Cloud computing

RISR CESI

Cloud Computing

Introduction

Introduction

What is cloud computing?

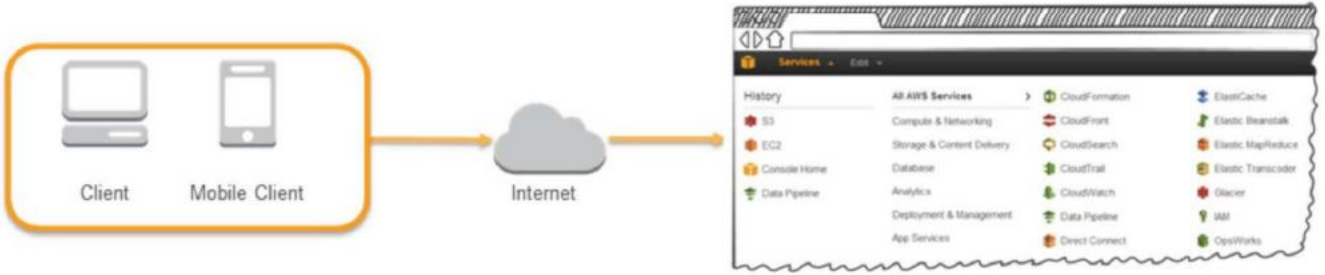
Cloud computing is on-demand delivery of IT resources and applications via the Internet with pay-as-you-go pricing.

Essential Characteristics

Essential Characteristics

On-Demand Self Services & Broad Network Access

- User provisions computing resources as needed.
- User interacts with cloud service provider through an online control panel.
- Clear solutions are available through a variety of network-connected devices and over varying platforms.



Essential Characteristics

Resource Pooling

Securely separate resources to service multiple customers.

Essential Characteristics

Rapid Elasticity

Resources are quickly scalable and flexible based on business needs.

Essential Characteristics

Measured Service

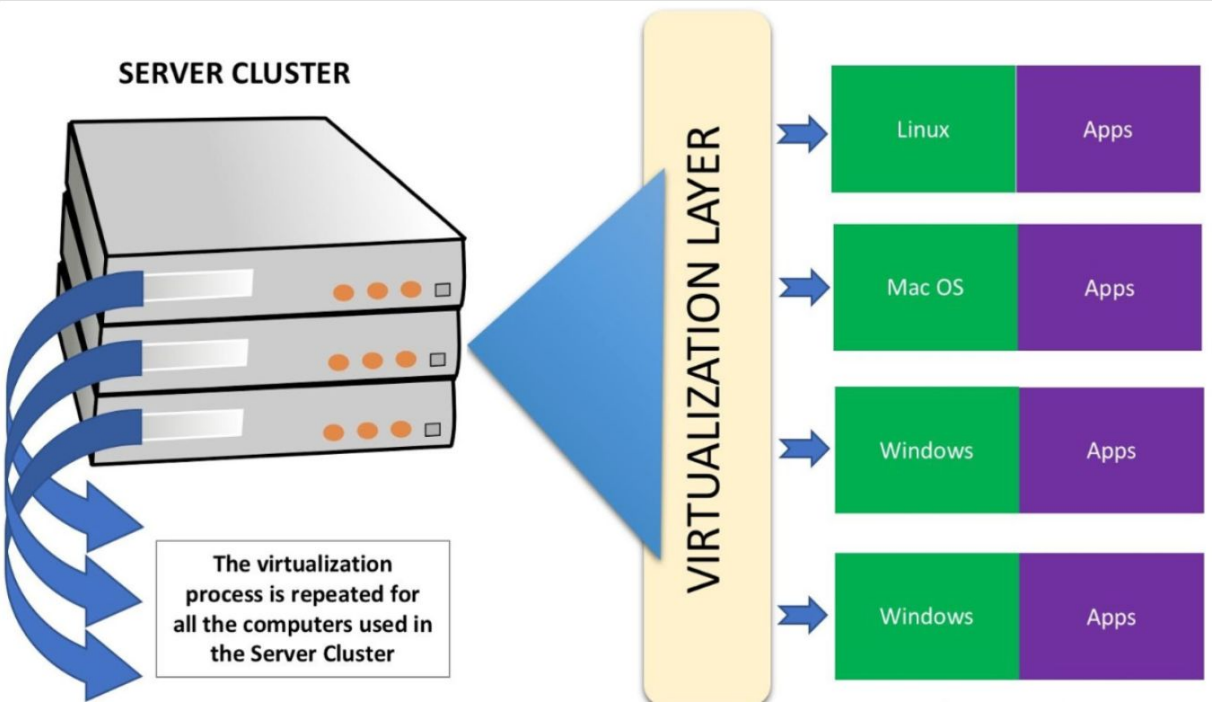
Pay for services as you go.

Electrical services analogy

Reminder

Infrastructure

Virtualisation

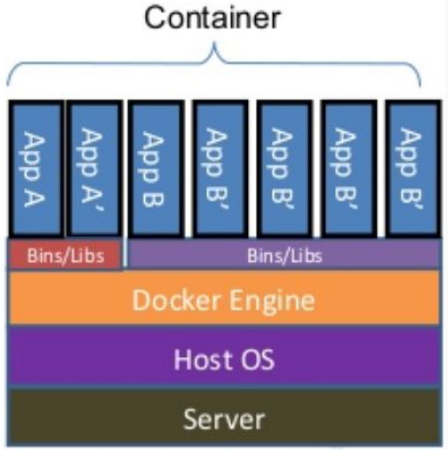


Containers

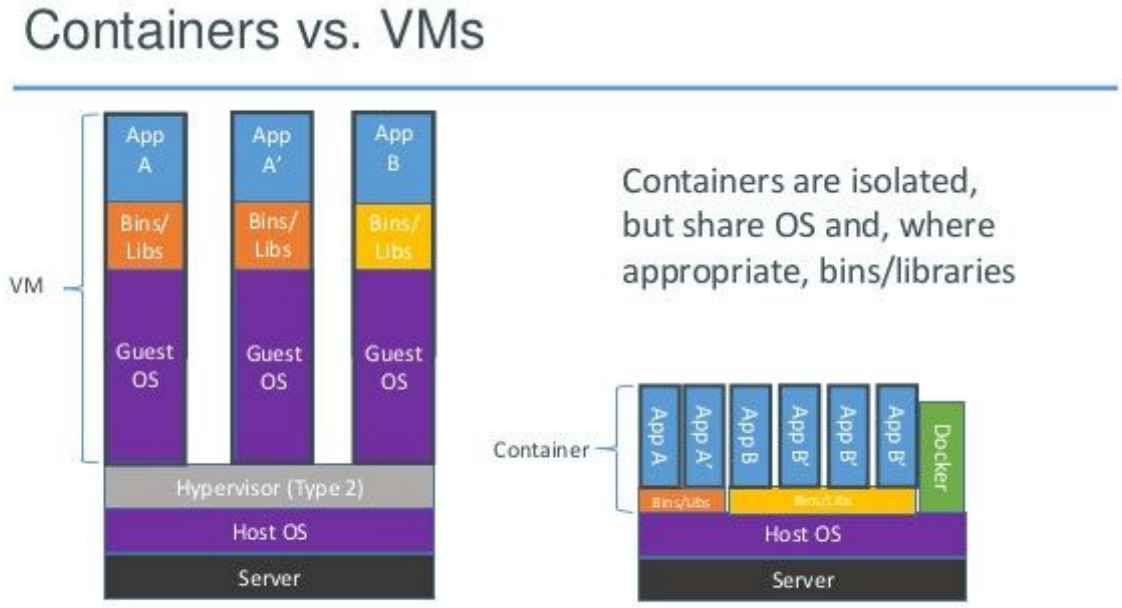
Containers

- It allows us to run a Linux system within another Linux system.
- A container is a group of processes on a Linux box, put together is an isolated environment.

- From the inside it looks like a VM
- From the outside, it looks like normal processes



Containers vs VMs



- Continuous Deployment
- Workload Portability
- Software Quality and Compliance
- Cost Optimization
- Infrastructure Agnostic

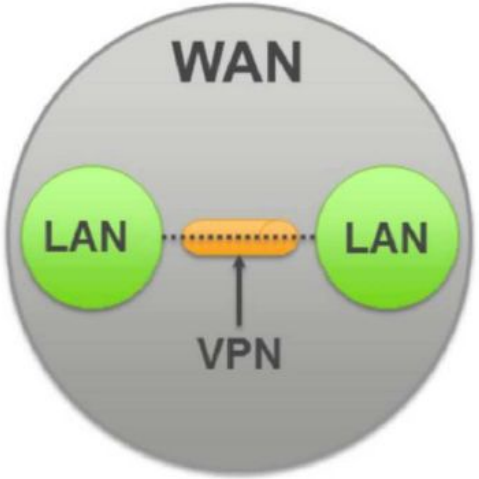
Network

What is a Network?

A network is two or more computers linked to share resources, exchange files, or allow electronic communications.

Network Types:

- Local Area Network (LAN)
- Wide Area Network (WAN)
- Virtual Private Network (VPN)



Network

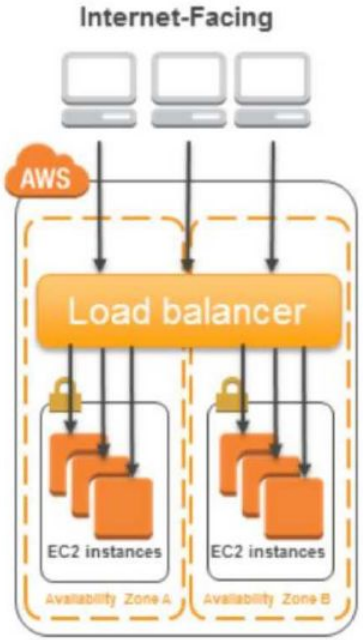
CIDR Address Blocks

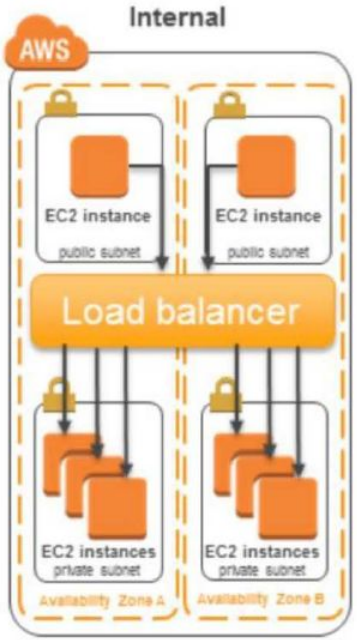
CIDR Prefix	Dotted Decimal Notation	# Node Addresses	# of Traditional Class Networks
/13	255.248.0.0	512K	8 B or 2048 C class
/14	255.252.0.0	256K	4 B or 1024 C class
/15	255.254.0.0	128K	2 B or 512 C class
/16	255.255.0.0	64K	1 B or 256 C class
/17	255.255.128.0	32K	128 C class
/18	255.255.192.0	16K	64 C class
/19	255.255.224.0	8K	32 C class
/20	255.255.240.0	4K	16 C class
/21	255.255.248.0	2K	8 C class
/22	255.255.252.0	1K	4 C class
/23	255.255.254.0	512	2 C class
/24	255.255.255.0	256	1 C class
/25	255.255.255.128	128	1/2 C class
/26	255.255.255.192	64	1/4 C class
/27	255.255.255.224	32	1/8 C class

