Create an EC2 instance with Amazon Linux

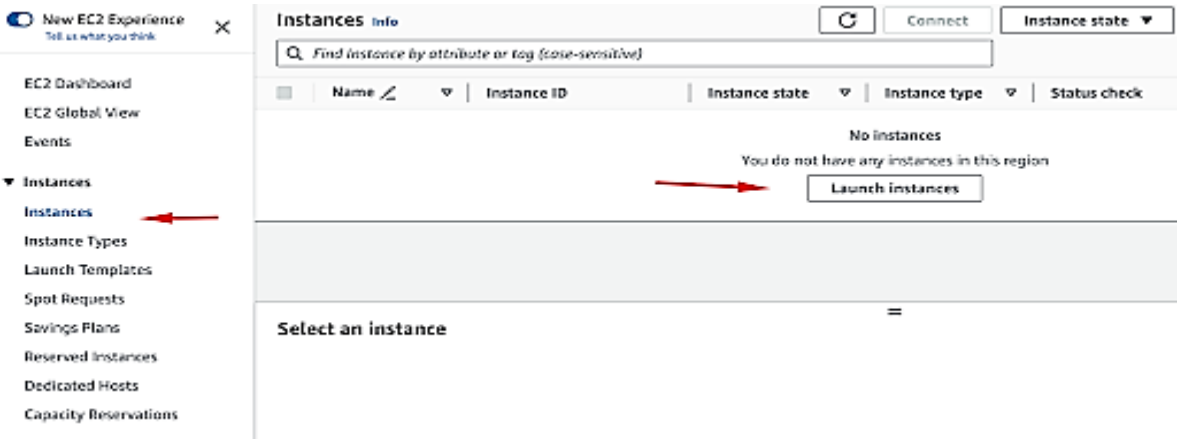
Step 1: Sign in to your AWS account

Before you can set up an EC2 instance with Amazon Linux, you need an AWS account.

