**Private and Public Subnet**

In AWS, **public** and **private subnets** are ways to organize your cloud network (specifically inside a **VPC**, or Virtual Private Cloud) to control how resources can connect to the internet.

Subnets are categorized as either **public** or **private**, depending on their routing configuration.

**Private Subnet:**

* If a subnet is associated with Rout Table, which is routed to the Internet Gateway (IGW), then that subnet is called as Public subnet.
* The Instance which is launched in a public subnet can receive a public IPv4 address (either automatically assigned or through an Elastic IP address). This allows them to initiate outbound traffic to the internet and receive inbound traffic from the internet (if the security group rules allow it).
* Public subnets are typically used for resources that need to be directly accessible from the internet, such as:

1. Web servers
2. Load Balancers
3. Bastion hosts (for securely accessing instances in private subnets)

**Private Subnet:**

* A private subnet is associated with a route table that **does not** have a direct route to an Internet Gateway.
* The Instance which is launched in a private subnet only have private IPv4 addresses within the VPC's CIDR block. They cannot directly initiate or receive traffic from the internet.
* To allow instances in a private subnet to access the internet (e.g., for software updates), you typically need to use a **NAT Gateway** or a **NAT instance** deployed in a public subnet.
* Private subnets are ideal for deploying resources that should not be directly exposed to the public internet, such as:

1. Application servers
2. Database servers
3. Internal processing systems

**Difference between the private and Public Subnets:**

|  | **Public Subnet** | **Private Subnet** |
| --- | --- | --- |
| **Internet Access** | Direct access to/from the internet via an **Internet Gateway**. | No direct internet access (unless through a **NAT Gateway** or **NAT instance**). |
| **Typical Resources** | Web servers, load balancers, bastion hosts. | Databases, application servers, backend services. |
| **Routing Table** | Routes traffic destined for 0.0.0.0/0 (i.e., any address) through an Internet Gateway. | Routes only within the VPC or through a NAT device for outbound internet access. |
| **Security** | Generally more open; use **Security Groups** and **NACLs** carefully. | More restricted and protected from public exposure. |

**Block Diagram:**

Internet

NAT

IGW

VPC

RT-02

RT-01

Private Subnet

Public Subnet

Instance-01

EIP address

Subnets association to the Rout Tables

Instance-02

**NAT Gateway**

* A **NAT Gateway (Network Address Translation Gateway)** is a managed AWS service that allows private subnets (with no internet access) to securely connect to the internet or other AWS services **without exposing their private IP addresses**.
* That means the resources (Instances) which are present in the private subnet can access internet (through) by using the NAT gateway, for software updates, installing software’s, downloading API’s and so on.
* It only sends/allow the outbound traffic to the external but cannot receive/allow the inbound traffic from the internet. That means the instances which are present in private subnet can be connect to the internet, but we cannot access/connect these instances from the internet.
* The NAT Gateway replaces the private IP address of the instance with its own public IP address (specifically, the Elastic IP address you associate with it). It also tracks the source port and destination to ensure return traffic is routed correctly.

Now let’s work with the NAT gateway practically by considering above block diagram

**Step1:** Create a VPC

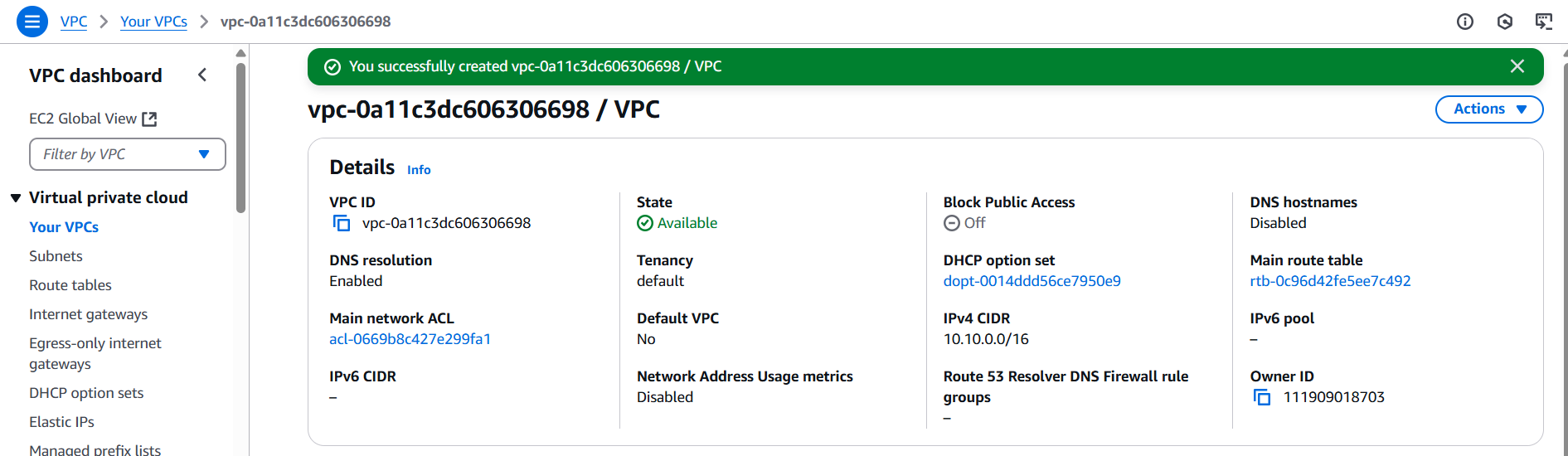


Fig: VPC is created successfully.

Step2: Create two subnets (Public-subnet & Private-subnet) within the VPC.

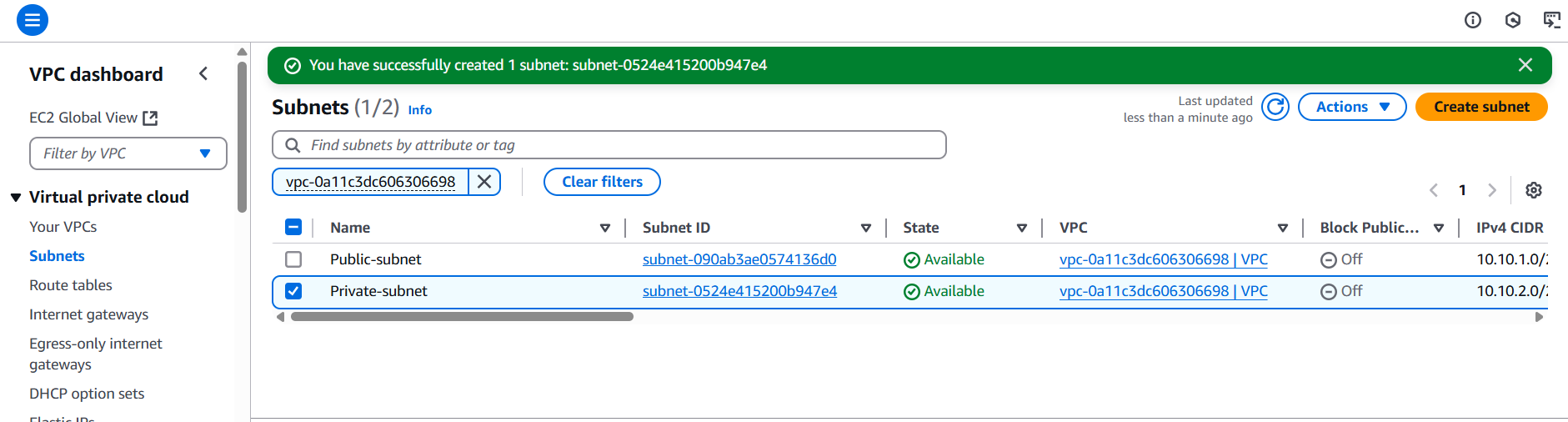


Fig: Two subnets (Public-subnet & Private-subnet) are created successfully.

