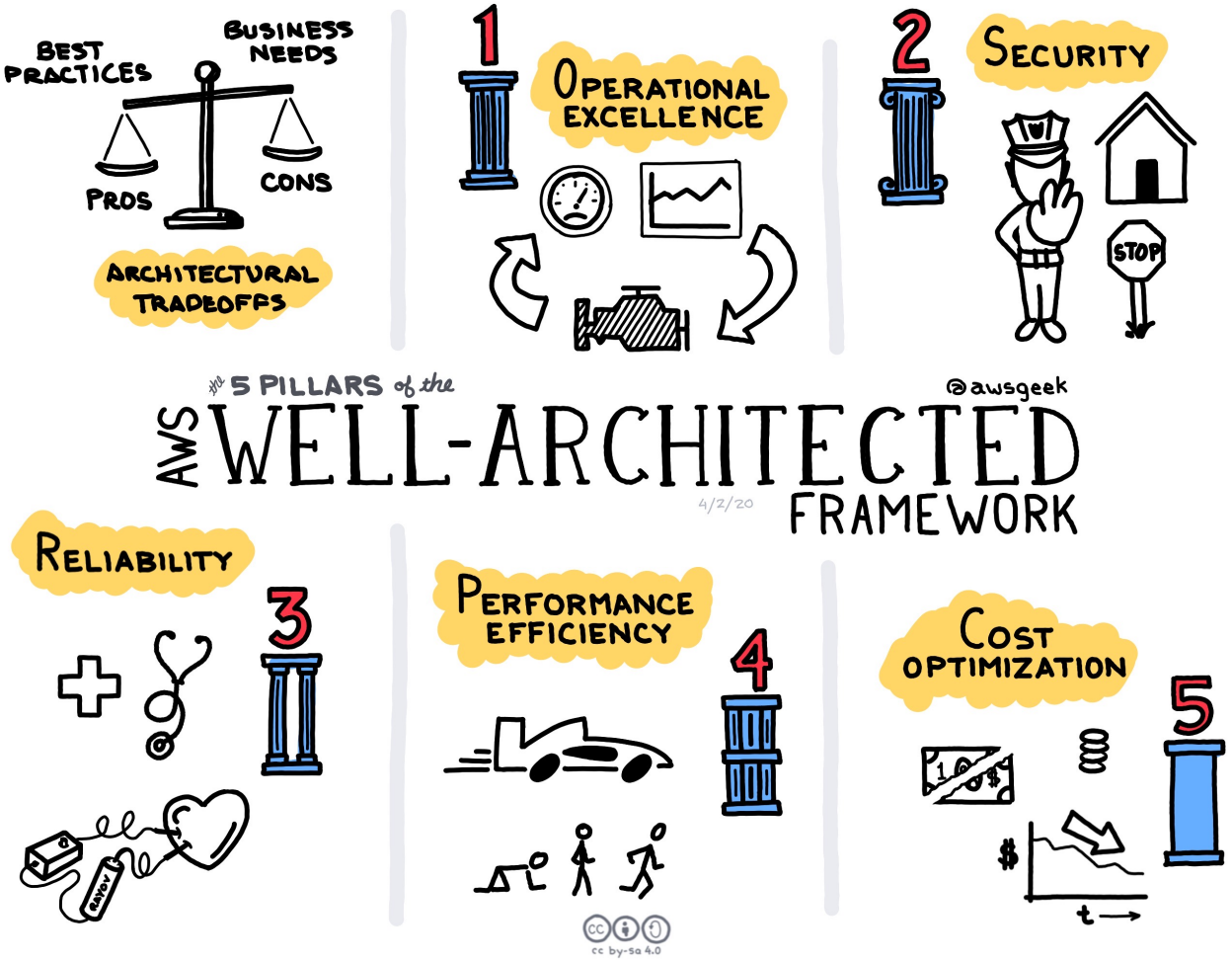
AWS02: AWS Core Services

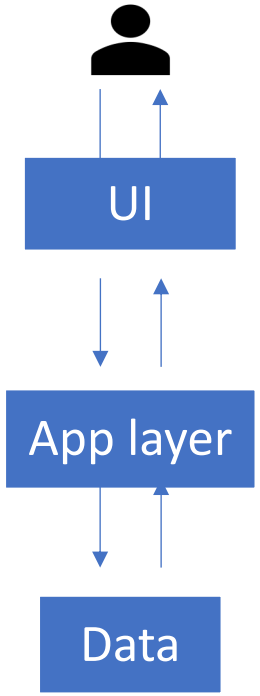


Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

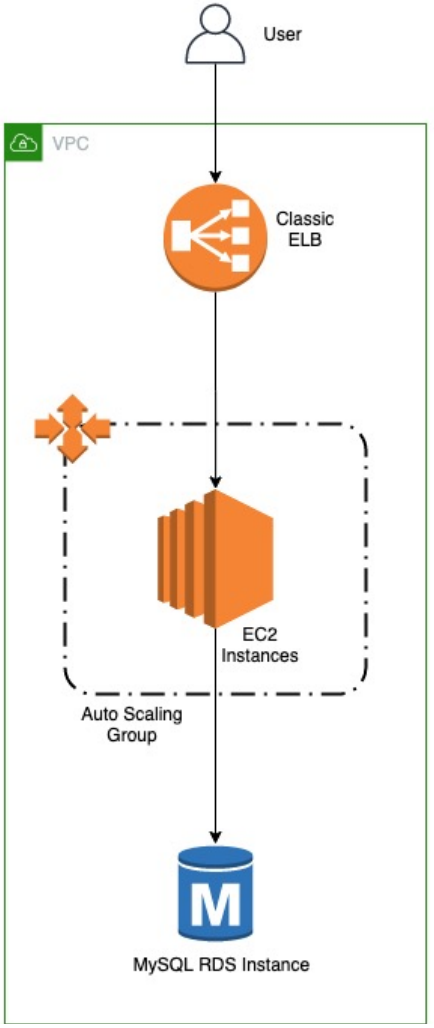
N-tiered Architecture of Web Applications

- tiers based on resource functions
- advantages
  - Flexibility to add, remove or upgrade each tier independently
  - Easy scalability based on usage
- three-tiered architecture
  - Presentation (UI)
  - Application processing
  - Data storage and management
- layers interact using APIs

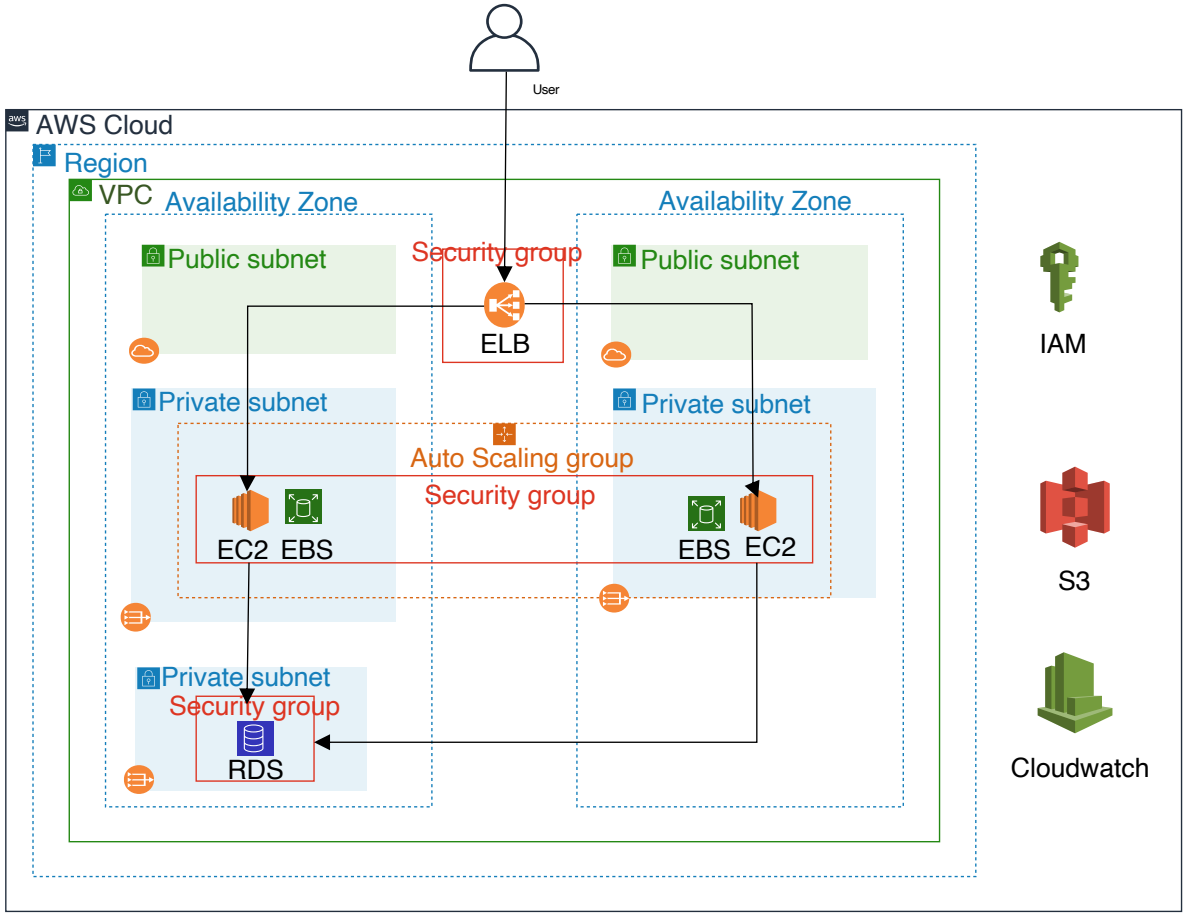


AWS Three-tier Architecture

- Client
- Server
  - load balancer distributes client requests for high-availability
  - cluster of web (application) servers
    - serve user requests received from the load balancer
    - deployed as an Auto Scaling group
- Data -> database
  - Stores persistent data
  - Serves queries received from web servers



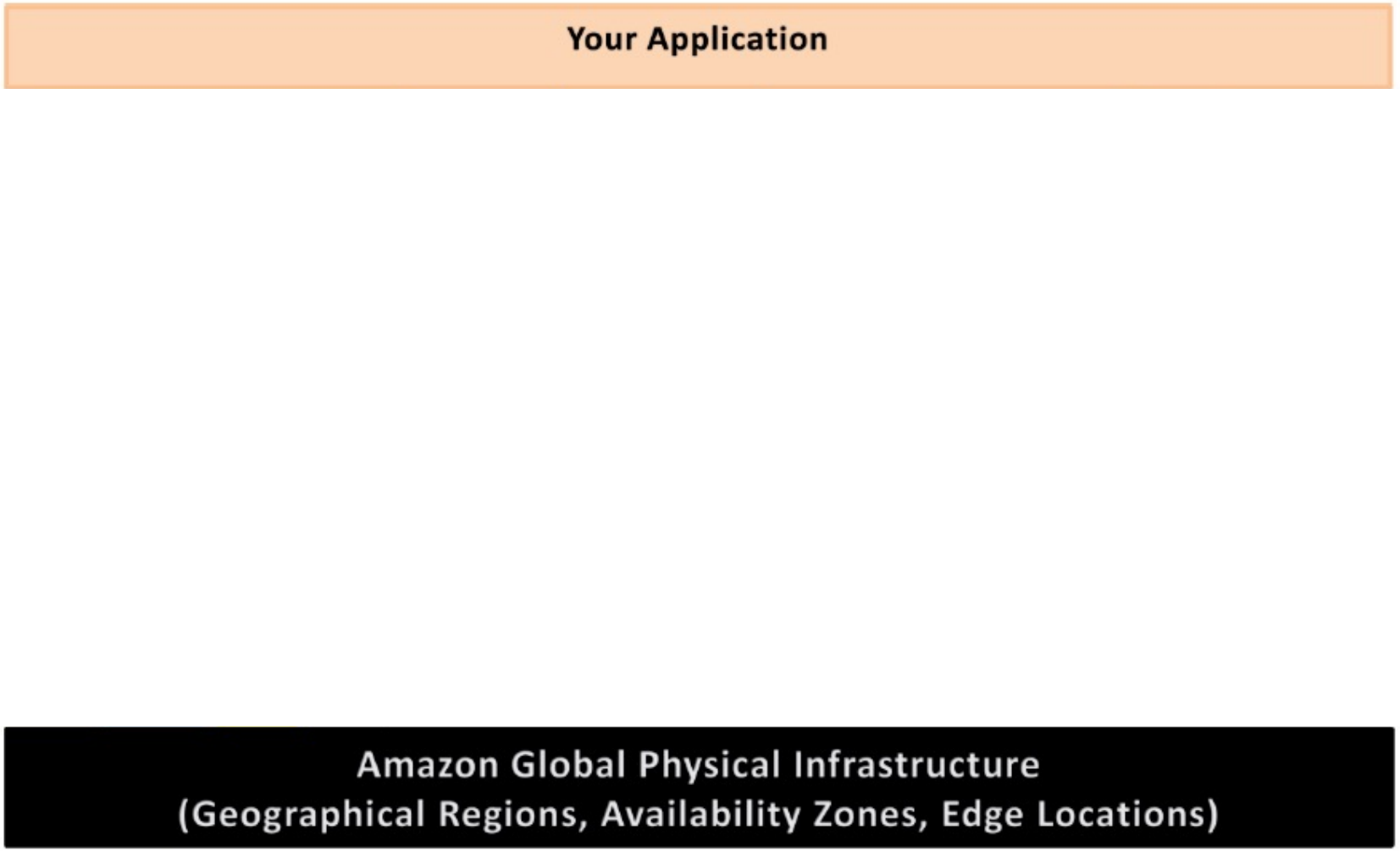
AWS Three-tier Architecture using Core Services



Components

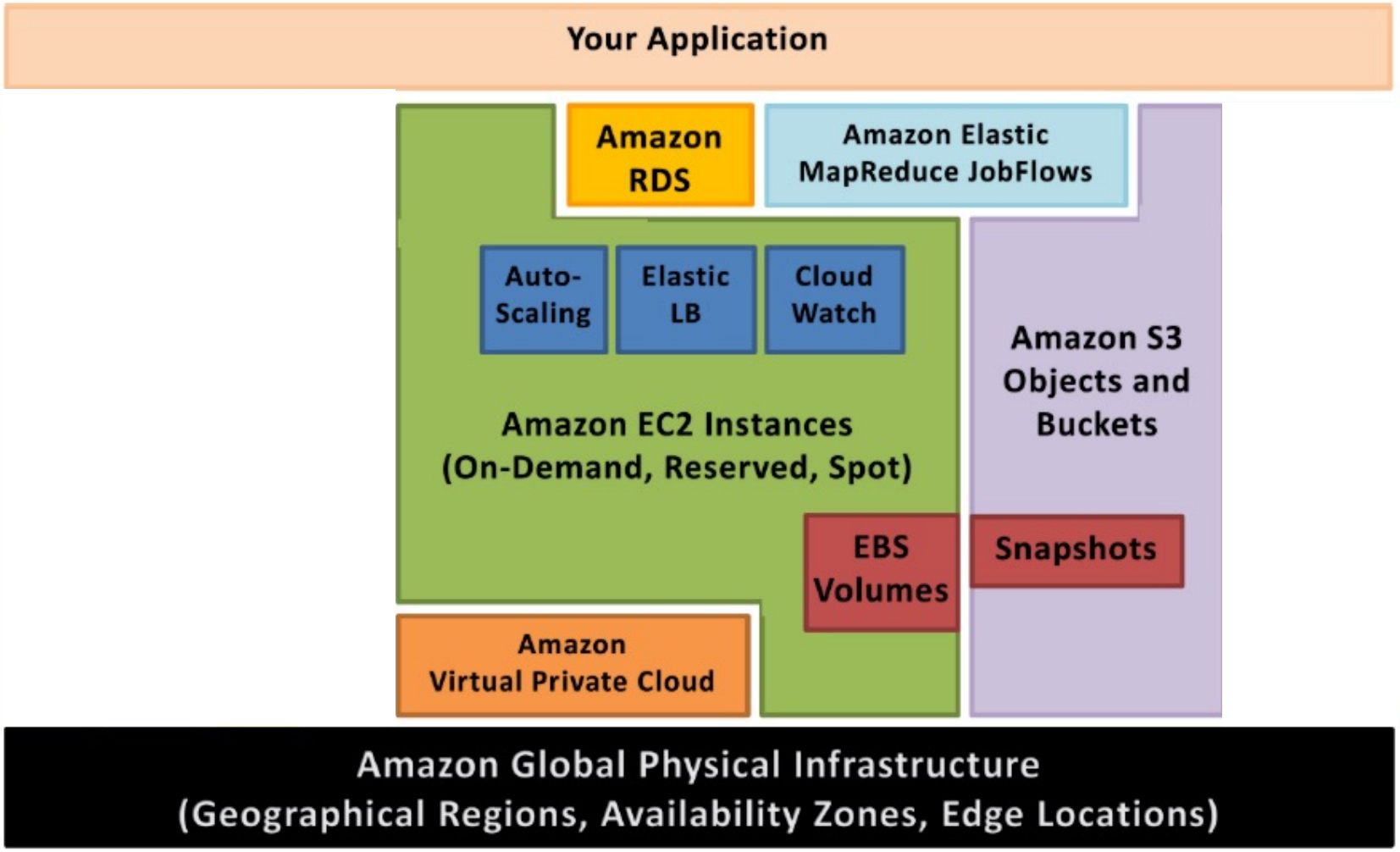
- Network
- Compute & Storage
- Elastic Load Balancer
- MySQL RDS Database
- S3 bucket
- Monitoring: Cloudwatch

Amazon Web Services (AWS) Ecosystem



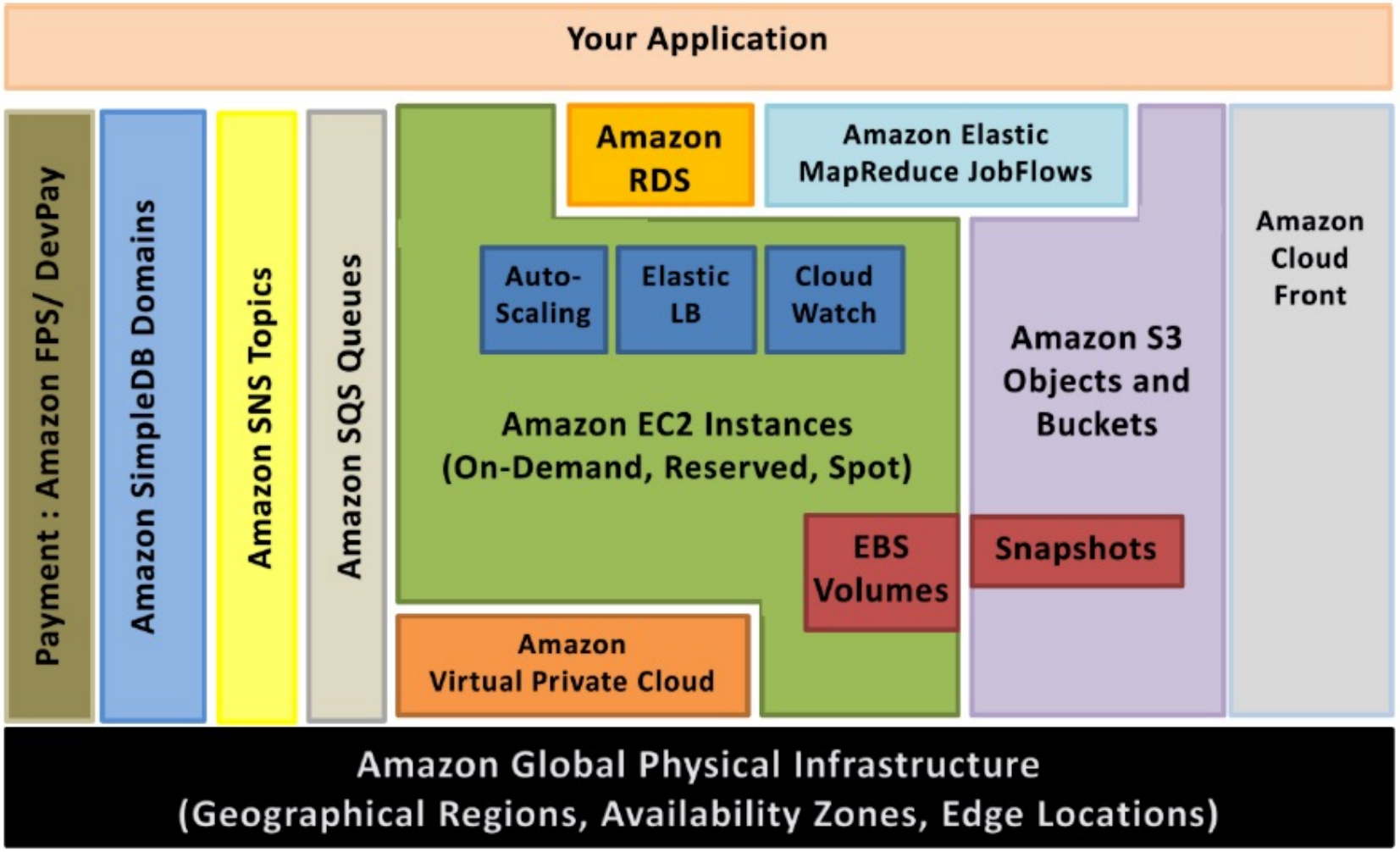
J. Varia, Architecting for the Cloud: Best Practices, Amazon Web Services, May 2010.

Amazon Web Services (AWS) Ecosystem



J. Varia, Architecting for the Cloud: Best Practices, Amazon Web Services, May 2010.

Amazon Web Services (AWS) Ecosystem



J. Varia, Architecting for the Cloud: Best Practices, Amazon Web Services, May 2010.

Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

Network Services - VPC

- What is Virtual Private Cloud (VPC)?

