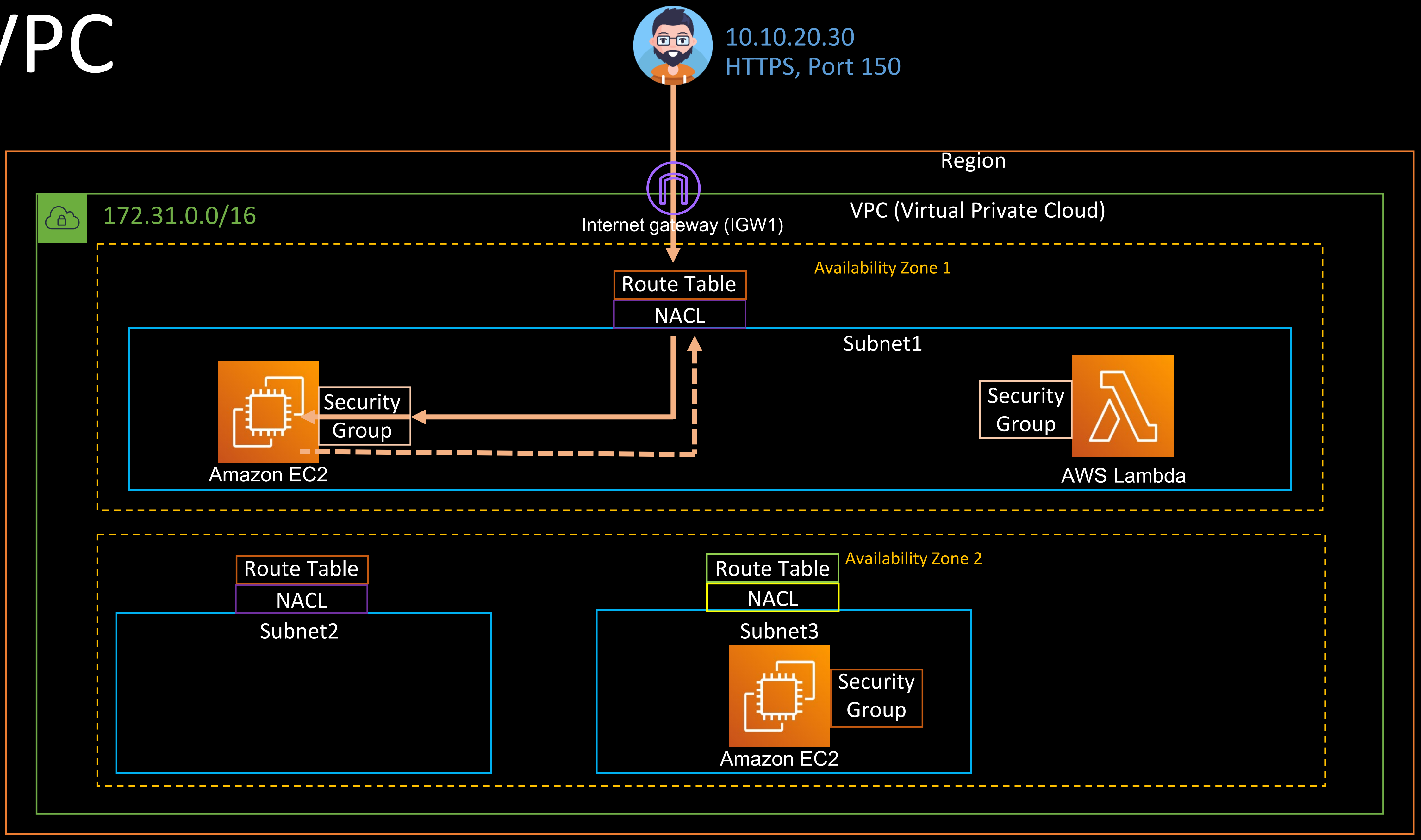
| Rule ID | Type | Protocol | Port range | Source |
|---|---|---|---|---|
| ID1 | HTTPS | All | All | 0.0.0.0/0 |
| ID2 | All traffic | All | All | 0.0.0.0/0 |

Outbound rules

| Rule ID | Type | Protocol | Port range | Source |
|---|---|---|---|---|
| ID3 | All traffic | All | All | 0.0.0.0/0 |

Security groups are stateful, irrespective of the rules

# VPC



10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
Security Group
Amazon EC2
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
Route Table
NACL
Subnet3
Security Group
Amazon EC2

# NACL (Network Access Control List)

10.10.20.30
HTTPS, Port 150

## NACL attached to subnet1

### Inbound rules

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | All traffic | All | All | 0.0.0.0/0 | Allow |
| 110 | HTTPS (443) | TCP (6) | 100-300 | 10.10.0.0/16 | Deny |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

### Outbound rules

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | All traffic | All | All | 0.0.0.0/0 | Allow |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

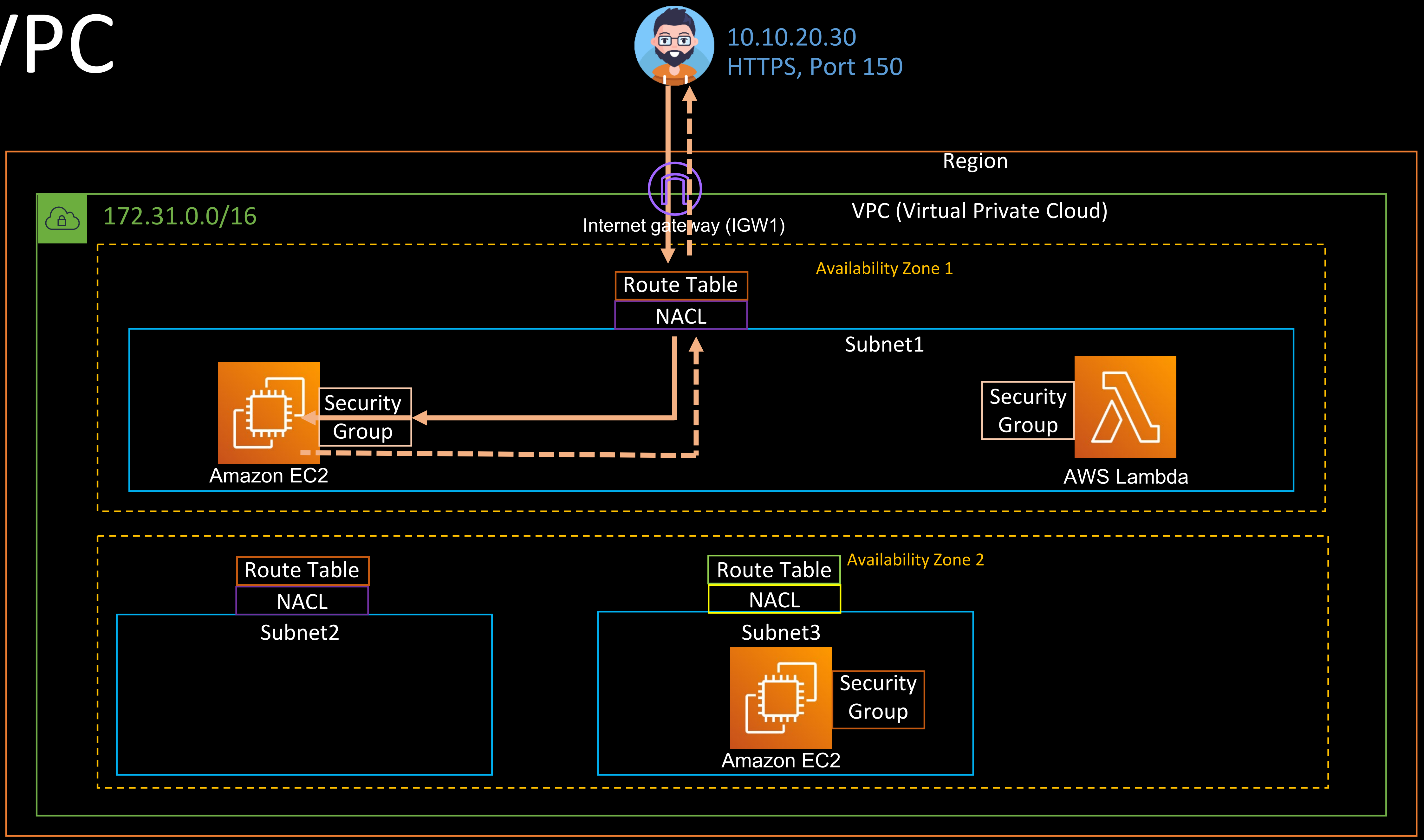
NACL are stateless, inbound and outbound traffic evaluated

# Route Table

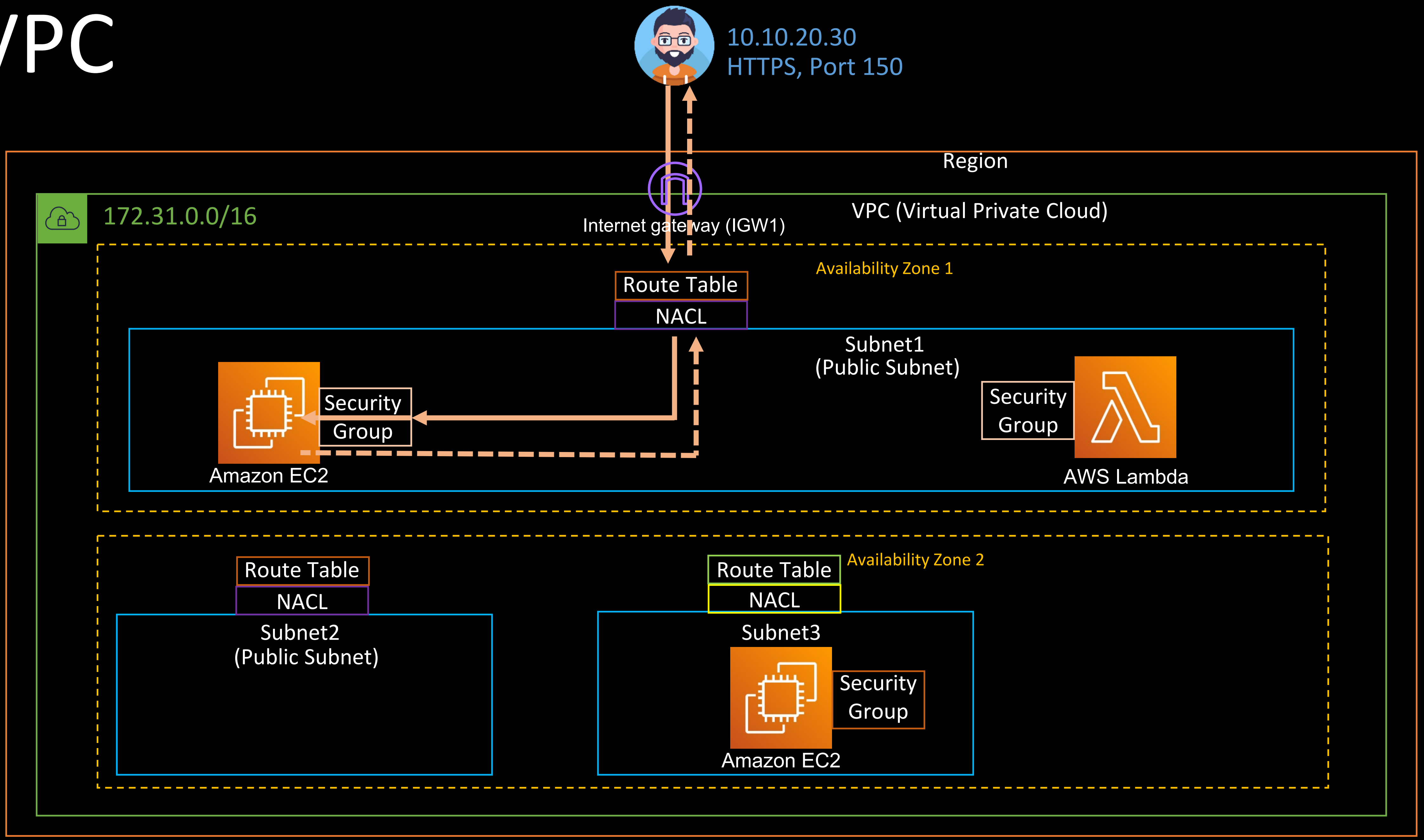
Route Table attached to subnet1

| Destination | Target |
|---|---|
| 0.0.0.0/0 | IGW1 |
| 172.31.0.0/16 | local |

# VPC


10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
Security Group
Amazon EC2
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
Route Table
NACL
Subnet3
Security Group
Amazon EC2

# VPC

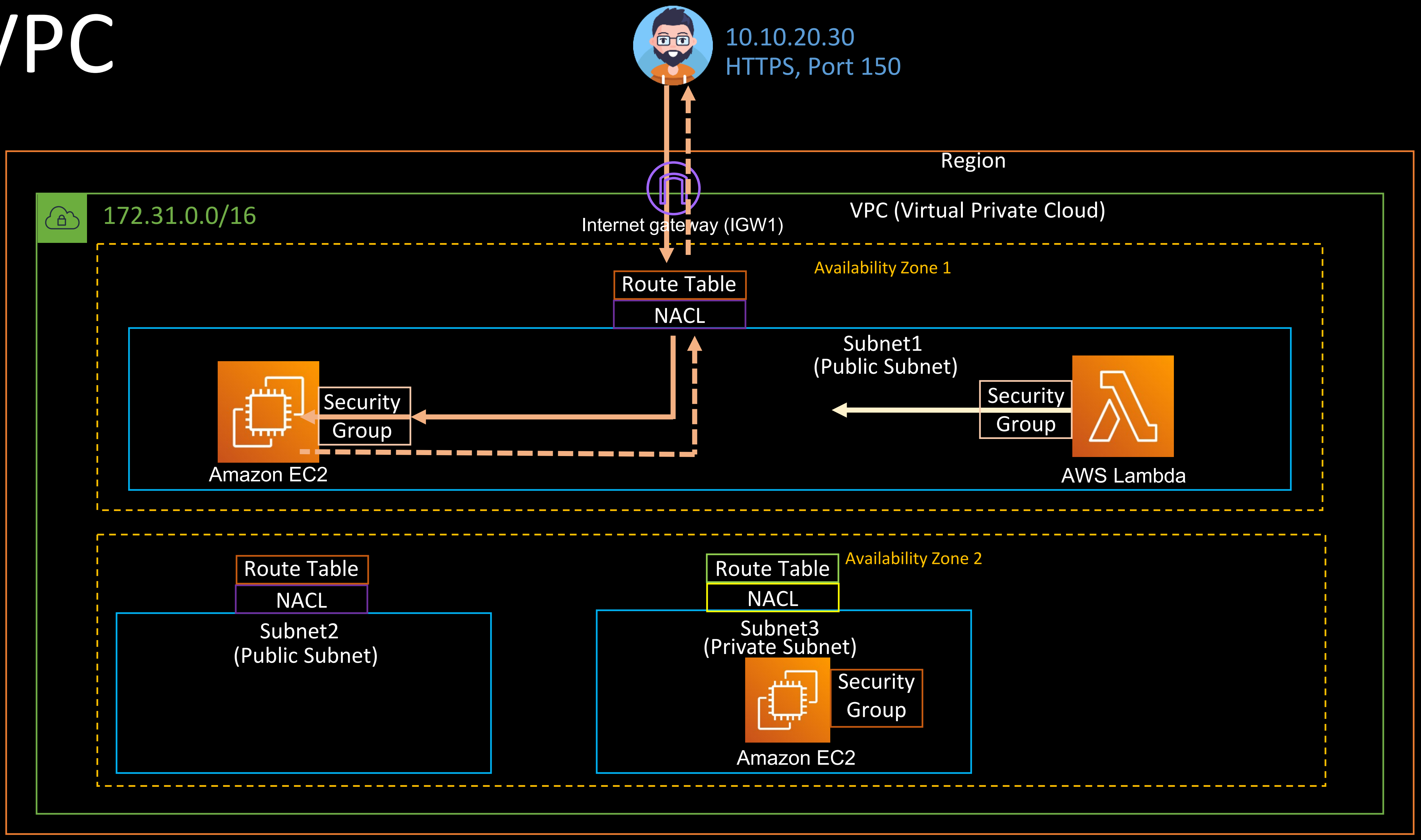

10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
(Public Subnet)
Security Group
Amazon EC2
Security Group
AWS Lambda
Route Table
NACL
Availability Zone 2
Route Table
NACL
Subnet2
(Public Subnet)
Subnet3
Security Group
Amazon EC2

# Route Table

Route Table attached to subnet3

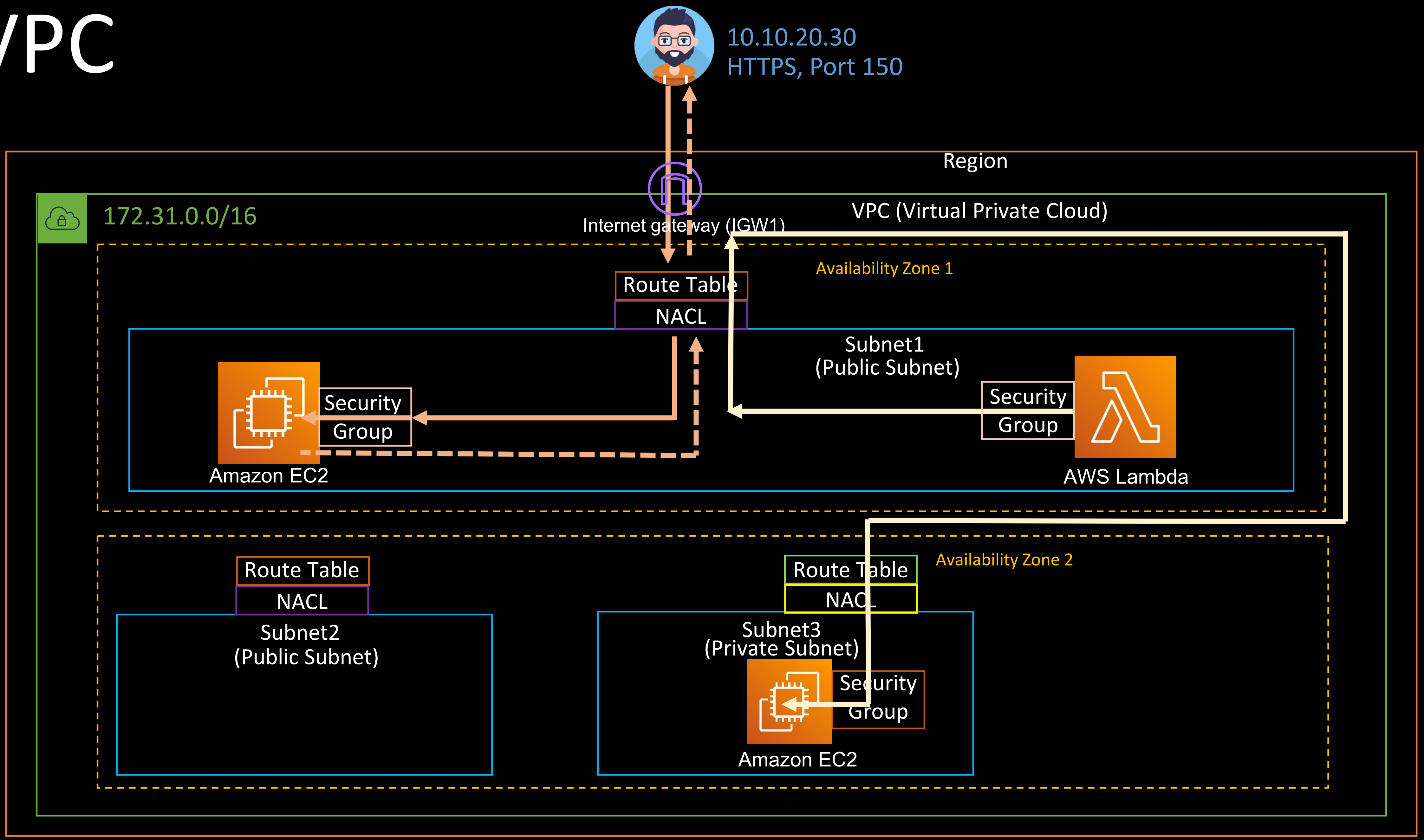
| Destination | Target |
|---|---|
| 172.31.0.0/16 | local |

# VPC

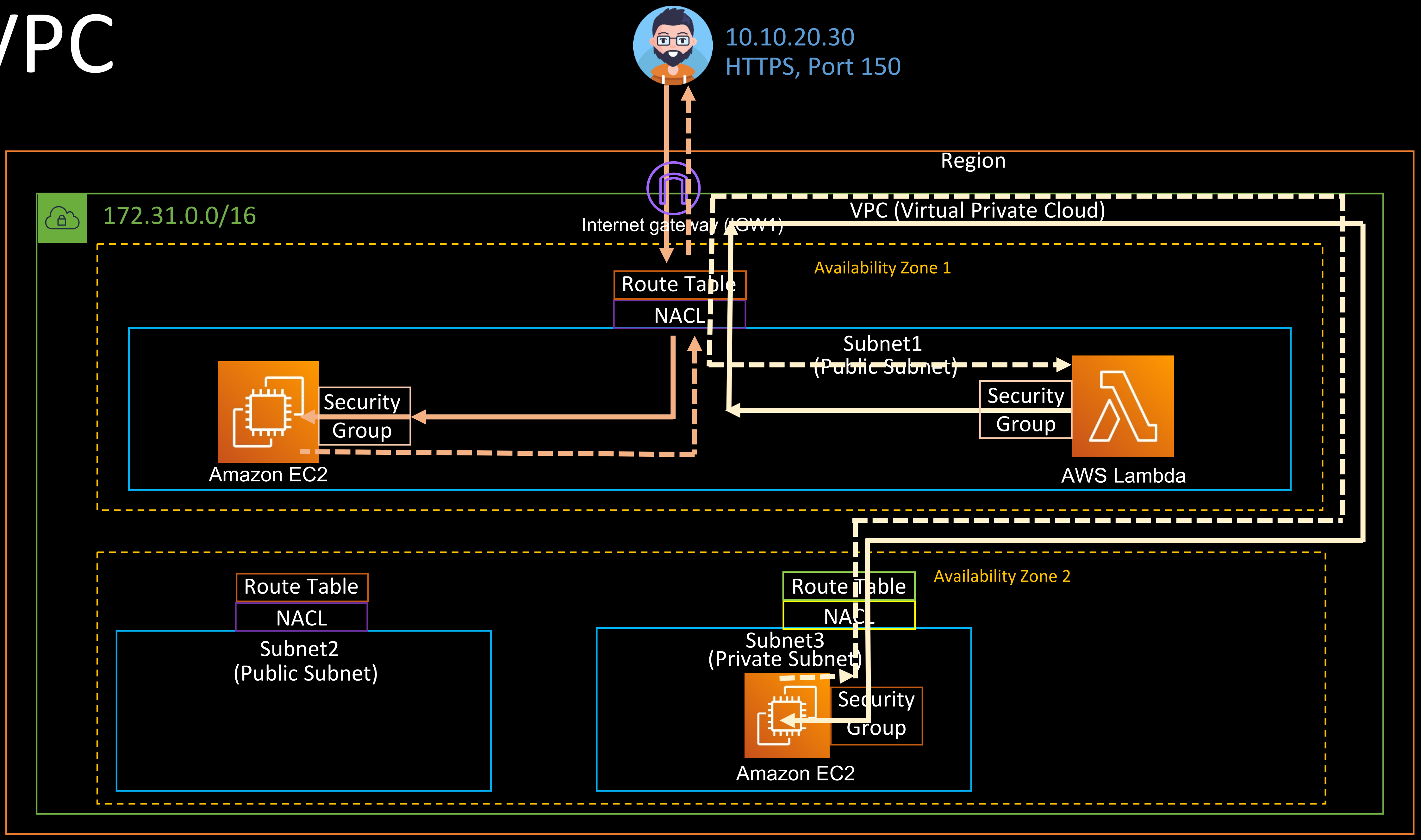


10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
(Public Subnet)
Security
Group
Amazon EC2
Security
Group
AWS Lambda
Route Table
NACL
Subnet2
(Public Subnet)
Route Table
NACL
Availability Zone 2
Subnet3
(Private Subnet)
Security
Group
Amazon EC2

# VPC

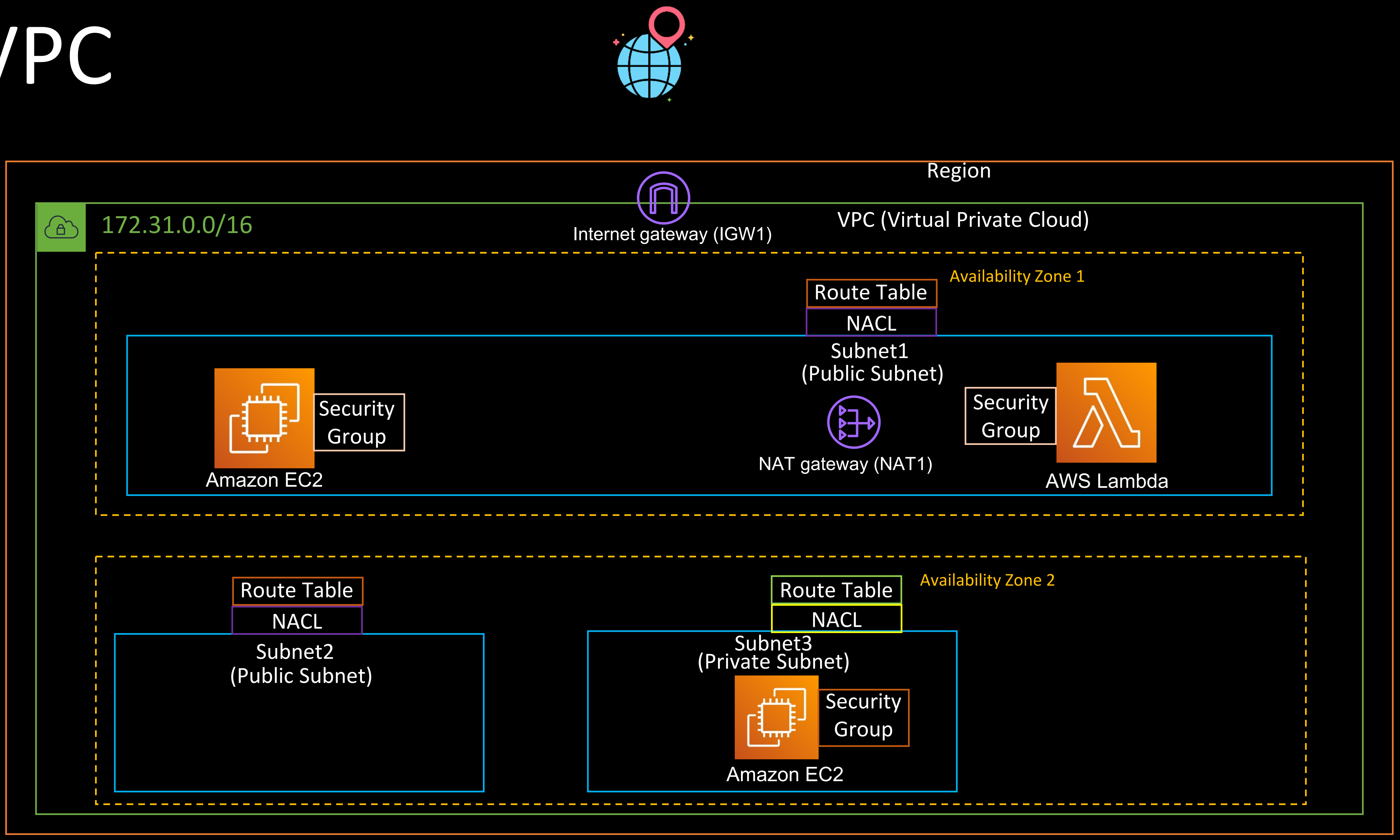

10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Route Table
NACL
Availability Zone 1
Subnet1
(Public Subnet)
Security
Group
Amazon EC2
Security
Group
AWS Lambda
Route Table
NACL
Subnet2
(Public Subnet)
Route Table
NACL
Availability Zone 2
Subnet3
(Private Subnet)
Security
Group
Amazon EC2

# VPC

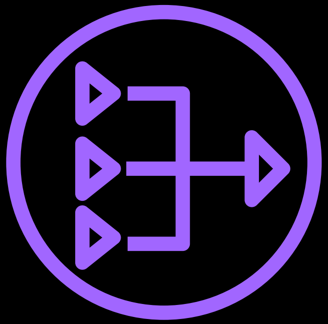


10.10.20.30
HTTPS, Port 150
Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
(Public Subnet)
Security
Group
Amazon EC2
Security
Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
(Public Subnet)
Route Table
NACL
Subnet3
(Private Subnet)
Security
Group
Amazon EC2

# VPC


Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
(Public Subnet)
Security Group
Amazon EC2
NAT gateway (NAT1)
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
(Public Subnet)
Route Table
NACL
Subnet3
(Private Subnet)
Security Group
Amazon EC2

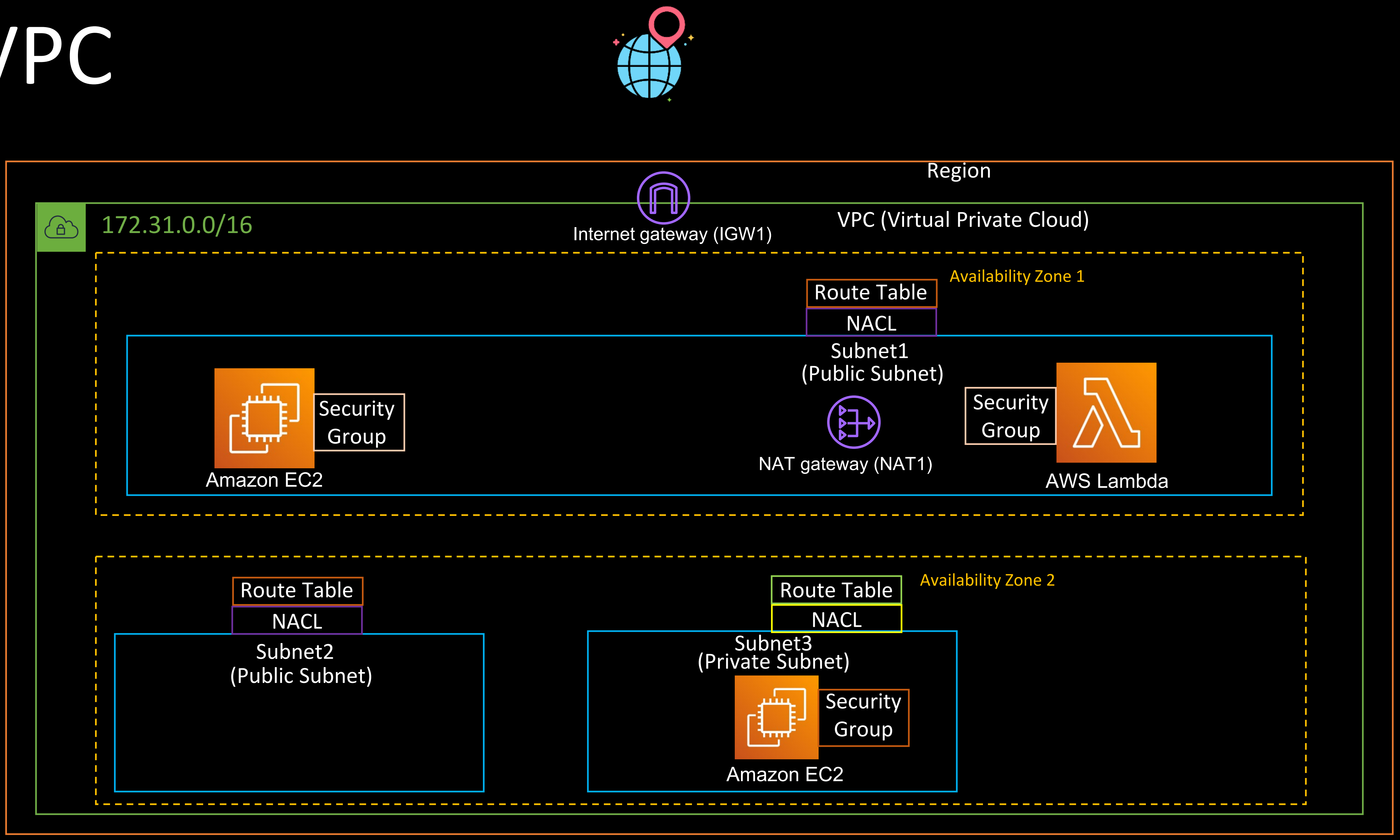
# NAT Gateway



NAT gateway

- NAT stands for Network Address Translation
- NAT gateway replaces the source IP addresses of the origin to it's own
- Private subnet to NAT gateway is allowed
    - External resources can't reach private subnet using NAT gateway
    - By default, NAT gateway created in public subnet
- NAT gateway is AWS managed and scales automatically
- NAT gateway is priced on:
    - Each hour it is available
    - Amount of data passed through it

