Terraform Advanced

Gagandeep Singh

Introduction

Your Name

Total experience

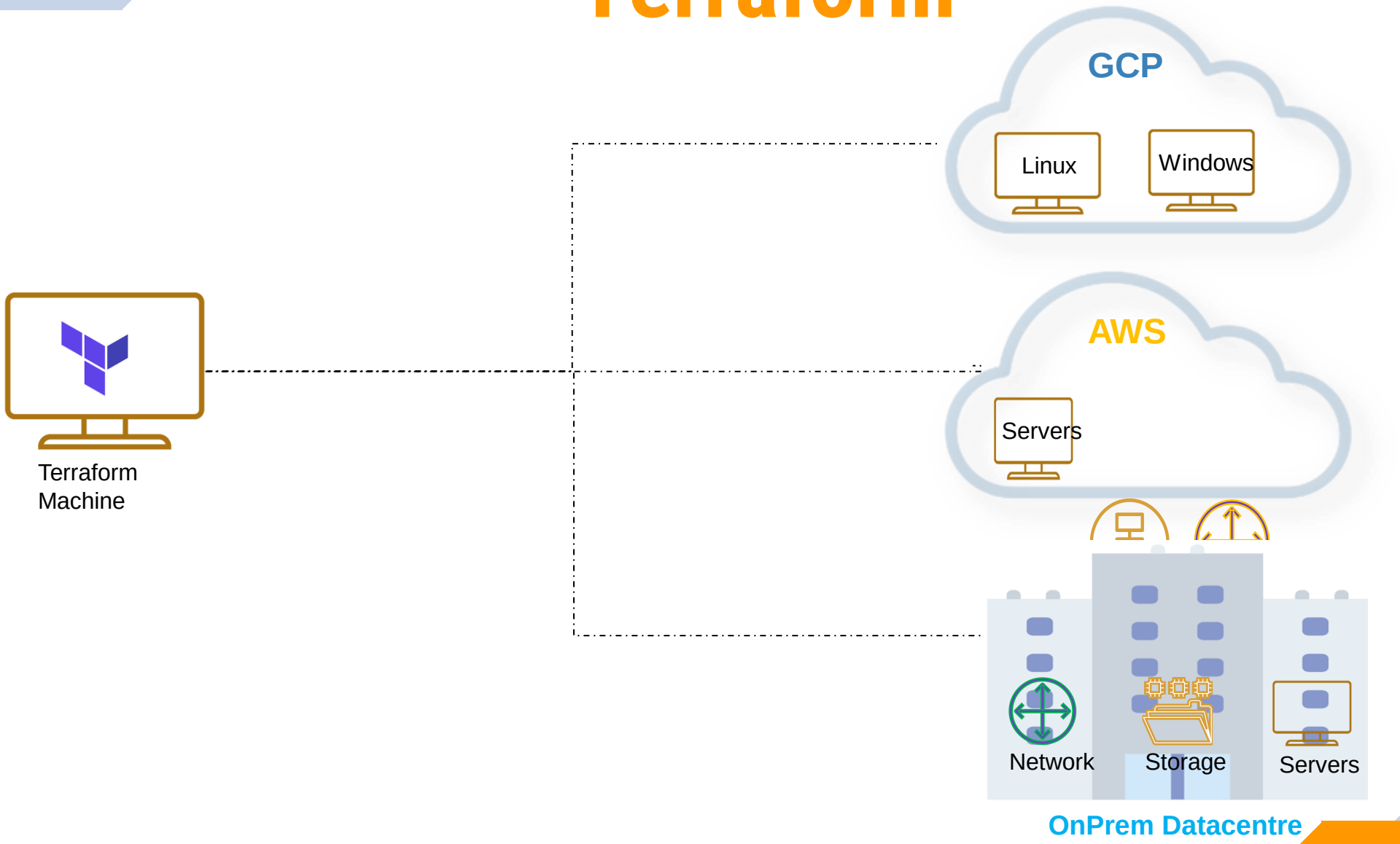
Background – Development / Infrastructure / Database / Network

Experience on Cloud and Terraform

Your expectations from this training

What is Orchestration?

Terraform


GCP
Linux
Windows
AWS
Servers
Terraform
Machine
Network
Storage
Servers
OnPrem Datacentre

