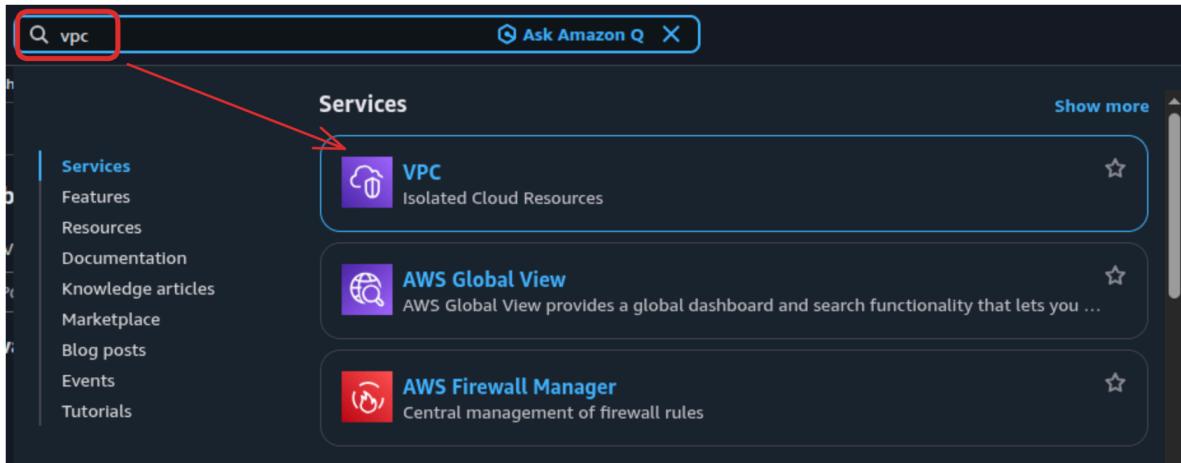


AWS Networking I

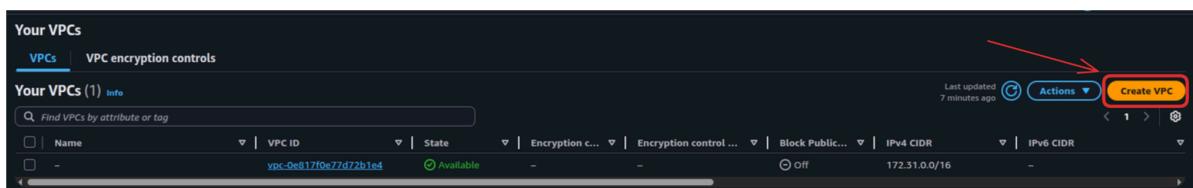
In this practice you will learn how to build a VPC architecture with public and private subnets.

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](#).

A screenshot of the 'Create VPC' configuration form. It includes fields for 'Name tag - optional' (jan26-vpc), 'IPv4 CIDR block' (selected 'IPv4 CIDR manual input', value '10.0.0.0/16'), and a note about CIDR size ('CIDR block size must be between /16 and /28').

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".

Subnets (6) Info				
Last updated C less than a minute ago Actions ▾ Create subnet				
<input type="checkbox"/>	Name	Subnet ID	State	VPC
<input type="checkbox"/>	-	subnet-0a5198e0ad0dfa1e1	Available	vpc-0e8
<input type="checkbox"/>	-	subnet-0418d53c1319a7ec1	Available	vpc-0e8
<input type="checkbox"/>	-	subnet-071c42b0e4dbb37e4	Available	vpc-0e8
<input type="checkbox"/>	-	subnet-06d44240d4df01b2b	Available	vpc-0e8
<input type="checkbox"/>	-	subnet-0f59cb6a860abf581	Available	vpc-0e8
<input type="checkbox"/>	-	subnet-0856b45a620644845	Available	vpc-0e8

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.

VPC
VPC ID
Create subnets in this VPC.

Associated VPC CIDRs
IPv4 CIDRs
10.0.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

Availability Zone [Info](#)
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 VPC CIDR block [Info](#)
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

IPv4 subnet CIDR block
 256 IPs

VPC created before
First on the VPC Region
Subnet CIDR

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).

