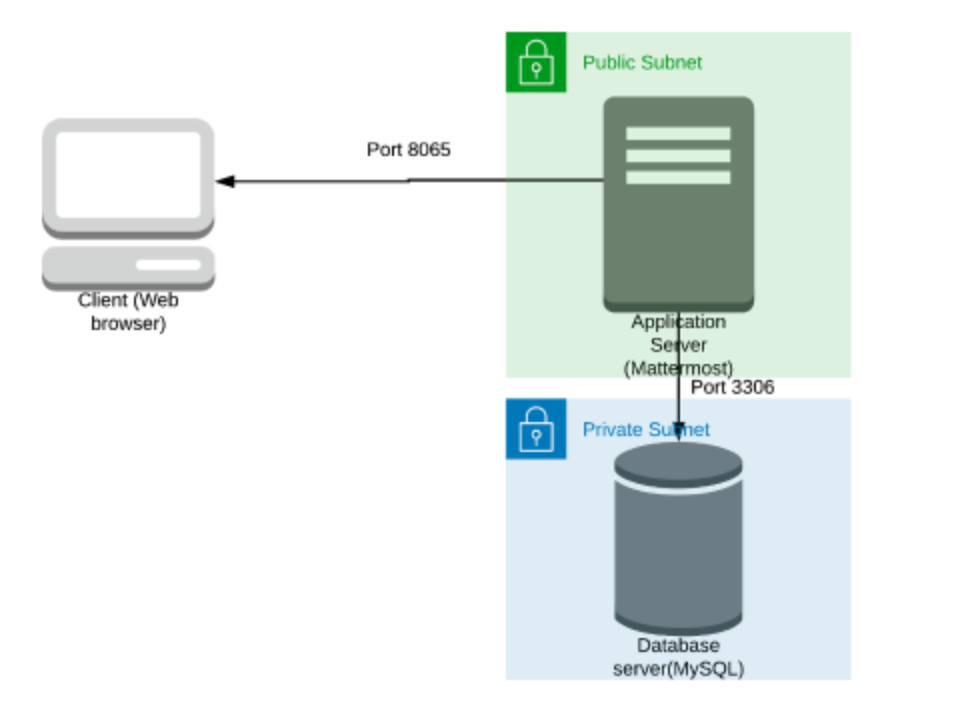
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Declaration** |  | | | | |
| Questions in this exercise are intentionally complex and could be convoluted or confusing. This is by design and to simulate real life situations where customers seldom give crystal clear requirements and ask unambiguous questions. | | | | | |
|
|
| I have read the above statement and agree to these conditions | | | | | |
| I AGREE | Vishwajeet bharti | | | | |
| <Enter your name above this line to indicate that you are in agreement> | | | | |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Instructions** |  |  |  |  |  |
| Every screenshot requested in this workbook is compulsory and carries 0.5 marks | | | | | |
| Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed. | | | | | |
| All screenshots must be in the order mentioned under "Expected Screenshots" for every step | | | | | |
| DO NOT WAIT UNTIL THE LAST MINUTE. | | | | | |
| The file should be renamed in the format BATCH\_FIRSTNAME\_LASTNAME\_PROJECT1.  For example: IITR\_FSD\_VIJAY\_DWIVEDI\_PROJECT1.docx | | | | | |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Resource Clean Up** | |  |  |  |  |
| Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you’ve completed your lab or project is always necessary. This is true whether you’re doing a lab or implementing a project at your workplace. | | | | | |
| After completing the lab, make sure to delete each resource created in reverse chronological order. | | | | | |

**Scenario**

Team communication and instant messaging solutions are an integral part of any business environment today. As of 2020, the total number of users of Slack and Microsoft Teams exceeded 20 million.

Some organizations might have compliance policies in place which do not allow them to use services managed by third parties. They will prefer solutions that can be managed and hosted on servers controlled by them. The same will extend to communication solutions as well.

**Architecture diagram**

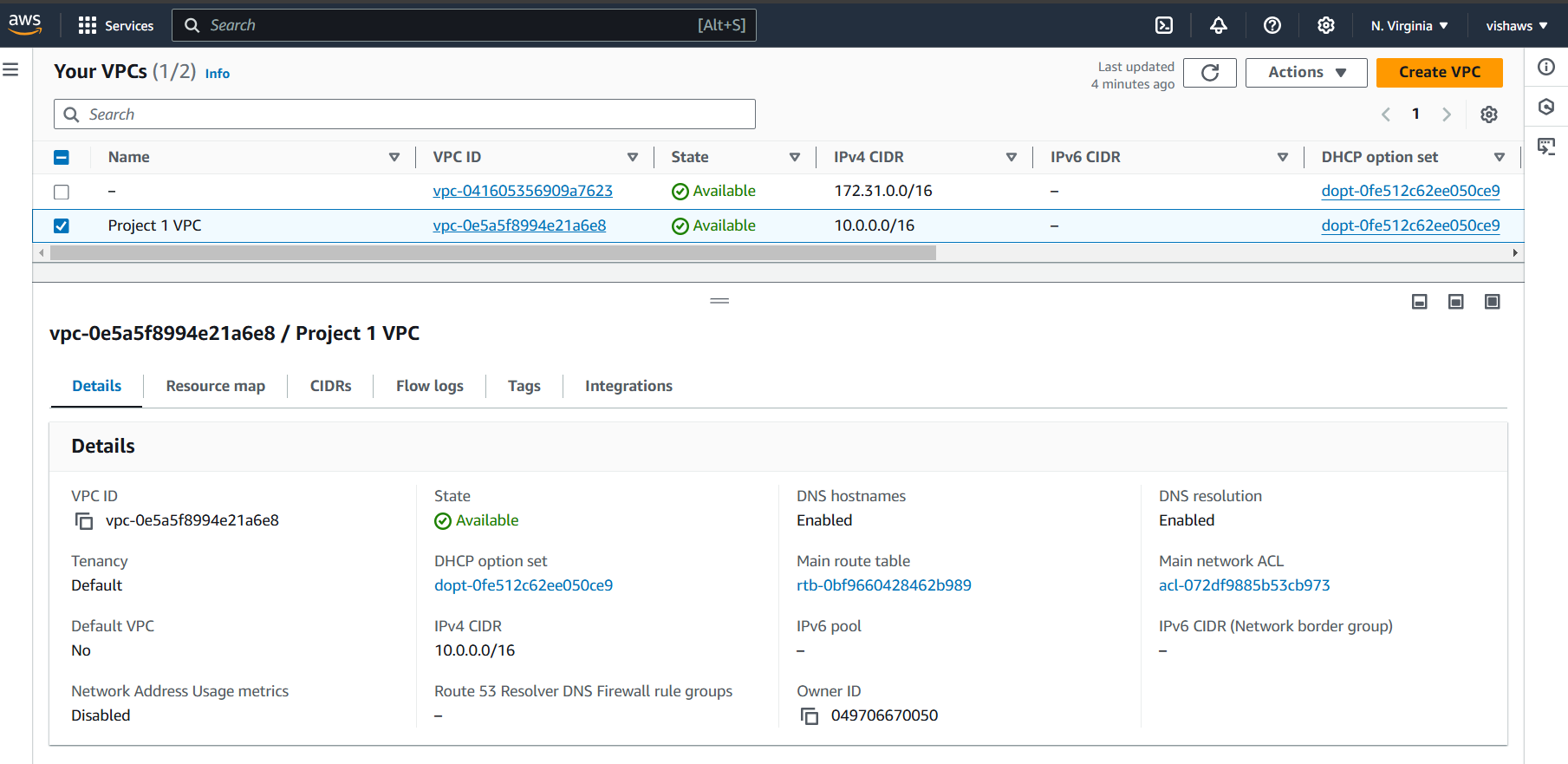
****

|  |  |
| --- | --- |
| **Architecture Implementation** | |
| 1 | Implement 2 different subnets (one public and the other private) in a custom VPC |
| 2 | Install and configure MySQL on an Amazon Linux 2 instance on the private subnet using the instructions provided. (Hint: Use a bastion host and a NAT gateway) |
| 3 | Install and configure Mattermost on an Amazon Linux 2 instance on the public subnet using the provided instructions. |
| 4 | Configure the security groups to allow the ports as shown in the architecture. |
| 5 | Test the installation by accessing the IP of the public instance in a browser via the port 8065. |

**Step 1: VPC and Subnet Creation**

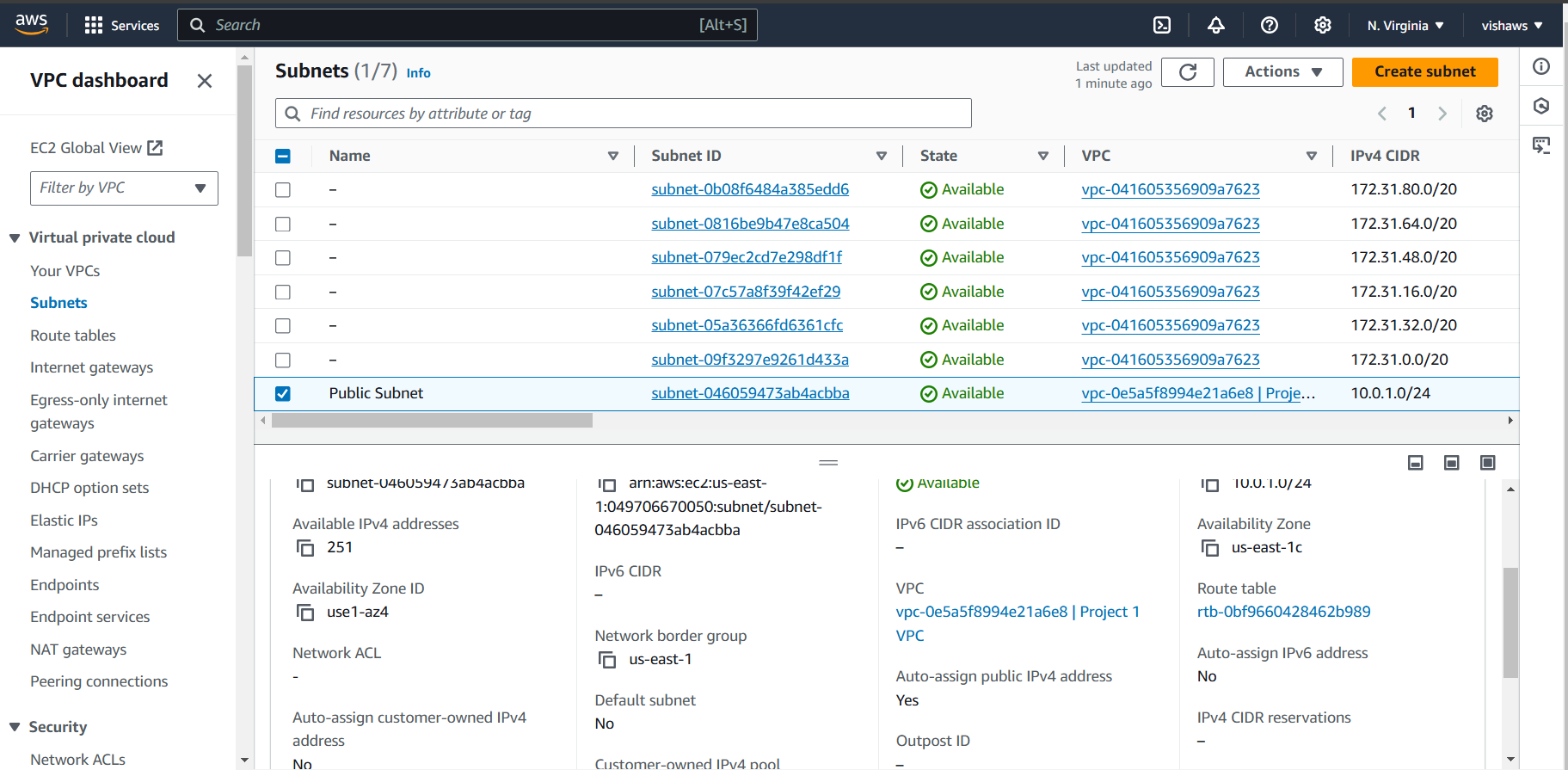
|  |  |  |  |
| --- | --- | --- | --- |
| Step number | A |  |  |
| Step name | Creation of VPC |  |  |
| Instructions | 1) Navigate to VPC using the Services button at the top of the screen  2) Select "Your VPCs" on the left side of the screen  3) Click on "Create VPC"  4) Enter the following fields :  Name: Project 1 VPC  IPv4 CIDR Block : 10.0.0.0/16  The rest of the options can be ignored  5) Select "Create VPC"  6) Select the VPC and click on Actions->Edit DNS hostnames  7) Enable DNS hostnames and click on Save |  |  |
| Expected screenshots | 1. Created VPC with properties visible |  |  |