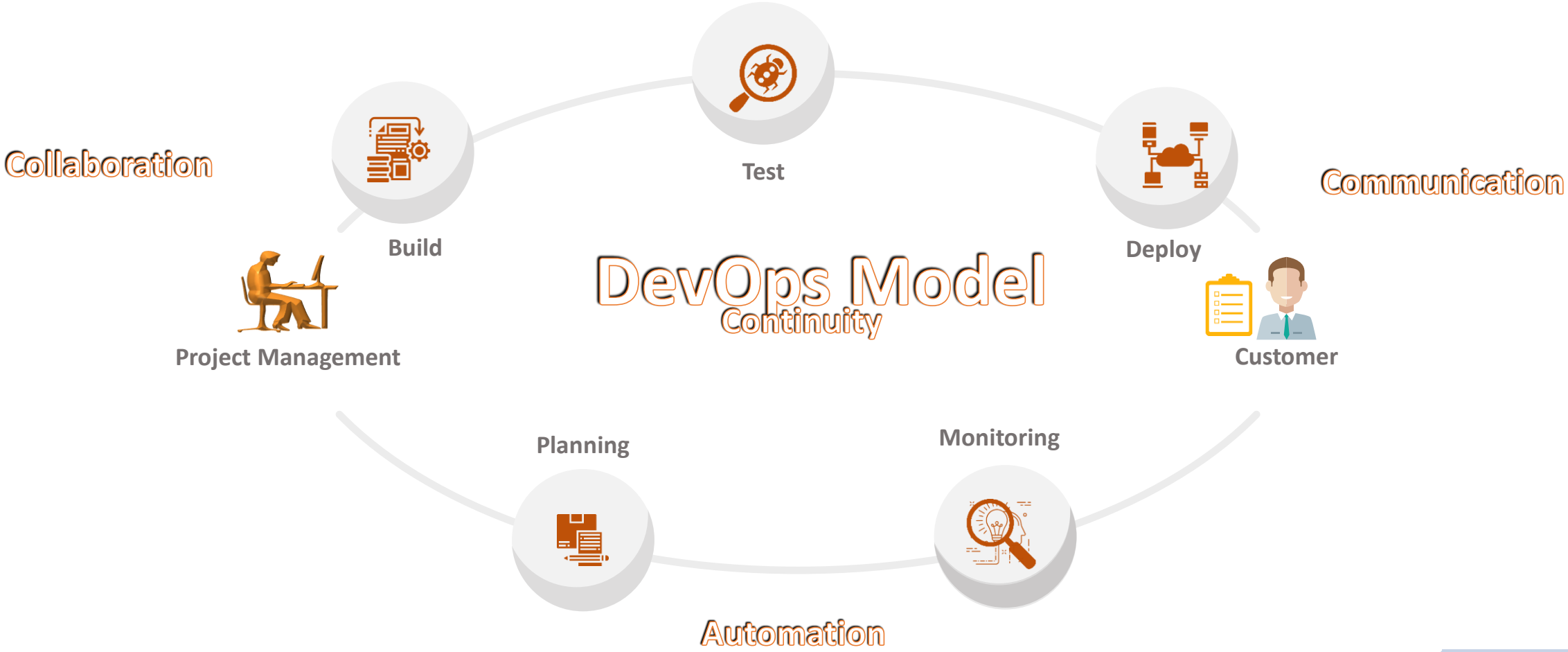
GUI vs CLI vs IAC

- **GUI (Graphical User Interface)**
  - ✓ Best for end user experience
  - ✓ Easy management
  - ✓ Bad for Automation
  - ✓ Not helpful for Administrators

- **CLI (Command Line Interface)**
  - Best for Admin Experience
  - Easy management for Admin level tasks
  - Bad for end user experience
  - Bad for maintaining desired state and consistency

- **IaC (Infrastructure as Code)**
  - Best for Admin Experience
  - Easy management for Admin tasks
  - Easy to understand for end users too
  - Can easily maintain consistency and desired state
  - Infrastructure is written in files, so can be versioned

DevOps


Collaboration
Build
Test
Deploy
Communication
Project Management
DevOps Model
Continuity
Customer
Planning
Monitoring
Automation

