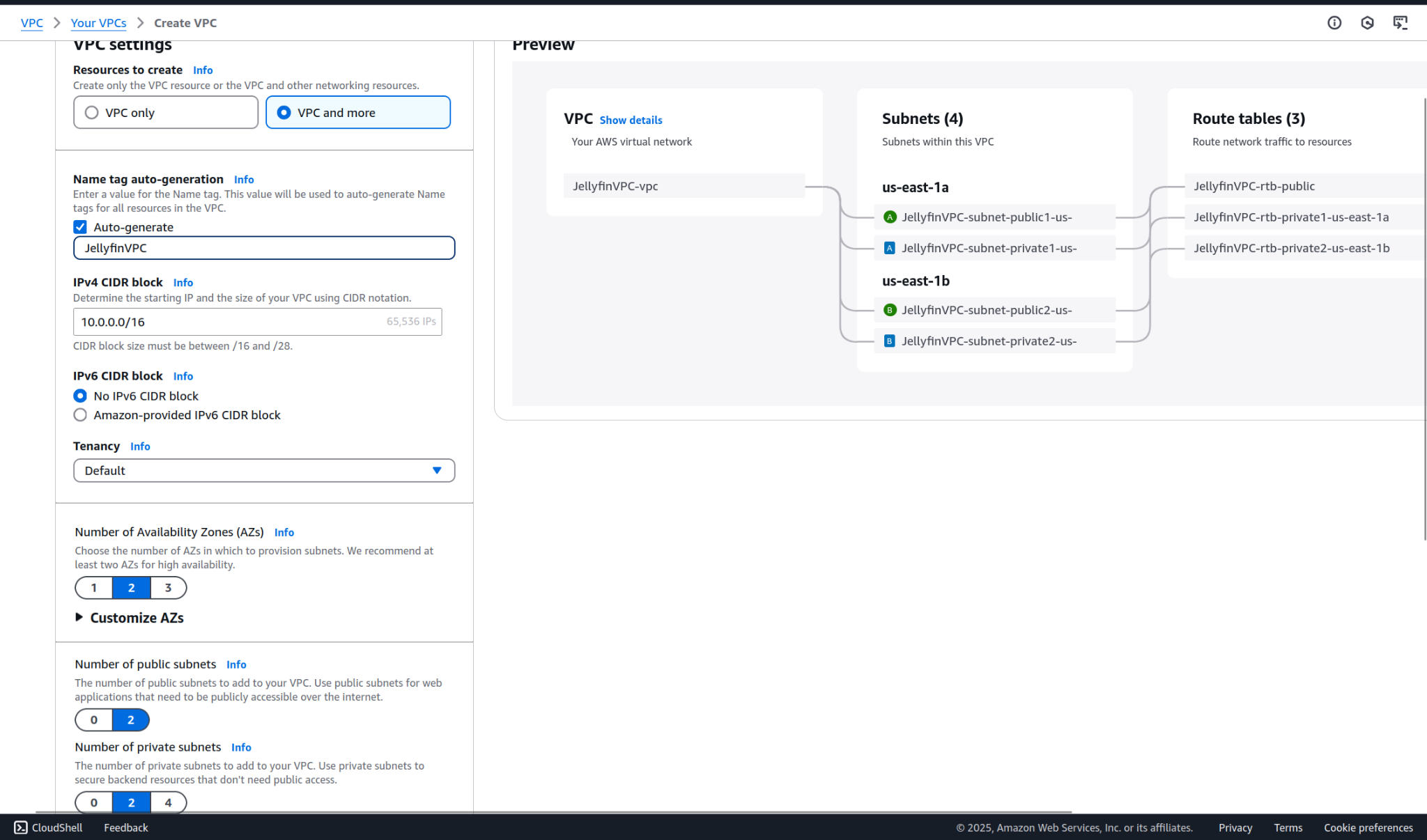
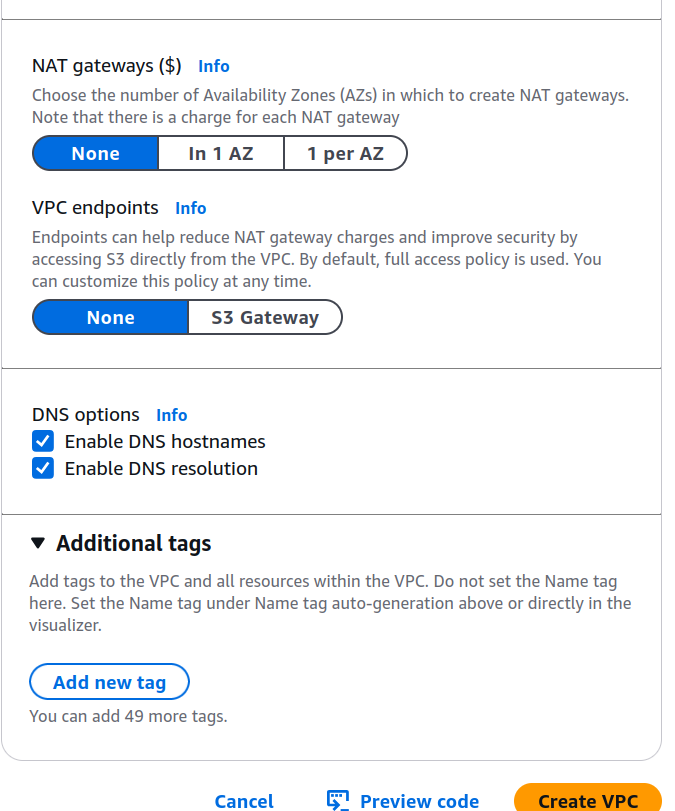
## Create VPC

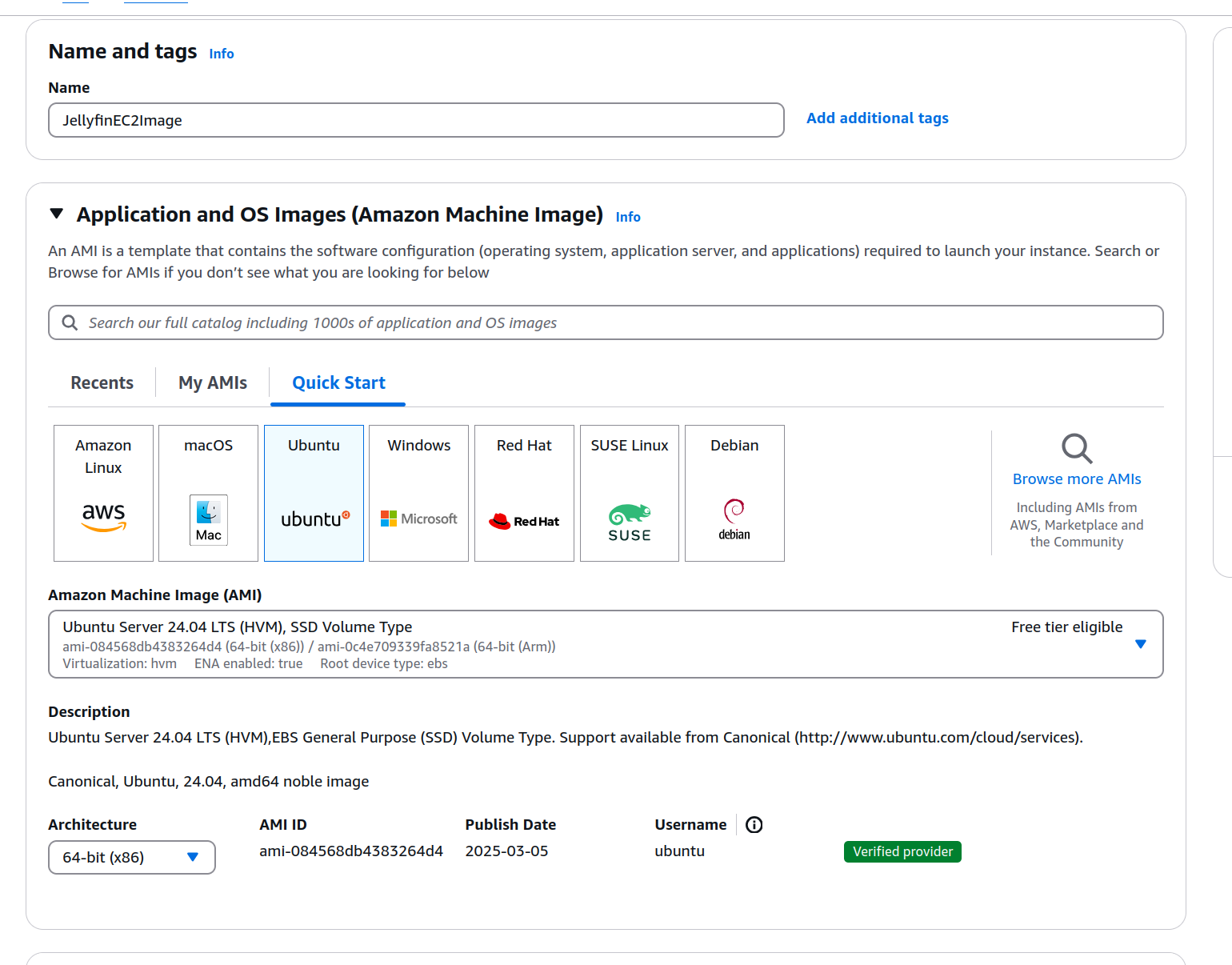
Create a VPC with 2 public subnets and a private subnet. The reason we are creating two public subnets is for the ALB for later that requires two availability zones (AZ).

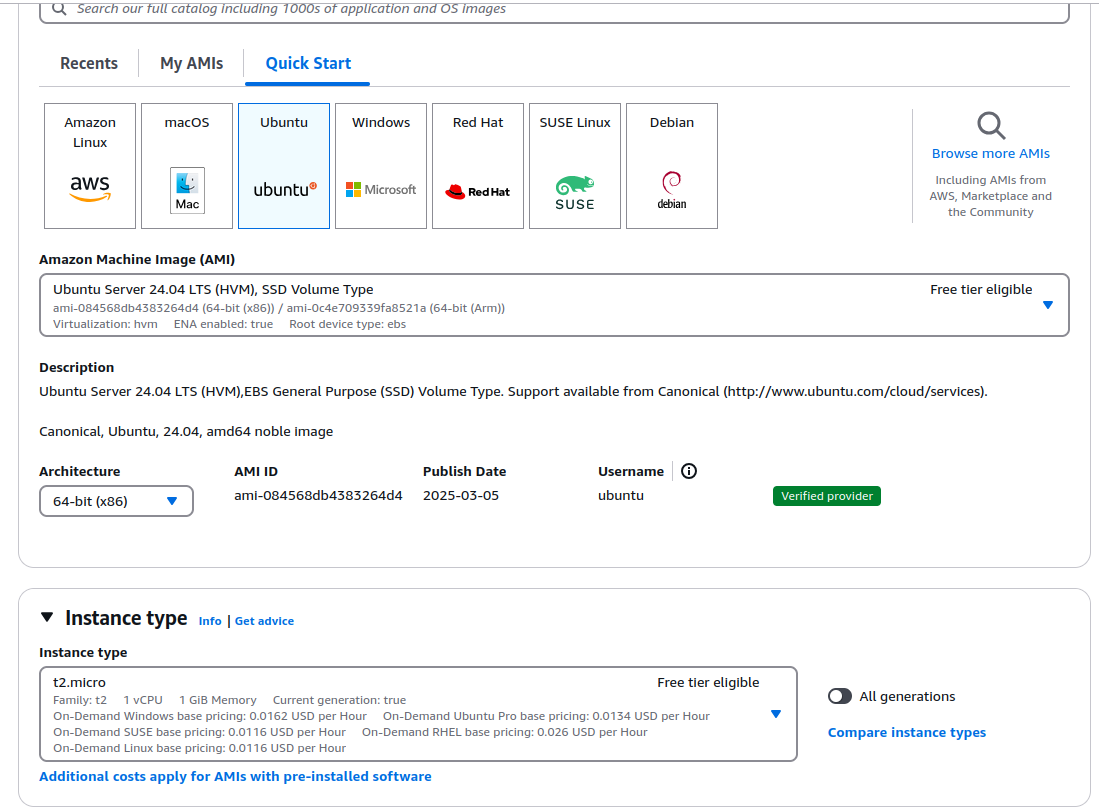




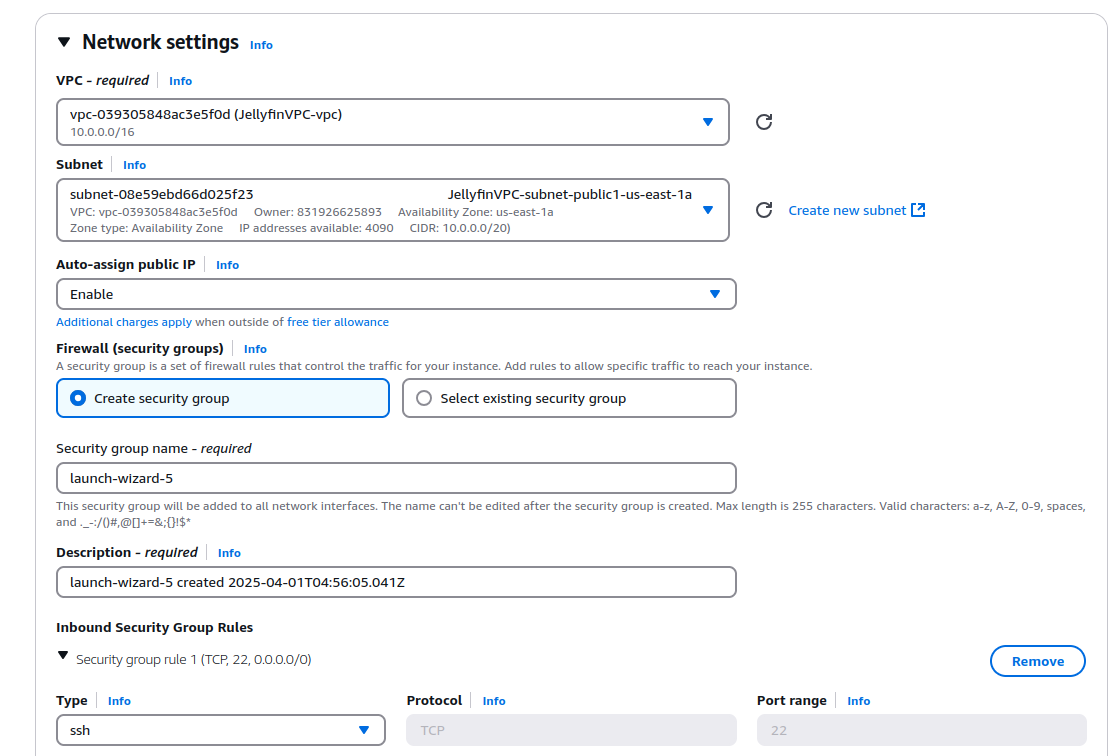
## Create EC2 Instance

We are naming it EC2Image because we will use this to make images (copies) later.

