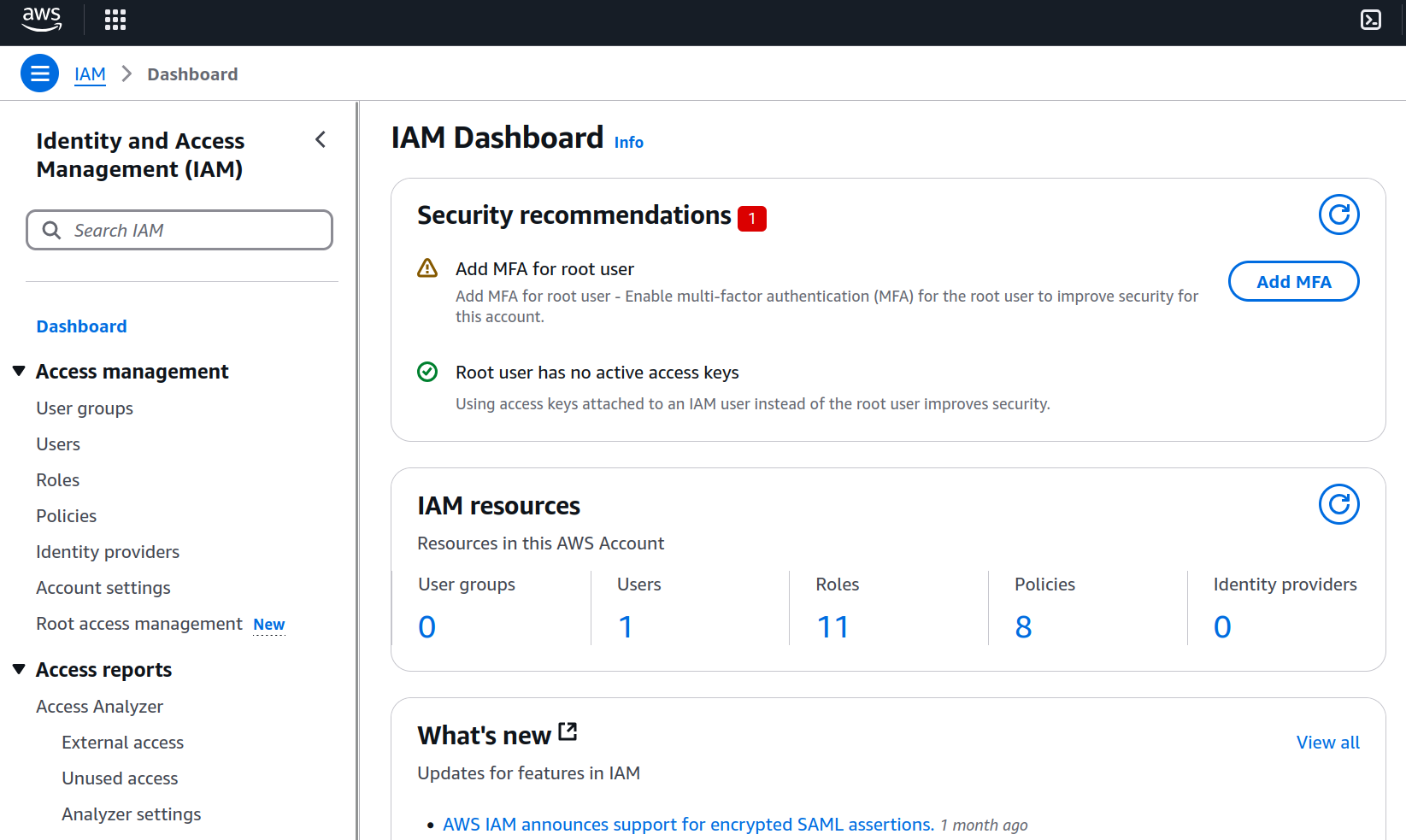
## Prerequisites

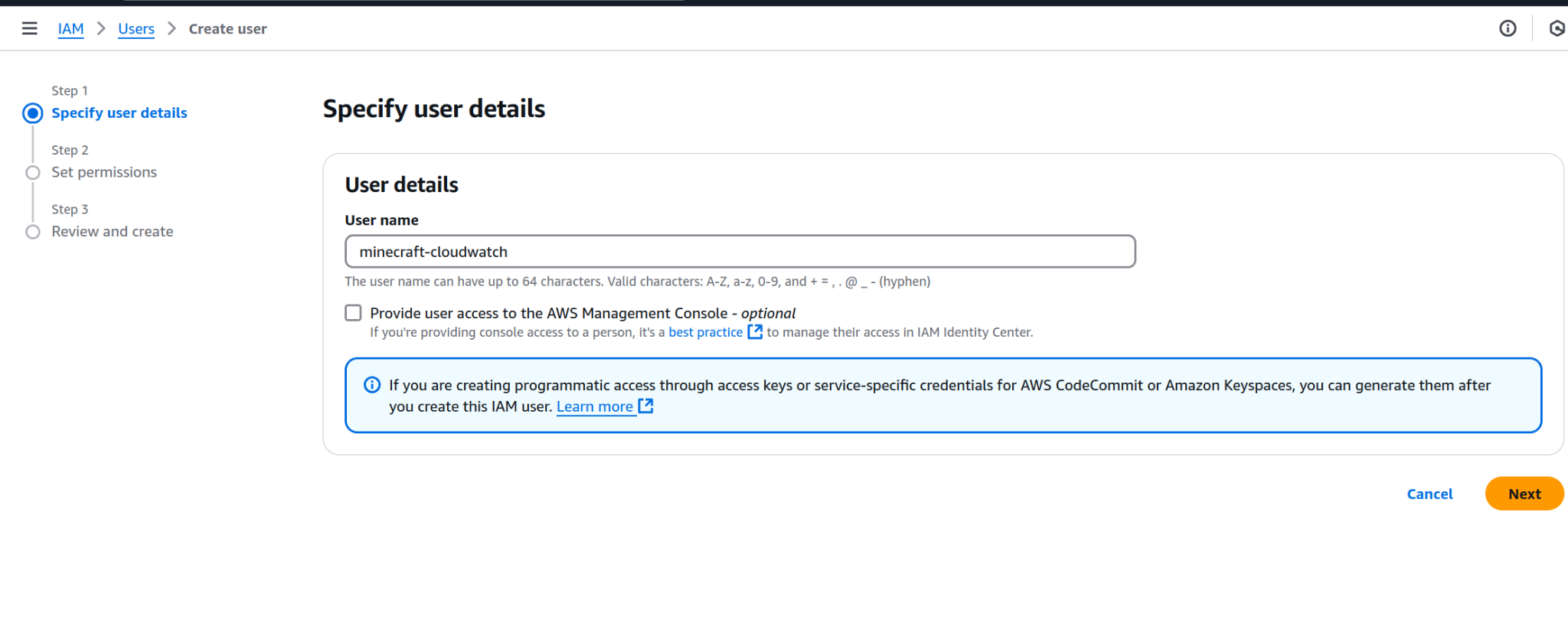
* Make sure you are on us-east-1 (virginia)
* You can only copy paste in lightsail terminal using right click to paste

## 

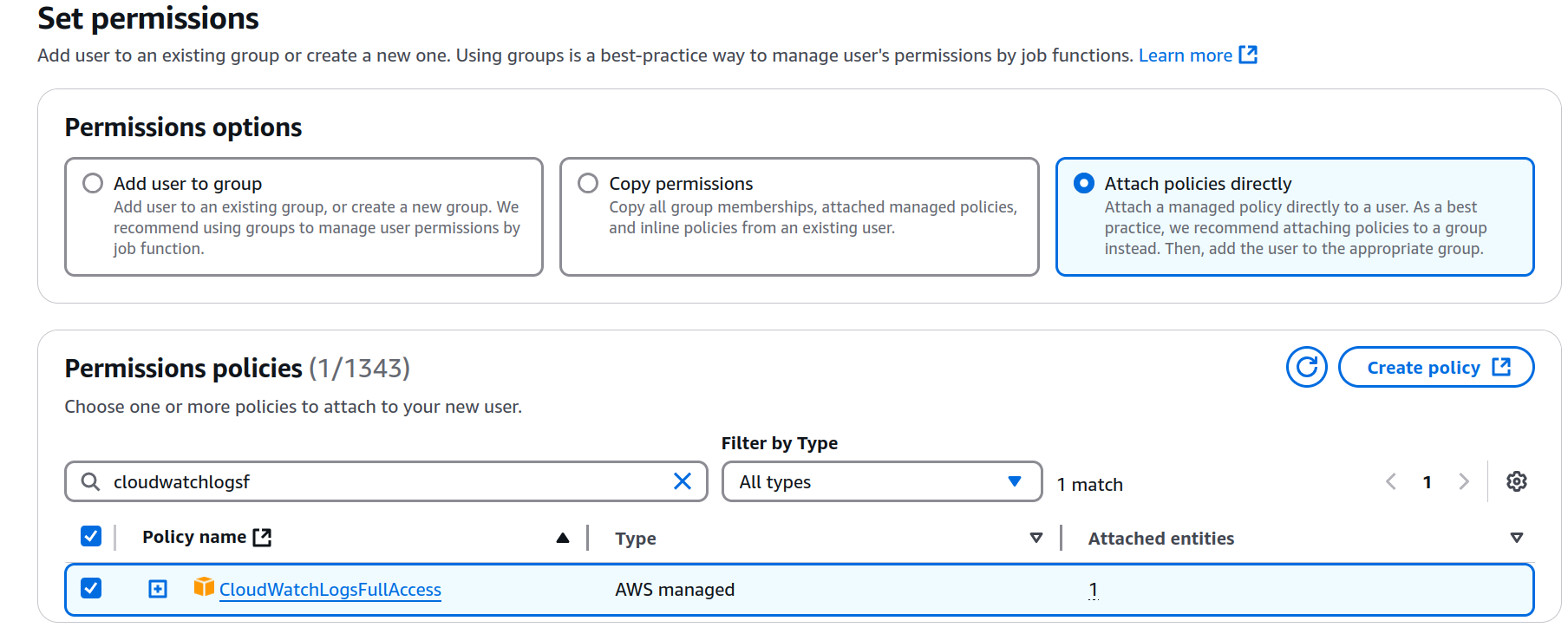
## Creating user for cloud watch with permissions

