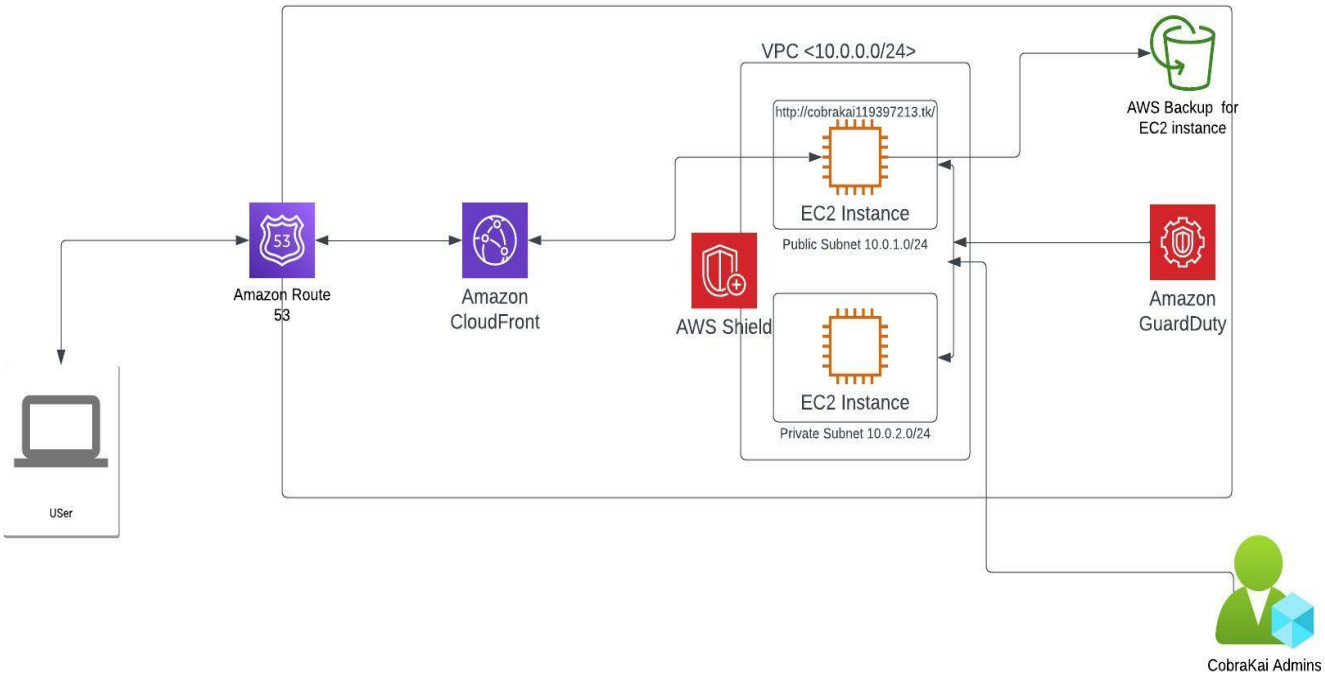
# ENPM665 - Final

**By Adithya Srinivas Parthasarathy**
**UID:119397213**

I wanted to provide seven recommendations after moving into AWS from the current on premise server. These recommendations include:

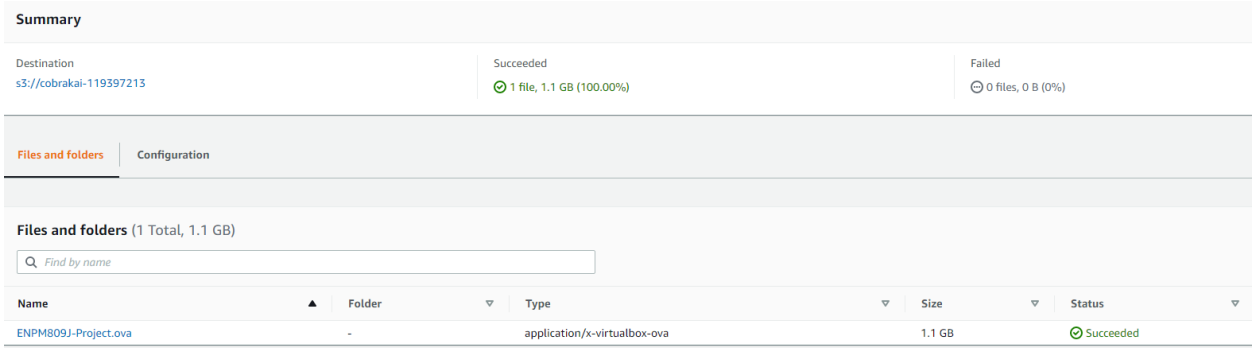
- IAM
- VPCs
- Security groups
- On demand Backup
- Route53
- Guard Duty
- Cloud Trail
- Maice
- CloudFront

The re-architected diagram of the entire scenario is :



