**Case Study - AWS SYSOPS**

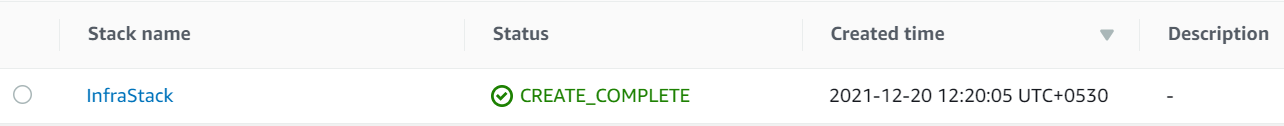
By

Geetha.M

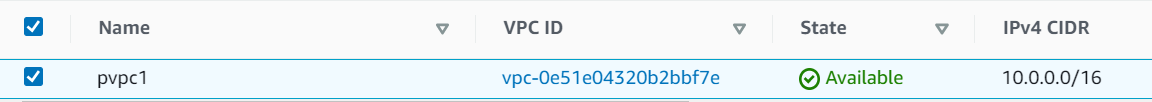
**Part -1: Automation**

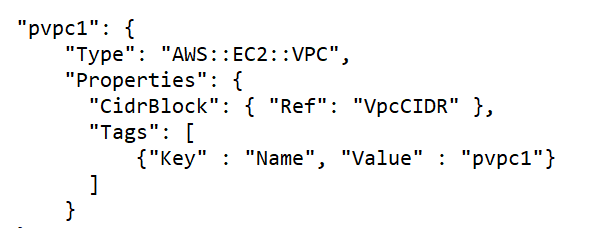
**Initial infrastructure using CloudFormation:**

1. Created CloudFormation stack to build infrastructure consists of VPC, Subnets, Internet gateway, NAT gateway, Route tables

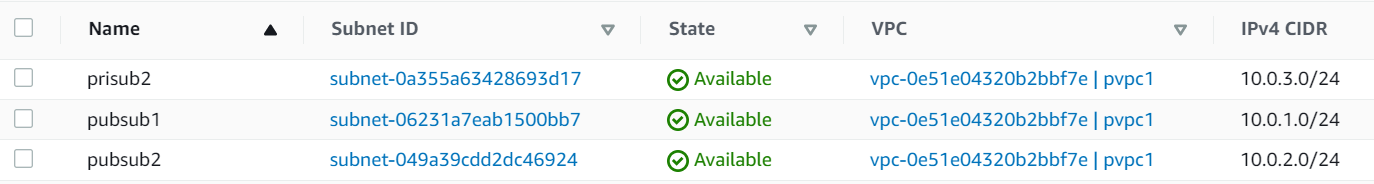


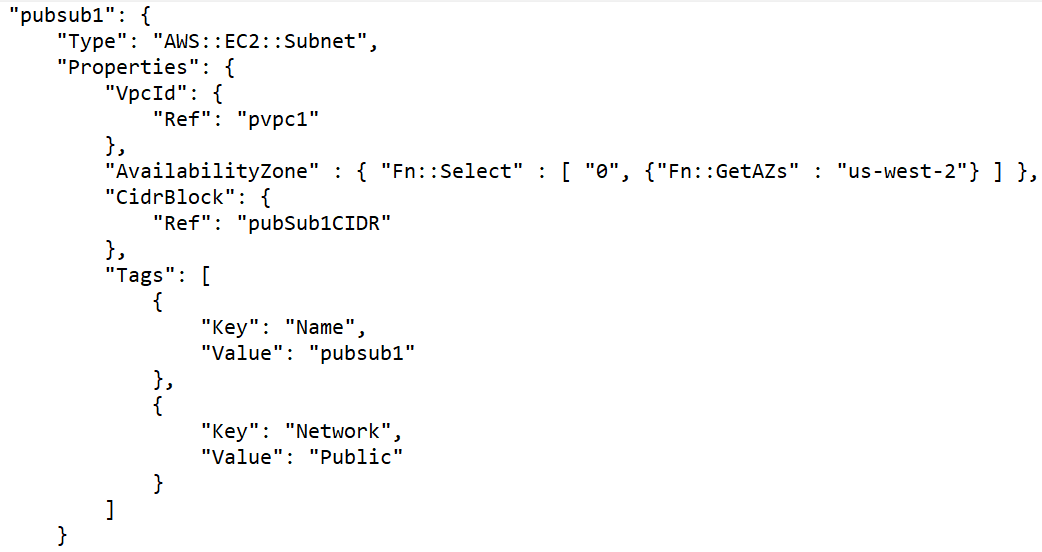
1. VPC has been created for CICR block 10.0.0.0/16 with name pvpc1





1. Created two public subnets( pubsub1 & pubsub2) and one private subnet(prisub1) with CIDR blocks 10.0.1.0/24, 10.0.2.0/24 and 10.0.3.0/24 respectively



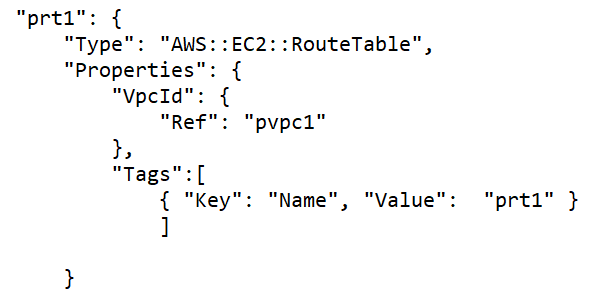


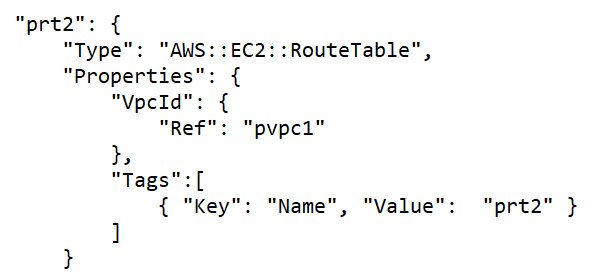




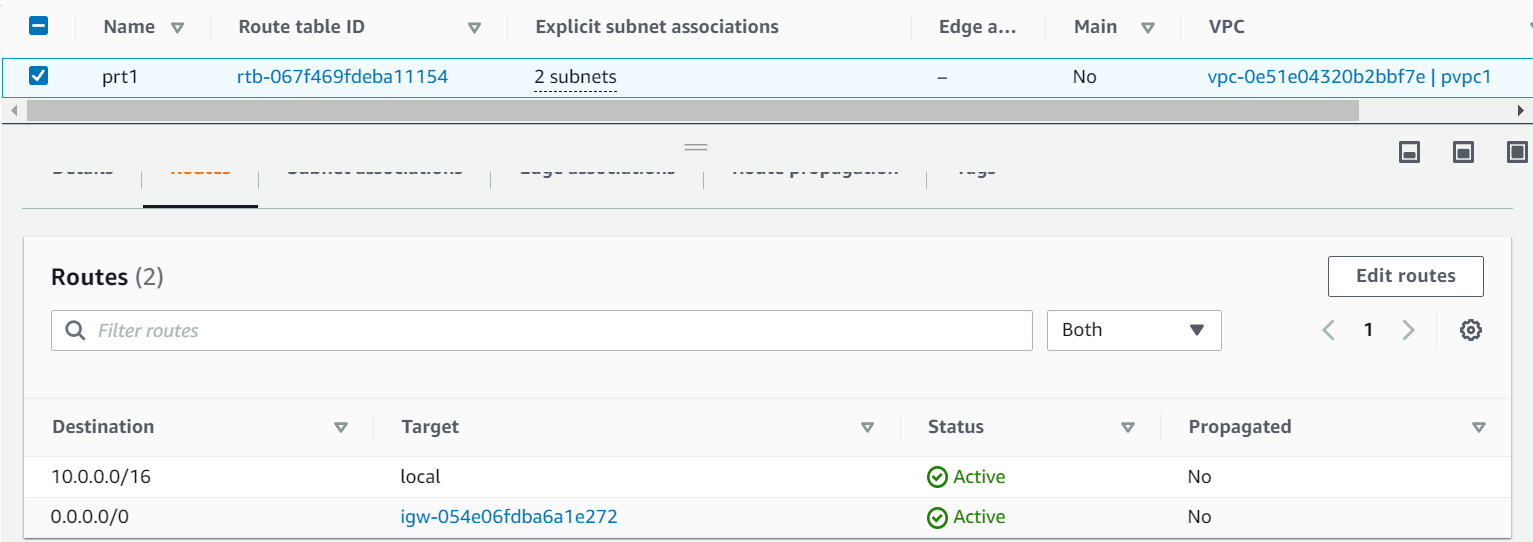
1. Created routing tables namely prt1 & prt2 and associated to subnets prt1 🡪 pubsub1 and pubsub2 and prt2 🡪 prisub1