# VPC

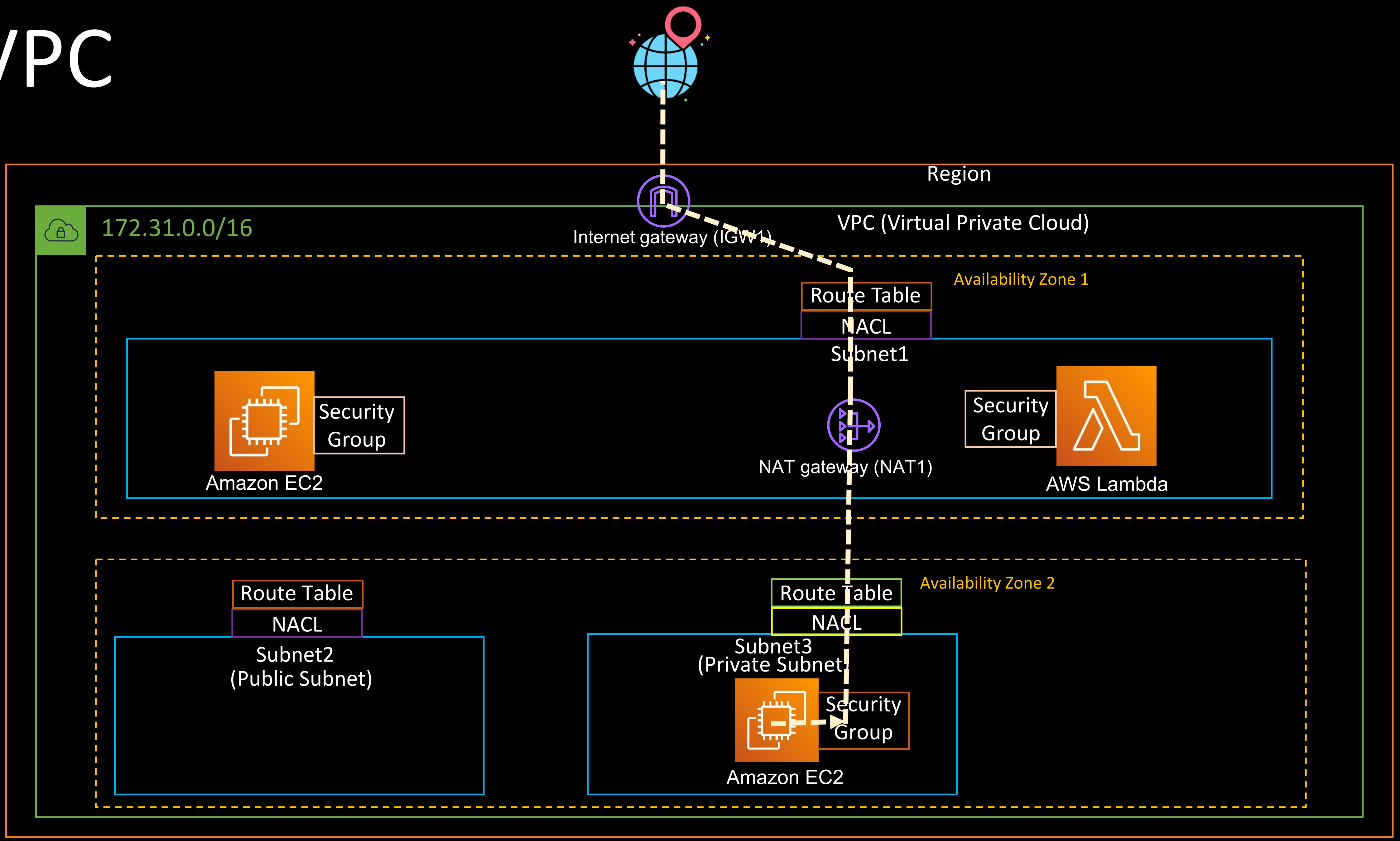

Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
(Public Subnet)
Security Group
Amazon EC2
NAT gateway (NAT1)
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
(Public Subnet)
Route Table
NACL
Subnet3
(Private Subnet)
Security Group
Amazon EC2

# Route Table

Route Table attached to subnet3

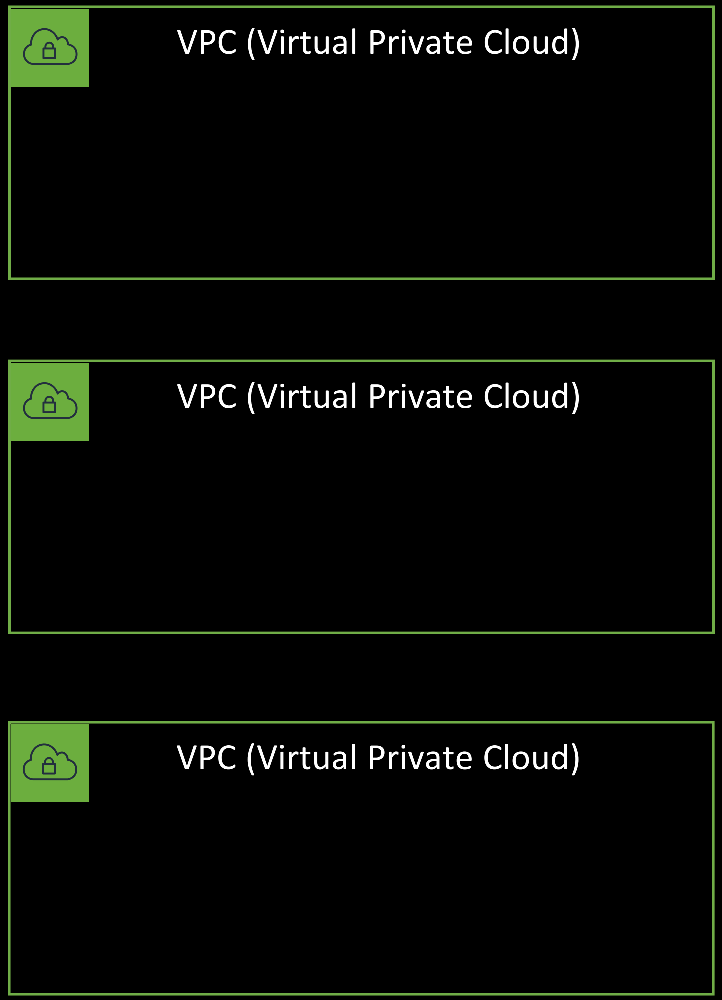
| Destination | Target |
|---|---|
| 172.31.0.0/16 | local |
| 0.0.0.0/0 | NAT1 |

# VPC



Region
172.31.0.0/16
Internet gateway (IGW1)
VPC (Virtual Private Cloud)
Availability Zone 1
Route Table
NACL
Subnet1
Security Group
Amazon EC2
NAT gateway (NAT1)
Security Group
AWS Lambda
Availability Zone 2
Route Table
NACL
Subnet2
(Public Subnet)
Route Table
NACL
Subnet3
(Private Subnet)
Security Group
Amazon EC2

# What Does All These Buy YOU?




- Creates logically isolated virtual network
    - app separation
    - dev, test, prod separation
- Logical security boundary
    - Choose IP range
    - NACL, subnet, route tables
- Enable detailed monitoring and logging for specific VPCs
- Create “no traffic via internet” architectures
- On-prem to AWS connectivity

# Demo - VPC

Raj Saha
cloudwithraj.com
Cloud With Raj

# High Availability Application

Raj Saha
cloudwithraj.com
Cloud With Raj

# High Availability

- System continues functioning even when some of its components fail

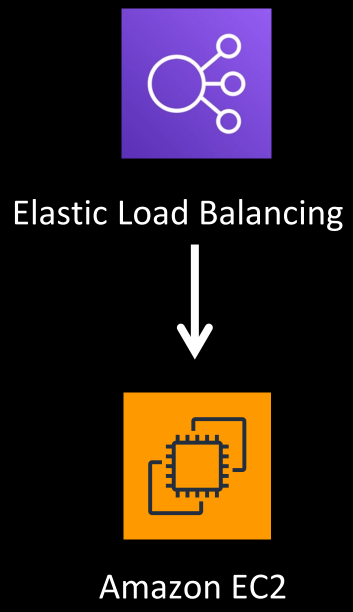
- System guarantees certain percentage of uptime

- Application runs in minimum two AZs

# Identifying Single Point of Failure

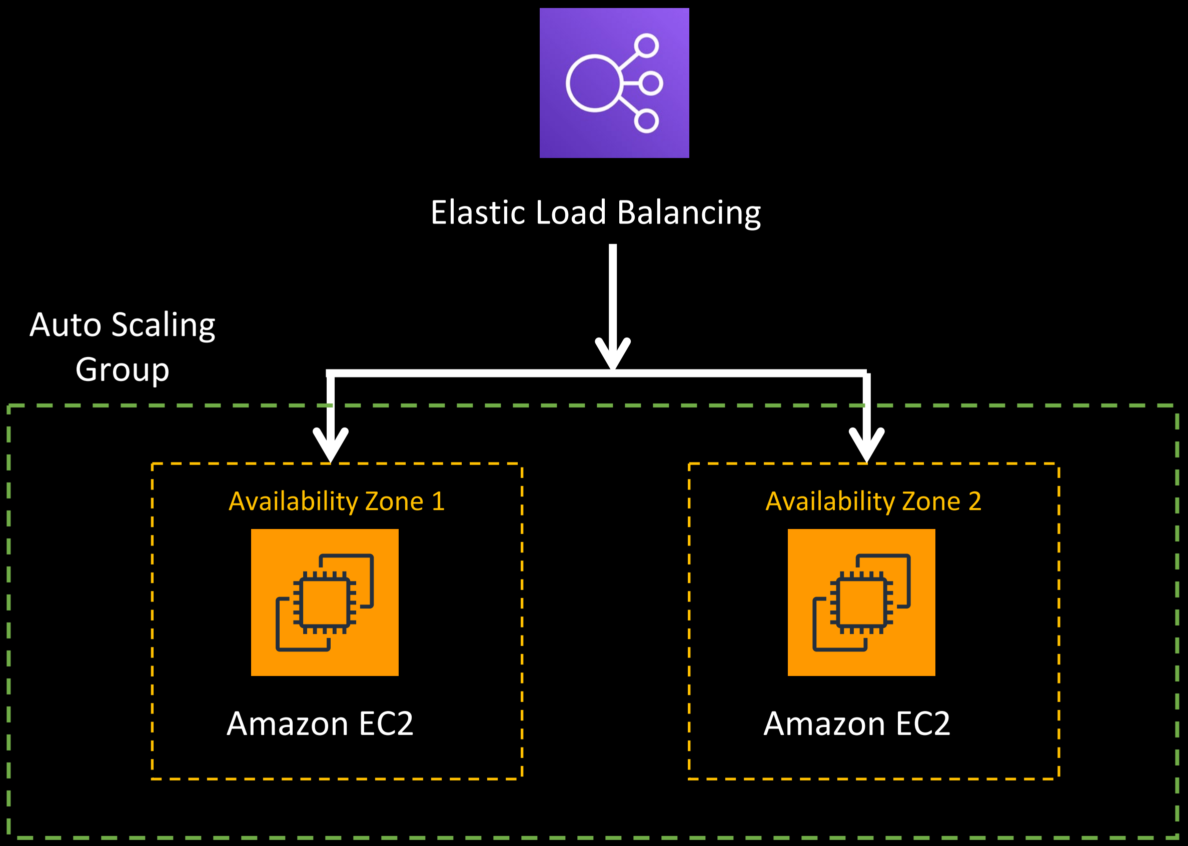
- Servers running your applications
- Database
- Load balancer
- Analyze each component and validate single point of failure

# Achieving High Availability on Cloud



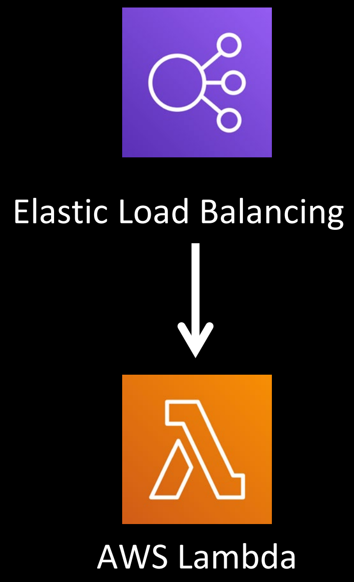

- Elastic Load Balancer is inherently highly available (managed by Cloud Provider)
- Auto Scaling Group makes the server scalable, not highly available
    - There is a delay to spin server up

# Achieving High Availability on Cloud



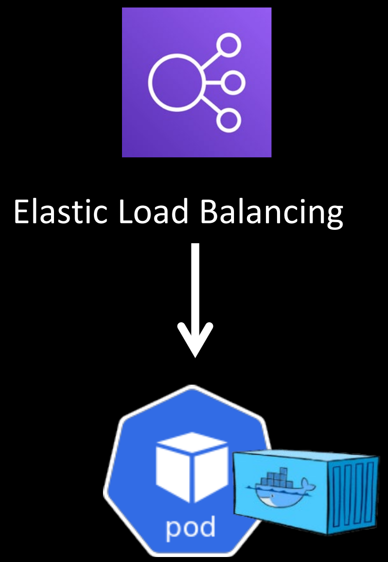

- Achieve high availability but costs extra money
- What is an option which is automatically highly available i.e. HA managed by Cloud Provider?

# Achieving High Availability on Cloud

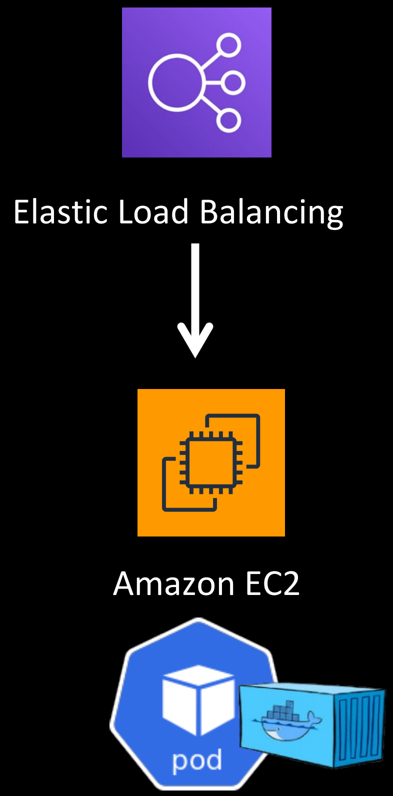



- How about Kubernetes?

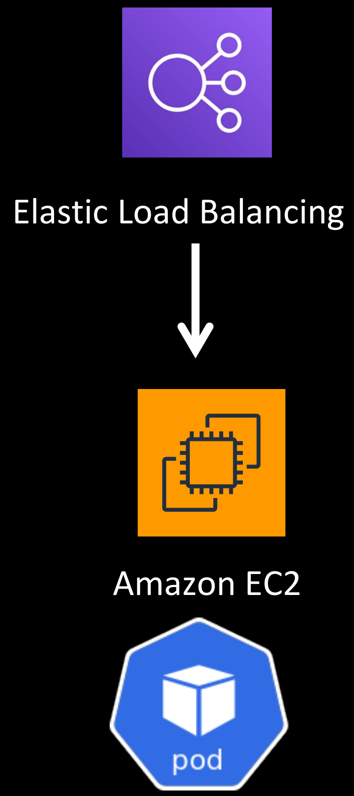
# Achieving High Availability on Cloud



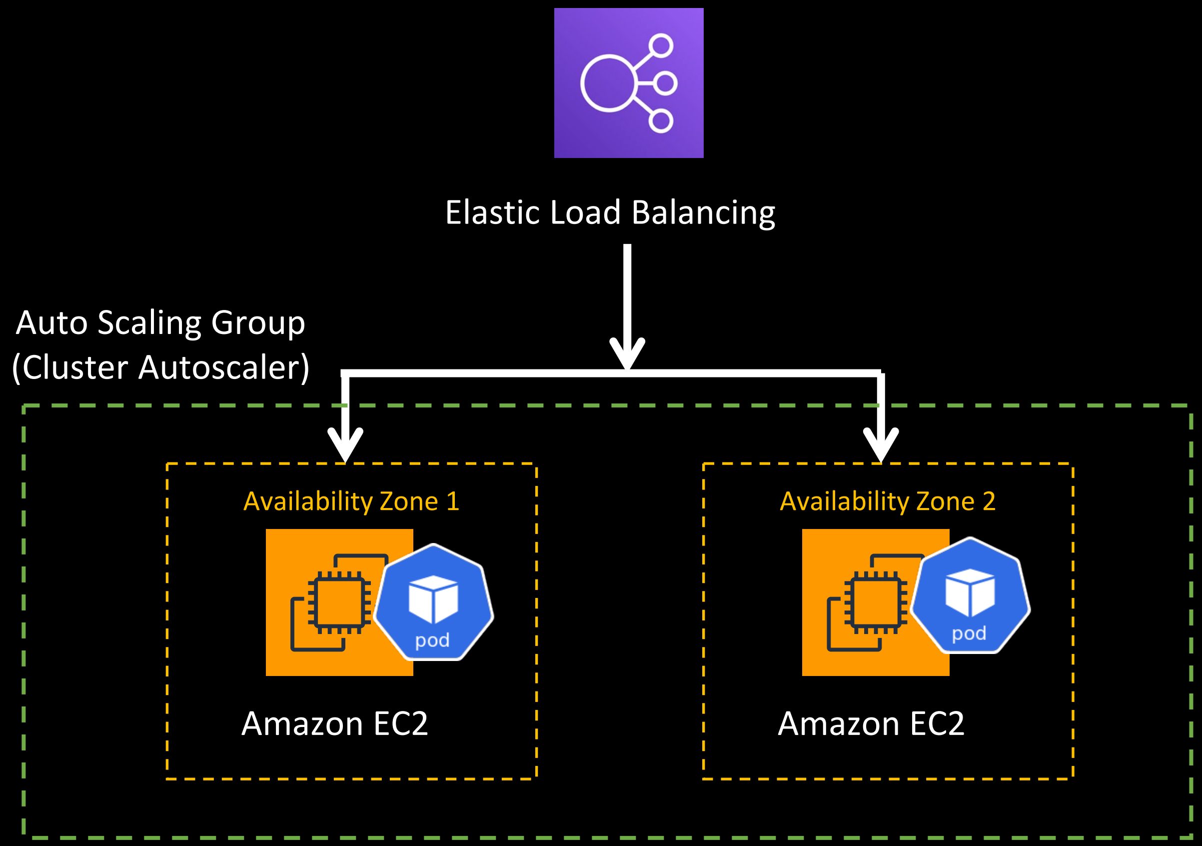
# Achieving High Availability on Cloud



# Achieving High Availability on Cloud



# Achieving High Availability on Cloud

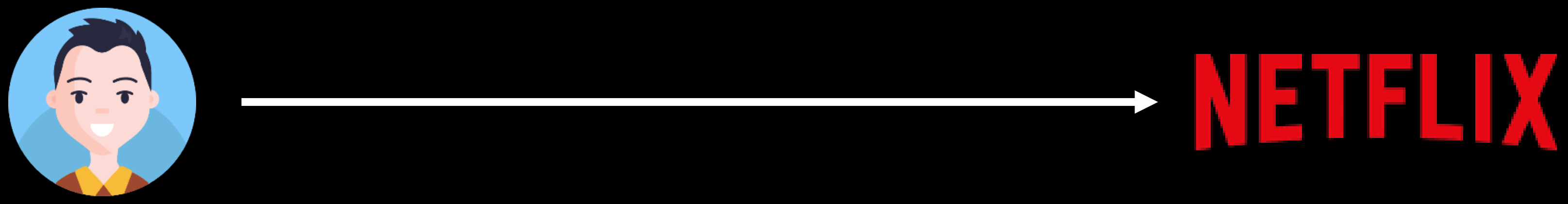



- Don’t over index on cost when you design or answer interview question

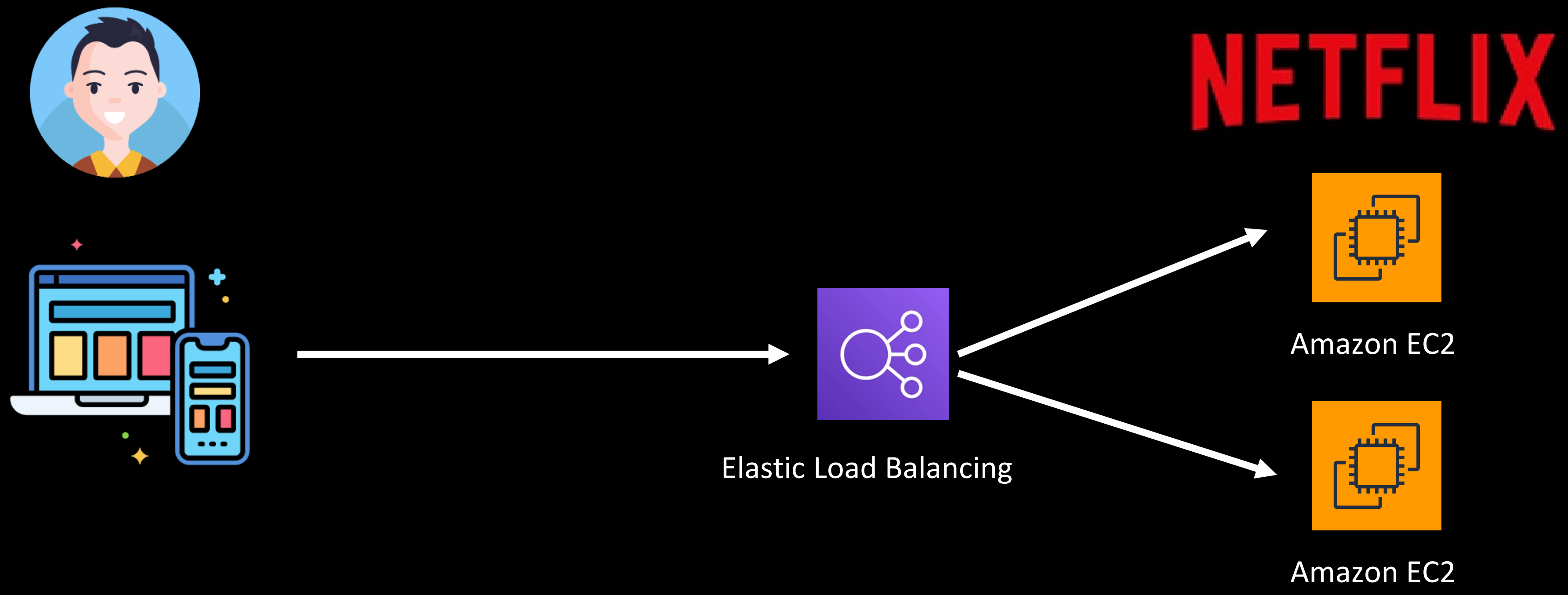
# OSI (Open System Interconnection) Layer

Raj Saha
cloudwithraj.com
Cloud With Raj

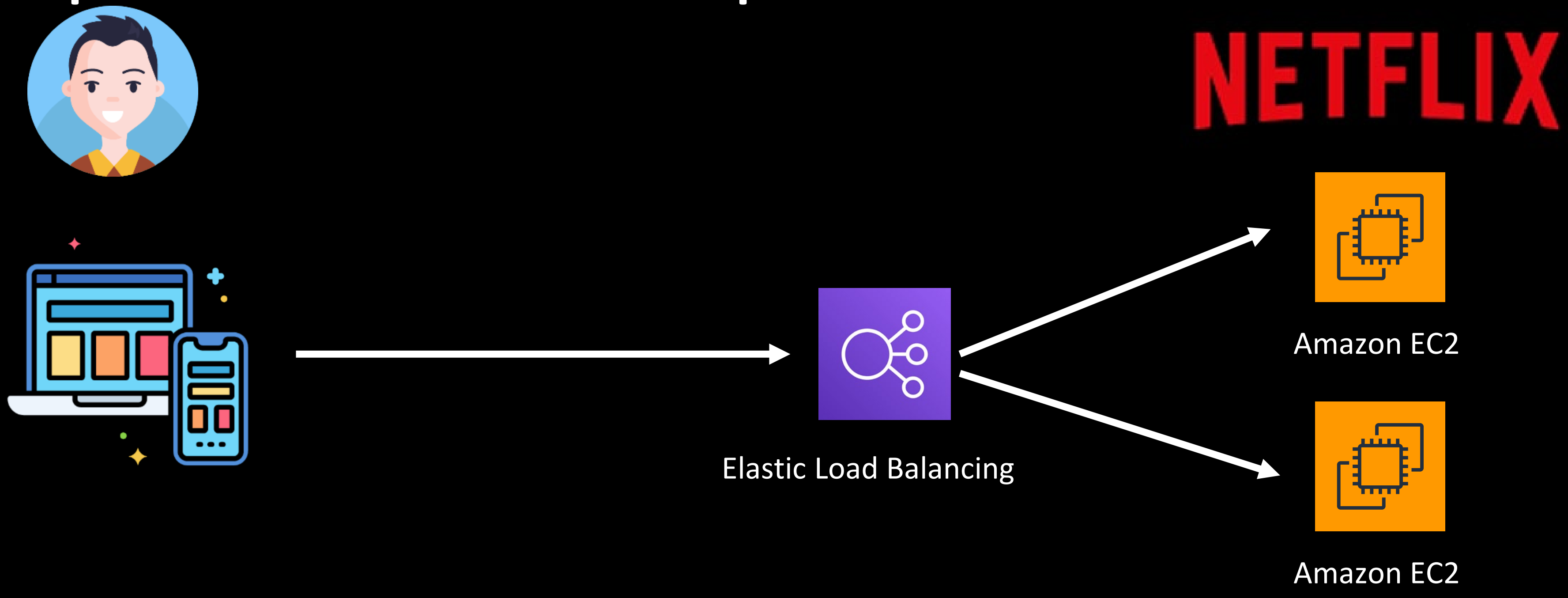
# Computer to Computer Interaction



# Computer to Computer Interaction

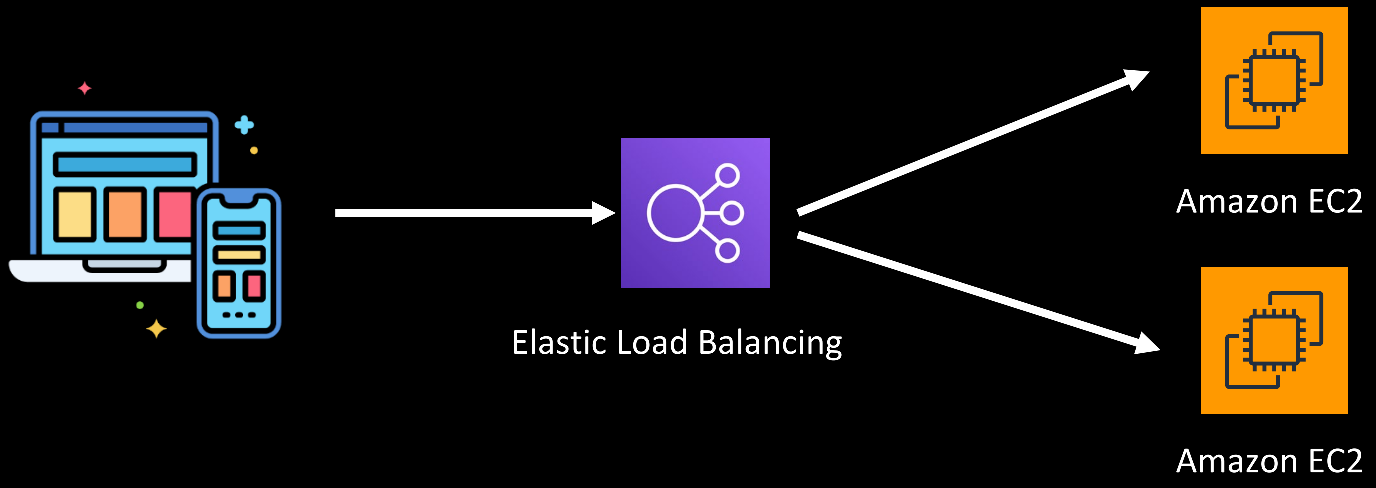


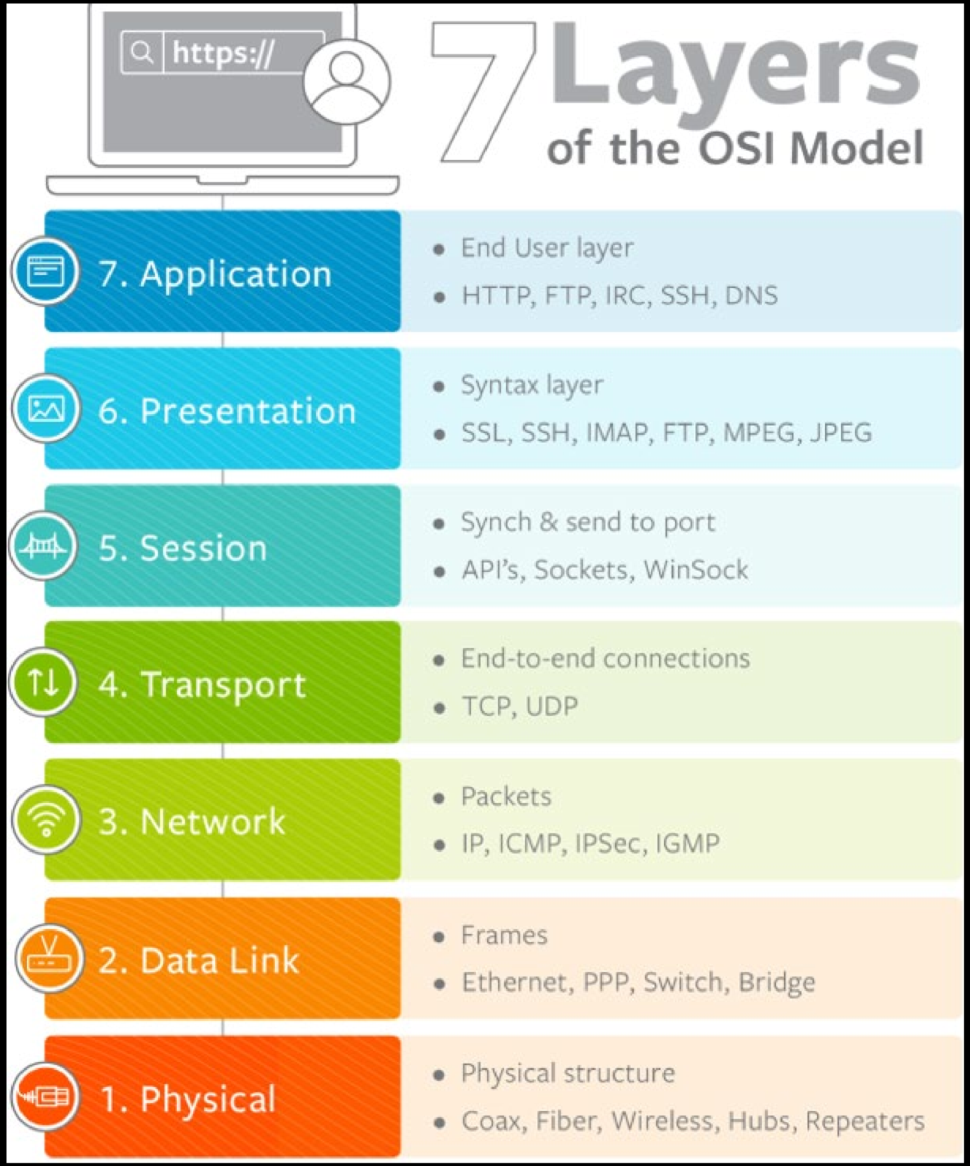
# Computer to Computer Interaction



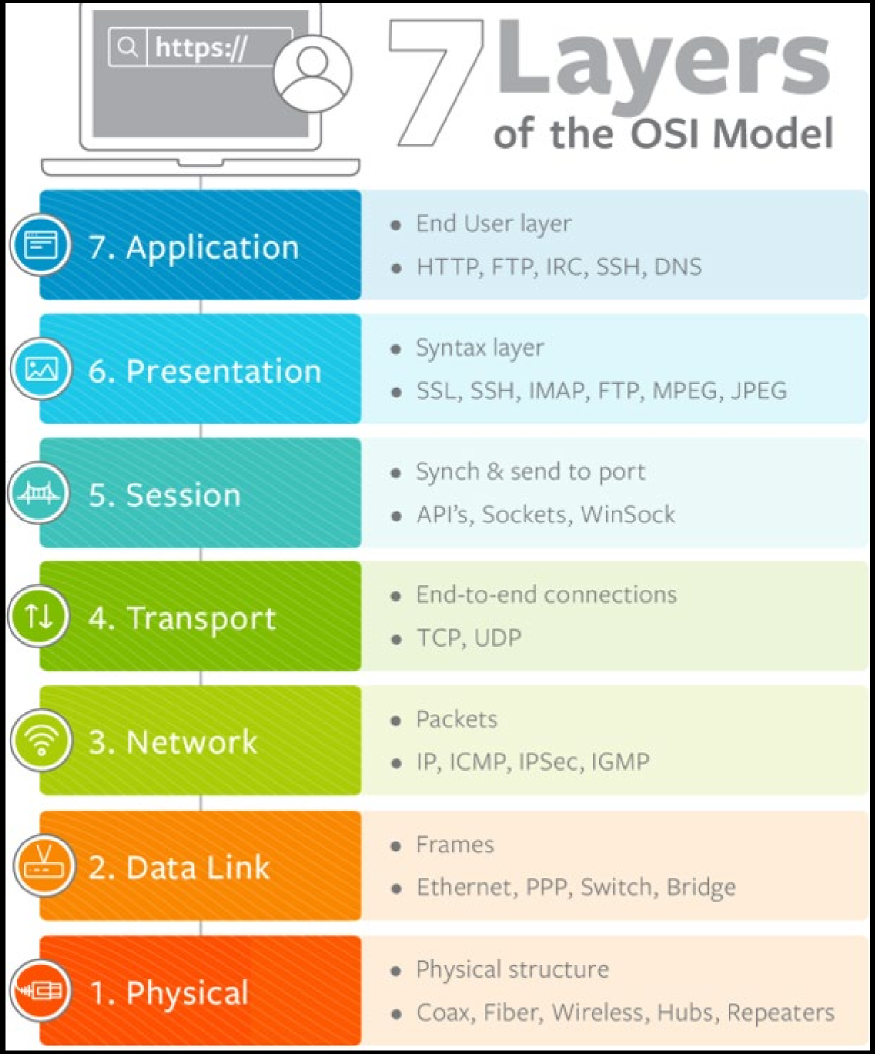

- Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network

# OSI 7 Layers (Memorize for Interviews)



# OSI 7 Layers (Memorize for Interviews)




- Different services operate at different layer
- L7 - Application Load Balancer, service mesh
- L4 – Network Load Balancer, Kubernetes network policy
- You will NOT be asked advanced questions on OSI for cloud devops interview
- Study in-depth if going for networking specialty

