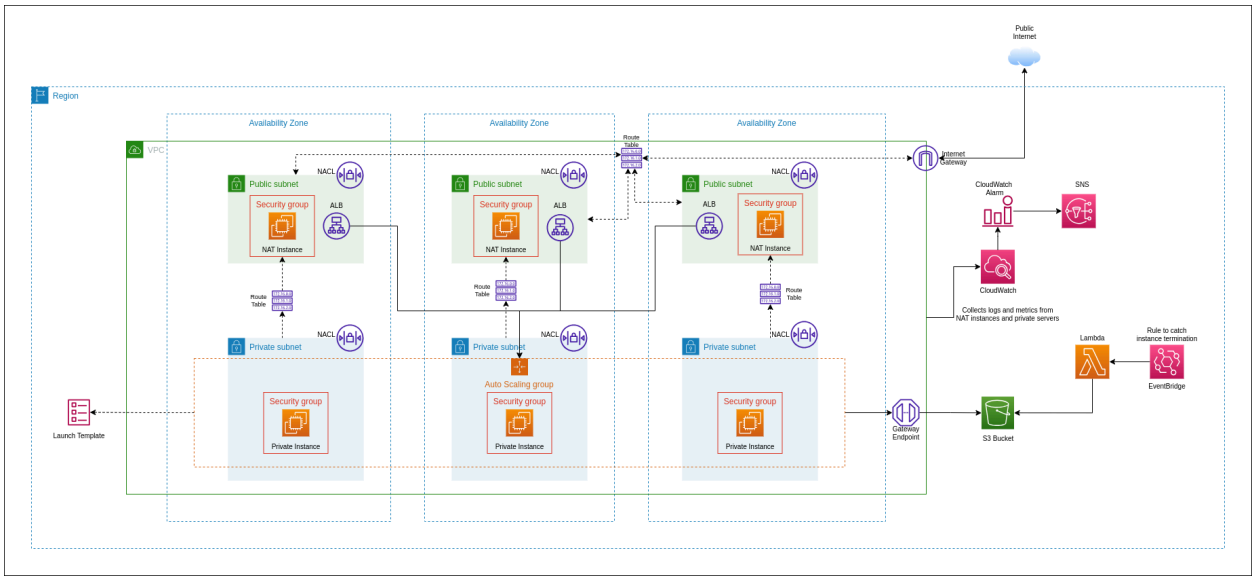
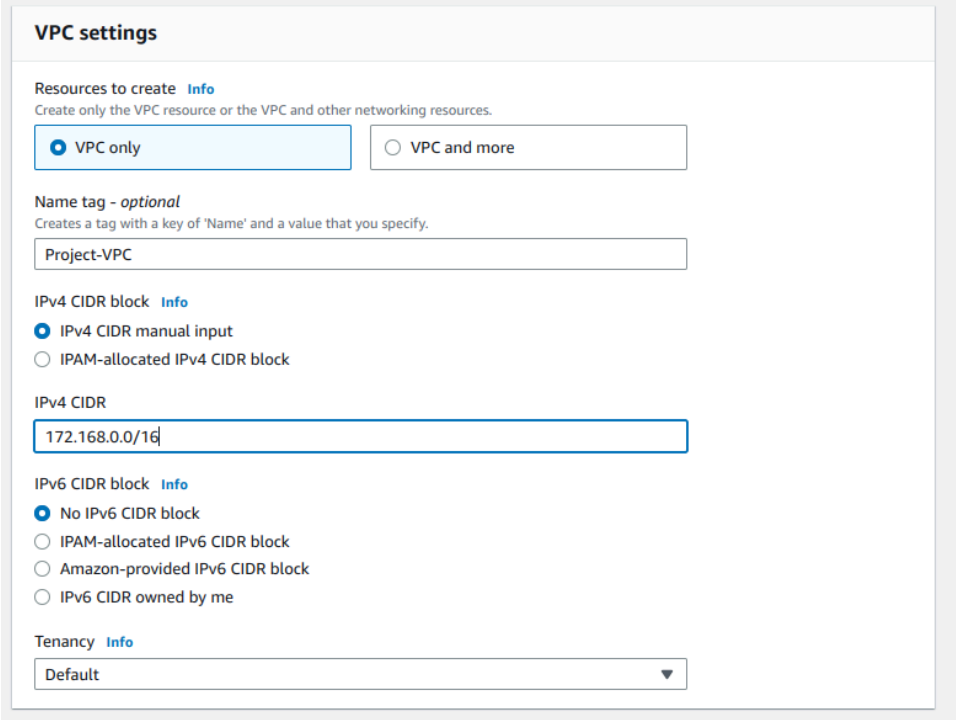
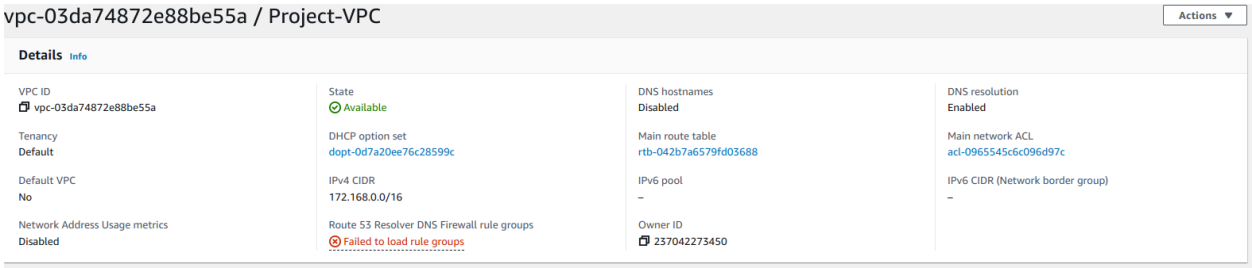
Demo project for AWS



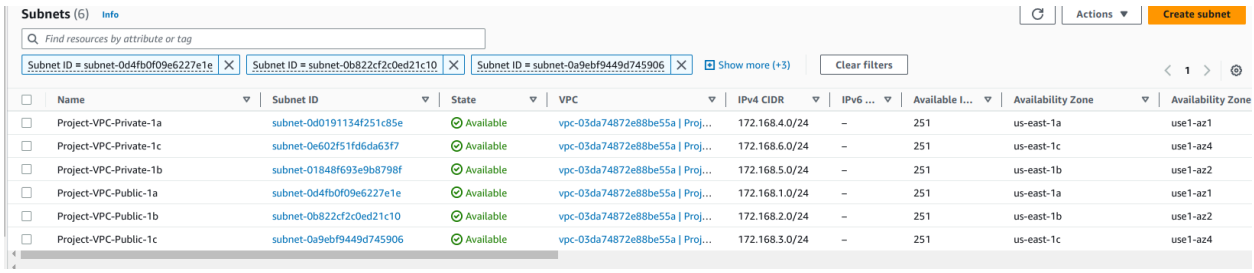
Architecture Diagram for the Solution

Step 1 - Create a VPC with the CIDR range of “172.168.0.0/16”

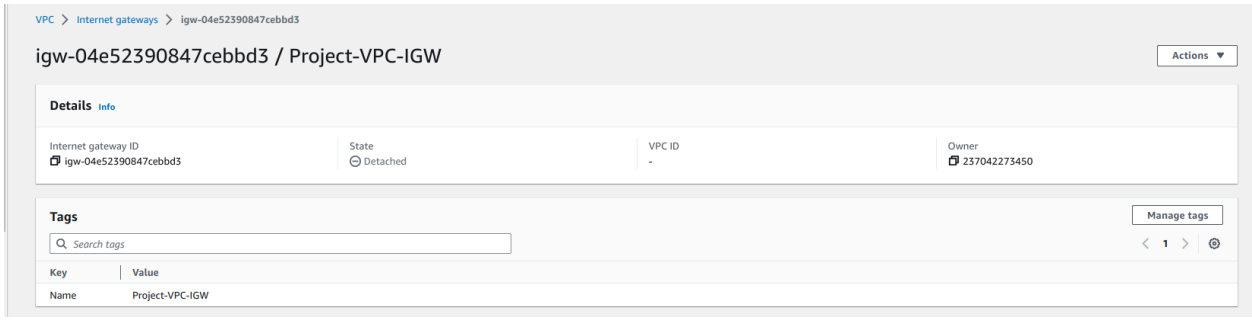


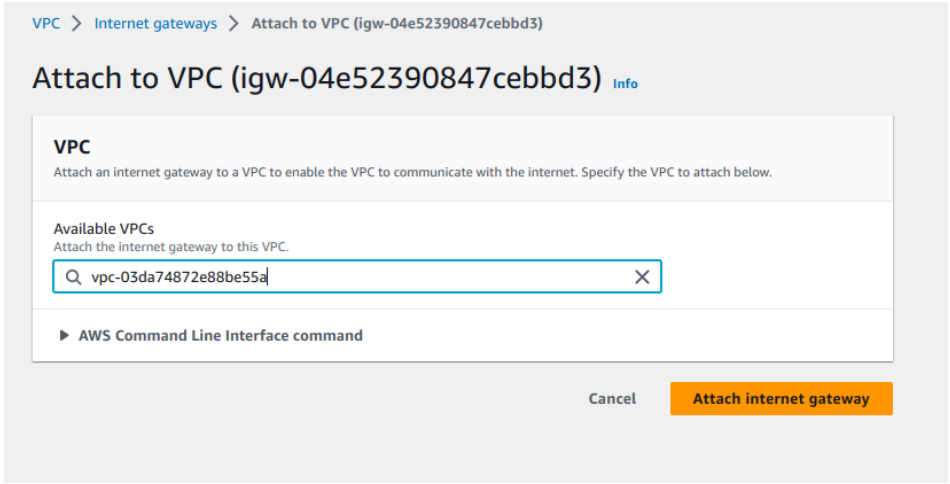


Step 2 - Create three public and three private subnets across three availability zones.

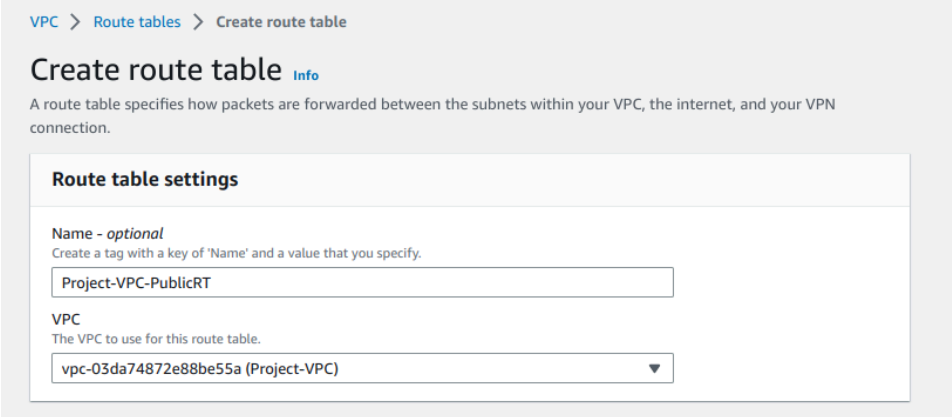


Step 3 - Create an Internet gateway and attach it to the VPC.

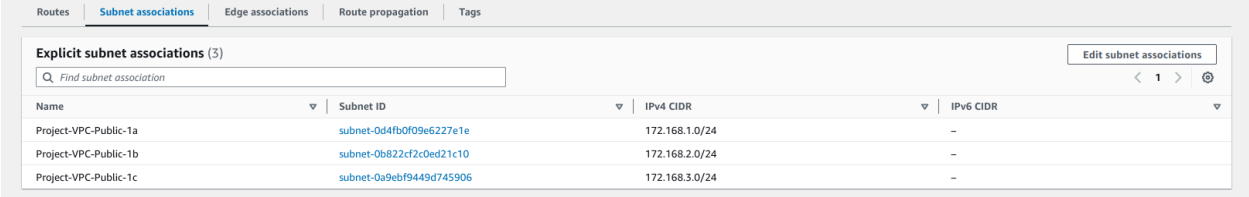




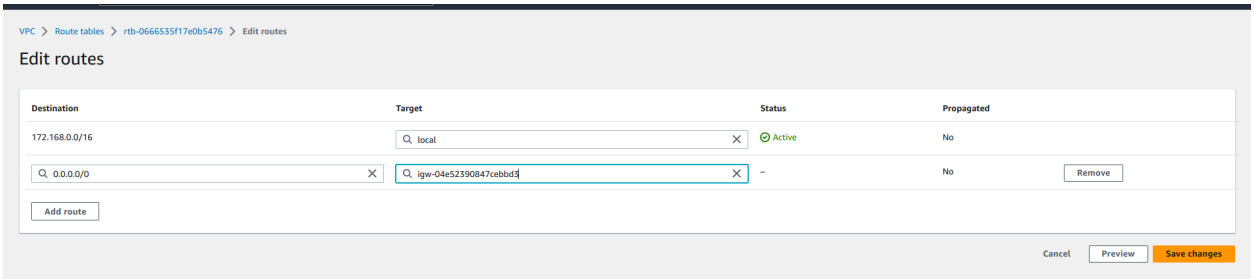
Step 4 - Create a route table for the public subnets.



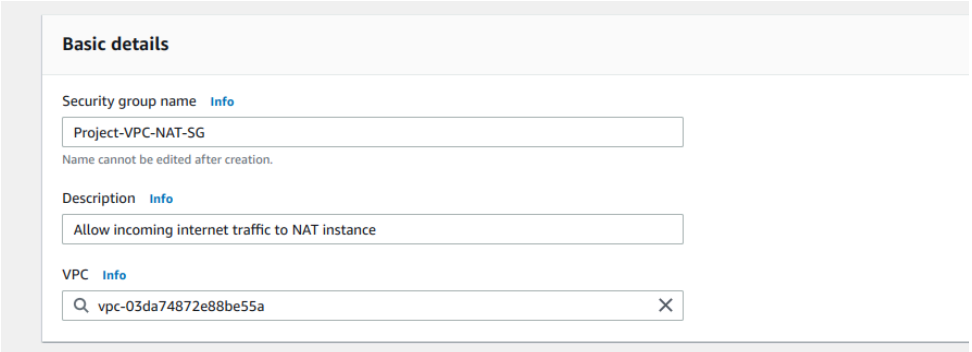
Edit the subnet association for the public route table to attach the three public subnets.



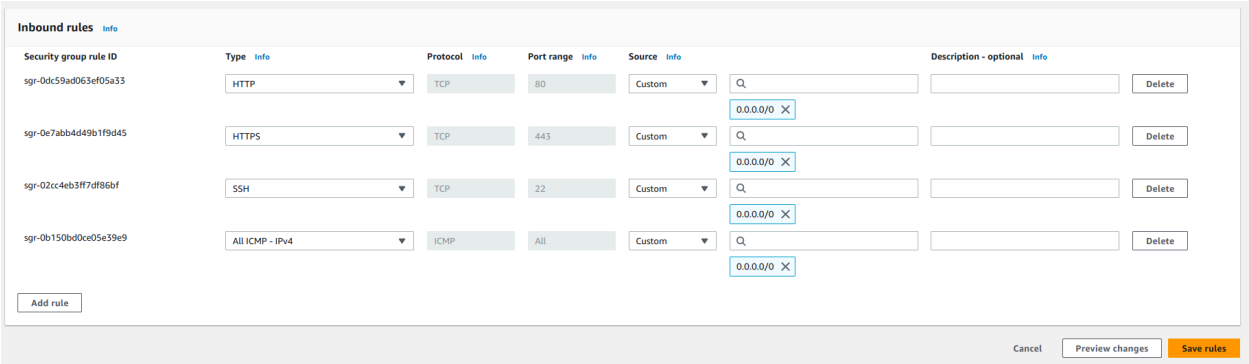
Edit the route of the table to forward the internet faced traffic to the internet gateway as a target.



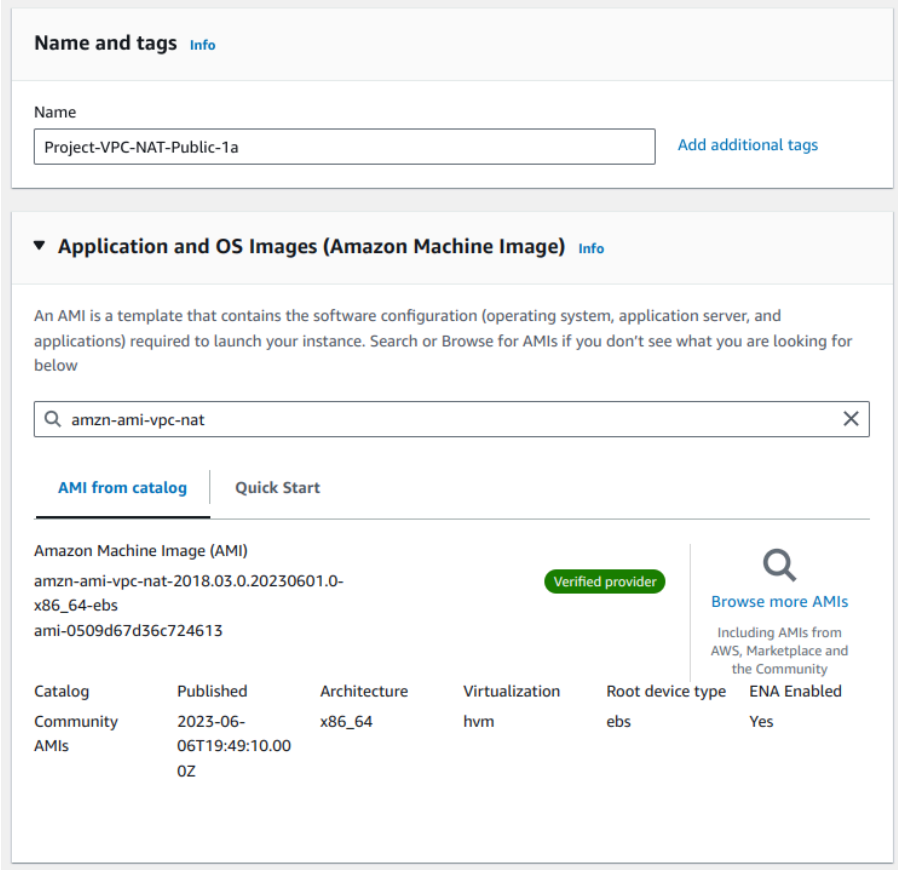
Step 5 - Create a Security group for the NAT instance, which will be created in the public subnet.

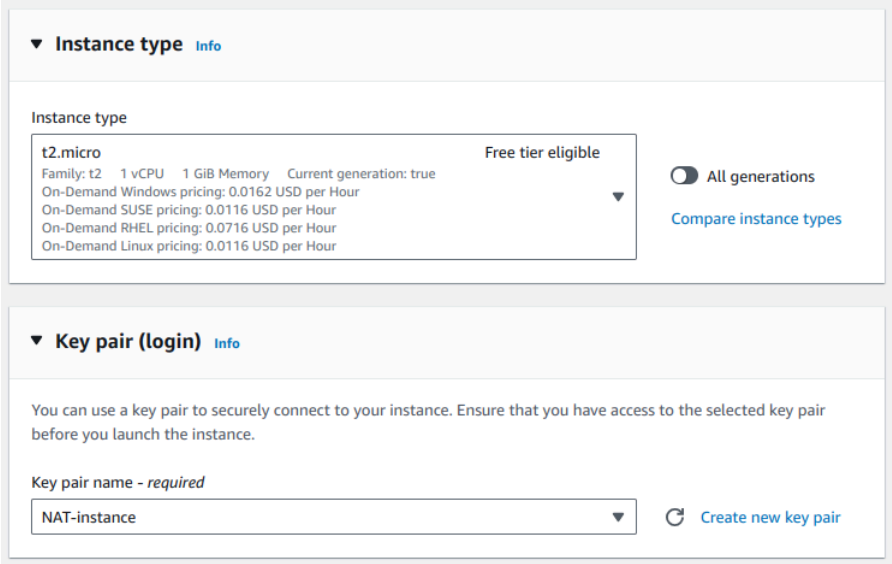


The security group allows incoming traffic for HTTP, HTTPS, SSH, and all ICMP from anywhere.