  "Amazon VPC enables you to launch Amazon Web Services (AWS) resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own datacenter, with the benefits of using the scalable infrastructure of AWS"

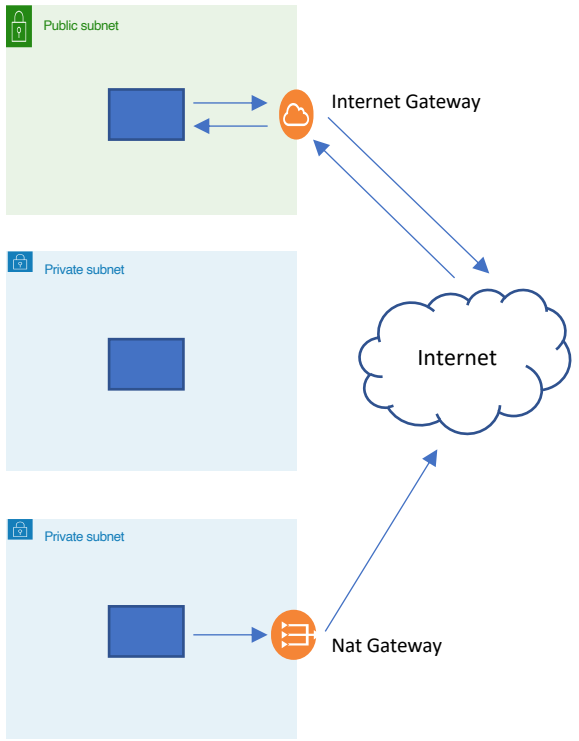
- Resembles an on-premise private corporate network
- In your VPC you can :
  - Define a CIDR block*

    Note: CIDR (Classless Inter-domain Routing) block - range of IPv4 addresses (e.g., 10.0.0.0/16)
  - Create multiple public and/or private subnets
  - Launch resources with your own private IP address
  - Define security groups, Access Control Lists, and Route Tables
- No additional charge for using a VPC

*read more - www.digitalocean.com/community/tutorials/understanding-ip-addresses-subnets-and-cidr-notation-for-networking

Network Services - Subnets

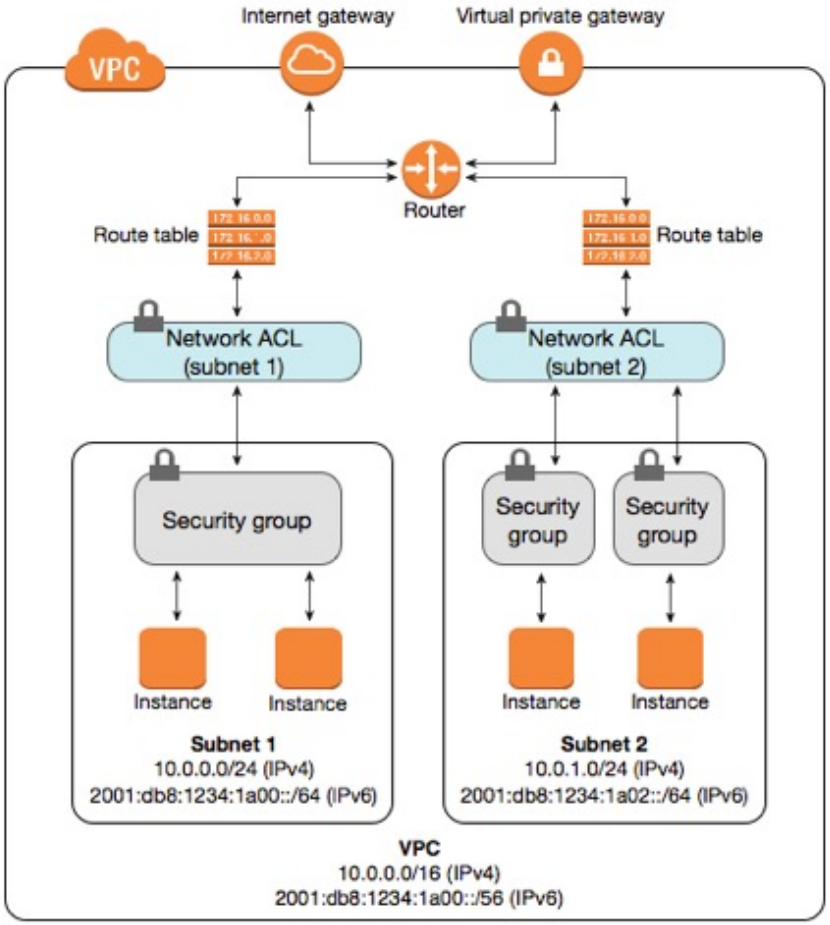
- Subnet is a range of IP addresses in a VPC, in one AZ
- Route table: Set of rules to determine where traffic is directed
- Types of Subnets:
  - Public Subnet
    - Network traffic directed to Internet Gateway
    - Resources have access to internet
    - Use for resources that are directly accessed
  - Private Subnet
    - By default, all new subnets are private
    - Resources do not have access to internet
    - To access internet, direct traffic to a NAT gateway



Network Services - Security

Two main features for network security:

- Security Groups
  - Act as firewall for EC2 instance
- Network Access Control List (NACL)
  - Act as a firewall for subnets



Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

Compute Services: EC2 Instances

- web service providing resizable computing capacity to build and host your software systems

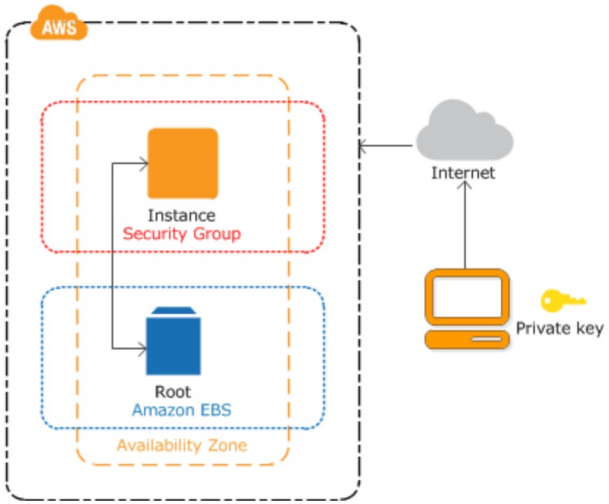
- an **instance** is a virtual server (VM) with a specified set of resources including CPU cycles, main memory, secondary storage, communication and I/O bandwidth

- user chooses:
  - *region* and *availability zone* where virtual server should be placed
  - an *instance type* from a menu of instance types

- pricing: on-demand, reserved, spot

EC2 Instances: Properties

- **Amazon Machine Image** (AMI): Software configuration template (OS, and applications)
- Instance Types: Compute Optimized, Memory optimized, General Purpose, etc.
- Instance Sizes: Based on vCPUs, Memory and Network Performance
- EBS (Elastic Block Storage) Root/Secondary Volumes: For persistent storage
- Key Pair: For SSH (Linux) or RDP (Windows) access
- Network configurations:
  - VPC and Subnet
  - Public and Private IPs
- IAM Role: AWS credentials
- Security Groups



Example: Instance Types

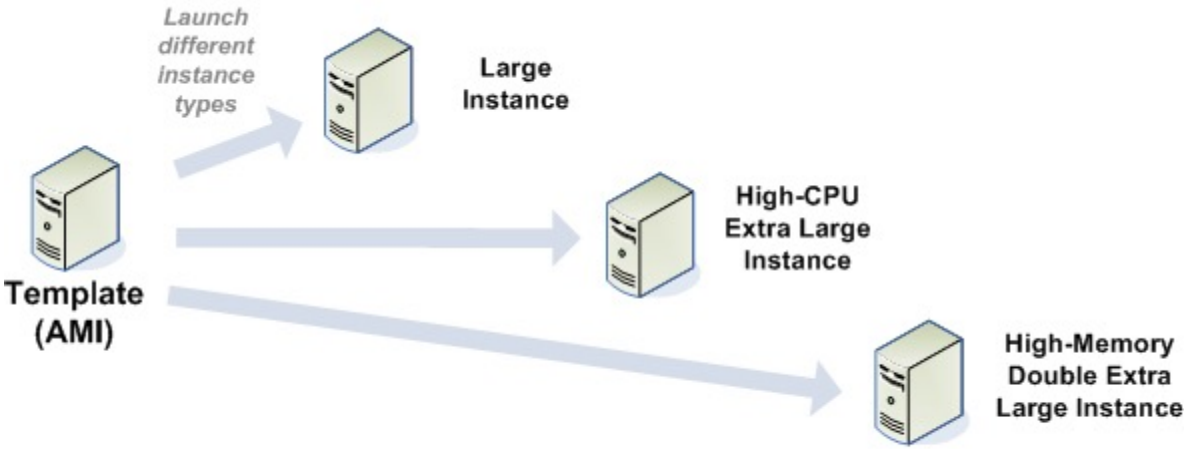
1. General Purpose: T2, M4
2. Compute Optimized: C4, C5
3. Memory Optimized: X1, R4
4. Accelerated Computing: F1, P2
5. Storage optimized: I3, D2

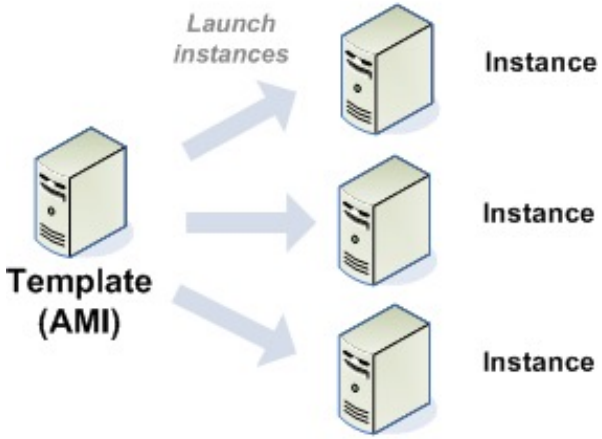
- Each instance type is further divided into subtypes based on features like the number of virtual CPUs (vCPUs), memory availability etc
- vCPU is defined as a hyper-thread of Intel Xeon E5-2670 v2 CPU (previously ECU (Elastic Compute Unit) – equivalent to CPU capacity of 2007 Opteron/Xeon processor)

Example: Instance Types

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t2.nano	1	Variable	0.5	EBS Only	$0.0059 per Hour
t2.micro	1	Variable	1	EBS Only	$0.012 per Hour
t2.small	1	Variable	2	EBS Only	$0.023 per Hour
t2.medium	2	Variable	4	EBS Only	$0.047 per Hour
t2.large	2	Variable	8	EBS Only	$0.094 per Hour
t2.xlarge	4	Variable	16	EBS Only	$0.188 per Hour
t2.2xlarge	8	Variable	32	EBS Only	$0.376 per Hour
m4.large	2	6.5	8	EBS Only	$0.1 per Hour
m4.xlarge	4	13	16	EBS Only	$0.2 per Hour
m4.2xlarge	8	26	32	EBS Only	$0.4 per Hour
m4.4xlarge	16	53.5	64	EBS Only	$0.8 per Hour
m4.10xlarge	40	124.5	160	EBS Only	$2 per Hour
m4.16xlarge	64	188	256	EBS Only	$3.2 per Hour

EC2: AMI


Launch different instance types
Large Instance
High-CPU Extra Large Instance
High-Memory Double Extra Large Instance
Template (AMI)


Launch instances
Instance
Instance
Instance
Template (AMI)

EC2 Instances: 3 Tasks

1. Launch an instance
   - a launched instance is secured by specifying a key pair and security group
   - to connect to your instance, you must specify the private key of the key pair that you specified when launching your instance

2. Connect to instance launched
   - AWS management console, CLI

3. Terminate your instance
   - deletes the instance and **stop incurring charges**

AWS Instances & Network Services

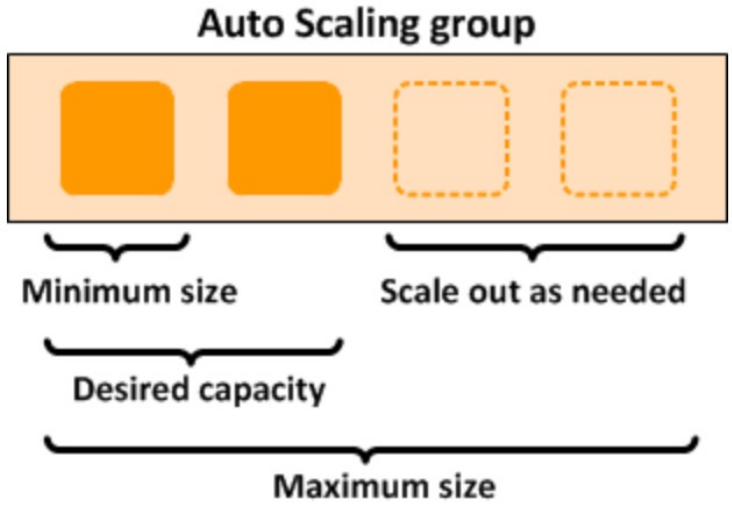
- when launched, an instance is provided with a DNS name that maps to a
  - *private IP address* → for internal communication within the internal EC2 communication network
  - *public IP address* → for communication outside the internal Amazon network, e.g., communication with the user that launched the instance and assigned for the lifetime of an instance

- an instance can request an *elastic IP address* (rather than a public IP), a static public IP address allocated to an instance from the available pool of the availability zone

- an elastic IP address is not released when the instance is stopped or terminated but released when no longer needed

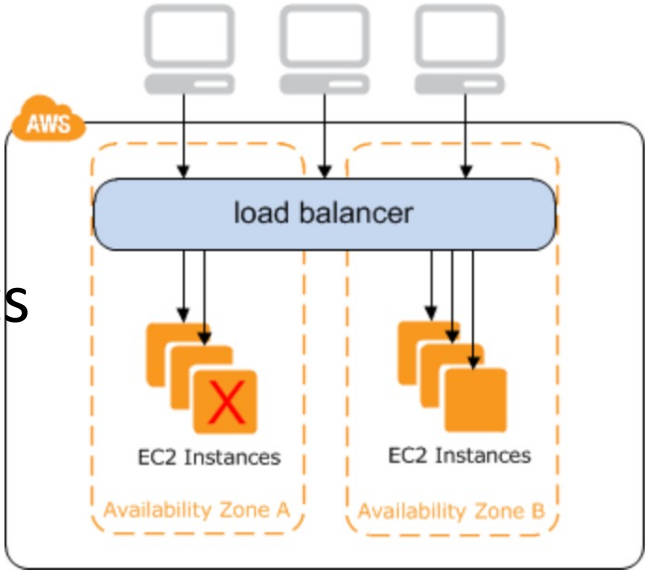
Compute Services: Auto Scaling

- scale EC2 capacity automatically
- for applications with usage variability
- no additional charge for using an Auto Scaling group
- uses a template called “Launch Configuration” to create EC2 instances
- advantages
  - increased availability
  - increased fault-tolerance
  - better cost management



Compute Services: Elastic Load Balancing

- Automatically distributes incoming traffic across targets
- Targets can be instances in one or more Availability Zones
- Single point of contact for the clients
- Routes requests only to healthy targets
- Advantages
  - Increases Fault-tolerance and availability
  - Targets can be added or removed without any disruption
  - Minimizes risk of overloading one server with requests
  - Supports operational monitoring and logging

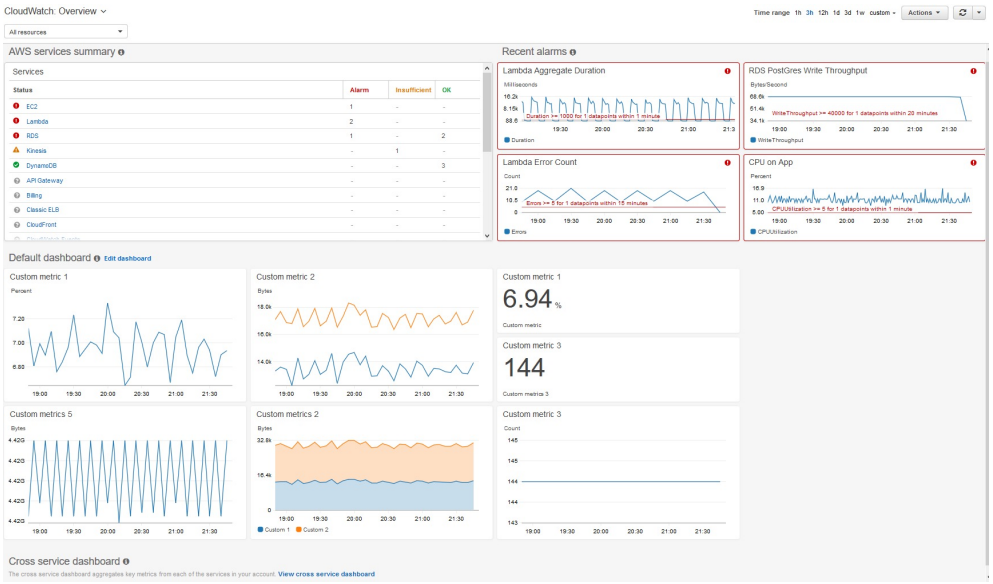


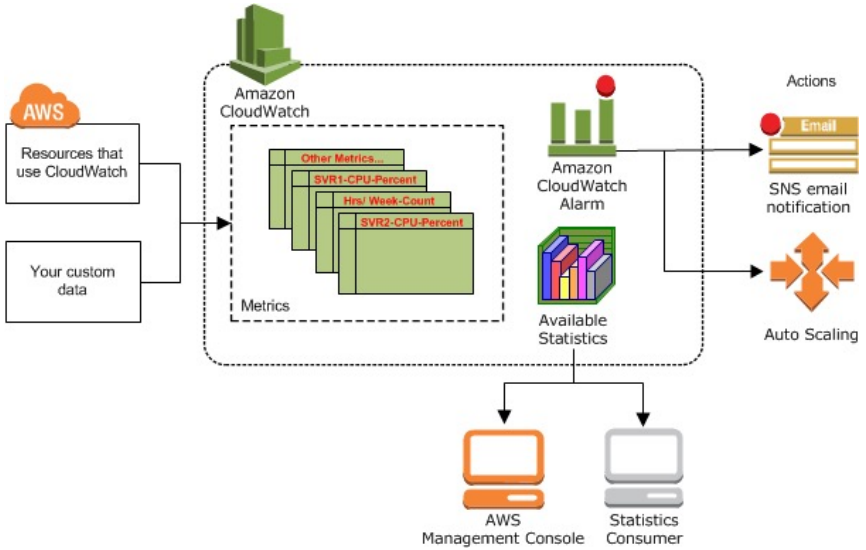
Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

Monitoring Service: Cloudwatch

- Manage and monitor AWS services
- Collect and track metrics and create alerts
- Monitor custom metrics using API



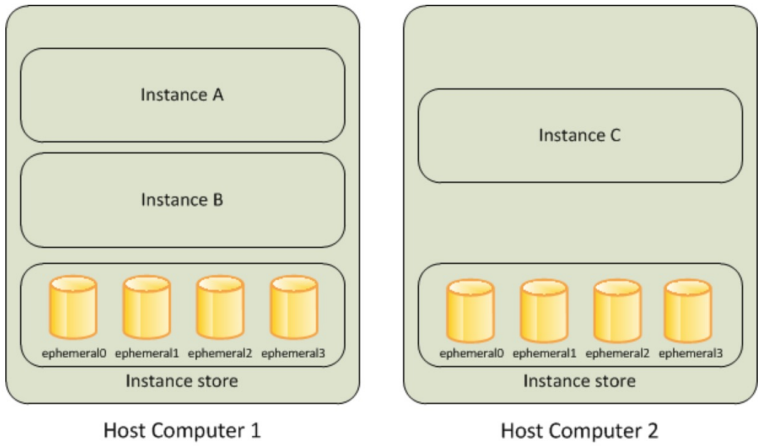


Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

Storage Services: Instance Storage

- Disks physically attached to EC2 host instances
- Ideal for temporary storage that changes frequently
- Data lost once the instance is stop / terminated but persists during reboots
- Data doesn't persist in AMIs
- Size depends on the instance type
- Cost is included in instance cost
- Example:
  - buffers, caches, scratch data

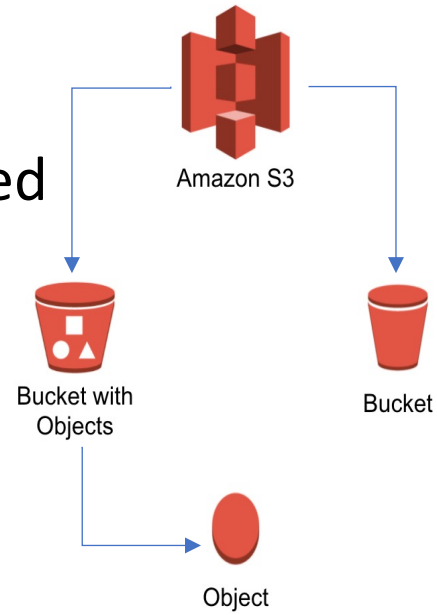


Storage Services
EBS (Elastic Block Store) Volumes

- Persistent block level storage volumes for EC2 instances, e.g database applications, file systems and applications using raw data devices
- Automatically replicated within its AZ
- Can be attached to instance as root or secondary volume
- Advantages
  - Reliable, secure storage (Supports encryption optional)
  - Quickly Backup and restore using Incremental Snapshots
  - Offers the consistent and low-latency performance
  - Quickly scale up, easily scale down within minutes
  - Can be copied across various regions using snapshots
  - Paying a low price for only what you provision

Storage Services: Simple Storage Service (S3)

- GLOBAL object-based storage service
- S3 buckets
  - objects are stored in buckets in a region selected by user and retrieved via unique names
  - set of functions: PUT (write), GET (read) and DELETE
  - User controls read/update access
- Advantages
  - Store/ retrieve any amount of data, anytime anywhere
  - Scalable, reliable, fast and durable
- Objects accessed through a URL:



http://johnsmith.s3.amazonaws.com/photos/puppy.jpg

Bucket

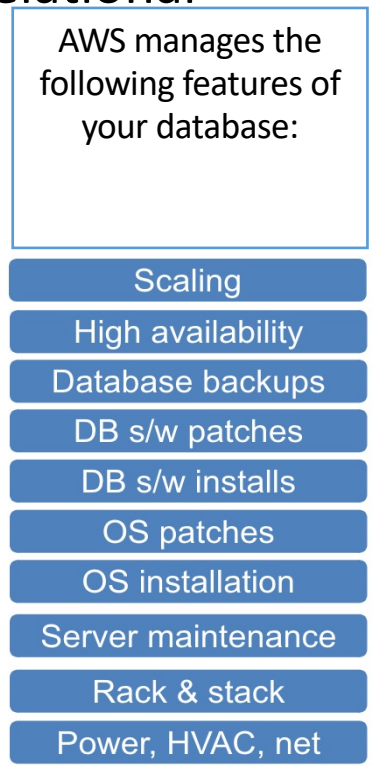
Object/Key

Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

Database Services
Relational Database Service (RDS)

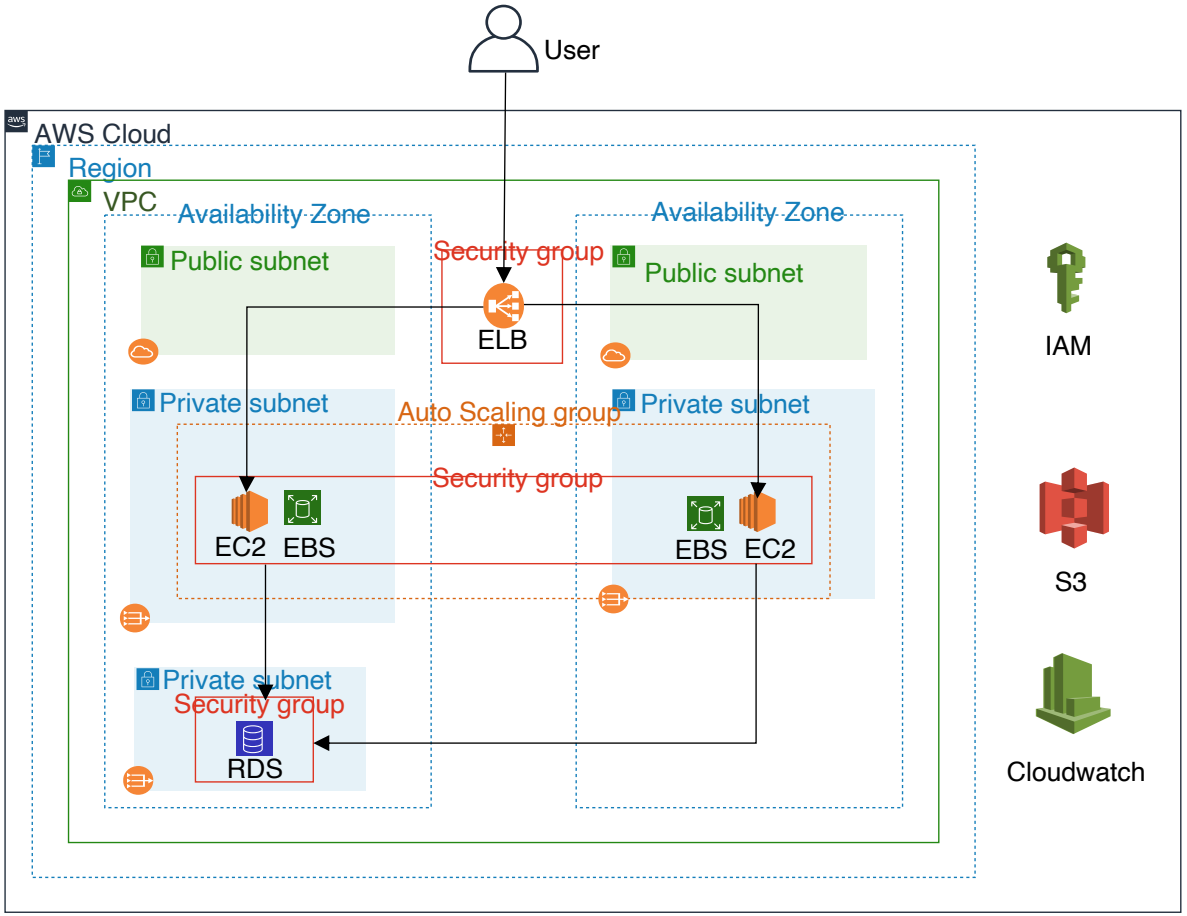
- Managed database service to set up, operate, and scale a relational database
- Basic building block is a DB instance
  - Isolated database environment
  - Contains multiple user-created databases
  - DB engines: MySQL, MariaDB, PostgreSQL, etc.
- Behavior controlled using Parameter Groups
- Available DB instance classes:
  - Standard: For general purpose workloads
  - Memory Optimized: For memory-intensive applications
  - Burstable Performance: provide a baseline performance level, with the ability to burst to full CPU usage
- Can be used in Multi-AZ deployment for High Availability



Outline

- N-tiered Architecture of Web Applications
- AWS Three-tier Architecture
- Types of Core Services
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Monitoring- Cloudwatch
  - Storage: S3, EBS
  - Database- RDS (MySQL)
- AWS Three-tier Architecture using Core Services

AWS Three-tier Architecture using Core Services



Steps

1. Select Region (us-east-1)
2. Create VPC
3. Create subnets in 2 AZs
   - 2 public subnets
   - 2 private subnets for webservers
   - 1 private DB subnet
4. Add IGW (Internet Gateway) and NAT (Network Address Translation Gateway)
5. Create Auto Scaling group (or 2 EC2 instances)
6. Create ELB
7. Create RDS MySQL Instance
8. Create S3 bucket
9. Monitor Cloudwatch metrics