DevOps in Action

Continuous Feedback

Continuous Improvement

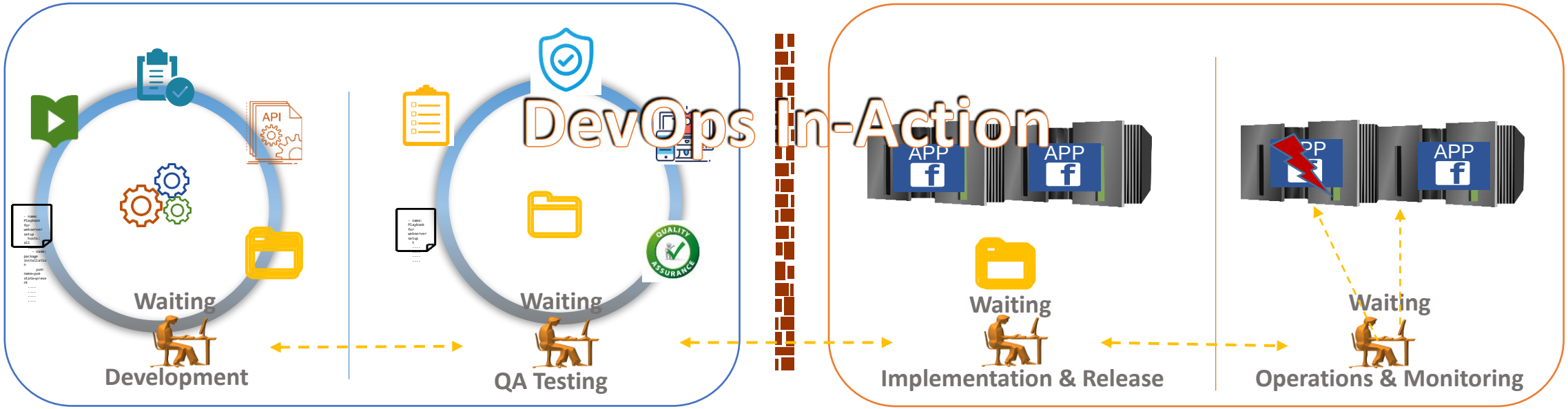
Continuous Planning

Continuous Integration

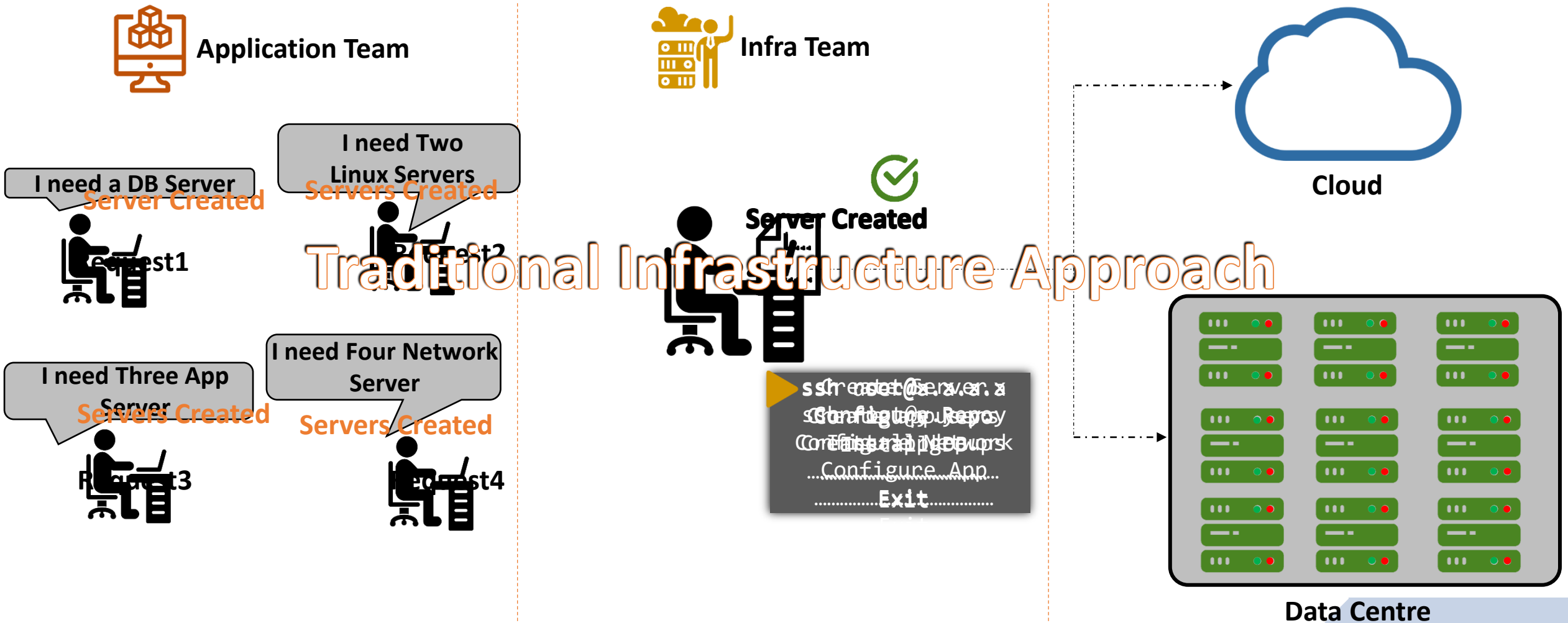
Continuous Delivery

Continuous Deployment

Continuous Monitoring



Why DevOps IaC



IAC (Infrastructure as Code)

Application Team

I need DB Server

Server Created

Request1

I need Two Linux Servers

Servers Created

Request2

need Three Linux Servers

Servers Created

Request3

I need Four Linux Servers

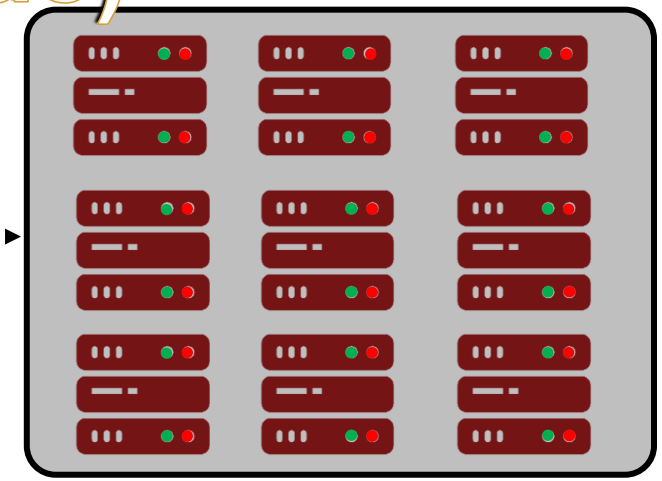
Servers Created

Request4

Infra Team

```
File is: main.tf
provider "aws" {
  region = "us-east-1"
}
resource "aws_instance" “requestfour" {
  count = "4"
  ami = ami-03d5c68bab01f3496
  instance_type = "t2.micro"
  tags = {
    Name = "DevOpsInAction"
  }
}
output "myawsserver" {
  value =
"${aws_instance.myawsserver.public_ip}"
}
```

Cloud



Data Centre

IaC is Managing Infrastructure in files rather than manually configuring resources in a user interface

Terraform

Terraform is an easy-to-use IT Orchestration & Automation Software for System Administrators & DevOps Engineers.