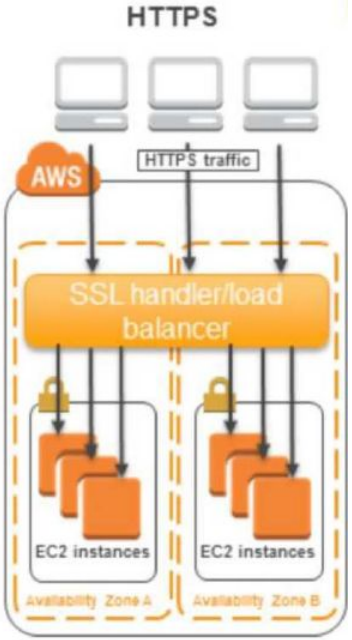
Network
192.168.0.0/23 → 512 addresses

Subnet
192.168.0.0/24 → 256 addresses
192.168.1.0/24 → 256 addresses

Load Balancing



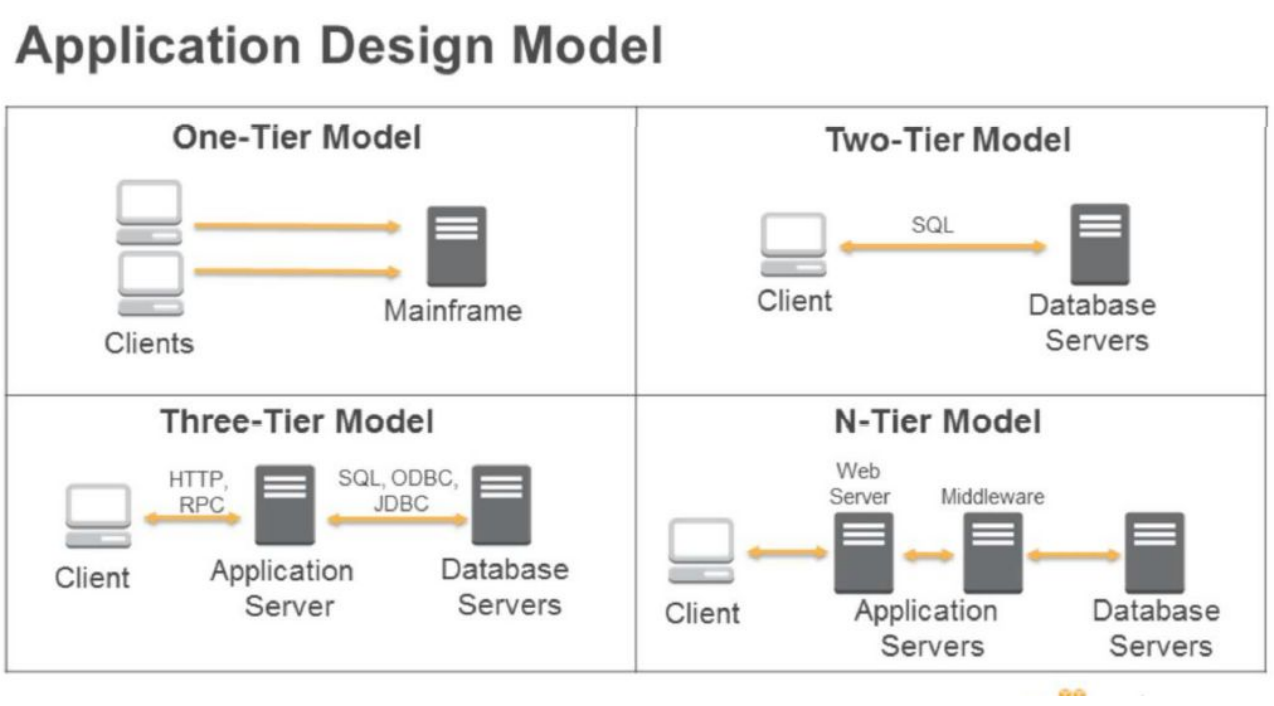




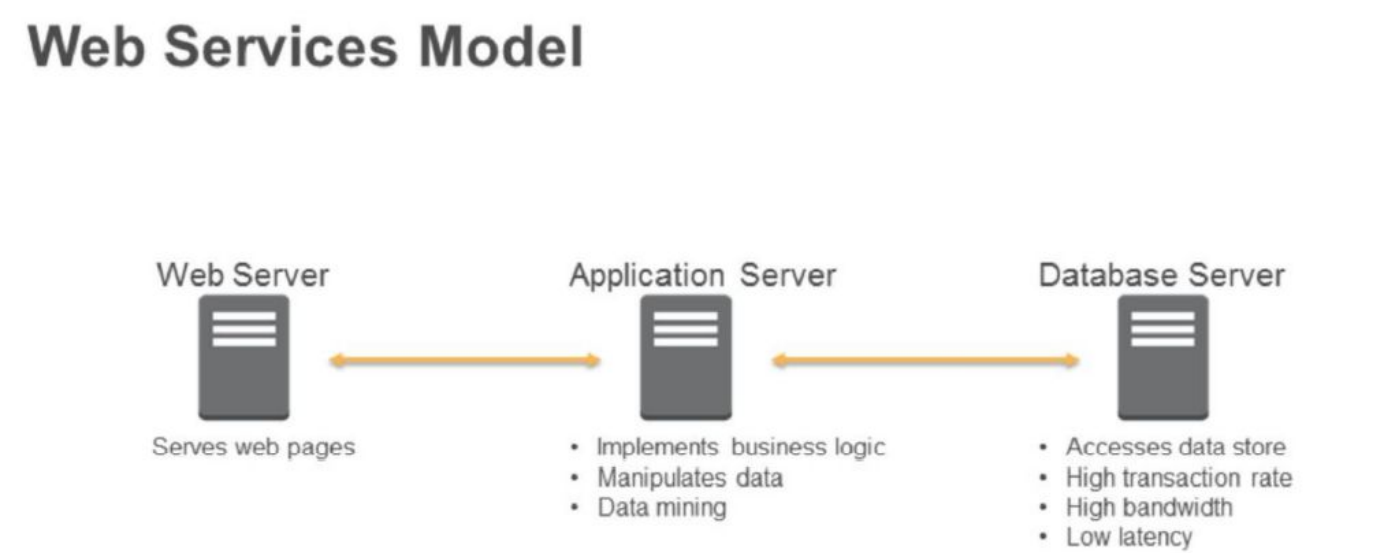
Reminder

Application models

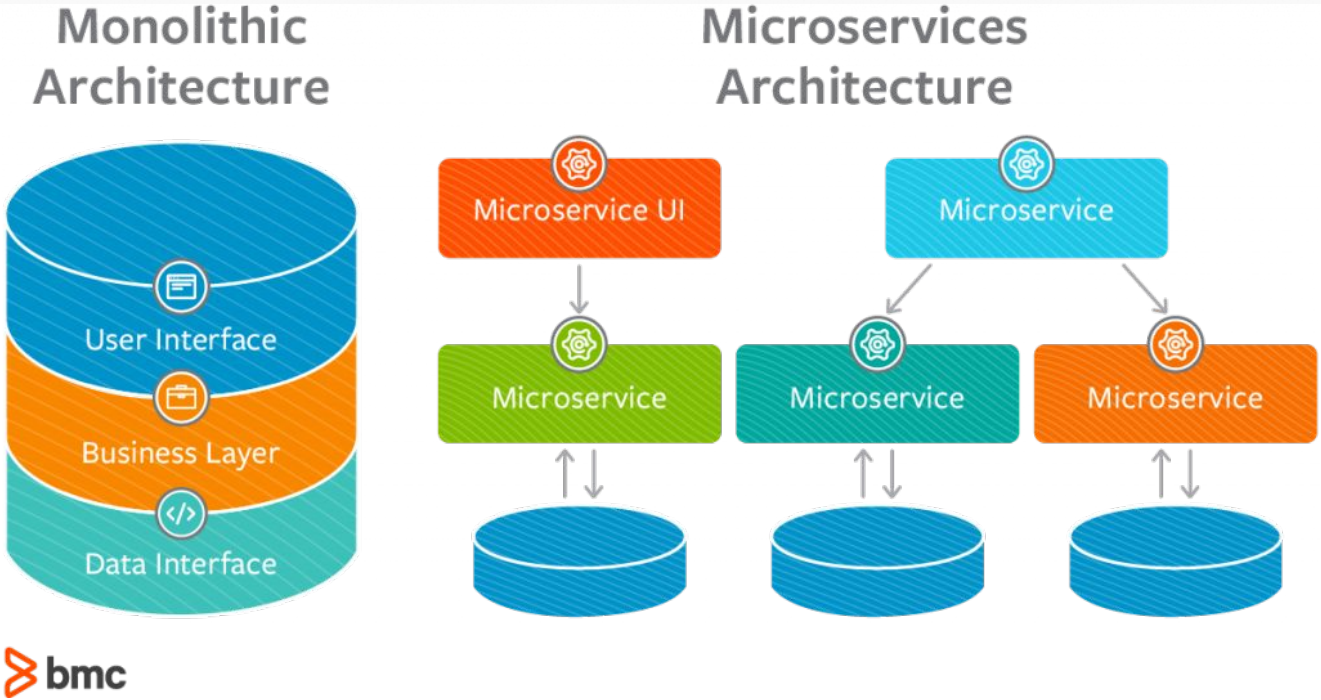
Application models



Application models

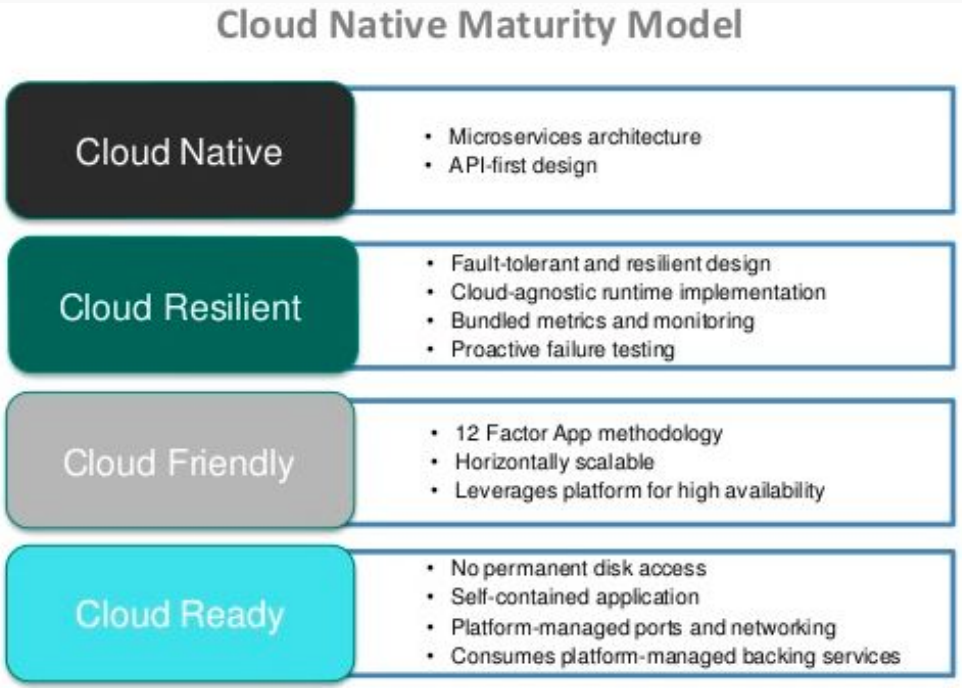


Application models



bmc

Application models



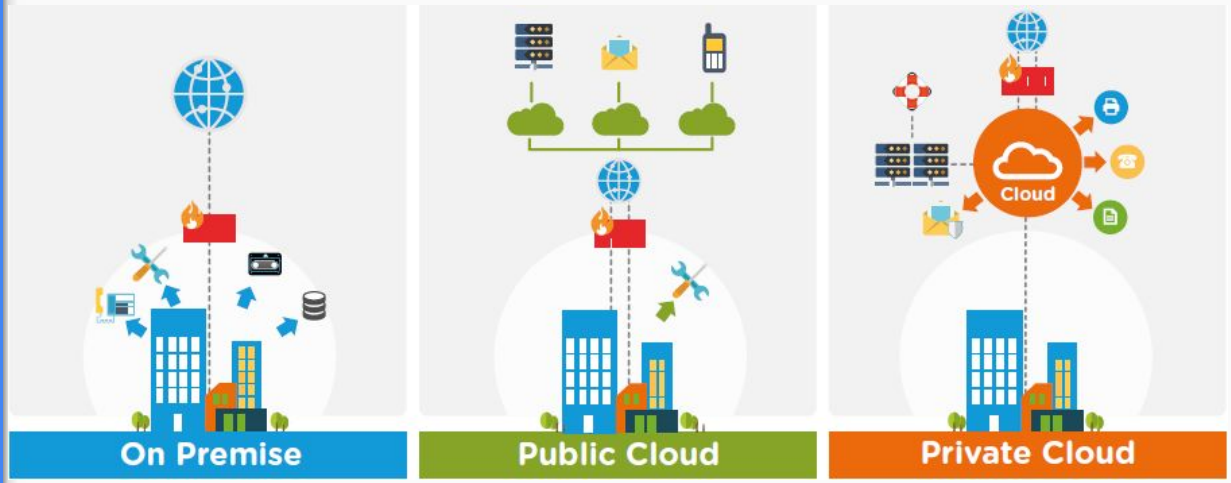
Cloud Computing

Types

Cloud Types

On-premise: Describes software and hardware that is housed and directly managed inside a business' physical location. On-premise hardware and software is usually always on and available.

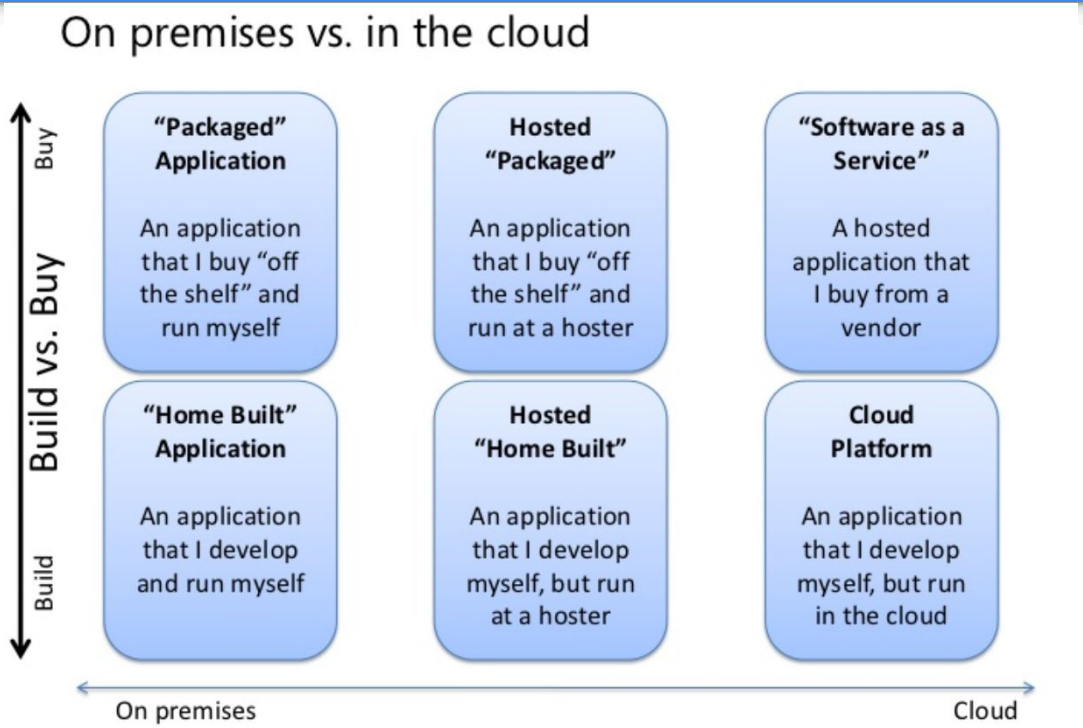
Cloud: In a hosted cloud, the service provider maintenance the systems on their server, accessible by the enterprise at any given time with related processes taken care of by the host-cloud service provider.



Cloud vs On-Prem

 Cloud software is delivered and hosted on the Internet.	 On-Premise software is managed and hosted locally on a device.
No installation, or increased costs for additional equipment or infrastructure.	Large initial cash outlay for equipment and replacement of outdated equipment.
The vendor handles updates and product enhancements and notifies you of changes.	"Feature lock-in"; usually limited to the solution features the product came with at time of purchase.
Accessibility for a broad range of devices through mobile and web applications.	Access generally limited to the device(s) the software was originally installed on.

Types


On premises vs. in the cloud
Build vs. Buy
Buy
Build
"Packaged" Application
An application that I buy "off the shelf" and run myself
Hosted "Packaged"
An application that I buy "off the shelf" and run at a hoster
"Software as a Service"
A hosted application that I buy from a vendor
"Home Built" Application
An application that I develop and run myself
Hosted "Home Built"
An application that I develop myself, but run at a hoster
Cloud Platform
An application that I develop myself, but run in the cloud
On premises
Cloud

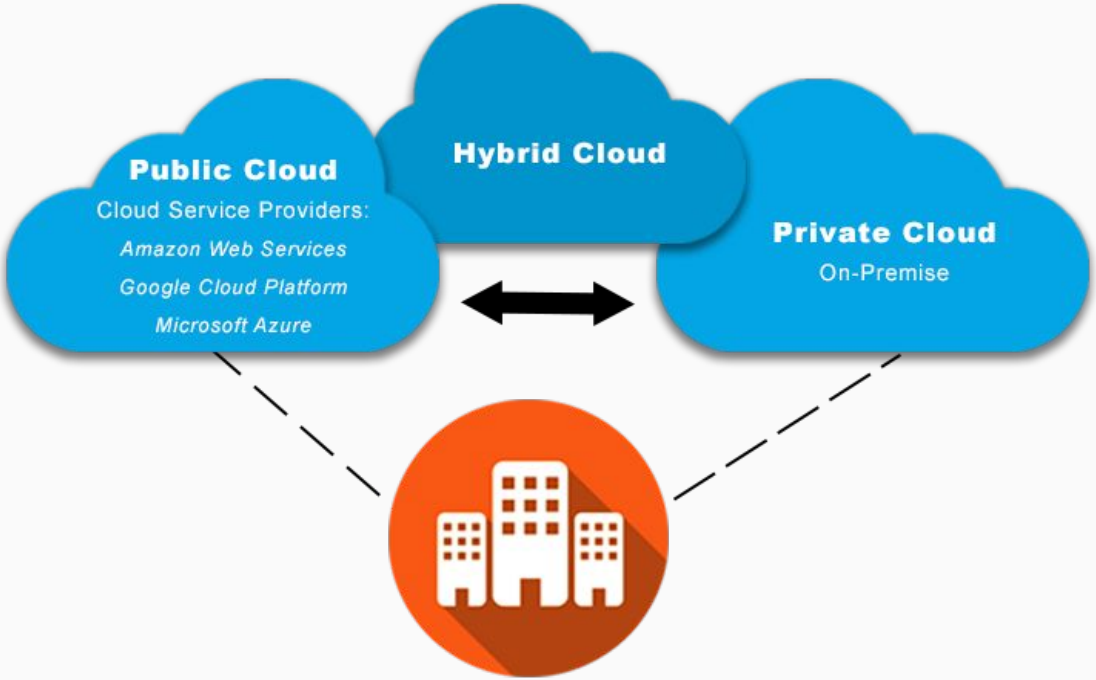
Hybrid Cloud

Hybrid cloud benefits include the following management tools:

Choice: The additional choice of multiple environments provides flexibility and the ability to avoid vendor lock-in.

Disaster Avoidance: Multiple environments ensures compute resources are always available to avoid downtime due to natural disaster or human error.

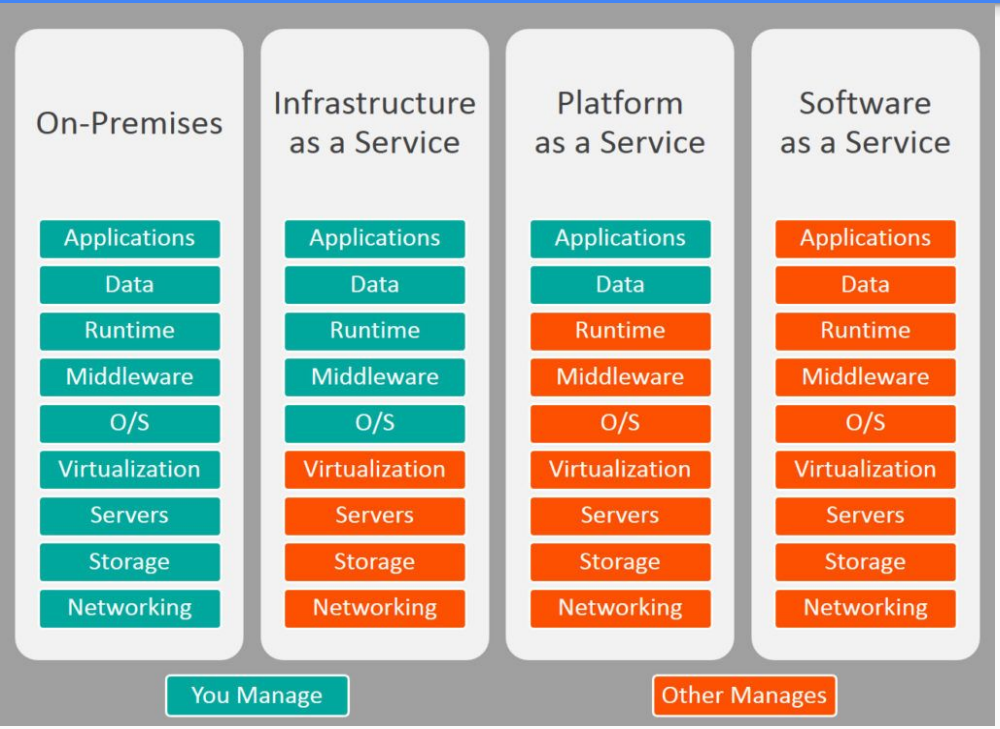
Compliance: Many hybrid cloud environments can help enterprises achieve their goals for governance, risk management and compliance regulations.



Cloud Computing

Models

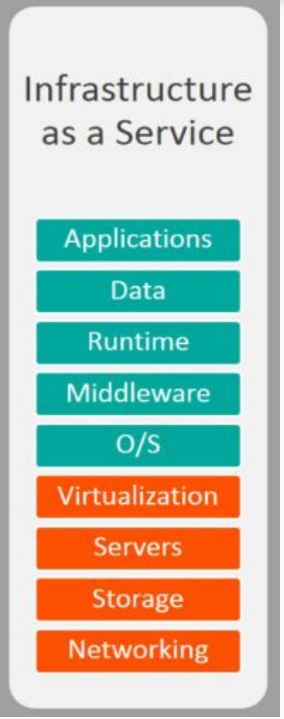
Cloud computing models



Cloud models come in three types, SaaS , IaaS and PaaS.

Each of the cloud models has their own set of benefits that could serve the needs of various businesses

Infrastructure as a Service

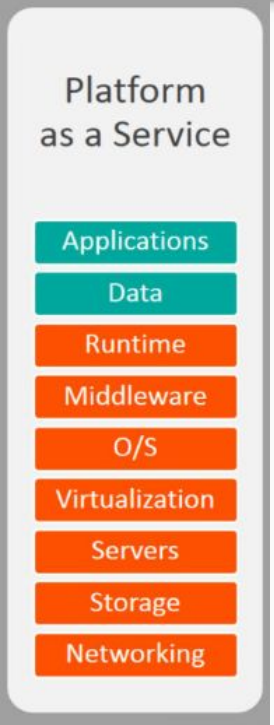


IaaS emulates the hardware that companies traditionally purchased to create internal and external networks.

The chore of installing hardware and connecting it to physical networks has been replaced with the task of creating robust configurations. Almost every part of a traditional network is offered as a cloud service.

Examples: Servers, Firewalls, Routers

Platform as a Service

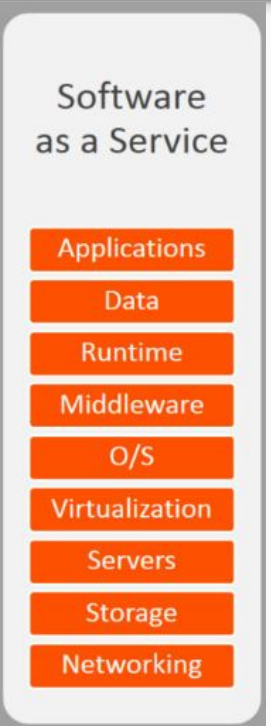


PaaS products sit in the middle between IaaS and SaaS.

PaaS services allow customers to create a customized experience on top of an existing set of tools.

Examples: Google App Engine, Beanstalk, RDS

Software as a Service

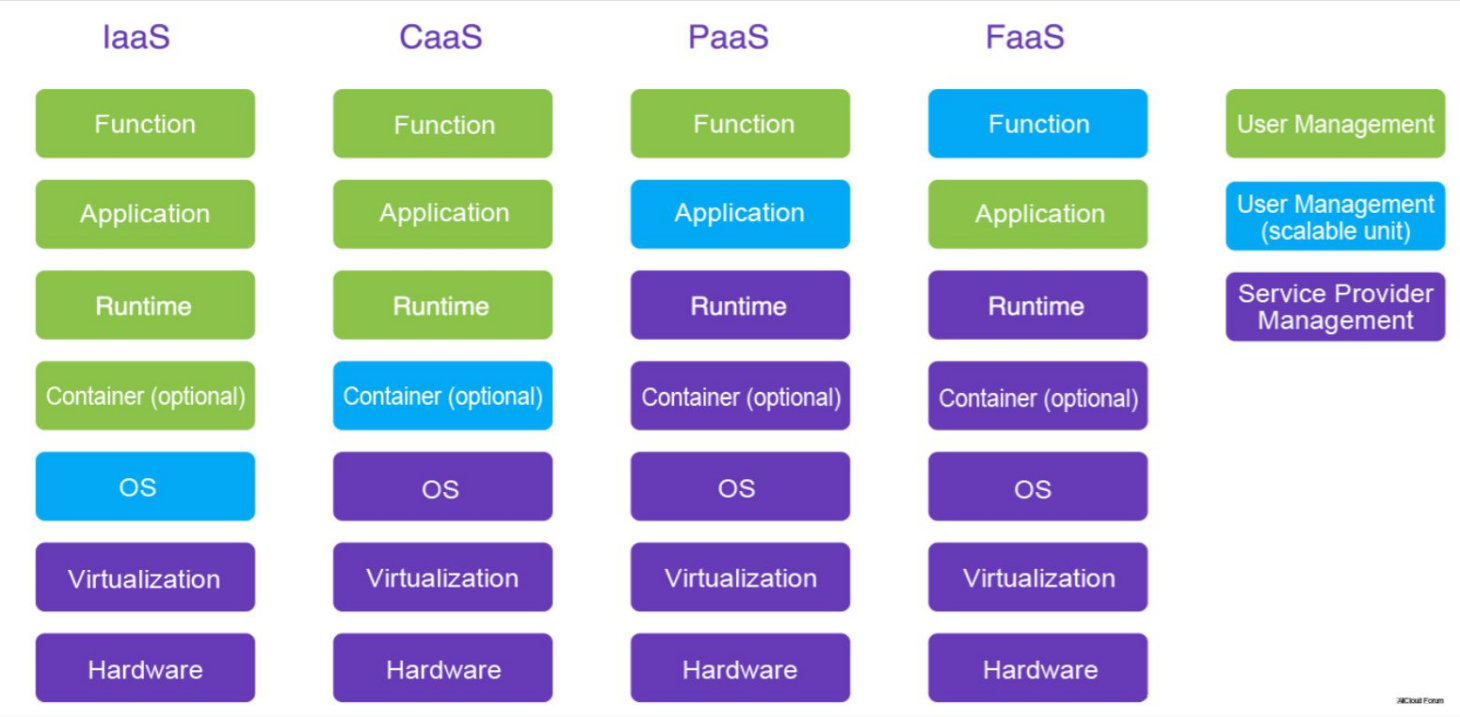


Software as a Service describes any Internet-based product that provides a specific set of operations. This includes everything from email to contact management.