Useful AWS Links

- AWS documentation - docs.aws.amazon.com
- AWS knowledge center (FAQ) – aws.amazon.com/premiumsupport/knowledge-center/
- AWS discussion forums – forums.aws.amazon.com/index.jspa
- 12-month free tier – aws.amazon.com/free
- Pricing - aws.amazon.com/ec2/pricing
- AWS simple monthly calculator – calculator.s3.amazonaws.com/index.html
- Total cost of ownership calculator – aws.amazon.com/tco-calculator
- AWS global infrastructure - aws.amazon.com/about-aws/global-infrastructure/

References

- **Reading**
  - AWS® Certified Solution Architect Study Guide, 2017
  - AWS 12-month Free Tier, aws.amazon.com/free

- **User Guides**
  - Network: Working with VPCs
  - EC2 Instances: Get Started with EC2
  - Auto Scaling: Get Started with Auto Scaling
  - ELB: Increase EC2 Availability
  - RDS: Get Started with RDS
  - S3: Backup files to Amazon S3
  - Cloudwatch: Get Started with Cloudwatch

- **Additional Readings**
  - AWS CLI: AWS CLI
  - Host static website on S3: Static Website Hosting
  - AWS Solution Implementations: Implementations
  - Security Groups vs NACL: SG vs NACL

AWS02: AWS Core Services (Lab)

Outline

- Objective
- Main Steps and Milestones
  - Network - VPC
  - Compute - EC2
  - Elastic Load Balancer
  - RDS DB instance
  - Cloudwatch Metrics
  - Autoscaling Group
- Cleanup
- Summary
- References

Objectives

Setup three-tier architecture on AWS

- Milestone 1: Network
  - Create VPC
  - Create 5 subnets in 2 AZs
  - Create Internet Gateway in each AZ
  - Create NAT Gateway in each AZ
  - Configure Routing tables for subnets
- Milestone 2: Compute & Load Balancing
  - Create one EC2 instance in each AZ
  - Configure Elastic Load Balancer
- Milestone 3: Database and Scaling
  - Configure RDS MySQL DB Instance
  - Observe Cloudwatch metrics
  - Add Auto Scaling Group

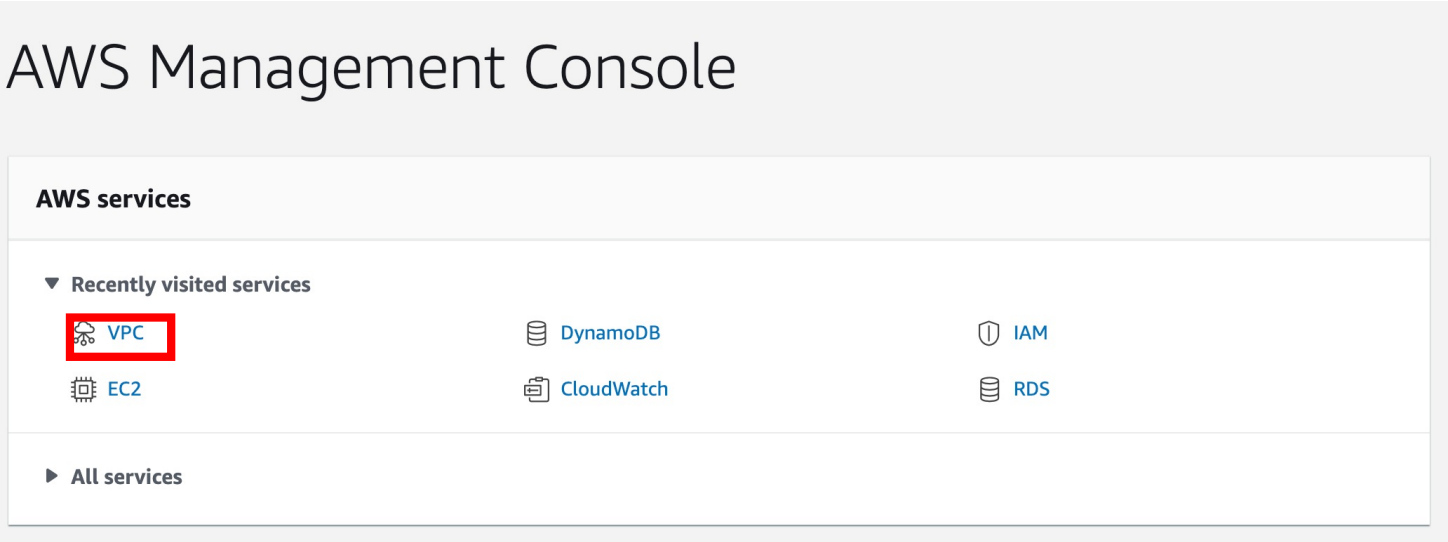
Main Steps - Network

- Sign in
- Select AWS Service
- Select Region
- Create VPC
- Create subnets
- Create Internet Gateway
- Create NAT Gateway
- Add to subnet routing tables

Step 1: Sign-in to AWS console

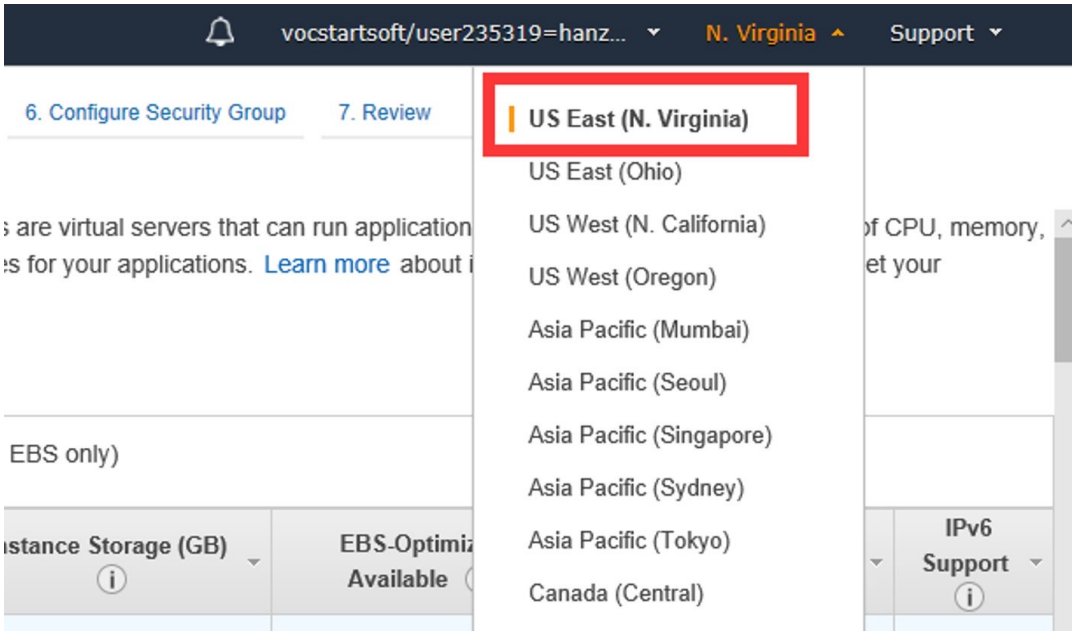
Use the IAM user account we have setup for you!

Step 2: Select VPC Service



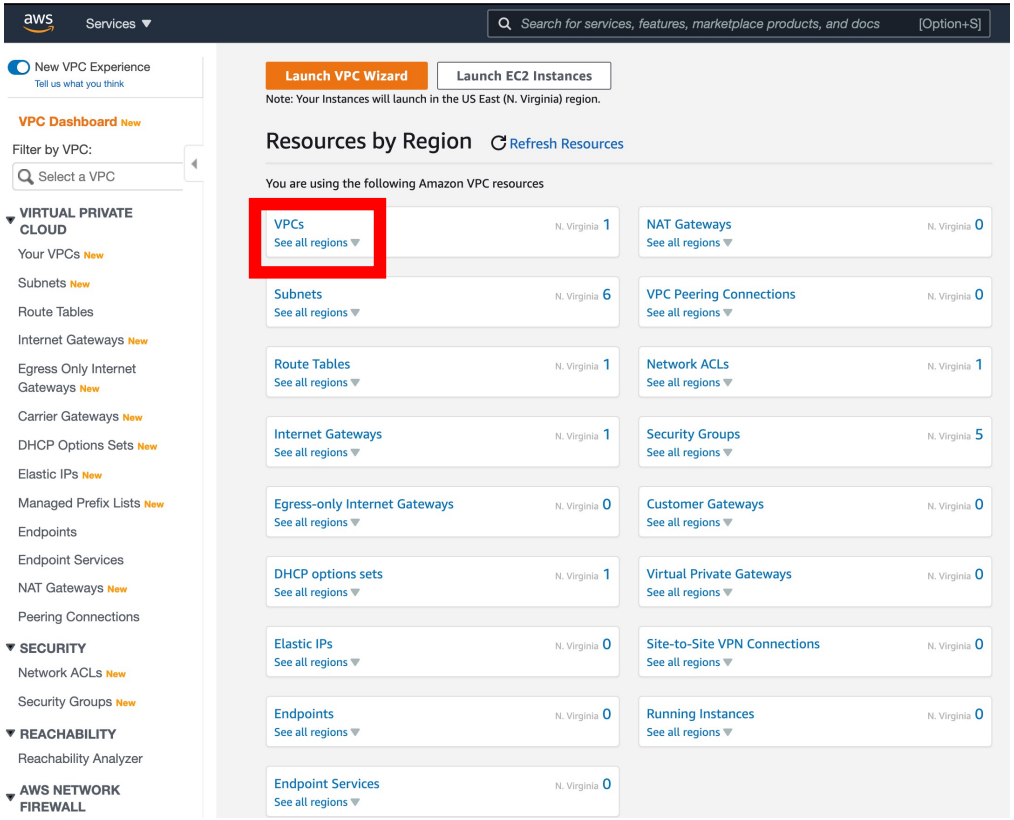
Step 3: Select Region

- Top right of the menu bar, between username and help:
- Choose US East (N. Virginia)



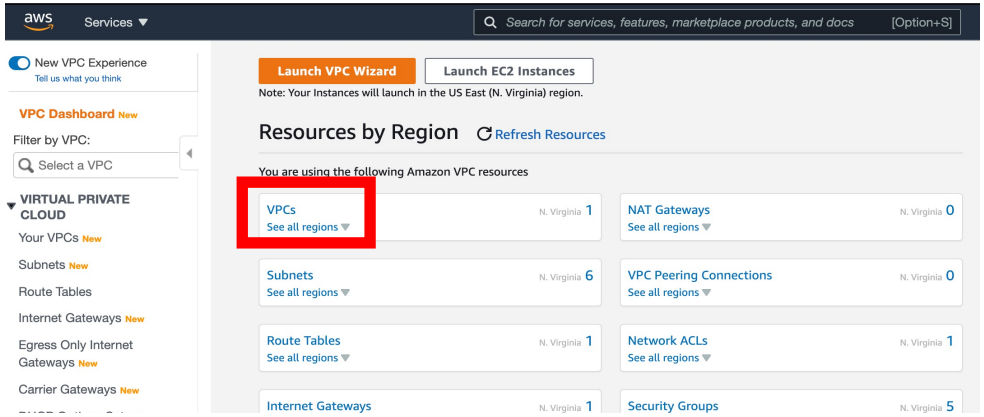
Step 4.1: Create VPC

View Existing default VPC



Step 4.2: Create VPC

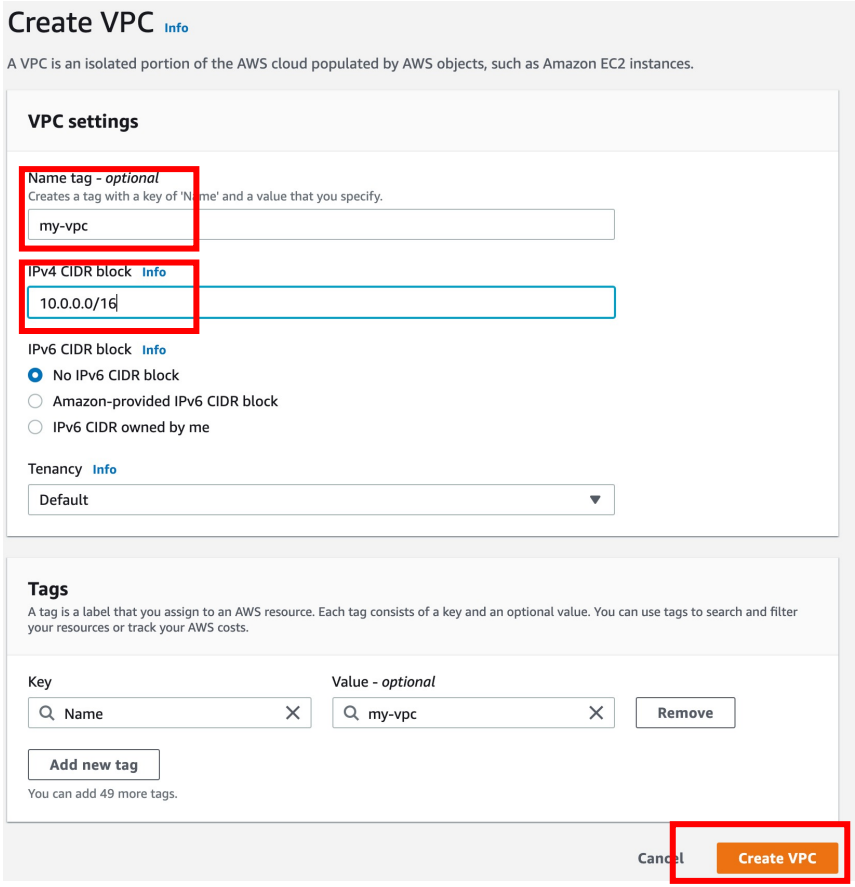
- View Existing default VPC



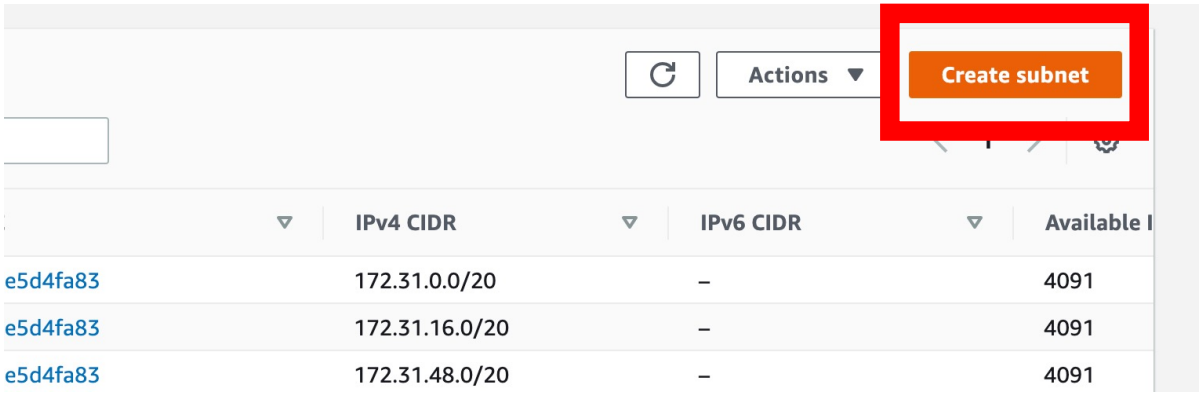
- Create VPC

Step 4.3: Create VPC

- Configure Name
- Configure IPv4 VPC CIDR
- Create

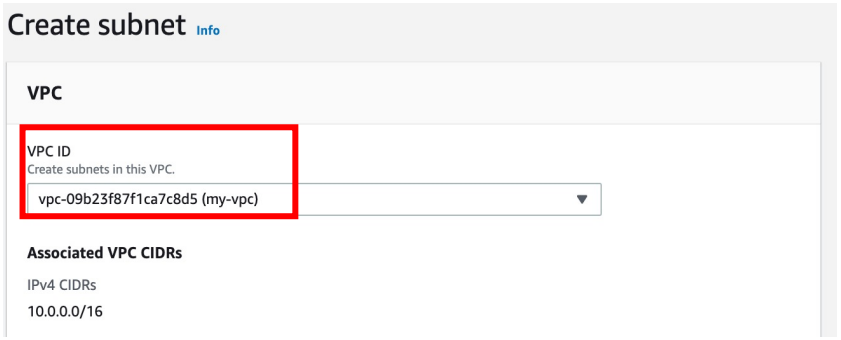


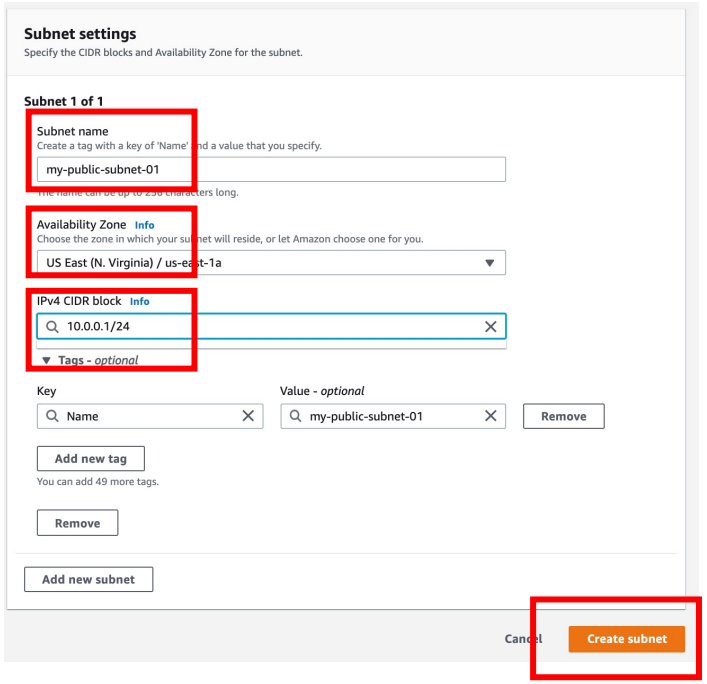
Step 5.1: Create Subnets



Step 5.2: Create Subnets

- Choose VPC
- Configure Name, AZ and CIDR and Create





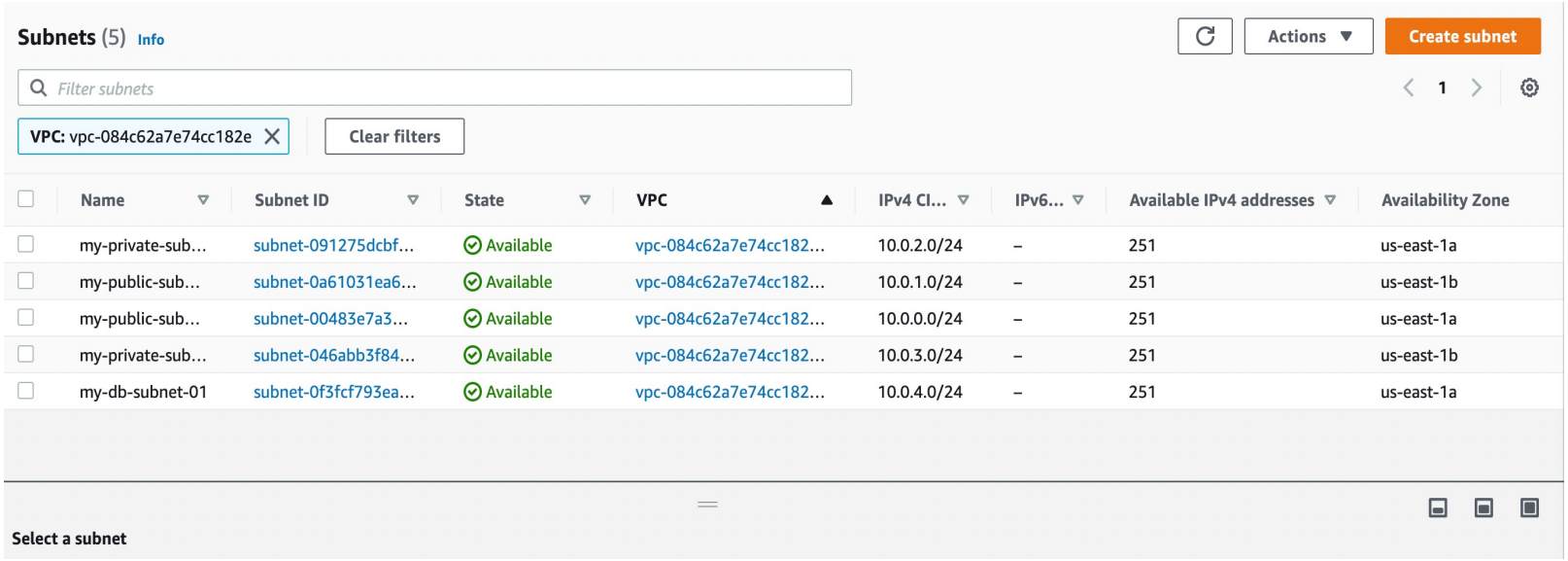
Step 5.3: Create Subnets

Create more subnets with the following configuration settings:

Subnet no	VPC	Name	AZ	CIDR
1	my-vpc	my-public-subnet-01	us-east-1a	10.0.0.0/24
2	my-vpc	my-public-subnet-02	us-east-1b	10.0.1.0/24
3	my-vpc	my-private-subnet-01	us-east-1a	10.0.2.0/24
4	my-vpc	my-private-subnet-02	us-east-1b	10.0.3.0/24
5	my-vpc	my-db-subnet-01	us-east-1a	10.0.4.0/24

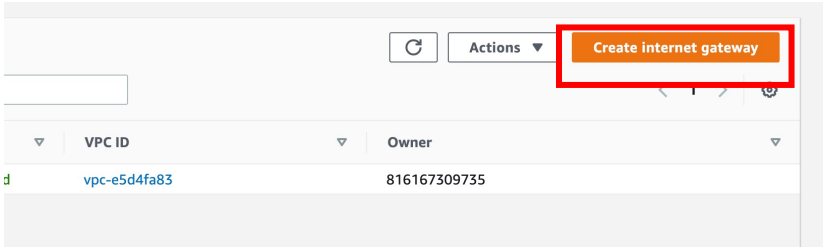
Step 5.4: Create Subnets

Verify the subnets created (filter by vpc)

