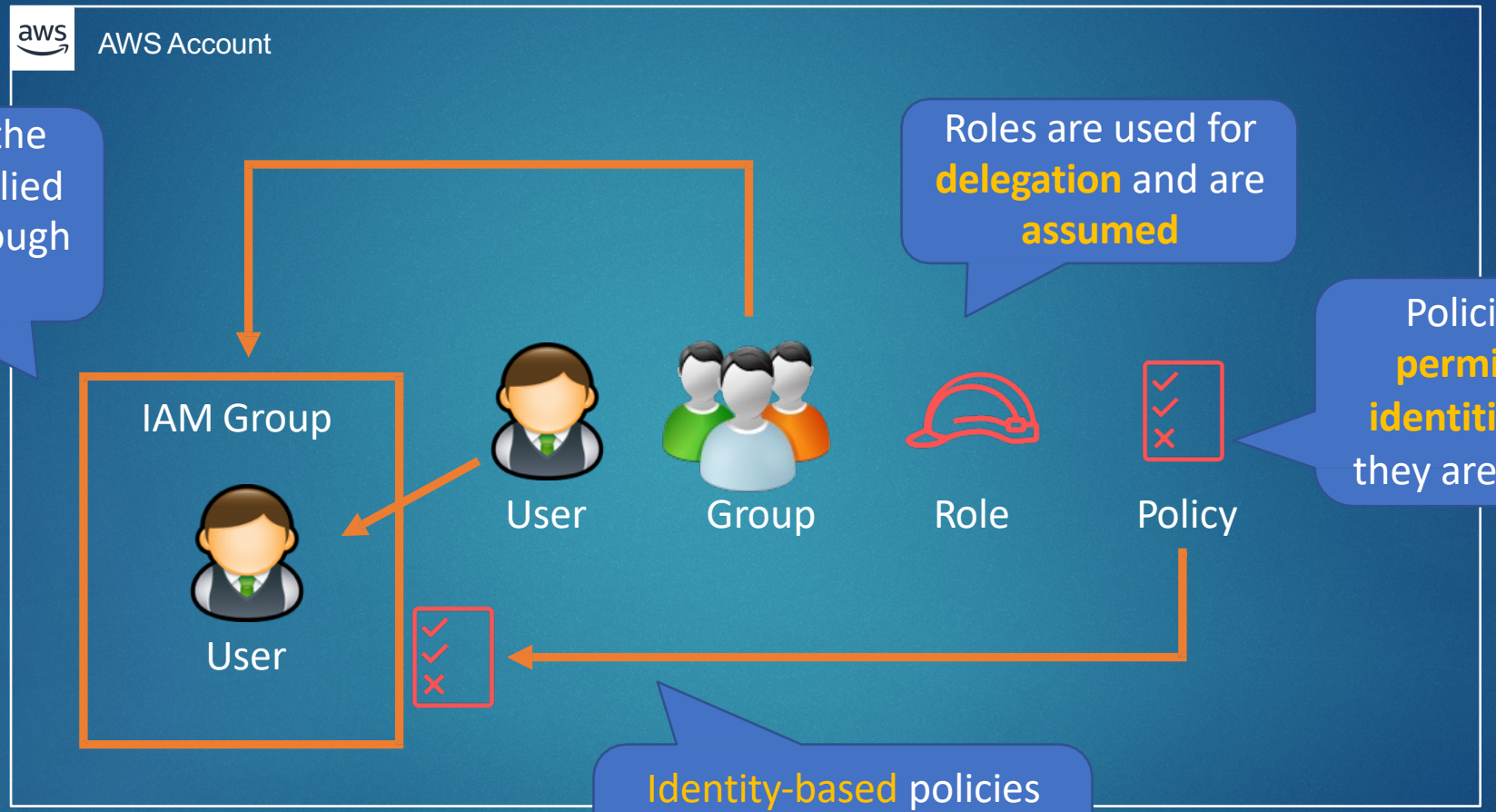
IAM Users, Groups, Roles, and Policies

Users, Groups, Roles and Policies



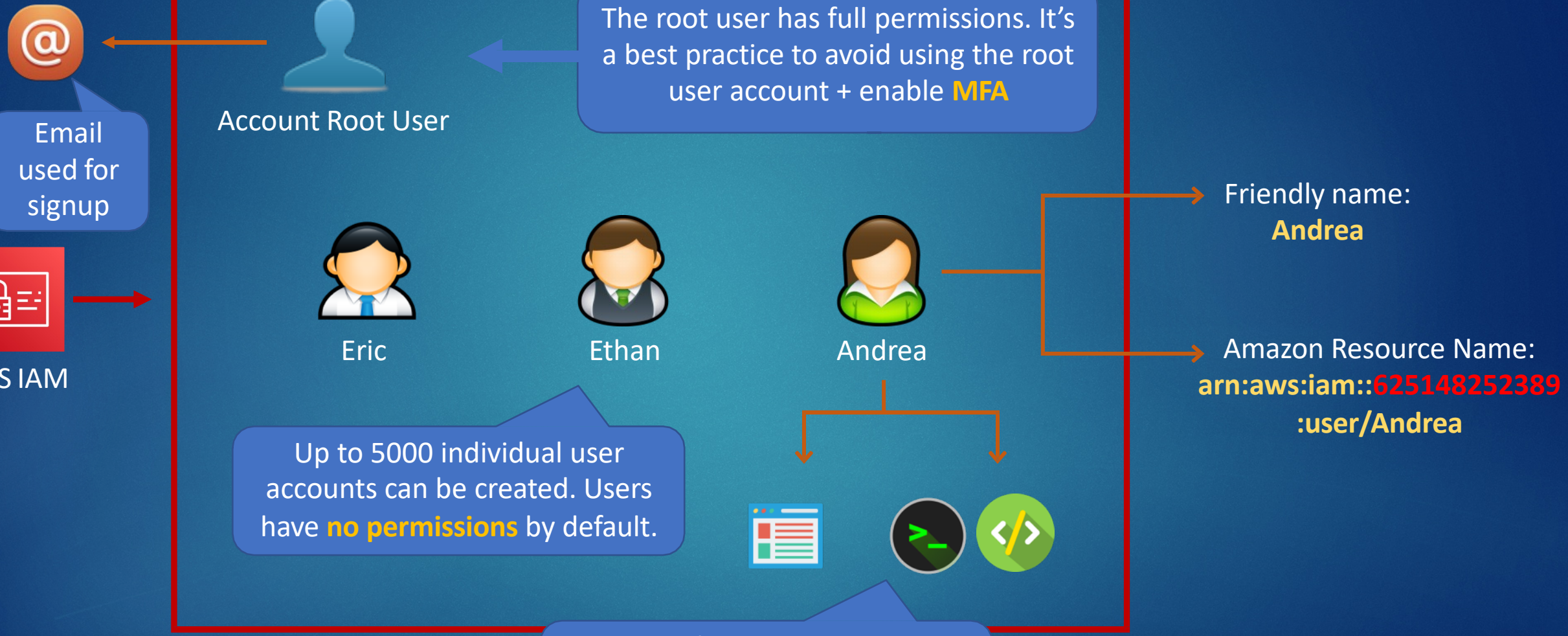
The user gains the **permissions** applied to the **group** through the **policy**

Roles are used for **delegation** and are **assumed**

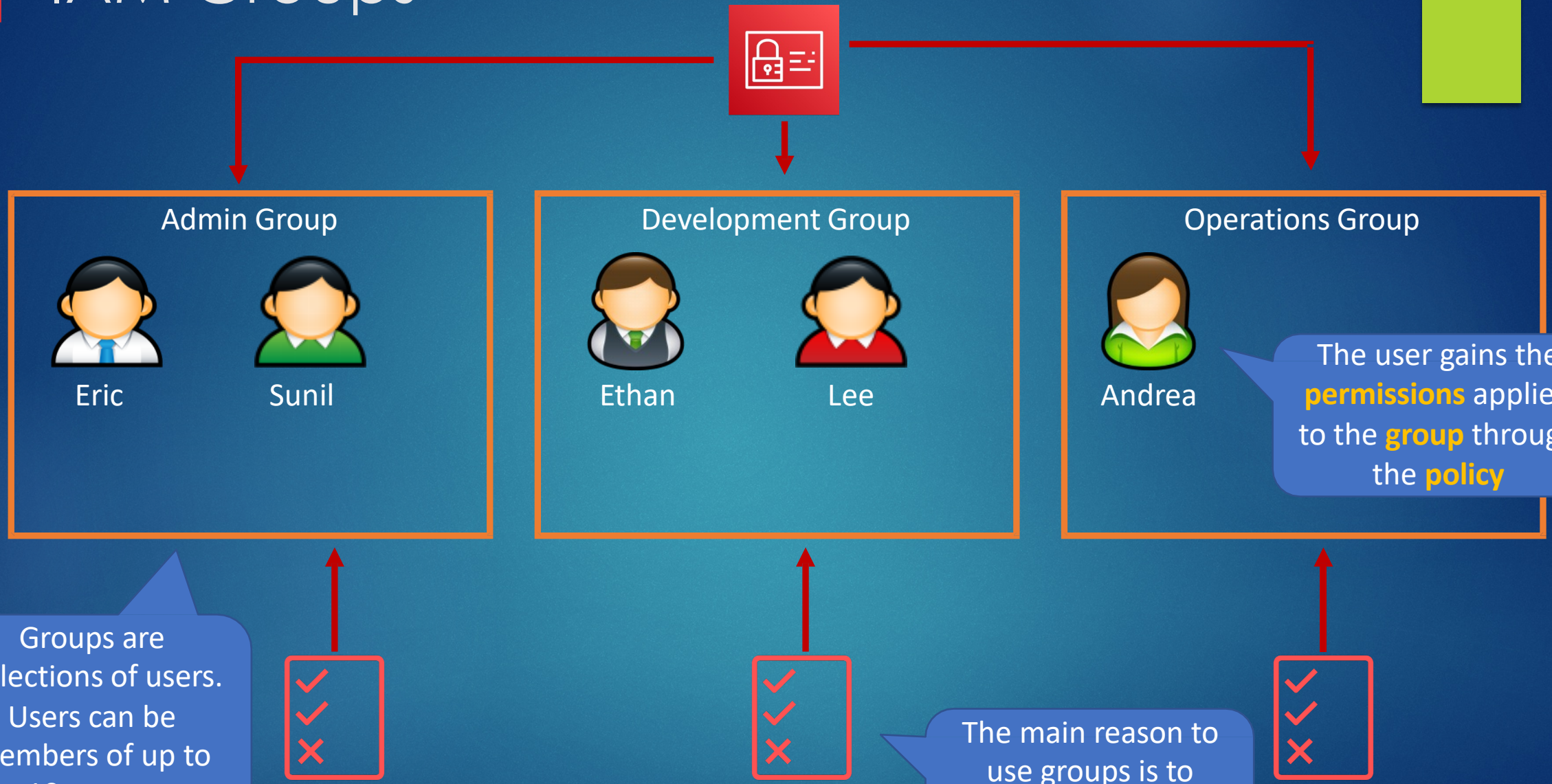
Policies define the **permissions** for the **identities** or **resources** they are associated with

