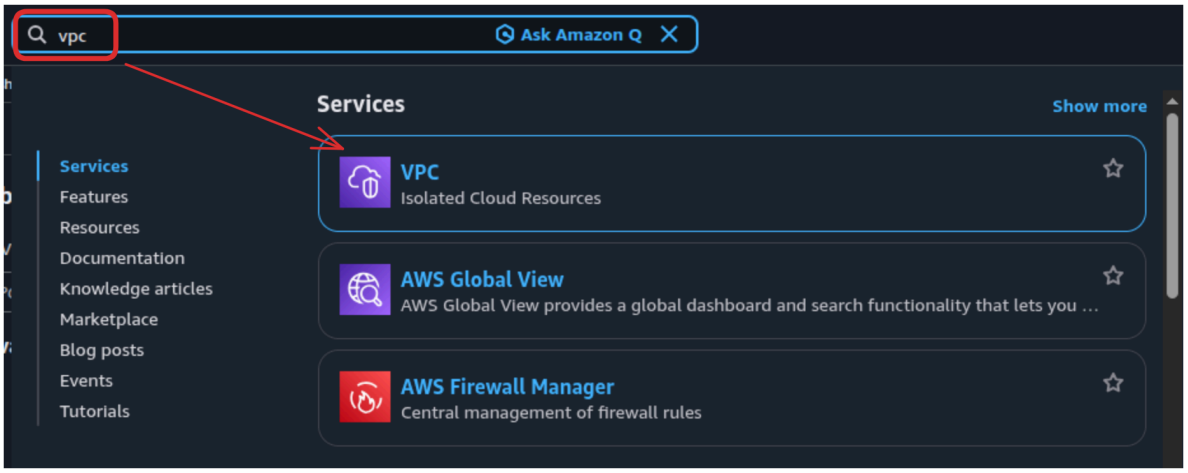
AWS Networking I

In this practice you will learn how to build a VPC architecture with public and private subnets.

VPC

To create a vpc go to the AWS console on VPC.



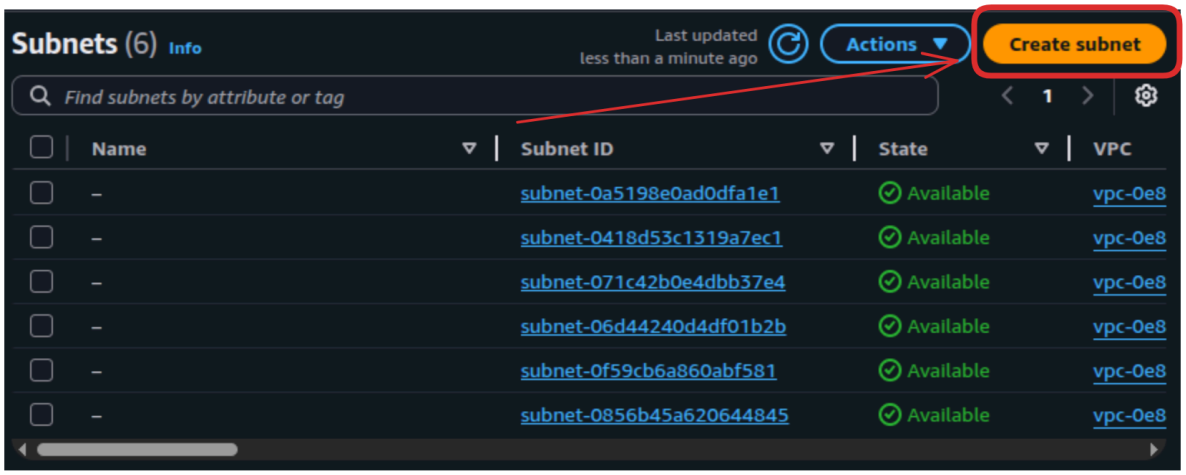
You will have a default VPC on your region (Example us-east-1), but usually will use a custom one for more control. Remember while VPCs are regional scoped, subnets are availability zone scoped (AZ us-east-1a for example).

You can set up a name and an IPv4 CIDR block as 10.0.0.0/16, depending on how many instances you want to be able to be hosted by the VPC. You can check how many you can allocate here.