- It is the infrastructure as code offering from Hashicorp.
- It is a tool for building, changing, and managing infrastructure in a safe, repeatable way.
- Configuration language called the HashiCorp Configuration Language (HCL) is used to configure the Infrastructure.
- Compatible with almost all major public and private Cloud service provider

Terraform

Infrastructure as code (IAC)

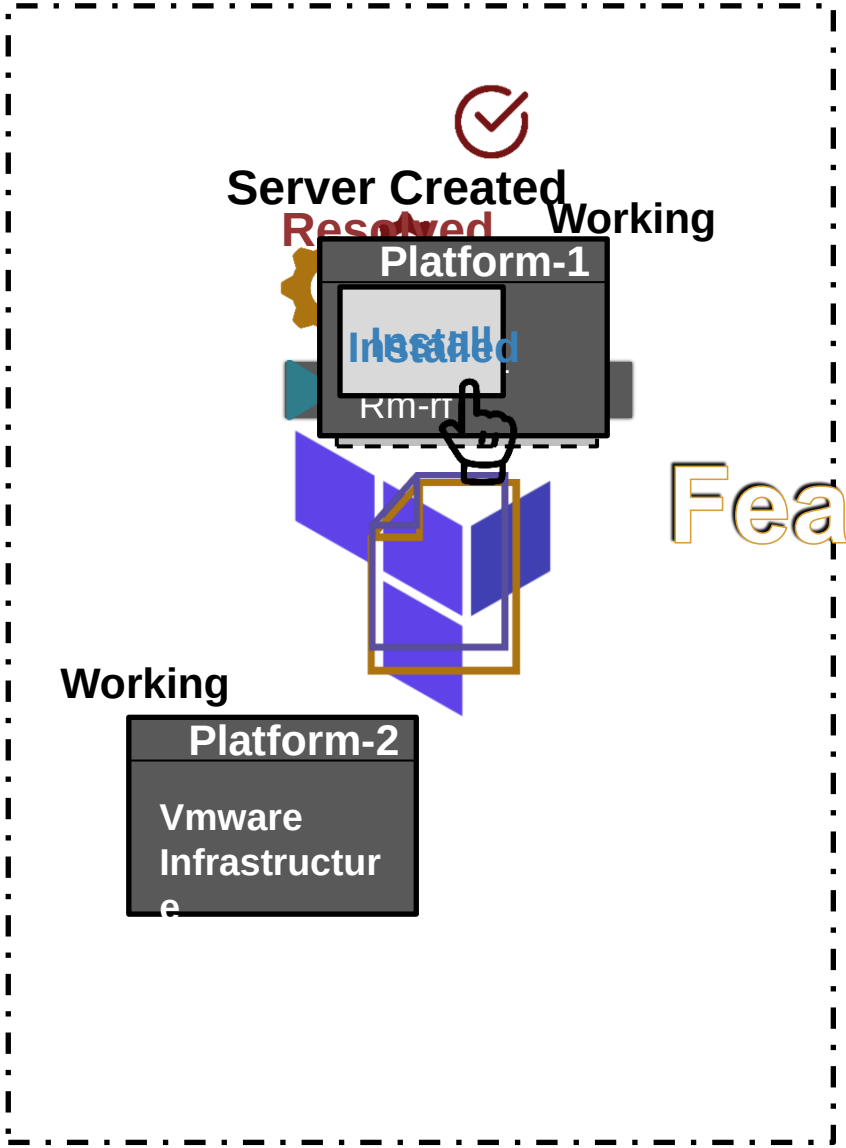
July 2014, HashiCorp

What is Terraform?

Opensource / Enterprise

HCL (Hashicorp Configuration Language)

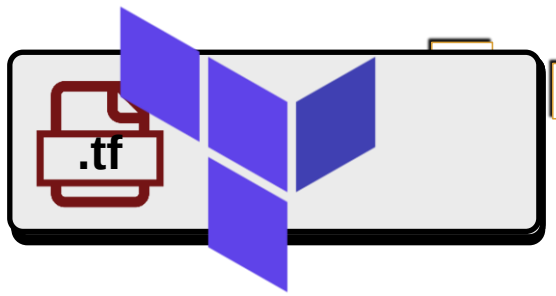
Terraform



Feature & Advantages

- Easy Installation
- Declarative in Nature
- Idempotent
- Intelligent Dependency Resolver
- Platform Agnostic
- Simple and easy to use

Terraform



Terraform Terminologies

Providers

Variables

Resources

Provisioners

DataSources

Outputs

Modules

File extension
.tf

Terraform

main.tf

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

Provider Block

```
resource "aws_instance" "myserver" {
  ami = "ami-030ff268bd7b4e8b5"
  instance_type = "t2.micro"
  tags = {
    Name = "DevOpsInAction"
  }
}
```

Resource Block

```
output "myserveroutputs" {
  description = "Display Servers Public IP"
  value = "${aws_instance.myserver.public_ip}"
}
```

Output Block

Terraform File (Sample Code)

Why Terraform?

- Infrastructure as Code – Write stuff in files, Version it, share it and collaborate with team on same.
- Declarative in Nature
- Automated provisioning
- Clearly mapped Resource Dependencies
- Can plan before you apply
- Consistent
- Compatible with multiple providers and infra can be combined on multiple providers
- 50+ list of official and verified providers
- Approx. 2500+ Modules readily available to work with
- Both Community and Enterprise versions available
- A best fit in DevOps IaC model

Why Terraform?

- **Platform Agnostic** – Manage Heterogeneous Environment
- **Perfect State Management** – Maintains the state and Refreshes the state before each apply action.

  Terraform state is the source of truth. If a change is made or a resource is appended to a configuration, Terraform compares those changes with the state file to determine what changes result in a new resource or resource modifications.
- **Confidence**: Due to easily repeatable operations and a planning phase to allow users to ensure the actions taken by Terraform will not cause disruption in their environment.