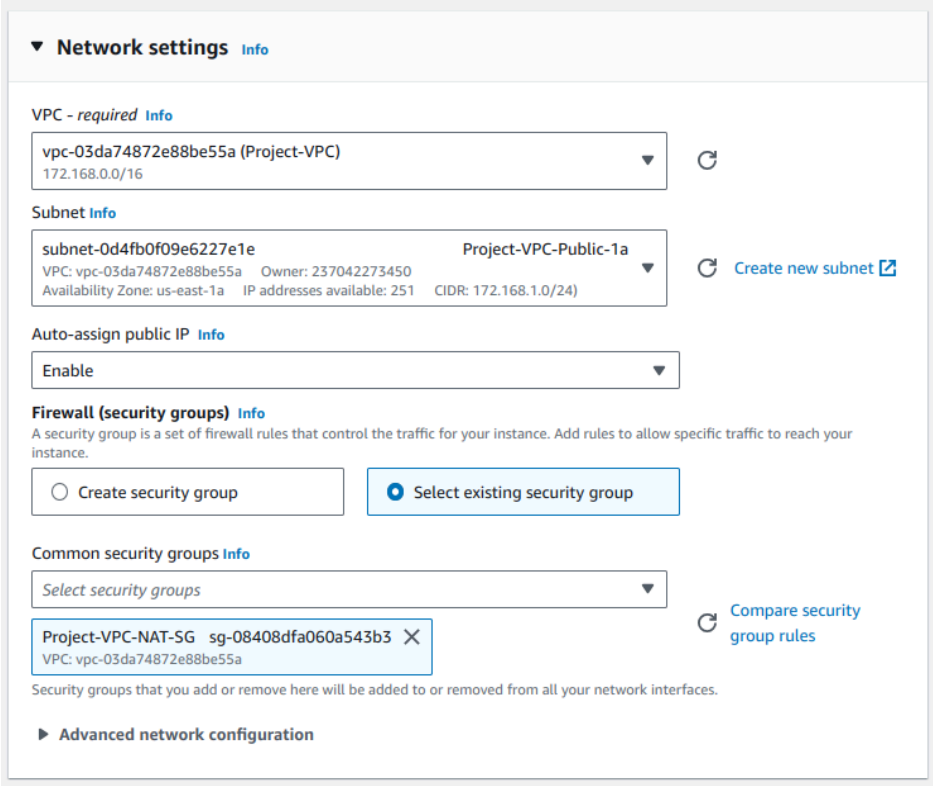


Step 6 - Launch an EC2 instance in one of the public subnets. Choose the AWS managed NAT AMI.

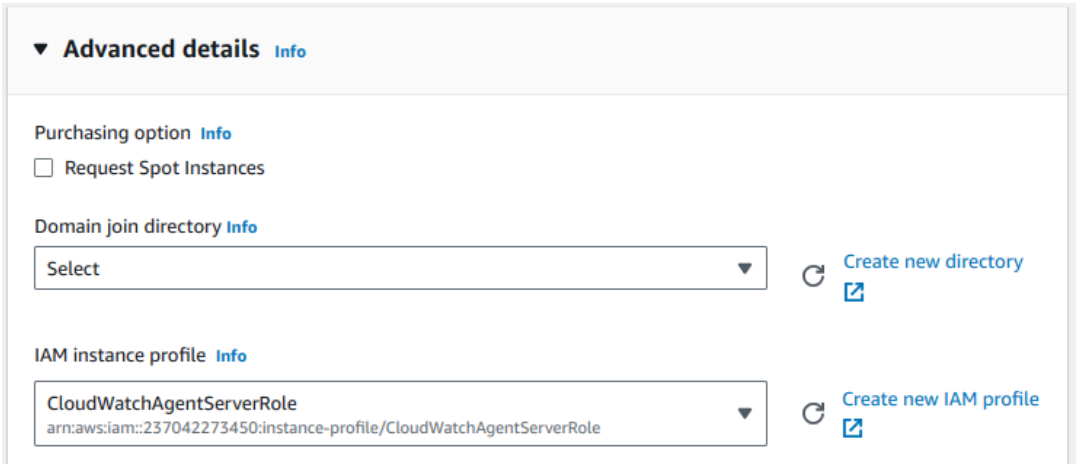




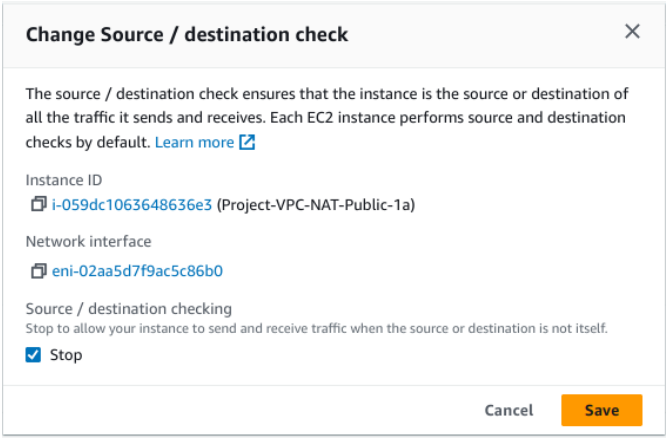
Attach the previously created security group to the instance.



Attach the “CloudWatchAgentServerRole” role to the instance profile, it will allow the instance to send logs and metrics to cloudwatch using cloudwatch agent.



After the NAT instance is launched, stop the source/destination checking on the instance. It will allow the instance to send and receive the traffic when it is not the source and destination for that traffic.

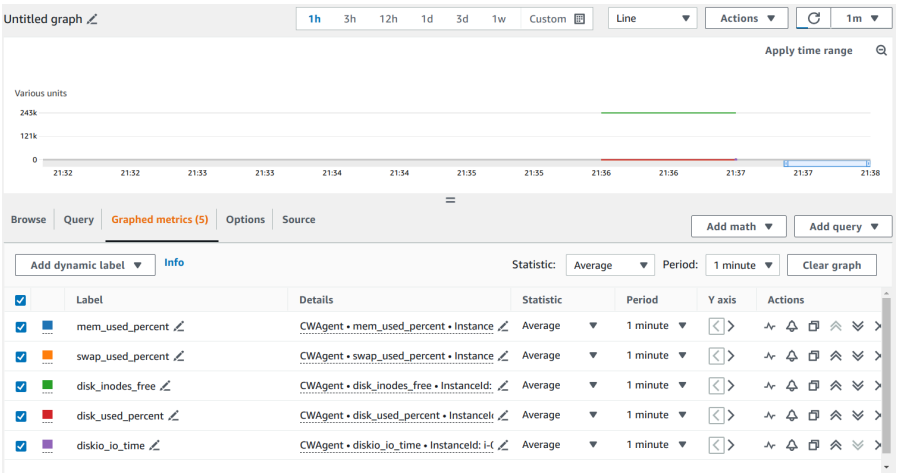


Step 7 - SSH into the NAT instance and configure the CloudWatch agent in it to send CPU, disk, and memory metrics to cloudwatch.

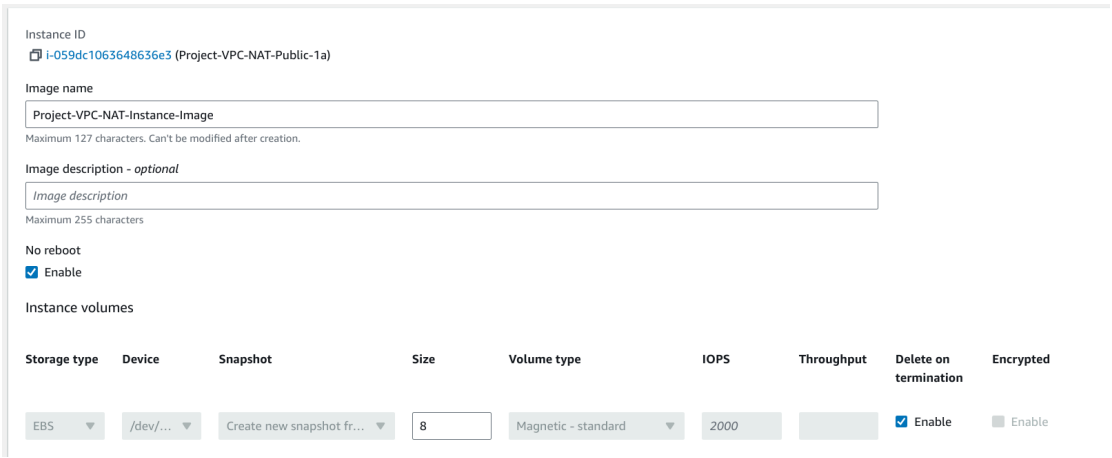
```
[ec2-user@ip-172-168-1-147 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
****** processing amazon-cloudwatch-agent ******
I! Trying to detect region from ec2 D! [EC2] Found active network interface Successfully fetched the config and saved in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp
Start configuration validation...
2023/08/07 16:06:37 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp ...
2023/08/07 16:06:37 I! Valid Json input schema.
2023/08/07 16:06:37 D! ec2tagger processor required because append_dimensions is set
2023/08/07 16:06:37 D! delta processor required because metrics with diskio or net are set
2023/08/07 16:06:37 D! ec2tagger processor required because append_dimensions is set
2023/08/07 16:06:37 Configuration validation first phase succeeded
I! Detecting run_as_user...
I! Trying to detect region from ec2
D! [EC2] Found active network interface
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -schematest -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml
Configuration validation second phase succeeded
Configuration validation succeeded
amazon-cloudwatch-agent stop/waiting
amazon-cloudwatch-agent start/running, process 19291
[ec2-user@ip-172-168-1-147 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a status
{
  "status": "running",
  "starttime": "2023-08-07T16:06:38+0000",
  "configstatus": "configured",
  "version": "1.300026.1b168"
}
```

The above snapshot shows that the cloudwatch agent is running in the instance.

It is seen that the metrics of the NAT instance are available in the cloudwatch.

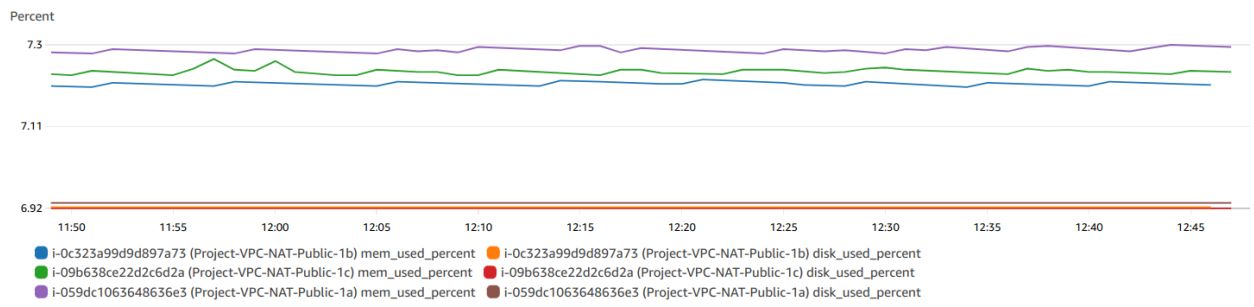


Step 8 - Create AMI of the above instance and launch NAT instances in other two subnets.

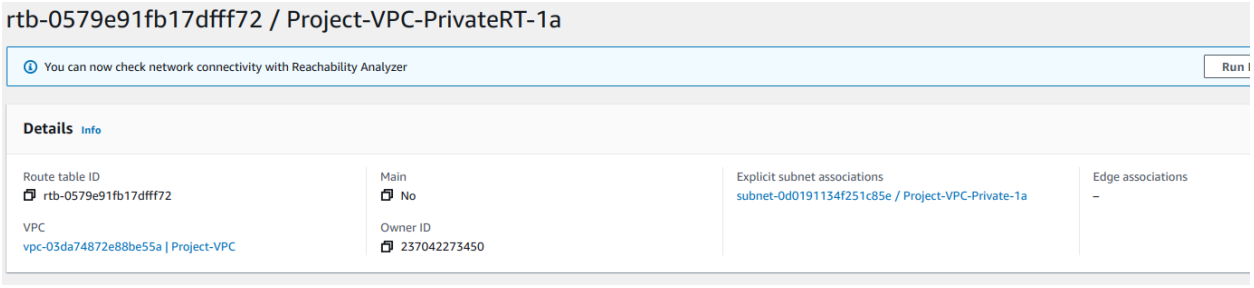


Launch two more NAT instances using the above AMI in different subnets.

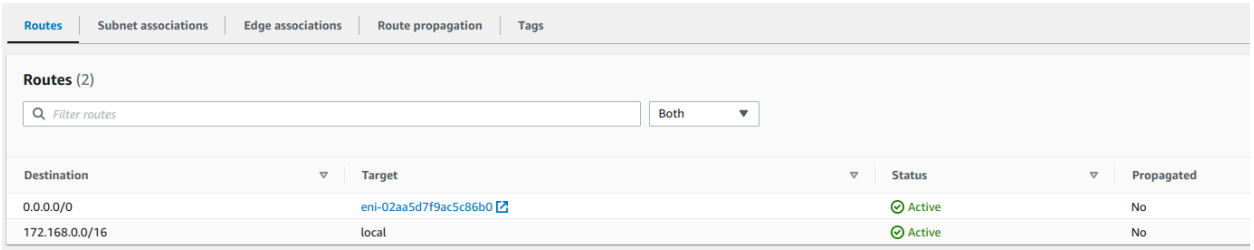
The snapshot shows that the metrics from all the three NAT instances are available in cloudwatch.

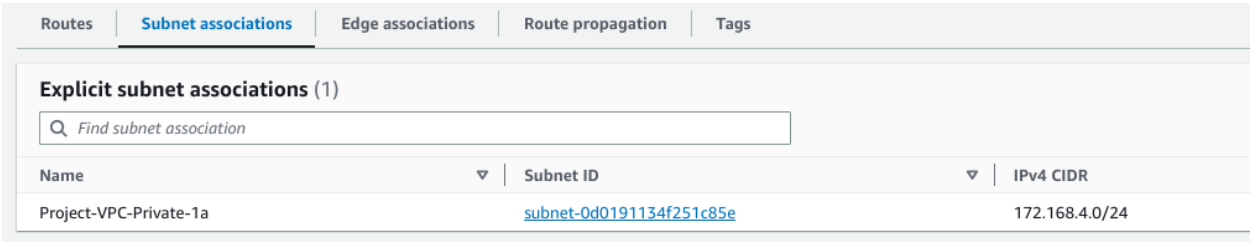


Step 9 - Create a route table for one of the private subnets.



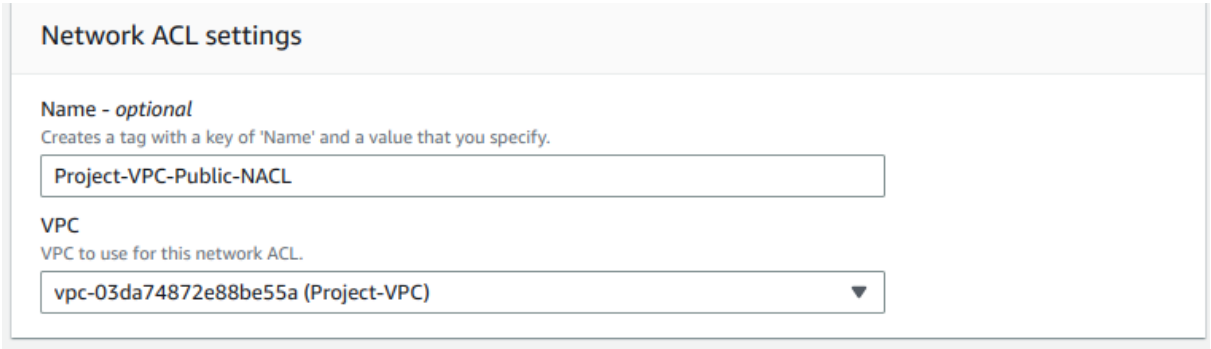
Edit the route to forward the internet facing traffic to the NAT instance hosted in the public subnet of the same AZ. This will allow the instances in the private subnet to access the public internet for updates and downloads.



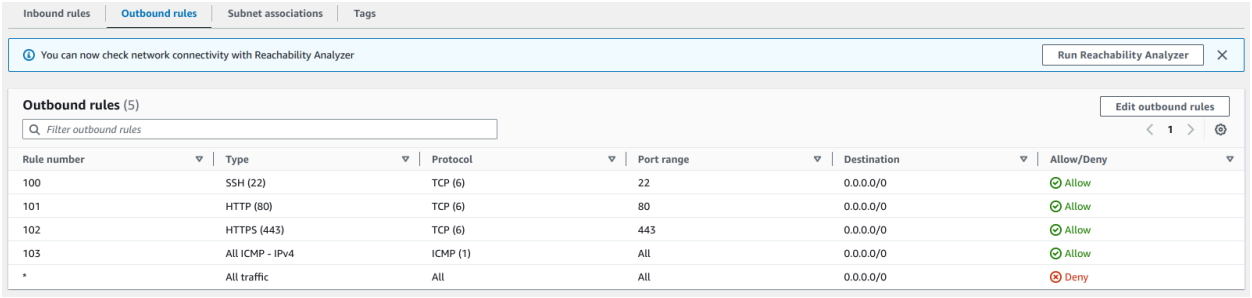


Create two more route tables for the other two private subnets and edit their route and subnet association. The route table forwards the internet facing traffic from the instances in those subnets to the NAT instances in the public subnet of the same AZ.

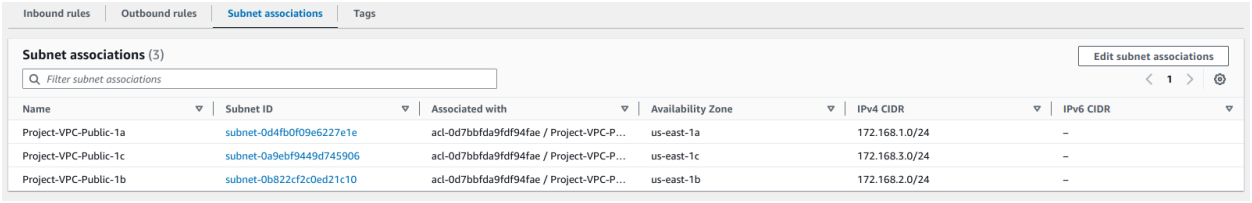
Step 10 - Create NACL for the public subnets.



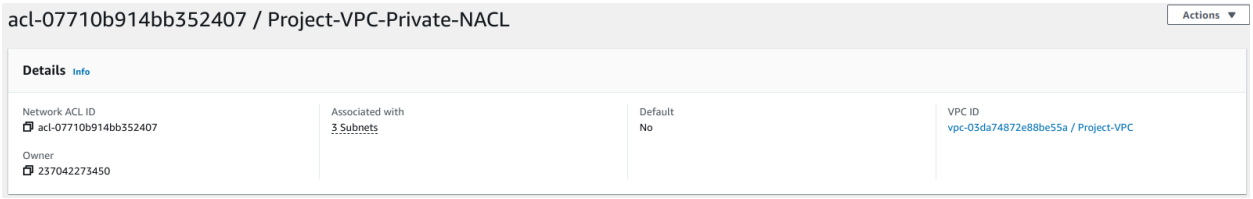
Allow inbound and outbound rules to allow only SSH, HTTP, HTTPS, and all ICMP traffic.

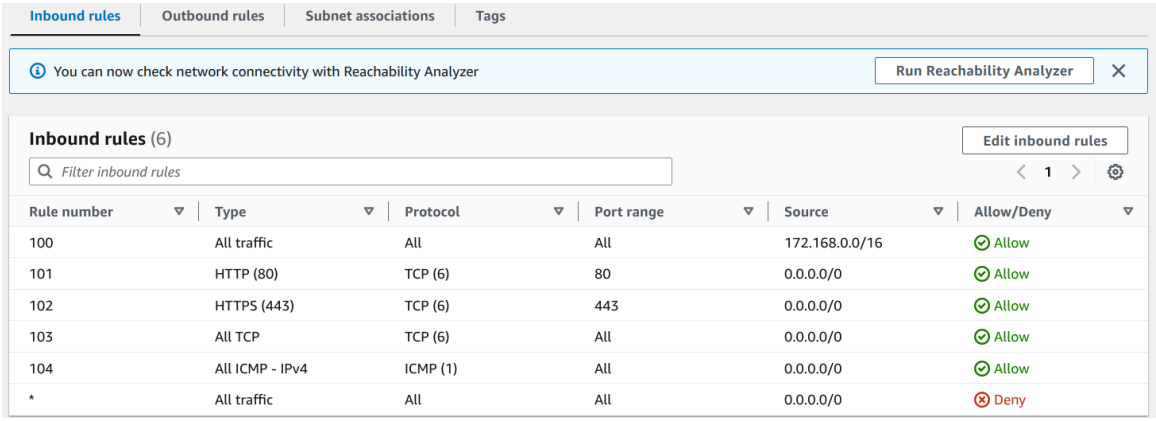


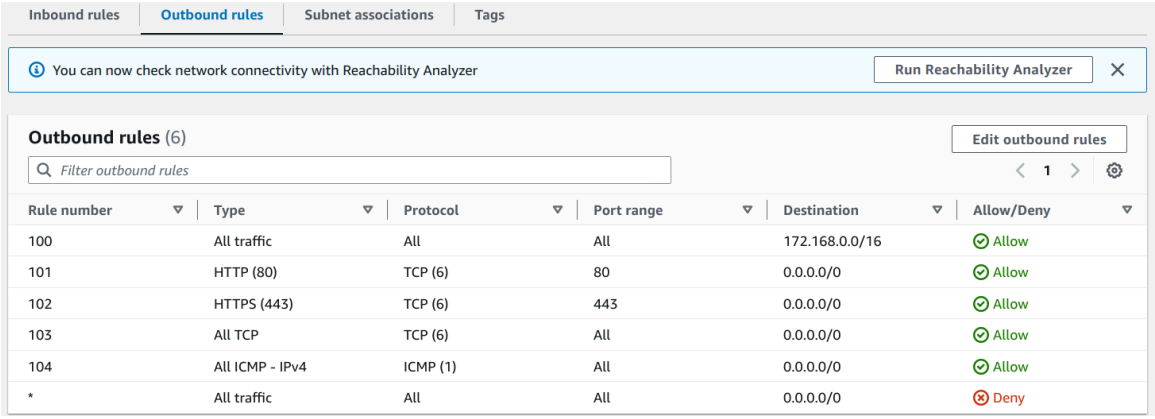
Associate the public subnets to the NACL.