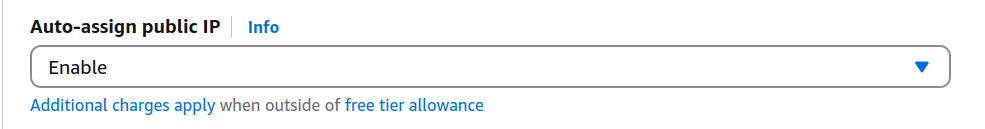




Configure the network to use the VPC we just made. Do not enable http access yet.

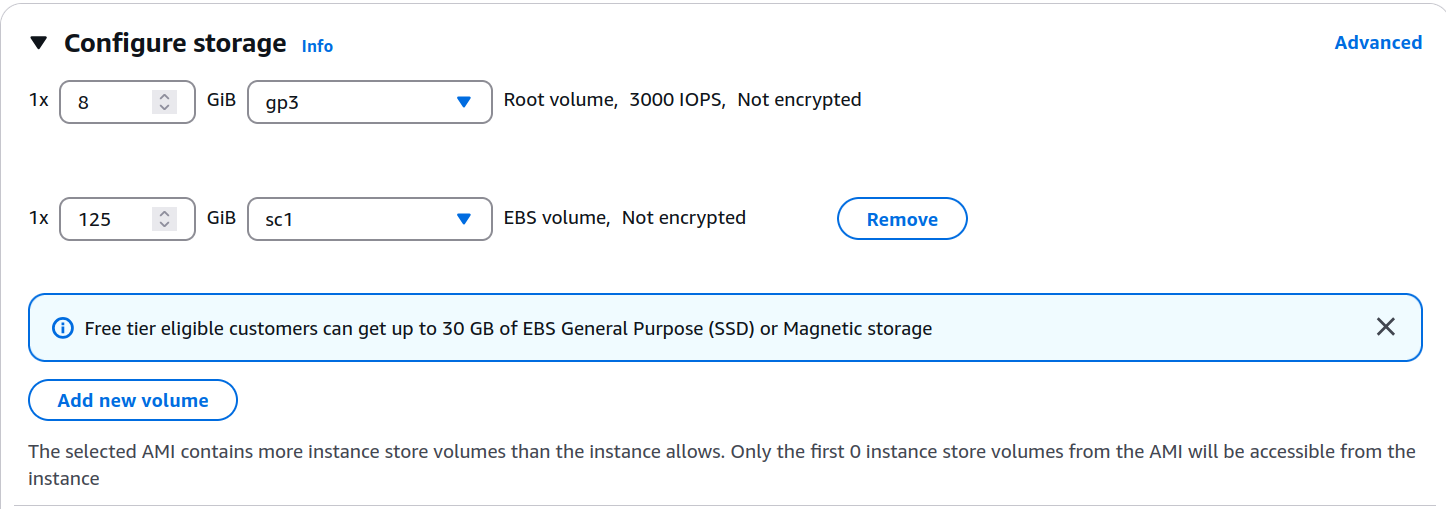


We’re going to need a public ip to connect to the EC2 for now. So enable this.



### Configuring EBS Storage

EBS storage is a storage device that can mount to EC2 for high throughput operations, such as video streaming. We’re using SC1 for cheaper(but still fast enough) usage.



## Setting up the EC2 Server

Connect to the EC2 instance using the browser connect.

Now let’s download jellyfin and install it.

| curl https://repo.jellyfin.org/install-debuntu.sh | sudo bash |
| --- |

Let it run through the process

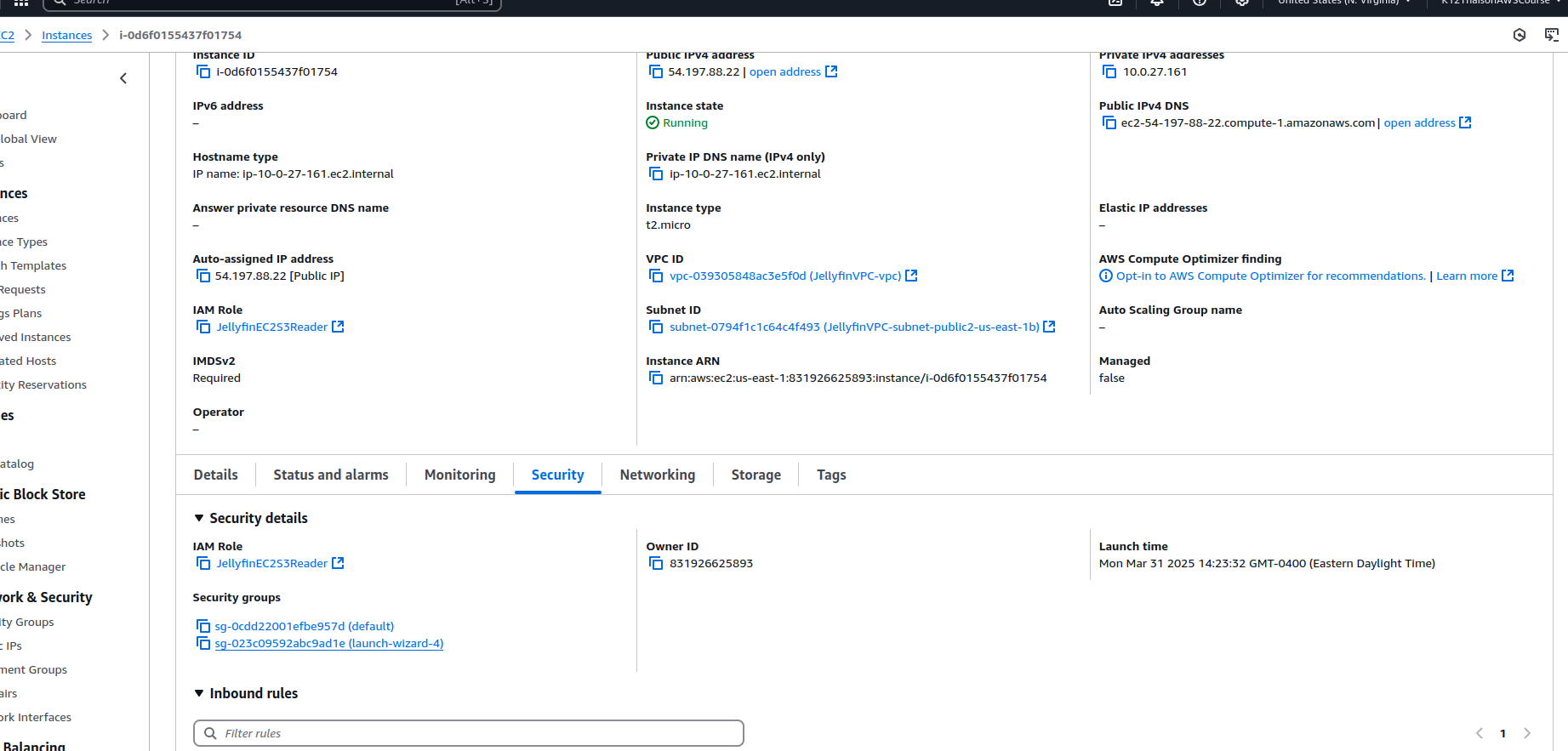
## Attach the EBS

| sudo mkfs -t ext4 /dev/xvdb sudo mkdir /content sudo mount /dev/xvdb /content sudo mkdir /content/Movies sudo chown -R jellyfin:jellyfin /content sudo chmod 755 /content echo "/dev/xvdb /content//Movies ext4 defaults,nofail 0 2" | sudo tee -a /etc/fstab |
| --- |

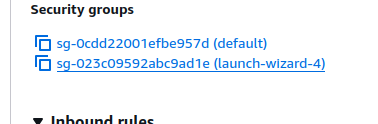
| sudo systemctl restart jellyfin |
| --- |

## Testing the EC2 server

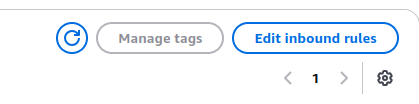
Let’s quickly test our server. Even though its public, we need to enable access to it in the security group



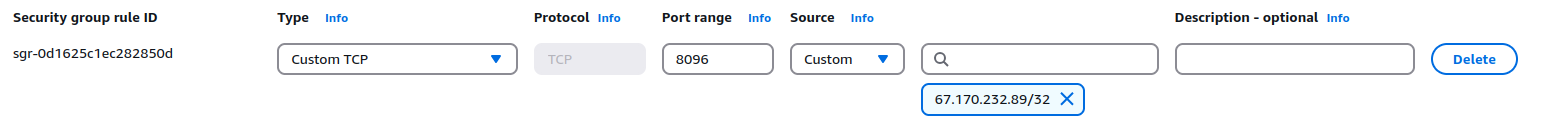
Click on launch wizard security group



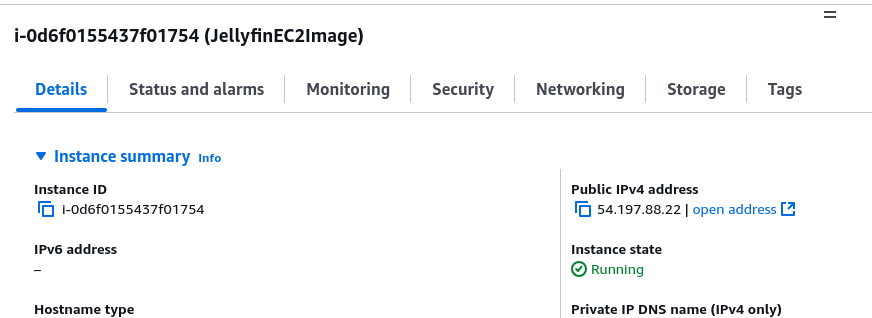
Edit the inbound rules



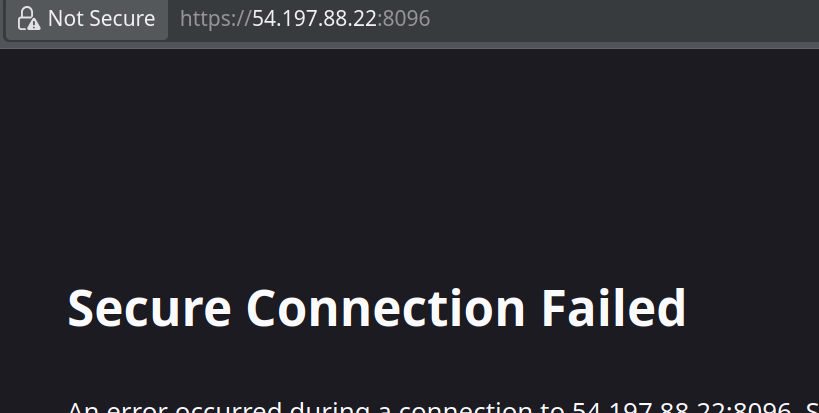
Add a TCP rule for 8096 for YOUR IP specifically.