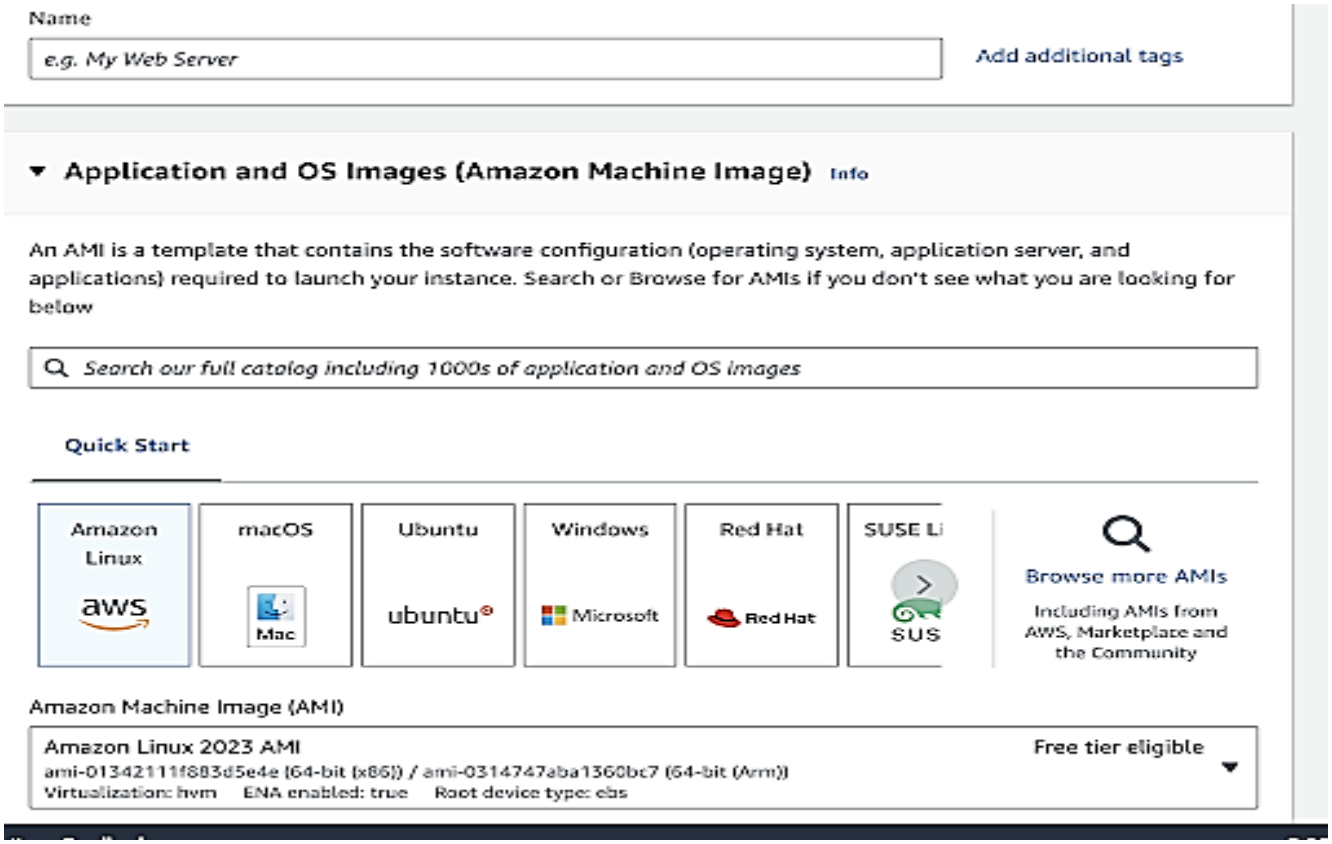
Step 2: Launch an EC2 instance

Once you have logged into your AWS account, in the search bar you can type EC2 in order to access the main page for provisioning and management.

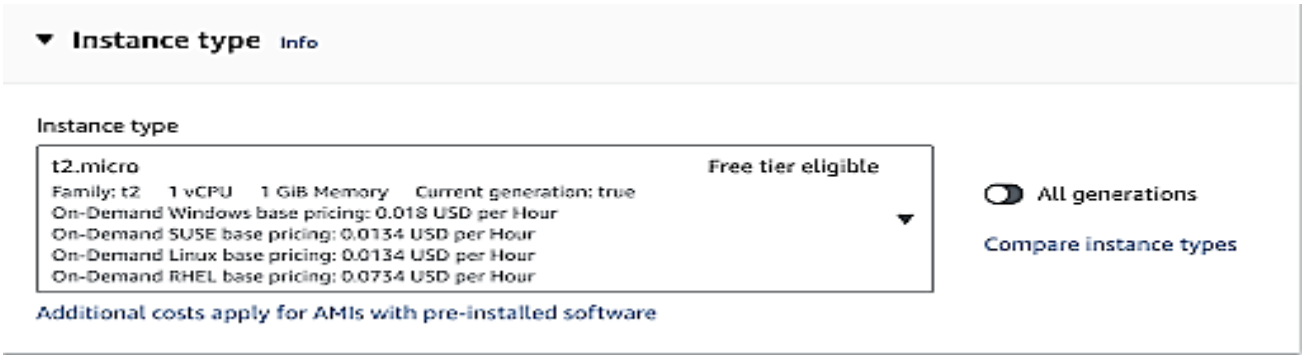
Next, under Instances, you can select the option “Launch Instances.”



We will land on the new page where we can select additional options, such as the Name of our instance, Application, and OS Images for our instance and we will select the default option — Amazon Linux.

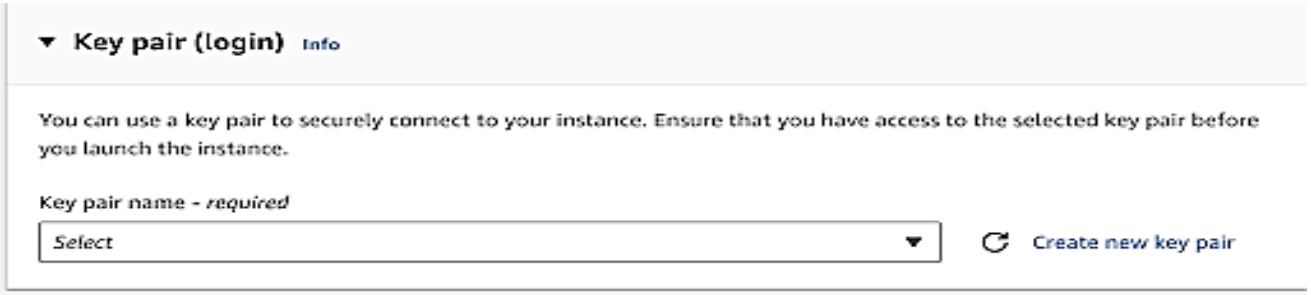


When we scroll below we can choose the type of our instance. Amazon Linux has a minimum requirement of 512MB of RAM and 1 CPU core, so if we are testing things, we can select the t2.micro instance which fits into the free tier in AWS.