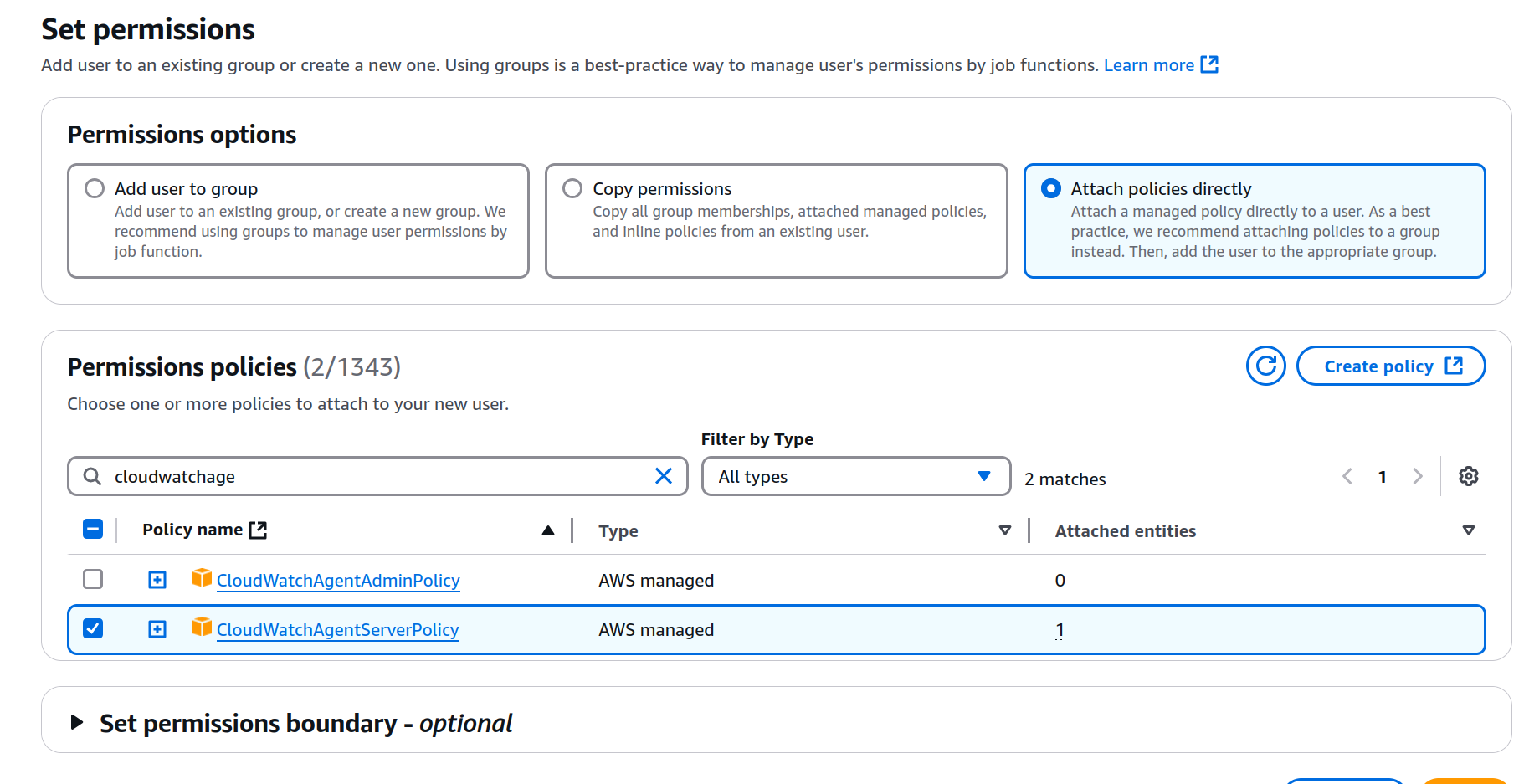
Go to users tab and create a user



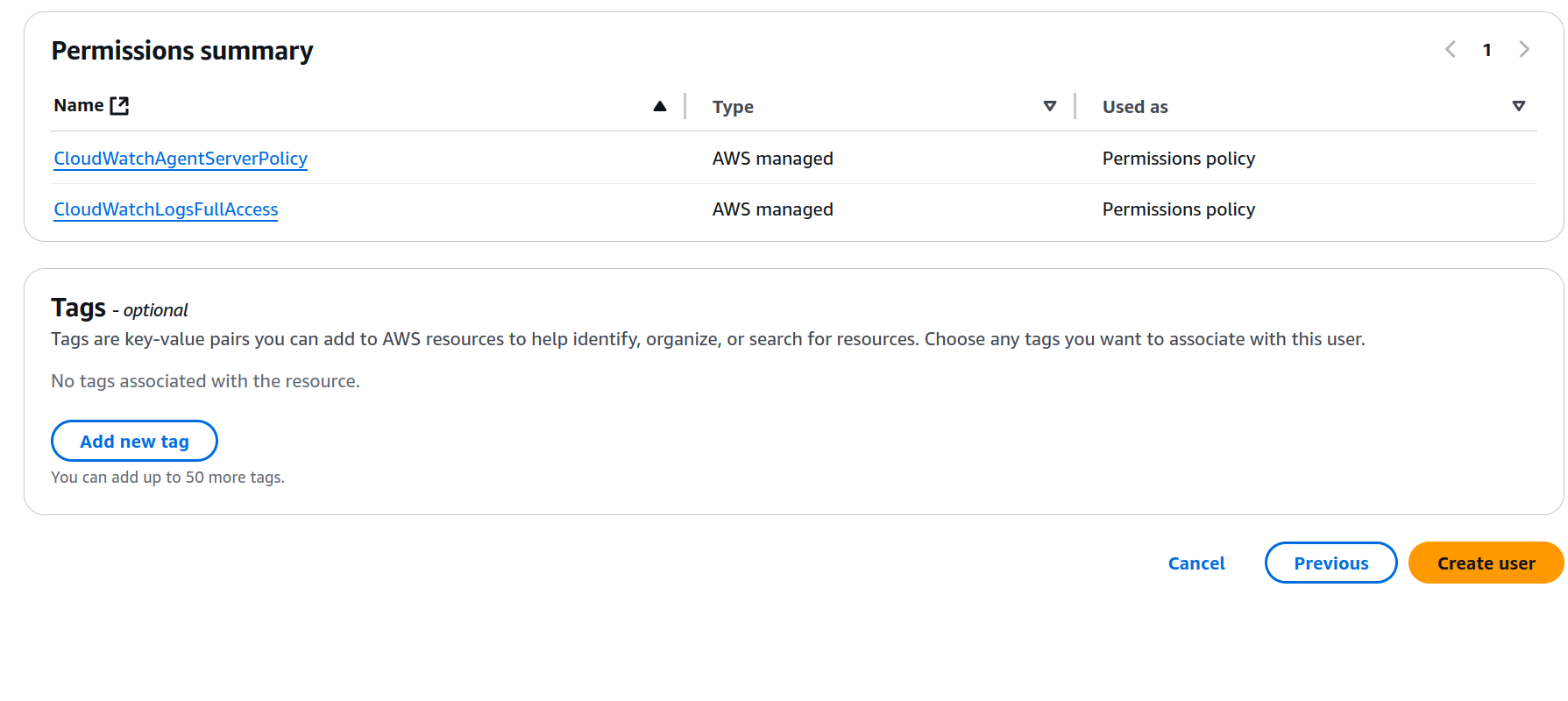


Attach policies to them

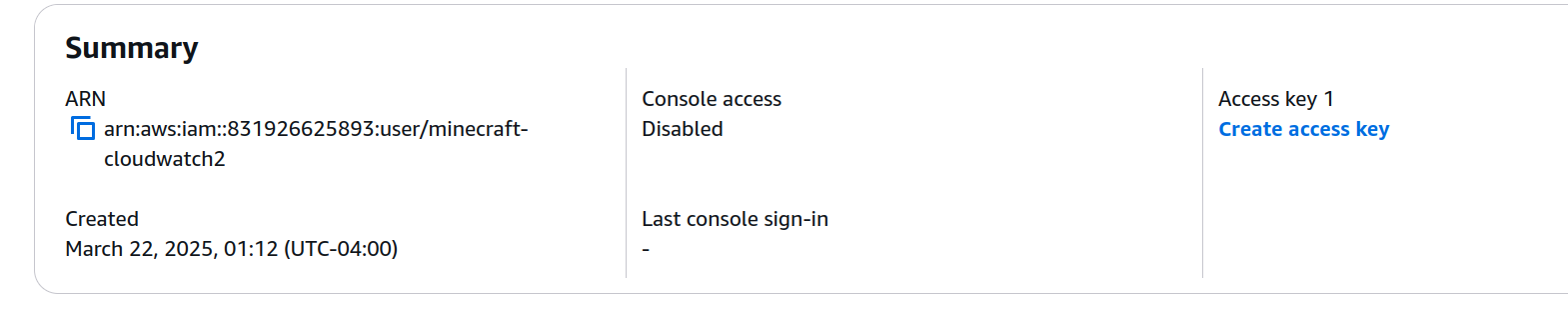


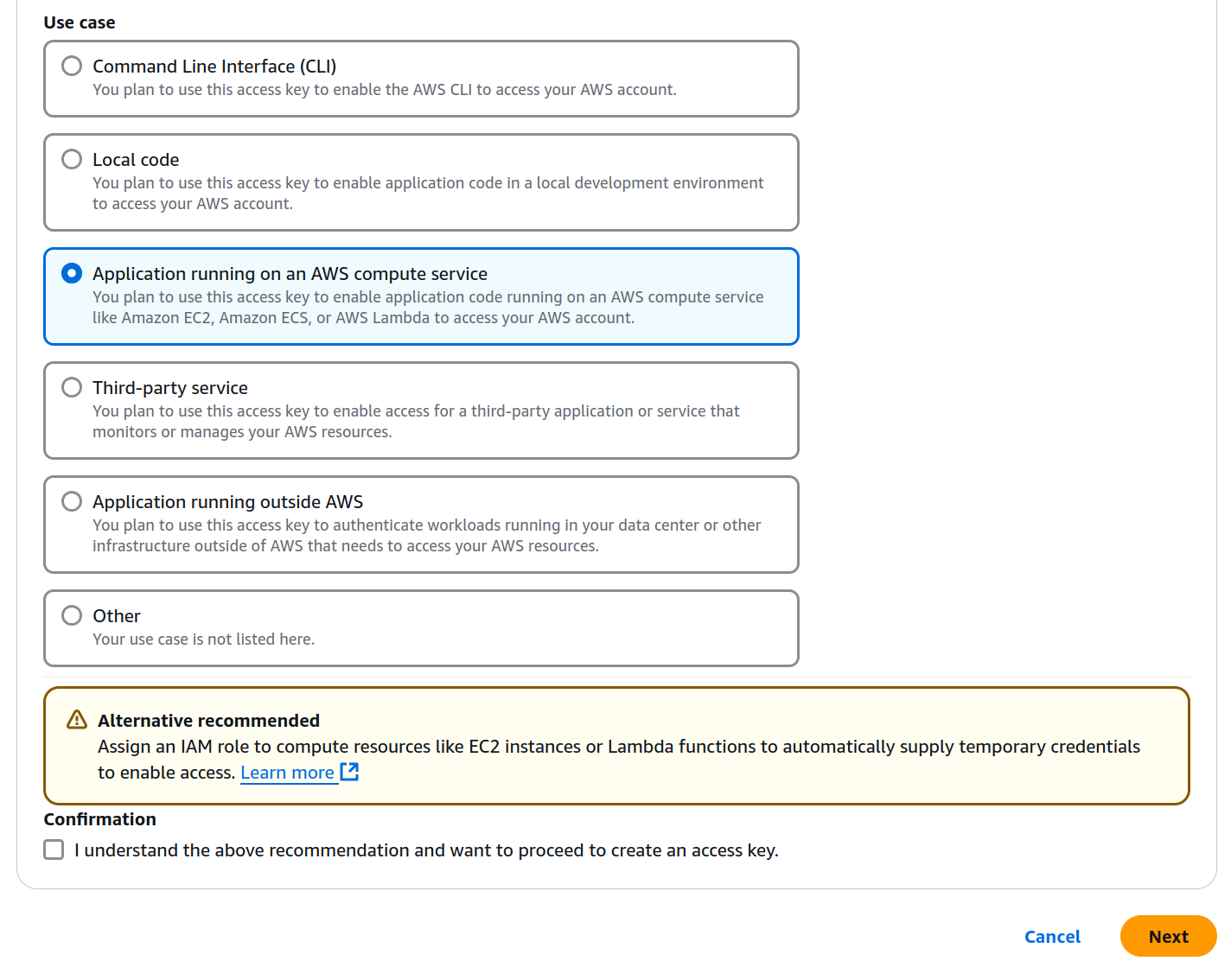


Make sure these are the policies at the end



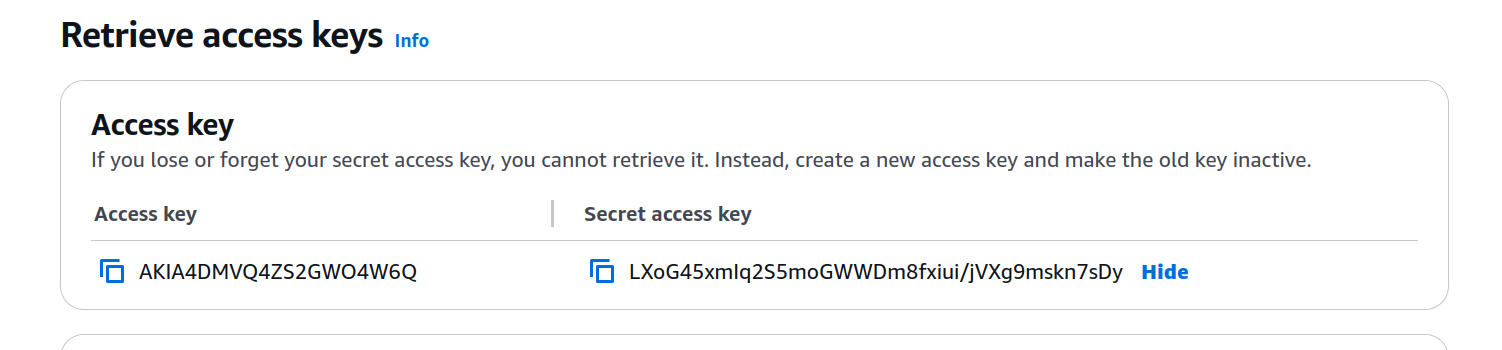