Step3: Create an Internet gateway (IGW) and attach it to the VPC.

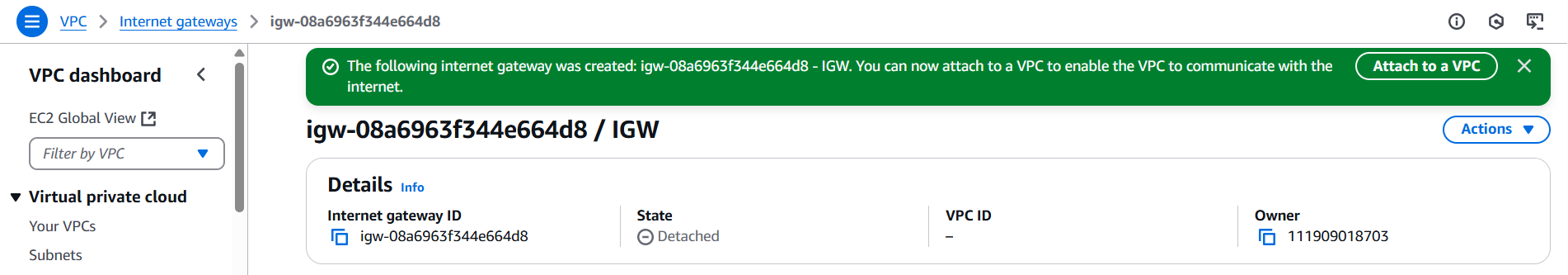


Fig: Internet gateway (IGW).

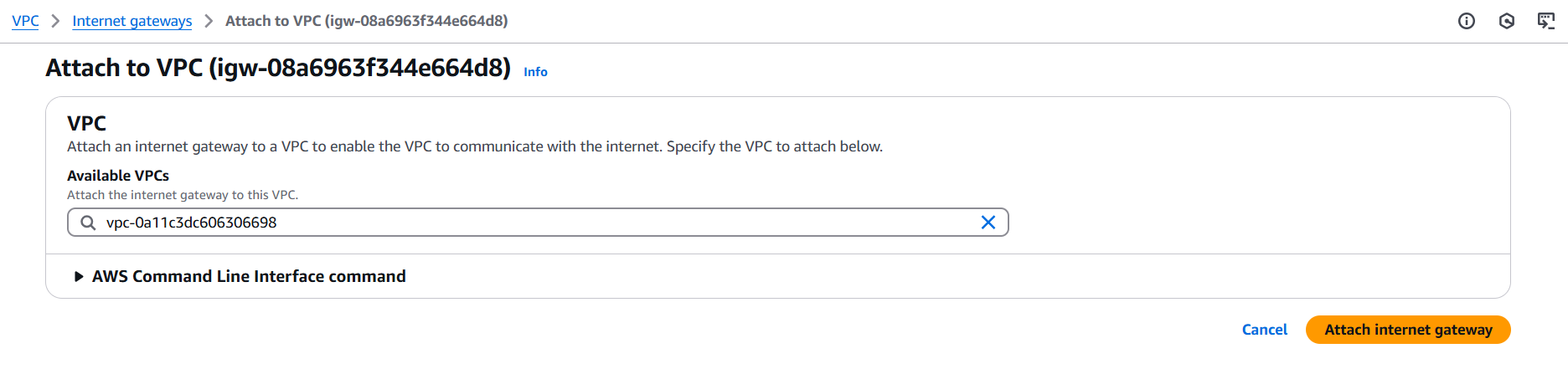


Fig: Attaching IGW to the VPC.

Step4: Create one Route Table (RT-01) and configure route to allow all traffic using internet gate way.

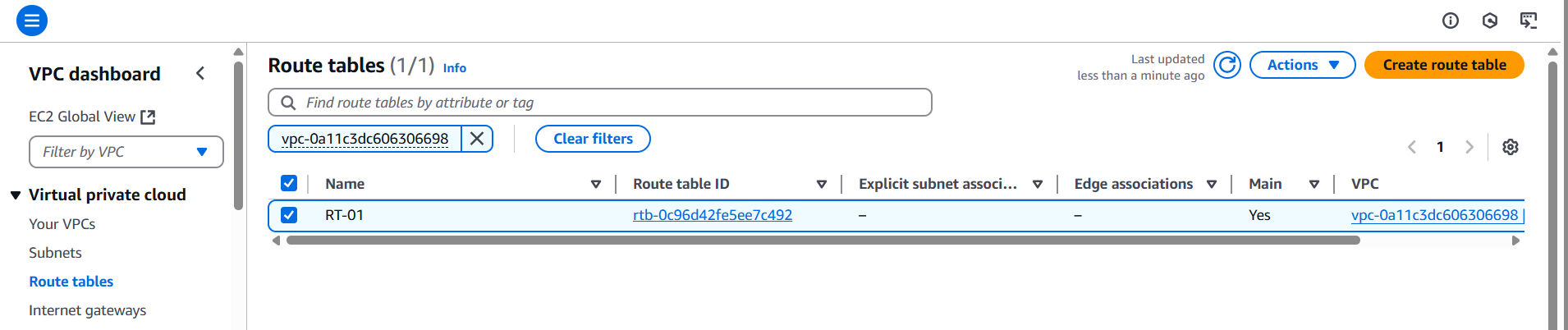


Fig: Route Table (RT-01).

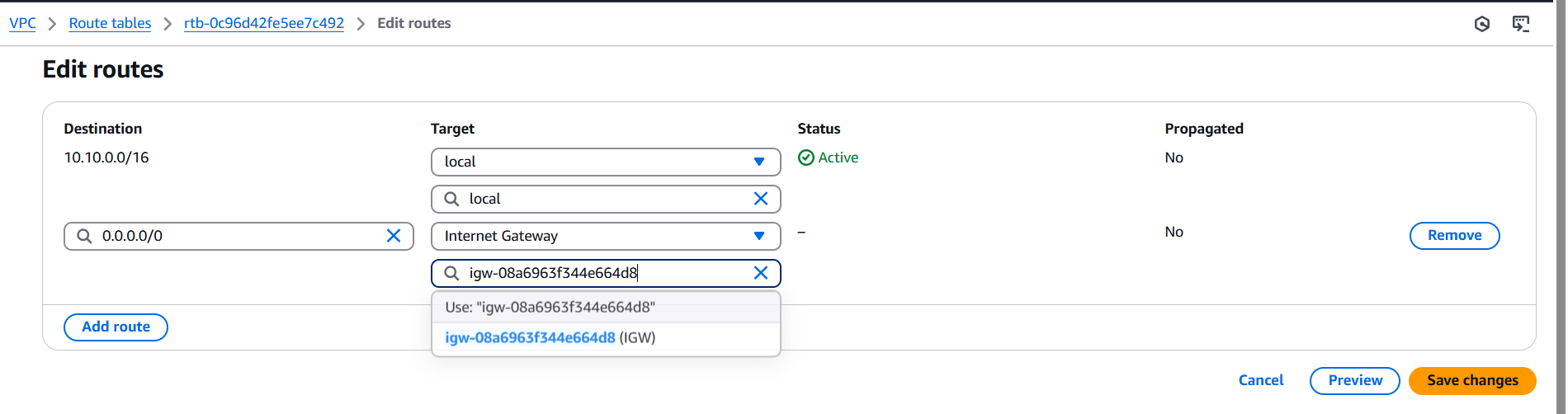


Fig: Configuring the route to allow all traffic (0.0.0.0/0) using IGW.