**<Insert Screenshot a(1) here>**



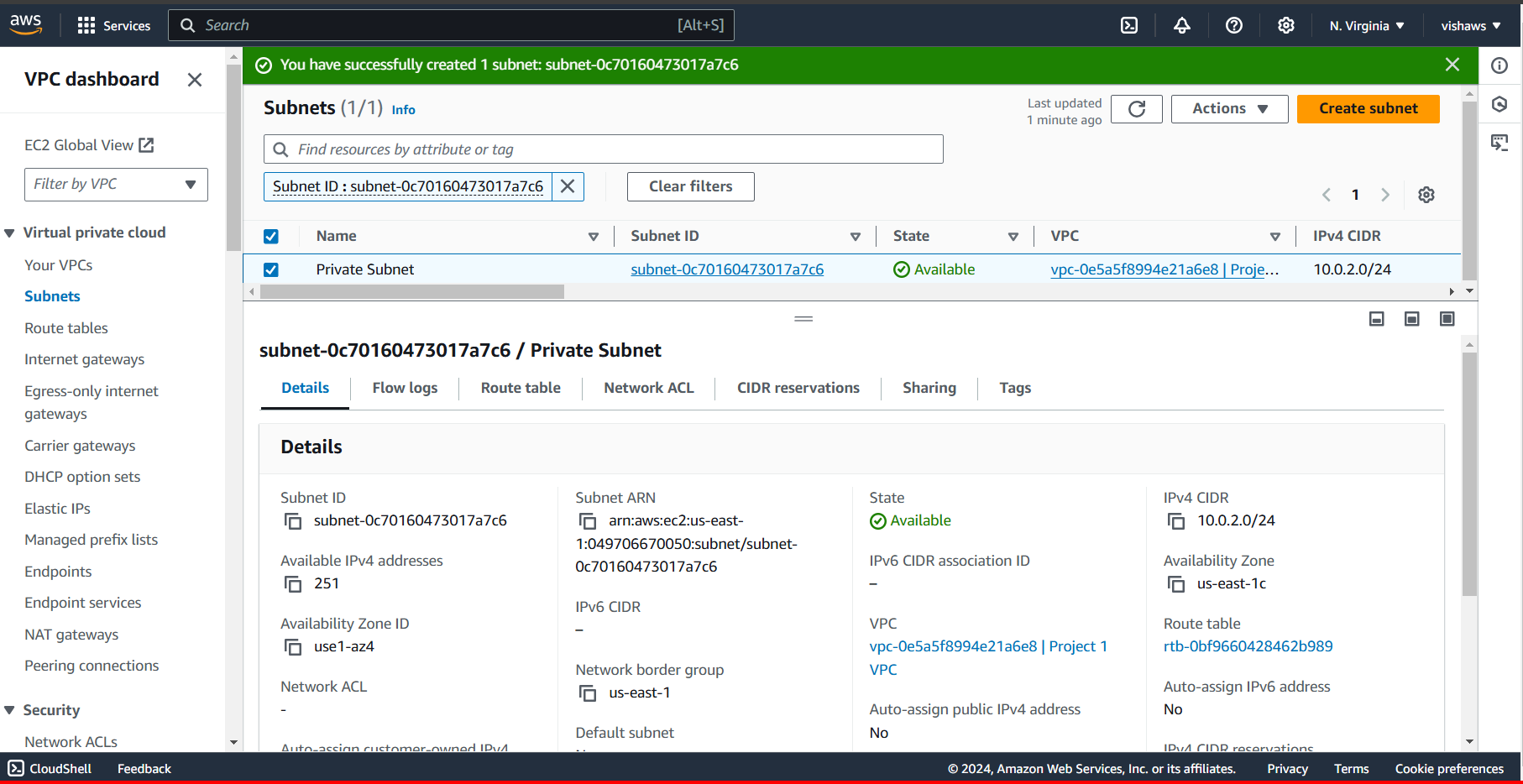
|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Creation of public subnet |  |
| Instructions | 1) Navigate to VPC->Subnets  2) Click on "Create Subnet"  3) Enter the following fields  Name tag : Public Subnet  VPC : Select the Project 1 VPC  IPv4 CIDR block : 10.0.1.0/24  The other options can be ignored  4) Click on Create  5) Once the subnet has been created, select the subnet and click on Actions->Modify Auto-assign IP settings  6) Enable the option "Auto assign IPv4" and select Save |  |
| Expected screenshots | 1. Subnet Creation screen |  |

**<Insert Screenshot b(1) here>**



|  |  |
| --- | --- |
| Step number | C |
| Step name | Creation of private subnet |
| Instructions | 1) Navigate to VPC->Subnets  2) Click on "Create Subnet"  3) Enter the following fields  Name tag : Private Subnet  VPC : Select the Project 1 VPC  IPv4 CIDR block : 10.0.2.0/24  The other options can be ignored  4) Click on Create |
| Expected screenshots | 1. Subnet Creation screen |

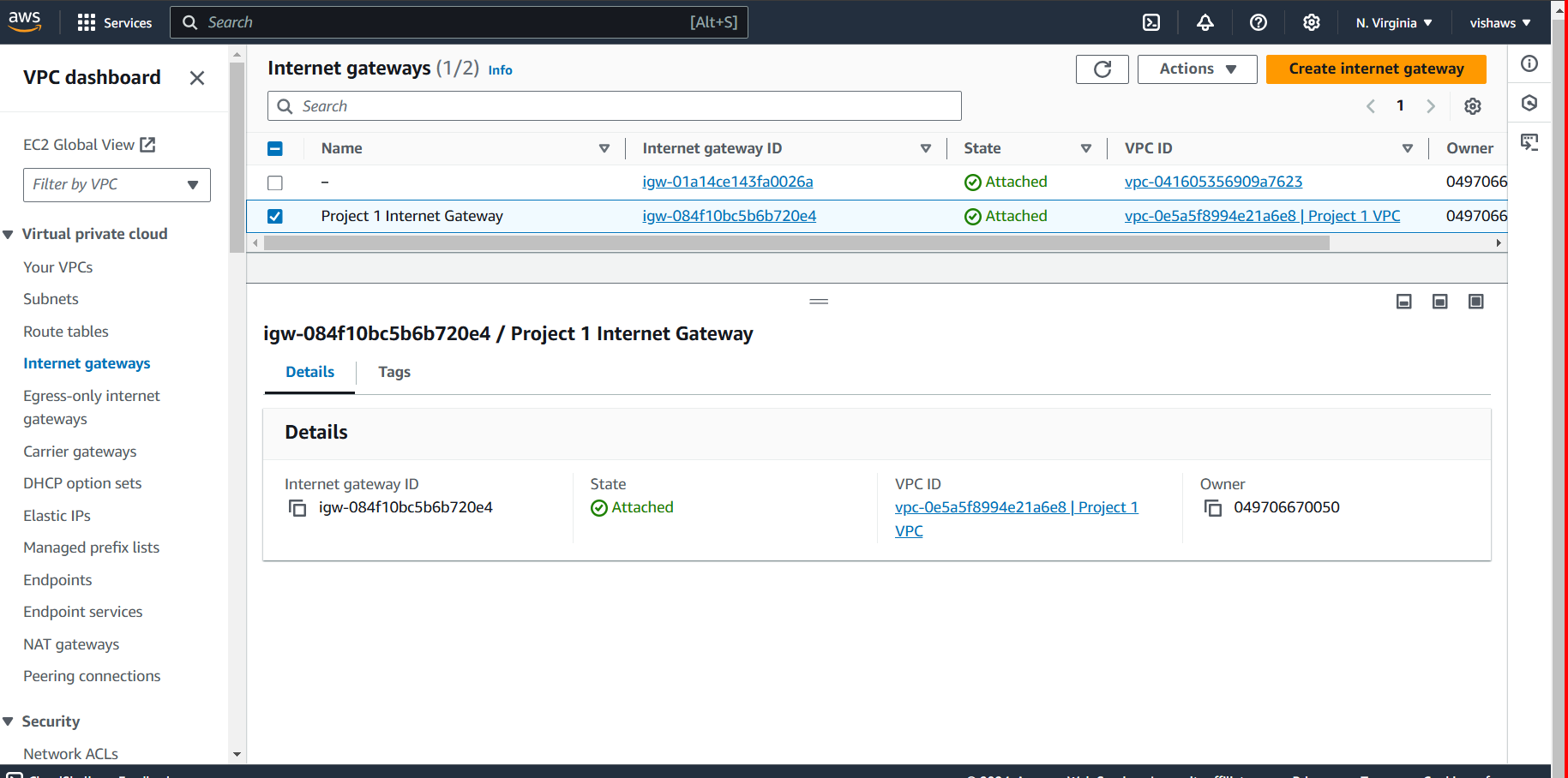
**<Insert Screenshot c(1) here>**



**Step 2 : Internet Gateway and VPC**

|  |  |  |
| --- | --- | --- |
| Step number | a |  |
| Step name | Creation and Configuration of Internet Gateway |  |
| Instructions | 1) Navigate to VPCs->Internet Gateway  2) Click on "Create Internet Gateway"  3) Enter the name tag "Project 1 Internet Gateway" and click on "Create Internet Gateway"  4) After the gateway is created, select it and click on Actions->Attach to VPC  5) Select the Project 1 VPC and click on "Attach Internet Gateway" |  |
| Expected screenshots | 1. Creation of Internet Gateway |  |