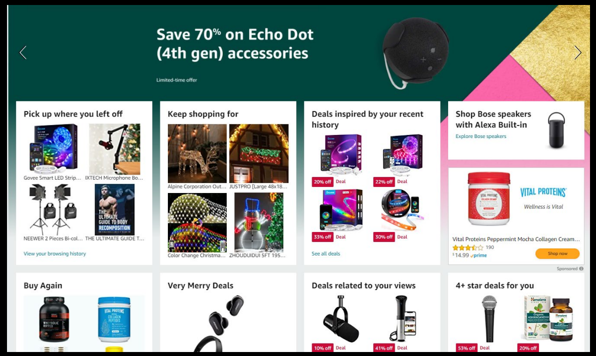
# What Happens When You Type an URL

Raj Saha
cloudwithraj.com
Cloud With Raj

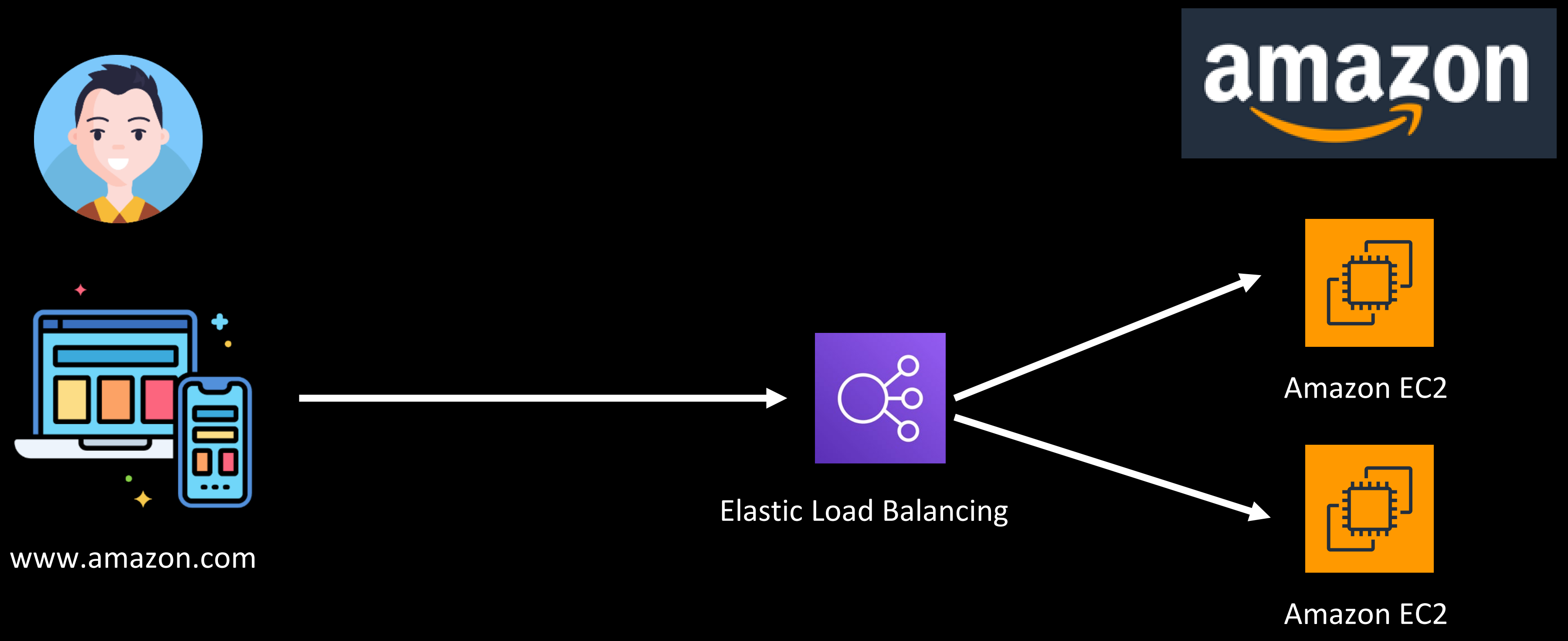
# Type an URL

www.amazon.com

amazon

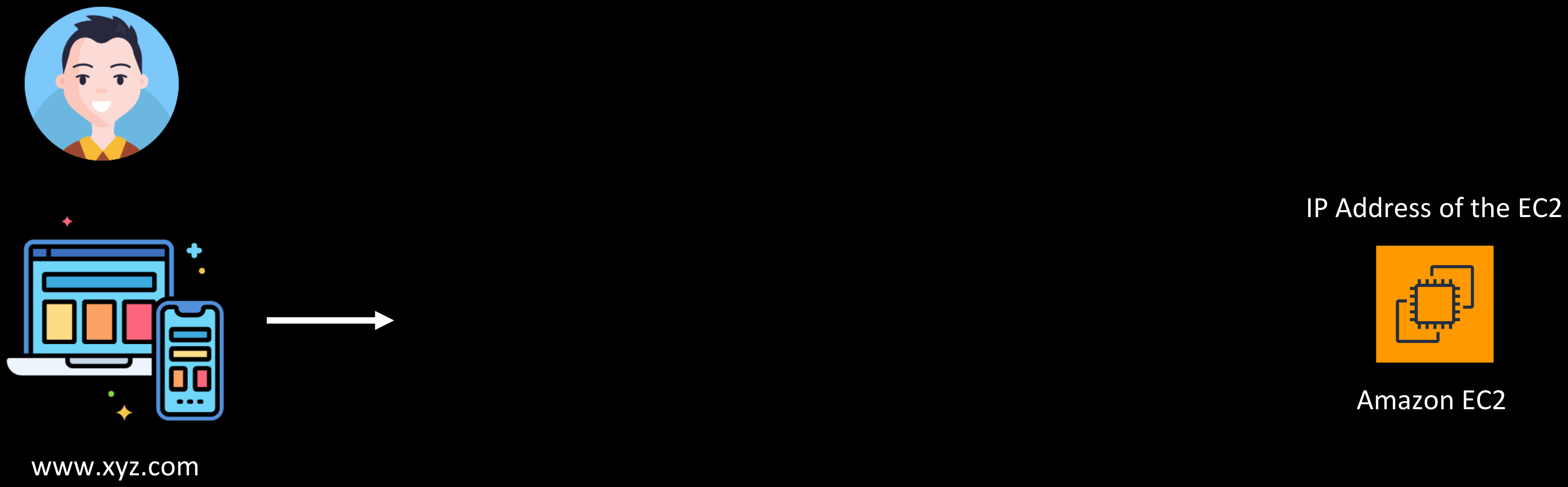


# Computer to Computer Interaction

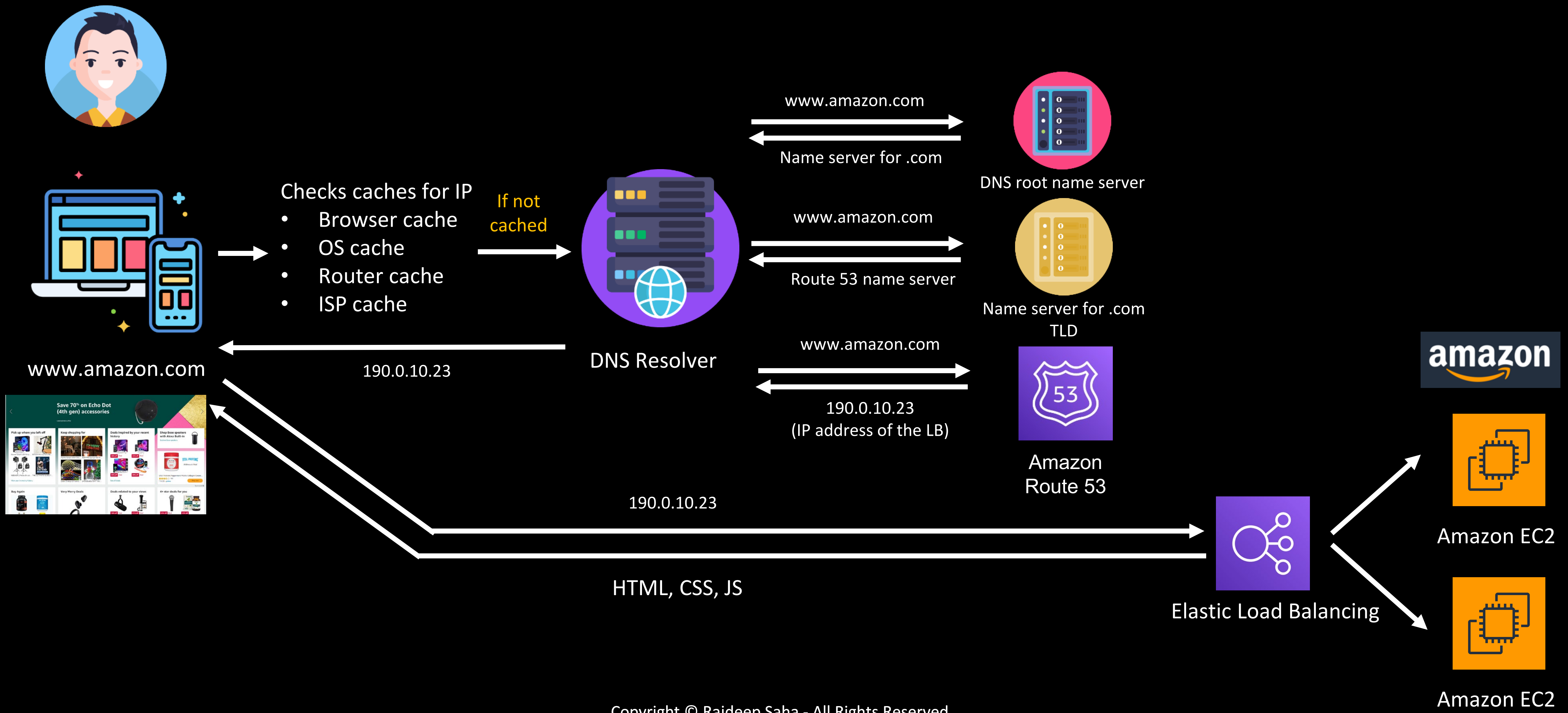


# Computer to Computer Interaction

# Computer to Computer Interaction



# Computer to Computer Interaction



# Disaster Recovery – What and Why

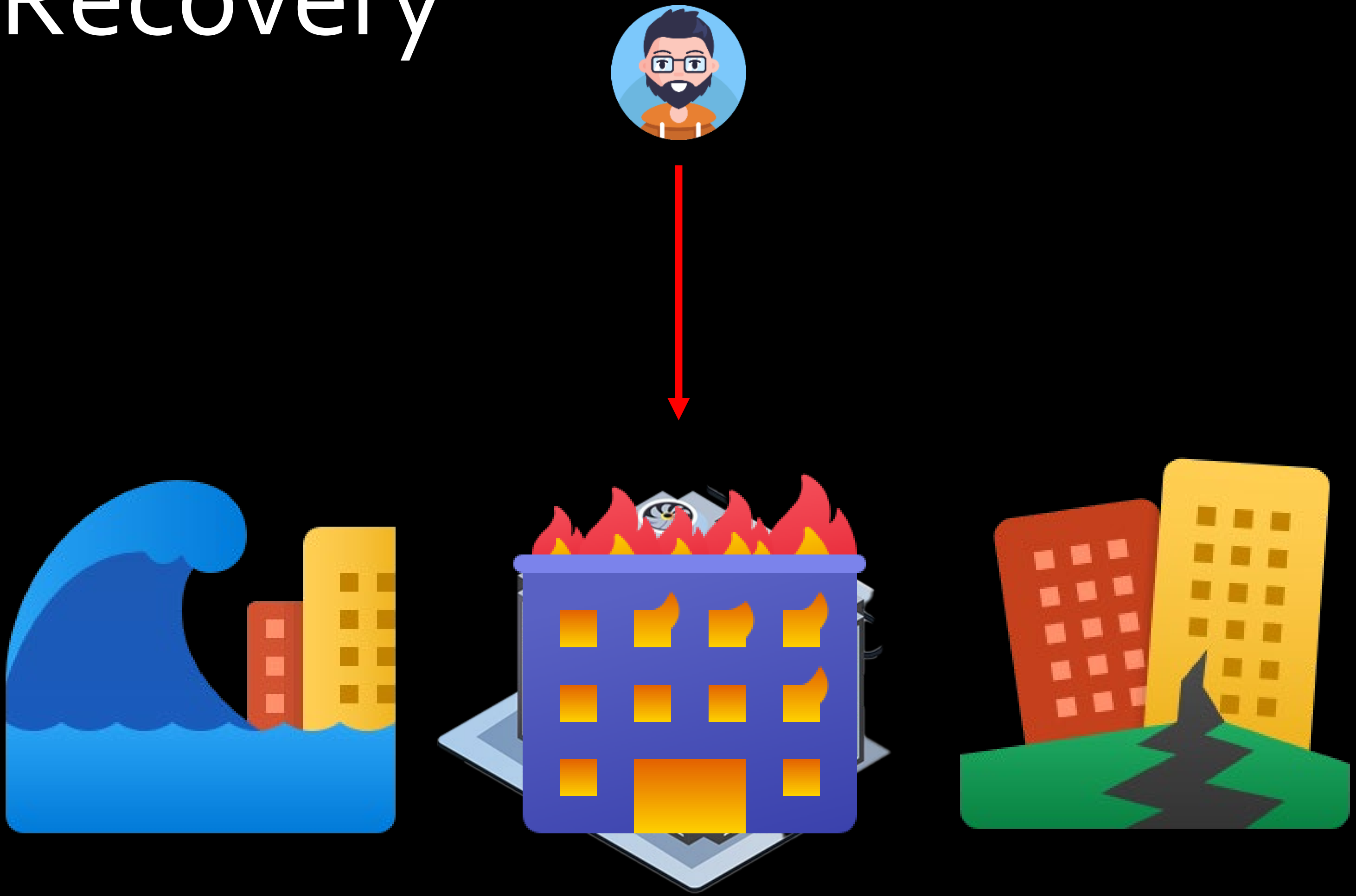
Raj Saha
cloudwithraj.com
Cloud With Raj

# Disaster Recovery


Data Center

- Physical building
- Electricity, cooling, manpower

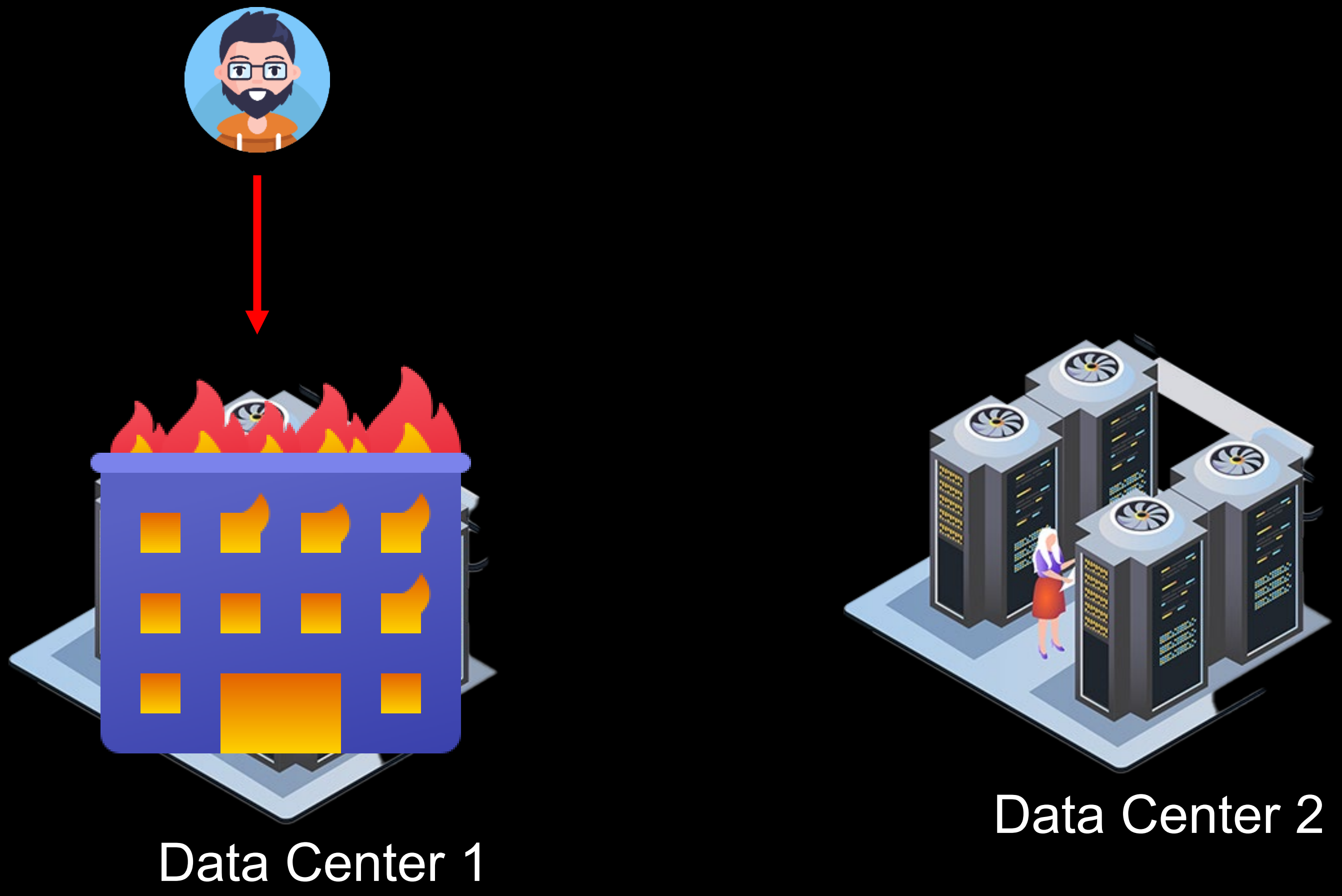
# Disaster Recovery



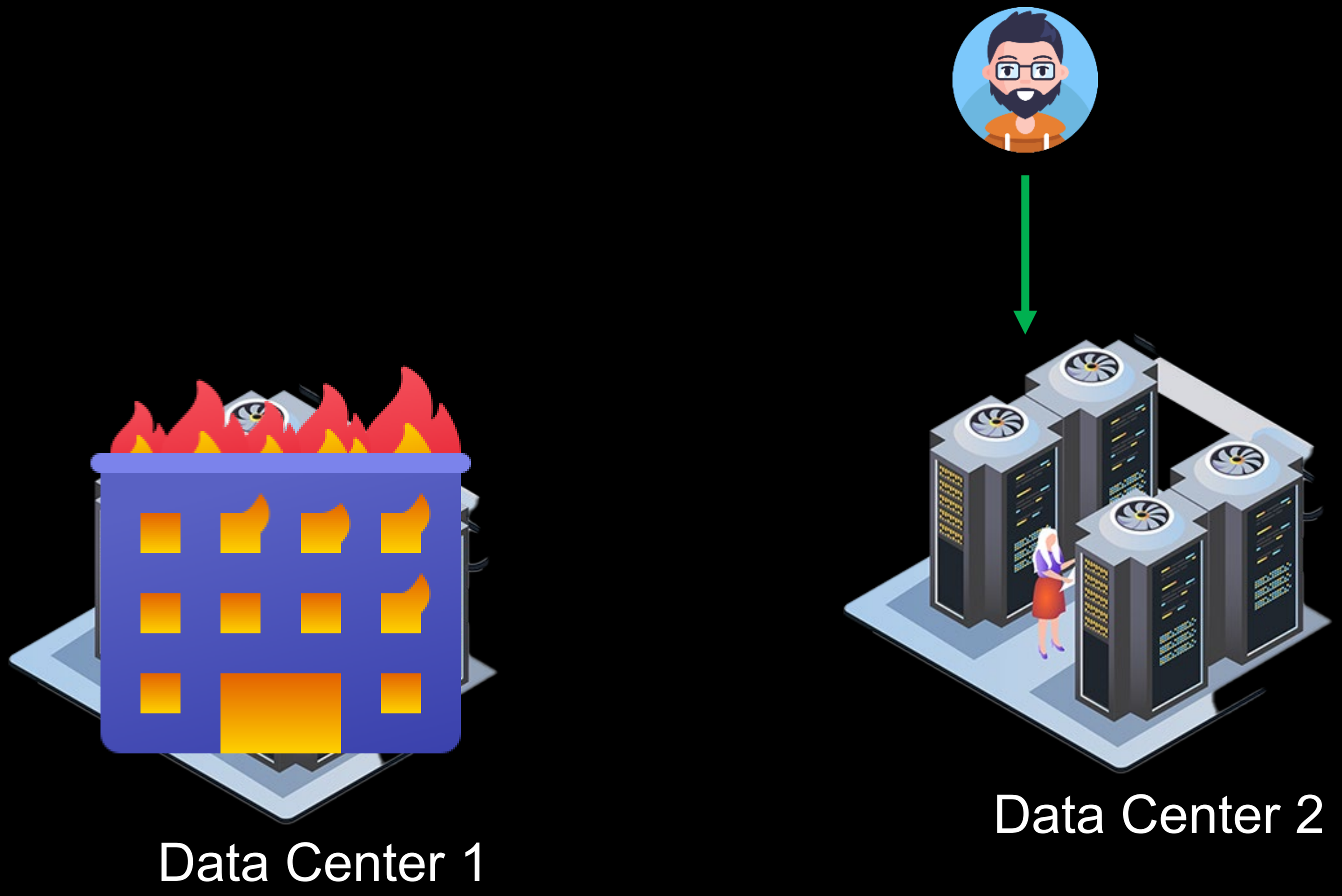
Data Center

- Disasters disrupting your application
- Disaster also extends to non-natural events
  - Security issues
  - Covid-19 etc

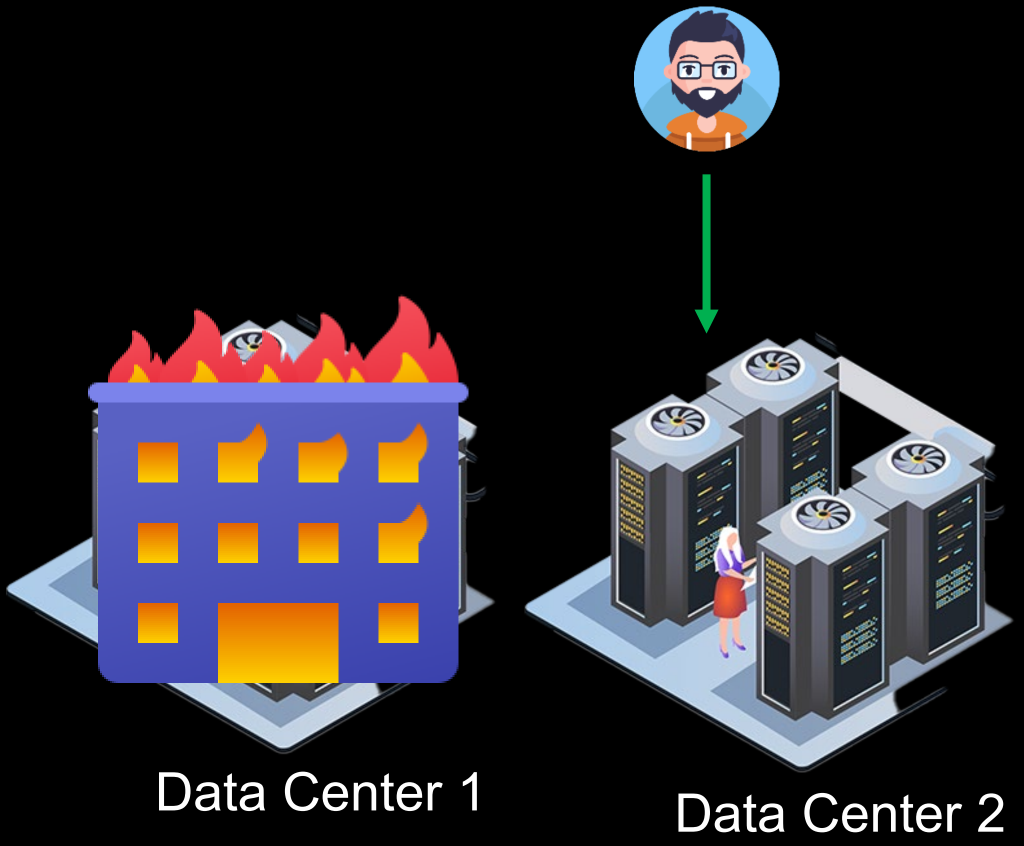
# Disaster Recovery



# Disaster Recovery



# Disaster Recovery



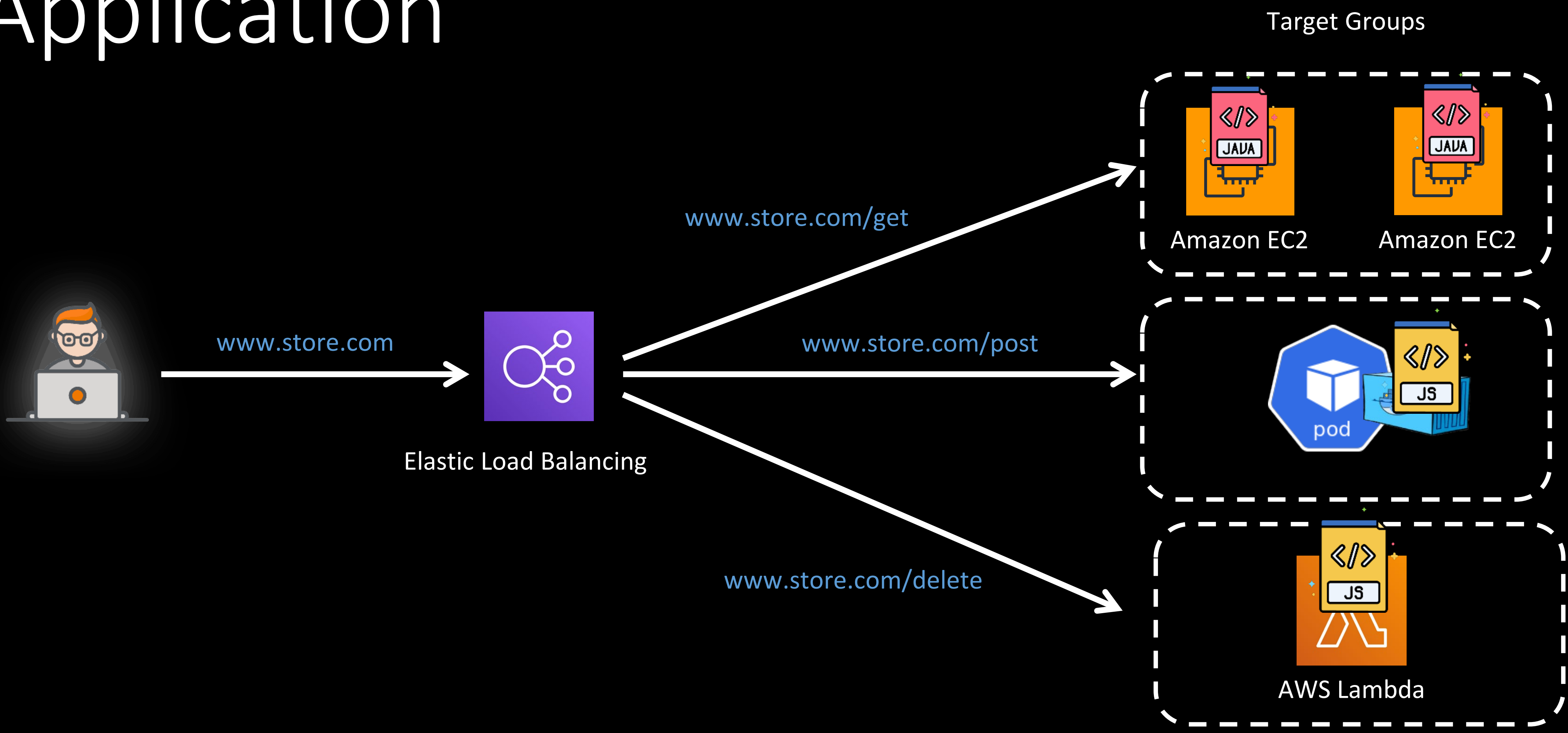

- Disaster Recovery (DR) is the process by which an organization anticipates and addresses technology-related disasters
- DR ensures business continuity
- DR enhances system security
- DR improves customer retention
- DR reduces recovery costs