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

IAM Users

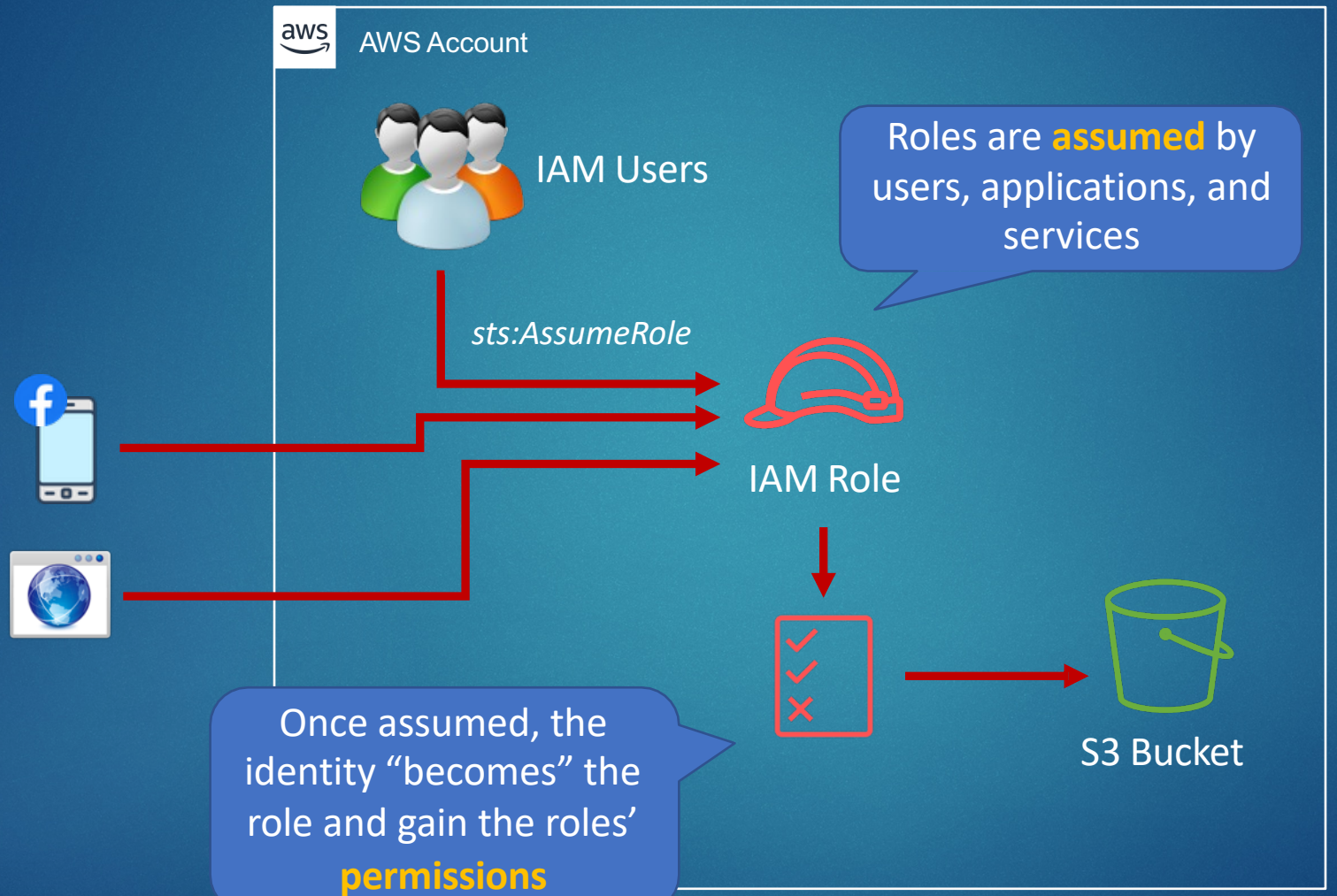

Email used for signup
AWS IAM
Account Root User
The root user has full permissions. It's a best practice to avoid using the root user account + enable MFA
Eric
Ethan
Andrea
Friendly name:
Andrea
Amazon Resource Name:
arn:aws:iam::625148252389
:user/Andrea
Up to 5000 individual user accounts can be created. Users have no permissions by default.
Authentication via username/password for console or access keys for API/CLI

IAM Groups


Admin Group
Eric
Sunil
Development Group
Ethan
Lee
Operations Group
Andrea
The user gains the permissions applied to the group through the policy
Groups are collections of users. Users can be members of up to 10 groups
The main reason to use groups is to apply permissions to users using policies

IAM Roles

An **IAM role** is an IAM **identity** that that has specific **permissions**



IAM Policies

Policies are **documents** that define **permissions** and are written in JSON

AdministratorAccess

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

All permissions are **implicitly denied** by default

User

Group

Role

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

Bucket Policy

```
{
    "Version": "2012-10-17",
    "Id": "Policy1561964929358",
    "Statement":[
    {
        "Sid": "Stmt1561964454052",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::515148227241:user/Paul"
        },
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::dctcompany",
        "Condition": {
            "StringLike": {
                "s3:prefix": "Confidential/*"
            }
        }
    }
    ]
}
```

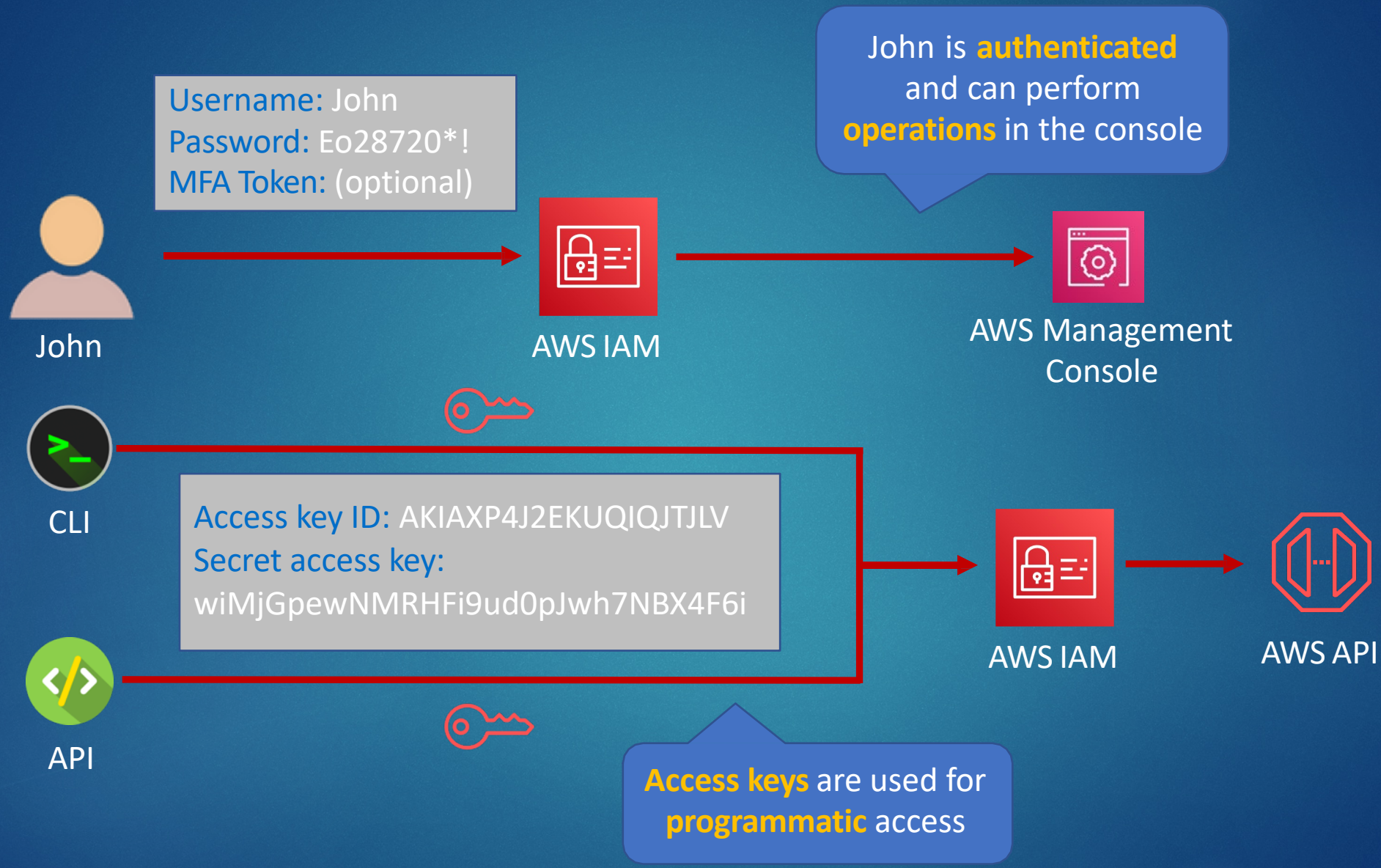
Resource-based policiesapply to **resources** such as S3 buckets or DynamoDB tables

S3 Bucket

Setup Individual User Account

IAM Authentication and MFA

IAM Authentication Methods



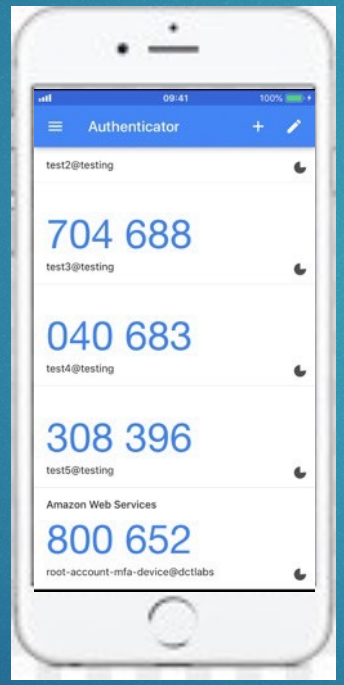
Multi-Factor Authentication

Something you **know**:

EJPx!*21p9%

Password

Something you **have**:



Something you **are**:

Multi-Factor Authentication

Something you **know**:

IAM User

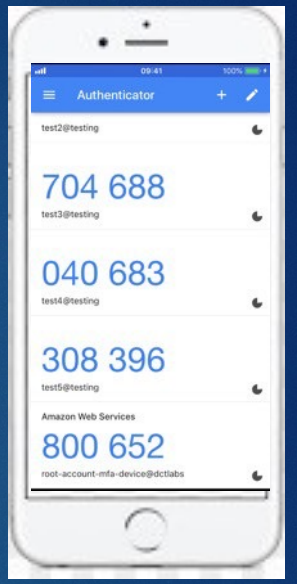
EJPx!*21p9%

Password

Something you **have**:

MFA Virtual MFA

e.g. Google Authenticator on your smart phone

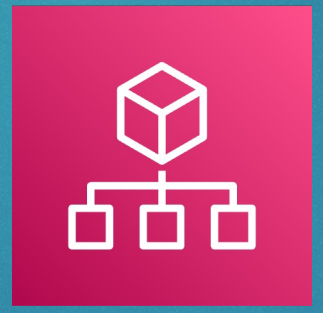


MFA Physical MFA

Physical tokens can be purchased from **third parties**

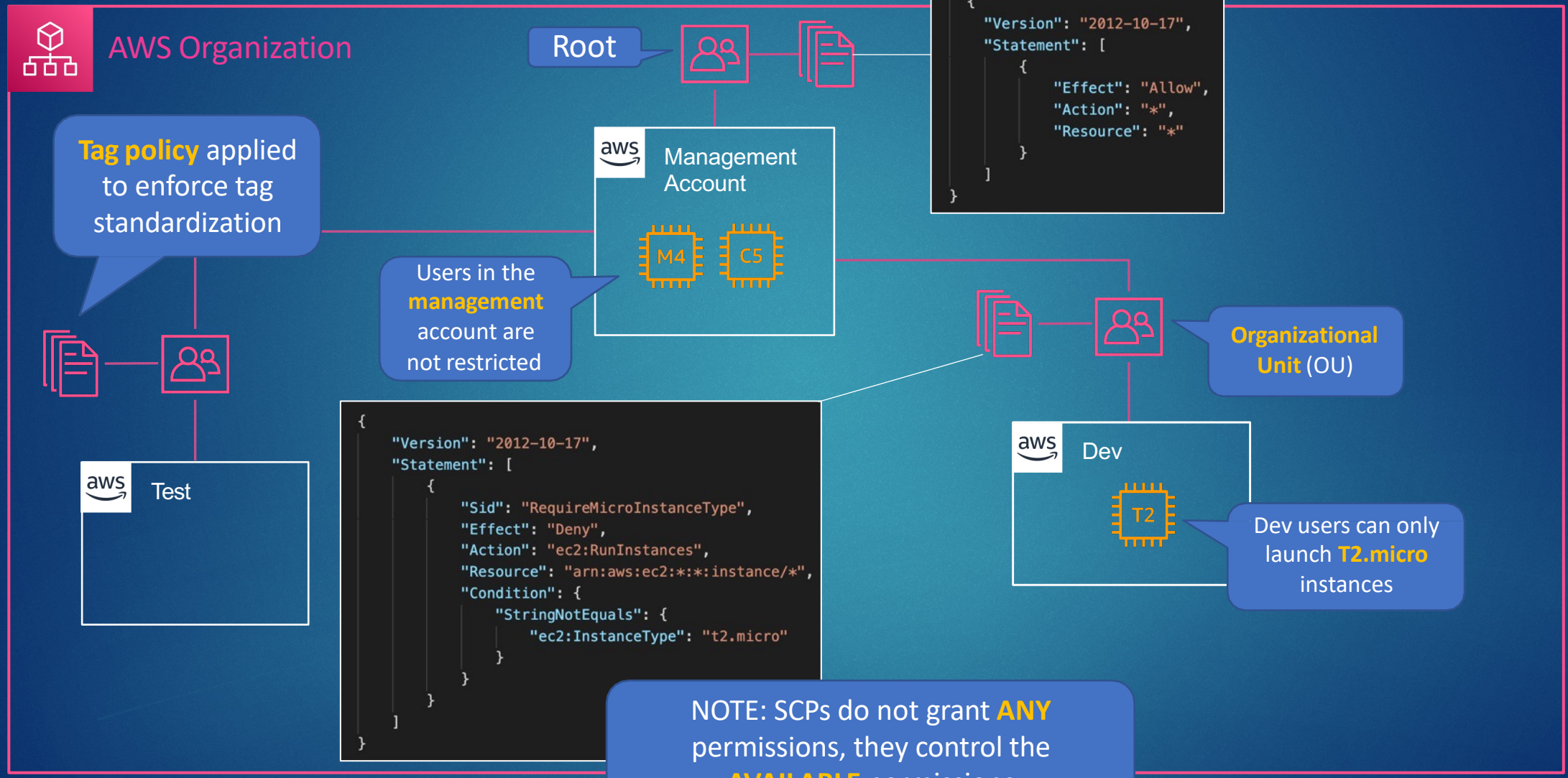
Setup Multi-Factor Authentication (MFA)

Service Control Policies (SCPs)