**<Insert Screenshot a(1) here >**



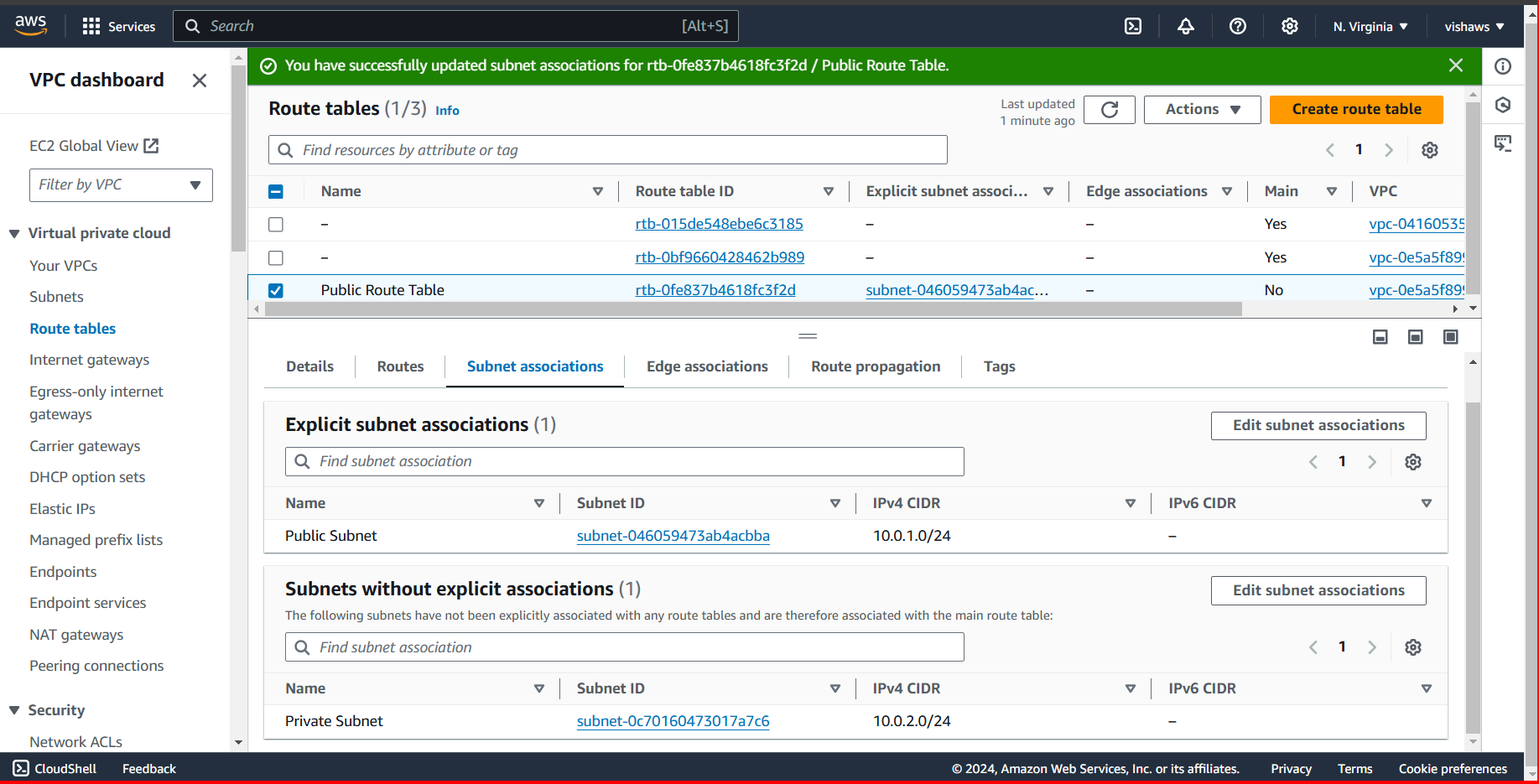
|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Creation of public route table |  |
| Instructions | 1) Navigate to VPC -> Route Tables and click on Create Route table  2) Enter the name tag "Public Route Table", select the Project 1 VPC from the dropdown and click on Create  3) Once the route table is created, select it and select the Routes tab below the list of route tables  4) Click in Edit Routes and add the following route (Don't edit the existing one)  - Destination : 0.0.0.0/0  - Target : Select Internet Gateway and the select the Project 1 Internet Gateway  Click on Save Routes  5) Select the Subnet Associations tab and click on Edit Subnet Associations  6) Select the Public Subnet from the list and click on Save |  |
| Expected screenshots | 1. Route list of the route table 2. Subnet Associations of the route table |  |

**<Insert Screenshot b(1) here>**

A screenshot of a computer

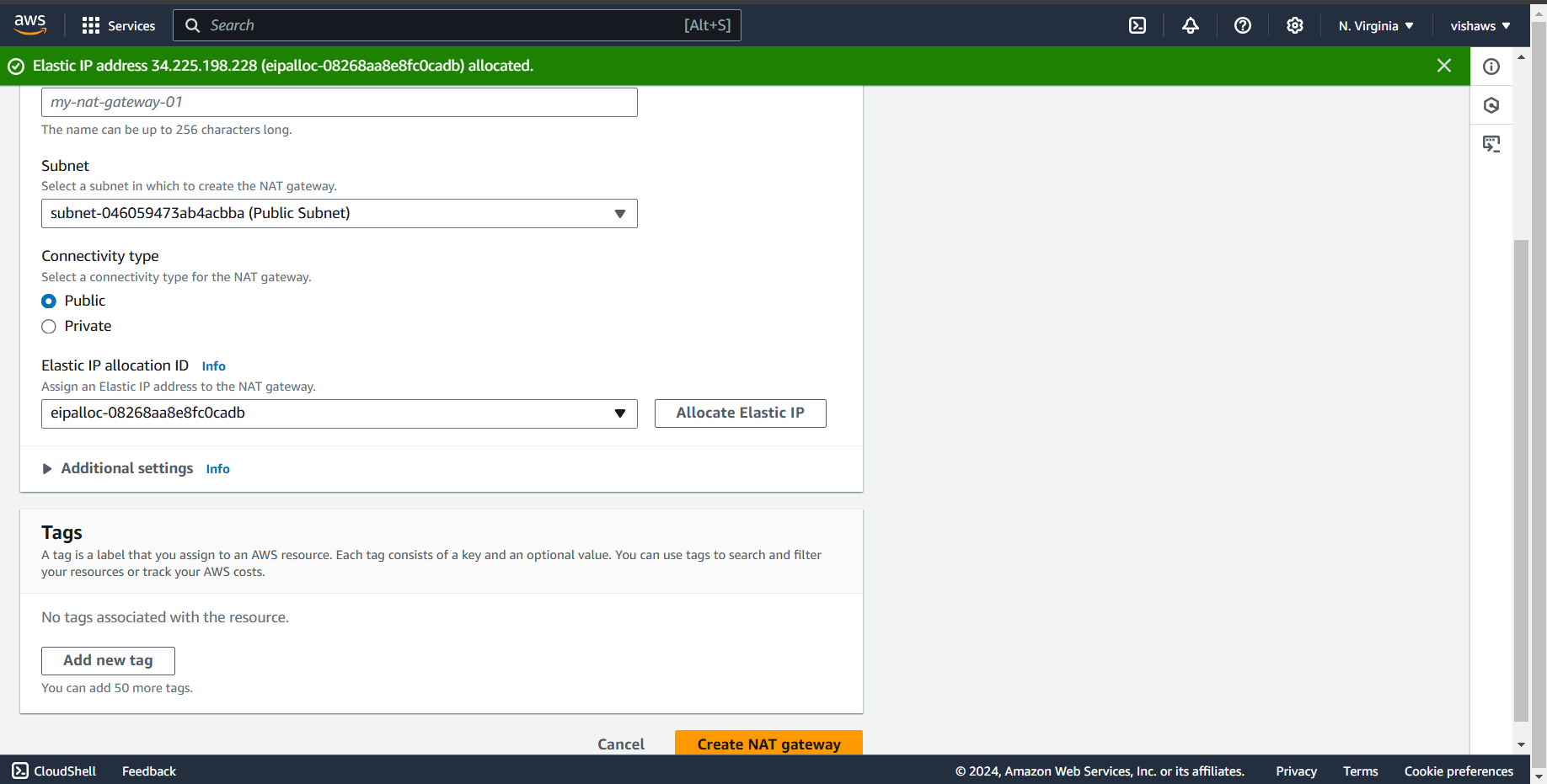
Description automatically generated

**<Insert Screenshot b(2) here>**

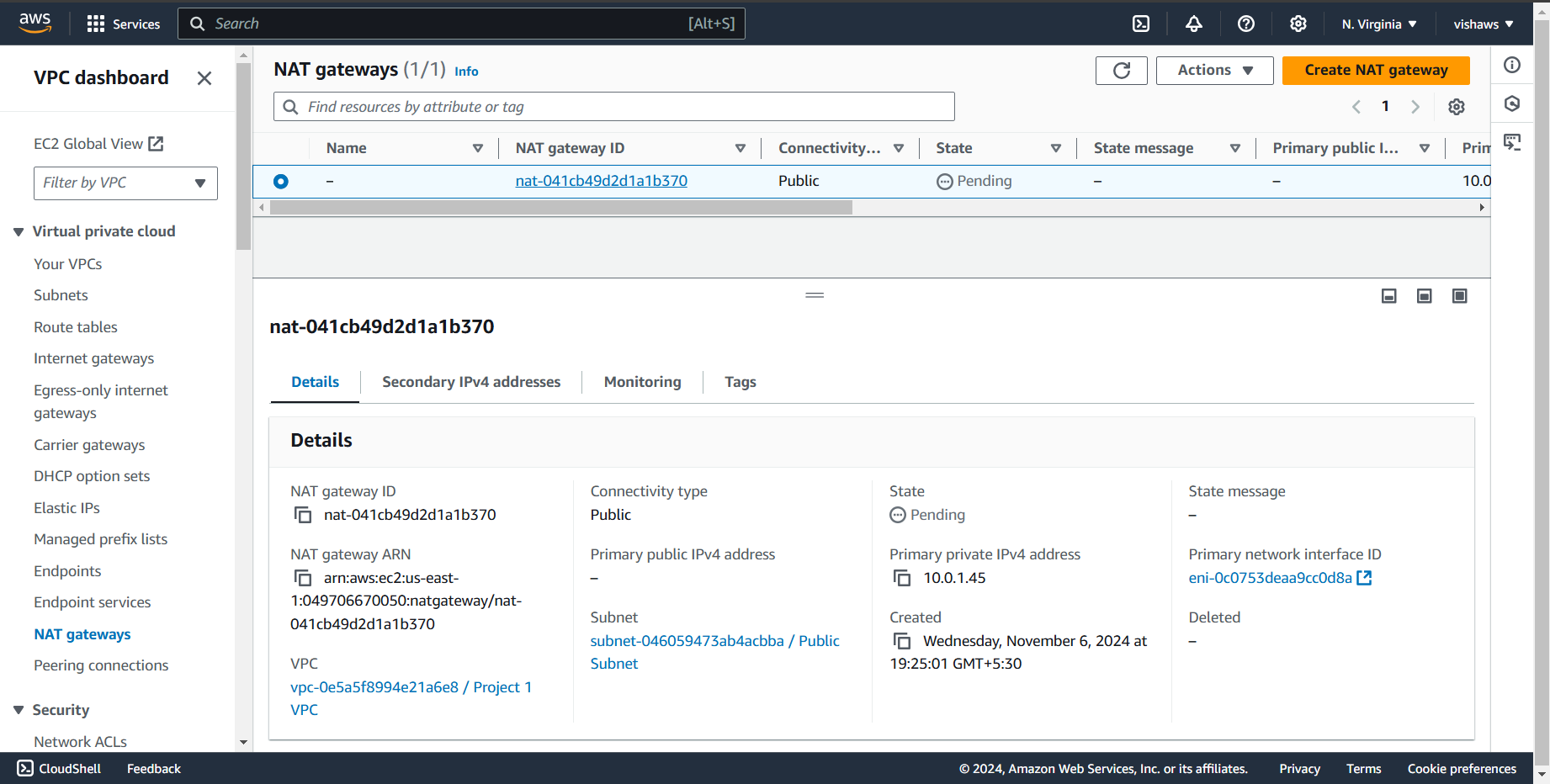


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | c |  |  |  |
| Step name | Creation of NAT gateway |  |  |  |
| Instructions | 1) Navigate to VPC using the Services button at the top of the screen  2) Select NAT Gateway at the left side of the screen  3) Click on Create NAT Gateway  - Deploy it in the public subnet  - Connectivity type : Public  - Allocate an elastic IP by clicking on “Allocate Elastic IP”  4) Click on “Create NAT Gateway” to create the gateway |  |  |  |
| Expected screenshots | 1. NAT gateway creation details 2. Gateway after creation |  |  |  |
|  |  |  |  |  |