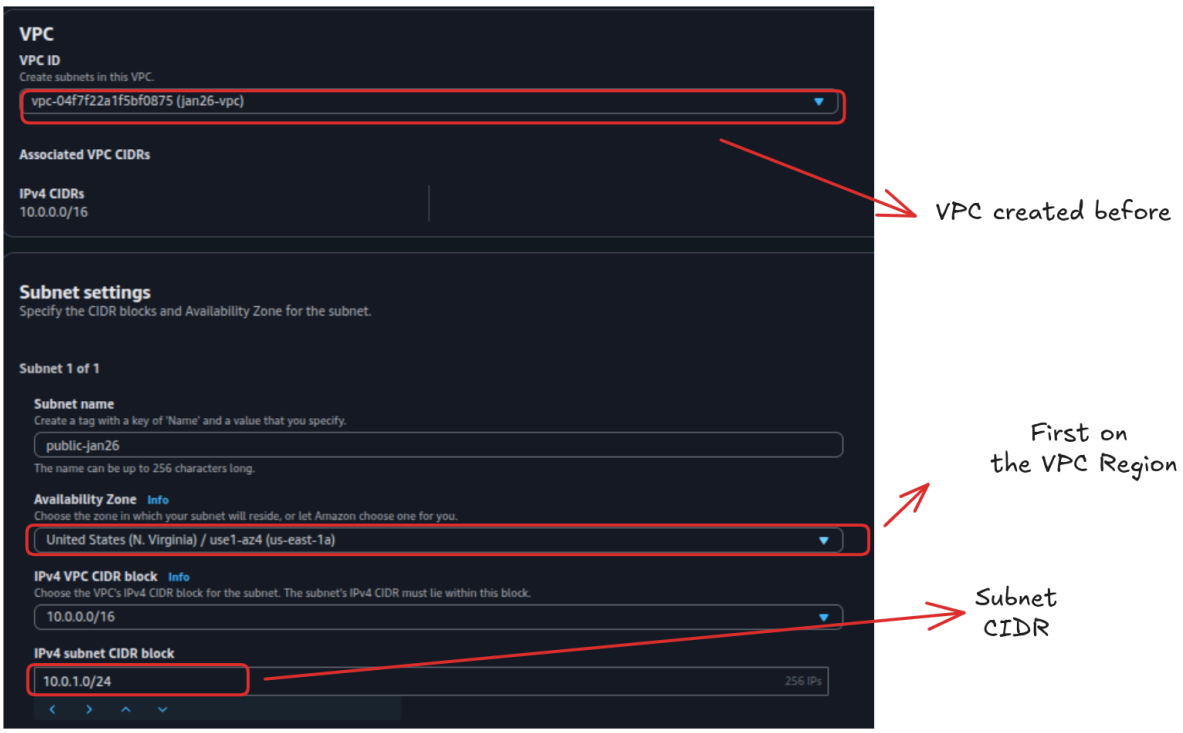
Subnet

Now let's create subnets, for this example will need a public and a private subnet, in order to emulate a production environment where usually you don't want all your instances or services to be reachable from outside (private subnet) but you want to be able to connect yourself, make updates and expose services to public usage (public subnet).

As before, go to subnets and then "Create subnet".



In the reference image above I have a couple of subnets that were created by default, we will proceed to create a public and a private subnets for this exercise.

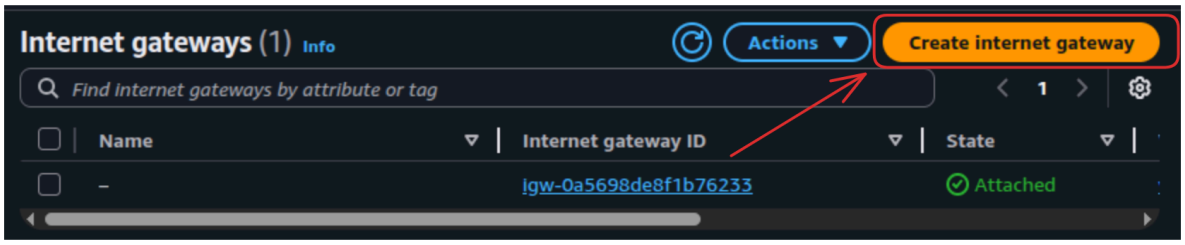


Note that the CIDR uses a “1” that the VPC CIDR didn't had, this is used to make a subset of all the possible IPs on the VPC for logical grouping, being able to use 251 IPs (self, default gateway, DNS, future usage, broadcast are blocked).

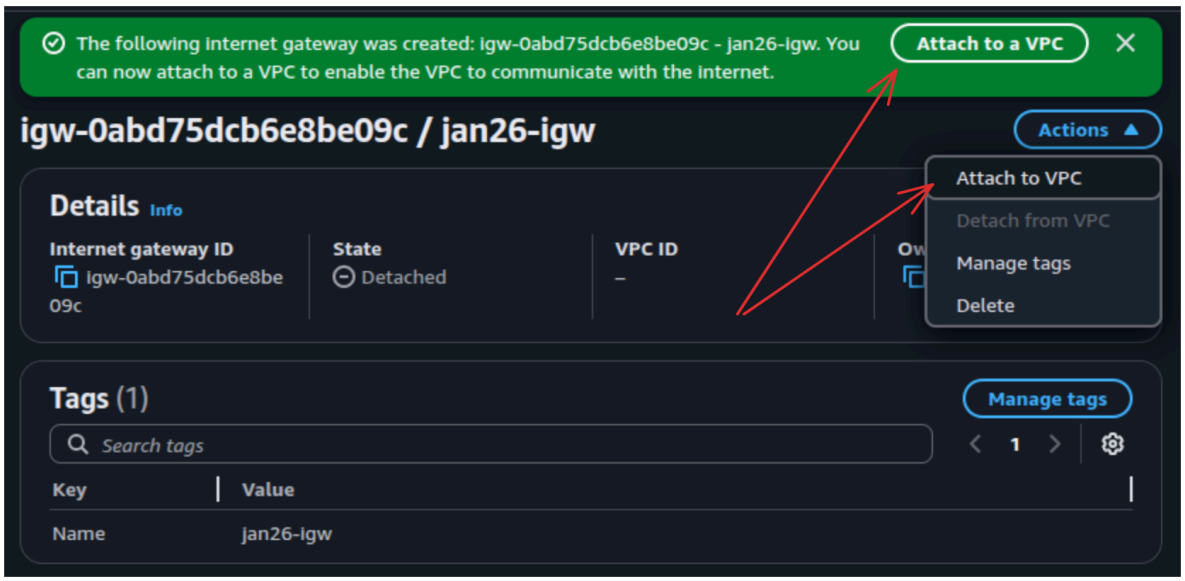
Then, create the private subnet as “private-jan26” using “10.0.2.0/24” CIDR.