# AWS Storage and Databases

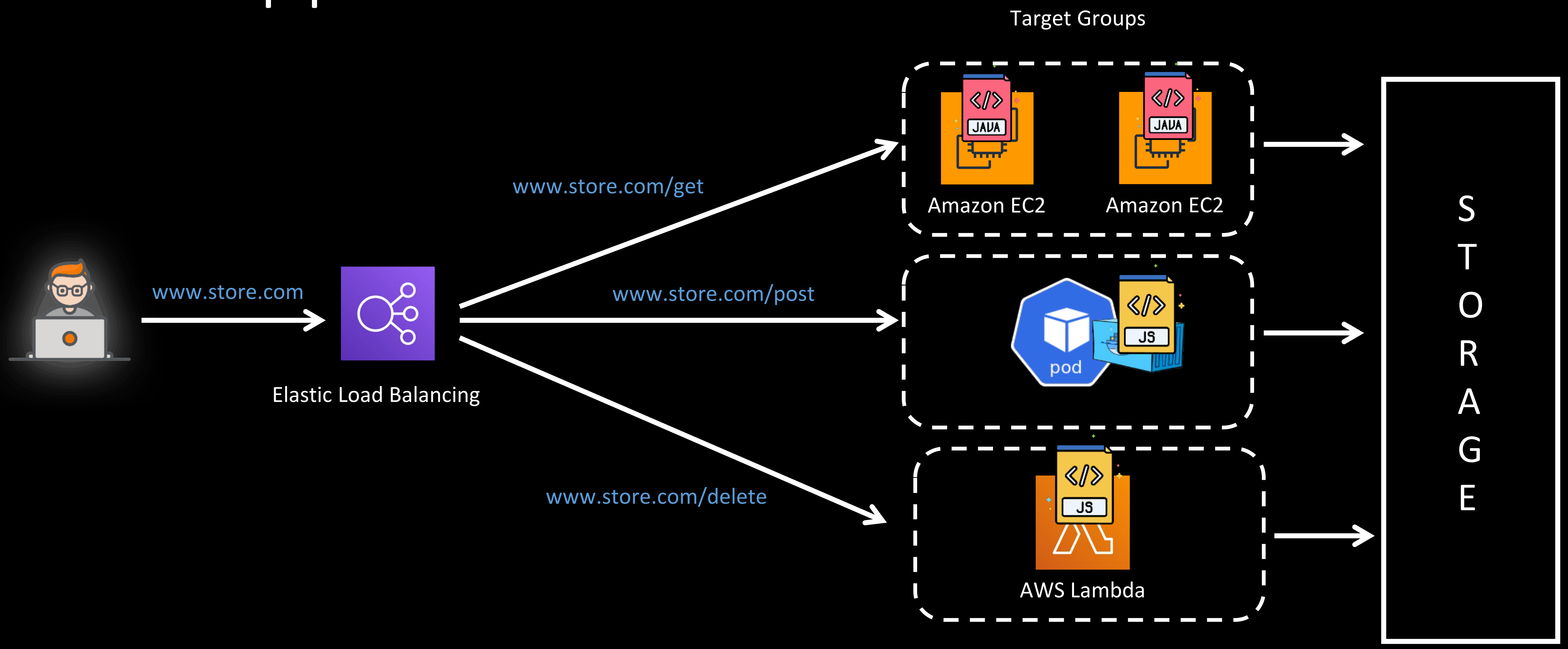
# Storage – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

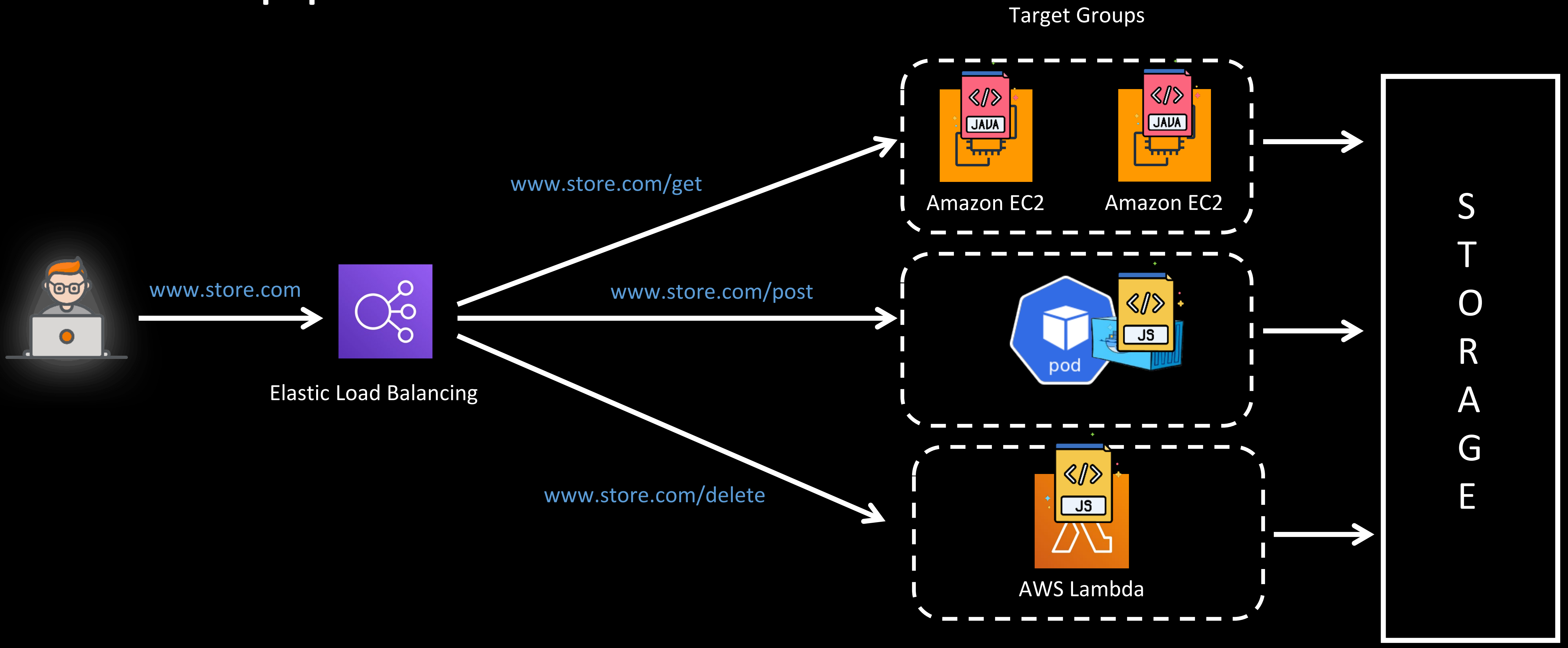
# Your Application



# Your Application

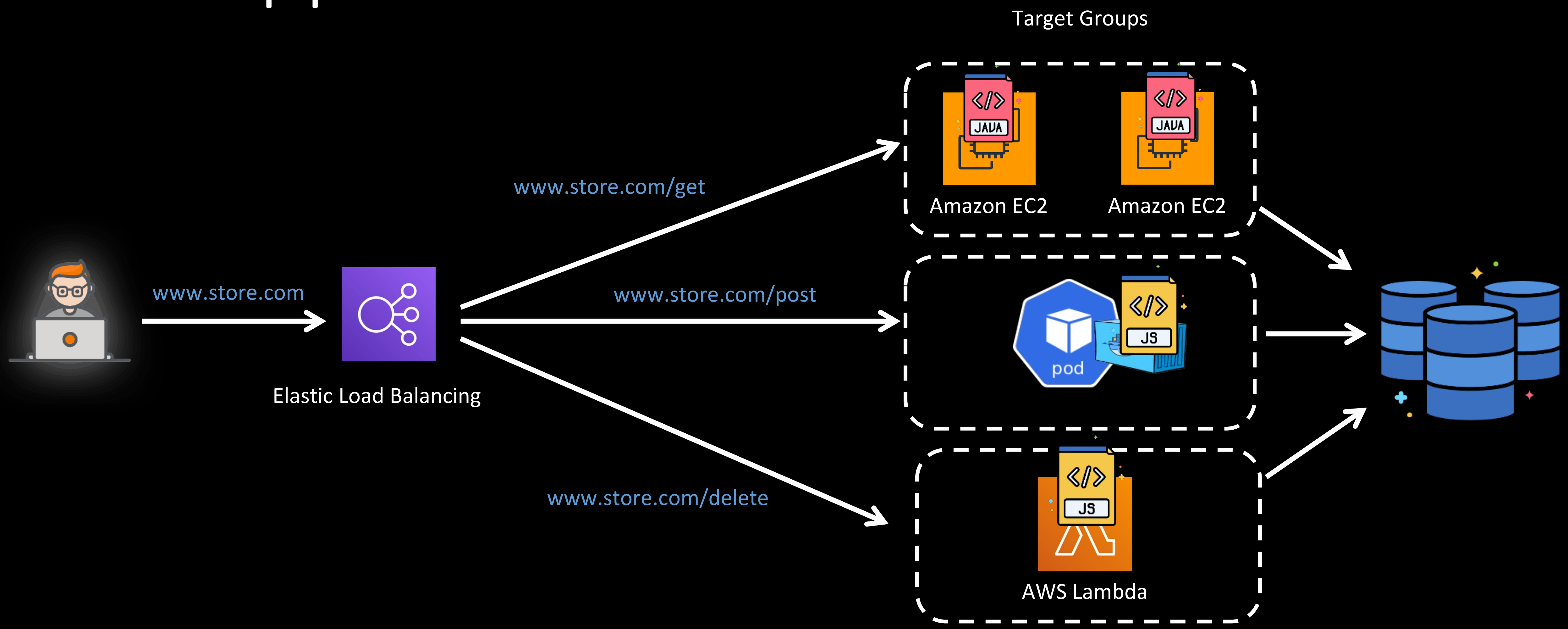


# Your Application

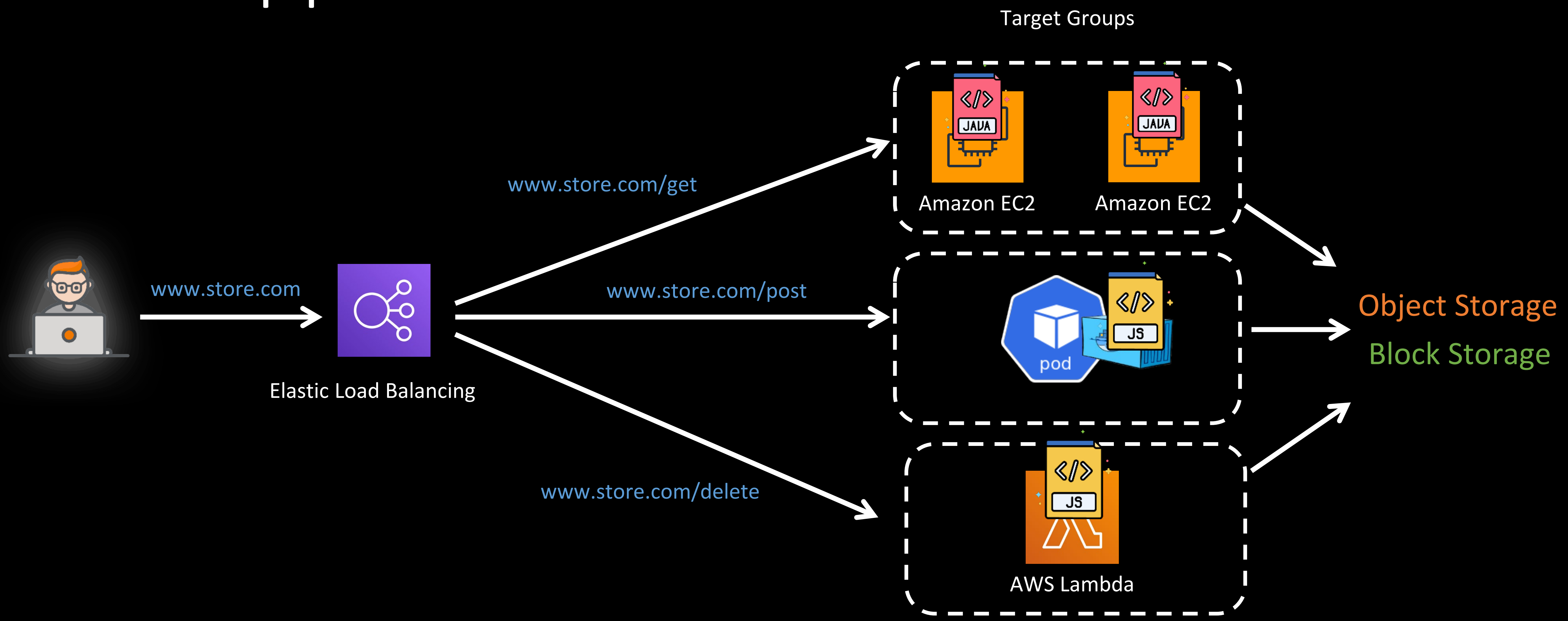



- Images
- Item info, prices
- Shopping cart

# Your Application



# Your Application



# Object vs Block vs Distributed vs File Storage

Raj Saha
cloudwithraj.com
Cloud With Raj

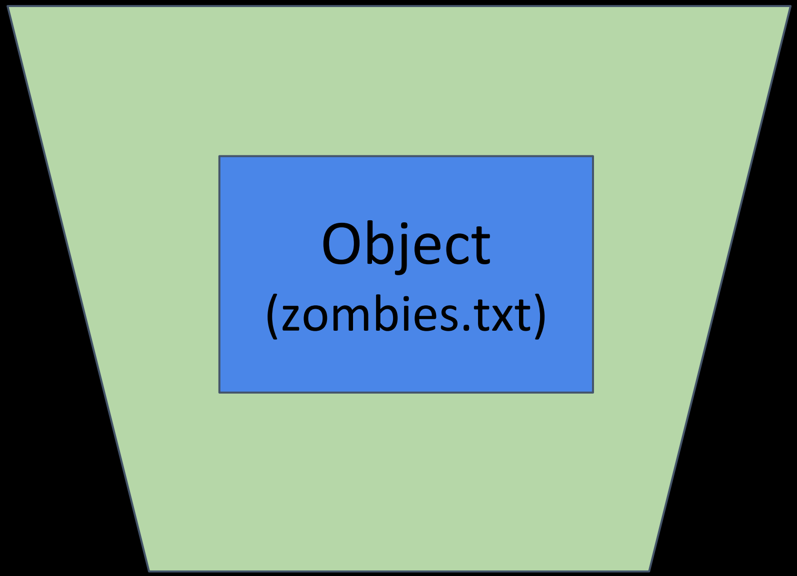
# In The Beginning..

# Object

Object (Literally any file)

Text files, Image File, Music, Movie etc.

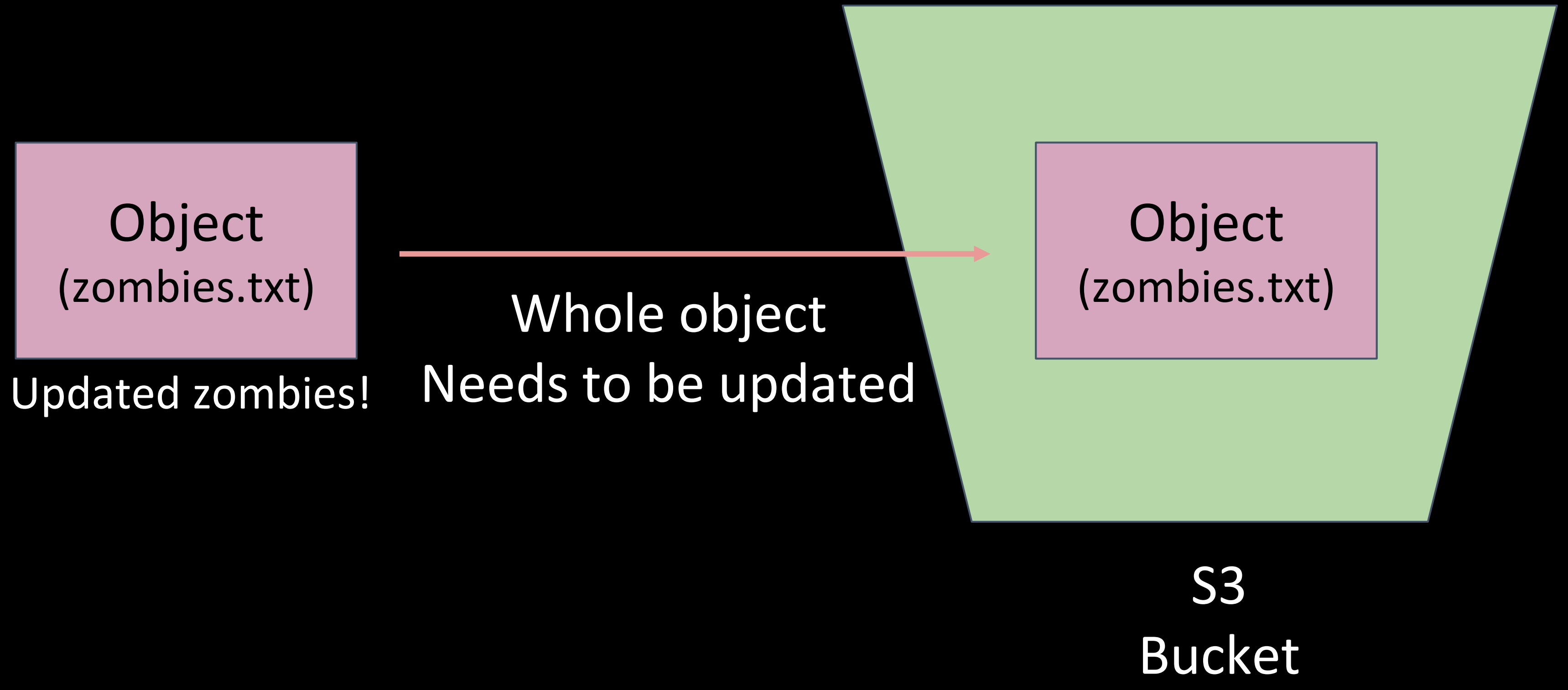
# Object Storage



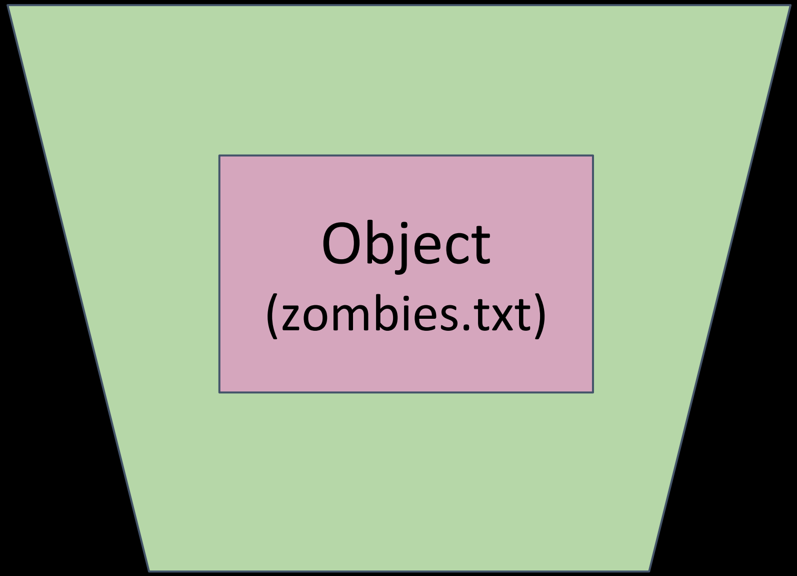

S3 Bucket

- Simple Storage Service
- Storage through web service interface
- Highly Scalable, Available and Durable

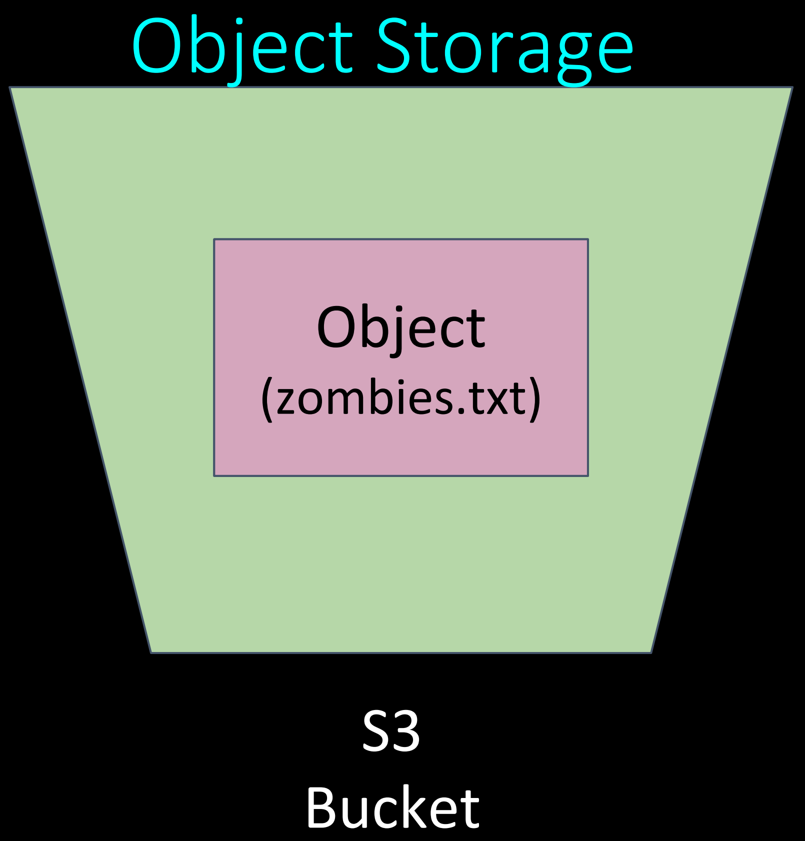
# Object Storage



# Object Storage



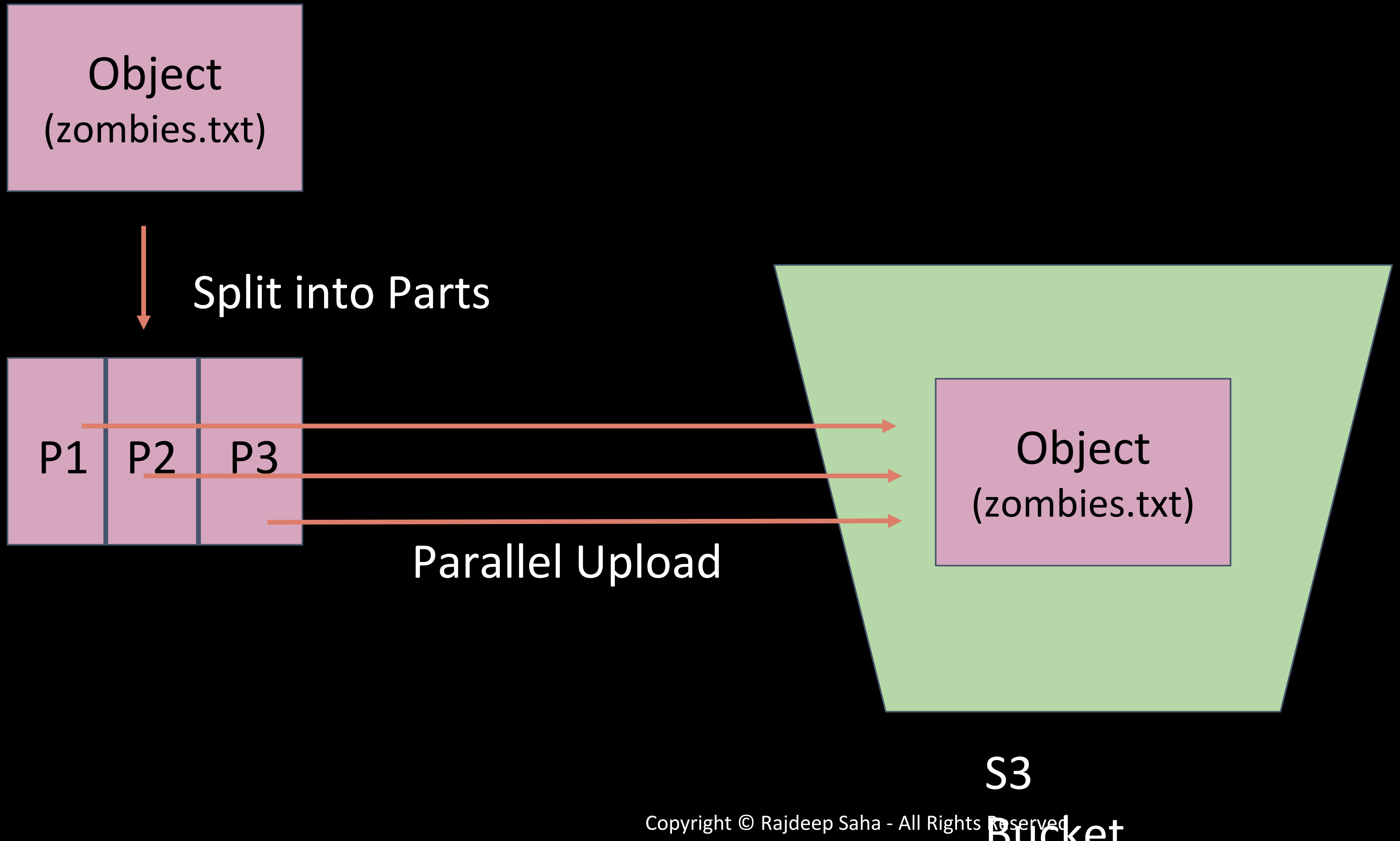

S3
Bucket




- For large objects, time consuming

# Object Storage MultiPart Upload/Download

## (Mention to Impress)



# Object Storage MultiPart Upload/Download

## (Mention to Impress)

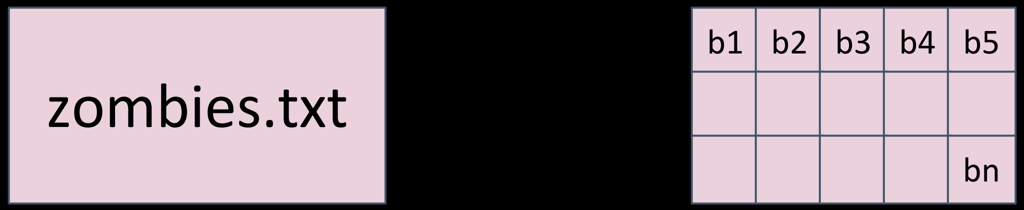
- Makes upload/download parallely and faster
- User has to coordinate
  - Breaking into parts
  - Calling API appropriately with parts. Handle abort etc.
- Recommended for objects larger than 100 MB

# Block

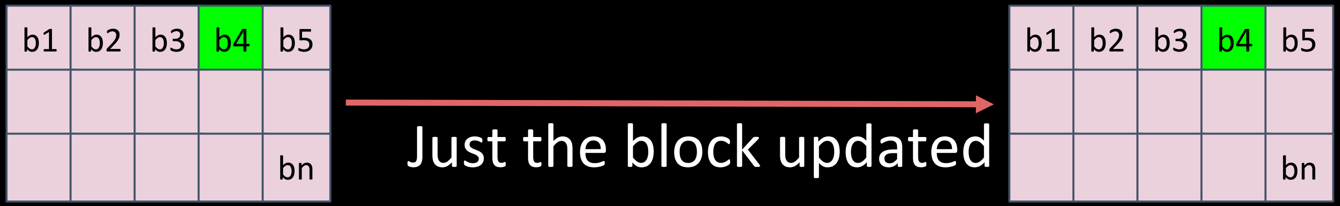
Literally any file

| | | | | |
|---|---|---|---|---|
| b1 | b2 | b3 | b4 | b5 |
| | | | | |
| | | | | bn |

# Block Storage



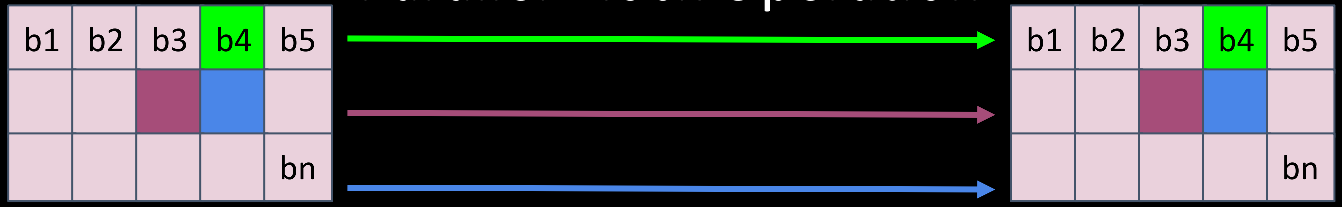
# Block Storage




Updated zombies!

# Block Storage

Parallel Block Operation



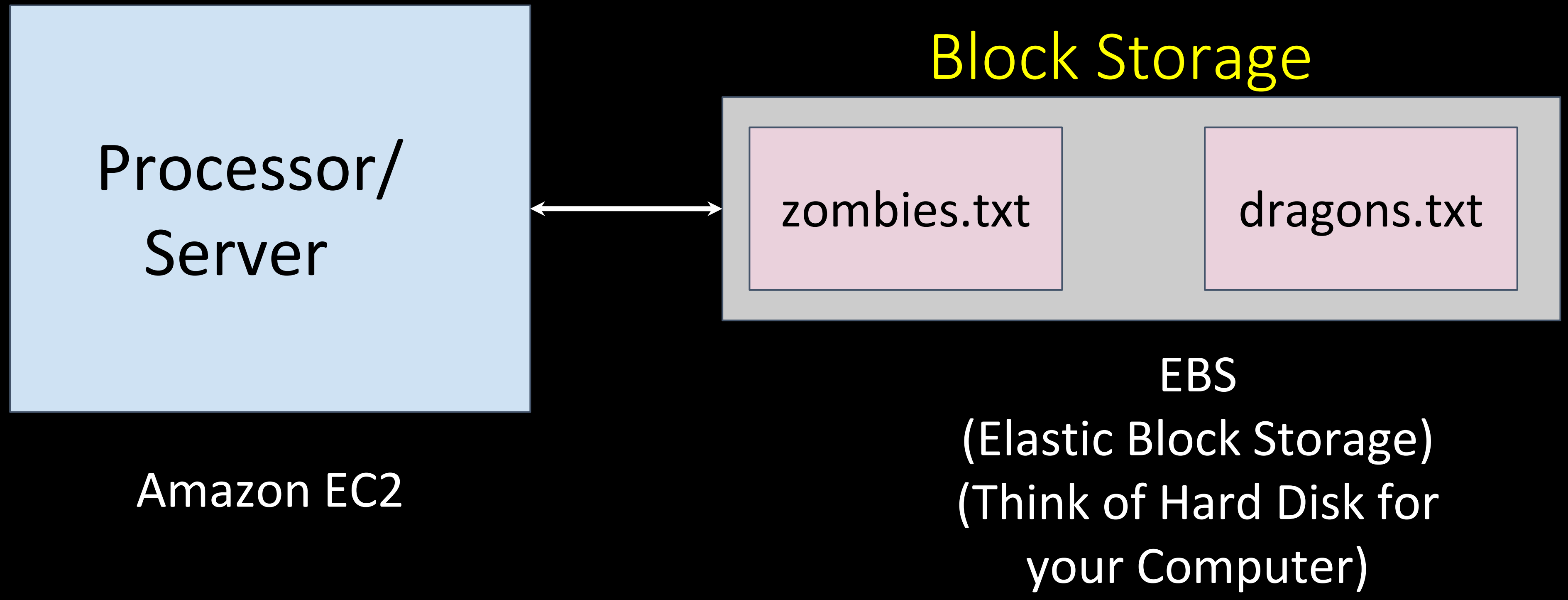

Updated zombies!

# Block Storage

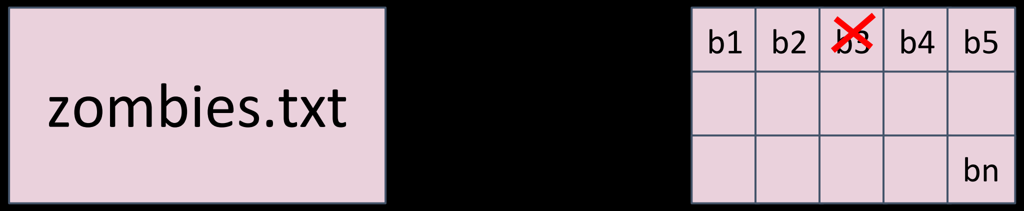

Updated zombies!