Service Control Policies

SCPs control the maximum available permissions




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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireMicroInstanceType",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringNotEquals": {
                    "ec2:InstanceType": "t2.micro"
                }
            }
        }
    ]
}
```

IAM Password Policy

IAM Best Practices

AWS IAM Best Practices

- Lock away your AWS account root user access keys
- Create individual IAM users
- Use groups to assign permissions to IAM users
- Grant least privilege
- Get started using permissions with AWS managed policies
- Use customer managed policies instead of inline policies
- Use access levels to review IAM permissions
- Configure a strong password policy for your users
- Enable MFA

AWS IAM Best Practices

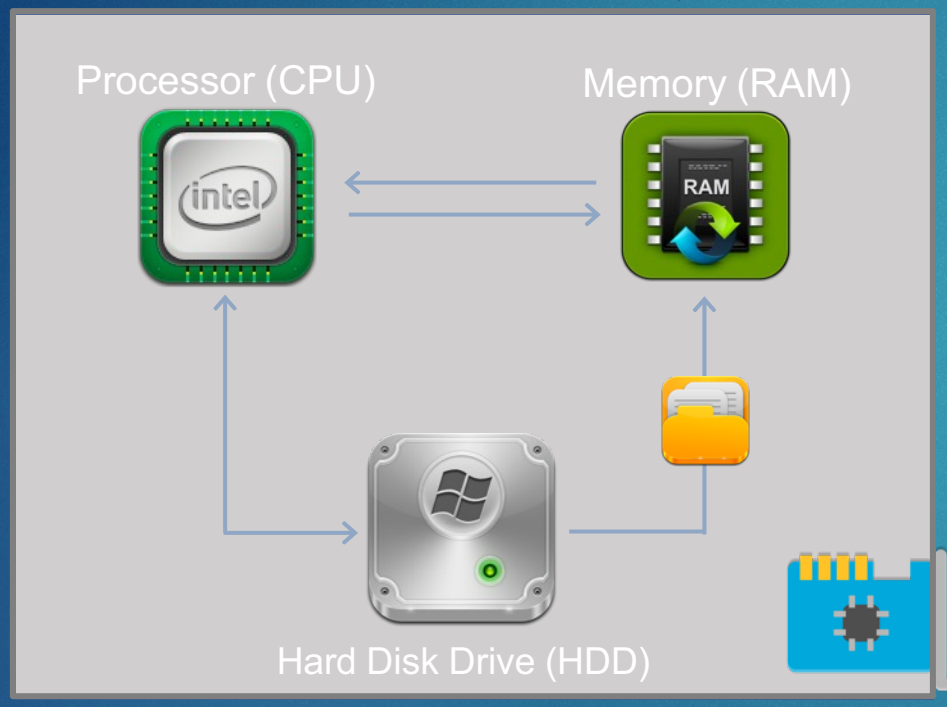
- Use roles for applications that run on Amazon EC2 instances
- Use roles to delegate permissions
- Do not share access keys
- Rotate credentials regularly
- Remove unnecessary credentials
- Use policy conditions for extra security
- Monitor activity in your AWS account

AWS Compute Services

Computing Basics

Central Processing Unit (CPU)

RAM is non-persistent storage



Random Access Memory (RAM)

Files/data are loaded into **memory**

Data is **persistent**

Network Interface Card (NIC)

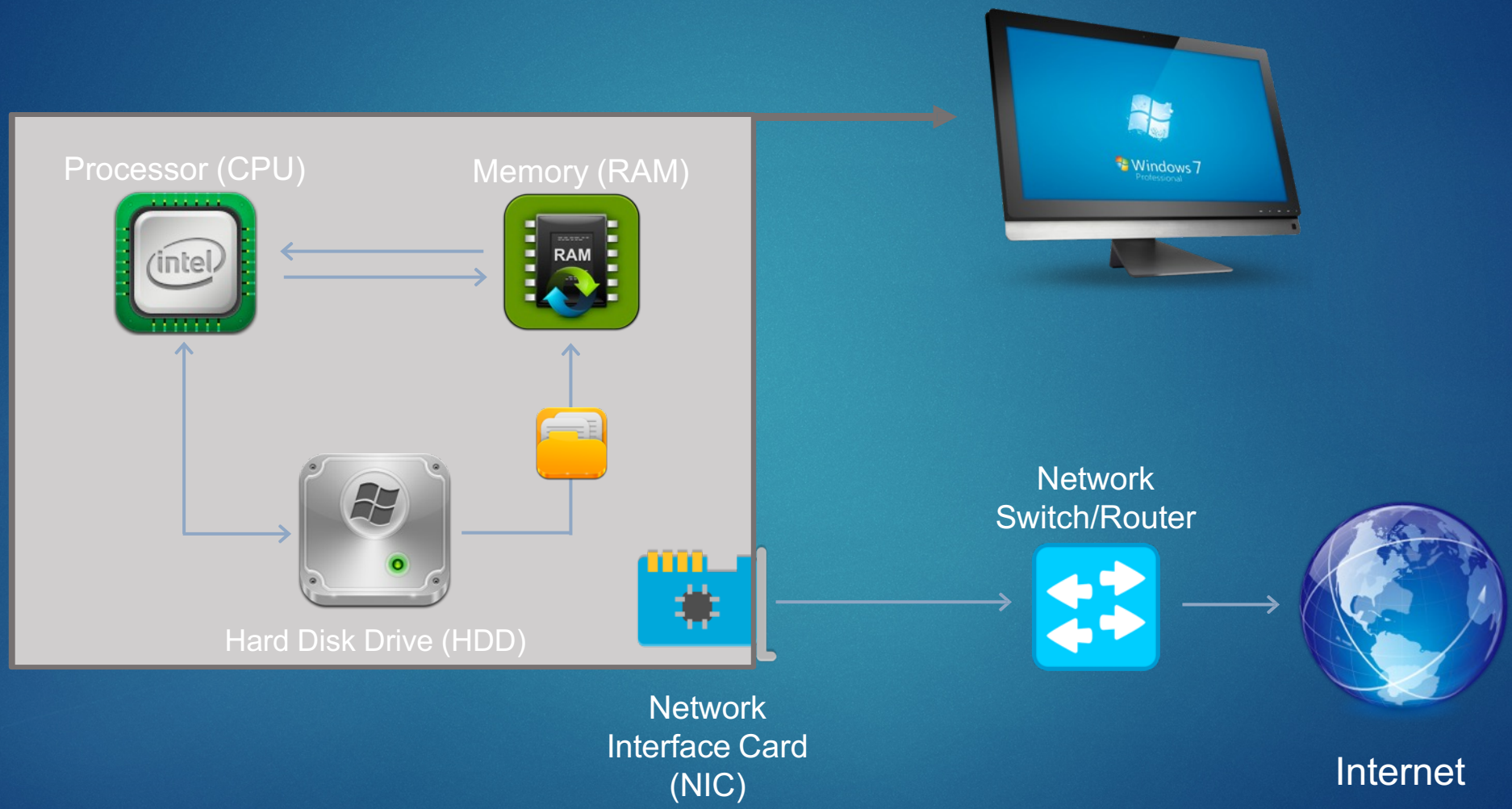
Network Switch/Router

Internet

Measurements:

- CPU is measured in Gigahertz (Ghz)
- RAM is measured in Gigabyte (GB)
- HDD is measured in Gigabyte (GB)
- NIC is measured in Megabits per second (Mbps) or Gigabits per second (Gbps)

Computing Basics



Servers vs Desktops/Laptops

Server

Laptop

Desktop

Server Hardware Build:

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

Servers can be used by many users over a network

Client / Server Computing

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

Web Server

Port: 80

Protocol: HTTP

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

File Server

Port: 445

Protocol: SMB

Email Server

Port: 25

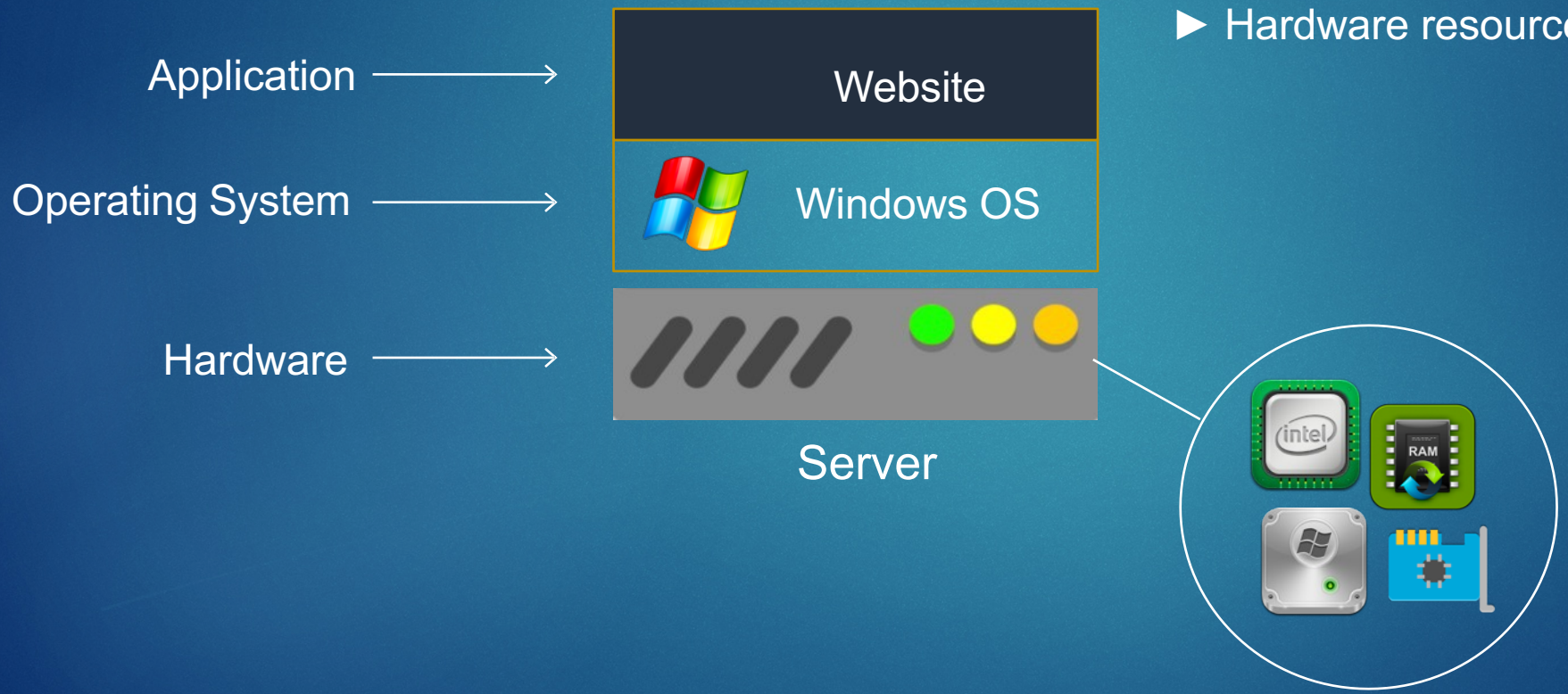
Protocol: SMTP

Server Virtualization

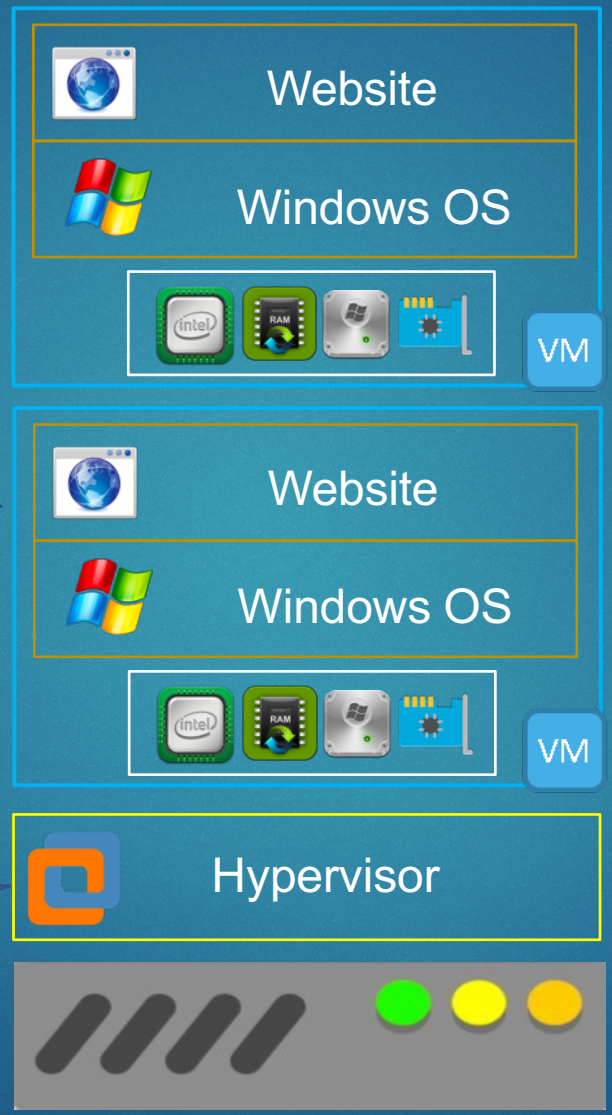
Without Server Virtualization

Limitations:

- OS is tied to hardware (no portability)
- Hardware resources may be underutilized



Server Virtualization