Internet Gateway

We also want internet access for the VPC, so we should create an internet gateway. Remember IGWs have 1:1 relations with VPCs.



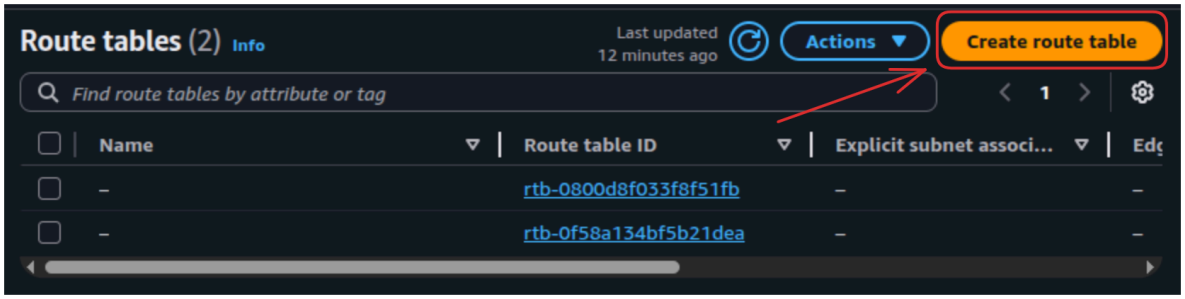
Then, proceed to attach it to the desired VPC by clicking on the referenced buttons below, and later selecting the VPC.



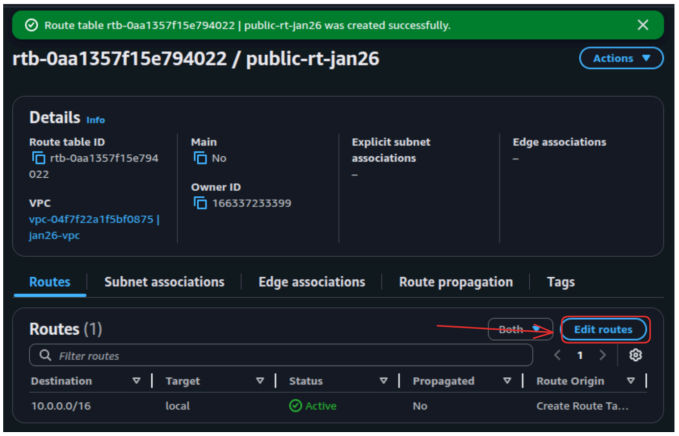
Route tables

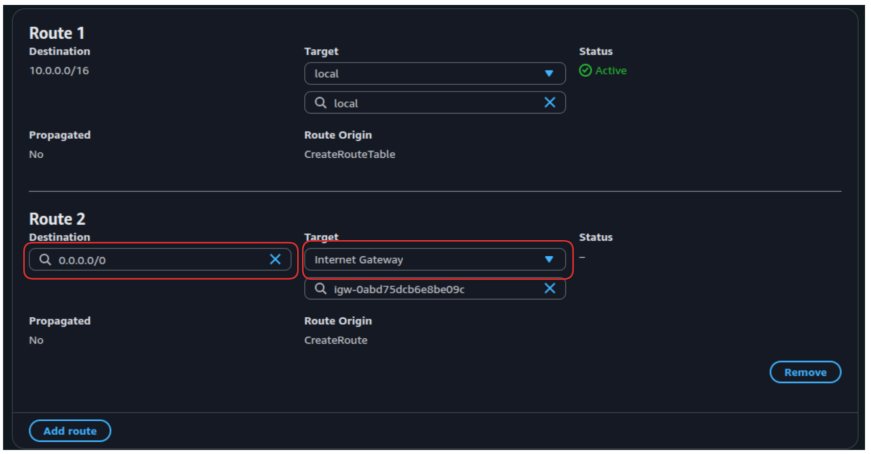
Also we’ll need routing tables, each subnet has a 1:1 relationship with them. The routing tables set up rules on subnets in order to connect to other subnets or internet gateways

- The subnet becomes “public” or “private” purely based on its route table.
- Route tables exist in VPC scope but work on subnet level. Local VPC CIDR allows connection among all VPC subnets.
- One route table → many subnets is allowed.
- One subnet → one route table, no more than one.

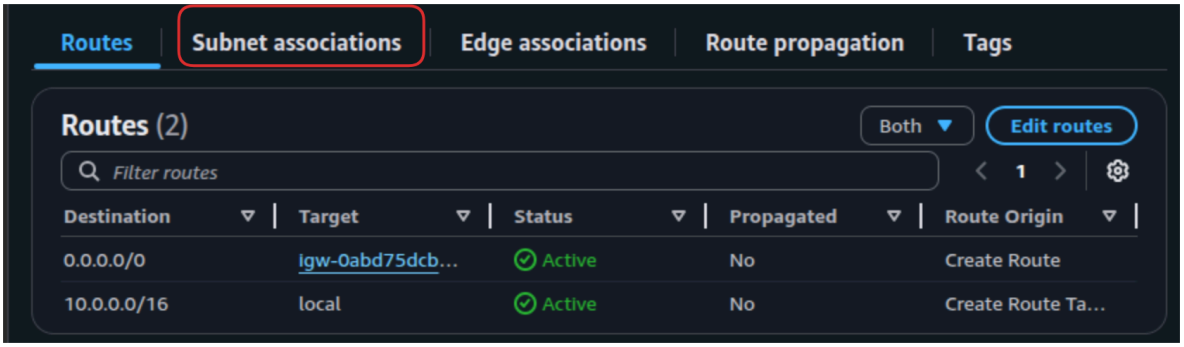


After creating the routing table, go to “Edit routes” as shown below, then “Add route”, here you will setup the 0.0.0.0 destination (all IPs) and use “Internet Gateway” previously created as target, in order to connect to the internet.