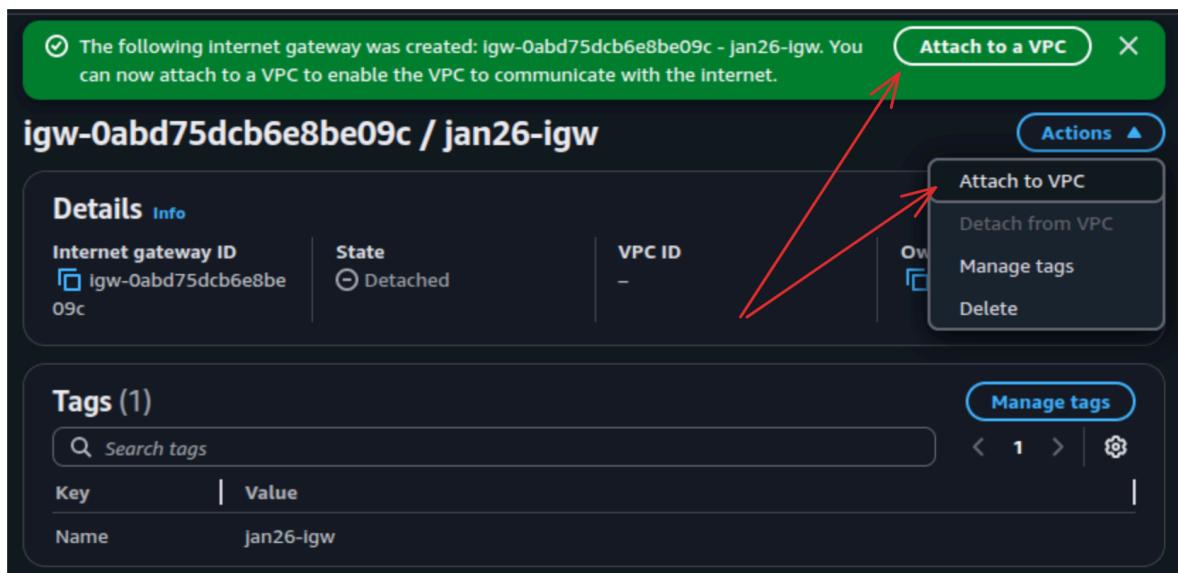
Then, create the private subnet as “private-jan26” using “10.0.2.0/24” CIDR.

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.

Route tables (2)		Info	Last updated	C	Actions	Create route table
	Name	Route table ID	Explicit subnet associ...	Ed...		
	-	rtb-0800d8f033f8f51fb	-	-		
	-	rtb-0f58a134bf5b21dea	-	-		

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.

Route table rtb-0aa1357f15e794022 | public-rt-jan26 was created successfully.

rtb-0aa1357f15e794022 / public-rt-jan26

Details		Info
Route table ID	rtb-0aa1357f15e794022	Main
VPC	vpc-04f7f22a1f5bf0875	No
Owner ID	166337233399	Explicit subnet associations
		Edge associations

Routes Subnet associations Edge associations Route propagation Tags

Routes (1)

Both Edit routes

Destination	Target	Status	Propagated	Route Origin
10.0.0.0/16	local	Active	No	Create Route Ta...

Route 1

Destination: 10.0.0.0/16

Propagated: No

Target: local

Status: Active

Route Origin: CreateRouteTable

Route 2

Destination: 0.0.0.0/0

Propagated: No

Target: Internet Gateway

Status: -

Route Origin: CreateRoute

Add route Remove

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.

Routes	Subnet associations	Edge associations	Route propagation	Tags
Routes (2)				
<input type="button" value="Both"/> < 1 > <input type="button" value="Edit routes"/>				
Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0abd75dc...	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Ta...

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".

Actions		Export security groups to CSV	Create security group
<input type="button" value="Find security groups by attribute or tag"/>			
<input type="checkbox"/> Name	▼	Security group ID	▼ Security group name
<input type="checkbox"/> -		sg-0b918398e49bc7b70	default
<input type="checkbox"/> -		sg-033cda9301f43cded	default

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).