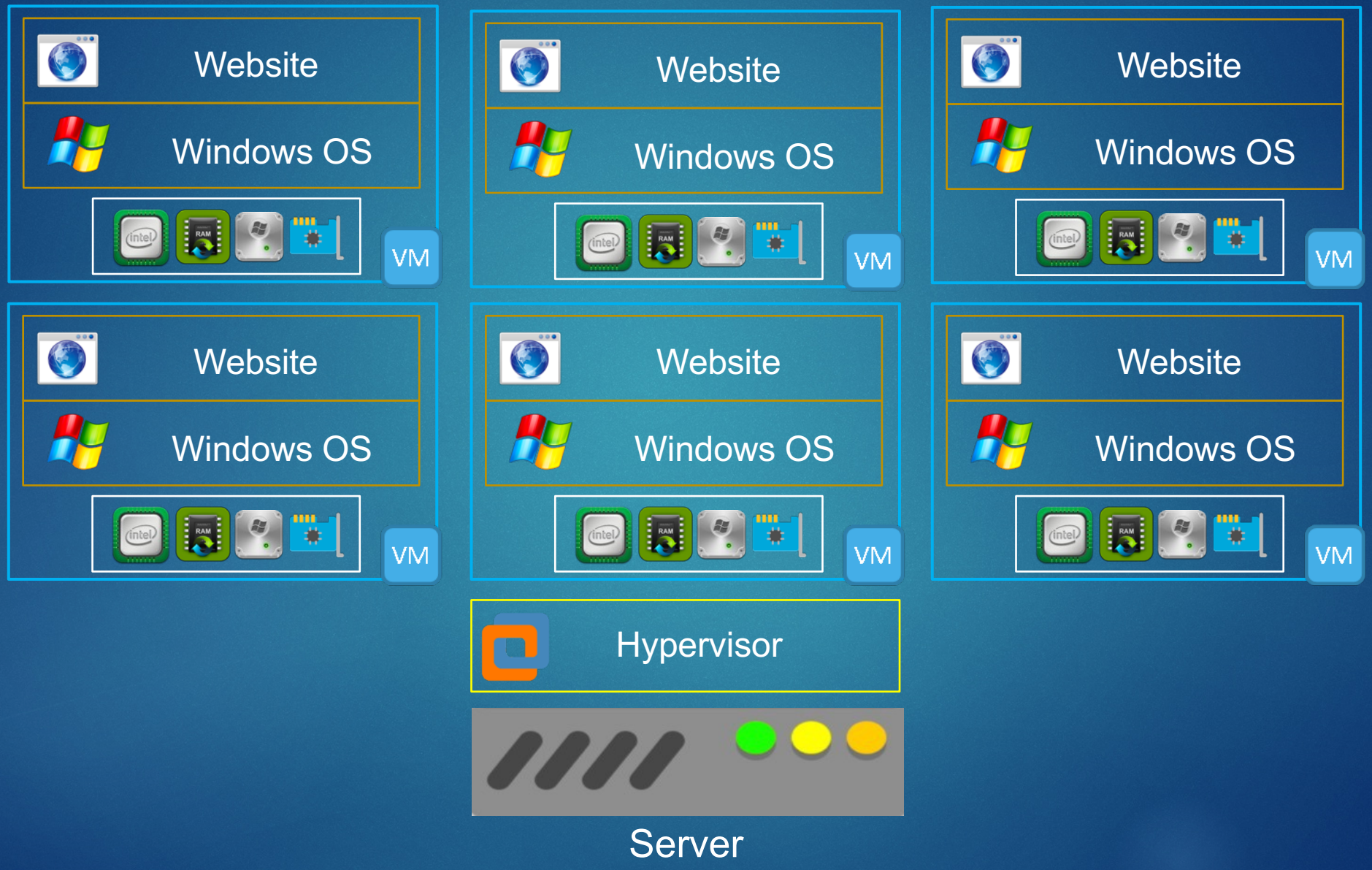
Many **VMs** can run on the same physical hardware

This is known as a virtual server, virtual machine, or **instance**

Virtual hardware is presented to the OS

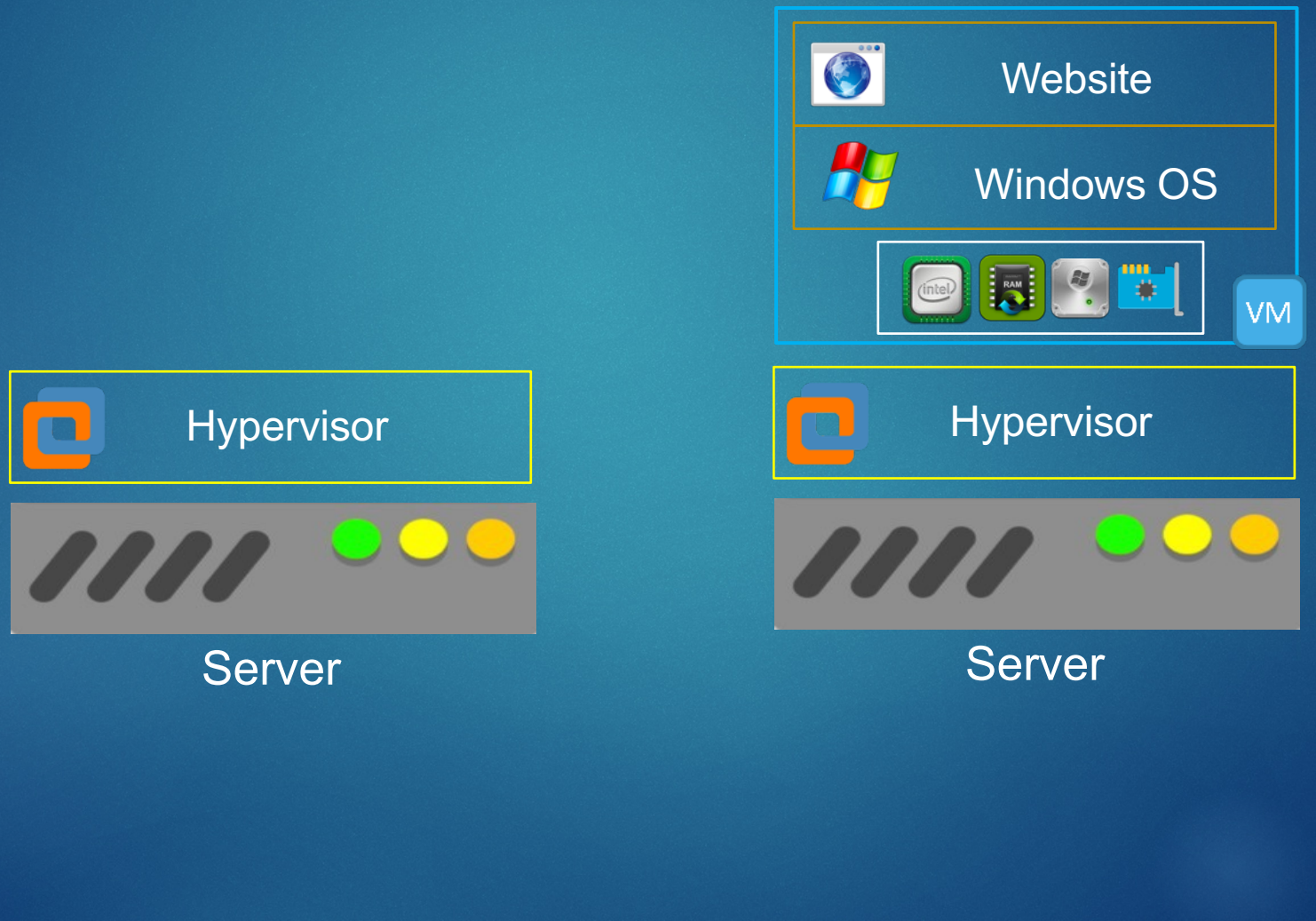
The **hypervisor** creates a layer of abstraction

Server Virtualization

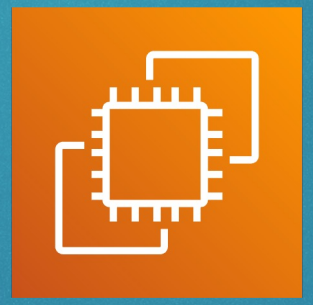


Server Virtualization

Server Virtualization



Amazon Elastic Compute Cloud (EC2)

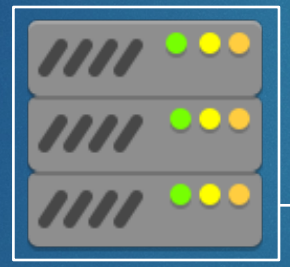


Amazon EC2

EC2 instances run Windows, Linux, or MacOS

An **EC2 instance** is a virtual server

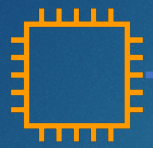
EC2 hosts are **managed by AWS**



EC2 Host Server

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

Launching an EC2 Instance



EC2 Instance

Select an **instance type**

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

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

Amazon Machine Image (AMI)

EBS Snapshot

Linux

Microsoft Windows

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

Customized AMI

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

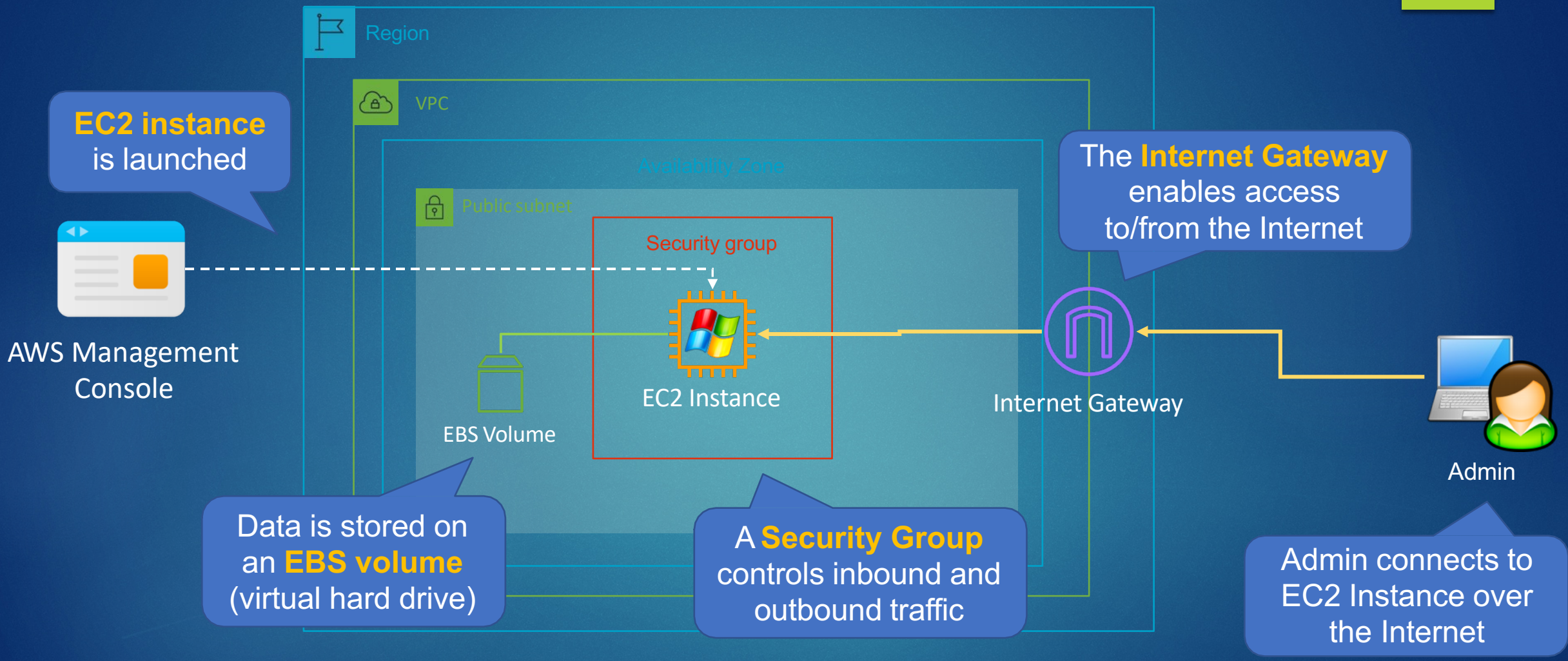
The **instance type** defines the hardware profile (and cost)

Benefits of Amazon EC2

- **Elastic computing** – easily launch hundreds to thousands of EC2 instances within minutes
- **Complete control** – you control the EC2 instances with full root/administrative access
- **Flexible** – Choice of instance types, operating systems, and software packages
- **Reliable** – EC2 offers very high levels of availability and instances can be rapidly commissioned and replaced
- **Secure** – Fully integrated with Amazon VPC and security features
- **Inexpensive** – Low cost, pay for what you use

Launch EC2 Instances (Windows + Linux)

Amazon EC2 Instance in a Public Subnet



EC2 Instance Connect and SSH

RDP to Windows Instance

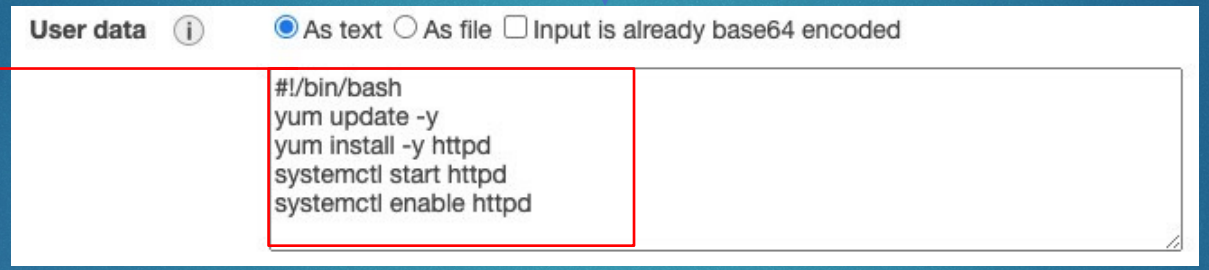
Amazon EC2 User Data and Metadata

Amazon EC2 User Data

AWS Management Console

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

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



Limited to **16 KB**

EC2 Instance

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

Amazon EC2 Metadata

- Instance metadata is data about your EC2 instance
- Instance metadata is available at http://169.254.169.254/latest/meta-data
- Examples:

```
[ec2-user@ip-172-31-42-248 ~]$ curl http://169.254.169.254/latest/meta-data
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
events/
hibernation/
hostname
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
```

Amazon EC2 Metadata

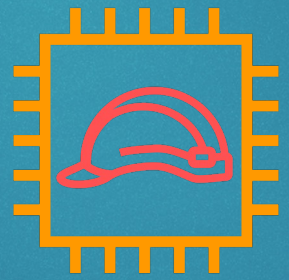
- Examples ctd.:

```
[ec2-user@ip-172-31-42-248 ~]$ curl http://169.254.169.254/latest/meta-data/local-ipv4
172.31.42.248[ec2-user@ip-172-31-42-248 ~]$
```

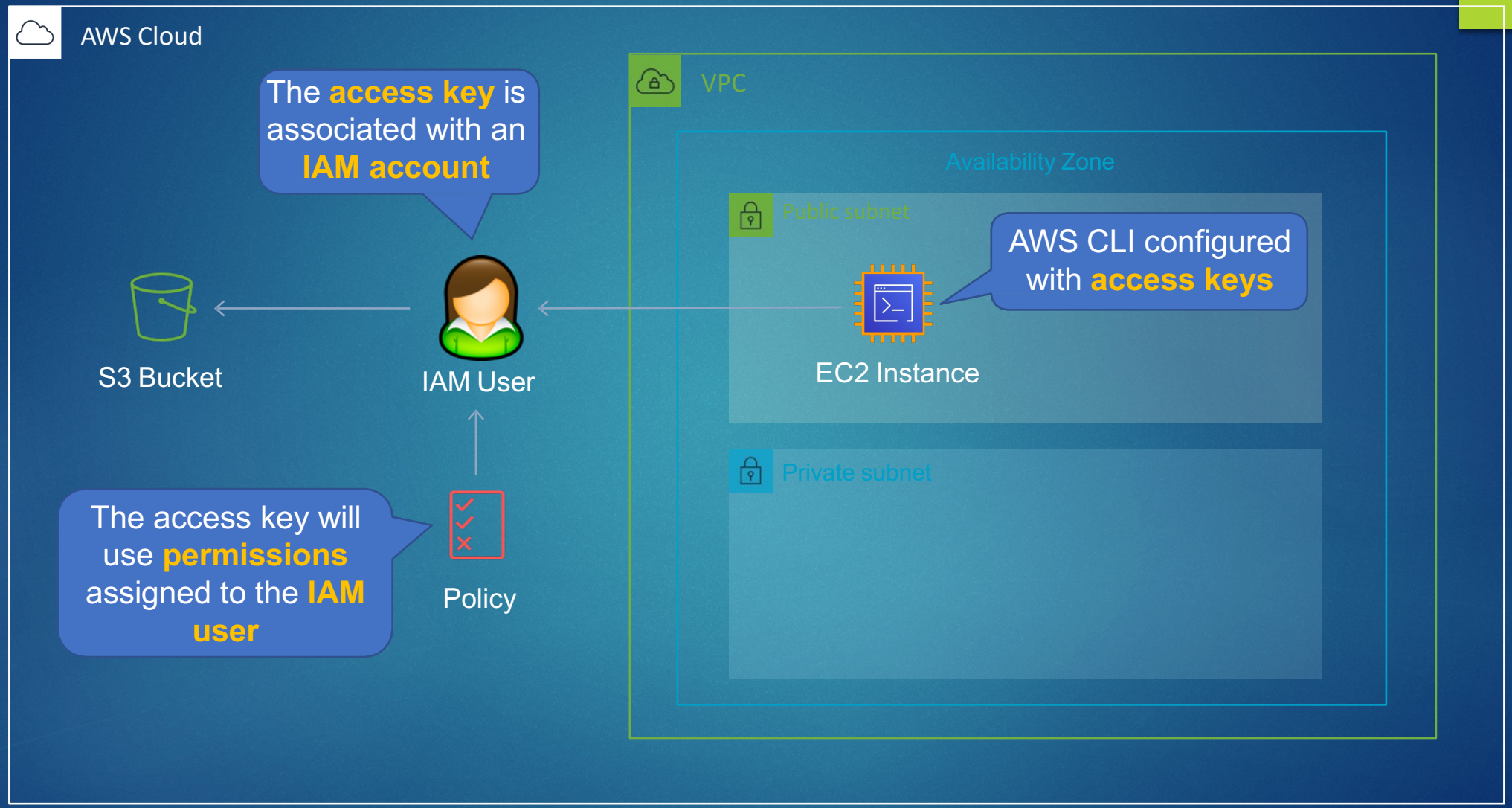
```
[ec2-user@ip-172-31-42-248 ~]$ curl http://169.254.169.254/latest/meta-data/public-ipv4
3.26.54.18[ec2-user@ip-172-31-42-248 ~]$
```

[HOL] Launch Instance with User Data and Metadata

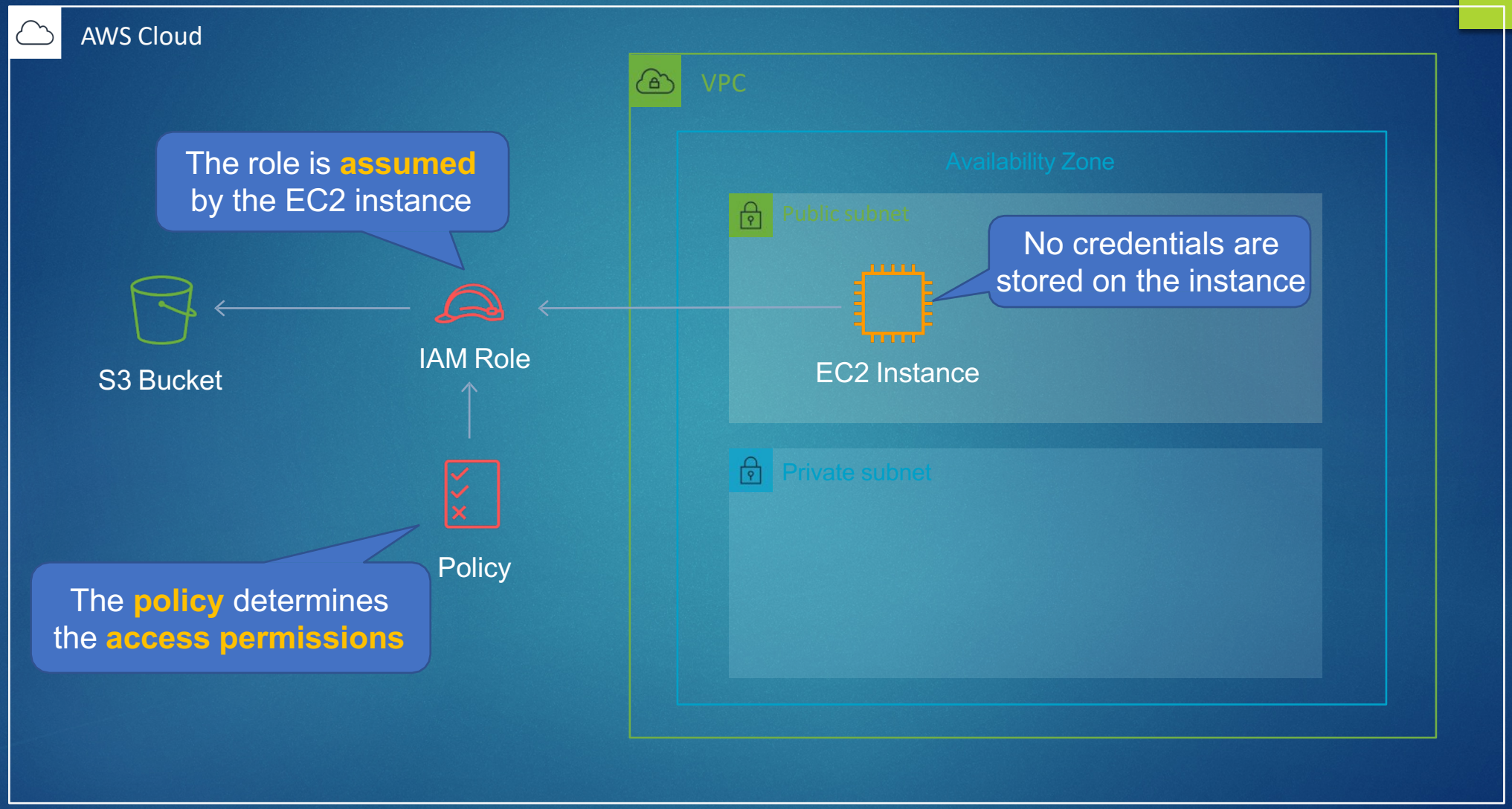
Accessing Services – Access Keys and IAM Roles



Access Keys


AWS Cloud
VPC
Availability Zone
Public subnet
Private subnet
The access key is associated with an IAM account
AWS CLI configured with access keys
S3 Bucket
IAM User
EC2 Instance
Policy
The access key will use permissions assigned to the IAM user

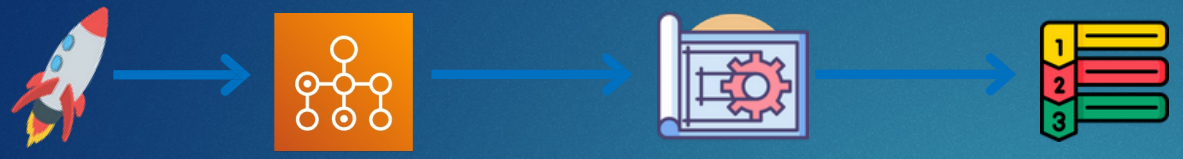
Amazon EC2 Instance Profiles (IAM Roles for EC2)



Access Keys and IAM Roles

AWS Batch

AWS Batch



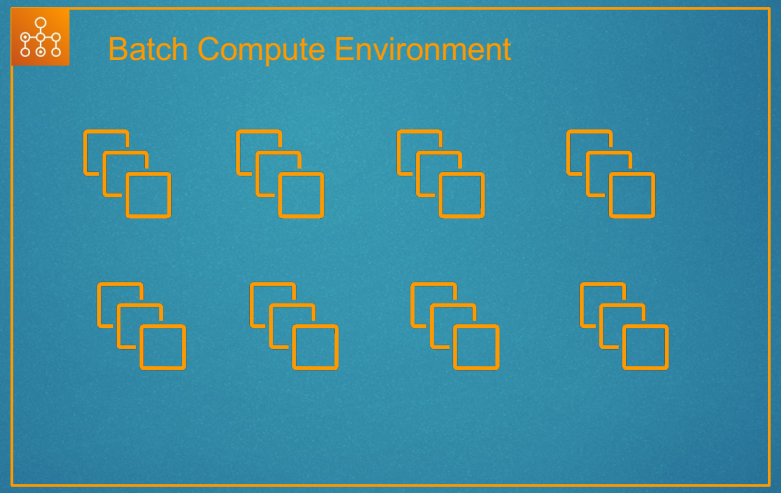
Launch a **Batch Job**

Job **Definition**

Job **Queue**

A job is submitted to a **queue** until **scheduled** onto a compute environment

A job is a unit of work such as a **shell script**, **executable** or **Docker container image**