Terraform and its Peers

- Chef
- Puppet
- SaltStack
- Ansible
- CloudFormation
- Terraform
- Kubernetes

Terraform and its Peers

Many tools available in Market. Few things to consider, before selecting any tool:

- Configuration Management vs Orchestration
- Mutable Infrastructure vs Immutable Infrastructure
- Procedural vs Declarative
- Client/Server Architecture vs Client-Only Architecture

Terraform and its Peers

	Chef	Puppet	Ansible	SaltStack	CloudFormation	Terraform
Code	Open source	Open source	Open source	Open source	Closed source	Open source
Cloud	All	All	All	All	AWS only	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable	Immutable
Language	Procedural	Declarative	Declarative	Declarative	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client-Only	Client/Server	Client-Only	Client-Only

Knowledge Checks

- What is Configuration Management?
- What is Orchestration?
- List a few available configuration Management tools.
- What are the Advantages of Terraform?

Summary: Terraform

Terraform is an easy-to-use IT Orchestration & Automation, Software for System Administrators & DevOps Engineers.

- Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently.
- Terraform can manage existing and popular service providers as well as custom in-house solutions.
- Maintain Desired State
- Highly scalable and can create a complete datacenters in minutes
- Agentless solution
- Declaration in nature than Procedural
- Uses Providers API to provision the Infrastructure
- Terraform creates a dependency graph to determine the correct order of operations.

AWS

Amazon Web Services

AWS (Amazon Web Services) is a group of web services (also known as cloud services) being provided by Amazon since 2006.

AWS provides huge list of services starting from basic IT infrastructure like CPU, Storage as a service, to advance services like Database as a service, Serverless applications, IOT, Machine Learning services etc..

Hundreds of instances can be build and use in few minutes as and when required, which saves ample amount of hardware cost for any organizations and make them efficient to focus on their core business areas.

Currently AWS is present and providing cloud services in more than 190 countries.

Well-known for IaaS, but now growing fast in PaaS and SaaS.

Why AWS?

Low Cost: AWS offers, pay as you go pricing. AWS models are usually cheapest among other service providers in the market.

Instant Elasticity: You need 1 server or 1000's of servers, AWS has a massive infrastructure at backend to serve almost any kind of infrastructure demands, with pay for what you use policy.

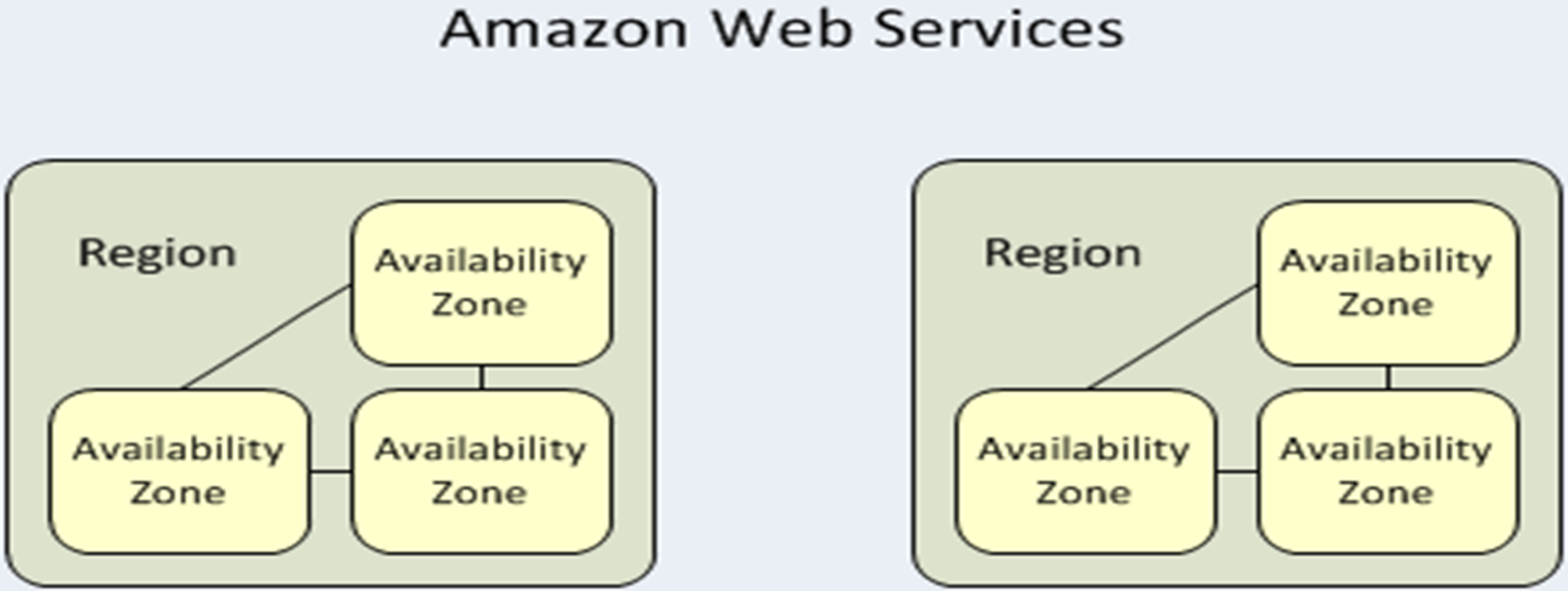
Scalability: Facing some resource issues, no problem within seconds you can scale up the resources and improve your application performance. This cannot be compared with traditional IT datacenters.