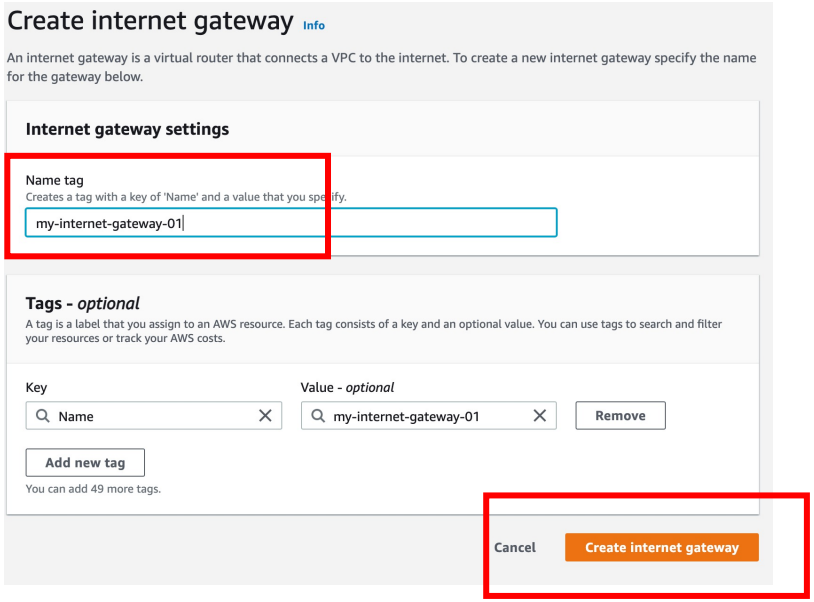


Step 6.1: Create Internet Gateway

- Create Internet Gateway



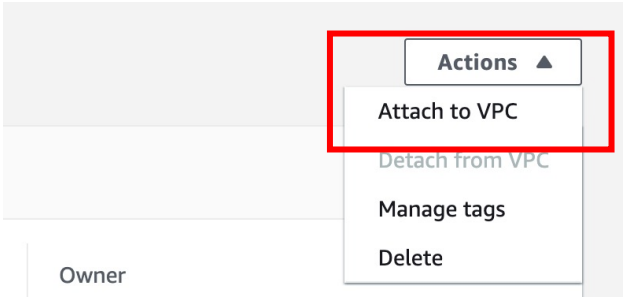
- Configure Gateway Name

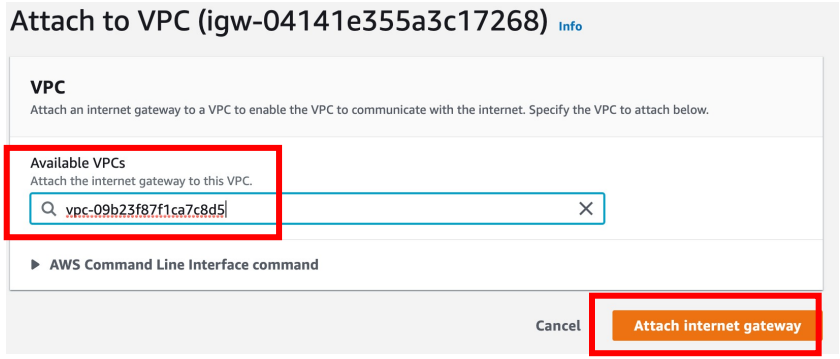


Step 6.2: Create Internet Gateway

Attach Internet Gateway to VPC

- Click on Actions
- Select “Attach to VPC”
- Choose VPC



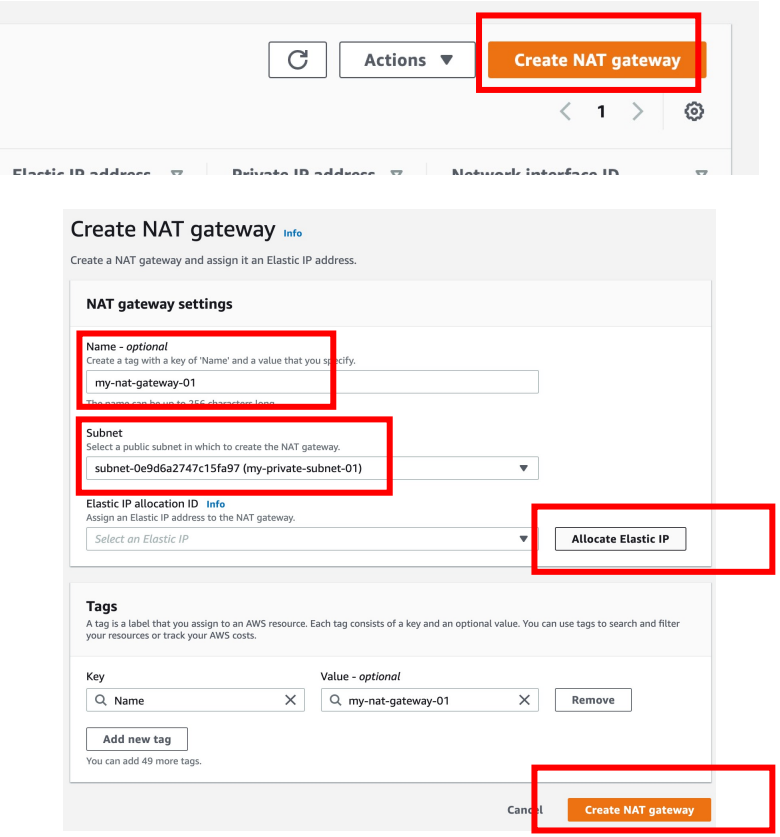


Step 7: Create NAT Gateway

- Configure NAT Gateway for each private subnet:
- Configure Name and subnet as:

NAT Gateway Name	Subnet
My-nat-gateway-01	My-private-subnet-01
My-nat-gateway-02	My-private-subnet-02
My-nat-gateway-03	My-db-subnet-01

- Allocate Elastic IP
- Create

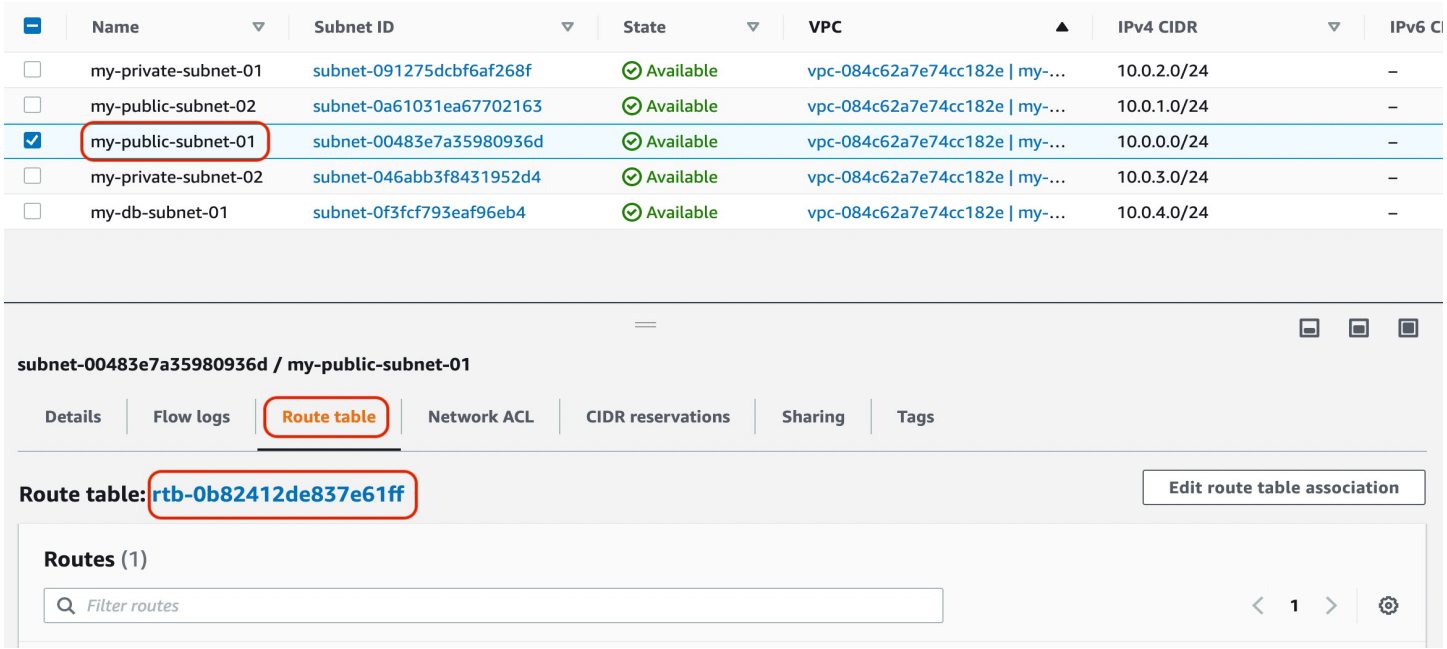


Step 8.1.1: Add to public subnet routing tables

Route Tables of Public subnets :

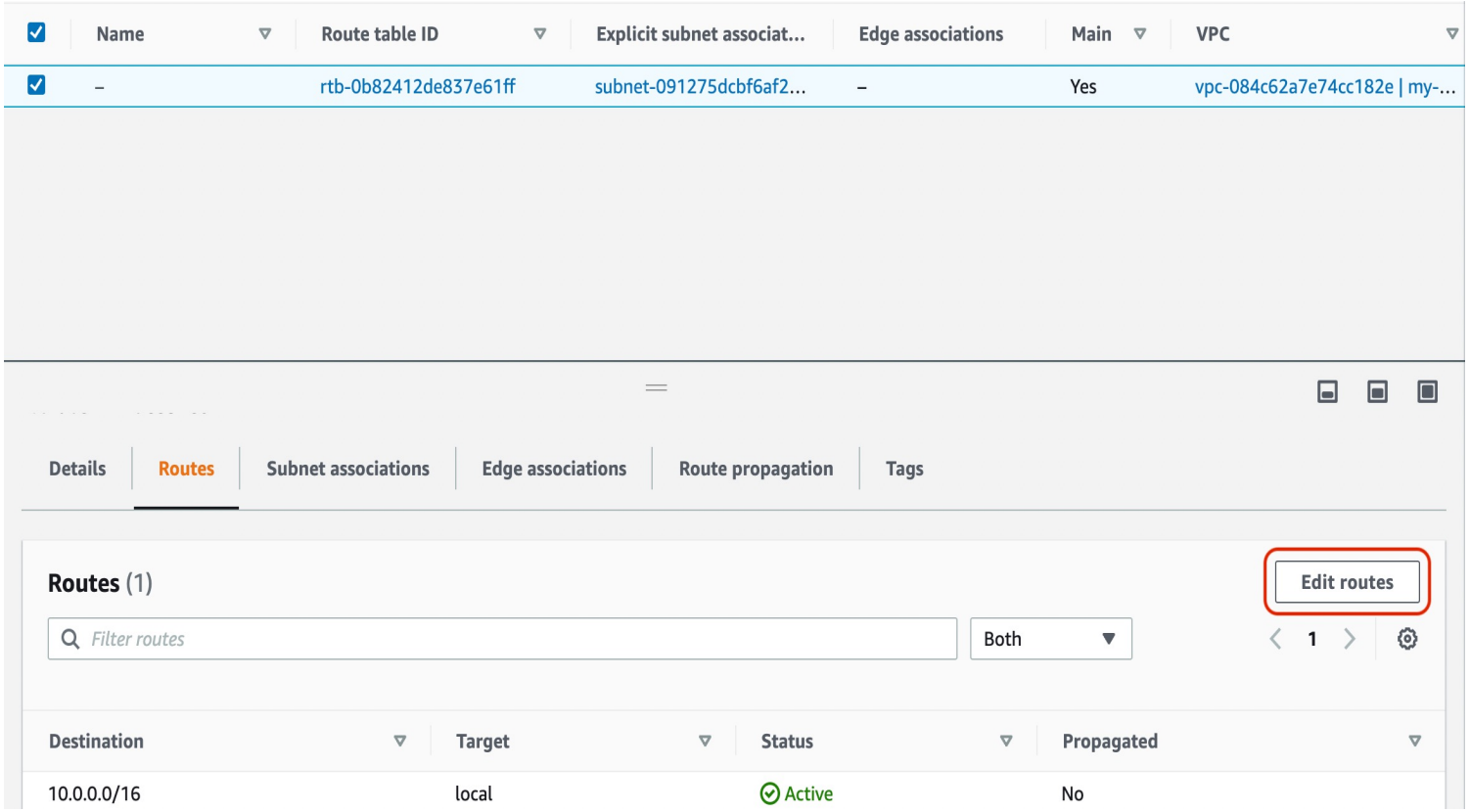
- Select a subnet
- View Route Table and click on the Route table

- Note: This is the default route table, which is currently attached to all subnets. For private subnets, we will create new route tables.



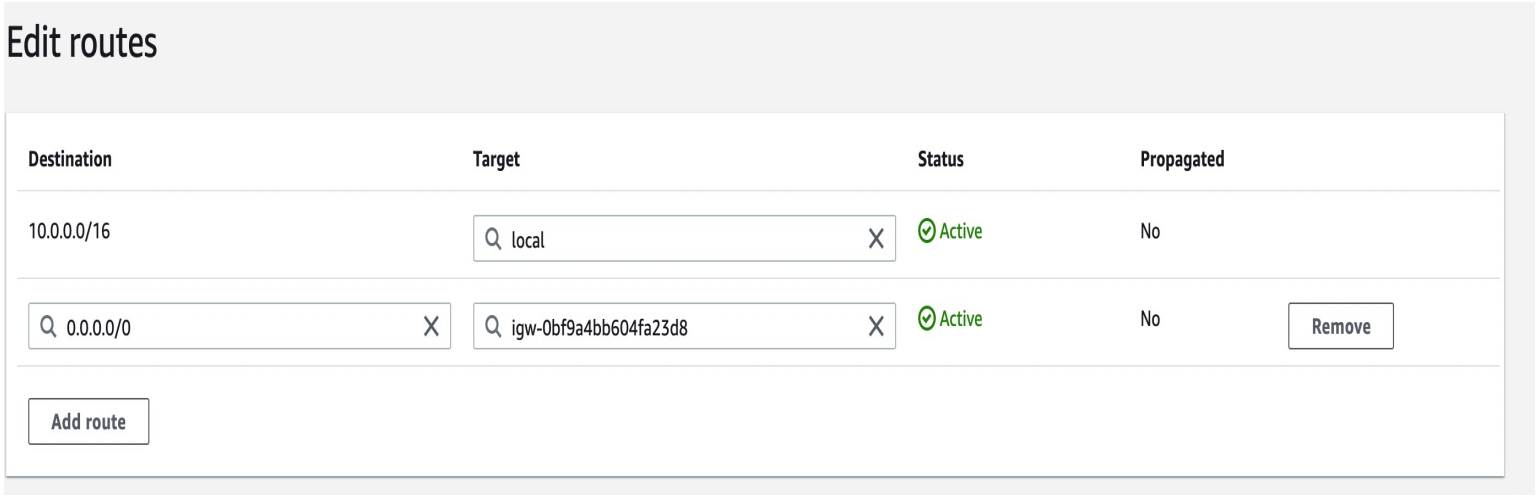
Step 8.1.2: Add to public subnet routing tables

- Click on Edit routes



Step 8.1.3: Add to public subnet routing tables

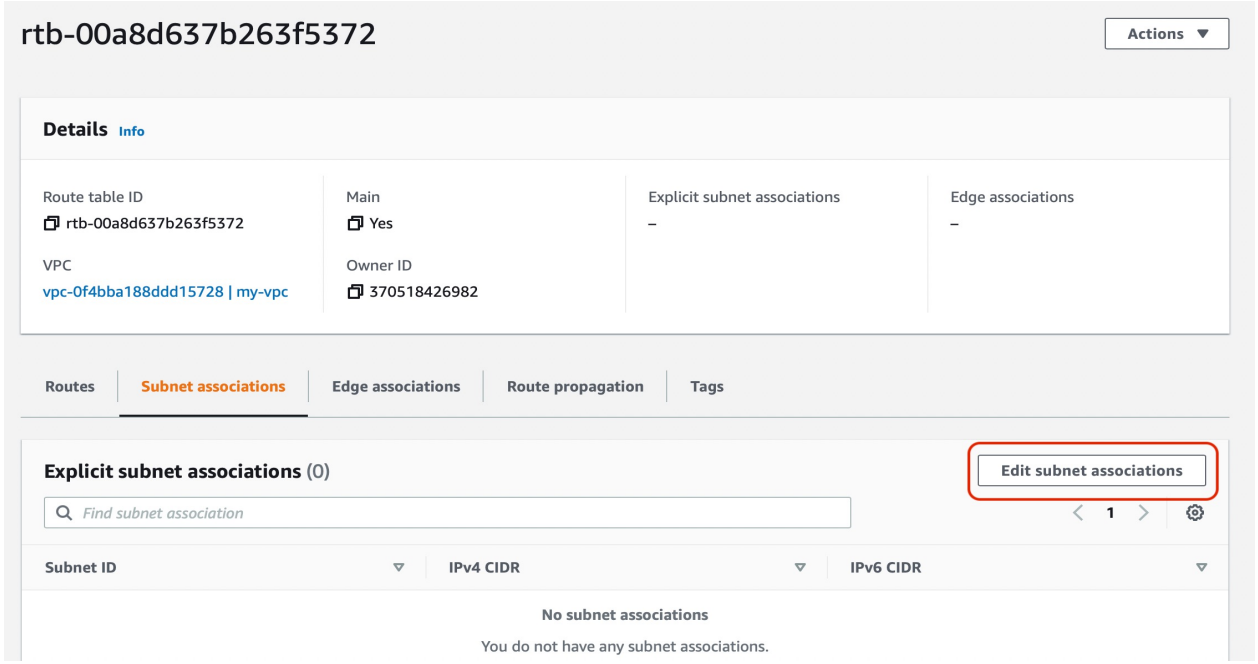
- Add route for all traffic to the internet gateway



Step 8.1.4: Add to subnet routing tables

For the route table of the public subnet:

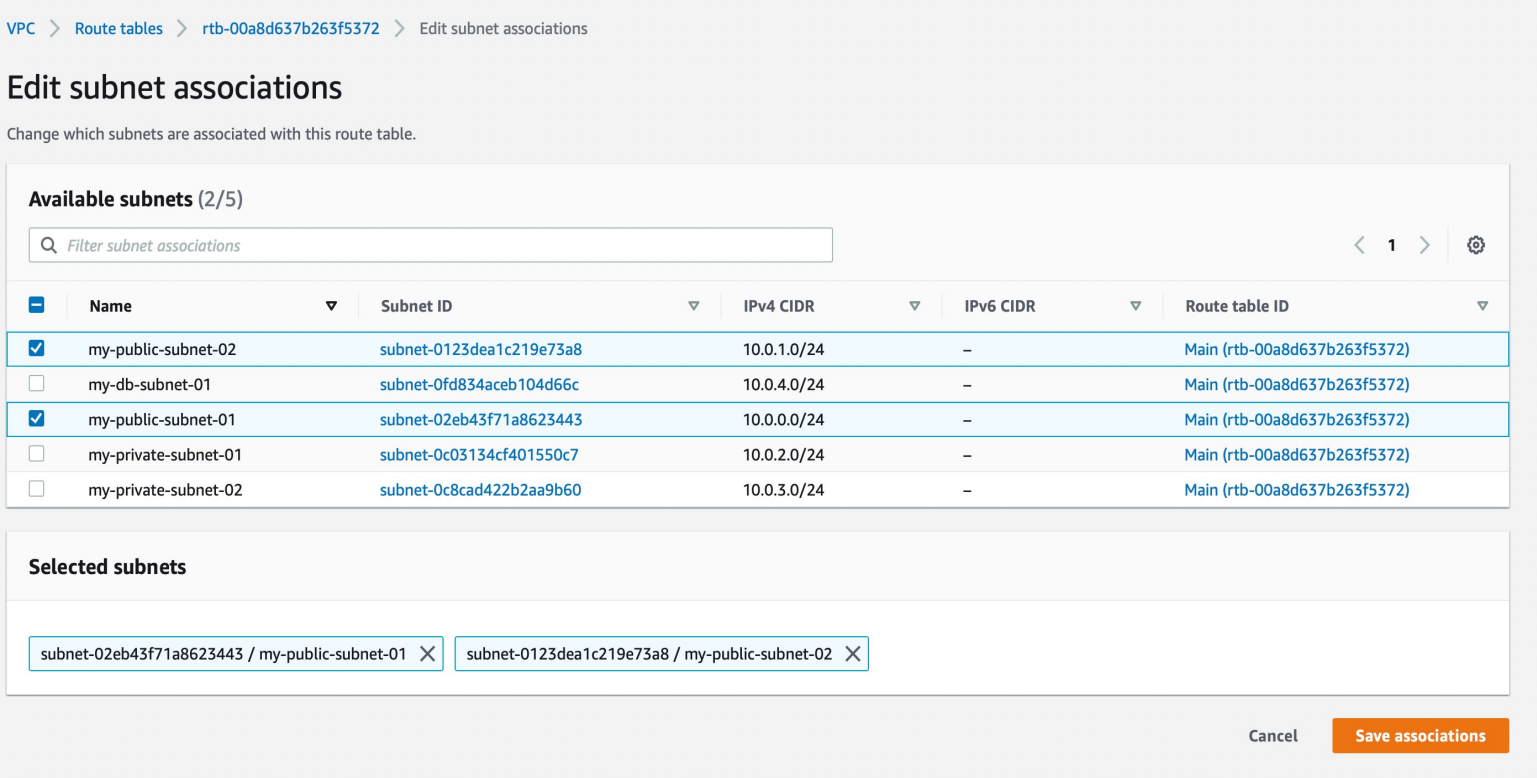
- Click on Edit subnet associations



Step 8.1.5: Add to subnet routing tables

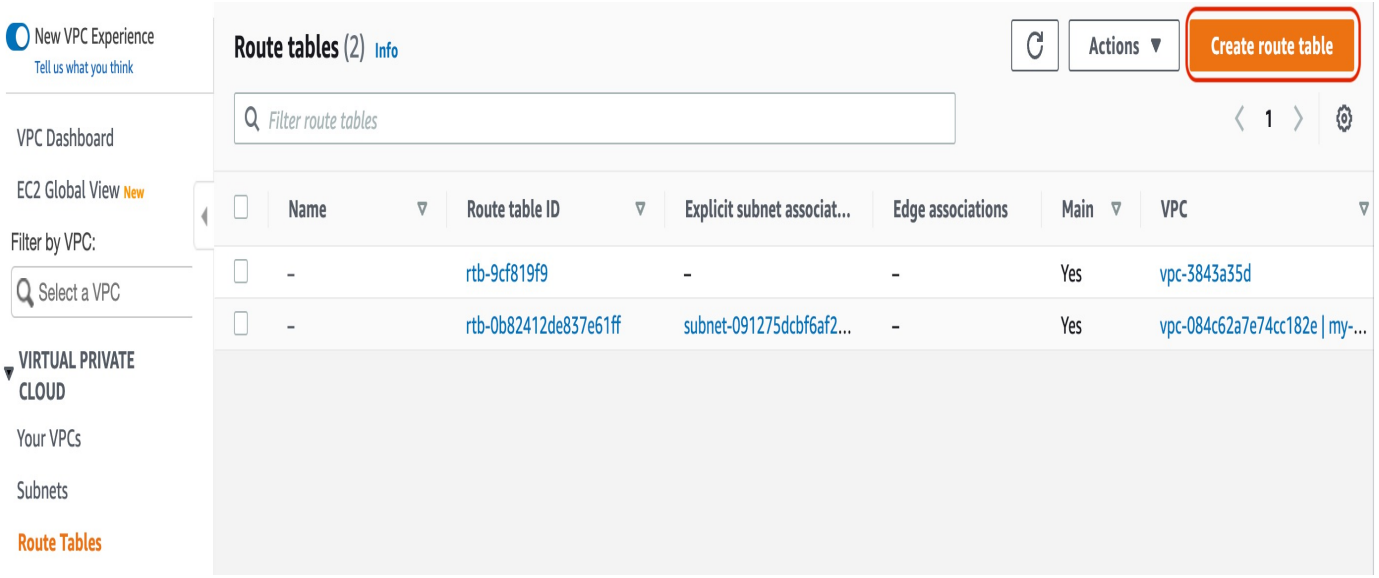
For the route table of the public subnet:

- Associate the two public subnets with the route table.