Interacting with a SaaS product had traditionally been done through a web browser.

Examples: Mail services, online storage (Dropbox)

Advanced Cloud models



Container as a Service

CaaS (also called Container-As-A-Service) is a business model in which the cloud service provider offers container based virtualization services as an online service that is scalable.

CaaS enables the users to use the container services without the need of having the required infrastructure.

Function as a Service

Function as a service (FaaS) is a cloud computing model that enables users to develop applications and deploy functionalities without maintaining a server, increasing process efficiency.

This is typically utilized when creating microservices such as web applications, data processors, chatbots and IT automation.

Examples: Web apps, Backends, Data/stream processing, Chatbots, Scheduled tasks, IT Automation

Cloud Providers

Introduction

Introduction

A cloud service provider, or CSP, is a company that offers some component of cloud computing -- typically infrastructure as a service (IaaS), software as a service (SaaS) or platform as a service (PaaS) -- to other businesses or individuals.

Some cloud service providers have differentiated themselves by tailoring their offerings to a vertical market requirements.

Major cloud providers

Azure

Great for enterprises that already use a lot of Microsoft software

AWS

The most mature with the widest array of products and services

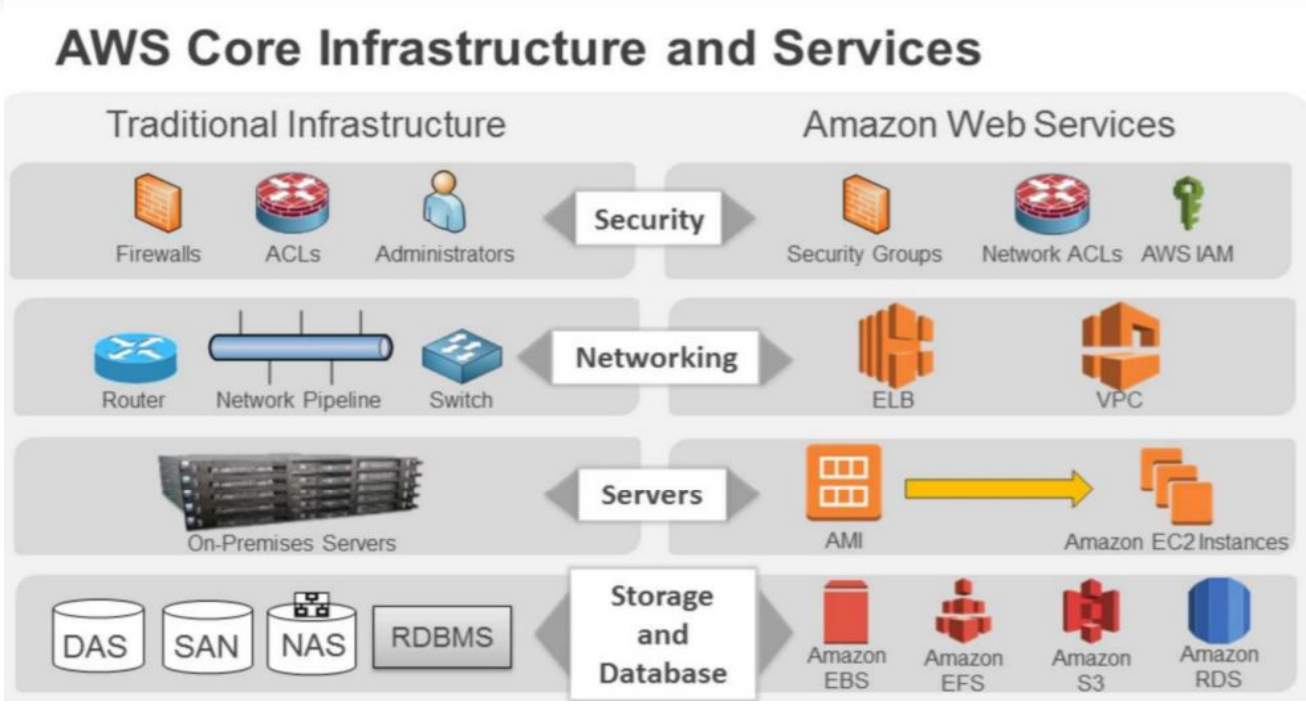
Google Cloud

Generally the cheapest of the three

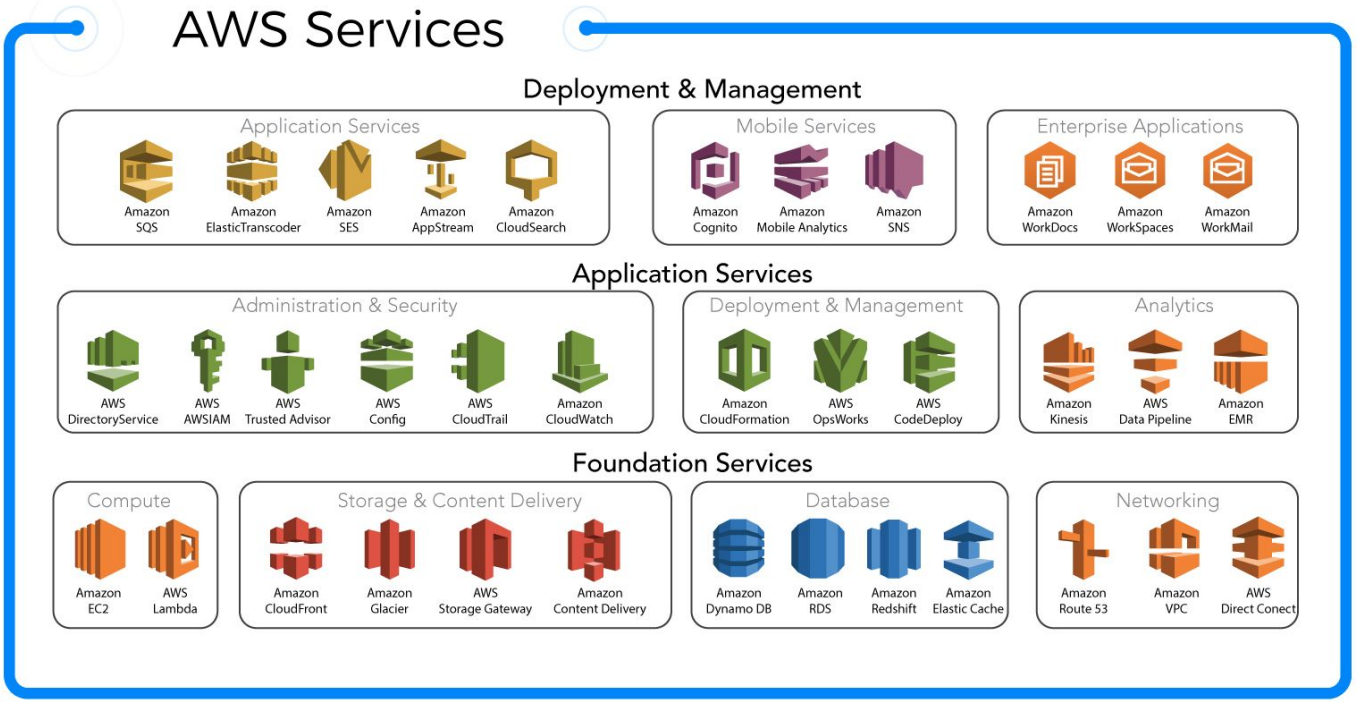
Differents names, same services

Category	Service	amazon web services	Azure	Google Cloud Platform
Compute	**Shared Web hosting**		Azure shared App Services	
Compute	**Virtual Server**	Amazon EC2	Azure Virtual Machine	Compute Engine
Compute	**Bare Metal Server**	Amazon EC2 Bare Metal Instance (Preview)	Azure Bare Metal Servers (Large Instance Only for SAP Hana)	
Compute	**Virtual Dedicated Host**	Amazon EC2 Dedicated Hosts		Sole Tenant Node (Beta)
Compute	**Container Registration Service**	Amazon EC2 Container Registry	Azure Container Registry	Container Registry
Compute	**Container Management Service**	Amazon EC2 Container Service Amazon Elastic Container Service for Kubernetes (EKS)	Azure Kubernetes Service (AKS) Azure Container Instances	Kubernetes Engine

On-premise and AWS



Main services



Much more services

Cloud Providers

Regions & Availability Zones

Infrastructure

AWS Global Infrastructure

Regions

- Geographic locations
- Consist of **at least two** Availability Zones

Availability Zones

- Clusters of data centers
- **Isolated from failures** in other Availability Zones

Infrastructure

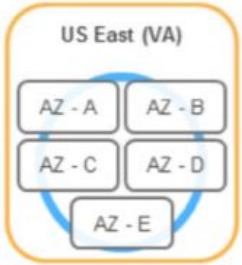
Regions & AZs

AWS Global Infrastructure

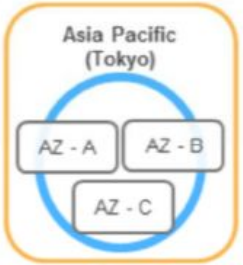
At least 2 Availability Zones per region.

Examples:

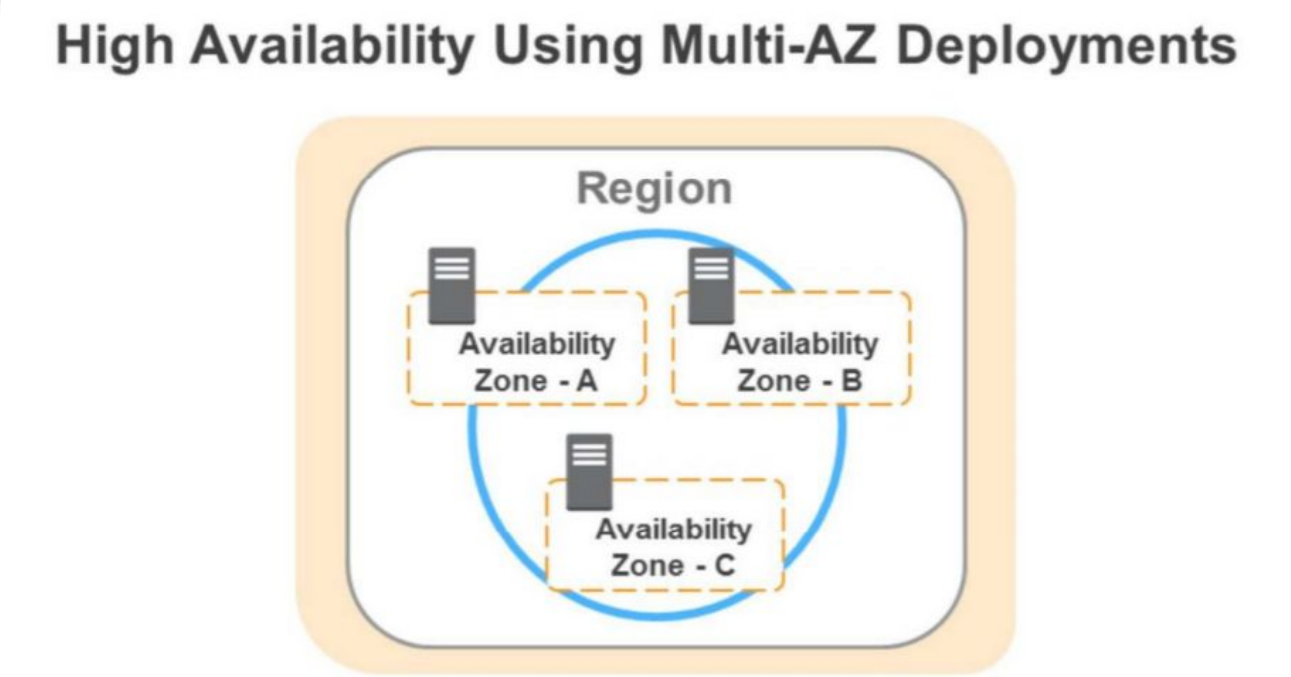
- US East (N. Virginia)
  - us-east-1a
  - us-east-1b
  - us-east-1c
  - us-east-1d
  - us-east-1e



- Asia Pacific (Tokyo)
  - ap-northeast-1a
  - ap-northeast-1b
  - ap-northeast-1c



High Availability



Cloud Providers

Servers

Instances

Amazon Elastic Compute Cloud (EC2)

Amazon EC2

- **Resizable** compute capacity
- Complete control of your computing resources
- **Reduced time required** to obtain and boot new server instances

Instances

Amazon EC2 Facts

- **Scale capacity** as your computing requirements change
- Pay only for capacity that you actually use
- Choose **Linux** or **Windows**
- Deploy across **AWS Regions** and **Availability Zones** for reliability
- Use **tags** to help manage your Amazon EC2 resources

Instances

Launching an Amazon EC2 Instance via the Management Console

1. **Determine the AWS Region** in which you want to launch the Amazon EC2 instance.
2. **Launch** an Amazon EC2 instance from a pre-configured Amazon Machine Image (AMI).
3. **Choose an instance type** based on CPU, memory, storage, and network requirements.
4. **Configure** network, IP address, security groups, storage volume, tags, and key pair.

Images

Amazon Machine Image (AMI) Details

An AMI includes the following:

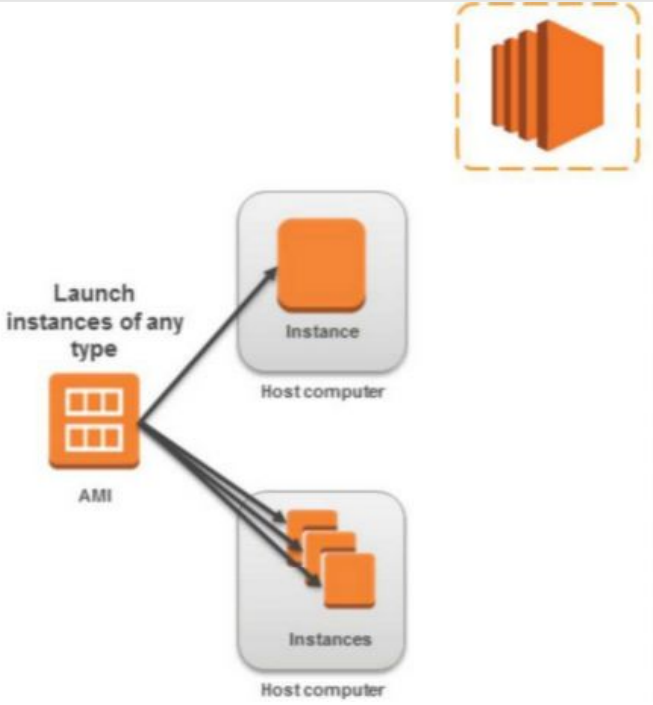
- A template for the **root volume** for the instance (for example, an operating system, an application server, and applications).
- **Launch permissions** that control which AWS accounts can use the AMI to launch instances.
- A block device mapping that specifies the **volumes to attach** to the instance when it is launched.

Images

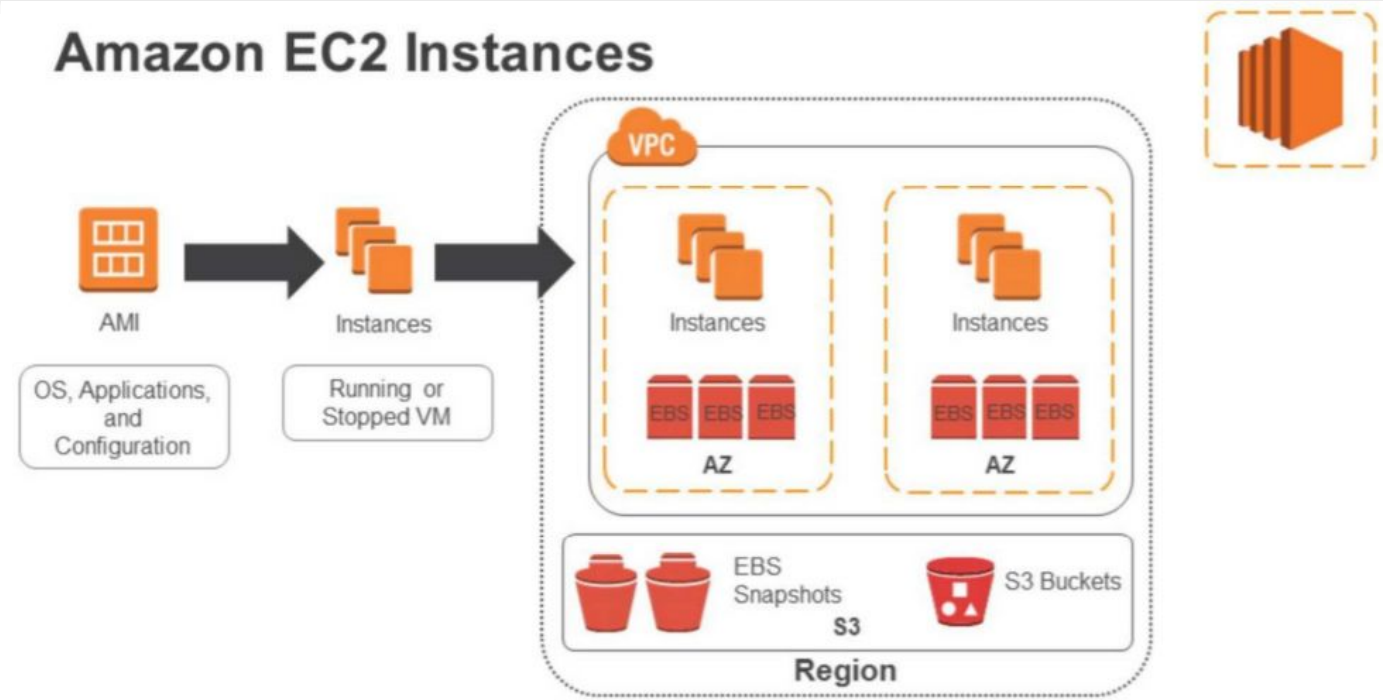
Instances and AMIs

Select an AMI based on:

- Region
- Operating system
- Architecture (32-bit or 64-bit)
- Launch permissions
- Storage for the root device



Deployment


Amazon EC2 Instances
AMI
OS, Applications, and Configuration
Instances
Running or Stopped VM
VPC
Instances
EBS
EBS
EBS
AZ
Instances
EBS
EBS
EBS
AZ
EBS Snapshots
S3
S3 Buckets
Region

Instance types

Current Generation Instances

Instance Family	Some Use Cases
General purpose (t2, m4, m3)	• Low-traffic websites and web applications • Small databases and mid-size databases
Compute-optimized (c4, c3)	• High performance front-end fleets • Video-encoding
Memory-optimized (r3)	• High performance databases • Distributed memory caches
Storage-optimized (i2, d2)	• Data warehousing • Log or data-processing applications
GPU instances (g2)	• 3D application streaming • Machine learning