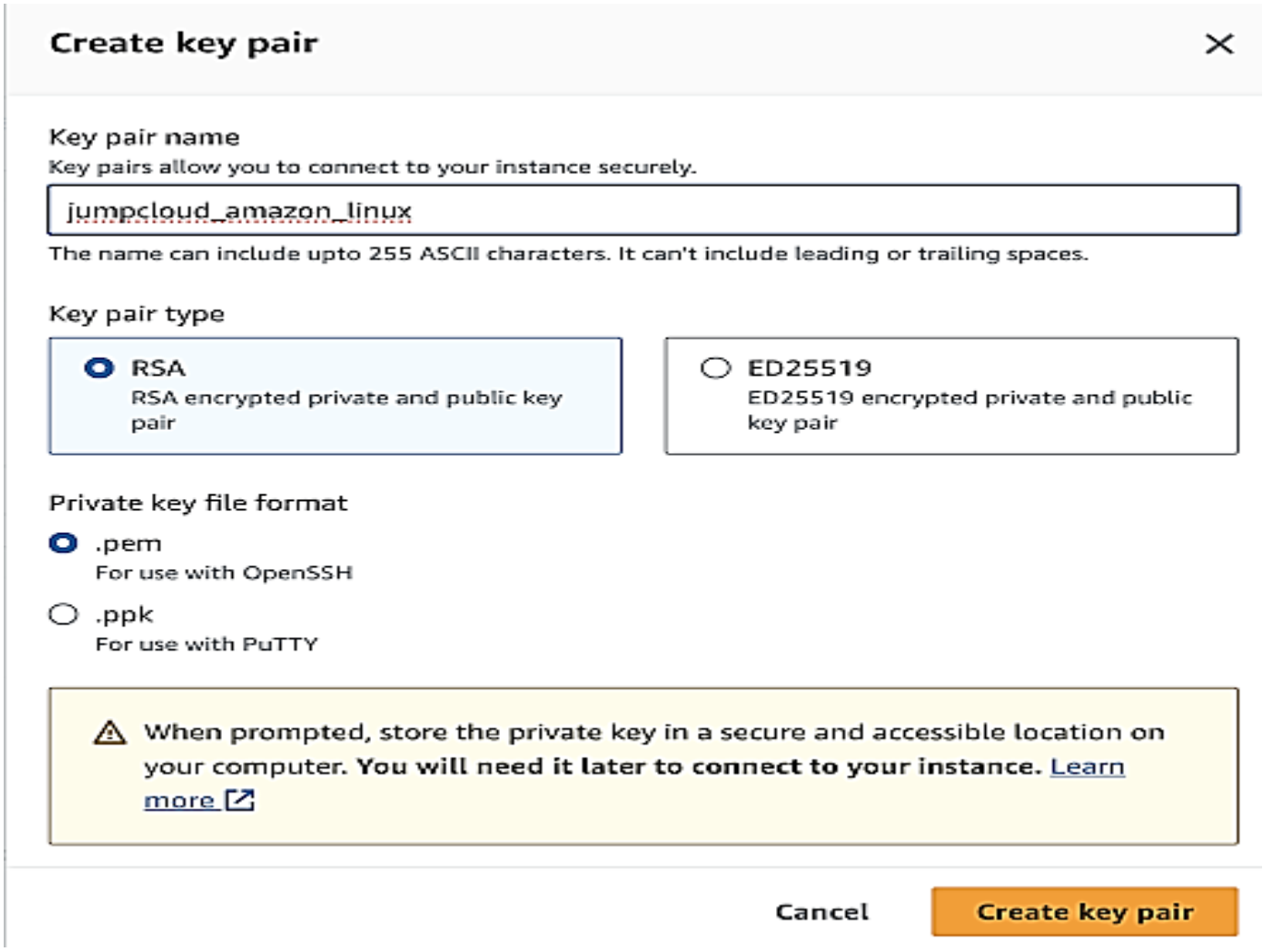


When you have a project and instance that should be production-ready, you can select instances with more RAM and CPU power, by choosing from the dropdown list.