Step 8.2.1: Add to private subnet routing tables

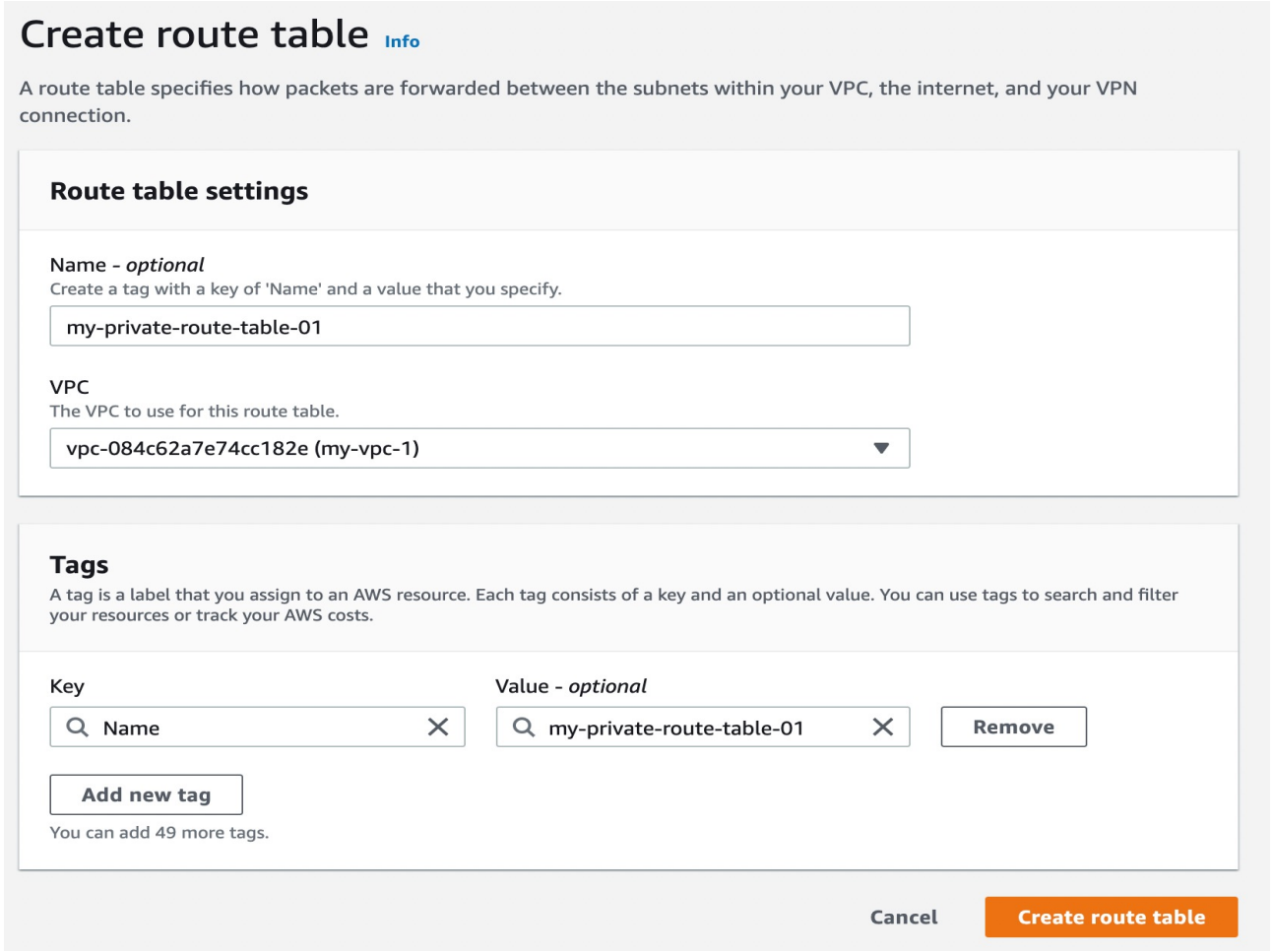
- Route Tables of 3 Private subnets :
  - Click on Create route table



4

Step 8.2.2: Add to private subnet routing tables

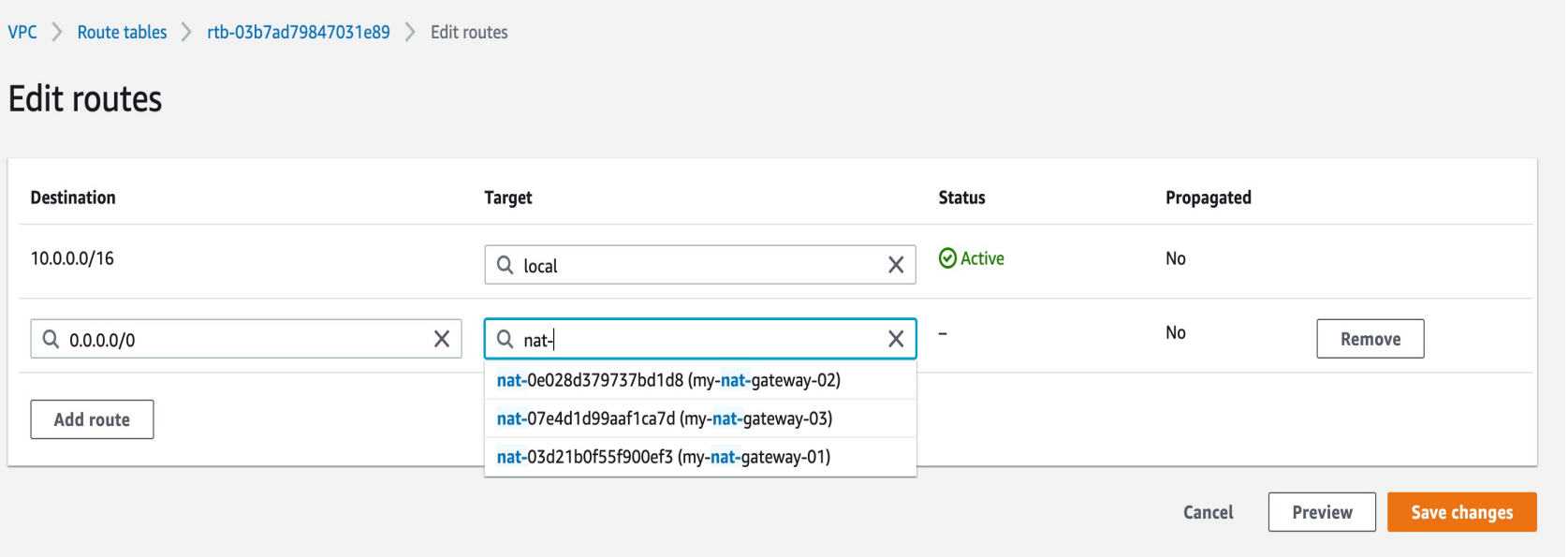
- Configure Names
- Select VPC
- Create routing tables - 'my-private-route-table-01', 'my-private-route-table-02' and 'my-db-route-table-01'



Step 8.2.3: Add to private subnet routing tables

- For each newly created route table, configure Routes to NAT.
  - Select the routing table
  - Click on Edit Routes

Route Table Name	Target
my-private-route-table-01	My-nat-gateway-01
my-private-route-table-02	My-nat-gateway-02
my-db-route-table-01	My-nat-gateway-03

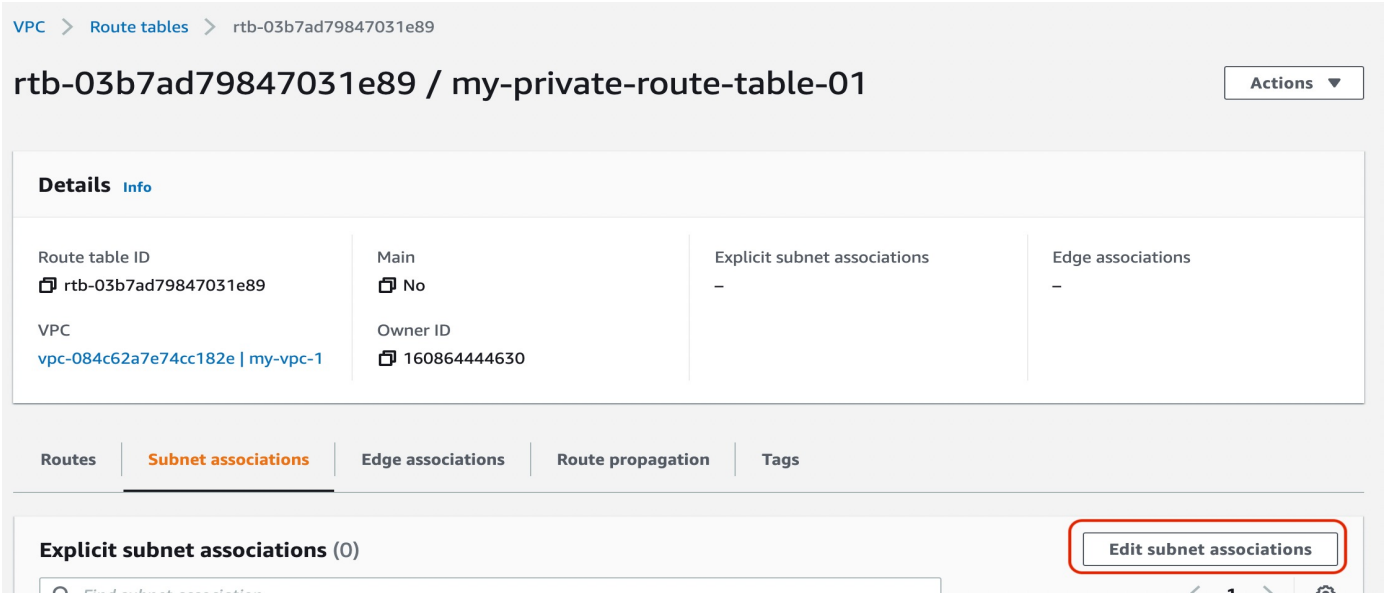


Step 8.3: Add to subnet routing tables

For each newly created Route Table:

- Select Route Table
- Click on Edit subnet associations
- Configure as follows:

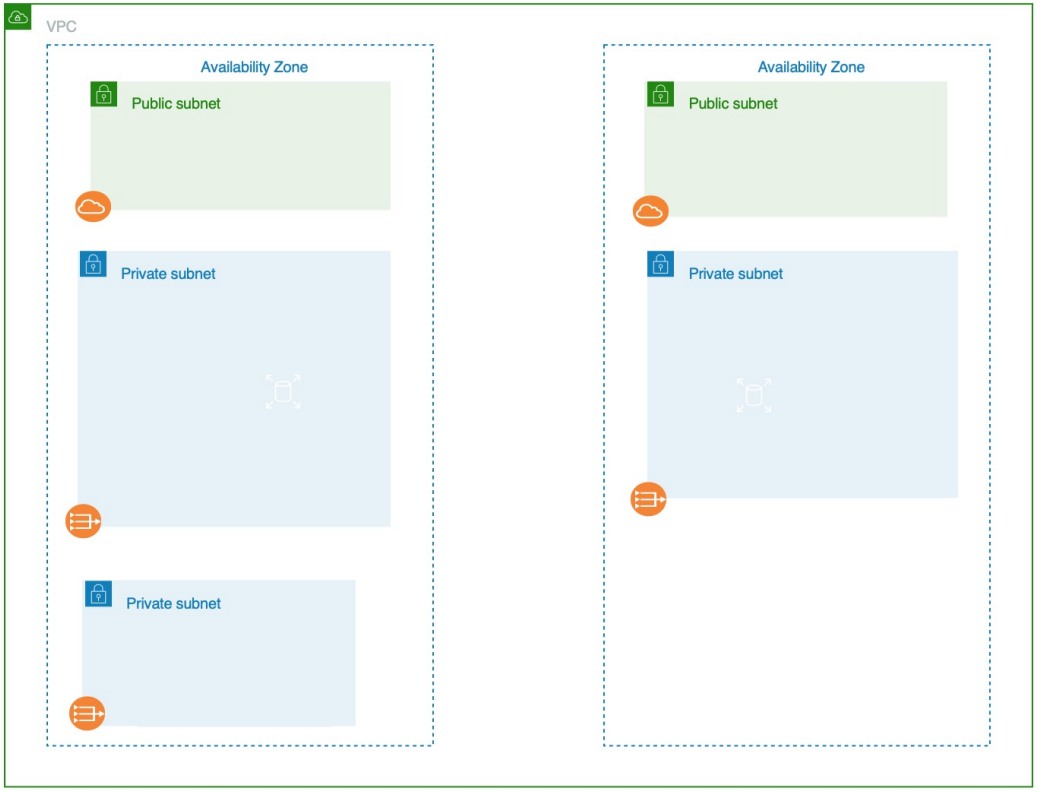
Route Table Name	Subnet
my-private-route-table-01	My-private-subnet-01
my-private-route-table-02	My-private-subnet-02
my-db-route-table-01	My-db-subnet-01



Milestone 1

Verify the resources created:

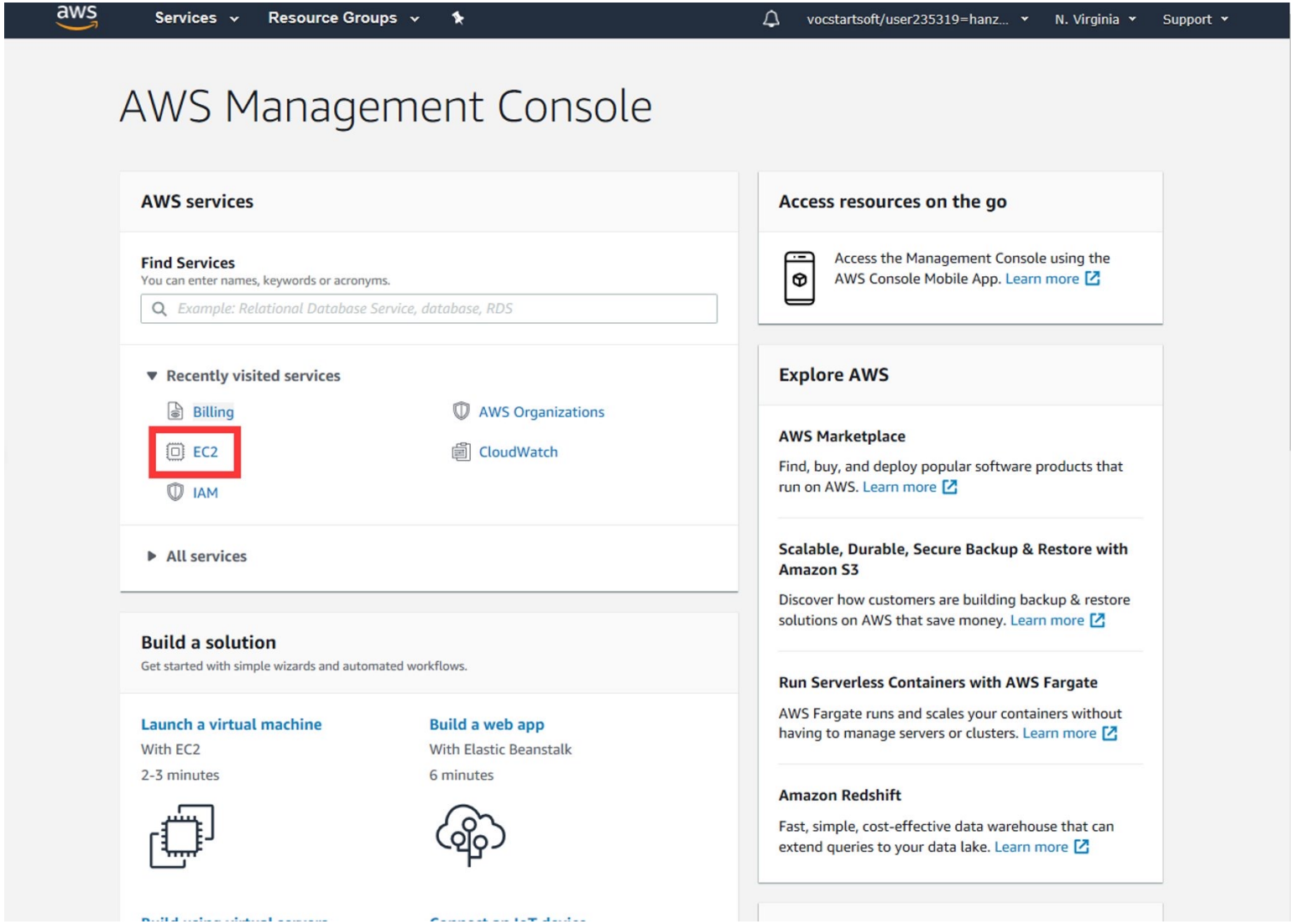
- VPC
- Subnets
- Internet Gateway
- NAT Gateway
- Route Tables



Main Steps - Computing

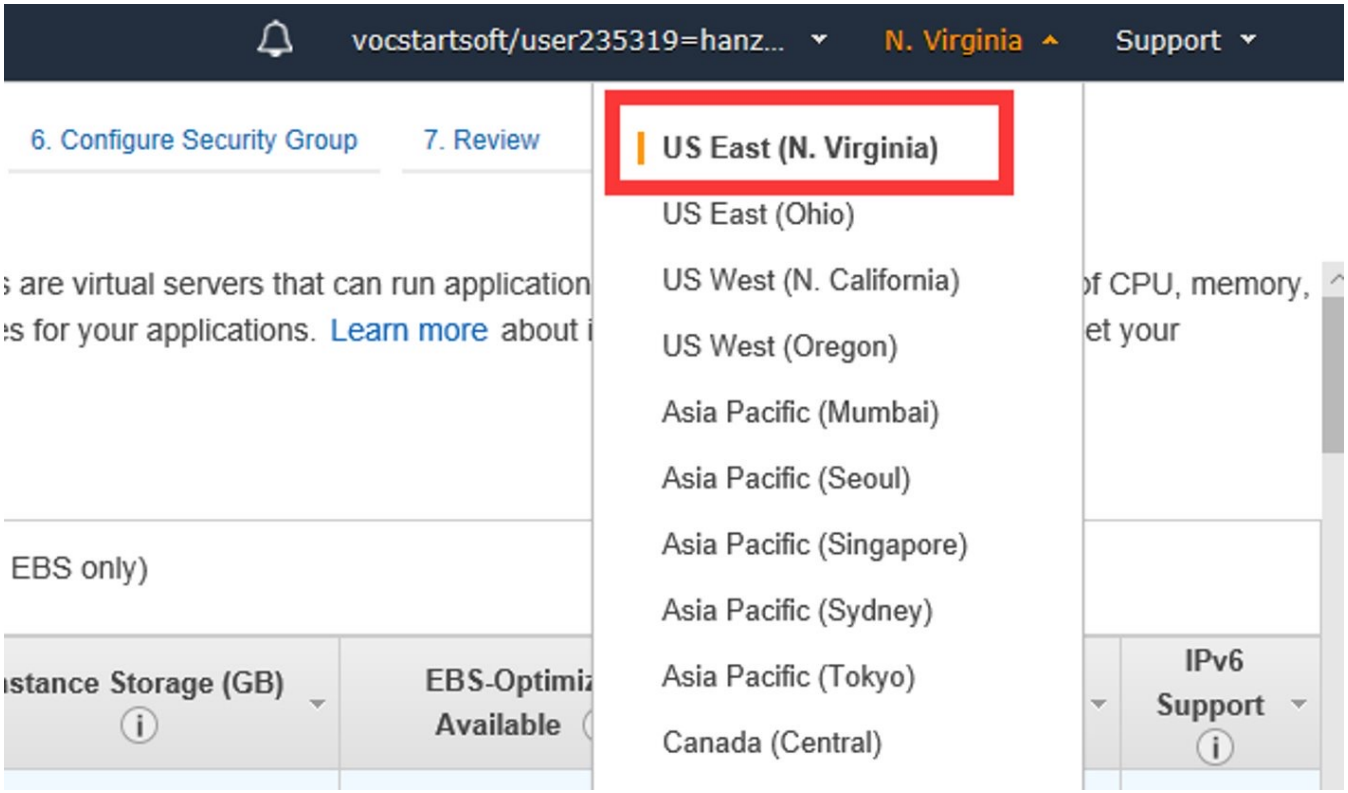
- Select AWS EC2 Service
- Select Region
- Launch instances
- View instance launched

Step 1: Select AWS EC2 Service



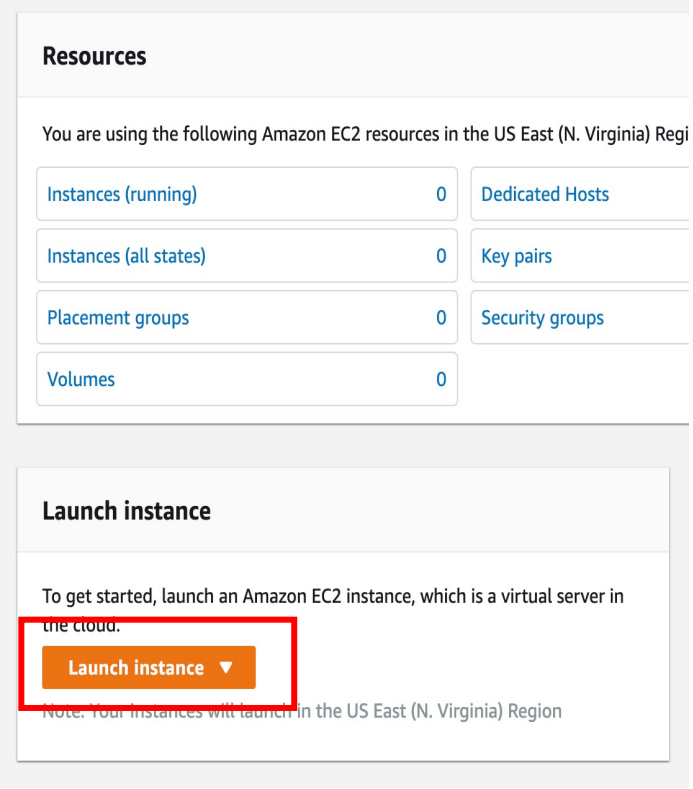
Step 2: Select Region

Top right of the menu bar, between username and help: Choose N.Virginia



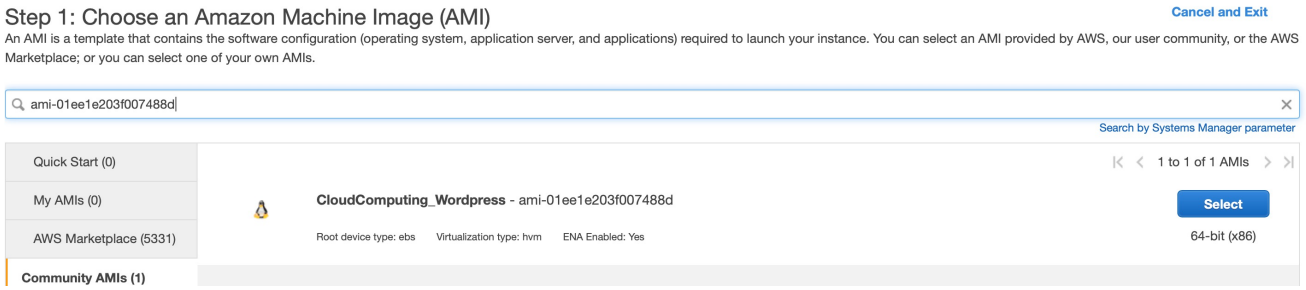
Step 3.1: Launch Instances

- Launch 2 instances one by one
- Note: All parameters (except the subnet) are the same

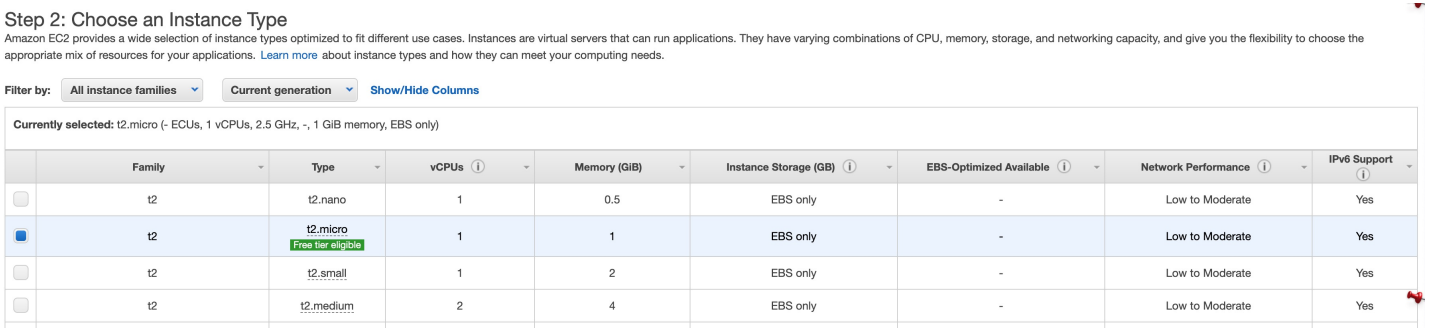


Step 3.2: Launch Instances

- Select AMI
- Search for ‘ami-01ee1e203f007488d’ under community ami's -> Select



- Select Instance Size (t2.micro)



Step 3.3: Launch Instances

Configure Instance

Instance 1:

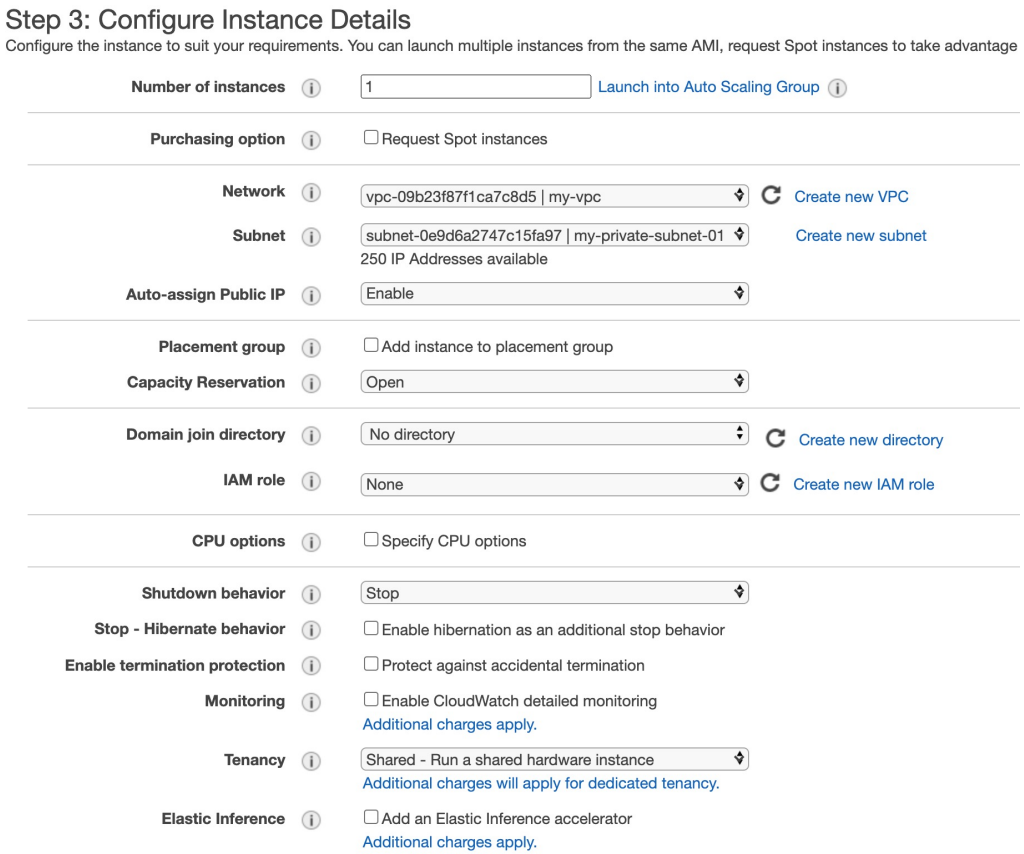
- Number of Instances: 1
- Network: my-vpc
- Subnet: my-private-subnet-01
- Auto-Assign Public IP: Disable
- Advanced Details – Userdata:

```
#!/bin/bash
sudo service apache2 restart
```

