DEVOPS TRAINING SCHOOL

CLOUD ENGINEERING

AWS

PROJECTS

COURSE OBJECTIVES

- AWS Project1
- AWS Project2
- AWS Project3

AWS SERVICES

DevOps on AWS

1- what are the tools(native) used for the CI/CD in AWS?

CI: Aws code commit(like GitHub) AWS code build

CD:AWS code deploy AWS code artifact

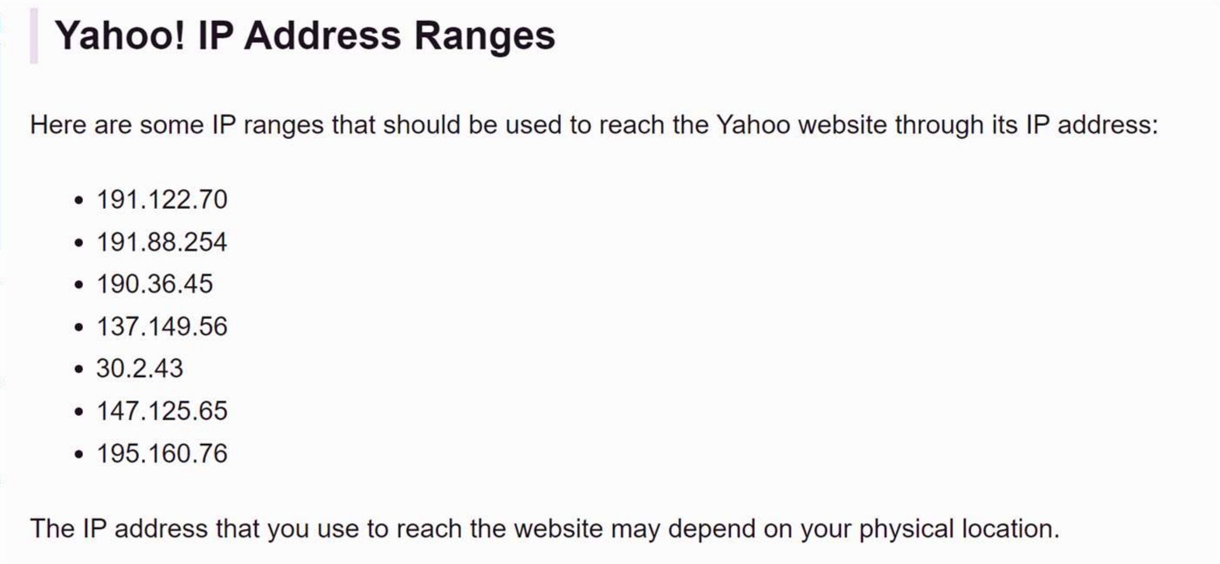
Or we can use this other tool for both CI/CD: AWS code pipeline

AWS SERVICES

AWS DNS

1- what is a DNS ?

Domain name service.



Yahoo! IP Address Ranges

Here are some IP ranges that should be used to reach the Yahoo website through its IP address:

- 191.122.70
- 191.88.254
- 190.36.45
- 137.149.56
- 30.2.43
- 147.125.65
- 195.160.76

The IP address that you use to reach the website may depend on your physical location.

It converts domain name into an ip address

The Domain Name System (DNS) is the phonebook of the Internet.

Humans access information online through domain names, like nytimes.com or espn.com.

Web browsers interact through Internet Protocol (IP) addresses.

DNS translates domain names to IP addresses so browsers can load Internet resource

AWS SERVICES

AWS DNS

1- what is an AWS DNS ?

It is called Route 53

2- what does route 53 do?

Domain registration/hosted zone/healthcheck/traffic flow

3- what are the route 53 routing policies?

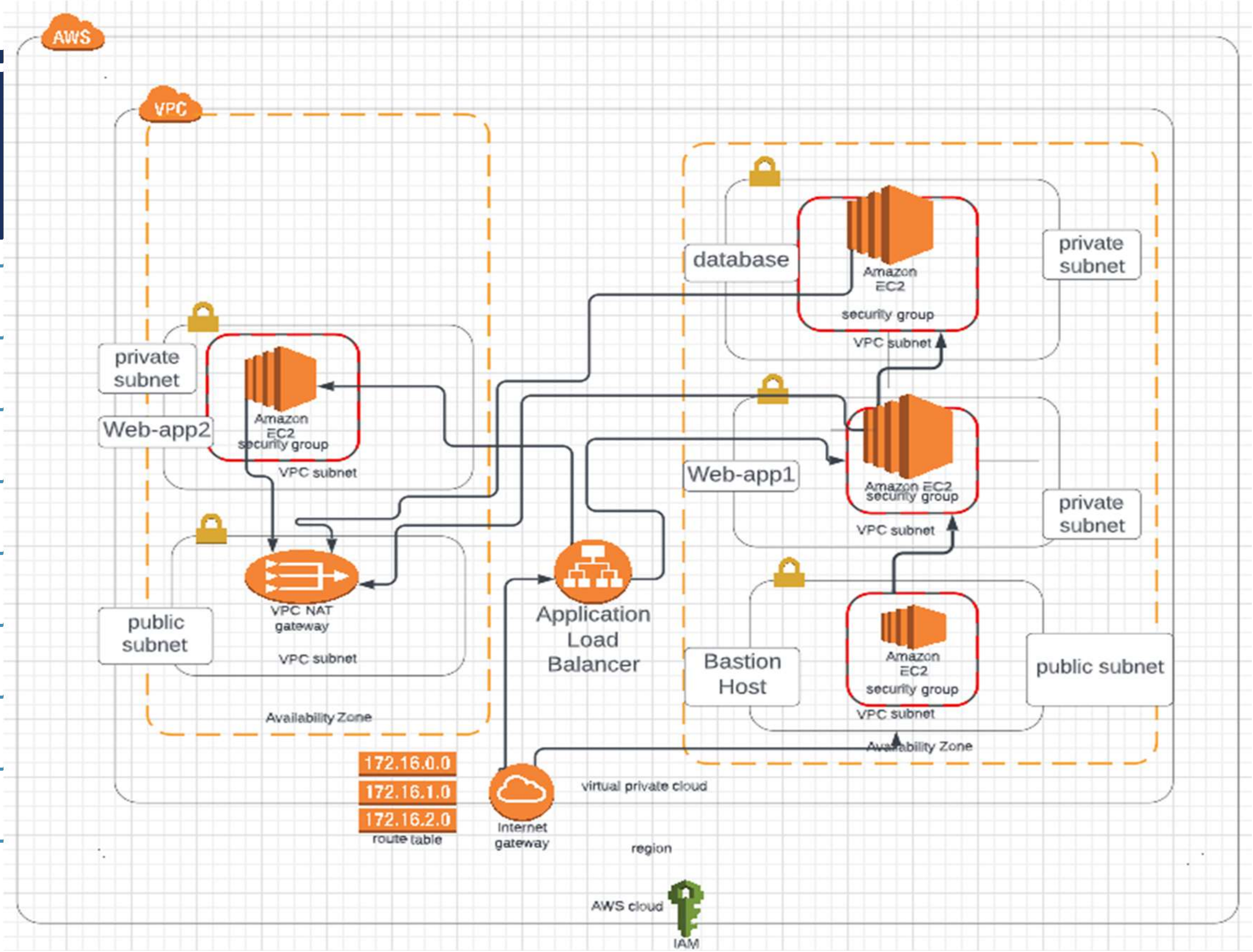
Simple /failover/geolocation/geoproximity/latency/weighted