User Data

Instance User Data

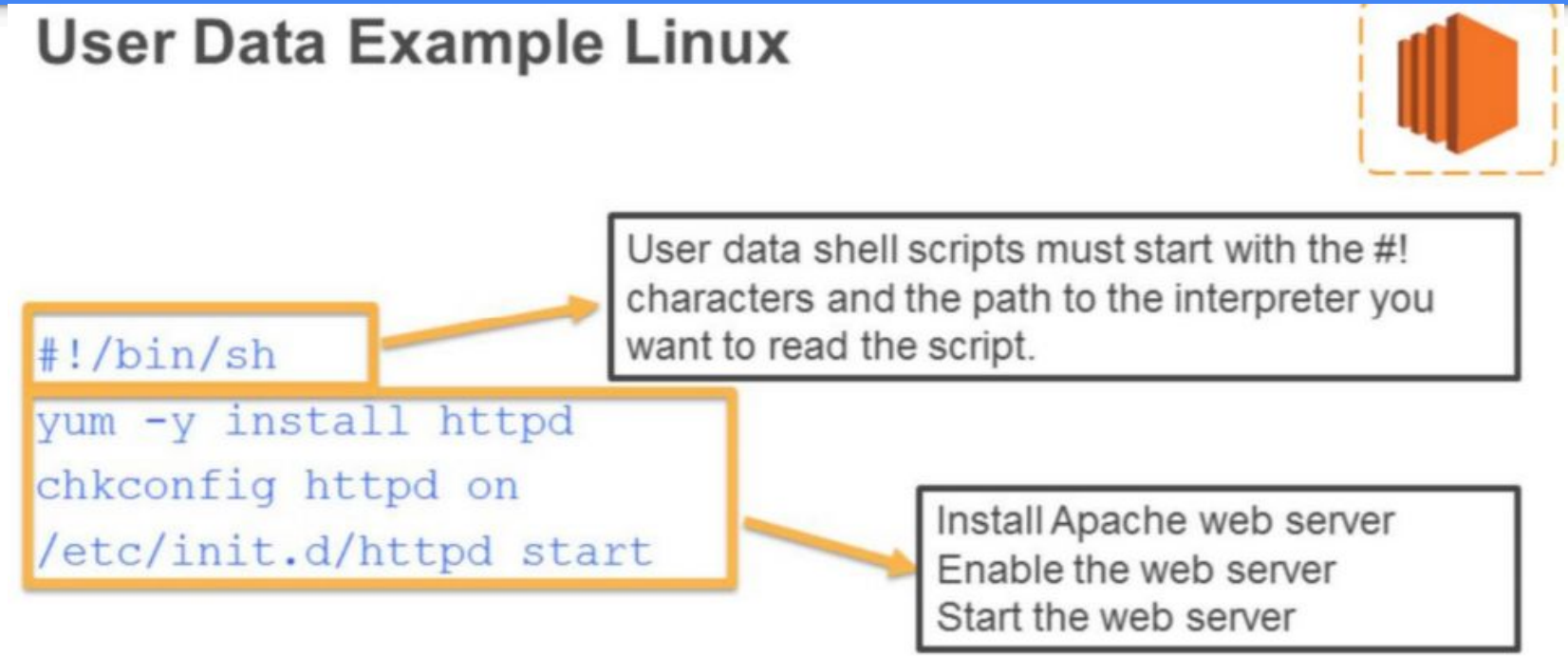
- Can be passed to the instance **at launch**.
- Can be used to perform common **automated configuration tasks**.
- Runs scripts after the instance starts.

User Data

Adding User Data

- You can specify user data when launching an instance.
- User data can be:
  - Linux script – executed by **cloud-init**
  - Windows batch or PowerShell scripts – executed by **EC2Config** service
- User data scripts run once per instance ID by default.

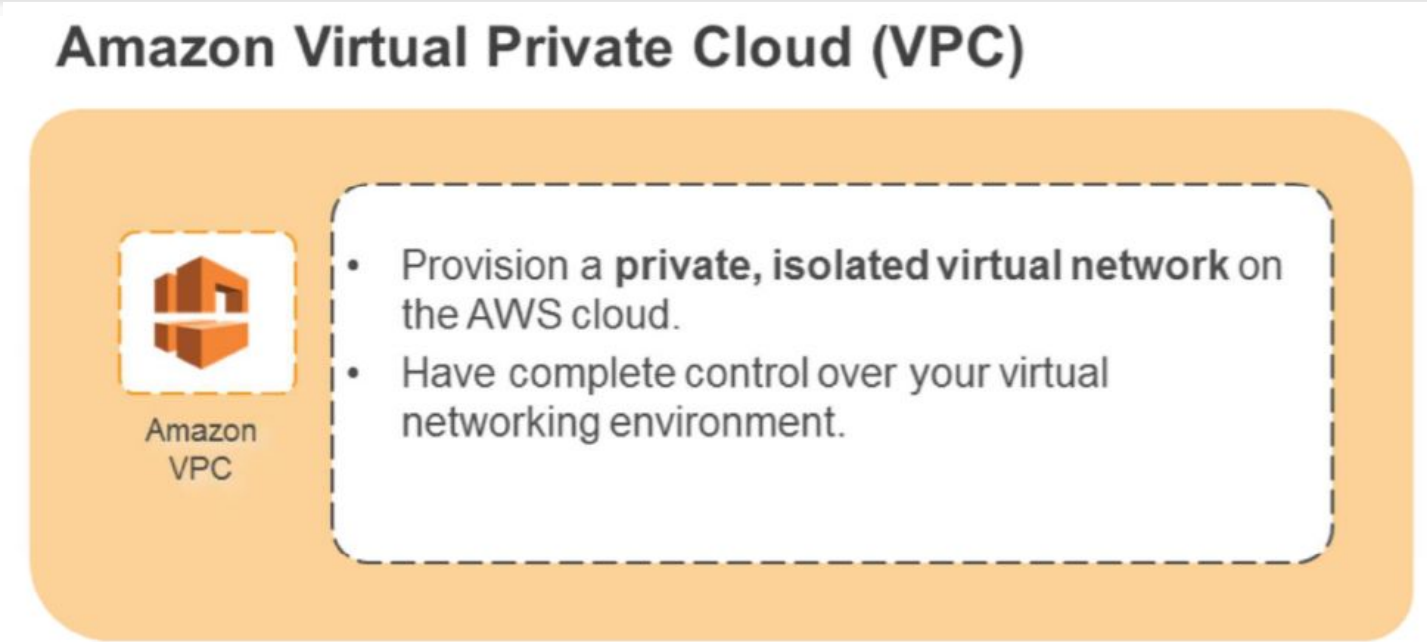
User Data


User Data Example Linux
#!/bin/sh
yum -y install httpd
chkconfig httpd on
/etc/init.d/httpd start
User data shell scripts must start with the #! characters and the path to the interpreter you want to read the script.
Install Apache web server
Enable the web server
Start the web server

Cloud Providers

Network

Virtual Private Cloud

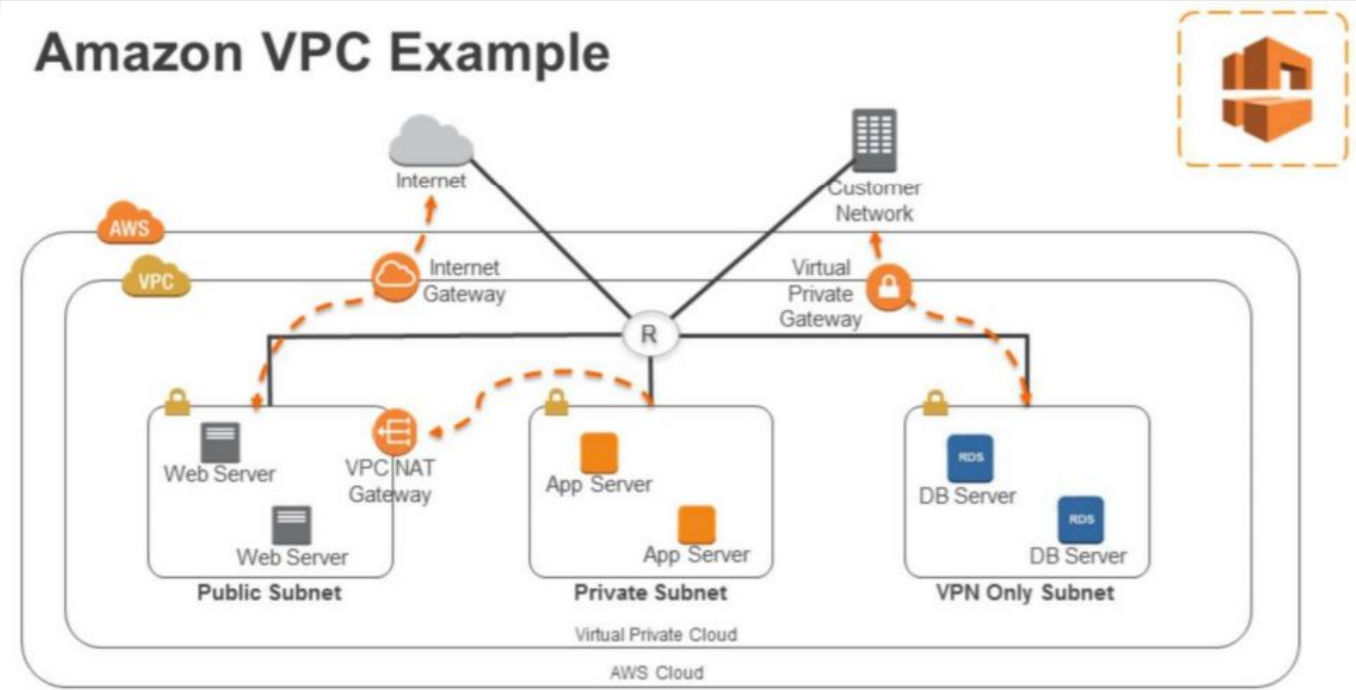


VPC & Subnets

VPCs and Subnets

- A **subnet** defines a range of IP addresses in your VPC.
- You can launch AWS resources into a subnet that you select.
- A **private subnet** should be used for resources that won't be accessible over the Internet.
- A **public subnet** should be used for resources that will be accessed over the Internet.
- Each subnet must reside entirely within one Availability Zone and cannot span zones.

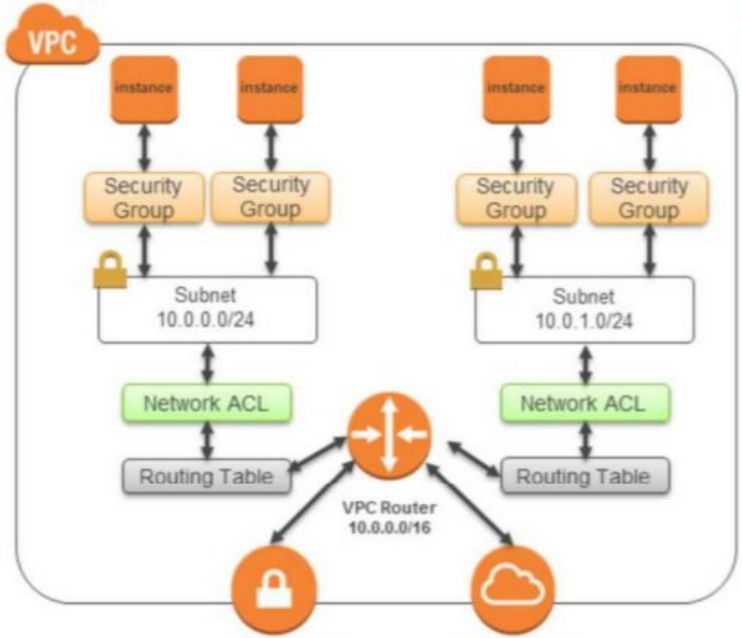
VPC example


Amazon VPC Example
Internet
Customer Network
AWS
VPC
Internet Gateway
Virtual Private Gateway
R
Web Server
VPC NAT Gateway
Web Server
Public Subnet
App Server
App Server
Private Subnet
RDS
DB Server
RDS
DB Server
VPN Only Subnet
Virtual Private Cloud
AWS Cloud

Security

Security in Your VPC

- Security groups
- Network access control lists (ACLs)
- Key Pairs



Security groups

Security Groups

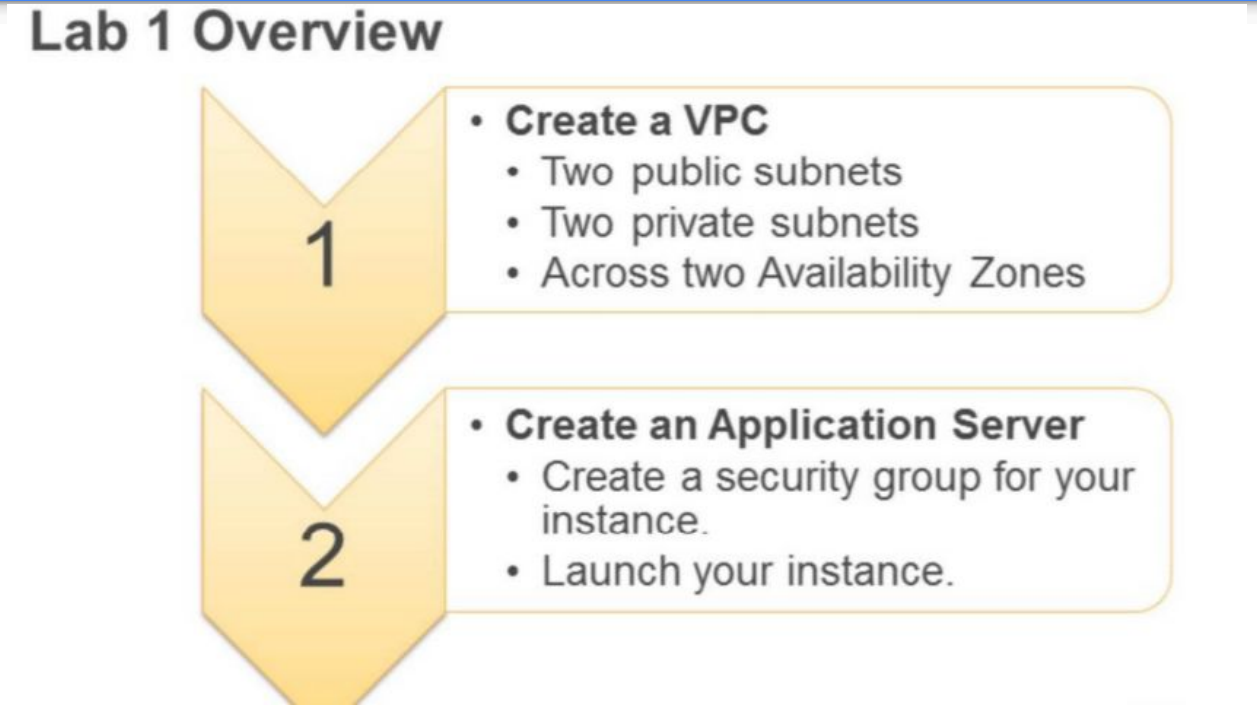
Allow access to IP address ranges or Amazon EC2 instances you specify.

Use VPC security groups to control access to a DB instance inside a VPC.

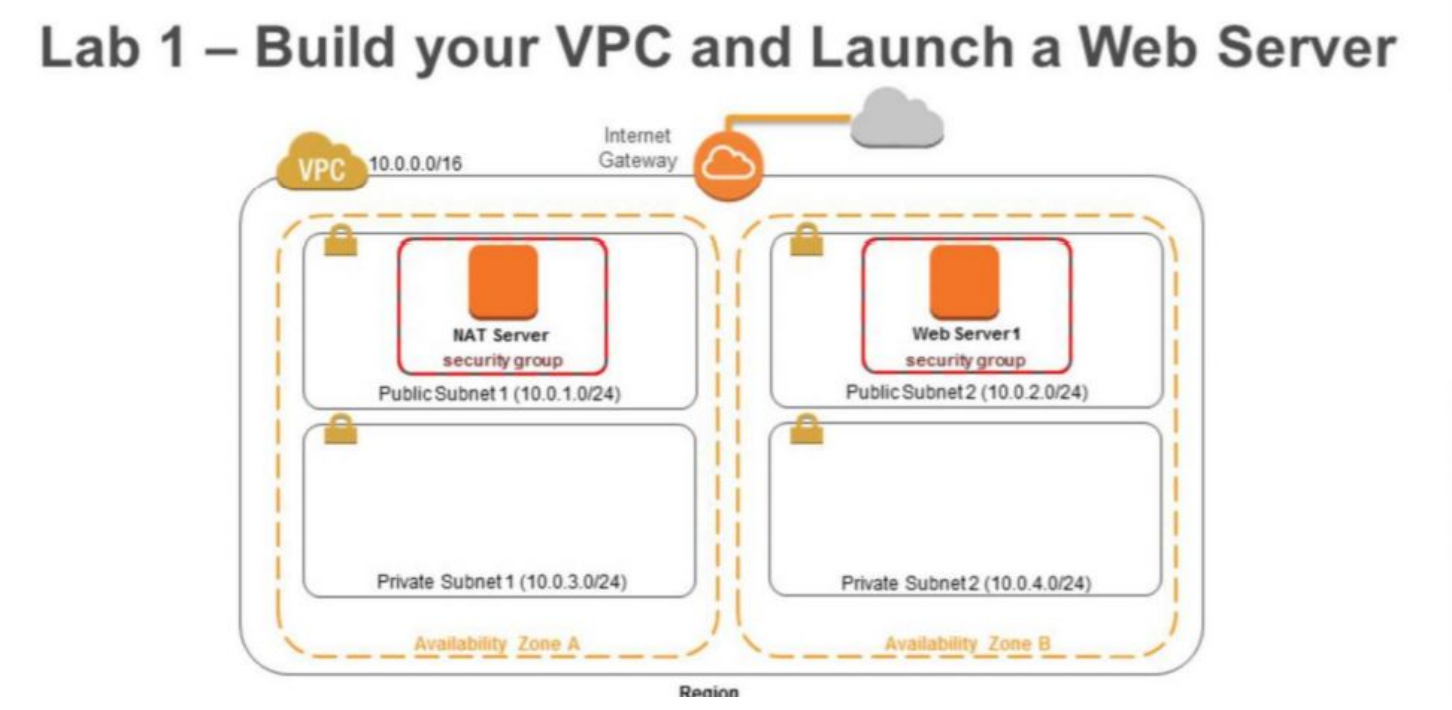
Lab 1: Build VPC & start Web server

Amazon Web Services

Lab 1: Overview



Lab 1: Architecture



Step-by-step

1. Create four subnets: two in us-east-1a, two in in us-east-1b.
2. Create an Internet Gateway and attach it to the VPC.
3. Create a route table; with a route 0.0.0.0/0 —> IGW, and attach it to the public subnets.
4. Deploy a Cloud9 environment in the public subnet.
5. Generate a key pair using Cloud9 terminal and the “ssh-keygen” Linux command.
6. Import the key into AWS via the cloud9 terminal and the following command: *aws ec2 import-key-pair --key-name "ec2-key" --public-key-material file://~/.ssh/id_rsa.pub*
7. Create an EC2 Instance with the following parameters: Ubuntu 18.04 64 bits, t2.micro, 8GB disk size, public-subnet, auto-assign public ip: enable, and choose the “ec2-key” key pair.
8. Edit the associated security group to add HTTP from anywhere in inbound rules.
9. Connect to the instance via Cloud9 terminal: “ssh ubuntu@IP”
10. Try to update your package repositories with the command: sudo apt-get update

https://github.com/skhedim/demo-aws

Storage

Amazon Web Services

Bucket

Amazon Simple Storage Service (S3)

Amazon S3

- Storage for the Internet
- Natively online, HTTP access
- Storage that allows you to store and retrieve **any amount of data**, any time, from anywhere on the web
- **Highly scalable**, reliable, fast and durable

Bucket

Amazon S3 Facts

- Can store an **unlimited number of objects** in a bucket
- Objects can be **up to 5 TB**; no bucket size limit
- Designed for **99.999999999%** durability and **99.99%** availability of objects over a given year
- Can use **HTTP/S** endpoints to store and retrieve any amount of data, at any time, from anywhere on the web
- Is highly scalable, reliable, fast, and inexpensive
- Can use optional server-side **encryption** using AWS or customer-managed provided client-side encryption
- Auditing is provided by access logs
- Provides standards-based **REST** and SOAP interfaces

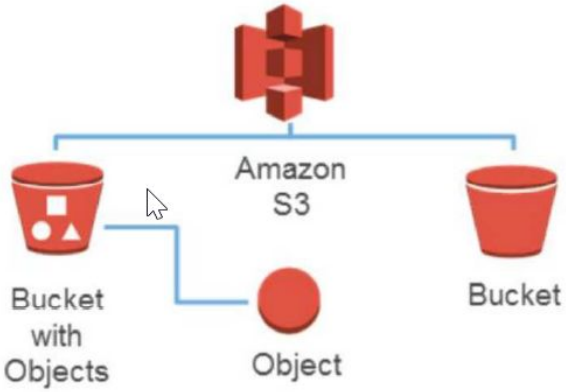
Bucket

Common Use Scenarios

- Storage and backup
- Application file hosting
- Media hosting
- Software delivery
- Store AMIs and snapshots

Bucket

Amazon S3 Concepts



- Amazon S3 stores data as objects within **buckets**
- An object is composed of a file and optionally any **metadata** that describes that file
- You can have **up to 100 buckets** in each account
- You can **control access** to the bucket and its objects

Bucket versioning

Amazon S3 Versioning

- Protects from **accidental overwrites and deletes** with no performance penalty.
- Generates a **new version with every upload**.
- Allows easily retrieval of deleted objects or **roll back** to previous versions.
- Three states of an Amazon S3 bucket
  - Un-versioned (default)
  - Versioning-enabled
  - Versioning-suspended

Versioning Enabled

Volumes

Amazon Elastic Block Store (EBS)

Amazon
EBS

- **Persistent block level storage** volumes offer consistent and low-latency performance.
- Stored data is automatically replicated within its Availability Zone.
- Snapshots are stored durably in Amazon S3.