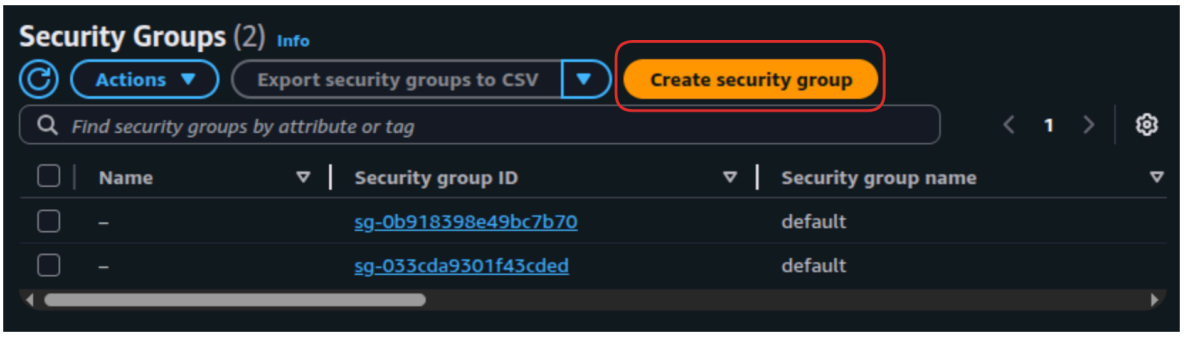
Now, we created the routing table but didn't associate it to the public subnet yet, so proceed to the “Subnet associations” tab, then click on “Edit subnet associations”, and select your “public-jan26” subnet.



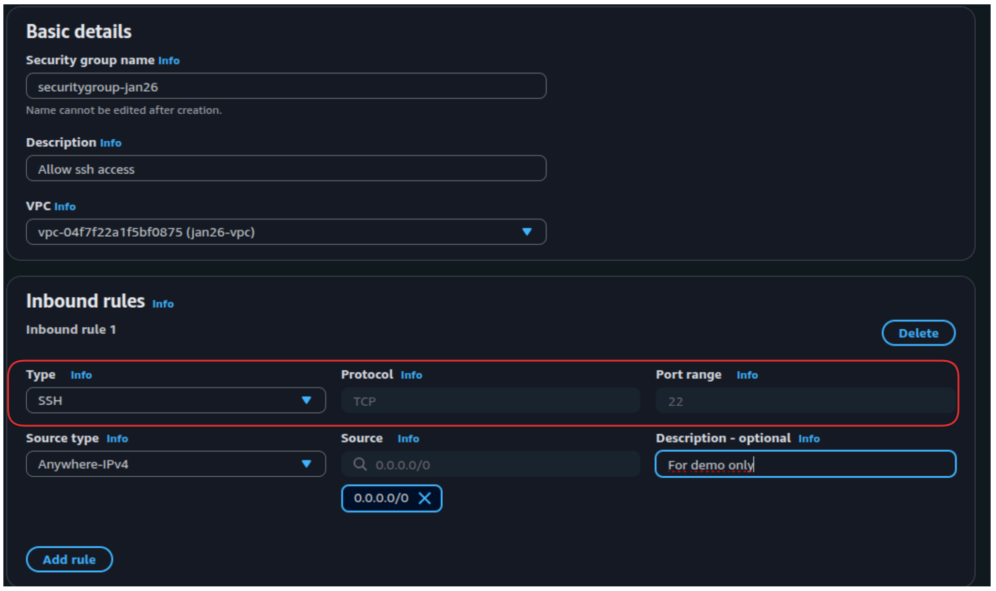
Security Groups

Let's set up security groups now. These work as firewalls to allow/deny access to instances, so remember, they work on instance level.

So, go to the "Security Groups" tab, then "Create security group".