Instance 2:

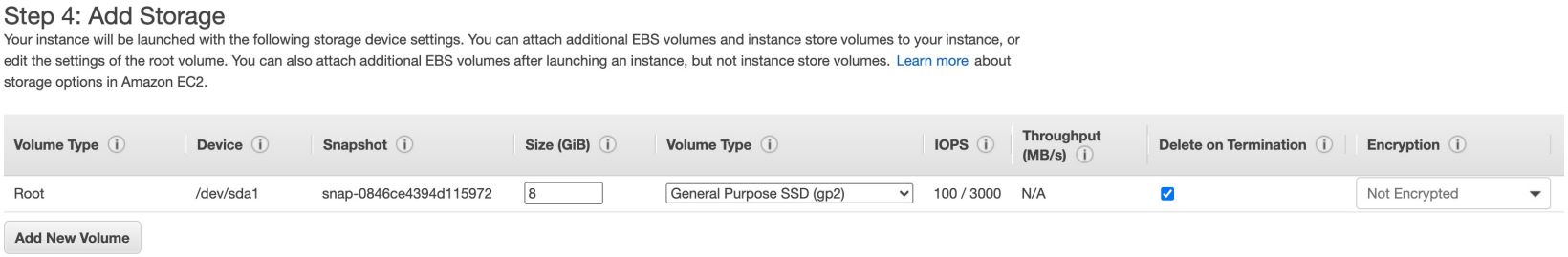
- Number of Instances: 1
- Network: my-vpc
- Subnet: my-private-subnet-02
- Auto-Assign Public IP: Disable
- Advanced Details – Userdata:

```
#!/bin/bash
sudo service apache2 restart
```



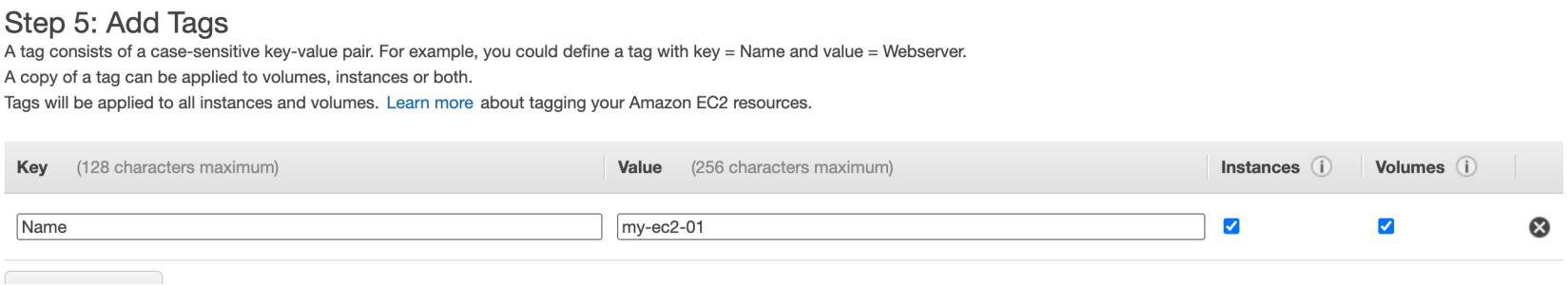
Step 3.4: Launch Instances

Add a root EBS volume of size 8GB to the instance



Step 3.5: Launch Instances

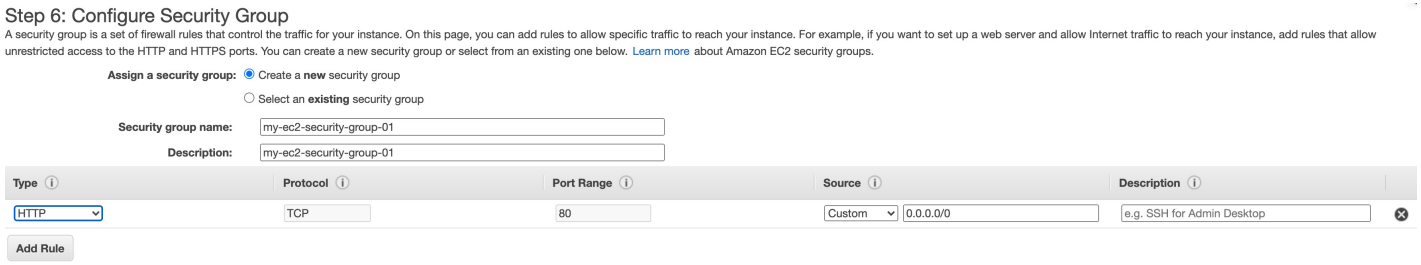
- Add Name Tags as:
  - Instance 1: my-ec2-01
  - Instance 2: my-ec2-02
- Optional: You can any other key-value tags



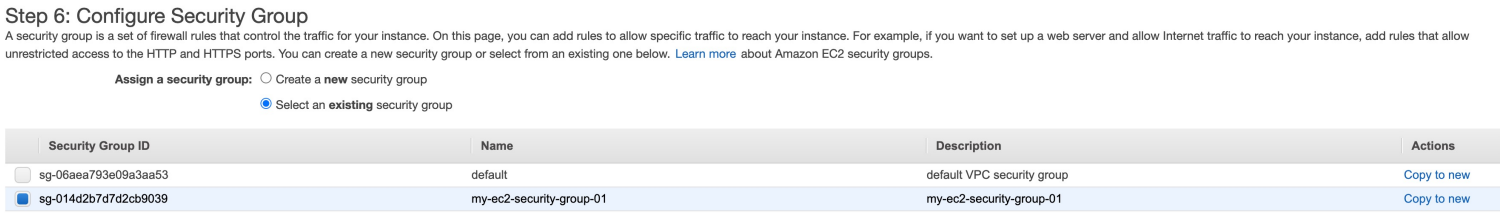
Step 3.6: Launch Instances

- Create a single Security Group for both instances
  - Allow HTTP access from all sources
- Note: This will be changed after creating the ELB

Instance 1:



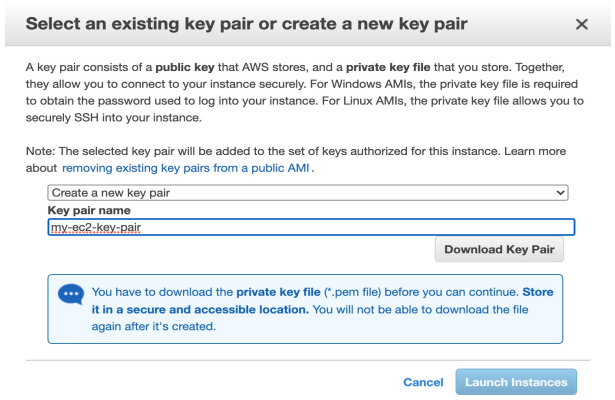
Instance 2:



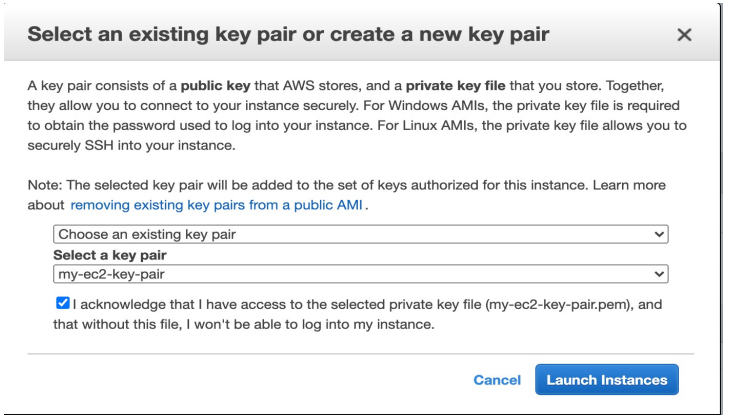
Step 3.7: Launch Instances

- Click on “Review and Launch”
- Click on “Launch”
- Configure a single Key Pair for both instances

Instance 1:



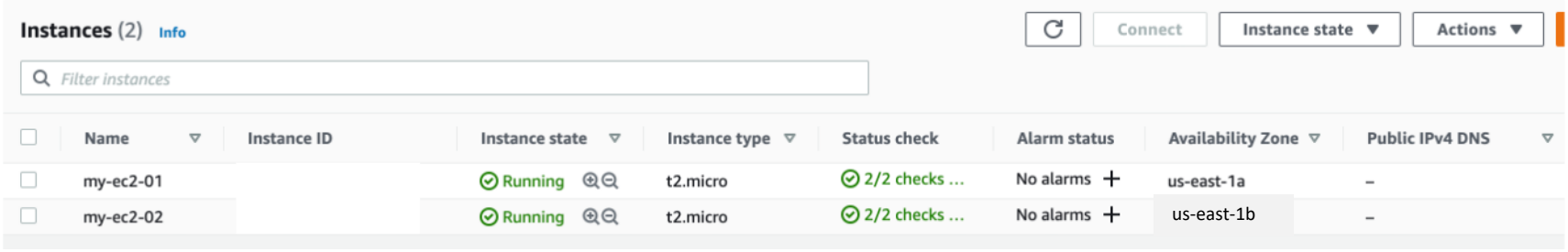
Instance 2:



- Click on “Download Key Pair”
- Then launch instance.

Step 4: View Instances

- Verify that instances have been successfully launched:
  - Check Instance State is Running
  - Status Checks should be passed (may take some time)
  - Select the instance to view more details and monitoring

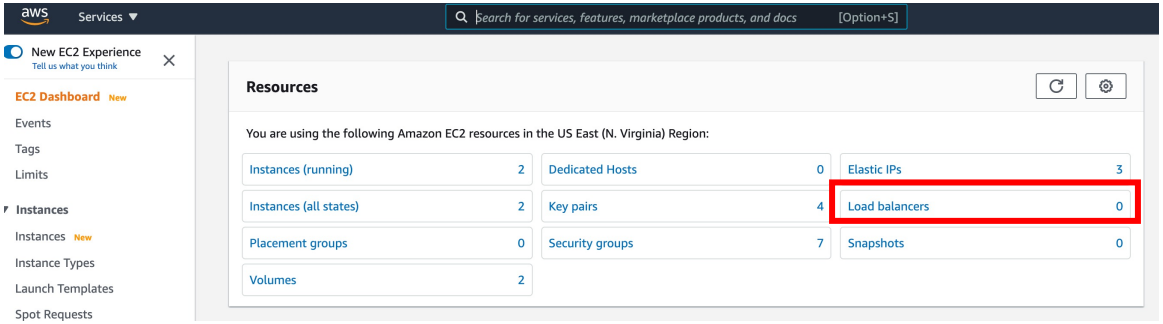


Main Steps – Elastic Load Balancer

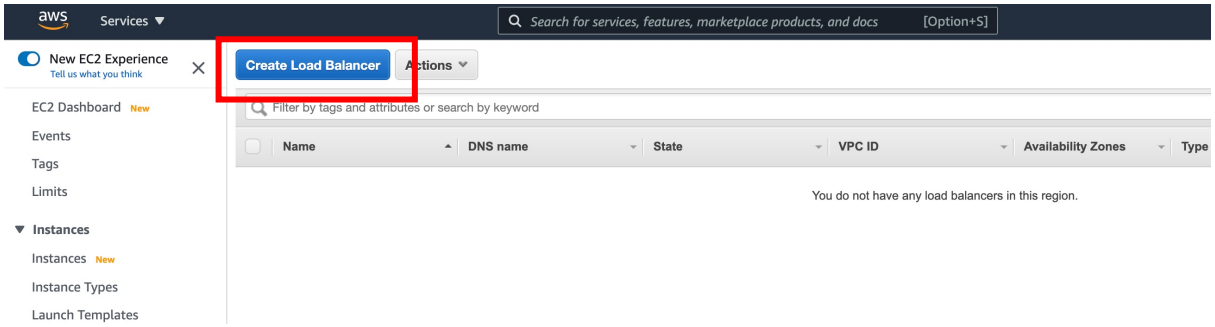
- Select AWS Service
- Select Load Balancer Type
- Basic Configuration
- Security Group
- Configure Health Check
- Add EC2 instances
- Verify successful creation
- Add access to EC2 security group

Step 1: Select AWS Service

On the EC2 console:

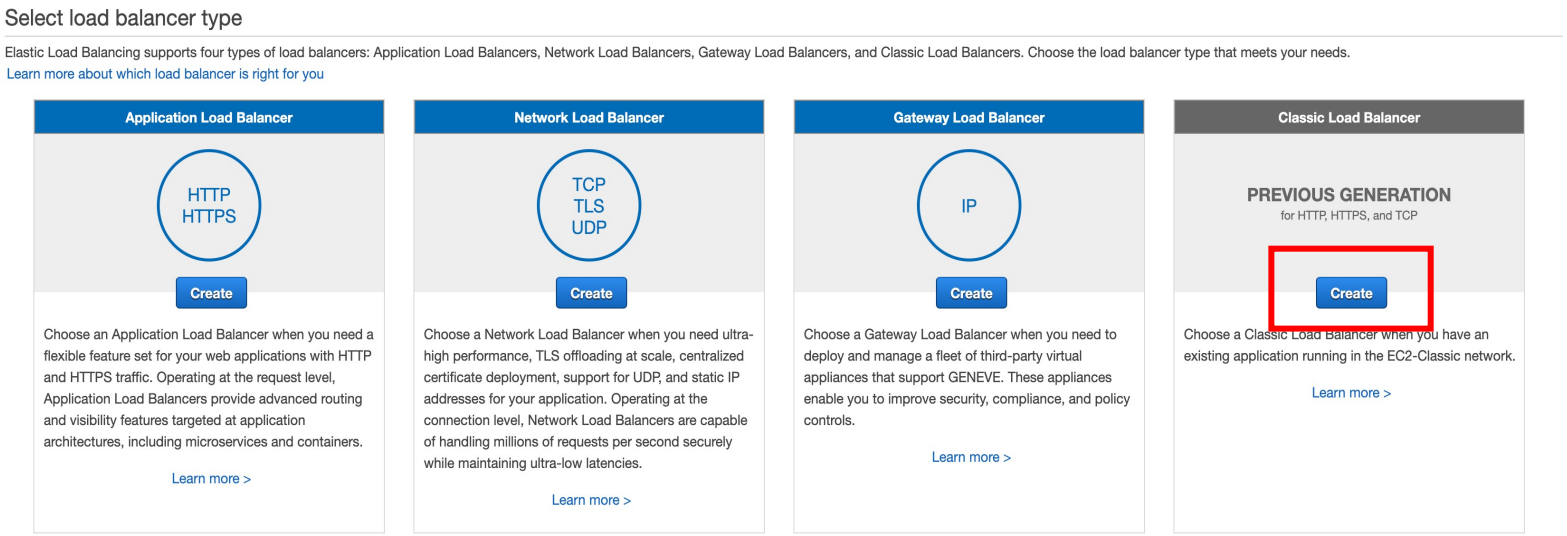


Create Load Balancer:



Step 2: Select Load Balancer Type

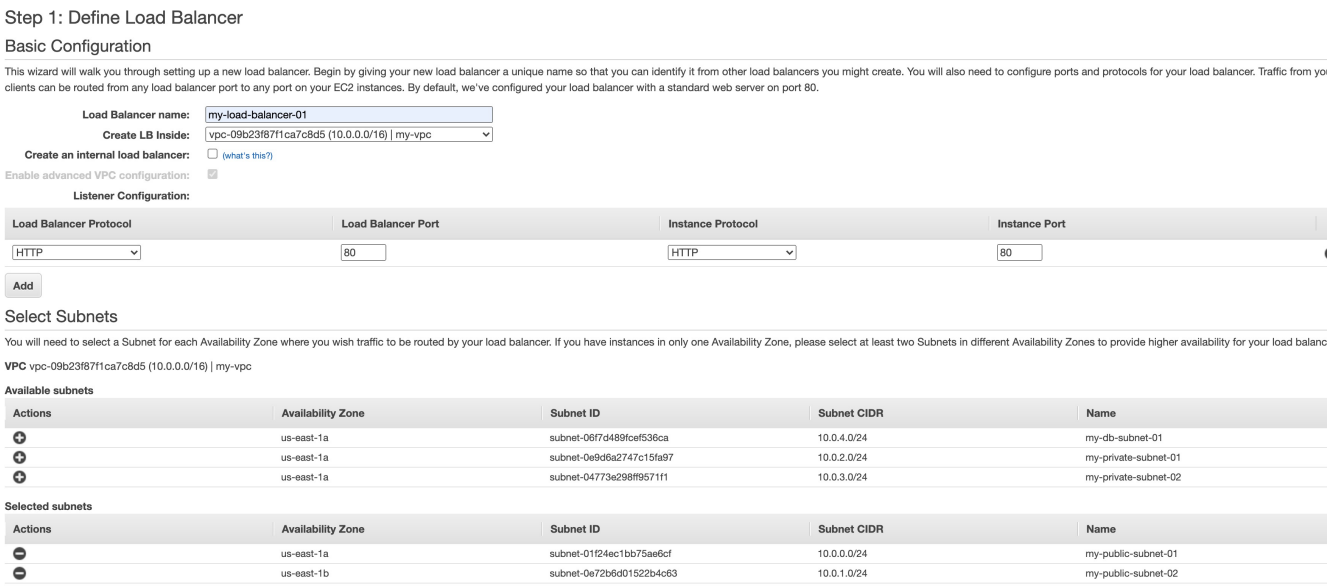
Select Classic Load Balancer:



Step 3: Basic configuration

- Name: my-load-balancer-01
- Create LB inside: my-vpc
- Select subnets: my-public-subnet-01, my-public-subnet-02

Note: Always assign public subnets to ELB in the same AZs as your EC2 instances.



Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from yo
clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name: my-load-balancer-01

Create LB Inside: vpc-09b23f87f1ca7c8d5 (10.0.0.0/16) | my-vpc

Create an internal load balancer: (what's this?)

Enable advanced VPC configuration:

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

Add

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balanc

VPC vpc-09b23f87f1ca7c8d5 (10.0.0.0/16) | my-vpc

Available subnets

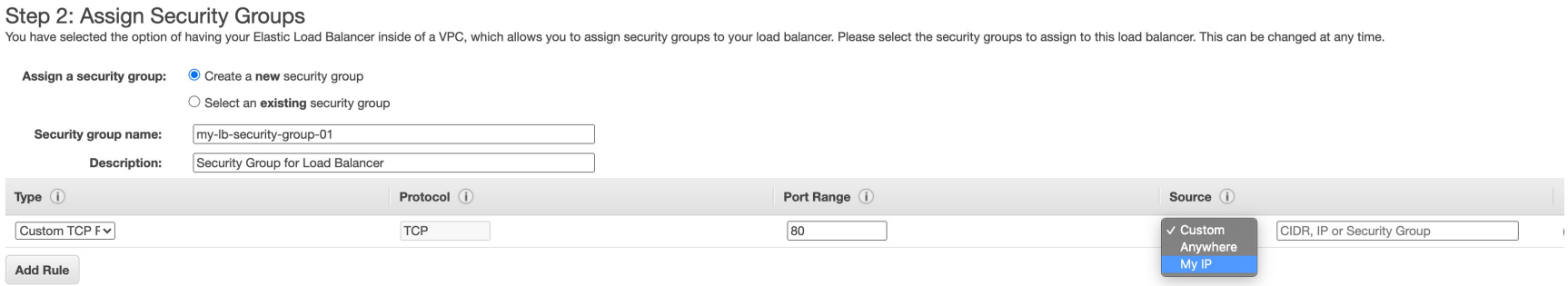
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
⊕	us-east-1a	subnet-06f7d489fcef536ca	10.0.4.0/24	my-db-subnet-01
⊕	us-east-1a	subnet-0e9d6a2747c15fa97	10.0.2.0/24	my-private-subnet-01
⊕	us-east-1a	subnet-04773e298ff9571f1	10.0.3.0/24	my-private-subnet-02

Selected subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
⊖	us-east-1a	subnet-01f24ec1bb75ae6cf	10.0.0.0/24	my-public-subnet-01
⊖	us-east-1b	subnet-0e72b6d01522b4c63	10.0.1.0/24	my-public-subnet-02

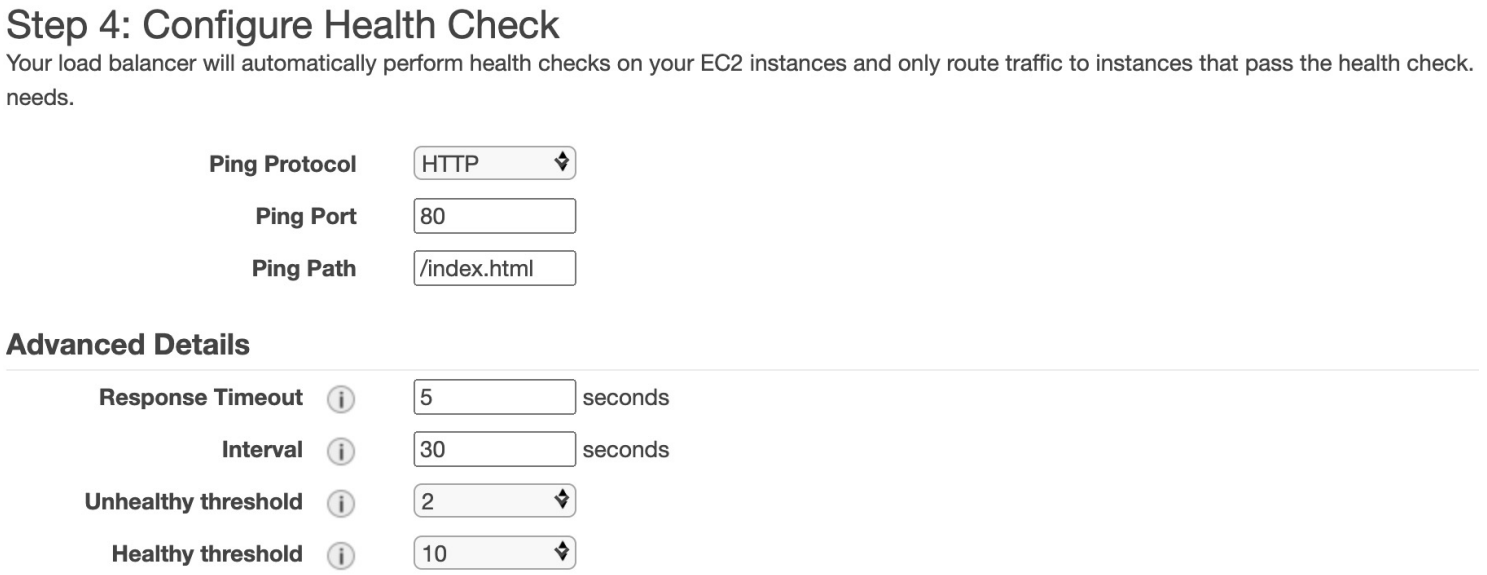
Step 4: Security Group

- Create a new Security Group for the Load Balancer:
- Allow HTTP (port 80) access from your IP