Basic details

Security group name [Info](#)
securitygroup-jan26
Name cannot be edited after creation.

Description [Info](#)
Allow ssh access

VPC Info
vpc-04f7f22a1f5bf0875 (jan26-vpc)

Inbound rules [Info](#)

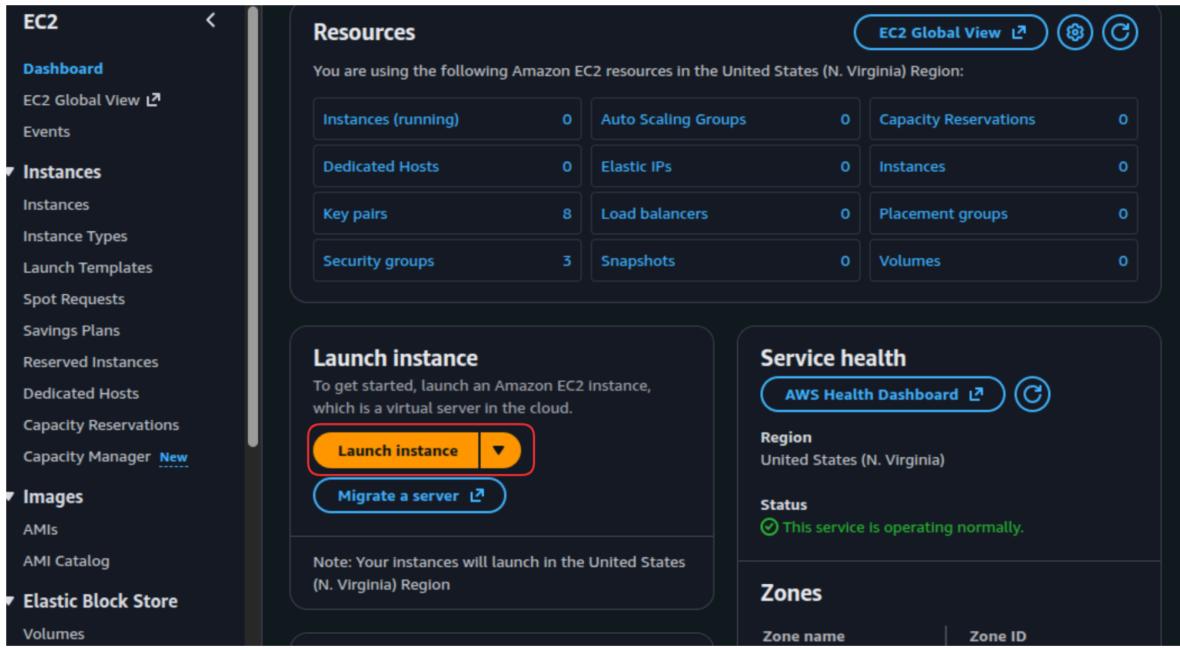
Inbound rule 1

Type Info SSH	Protocol Info TCP	Port range Info 22
Source type Info Anywhere-IPv4	Source Info <input type="text" value="0.0.0.0/0"/> <input type="button" value="X"/>	Description - optional Info For demo only

EC2 instance (public)