Now connect to it with its public ip



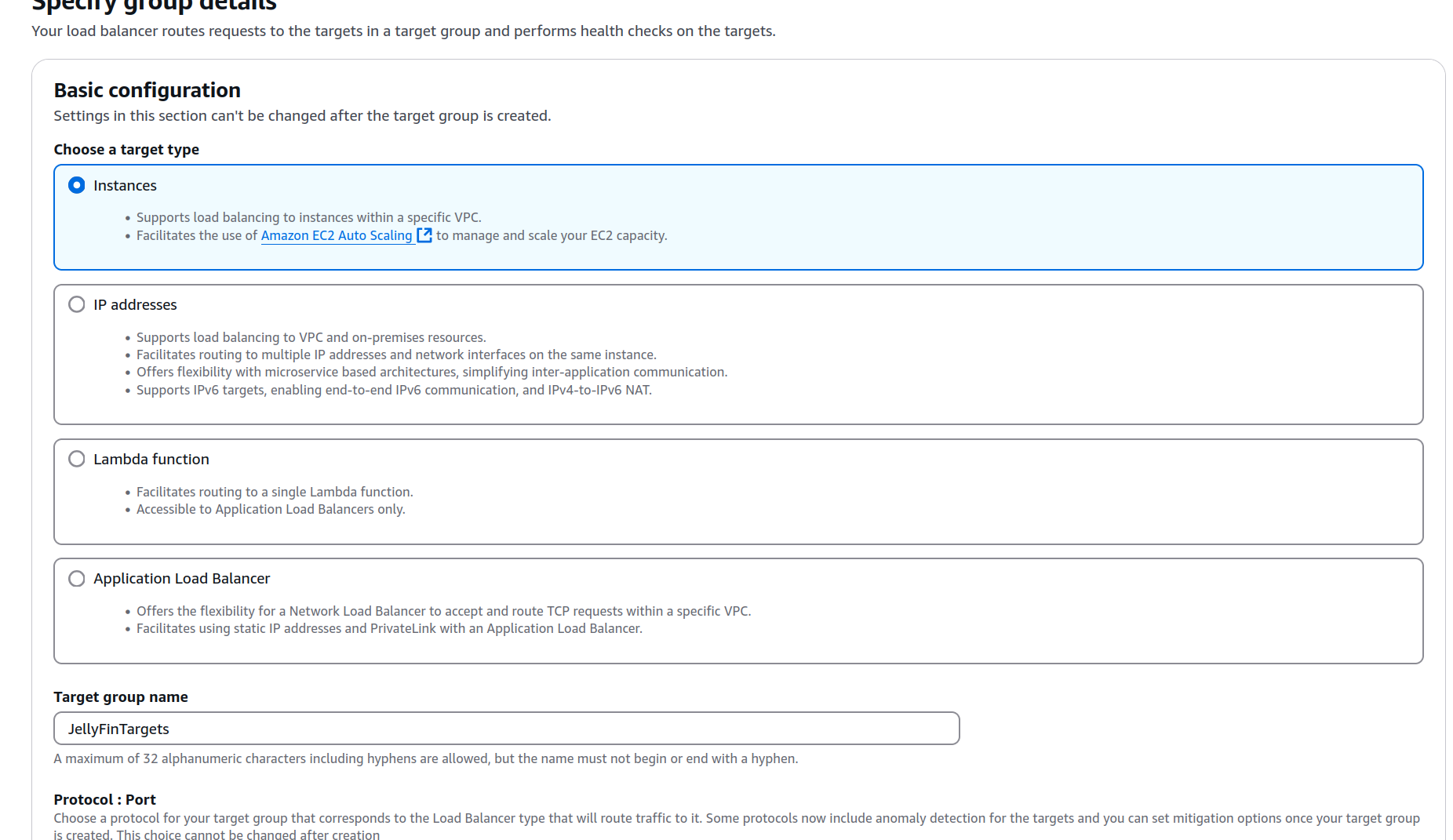
Will have to use http, so replace the https with http



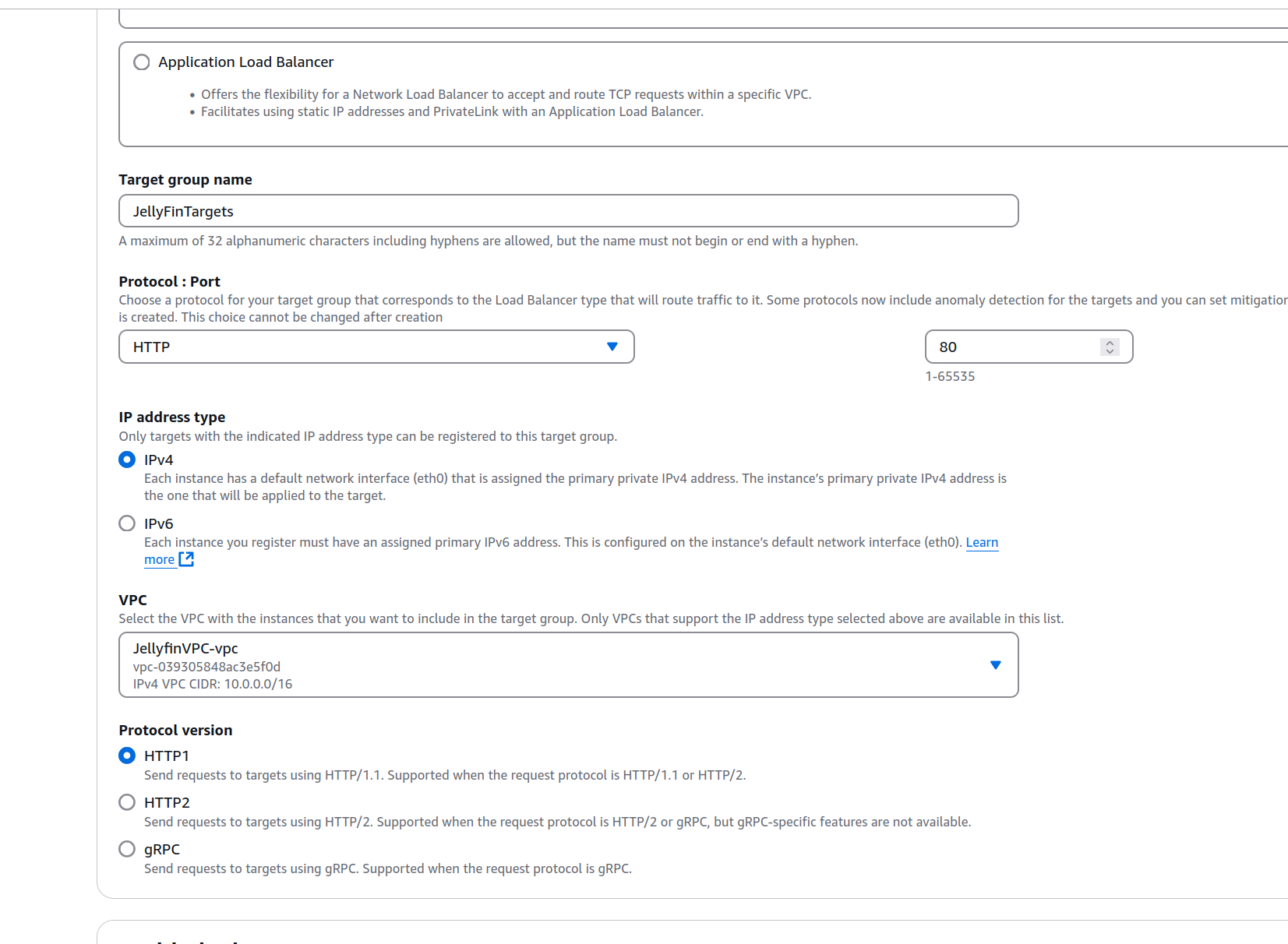
Do **NOT** go through the set up wizard yet. You are on http! It’s not safe.

## Setting up HTTPS -> Create a target group

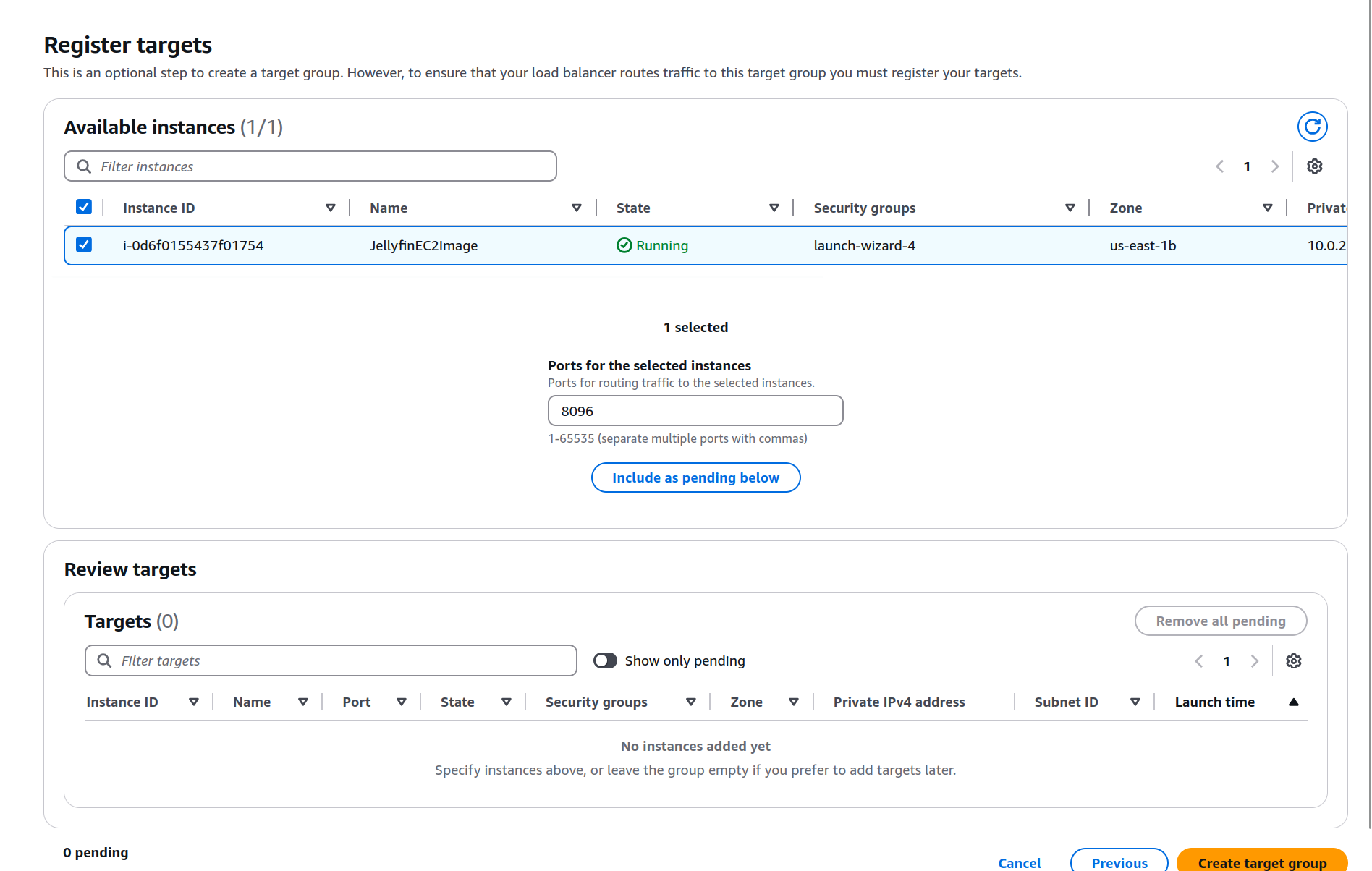
On the EC2 Homepage, go to target groups and create one for instances. Name it something fitting.



We’re going to be using http(not https) protocol to communicate within the VPC.

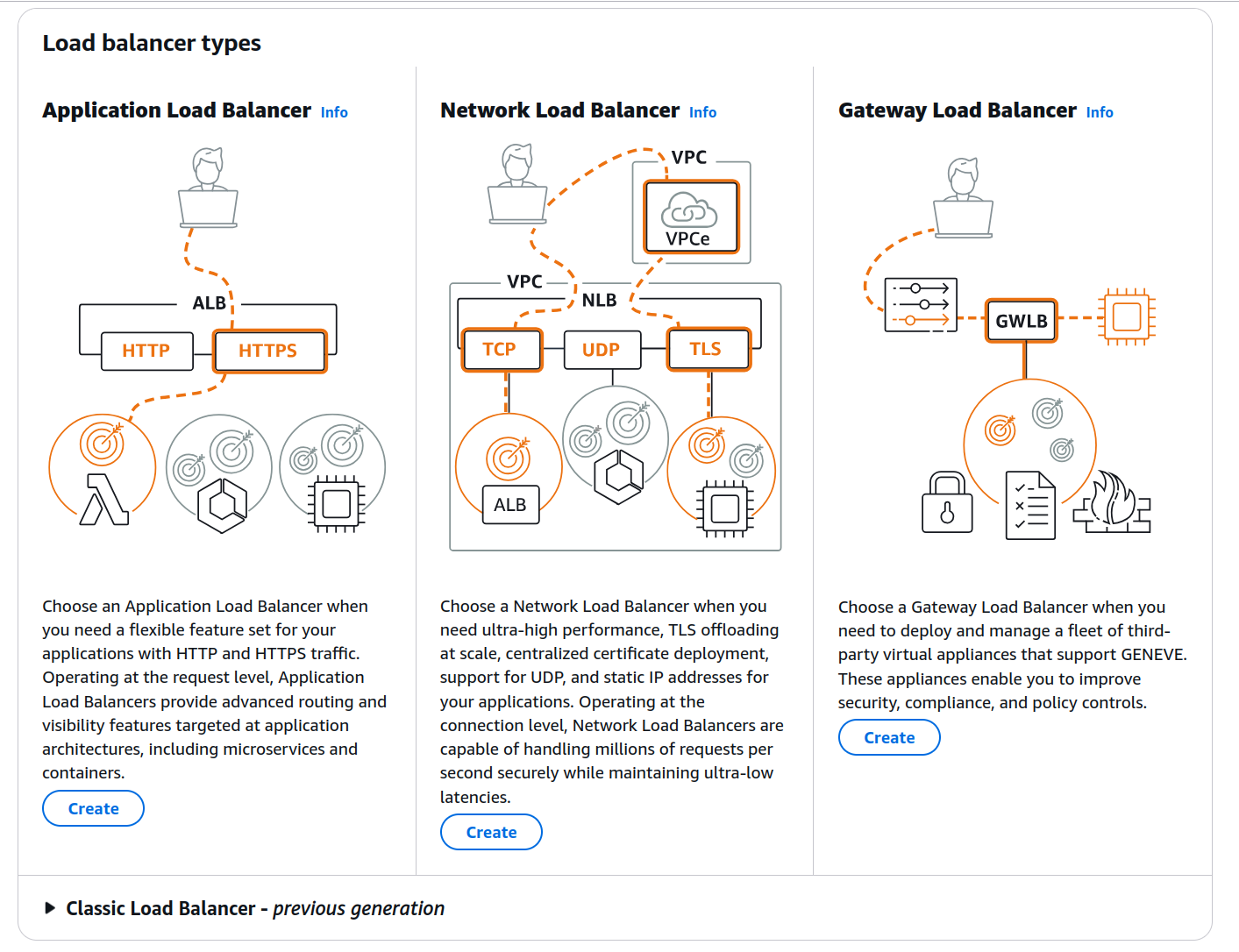


Now target the 8096 port on the Jellyfin EC2. This is the port jellyfin will start on. Press include as a pending below, then create the target group.