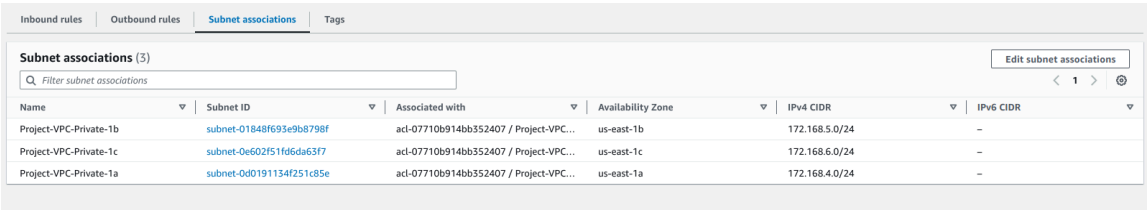


Step 11 - Create NACL for the private subnets.





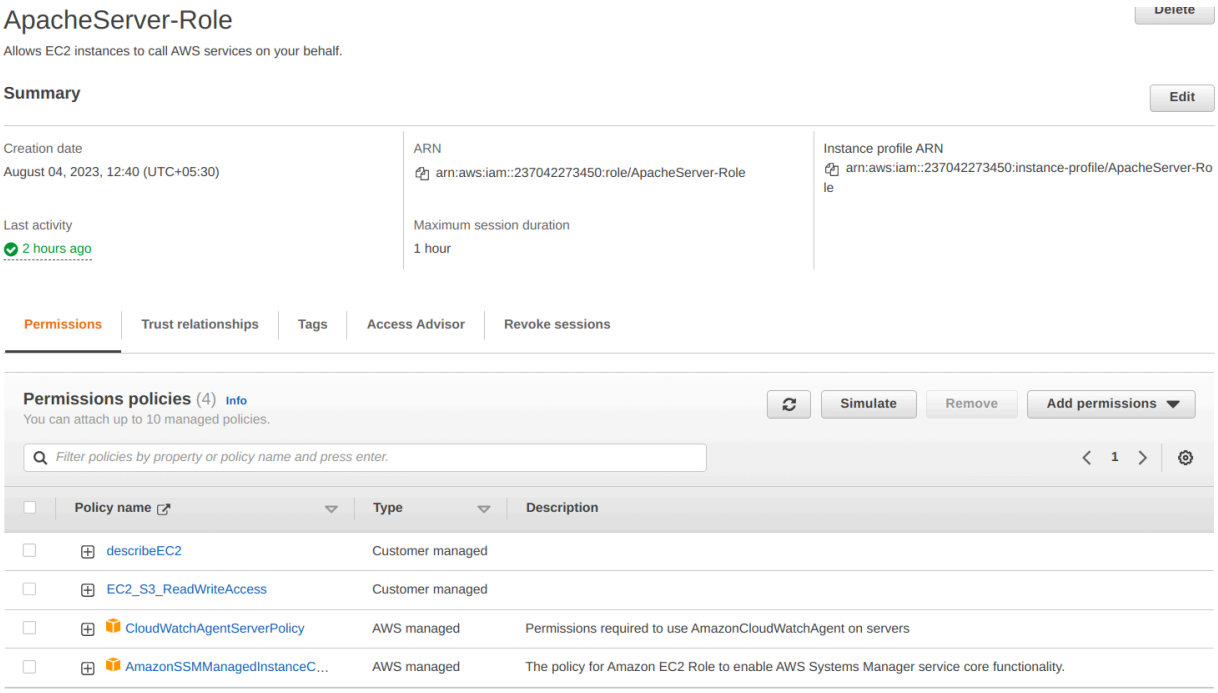




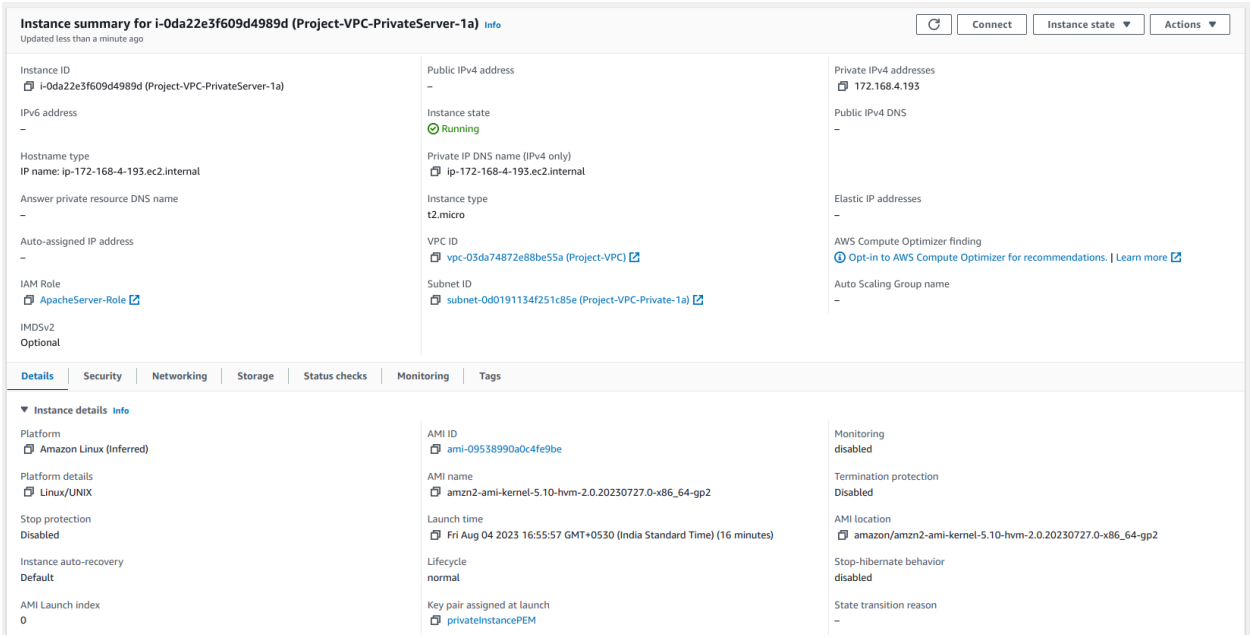
==

Step 12 - Create an IAM role for the private instances.

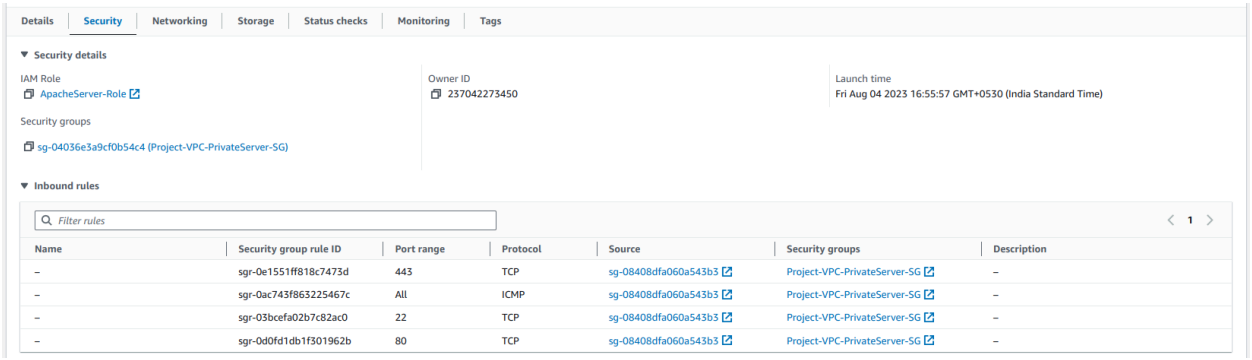
The policies attached to the roles allows the instances to describe other instances, perform actions with S3, send logs to cloudwatch, and SSM connection.

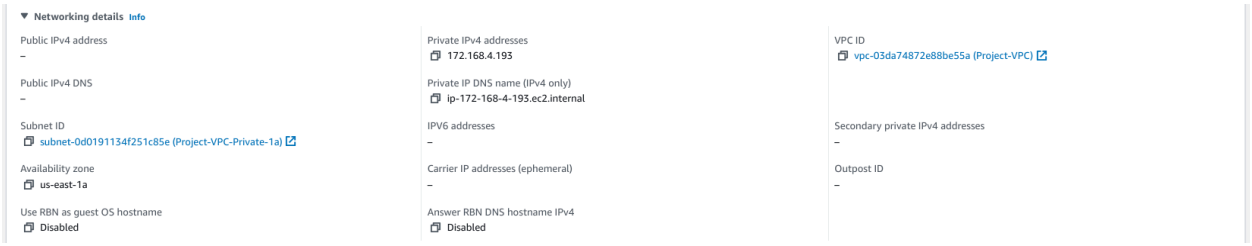


Step 13 - Launch EC2 instance in one of the private subnets.



The security group attached to the private instance allows incoming HTTP, ICMp, SSH, and HTTPS traffic from the security group of the NAT instance only.





Step 14 - Connect to the NAT instance using SSH or Session manager.

To SSH the private instance from the NAT instance, copy the content of the key pair attached to the private instance and paste it into a new file created in the NAT instance.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAuk6RSvt3G4grXpxKR5P4GYlkZdCLt+ikAoDTgx48iKFQbN99
OGx726r4t9urur/JsOOf2YwjislD/PdgPbB/1rBNhELd/01LlJnb8mz5GQ+KfjvK
6pxl3PkcxklP62hRQXnB6SehrgAFRAOxkXEv4kLfT4TPf2p6WTD02jHq/bLsZPqc
E9tBASiS/0NOEU4nDZfugvqKTR+F2H3HxGAk1fOghHRmH4qLT9C0Pdf1SRsK5lDx
VBYb3MM5YHw1jHk3D3KN5Dq+oC7k9LMFj/VL+HbKdRUfJYouM0bs5TK4SrwWcKTJ
NNbAoZqrxiUiyeCdA8yFteydK+qQ36pSZt7JNwIDAQABAoIBAB0VnNr+zMZQiKy5
a2MnN2kNKM44M0efmtMFN3/VnY+a10pXzV2cme1eEHvrYDDENgFWV6MHnJjS+Lkq
CS04O4zRcxh9s9hiQbVOG6cJ2g1OfslcHj5n8gMAJLzTvxU58M1u65WxaS0Xtbq7
dOIGsHn7BI+OZTI39iesW90gQ9Yc7xDtex1eWqNrEYW4UkZVDgQZYXmoIXJMR6eC
y33kWK8W249oj0VagVcb71c3k/Z/G0E1rob98wydG5rf94Zayg/zm1HeUGE3QPvt
gSWtjq1MSxS3flBi6g6STPjM/7eadmYCIBxvkKEIBy8zaBxjoU/QgRvSR/WnQKzK
UxjMYAECgYEA7xntanXnXY9lkZ+jLOYNydbIP0qOVTxKo/B3lFaqHJRFNCWx2ZCm
yhnzeZ4i4bC1mTBVMn/DdGUKCY5TyGXoZmxrU5E4w52Wjj9v+SRMalMJag55e04y
nzIXkhaZw2Cj3KxcV14ecLVajHMsW8e/+knB4ugEUsZL29DYFljtcAECgYEAx3lt
ukQP3OTx/4b2x+YutwOhoS6fuZi1NK8wIz9GODZ7oy/Kafrde1vQzozRWM3kPKGd
+DaFsrA+f53Sl0eWO/zXJ17nACbC0i/h6Uj8kjR6jb2LhmsXV1GL8pAX4VtwlAjH
tBujfgYBtk0TqxBYsUDdWRs9mJeiM75AG+XruTcCgYAgbDyknh5ksohMagBfm0at
YyG+a2bHqihY+MfOZr4kp/qGasImRZZIe6JnkIkcX7EQd9xlv6uKghUNXK6B00fj
c7pOF7hB1oHPAk011LVECcSSDCpvDRiSO+G0Xro1I0q1Ju5SrqGkGQORXiDaxUiE
MTJIibvkWZ15zmUvCpoQAQKBgF/oHD/QdagJU+k95O1b2N4Zmt66YgWETKuWB5kK
1SI13+AoHDQ0b8Td8lk9D3KZG8GjGya6YDH31zTYluYxfyQ8ft3pHTvPZyBT8VSO
yREqtjnsBmeMmd5b0OPBMBokMs6aIhqQp1zyWmRiTrCS3XHr9ADhvg3d6hjiwkQn
mdylAoGBAO6cZABTAF3wKwk0Wx05WrLU3qnM7MeL9fYbXAO9he0ozNIcirpespeq
WR6nyfDU3CYe4v1hpoIz6kctKWmokMJncGUcK4fXmqi+E4tnmixHc1RuWK2BcUq6
U63Egr/xU5TRIZmMelYiBolN9wODVGYBQ3zJlqmXIgFBrIWOmBR6
-----END RSA PRIVATE KEY-----
~
~
```

```
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ ssh -i "Downloads/NAT-instance.pem" ec2-user@52.3.223.244
Last login: Fri Aug  4 09:45:27 2023 from 106.51.73.138

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
8 package(s) needed for security, out of 12 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-168-1-147 ~]$ hostname
ip-172-168-1-147
[ec2-user@ip-172-168-1-147 ~]$
```

After pasting the content of the private instance key pair to a new file, use the “chmod” command to change the permissions of the file.

```
[ec2-user@ip-172-168-1-147 ~]$ sudo vi privateInstancePEM.pem
[ec2-user@ip-172-168-1-147 ~]$ sudo chmod 400 privateInstancePEM.pem
[ec2-user@ip-172-168-1-147 ~]$
```

SSH into the private instance.

```
[ec2-user@ip-172-168-1-147 ~]$ sudo ssh -i privateInstancePEM.pem ec2-user@172.168.4.193
The authenticity of host '172.168.4.193 (172.168.4.193)' can't be established.
ECDSA key fingerprint is SHA256:cJwxh0QUP1qifWNjljcbuHa3GSIDCH+c5sUfDhf1IJE.
ECDSA key fingerprint is MD5:72:21:41:d9:29:38:40:cd:06:1a:64:b0:7f:02:9b:4d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.168.4.193' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 2 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-168-4-193 ~]$ clear
```

Step 15 - Install apache in the private instance then start and enable the apache service (httpd.service)

```
[root@ip-172-168-4-193 ec2-user]# systemctl start httpd
[root@ip-172-168-4-193 ec2-user]# systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[root@ip-172-168-4-193 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2023-08-04 11:46:30 UTC; 14s ago
     Docs: man:httpd.service(8)
 Main PID: 3672 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─3672 /usr/sbin/httpd -DFOREGROUND
           ├─3673 /usr/sbin/httpd -DFOREGROUND
           ├─3674 /usr/sbin/httpd -DFOREGROUND
           ├─3675 /usr/sbin/httpd -DFOREGROUND
           ├─3676 /usr/sbin/httpd -DFOREGROUND
           └─3677 /usr/sbin/httpd -DFOREGROUND

Aug 04 11:46:30 ip-172-168-4-193.ec2.internal systemd[1]: Starting The Apache HTTP Server...
Aug 04 11:46:30 ip-172-168-4-193.ec2.internal systemd[1]: Started The Apache HTTP Server.
```

Navigate to the document root file, which is index.html here. Edit the index.html file.

```
[ec2-user@ip-172-168-4-79 ~]$ cd /
[ec2-user@ip-172-168-4-79 /]$ cd var/www/html
[ec2-user@ip-172-168-4-79 html]$ ls
[ec2-user@ip-172-168-4-79 html]$ sudo vi index.html
[ec2-user@ip-172-168-4-79 html]$
```

Curl the private ip of the instance in the nat instance to check if the apache server is working correctly.

```
[ec2-user@ip-172-168-1-147 ~]$ curl 172.168.4.193
<!doctype html>
<html lang=en>
<head>
<meta charset=utf-8>
<title>Group 1 Project</title>
</head>
<body>
<h1>Axcess.io</h1>
<p>Akash</p>
<p>Shreyas</p>
<p>Cloud Engineers</p>
</body>
</html>

[ec2-user@ip-172-168-1-147 ~]$
```

Step 16 - Configure the cloudwatch agent in the private instance.

```
[ec2-user@ip-172-168-4-79 /]$ sudo yum install amazon-cloudwatch-agent -y
Last metadata expiration check: 0:11:03 ago on Fri Aug  4 10:21:28 2023.
Dependencies resolved.
================================================================================
 Package                          Architecture                  Version
================================================================================
Installing:
 amazon-cloudwatch-agent          x86_64                        1.300025.0

Transaction Summary
================================================================================
Install  1 Package

Total download size: 69 M
Installed size: 341 M
Downloading Packages:
amazon-cloudwatch-agent-1.300025.0-1.amzn2023.x86_64.rpm
--------------------------------------------------------------------------------
```

```
[ec2-user@ip-172-168-4-193 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
============================================================
= Welcome to the Amazon CloudWatch Agent Configuration Manager =
=                                                          =
= CloudWatch Agent allows you to collect metrics and logs from =
= your host and send them to CloudWatch. Additional CloudWatch =
= charges may apply.                                       =
============================================================
On which OS are you planning to use the agent?
1. linux
2. windows
3. darwin
default choice: [1]:
1
Trying to fetch the default region based on ec2 metadata...
Are you using EC2 or On-Premises hosts?
1. EC2
2. On-Premises
default choice: [1]:
1
Which user are you planning to run the agent?
1. root
2. cwagent
3. others
default choice: [1]:
1
Do you want to turn on StatsD daemon?
1. yes
2. no
default choice: [1]:
2
Do you want to monitor metrics from CollectD? WARNING: CollectD must be installed or the Agent will fail to start
1. yes
2. no
default choice: [1]:
2
Do you want to monitor any host metrics? e.g. CPU, memory, etc.
1. yes
2. no
default choice: [1]:
1
Do you want to monitor cpu metrics per core?
1. yes
2. no
```

```
Do you want to add ec2 dimensions (ImageId, InstanceId, InstanceType, AutoScalingGroupName) into all of your metrics if the info is available?
1. yes
2. no
default choice: [1]:
1
Do you want to aggregate ec2 dimensions (InstanceId)?
1. yes
2. no
default choice: [1]:
1
Would you like to collect your metrics at high resolution (sub-minute resolution)? This enables sub-minute resolution for all metrics, but you can customize for specific met
rics in the output json file.
1. 1s
2. 10s
3. 30s
4. 60s
default choice: [4]:
4
Which default metrics config do you want?
1. Basic
2. Standard
3. Advanced
4. None
default choice: [1]:
2
```

```
Are you satisfied with the above config? Note: it can be manually customized after the wizard completes to add additional items.
1. yes
2. no
default choice: [1]:
1
Do you have any existing CloudWatch Log Agent (http://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AgentReference.html) configuration file to import for migration?
1. yes
2. no
default choice: [2]:
2
Do you want to monitor any log files?
1. yes
2. no
default choice: [1]:
1
Log file path:
/var/log/httpd/access_log
Log group name:
default choice: [access_log]
/var/log/httpd/access_log
Log stream name:
default choice: [{instance_id}]
{instance_id}/httpd/access_log
```

```
22. 3288
23. 3653
default choice: [1]:
5
Do you want to specify any additional log files to monitor?
1. yes
2. no
default choice: [1]:
1
Log file path:
/var/log/httpd/error_log
Log group name:
default choice: [error_log]
/var/log/httpd/error_log
Log stream name:
default choice: [{instance_id}]
{instance_id}/httpd/error_log
```

```
default choice: [1]:
5
Do you want to specify any additional log files to monitor?
1. yes
2. no
default choice: [1]:
1
Log file path:
/var/log/messages
Log group name:
default choice: [messages]
/var/log/messages
Log stream name:
default choice: [{instance_id}]
{instance_id}/systemLogs/messages
```

```
default choice: [1]:
5
Do you want to specify any additional log files to monitor?
1. yes
2. no
default choice: [1]:
2
Saved config file to /opt/aws/amazon-cloudwatch-agent/bin/config.json successfully.
```

```
Please check the above content of the config.
The config file is also located at /opt/aws/amazon-cloudwatch-agent/bin/config.json.
Edit it manually if needed.
Do you want to store the config in the SSM parameter store?
1. yes
2. no
default choice: [1]:
2
Program exits now.
```

Edit the configuration file to group the metrics based on the ASG.