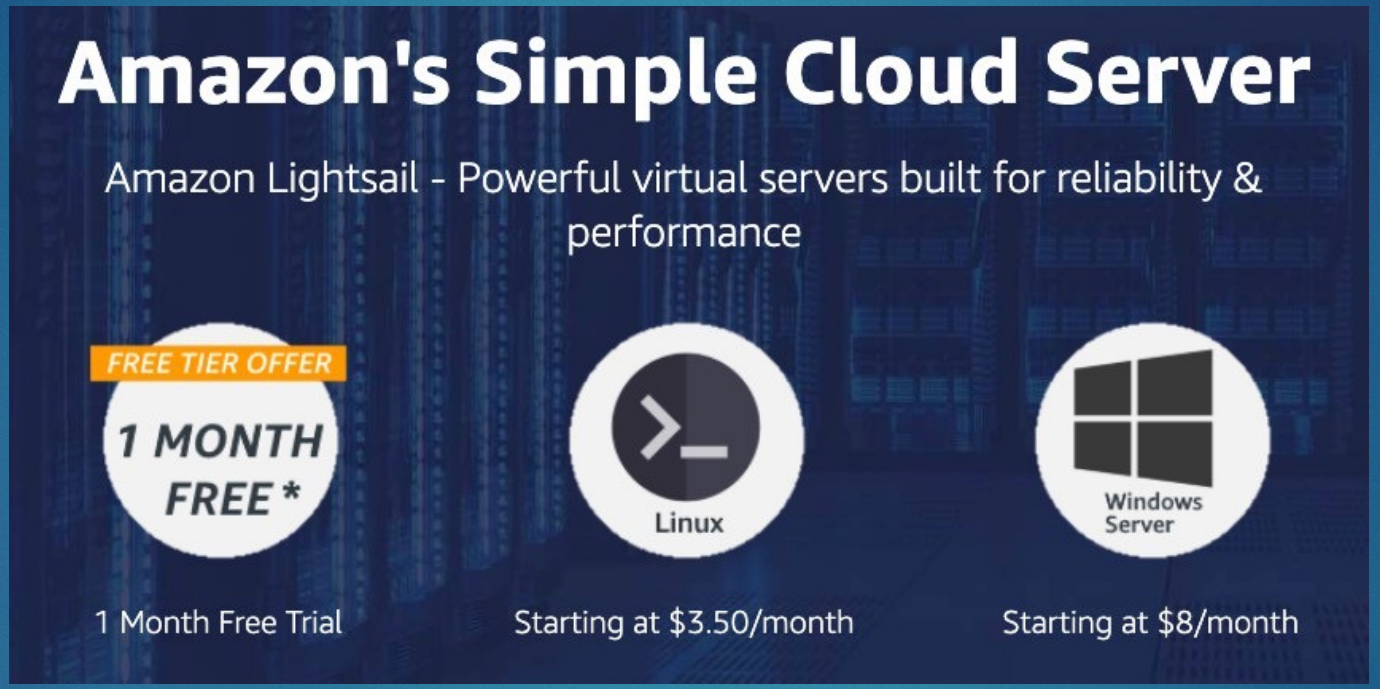
Managed or **unmanaged** resources used to run the job

Batch **launches**, **manages**, and **terminates** resources as required (EC2 and ECS/Fargate)

Amazon LightSail

Amazon LightSail

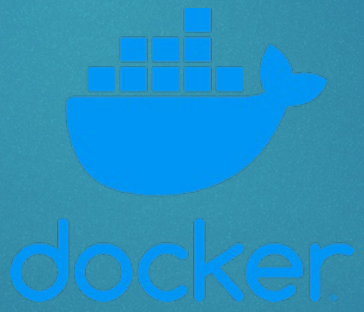
- Low cost and ideal for users with less technical expertise
- Compute, storage, and network
- Preconfigured virtual servers



- Virtual servers, databases and load balancers
- SSH and RDP access
- Can access Amazon VPC

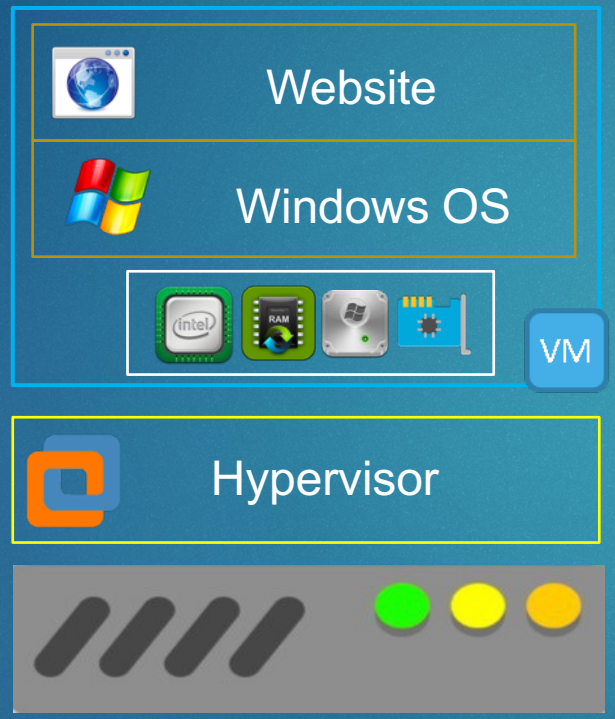
Exam tip: typically comes up in use cases where an easy method of deploying a virtual server is required by a user with little or no AWS expertise

Docker Containers and Microservices



Server Virtualization vs Containers

Every VM/instance needs an **operating system** which uses significant resources



Docker Containers

Containers **start up** very **quickly**

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

Containers are very resource **efficient**

Each container is **isolated** from other containers

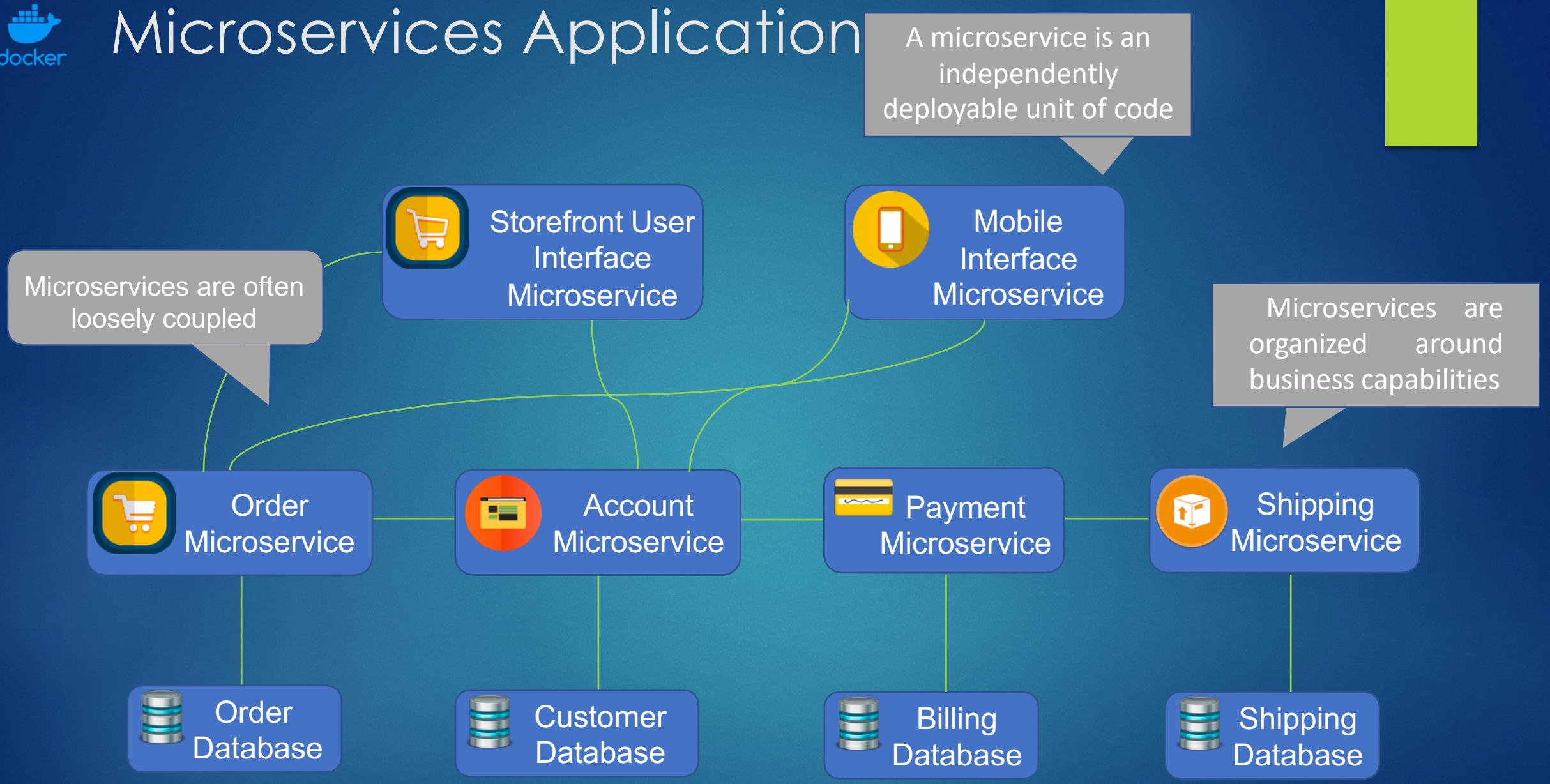
Monolithic Application

Monolithic Application

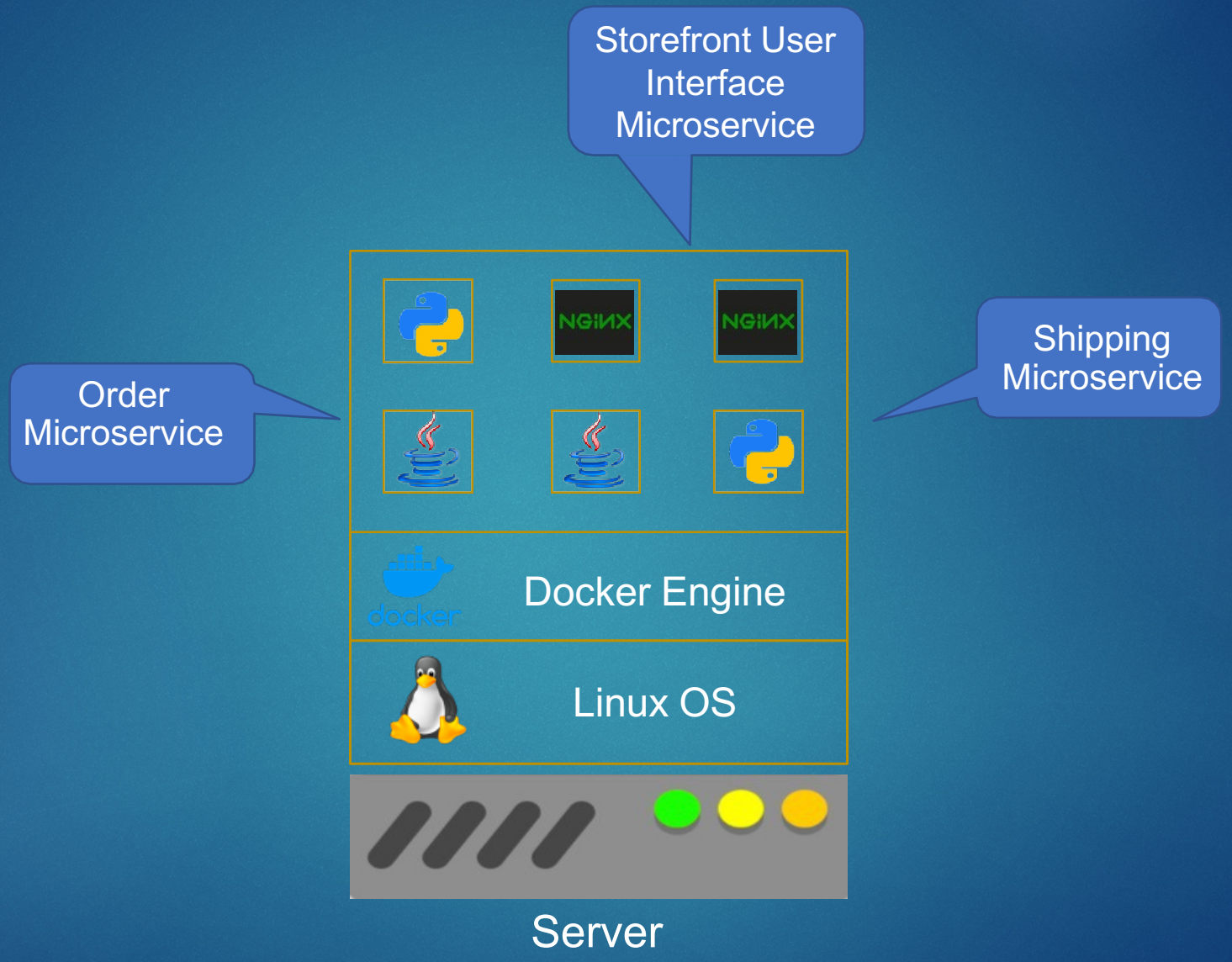
Updates to, or failures of, any single component can take down the whole application

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

Microservices Application



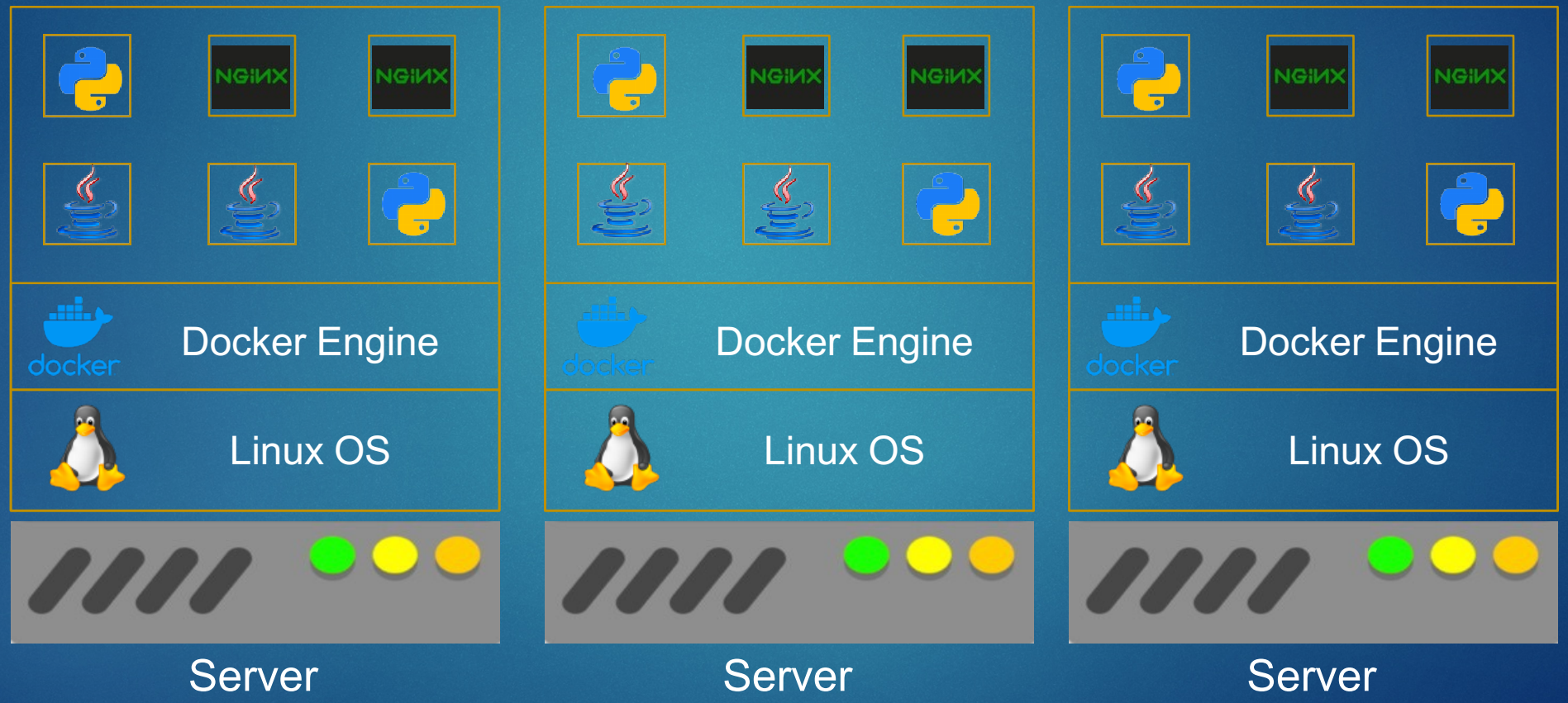