**Step5:** Create one more Rout Table (RT-02) inside the same VPC and do not configure route to the internet gateway (IGW). (Allow only internal local access only).

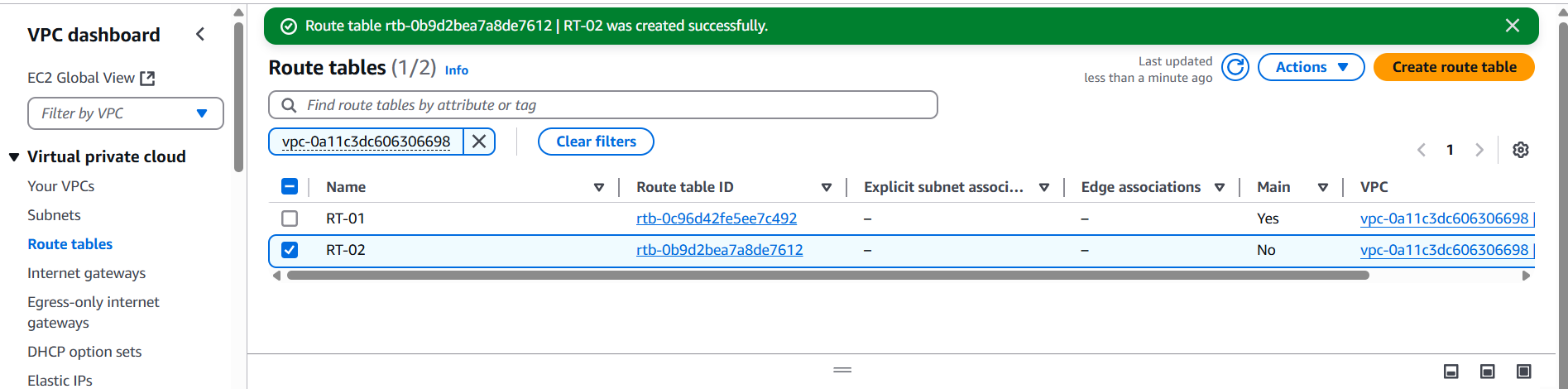


Fig: Rout Table (RT-02).

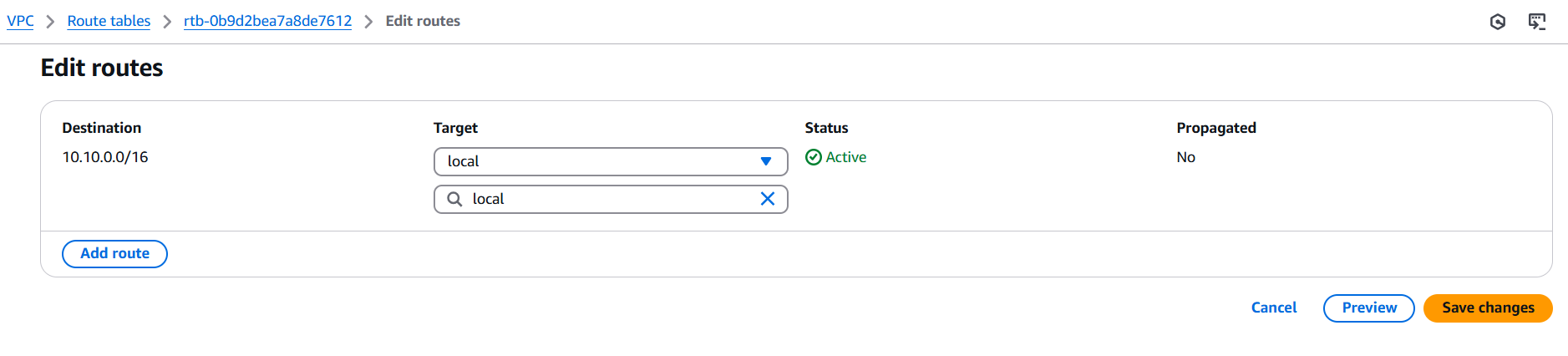


Fig: Configuring of RT-02 to allow only local access/connection.

Step6: Associate the Public-subnet to the Rout Table (RT-01), which is routed to the internet gateway (IGW).

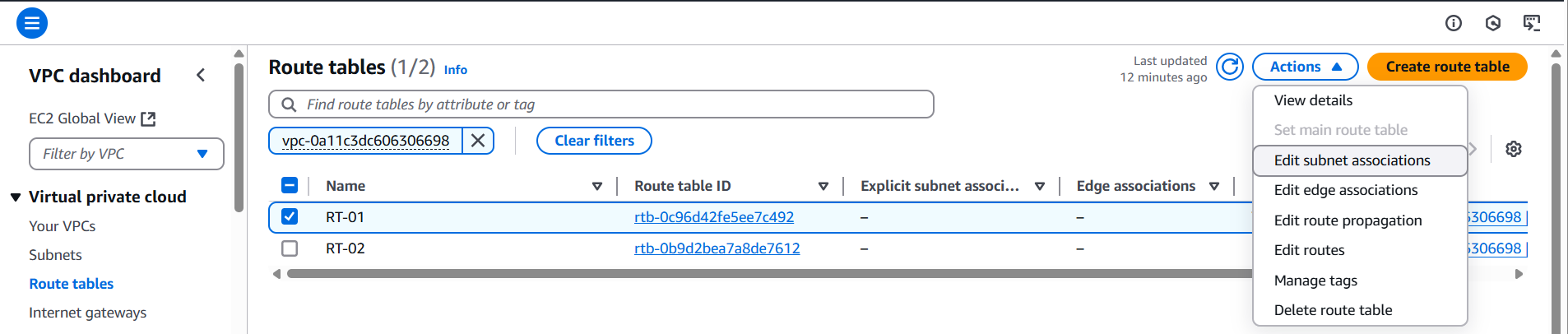


Fig: Association of public-subnet to RT-01.