- Block level operation is faster than object level
- Automatic behind the scene, YOU focus on zombies only!
  (remember, multipart object upload/download has overhead)
- Default block size 256 KB
- SAN (Storage Area Network)

# How Do I access EBS?

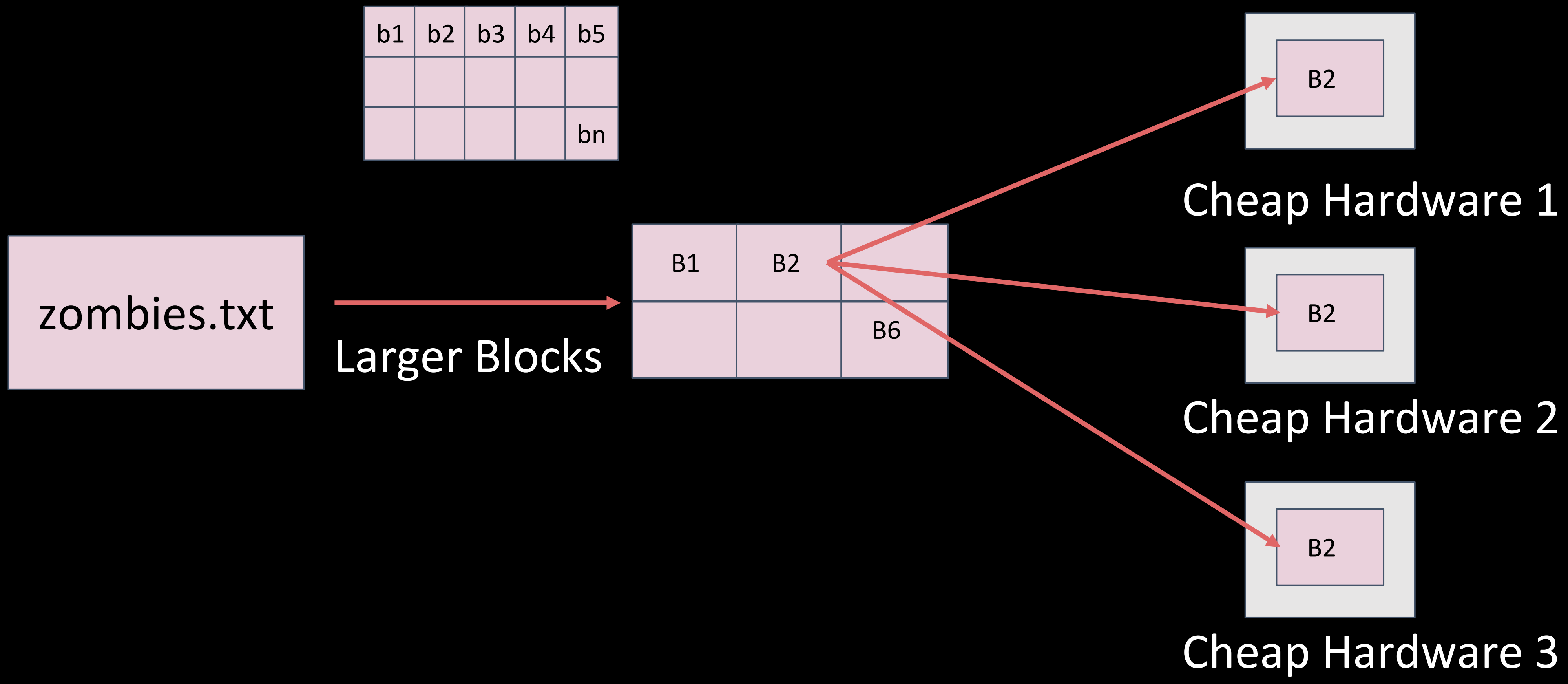


# Distributed File System

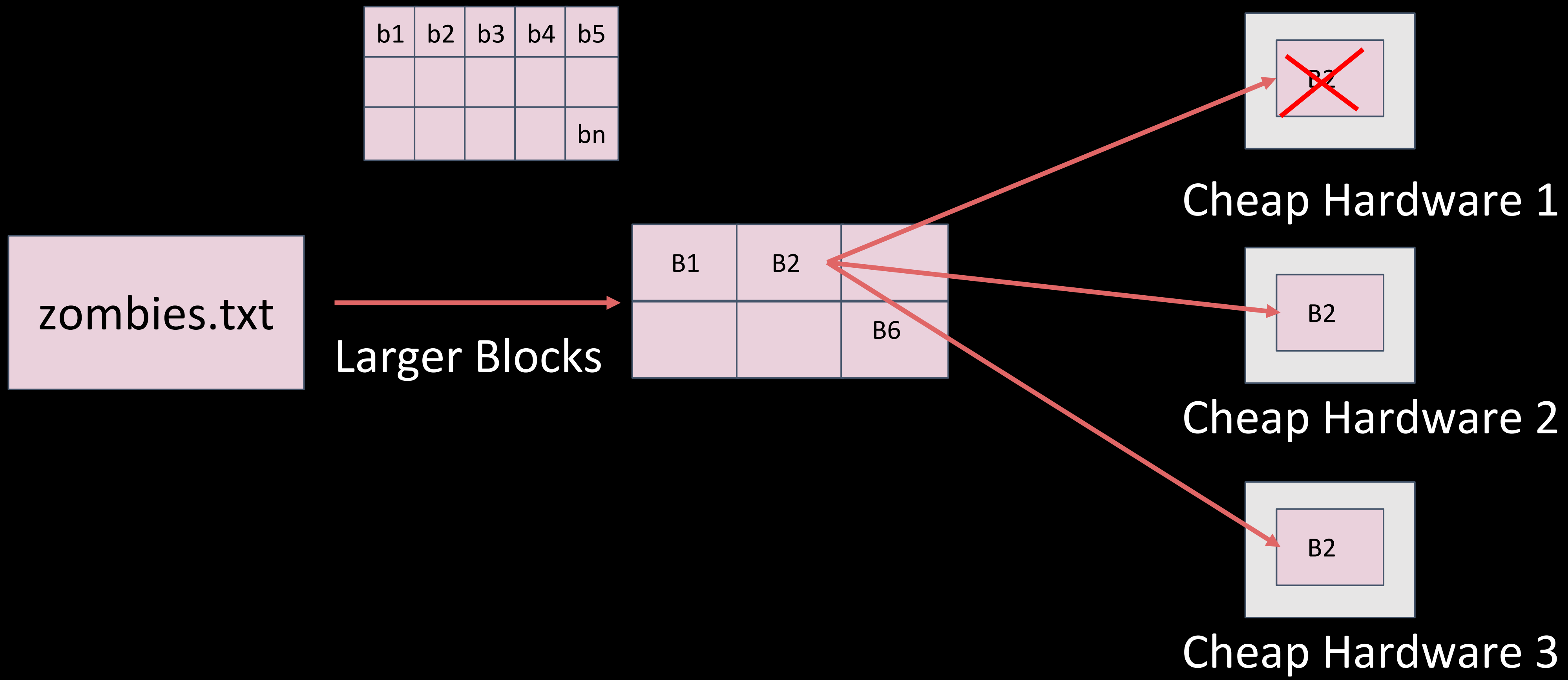



- Block Storage can be expensive
- How do you store lots of data cheaper and durable fashion?

# Distributed File System



# Distributed File System



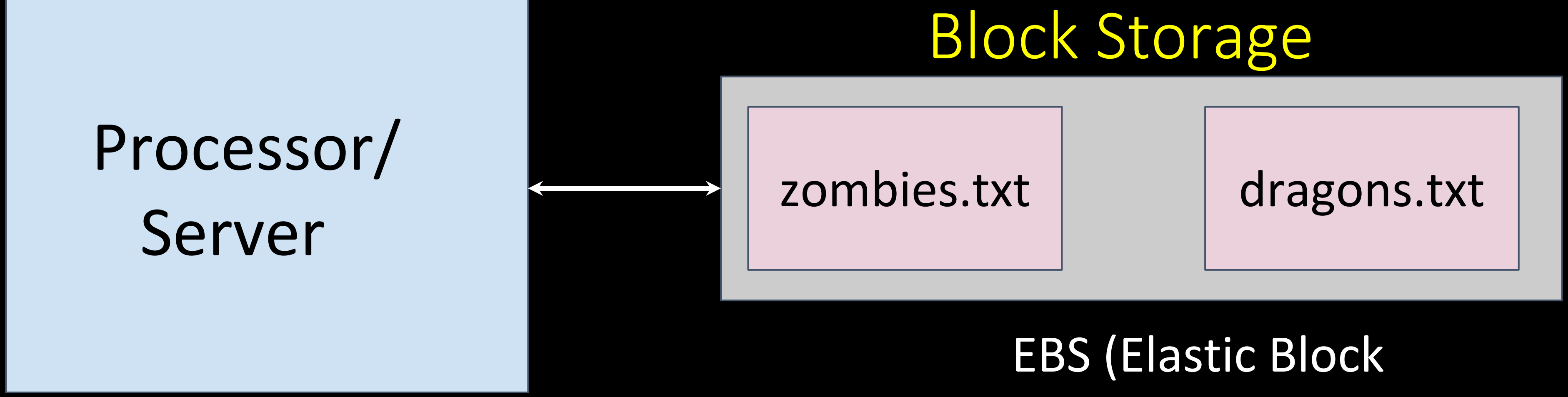
# Distributed File System

- File blocks Distributed over commodity hardware
- System automatically replicates
- Block size way larger than EBS - 128MB
- Suited for big file - remember smallest size that can be updated is block!

# Hadoop Distributed File System

- Highly Scalable, Fault Tolerant
- Supports parallel reading and processing - using Map Reduce
- Can be queried using Hive
- Utilized is AWS Services - Athena to query S3, Amazon EMR etc.

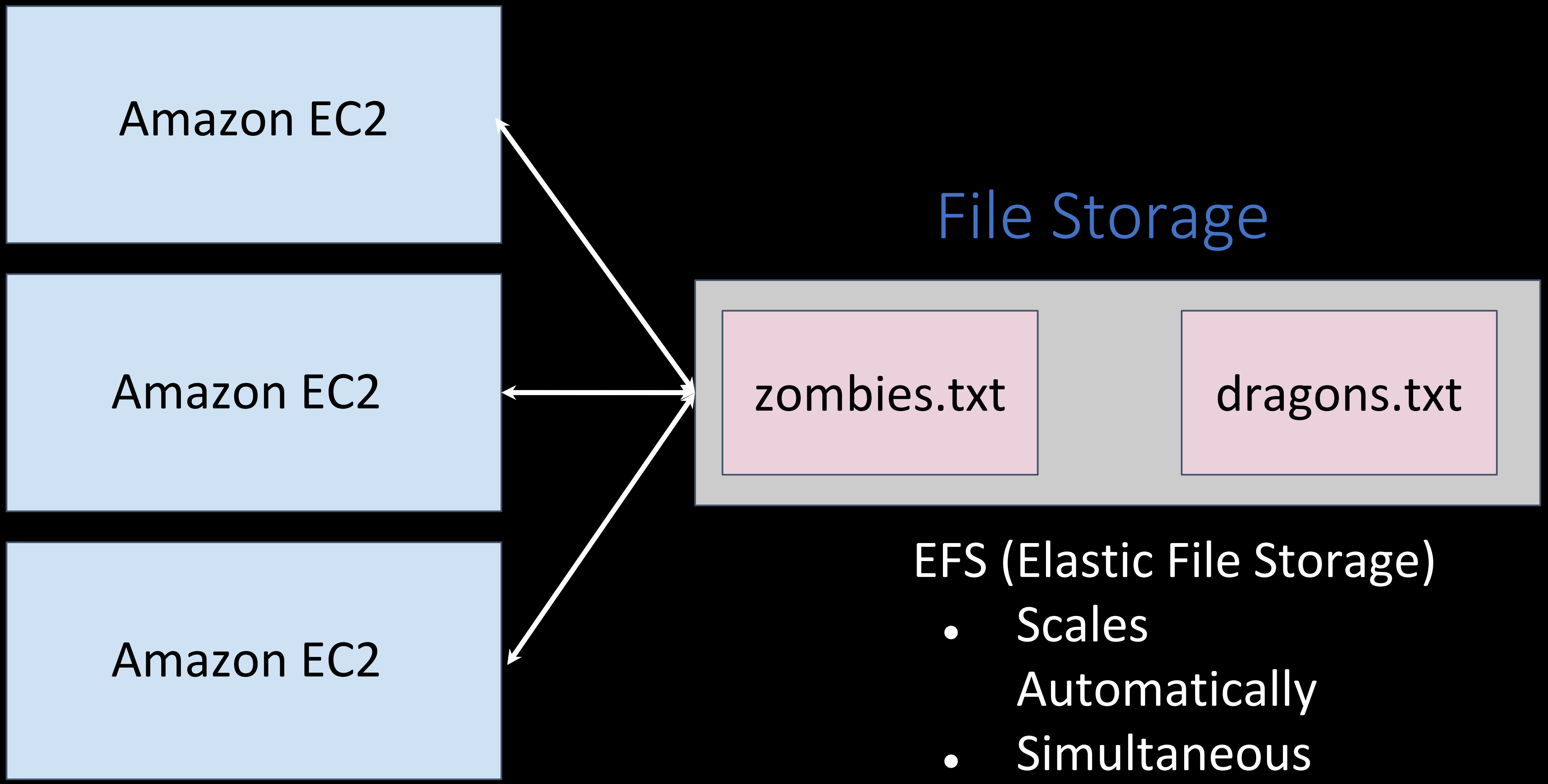
# Zombies Getting Popular




Amazon EC2

EBS (Elastic Block Storage)
(Think of Hard Disk for your Computer with each disk of fixed capacity)

# One File Across Many Servers!




EFS (Elastic File Storage)

- Scales Automatically
- Simultaneous Read/Writes from Multiple Servers
- Pay what you use

# The Good Old Comparison Chart

S3

EBS

HDFS

EFS

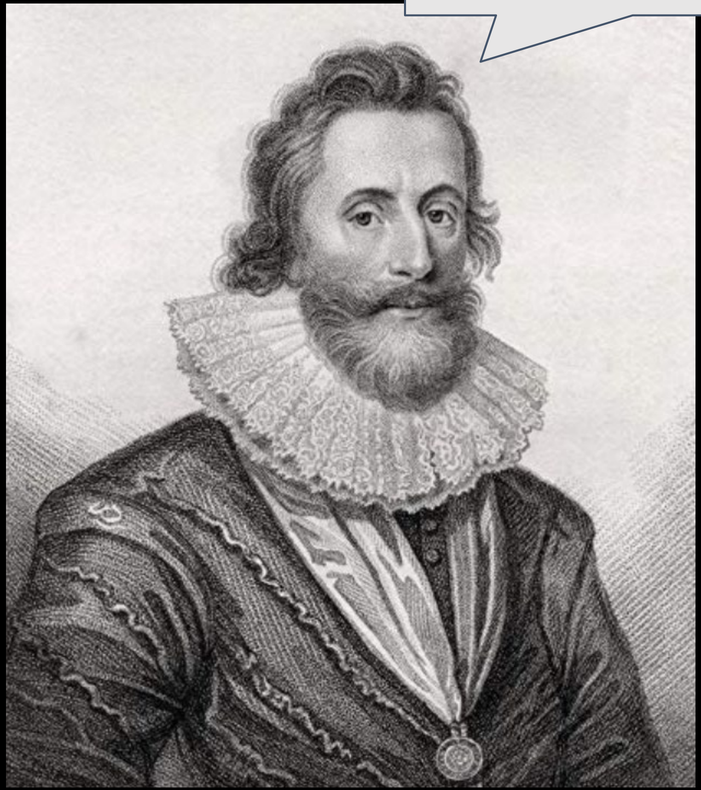
# TAKING IT ALL IN - RIGHT TOOL FOR RIGHT JOB!

Do not use a cannon to kill a mosquito.

~ Confucius

# DIVE DEEP

S3 so Slow, use
EBS everytime



Pen And paper
Architect

Good sir, can we discuss:

1. Specific use case
2. Response time requirement
3. Cost factors
4. Etc.

Agent of Change

# Sample Use Case - Bill Storage and Retrieval

WAITSFIELD TELECOM CHAMPLAIN VALLEY

Waitsfield and Champlain Valley Telecom

JOHN SMITH
PO BOX 9
WAITSFIELD, VT 05673-0009

1

**Account Summary**

**Primary Line:** 496-XXXX
Account Number: 000000000
Billing Period: 04/01/16 - 04/30/16
**Total Amount Due: $126.82**

Current Charges Due By: 04/28/16

***Monthly Internet Data Usage (GB)**
Upload: .5
Download: 38.9
**Total:** 39.4

**Thanks for keeping your business local!**

2

| Previous Balance | Payments | Adjustments | Balance Forward | Current Charges | Total Amount Due |
|---|---|---|---|---|---|
| $124.06 | $124.06- | $0.00 | $0.00 | $126.82 | **$126.82** |

## SUMMARY OF CURRENT CHARGES

3

| | |
|---|---|
| Services | $120.49 |
| Savings & Charges | $1.23- |
| Local Measured Service | $.10 |
| Long Distance | $2.24 |
| Federal Tax | $1.02 |
| GMLDS Connectivity Charge | $.97 |
| Vermont Telecommunications Tax | $2.42 |
| Vermont USF Charge | $.81 |
| **TOTAL CURRENT CHARGES** | **$126.82** |

**Customer Service**
(802) 496-3391 or toll free at
(800) 496-3391
**Pay Online:** www.wcvt.com
Choose "Manage My Account"
**Pay by Phone:** 1-866-863-8070
**Office Hours:**
Monday - Friday 8:00AM-5:00PM
www.wcvt.com (LIVE CHAT available)
csdept@wcvt.com

Important Information

**Mad River Valley Telephone Directory Closes April 15, 2016**

In preparation for our 2016-2017 Mad River Valley Telephone Directory, please take a moment to verify your current listing(s). To make a change in your directory listing, please call our Customer Service Department at 496-3391. For yellow page advertising (business accounts), contact Alan Jones at 496-8328 or ajones@wcvt.com.

Notice of Federal Universal Service and Connectivity Charge Decreases
Please refer to the last page of your bill for information on decreases in the Federal Universal Service and Connectivity Charges effective April 1, 2016.

4

PLEASE DETACH AND RETURN THIS FORM WITH YOUR PAYMENT

| Due Date | Total Amount Due | Amount Enclosed |
|---|---|---|
| 04/28/16 | **$126.82** | |

5

0100518449 000012682 BADOR 9

JOHN SMITH
Primary Line: 496-XXXX Billing Date: 04/01/16

000000000$0000126.82

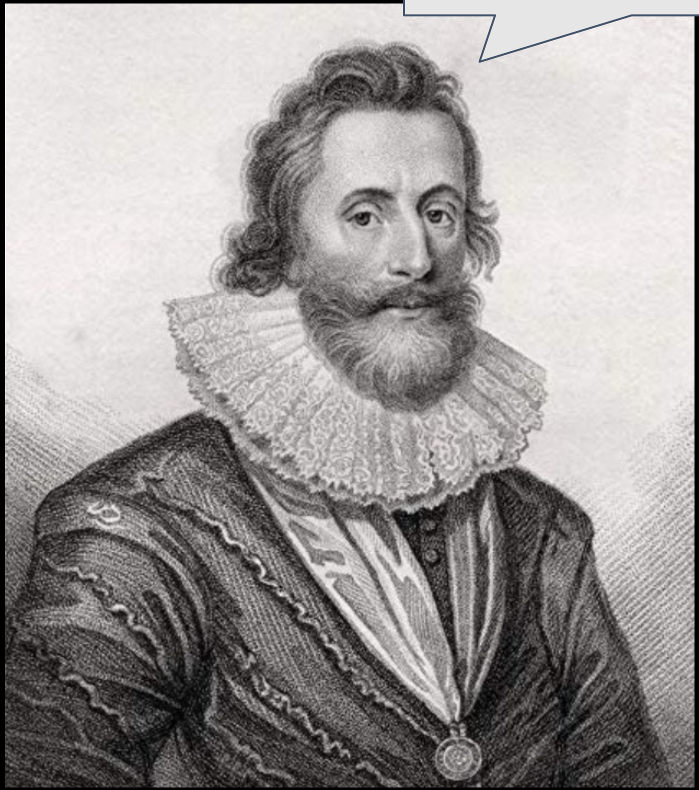
WAITSFIELD AND CHAMPLAIN VALLEY TELECOM
P.O. BOX 412
BRATTLEBORO VT 05302-0412

Check Here for Address Change: ☐ (Please write new information on reverse)

# Sample Use Case - Bill Storage and Retrieval

- Customers generally don’t access bills older than 3 months
- Bill retrieval time in seconds okay, no need for millisecond response
- Need to save bill for 7 years

I have just the perfect idea - save the Bills in Database



Pen And paper Architect

Good sir, all the requirements can be achieved using S3 and it will be much cheaper

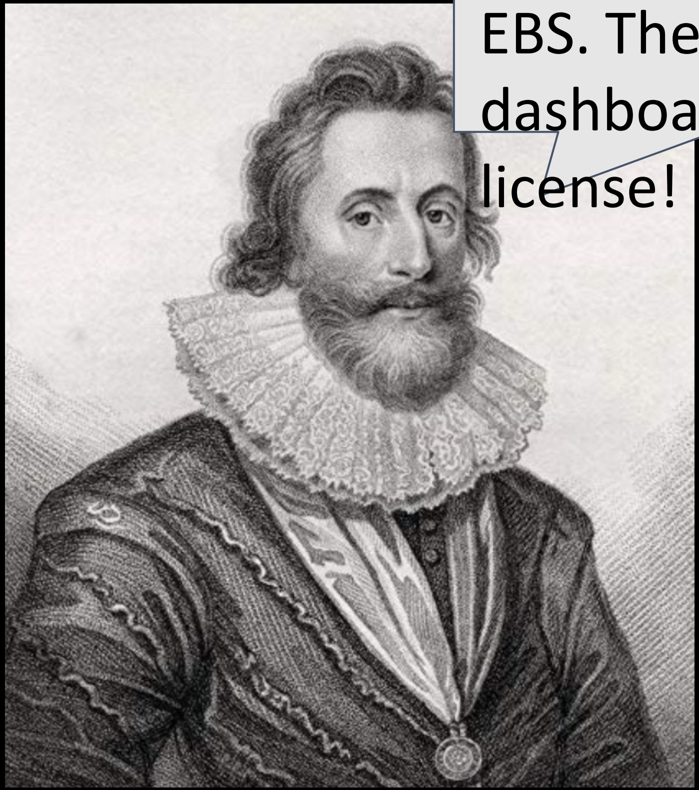
Agent Of

# Sample Use Case - Querying Large Data

- Amount of data is on Petabyte scale
- Data is unstructured, semi-structured and structured
- Query NOT from transactional system
- In future, dashboards are planned

# Sample Use Case - Querying Large Data

only one thing - DATABASE. Let's spin up fastest RDS possible with fastest EBS. Then buy shiny dashboard software license!



Pen And paper Architect

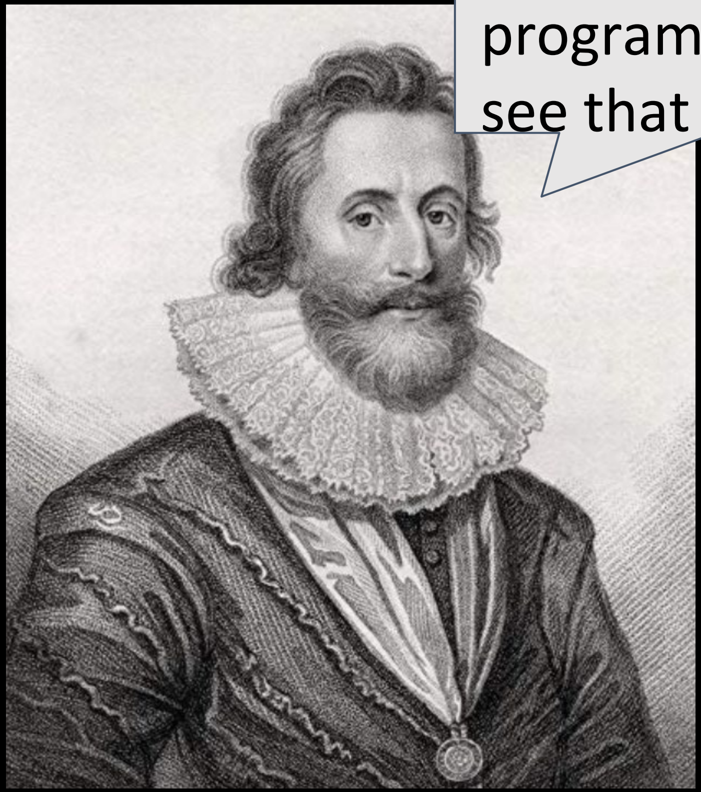
Good sir, let's consider following:

1. Use S3 to store data
2. Use Amazon Athena to query data
3. We can build dashboards in Quicksight

Agent of Change

# Sample Use Case - AI/ML

My best friend is in AI/ML! I suggest loading data into database, write ML programs on EC2 and see that thing run...



Pen And paper Architect

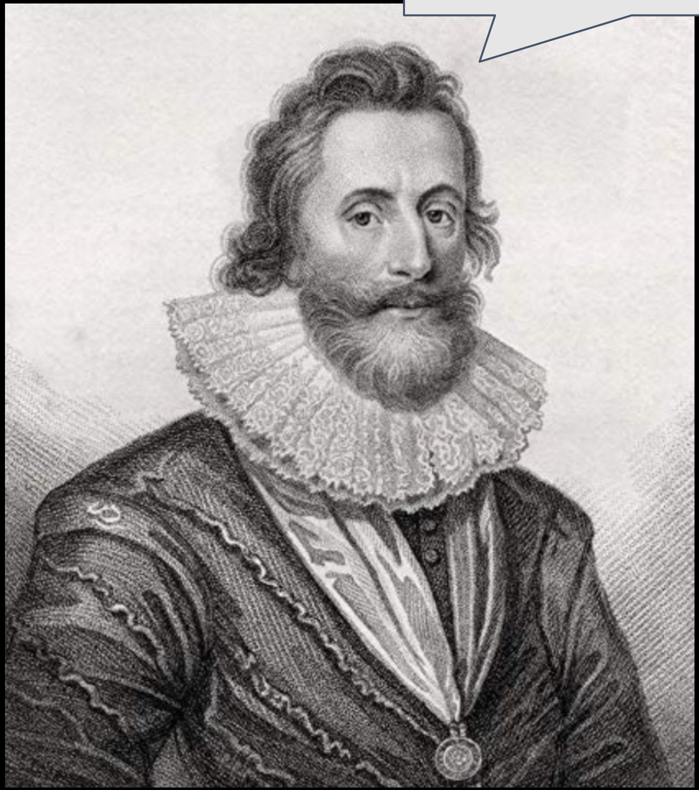
Good sir, let's consider following:

1. Use S3 to store data
2. Use SageMaker to run ML
3. Profit!

Agent of Change

# Sample Use Case - Non Prod WebServer

Finally! Some EBS Action! Give me some Provisioned IOPS SSD EBS!



Pen And paper Architect

Good sir, let's check input output per second requirement and use General purpose SSD (gp2) EBS and save money. Also let's check out Serverless!

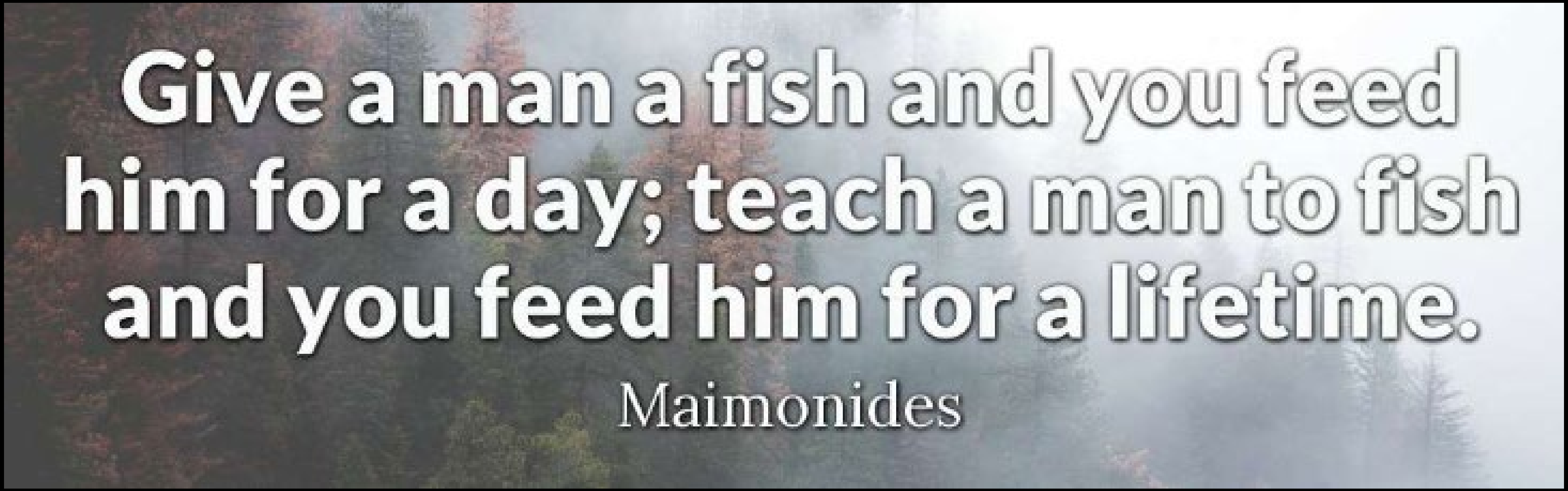
Agent of Change

# When in Interview!

- Focus on Why, not What
- Remember - technology is changing all the time. Demonstrate you keep up to date

# When in Interview!

- Focus on Why, not What
- Remember - technology is changing all the time. Demonstrate you keep up to date



# S3 – Demo

Raj Saha
cloudwithraj.com
Cloud With Raj

# Database Basics – Table, Query

Raj Saha
cloudwithraj.com
Cloud With Raj

# In The Beginning - File

# In The Beginning - File

Customer

Product

Purchase History

# In The Beginning - File

Customer
Product
Purchase History

Challenges

- Hard to correlate
- As the file size grows, harder to find entries
- Update is cumbersome

# Table

Product Purchase History

Customer

| Customer_ID | First_Name | Last_Name | DOB | Zip |
| --- | --- | --- | --- | --- |
| 01 | John | Doe | 07/10/1980 | 10001 |
| 02 | Jane | Smith | 01/20/1983 | 19005 |
| 03 | Ravi | Patel | 05/25/2000 | 51003 |

# Table

Product Purchase History

Customer

| Customer_ID | First_Name | Last_Name | DOB | Zip |
|---|---|---|---|---|
| 01 | John | Doe | 07/10/1980 | 10001 |
| 02 | Jane | Doe | 01/20/1983 | 19005 |
| 03 | Ravi | Patel | 05/25/2000 | 51003 |

- Programmatic access
- Easy to update
- Faster than flat file
- Huge capacity

# Table

Customer

| Customer_ID | First_Name | Last_Name | DOB | Zip |
|---|---|---|---|---|
| 01 | John | Doe | 07/10/1980 | 10001 |
| 02 | Jane | Doe | 01/20/1983 | 19005 |
| 03 | Ravi | Patel | 05/25/2000 | 51003 |

Product

| Product_ID | Product_Name | Description | Price | Quantity |
|---|---|---|---|---|
| P01 | 70” TV | Samsung | $500 | 5 |
| P02 | Toothbrush | Philips | $30 | 20 |
| P03 | Shampoo | Pantene | $10 | 50 |

- Programmatic access
- Easy to update
- Faster than flat file
- Huge capacity
- Easy to relate to other tables

Purchase_History

| Customer_ID | Product_ID | Purchase_Date | Quantity |
|---|---|---|---|
| 01 | P01 | 07/20/2022 | 1 |
| 01 | P03 | 07/20/2022 | 2 |
| 03 | P02 | 12/20/2022 | 4 |

# Query (SQL – Structured Query language)

Customer

| Customer_ID | First_Name | Last_Name | DOB | Zip |
|---|---|---|---|---|
| 01 | John | Doe | 07/10/1980 | 10001 |
| 02 | Jane | Doe | 01/20/1983 | 19005 |
| 03 | Ravi | Patel | 05/25/2000 | 51003 |

Product

| Product_ID | Product_Name | Description | Price | Quantity |
|---|---|---|---|---|
| P01 | 70" TV | Samsung | $500 | 5 |
| P02 | Toothbrush | Philips | $30 | 20 |
| P03 | Shampoo | Pantene | $10 | 50 |

Purchase_History

| Customer_ID | Product_ID | Purchase_Date | Quantity |
|---|---|---|---|
| 01 | P01 | 07/20/2022 | 1 |
| 01 | P03 | 07/20/2022 | 2 |
| 03 | P02 | 12/20/2022 | 4 |

Find all the products purchased by Customer_ID 01:

Select * from Purchase_History where Customer_ID=01

| Customer_ID | Product_ID | Purchase_Date | Quantity |
|---|---|---|---|
| 01 | P01 | 07/20/2022 | 1 |
| 01 | P03 | 07/20/2022 | 2 |

Select * from Purchase_History A, Product B where A.Customer_ID=01 and A.Product_ID=B.Product_ID

| Customer_ID | Product_ID | Purchase_Date | Quantity | Product_ID | Product_Name | Description | Price | Quantity |
|---|---|---|---|---|---|---|---|---|
| 01 | P01 | 07/20/2022 | 1 | P01 | 70" TV | Samsung | $500 | 5 |
| 01 | P03 | 07/20/2022 | 2 | P03 | Shampoo | Pantene | $10 | 50 |