Volumes types

Amazon EBS Volume Types

- SSD-backed volumes are
  - Optimized for **transactional** workloads that involve **frequent read/write** operations with **small I/O** size.
  - Dominant in **IOPS** performance.
- HDD-backed volumes are
  - Optimized for **large streaming** workloads.
  - Dominant in **throughput** (measured in MiB/s).

Volumes

Amazon EBS Facts

- EBS is recommended when data must be **quickly accessible** and requires **long-term persistence**.
- You can launch your EBS volumes as **encrypted** volumes – data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted.
- You can create **point-in-time snapshots** of EBS volumes, which are persisted to Amazon S3.

Volumes

Amazon EBS Use Cases

- **OS:** Use for boot/root volume, secondary volumes
- **Databases:** Scales with your performance needs
- **Enterprise applications:** Provides reliable block storage to run mission-critical applications
- **Business continuity:** Minimize data loss and recovery time by regularly backing up using EBS Snapshots
- **Applications:** Install and persist any application

Database

Amazon Web Services

Database

Amazon Relational Database Service (RDS)

Amazon RDS

- Cost-efficient and **resizable capacity**
- Manages time-consuming **database administration** tasks
- Access to the full capabilities of **Amazon Aurora**, **MySQL**, **MariaDB**, **Microsoft SQL Server**, **Oracle**, and **PostgreSQL** databases

DB types

Database Considerations

If You Need	Consider Using
A relational database service with minimal administration	**Amazon RDS** • Choice of Amazon Aurora, MySQL, MariaDB, Microsoft SQL Server, Oracle, or PostgreSQL database engines • Scale compute and storage • Multi-AZ availability 
A fast, highly scalable NoSQL database service	**Amazon DynamoDB** • Extremely fast performance • Seamless scalability and reliability • Low cost 
A database you can manage on your own	Your choice of **AMIs** on Amazon EC2 and Amazon EBS that provide scale compute and storage, complete control over instances, and more. 

RDS

Amazon RDS

- Simple and **fast to deploy**
- Manages common database administrative tasks
- **Compatible** with your applications
- Fast, predictable performance
- Simple and **fast to scale**
- Secure
- Cost-effective

MySQL
PostgreSQL
SQL Server
MariaDB
Amazon Aurora
ORACLE

Snapshots

Cross-Region Snapshots

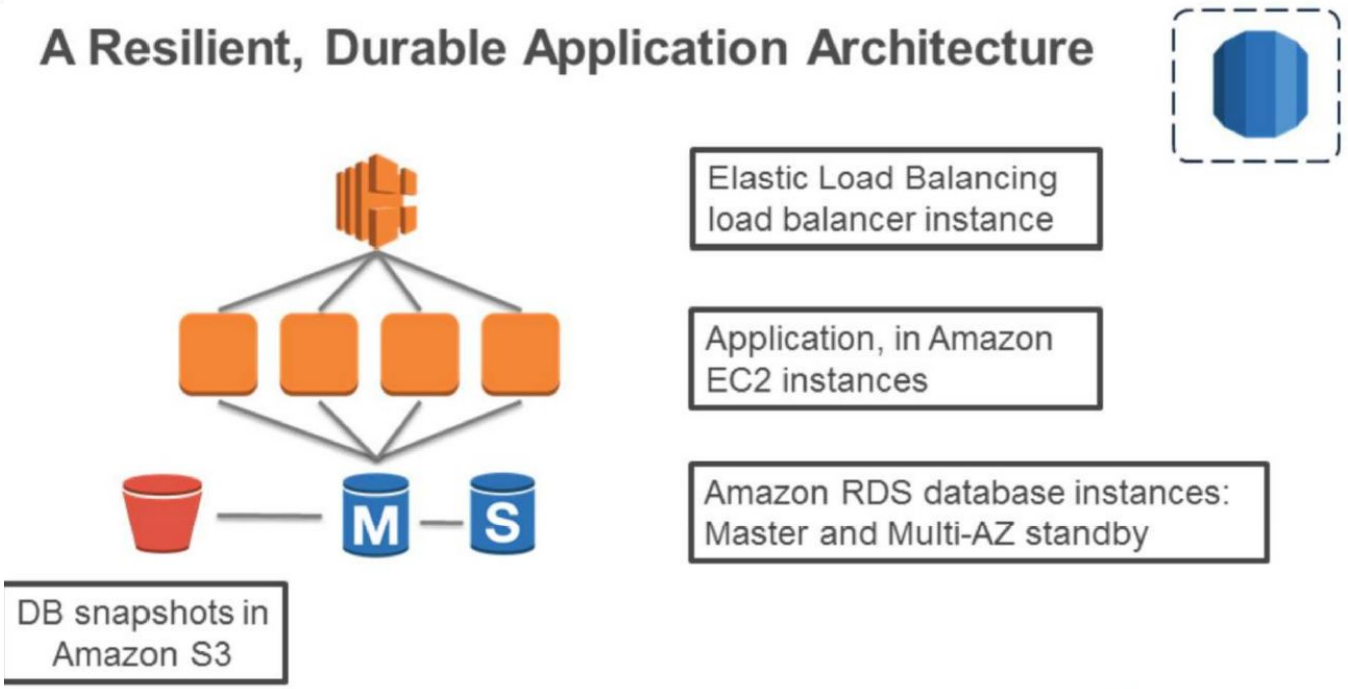
- Are a **copy** of a **database** snapshot stored in a **different AWS Region**.
- Provide a backup for disaster **recovery**.
- Can be used as a **base** for **migration** to a different region.

Multi-zones

Multi-AZ RDS Deployment

- With **Multi-AZ** operation, your database is **synchronously replicated to another Availability Zone** in the same AWS Region.
- **Failover** to the standby **automatically** occurs in case of master database failure.
- Planned maintenance is applied first to standby databases.

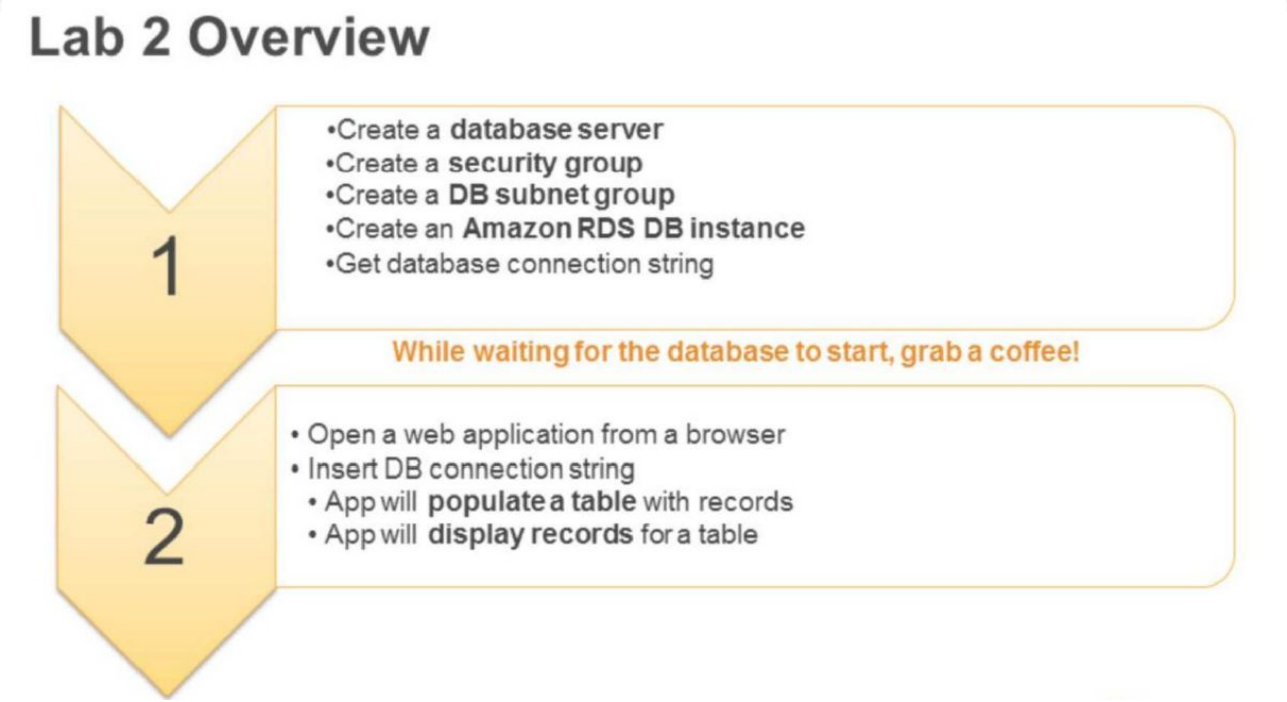
Resilient application



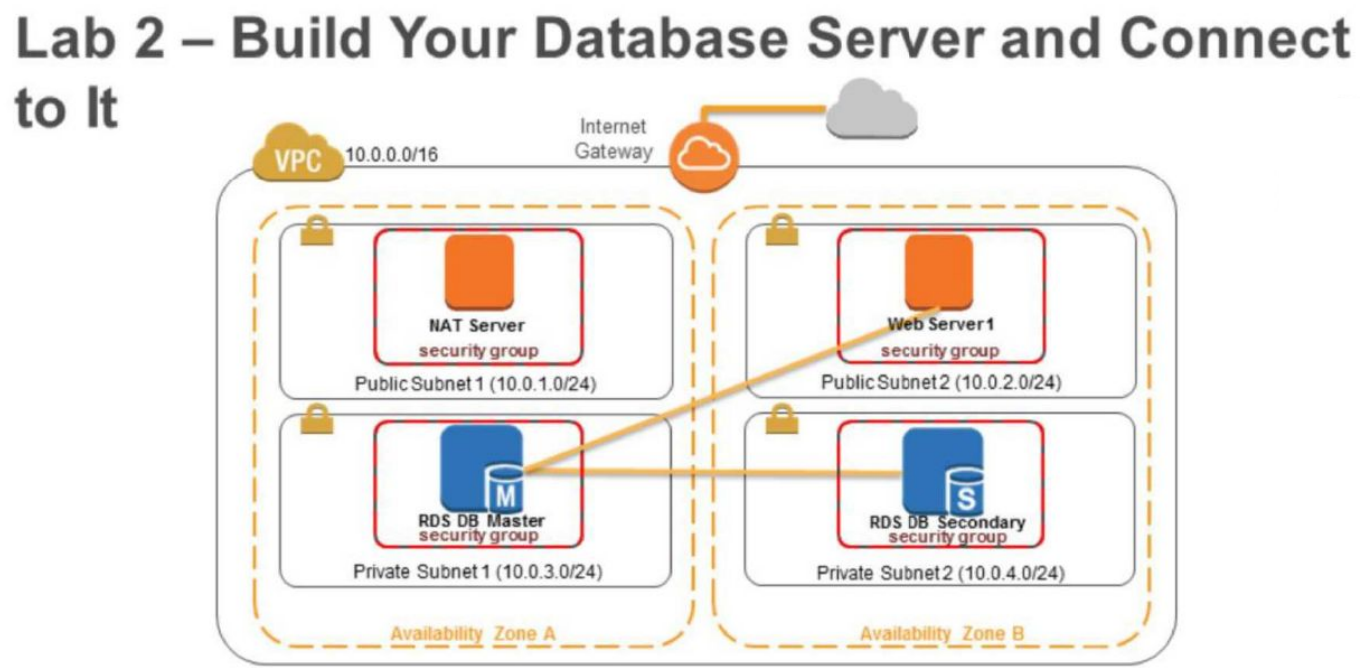
Lab 2: Build DB & configureserver

Amazon Web Services

Lab 2: Overview



Lab 2: Architecture



Lab 2: Step-by-step

1. Create a DB subnet group with the 2 private subnets
2. Create a SG to authorize the web server to access the RDS instance (3306 port)
3. Create a RDS instance with MySQL engine, Free tier option, choose your previously created subnet group and SG
4. Connect to the web server via Cloud9 terminal
5. Apply the wordpress script at this address https://github.com/skhedim/demo-aws
6. Connect to the RDS instance via the web server instance using mysql CLI: mysql -u admin -p -h YOUR-RDS-endpoint
7. Create a database for the website: CREATE DATABASE wordpress; then quit the terminal: QUIT;
8. Configure the database connection from your web browser via the public IP of your web server instance

Elasticity

Amazon Web Services

Load Balancing

Elastic Load Balancing

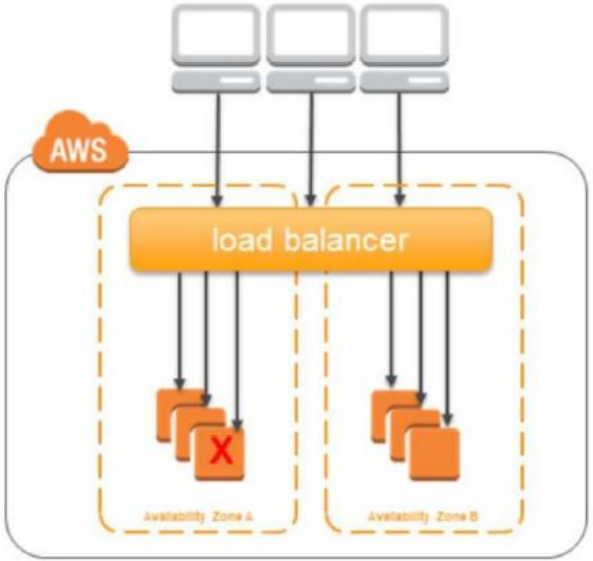
Elastic Load Balancing

- **Distributes** traffic across multiple EC2 instances, in multiple Availability Zones
- Supports **health checks** to detect unhealthy Amazon EC2 instances
- Supports the **routing and load balancing** of HTTP, HTTPS, SSL, and TCP traffic to Amazon EC2 instances

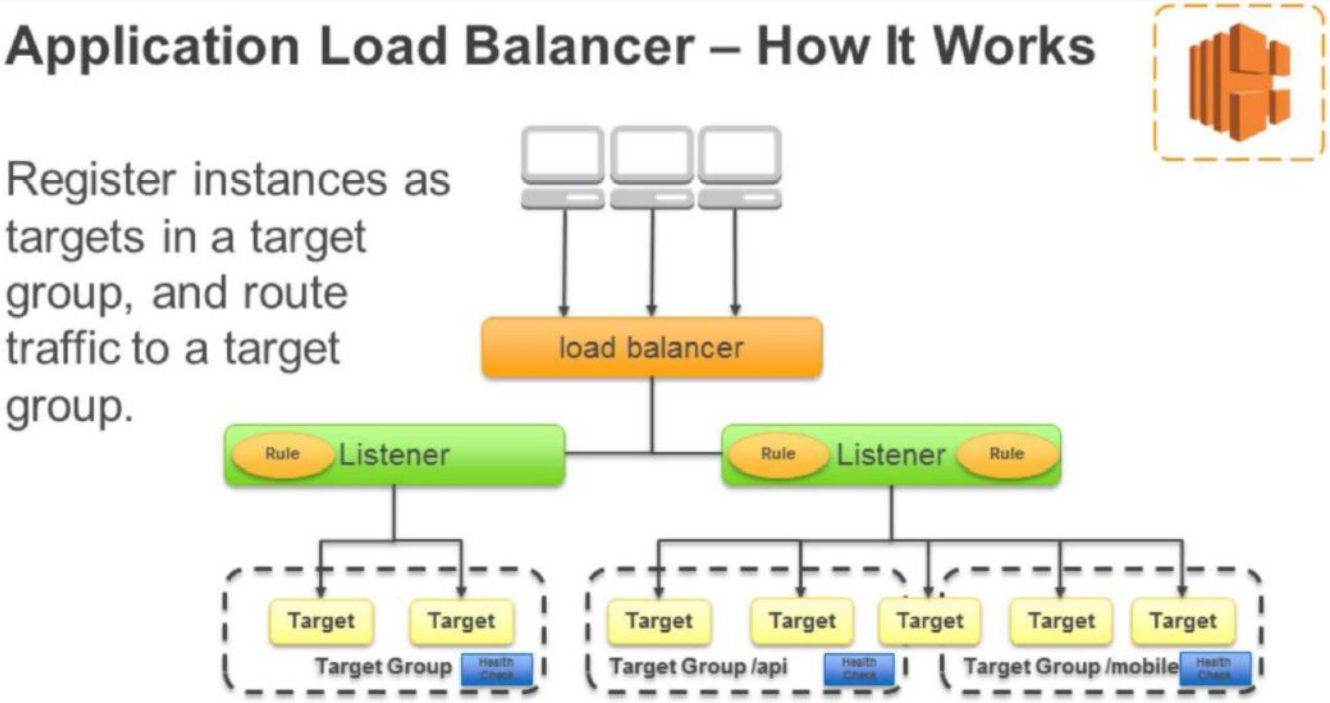
Load Balancer

Classic Load Balancer - How It Works

Register instances with your load balancer.