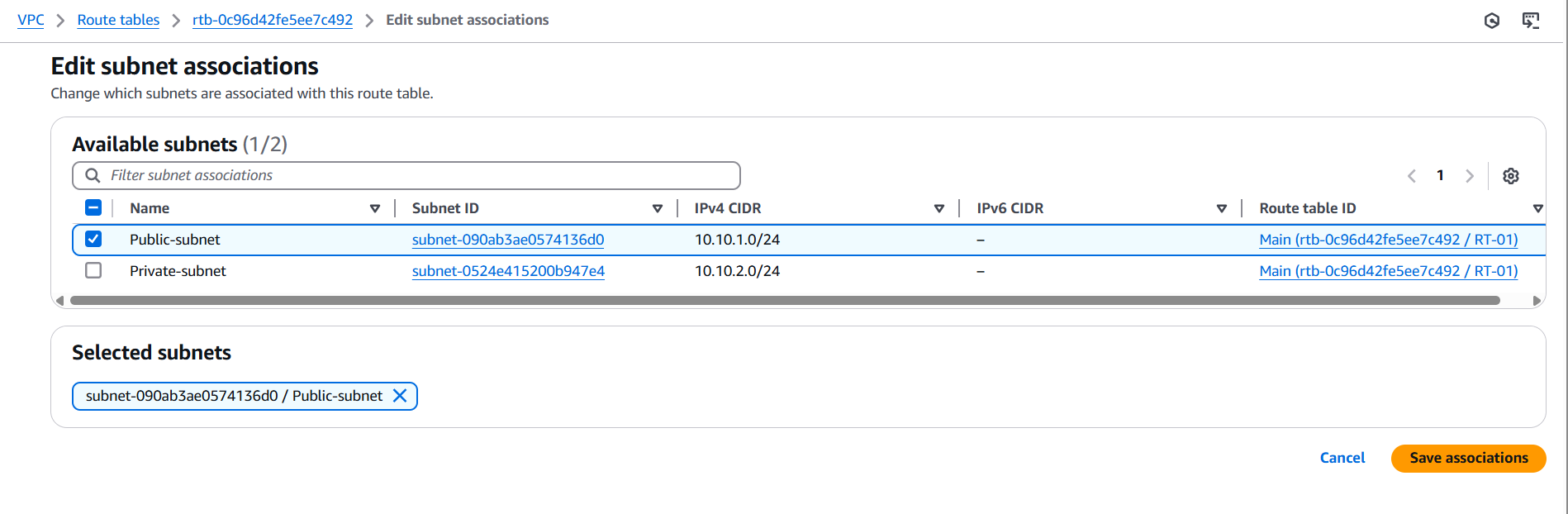


Fig: Save Association.

**Step7:** Similarly now associate the private-subnet to the RT-02, which not routed to the internet gateway.

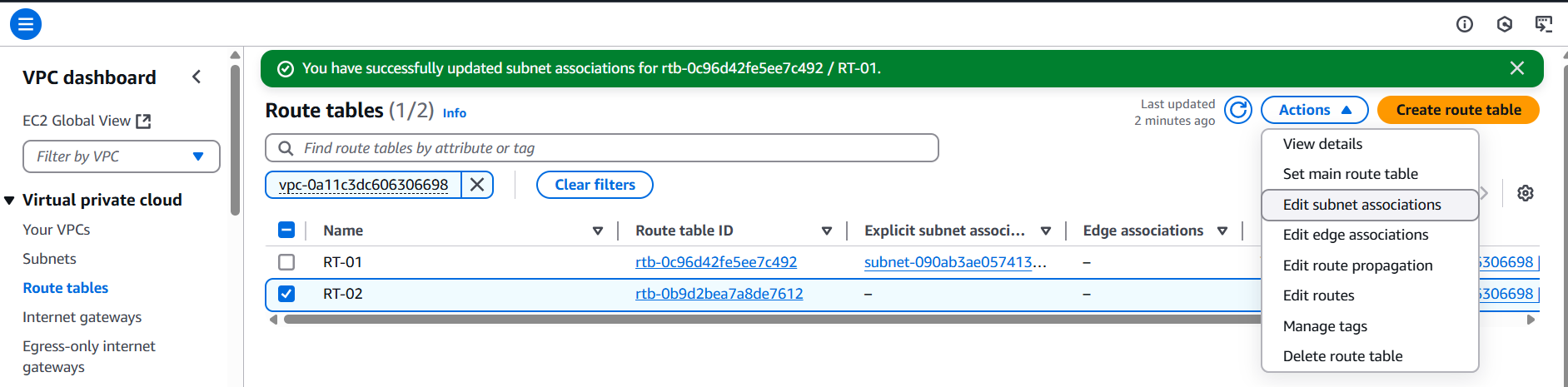


Fig: Association of private-subnet to RT-02.

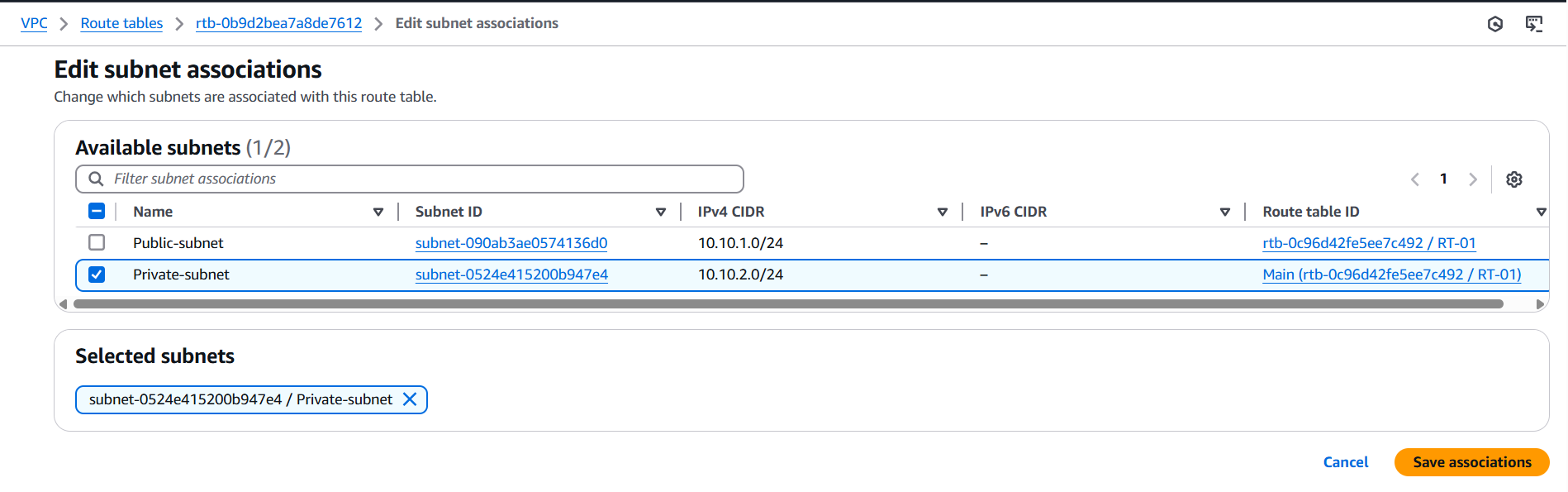


Fig: Save Association.

Step8: Create Security group and assign it to the VPC by allowing all traffic.

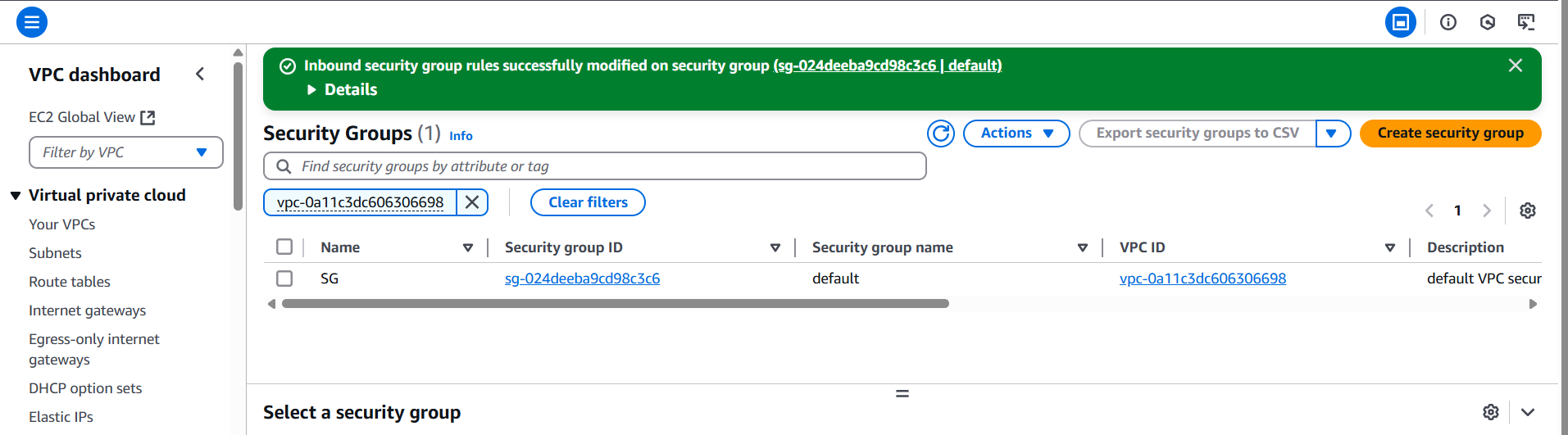


Fig: Security group (SG).