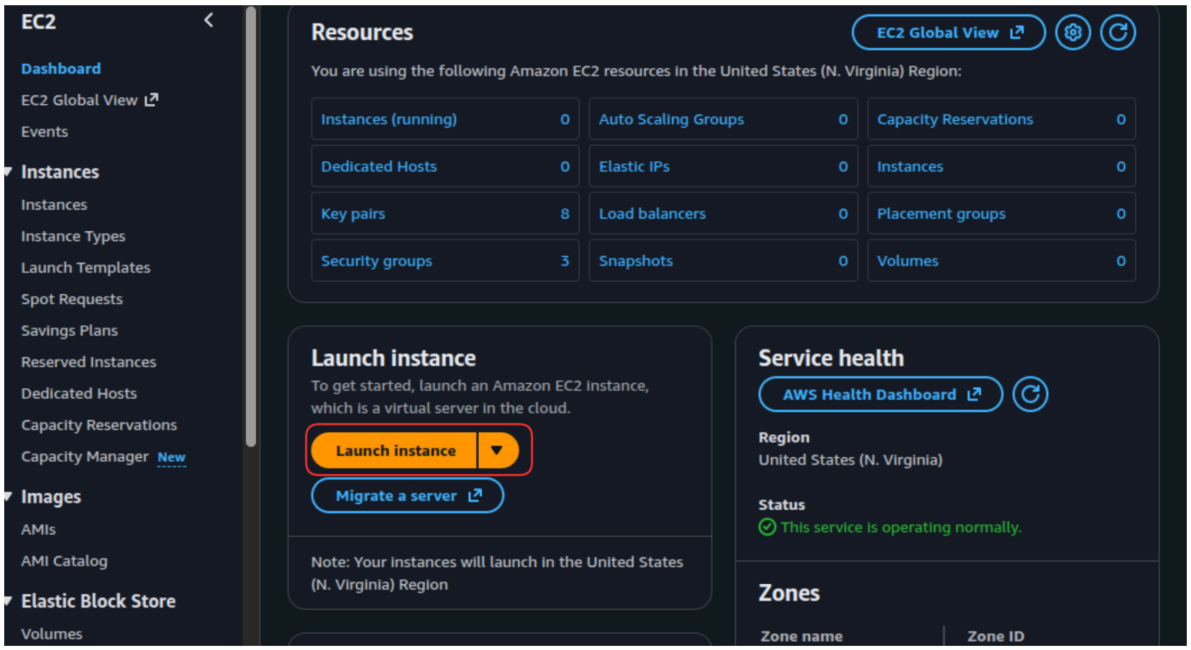
As settings, click on "Edit inbound rules", and add a SSH rule with source 0.0.0.0/0 (this is only for demo purpose, in production must allow only specific IPs for security reasons).



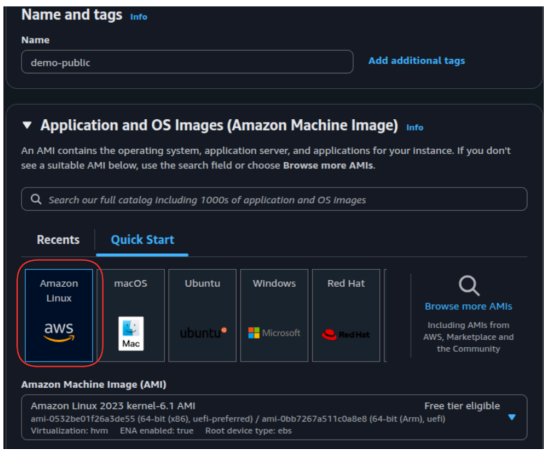
EC2 instance (public)

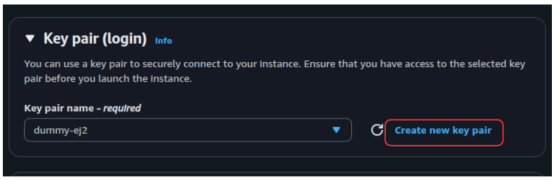
With everything already set up will now create the public and private EC2 instances, starting with the public one. This is not more than assigning each of them to their respective subnet following steps detailed below.

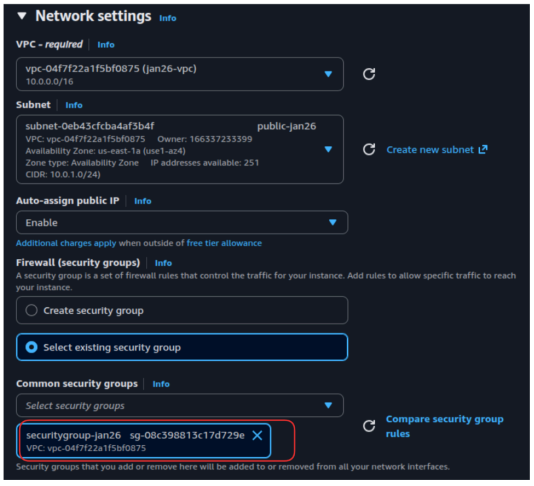
First, go to EC2 and click on “Launch Instance”.



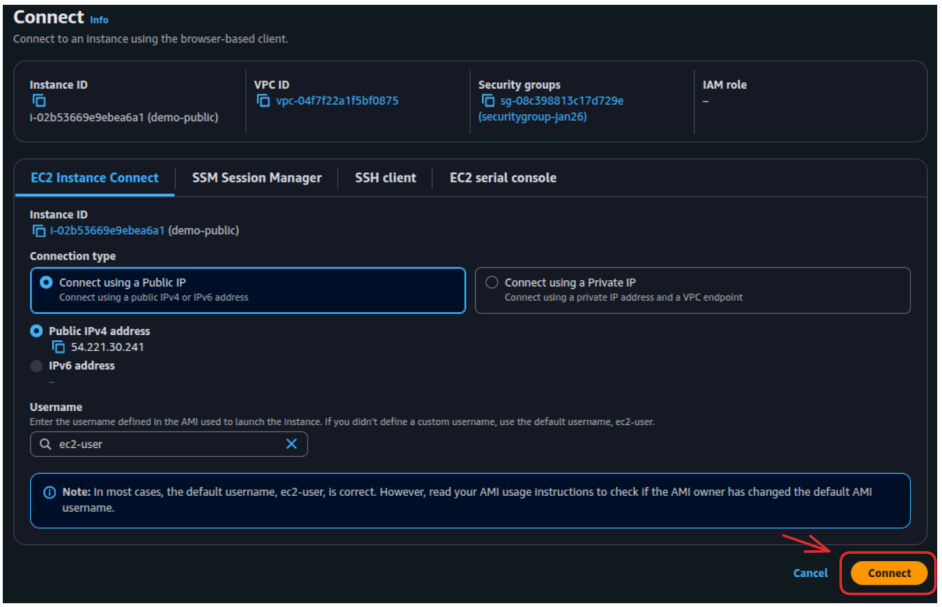
Then you will have to setup the following: OS Image (Amazon Linux), Instance Type (free tier), create key pair, Edit Network settings selecting all the components previously created (VPC, public subnet, Enable “auto assign public IP”, select security group), and then launch the instance (below the important commands are highlighted).