Application load balancer



Load balancer

Load Balancer Comparison

Classic Load Balancer benefits include support for:

- EC2-Classic.
- VPC.
- TCP and SSL listeners.
- Sticky sessions.
- OSI Layer 4
  (network protocol level)

ALB benefits include support for:

- Path-based routing.
- Routing requests to multiple services on a single EC2 instance.
- Containerized applications.
- Monitoring the health of each service independently.
- OSI Layer 7
  (application level)

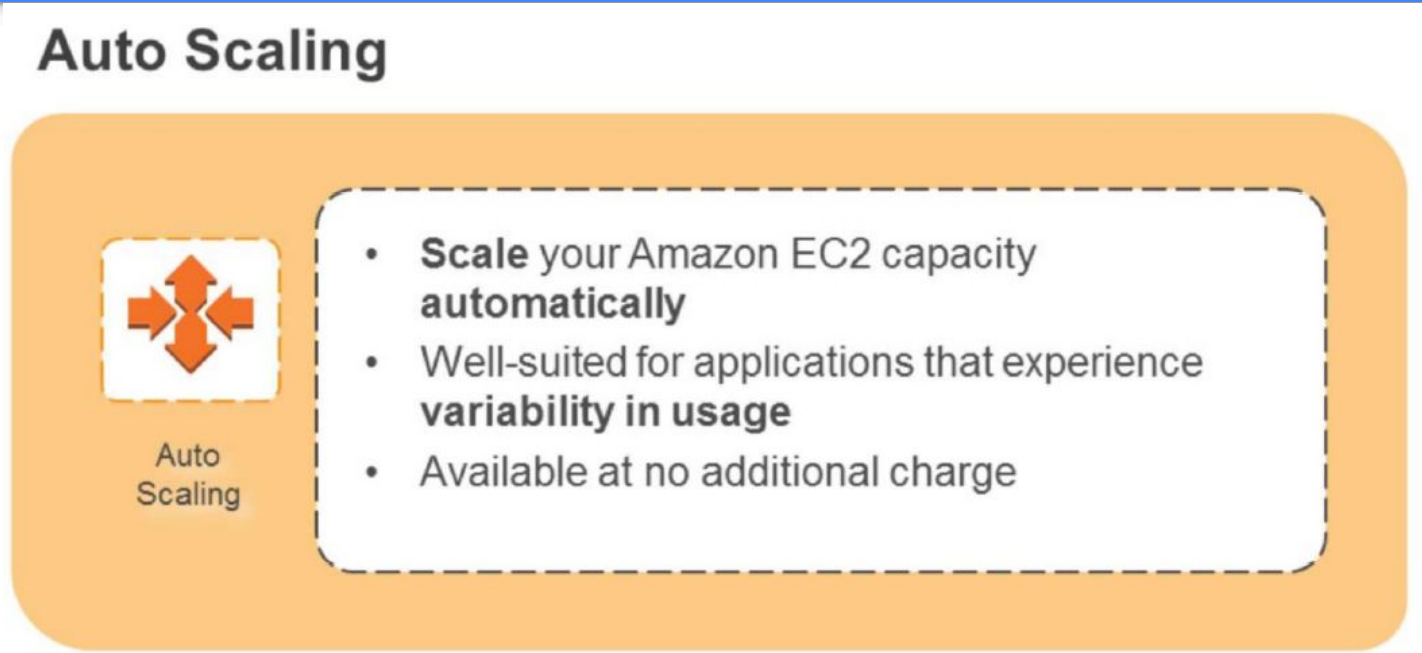
Monitoring

Amazon CloudWatch

Amazon CloudWatch

- A **monitoring service** for AWS cloud resources and the applications you run on AWS
- **Visibility into** resource utilization, operational performance, and overall demand patterns
- **Custom application-specific** metrics of your own
- **Accessible** via AWS Management Console, APIs, SDK, or CLI

Auto-scaling



Auto scaling

Auto Scaling Benefits

Better Fault Tolerance

Better Availability

Better Cost Management

Auto-scaling Templates

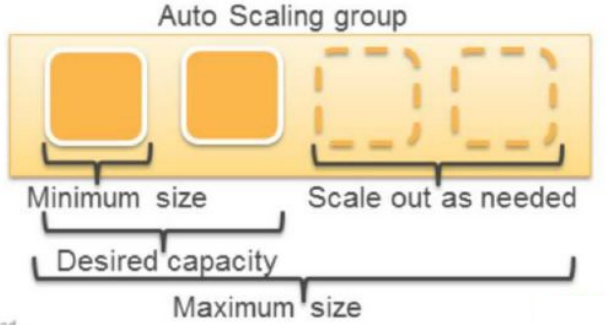
Launch Configurations

- A **launch configuration** is a template that an Auto Scaling group uses to launch EC2 instances.
- When you create a launch configuration, you can specify:
  - AMI ID
  - Instance type
  - Key pair
  - Security groups
  - Block device mapping
  - User data

Groups

Auto Scaling Groups

- Contain a collection of EC2 instances that share similar characteristics.
- Instances in an Auto Scaling group are treated as a **logical grouping** for the purpose of instance scaling and management.

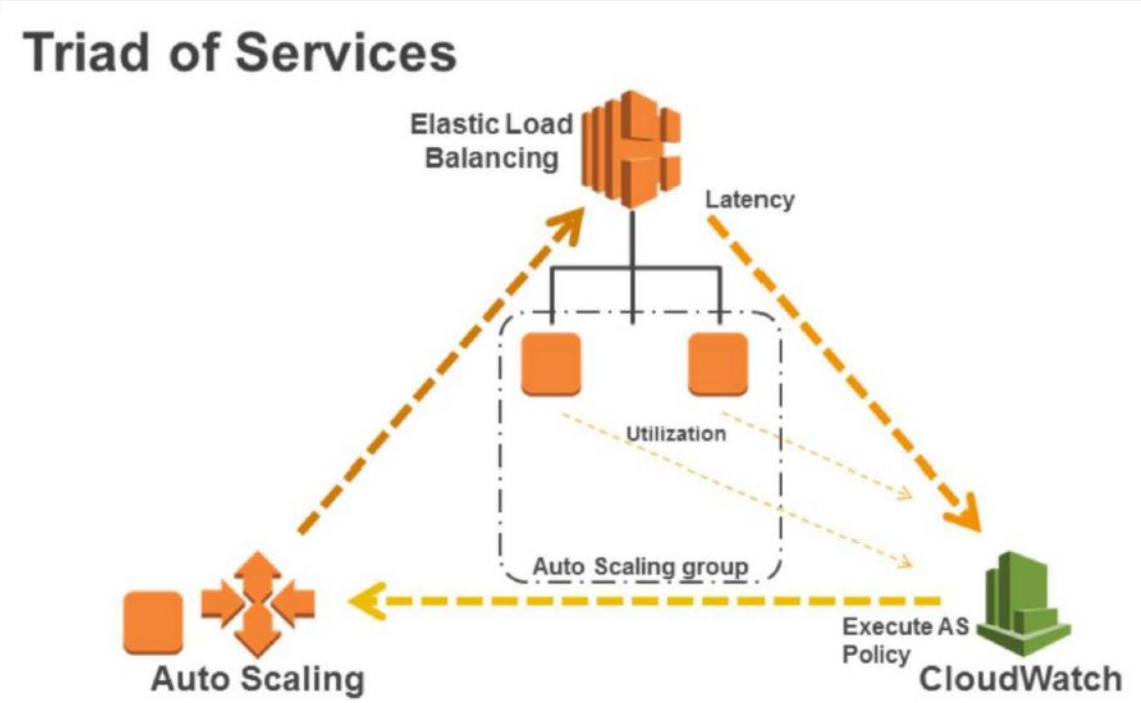


Auto scaling

Dynamic Scaling

- You can create a scaling policy that uses **CloudWatch alarms** to determine:
  - When your Auto Scaling group should **scale out**.
  - When your Auto Scaling group should **scale in**.
- You can use alarms to monitor:
  - Any of the metrics that AWS services send to Amazon CloudWatch.
  - Your own **custom metrics**.

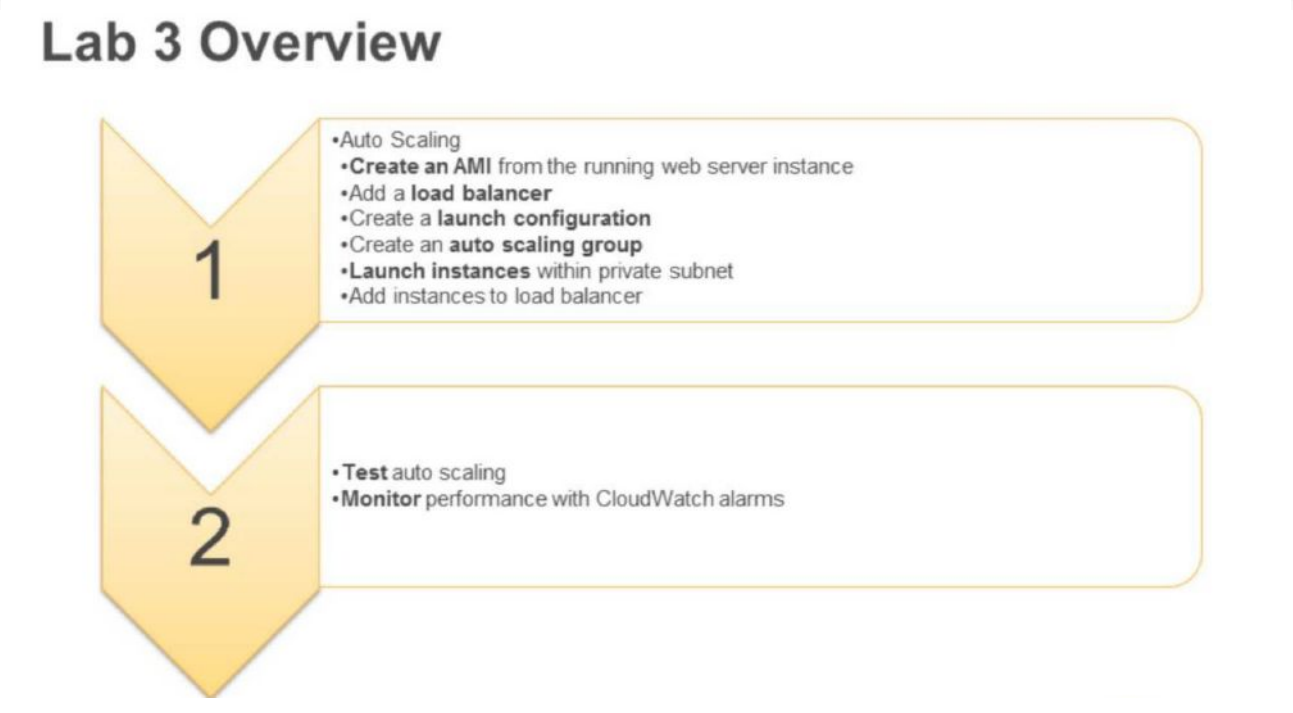
A set of services



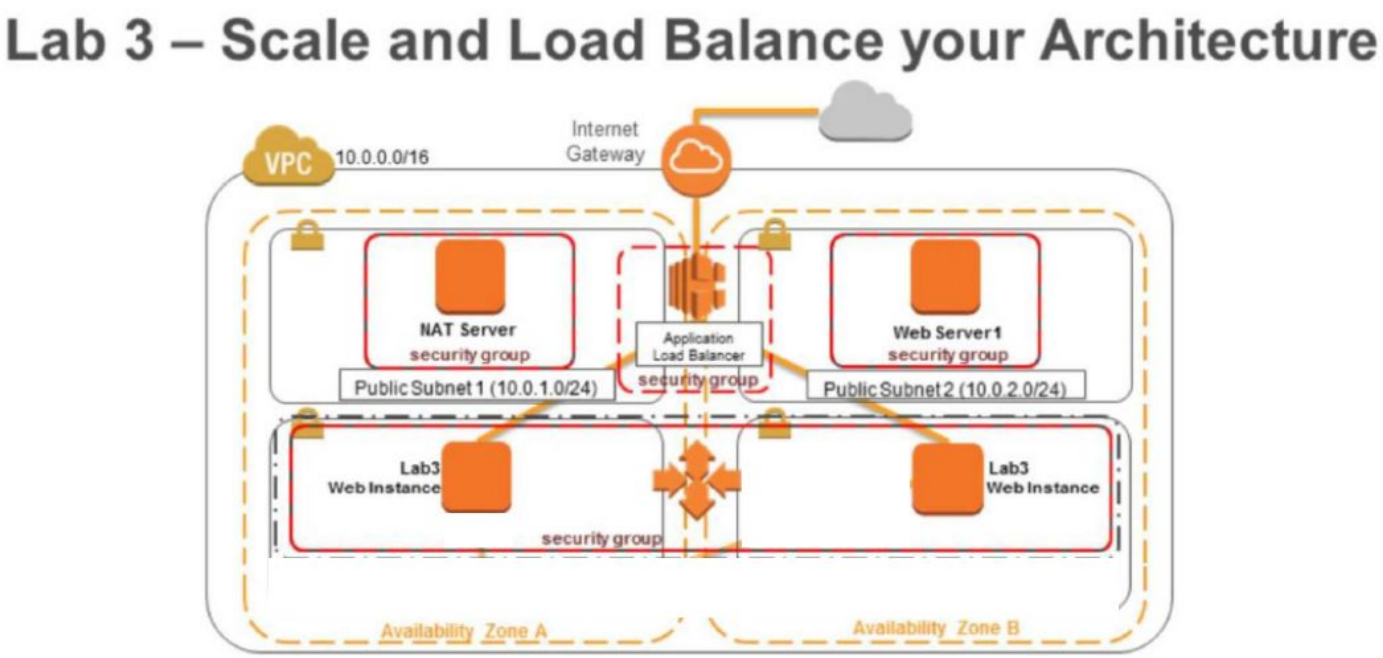
Lab 3: Scale & Load Balance

Amazon Web Services

Lab 3: Overview



Lab 3: Architecture



Lab 3: Step-by-step

1. Create a NAT GW and attach it to a public subnet.
2. Create an EC2 instance with a minimal web server in a private subnet **without** public IP
3. Create an AMI from the previously created instance
4. Create a route table; with a route 0.0.0.0/0 —> NAT GW, and attach it to the private subnets.
5. Create a Target Group for the autoscaling group
6. Create a Configuration template with the previously created AMI and add the web server security group
7. Create the autoscaling group associated to the configuration template, scale from 1 to 3 based on CPU usage (20%).
8. Edit the details of the autoscaling group and add it to the previously created target group
9. Create an ALB and add the previously created target group to it.
10. Try to access your website via the load balancer DNS name
11. Connect into a scaled machine and generate CPU utilization with this command: while true;do a=1;done

Infrastructure as Code

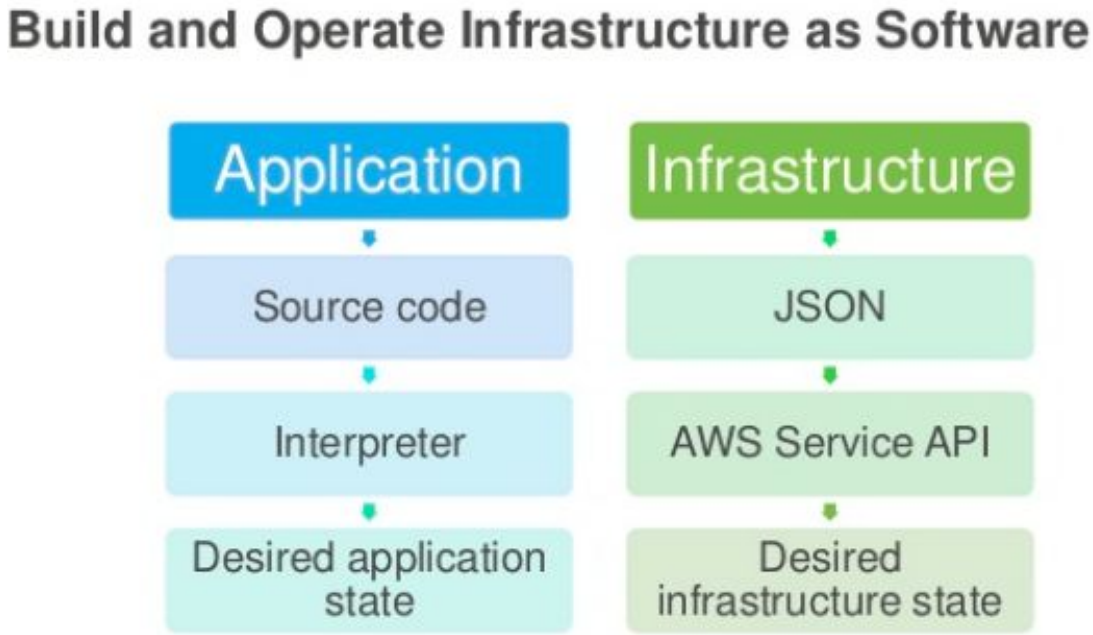
Amazon Web Services

What IaC is?

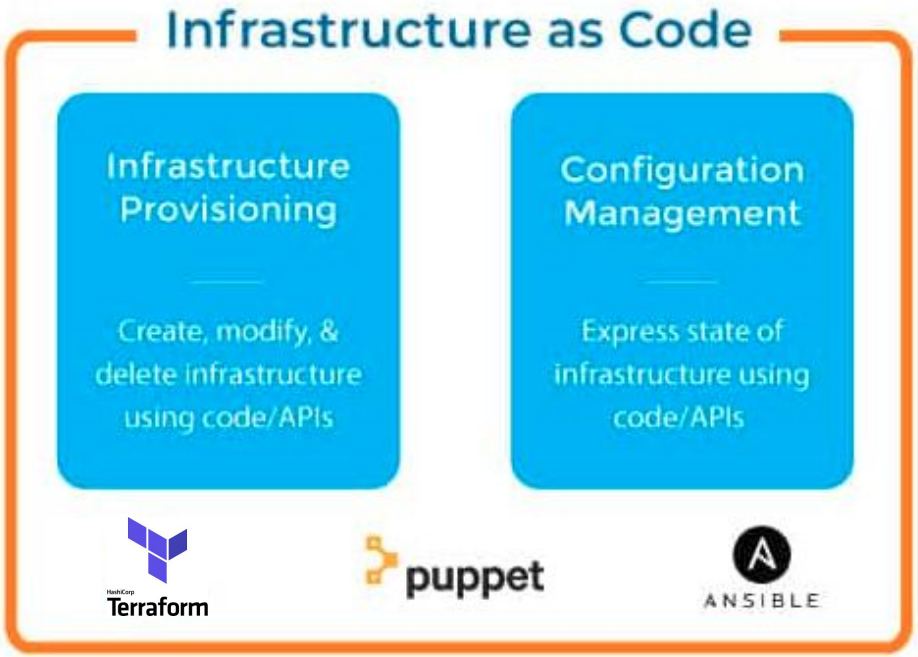
Infrastructure as Code:

Techniques, practices, and tools from software development applied to creating **reusable, maintainable, extensible, and testable** infrastructure.

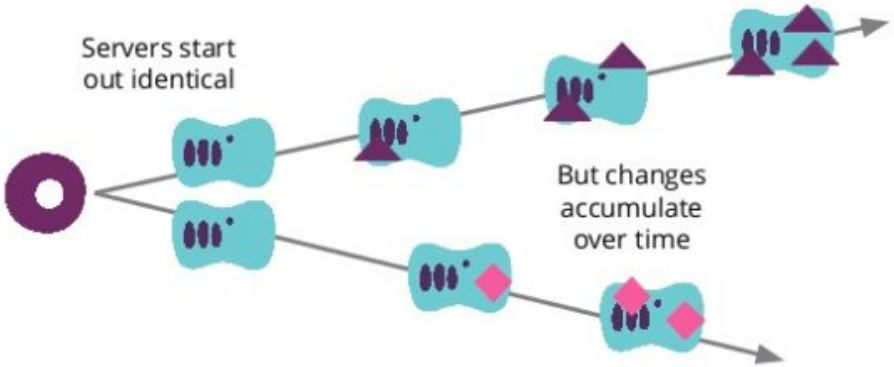
Everything is code



Infrastructure to configuration



Why IaC?