```
[ec2-user@ip-172-168-4-193 ~]$ sudo vi /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": "/var/log/httpd/access_log",
						"log_group_name": "/var/log/httpd/access_log",
						"log_stream_name": "{instance_id}/httpd/access_log",
						"retention_in_days": 7,
						"timestamp_format": "%Y-%m-%d %H:%M:%S"
					},
					{
						"file_path": "/var/log/httpd/error_log",
						"log_group_name": "/var/log/httpd/error_log",
						"log_stream_name": "{instance_id}/httpd/error_log",
						"retention_in_days": 7,
						"timestamp_format": "%Y-%m-%d %H:%M:%S"
					},
					{
						"file_path": "/var/log/messages",
						"log_group_name": "/var/log/messages",
						"log_stream_name": "{instance_id}/systemLogs/messages",
						"retention_in_days": 7,
						"timestamp_format": "%Y-%m-%d %H:%M:%S"
					}
				]
			}
		}
	},
```

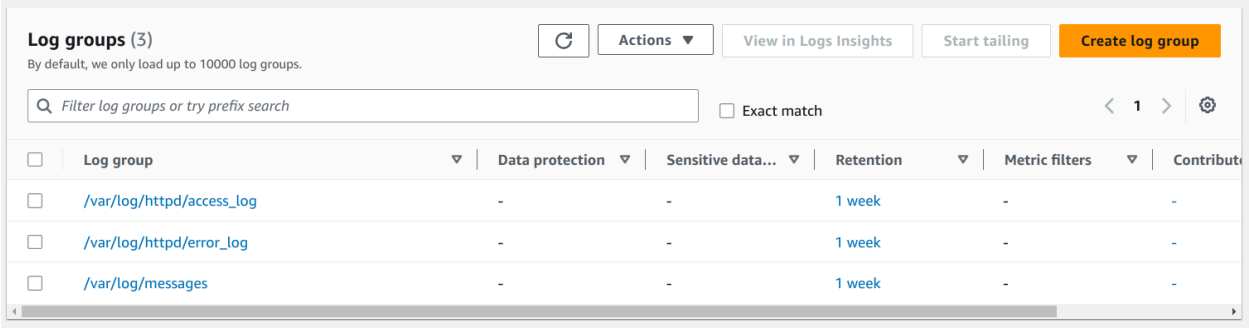
```
        "metrics": {
                "aggregation_dimensions": [
                        [
                                "InstanceId"
                        ]
                ],
                "append_dimensions": {
                        "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
                        "InstanceId": "${aws:InstanceId}"
                },
                "metrics_collected": {
                        "cpu": {
                                "measurement": [
                                        "cpu_usage_idle",
                                        "cpu_usage_iowait",
                                        "cpu_usage_user",
                                        "cpu_usage_system"
                                ],
                                "metrics_collection_interval": 60,
                                "totalcpu": false
                        },
                        "disk": {
                                "measurement": [
                                        "used_percent",
                                        "inodes_free"
                                ],
                                "metrics_collection_interval": 60,
                                "resources": [
                                        "*"
                                ]
                        },
                        "diskio": {
                                "measurement": [
                                        "io_time"
                                ],
                                "metrics_collection_interval": 60,
                                "resources": [
                                        "*"
                                ]
                        },
                        "mem": {
                                "measurement": [
                                        "mem_used_percent"
```

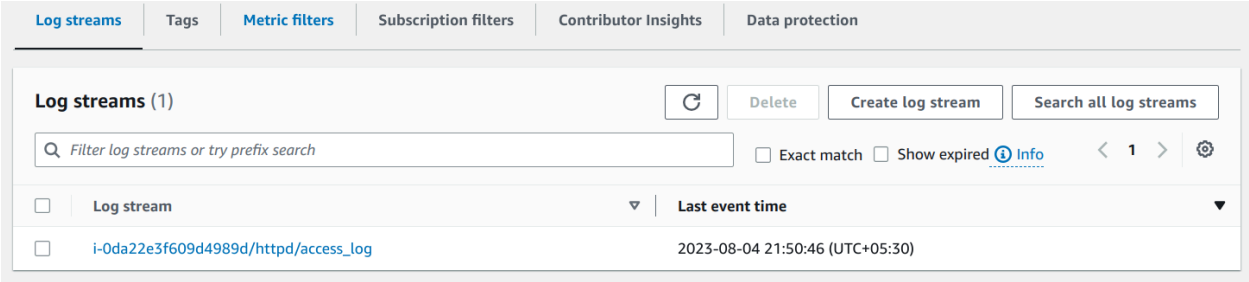
```
                        "swap": {
                                "measurement": [
                                        "swap_used_percent"
                                ],
                                "metrics_collection_interval": 60
                        }
                }
        }
}
```

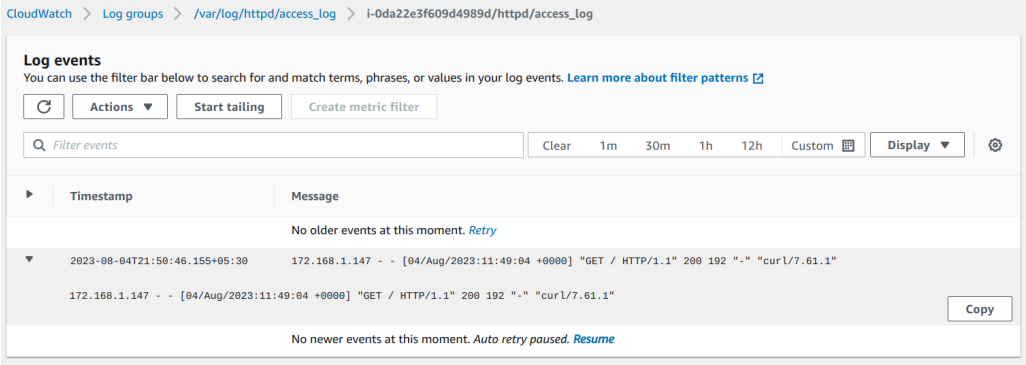
Start the CloudWatch agent. Check if it's running or not.

```
[ec2-user@ip-172-168-4-193 ~]$ 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
****** processing amazon-cloudwatch-agent ******
I! Trying to detect region from ec2 D! [EC2] Found active network interface Successfully fetched the config and saved in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp
Start configuration validation...
2023/08/04 16:20:38 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp ...
2023/08/04 16:20:38 I! Valid Json input schema.
2023/08/04 16:20:38 D! ec2tagger processor required because append_dimensions is set
2023/08/04 16:20:38 D! delta processor required because metrics with diskio or net are set
2023/08/04 16:20:38 D! ec2tagger processor required because append_dimensions is set
2023/08/04 16:20:38 Configuration validation first phase succeeded
I! Detecting run_as_user...
I! Trying to detect region from ec2
D! [EC2] Found active network interface
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -schematest -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml
Configuration validation second phase succeeded
Configuration validation succeeded
amazon-cloudwatch-agent has already been stopped
Created symlink from /etc/systemd/system/multi-user.target.wants/amazon-cloudwatch-agent.service to /etc/systemd/system/amazon-cloudwatch-agent.service.
[ec2-user@ip-172-168-4-193 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a status
{
  "status": "running",
  "starttime": "2023-08-04T16:20:39+0000",
  "configstatus": "configured",
  "version": "1.300025.0"
}
```

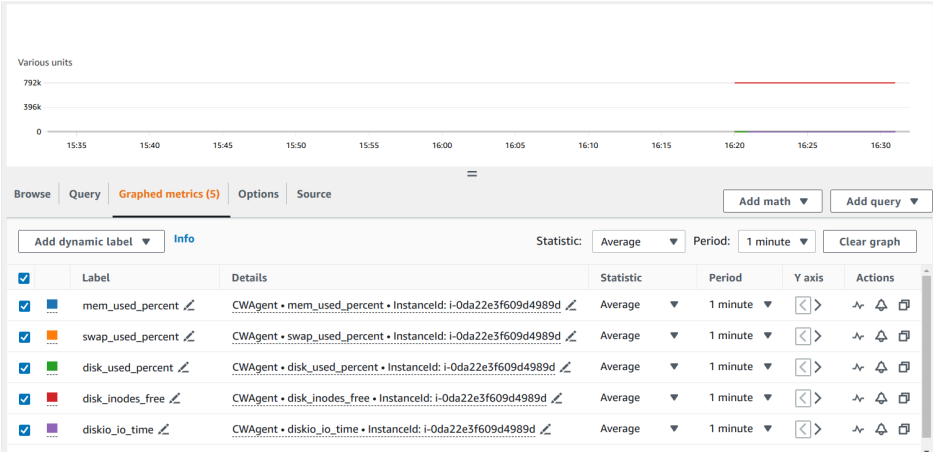
Step 17 - Check the cloudwatch for the logs.



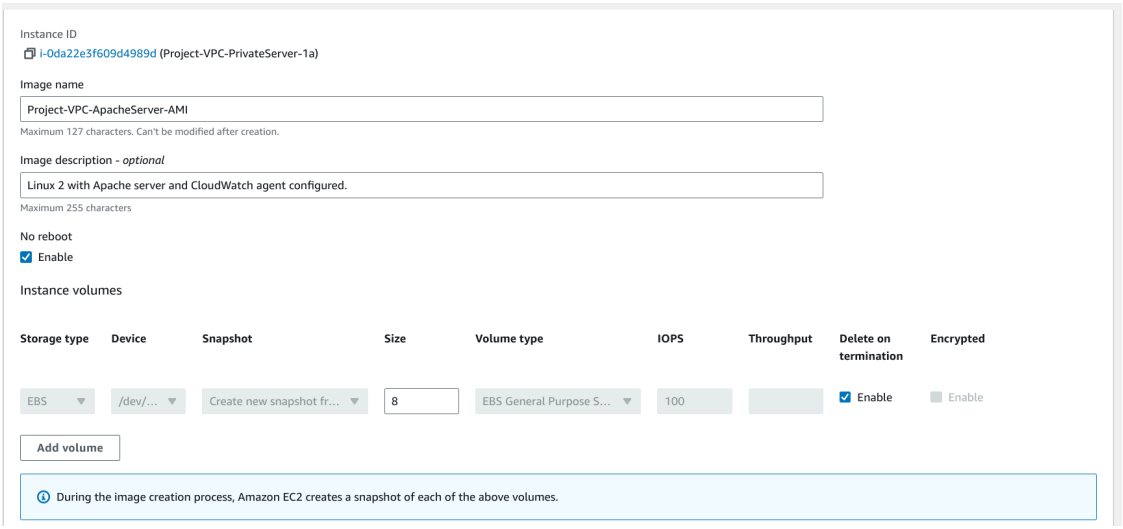




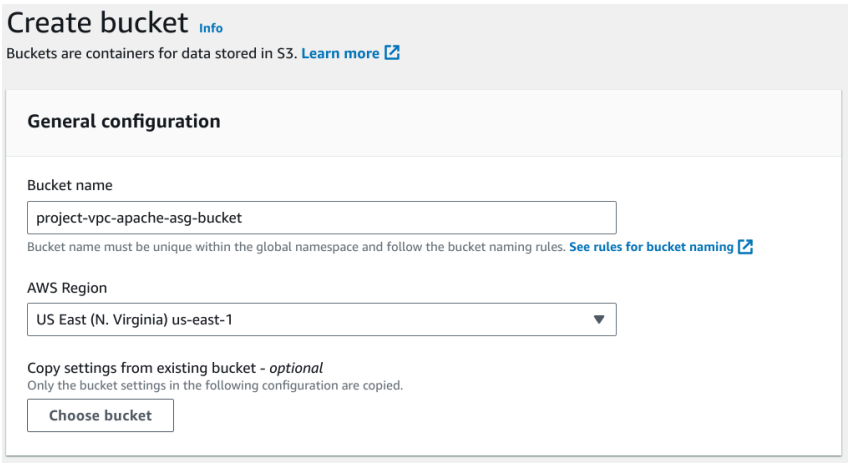
Metrics collected from the instance.



Step 18 - Create an AMI of the instance. This AMI will be used in the launch template of the ASG.

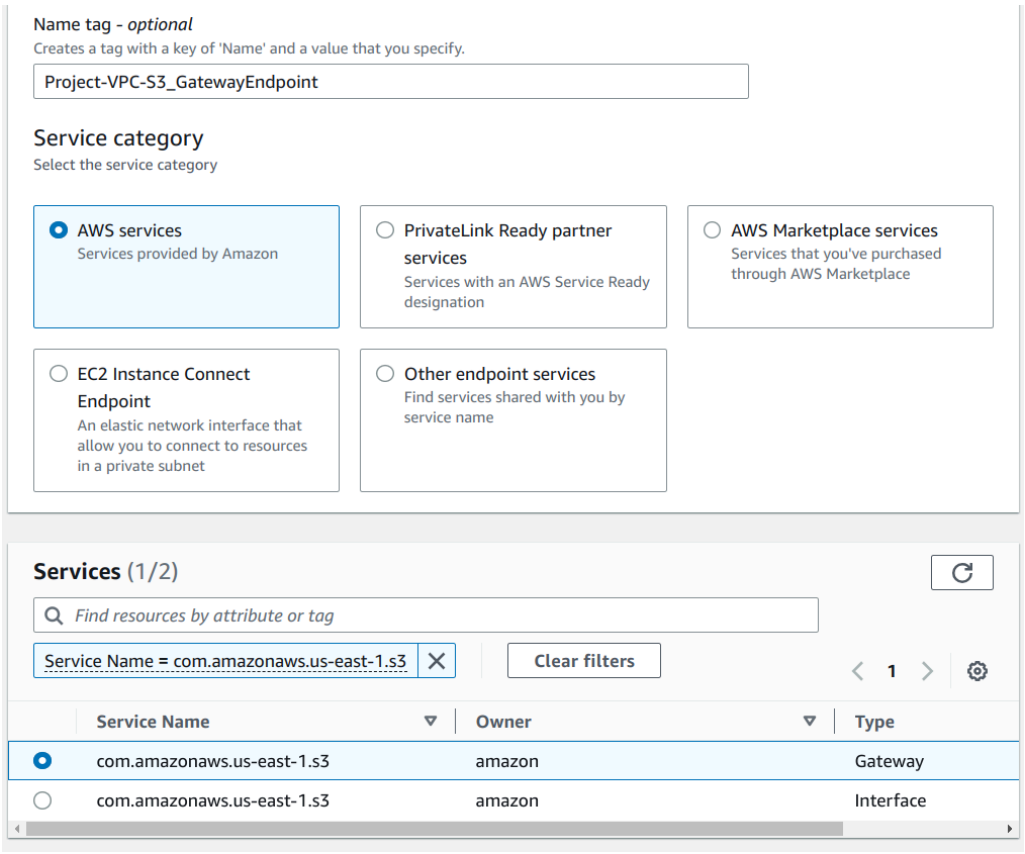


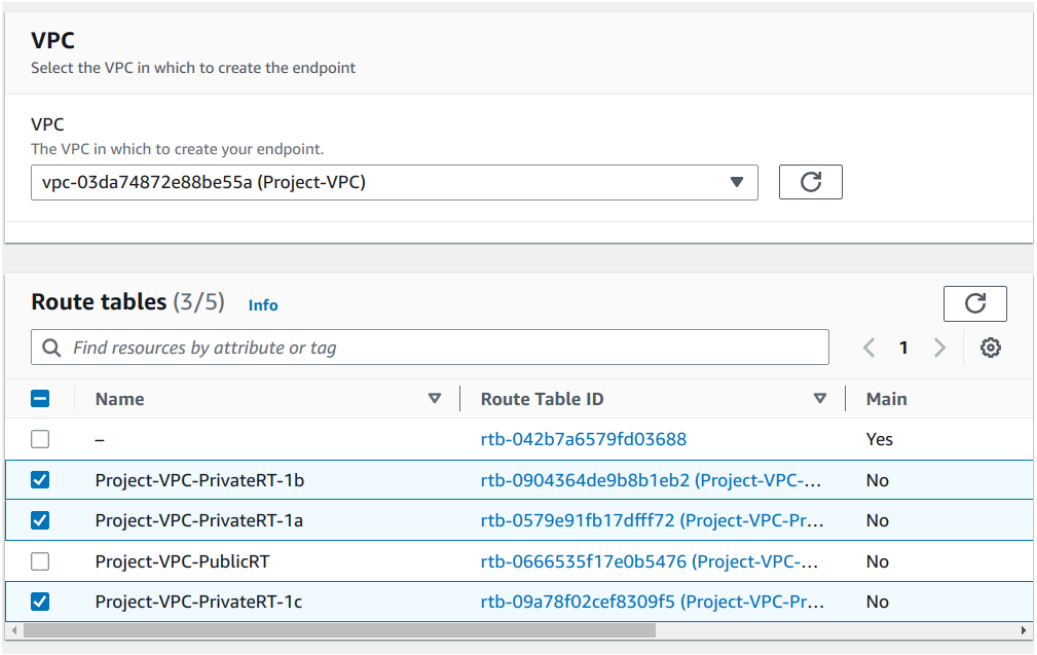
Step 19 - Create an S3 bucket which will be used to store the private IP addresses of the instances launched in the ASG.

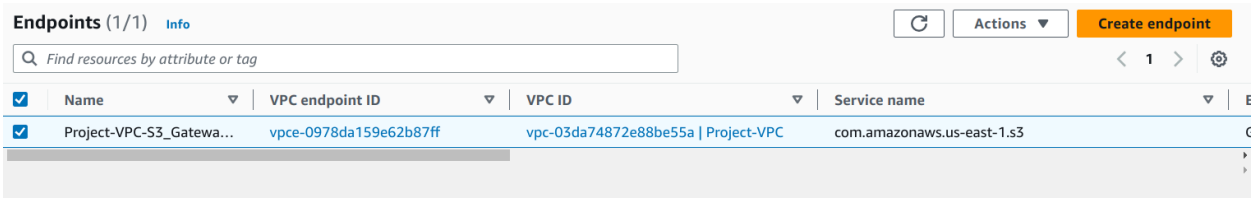


Step 20 - Create S3 gateway endpoint.

Any traffic for S3 bucket will go through the private IP on the AWS private network instead of the public internet. This is achieved by configuring the route tables of the private subnets to route traffic which have the destination of the S3 bucket to be forwarded through the endpoint as a target.

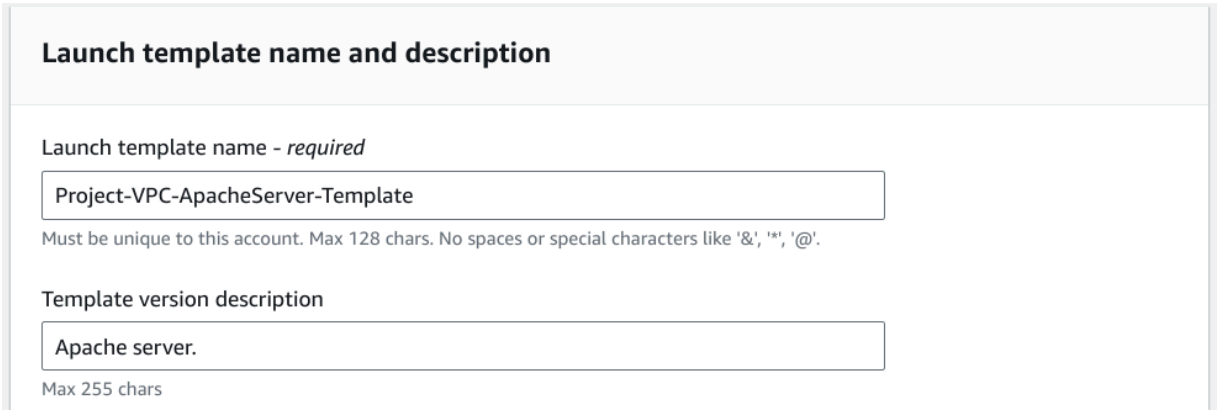


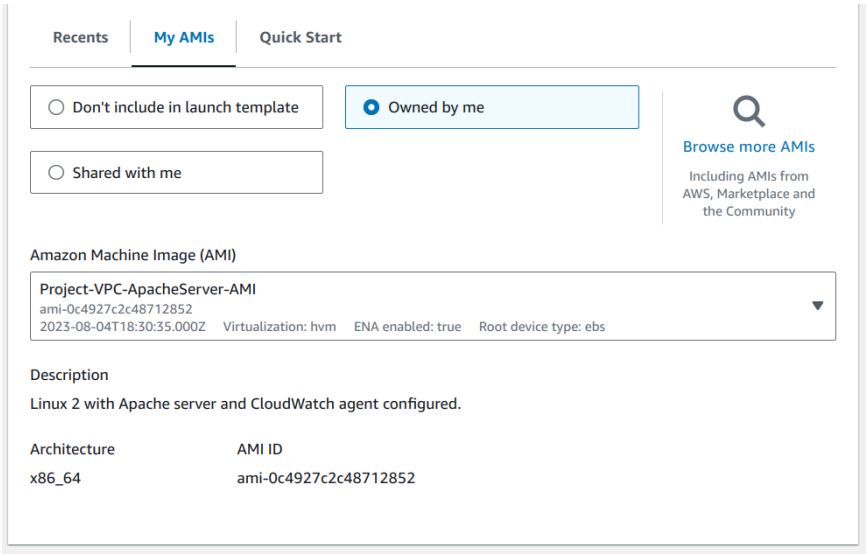


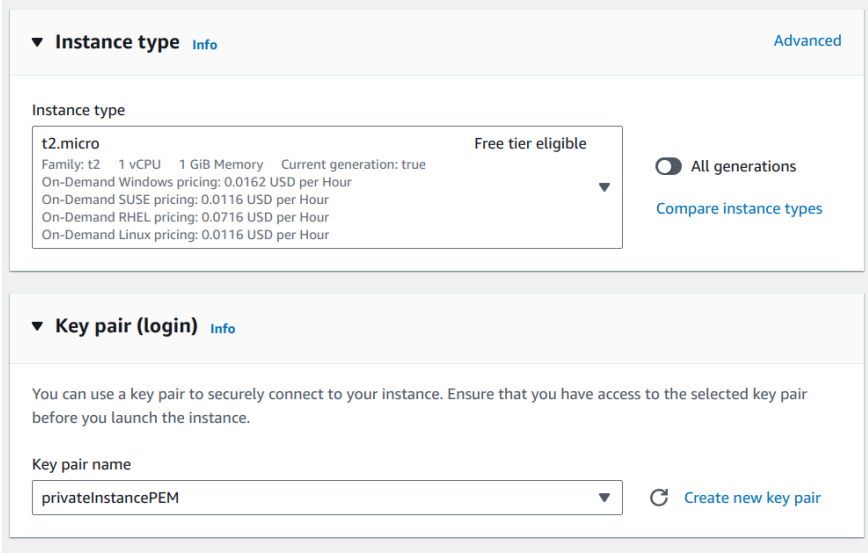


Step 21 - Create a launch template.

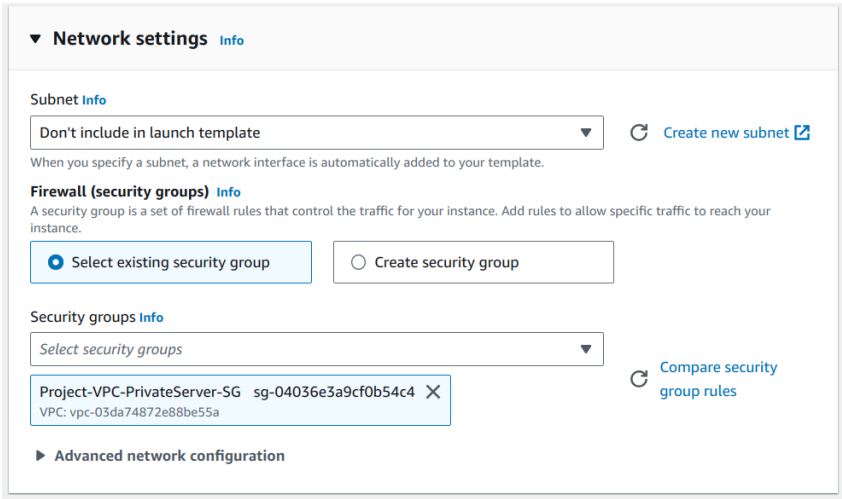
Attach the AMI created from the private instance to the launch template.

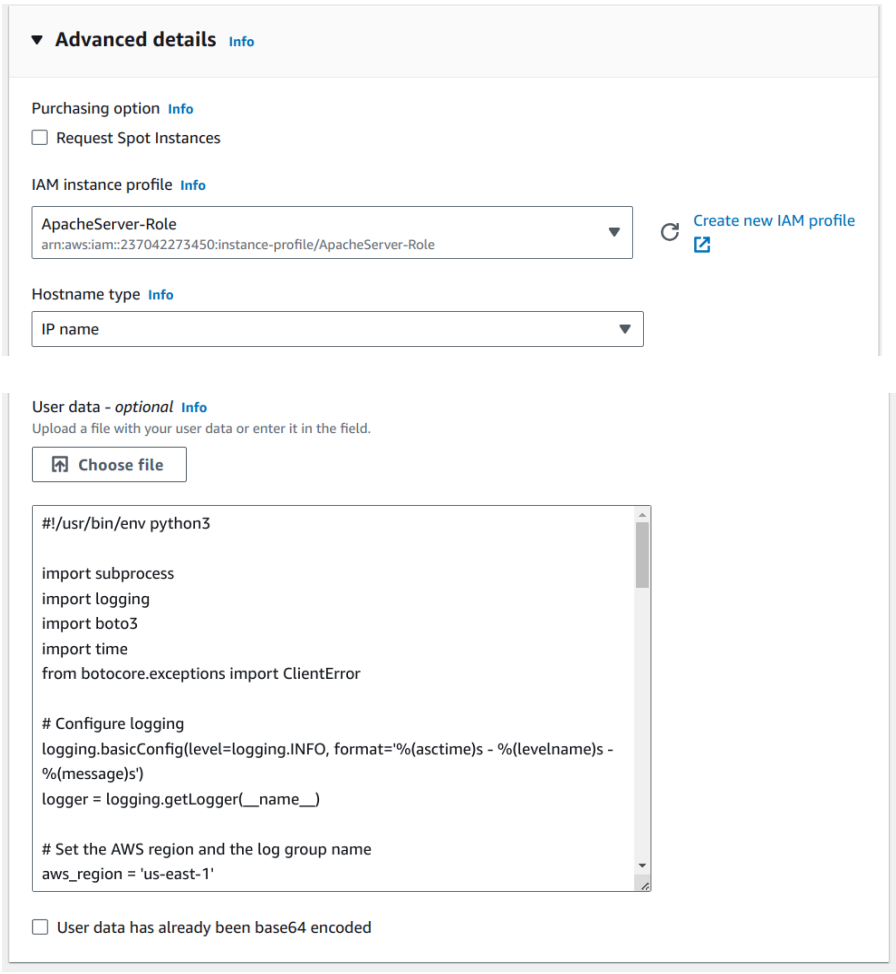



Recents
My AMIs
Quick Start
Don't include in launch template
Owned by me
Shared with me
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Project-VPC-ApacheServer-AMI
ami-0c4927c2c48712852
2023-08-04T18:30:35.000Z Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Linux 2 with Apache server and CloudWatch agent configured.
Architecture
x86_64
AMI ID
ami-0c4927c2c48712852


Instance type Info
Advanced
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour
All generations
Compare instance types

Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name
privateInstancePEM
Create new key pair


Network settings Info
Subnet Info
Don't include in launch template
Create new subnet
When you specify a subnet, a network interface is automatically added to your template.
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Select existing security group
Create security group
Security groups Info
Select security groups
Project-VPC-PrivateServer-SG sg-04036e3a9cf0b54c4
VPC: vpc-03da74872e88be55a
Compare security group rules
Advanced network configuration



The user data is a script written in python. Here the script uses the boto3 package to interact with the AWS services. The script here on first launch retrieves the private IP of the current instance by its instance id, then stores the instance id and private ip in key:value pair in a document stored in S3 bucket. Then the instance retrieves the IP addresses from the S3 bucket and runs ping operation on those IPs. It then sends the results to the Cloudwatch logs.