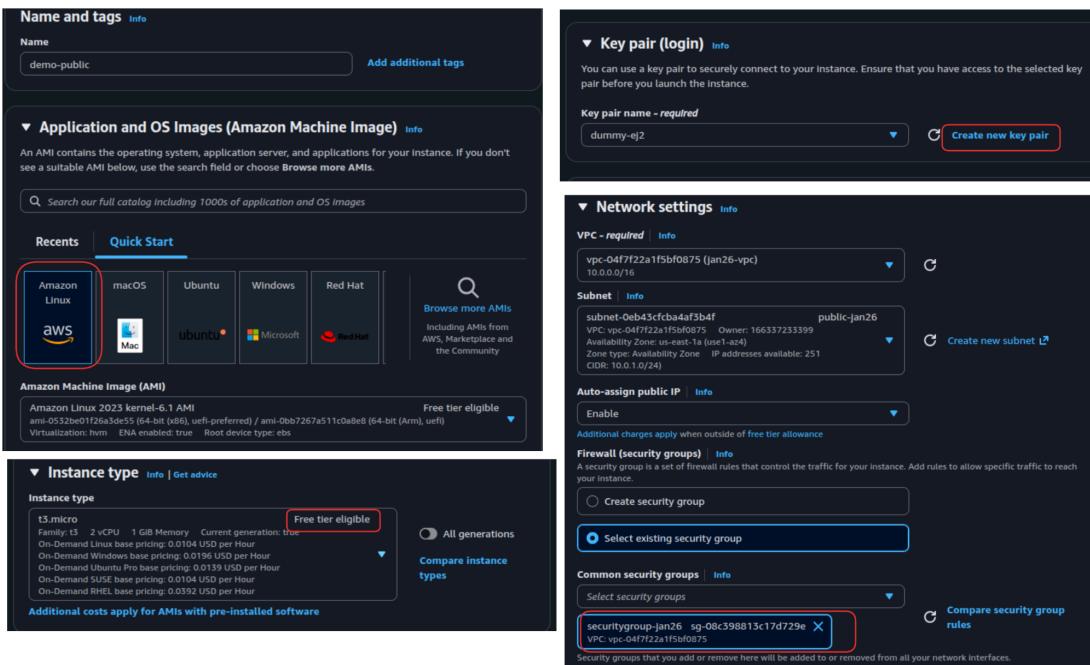
With everything already set up will now create the public and private EC2 instances, starting with the public one. This is not more than assigning each of them to their respective subnet following steps detailed below.

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



The screenshot shows the AWS EC2 console. On the left sidebar, under the 'Instances' section, the 'Launch Templates' option is selected. In the main content area, there's a 'Launch instance' section with a large 'Launch instance' button highlighted by a red box. Below it is a 'Migrate a server' button. A note at the bottom says 'Note: Your instances will launch in the United States (N. Virginia) Region'. To the right, there's a 'Service health' section showing 'AWS Health Dashboard' and 'Region: United States (N. Virginia)' with a green status indicator. At the top right, there are 'EC2 Global View', settings, and refresh icons.

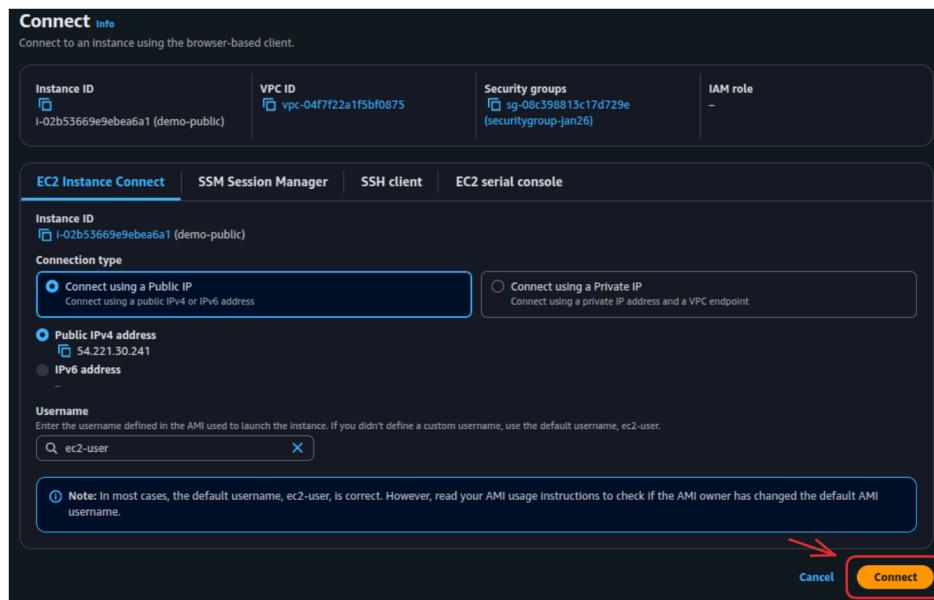
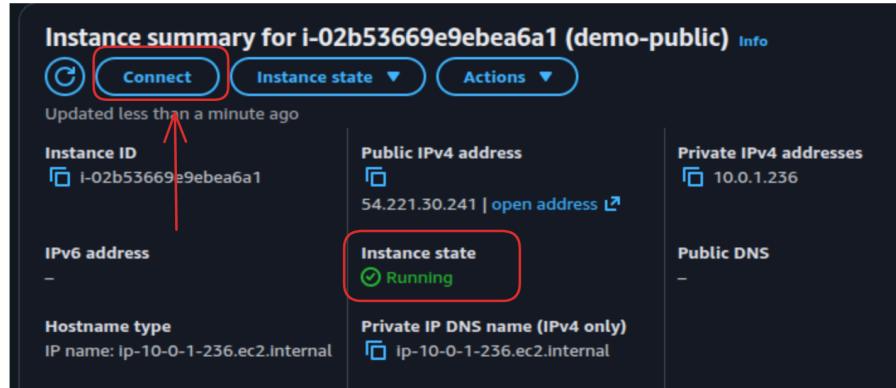
Then you will have to setup the following: OS Image (Amazon Linux), Instance Type (free tier), create key pair, Edit Network settings selecting all the components previously created (VPC, public subnet, Enable “auto assign public IP”, select security group), and then launch the instance (below the important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