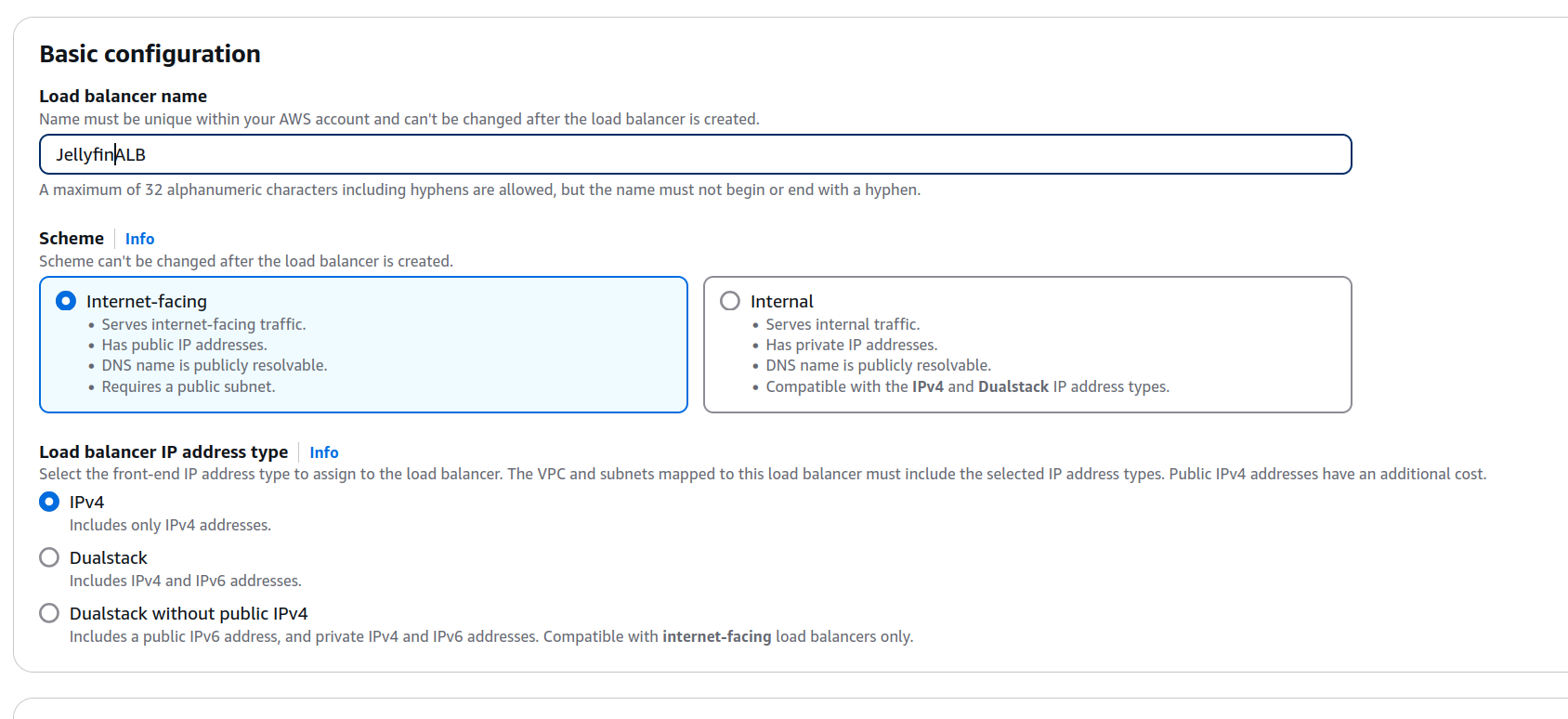


## Create Application Load Balancer

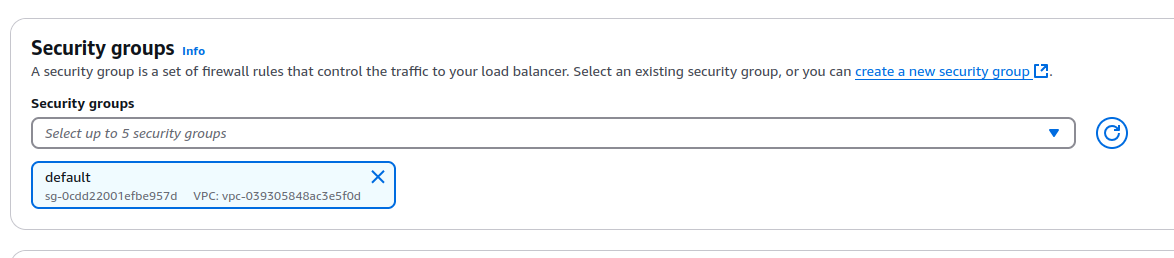
We’ll be using a public application load balancer has the public facing access for our EC2s



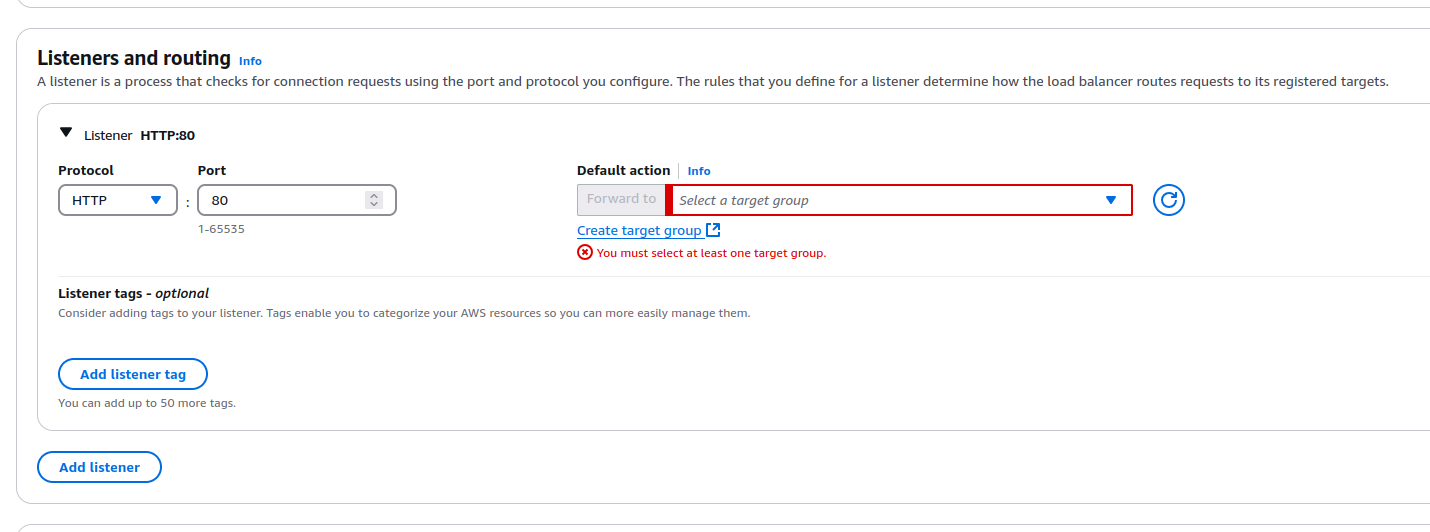
Create the ALB with these settings







Set the target as the previous target group we had earlier.

 Create the ALB

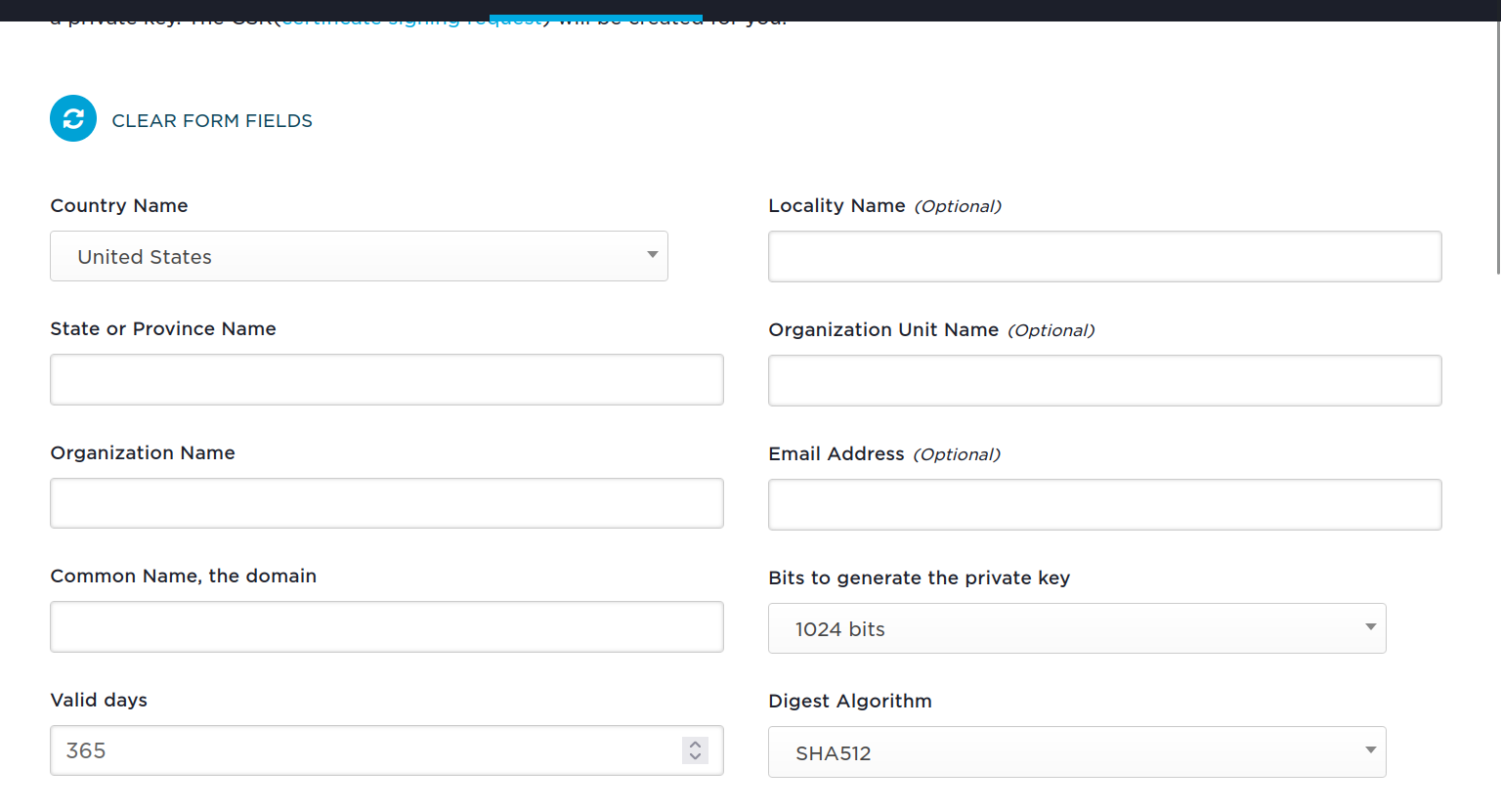
## Creating a self signed certificate

Create a self signed certificate here

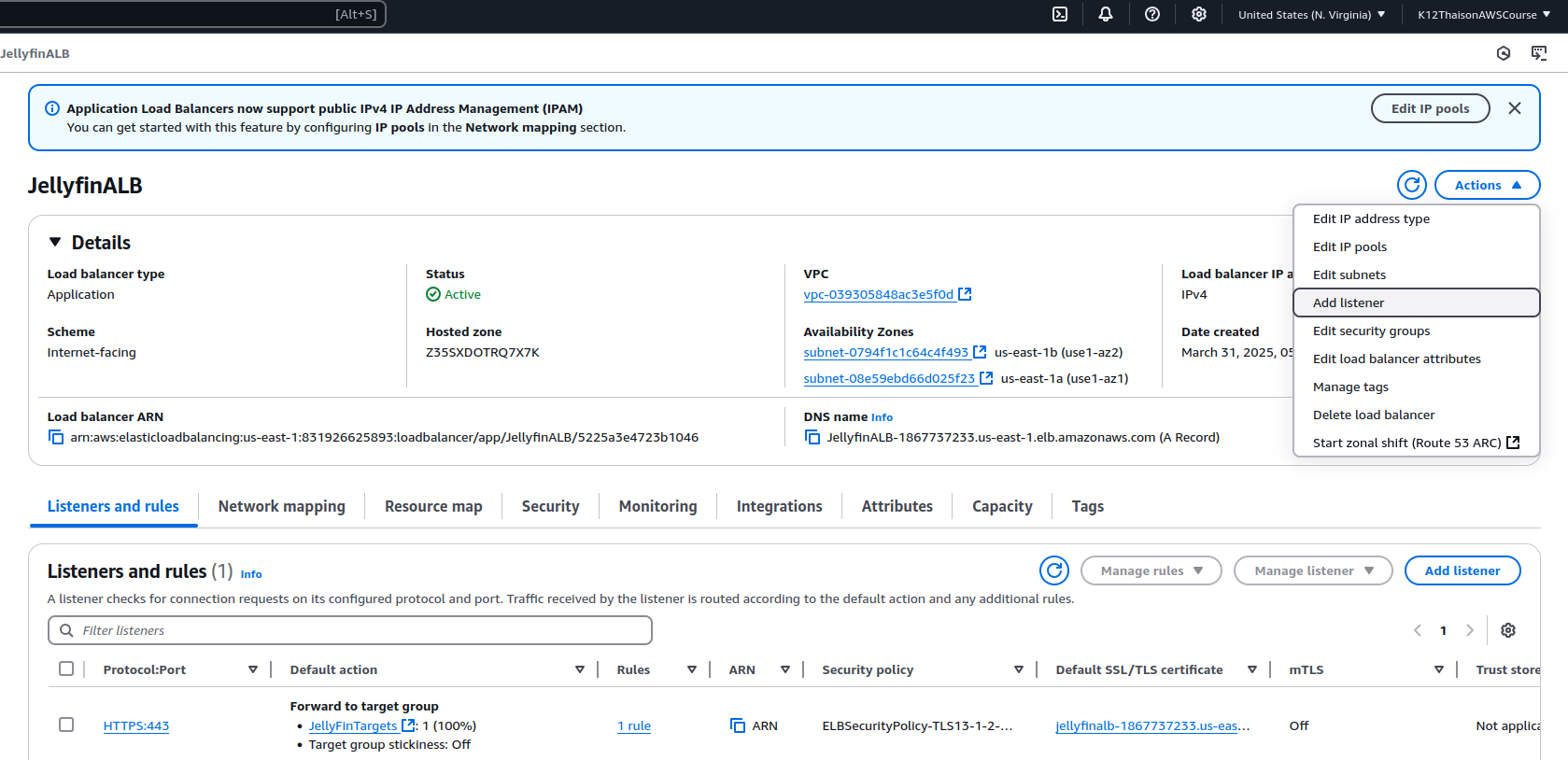
https://www.samltool.com/self\_signed\_certs.php