Next, we will make sure to create a new security group for our instance.



Creating a new security group for your new EC2 instance is a fundamental security best practice in AWS. It allows you to define and enforce customized network access controls, adhere to the **least privilege principle**, and maintain better isolation and security for your EC2 instances.

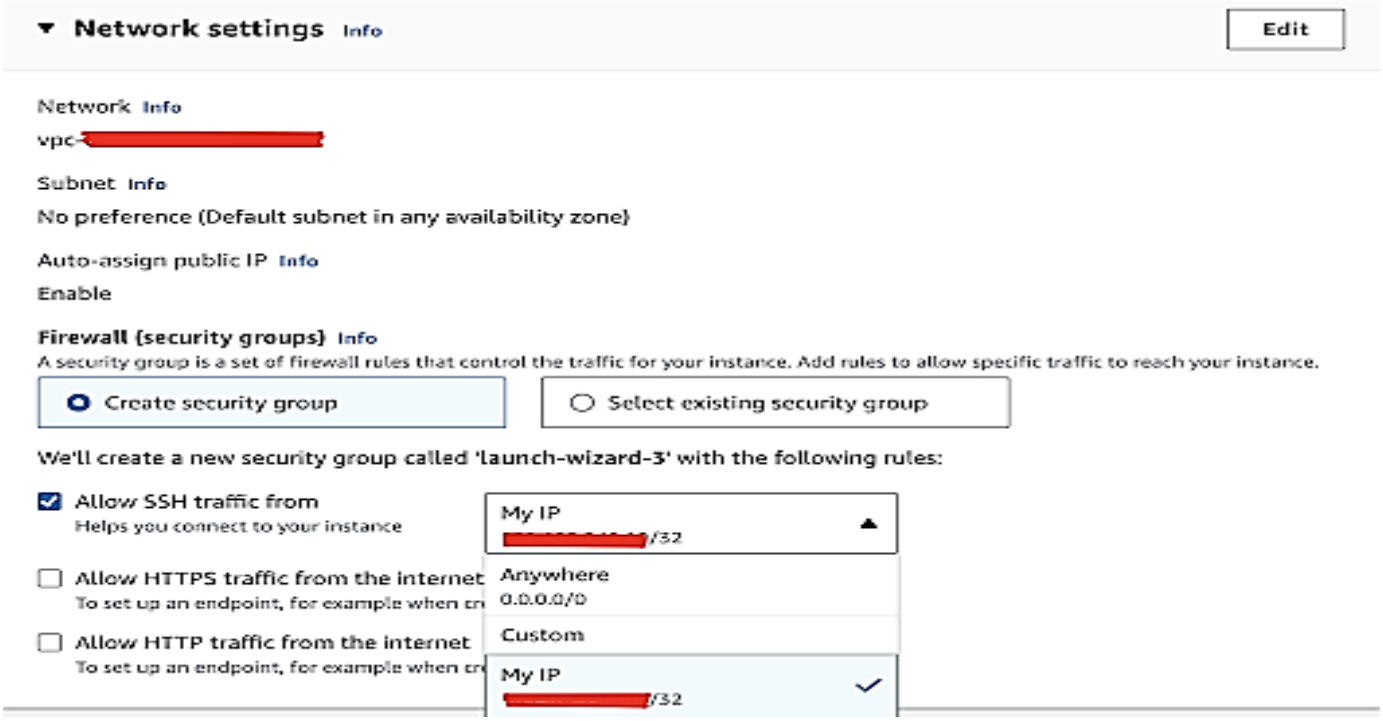


We can select Private key file format which can be either in .pem format or .ppk in case you use a Windows SSH client such as PuTTy.

In our case, we will select the .pem format since we will be using the terminal.

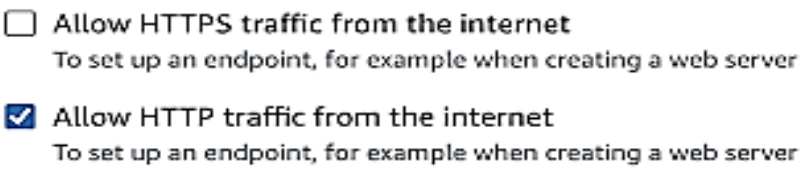
Finally, click on the “Create key pair” option in order to generate the SSH key and this will prompt the download in your browser. Also, there is a warning from AWS that we should store our private keys in a secure and accessible location on our computers.