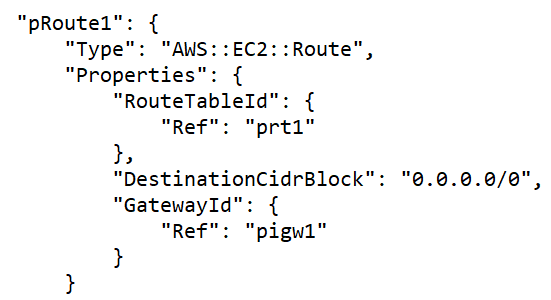


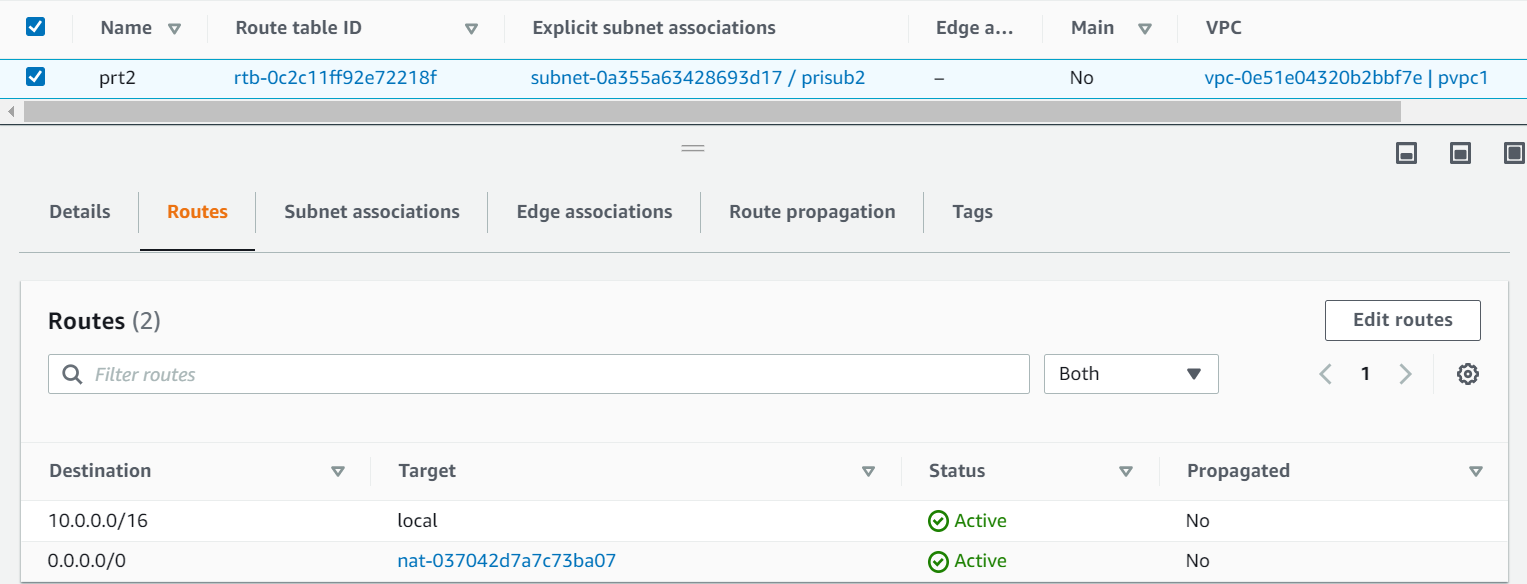


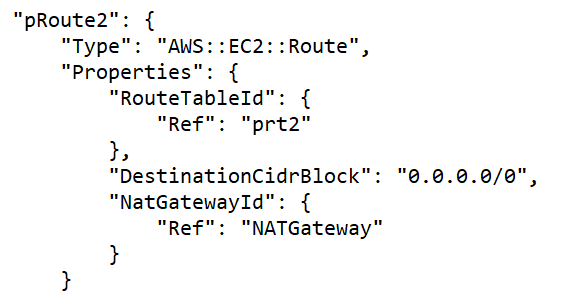


1. Created routes to local, internet gateway and NAT gateway

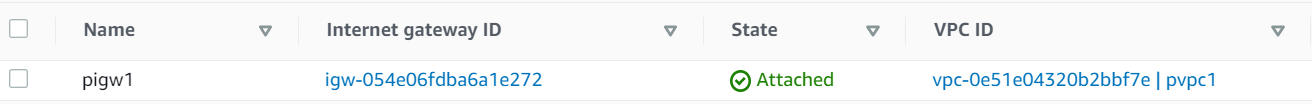






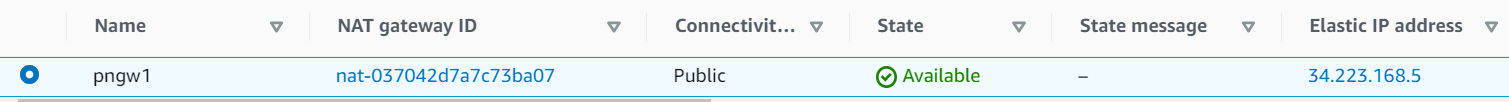


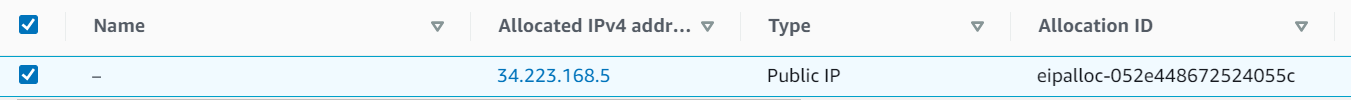
1. Created internet gateway(pigw1) attached to vpc

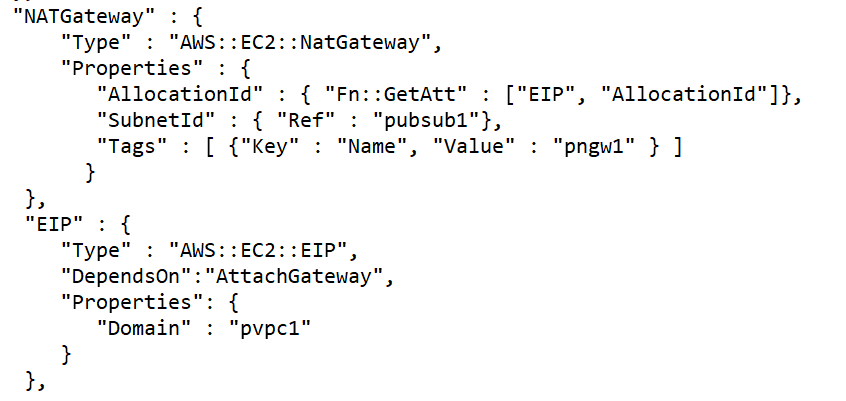




1. Created NAT gateway and Elastic IP

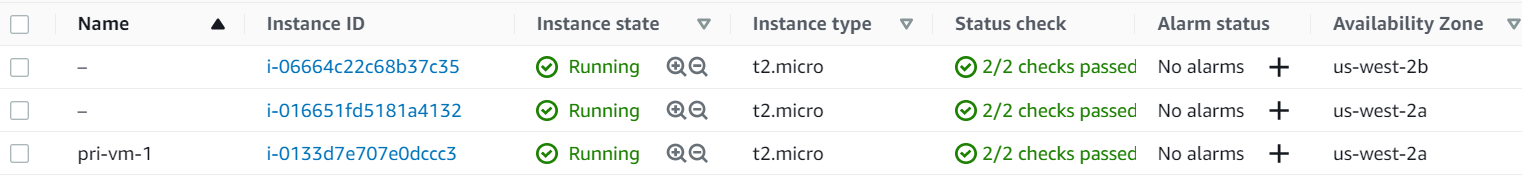




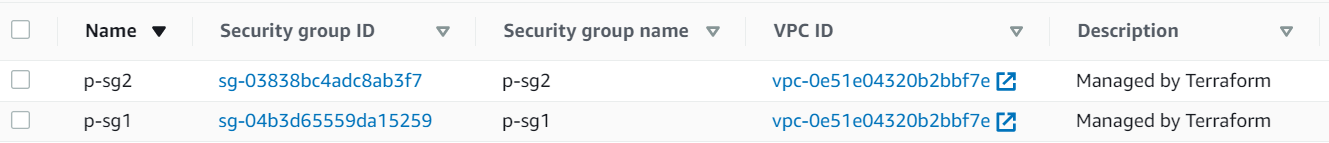


**Application Infrastructure using Terraform:**

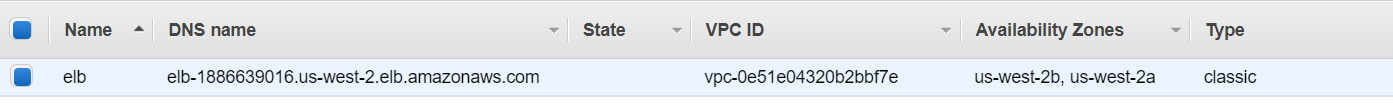
1. Created 2 Public VMs(pub-vm1 & pub-vm2) using launch configuration and one private VM(pri-vm1) – Ubuntu 18.04 with t2.micro instance type is used to launch VMs.