Servers start
out identical
But changes
accumulate
over time


I make changes
outside my
automation tool
My servers are
inconsistent
FEAR!
I'm afraid that running
my automation tool will
break something

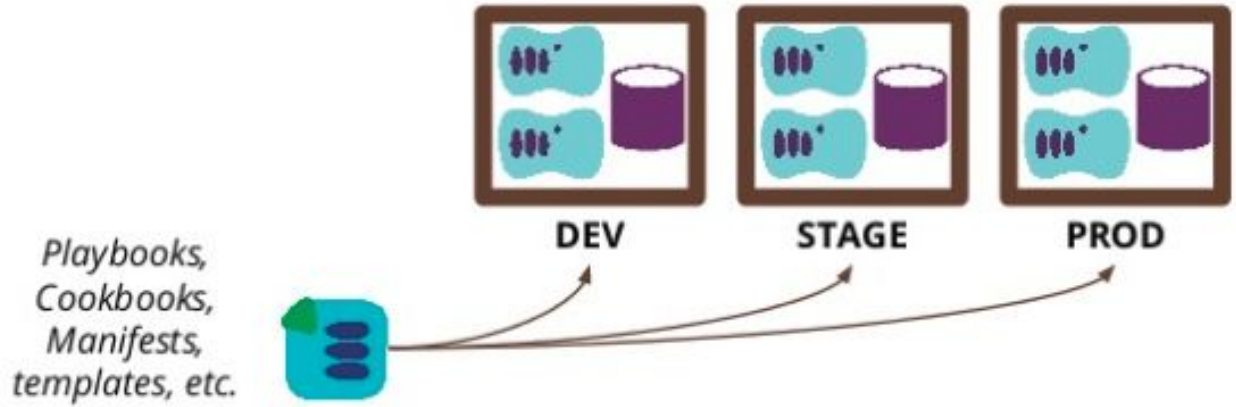
Automation

Automation isn't just for new servers!

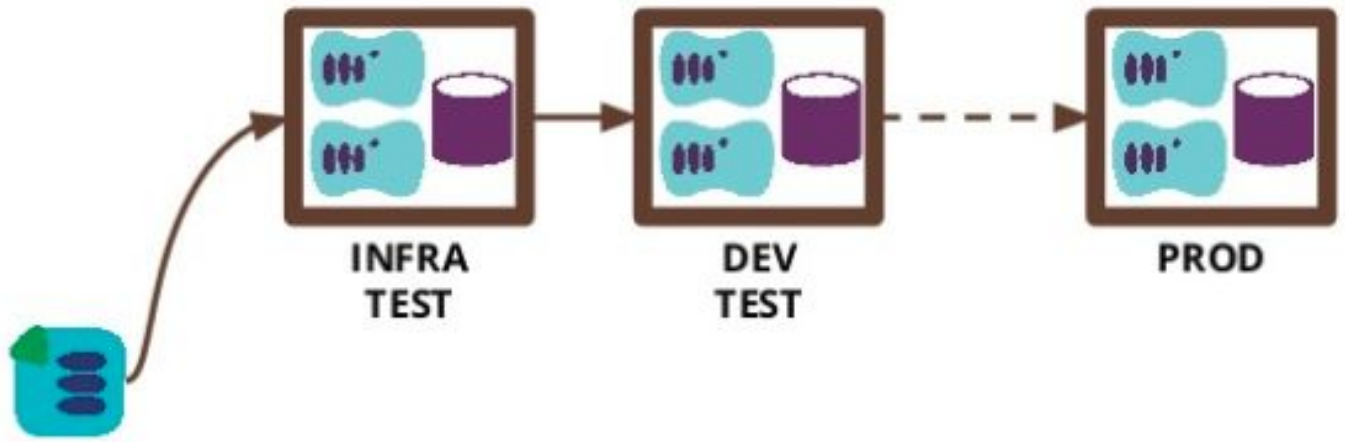
Configuration synchronization	Run Chef, Puppet, Ansible, etc. on a schedule
Immutable servers	Apply changes by rebuilding servers
Containerized servers	Apply changes by deploying new container instances

Don’t repeat yourself!

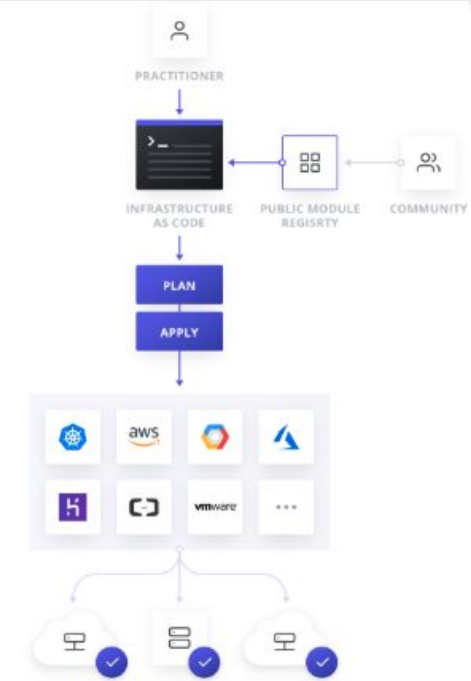
Re-use the same definition files across environments for a given application or service



Preventing mistakes



Terraform



A provisioning declarative tool that based on Infrastructure as a Code

Uses own syntax - HCL (Hashicorp Configuration Language)

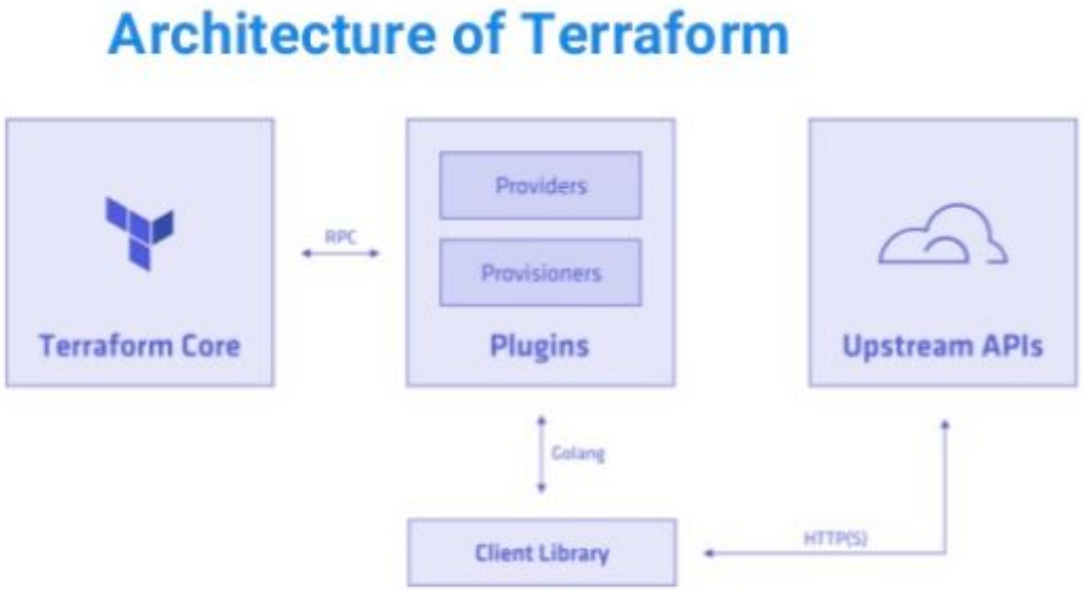
Written in Golang.

Helps to evolve you infrastructure, safely and predictably

Terraform in action

```
      + ipv6_association_id             = (known after apply)
      + ipv6_cidr_block                 = (known after apply)
      + main_route_table_id             = (known after apply)
      + owner_id                        = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.example: Creating...
aws_vpc.example: Creation complete after 2s [id=vpc-0f4231cc2068c6
0a4]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
terraformUser$ _
```

Architecture



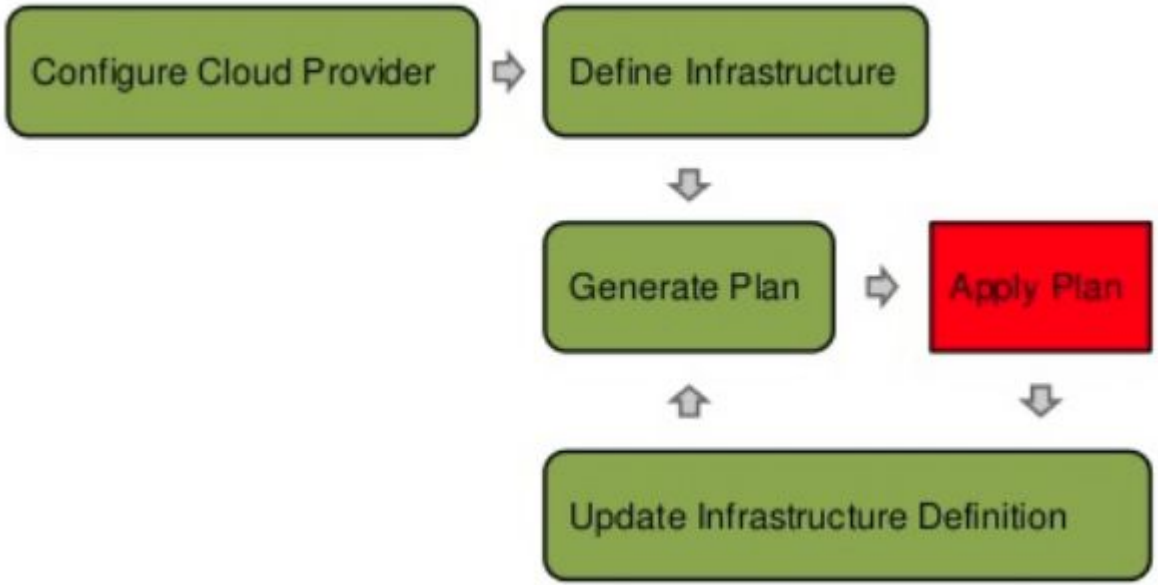
Core concepts

- **Providers:** A source of resources. [With an API endpoint and authentication. E.g AWS]
- **Resource:** Everything that has a set of configurable attributes and a lifecycle such as create, read, update, delete. [aws ec2 instance] -- impies id and state
- **Data :** information read from providers. E.g. lookup from own AWS account for ami_id or keypairs.
- **Provisioner:** initialize a resource from a local or remote script.

terraform state

- Terraform State stores status of your managed infrastructure and configuration.
- This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.
- State is stored by default in a local file named "terraform.tfstate", but it can also be stored remotely, which works better in a team environment (backends).

Workflow


Configure Cloud Provider
Define Infrastructure
Generate Plan
Apply Plan
Update Infrastructure Definition

Code example

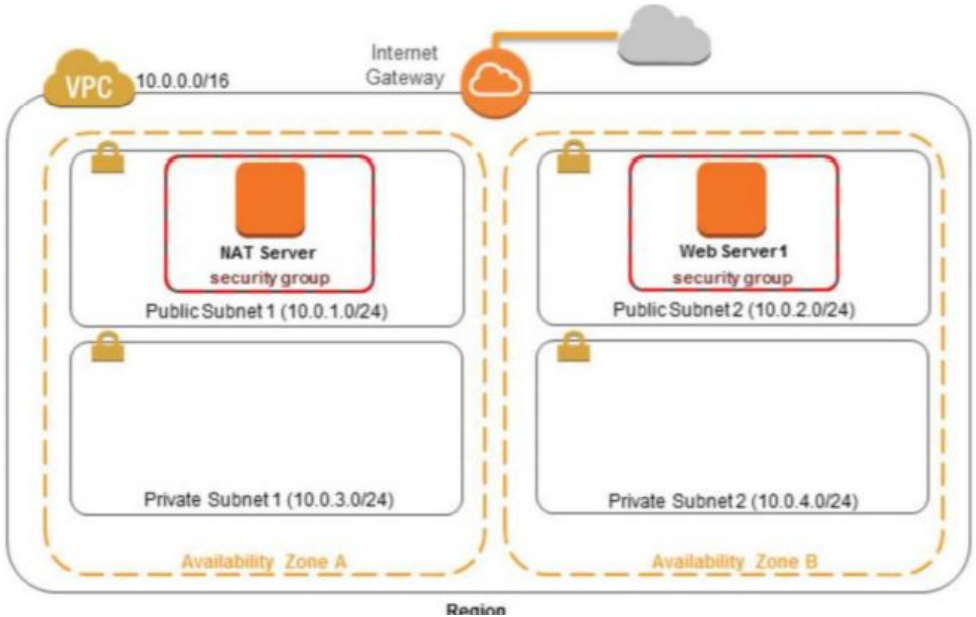
```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "example" {
    ami = "ami-408c7f28"
    instance_type = "t2.micro"
    tags { Name = "terraform-example" }
}
```

This template creates a single EC2 instance in AWS

Lab 4: Deploy with terraform

Amazon Web Services

Lab 4: Architecture



Lab 4: Step-by-step

1. Install terraform in your cloud9 environment via the install-terraform script
2. Open a new file in Cloud9 to write your first terraform code
3. Go to https://www.terraform.io/docs/providers/aws/index.html, add the block code for the provider, Configure the us-east-1 region.
4. Add a VPC resource with a tag including terraform, and and a IP Range (10.0.0.0/16).
5. Create the VPC via terraform with: “terraform init”, “terraform plan”, check the result and try to “terraform apply” Check in the AWS console to see the new VPC
6. Try to destroy the VPC with “terraform destroy” and check in the AWS console
7. Add a subnet to the VPC with the range 10.0.0.0/24 and a tag. Try to apply the new code.
8. Now add the IGW, Route tables and a web server instance. Good luck !

Lab 4: Step-by-step

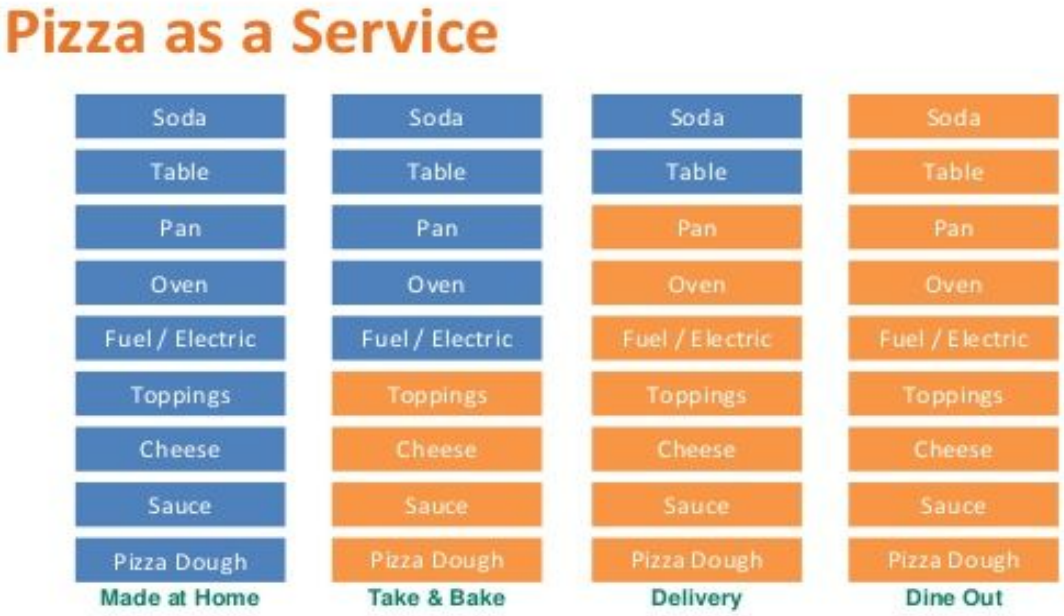
1. Instance
2. ubuntu 18.04
3. t2.micro
4. security group
5. Public ip
6. subnet public a or b
7. Key pair

Userdata: script wordpress

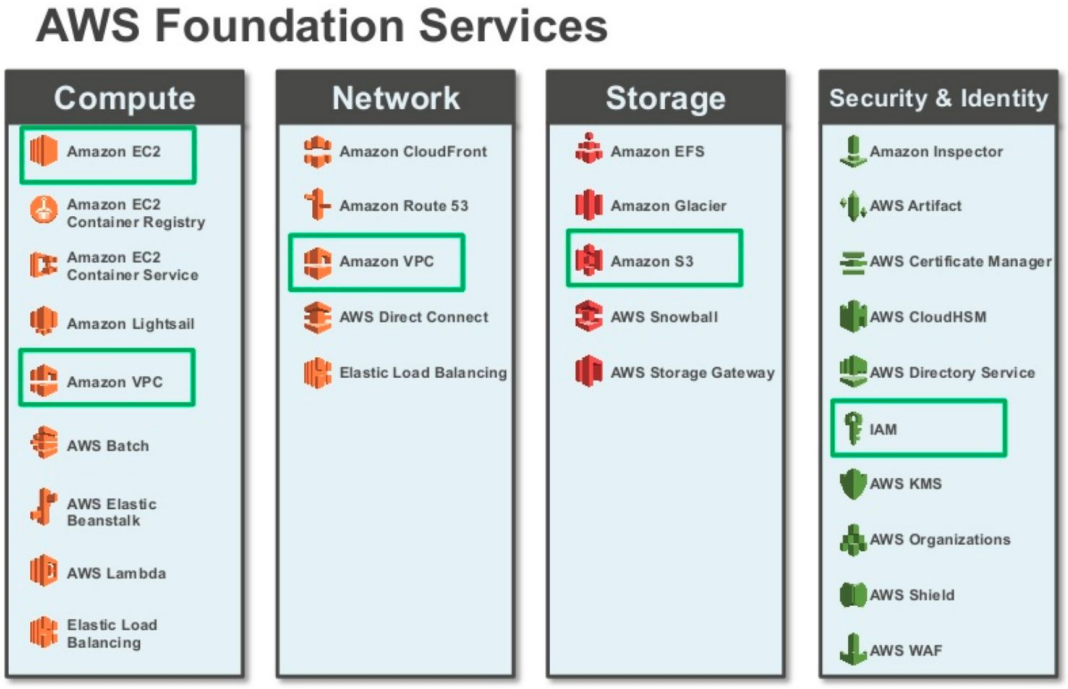
Cloud Computing

Conclusion

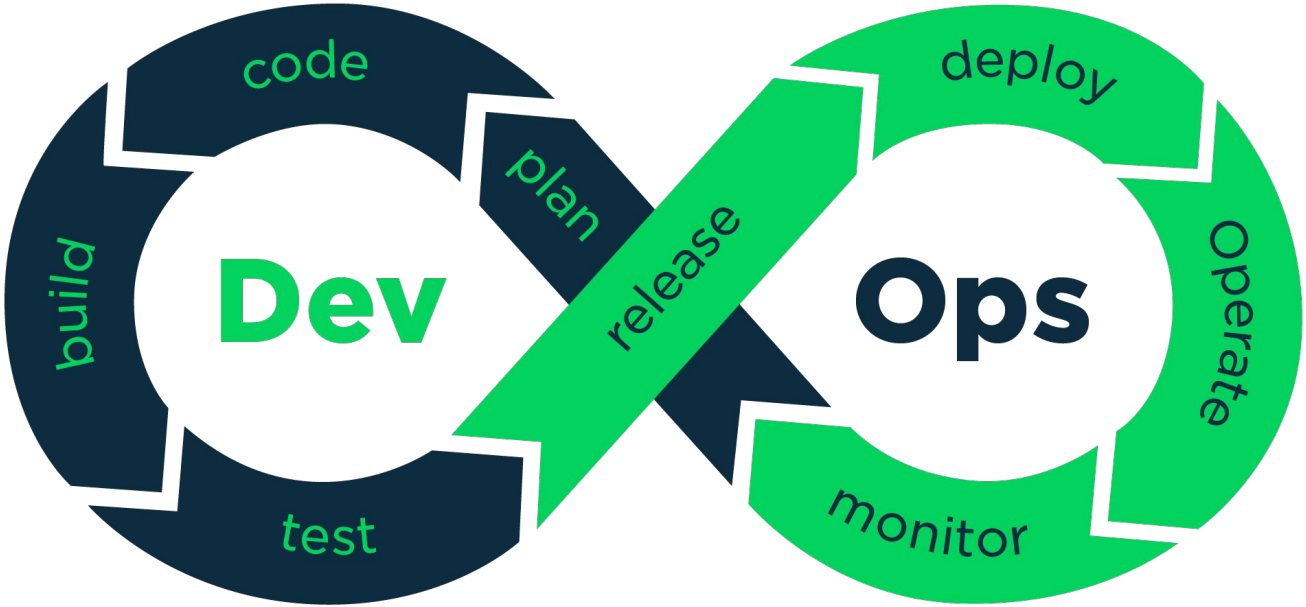
Models


Pizza as a Service
Soda
Table
Pan
Oven
Fuel / Electric
Toppings
Cheese
Sauce
Pizza Dough
Made at Home
Soda
Table
Pan
Oven
Fuel / Electric
Toppings
Cheese
Sauce
Pizza Dough
Take & Bake
Soda
Table
Pan
Oven
Fuel / Electric
Toppings
Cheese
Sauce
Pizza Dough
Delivery
Soda
Table
Pan
Oven
Fuel / Electric
Toppings
Cheese
Sauce
Pizza Dough
Dine Out