Multiple OS's: Choice and use any supported Operating systems.

Multiple Storage Options: Choice of high I/O storage, low cost storage. All is available in AWS, use and pay what you want to use with almost any scalability.

Secure: AWS is PCI DSS Level1, ISO 27001, FISMA Moderate, HIPAA, SAS 70 Type II passed. In-fact systems based on AWS are usually more secure than in-house IT infrastructure systems.

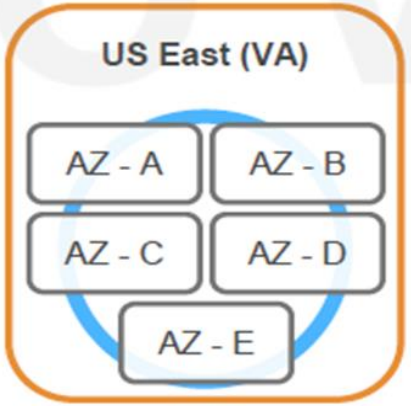
Amazon Web Services

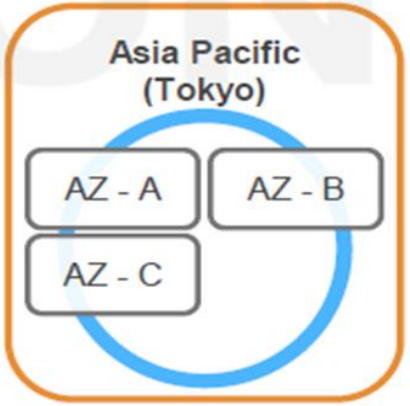


Amazon Web Services

At least 2 AZs 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





Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

Amazon Web Services

AWS Regions:
-Geographic Locations
-Consists of at least two Availability Zones(AZs)
-All of the regions are completely independent of each other with separate Power Sources, Cooling and Internet connectivity.

AWS Availability Zones
-AZ is a distinct location within a region
-Each Availability Zone is isolated, but the Availability Zones in a Region are connected through low-latency links.
-Each Region has minimum two AZ’s
-Most of the services/resources are replicated across AZs for HA/DR purpose.

Amazon Web Services

AWS Regions:

-Geographic Locations

-Consists of at least two Availability Zones(AZs)

-All of the regions are completely independent of each other with separate Power Sources, Cooling and Internet connectivity.

-This achieves the greatest possible fault tolerance and stability.

-There is a charge for data transfer between Regions.

-When you view your resources, you'll only see the resources tied to the Region you've specified.

-An AWS account provides multiple Regions so that you can launch Amazon EC2 instances in locations that meet your requirements. For example, you might want to launch instances in Europe to be closer to your European customers or to meet legal requirements.

-Resources aren't replicated across regions unless you do so specifically.

Amazon Web Services

AWS Availability Zones

-AZ is a distinct location within a region

-Each Availability Zone is isolated, but the Availability Zones in a Region are connected through low-latency links.

-Each Region has minimum two AZ’s

-Most of the services/resources are replicated across AZs for HA/DR purpose.

-While launching instance you should specify an Availability Zone if your new instances must be close to, or separated from, your running instances.

Amazon Web Services

Current:

22 AWS Regions

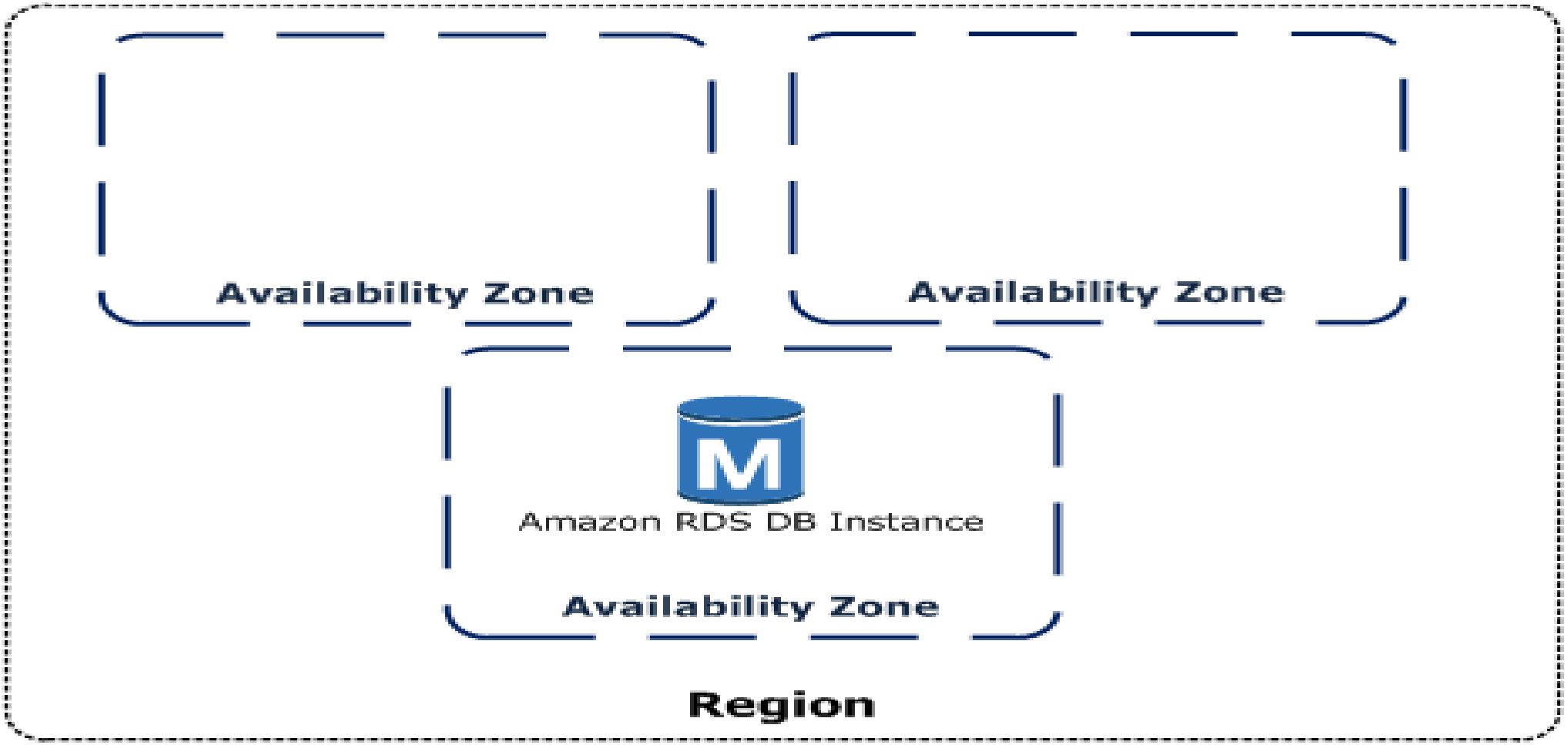
69 AZs

Upcoming:

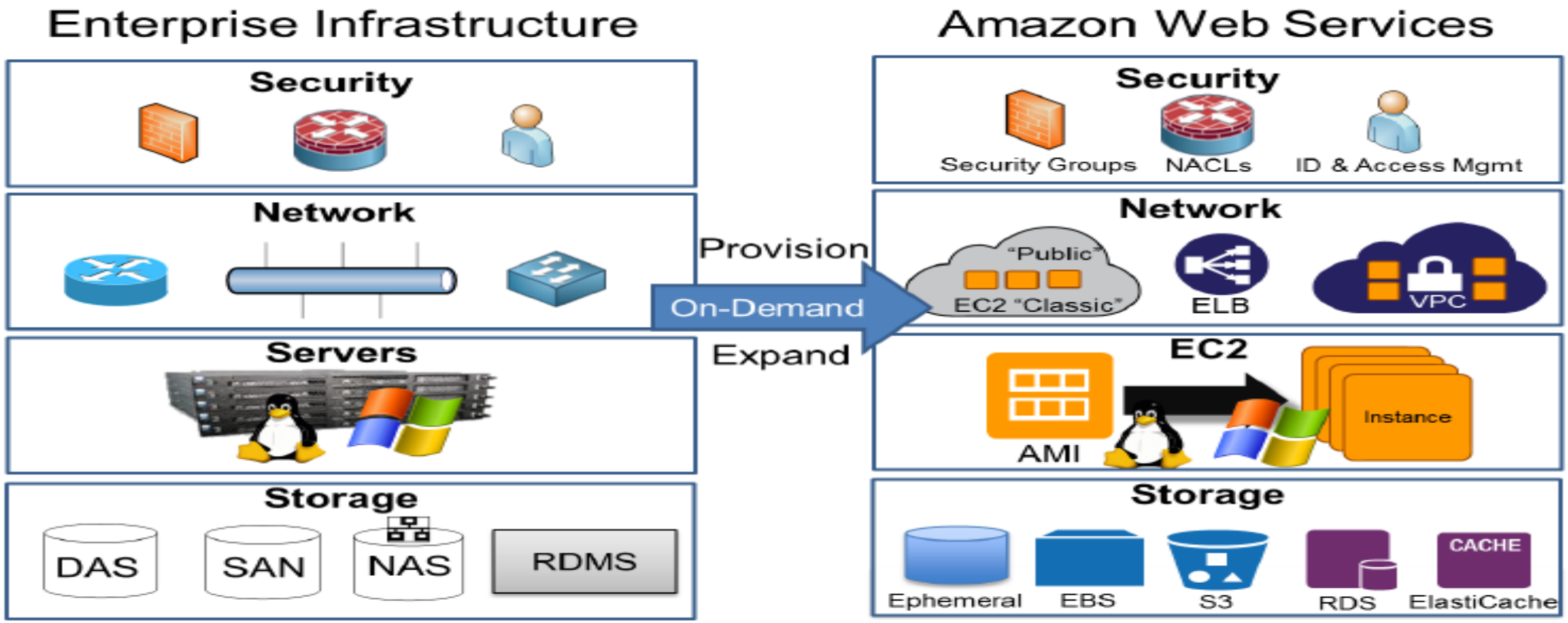
4 Regions

13 AZs

Amazon Web Services



AWS


Enterprise Infrastructure
Security
Network
Servers
Storage
DAS
SAN
NAS
RDMS
Provision
On-Demand
Expand
Amazon Web Services
Security
Security Groups
NACLs
ID & Access Mgmt
Network
"Public"
EC2 "Classic"
ELB
VPC
EC2
AMI
Instance
Storage
Ephemeral
EBS
S3
RDS
CACHE
ElastiCache

Knowledge Checks

- To Achieve HA in AWS, my Servers should be in different (Racks, Datacenters, Availability Zones, Regions)?

- To Achieve DR/High Durability, where should I have by Server backup (Different AZ or Different Regions)?