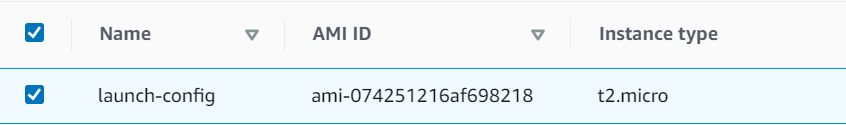
1. Created 2 security groups p-sg1 for ASG instances and p-sg2 for private instance



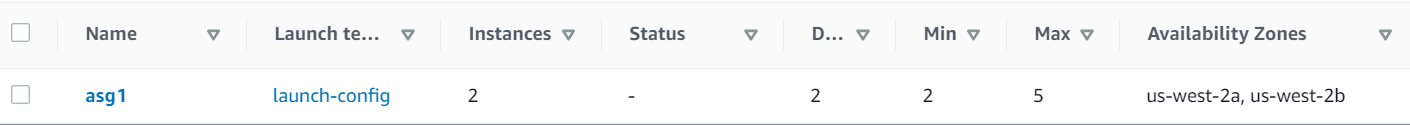
1. Created Elastic Load Balancer between two public VMs that are created in auto scaling group



1. Created launch configuration to deploy public VMs in ASG



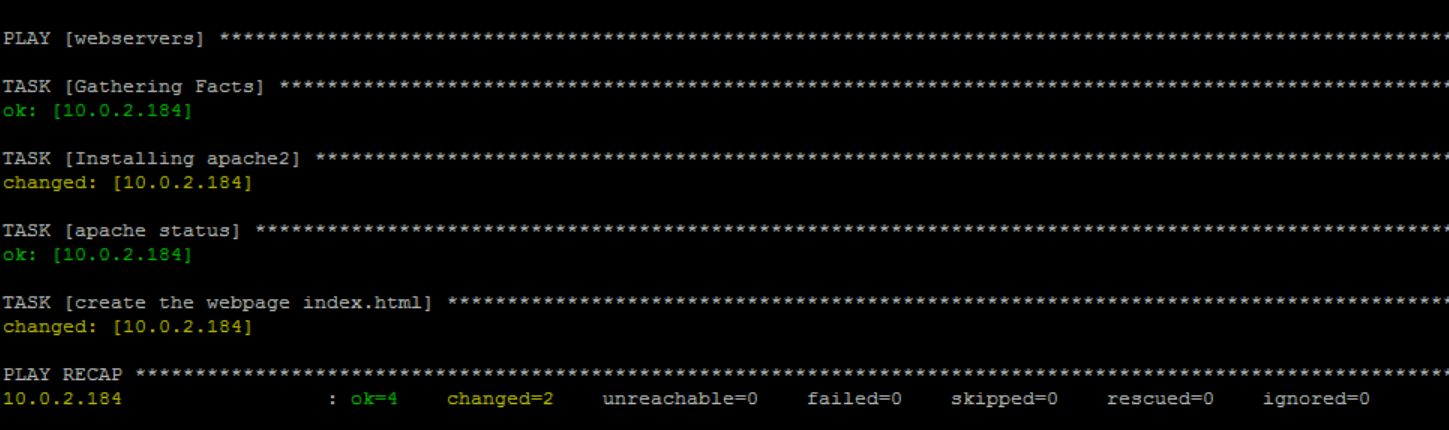
1. Created auto scaling group(asg1) with minimum 2, maximum 5 and desired capacity 2.

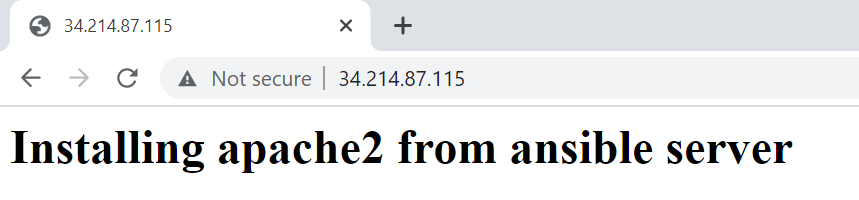


**Install Application using Ansible:**

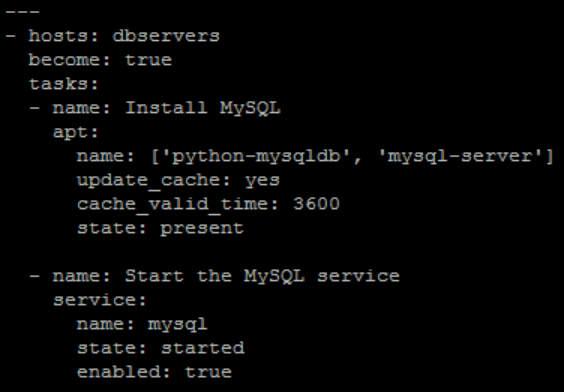
1. Apache2 is installed from ansible server(pub-vm1) to ansible node/apache server(pub-vm2), created an index file in ansible server and copied on the apache server using ansible. Accessed the apache server IP via browser to see the index file content.