# Primary Key

Customer

| Customer_ID | First_Name | Last_Name | DOB | Zip |
|---|---|---|---|---|
| 01 | John | Doe | 07/10/1980 | 10001 |
| 02 | Jane | Doe | 01/20/1983 | 19005 |
| 03 | Ravi | Patel | 05/25/2000 | 51003 |

Product

| Product_ID | Product_Name | Description | Price | Quantity |
|---|---|---|---|---|
| P01 | 70" TV | Samsung | $500 | 5 |
| P02 | Toothbrush | Philips | $30 | 20 |
| P03 | Shampoo | Pantene | $10 | 50 |

Purchase_History

| Customer_ID | Product_ID | Purchase_Date | Quantity |
|---|---|---|---|
| 01 | P01 | 07/20/2022 | 1 |
| 01 | P03 | 07/20/2022 | 2 |
| 03 | P02 | 12/20/2022 | 4 |

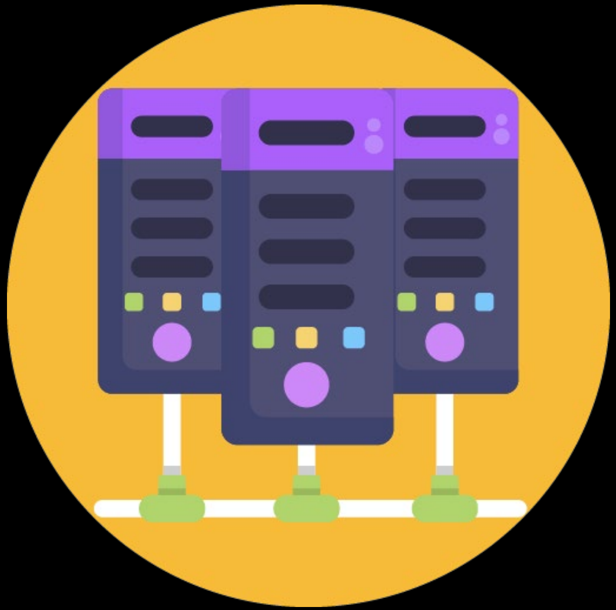
# Database

- Collection of data (Tables)
- Makes data management easy
- Logical separation of systems
- Example – Oracle, SQL Server, Amazon Aurora

# On-premises Database Disadvantages

Raj Saha
cloudwithraj.com
Cloud With Raj

# On-Premises Databases



- Fixed capacity
- High licensing cost
- Expensive DR solutions
- High management overhead

SQL | Amazon Aurora

VS

NoSQL | Amazon DynamoDB

# What We Going To Learn...

- SQL Vs NoSQL
- AWS Database Options
- Amazon Aurora Vs DynamoDB
- Conclusion

# SQL Vs NoSQL Database

| SQL Database (RDBMS) | NoSQL Database |
|---|---|
| Tables have predefined schema | Schemaless |
| Holds structured data | Holds structured and unstructured data |
| Good fit for joins and complex queries | Generally, not good fit for complex multi table queries |
| Emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability) | Follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance ) |
| Generally, scales vertically | Generally, scales horizontally. AWS DynamoDB scales automatically! |

# Schema Vs Schemaless

## Schema

| Artist | Song Title | Album Title | Price | Genre | Critic Rating |
|---|---|---|---|---|---|
| | | | | | |

## Schemaless

```
{
    "Artist": "No One You Know",
    "SongTitle": "My Dog Spot",
    "AlbumTitle": "Hey Now",
    "Price": 1.98,
    "Genre": "Country",
    "CriticRating": 8.4
}
```

```
{
    "Artist": "No One You Know",
    "SongTitle": "Somewhere Down The Road",
    "AlbumTitle": "Somewhat Famous",
    "Genre": "Country",
    "CriticRating": 8.4,
    "Year": 1984
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Still in Love",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 2.47,
    "Genre": "Rock",
    "PromotionInfo": {
        "RadioStationsPlaying": [
            "KHCR",
            "KQBX",
            "WTNR",
            "WJJH"
        ],
        "TourDates": {
            "Seattle": "20150625",
            "Cleveland": "20150630"
        },
        "Rotation": "Heavy"
    }
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Look Out, World",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 0.99,
    "Genre": "Rock"
}
```

# SQL Vs NoSQL in AWS

## SQL DATABASES

Amazon Aurora

Amazon RDS

## NOSQL DATABASES

Amazon DynamoDB

Amazon DocumentDB (with MongoDB compatibility)

Amazon Managed Apache Cassandra Service

Note - You can always run your favorite non-AWS database on EC2

# Amazon DynamoDB

## Amazon Aurora

MySQL and PostgreSQL compatible relational database built for the cloud. 5 times faster than standard MySQL, 3 times faster than standard PostgreSQL at 1/10th the cost

Multi-Master Supported for MySQL

Cross region Active-Passive replication Supported for MySQL (Global Database)

Choosing Multi-AZ & Read Replicas provide High Availability

Vertical scaling. Serverless Aurora scales automatically, not as scalable as Dynamo.

Has integrated caches, can’t be adjusted

Enable backups, snapshots for DR

## Amazon DynamoDB

Key-value and document database with single-digit millisecond performance AT ANY SCALE

Multi-Master

Cross region Active-Active replication Supported (Global Tables)

Inherently replicates across three AZs - HA and Durable

Inherently Scalable, can handle more than 10 trillion requests/day & peaks of more than 20 million requests/second

Provides adjustable in-memory caching via DAX

Inherently durable, Point In Time Backups can be enabled

# Customer References

Amazon Aurora

Amazon DynamoDB

# TAKING IT ALL IN - RIGHT TOOL FOR RIGHT JOB!

Do not use a cannon to kill a mosquito.

~ Confucius

SQL | Amazon Aurora

VS

NoSQL | Amazon DynamoDB

SQL | Amazon Aurora

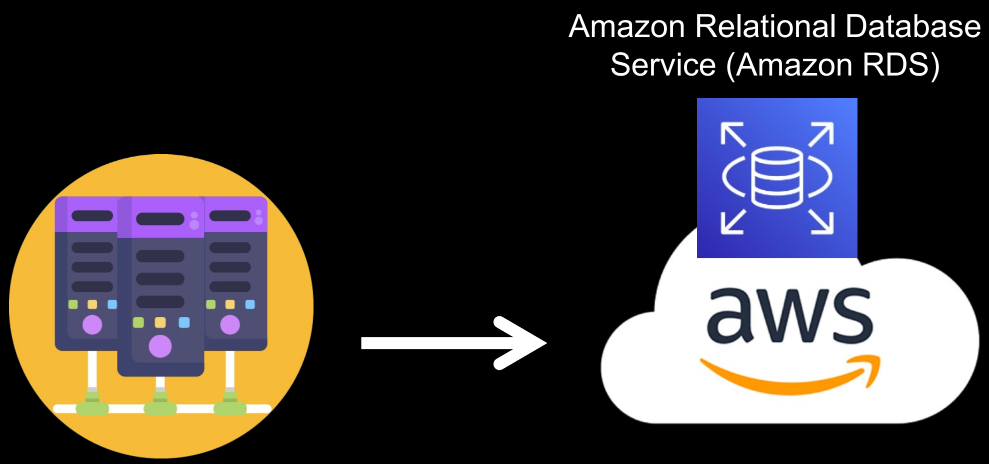
&

NoSQL | Amazon DynamoDB

# AWS Databases – SQL and NoSQL

Raj Saha
cloudwithraj.com
Cloud With Raj

# On-Premises Databases to AWS




- Modernization takes time
    - Train team
    - Change codes
    - Migrate data
- Remaining licensing term
- People, Process, Technology
- Can choose familiar databases on AWS

# AWS Databases

SQL

Amazon Relational Database Service (Amazon RDS)

Amazon Aurora, MySQL, MariaDB, PostgreSQL, Oracle, MS SQL Server

NoSQL

Amazon DynamoDB

# SQL

Amazon Relational Database Service (Amazon RDS)

- Lower administrative burden
  - Automatic patching
  - Best practice recommendation
- Highly performant
  - Choose between General Purpose (SSD) and Provisioned IOPS (SSD) storage
- Scalable
  - Easy vertical scaling
  - Storage scaling with NO downtime
  - Read replica
- Availability and durability
  - Automated backups
  - Database snapshots
  - Multi-AZ deployments
- Security
- Cost effectiveness
  - Use and delete (Pay as you go)
  - Stop and start

# NoSQL

Amazon DynamoDB

- Serverless key value datastore
- Highly performant
  - Milliseconds latency
  - Microseconds latency with DAX (DynamoDB Accelerator)
- Scalable
  - Horizontal scaling
  - Automatic scale up and down
- Availability and durability
  - Data automatically replicated across multiple AZs
  - Automated backups
  - Global table
- Security
- Cost effectiveness
- Tight integration with AWS Lambda
  - Trigger
  - Dynamo Streams

# AWS Observability and Security

# Observability – What and Why

Raj Saha
cloudwithraj.com
Cloud With Raj

# Your Application

# Your Application

# Your Application

# Observability


Observability is the ability to measure a system's current state based on the data it generates, such as logs, metrics, and traces.

# Logs


## Application Log

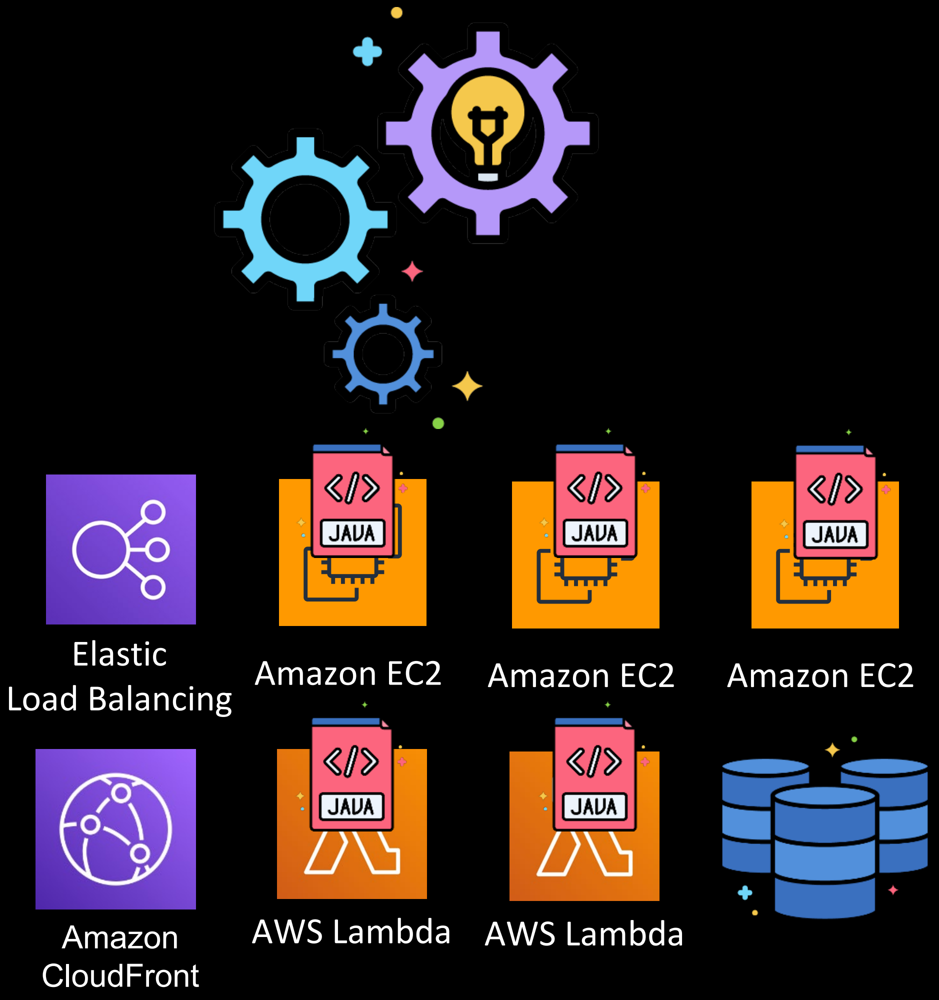
- Errors, informational events, and warnings logged by a software application
- Examples
    - Finance application writes errors when a money transfer is rejected
    - Finance application creates balance low warning messages
    - Streaming application writes total number of viewers for a show

# Logs


Infrastructure Log

- Who or what process did changes to the infrastructure
- Examples
  - Who deleted the S3 bucket
  - Who created the large EC2
  - Who created the Kubernetes cluster

# Metrics




## Metrics

- Real-world example
  - How did you perform in a test
  - How is your restaurant running
- Statistics/data about your application and infrastructure
  - How much resource is being used
  - How many errors in a month

# Metrics


## Metrics

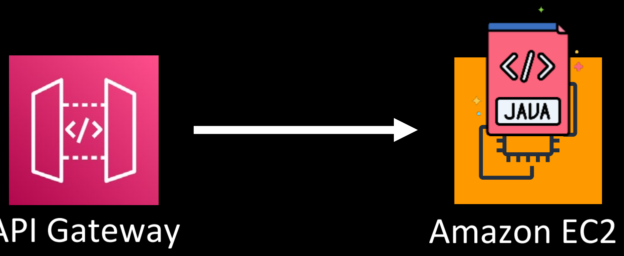
- Real-world example
    - How did you perform in a test
    - How is your restaurant running
- Statistics/data about your application and infrastructure
    - How much resource is being used
    - How many errors in a month
- Synonymous with monitoring

# Traces


Traces

- Complete view of the execution requests as they travel through your code
- Identify bottlenecks and improve performance

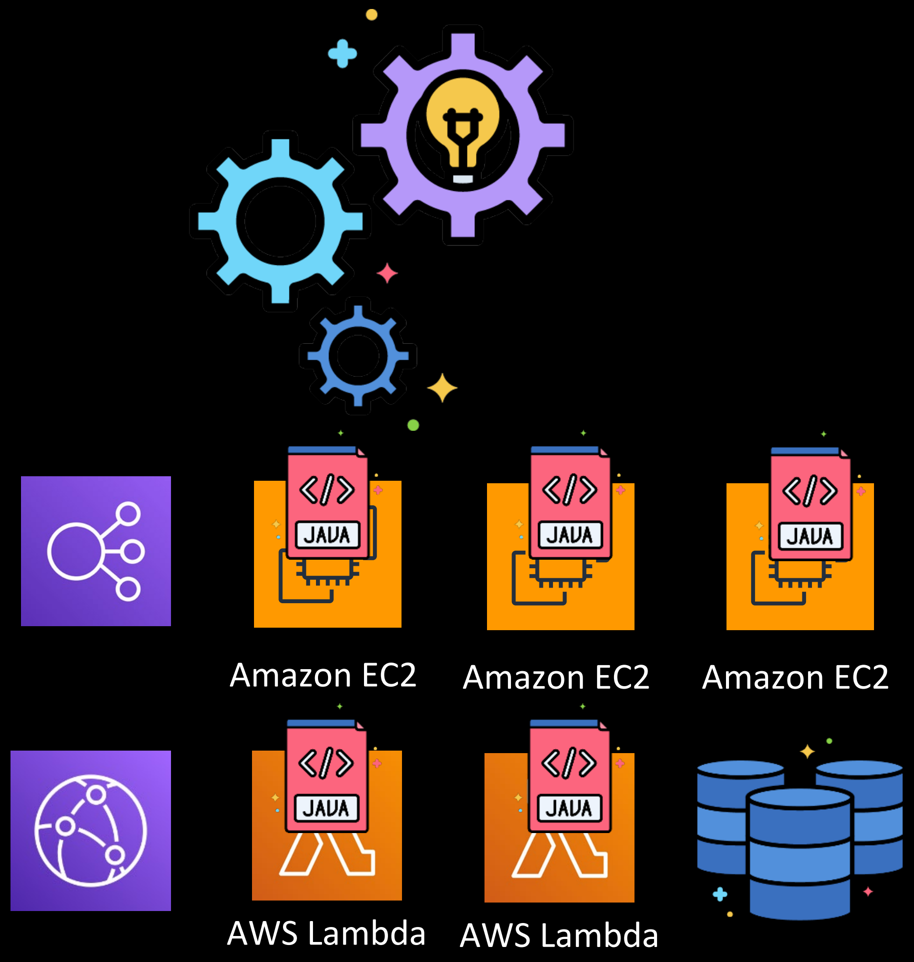
# Traces




## Traces

- Complete view of the execution requests as they travel through your code
- Identify bottlenecks and improve performance
- Available for multiple service interaction

# Logs, Metrics, Traces



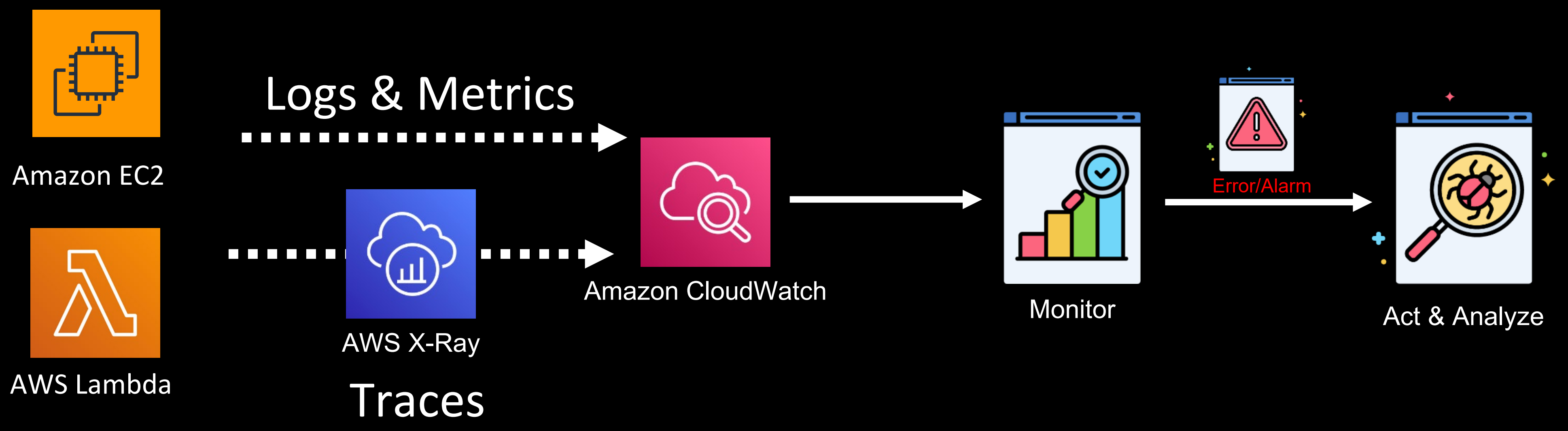

Observability – Logs, Metrics, Traces

- Make your system better performant
- Recover from errors faster
  - Identify root cause
- Critical to implement in real-world applications
  - And for interviews!

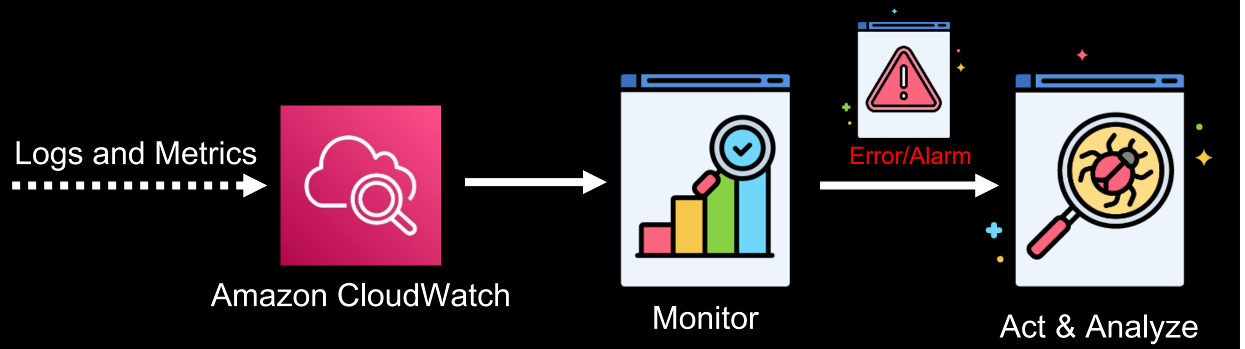
# Amazon CloudWatch – Boring or Powerful?

Raj Saha
cloudwithraj.com
Cloud With Raj

# Amazon CloudWatch



# Amazon CloudWatch




- Observability service that provides data and actionable insights for AWS
  - Logs and metrics for AWS services out of the box
  - Traces done by X-Ray shown in CloudWatch
- Managed by AWS
- Monitor key metrics and logs
  - Visualize
  - Create alarms
  - Query log data
  - Faster mean time to resolution
- Optimize resources proactively
- Identify trends and patterns
  - Anomaly detection
  - Synthetics
- Stream logs to third-party applications (Splunk, ElasticSearch etc.), or S3 bucket

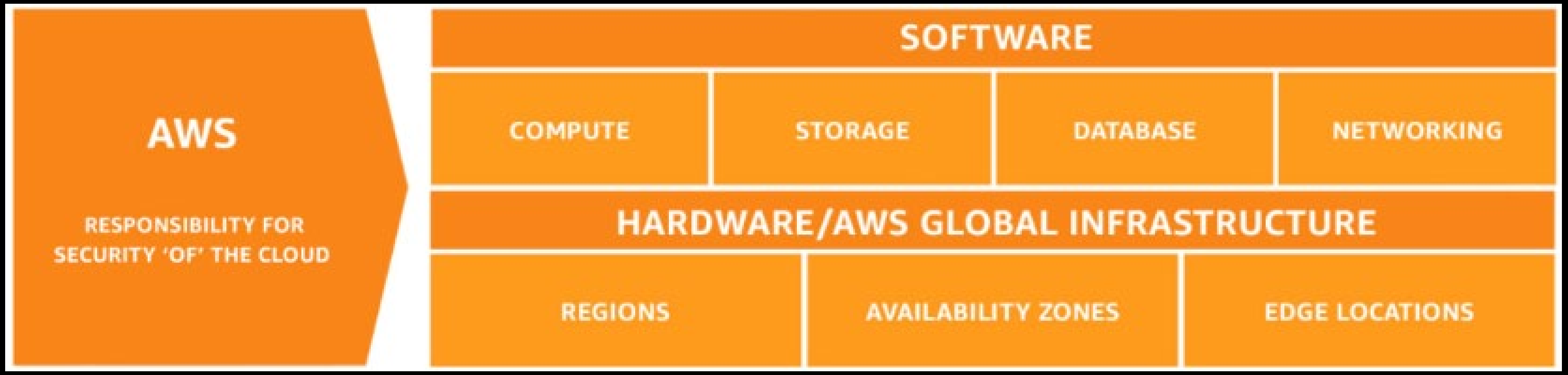
# Demo - Amazon CloudWatch

Raj Saha
cloudwithraj.com
Cloud With Raj

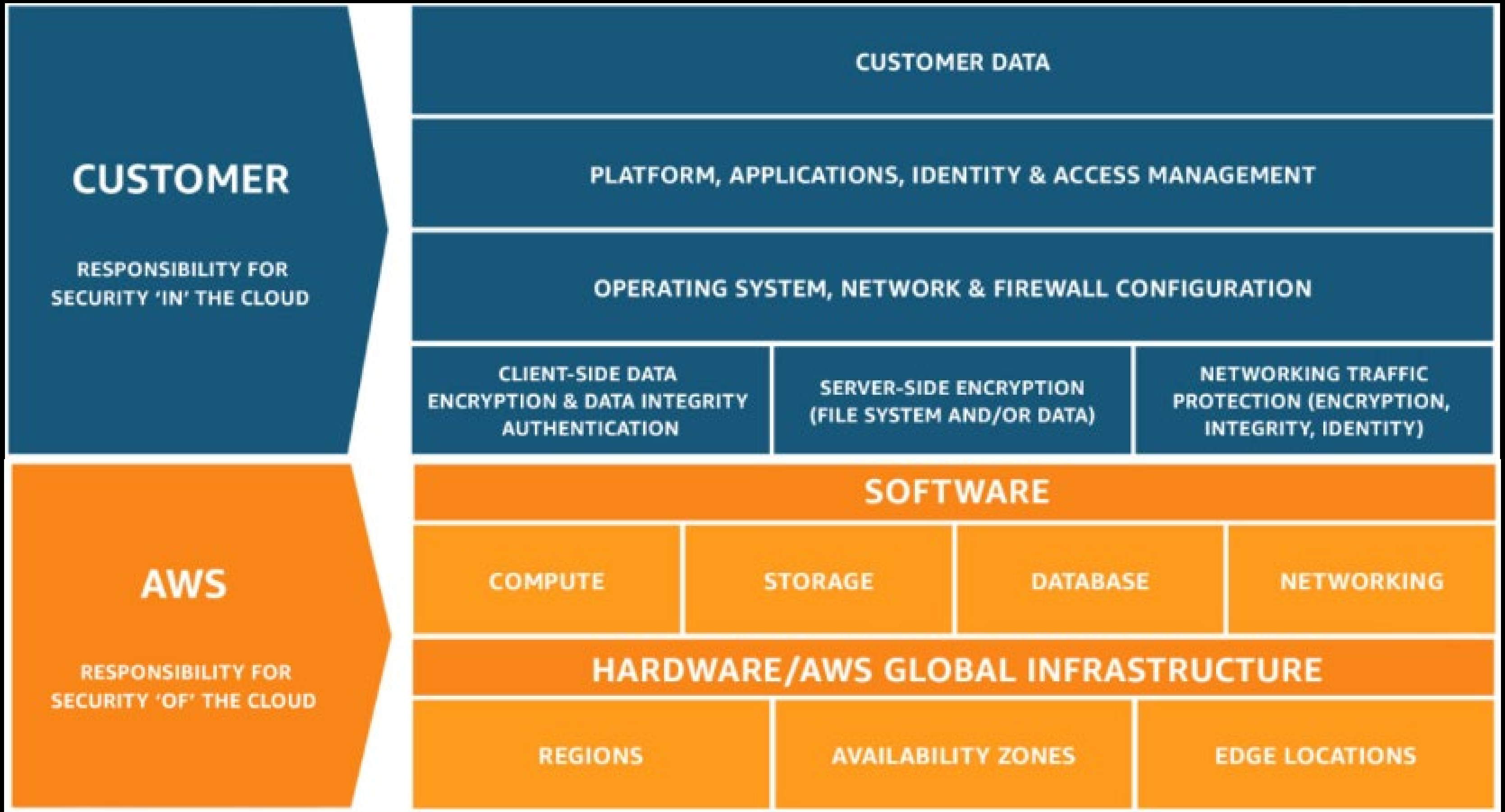
# Shared Responsibility Model

Raj Saha
cloudwithraj.com
Cloud With Raj

# Shared Responsibility



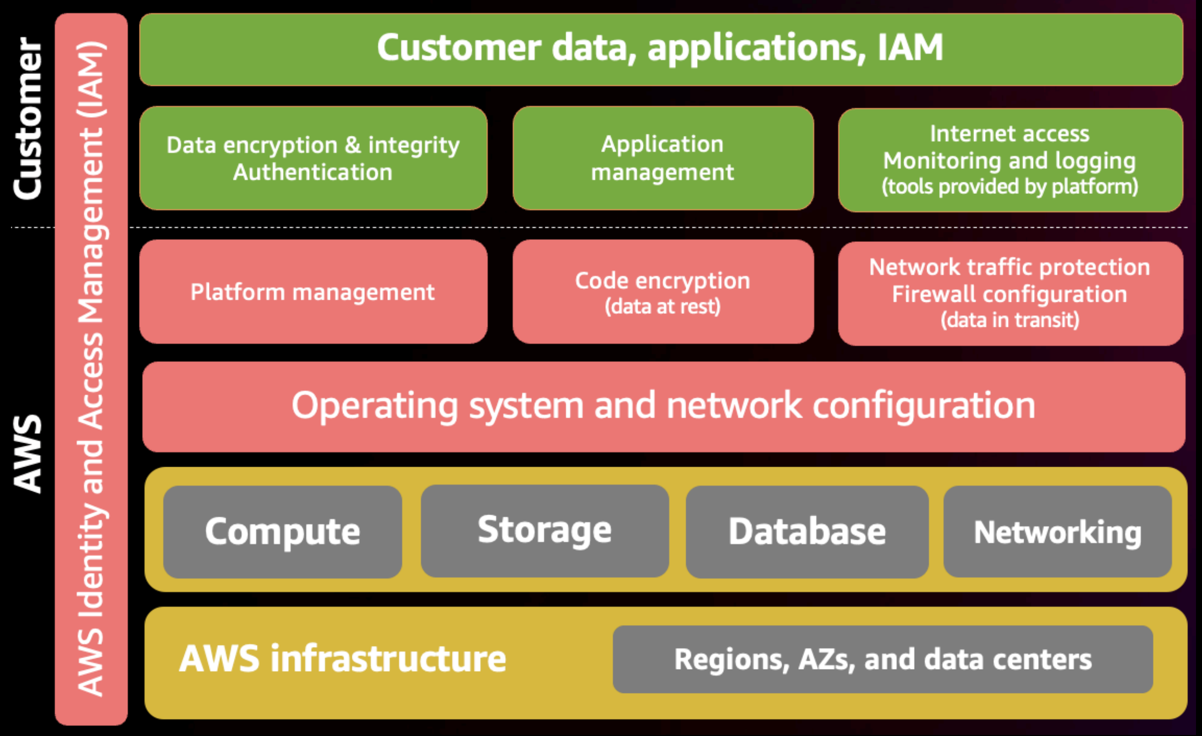

- AWS responsibility "Security of the Cloud" - AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.




- AWS responsibility “Security of the Cloud” - AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.
- Customer responsibility “Security in the Cloud” - Customer responsibility will be determined by the AWS Cloud services that a customer selects. This determines the amount of configuration work the customer must perform as part of their security responsibilities.