Step9: Create two EC2 instances (instance-01 & instance-02) in each subnet (private & public).

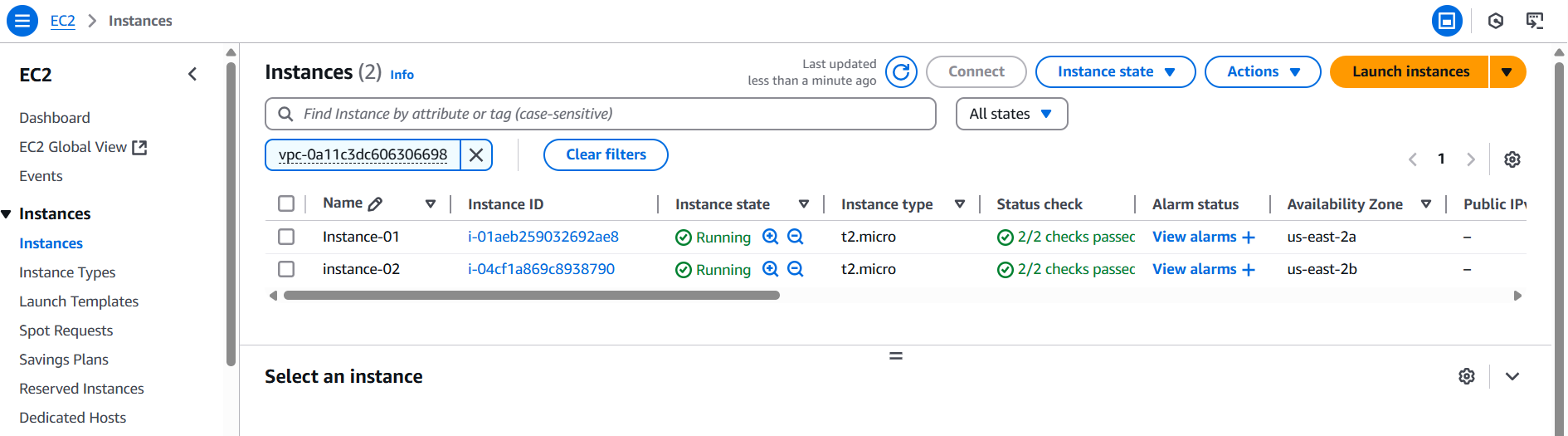


Fig: Two EC2 instances (instance-01 & instance-02) are created successfully.

**Step10:** Login in to the two instances using connection option or putty or mobaxterm.

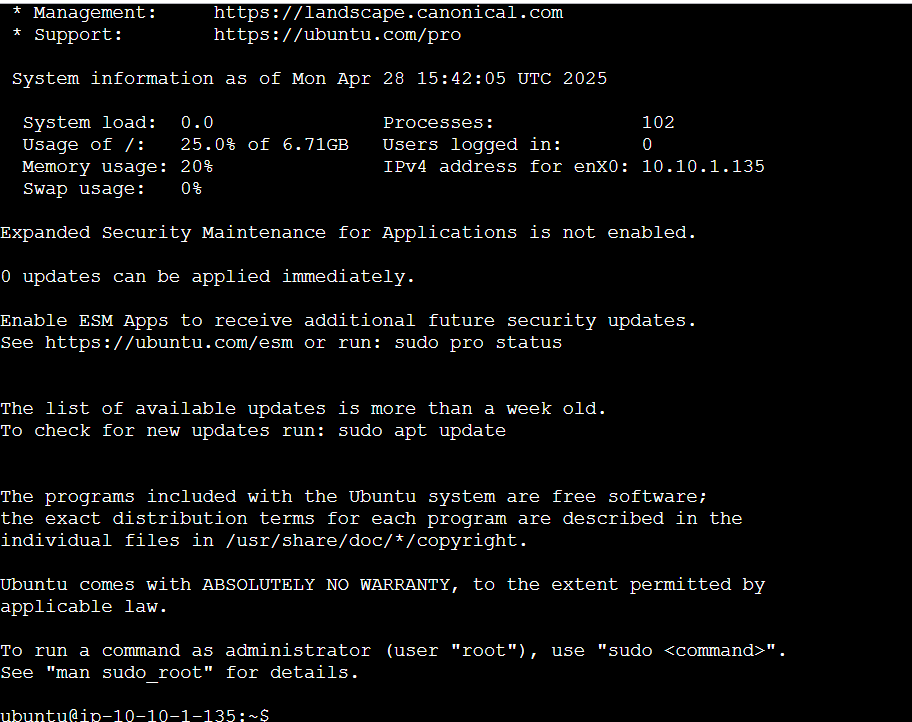


Fig: Successfully login into the instance-01.