### Create an access key





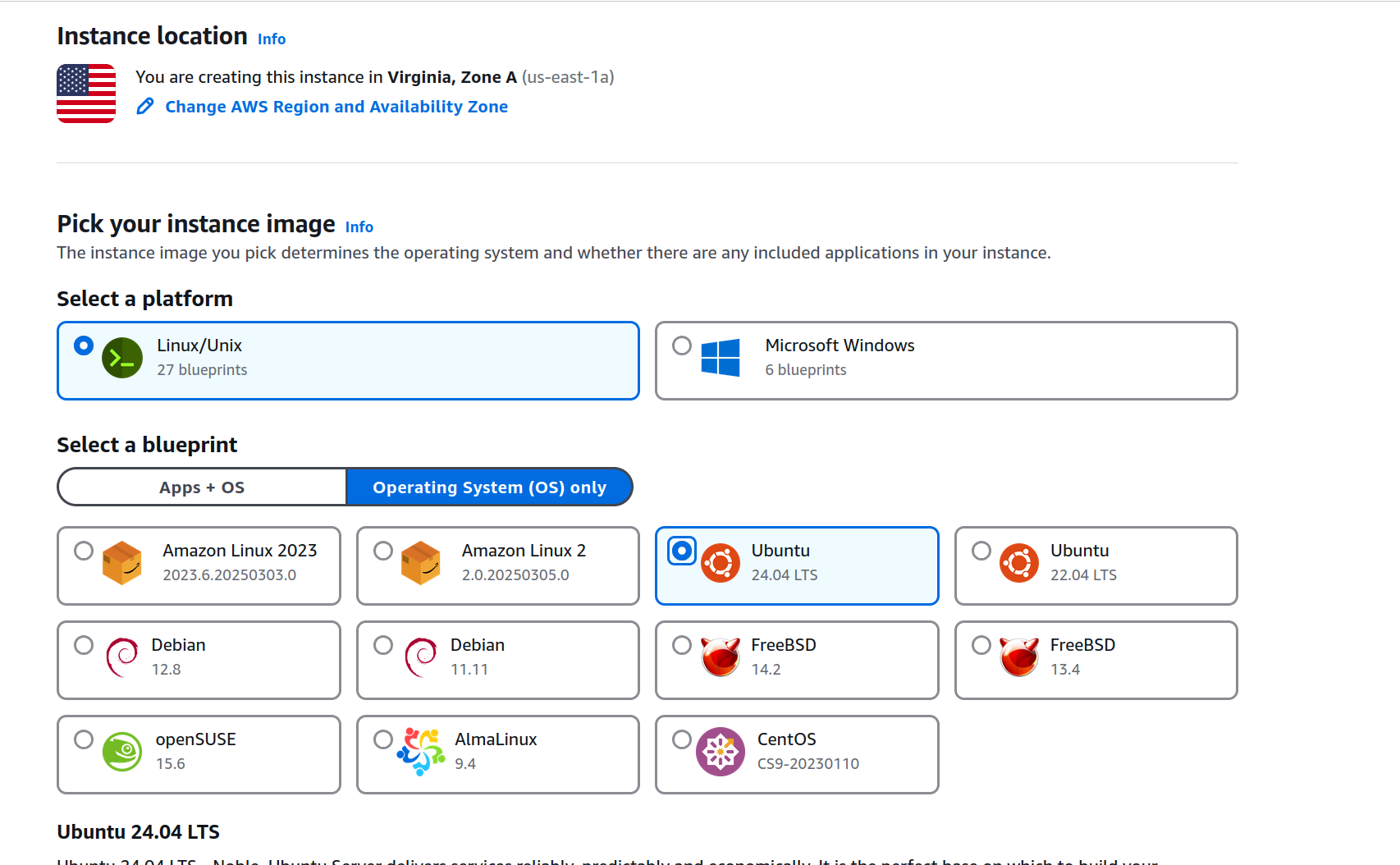
Create a descriptive name if you want

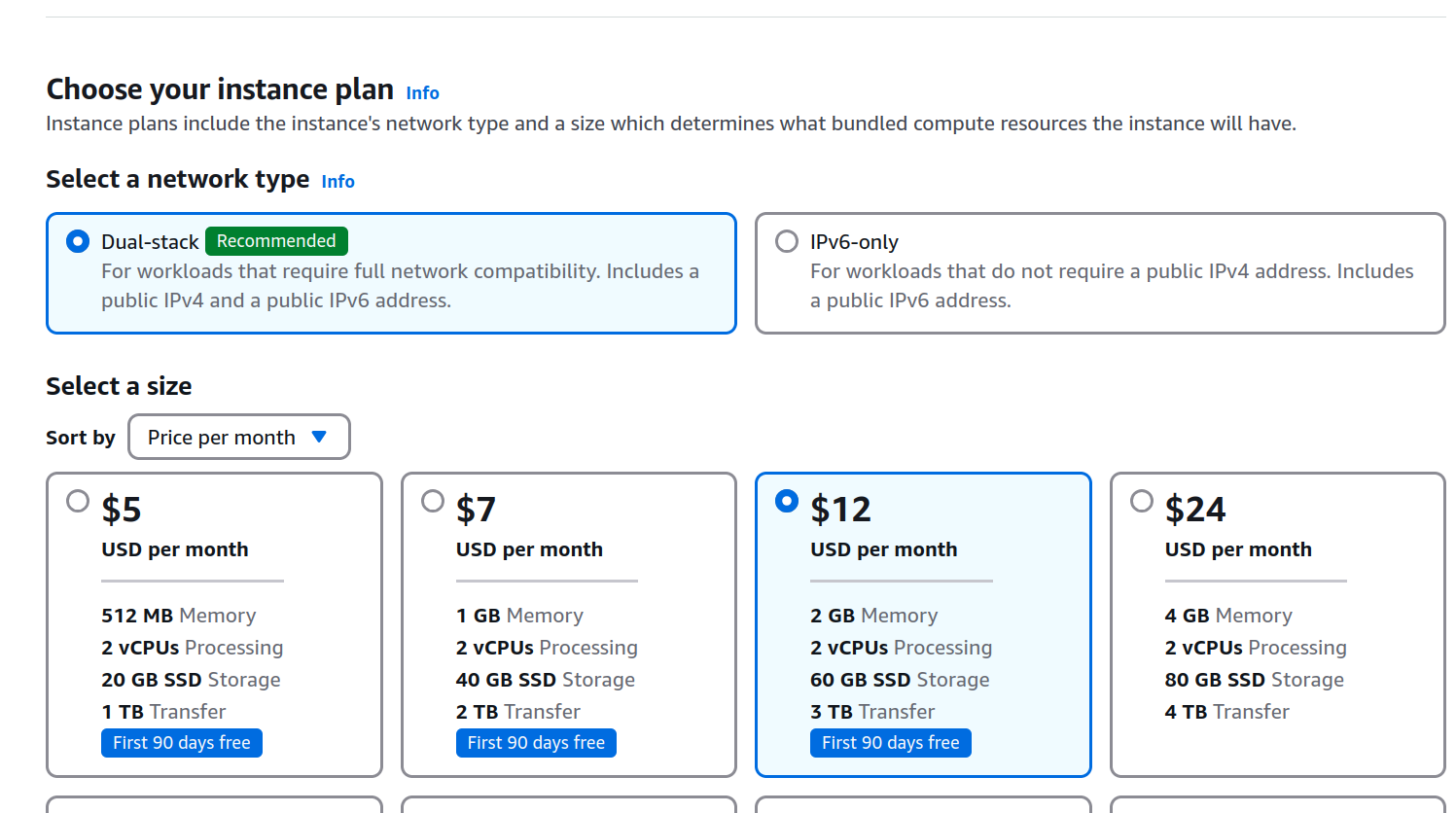
Copy these two keys somewhere!



## Create Lightsail instance

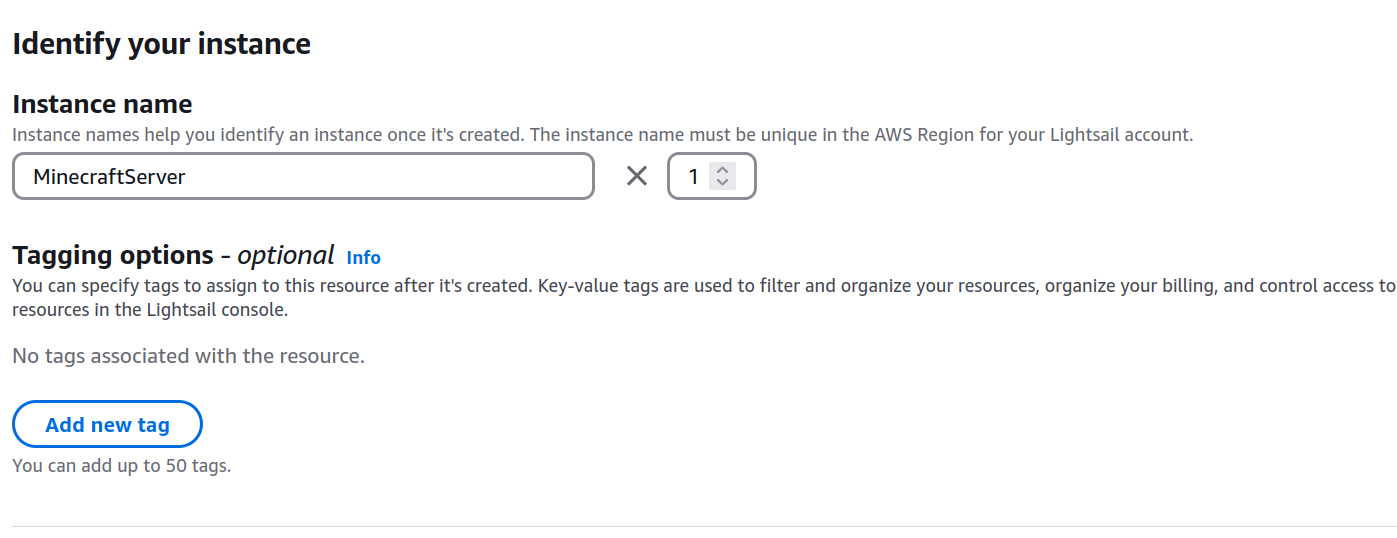
Make sure you are using us-east-1a and using Ubuntu 24.04