4- What are the type of record types?

A record/ CNAME record / AAAA record / NS / …more

AWS SERVICES

Project4:



AWS SERVICES

Project4:

The project is a 3-tier application

We don’t want the webserver and database to be accessible directly from the outside.

We want them to be in private network

For high availability , we will create the exact copy of the webserver to another AZ.

To access them , we will create a bastion host in a public subnet that can be accessible only through SSH

The bastion host or jump server will have direct access to internet(public ip address)

To access internet , we will use an Internet gateway , associated with a router(route tables)

To access application from the webserver, we will use a load balancer.

The load balancer will be connected to the IGW directly through port 80 and 443

For our private servers to access internet for installing applications or patching(update; for example , yum update –y), we will use

Network address translation in another Az and its own subnet.

AWS SERVICES

Points to remember

* Subnets within a VPC cannot have overlapping IP ranges

* Choose   RFC 1918 range for your VPC CIDR range

10.0.0.0/16

172.16.0.0/16

192.168.0.0/16

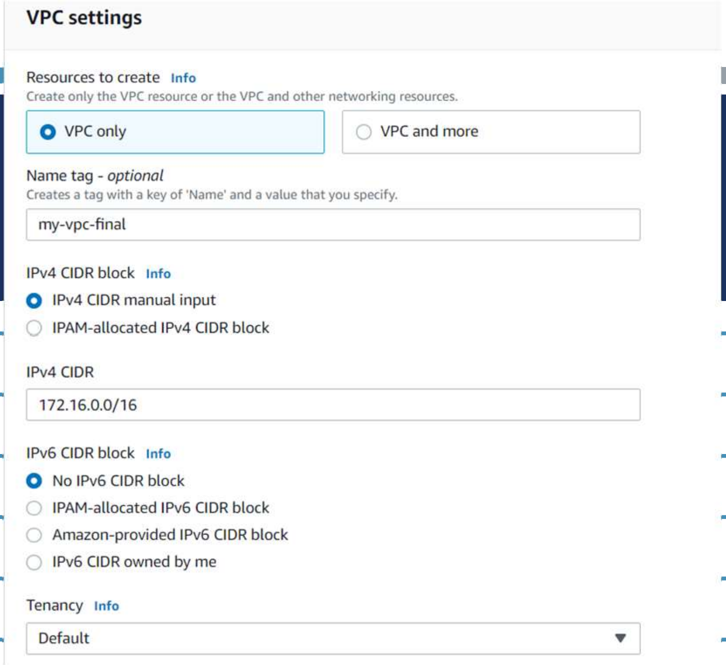
We already used the 10.0.0.0/16 a lot , this time we will use 172.16.0.0/16

We cannot modify or change the CIDR block of your VPC after creating it.

Let's jump and create all our services

AWS SERVICES

Step1: create our VPC with CIDR=172.16.0.0/16



VPC settings

Resources to create Info
Create only the VPC resource or the VPC and other networking resources.

● VPC only ○ VPC and more

Name tag - *optional*
Creates a tag with a key of 'Name' and a value that you specify.

my-vpc-final

IPv4 CIDR block Info

● IPv4 CIDR manual input
○ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

172.16.0.0/16

IPv6 CIDR block Info

● No IPv6 CIDR block
○ IPAM-allocated IPv6 CIDR block
○ Amazon-provided IPv6 CIDR block
○ IPv6 CIDR owned by me

Tenancy Info

Default ▼

Step2: create our subnets

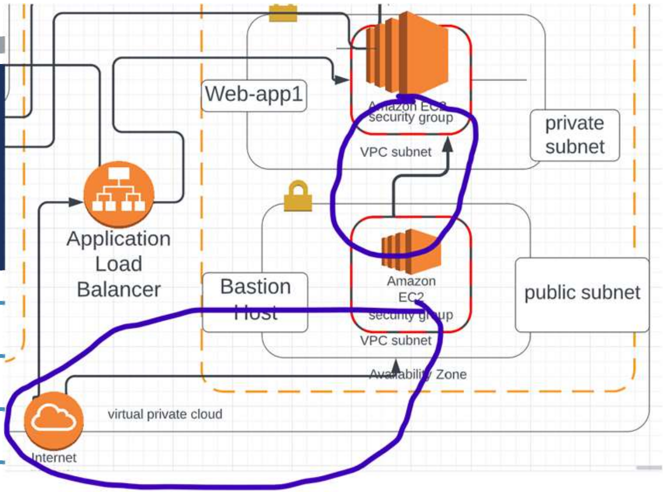
- Webserver1: name=web-subnet1, CIDR= 172.16.1.0/24
- Database server: name=db-subnet , CIDR=172.16.2.0/24
- Bastion host subnet: name = bastion-subnet, CIDR=172.16.3.0/24

All the above should be in the same availability zone.