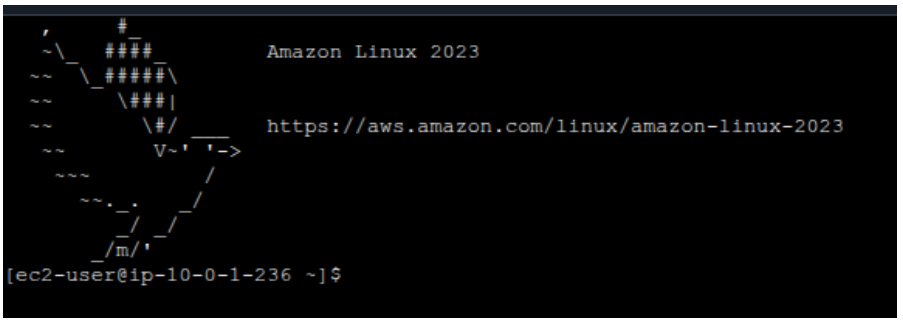






To check if all went right, click on the instance, when the instace state is “running” you should be able to connect.





EC2 instance (private)

Must follow the same steps, could use the same .pem key created before to connect, VPC is the same but the subnet is the private one, Auto-assign public IP: DISABLE, Security group: Same as public instance.

Then, in order to connect will be necessary to copy the .pem key to the public instance as it allows you to connect through ssh, which is allowed by the security group but just in the same network.

Copy with the following command:

scp -i dummy-ej2.pem dummy-ej2.pem ec2-user@<PUBLIC_IP>:~/

Then connect to the private instance through the public one using:

- SSH into public instance first
- From public instance, SSH to private:

chmod 400 dummy-ej2.pem
ssh -i jdummy-ej2.pem ec2-user@<PRIVATE_IP>

- Check hostname to confirm you're on different machine:

  hostname

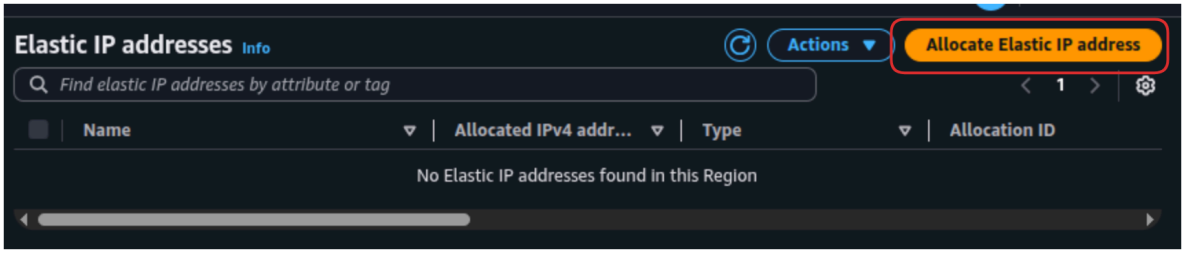
You're now in a private instance via jump box!

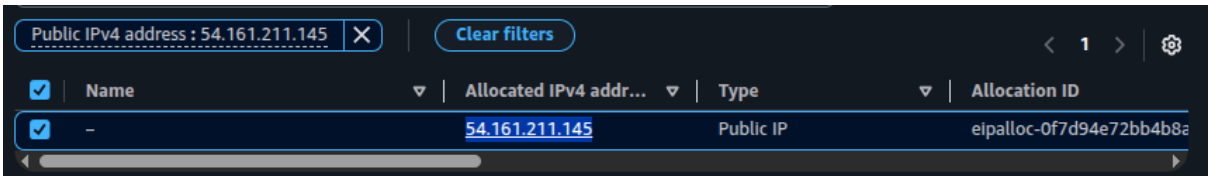
```
[ec2-user@ip-10-0-1-236 ~]$ ssh -i "dummy-ej2.pem" ec2-user@10.0.2.133
The authenticity of host '10.0.2.133 (10.0.2.133)' can't be established.
ED25519 key fingerprint is SHA256:aOtrFbnzt4Dq53Hql6/A+TABMoHFGk7eomYg+Ut9QM8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.133' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-2-133 ~]$
```