### Start up script

| apt-get update  cd /home/ubuntu/  wget https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb  dpkg -i -E ./amazon-cloudwatch-agent.deb  apt install -y tmux  apt install -y default-jre  mkdir minecraft-server  cd minecraft-server  wget https://piston-data.mojang.com/v1/objects/4707d00eb834b446575d89a61a11b5d548d8c001/server.jar  java -Xmx1G -Xms1G -jar server.jar nogui |
| --- |



## Adding Cloudwatch agent to Lightsail Instance

sudo nano /opt/aws/amazon-cloudwatch-agent/bin/config.json

| {  "agent": {  "metrics\_collection\_interval": 60,  "run\_as\_user": "root"  },  "logs": {  "logs\_collected": {  "files": {  "collect\_list": [  {  "file\_path": "/home/ubuntu/minecraft-server/logs/latest.log",  "log\_group\_name": "minecraft-server-logs",  "log\_stream\_name": "{instance\_id}-minecraft",  "retention\_in\_days": 14  }  ]  }  }  } } |
| --- |

Ctrl+x, y, then enter to save

| sudo nano /opt/aws/amazon-cloudwatch-agent/etc/common-config.toml |
| --- |

Copy paste this at the end, after all the comments

| [credentials] shared\_credential\_profile = "minecraft-cloudwatch" shared\_credential\_file = "/home/ubuntu/.aws/credentials" |
| --- |

| mkdir ~/.aws/  nano ~/.aws/credentials |
| --- |

paste this in, replacing with the values you copied earlier

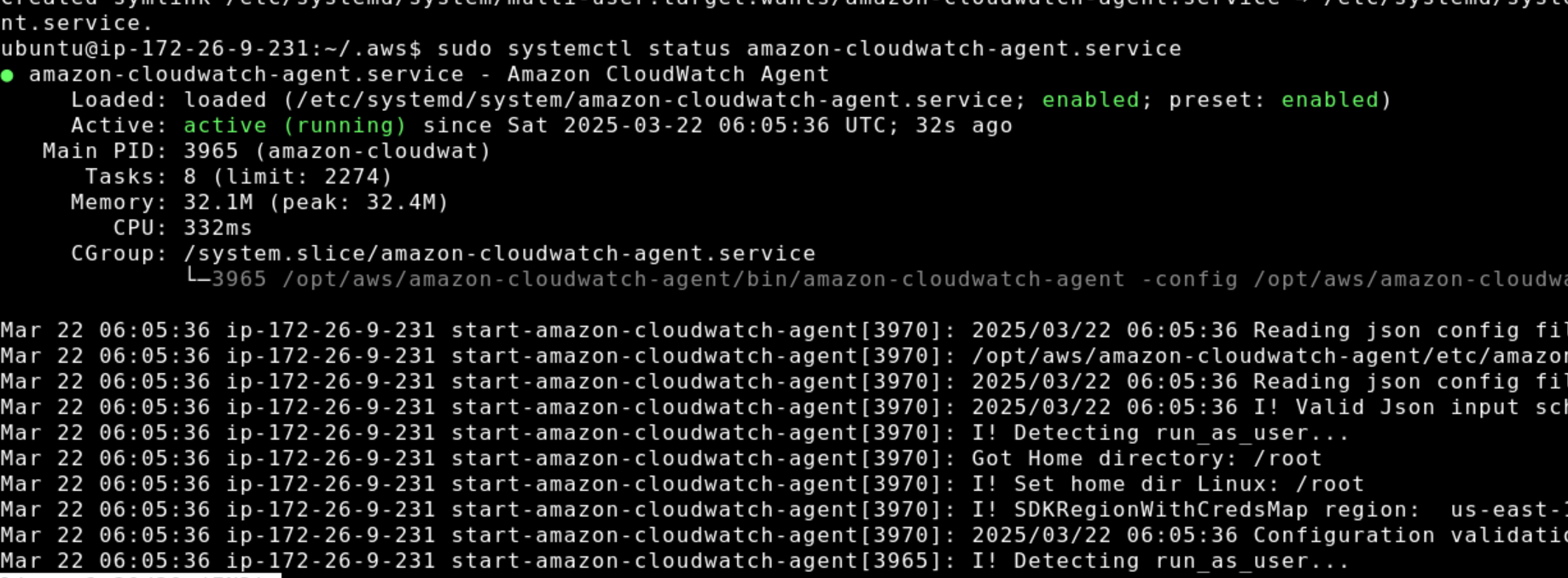
| [minecraft-cloudwatch]  aws\_access\_key\_id = YOUR\_ACCESS\_KEY\_ID  aws\_secret\_access\_key = YOUR\_SECRET\_ACCESS\_KEY  region = us-east-1 |
| --- |

now paste this whole thing in, all one line

| sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json |
| --- |

Check if its running properly

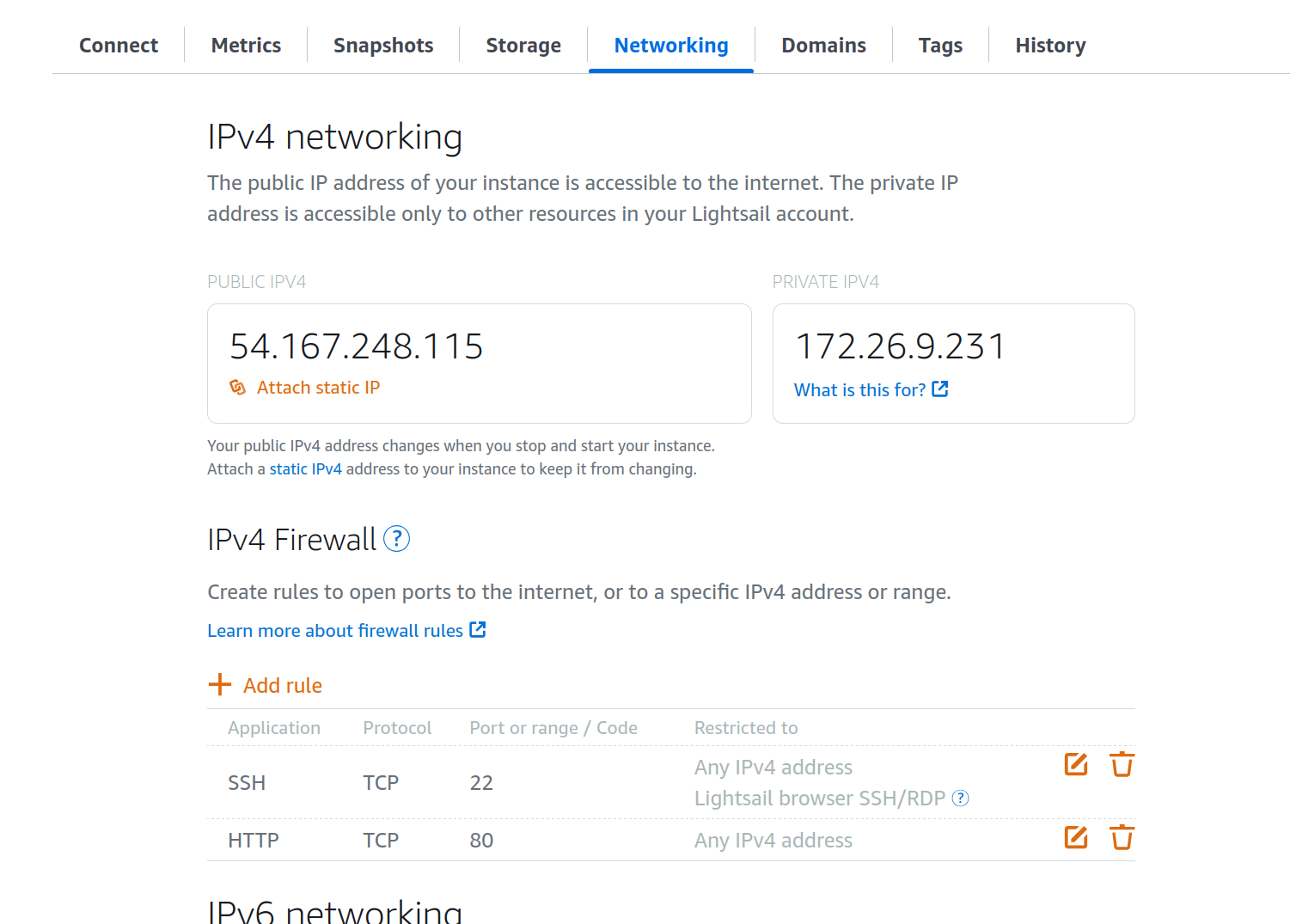
| sudo systemctl status amazon-cloudwatch-agent |
| --- |

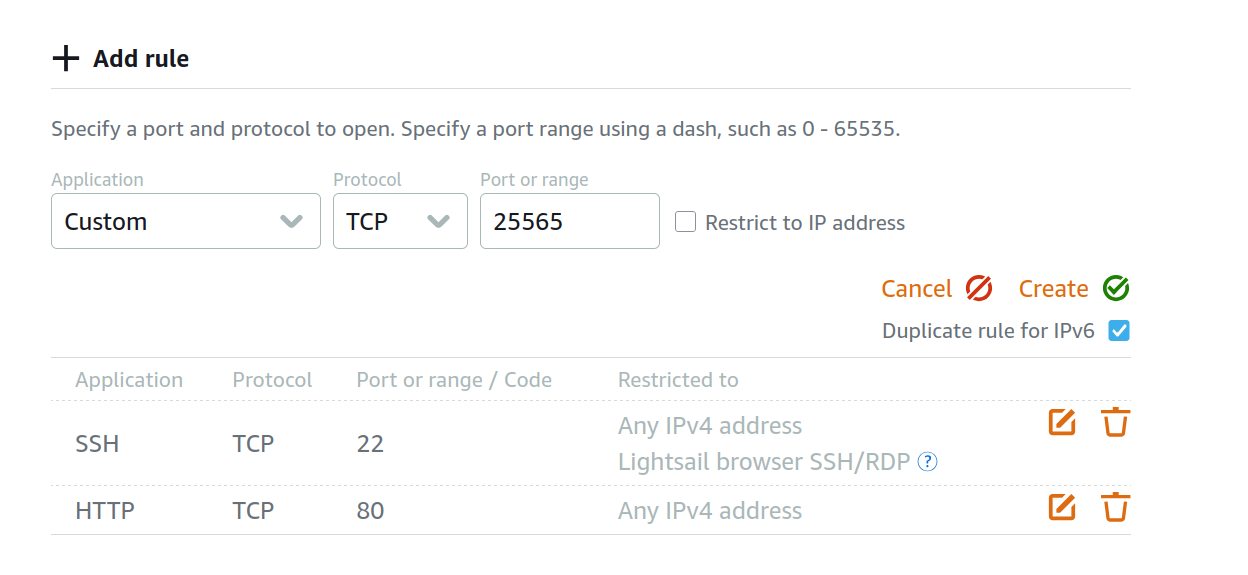


Should be active (running)