Further in our setup process, we would need to select our Network Settings.

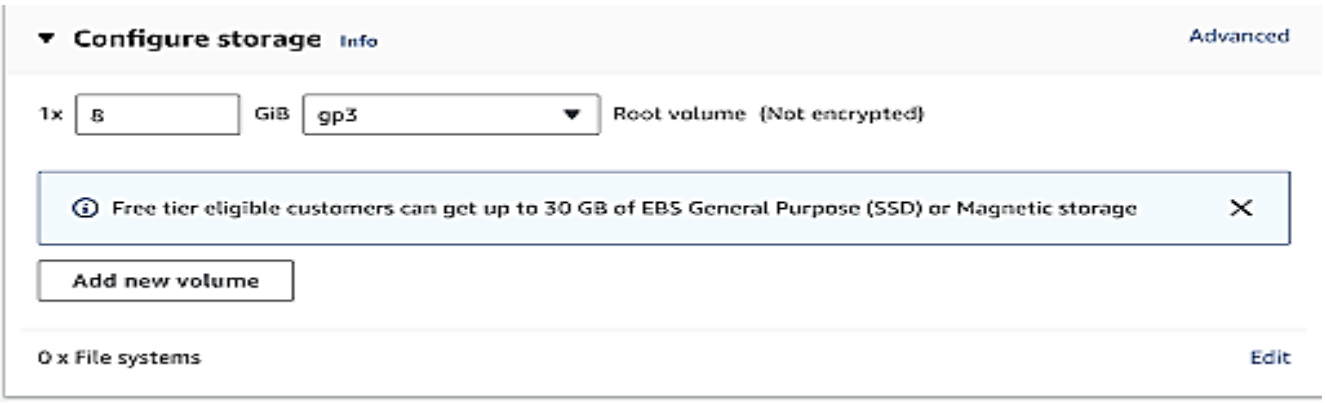


For the first option, we need to access our EC2 instance through SSH but instead of allowing SSH traffic from “0.0.0.0/0” (which means from anywhere), restrict it to only allow connections from specific trusted IP addresses or ranges. This limits the exposure of your SSH port to the internet and reduces the risk of unauthorized access. You can also use CIDR notation to specify the IP ranges (e.g., 10.100.0.0/16,).

Since we would like to install the nginx web server as a proof of concept later, we will check the “Allow HTTP traffic from the internet” option.