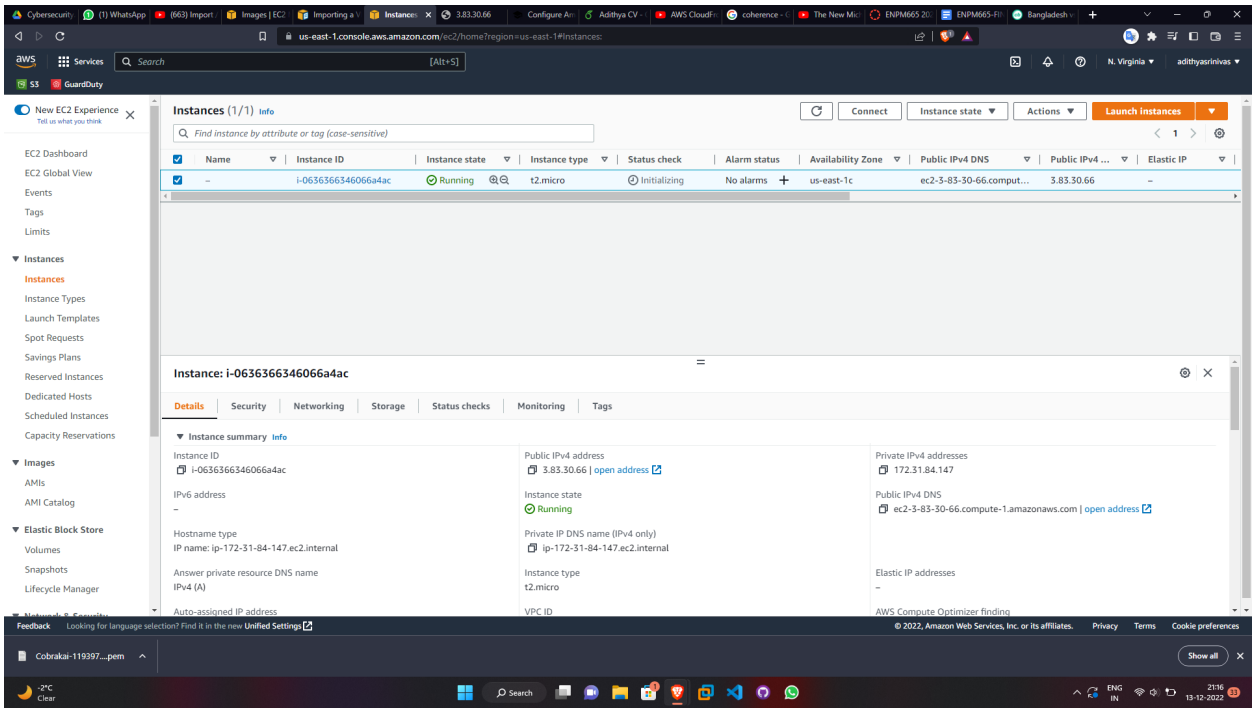
Firstly we migrate the on premise server to aws as suggested with help of S3 bucket.

We do this by taking the image given and by adding it to S3 bucket with certain configurations of route-policy.json and trust-policy.json inorder to correctly import the given image into the bucket.




Once that is done, now with the help of Amazon Machine Image (AMI) that we just imported, we can run an EC2 instance with certain security configurations to get the most out of it.

So the idea is to create Virtual Private clouds, security groups, network access control list (ACL) , so that we could add these to our EC2 instance that we will create out of the AMI.



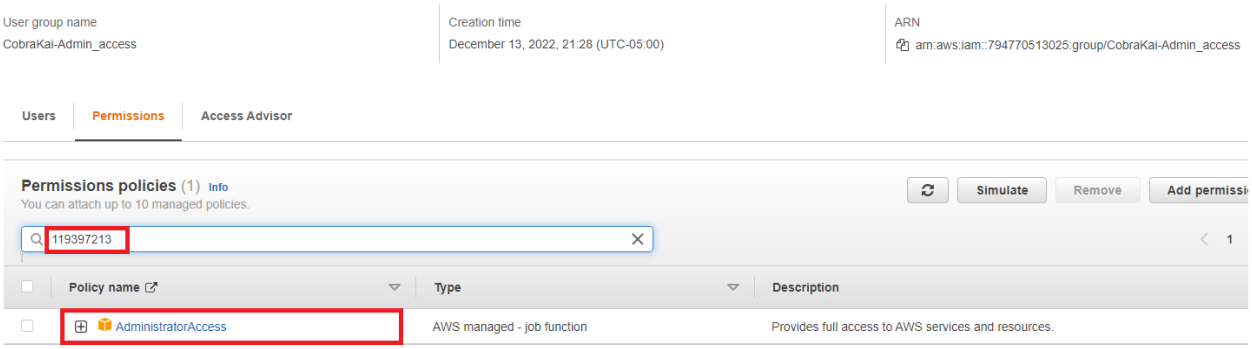
# 1) IAM:

Management of identity and access to technological resources is the process of ensuring that the right people have access to the right resources through policies. We could do this by formation of policies and groups which have permission for only certain resources.

We create three **groups** for this purpose:

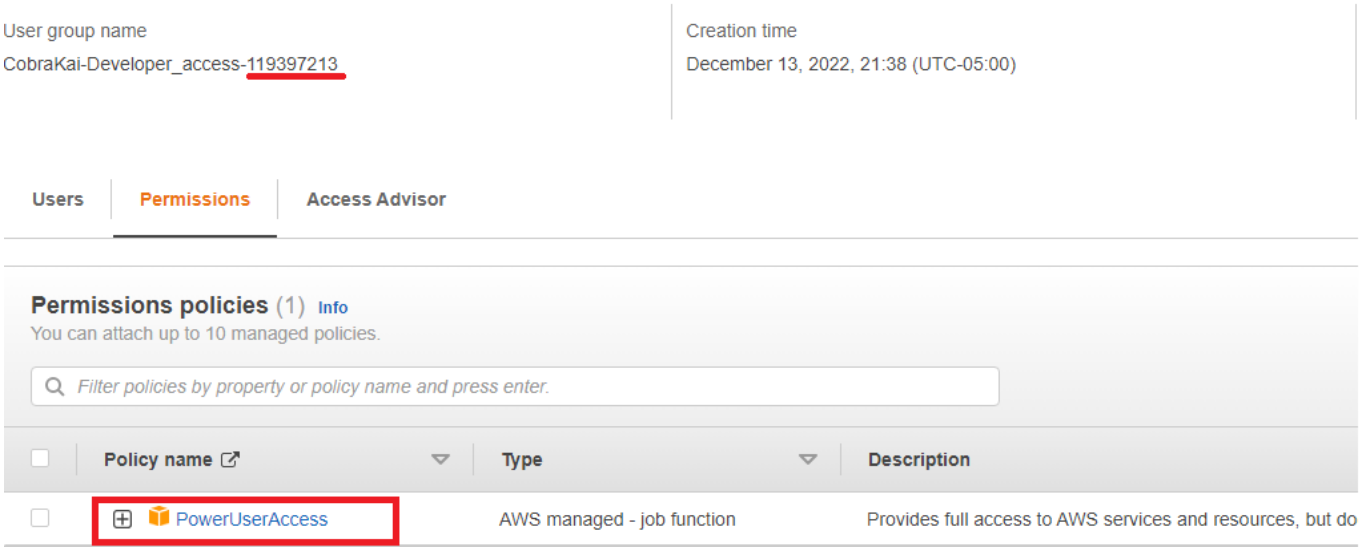
- Admin
- Developer
- System Administrators