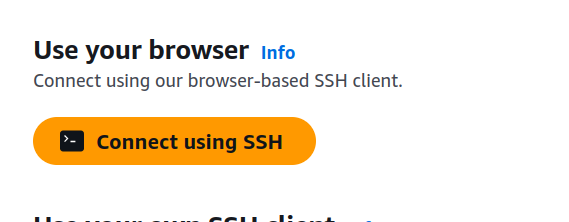
## Start Minecraft Server

First open up port 25565 for Minecraft





Now use browser connect to ssh



| tmux new -s minecraft cd ~/minecraft-server |
| --- |

| sudo nano eula.txt |
| --- |

change eula=true

ctrl+x and press Y and then enter to save

then do

| sudo java -Xmx1G -Xms1G -jar server.jar nogui |
| --- |

press ctrl+b and then press d

Try joining your minecraft server by using your lightsail instance ip and port 25565

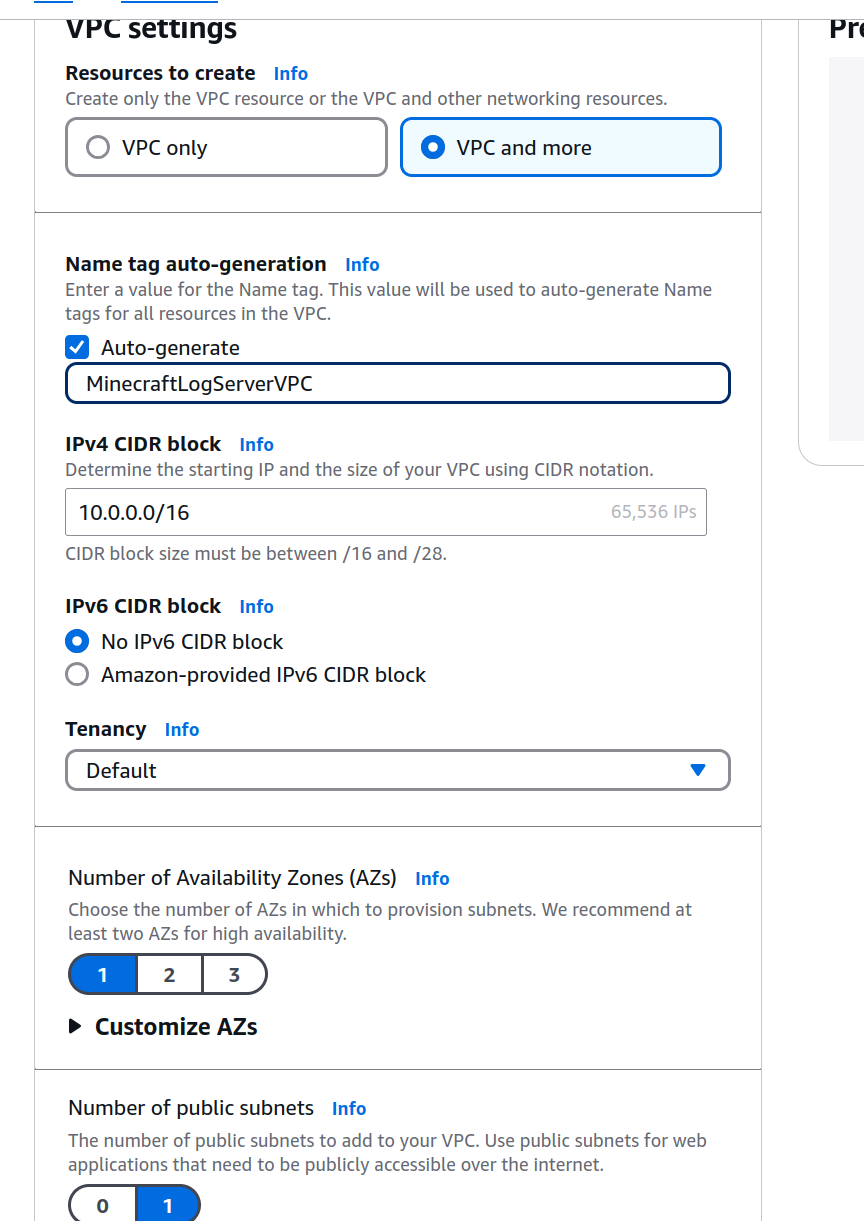
restart the cloudwatch amazon agent. This is because a new log file has been made, so it must be restarted

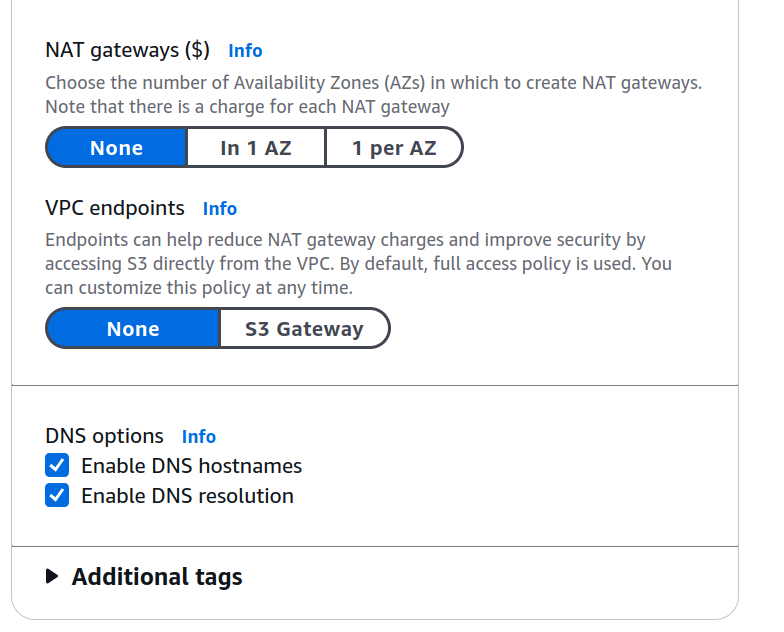
| sudo systemctl restart amazon-cloudwatch-agent |
| --- |

Check this to make sure there are no errors

| sudo tail -n 100 /var/log/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.log |
| --- |