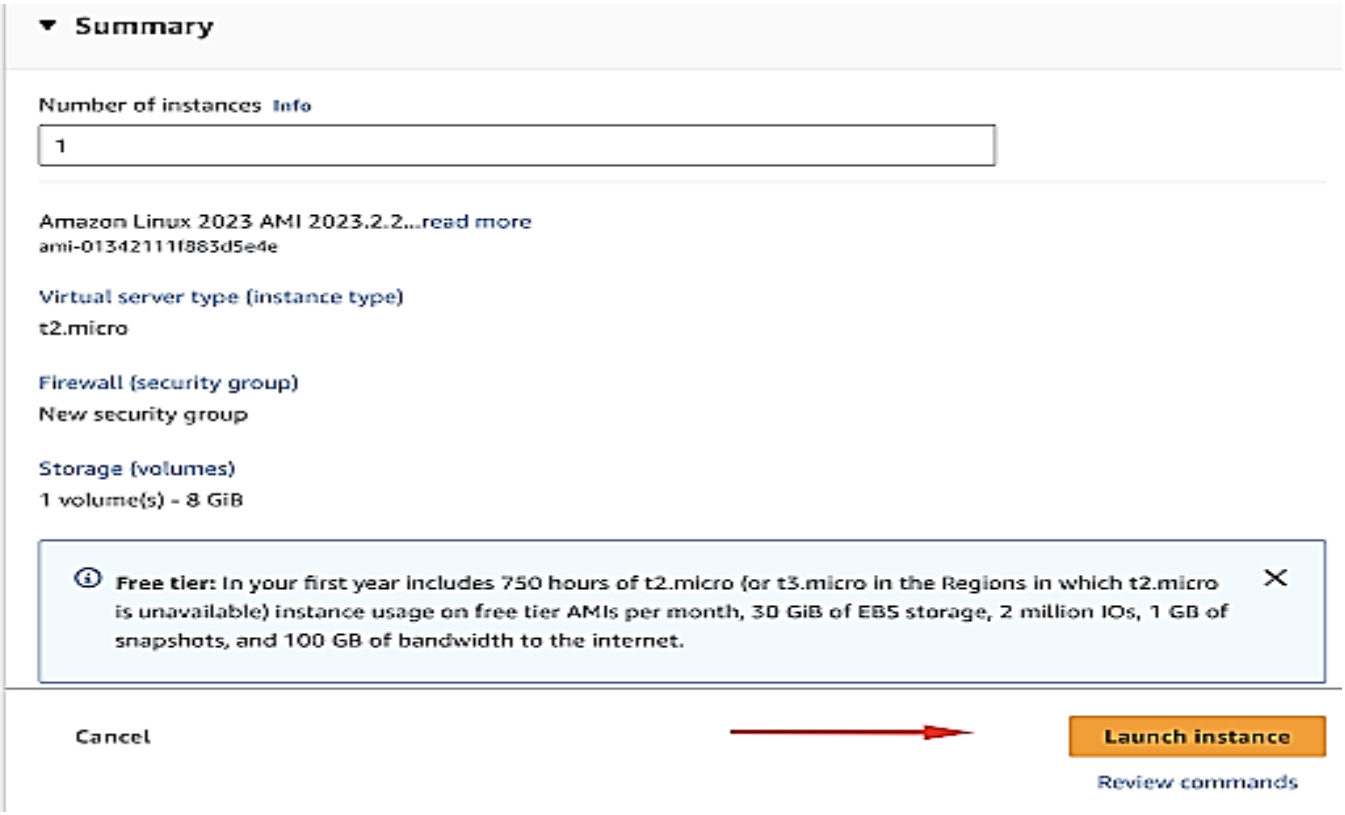


Next, we will select our storage options from the menu:

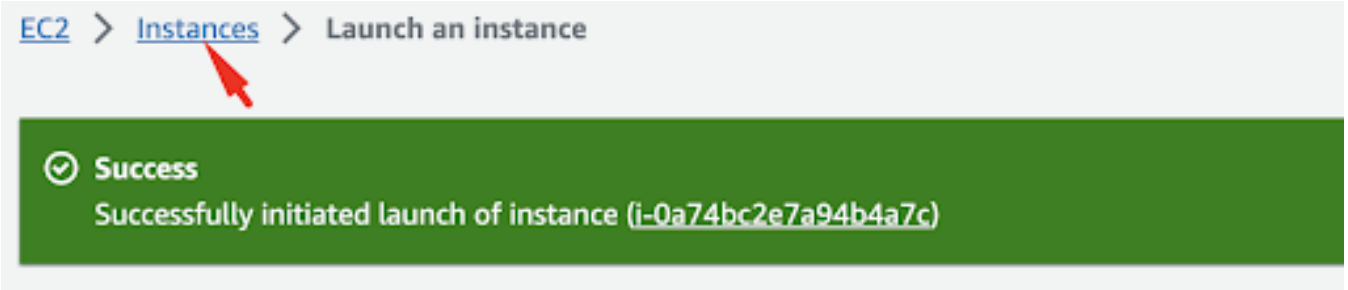


In this case, we have selected a new volume with 8 GB of storage with the general purpose, gp3 type of volume.