```python
#!/usr/bin/env python3

import subprocess
import logging
import boto3
import requests
import time
from botocore.exceptions import ClientError
```

```python
# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
    (message)s')
logger = logging.getLogger(__name__)

# Set the AWS region and the log group name
aws_region = 'us-east-1'
log_group_name = 'apacheServers/pingResponse'
s3_bucket_name = "project-vpc-apache-asg-bucket"
s3_privateIp_file = "Project-VPC-ApacheASG-PrivateIPs.txt"

s3_client = boto3.client('s3', region_name=aws_region)

# Initialize CloudWatch Logs client and log stream name
cloudwatch_client = boto3.client('logs', region_name=aws_region)
log_stream_name = None
```

```python
#get IPs from S3 bucket
def retrievePrivateIps_S3():
    try:
        response = s3_client.get_object(Bucket=s3_bucket_name, Key
            =s3_privateIp_file)
        json_bytes = response['Body'].read()
        json_string = json_bytes.decode('utf-8')
        private_ips = json.loads(json_string)

        return private_ips
    except Exception as e:
        logger.error(f"Error getting private IPs from S3 bucket: {e}")
        return []

# Function to ping other instances in the ASG
def ping_other_instances():
    try:
        privateIPs_list = retrievePrivateIps_S3().values()
```

```python
        for privateIP in privateIPs_list:
            if privateIP == get_private_ip(get_current_instance_id()):
                # Skip pinging the current instance itself
                continue

            ping_command = ['ping', '-c', '4', privateIP]
            result = subprocess.run(ping_command, capture_output=True, text=True)

            # Push the ping results to CloudWatch Logs
            log_message = f"Ping from {get_current_instance_id()} to {privateIP}:
                {result.stdout.strip()}"
```

```
            push_to_cloudwatch_logs(log_message)

    except Exception as e:
        logger.error(f"Error: {e}")

# Function to get the current EC2 instance ID
def get_current_instance_id():
    try:
        response = requests.get('http://169.254.169.254/latest/meta-data/instance
            -id', timeout=2)
        response.raise_for_status()
        return response.text.strip()
    except requests.exceptions.RequestException as e:
        logger.error(f"Error getting instance ID: {e}")
        return None

# Function to get the private IP of an instance
def get_private_ip(instance_id):
    ec2_resource = boto3.resource('ec2', region_name=aws_region)
    instance = ec2_resource.Instance(instance_id)
    return instance.private_ip_address

# Function to get the log stream name for the current instance
def get_log_stream_name():
    global log_stream_name
```

```
    if log_stream_name is None:
        log_stream_name = get_current_instance_id()

        try:
            response = cloudwatch_client.describe_log_streams(
                logGroupName=log_group_name,
                logStreamNamePrefix=log_stream_name,
                limit=1
            )
            if 'logStreams' in response:
                for stream in response['logStreams']:
                    if stream['logStreamName'] == log_stream_name:
                        return log_stream_name

            # If log stream doesn't exist, create a new one
            cloudwatch_client.create_log_stream(logGroupName=log_group_name,
                logStreamName=log_stream_name)
            return log_stream_name

        except ClientError as e:
            logger.error(f"Error retrieving log stream name: {e}")

    return log_stream_name
```

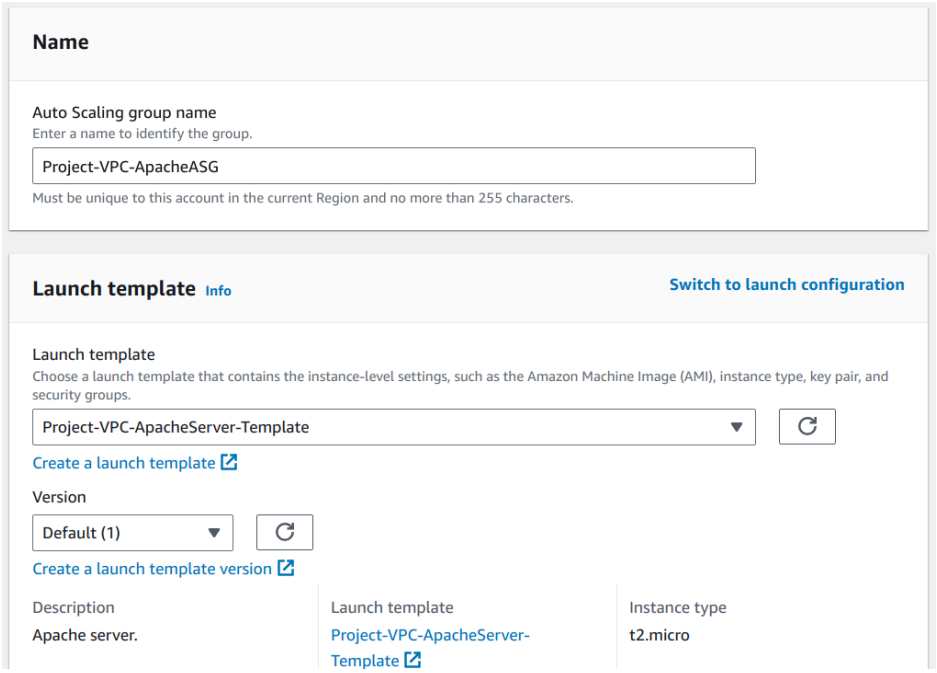
```python
# Function to push logs to CloudWatch Logs
def push_to_cloudwatch_logs(log_message):
    log_stream_name = get_log_stream_name()

    try:
        cloudwatch_client.put_log_events(
            logGroupName=log_group_name,
            logStreamName=log_stream_name,
            logEvents=[{'timestamp': int(time.time() * 1000), 'message':
                log_message}]
        )
        logger.info(f"Pushed logs to CloudWatch Logs: {log_message}")
    except ClientError as e:
        logger.error(f"Error pushing logs to CloudWatch Logs: {e}")
```

```python
def main():
    instance_id = get_current_instance_id()
    currentInstance_privateIP = get_private_ip(instance_id)
    objectList = s3_client.list_objects(Bucket=s3_bucket_name)
    objects = []
    try:
        for file in objectList['Contents']:
            objects.append(file['Key'])
    except KeyError as e:
        pass

    if s3_privateIp_file not in objects:
        s3_client.put_object(Bucket=s3_bucket_name, Key=s3_privateIp_file)

    response = s3_client.get_object(Bucket=s3_bucket_name, Key
        =s3_privateIp_file)
    json_bytes = response['Body'].read()

    # Convert bytes to a string
    json_string = json_bytes.decode('utf-8')

    # Parse the JSON string
    existingIps = json.loads(json_string)
```

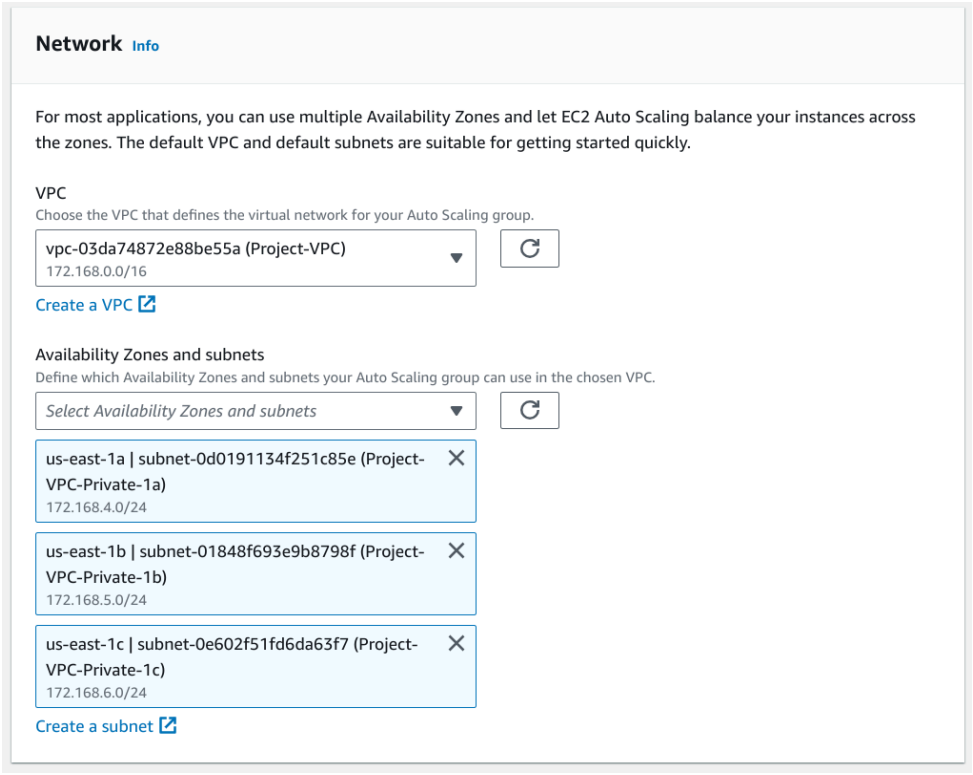
```python
    if instance_id not in existingIps.keys():
        existingIps[instance_id] = currentInstance_privateIP
        updatedIps = json.dumps(existingIps, indent=2)
        s3_client.put_object(Bucket=s3_bucket_name, Key=s3_privateIp_file,
            Body=updatedIps)
```