- Webserver2: name=web-subnet2, CIDR=172.16.4.0/24 ,AZ= another AZ than the one above
- NAT : name=nat-subnet, CIDR=172.16.5.0/24

AWS SERVICES

Step3: Create an IGW(Internet gateway) named igw-01 and attach it to the VPC



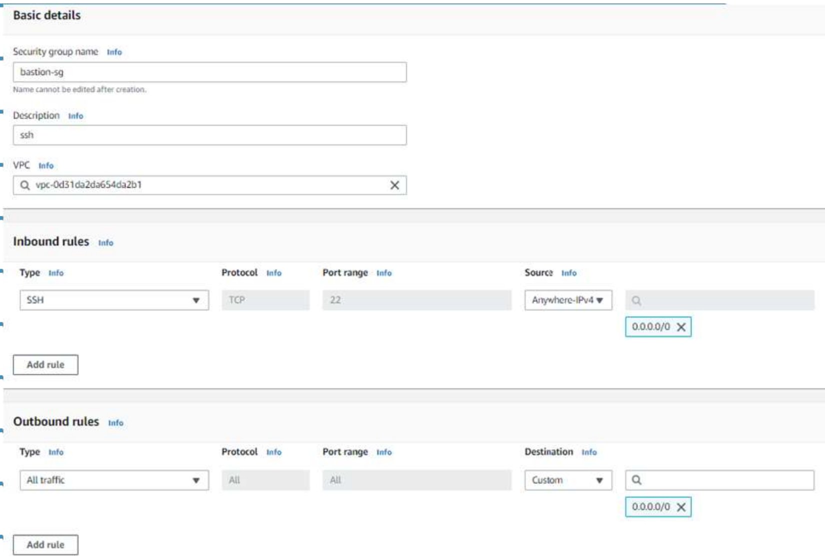
Step4: Create security group for each ec-2 instance(server).

* Bastion host:

1-Inbound: We will need to ssh to the bastion host only, so it needs to have access to internet.

2- Outbound: The bastion host needs to ssh to other servers as well

We will name it bastion-sg



Basic details

Security group name: bastion-sg

Name cannot be edited after creation.

Description: ssh

VPC: vpc-0d31da2da654da2b1

Inbound rules

Type	Protocol	Port range	Source	
SSH	TCP	22	Anywhere-IPv4	0.0.0.0/0

Outbound rules

Type	Protocol	Port range	Destination	
All traffic	All	All	Custom	0.0.0.0/0

* Load balancer:

1-Inbound: We will need to access our web pages from the internet.

2- Outbound: the load balancer will need to access webservers

We will name it lb-sg

Security group Name		Type	destination
lb-sg	Inbound:	http	all
	Outbound:	http	web-sg1

For the outbound we will come later after creating webservers security group

AWS SERVICES

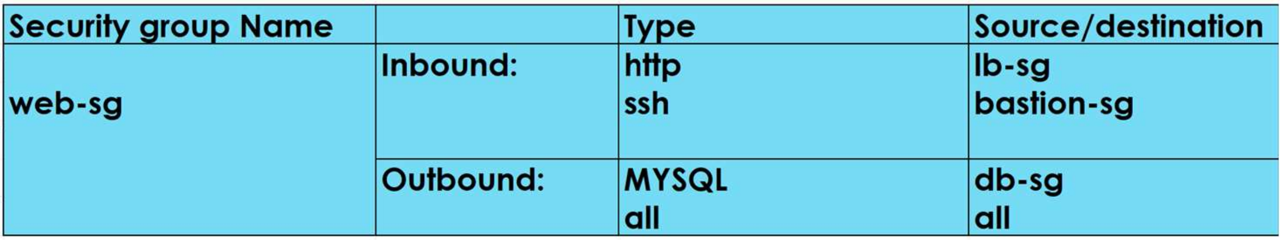
Step4: Create security group for each ec-2 instance(server).

*Webserver1 and 2:

1-Inbound: all webservers need to open http/https access for the loadbalancer and ssh for the bastion host

2- Outbound: the webservers need to access the database and the internet through the NAT

We will name it web-sg



Security group Name		Type	Source/destination
web-sg	Inbound:	http ssh	lb-sg bastion-sg
	Outbound:	MYSQL all	db-sg all

AWS SERVICES

Step4: Create security group for each ec-2 instance(server).

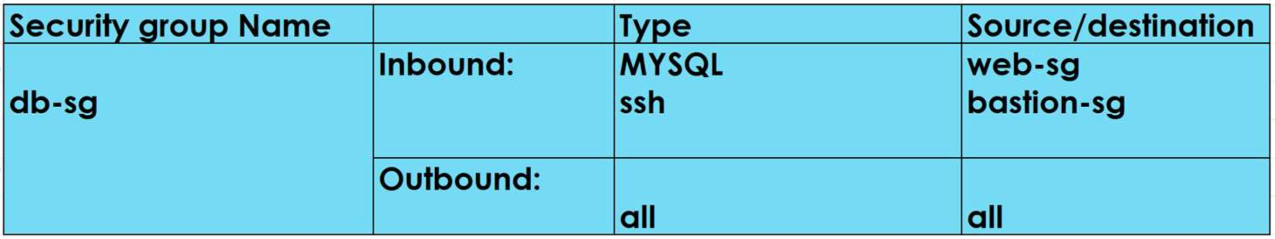
* database:

1-Inbound: all webservers need to access for the database and the bastion host needs to access the database through ssh.