Step 5: Configure Health Check

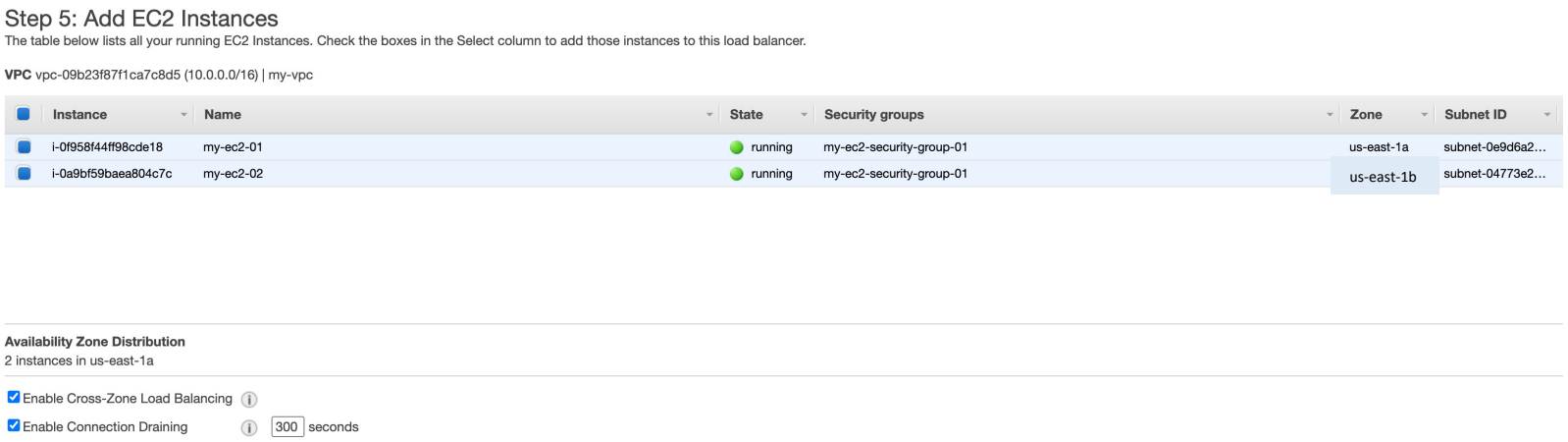
- Used by the load balancer to determine if instance is healthy



Step 6: Add EC2 instances

- Add EC2 instances as targets for the ELB

Note: If you don’t see the correct instances here, check the subnet configurations of EC2 and ELB

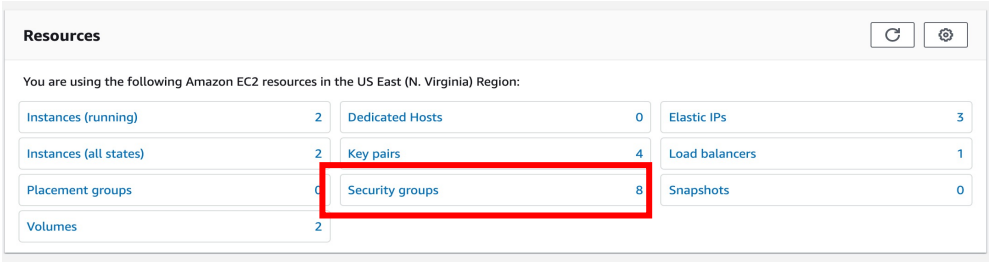


Step 7: Verify successful creation

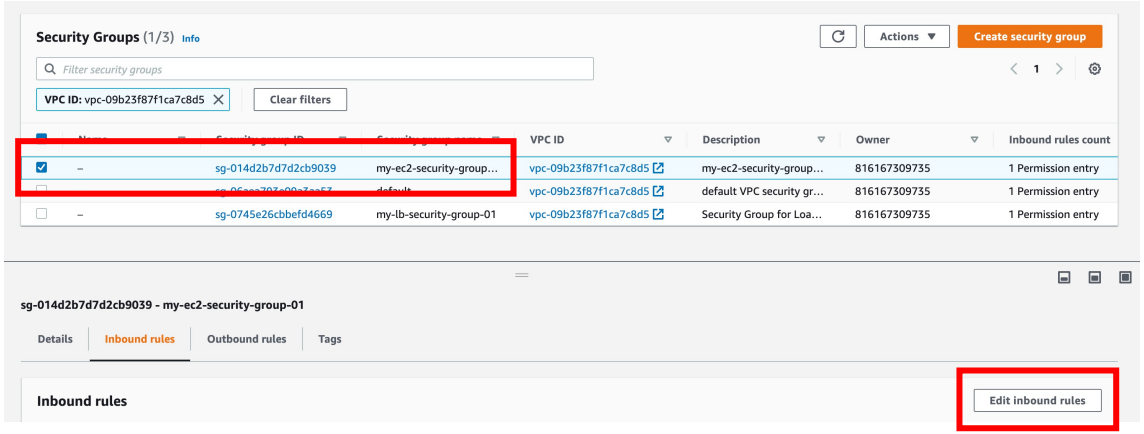
- Verify creation
- Verify status of instances is “InService”
- Note down the DNS name of the ELB

Step 8.1: Add access to EC2 security group

- On EC2 console, open security groups

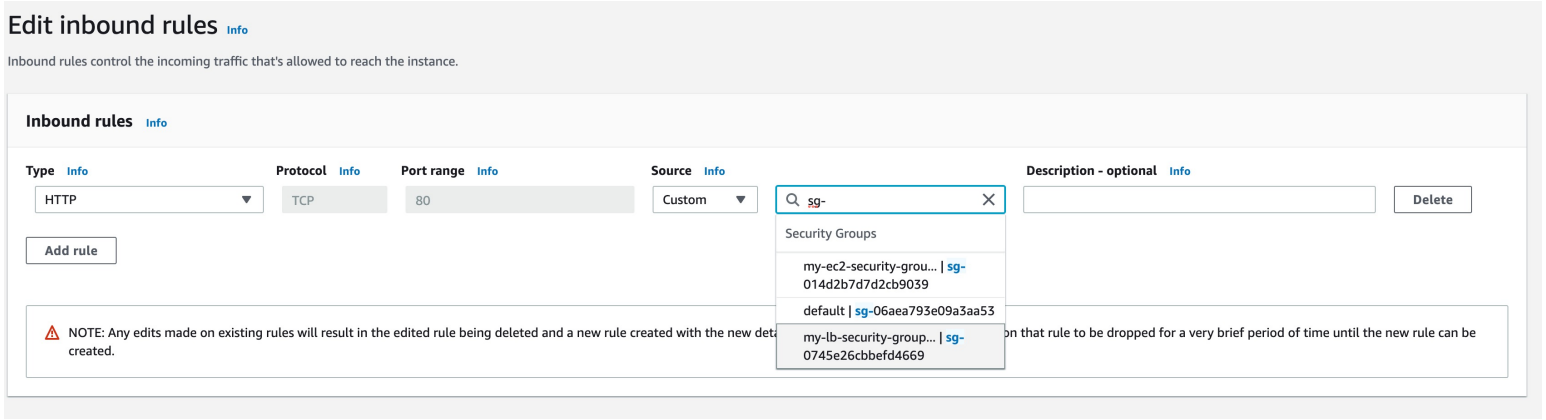


- Select security group and Edit Inbound Rules



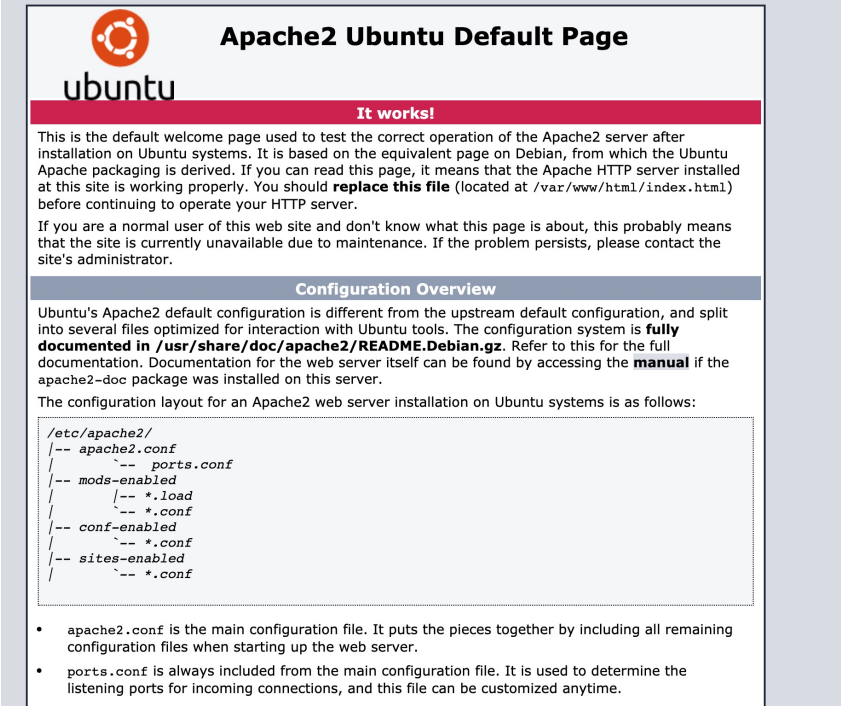
Step 8.2: Add access to EC2 security group

- Edit Inbound rule
  - Protocol: HTTP
  - Port: 80
  - Source: Security group of ELB



Milestone 2

You should be able to access application on EC2 using DNS name of ELB

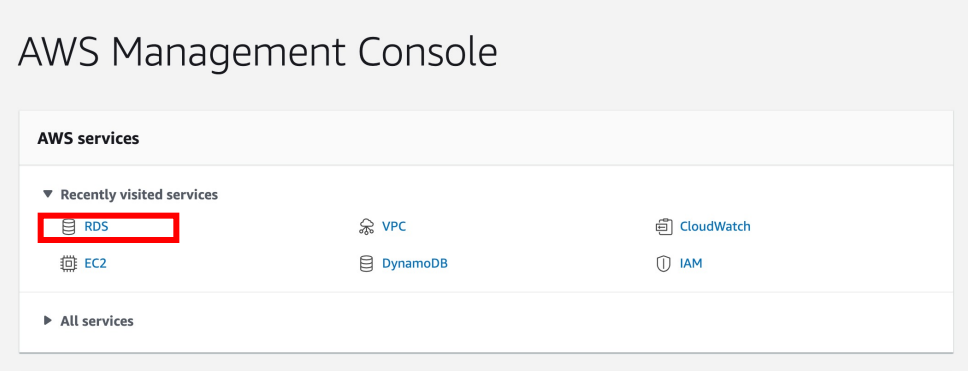


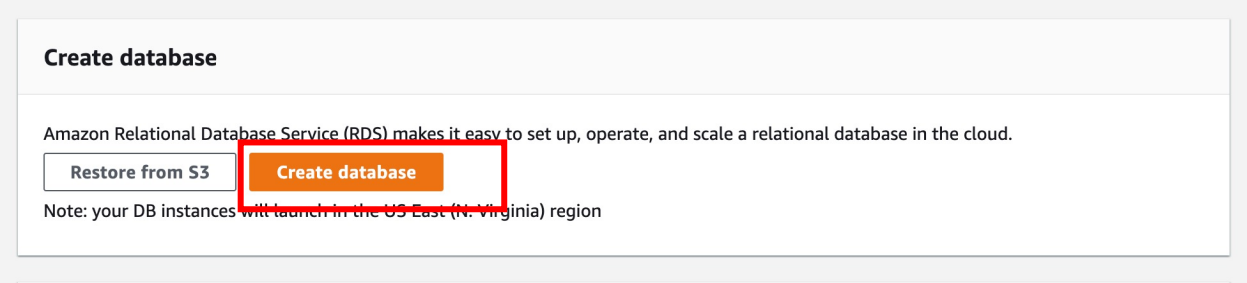
Main Steps – RDS DB Instance

- Select AWS Service
- Create DB Instance
- Add inbound rule to DB Security Group
- Verify Creation

Step 1: Select AWS Service

On AWS Console:

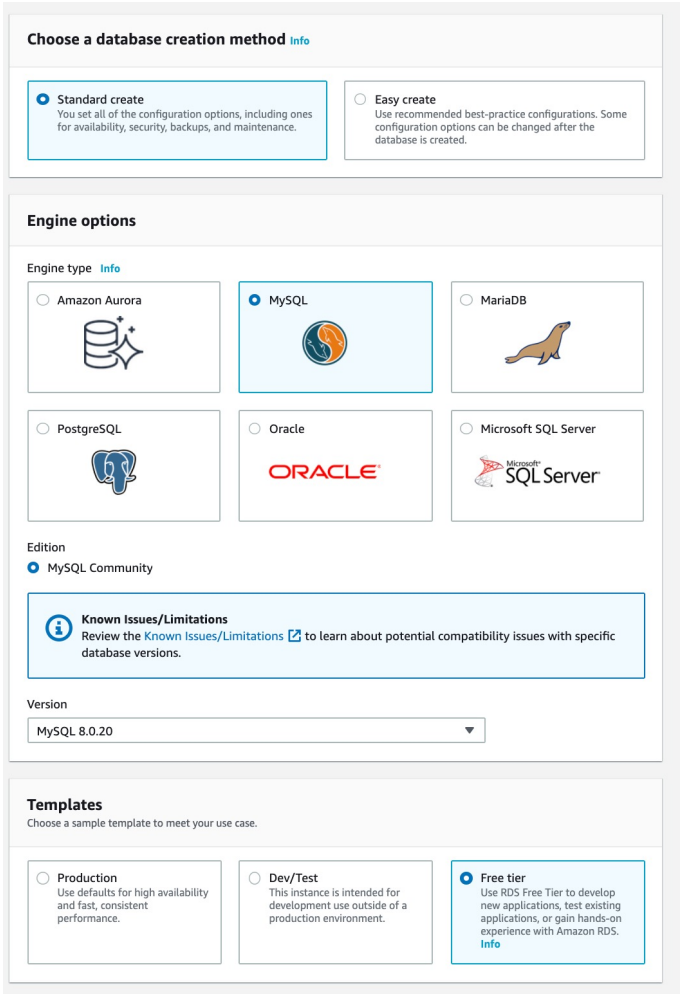




Step 2.1: Create DB Instance

DB Instance Configuration:

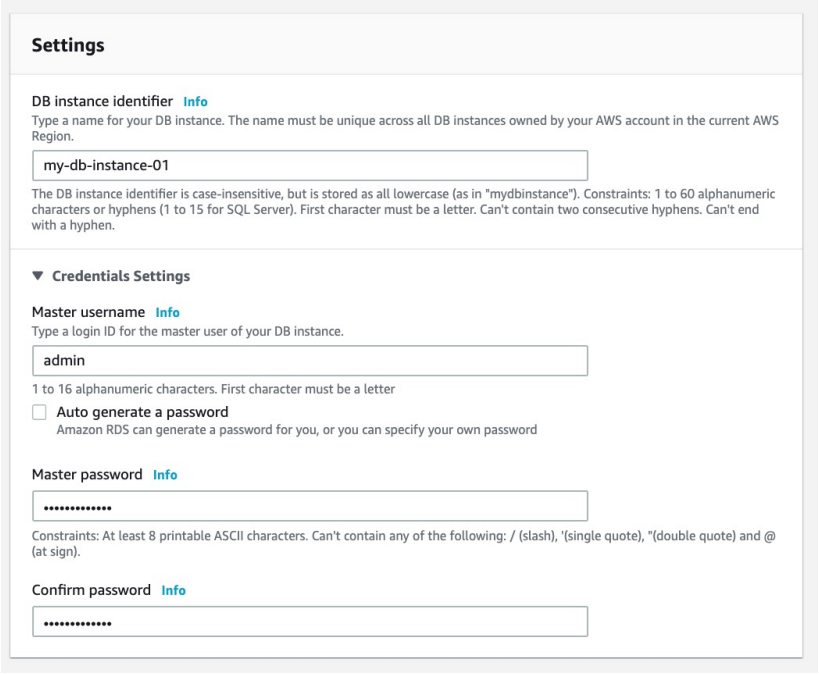
- Creation Method: Standard Create
- Engine Type: MySQL
- Template: Free Tier



Step 2.2: Create DB Instance

DB Instance Configuration:

- DB Instance Identifier: my-db-instance-01
- Master username: admin
- Master password: adminpassword



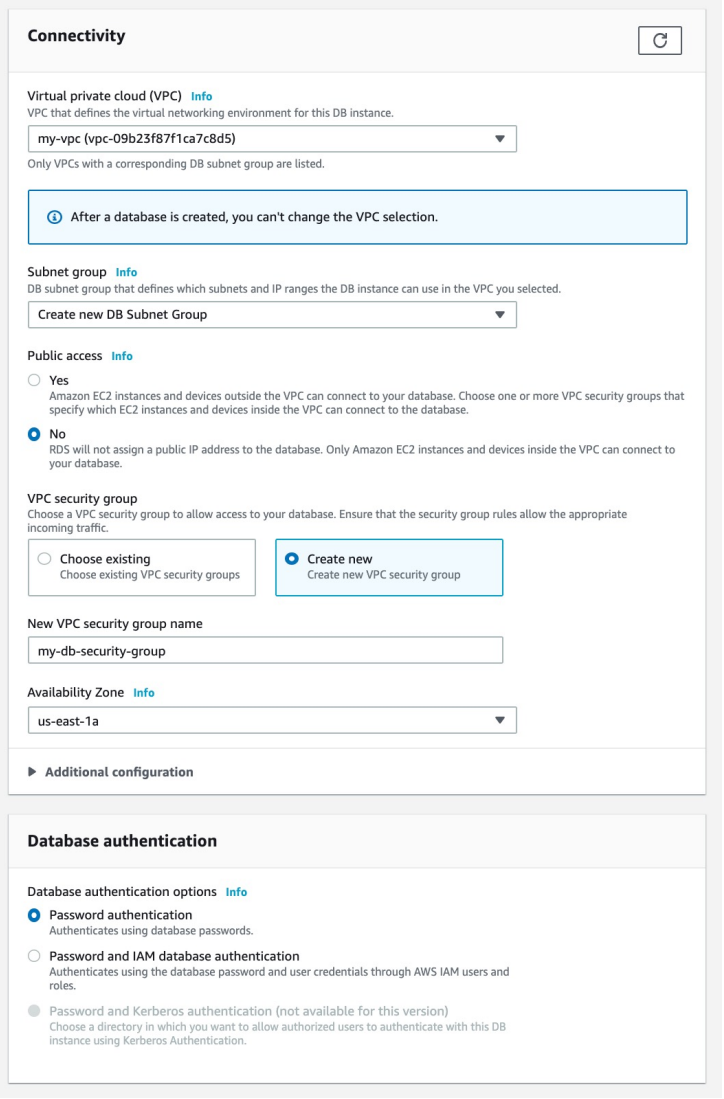
Step 2.3: Create DB Instance

DB Instance Configuration:

- VPC: my-vpc
- Public Access: No
- VPC Security Group: Create new
- New VPC security group name: my-db-security-group
- Availability Zone: us-east-1a
- Authentication: Password authentication
- Additional Configuration:
    - Initial database name: wordpress

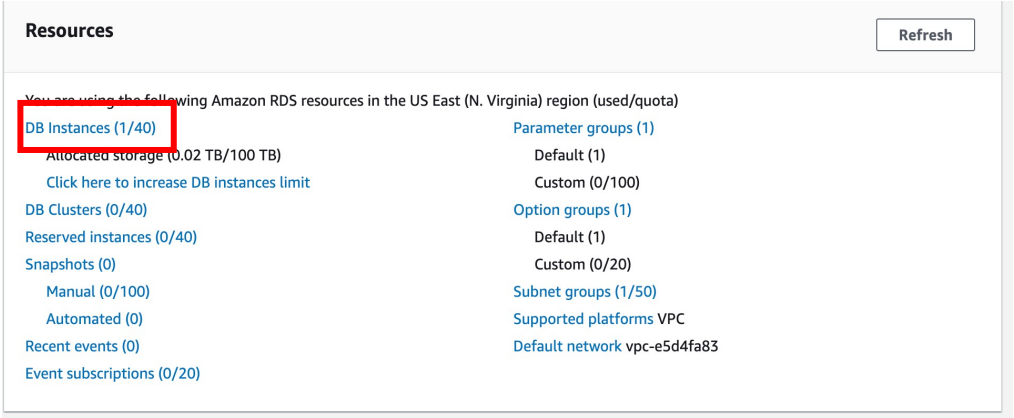
Create Database

Note: It will take some time for the instance to be available. Move on to the next step.

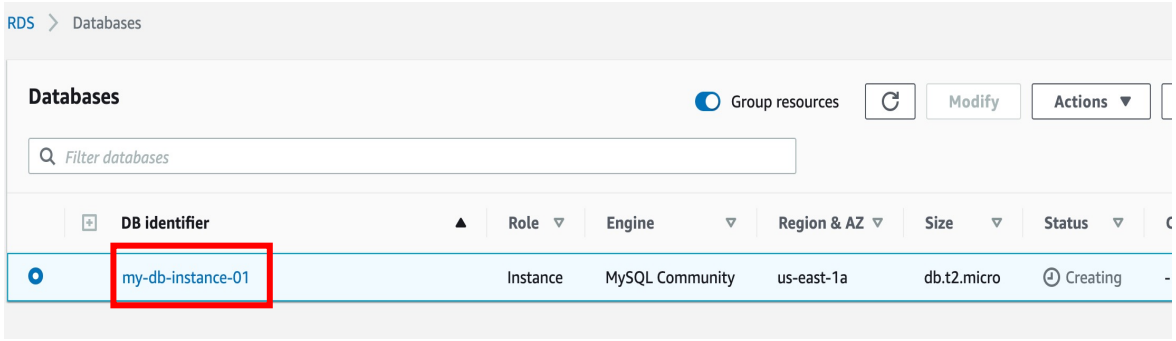


Step 3.1: Add inbound rule to DB Security Group

- On RDS Console:

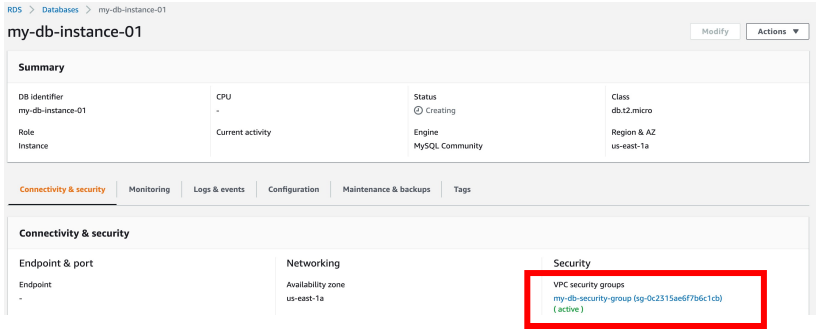


- Select DB instance:

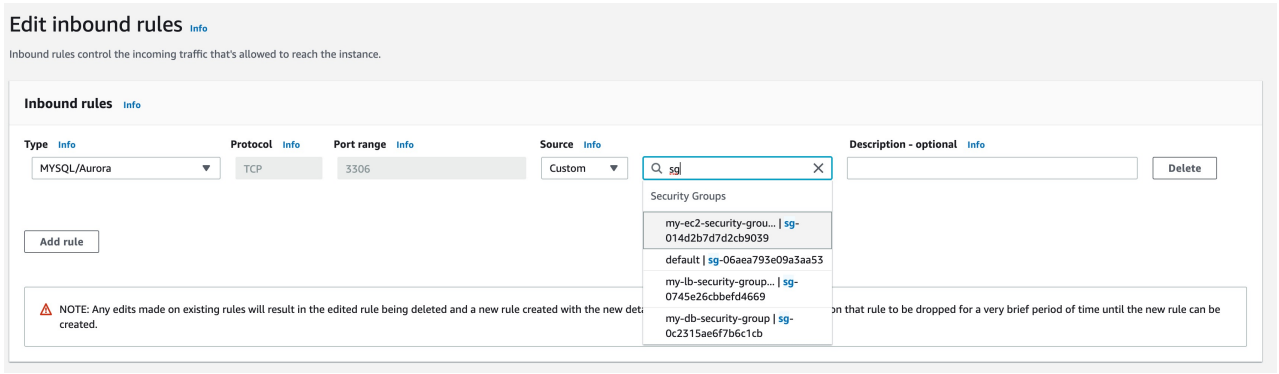


Step 3.2: Add inbound rule to DB Security Group

- Select Security Group

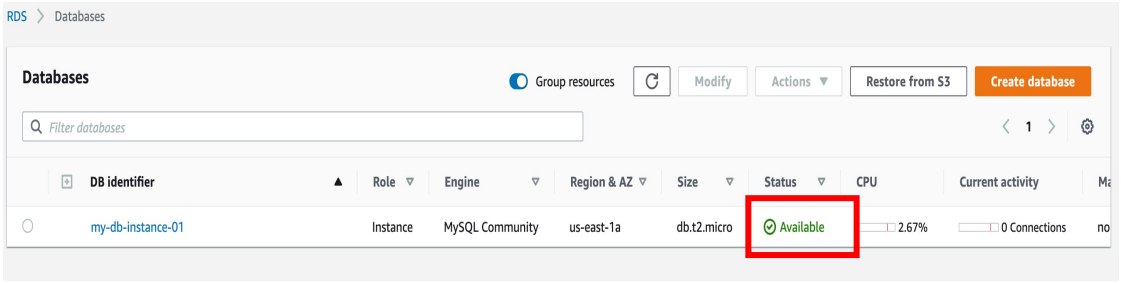


- Add Inbound Rule:
  - Port – 3306, Source – EC2 Security Group



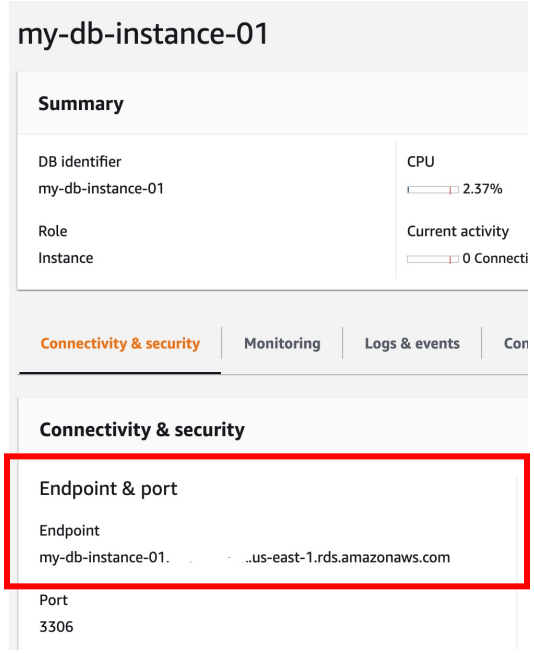
Step 4: Verify Creation

Wait for “Available” status



Click on the “DB identifier”