```python
    while True:
        ping_other_instances()

        time.sleep(60)

if __name__ == "__main__":
    main()
```

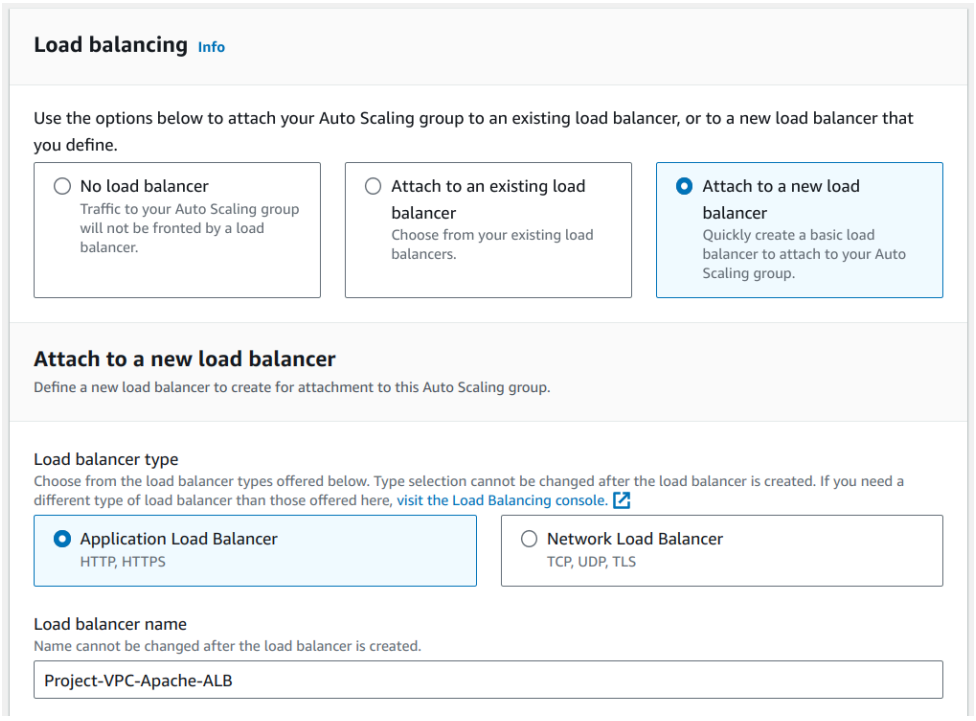
Step 22 - Create an Auto Scaling Group.

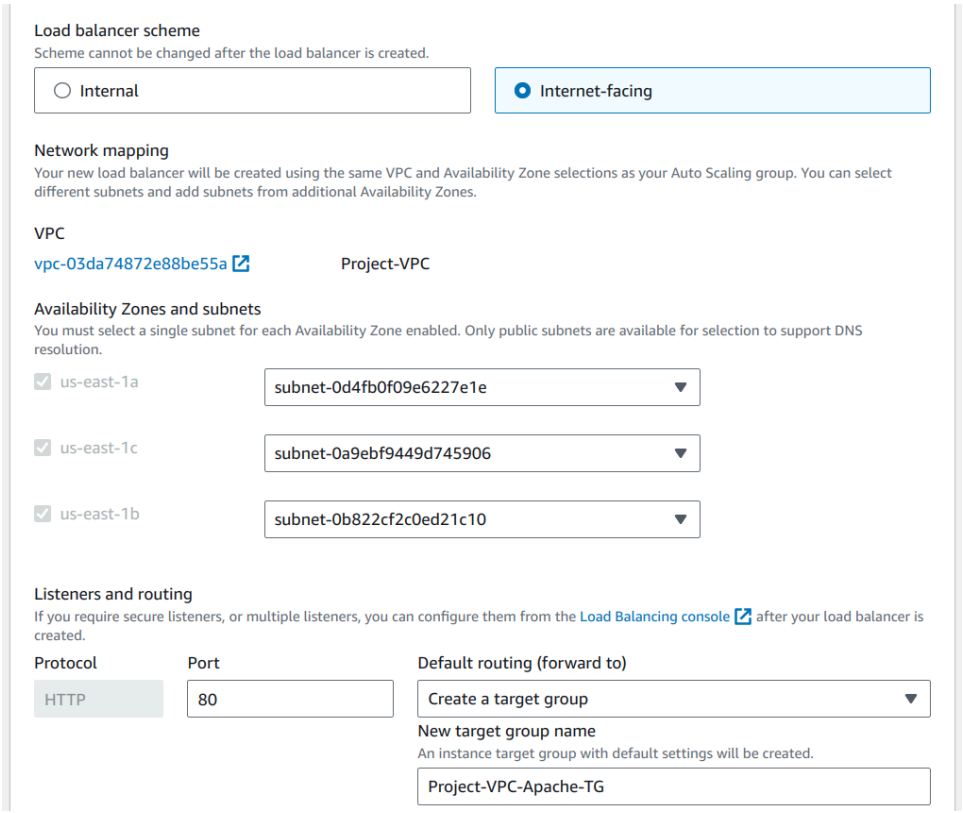


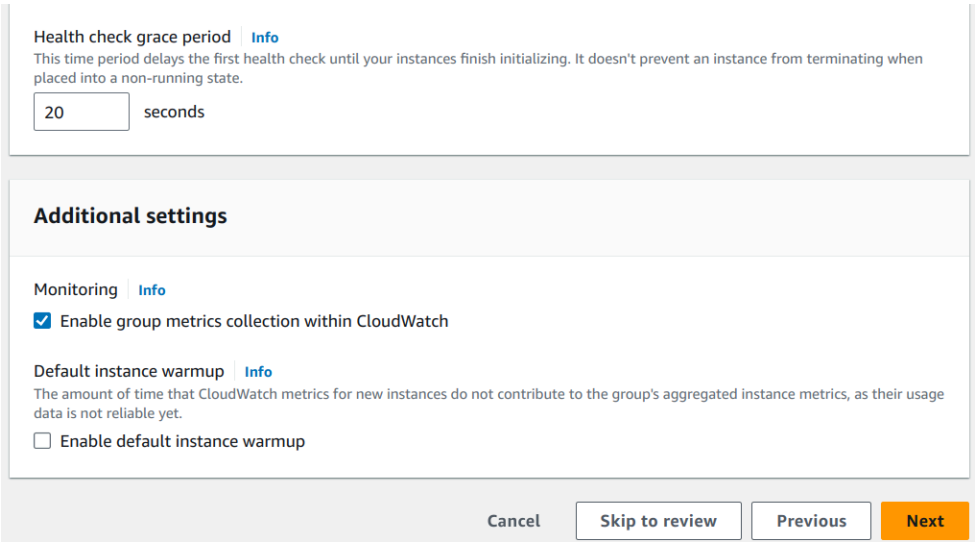
Select the three private subnets for the private instances.

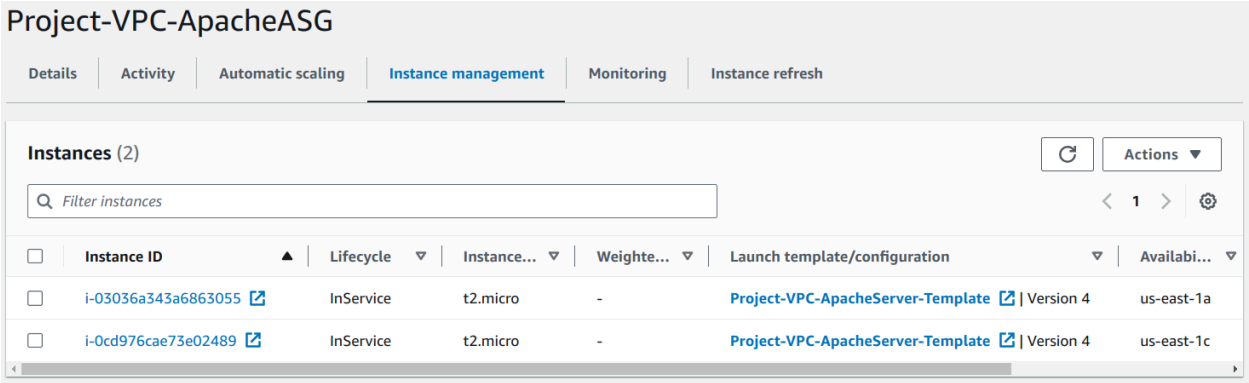


Create an Application Load Balancer which will distribute the incoming traffic on apache server to the instances launched in the ASG.

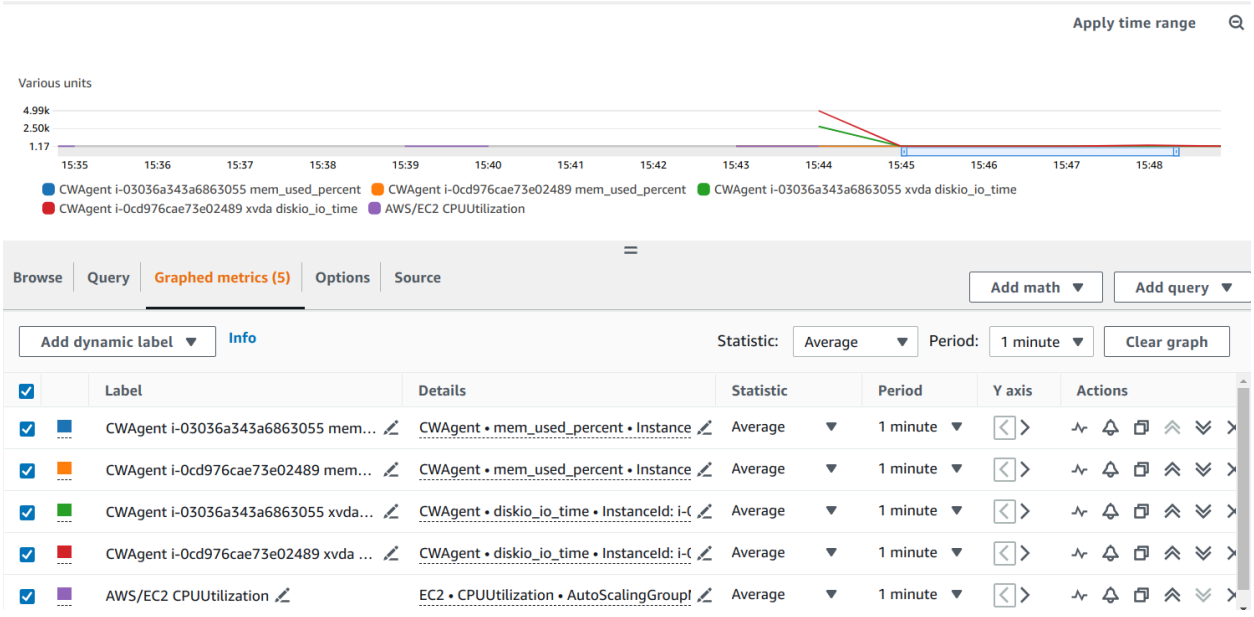


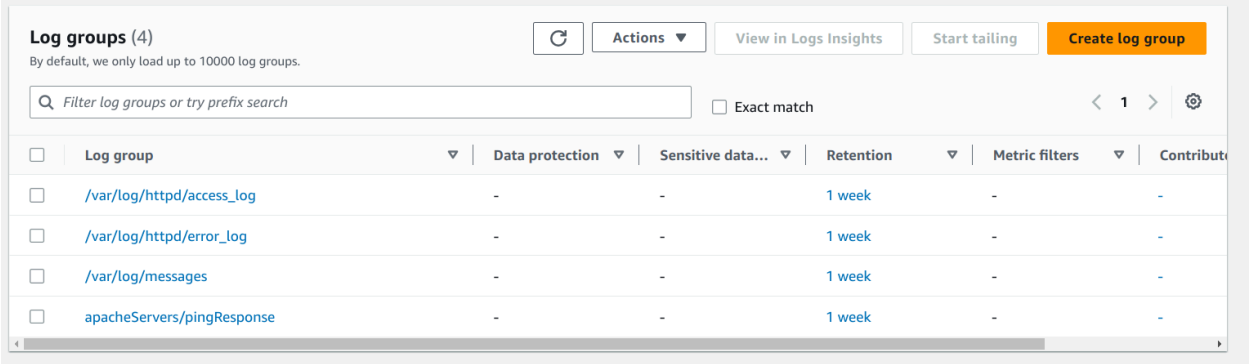




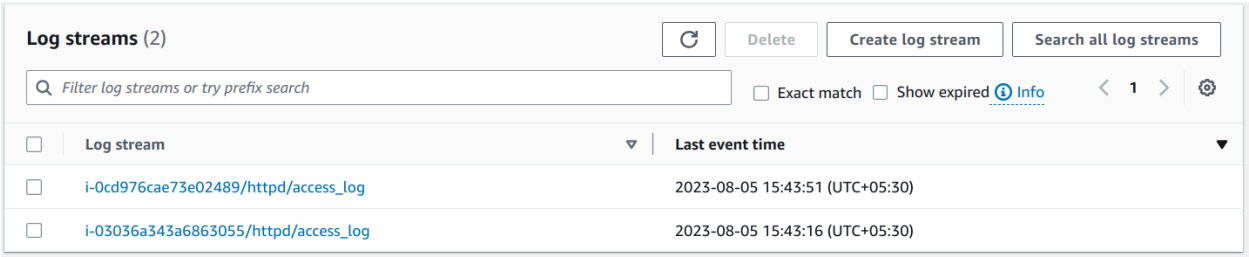


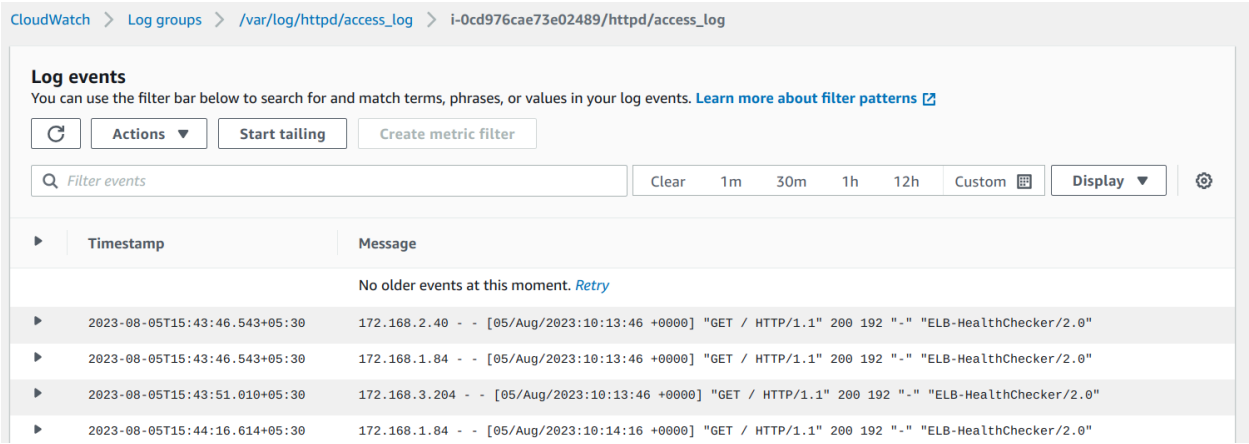
After the instances are launched it sends metrics to cloudwatch based on the Auto Scaling Group name. As it was configured in the custom AMI.

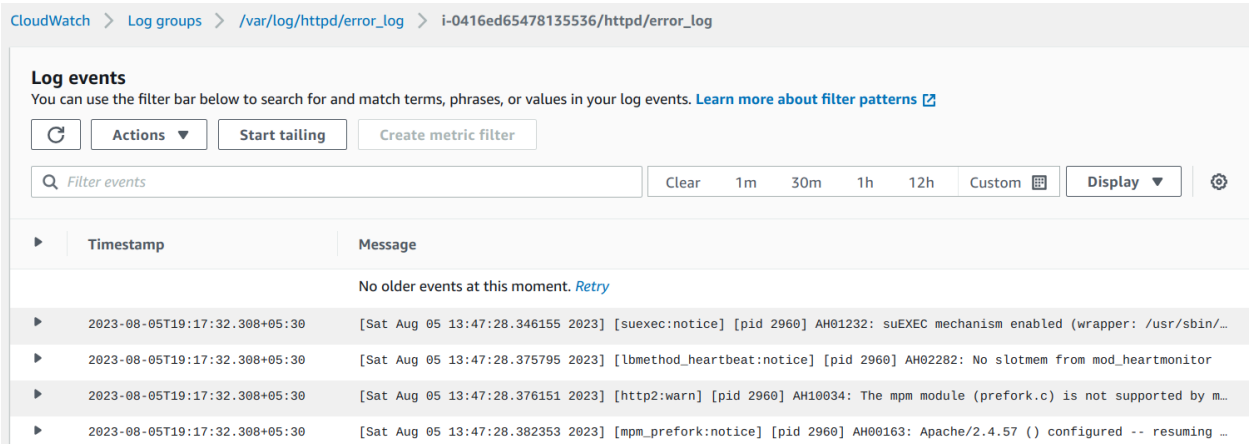




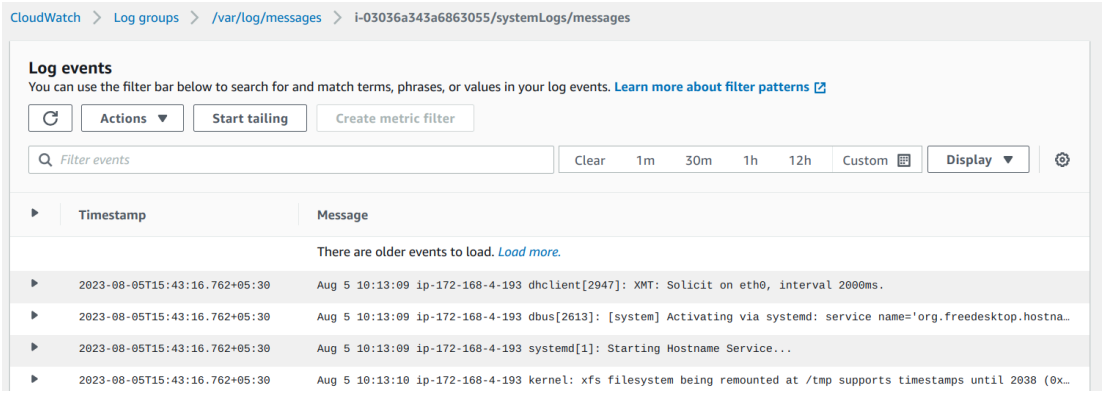
Application access and error logs from the Apache server running on the instance.



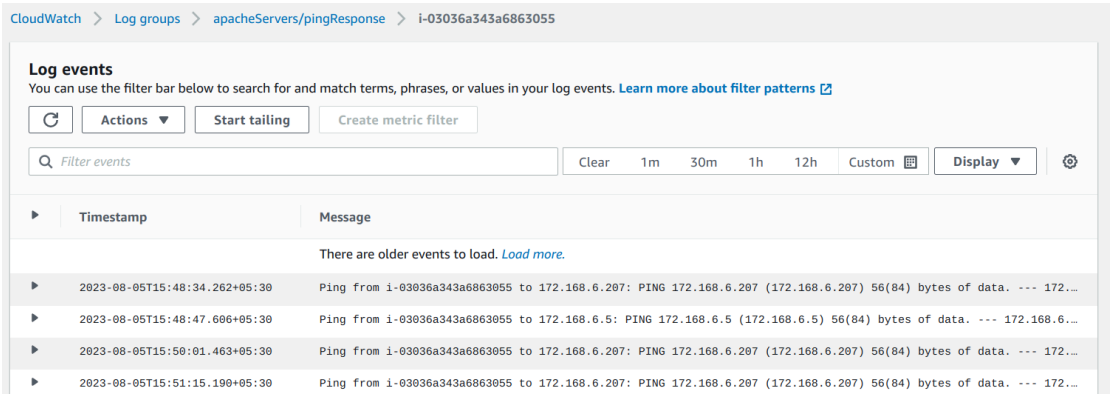




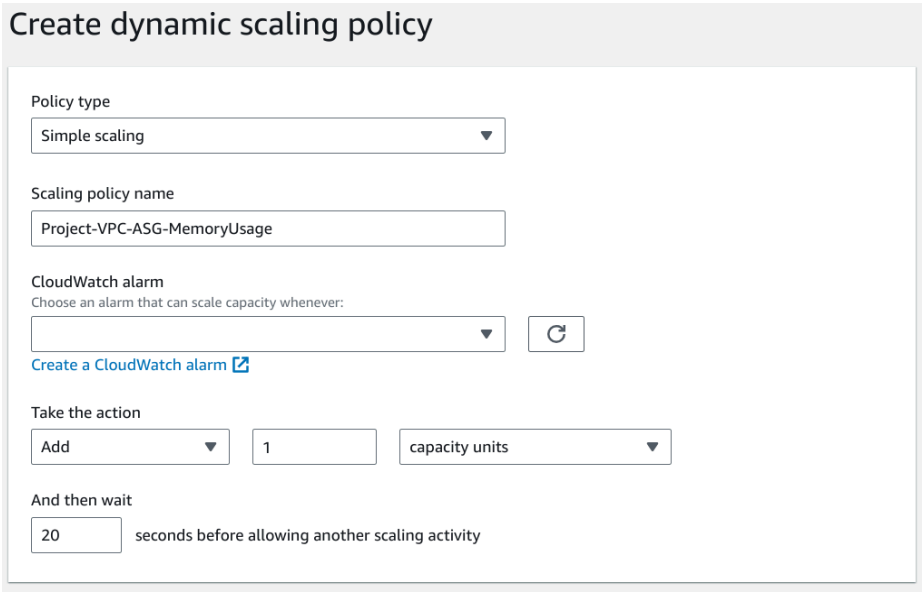
System logs from the instances.



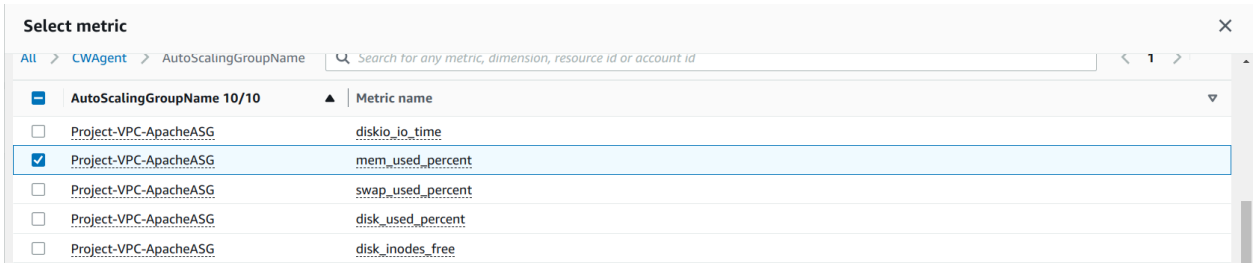
The instances ping other instances every minute and send logs to the cloudwatch.



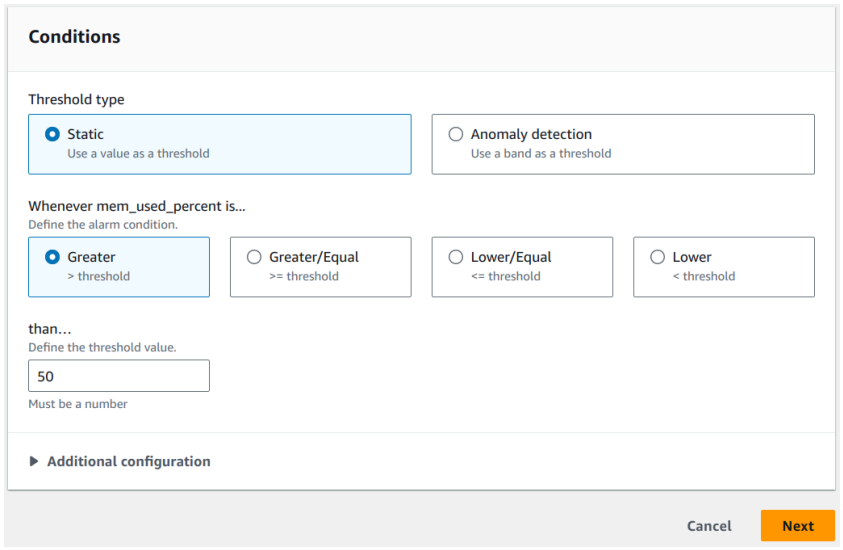
Step 23 - Create ASG memory based alarm and scaling policy to launch a new instance whenever the memory usage goes over 50% of threshold.



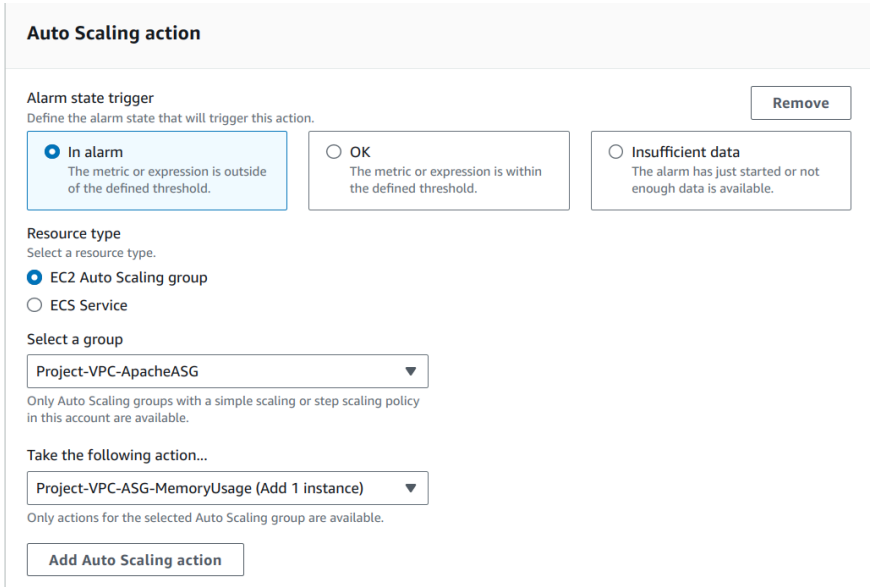
After creating dynamic simple scaling policy in the ASG, create an alarm in CloudWatch.



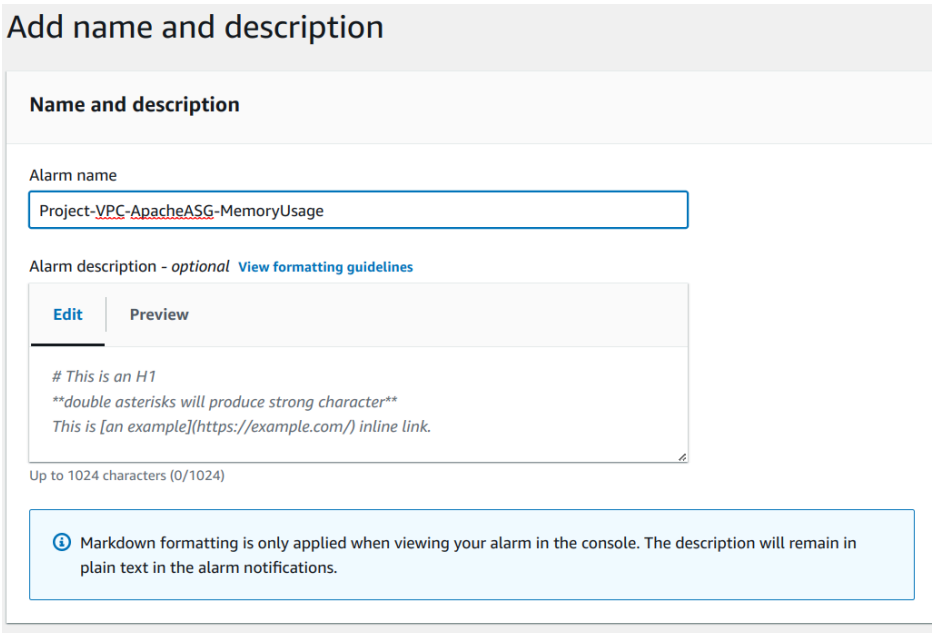
The alarm will be in “in-alarm” state if the metric value crosses the threshold of 50%.

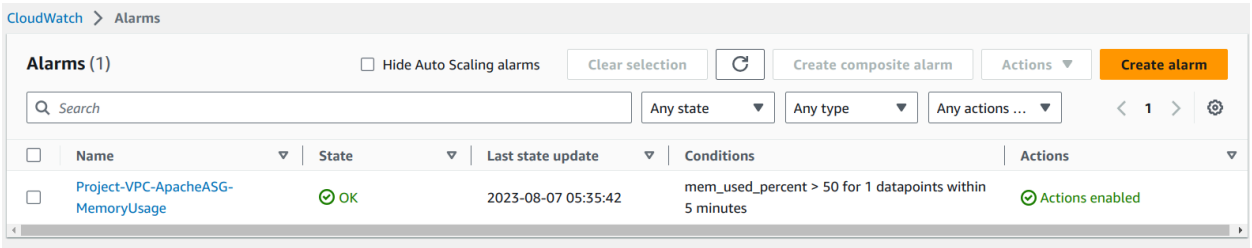


The alarm will trigger auto scaling action.



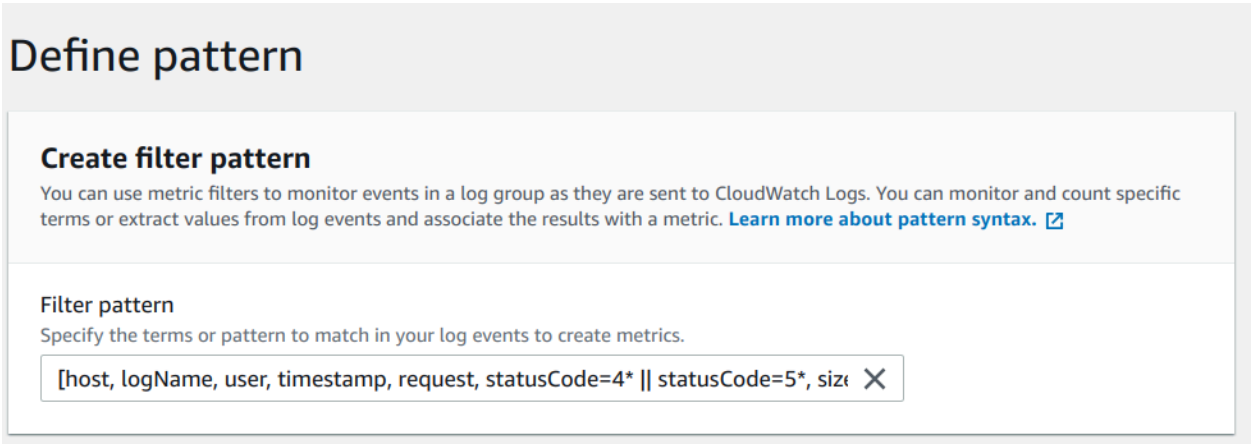
Name of the alarm.

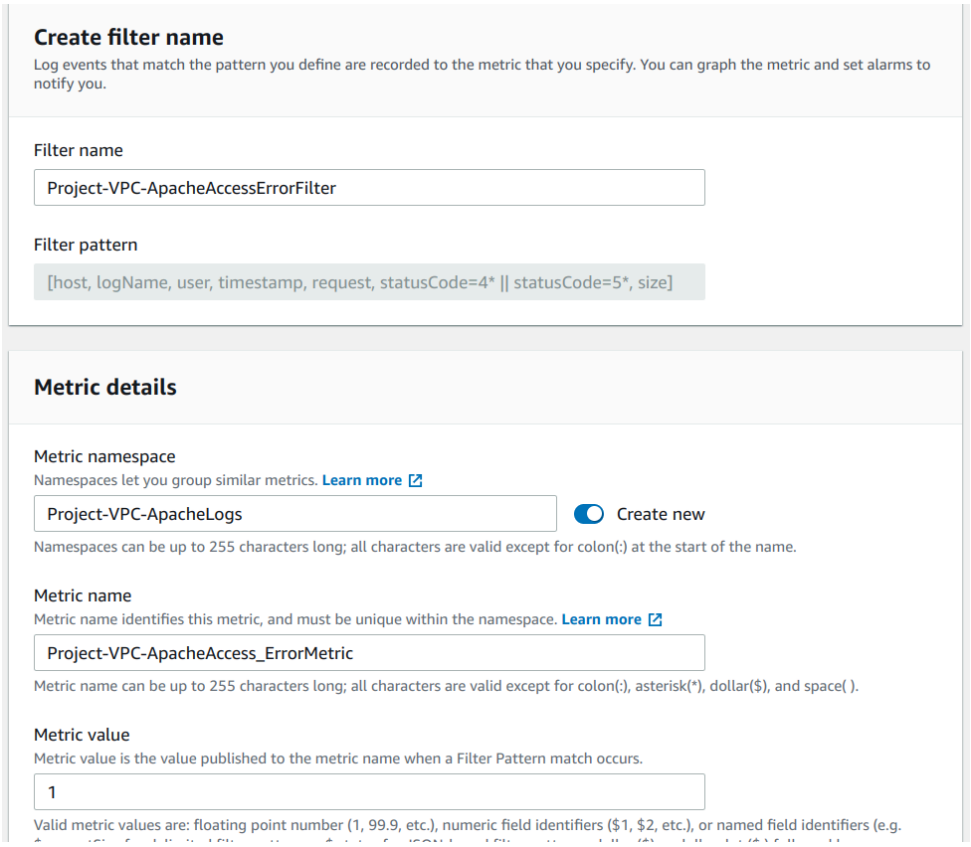




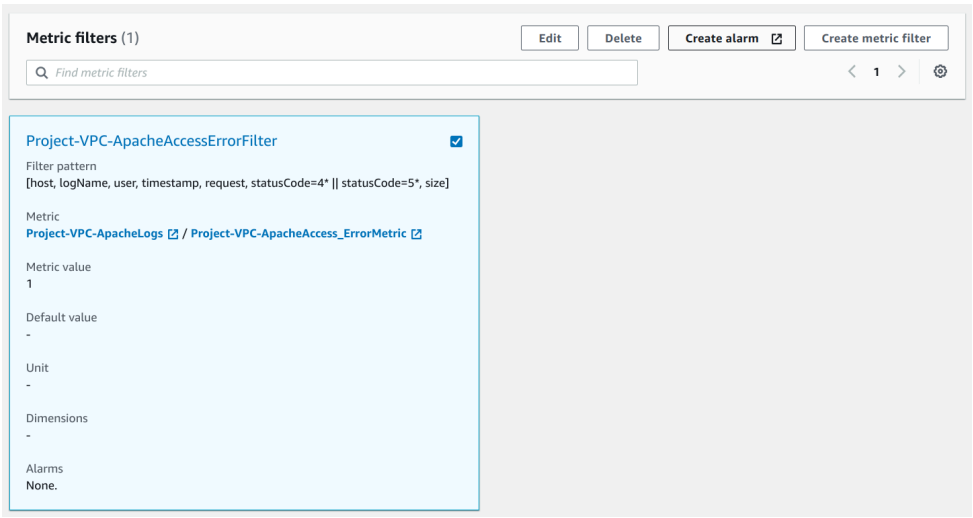
Step 24 - Create metric filters in the application log group to catch errors and exceptions.

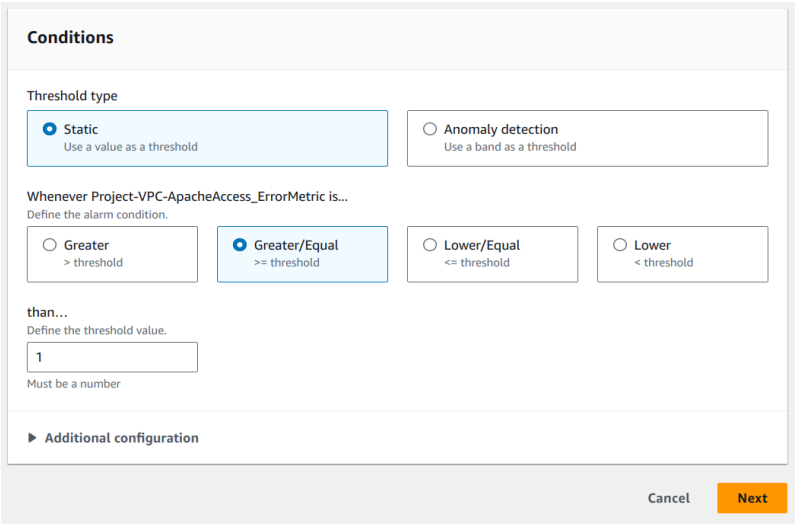
The filter pattern will catch the logs which contain 4xx and 5xx status codes.

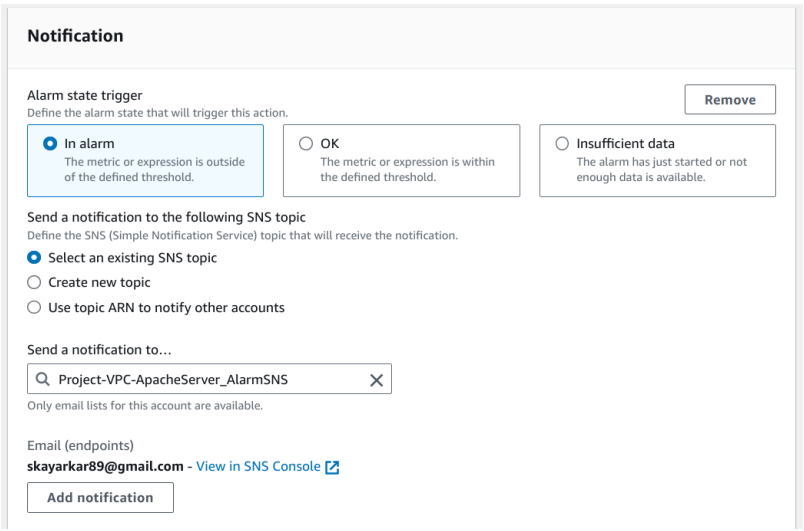


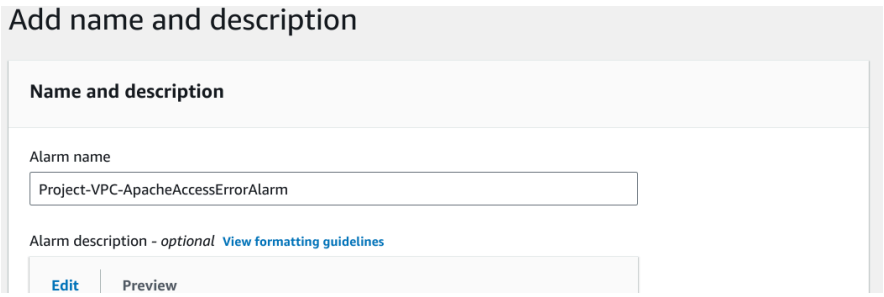


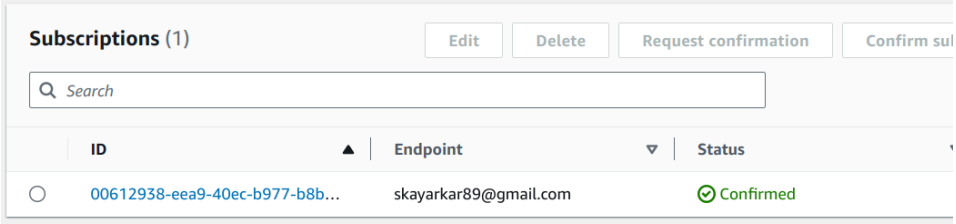
Step 25 - Create an alarm on the metric filter, which will be triggered whenever the filter catches a matching event.



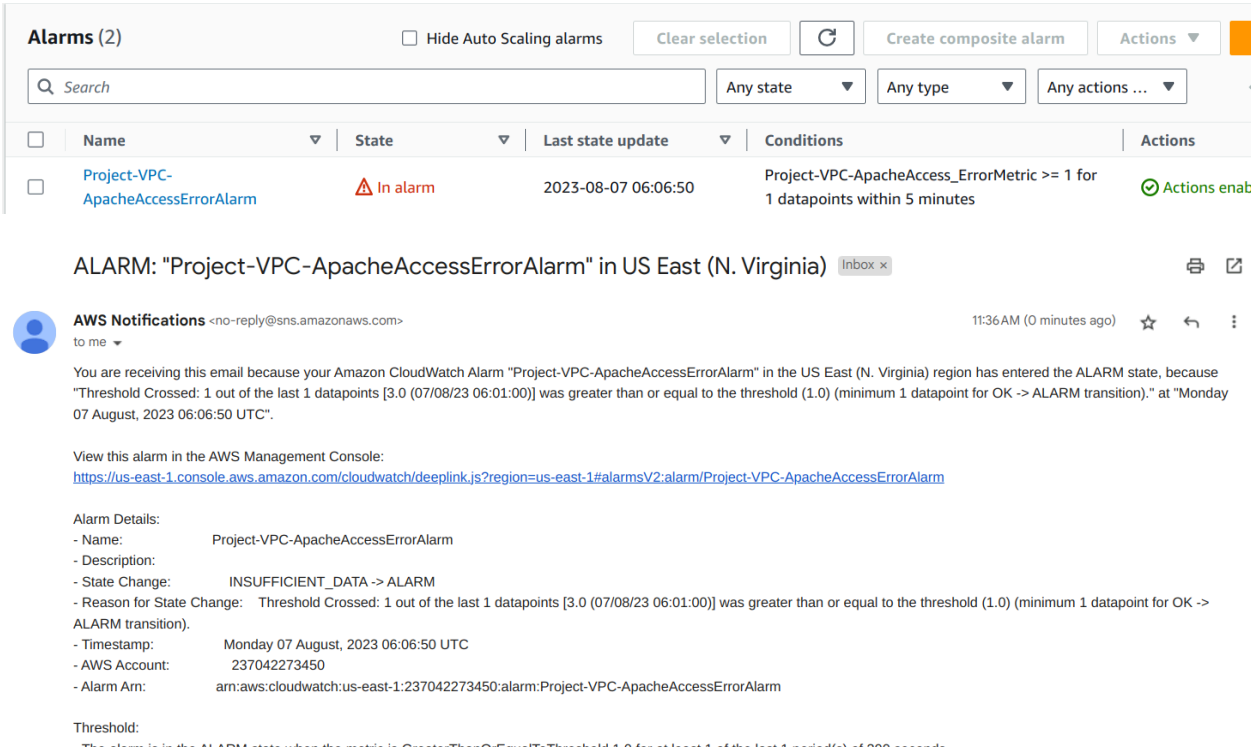

Conditions
Threshold type
Static
Use a value as a threshold
Anomaly detection