2- Outbound: the database need to access internet through NAT

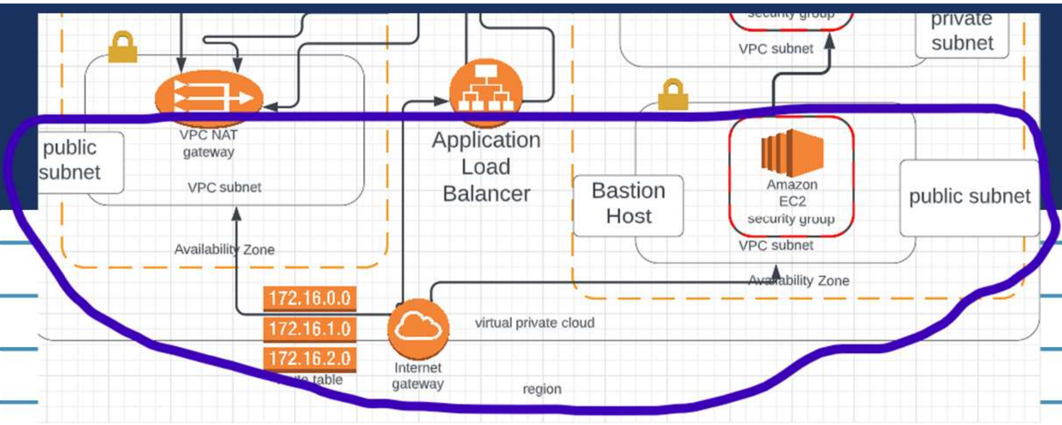
We will name it db-sg

Let fix the previous security groups



Security group Name		Type	Source/destination
db-sg	Inbound:	MYSQL ssh	web-sg bastion-sg
	Outbound:	all	all

AWS SERVICES



Step5: Create NAT name NAT-01

Subnet: nat-subnet

Connectivity type : public

Allocate elastic IP

Step6: Create routing

Let create route tables:

We will have two routes:

* Private subnets routes

* Public routes

Route Table	Priv-rt	
Routes	Destination	target
	172.16.0.0/16	local
	0.0.0.0/0	Nat
subnet associations	explicit subnet	
	db-subnet web-subnet1 web-subnet2	

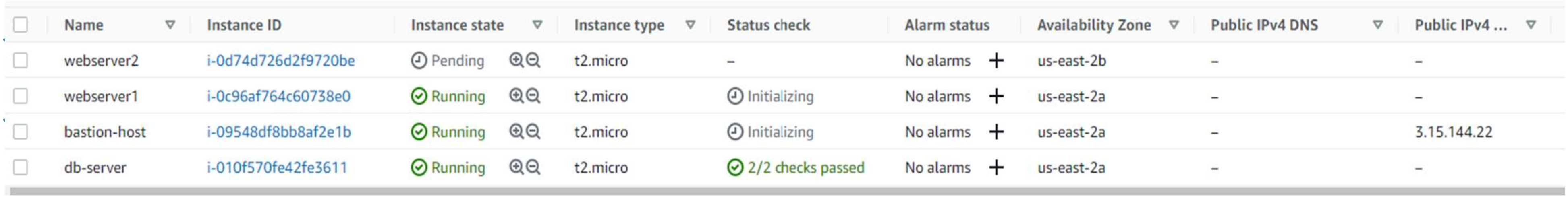
Route Table	Public-rt	
Routes	Destination	target
	172.16.0.0/16	local
	0.0.0.0/0	IGW
subnet associations	explicit subnet	
	bastion-subnet nat-subnet	

AWS SERVICES

Step7: create the ec-2 instances

* Bastion host with public IP address

* The other ec-2 instances are private (disable public ip address)



□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
□	webserver2	i-0d74d726d2f9720be	Pending	t2.micro	–	No alarms +	us-east-2b	–	–
□	webserver1	i-0c96af764c60738e0	✓ Running	t2.micro	Initializing	No alarms +	us-east-2a	–	–
□	bastion-host	i-09548df8bb8af2e1b	✓ Running	t2.micro	Initializing	No alarms +	us-east-2a	–	3.15.144.22
□	db-server	i-010f570fe42fe3611	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	us-east-2a	–	–

AWS SERVICES

Step8:

Ssh to bastion host

Then ssh to the webservers and install httpd

Ssh to database server and install mariadb and all requirements from this link

https://github.com/yannickeboo/LAMP-Stack

1- Ssh to Bastion:

Before ssh , we need to add our key to the known host

Cd into the director that has the key pair

# ssh-agent bash

# ssh-add keyname

check if the key pair is added: # ssh-add –l

Ssh to the bastion host: # ssh –A ec2-user@ipAddress

2- ssh to each other server:

Ssh ec2-user@private.ip.address for each server

AWS SERVICES

Step9: Creating load balancer

1- create target groups first .

- Go to services, ec2 , then click on target group
- Click on create target group , select instances
- Name the target group: tg-1, select the VPC & click on next
- Select the webserver instances & click on include as pending below
- Click on create target group

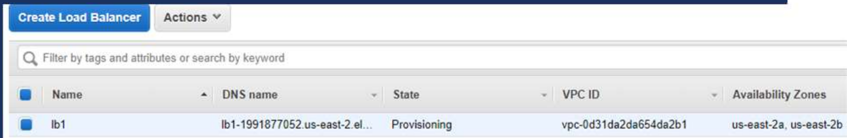
2- Create a load balancer with tg-1 as target group

- Go to services, ec2, then load balancers
- Click on create load balancer & select Application load balancer
- Click on create , give it a name lb-1 , it should be internet facing
- Select the VPC & two public availability zones(bastion host subnet & Nat subnet) & click on next