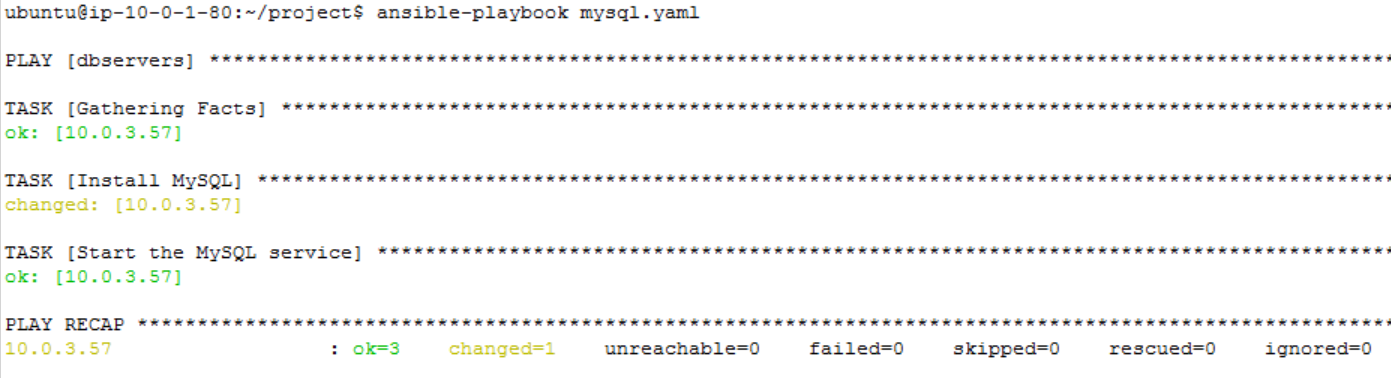


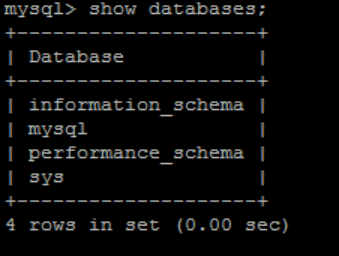


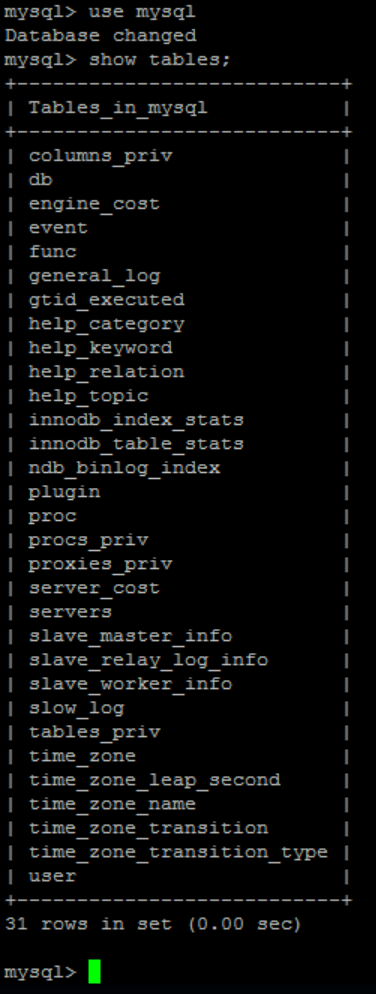


1. Installed MYSQL on private vm(pri-vm1) using ansible and able to see default databases.



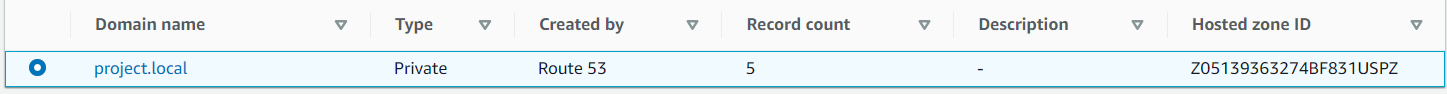


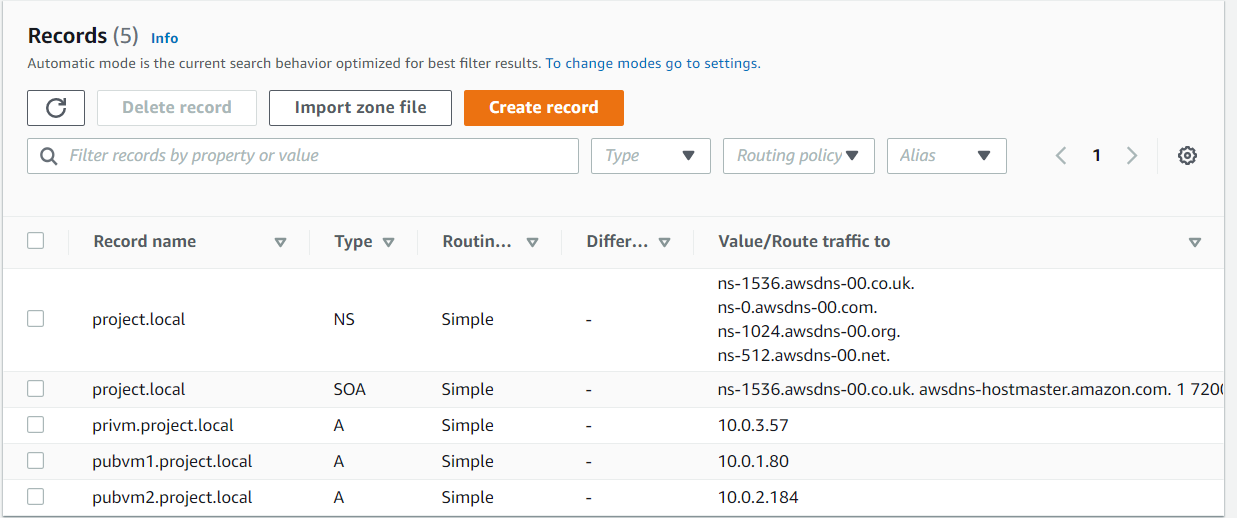




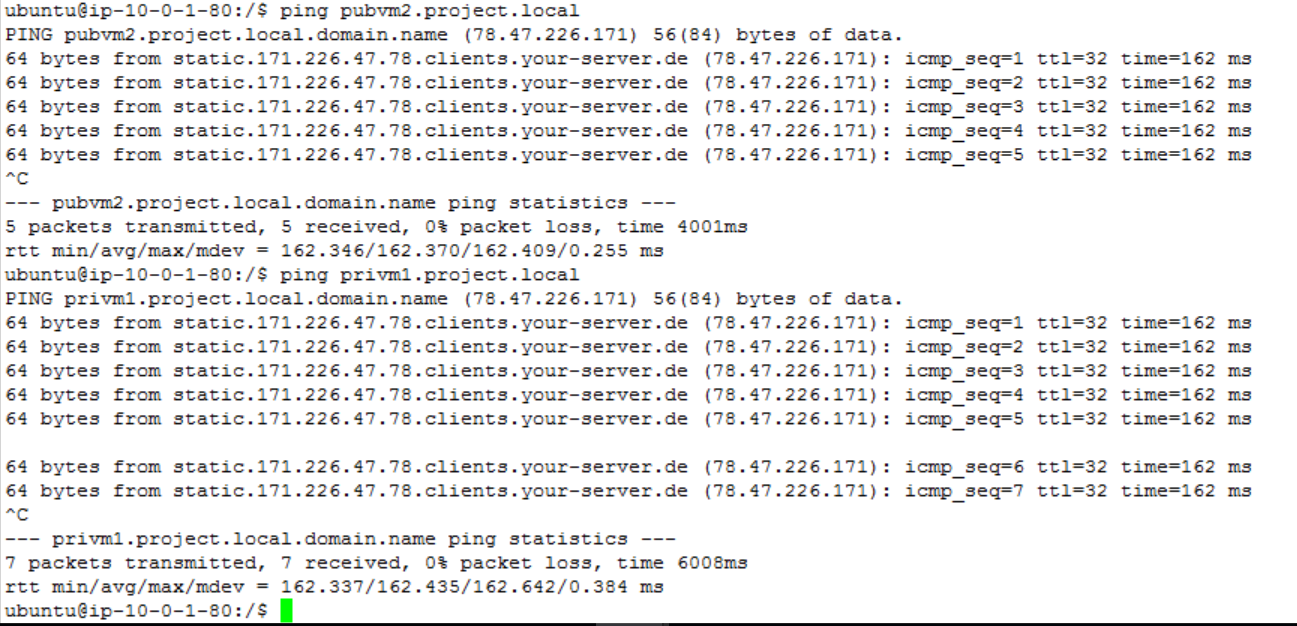
**Part - 2: Secure Communication and Security Management for Application (Manual Configuration)**

1. In Route53 configuration for the VPC internal communication, created hosted zone(project.local) and records in it. Servers are accessible using the record name.