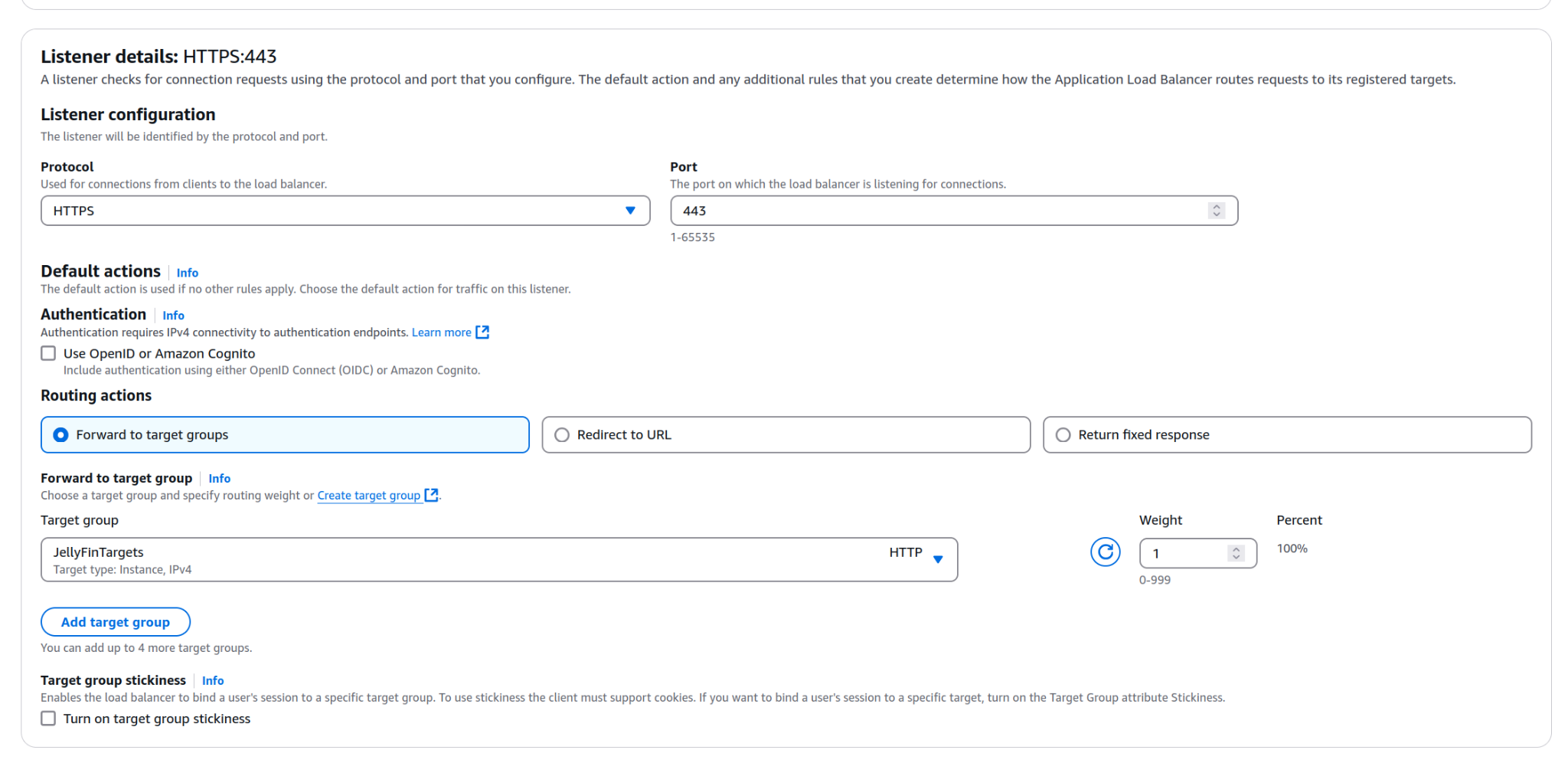
This will allow us to use https on our server.

Use the ALB DNS for the Common Name.



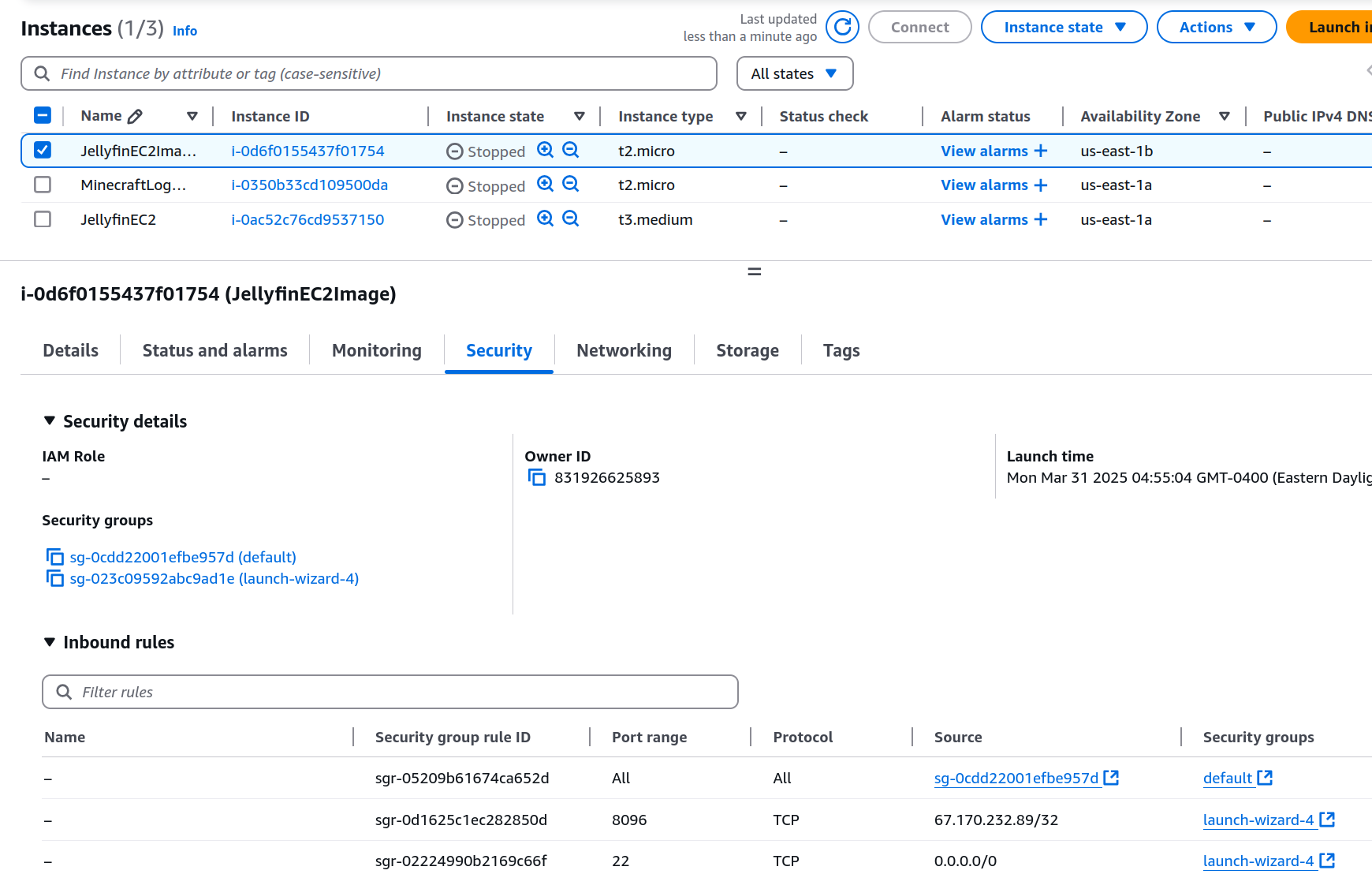
## Add an https listener to our ALB.



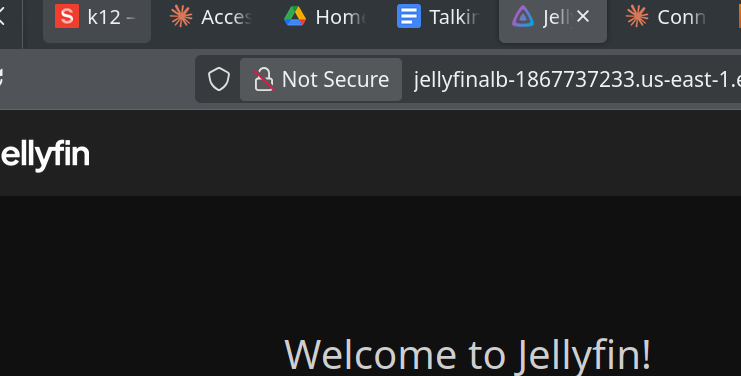




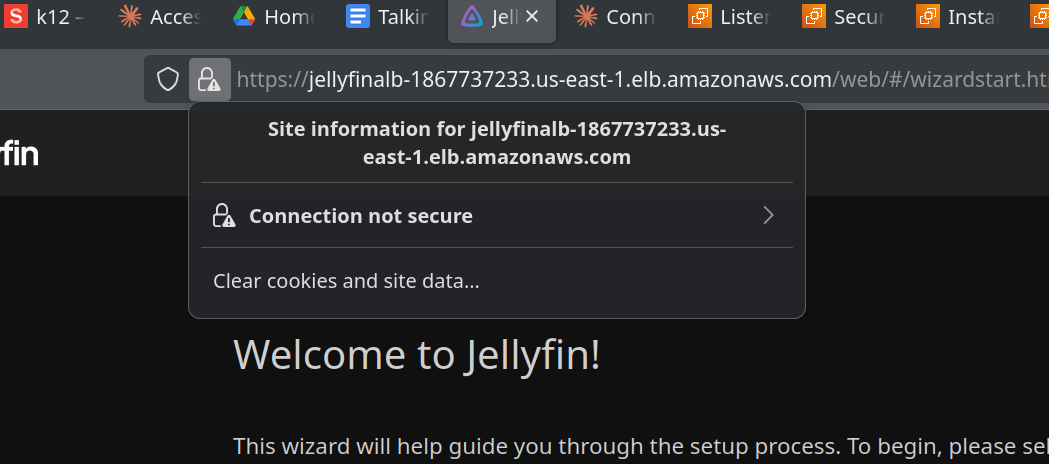
Add the default security group to the EC2, so that the ALB can send data to the EC2. This is because you can send data to other services in the same security group.



Now try connecting to the ALB DNS. Make sure we use https:// by putting that in front of the dns. If it shows not secure, it’s http.

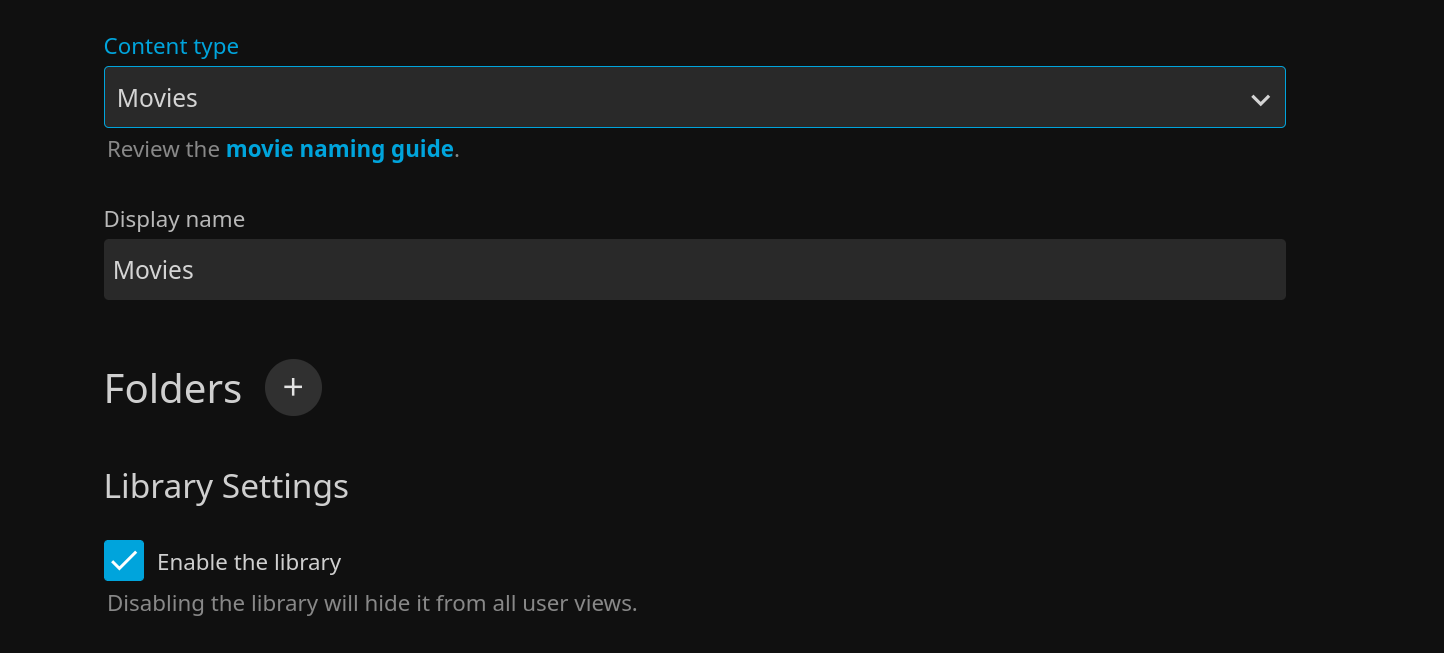


Example of https. Still says connection not secure though (because self signed cert)