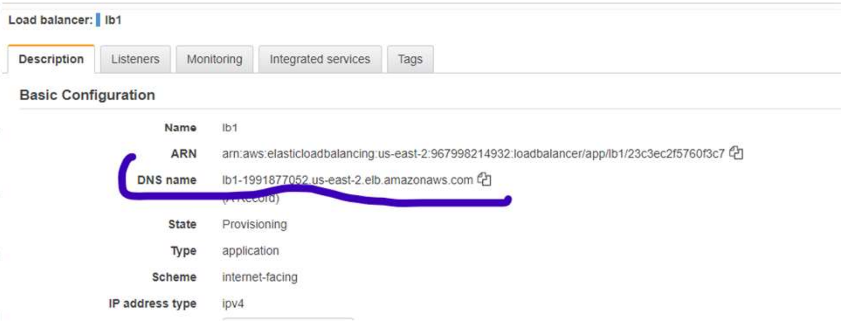
Click on next : configure security group

AWS SERVICES

- Click on next : configure security group and next again
- Click on select an existing security group and select the lb-sg & click on next
- On configure routing , select existing target group named tg-1
- Click on next , then next , & create



Name	DNS name	State	VPC ID	Availability Zones
lb1	lb1-1991877052.us-east-2.el...	Provisioning	vpc-0d31da2da654da2b1	us-east-2a, us-east-2b

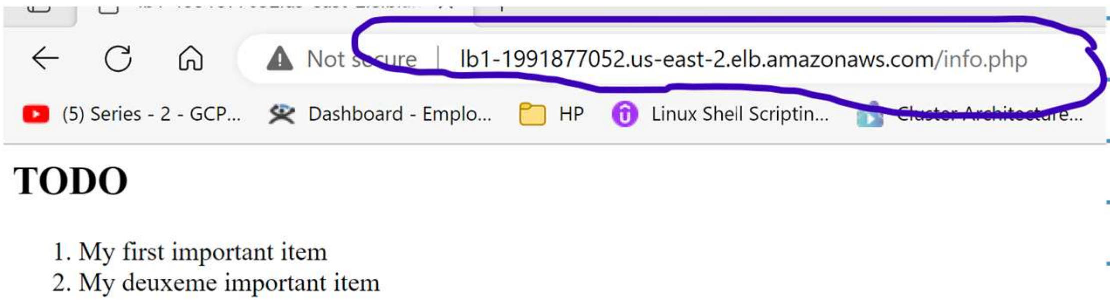


Basic Configuration

Name	lb1
ARN	arn:aws:elasticloadbalancing:us-east-2:967998214932:loadbalancer/app/lb1/23c3ec2f5760f3c7
DNS name	lb1-1991877052.us-east-2.elb.amazonaws.com (A Record)
State	Provisioning
Type	application
Scheme	internet-facing
IP address type	ipv4

2- How to access the load balancer?

Copy the DNS name and paste on the browser. Put info.php at the end



lb1-1991877052.us-east-2.elb.amazonaws.com/info.php

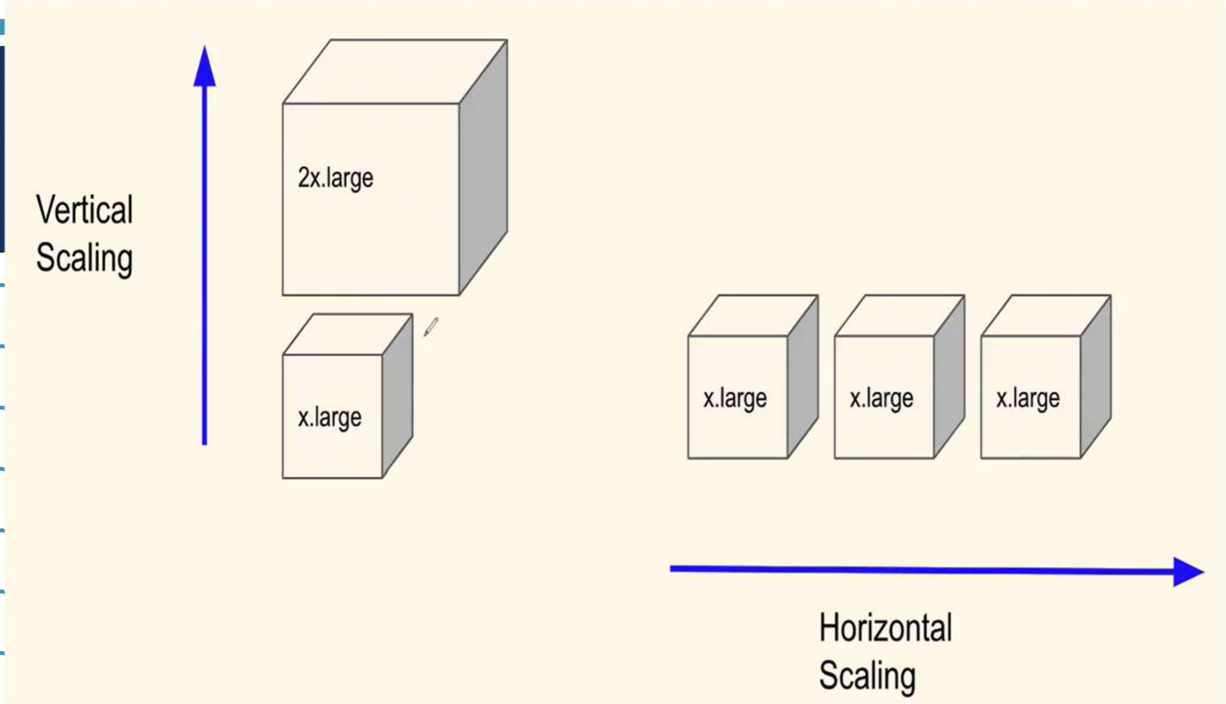
TODO

1. My first important item
2. My deuxeme important item

AWS SERVICES

Autoscaling

Autoscaling **provides users with an automated approach to increase or decrease the compute, memory or networking resources they have allocated, as traffic spikes and use patterns demand**



1- what are the types of Scaling?