**<Insert Screenshot c(1) here>**

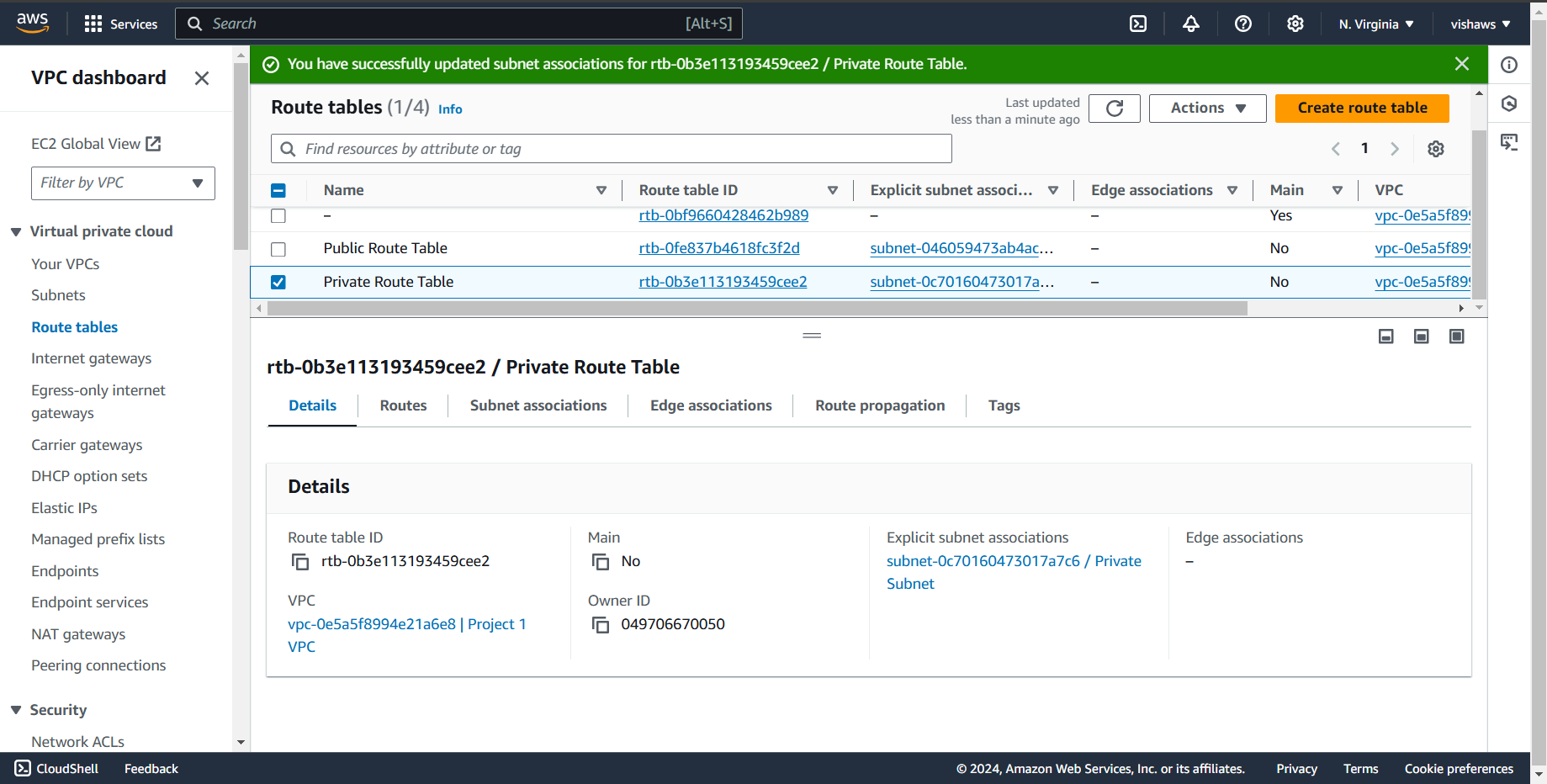


**<Insert Screenshot c(2) here>**

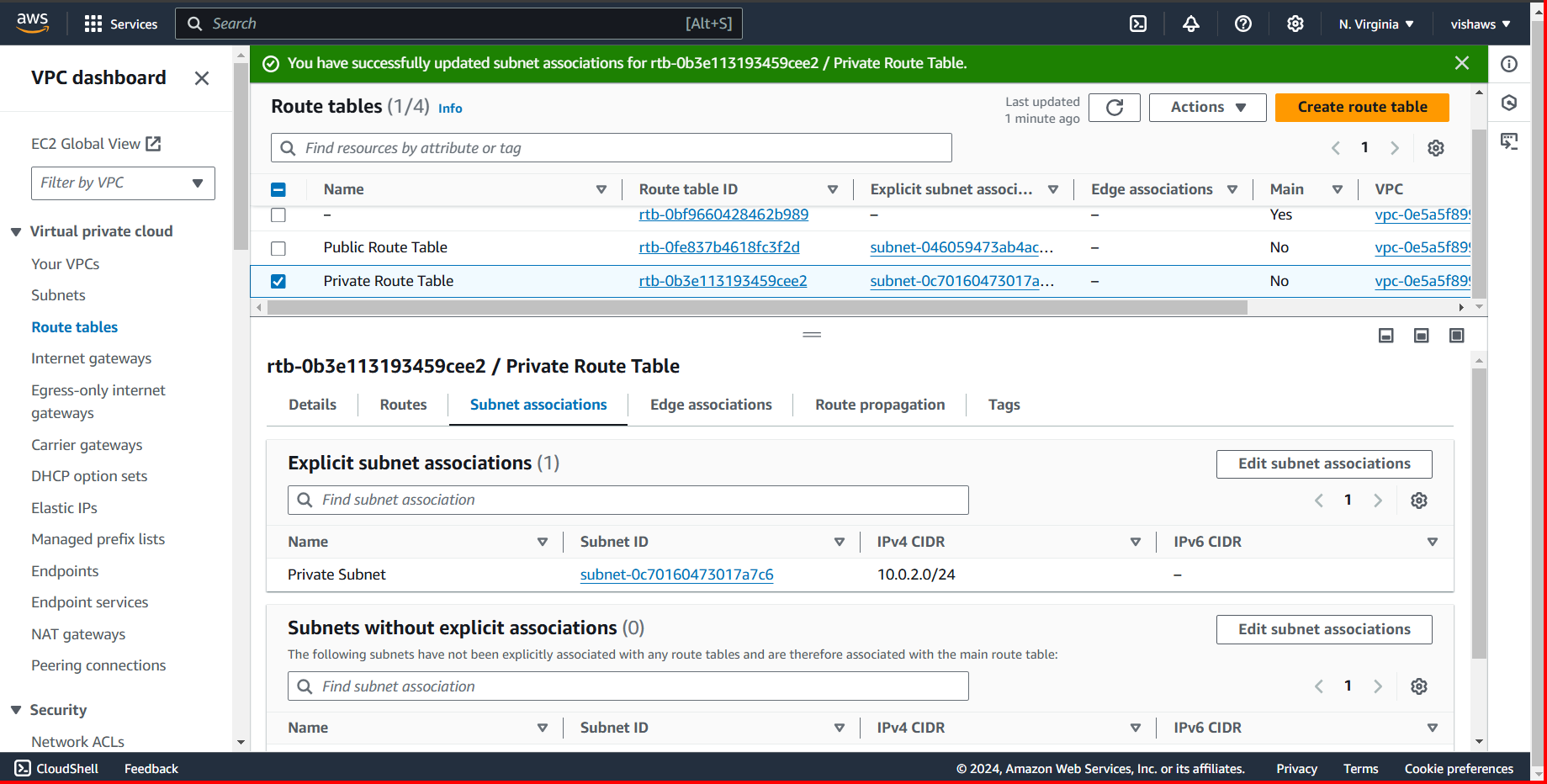


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | d |  |  |  |
| Step name | Creation of private route tables |  |  |  |
| Instructions | 1) Navigate to VPC -> Route Tables and click on Create Route table  2) Enter the name tag "Private Route Table", select the Project 1 VPC from the dropdown and click on Create  3) Once the route table is created, select it and select the Routes tab below the list of route tables  4) Click in Edit Routes and add the following route (Don't edit the existing one)  - Destination : 0.0.0.0/0  - Target: Select NAT Gateway and select the NAT Gateway created in the previous step  Click on Save Routes  5) Select the Subnet Associations tab and click on Edit Subnet Associations  6) Select the private Subnet from the list and click on Save |  |  |  |
| Expected screenshots | 1. Route list of the route table 2. Subnet association of the route table |  |  |  |