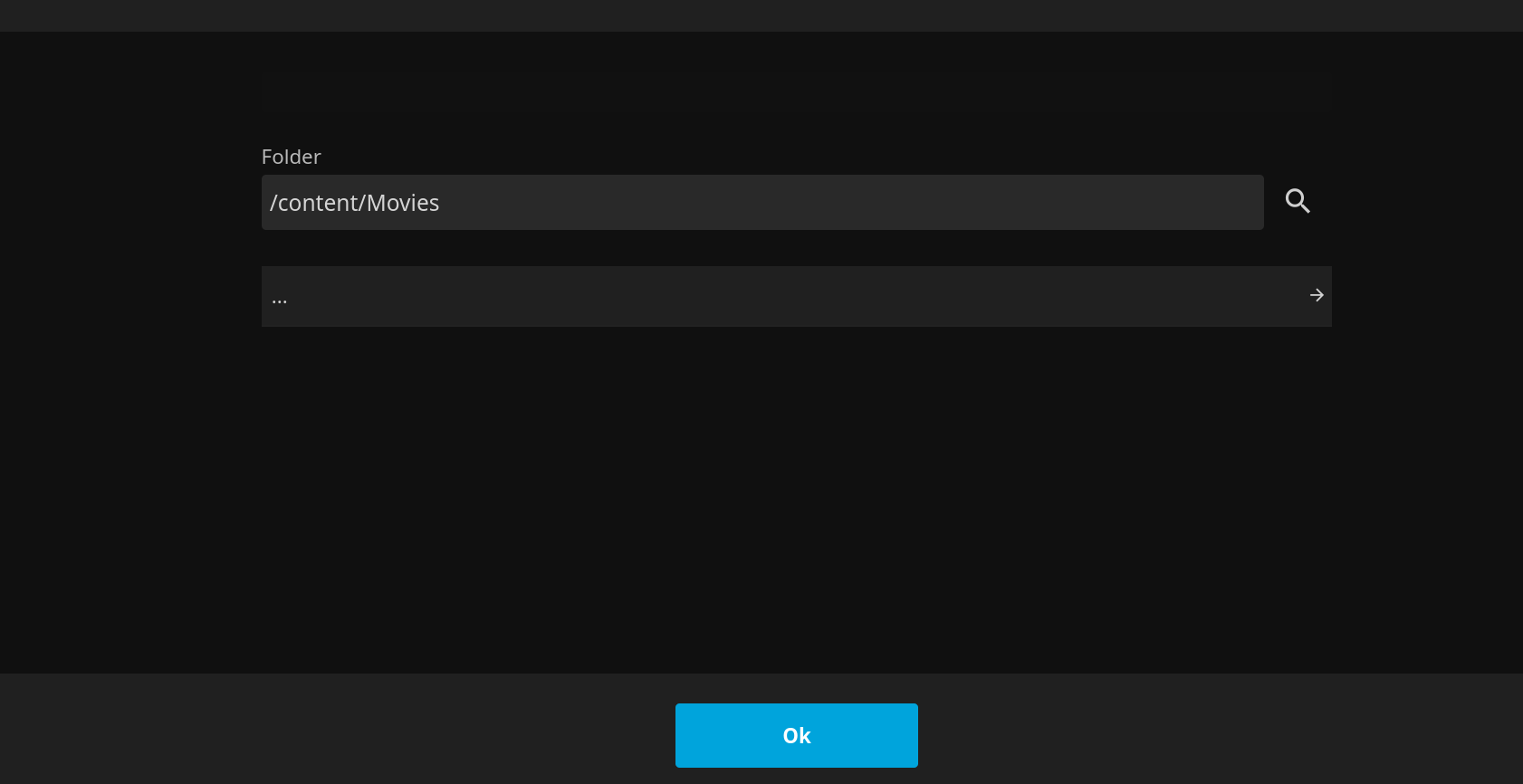


Now go through the process with https setup. Set up the admin credentials (Remember this!) Set up a movie folder when prompted to create a library.

Change content type to movies. Press plus button on Folders



Put in this path

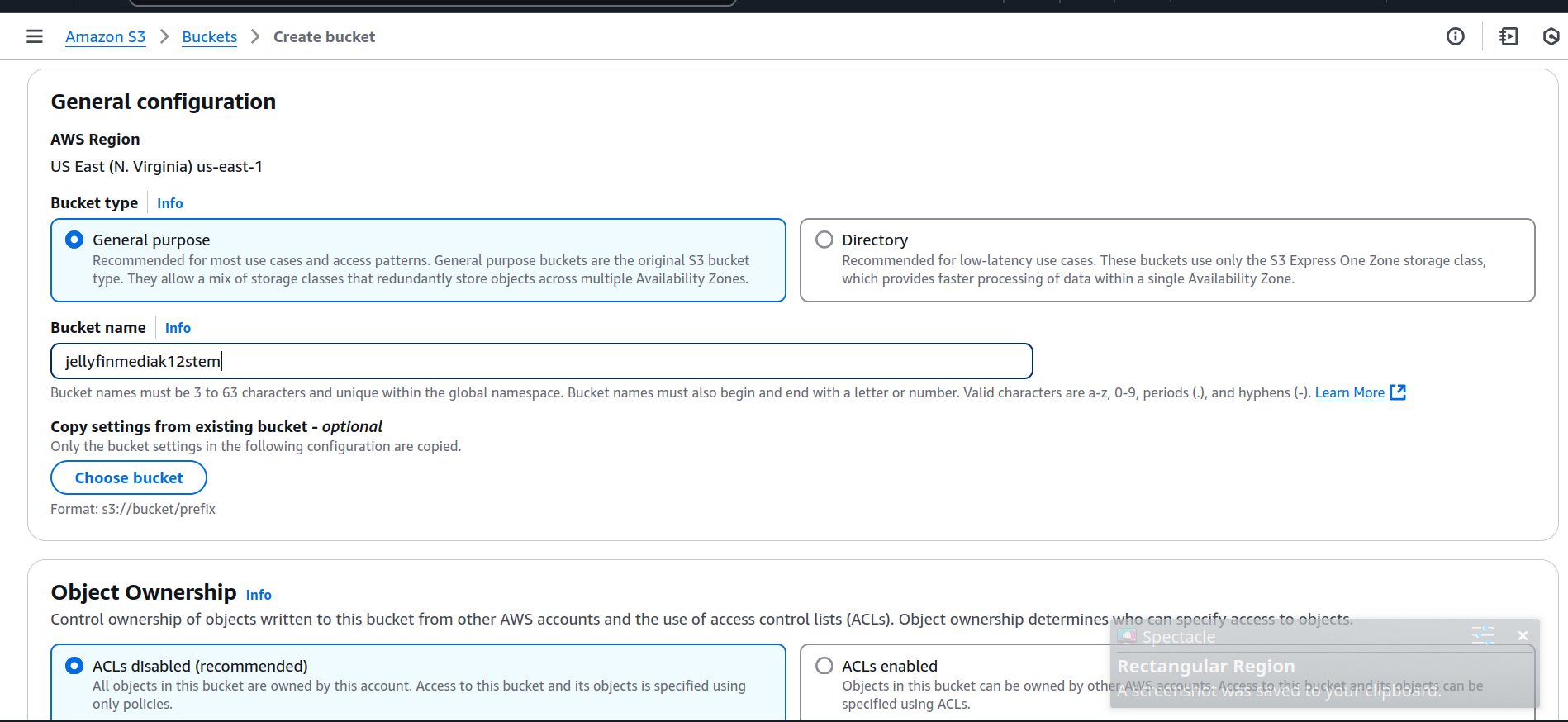


Finish installation and login with admin and password credentials created earlier

## Creating S3 Bucket

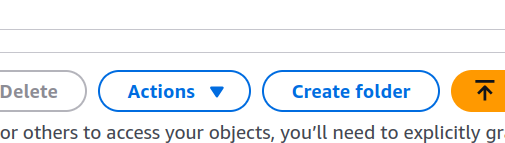
We will use an S3 bucket to easily upload files and store the videos.

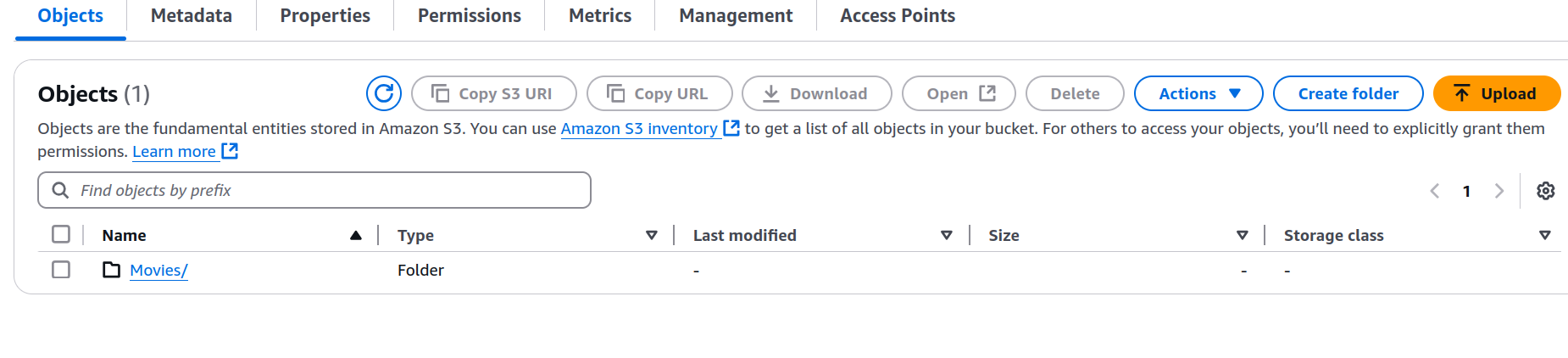
Create the S3 bucket. Remember, bucket must be unique (globally too, across all users)