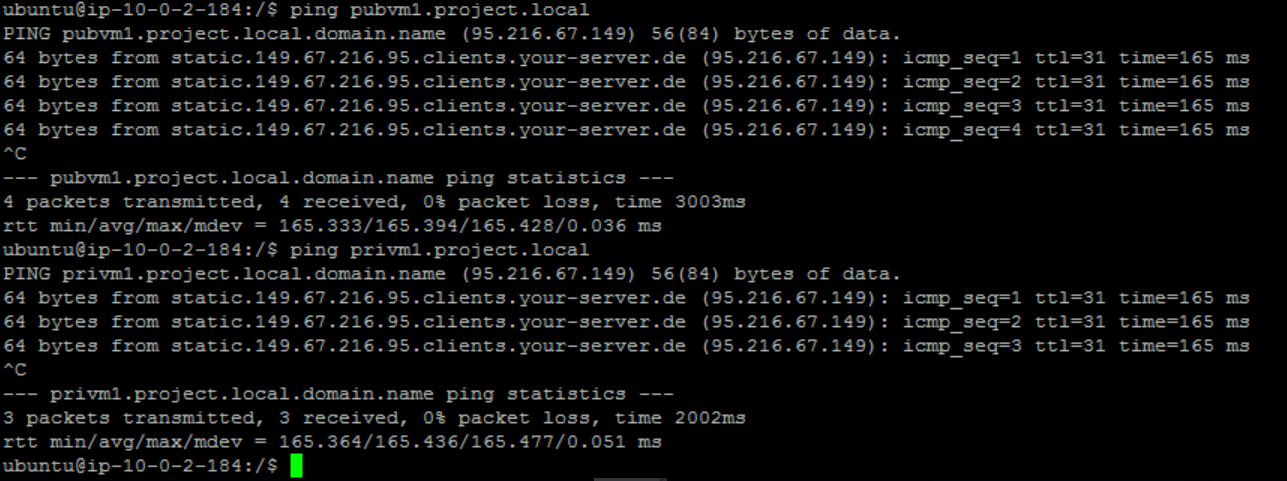




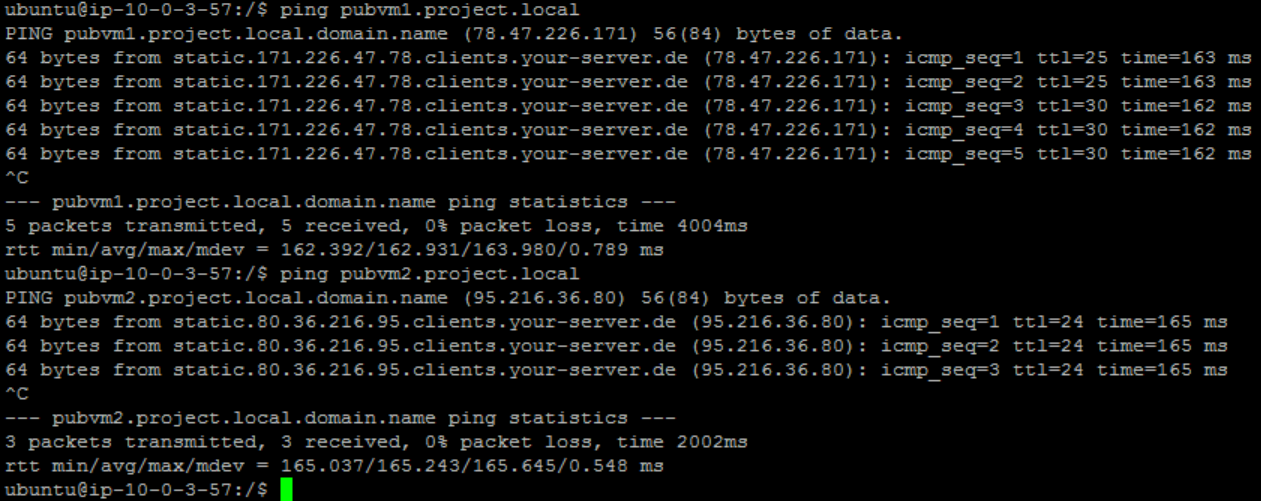
**From ansible server:**



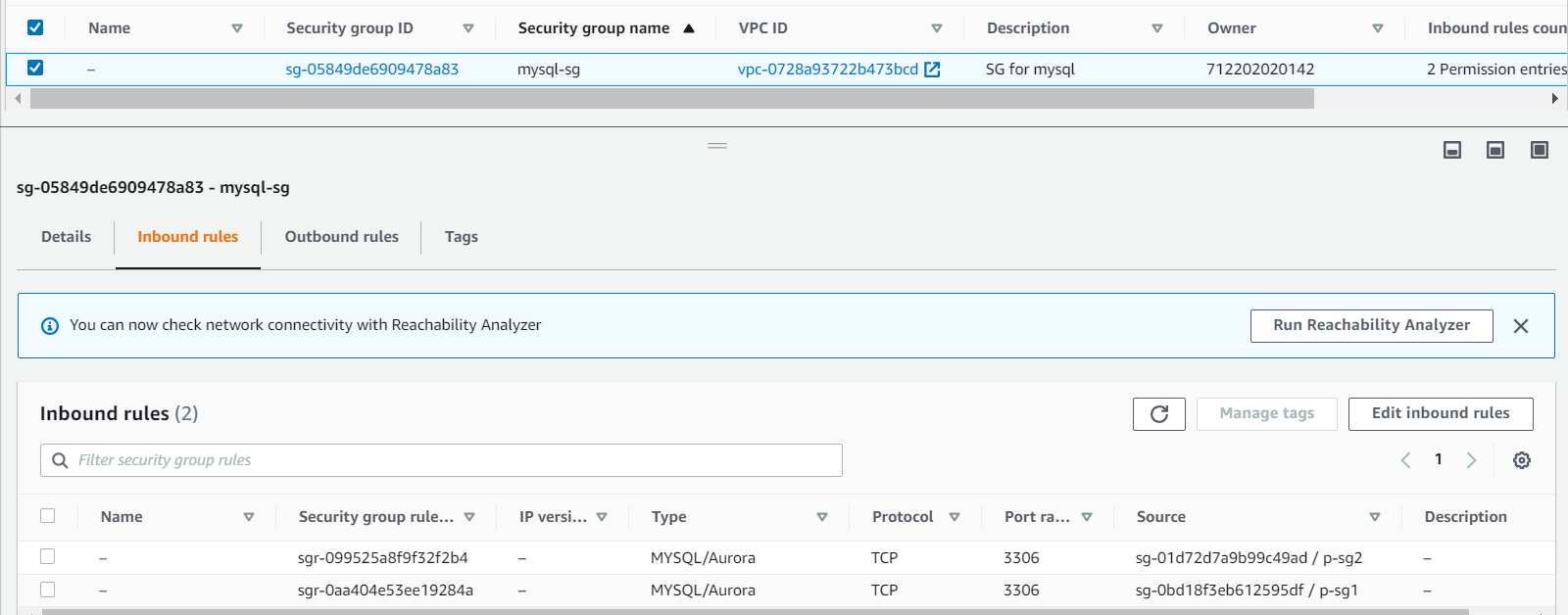
**From apache server:**



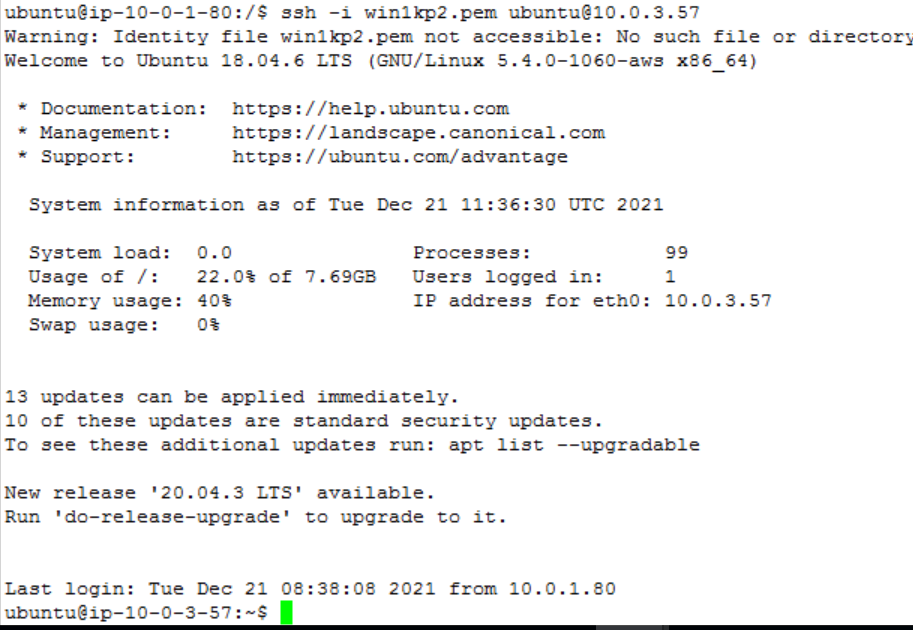
**From mysql server:**



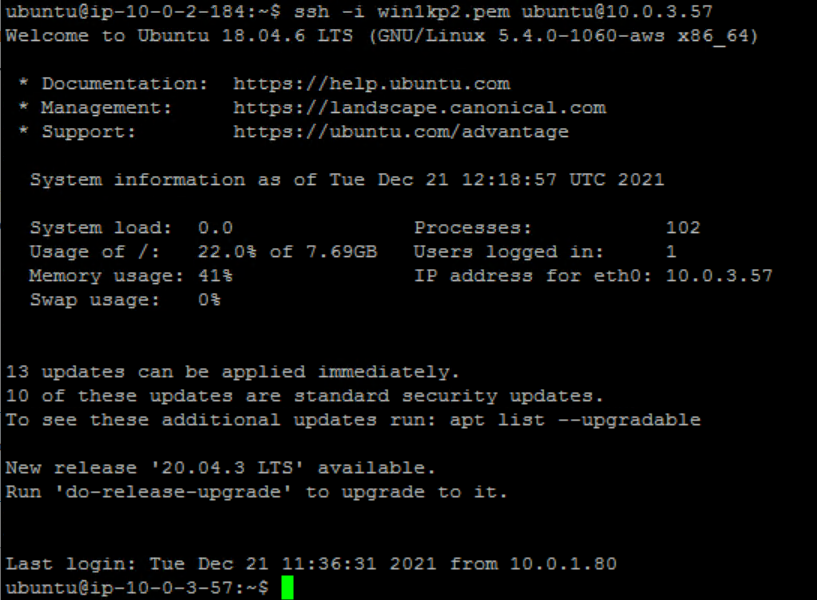
1. Created a security group for MYSQL DB and added the security group of ASG instances to access MYSQL DB EC2 instance from ASG instances. Tested with other EC2 instances and it is not accessible.