i)**Admin**



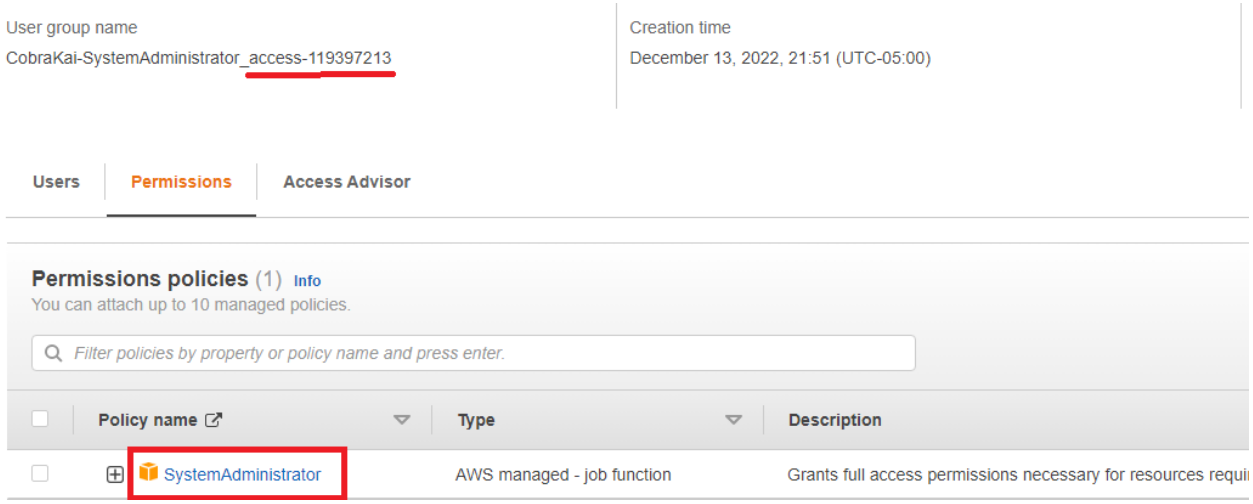

The administrator group is provided with 'AdminstratorAccess' policy which makes sure all the access is granted. We make this for C-Suite executives and founder.

ii)**Developer**



The Developer group is provided with 'PowerUserAccess' policy which provides full access to AWS resources and services, but does not allow management of users and groups. This group is created for Developers.

iii)**system administration**:




The System administrator group is provided with 'SystemAdminstrator' policy which grants full access permissions necessary for resources required for development operations.We make this for system admins.

**Users:**

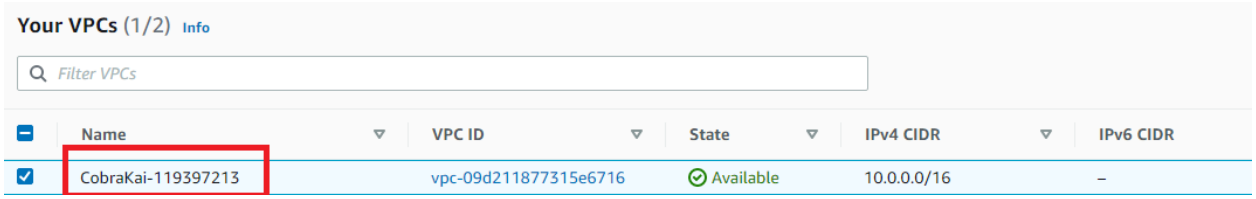
We add a sample of 5 users as given to us in the document. We add all the C-Suite executives (Johnny Lawrence, Miguel Diaz, Aisha Robinson, Eli "Hawk" Moskowitz) to the CobraKai_admin_acess group.

We add Demetri to the Developer-access group followed by Bert to SystemAdministrator groups respectively.

| | | | | | |
|---|---|---|---|---|---|
| | AishaRobinson | CobraKai-Admin_access | Never | None | None |
| | Bert | CobraKai-SystemAdministrator_access-119397213 | Never | None | None |
| | Demetri | CobraKai-Developer_access-119397213 | Never | None | None |
| | Eli_Hawk_Moskowitz | CobraKai-Admin_access | Never | None | None |
| | JohnnyLawrence | CobraKai-Admin_access | Never | None | None |
| | MiguelDiaz | CobraKai-Admin_access | Never | None | None |