**<Insert Screenshot for d(1) here >**



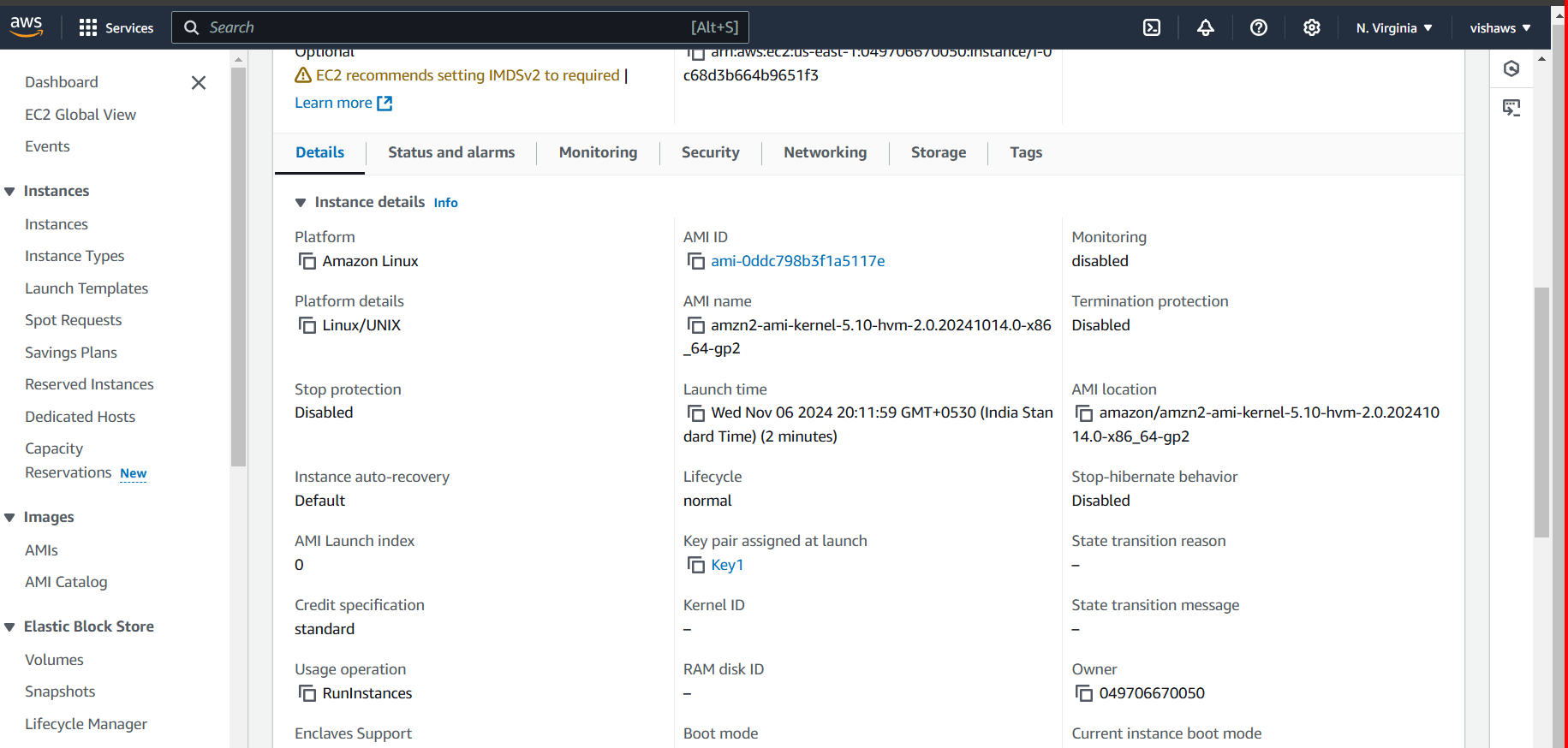
**<Insert Screenshot for d(2) here>**



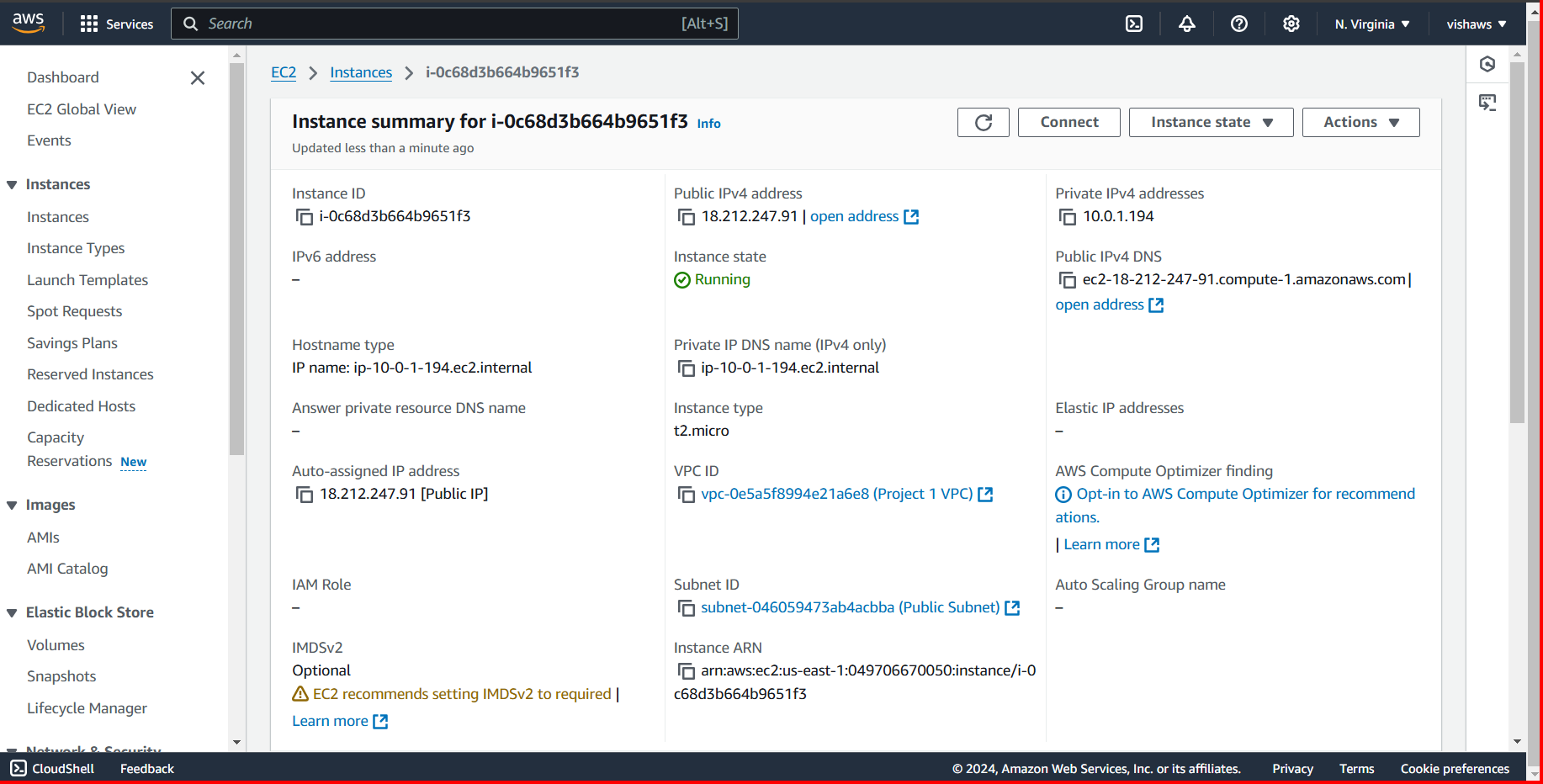
**Step 3 : Creation of database and application servers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | a |  |  |  |
| Step name | Creation of application server |  |  |  |
| Instructions | 1) Navigate to EC2 using the Services button at the top of the screen  2) Select Instances at the left side of the screen  3) Click on Launch Instance  - Select the AMI Amazon 2 Linux  - Select the instance type t2.micro  - Select Network as "Project 1 VPC" and subnet as "Public Subnet"  - For the security group, open the ports 80,443, 22 and 8065 for source set to "Anywhere"  4) Launch the instance after creating a new pem file and downloading it | | | |
| Expected screenshots | 1. AMI used 2. Instance configuration screen 3. Security group rules 4. Instance after creation |  |  |  |