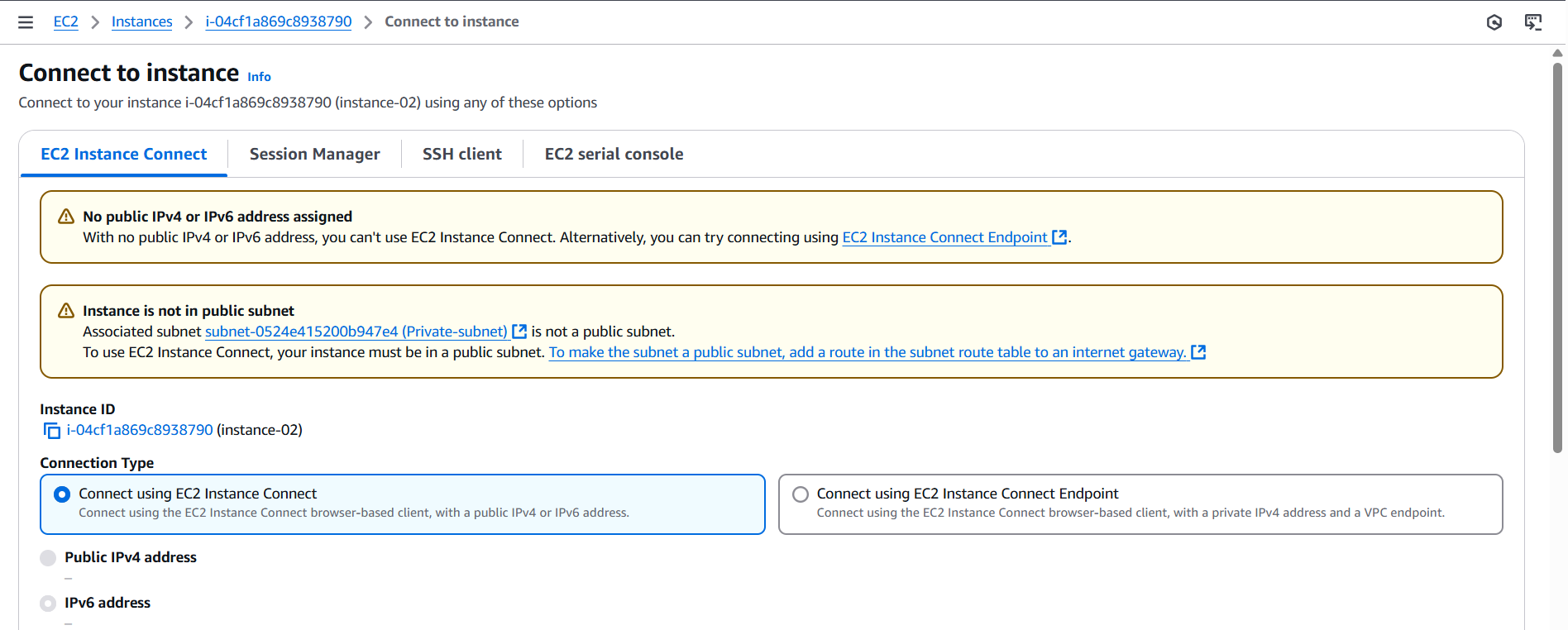


Fig: Login into the instance-02.

While Login into the instance-02 which is present in the Private-subnet, it showing the notification as shown above figure.

That notification describes that without public IP and Instance-02 is not in a public subnet, we cannot login/connect to the instance directly.

So in order to Login into the instance (instance-01) which is present in the private subnet we use the other instance (instance-02) which present in the public subnet of same VPC using the key pair (.pem) as shown in below.

1. Login into instance-01 (which is present in public subnet).
2. Create a .pem file in the instance-01
3. And copy and paste the key-pair in the created .pem file
4. Now change the mode of the .pem file to 400 (owner can only read).
5. Then execute the below command

**Command:** ssh –i .pem ubuntu@private IP of instance-02

**Ex:** ssh – i harish.pem [ubuntu@10.10.2.197](mailto:ubuntu@10.10.2.197)

**Change mode Linux command:**

Command: chmod Owner | Group | other users.

 **4**: Read permission (r)

 **2**: Write permission (w)

 **1**: Execute permission (x)

 **0**: No permission (-)

**Example:** chmod 400 <file-name>

 **4 (Owner)**: The owner of the file has **read** permission (4). They cannot write to (2) or execute (1) the file.

 **0 (Group)**: Members of the file's group have **no** permissions (0). They cannot read, write, or execute the file.

 **0 (Others)**: All other users on the system have **no** permissions (0). They cannot read, write, or execute the file.

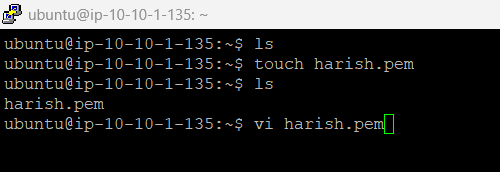


Fig: Creating of .pem file.

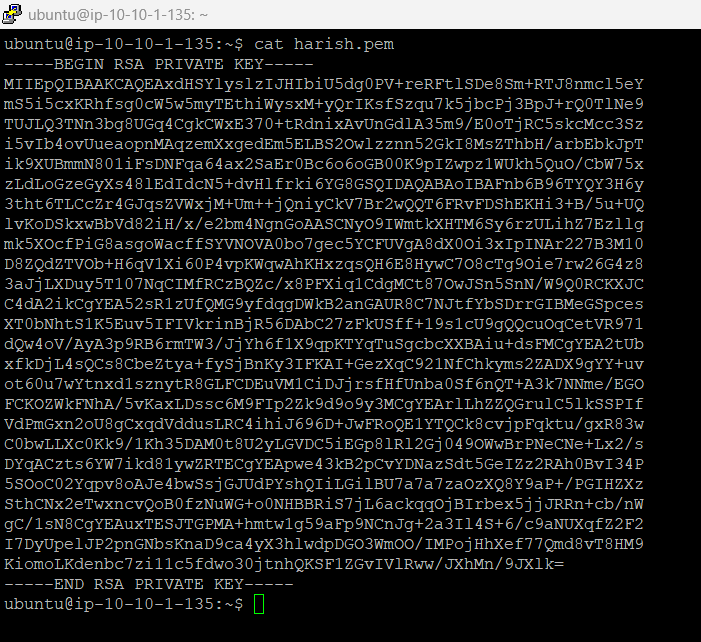


Fig: Pasting the key pair content in to the newly created .pem file in the Ubuntu instance-01.

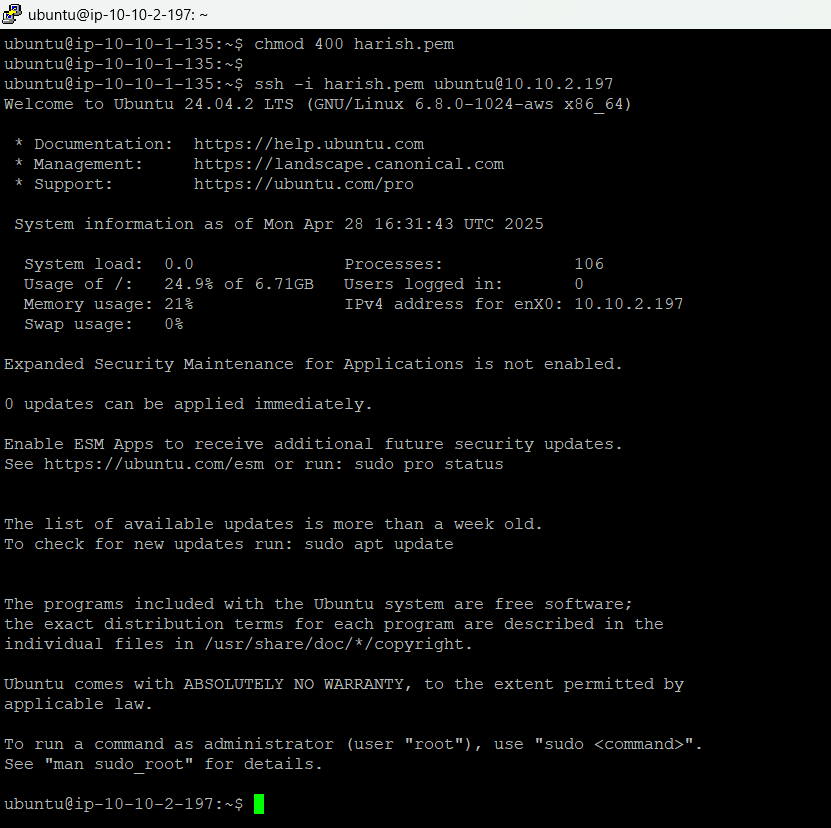


Fig: Successfully Login into the instance-02 which is present in the private subnet.