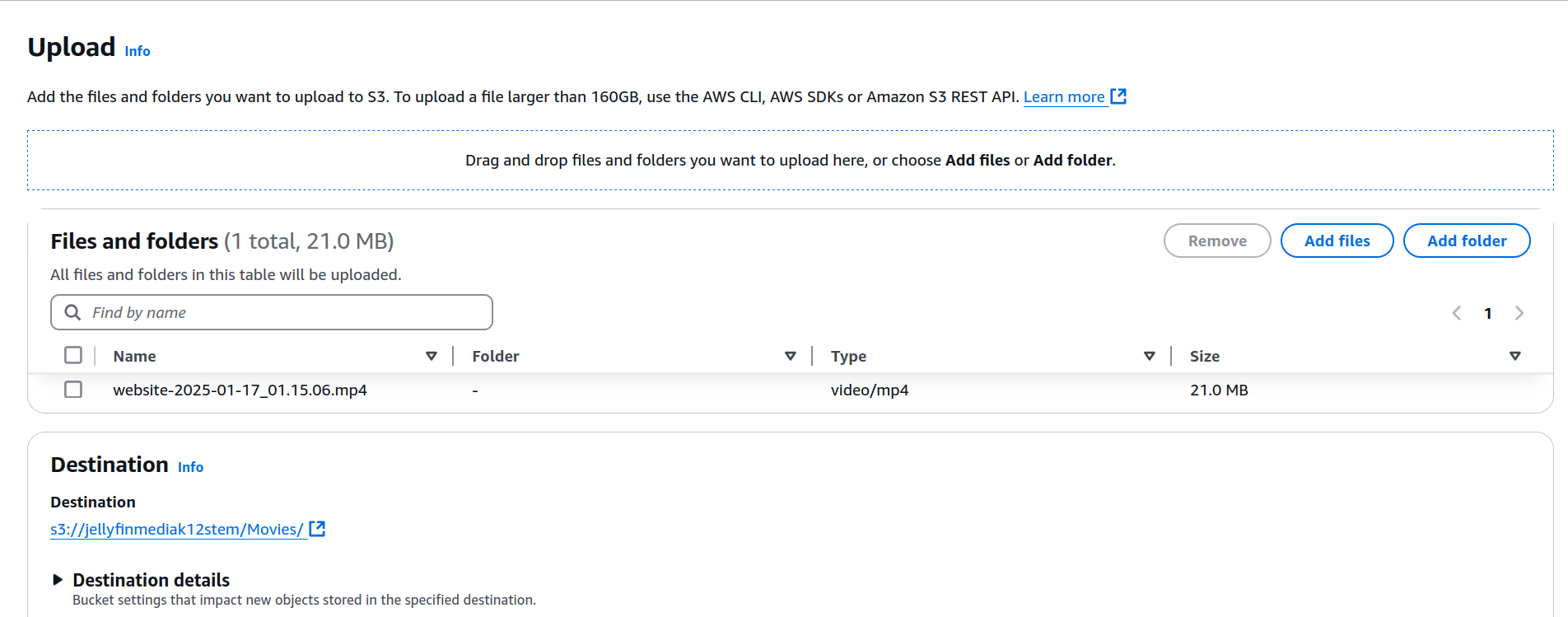
keep all default settings

Create a folder called Movies/





Upload your video of choice in the movies folder

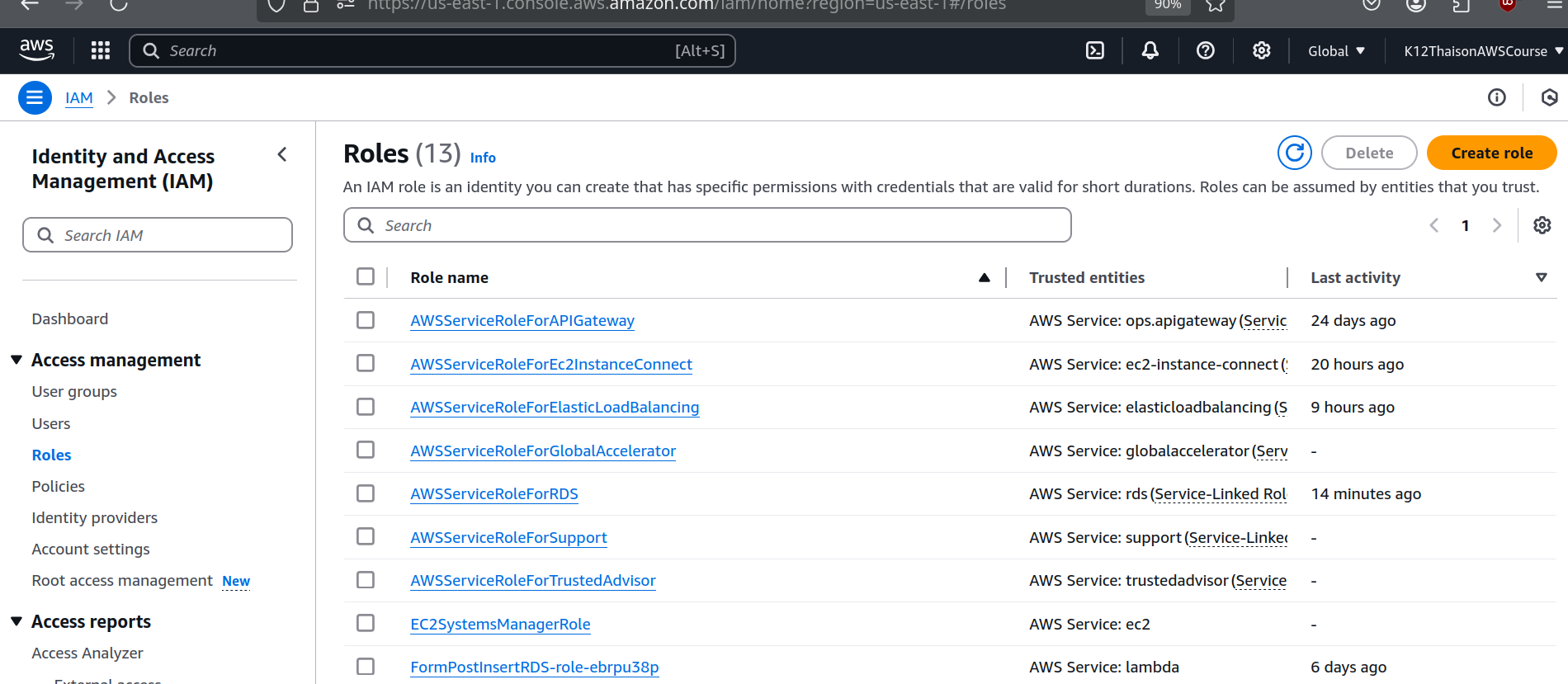


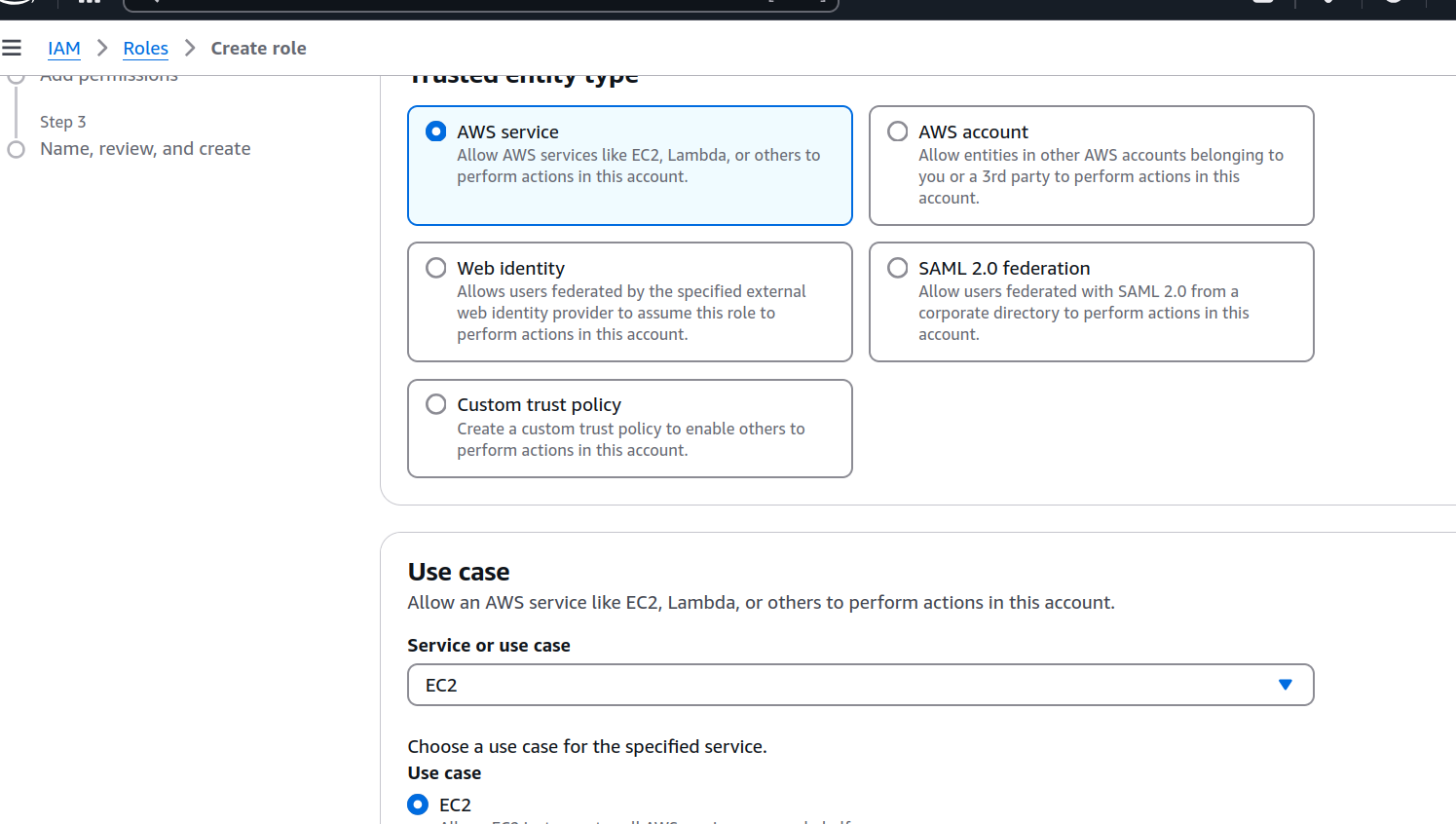
## 

## Syncing -> Creating an IAM Role

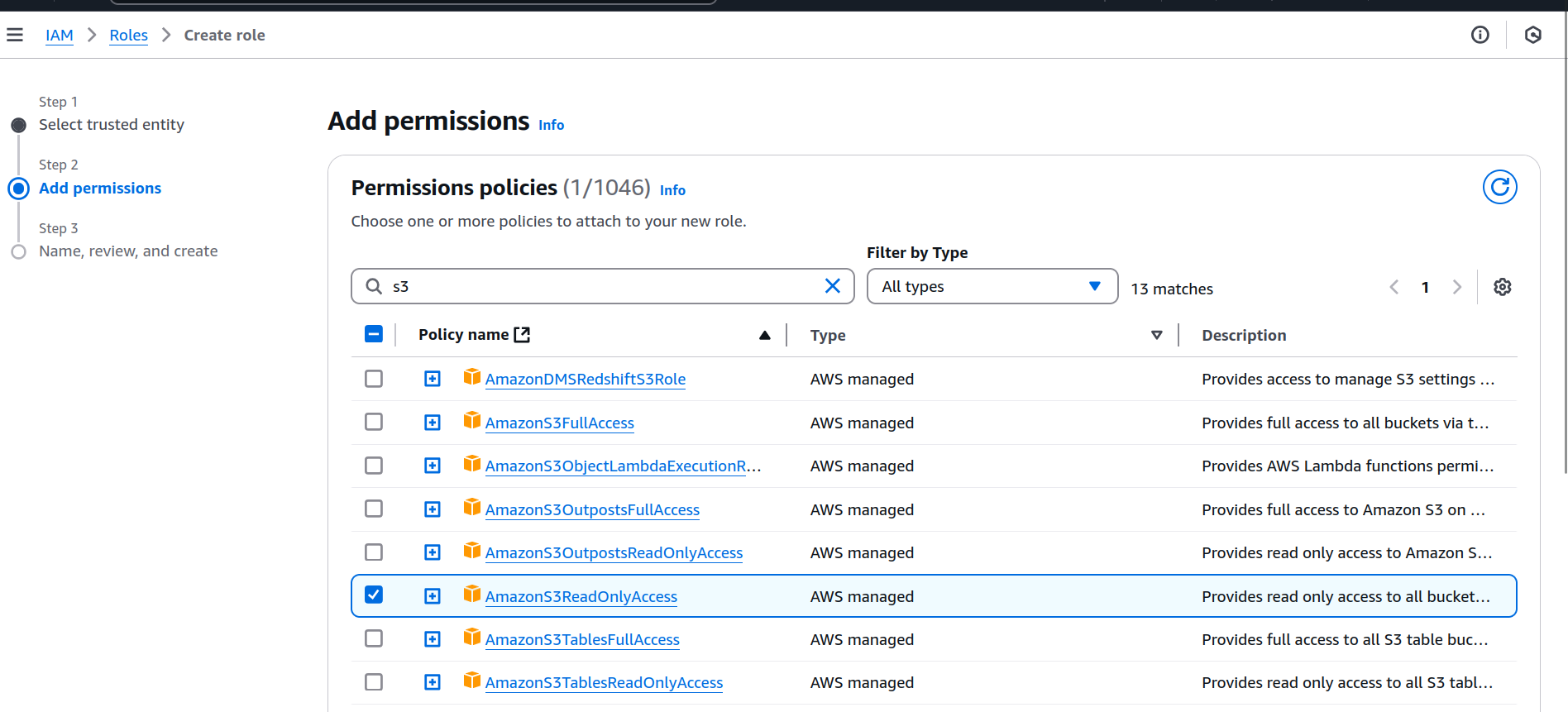
We need to create an IAM role to allow the EC2 to sync with the S3 bucket.

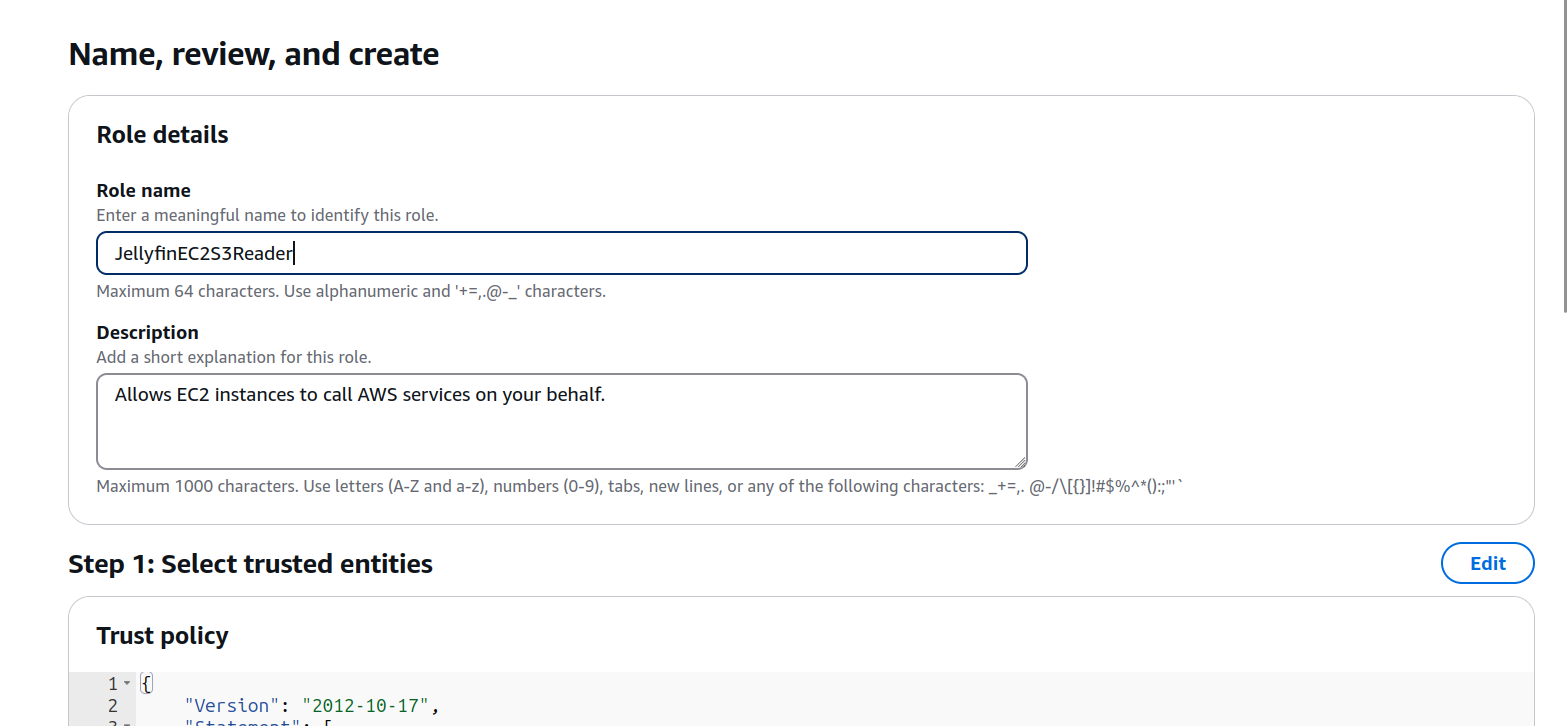
Go to IAM dashboard and create a new role.