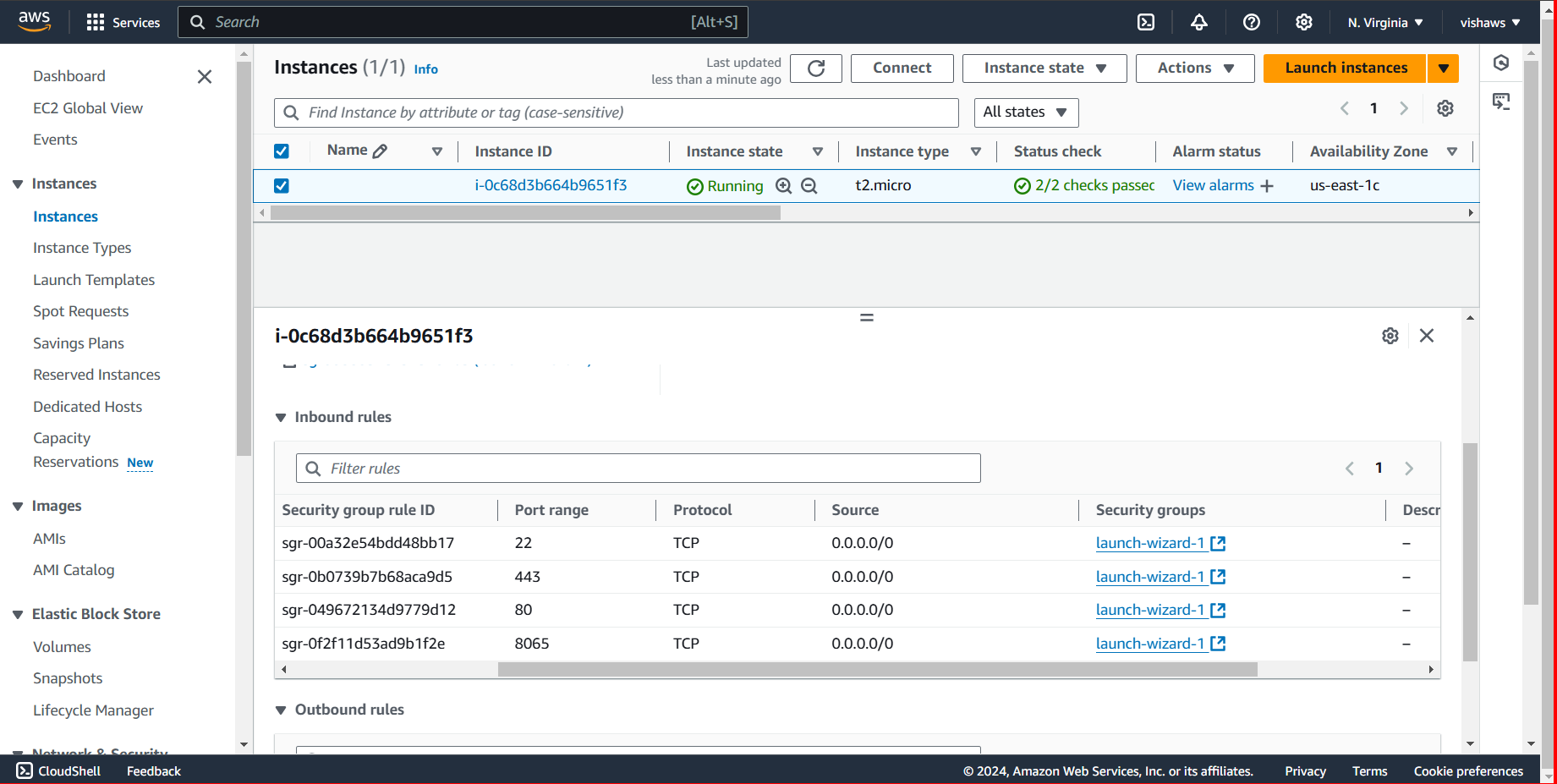
**<Insert screenshot a(1) here>**



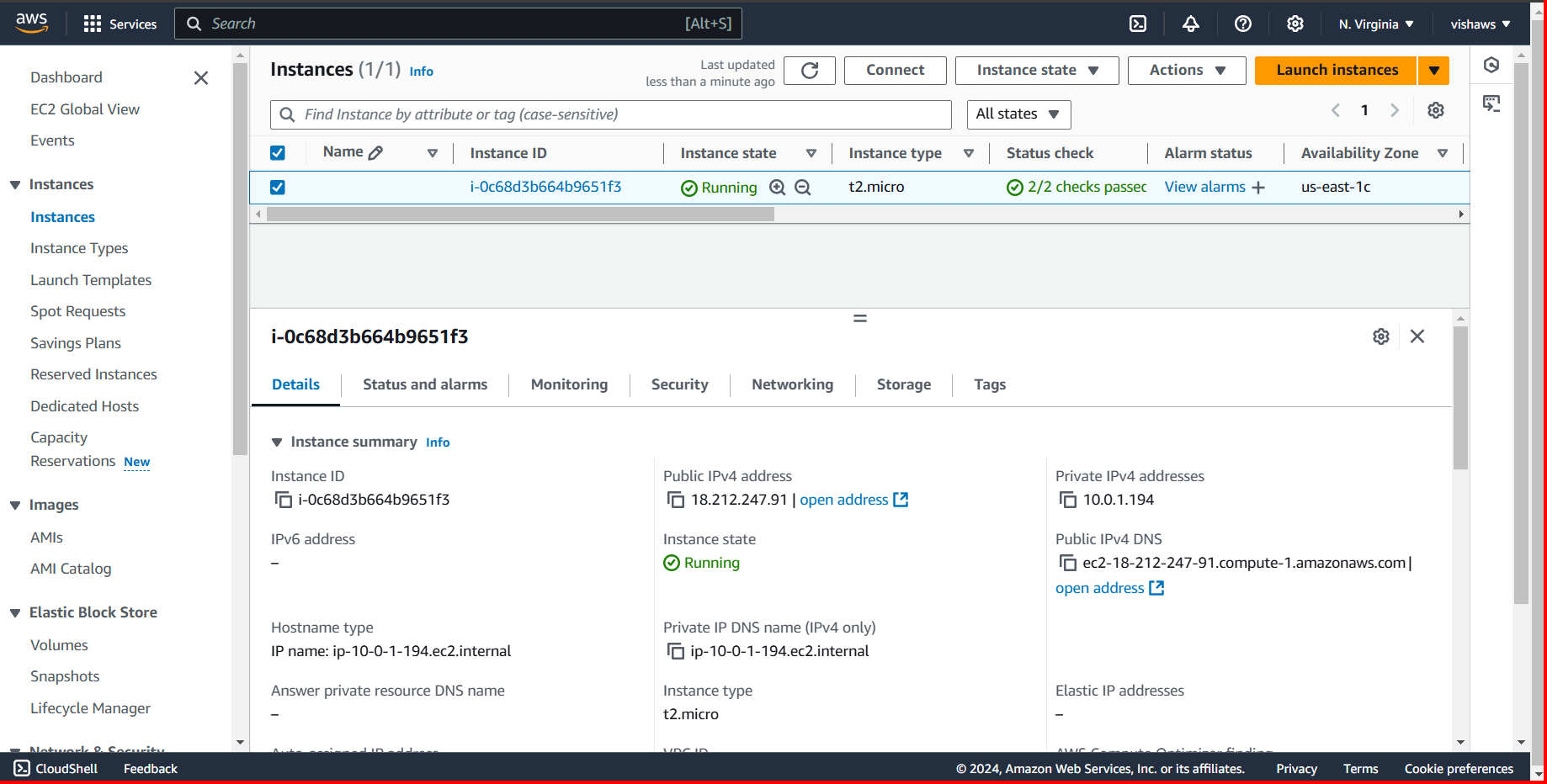
**<Insert screenshot a(2) here>**



**<Insert screenshot a(3) here>**

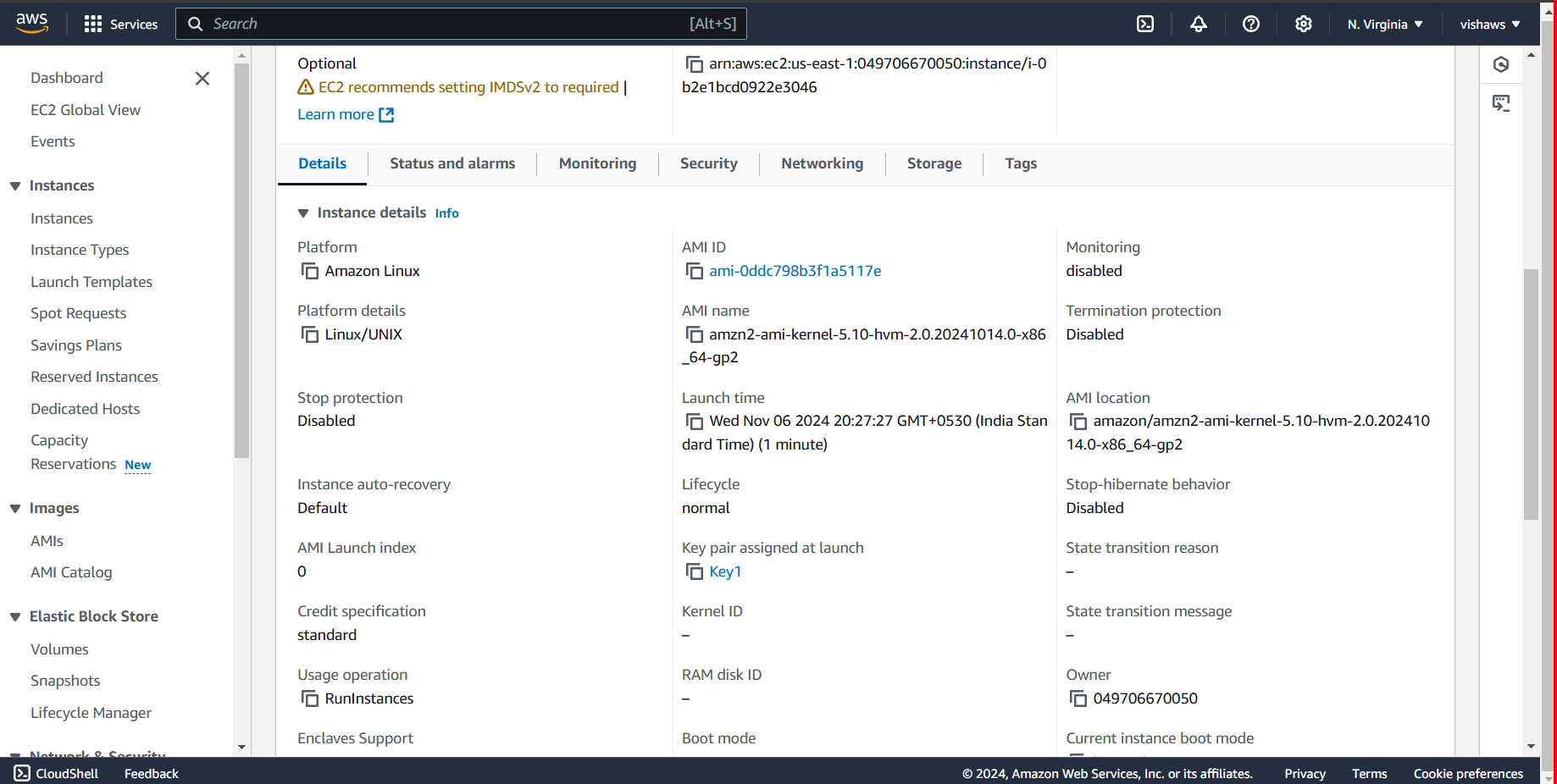


**<Insert screenshot a(4) here>**

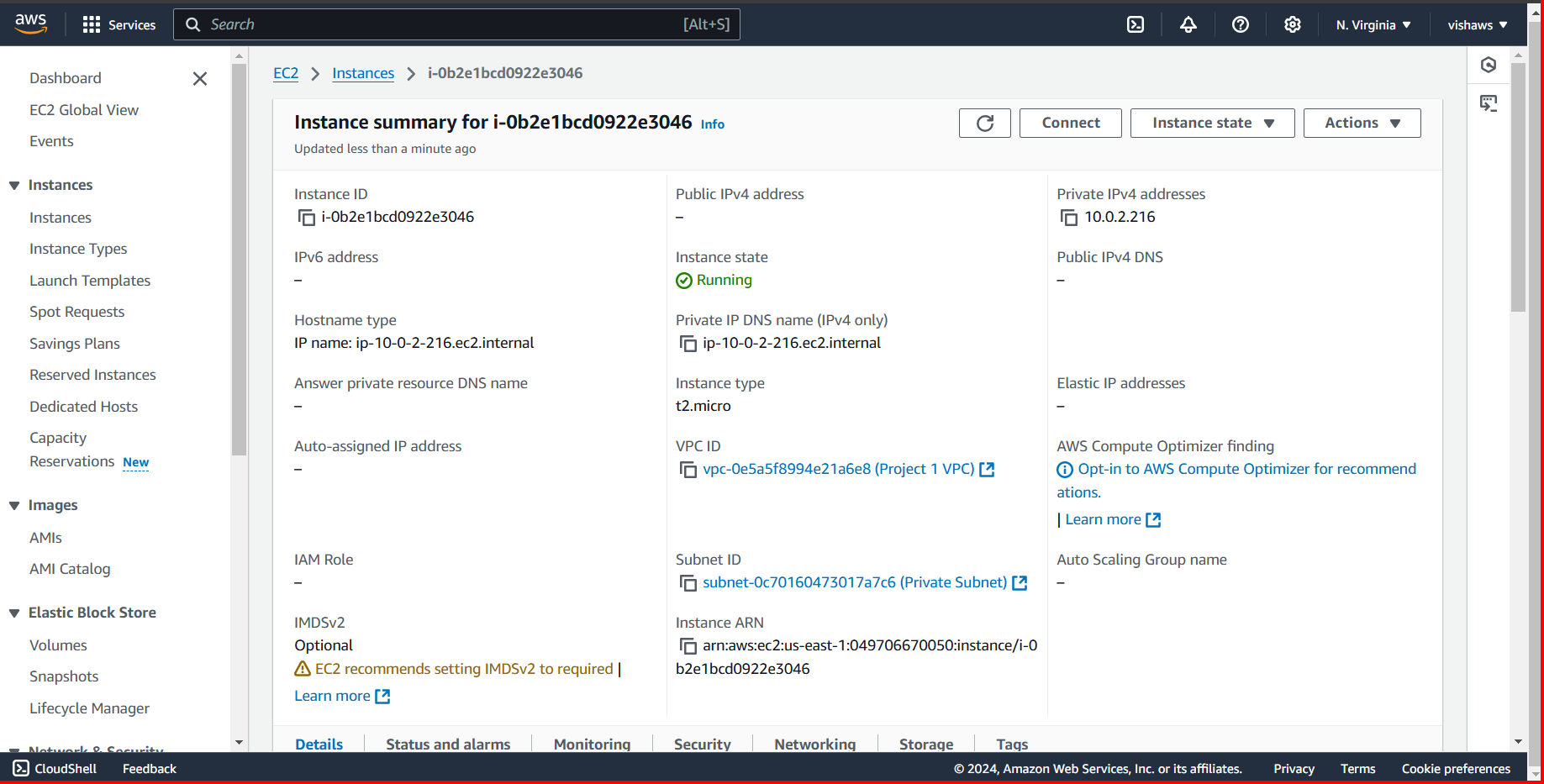


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | b |  |  |  |
| Step name | Creation of database server |  |  |  |
| Instructions | 1) Navigate to EC2 using the Services button at the top of the screen  2) Select Instances at the left side of the screen  3) Click on Launch Instance  - Select the AMI Amazon 2 Linux  - Select the instance type t2.micro  - Select Network as "Project 1 VPC" and subnet as "Private Subnet"  - For the security group, open the ports 80, 443,22 and 3306 for source set to "Anywhere"  4) Launch the instance by selecting the same pem file created in the previous step | | | |
| Expected screenshots | 1. AMI used 2. Instance configuration screen 3. Security group rules 4. Instance after creation |  |  |  |