We have horizontal auto scaling and vertical autoscaling.

2- How the vertical autoscaling work?

It increases/decreases the memory , or cpu , or network resources when needed.

3- How horizontal auto scaling work?

A "horizontally scalable" system is one that can increase capacity by adding more computers to the system

AWS SERVICES

Project5: Autoscaling

Autoscaling **provides users with an automated approach to increase or decrease the compute, memory or networking resources they have allocated, as traffic spikes and use patterns demand**

Make sure that we are on EC2 Dashboard

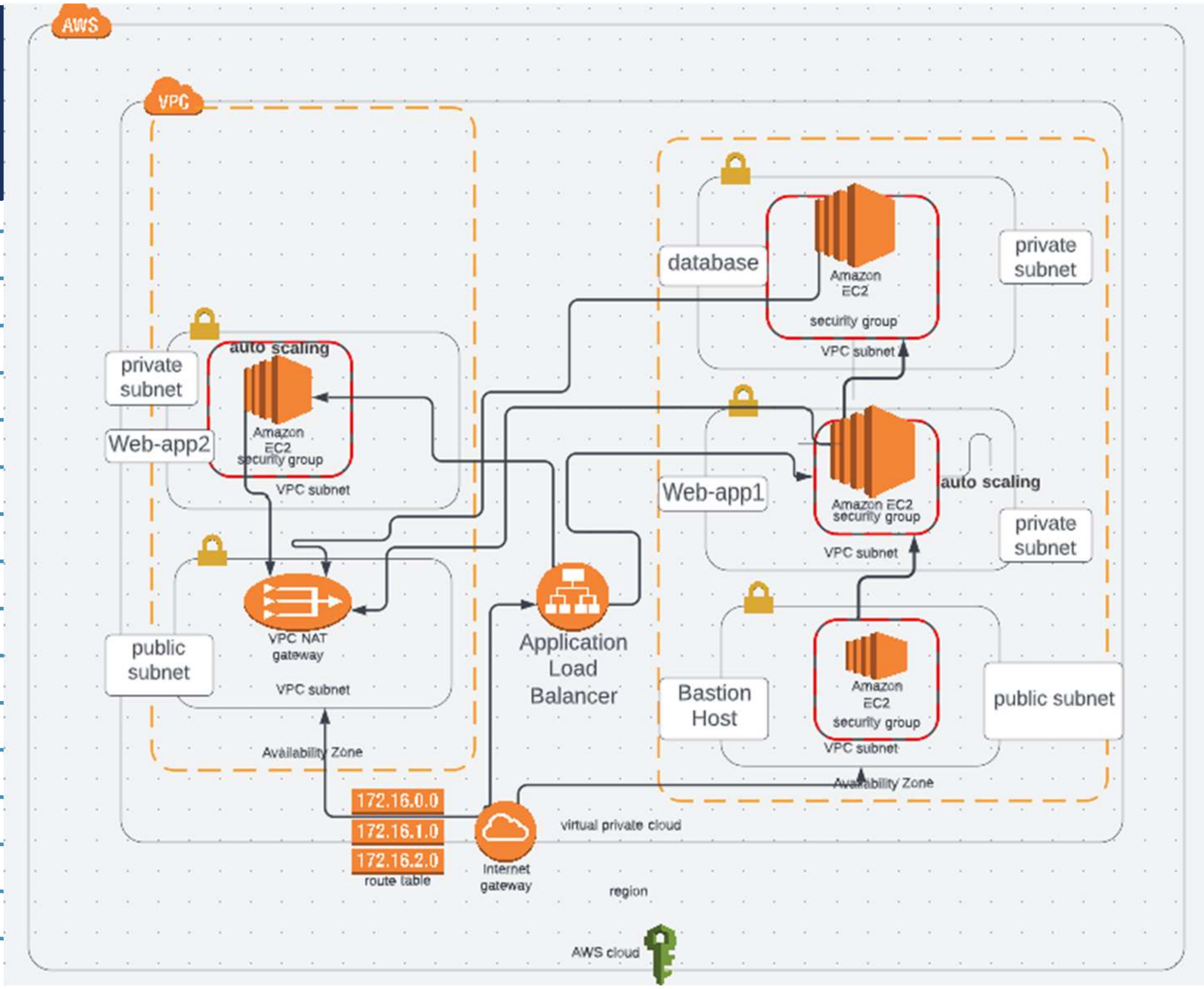
Step1: create an image of our webserver

Step2: Create a launch configurations

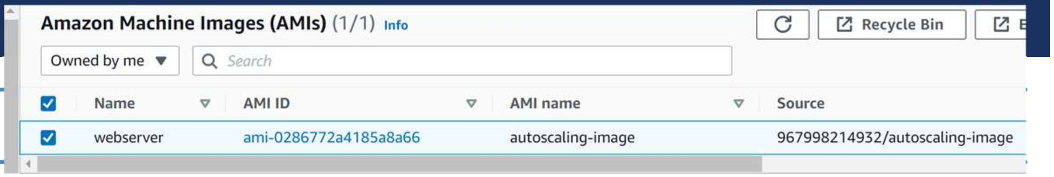
Step3: create an autoscaling group

Step4: verify that target group is created

Step5: verify all



AWS SERVICES



Amazon Machine Images (AMIs) (1/1) Info

☑	Name	AMI ID	AMI name	Source
☑	webserver	ami-0286772a4185a8a66	autoscaling-image	967998214932/autoscaling-image