## 2) VPCs:

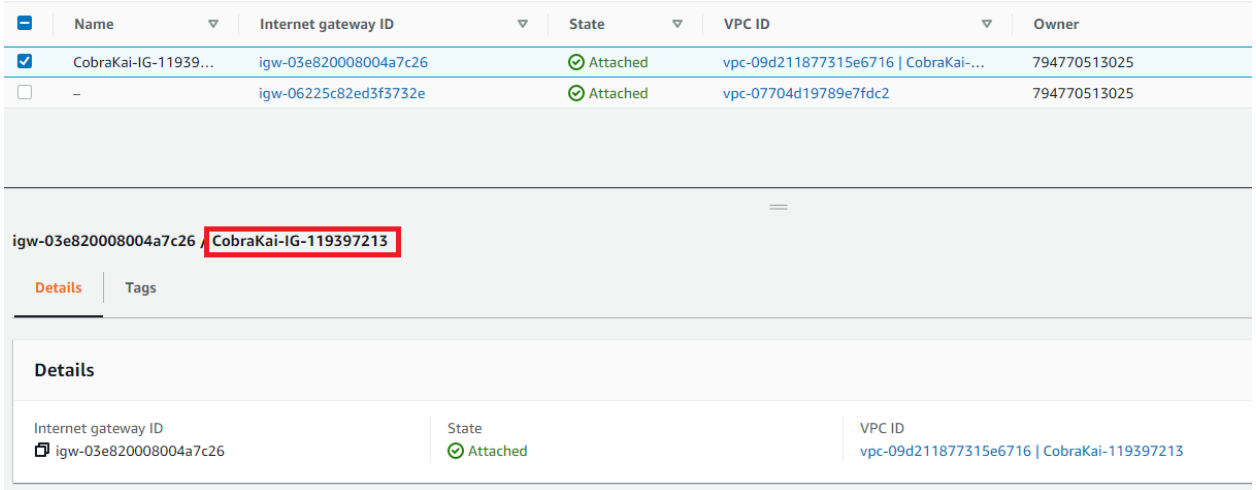
An Amazon Virtual Private Cloud (Amazon VPC) provides you with a virtual network where you can launch Amazon Web Services resources. The AWS infrastructure enables you to operate this virtual network in a similar way to a traditional data center network with the added bonus of scaling with AWS resources. It provides us with a virtual private network in a given public cloud environment. This can be achieved using Subnets, Network Access control lists and Internet gateways in order to use the VPC to allow only access to certain IP address.




We name the VPC as CobraKai-119397213 where '119397213 ' is the UID.

2.1) **Internet Gateways**:

To connect the VPCs to another network we usually need a Gateway. The Internet Gateway helps us to connect our VPC (CobraKai-119397213) to the internet.

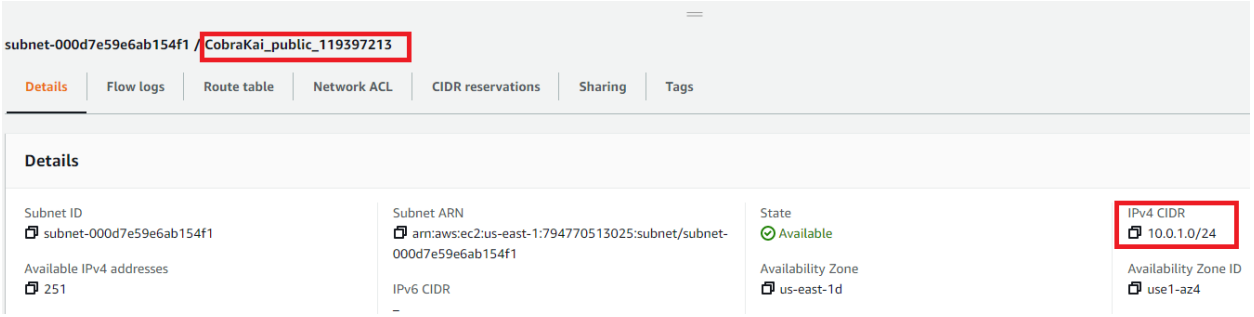



We attach our already existing VPC that we created for this purpose.

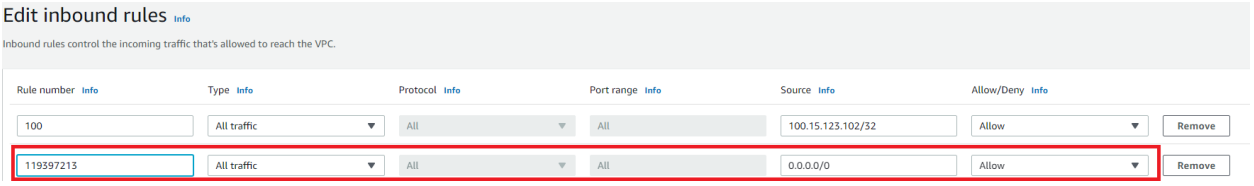
2.2) **Subnets**:

An IP network can be divided logically into subnetworks. Creating a subnet is one way to reduce traffic on a larger network by splitting it into smaller, interconnected networks. As a result, traffic does not have to travel through unnecessary routes, which increases the speed of the network. So we basically split it as a public subnet and private subnet each with its own IP ranges.We need this division since the resources that need to be accessed across the internet need to be in a public subnet, while the ones that don't need to be accessed should be in a private network.

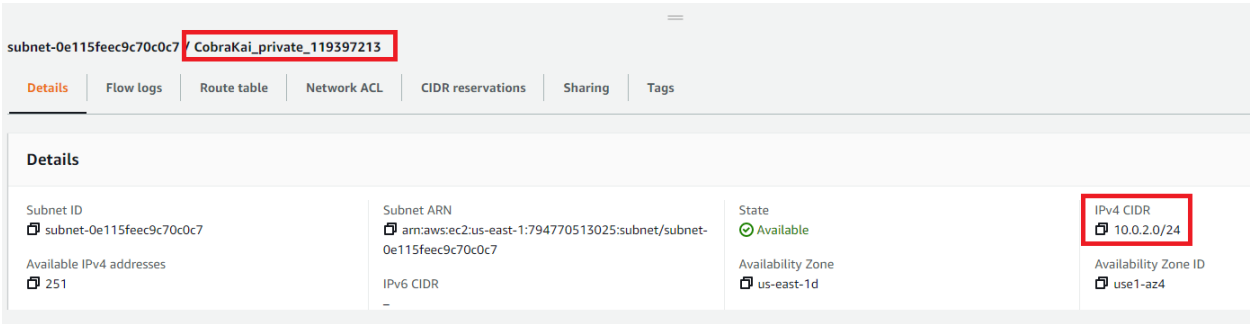
Public subnet:




Here we assign the CIDR range of **10.0.1.0/24** for the public subnet.
For the public subnet we created, we add a routing rule as 0.0.0.0/24 inorder to allow access from anywhere from the internet.