**<Insert screenshot b(1) here>**



**<Insert screenshot b(2) here>**

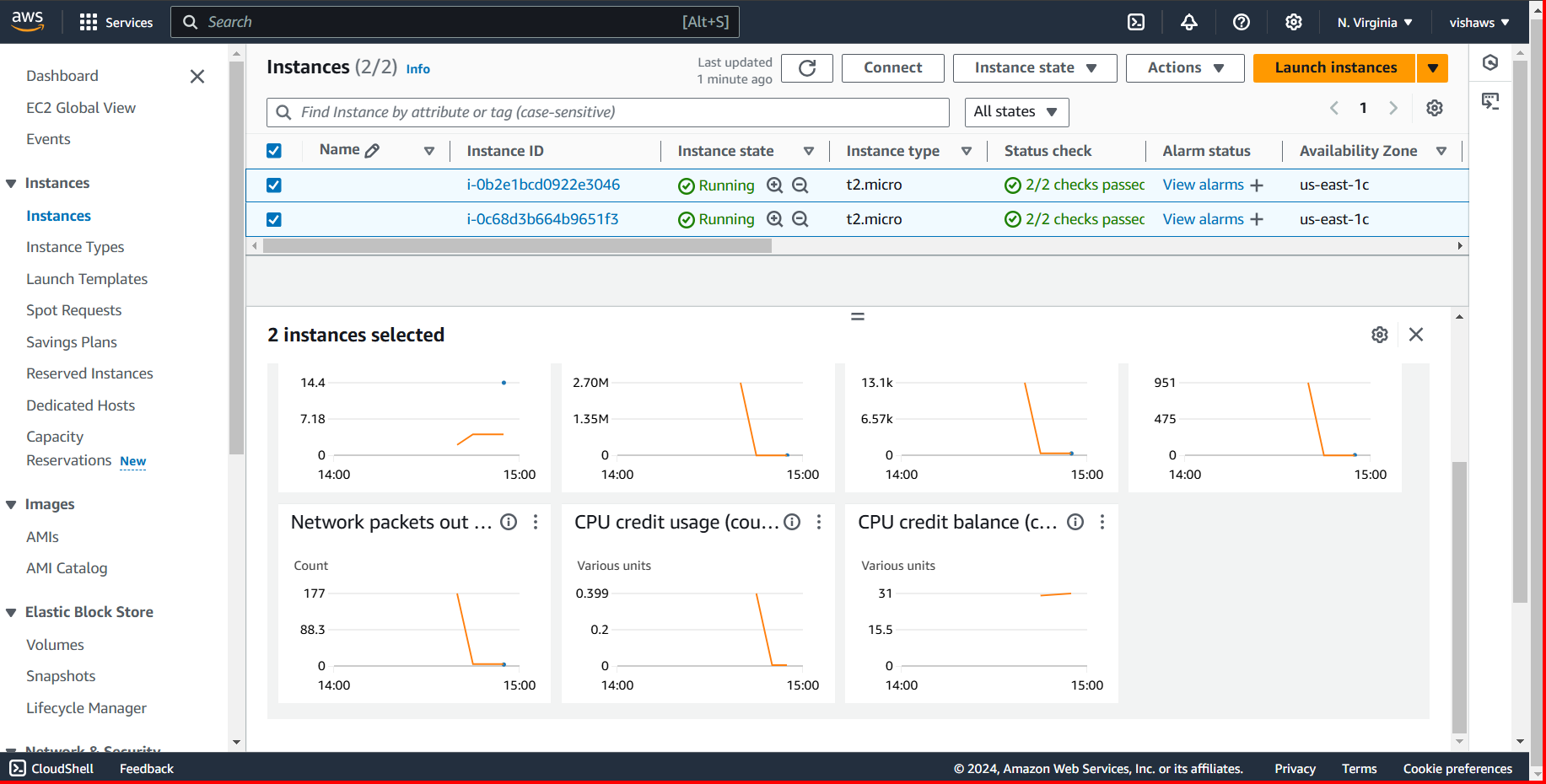


**<Insert screenshot b(3) here>**

A screenshot of a computer

Description automatically generated

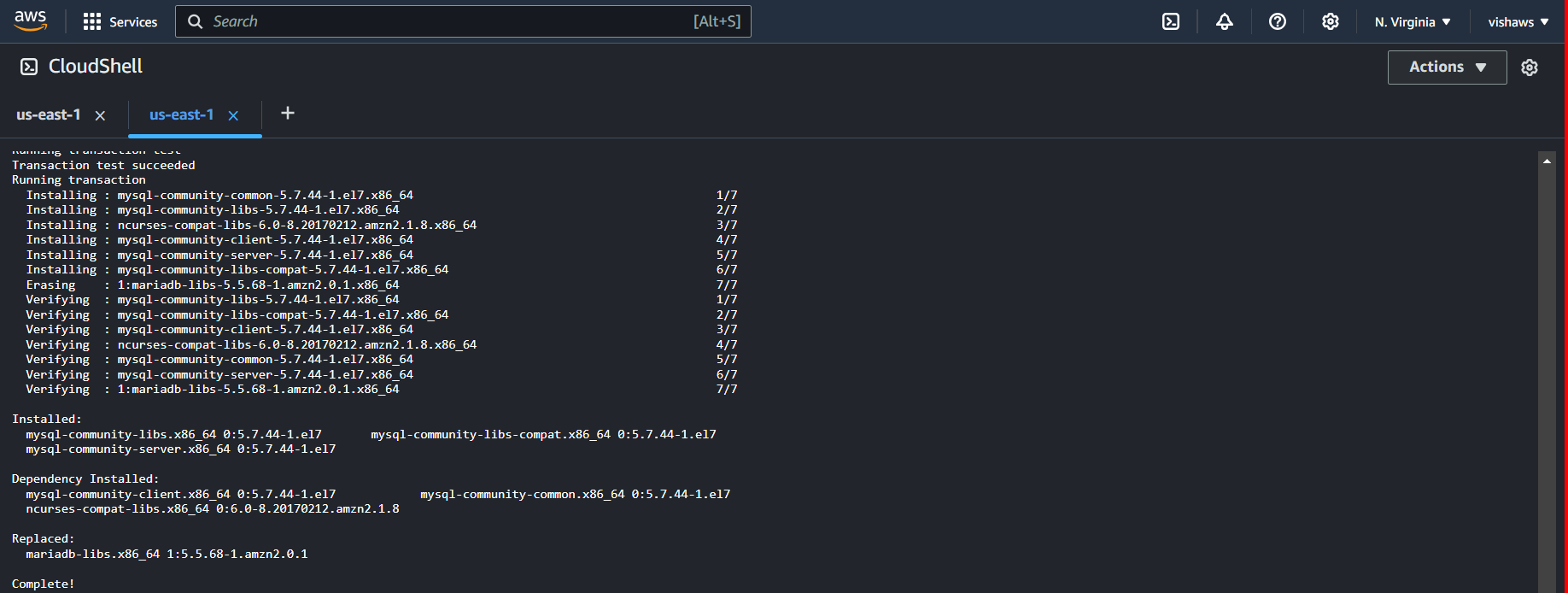
**<Insert screenshot b(4) here>**



**Step 4: Application and Database Installation and Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | a |  |  |  |
| Step name | Installation and configuration of MySQL |  |  |  |
| Instructions | 1) Copy the database pem file into the application server using the below command  *scp -i <application server pem file> <database server pem file > ec2-user@<application server public IP>:/home/ec2-user scp -i "Key1.pem" "Key1.pem" ec2-user@ec2-18-212-247-91.compute-1.amazonaws.com:/home/ec2-user*  2) Log into the application server using SSH/Putty  3) From the application server, log into the database server using the pem file copied in step 1and the private IP address of the database server with the following command  *ssh -i <database server pem file> ec2-user@<private IP of database server>*  4) Enter the following commands to install and configure MySQL on the database server  *sudo yum update* *wget http://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm sudo yum localinstall mysql57-community-release-el7-9.noarch.rpm -y*  *sudo yum install mysql-community-server -y --nogpgcheck*  *sudo systemctl start mysqld.service*  Run the below command to retrieve a temporary password for MySQL *sudo grep 'temporary password' /var/log/mysqld.log | rev | cut -d" " -f1 | rev | tr -d "."*  Log in to MySQL with the below command and enter the above password when prompted *mysql -u root -p*  Enter the below command after you login to MySQL  *ALTER USER 'root'@'localhost' IDENTIFIED BY 'Password42!';*  Type ‘exit’ into the MySQL prompt and press Enter to exit out of the MySQL environment. Enter the below commands to complete the setup. Ignore any warning messages you receive.  *wget https://d6opu47qoi4ee.cloudfront.net/install\_mysql\_linux.sh*  *chmod 777 install\_mysql\_linux.sh*  *sudo ./install\_mysql\_linux.sh*  5) Type *exit* to exit the database server and go back to the application server | | | |
| Expected screenshots | 1. Installation of MySQL 2. Retrieving the temporary password 3. Executing the provided script |  |  |  |

**<Insert screenshot a(1) here>**



**<Insert screenshot a(2) here>**

A screenshot of a computer

Description automatically generated

**<Insert screenshot a(3) here>**

A screenshot of a computer program

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | b |  |  |  |
| Step name | Installation and configuration of Mattermost |  |  |  |
| Instructions | 1) Enter the following commands after logging into the application server via SSH to install and configure Mattermost  *wget https://d6opu47qoi4ee.cloudfront.net/install\_mattermost\_linux.sh*  *sudo yum install dos2unix -y*  *sudo dos2unix install\_mattermost\_linux.sh*  *chmod 700 install\_mattermost\_linux.sh*  *sudo ./install\_mattermost\_linux.sh <private IP of MySQL server>*  Example : sudo ./*install\_mattermost\_linux* 173.65.34.7  *sudo chown -R mattermost:mattermost /opt/mattermost*  *sudo chmod -R g+w /opt/mattermost*  *cd /opt/mattermost*  *sudo -u mattermost ./bin/mattermost*  2) Check whether the server has been successfully deployed by navigating to the following URL in your web browser. The web page might take a couple of minutes to load.  <public IP of the application server>:8065 | | | |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |  |  |  |
| Expected screenshots | 1. Executing the script 2. Starting the Mattermost server 3. Accessing the application via web browser |  |  |  |