Project5: Autoscaling

Step1: create an image of our webserver

- Select the ec2 instance that we will us to create an image

- Click on Actions , then Image and templates, finally create image

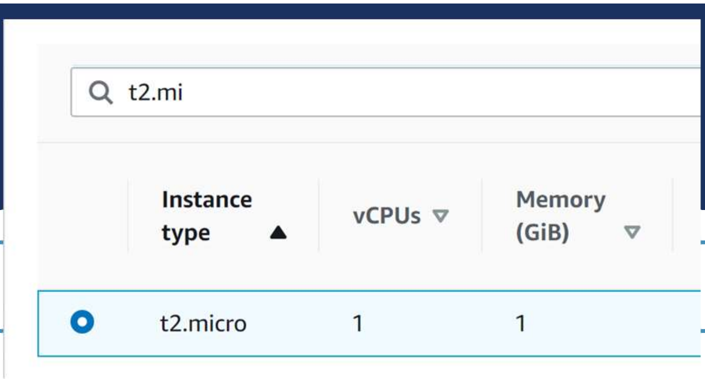
- Give the image a name: autoscaling-image

& description

- On tags, select the Tag image and snapshot together

Checking the image: click on AMIs

AWS SERVICES



t2.mi

	Instance type ▲	vCPUs ▽	Memory (GiB) ▽
●	t2.micro	1	1

Project5: Autoscaling

Step2: create a Launch configurations

- Click on launch configurations ,then create launch configuration

- Name: my-autoscaling-launch-config

- Select the AMI image that we created in the previous step: autoscaling-image

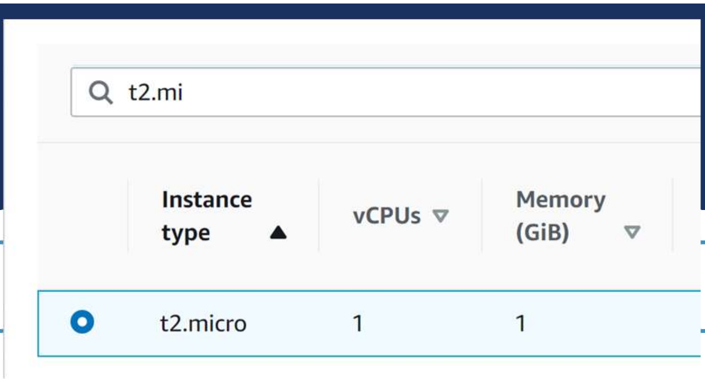
- Select the instance type: t2.micro 1 1

- On security group, select the existing security group & the same security group as the web server: web-tg

- Select existing key pair & select the key pair used to ssh the webserver

Click on the acknowledgement and click on create launch configuration

AWS SERVICES



🔍 t2.mi

	Instance type ▲	vCPUs ▽	Memory (GiB) ▽
●	t2.micro	1	1

Project5: Autoscaling

Step2: create a Launch configurations

- Click on launch configurations ,then create launch configuration

- Name: my-autoscaling-launch-config

- Select the AMI image that we created in the previous step: autoscaling-image

- Select the instance type: t2.micro 1 1

- On security group, select the existing security group & the same security group as the web server: web-tg

- Select existing key pair & select the key pair used to ssh the webserver

Click on the acknowledgement and click on create launch configuration

AWS SERVICES

Project5: Autoscaling

Step3: Create an autoscaling group

- Click on auto scaling groups and select create auto scaling group

Name: webserver-autoscaling-group

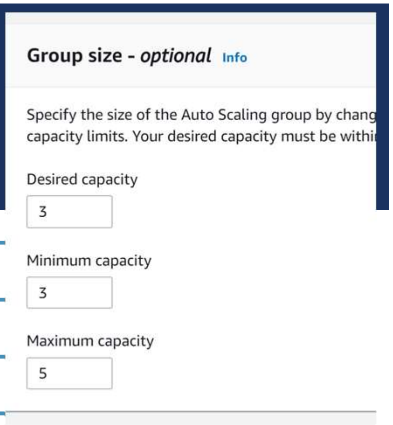
- Click on switch to launch configuration and select the launch configuration created at the previous step.

Then click on next

- ON network, Select the VPC and the AZ (availability zones and subnets)
- On Configure advanced options, select attach to a new load balancer
- On attach to new load balancer, we will select Application load balancer
- name: autoscaling-lb
- Select internet-facing
- On listeners and routing click on create a target group
- New target group name: autoscaling-tg

Click on next

AWS SERVICES



Group size - *optional* Info

Specify the size of the Auto Scaling group by chang
capacity limits. Your desired capacity must be withi

Desired capacity

3

Minimum capacity

3

Maximum capacity

5

Project5: Autoscaling

Step3: Create an autoscaling group

On group size and scaling policies,

We will select

* Desired capacity= 3

* Minimum capacity=3

* Maximum capacity=5

On scaling policies, click on target tracking scaling policies and

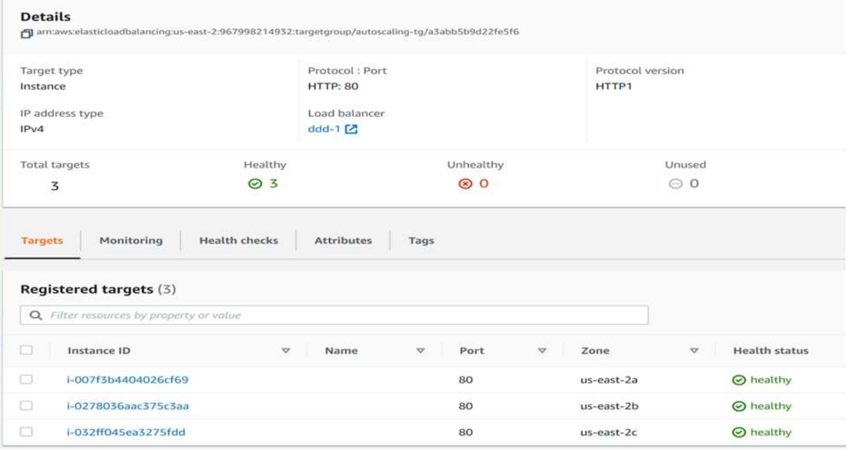
select the metric type you want the autoscaling the work with

- We will select CPU as metric type, and target value=70 , meaning when a Web Server hits 70% of CPU usage ,

a new ec2-instance will be created.