Use a band as a threshold
Whenever Project-VPC-ApacheAccess_ErrorMetric is...
Define the alarm condition.
Greater
> threshold
Greater/Equal
>= threshold
Lower/Equal
<= threshold
Lower
< threshold
than...
Define the threshold value.
1
Must be a number
Additional configuration
Cancel
Next


Notification
Alarm state trigger
Define the alarm state that will trigger this action.
Remove
In alarm
The metric or expression is outside of the defined threshold.
OK
The metric or expression is within the defined threshold.
Insufficient data
The alarm has just started or not enough data is available.
Send a notification to the following SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.
Select an existing SNS topic
Create new topic
Use topic ARN to notify other accounts
Send a notification to...
Project-VPC-ApacheServer_AlarmSNS
Only email lists for this account are available.
Email (endpoints)
skayarkar89@gmail.com - View in SNS Console
Add notification

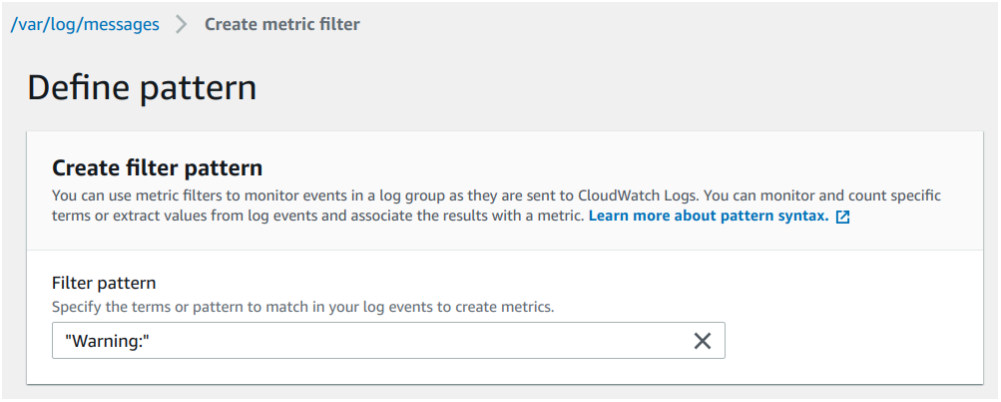

Add name and description
Name and description
Alarm name
Project-VPC-ApacheAccessErrorAlarm
Alarm description - optional View formatting guidelines
Edit
Preview

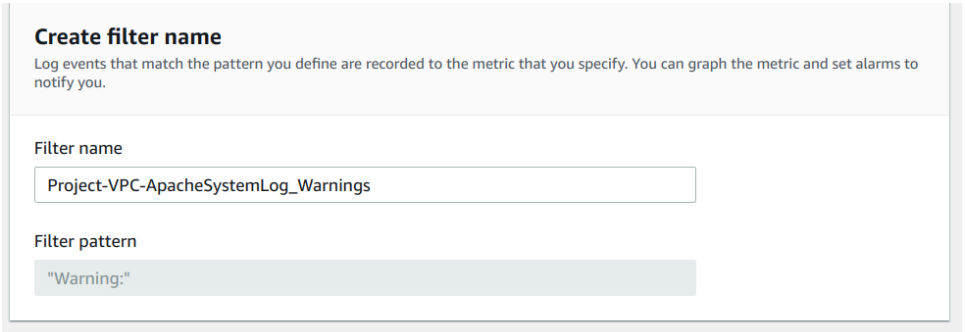

Subscriptions (1)
Edit
Delete
Request confirmation
Search
ID
Endpoint
Status
00612938-eea9-40ec-b977-b8b...
skayarkar89@gmail.com
Confirmed

The alarm is in “in-alarm” state because previously the log group recorded one log with 404 status code, which crosses the threshold limit of 1.



Step 26 - Metric filter for system logs with warning and error.





Metric details

Metric namespace

Namespaces let you group similar metrics. Learn more

Project-VPC-ApacheSystemLogMetrics — Create new

Namespaces can be up to 255 characters long; all characters are valid except for colon(:) at the start of the name.

Metric name

Metric name identifies this metric, and must be unique within the namespace. Learn more

Project-VPC-ApacheSystemWarningMetric

Metric name can be up to 255 characters long; all characters are valid except for colon(:), asterisk(*), dollar($), and space( ).

Metric value

Metric value is the value published to the metric name when a Filter Pattern match occurs.

1

Valid metric values are: floating point number (1, 99.9, etc.), numeric field identifiers ($1, $2, etc.), or named field identifiers (e.g.

Conditions

Threshold type

- (●) Static — Use a value as a threshold
- ( ) Anomaly detection — Use a band as a threshold

Whenever Project-VPC-ApacheSystemWarningMetric is...