The screenshot shows the 'Create New Instance' wizard. It has several sections:

- Name and tags**: Shows a 'Name' field with 'demo-public' and an 'Add additional tags' button.
- Application and OS Images (Amazon Machine Image)**: Shows a search bar and a 'Recent' section where 'Amazon Linux' is highlighted with a red box. Other options include macOS, Ubuntu, Windows, and Red Hat.
- Instance type**: Shows the 't3.micro' instance type as selected, with a 'Free tier eligible' badge. It provides details like family, CPU, memory, and current generation.
- Network settings**: Shows a VPC dropdown set to 'vpc-04f7f22a1fsbf0875 (jan26-vpc)', a subnet dropdown for 'subnet-0eb43fcba4af3b4f', and a 'Create new subnet' button.
- Common security groups**: Shows a dropdown with 'securitygroup-jan26 sg-08c398815c17d729e' selected, with a red box around it. It also includes a 'Compare security group rules' button.

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



```
#_/\_###_#_ Amazon Linux 2023
~~\_\_###\#\_
~~ \#\#\|_ https://aws.amazon.com/linux/amazon-linux-2023
~~ \#/ \_>
~~ V~ \_>
~~ \_ . / /
~~ \_ / \_ / [ec2-user@ip-10-0-1-236 ~]$
```

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!