- Click on next , & next , & next ,then create auto scaling group

AWS SERVICES



Details

arn:aws:elasticloadbalancing:us-east-2:967998214932:targetgroup/autoscaling-tg/a3abb5b9d22fe5f6

Target type
Instance

Protocol : Port
HTTP: 80

Protocol version
HTTP1

IP address type
IPv4

Load balancer
ddd-1

Total targets	Healthy	Unhealthy	Unused
3	✓ 3	⊗ 0	0

Registered targets (3)

□	Instance ID	Name	Port	Zone	Health status
□	i-007f3b4404026cf69		80	us-east-2a	✓ healthy
□	i-0278036aac375c3aa		80	us-east-2b	✓ healthy
□	i-032ff045ea3275fdd		80	us-east-2c	✓ healthy

Project5: Autoscaling

Step4: verify the target group is created

- Click on target groups and verify that we have a target with autoscaling-tg name.

Step5:Verify that we have a load balancer working properly

- Click on Load balancer , then we should see a load balancer working properly

Let very that we have autoscaling working:

- Go to auto scaling group and click on the name of our auto scaling group , we should see all the details about the group
- Go to target groups and click on the target group name , we should see all the instances created by the autoscaling
- Go to ec2 dashboard, running instances , we should see and the newly created instances by auto scaling

Please don’t forget to delete the auto scaling and the vm after this

AWS SERVICES

Project7:

Create a S3 bucket

The name should be unique

Create a user and give him access to the S3 bucket only

AWS SERVICES

Interview questions and answers:

https://mindmajix.com/aws-interview-questions

https://intellipaat.com/blog/interview-question/amazon-aws-interview-questions/

AWS SERVICES

Project8:VPC peering

We will create two VPCs

The first want will have the webserver instance

The second one will have the database

Our goal is to peer the VPCs so that the webserver

can talk with the database

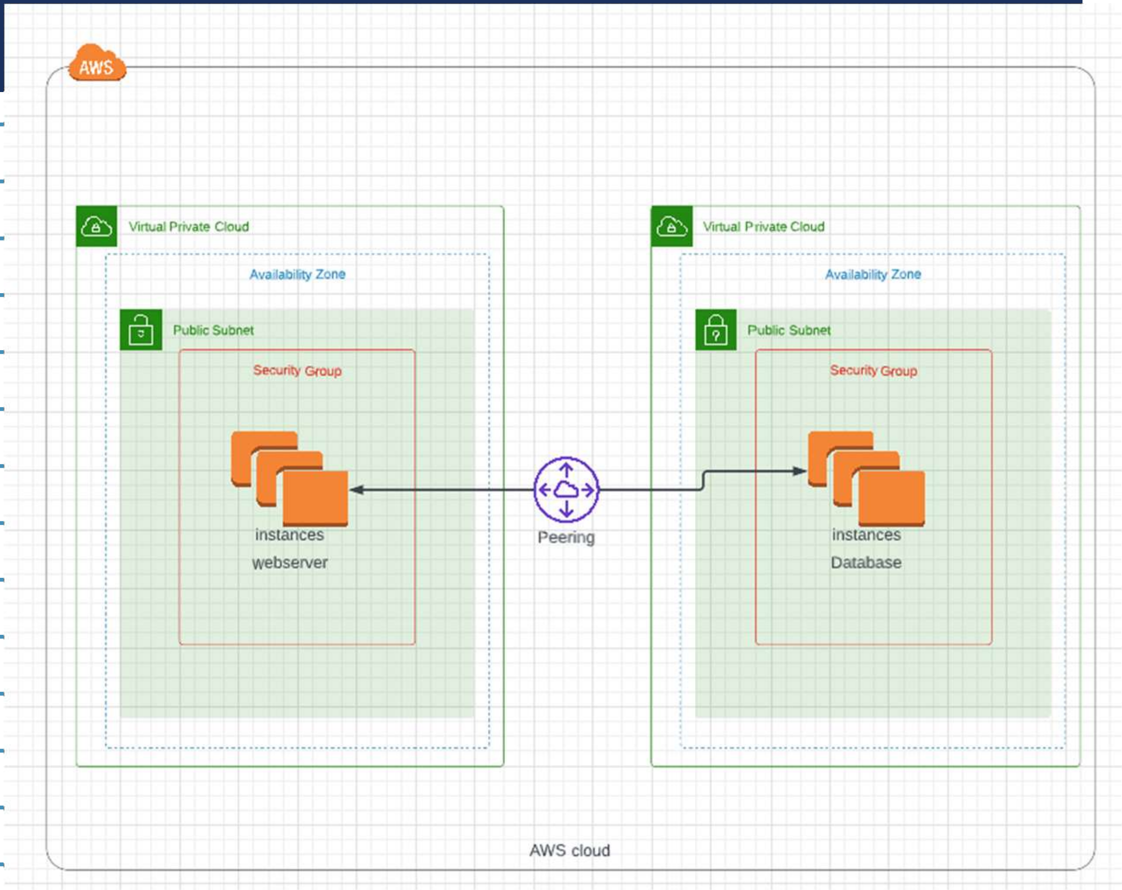
VPC peering enables routing using private IP addresses

The two VPCs cannot have overlapping IP address ranges

Please , use default VPC so you don’t have to go through all the

Process.

DUE DATE 07/13/2022



AWS SERVICES

Project8:VPN client

Create an AWS Client VPN so we can connect safely to our VPCs

Due date : 07/27/2022

AWS SERVICES

Project9: Application migration

AWS SERVICES

Projects:

RESEARCH ABOUT SNS, ELASTICACHE,