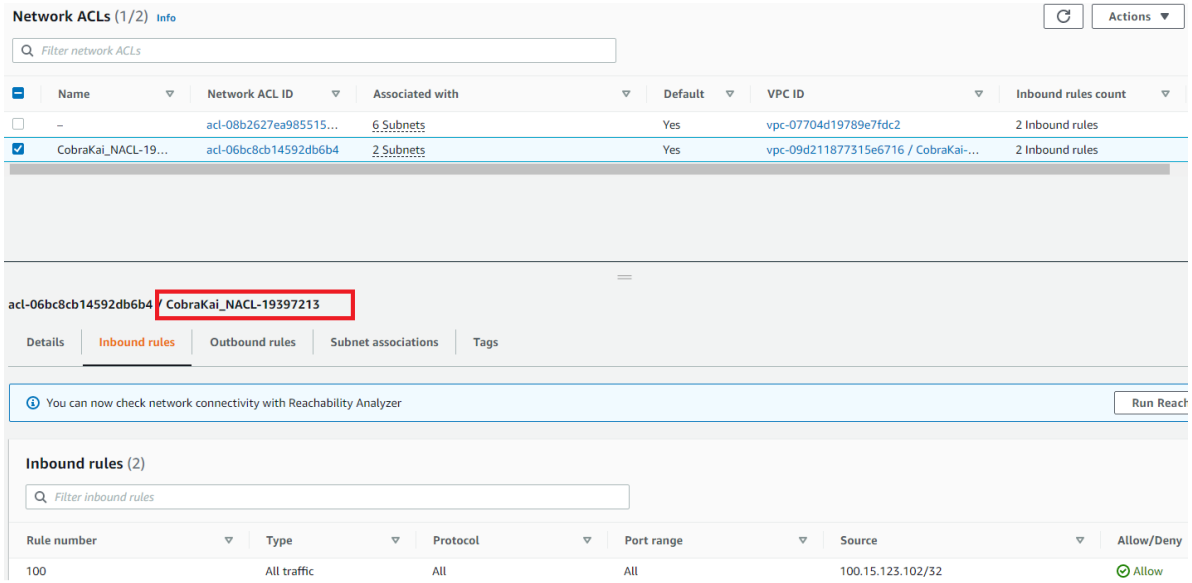



Private subnet:



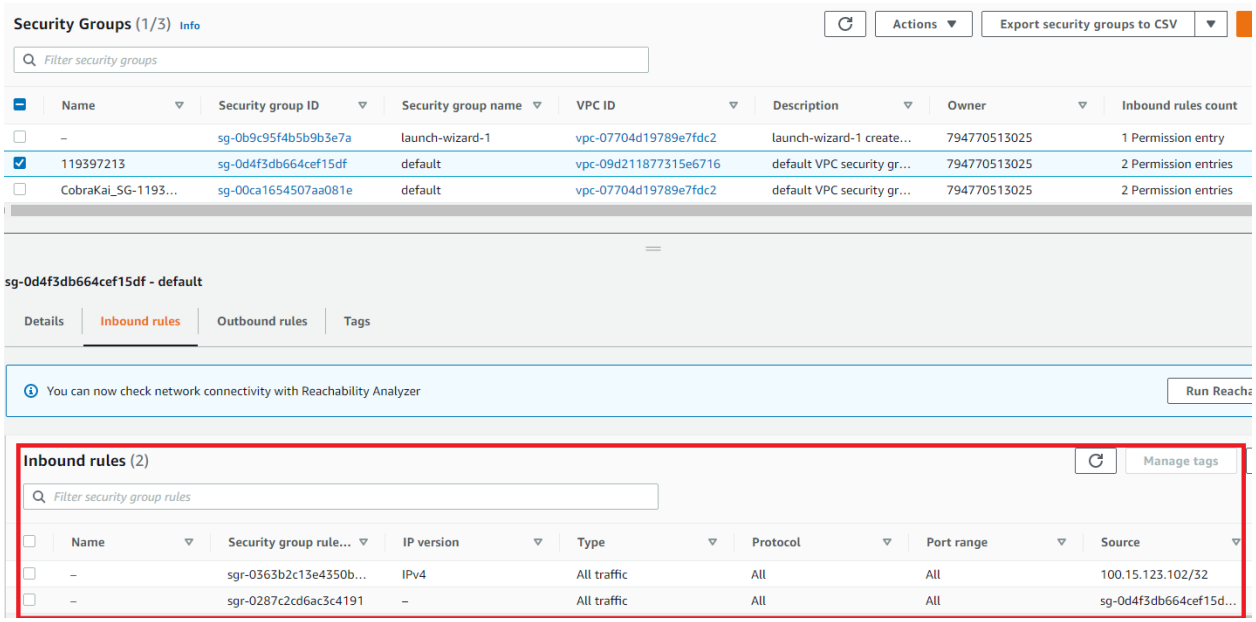

Here we assign the CIDR range of 10.0.2.0/24 for the private subnet. And we save subnet associations with each subnet respectively.

2.3) **Network Access Control List**:

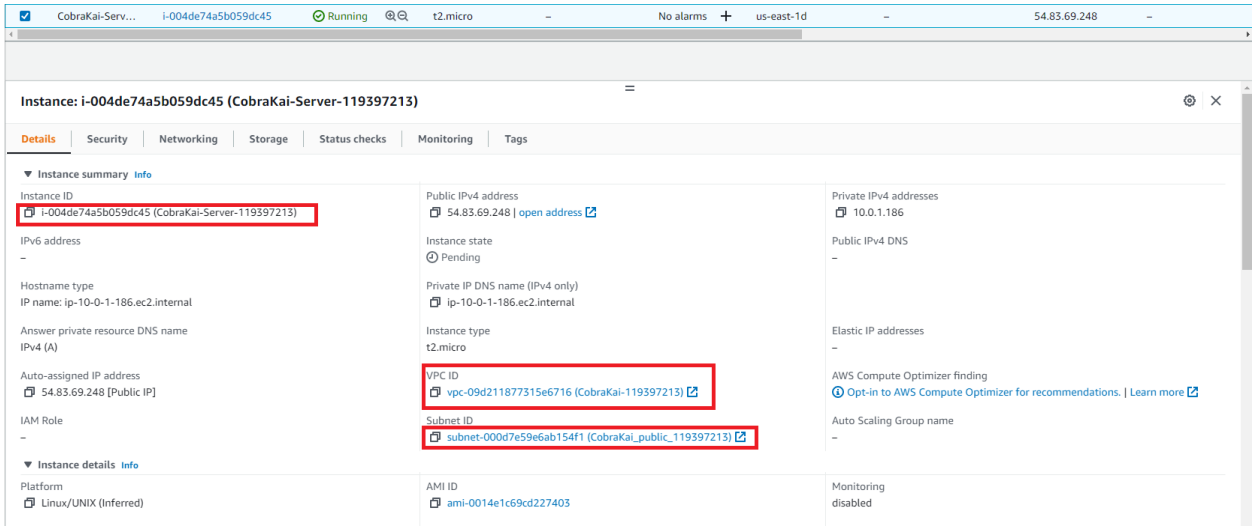


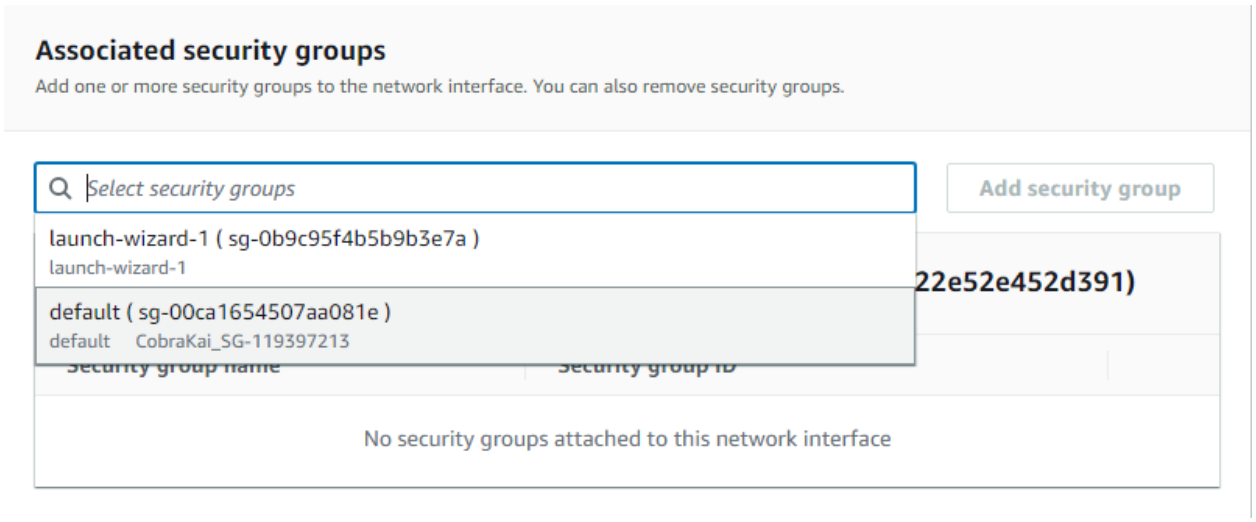

## 2.4) Security groups:

We add security group with set of defined inbound and outbound rules:



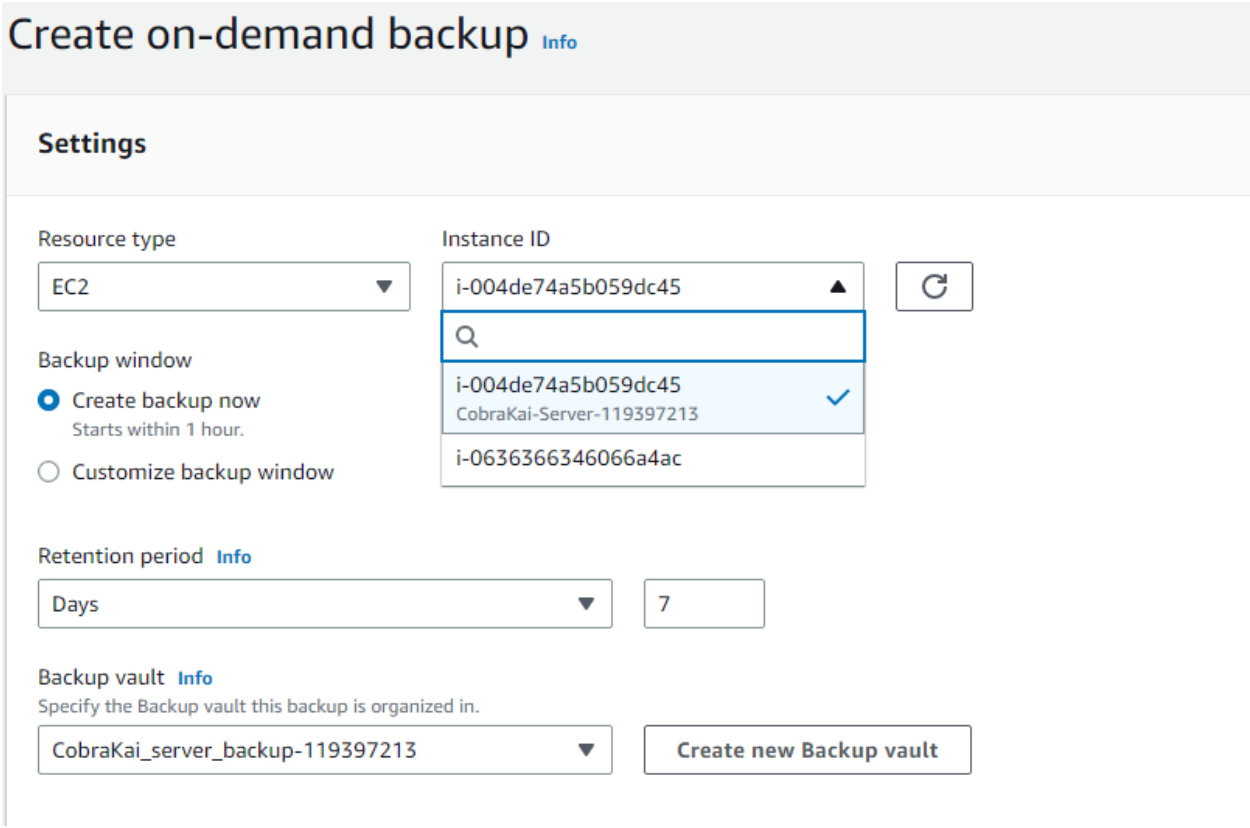

**With all this created, we can now spin off an EC2 instance with created Internet gateway, subnets and VPCs.**




## 3) Creation of Backups:

There is a need for Backup of the EC2 instance because in case the instance is compromised or attacked under ransomware, the team could always retrieve the latest image of the instance . In addition to automating backup schedules and retention management, the team can create backup policies as well.