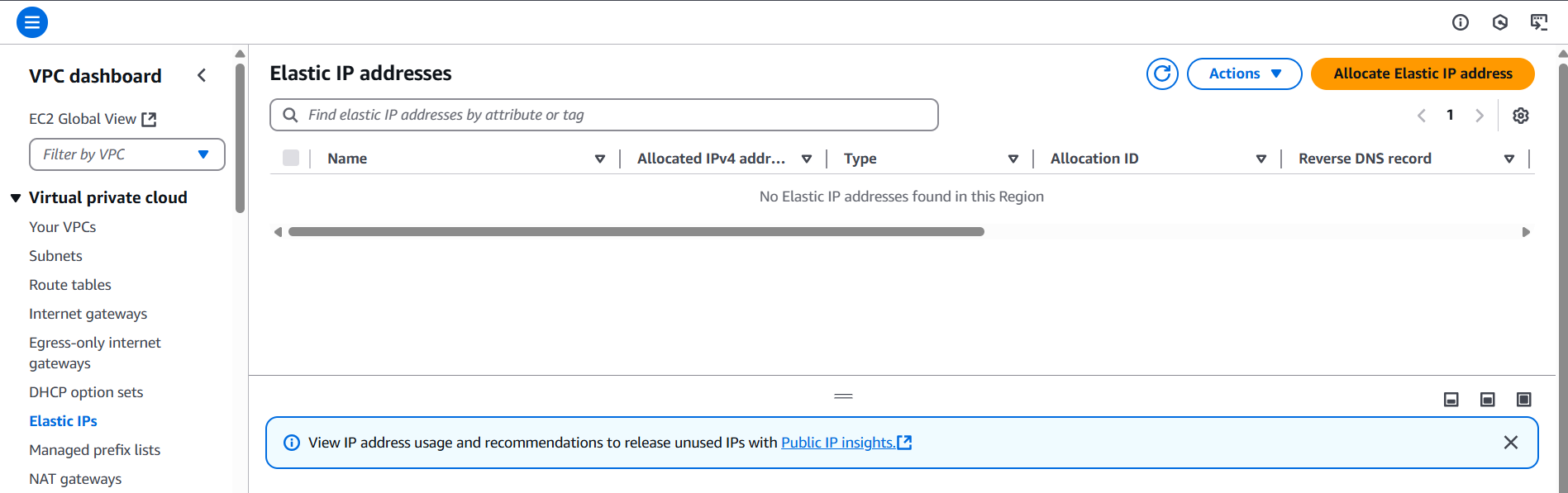
Since **instance-02** is in a **private subnet**, it doesn’t have a direct route to the Internet Gateway.  
Because of this, if you try to run commands like yum update or install Nginx, they will **fail** — the instance has **no internet connectivity**.

To allow **instance-02** (or any private subnet instance) to **update, install, or download software**, we use a **NAT Gateway**.

The **NAT Gateway** works by **replacing (masking)** the private IP address of the instance with its own **Elastic IP address** (public IP).  
This NAT Gateway is deployed inside a **public subnet**, which is connected to the Internet Gateway.

In order to configure the NAT gateway first we have to create an Elastic IP address.

**Step11:** Create an Elastic IP address.



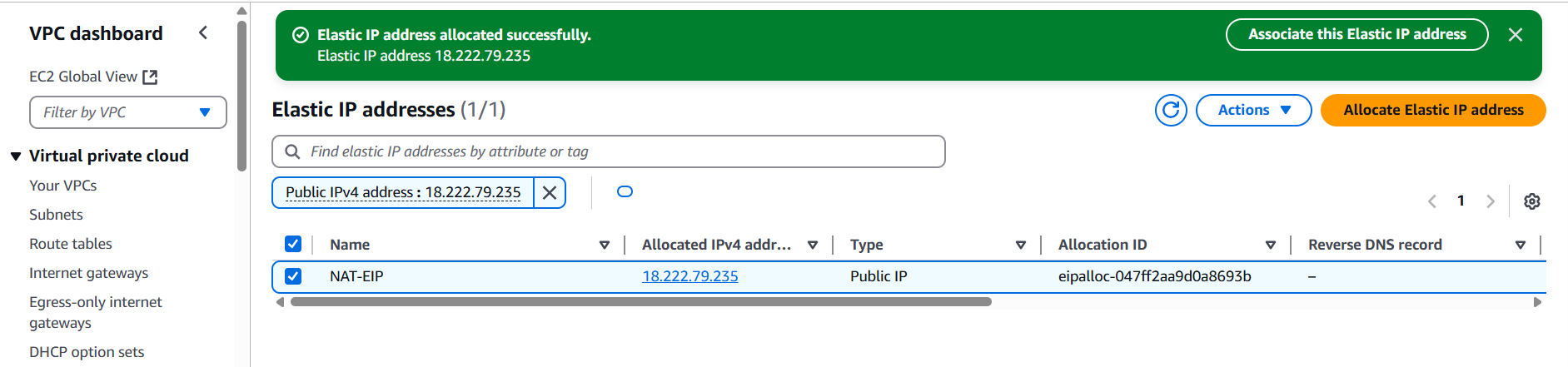
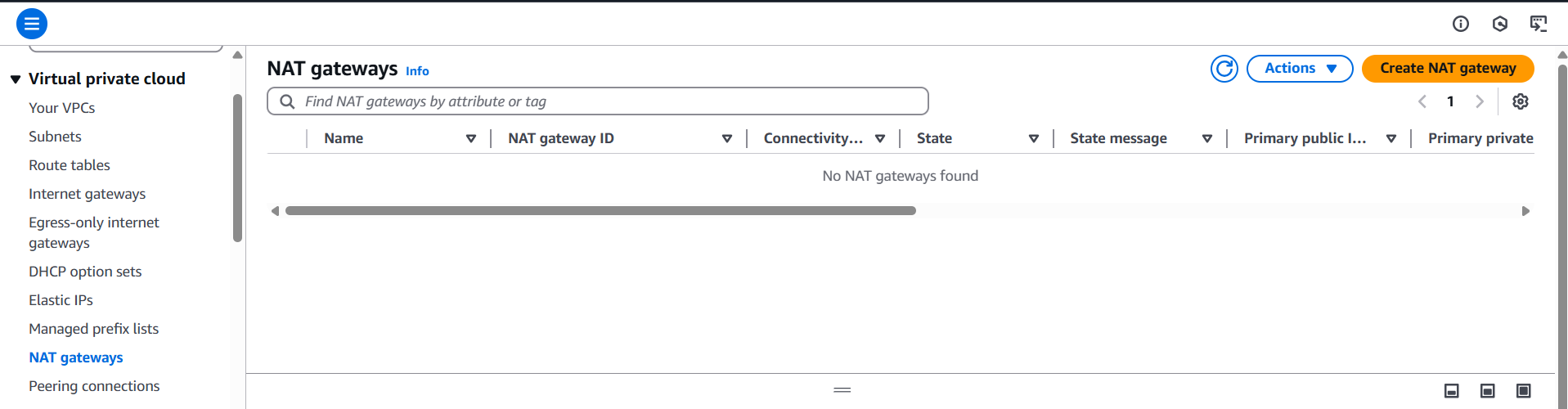


Fig: EIP is created successfully.

Step12: Create a NAT gate way.