**<Insert screenshot b(1) here>**

A screen shot of a computer

Description automatically generated

**<Insert screenshot b(2) here>**

A screenshot of a computer

Description automatically generated

**<Insert screenshot b(3) here>**

A screenshot of a computer

Description automatically generated

**Step 5: Answer the following questions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Answer the following questions** | | | |  |
| Q1 | What is the default setting for DNS hostnames when a new VPC is created? | | |  |
|  | a) Enabled |  |  |  |
|  | b) Disabled |  |  |  |
|  | c) Can be set during VPC creation |  |  |  |
|  | d) Depends on the region used |  |  |  |
|  | Enter your answer here | a) Enabled |  |  |
|  |  |  |  |  |
| Q2 | What is the term used for the machine when we use it to log into the database server? | | |  |
|  | a) Bastion Host |  |  |  |
|  | b) NAT Gateway |  |  |  |
|  | c) Tunnel Interface |  |  |  |
|  | d) SSH Gateway |  |  |  |
|  | Enter your answer here | a) Bastion Host |  |  |
|  |  |  |  |  |
| Q3 | The database server security group in this exercise has to keep port 3306 open. Which protocol uses this port to communicate? | | |  |
|  | a) HTTPS |  |  |  |
|  | b) RDP |  |  |  |
|  | c) TCP |  |  |  |
|  | d) SCP |  |  |  |
|  | Enter your answer here | c) TCP |  |  |
|  |  |  |  |  |
| Q4 | Which port is being used by Mattermost to communicate with the client application | | |  |
|  | a) 8080 |  |  |  |
|  | b) 80 |  |  |  |
|  | c) 443 |  |  |  |
|  | d) 8065 |  |  |  |
|  | Enter your answer here | d) 8065 |  |  |
|  |  |  |  |  |
| Q5 | Which of the following is a reason why we cannot set the CIDR block for the public subnet to 10.0.2.0/16, assuming the values for the other CIDR blocks are the same as mentioned in the instructions? | | |  |
|  | a) CIDR block overlaps with existing block |  |  |  |
|  | b) CIDR block is not a valid CIDR |  |  |  |
|  | c) CIDR block does not fall within the VPC |  |  |  |
|  | d) There is no reason, this is a perfectly valid CIDR |  |  |  |
|  | Enter your answer here | a) CIDR block overlaps with existing block |  |  |
|  |  |  |  |  |
| Q6 | Assume that you have been asked to create 3 EC2 instances - application server, the database server and NAT instance. Each of these instances have their own security groups with a set of ports to be kept open. One of those ports is entirely unnecessary for the given architecture to function. Which of the ports given in the option below could it be? | | |  |
|  | a) Port 22 on the NAT instances |  |  |  |
|  | b) Port 3306 on the database server |  |  |  |
|  | c) Port 443 on the NAT instance |  |  |  |
|  | d) Port 22 on the application server |  |  |  |
|  | Enter your answer here | c) Port 443 on the NAT instance |  |  |
|  |  |  |  |  |
| Q7 | Describe the steps you would take to increase security of the servers you have deployed so that they are not reachable from external sources | | |  |
|  |  | | |  |
|  | **1. Implement Network Security (VPC and Security Groups)**   * **Isolate servers in private subnets**: Ensure that your servers (such as **application** and **database servers**) are placed in **private subnets** within your **VPC (Virtual Private Cloud)**. This prevents them from being directly accessible from the **internet**. * **Use VPC Security Groups**: Configure **Security Groups** to restrict inbound and outbound traffic for each server:   + Only allow **necessary traffic** (e.g., port 22 for SSH on the application server, port 3306 for MySQL on the database server).   + Deny inbound traffic from any external IPs unless it's specifically required for administration or application access.   + For example, allow SSH (port 22) on the application server only from specific IP addresses or CIDR blocks (e.g., your office IP range or a jump box).   **2. Use a Bastion Host (Jump Server)**   * **Set up a Bastion Host** in a **public subnet** to provide secure access to the private servers. The bastion host is the only server exposed to the internet, and you can access your private servers through it.   + Only allow SSH or RDP (depending on your OS) access to the bastion host.   + Use **multi-factor authentication (MFA)** for SSH access to the bastion host.   + From the bastion host, you can SSH into the application and database servers. * Alternatively, use **AWS Systems Manager Session Manager** for secure access without needing an open SSH port.   **3. Implement Network ACLs (Optional but Recommended)**   * **Network Access Control Lists (ACLs)** can be used in addition to security groups to add a layer of security for controlling traffic at the subnet level. * Define **stateless rules** in your NACLs to block inbound and outbound traffic to the servers from external sources unless explicitly required.   **4. Close Unused Ports**   * **Audit open ports** on each server and close any **unused ports** in both your **security groups** and **NACLs**. For example, if you're not running a web service, close **port 80** and **port 443**. * **Limit SSH access (port 22)**: Restrict SSH access to specific IP addresses or ranges. Avoid allowing SSH access from 0.0.0.0/0 (anywhere). * **Use non-standard ports**: If applicable, consider using **non-standard ports** for SSH or other services to reduce exposure to automated attacks.   **5. Enable Host-Based Firewalls**   * Use host-based firewalls (e.g., **iptables** for Linux, **Windows Firewall** for Windows) on each server to restrict traffic that isn’t necessary for the operation of the server. * For example, on a Linux server, configure **iptables** to only allow traffic from specific IP addresses or subnets.   **6. Secure SSH and Remote Access**   * **Disable password authentication for SSH** and use **SSH keys** for access to servers. This helps prevent brute force attacks. * If you're using **RDP** (for Windows servers), ensure that **Network Level Authentication (NLA)** is enabled, and **RDP access** is restricted to trusted IP addresses. * **Use MFA** for SSH or RDP access, especially for privileged accounts. * Disable or lock accounts that are not in use and use **least privilege** for access control.   **7. Implement Proper Identity and Access Management (IAM)**   * **Use IAM roles and policies**: For AWS, ensure you are using **IAM roles** and **policies** to restrict access to resources on the server. For example, if an EC2 instance doesn't need S3 access, ensure it doesn't have an IAM role that provides that access. * **Least Privilege**: Apply the principle of **least privilege** to your IAM users and roles to limit access to only what is necessary. * Regularly **review and rotate credentials** for admin accounts.   **8. Enable Logging and Monitoring**   * **Enable logging** for all access attempts, especially SSH and RDP access. For Linux servers, ensure that **/var/log/auth.log** (for SSH logs) is enabled, and for Windows servers, ensure **Windows Event Logs** are enabled. * Use tools like **AWS CloudTrail** or **AWS Config** to monitor and log all API activity and configuration changes. * Implement **intrusion detection systems (IDS)** or **intrusion prevention systems (IPS)** such as **Snort**, **OSSEC**, or **AWS GuardDuty** to detect abnormal or unauthorized activity.   **9. Encrypt Data at Rest and in Transit**   * **Encrypt sensitive data** stored on your servers using technologies like **EBS encryption** for AWS or **Disk encryption** for on-premise servers. * Use **TLS/SSL** to encrypt communication between servers, especially for sensitive data exchanges such as with your database or application layer. * Enable **Encryption in Transit** for internal communications between servers to prevent eavesdropping or tampering.   **10. Patch and Update Servers Regularly**   * **Regularly patch** your servers to ensure they are up-to-date with the latest security fixes. Implement a **patch management policy** to ensure this process is automated or scheduled. * Subscribe to security advisories related to the OS and applications you are using to stay informed about potential vulnerabilities.   **11. Use VPN or Direct Connect (for Internal Communication)**   * If your servers need to communicate over the internet, ensure this communication is encrypted using a **VPN** or **Direct Connect** to establish a private, secure link between your internal infrastructure and cloud resources. * If your infrastructure involves **hybrid environments**, ensure secure communication using **VPN** or private links rather than relying on open internet channels.   **12. Automate Security Compliance with Tools**   * Use tools like **AWS Config**, **AWS Security Hub**, or **Terraform** to enforce security standards and ensure compliance with industry regulations like **PCI-DSS**, **HIPAA**, or **GDPR**. * Consider using **security scanning tools** such as **Amazon Inspector** to identify vulnerabilities in your EC2 instances.   **Summary of Steps:**   1. **Place servers in private subnets** with appropriate **security groups** and **NACLs**. 2. **Use a Bastion Host** for secure administrative access. 3. **Disable unnecessary ports** and restrict access to **required services** only. 4. **Implement SSH key-based authentication** and disable password access. 5. Use **host-based firewalls** for additional access control. 6. Use **IAM roles** with the principle of **least privilege**. 7. Enable **logging and monitoring** to detect unauthorized access. 8. Regularly **patch** and **update** servers. 9. **Encrypt sensitive data** both in transit and at rest. 10. Use a **VPN** or **Direct Connect** for secure internal communication. 11. **Automate security compliance** and vulnerability scanning.   By following these steps, you can significantly reduce the risk of unauthorized access and ensure your servers are protected from external threats. | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Q8 | Describe the steps required to deploy the given application in an autoscaling environment | | |  |
|  |  |  |  |  |
|  | **1. Prepare the Application for Scaling**   * Ensure the application is **stateless**:   + Stateless means the application does not rely on a single instance for its operation. Each instance must be able to handle any request independently. Use an **external database** or a **distributed cache** (e.g., **Redis**, **ElastiCache**) to store session data or state information. * **Containerization** (Optional):   + If the application is containerized (e.g., using **Docker**), ensure the containers are designed to work in an **elastic environment** and can be replicated as needed.   **2. Set Up the Infrastructure (Virtual Machines, Containers, etc.)**   * **Create an Image for EC2 Instances (or VMs)**:   + In AWS, for example, use **Amazon Machine Images (AMIs)**. Ensure the AMI contains all necessary configurations (software, libraries, environment settings) required to run the application. * If using **containerized applications**, ensure that the Docker images or Kubernetes pods are pre-configured and ready for scaling.   **3. Set Up a Load Balancer**   * **Configure a Load Balancer** to distribute traffic across the instances:   + **AWS**: Use **Elastic Load Balancer (ELB)**, such as **Application Load Balancer (ALB)** for HTTP/HTTPS traffic or **Network Load Balancer (NLB)** for TCP traffic.   + **Azure**: Use **Azure Load Balancer** or **Azure Application Gateway**.   + **Google Cloud**: Use **Google Cloud Load Balancer**. * The load balancer should be **connected to your auto-scaling group** to ensure traffic is evenly distributed to the healthy instances in the pool. * **Health Checks**: Set up **health checks** on the load balancer to ensure that only healthy instances receive traffic. The health check will monitor the application’s availability (usually by checking an endpoint like /health).   **4. Set Up Auto-Scaling Group**   * **Create an Auto-Scaling Group** (ASG) to automatically manage the scaling of the application instances.   + In **AWS**, you create an **Auto Scaling Group** (ASG) by defining a launch template or launch configuration.   + Define the **minimum**, **maximum**, and **desired instance count**. For example, you may start with 2 instances and scale up to 10 instances based on demand. * **Scaling Policies**:   + Define **scaling policies** to trigger scaling actions. These policies typically monitor **metrics** like **CPU utilization**, **memory usage**, or **network traffic**.     - Example: Scale up when average CPU utilization exceeds **70%** and scale down when CPU utilization is below **30%**. * **Instance Types**:   + Choose the appropriate **instance types** (e.g., t3.medium, m5.large, c5.xlarge) based on the application’s resource requirements.   **5. Configure Auto-Scaling Triggers (Scaling Policies)**   * **Metrics and CloudWatch Alarms**:   + In AWS, configure **CloudWatch alarms** for the selected metrics (e.g., CPU utilization, network traffic, request count, etc.).   + Create a scaling policy based on these alarms to either scale in (remove instances) or scale out (add instances).   + Example: **Scale Out** if CPU usage exceeds 80% for 5 minutes, and **Scale In** if CPU usage is below 40% for 5 minutes. * **Scheduled Scaling**:   + In addition to dynamic scaling based on demand, you can set up **scheduled scaling** if you know traffic will increase or decrease at specific times of the day or week.   **6. Set Up Storage (Database and Persistent Storage)**   * **Databases**: Use a **managed database service** like **Amazon RDS**, **Azure SQL Database**, or **Google Cloud SQL** for the database backend.   + Ensure that the database is **highly available** and **scalable** (e.g., using **Read Replicas**, **Multi-AZ** deployments).   + Configure **auto-scaling for the database** if needed (e.g., using **Amazon Aurora** with Aurora Replicas). * **Shared Storage**: If your application requires **shared storage** (e.g., file storage for media), use services like **Amazon S3** or **Azure Blob Storage**. * Ensure **data persistence** is not tied to any specific instance. For example, use **Elastic File System (EFS)** or **Amazon S3** for shared file storage.   **7. Configure Security Groups and Network Settings**   * **Security Groups**:   + Configure **Security Groups** for your instances and the load balancer. These security groups should allow necessary traffic (e.g., HTTP/HTTPS, SSH for management) while blocking unnecessary access.   + Ensure **SSH** (port 22) access is restricted to trusted IP addresses or through a **Bastion Host**. * **VPC Setup**: Ensure the instances are deployed in the appropriate **subnets** (public or private) within a **VPC**.   + Use **private subnets** for application instances and **public subnets** for load balancers if necessary.   + Use **Network ACLs** to control the traffic at the subnet level.   **8. Configure Monitoring and Logging**   * **Monitor your Auto-Scaling Environment**:   + Enable **CloudWatch** or a similar monitoring service to track key metrics like CPU utilization, memory usage, response times, etc.   + Use **AWS CloudWatch Logs** (or the equivalent in other cloud platforms) to collect and analyze logs from your application. * **Set up Alarms**: Create alarms for critical metrics like CPU utilization, disk space, or application errors that may indicate problems with the application.   **9. Test the Auto-Scaling Environment**   * **Test scaling functionality** by simulating traffic spikes to trigger scaling policies:   + For example, use **load testing** tools (e.g., **Apache JMeter**, **Gatling**) to simulate traffic and see if the scaling policies correctly scale out/in the number of instances.   + Ensure the application remains responsive and healthy as new instances are added or removed.   **10. Deploy Application Code**   * **Deploy the application** to the auto-scaling instances. If using a **CI/CD pipeline**, automate the deployment process to push updates to the instances in the auto-scaling group. * Ensure that any configuration or application data that needs to be shared between instances is stored externally (e.g., in an **S3 bucket**, **Redis cache**, or **RDS** database).   **11. Set Up Continuous Deployment (Optional)**   * Use **CI/CD tools** like **AWS CodePipeline**, **Jenkins**, **GitLab CI**, or **Azure DevOps** to automate code deployment into the auto-scaling environment. * Automate testing, staging, and production deployments to ensure smooth updates to your application.   **Summary of Steps:**   1. **Prepare the application** to be stateless and containerized (if applicable). 2. **Set up the infrastructure** for EC2 instances, containers, or Kubernetes clusters. 3. **Set up a Load Balancer** to distribute traffic and connect it to your auto-scaling group. 4. **Create an Auto-Scaling Group** with proper instance configurations and scaling policies. 5. **Set up scaling triggers** based on metrics (CPU, memory, etc.) using CloudWatch alarms. 6. **Configure storage** (database and file storage) to be scalable and persistent. 7. **Configure security** (Security Groups, Network ACLs, IAM roles). 8. **Monitor and log** performance and application behavior to ensure proper scaling. 9. **Test the auto-scaling environment** by simulating load to verify scaling policies work as expected. 10. **Deploy application code** to the auto-scaling environment, and automate deployment where possible. 11. **Set up continuous deployment** if required for ongoing updates.   By following these steps, you'll ensure that your application can dynamically scale based on traffic demand, while maintaining performance, availability, and security. | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |
| --- | --- |
| **Grades distribution** |  |
| MCQs | 6 (1 mark each) |
| Subjective questions | 20 marks (10+10) |
| Implementation screenshots | 24 marks (1 marks each) |
| Total | 50 marks |