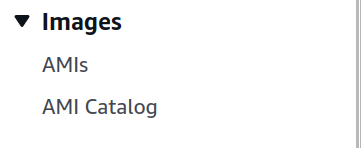
## Create a VPC for the other services

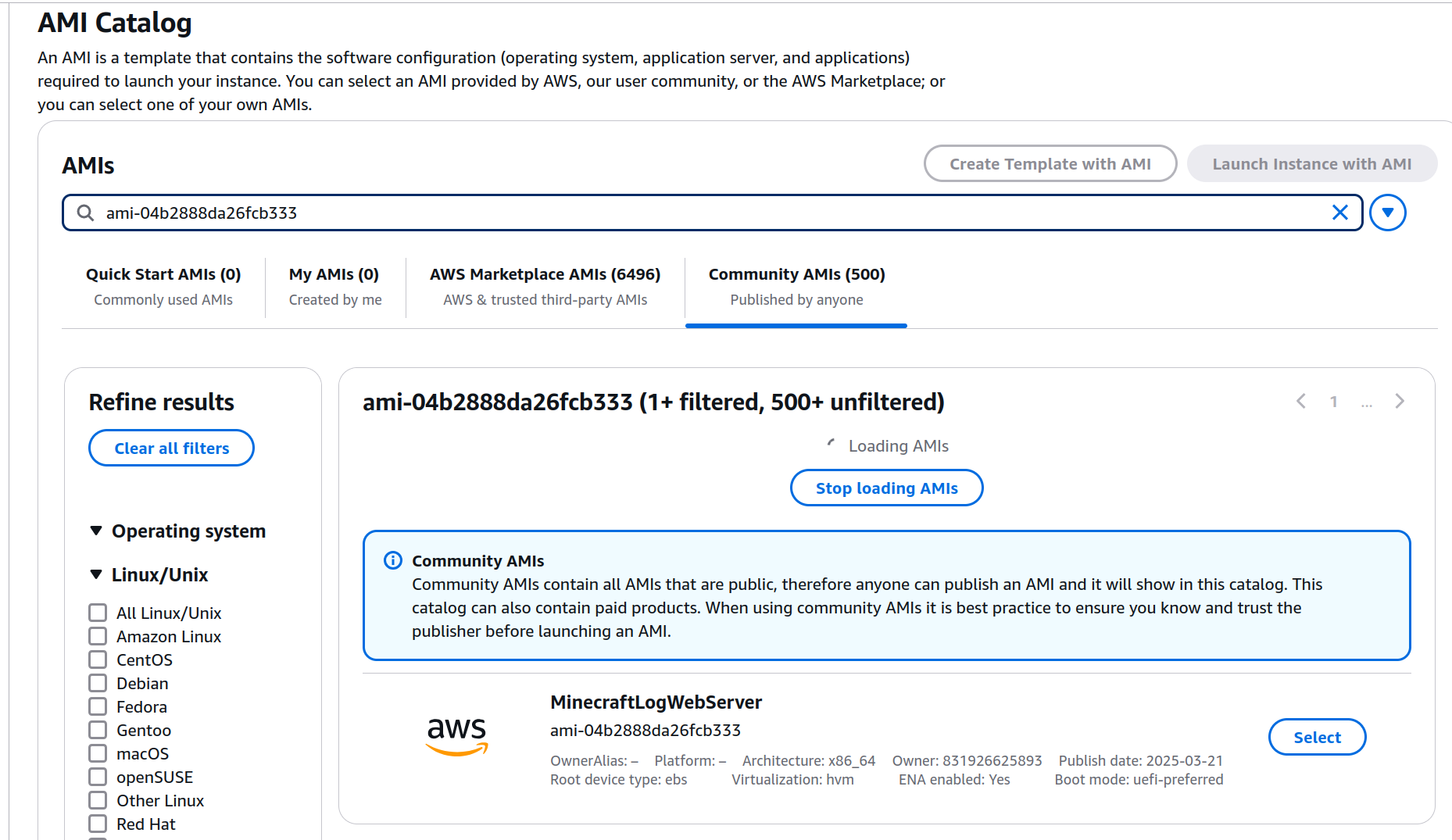




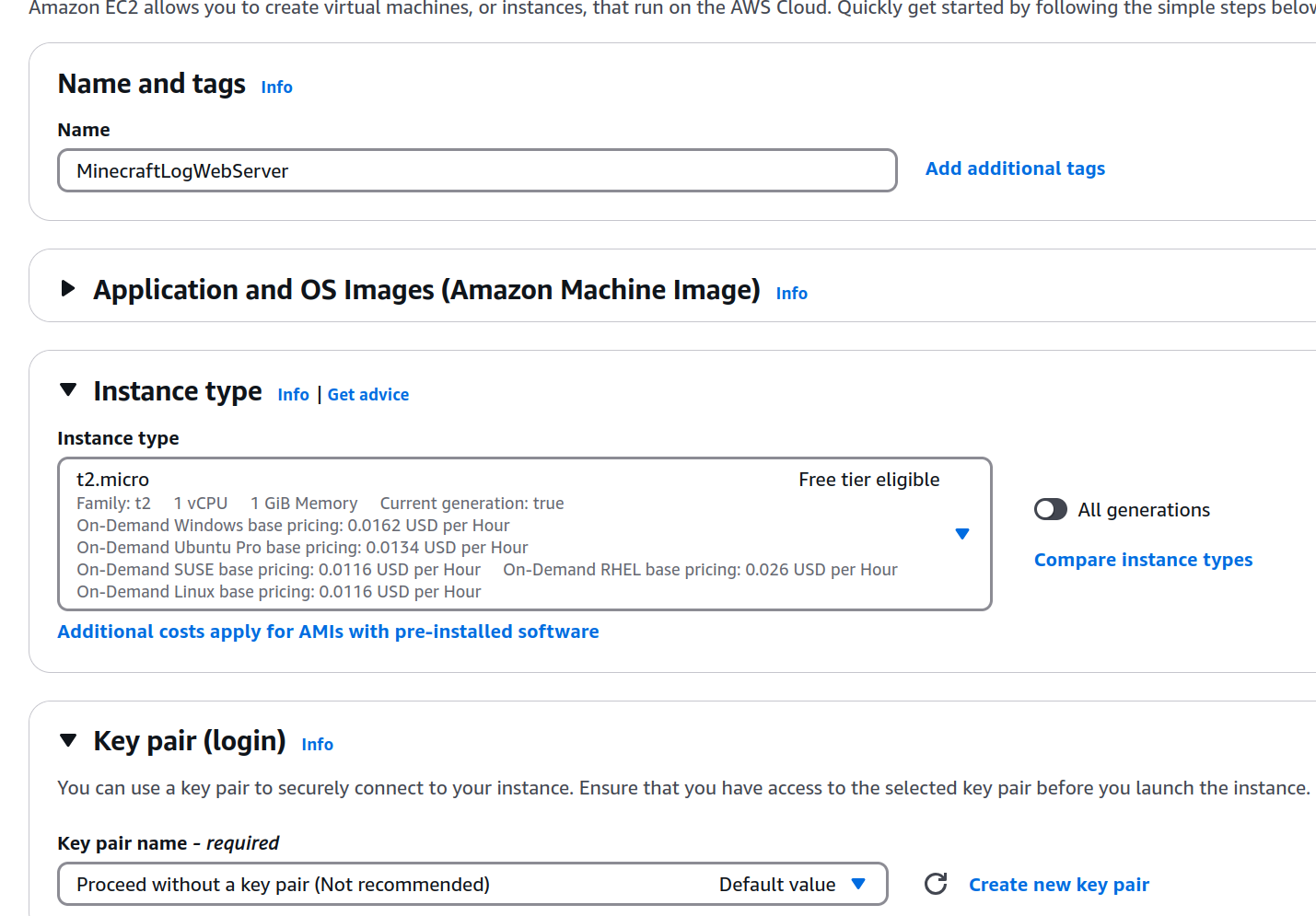
## Creating EC2 Web Server

Go to AMI catalog and search up the AMI ami-04b2888da26fcb333 in the catalog

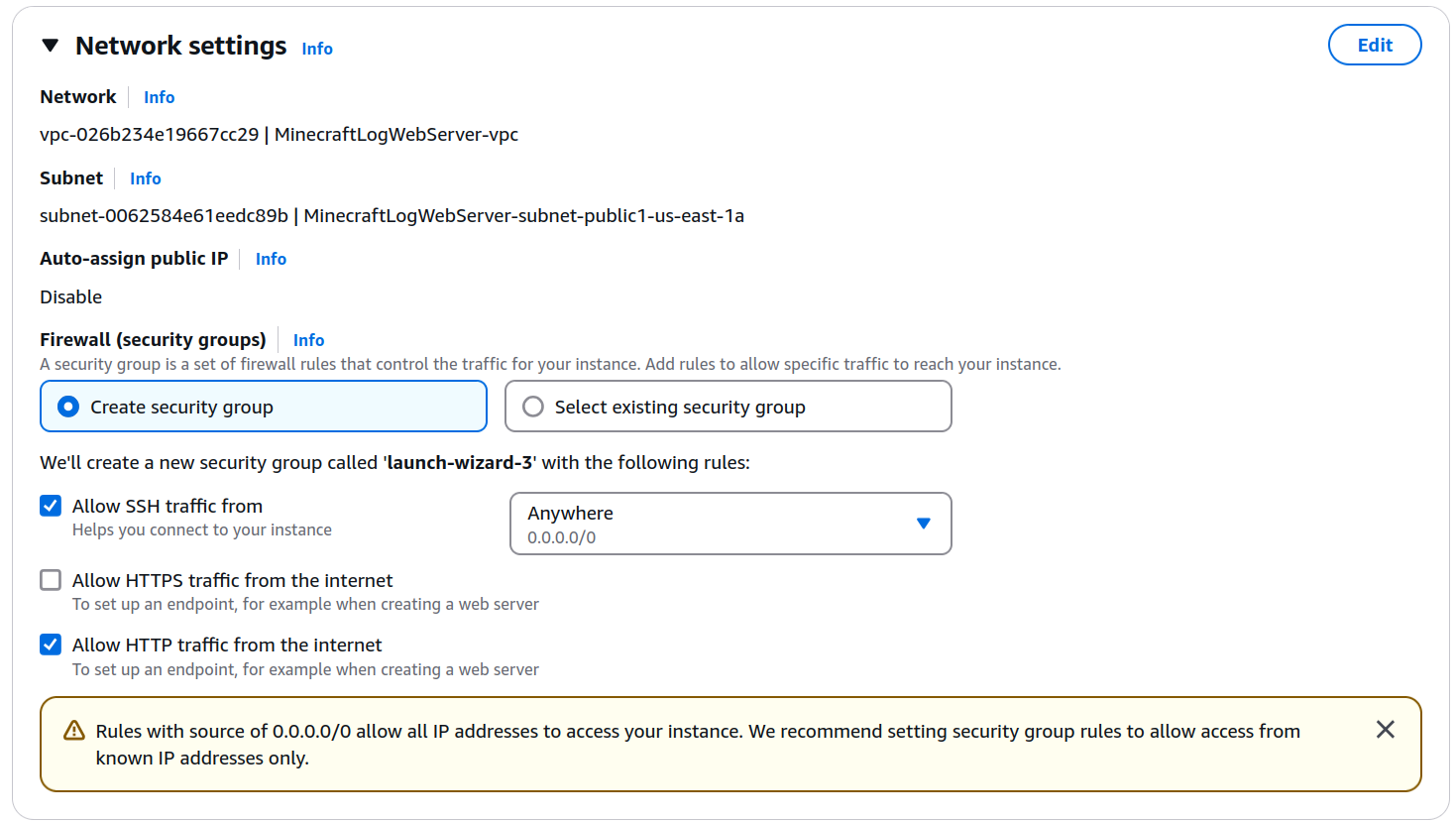




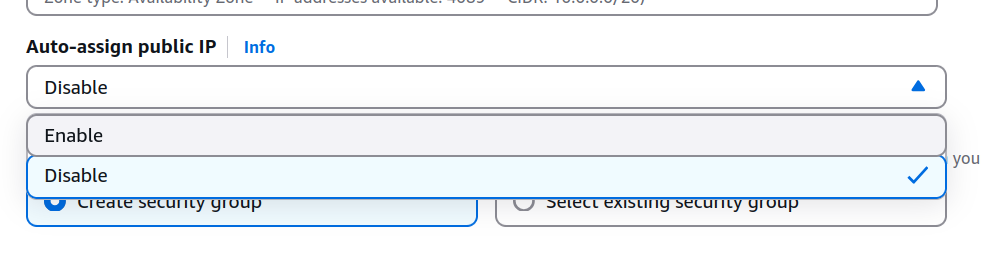
Select it and launch an instance



Allow HTTP Traffic. Then Click edit to further edit network settings

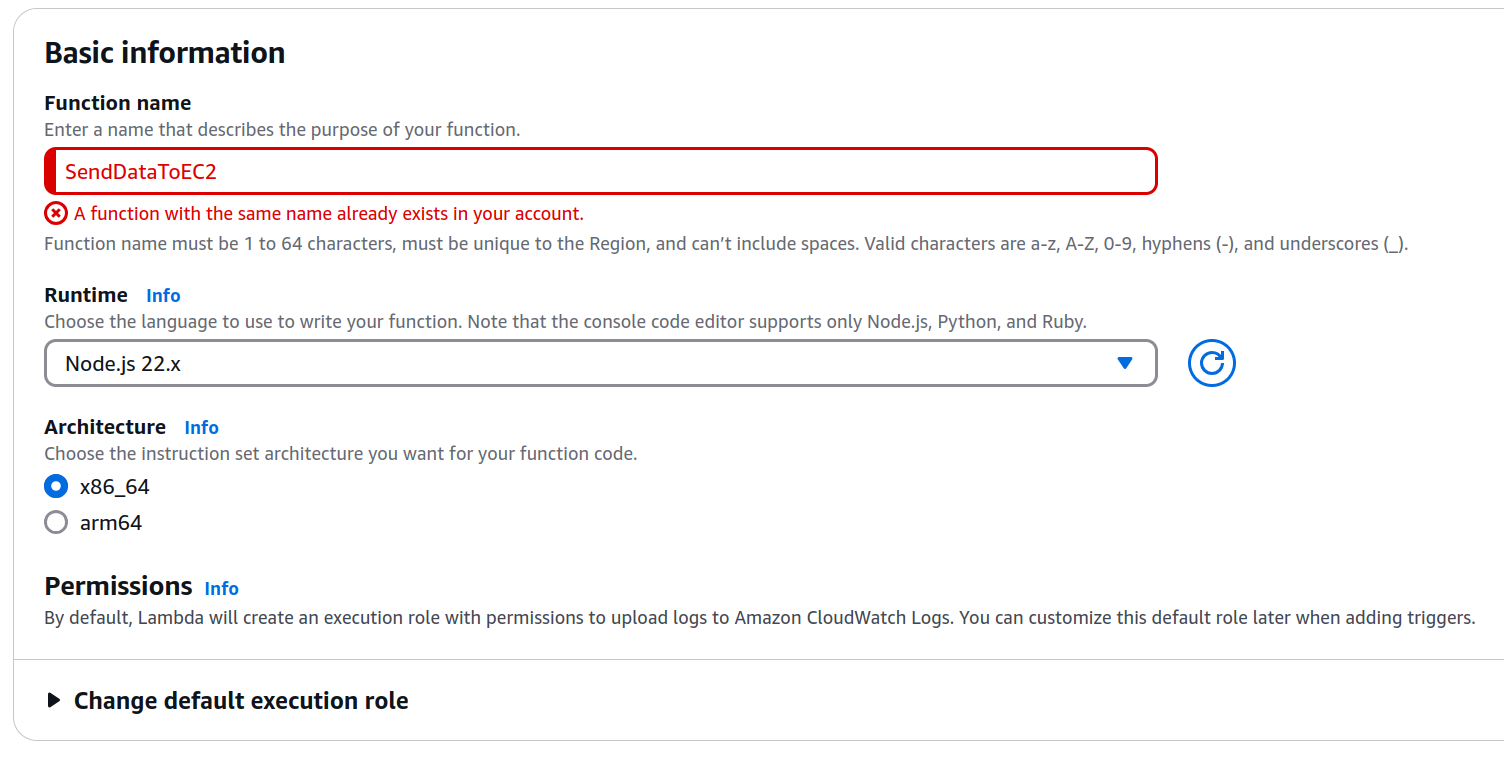


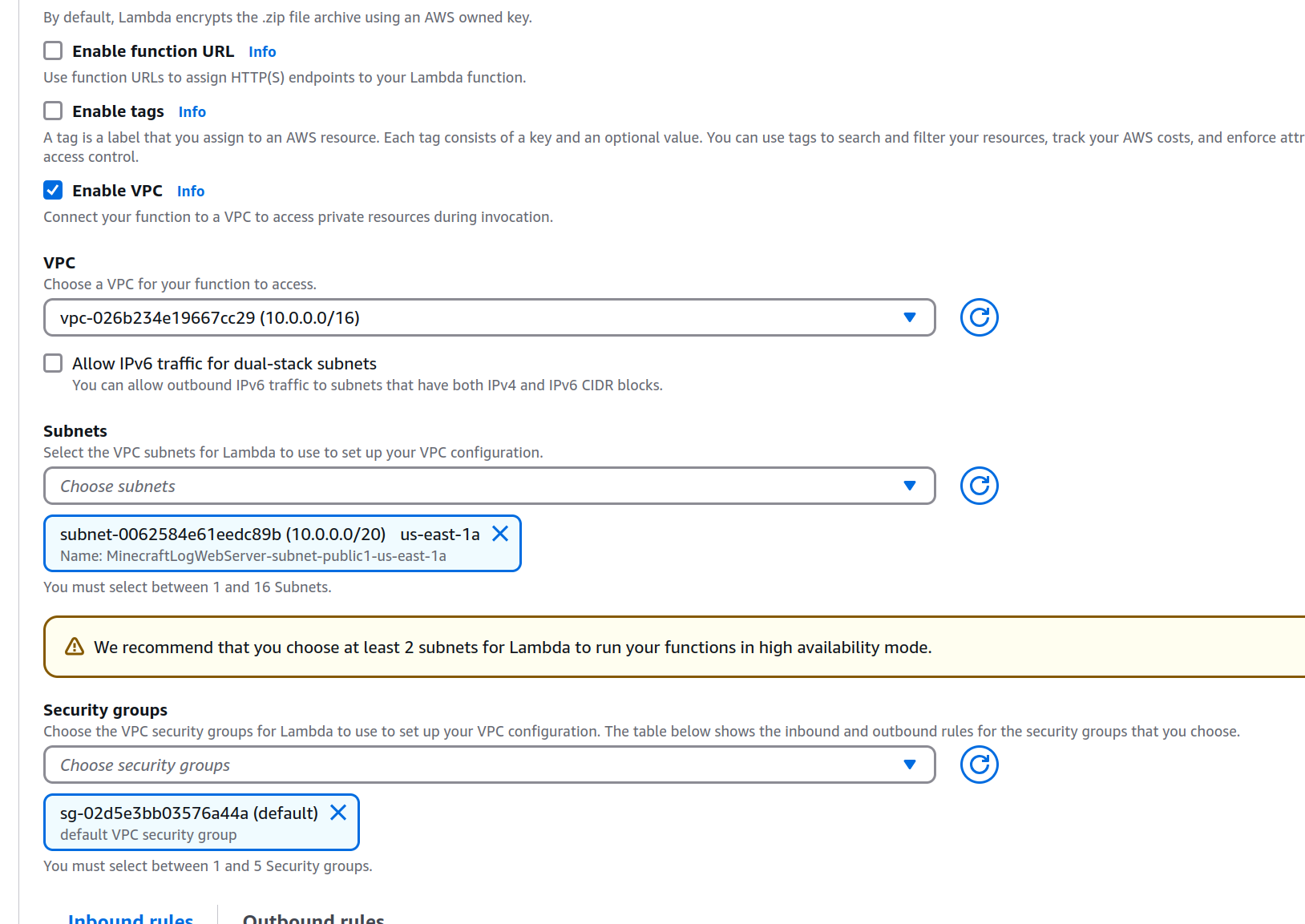
Enable the public ip



Create the instance and finish. Record the internal DNS somewhere of the instance.

## Creating a lambda function to send information to EC2



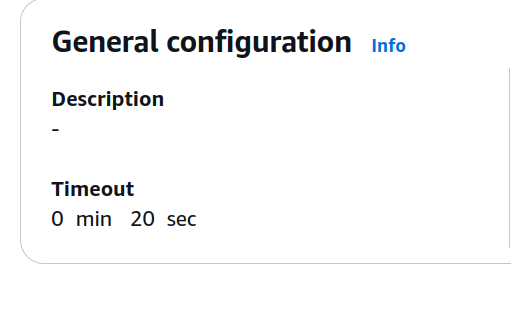


Edit the following variables for your function:

* TargetIP (Should be the internal DNS of the EC2 Instance)

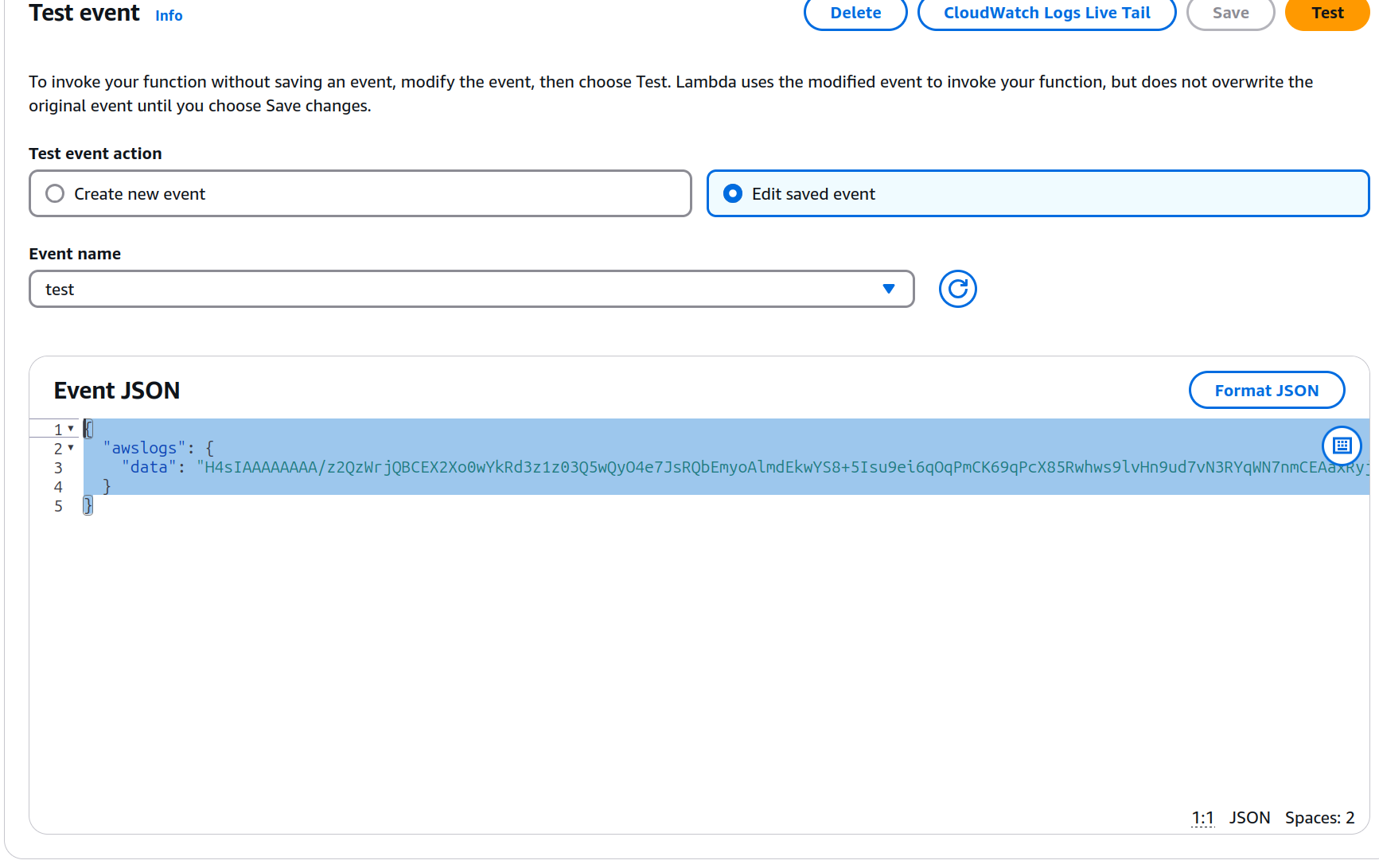
| import \* as https from 'node:https'; import \* as http from 'node:http'; import \* as zlib from 'node:zlib';  /\*\* \* Lambda function that receives CloudWatch events and forwards them to an external API \* \* @param {Object} event - The CloudWatch event object \* @returns {Object} Response object with status code and body \*/ export const handler = async (event) => {  try {  console.log('Received event:', JSON.stringify(event, null, 2));    // Configuration for the external API  const targetIP = 'ip-10-0-1-67.ec2.internal';  const targetPort = 80; // HTTP port  const targetPath = '/log';  const useHttps = false;    // Extract and decode the CloudWatch Logs data  if (event.awslogs && event.awslogs.data) {  // Decode base64 and decompress  const compressed = Buffer.from(event.awslogs.data, 'base64');  const decompressed = zlib.gunzipSync(compressed).toString('utf-8');  const logData = JSON.parse(decompressed);    // Process log events  if (logData.logEvents && logData.logEvents.length > 0) {  for (const logEvent of logData.logEvents) {  // Send each log message as plain text  const logMessage = logEvent.message;    await sendHttpRequest(  targetIP,  targetPort,  targetPath,  logMessage, // Send the raw log message