NAT Gateway

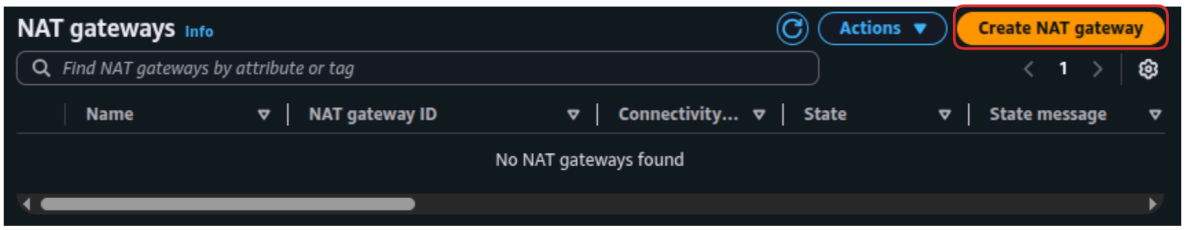
Finally, to allow private instances to connect to the internet for updates (only outbound connection) we use NAT Gateway.

We'll need a Elastic IP, so will create one in VPC "Elastic IP adresses".

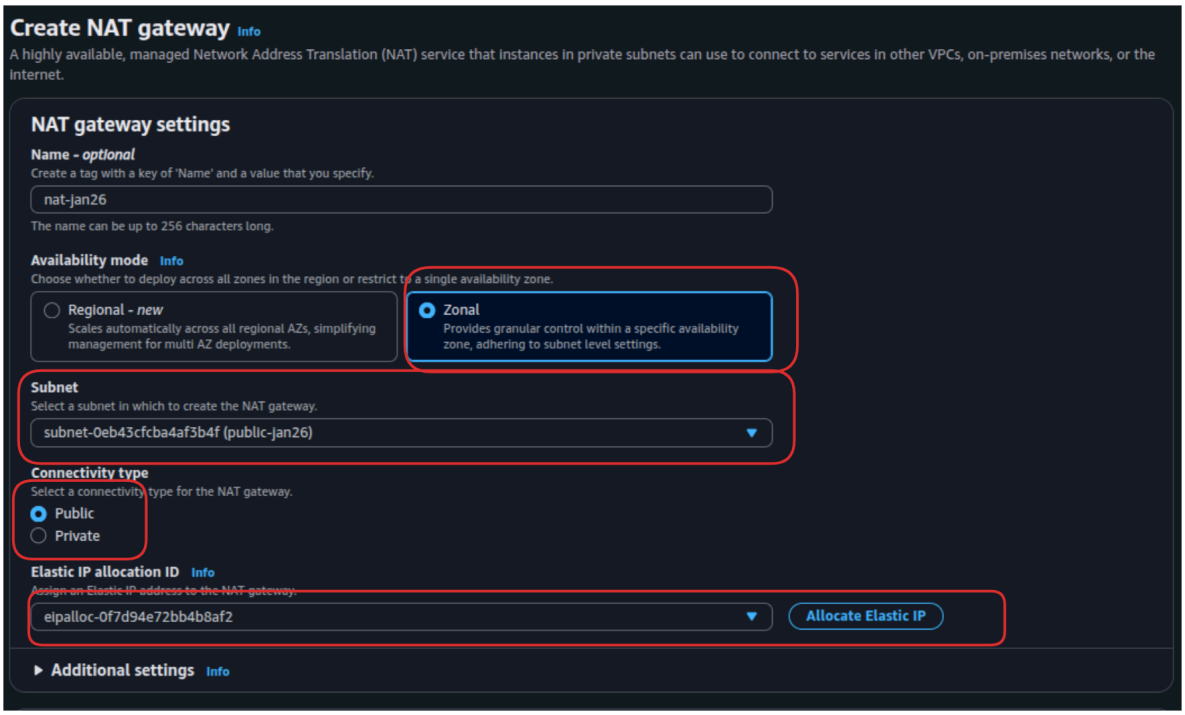




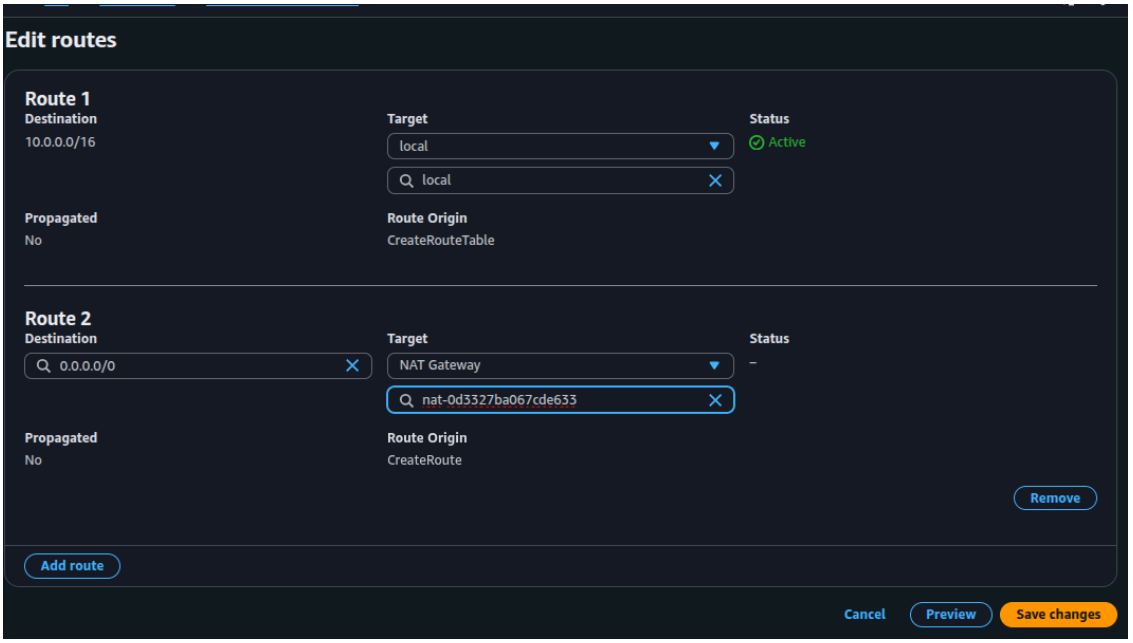
Now will create a NAT Gateway.