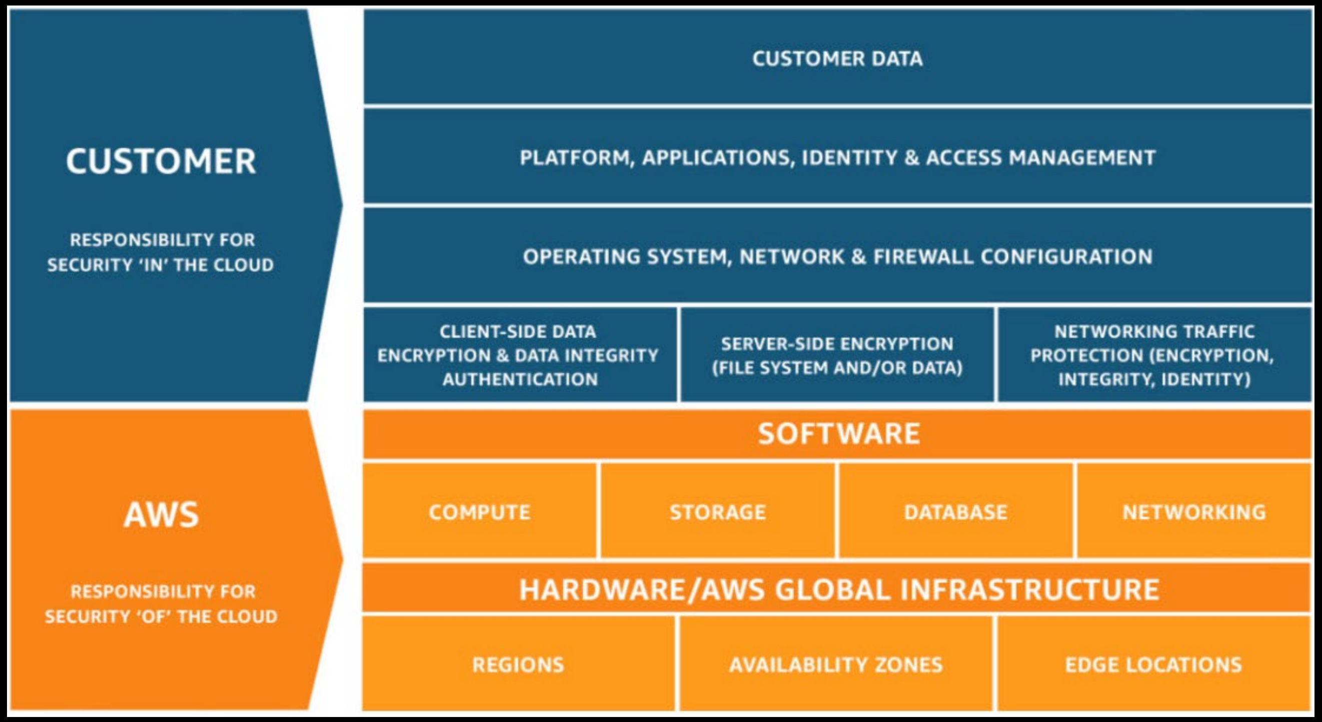
- The responsibility model changes based on AWS service

## Lambda



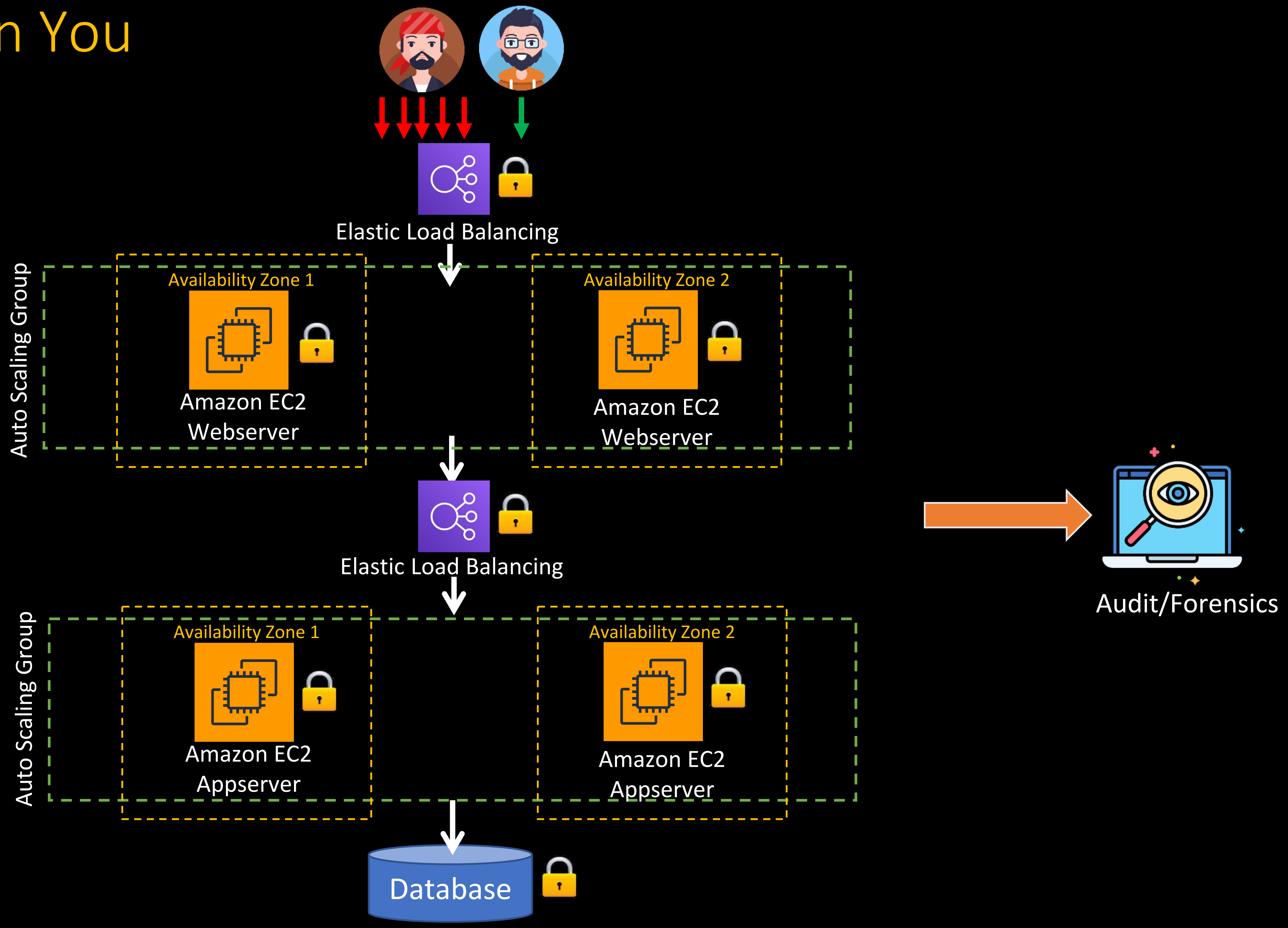
Customer
AWS
AWS Identity and Access Management (IAM)
Customer data, applications, IAM
Data encryption & integrity
Authentication
Application management
Internet access
Monitoring and logging
(tools provided by platform)
Platform management
Code encryption
(data at rest)
Network traffic protection
Firewall configuration
(data in transit)
Operating system and network configuration
Compute
Storage
Database
Networking
AWS infrastructure
Regions, AZs, and data centers


## EC2

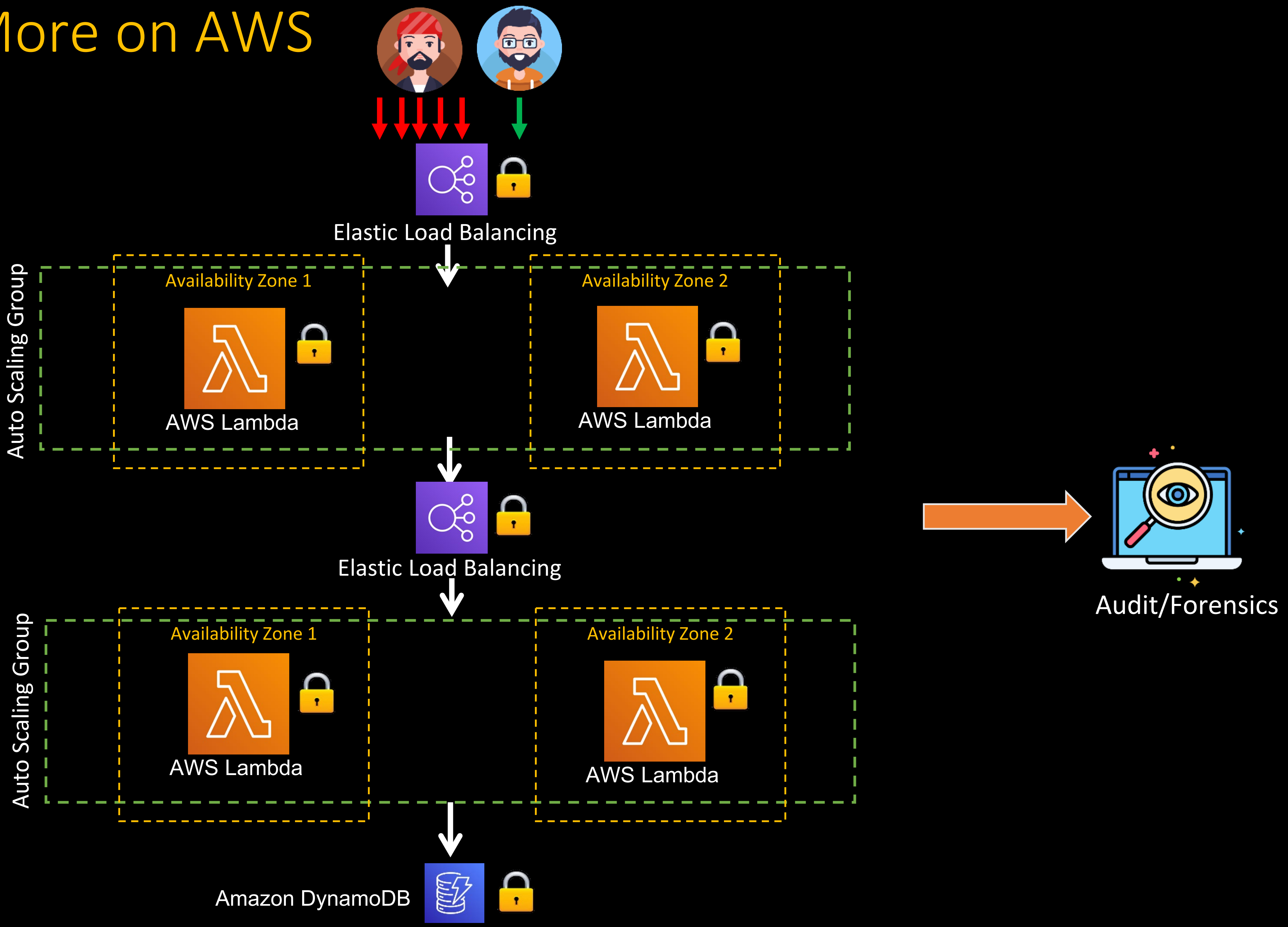


CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

# EC2 – More on You



# Serverless – More on AWS



# IAM – Group, Policy, User, Role

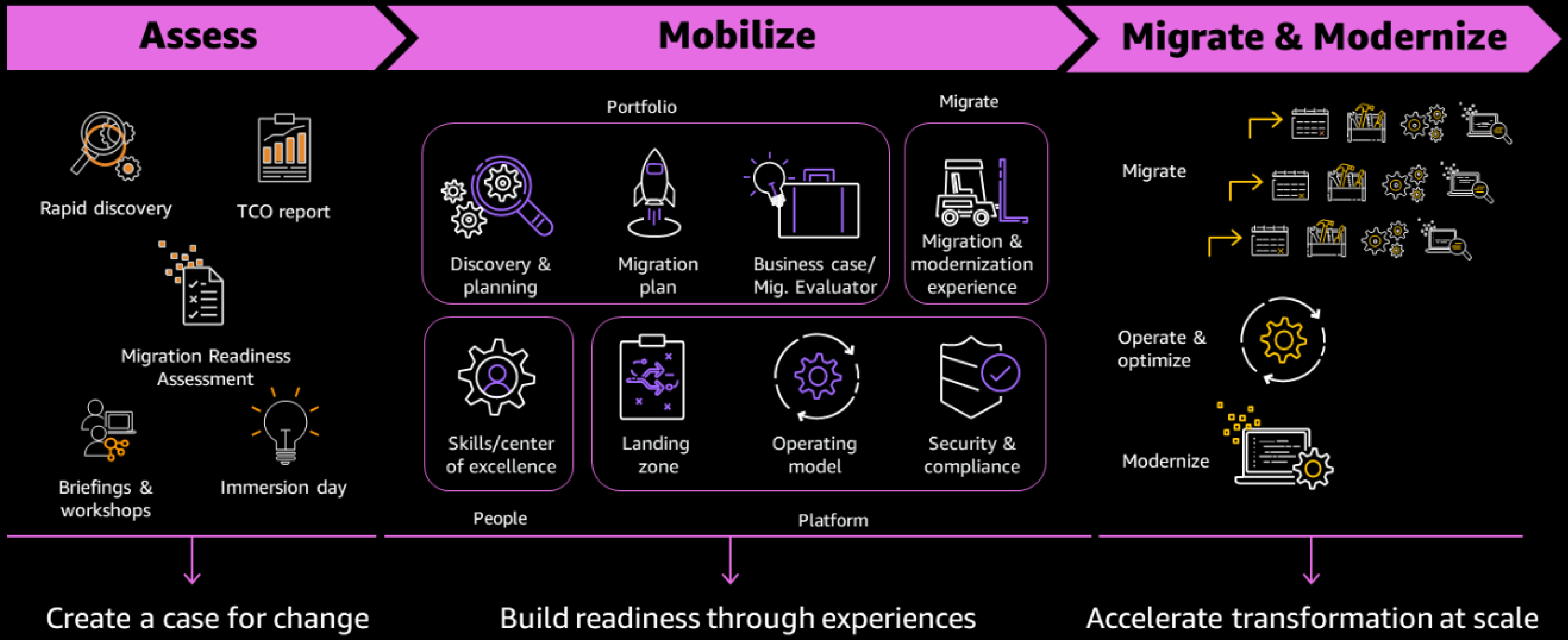
Raj Saha
cloudwithraj.com
Cloud With Raj

# Migrating to AWS

# End to End Flow – Assess, Mobilize, Migrate

Raj Saha
cloudwithraj.com
Cloud With Raj

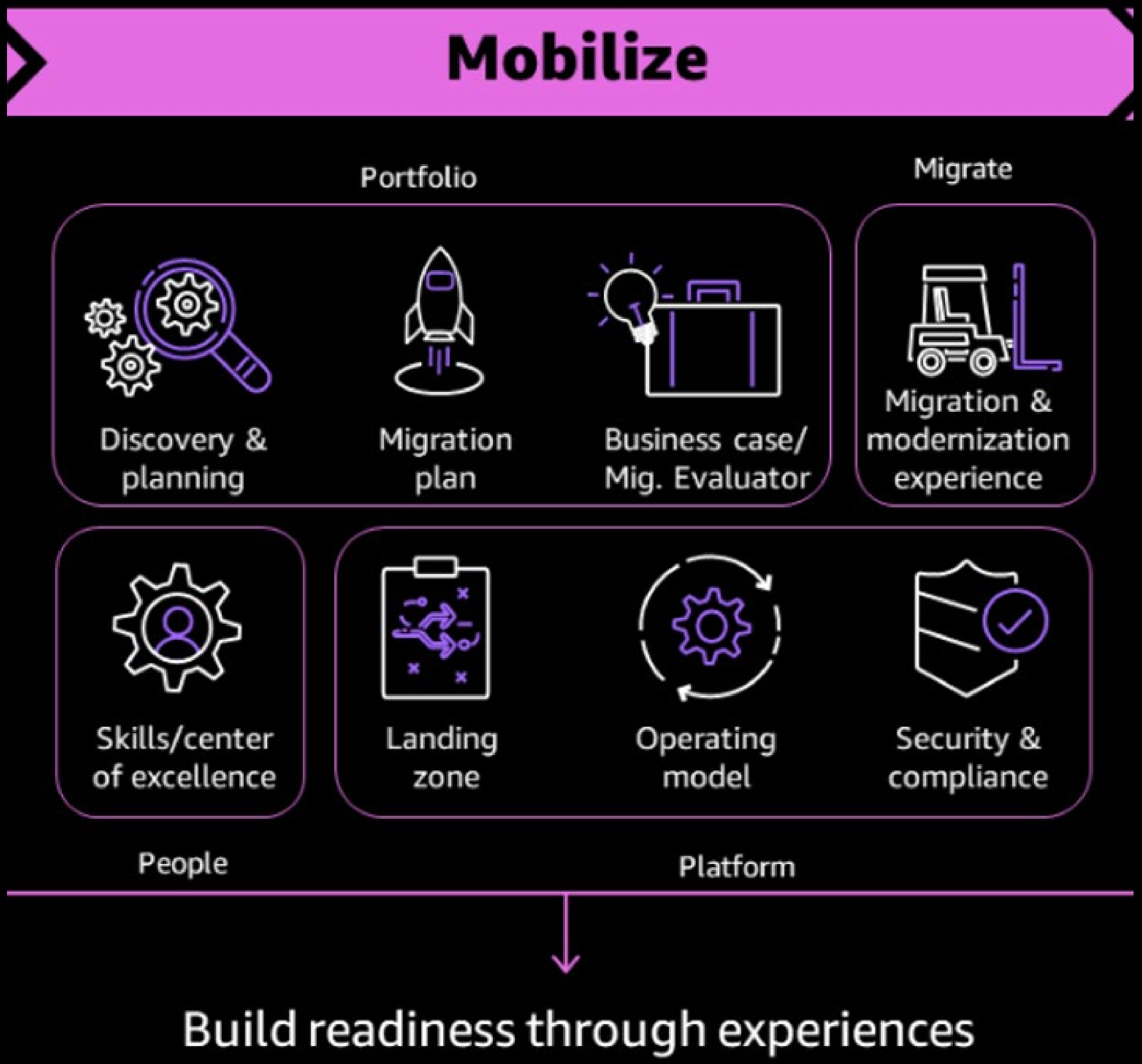
# Overall Migration Flow



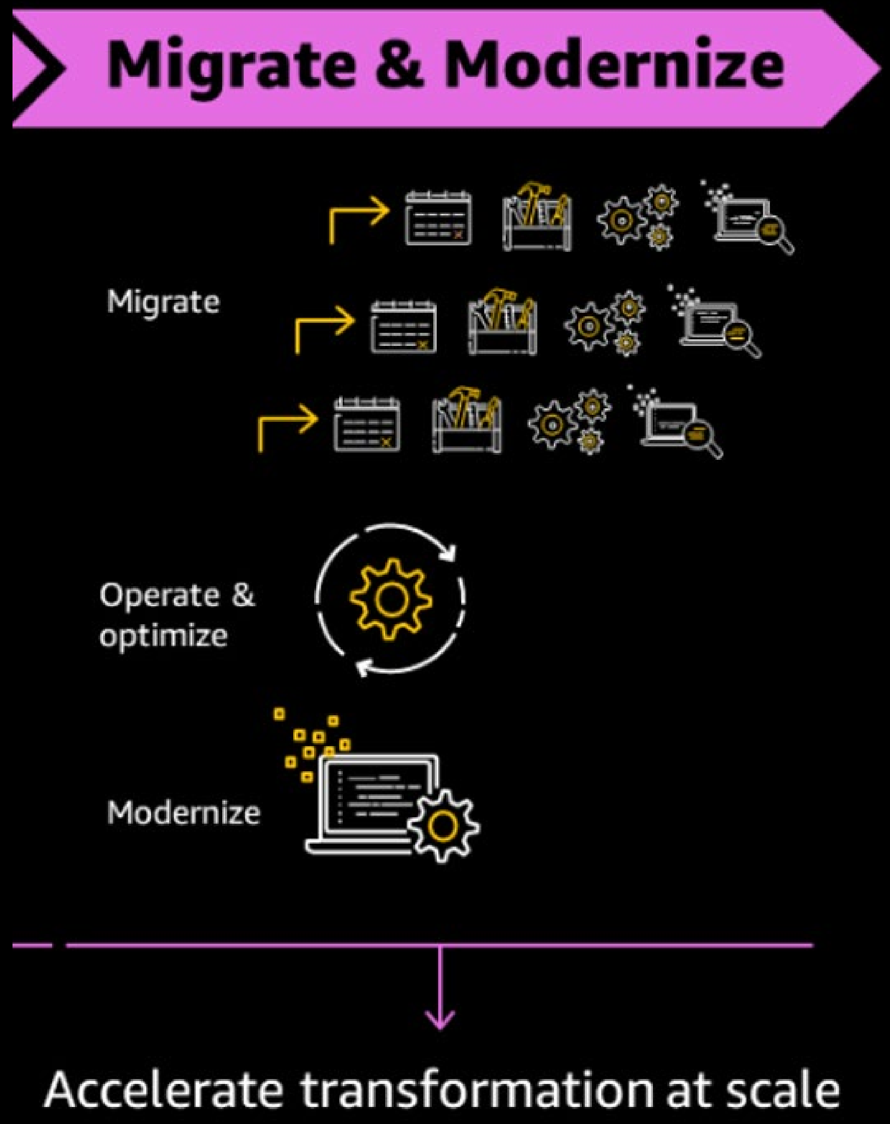

- Begins with a cloud readiness assessment of the current state
  - What you will migrate
  - Business case of migration
  - TCO analysis of migration

- Migration Readiness Assessment (MRA)
  - A set of questions
  - Understands how far along your enterprise is in the cloud journey
  - An action plan to close identified gaps

- Business, people, governance, platform, security, and operations
  - Based on AWS Cloud Adoption Framework (CAF)




- Build foundational capabilities
  - Create CCoE
- Migrate initial applications to AWS
- Few activities include:
  - Detailed portfolio discovery
  - Security, risk, compliance
  - Operations
  - Skills, culture, change, and leadership
  - Implement AWS Landing Zone
- Determine migration strategy from 7 Rs




- Migrate applications at scale
- Utilize best practices and lessons learned from earlier phases
- Use the outputs from CCoE

# Migration Strategies – 7 Rs

Raj Saha
cloudwithraj.com
Cloud With Raj

# 7 Rs

- Relocating
- Rehosting
- Replatforming
- Refactoring/Re-architecting
- Repurchasing
- Retaining
- Retiring

# Relocating

- Known as hypervisor-level lift and shift
- Move application to AWS without any changes
- Example - onprem VMware to VMware on AWS

# Rehosting

- Known as lift and shift
- Move application to AWS without code changes but with using AWS IaaS offerings
- Ideal for quick migrate to cloud before data center lease runs out
- Workforce learns cloud concepts while migrating and use it to modernize the app down the line
- Example – On-premises application running on VM to EC2
- Example – On-premises Oracle to Oracle on EC2

# Replatforming

- Known as lift, tinker, and shift (also lift and reshape)
- Make few optimizations with core architecture unchanged
- Reduce time spent on managing infrastructure
- Example – move on prem Oracle to RDS Oracle
- Example – migrate your app to fully managed platform like Amazon Elastic Beanstalk
- More cost effective than Rehosting

# Refactoring/Re-architecting

- Refactored the app using cloud native services
- Driven by strong business need to scale, and faster time to market
- Example – monolith to microservices using serverless
- Example – containerizing your app
- Example – oracle to Amazon Aurora
- More benefits than Rehost and Replatform, but takes more time

# Repurchasing

- Moving from a traditional license to a software-as-a-service model
- Example – moving a CRM to salesforce, HR system to Workday

# Retaining

- Do nothing for now
- Application will be deprecated, not prioritized, not much business sense to migrate

# Retire

- Remove applications no longer needed
- Allows more attention to appropriate apps

# Level of Effort

Refactor

Replatform

Repurchase

Rehost

Relocate

Retain

Retire

*Level of effort*

# Choosing Migration Strategy

Raj Saha
cloudwithraj.com
Cloud With Raj

# The Complicated Decision Chart

https://docs.aws.amazon.com/prescriptive-guidance/latest/application-portfolio-assessment-guide/prioritization-and-migration-strategy.html#migration-r-type

# Choosing One

- Some factors
  - timeline
  - workforce cloud knowledge
  - complexity
  - interdependence
  - architecture/tech stack
  - licensing terms
  - internal politics (mention as “rate of change” in interview)

- Strongly recommended to start with low complexity ones

- Multiple migration strategies can be chosen

- Irrespective of the strategy, moving to cloud is always beneficial than managing data center

# Real-world tips

- Most interview questions will be on Refactoring
  - Companies are forward looking
- AWS also offers hands on help to migrate - Professional Services
- AWS offers free training –immersion days, workshops
- Utilize AWS partners to migrate
- As a fresher – stay away from Repurchase/Retain/Retire even though it means less work

# Cloud Center of Excellence

Raj Saha
cloudwithraj.com
Cloud With Raj

# CCoE (Cloud Center of Excellence) – What and Why?

- What if all the projects in a company start cloud migration?
  - Each team will choose various AWS service
  - Duplication of effort on PoC, securing AWS services, creating best practices etc.
  - Cost and time inefficient
  - Creates friction amongst different teams on migration approach

- CCoE is an organizational entity that helps drive cloud acceptance and adoption within the organization

- If you are employed and trying to switch to cloud projects
  - Make good relationship and partner with CCoE!

# CCoE (Cloud Center of Excellence) – Responsibilities

- Select a project team and partner with them to migrate them to AWS
  - First project should be quick (few weeks, rather than months)
  - Help them with system design, PoC, AWS liaison, security, automation, reliability etc.
  - Quick wins build momentum

- Build resusable patterns and reference architectures from the win
  - Example – Service Catalog for reusable CloudFormation templates with security best practices built in
  - Example – Create EKS cluster template with standard tools baked in

- Evaluate various tools (AWS/Open source/Third Party) and help standardize

# CCoE (Cloud Center of Excellence) – Responsibilities contd.

- Create cost optimization techniques

- Evangelize, scale the wins and influence other projects
    - Conduct internal webinars, hands on sessions, workshops etc.
    - Highlight migrated project teams
    - Talent transformation is the hardest part of cloud migration

- Gain leadership support
    - Company wide migration is only successful when it's mandated from top down
    - Encourage getting AWS certs with executive sponsorship

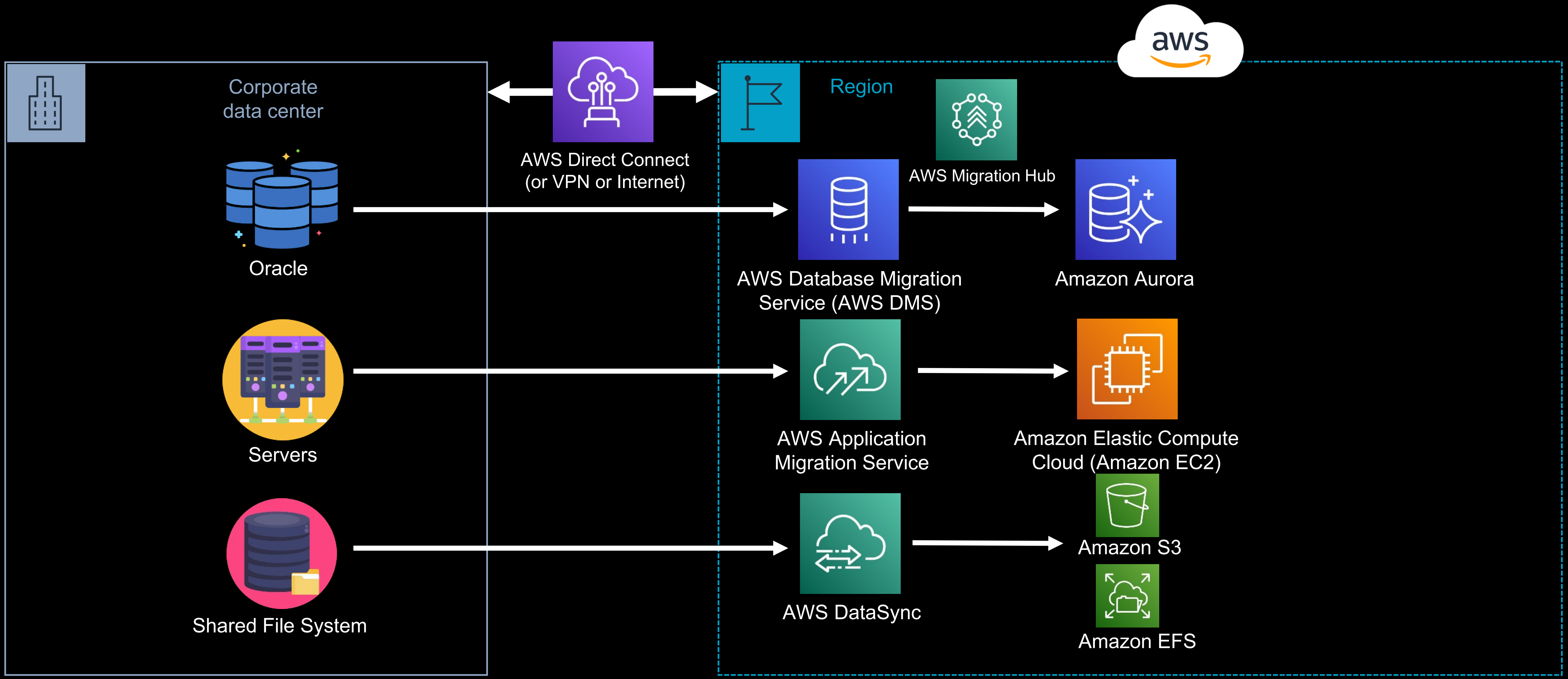
# CCoE (Cloud Center of Excellence) – Forming the Team

- Form a small tiger team
  - No larger than two-pizza team
  - Folks are experiment driven – shows bias for action rather than analysis paralysis
    - Able to learn from failure and iterate quickly
  - Deliver results
  - Customer obsessed (other projects are their customer)
  - Important – Able to evangelize, scale, and influence the learnings

# Notable AWS Migration Tools

Raj Saha
cloudwithraj.com
Cloud With Raj

# Notable AWS Migration Tools



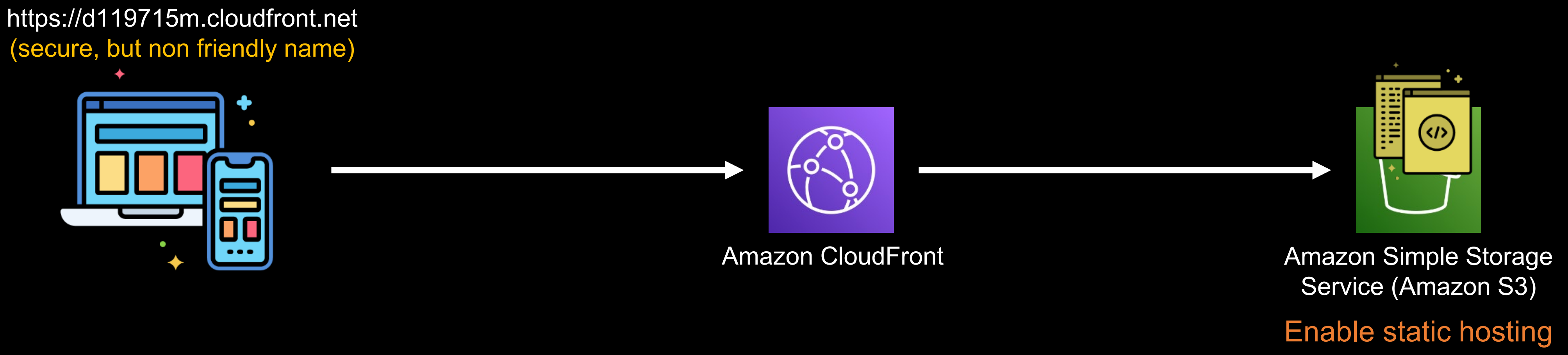
# Hands-on Projects

# Resume from Static Website

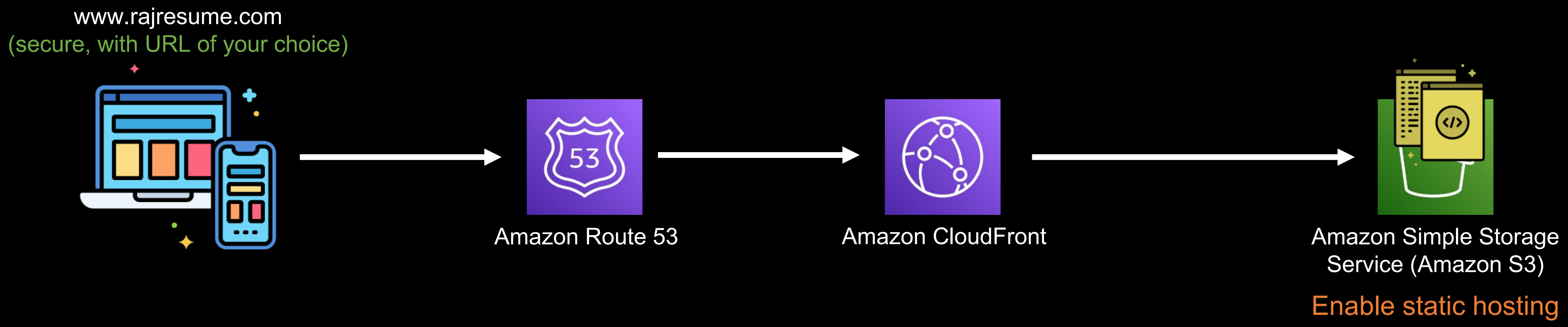
Raj Saha
cloudwithraj.com
Cloud With Raj

# Architecture (free of charge)

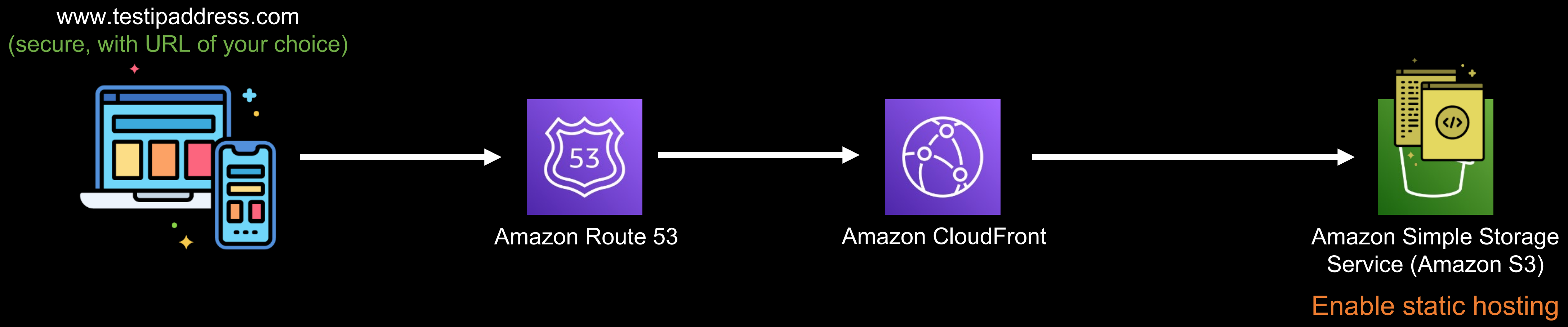
# Architecture (under free tier)