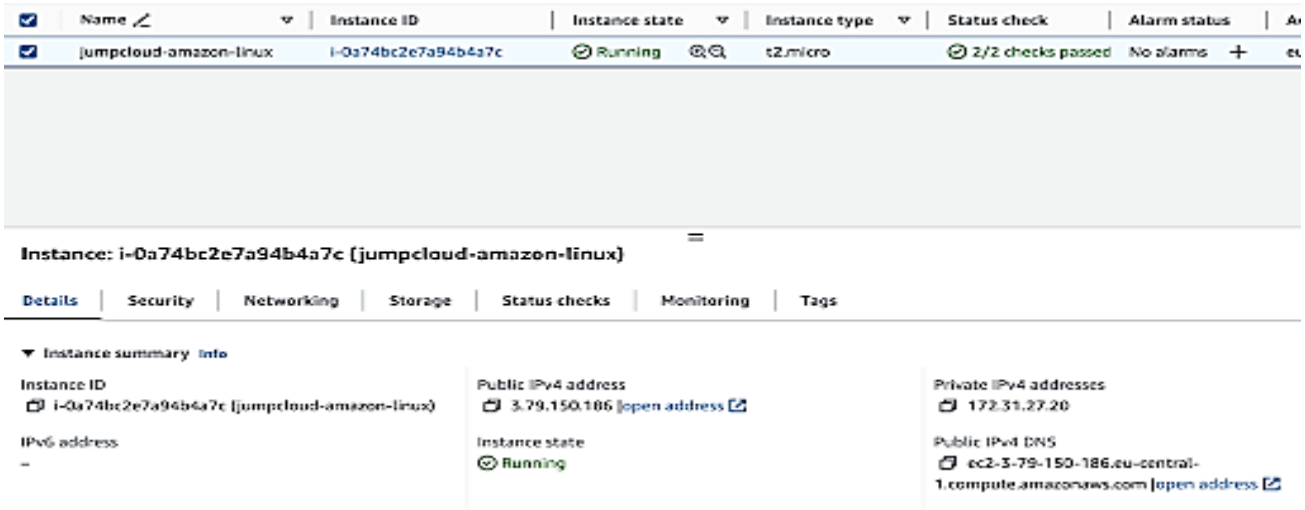
Our final step is confirming the selected options in the summary and we can proceed to launch the instance. Encrypt EBS snapshots and volumes.



You will get the notification that the instance is successfully launched and you'll return to your Instances list.

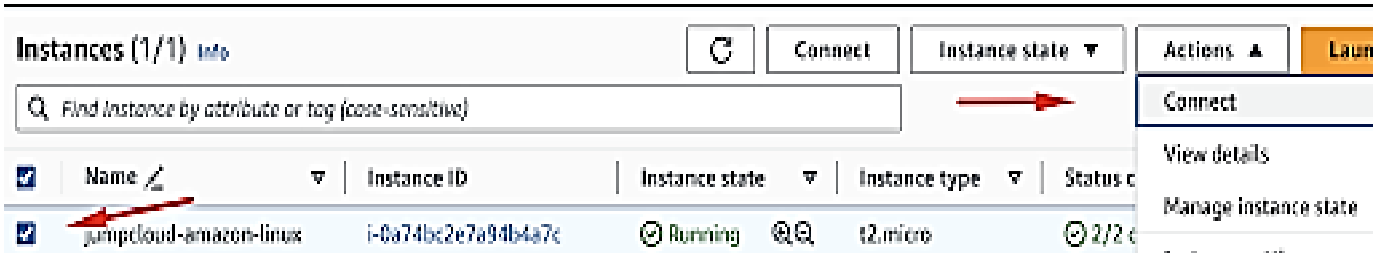


After a short wait, typically with a fast provisioning time, you'll be able to locate your instance in the list.