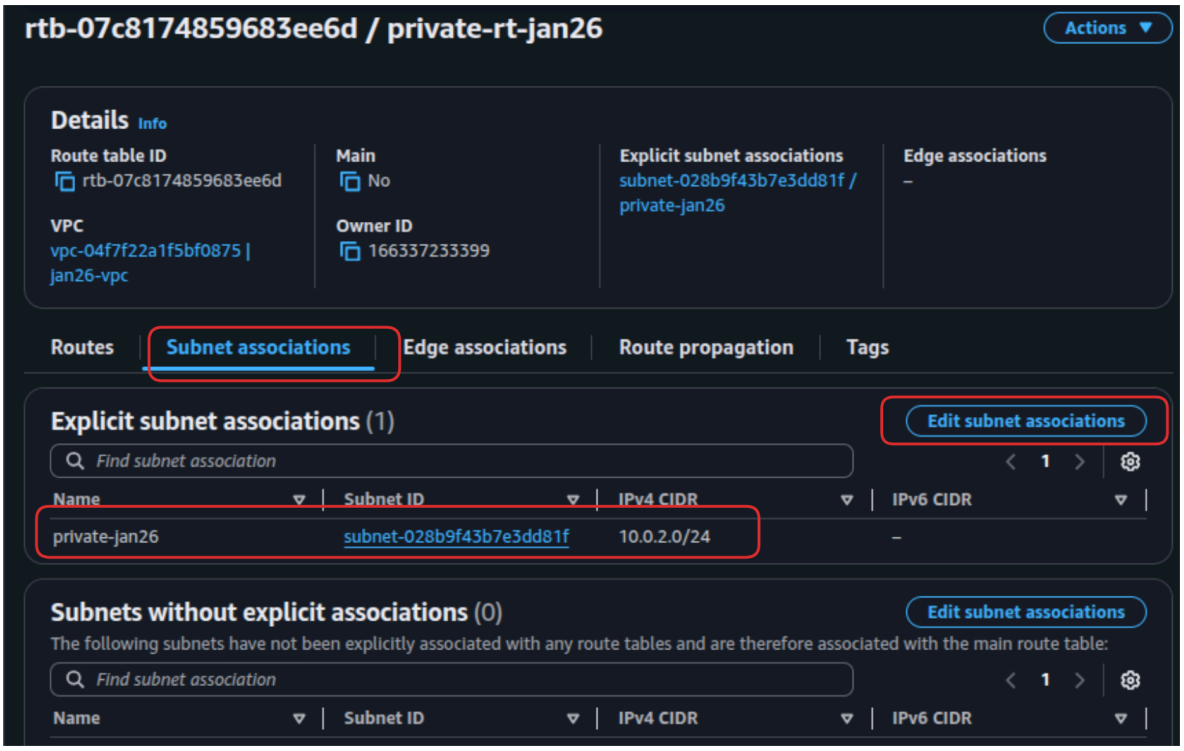
With the following settings (note that subnet is public, as it allows internet connection, private subnet will receive connection through NAT Gateway and public subnet).



Now create the private route table as the public describe above (route table section), and in the “Edit routes” section add a NAT Gateway route as below.



Then associate the route table to the private subnet (similar to the “route table” section described earlier).

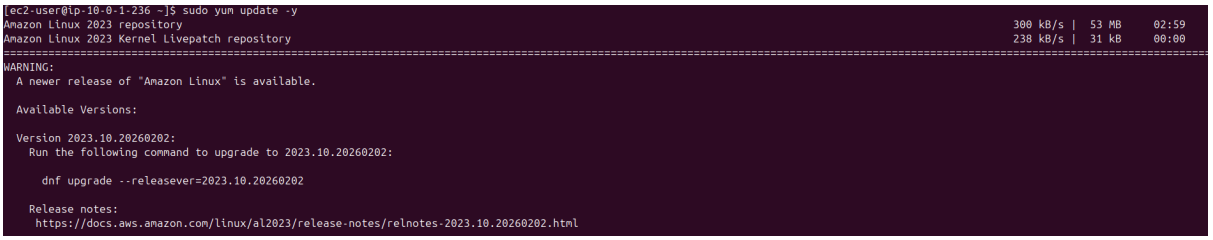


In the security group also is important to add the outbound rule that allows https connection, so the instances can get this tipe of connection.

Finally check if everything works running:

sudo yum update -y

If it receives packages, everything is right, if not, check steps again.



Remember to always delete all resources used (nat gateway, elastic IP, instances, subnets, VPCs, route tables, etc.) so don’t get surprise charges.