Define the alarm condition.

- ( ) Greater — > threshold
- (●) Greater/Equal — >= threshold
- ( ) Lower/Equal — <= threshold
- ( ) Lower — < threshold

than...

Define the threshold value.

1

Must be a number

▶ Additional configuration

Notification

Alarm state trigger — Remove

Define the alarm state that will trigger this action.

- (●) In alarm — The metric or expression is outside of the defined threshold.
- ( ) OK — The metric or expression is within the defined threshold.
- ( ) Insufficient data — The alarm has just started or not enough data is available.

Send a notification to the following SNS topic

Define the SNS (Simple Notification Service) topic that will receive the notification.

- (●) Select an existing SNS topic
- ( ) Create new topic
- ( ) Use topic ARN to notify other accounts

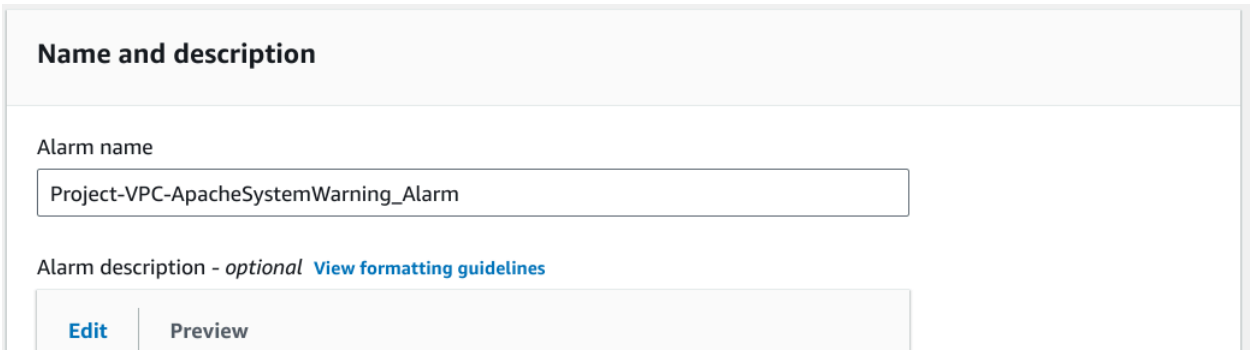
Send a notification to...

Project-VPC-ApacheServer_AlarmSNS

Only email lists for this account are available.

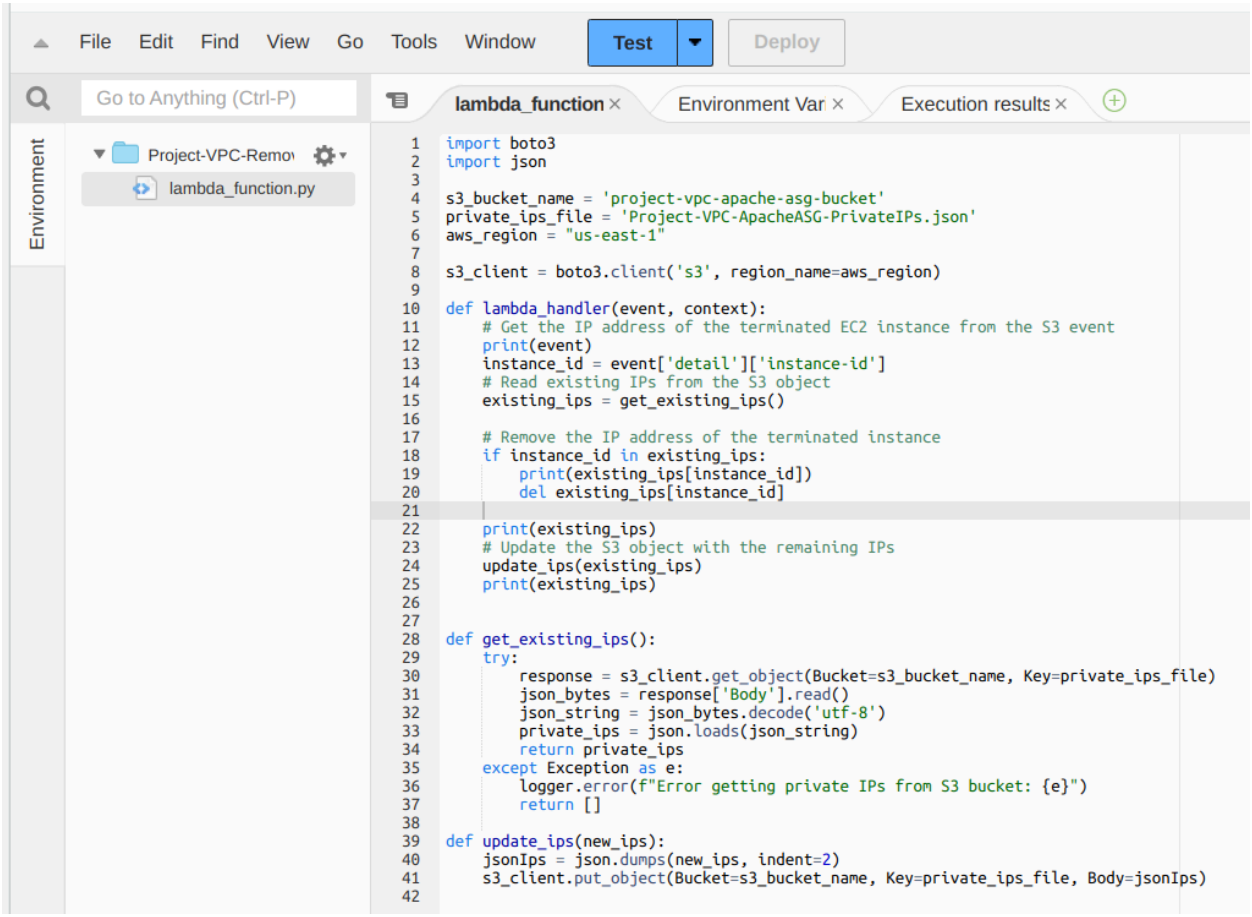
Email (endpoints)

skayarkar89@gmail.com - View in SNS Console

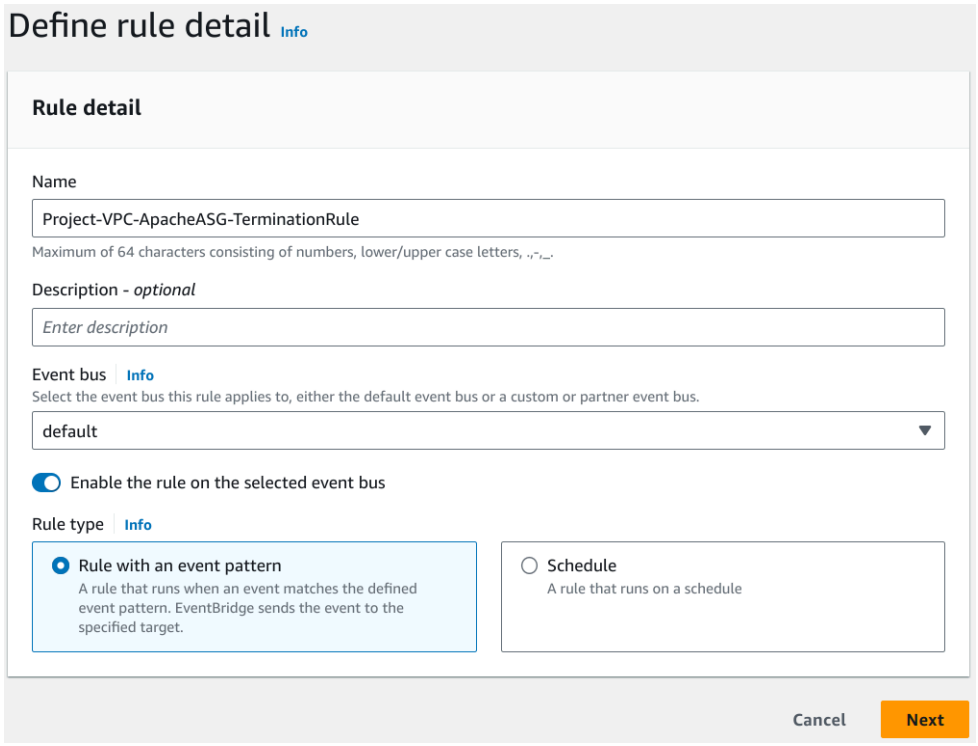


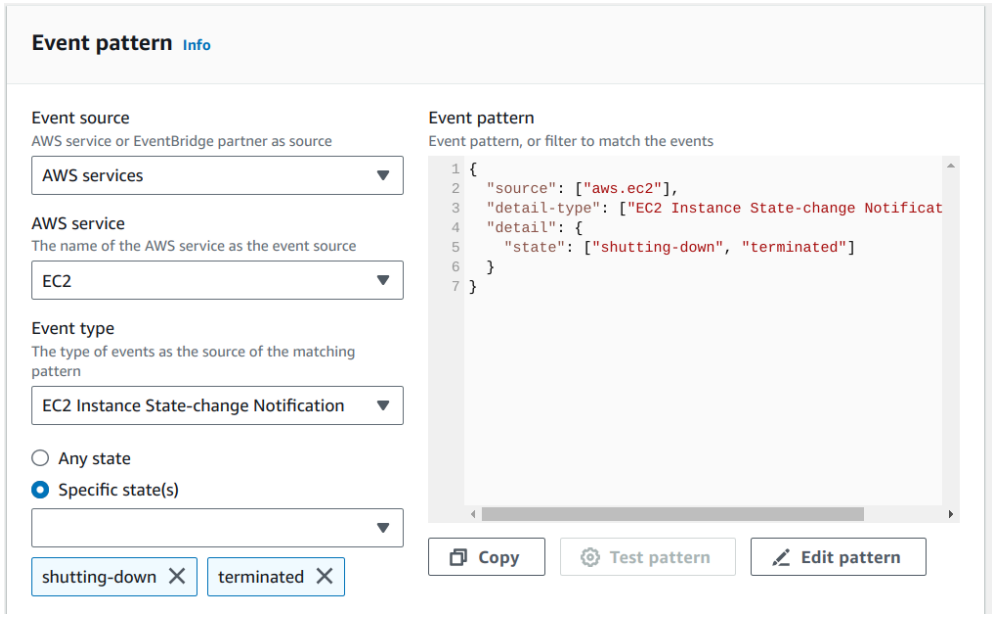
Step 27 - Create an event bridge rule to track EC2 termination events and delete the private IP entry for that instance from the s3 bucket.

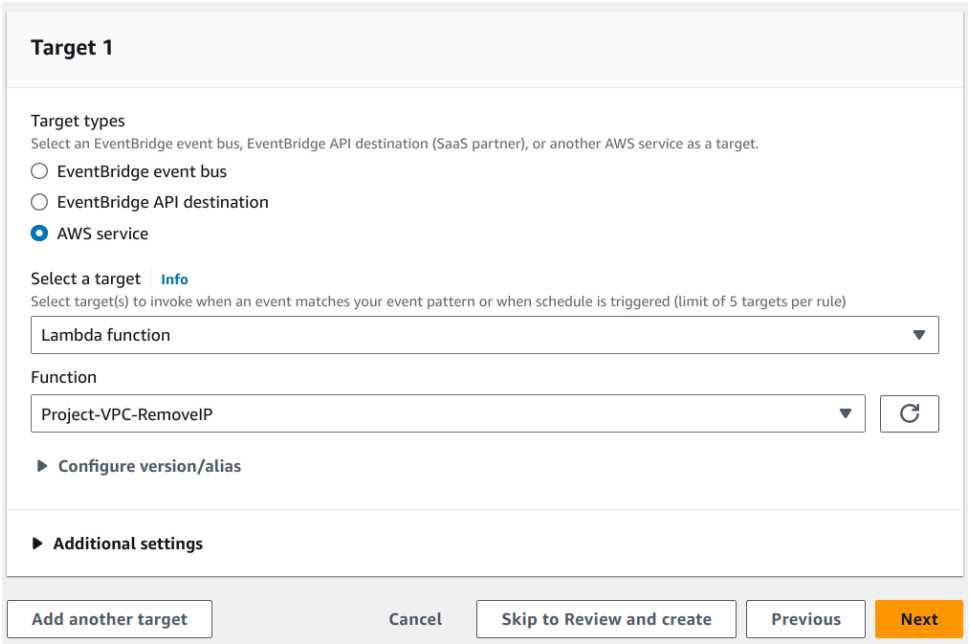
Lambda for the event trigger. The lambda function will retrieve the instance id of the terminated instances from the event record. Then delete the matching record in the S3 document of private IPs and again upload the modified document to S3.



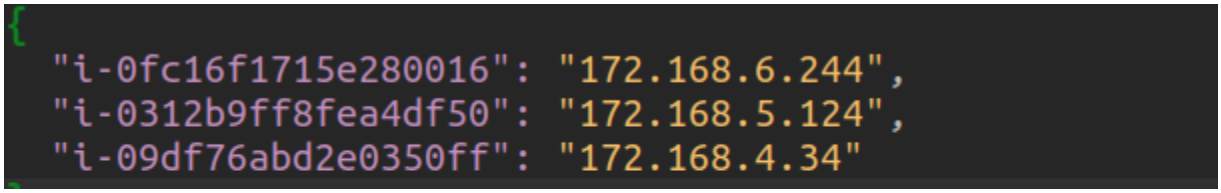
The rule will catch instance termination events and trigger the lambda function.



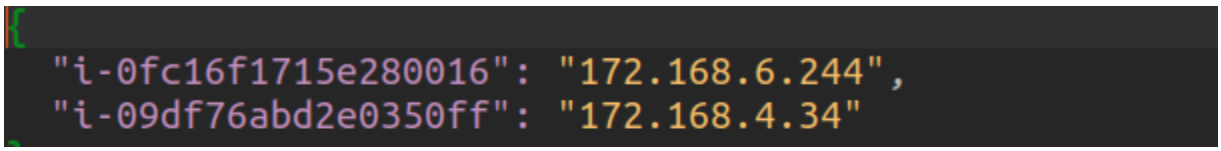


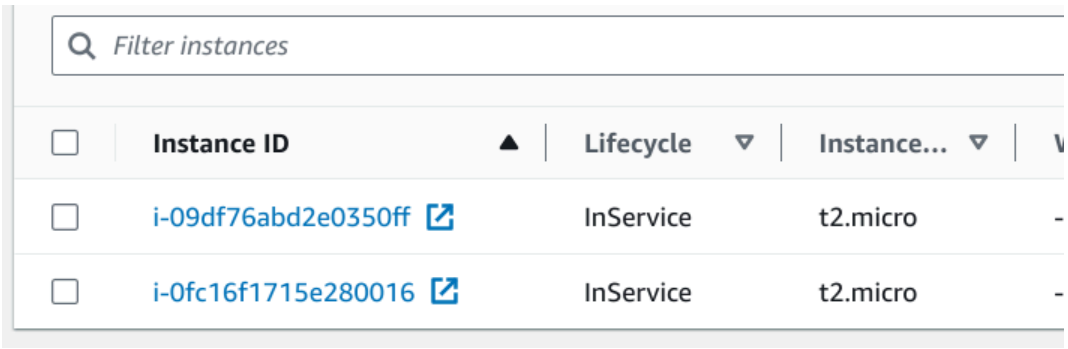


Step 28 - Currently the file which stores the private IPs of the instances contain the following.



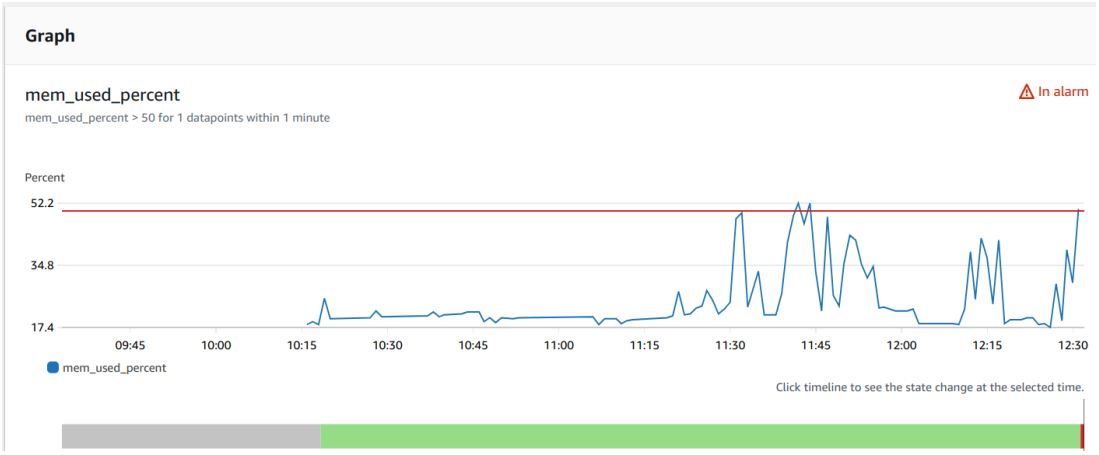
After changing the desired capacity of the ASG back to two, one of the instances will be terminated and through the event bridge rule, this event will trigger the lambda function to remove the associated IP address from the bucket object.





Step 29 - Check the memory based auto scaling policy. Run stress command in one of the private instances in the ASG.

```
[root@ip-172-168-5-124 /]# stress -c 2 -i 1 -m 1 --vm-bytes 558M
stress: info: [4347] dispatching hogs: 2 cpu, 1 io, 1 vm, 0 hdd
```



The alarm turns to "in-alarm" state and triggers the auto scaling which launches one more instance.