as plain text  useHttps  );  }  }  }    // Return a success response  return {  statusCode: 200,  body: JSON.stringify({  message: 'Successfully forwarded CloudWatch logs'  })  };  } catch (error) {  console.error('Error processing CloudWatch event:', error);    // Return an error response  return {  statusCode: 500,  body: JSON.stringify({  message: 'Error processing CloudWatch event',  errorDetails: error.message  })  };  } };  /\*\* \* Sends an HTTP request to the specified endpoint \* \* @param {string} host - The target host/IP \* @param {number} port - The target port \* @param {string} path - The target path \* @param {string} data - The data to send in the request body \* @param {boolean} useHttps - Whether to use HTTPS or HTTP \* @returns {Promise<Object>} The response data \*/ function sendHttpRequest(host, port, path, data, useHttps = false) {  return new Promise((resolve, reject) => {  // Configure the request options  const options = {  hostname: host,  port: port,  path: path,  method: 'POST',  headers: {  'Content-Type': 'text/plain',  'Content-Length': Buffer.byteLength(data)  }  };    // Choose the appropriate protocol (HTTP or HTTPS)  const protocol = useHttps ? https : http;    // Create and send the request  const req = protocol.request(options, (res) => {  let responseData = '';    // Collect the response data  res.on('data', (chunk) => {  responseData += chunk;  });    // Resolve the promise when the response is complete  res.on('end', () => {  try {  const parsedData = responseData ? JSON.parse(responseData) : {};  resolve({  statusCode: res.statusCode,  headers: res.headers,  body: parsedData  });  } catch (e) {  resolve({  statusCode: res.statusCode,  headers: res.headers,  body: responseData  });  }  });  });    // Handle request errors  req.on('error', (error) => {  reject(error);  });    // Send the request data  req.write(data);  req.end();  }); } |
| --- |

Change the timeout of the function to 20 seconds



## Creating test cases for lambda

Write a test case for your lambda function and test it



Try this test code. This is what cloudwatch will send to the lambda function:

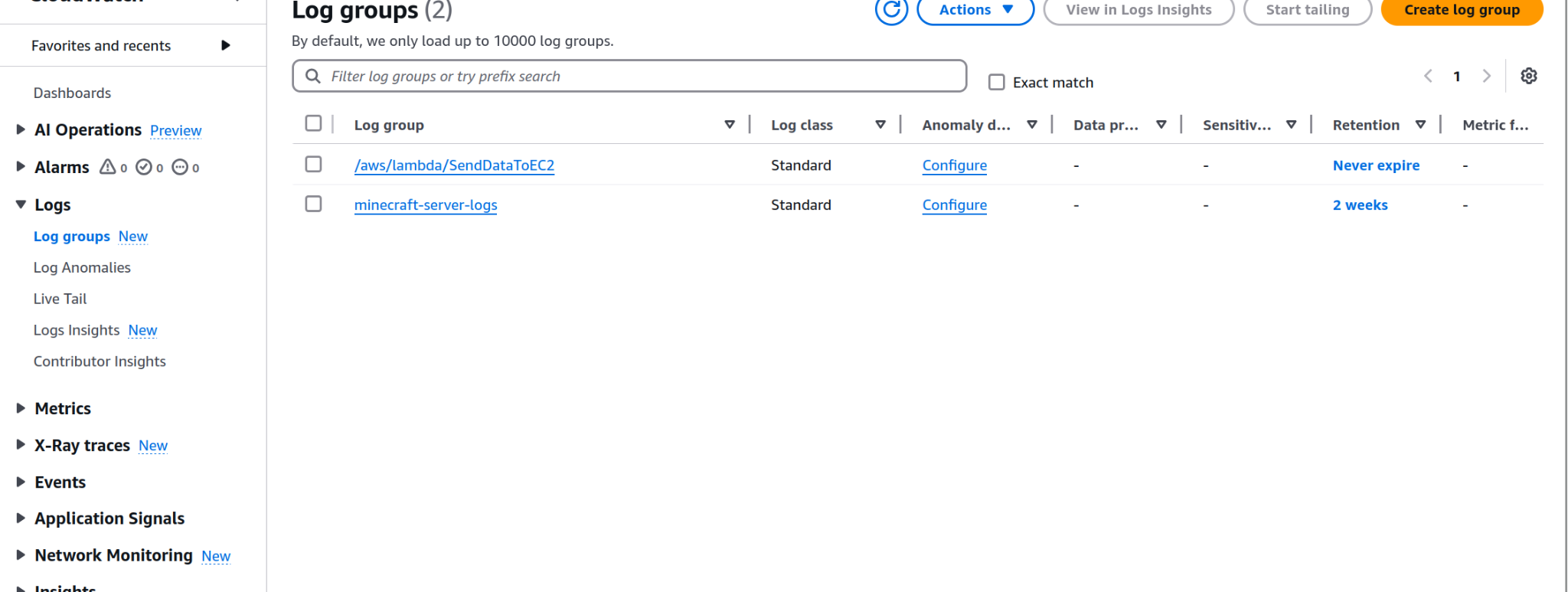
| {  "awslogs": {  "data": "H4sIAAAAAAAA/z2QzWrjQBCEX2Xo0wYkRd3z1z03Q5wQyO4e7JsRQbEmyoAlmdEkwYS8+5Isu9ei6qOqPmCK69qPcX85Rwhws9lvHn9ud7vN3RYqWN7nmCEAaxRyjiyLhgpOy3iXl9czBJjSHI+5fy71GvNbzPVpGde/ll3JsZ8gQKrbwZqoB+be2IgYTf0/BxWsr0/rMadzSct8m04l5hXCAR6WcYwZum/Y9i3O5Uv+gDRAAM1sRbR15AW1dtISkVjRBgVbdi2j11bQMXlyTE7Ie2QLFZQ0xbX00xkCekNWi/fCqKt/X0CAQ+uCkaClU4fd9zBVXnLsh+v7X7e/u6D2y7QcU7nQ4dpjQ8Y31FKDGCxZ4k6dvsoPKs3qPZUXFeeSykWlQVnvtOqL+lEjNUQO0XmxGq2vFJumrVTdNiSO2HhmYbZWtL+Cz+7zDw/NNWGtAQAA"  } } |
| --- |

Now Run it in the test

## Sending Logs from cloudwatch to lambda

### Going to log streams

Go to cloudwatch dashboard and go to log groups. Click on minecraft-server-logs

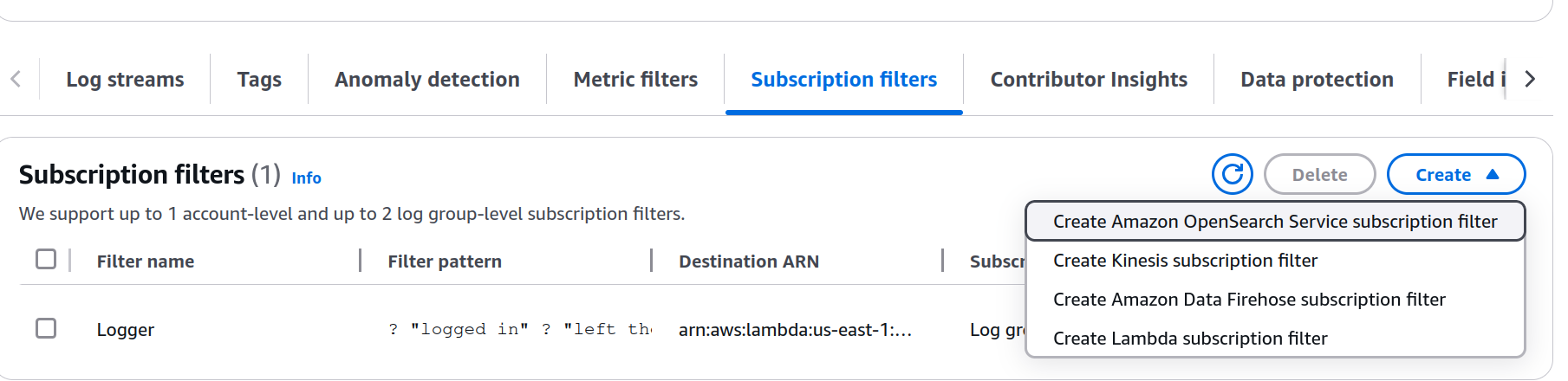


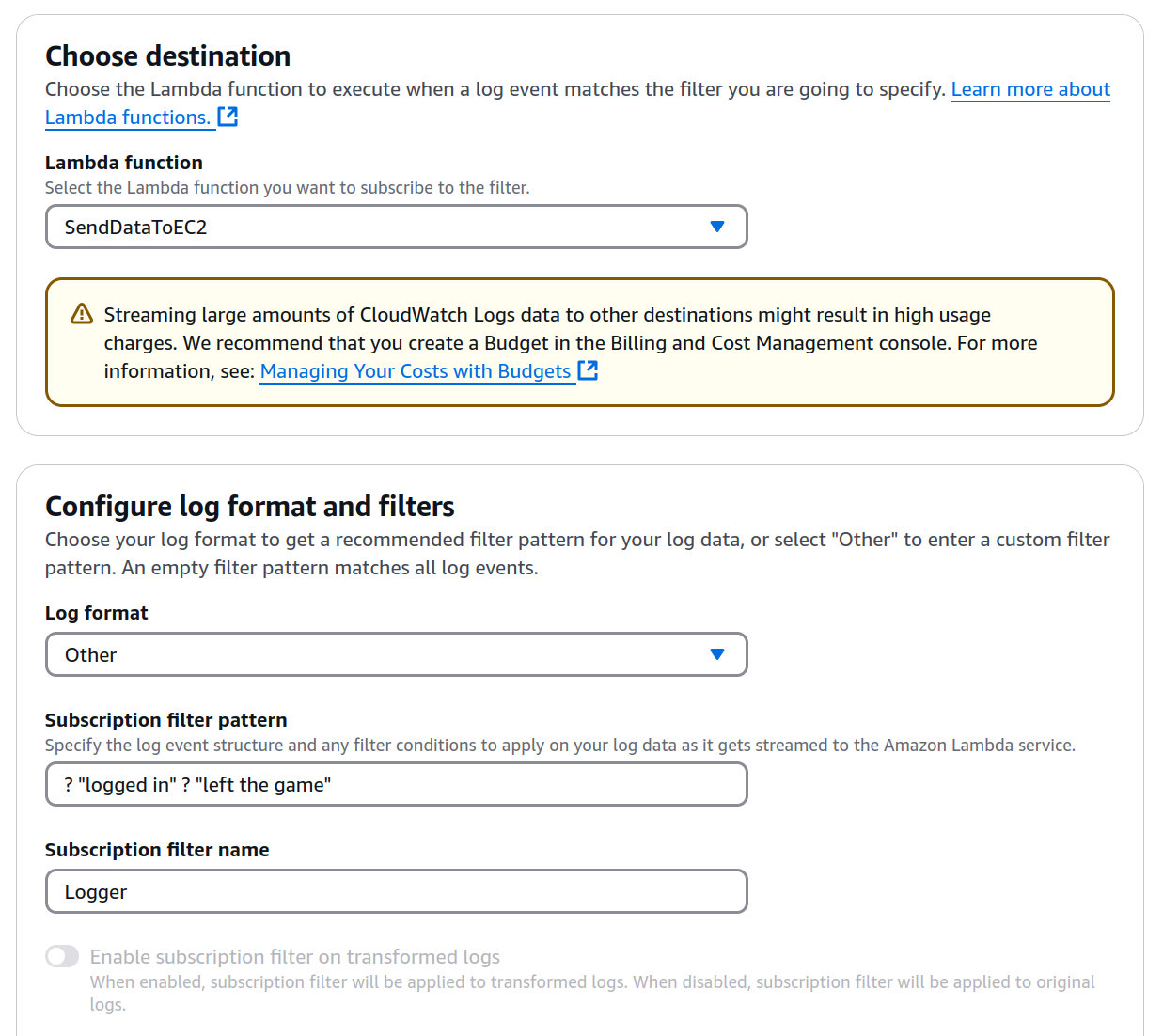
There should now be a logstream called {lightsail instance}-minecraft. Make sure its there.

### Creating a subscription filter

The subscription filter will send the data to the lambda function itself

Go to subscription filters and create a lambda filter





Test the pattern out with this

| 2025-03-22T06:15:36.977Z [06:11:30] [User Authenticator #1/INFO]: UUID of player Tomocity2 is 787b9319-d89f-4498-b98e-d7557bc10ed6 2025-03-22T06:15:36.977Z [06:11:30] [Server thread/INFO]: Tomocity2[/71.247.203.11:44402] logged in with entity id 134 at (21.180475311874886, 75.0, 21.98323143604005) 2025-03-22T06:15:42.188Z [06:11:30] [Server thread/INFO]: Tomocity2 joined the game 2025-03-22T06:15:50.264Z [06:15:50] [Server thread/INFO]: Tomocity2 lost connection: Disconnected 2025-03-22T06:15:52.521Z [06:15:50] [Server thread/INFO]: Tomocity2 left the game 2025-03-22T06:15:53.023Z [06:15:52] [User Authenticator #2/INFO]: UUID of player Tomocity2 is 787b9319-d89f-4498-b98e-d7557bc10ed6 2025-03-22T06:15:53.023Z [06:15:52] [Server thread/INFO]: Tomocity2[/71.247.203.11:55440] logged in with entity id 804 at (7.638552196096684, 72.0, 48.305932660813625) 2025-03-22T06:15:57.977Z [06:15:52] [Server thread/INFO]: Tomocity2 joined the game |
| --- |

start streaming

## Start the python web server

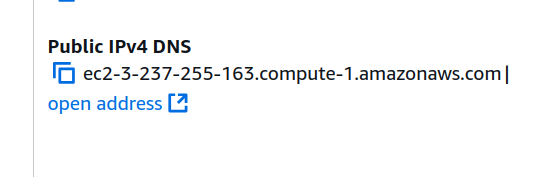
Connect to your EC2 and do

python3 server.py

## All done!

### Time to try it out

Connect to the public EC2 url. Don’t hit open address, because it will use https. Instead copy the public dns and do [http://{dns](about:blank)} in your browser



### Now try to connect to the server

If it works, it should show up in 3-10 seconds