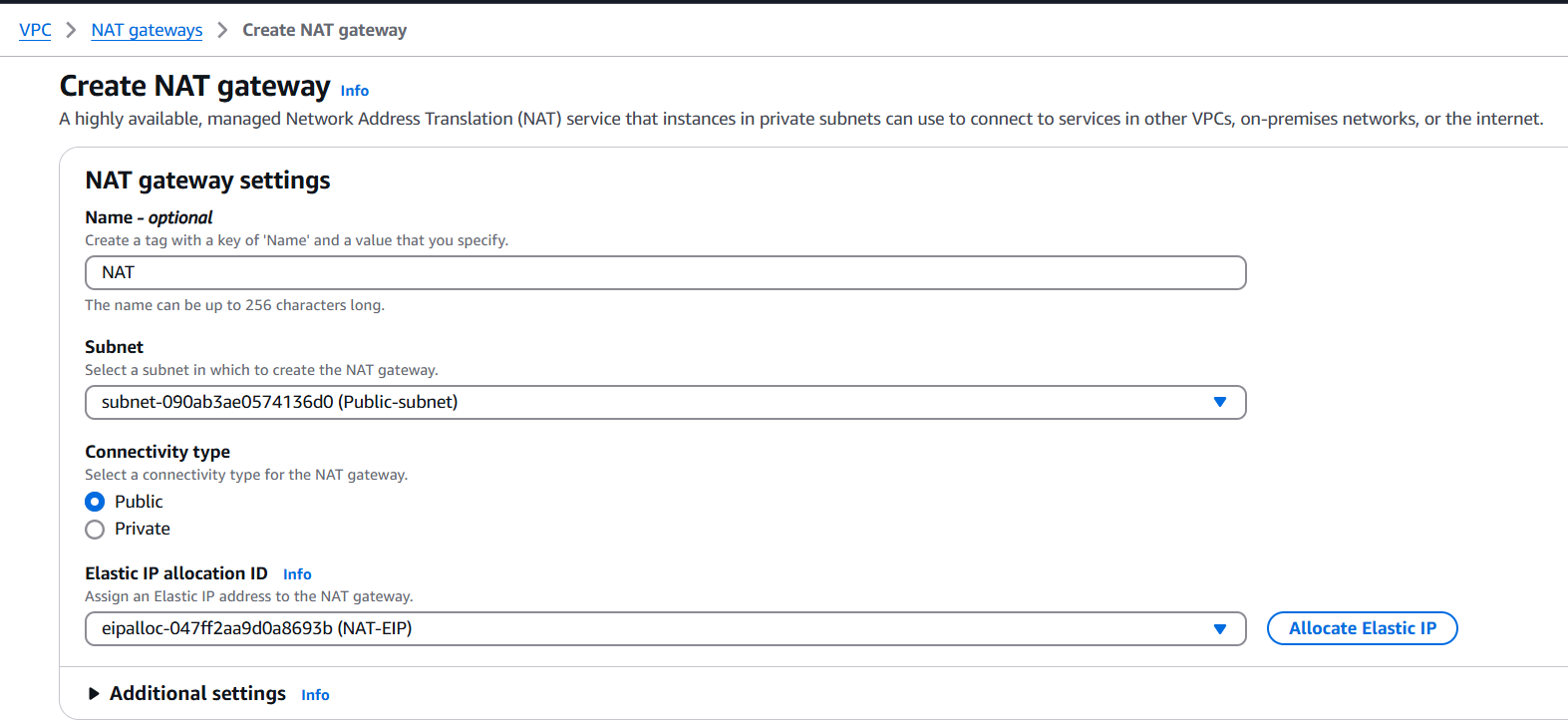
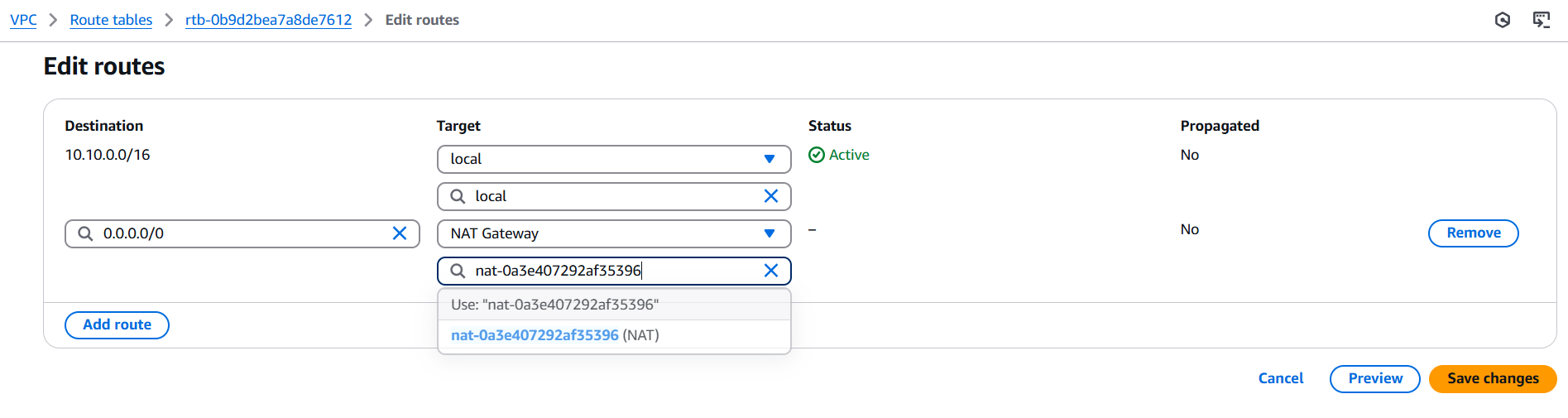


Fig: Assigning NAT to the public subnet.

**Step13:** Now write the route table (RT-02 private) in such a way to rout traffic using the NAT gate way.



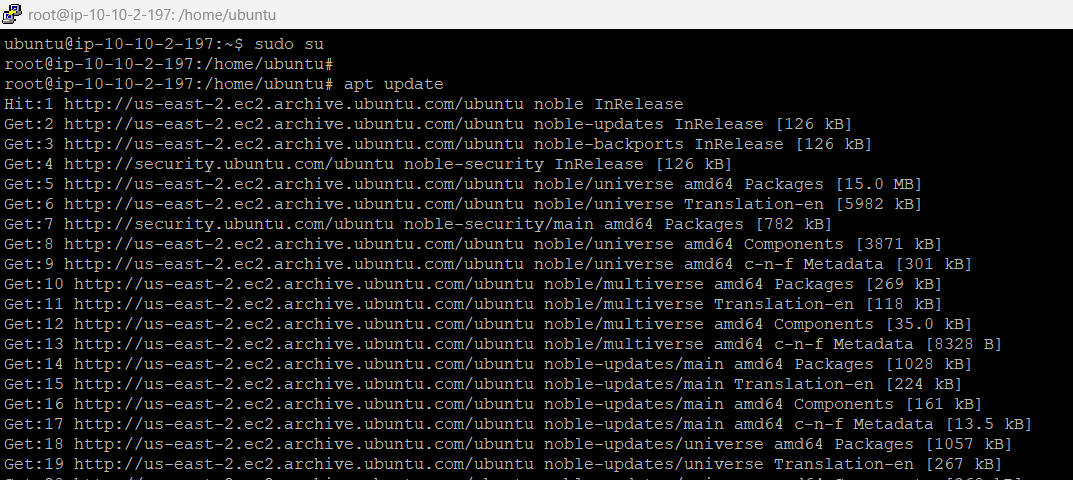


Fig: Updated successfully.

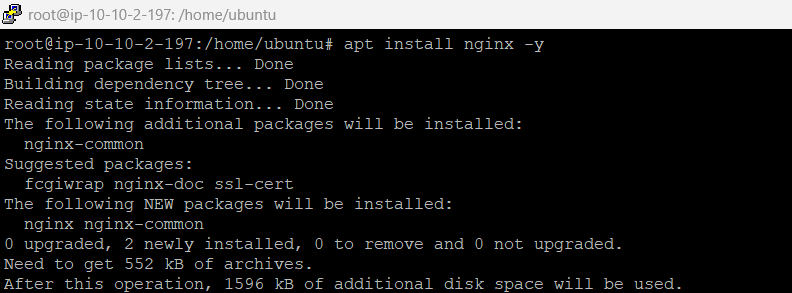


Fig: Installing of Nginx in the instance-02 which is present in the private subnet using NAT gate.