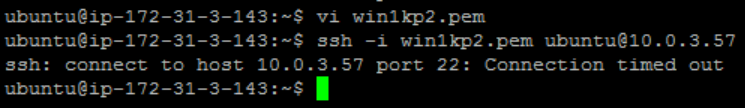
**MYSQL DB EC2 instance is accessible from pub-vm1:**



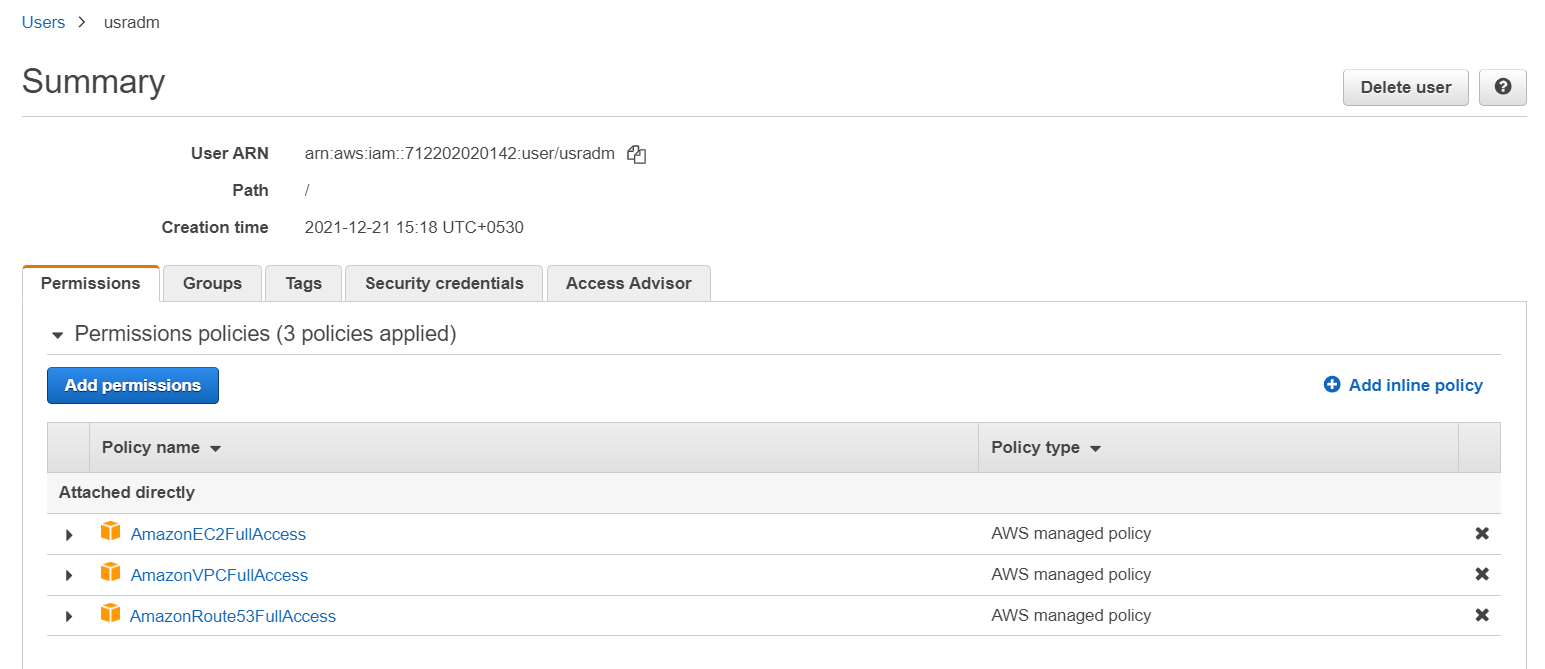
**MYSQL DB EC2 instance is accessible from pub-vm2:**



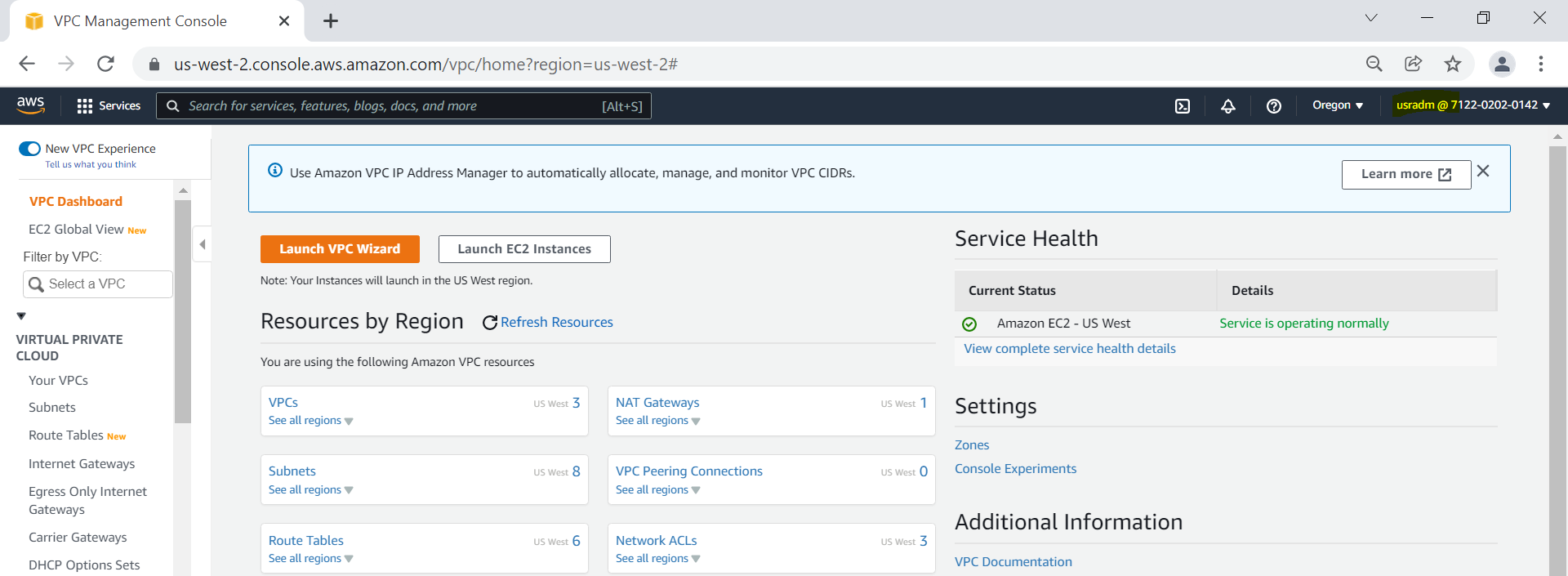
**MYSQL DB EC2 instance is not accessible from non-asgvm(test vm):**