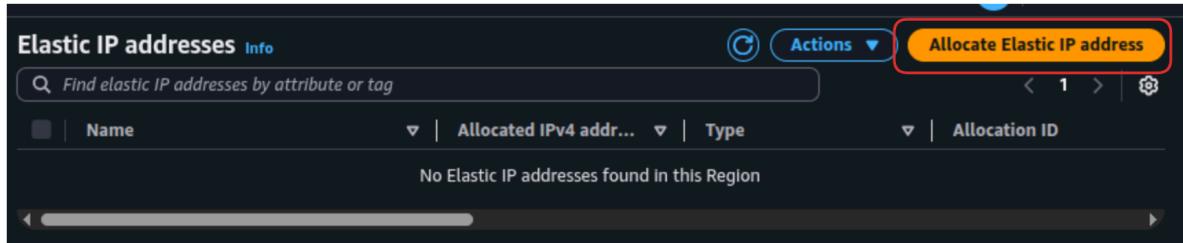
```
[ec2-user@ip-10-0-1-236 ~]$ ssh -i "dummy-ej2.pem" ec2-user@10.0.2.133
The authenticity of host '10.0.2.133 (10.0.2.133)' can't be established.
ED25519 key fingerprint is SHA256:a0trFbnzt4Dq53Hql6/A+TABMoHFGk7eomYg+Ut9QM8.
This key is not known by any other names
Are you sure you want to continue 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

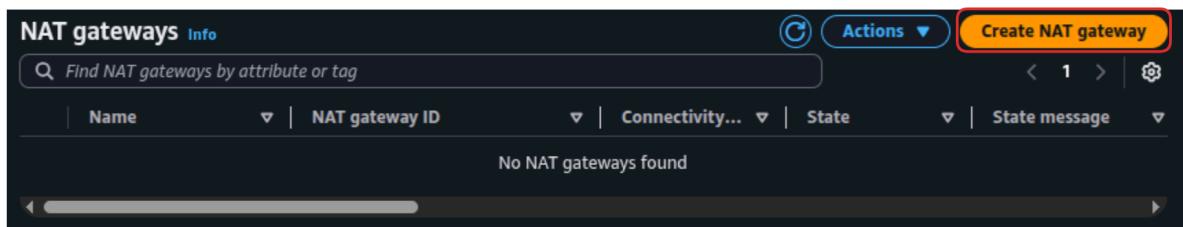
Finally, to allow private instances to connect to the internet for updates (only outbound connection) we use NAT Gateway.

We'll need a Elastic IP, so will create one in VPC “Elastic IP addresses”.



Public IPv4 address : 54.161.211.145			
Clear filters			
Name	Allocated IPv4 addr...	Type	Allocation ID
<input checked="" type="checkbox"/> -	54.161.211.145	Public IP	eipalloc-0f7d94e72bb4b8af

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).

Create NAT gateway [Info](#)

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the Internet.

NAT gateway settings

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

The name can be up to 256 characters long.

Availability mode [Info](#)
Choose whether to deploy across all zones in the region or restrict to a single availability zone.

Regional - new
Scales automatically across all regional AZs, simplifying management for multi AZ deployments.

Zonal
Provides granular control within a specific availability zone, adhering to subnet level settings.

Subnet
Select a subnet in which to create the NAT gateway.

Connectivity type
Select a connectivity type for the NAT gateway.
 Public
 Private

Elastic IP allocation ID [Info](#)
Assign an Elastic IP address to the NAT gateway.
 [Allocate Elastic IP](#)

Additional settings [Info](#)

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.

The screenshot shows the 'Edit routes' interface with two entries:

- Route 1**: Destination 10.0.0.0/16, Target local, Status Active.
- Route 2**: Destination 0.0.0.0/0, Target NAT Gateway, Status -.

Buttons at the bottom include 'Add route', 'Remove', 'Cancel', 'Preview', and 'Save changes'.

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

The screenshot shows the details of the route table 'rtb-07c8174859683ee6d / private-rt-jan26'.

Details (Info):

- Route table ID: rtb-07c8174859683ee6d
- VPC: vpc-04f7f22a1f5bf0875 | jan26-vpc
- Main: No
- Owner ID: 166337233399
- Explicit subnet associations: subnet-028b9f43b7e3dd81f / private-jan26
- Edge associations: -

Subnet associations tab is selected. It lists one association:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
private-jan26	subnet-028b9f43b7e3dd81f	10.0.2.0/24	-

Edit subnet associations button is present.

Subnets without explicit associations (0):

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
------	-----------	-----------	-----------

Edit subnet associations button is present.

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

Outbound rules (2)					
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol
<input type="checkbox"/>	-	sgr-010e1ee6e4a8f0851	IPv4	SSH	TCP
<input type="checkbox"/>	-	sgr-0e72ef804784eef4a	IPv4	HTTPS	TCP

Finally check if everything works running:

```
sudo yum update -y
```

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

```
[ec2-user@ip-10-0-1-236 ~]$ sudo yum update -y
Amazon Linux 2023 repository
Amazon Linux 2023 Kernel Livepatch repository
=====
WARNING: A newer release of "Amazon Linux" is available.
Available Versions:
Version 2023.10.20260202:
Run the following command to upgrade to 2023.10.20260202:
dnf upgrade --releasever=2023.10.20260202
Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.10.20260202.html
300 kB/s | 53 MB 02:59
238 kB/s | 31 kB 00:06
```

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