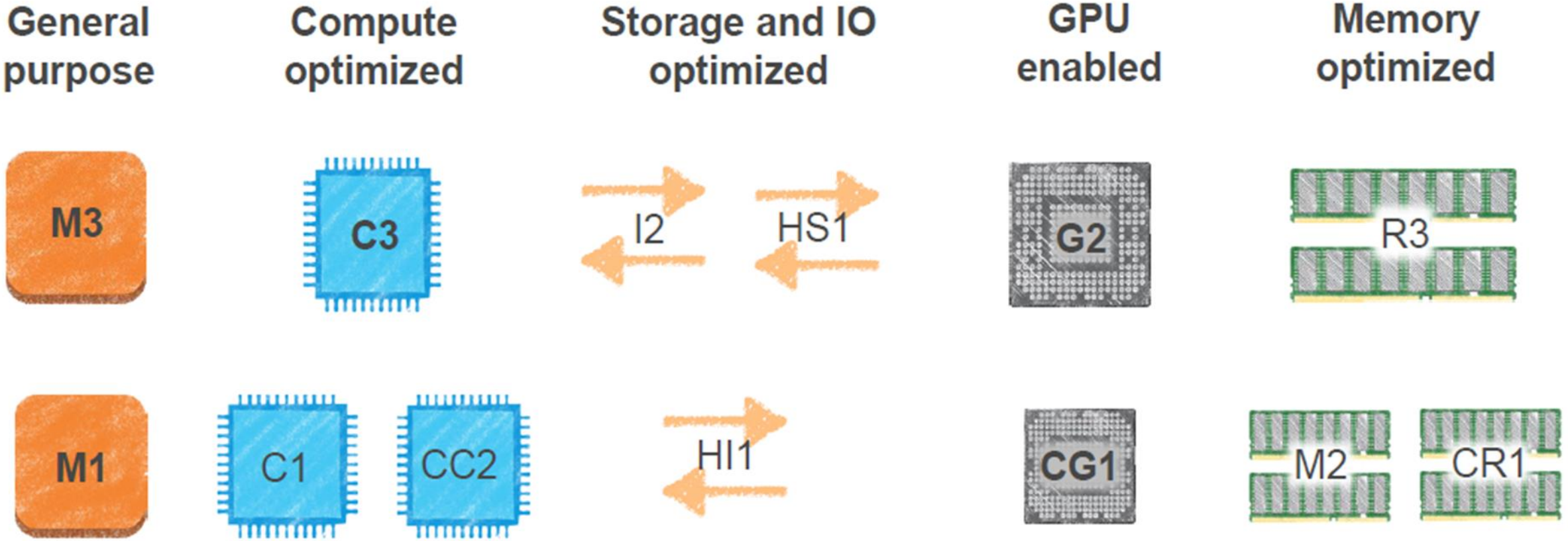
AWS
Compute Services

AWS Elastic Compute Cloud

- Amazon EC2 stands for Elastic Compute Cloud, and is the Primary AWS web service.
- Provides Resizable compute capacity
- Reduces the time required to obtain and boot new server instances to minutes
- There are two key concepts to Launch instances in AWS:
  - Instance Type
  - AMI

- EC2 Facts:
  - Scale capacity as your computing requirements change
  - Pay only for capacity that you actually use
  - Choose Linux or Windows OS as per need. You have to Manage the OS and Security of same.
  - Deploy across AWS Regions and Availability Zones for reliability/HA

AWS EC2


General purpose
Compute optimized
Storage and IO optimized
GPU enabled
Memory optimized
M3
C3
I2
HS1
G2
R3
M1
C1
CC2
HI1
CG1
M2
CR1

EC2 Security Group

Security Group is a Virtual Firewall Protection.

AWS allows you to control traffic in and out of your instances through virtual firewalls called security groups.

Security groups allow you to control traffic based on port, protocol, and source(inbound)/destination(outbound).

Security groups are associated with instances when they are launched. Every instance must have at least one security group. Though they can have more.

A security group is default deny.

AWS CLI

AWS CLI

AWS CLI is a command based utility to manage AWS resources

The primary distribution method for the AWS CLI on Linux, Windows, and macOS is pip, a package manager for Python that provides an easy way to install, upgrade, and remove Python packages and their dependencies

http://docs.aws.amazon.com/cli/latest/userguide/installing.html

Requirements

- Python 2 version 2.6.5+ or Python 3 version 3.3+
- Windows, Linux, macOS, or Unix
- Pip package should be present (else install python-pip)

Install AWSCLI: pip install awscli --upgrade --user
For Windows, directly download the Windows installer from CLI webpage

AWS CLI

Lets install an AWSCLI
https://aws.amazon.com/cli

aws --version

aws help

aws ec2 help / aws s3 help / aws <anysubcommand> help

Configure your default keys and region:

root@ip-172-31-28-145:~# aws configure
AWS Access Key ID [None]: ##########
AWS Secret Access Key [None]: ################
Default region name [None]: us-west-2
Default output format [None]:
root@ip-172-31-28-145:~#

AWS CLI

Check the details for all running instances using CLI

- aws ec2 describe-instances

Creation of an AWS Instance using CLI:

- aws ec2 run-instances help
- aws ec2 run-instances --image-id ami-76d6f519 --instance-type t2.micro --key-name test
- aws ec2 describe-instances
- aws ec2 stop-instances --instance-ids i-02e6b6c6c4dd3bbe0
- aws ec2 terminate-instances --instance-ids i-0297acea9e1b39a56