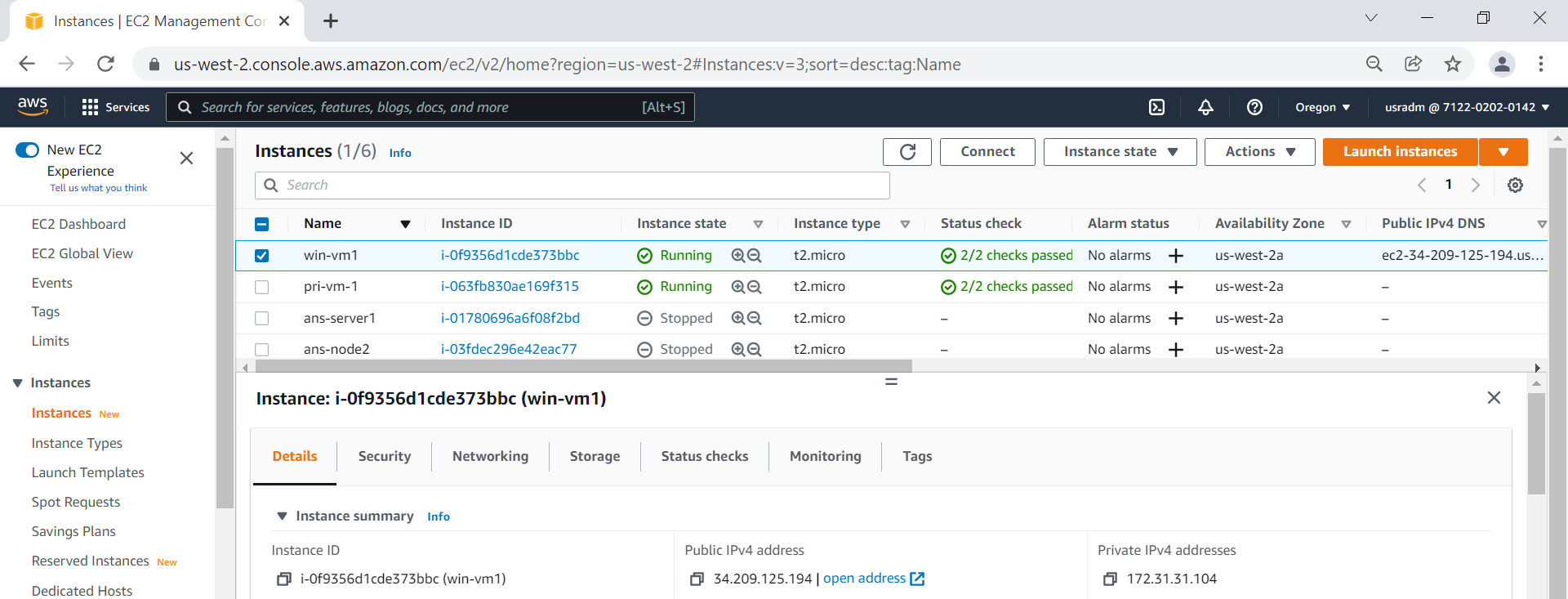
1. Created an user called usradm with full access to infrastructure VPC, EC2 and Route53. Logged in using the new user and check these resources and is accessible. Tested user (usradm) with resource S3 for which the permission is not granted.