Foundation services

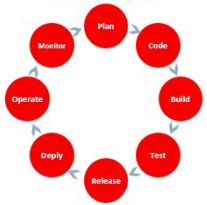
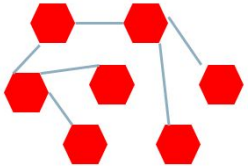
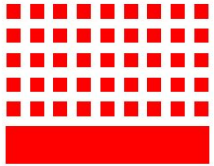
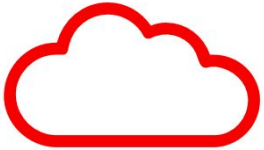
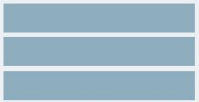
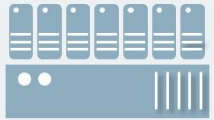


DevOps

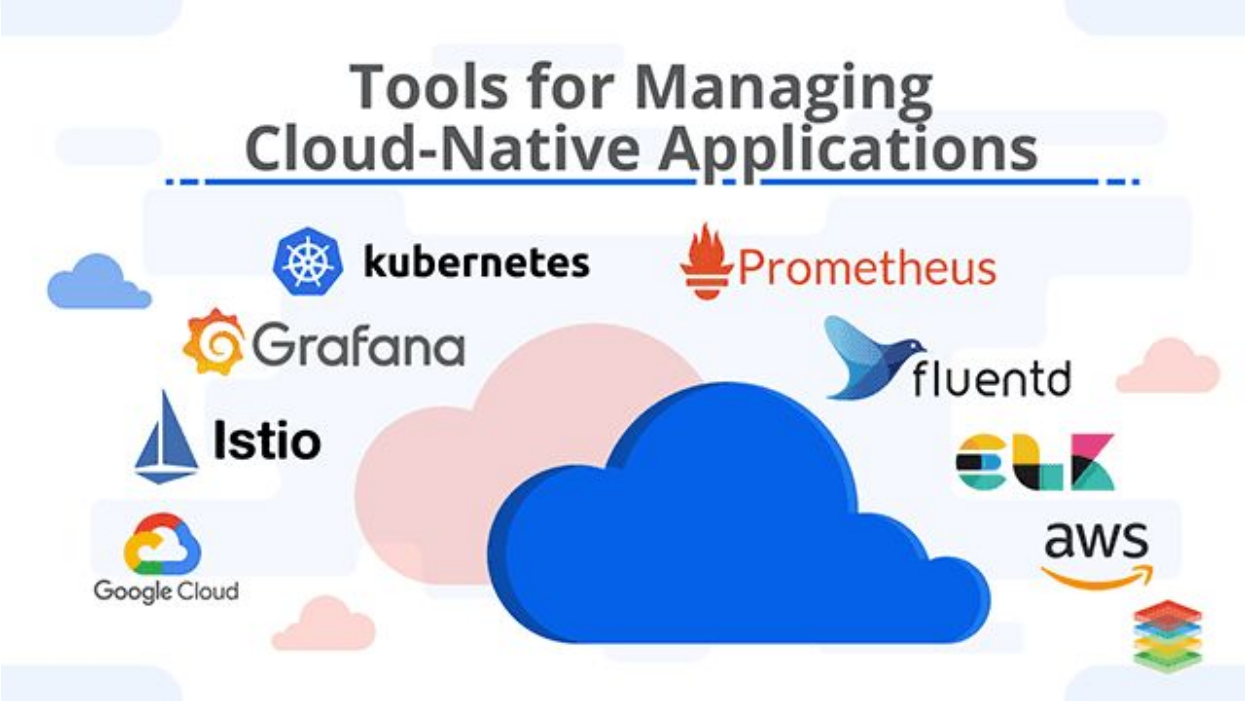

code
plan
build
Dev
test
release
deploy
Ops
operate
monitor

Cloud Native

Cloud Native

	Development Process	Application Architecture	Deployment and Packaging	Infrastructure
Now	DevOps 	Microservices 	Containers 	Cloud 
~2000	Agile 	N-Tier 	Virtual Servers 	Hosted 
~1980 ~1990	Waterfall 	Monolithic 	Physical Server 	DataCenter 

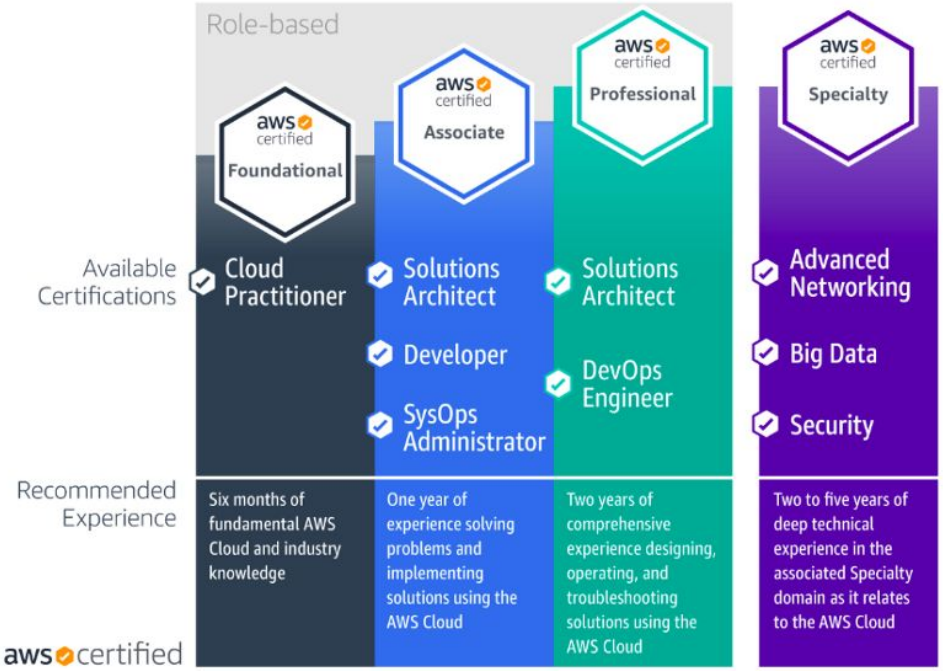
Tools


Tools for Managing
Cloud-Native Applications
kubernetes
Prometheus
Grafana
fluentd
Istio
ELK
aws
Google Cloud

Landscape


Cloud Native Landscape
v0.9.9
Database & Data Analytics
Streaming
SCM
Application Definition
CI/CD
App Definition & Development
Scheduling & Orchestration
Coordination & Service Discovery
Service Management
Orchestration & Management
kubernetes
CNCF Project
CoreDNS
linkerd
gRPC
envoy
Cloud-Native Storage
Container Runtime
Cloud-Native Network
Runtime
container
rkt
CNI
Host Management / Tooling
Infrastructure Automation
Container Registries
Secure Images
Key Management
Provisioning
notary
Public
Private
Cloud
aws
Azure
ORACLE
MAAS
openstack
Platforms
PaaS / Container Service
Serverless/Event-based
Observability & Analysis
Monitoring
Prometheus
Logging
fluentd
Tracing
OPENTRACING
JAEGER
github.com/cncf/landscape
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify
Greyed logos are not open source

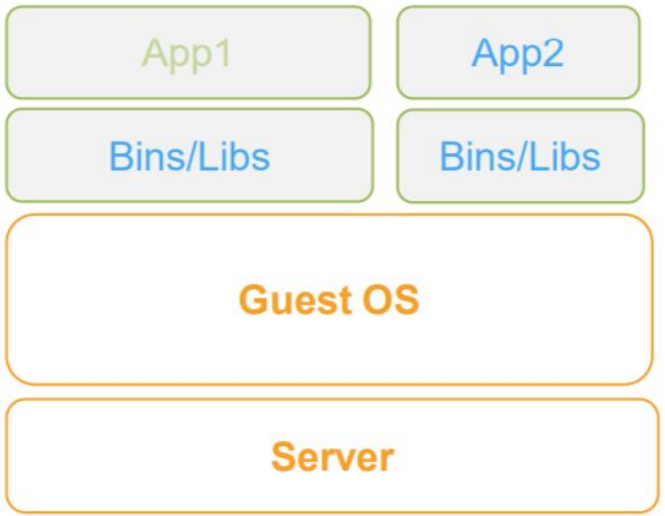
Certifications



Containers

Amazon Web Services

What are containers?



OS virtualization

Process isolation

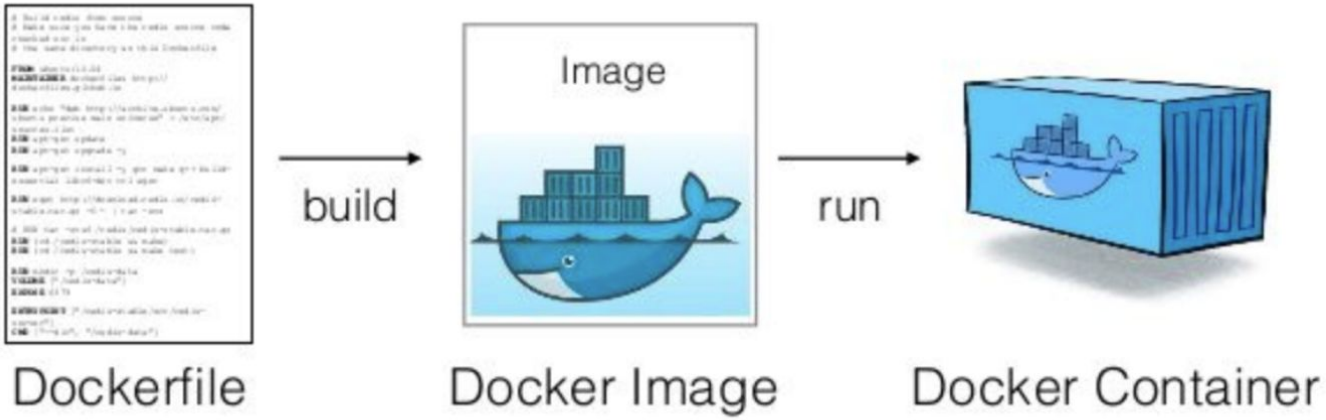
Images

Automation

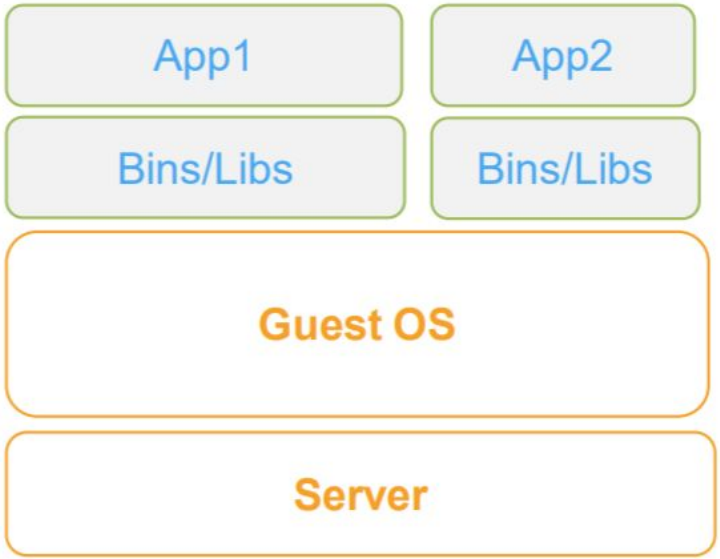
How it works?

FROM: ubuntu:latest

RUN apt-get update
RUN apt-get install apache2



Advantages



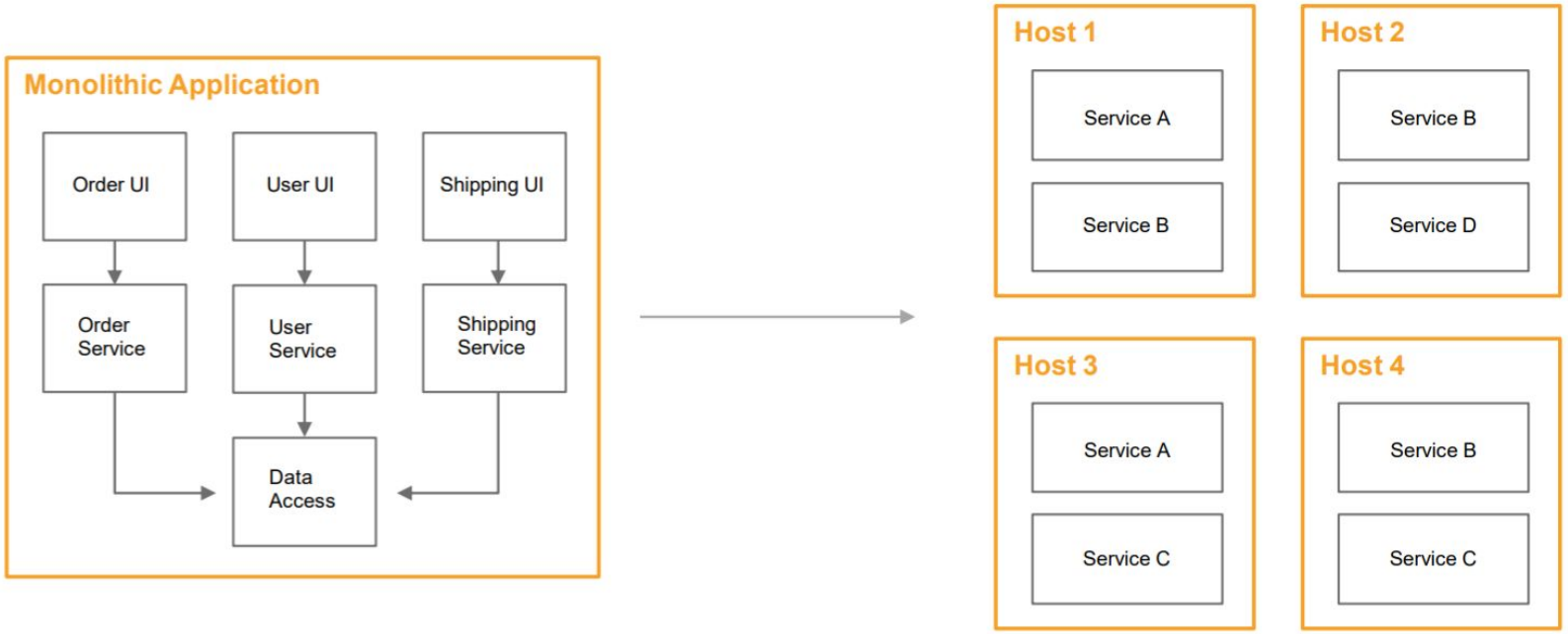
Portable

Flexible

Fast

Efficient

Microservice application



Orchestration

Running one container is easy

Running many containers is hard…

Some challenges

Cluster management

Availability

Scheduling

Security

Monitoring

Integration with AWS services

ECS

Amazon EC2 Container Service (ECS)

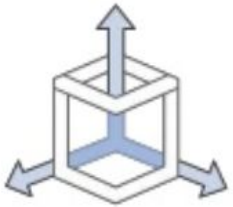
Highly scalable, high performance container management system.

Eliminates the need to install, operate, and scale your own container management infrastructure.

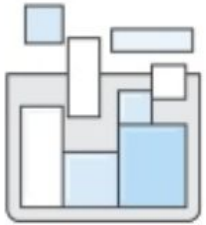
Main features

Amazon EC2 Container Service (ECS)

ECS provides a managed platform for:



Cluster management



Container orchestration

Deep AWS integration

Auto-scaling Templates

How does ECS map to traditional workloads?

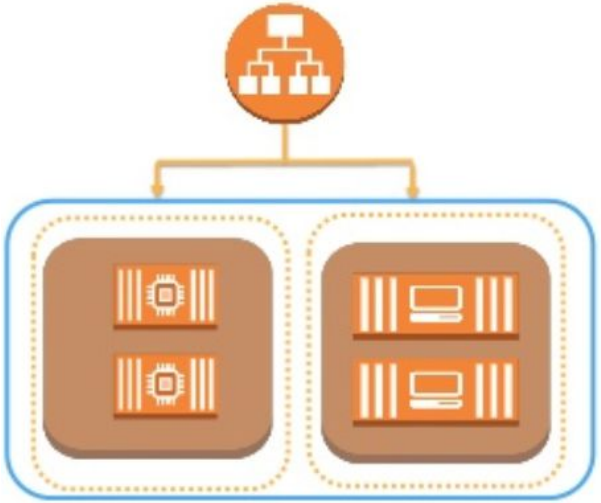
Instances: standard EC2 boxes. Once registered to a Cluster, your Tasks run here

Services: layer that manages and places Tasks

Tasks: container wrapper and configuration around processes running on the instance

Auto-scaling Templates

How does ECS work?



Load balancer: (ALB or EC2 classic) routes traffic to the cluster instances.

Cluster is made up of one or more EC2 instances

Each **cluster instance** runs one or more **Services**

Auto-scaling Templates

How does ECS work?

Each **cluster instance** runs one or more **Services**

A **Service** controls things like the number of copies of a Task you want running (Desired Count), and registers your Service with a load balancer

A **Task Definition** controls things like container image, environment variables, resource allocation, logger, and other parameters

Fargate

Networking only

Resources to be created:

Cluster

VPC (optional)

Subnets (optional)

Powered by AWS Fargate

EC2 Linux + Networking

Resources to be created:

Cluster

VPC

Subnets

Auto Scaling group with Linux AMI

Lab 4: Deploy containers

Amazon Web Services

Lab 4: Step-by-step

1. Try to deploy a nginx container from the ECS get started menu, add an ALB to your service.
2. Access to your nginx container from the ALB DNS name.
3. Create a ECS cluster **EC2 Linux + Networking**
4. Create a Task definition: One nginx container
5. Create a Service In the previously created ECS cluster, attach the previous Task definition, add a Load Balancer.
6. Access to your nginx container from the ALB DNS name.
7. Try to create a new task definition with the container you want (find docker images on dockerhub.com)
8. Deploy a new Service with your new task definition