# Architecture (charged)



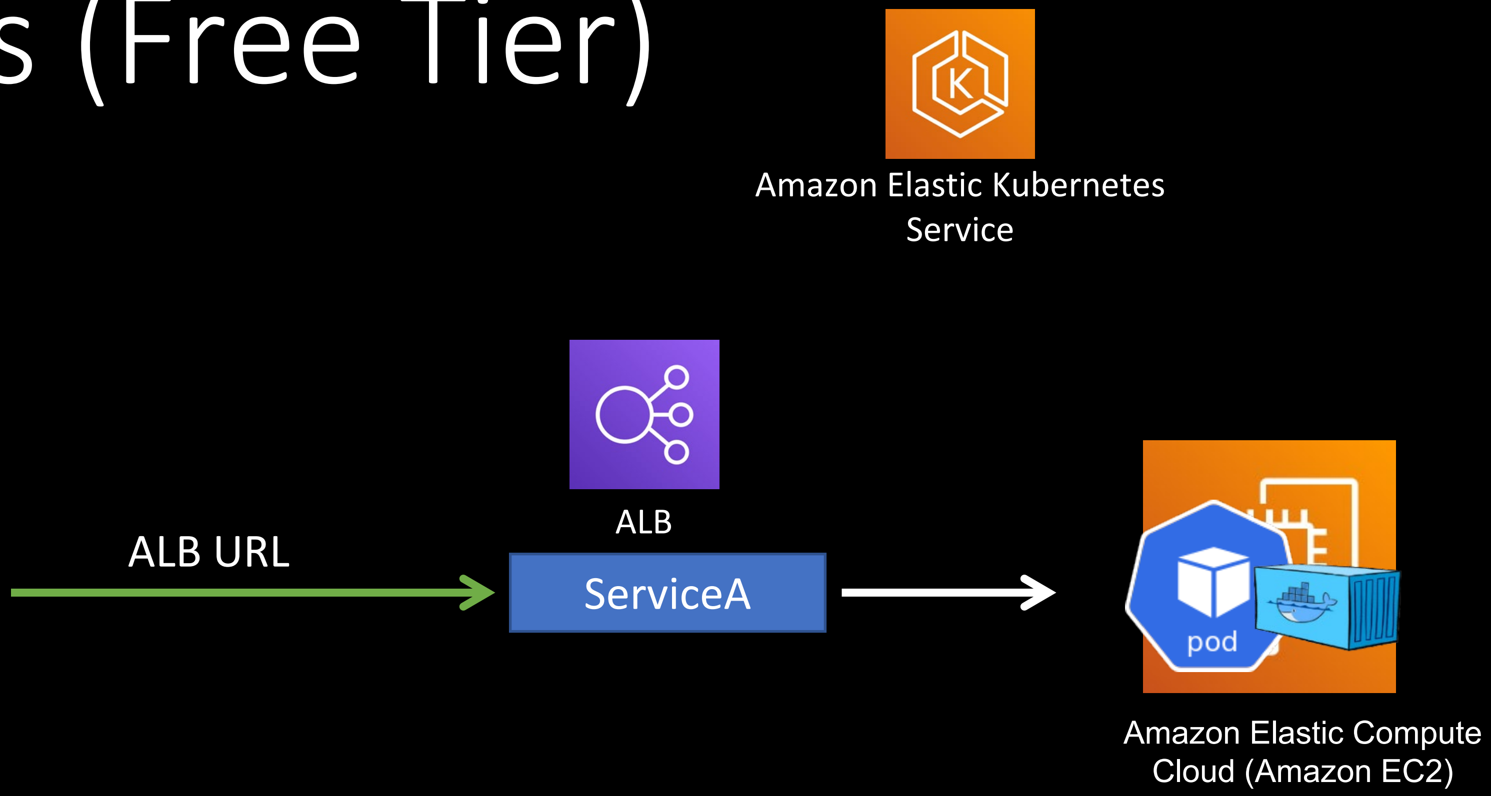
# Architecture (charged)



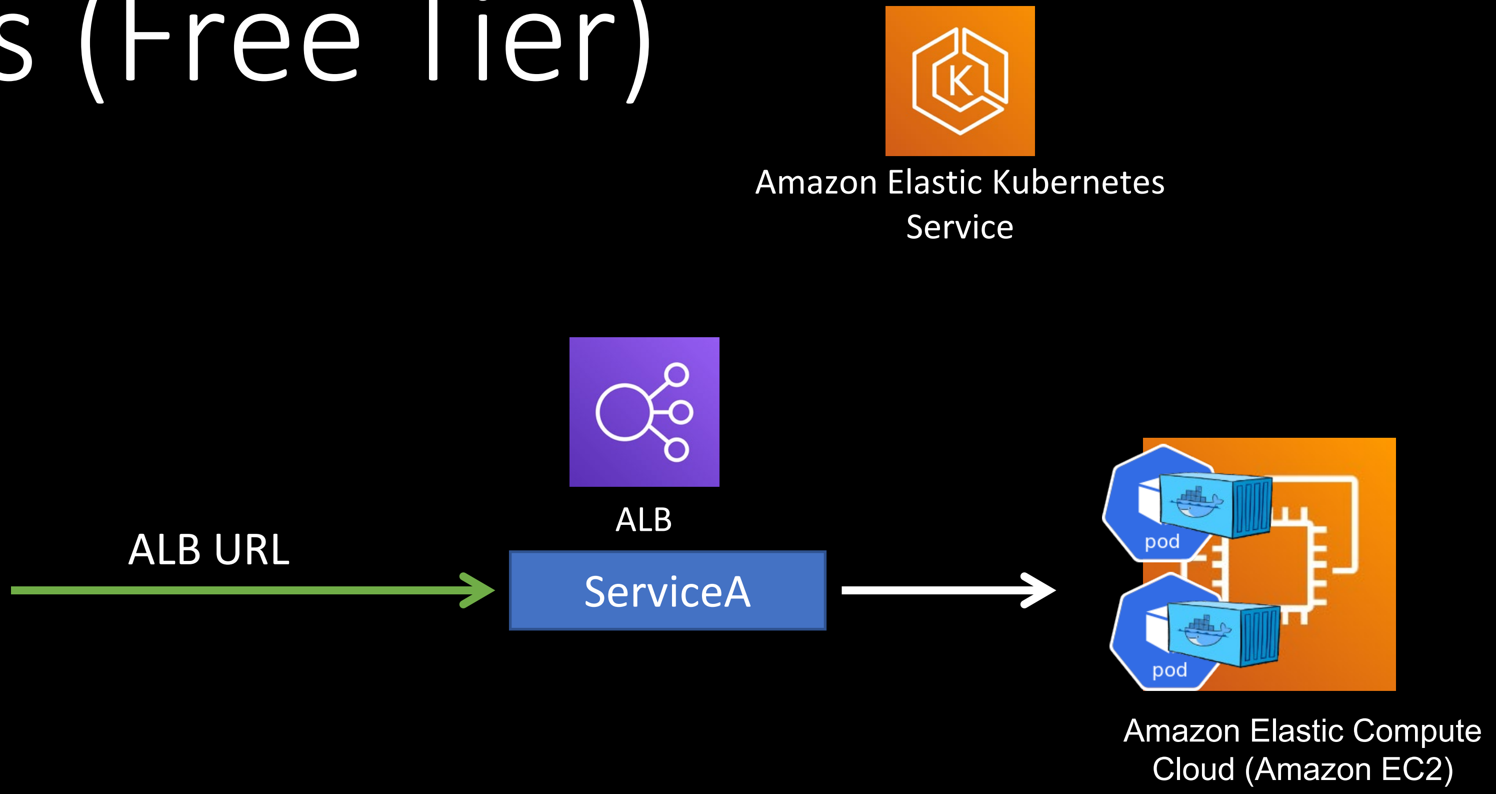
# CV/Resume From Kubernetes

Raj Saha
cloudwithraj.com
Cloud With Raj

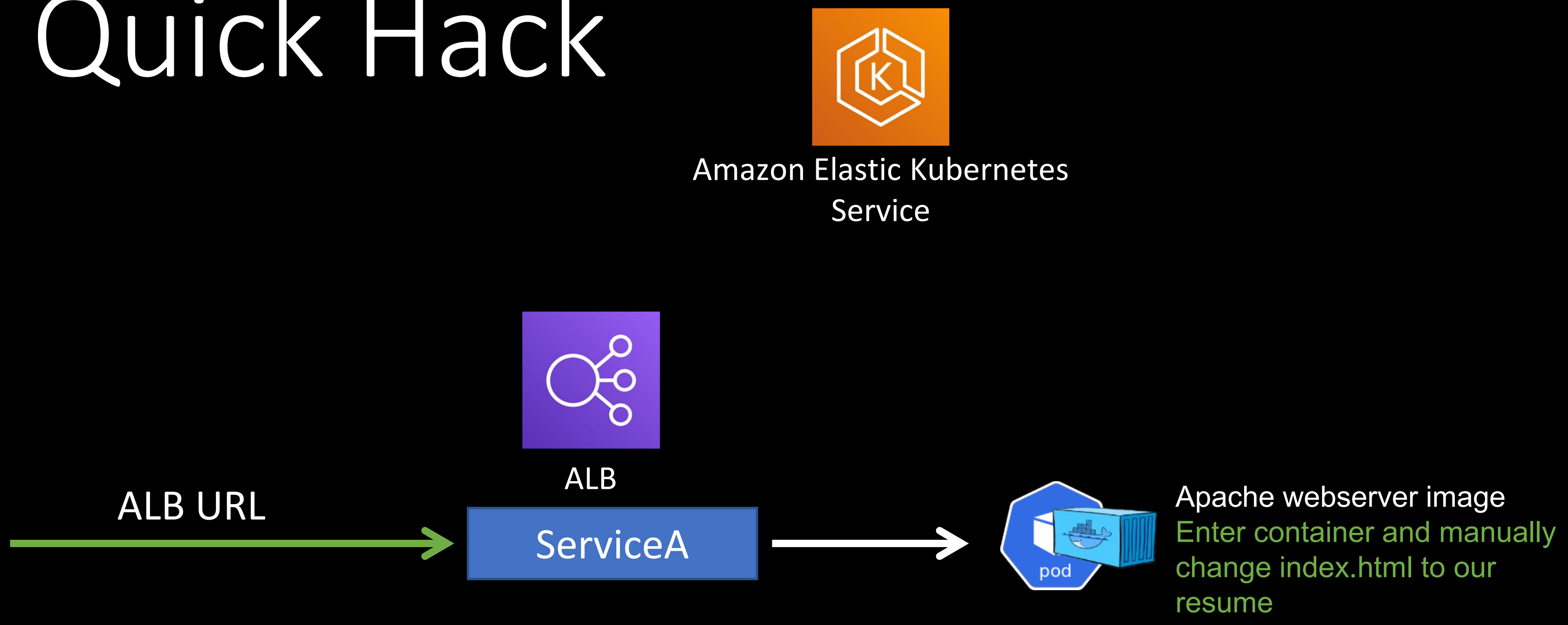
# Kubernetes (Free Tier)



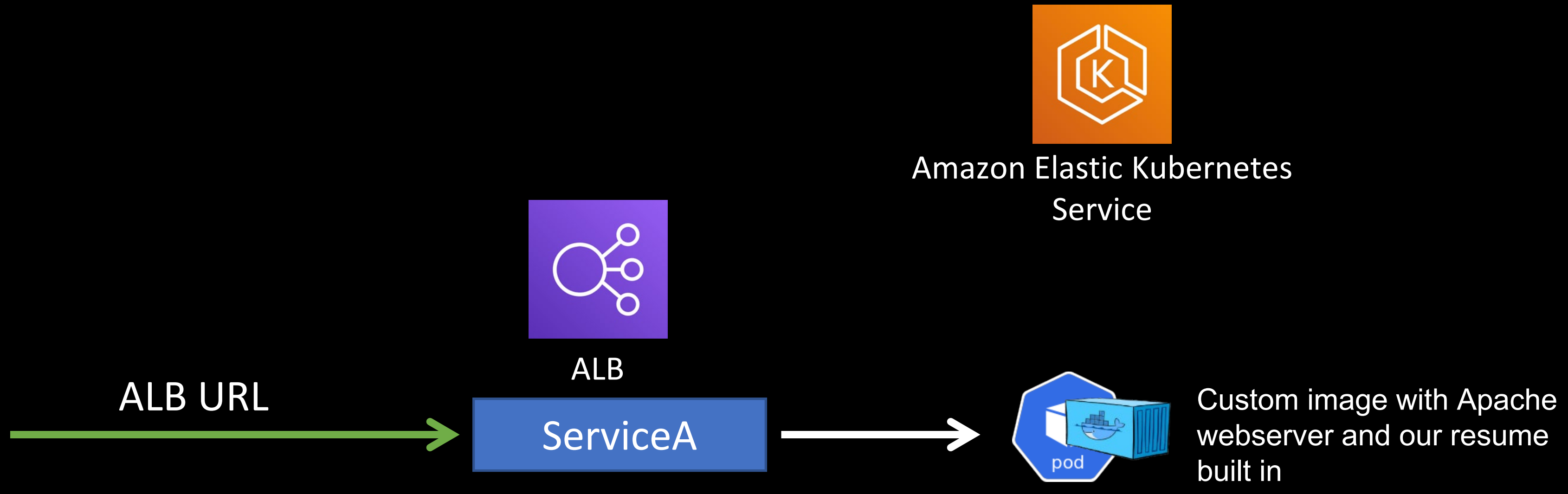
# Kubernetes (Free Tier)



# Solution1 - Quick Hack



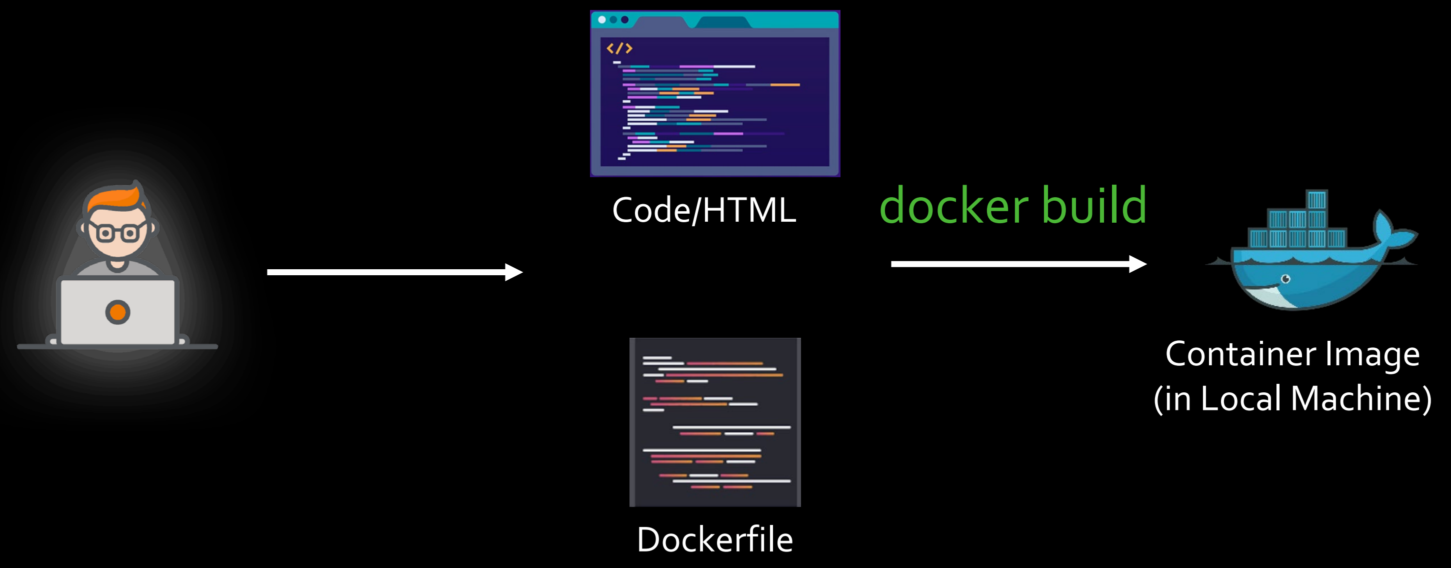
# Solution2 – Real World Way



# Containerizing Your App Workflow

Raj Saha
cloudwithraj.com
Cloud With Raj

# #1 Containerizing the Code

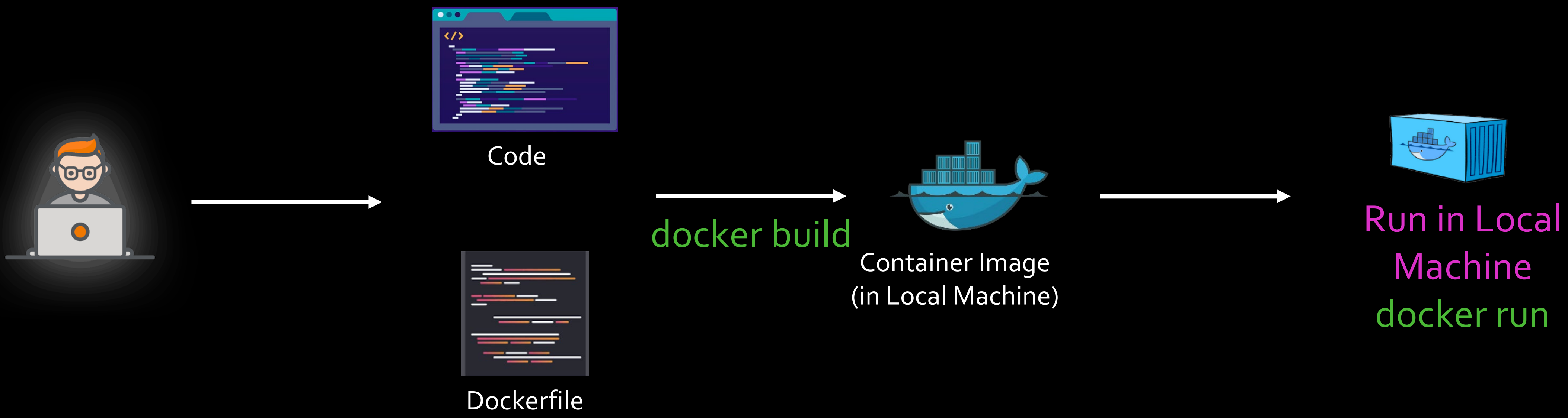


# #2 Dockerfile

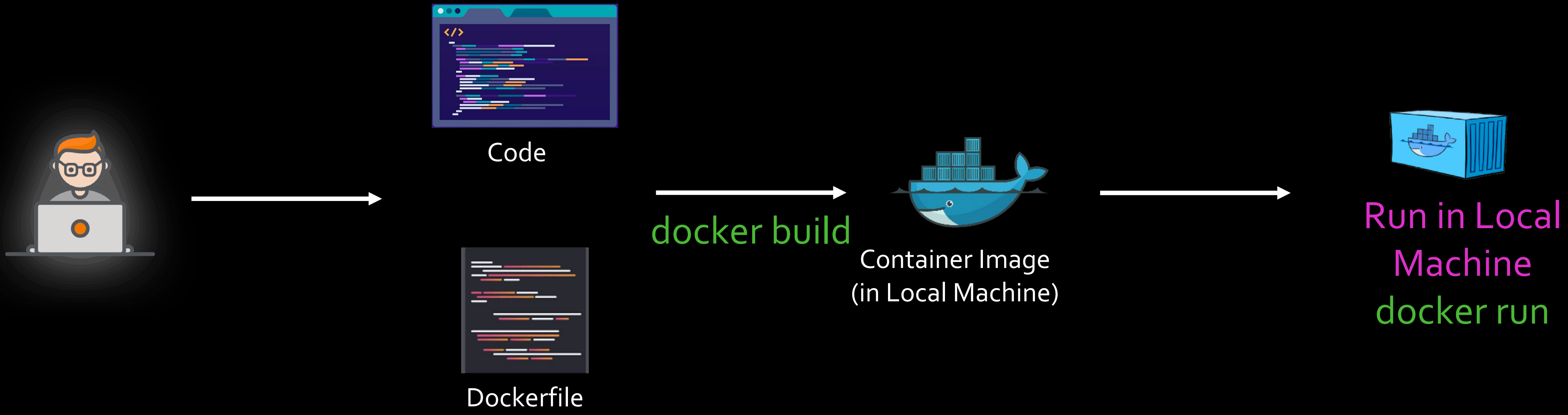
C: > Users > Ashdeep > awscourseforbeginners > resumefromkubernetes > Dockerfile > ...

```
FROM httpd
COPY detailed-resume.html /usr/local/apache2/htdocs/index.html
COPY style.css /usr/local/apache2/htdocs/style.css
COPY headshot.jpg /usr/local/apache2/htdocs/headshot.jpg
```

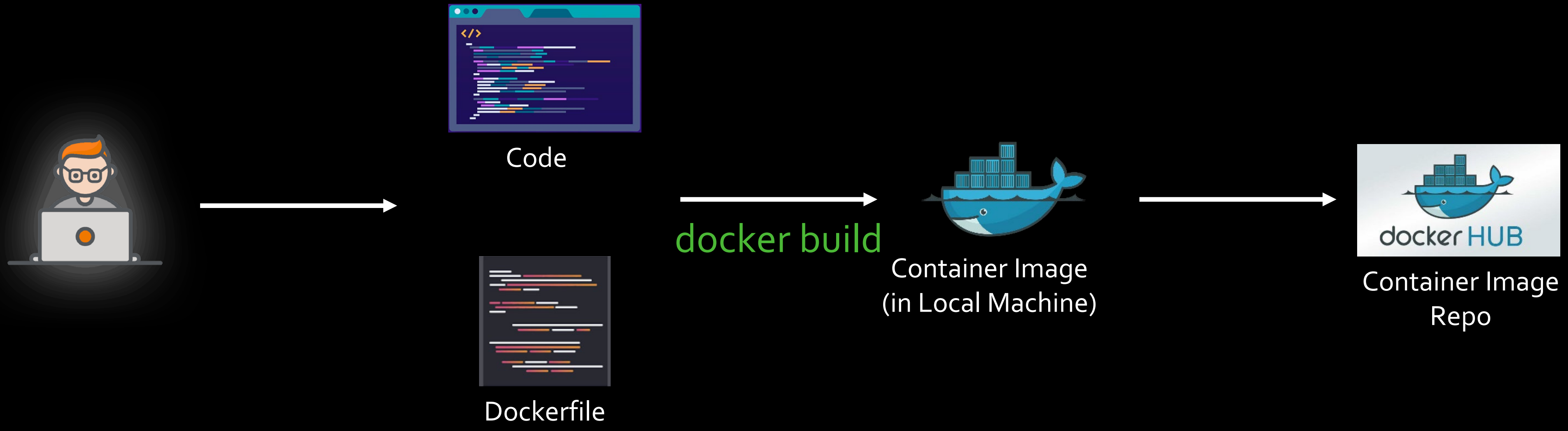
# #3 Run Container in Local Machine



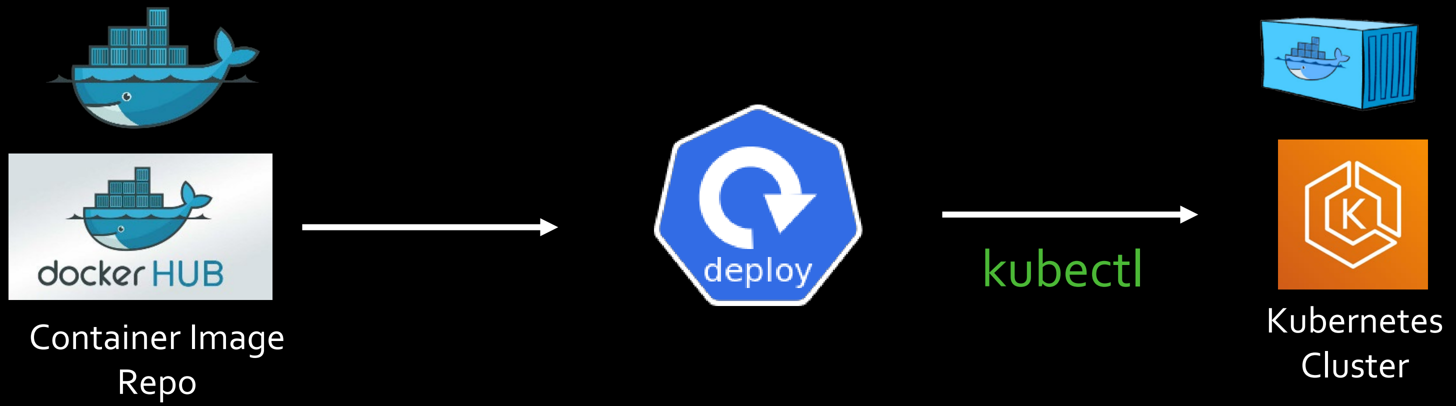
# #4 Prepare for Cloud



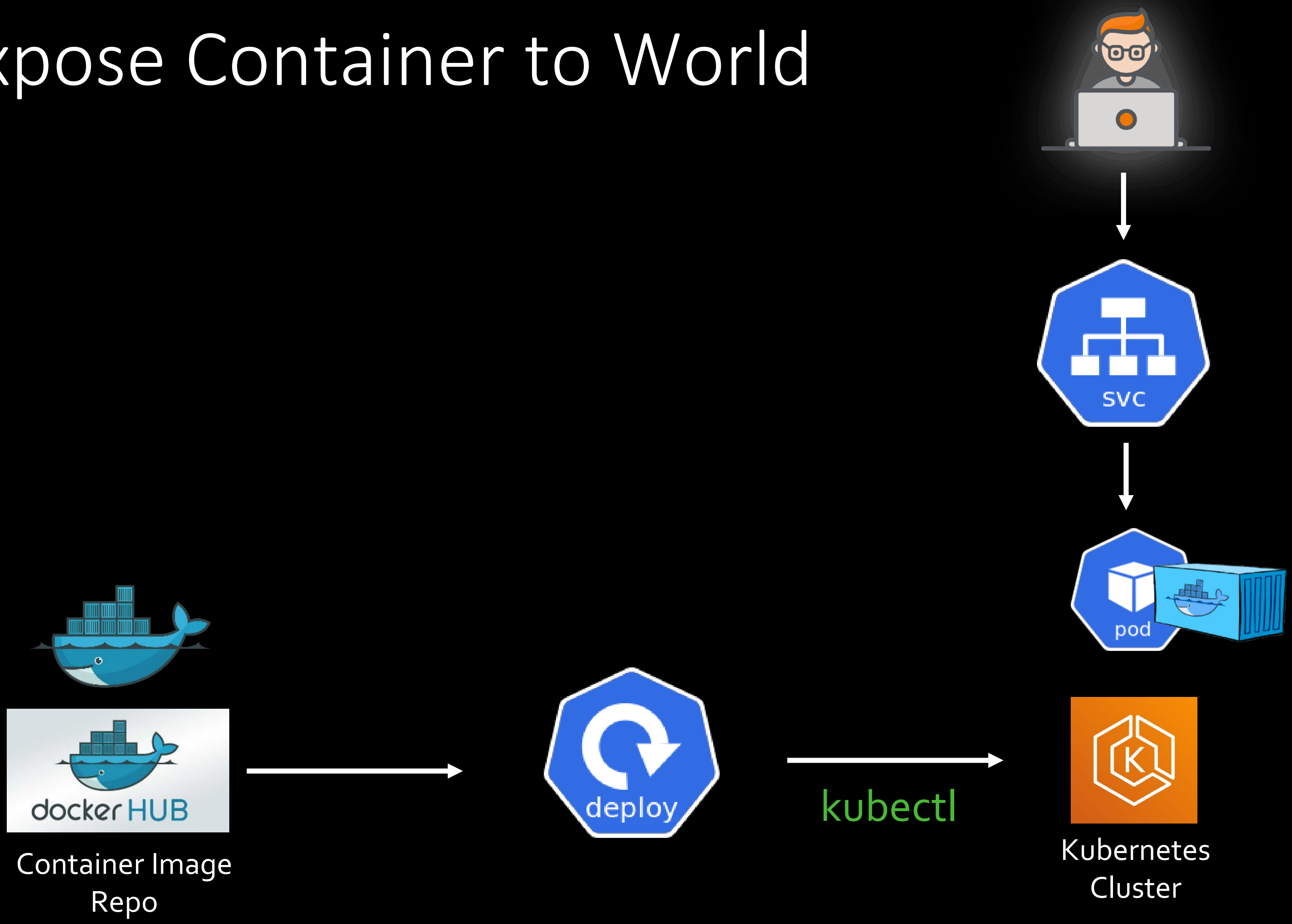
# #4 Prepare for Cloud – Image Repo



# #5 Deploy into Cloud (AWS)



# #5 Expose Container to World



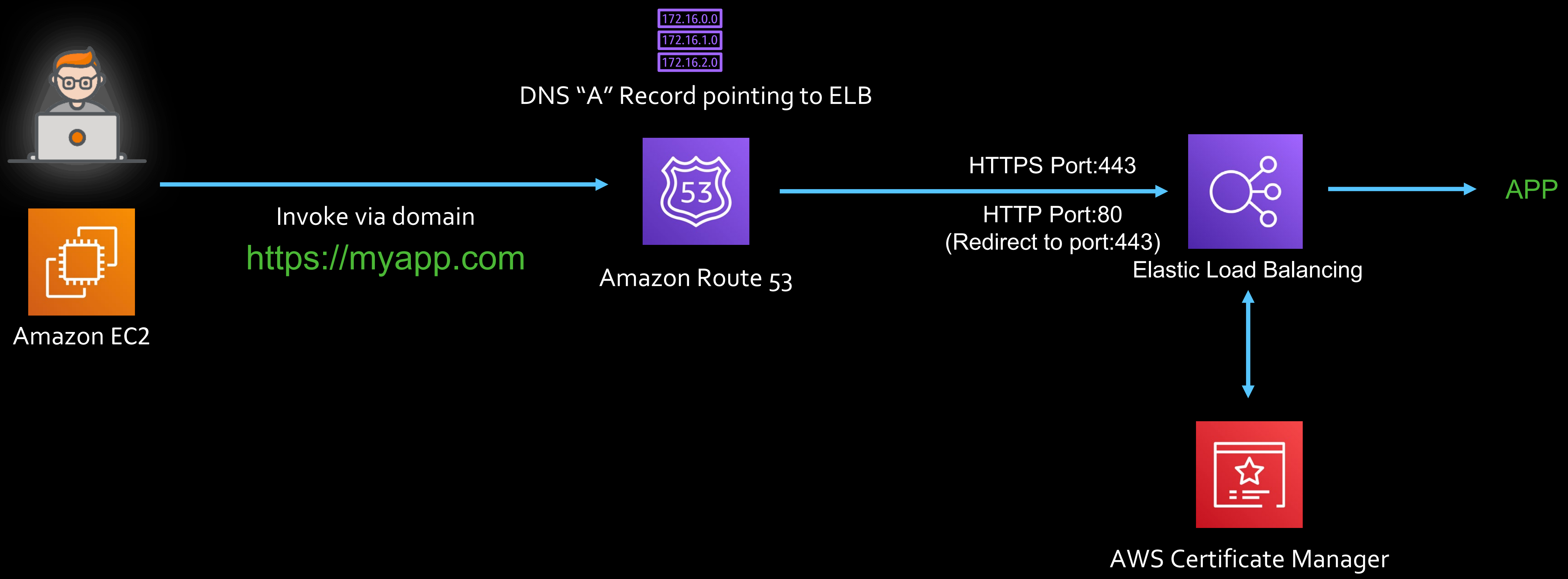
# Demo – Run CV from Kubernetes with Containerization Workflow

Raj Saha
cloudwithraj.com
Cloud With Raj

# Resume from EC2 ALB

Raj Saha
cloudwithraj.com
Cloud With Raj

# Load Balancer Custom Domain SSL Flow



Raj Saha
cloudwithraj.com
Cloud With Raj
linkedin.com/in/rajdeep-sa-at-aws/

Instructor Bio:

Sr. Specialist SA – Containers/Serverless@aws
Bestselling Udemy/Pluralsight author
Tech Advisor of crypto startup
Public speaker and guest lecturer
Author of multiple official AWS blogs
YouTuber with 54K subscribers
Previously - Distinguished Cloud Architect @Verizon

*Opinions are my own*