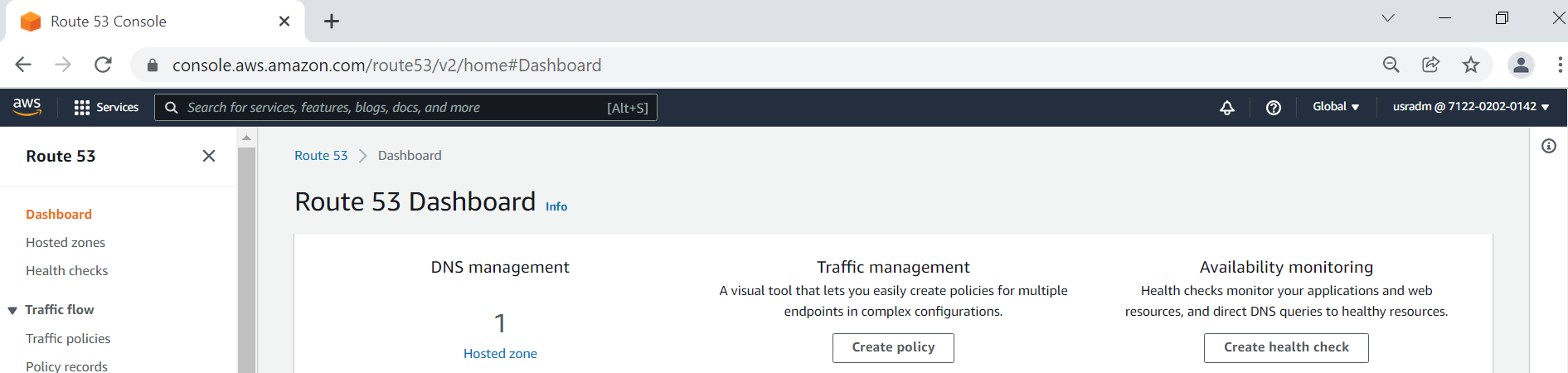
**User usradm can access VPC:**



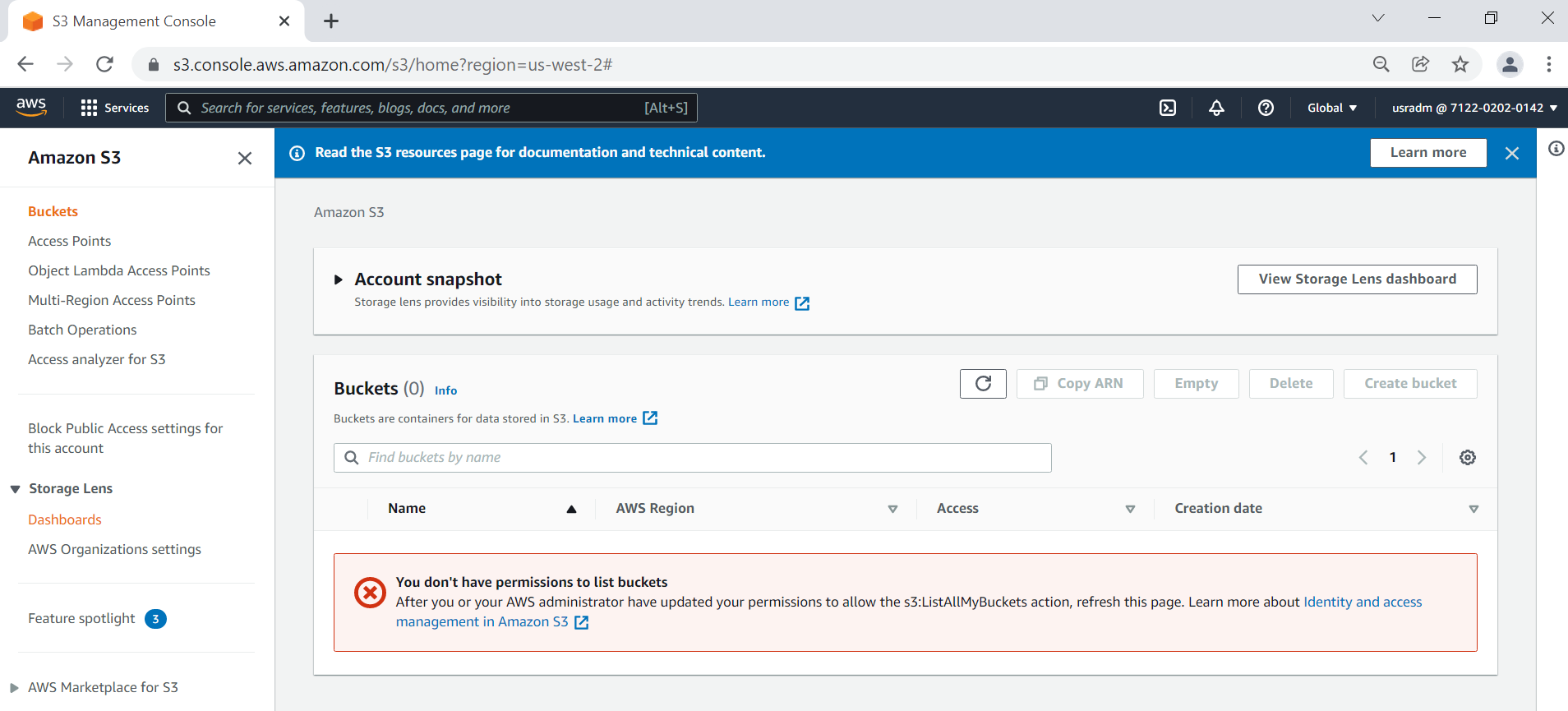
**User usradm can access EC2:**



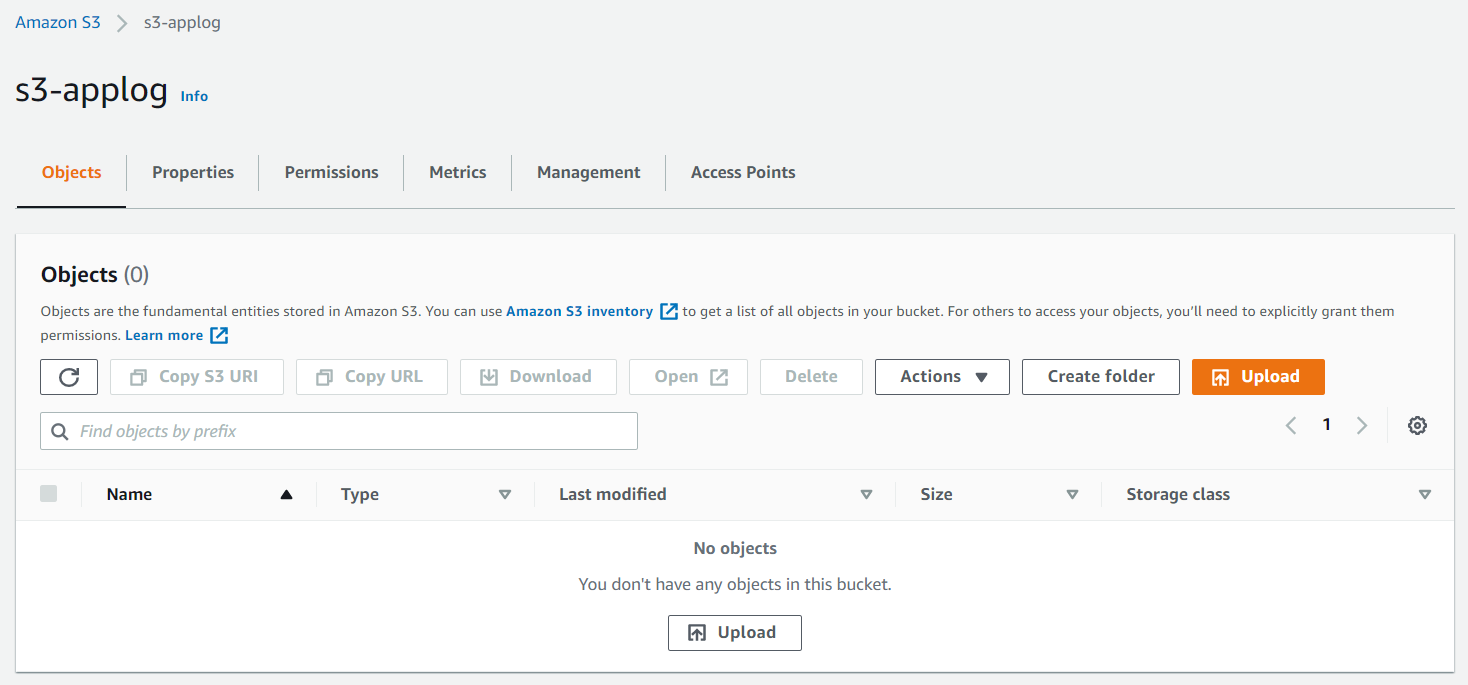
**User usradm can access Route53:**

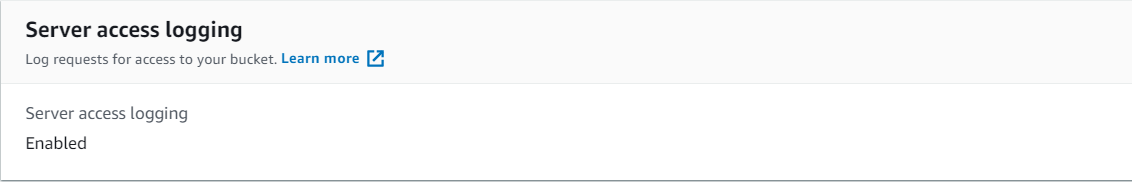


**User usradm is not able to access S3:**

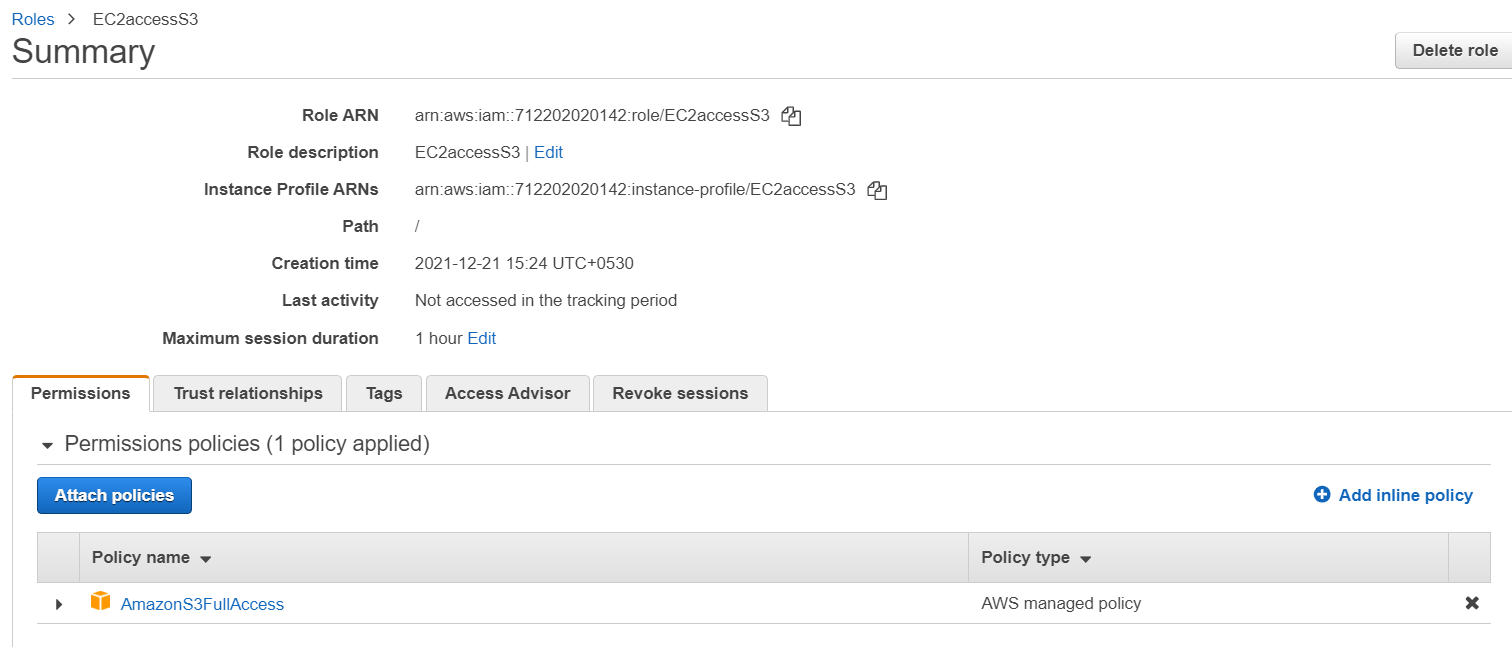


1. Created S3 bucket(s3-applog) to store log files. Server access logging is enabled.



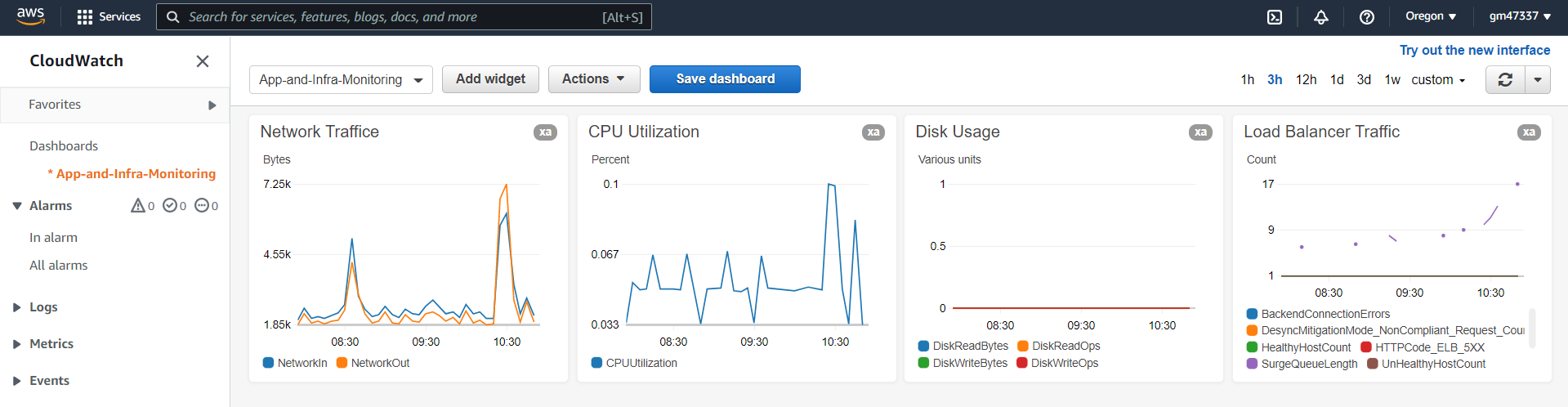


1. Created IAM roles to have permission to access S3 for all EC2 instances



**Part - 3: Monitoring App and App Infrastructure (Manual Configuration)**

1. Create a CloudWatch Dashboard to monitor Network I/O traffic, CPU Utilization, Disk usage and Load Balancer traffic



--- End of the document ---