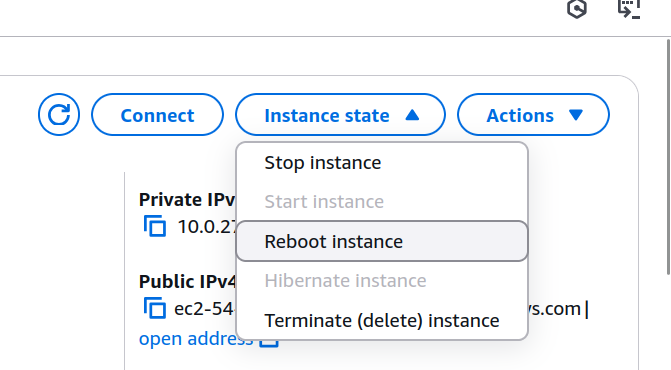
Use the s3readonly policy. Name the role something reasonable.





Add the role to the EC2 instance





## 

### Sync bucket

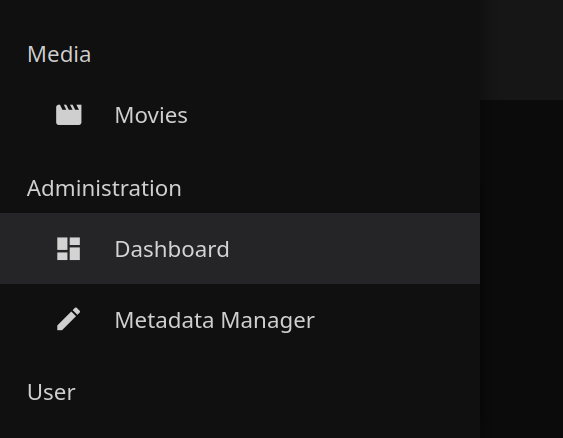
Sync your bucket by going through these commands

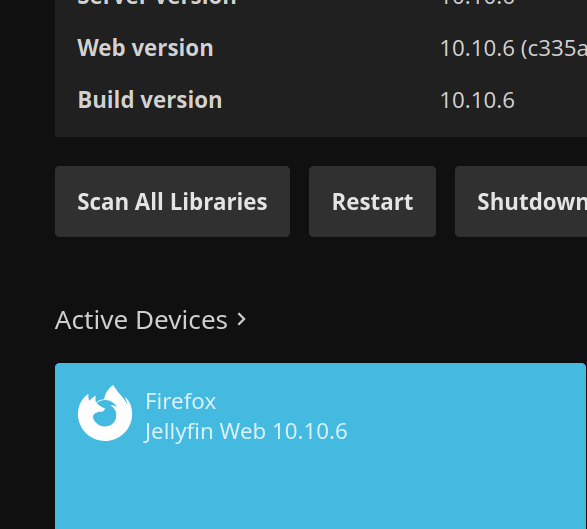
| sudo apt update sudo snap install aws-cli --classic  sudo -u jellyfin aws s3 sync s3://your-bucket-name/ /content/ example: sudo -u jellyfin aws s3 sync s3://jellyfinmediak12stem/ /content/ |
| --- |

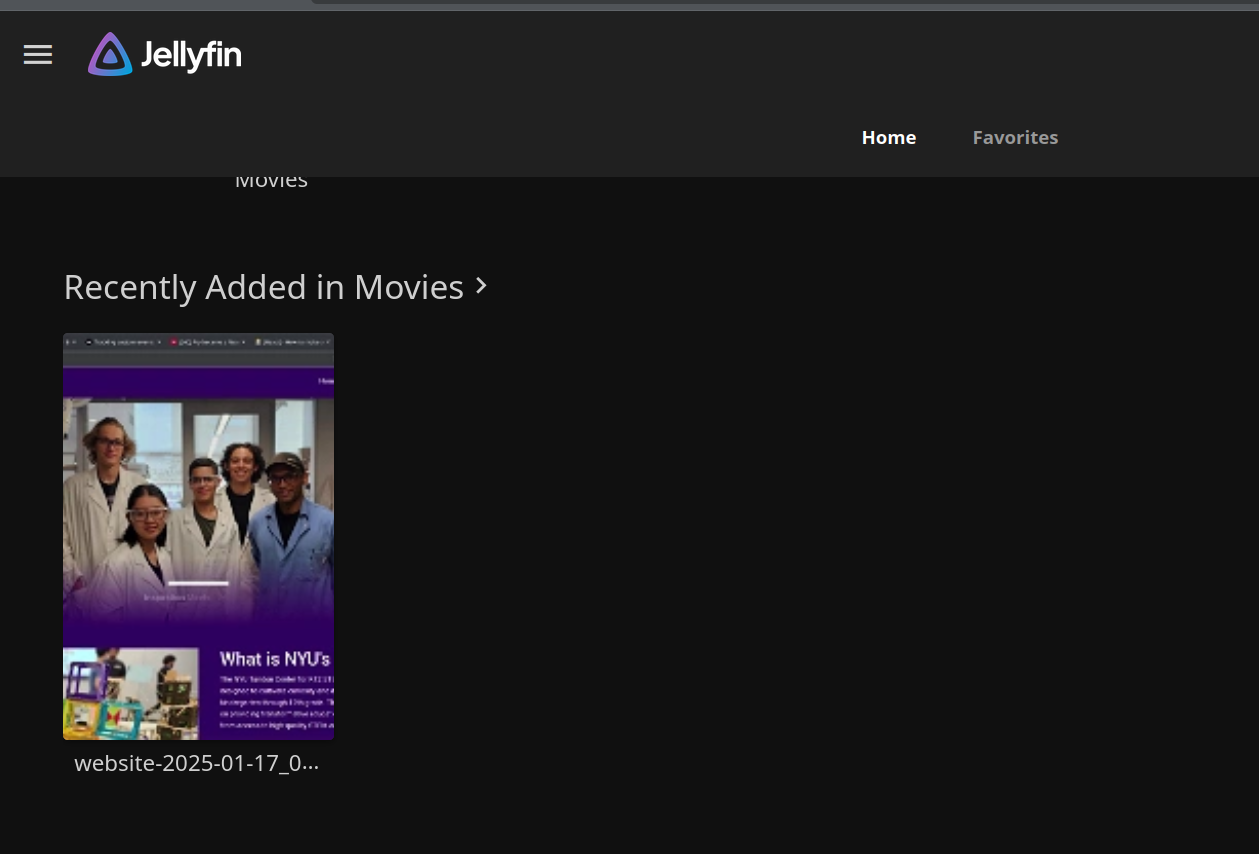
The video should now show up in your folders.

## Refresh Jellyfin

Go to the administrator panel and refresh the library for jellyfin to find the files



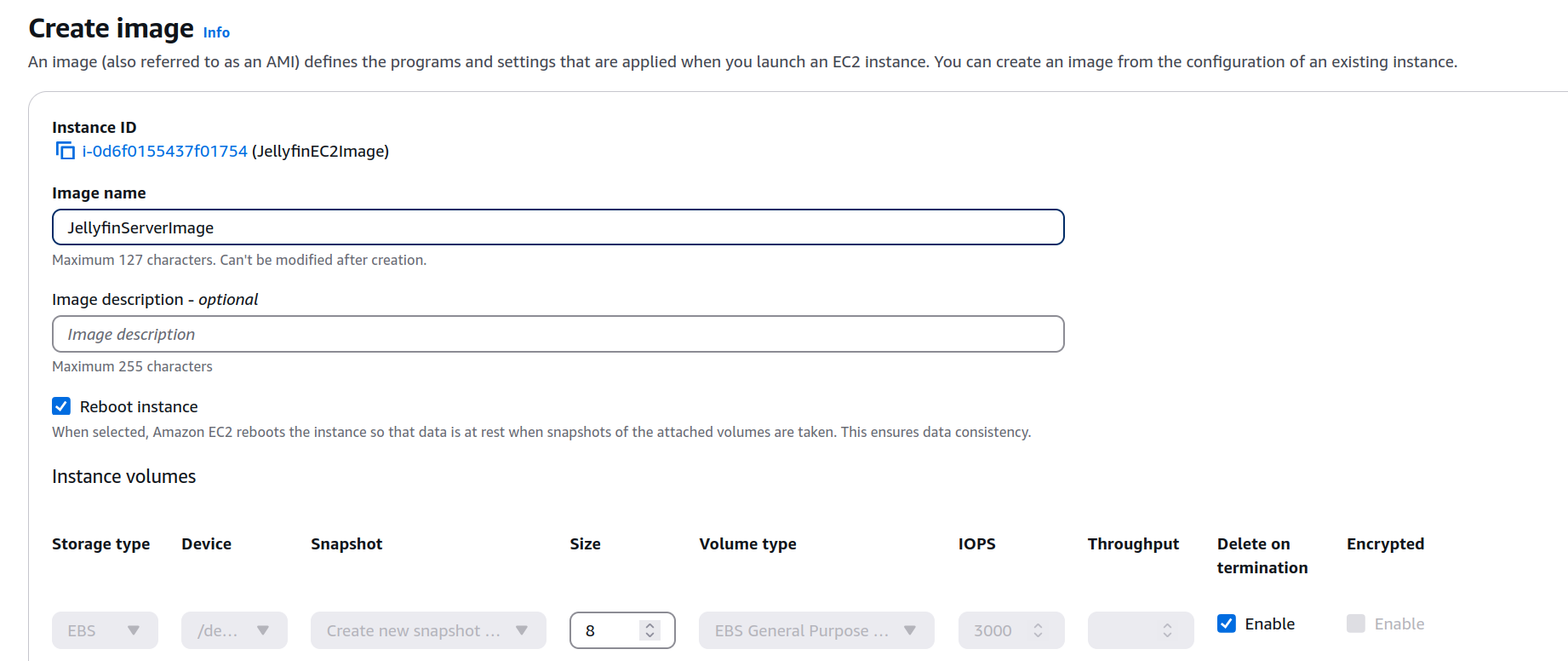




## Creating an Image

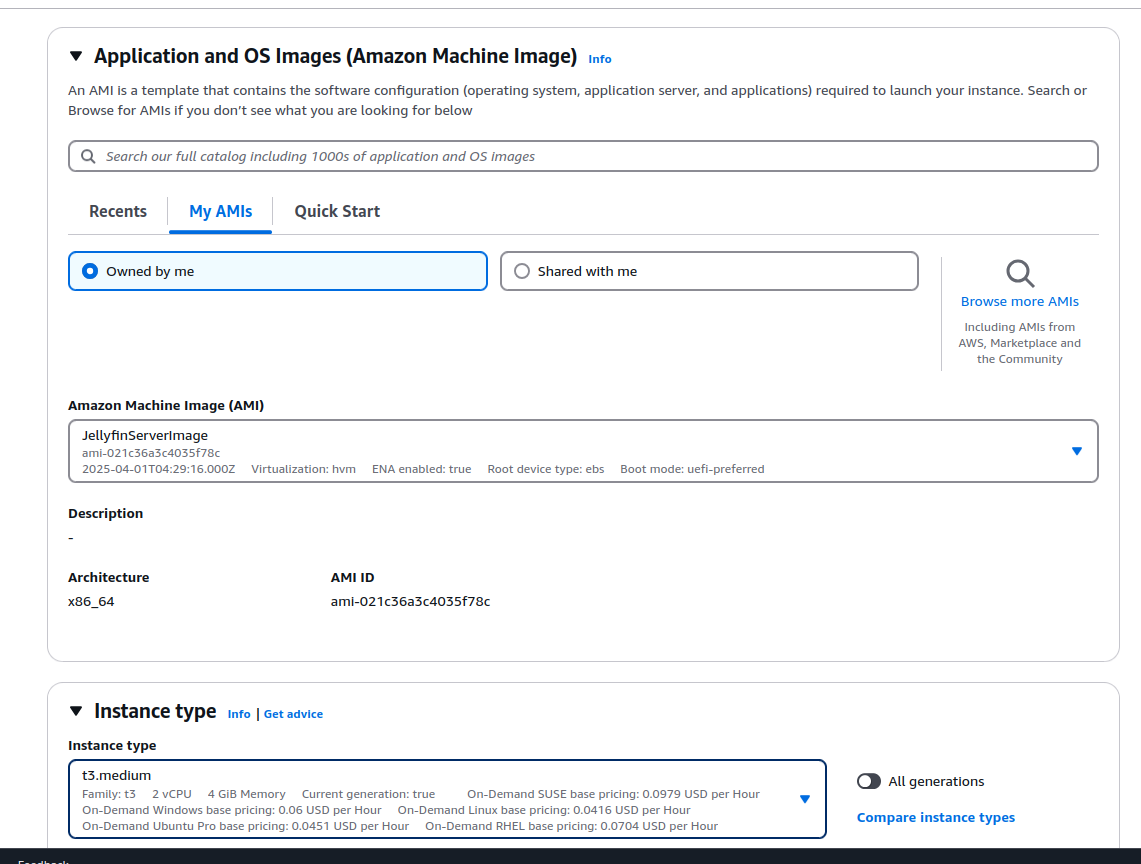
Let’s create an Image of the server we just made (to save the work and also to create copies later)





## Launching an EC2 instance with more Oomph (Do it yourself)

You can now use this image to launch instances that are stronger. The current t2.micro can only support direct play on very small videos. It will **LAG** on bigger videos like 1080p and 4k videos with multiple people watching at once. That’s why we can use the image to create an instance that is exactly the same, but running with better hardware. We can also put it behind a private subnet now, if you automate the syncing, so that you don’t have to interact with the server anymore.



## End

Make sure to delete/stop and resources you are not using. Things to note:

* Your EBS storage volume is not automatically deleted when terminating the EC2. You have to manually delete it
* Your ALB must be deleted, it cannot be just turned off