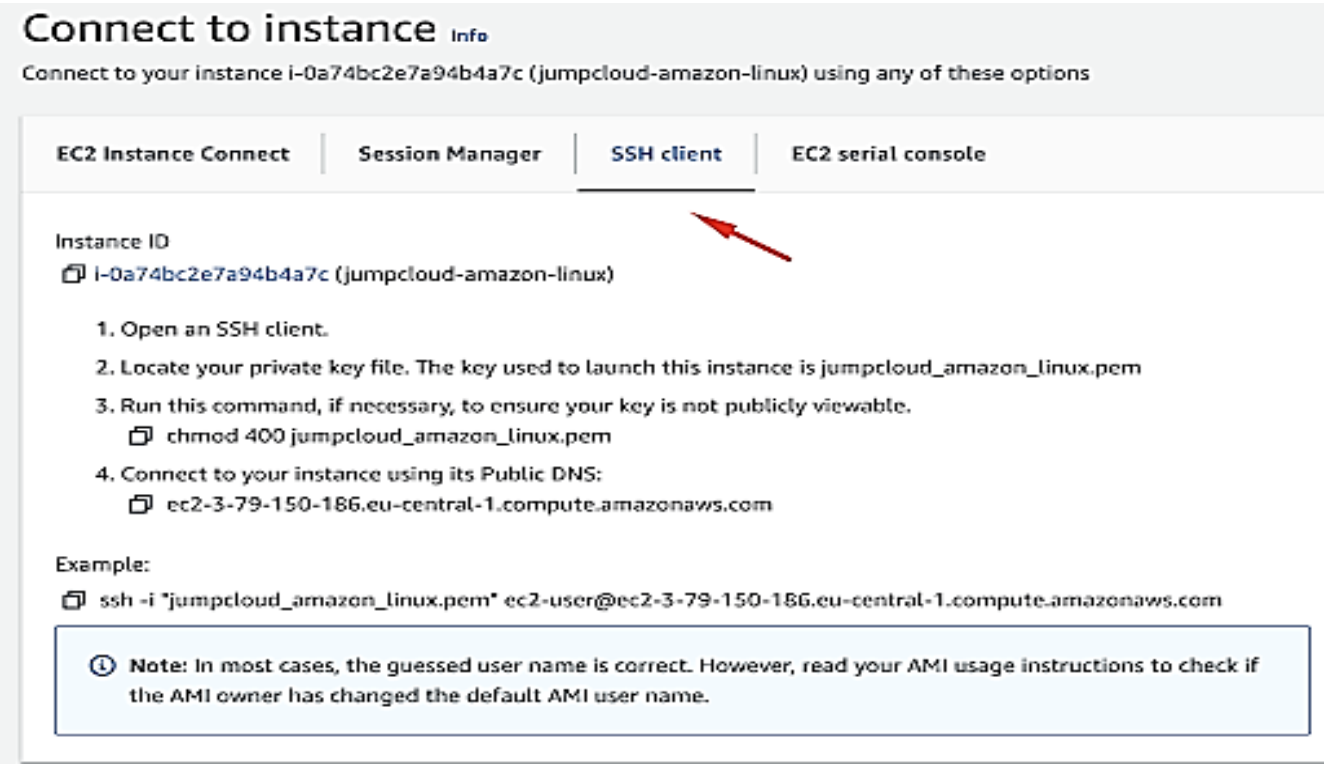


Step 3: Connect to your Amazon Linux instance

Now the next step is to connect to the instance via SSH. We can do so by using our Instances menu and clicking Connect:



Here we will see a new menu where we will select the SSH client menu item:



We will now follow the instructions provided by AWS in order to connect to our instance.

After opening your terminal and finding where you have downloaded the .pem key, it is important to change permissions to it, for security reasons, so that the key is not publicly viewable, and in our case, we will run the following command:

```
chmod 400 jumpcloud_amazon_limux.pem
```

Next, we will use the SSH key to connect to our instance:

```
ssh -i “jumpcloud_amazon_linux.pem” ec2-user@ec2-3-79-150-186.eu-central-1.compute.amazonaws.com
```

Here, the -i flag in SSH is used to specify the path to the private key file to be used for authentication when connecting to a remote server. It allows you to choose a specific key file when you have multiple key pairs or non-standard key file names and locations.

When we log into our Amazon Linux instance for the first time, we need to confirm the authenticity of the host. Here you can type yes and press Enter.

```
The authenticity of host 'ec2-3-79-150-186.eu-central-1.compute.amazonaws.com (3
.79.150.186)' can't be established.
ED25519 key fingerprint is SHA256:7+l4e5rWSJ3DfH2UOJ5mDdG7LOK23hVked2L/IlTa48.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

After this action, we will be logged into our instance:

```
2-3-79-150-186.eu-central-1.compute.amazonaws.com
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-27-20 ~]$
```

You can always verify the version of your Amazon Linux by typing:

cat /etc/os-release

```
[ec2-user@ip-172-31-27-20 ~]$ cat /etc/os-release
NAME="Amazon Linux"
VERSION="2023"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2023"
PLATFORM_ID="platform:al2023"
PRETTY_NAME="Amazon Linux 2023"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2023"
HOME_URL="https://aws.amazon.com/linux/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2023"
SUPPORT_END="2028-03-01"
```

We can list updates to our system by running the following command:

sudo dnf update

This command requires higher system privileges so make sure you use the sudo.

Often the repositories are up to date once the instance has been provisioned, but it is always a good idea to check, mainly for security and compatibility reasons.

```
[ec2-user@ip-172-31-27-20 ~]$ sudo dnf update
Last metadata expiration check: 0:57:44 ago on Sat Sep 30 14:09:52 2023.
Dependencies resolved.
Nothing to do.
Complete!
```