Microservices Application



Microservices Application

Microservices can also be **spread** across hosts

Many instances of each microservice can run on each host



Amazon Elastic Container Service (ECS)

Amazon ECS

Amazon Elastic Container Service

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

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

Task Definition

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

Availability Zone

Availability Zone

ECS Cluster

ECS Service

Auto Scaling group

ECS Container instance

Task

Task

ECS Container instance

Task

Task

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

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

Amazon Elastic Container Registry

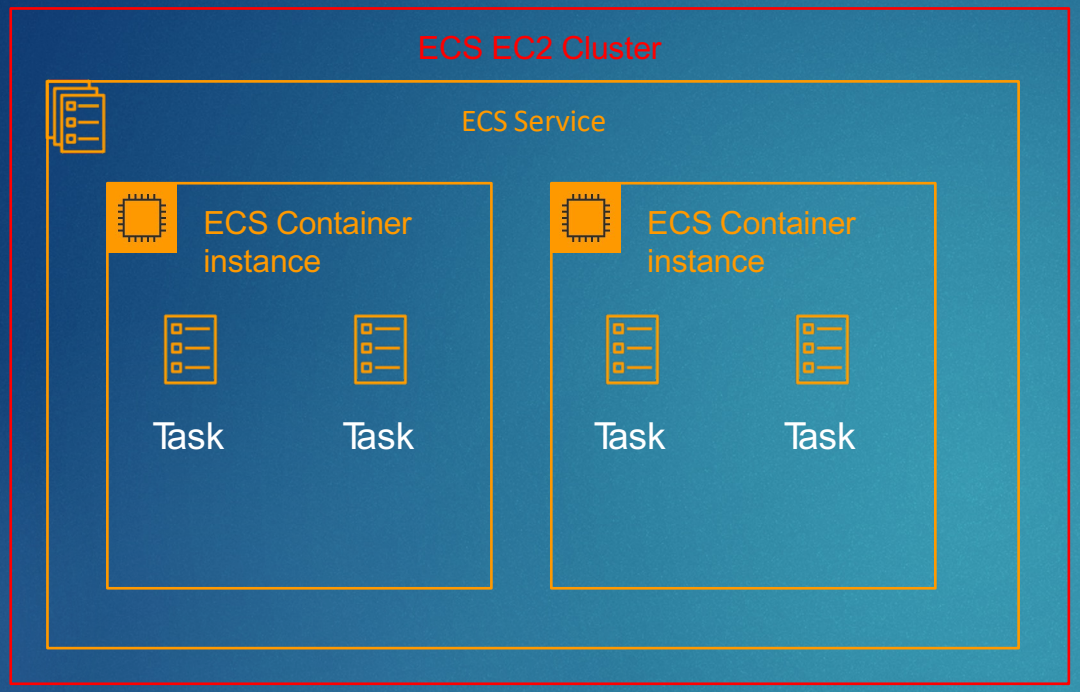
Registry

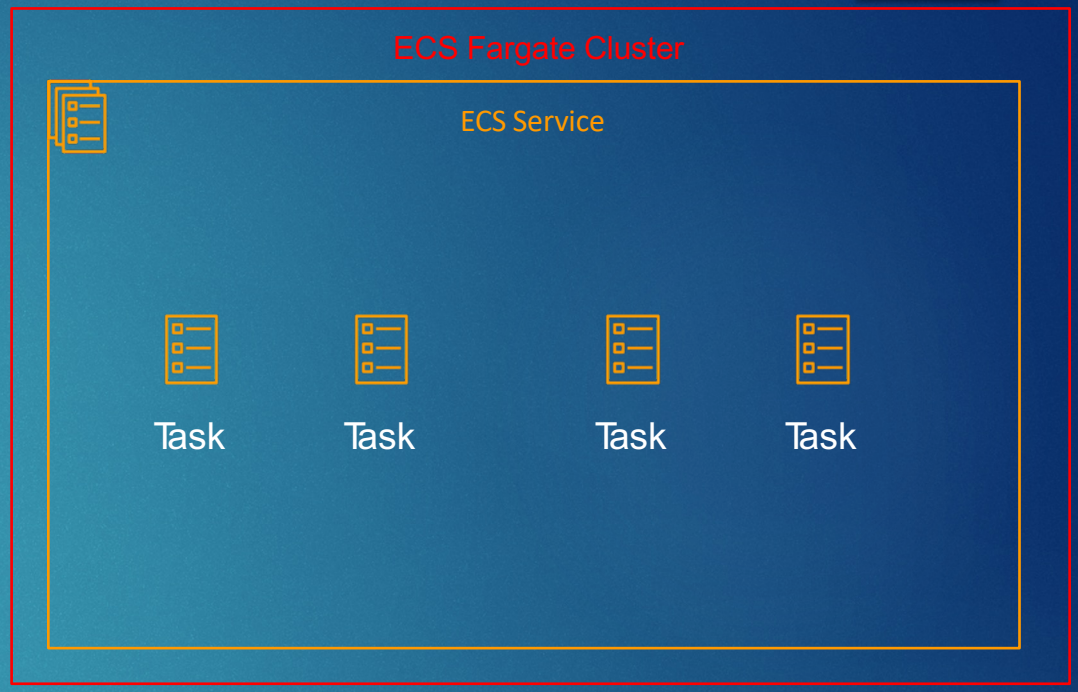
Image

Image

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

Amazon ECS





EC2 Launch Type

- You explicitly provision EC2 instances
- You're responsible for managing EC2 instances
- Charged per running EC2 instance
- EFS and EBS integration
- You handle cluster optimization
- More granular control over infrastructure

Fargate Launch Type

- Fargate automatically provisions resources
- Fargate provisions and manages compute
- Charged for running tasks
- No EFS and EBS integration
- Fargate handles cluster optimization
- Limited control, infrastructure is automated

Launch Docker Container on ECS