Note down the endpoint of the DB instance

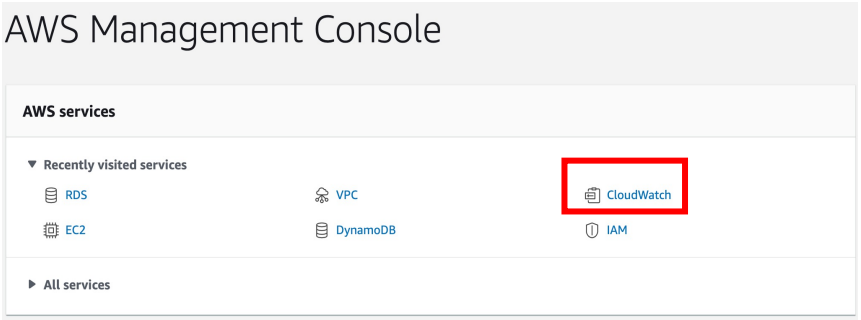


Main Steps - Monitoring

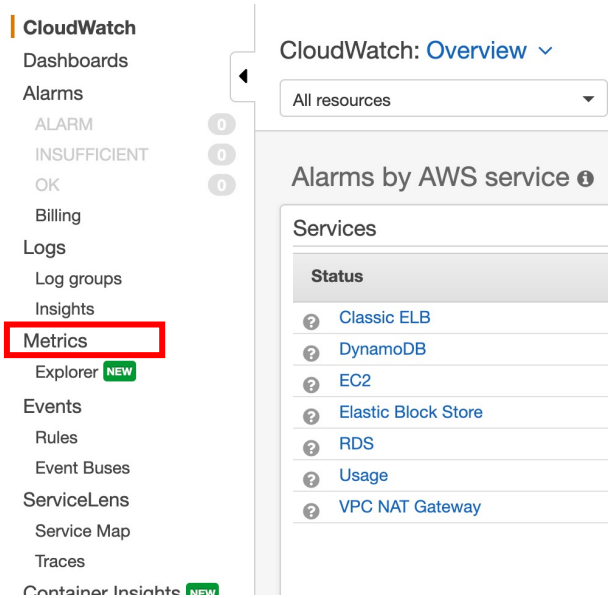
- Select AWS Service
- Observe EC2 CPU utilization

Step 1: Select AWS Service

- On AWS Console:

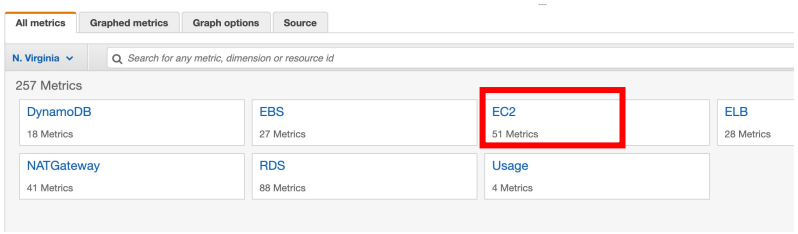


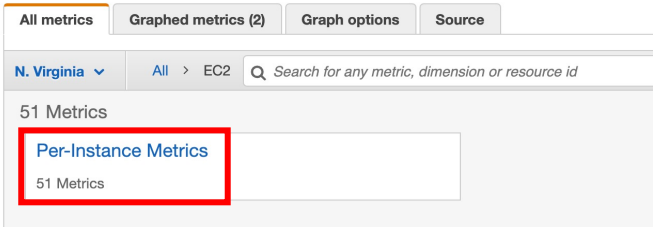
- Open Metrics:



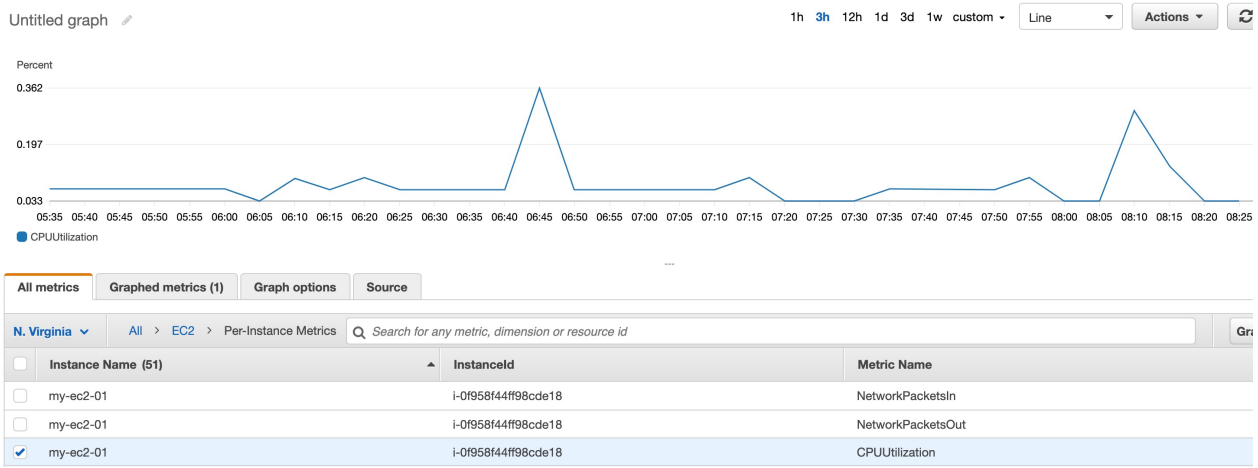
Step 2: Observe EC2 CPU utilization

- Select EC2 -> Per-Instance Metrics





- Select CPU Utilization Metric for my-ec2-01

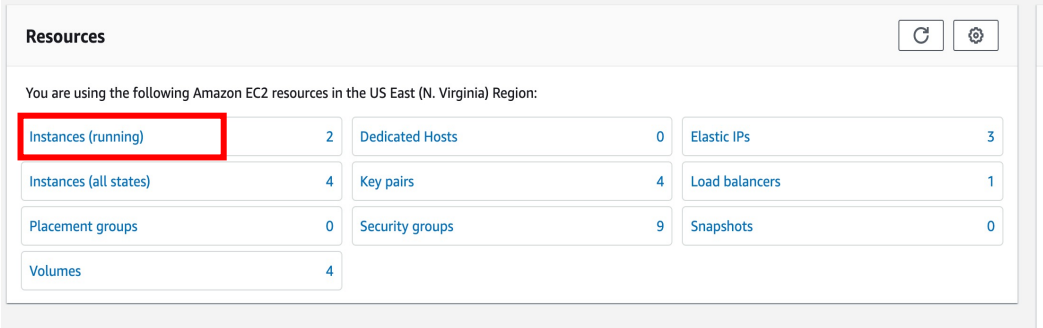


Main Steps – Add Auto Scaling

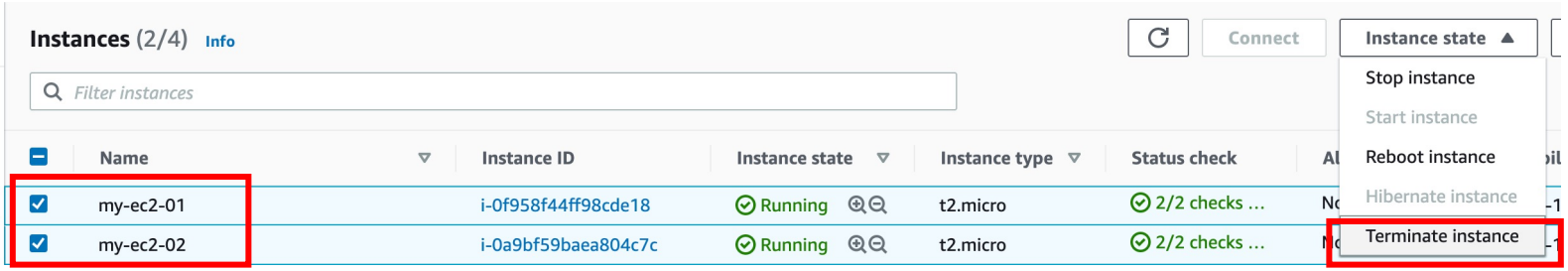
- Delete existing EC2 instances
- Create Launch Configuration
- Create Auto Scaling Group

Step 1: Delete existing instances

- On EC2 Dashboard

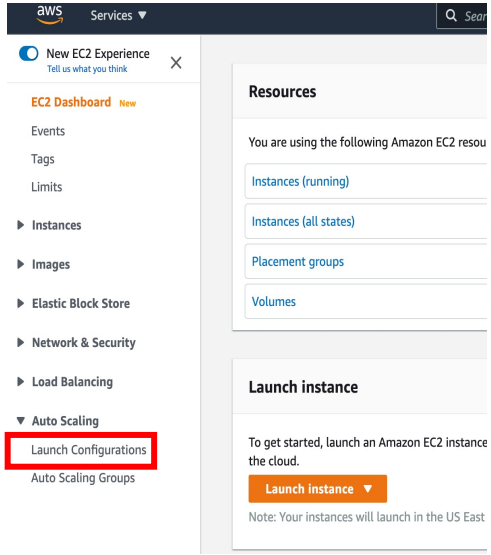


- Select instances and terminate them

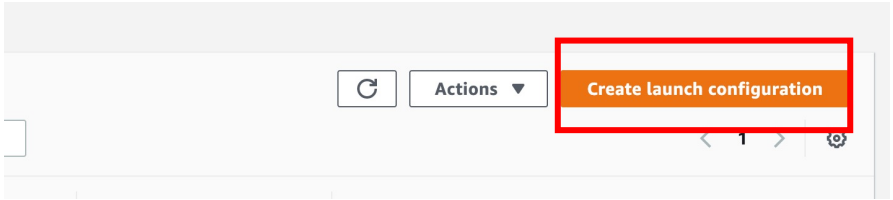


Step 2: Create Launch Configuration
Step 2.1: Select AWS Service

- On EC2 Dashboard



- Create Launch Configuration



Step 2: Create Launch Configuration
Step 2.2: Configuration

Name: my-launch-config-01

AMI: ami-01ee1e203f007488d

Instance Type: t2.micro

Additional configuration -> Advanced details:

Userdata: (as text)

```
#!/bin/bash
sudo bash -c "cat >/etc/wordpress/config-us-east-1.elb.amazonaws.com.php" <<"EOT"
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'admin');
define('DB_PASSWORD', 'adminpassword');
define('DB_HOST', '<REPLACE WITH RDS DB INSTANCE ENDPOINT>');
define('DB_COLLATE', 'utf8_general_ci');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>
EOT
sudo service apache2 restart
```

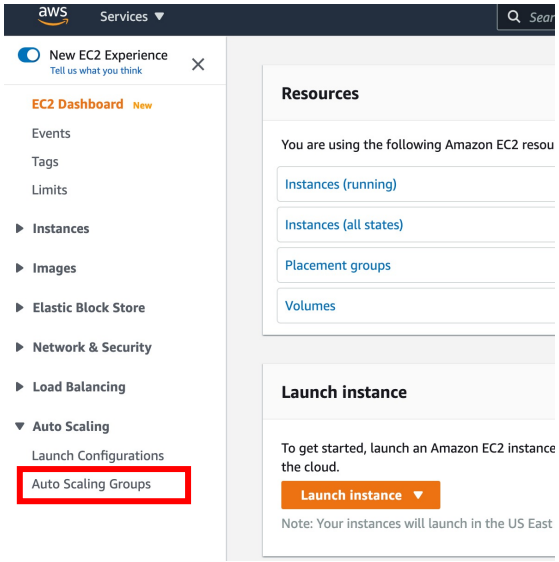
Username, password and hostname of RDS DB instance

Security Group: my-ec2-security-group-01

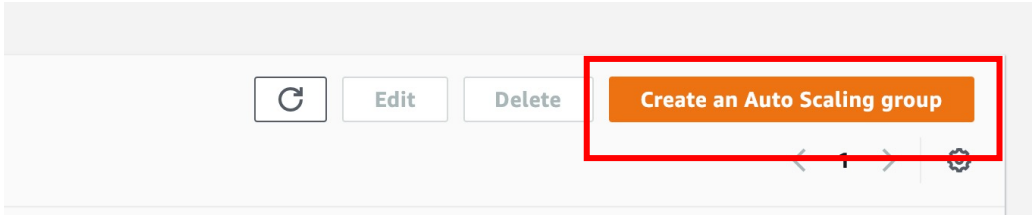
Key Pair: my-ec2-key-pair

Step 3: Create Auto Scaling Group
Step 3.1: Select AWS Service

On EC2 Dashboard



Create Auto Scaling Group

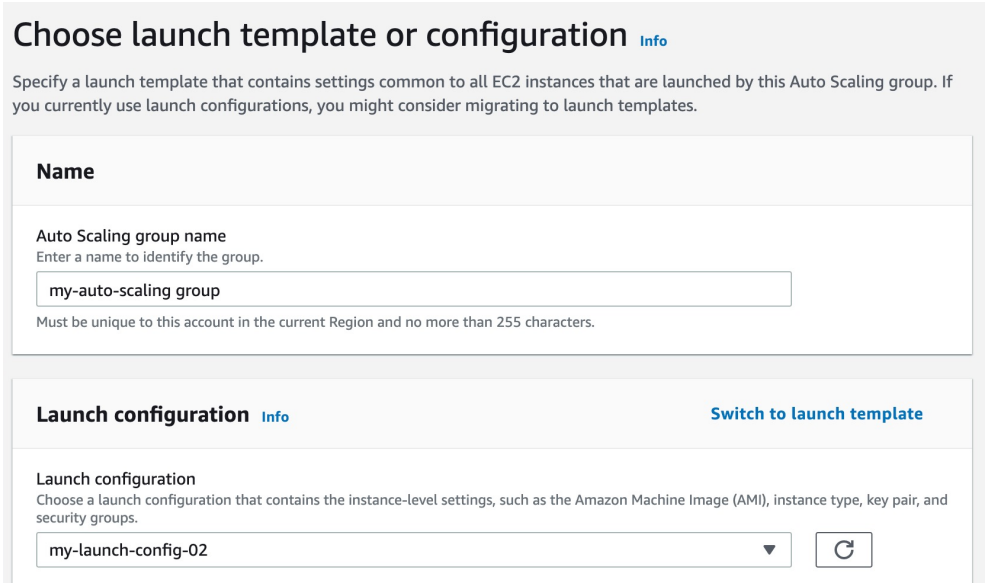


Step 3: Create Auto Scaling Group
Step 3.2: Choose Launch Configuration

Name: my-auto-scaling-group

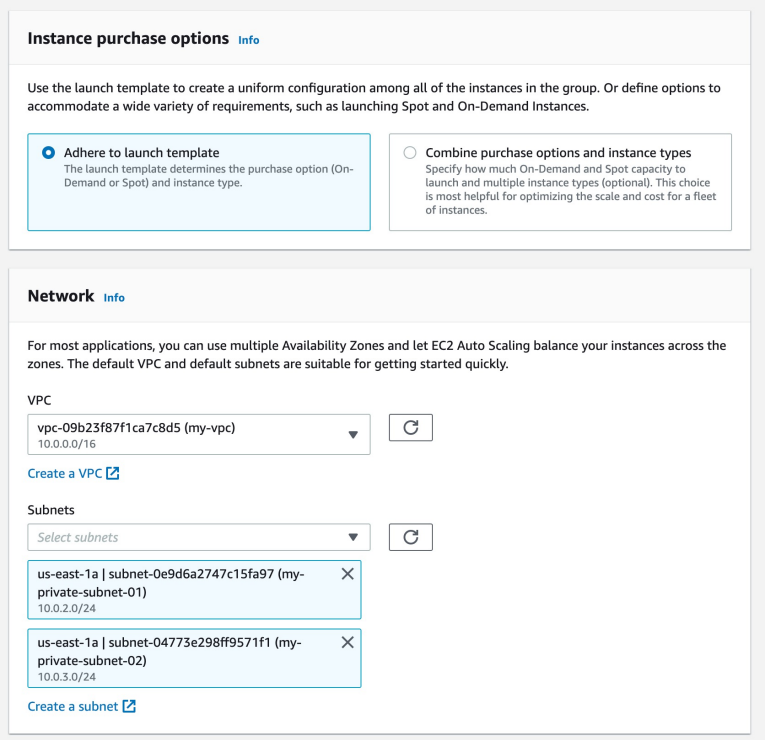
Switch to launch configuration

Launch Template: my-launch-config-01



Step 3: Create Auto Scaling Group
Step 3.3: Configure Settings

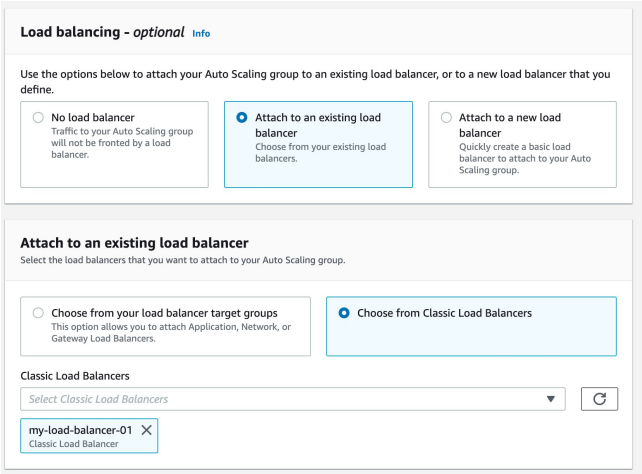
- Instance Purchase Options:
  - Adhere to launch template
- Network:
  - VPC: my-vpc
  - Subnets: my-private-subnet-01
  - my-private-subnet-02

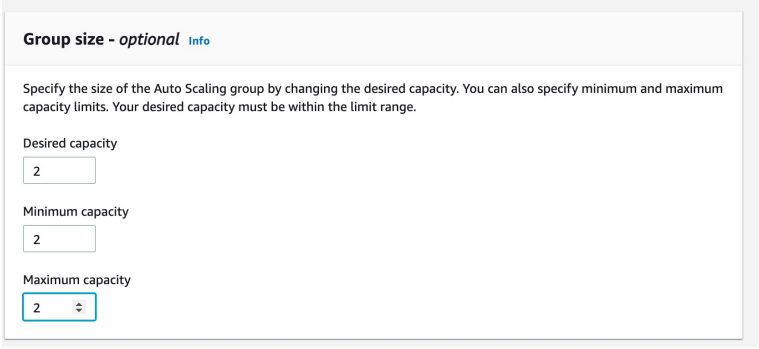


Step 3: Create Auto Scaling Group
Step 3.4: Configure advanced options

- Load balancing
  - Attach to an existing load balancer
  - Choose my-load-balancer-01
- Group Size
  - Set a constant capacity of 2
- Tags
  - Name: my-ec2
  - Tag new Instances - checked

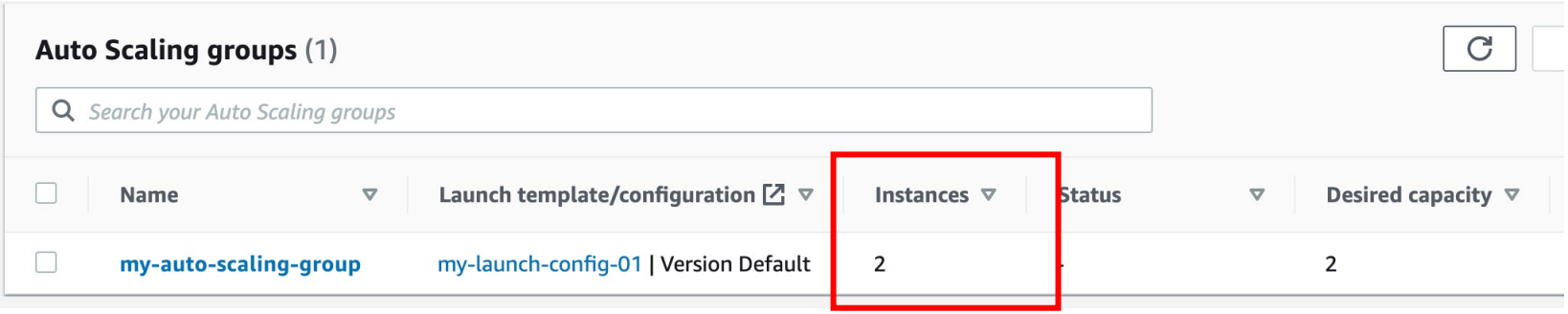
Create Auto Scaling Group





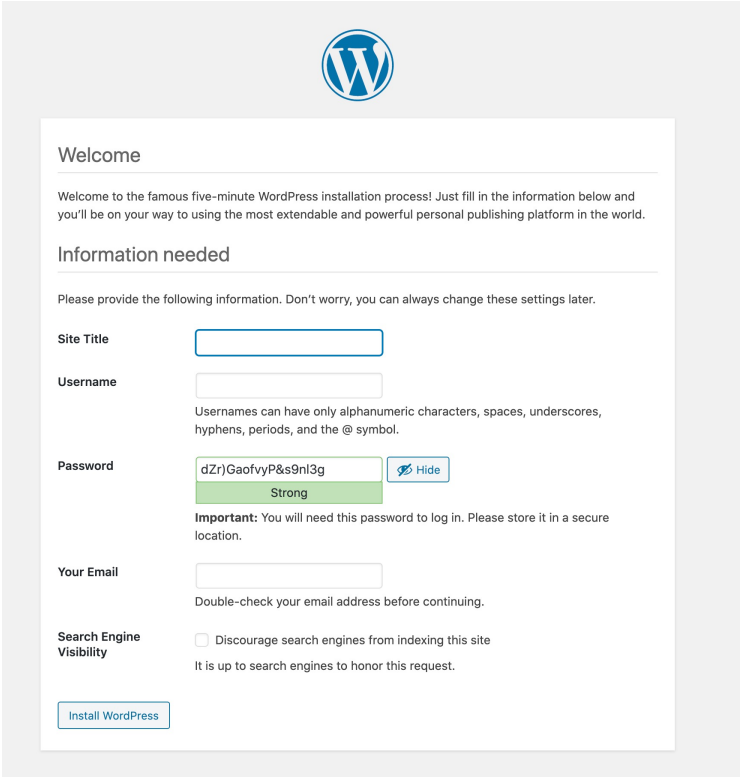
Step 3: Create Auto Scaling Group
Step 3.5: Verify

- Verify number of Instances (=2)
- Verify 2 Running Instances on EC2 dashboard



Milestone 3

Access ELB again as : <ELB-DNS-NAME>/blog



Clean up

Remember to delete resources:

- Delete auto scaling group – deletes the instances too (docs)
- Delete ELB (docs)
- Delete RDS DB instance (docs)
- Delete NAT Gateways (docs)
- Release Elastic IPs associated with NATs (docs)
- Delete Internet gateways (docs)

Summary

- Core services covered
  - Network: VPC
  - Compute: EC2, ELB, Auto Scaling
  - Storage: S3, EBS
  - Database: RDS
  - Monitoring: Cloudwatch