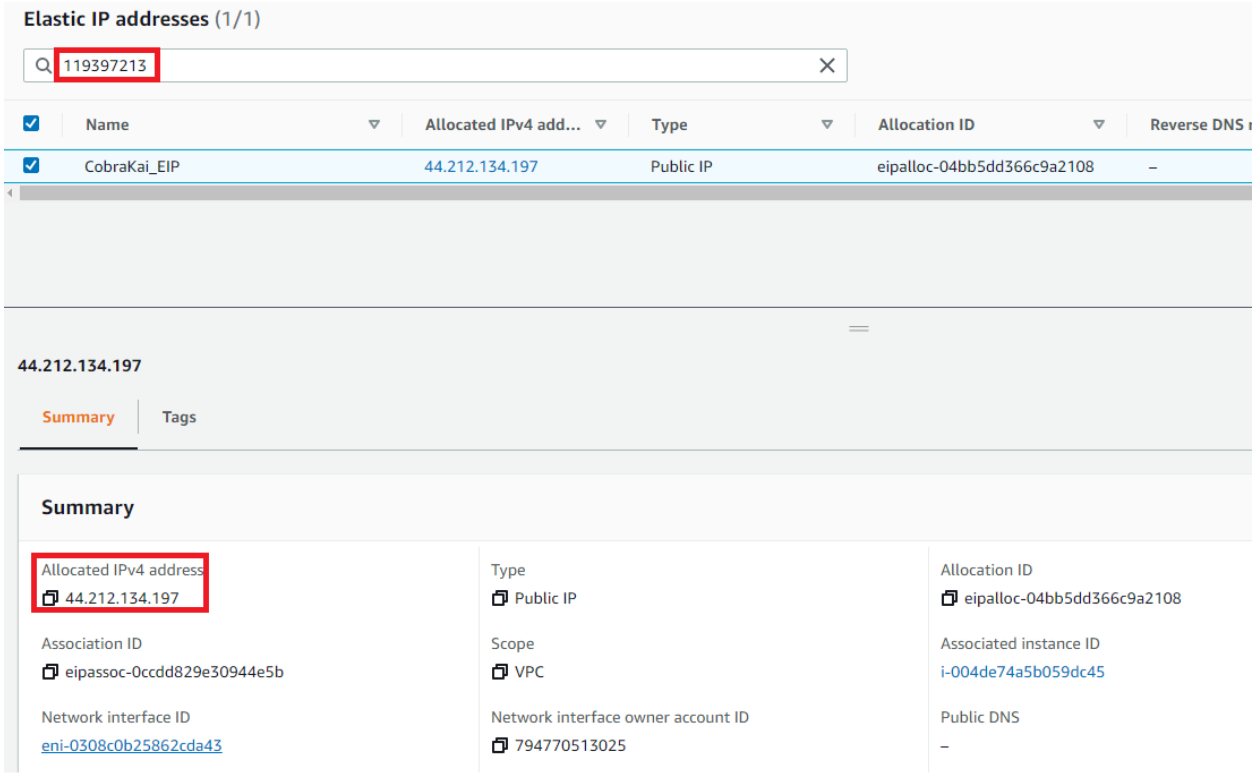



We choose our EC2 instance and set the retention period as 7 days (basically once every week). We create a separate backup vault for the same.

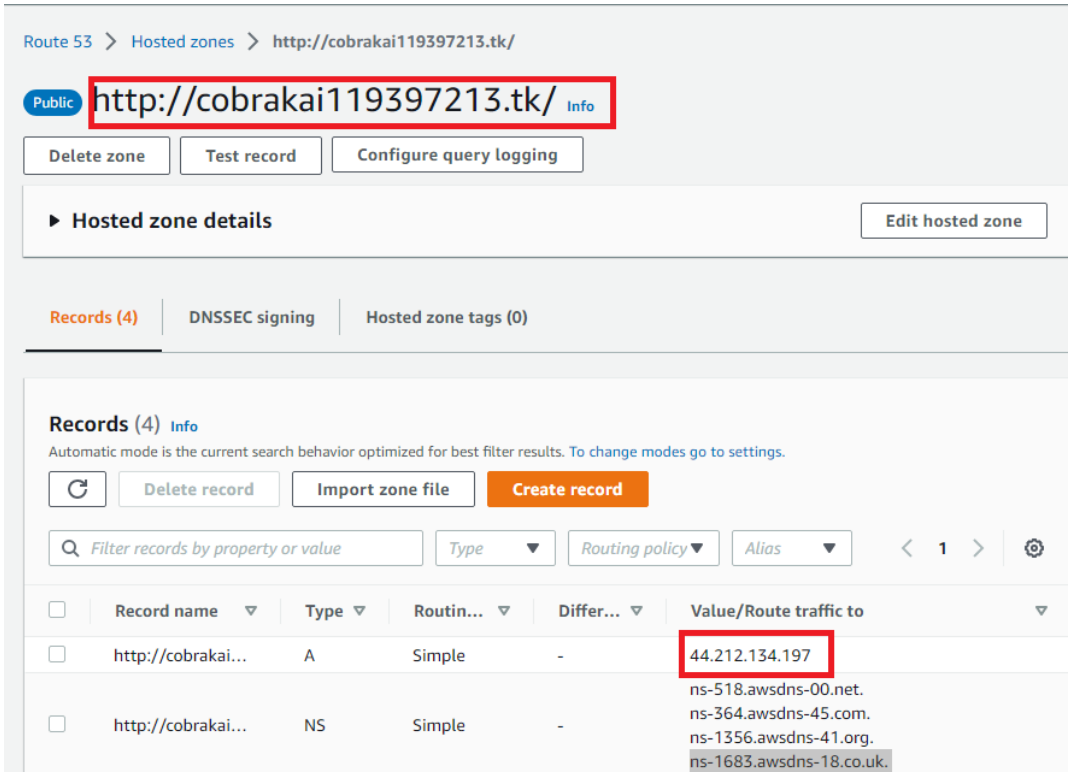
## 4) Route 53:

The Amazon Route 53 is a DNS web service that is scalable and highly available. User requests are routed to internet applications running on AWS or on-premises via Route 53.It can load balance and accordingly change origin address dynamically.

For this first we have to create an elastic ip for our ec2 instance. This makes sure that even after restart of ec2 instance the IP address won't be different.




For this first we register a domain name called ‘http://cobrakai119397213.tk/’. Now we attach this elastic ip to the hosted zone by creating a record so that we get a set of DNS values. We now add these values to the DNS record of the website.

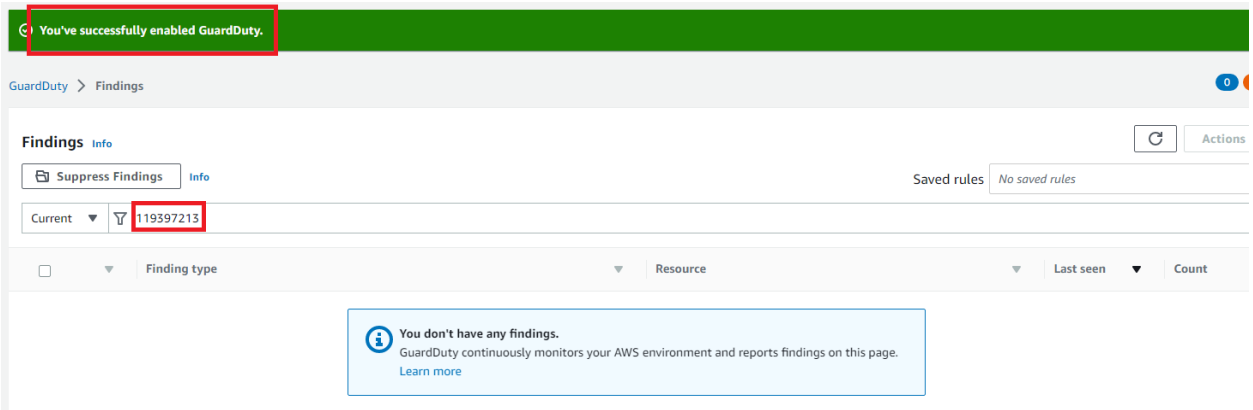



This makes sure that even if one of them fail there are other three that could still provide the service accordingly so that CobraKai team does not lose on customers and the customers don't lose when there is downtime in one line.

## 5)Guard duty:

Amazon GuardDuty processes data sources ( Eg: AWS CloudTrail data events for Amazon S3 logs, CloudTrail management event logs, and various other logs) and monitors them continuously and analyzes them for threats. It does this so by matching with the default premade list of malicious IPs and threat lists and also uses Machine Learning to identify unauthorized, unexpected, and potential malicious activity within a given AWS environment.

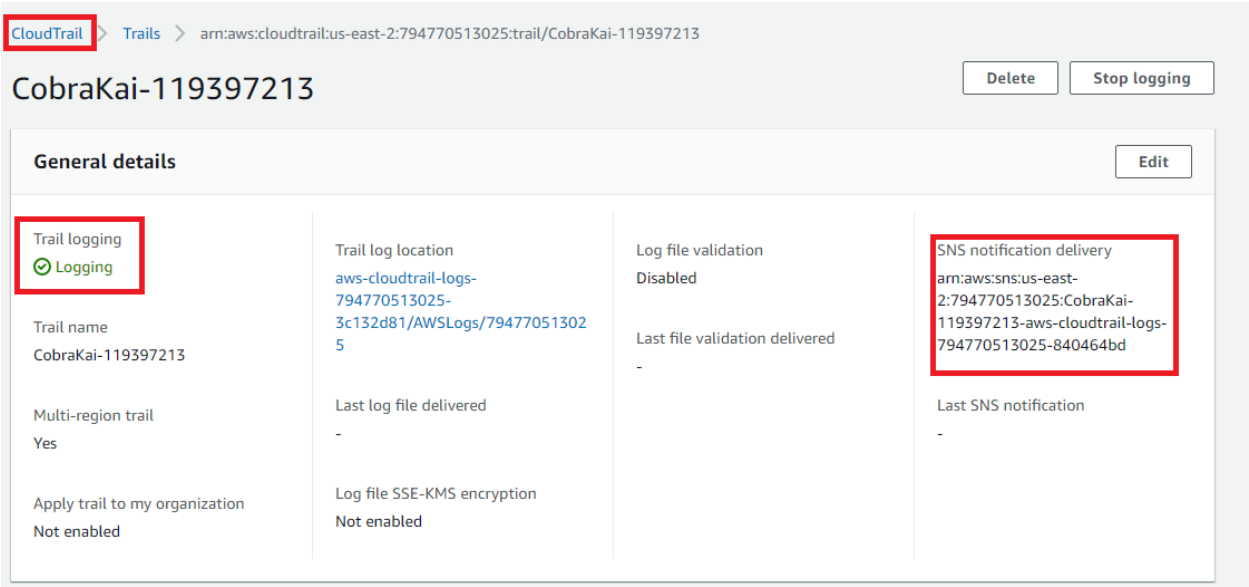
Enabling AWS Guard duty for the account.



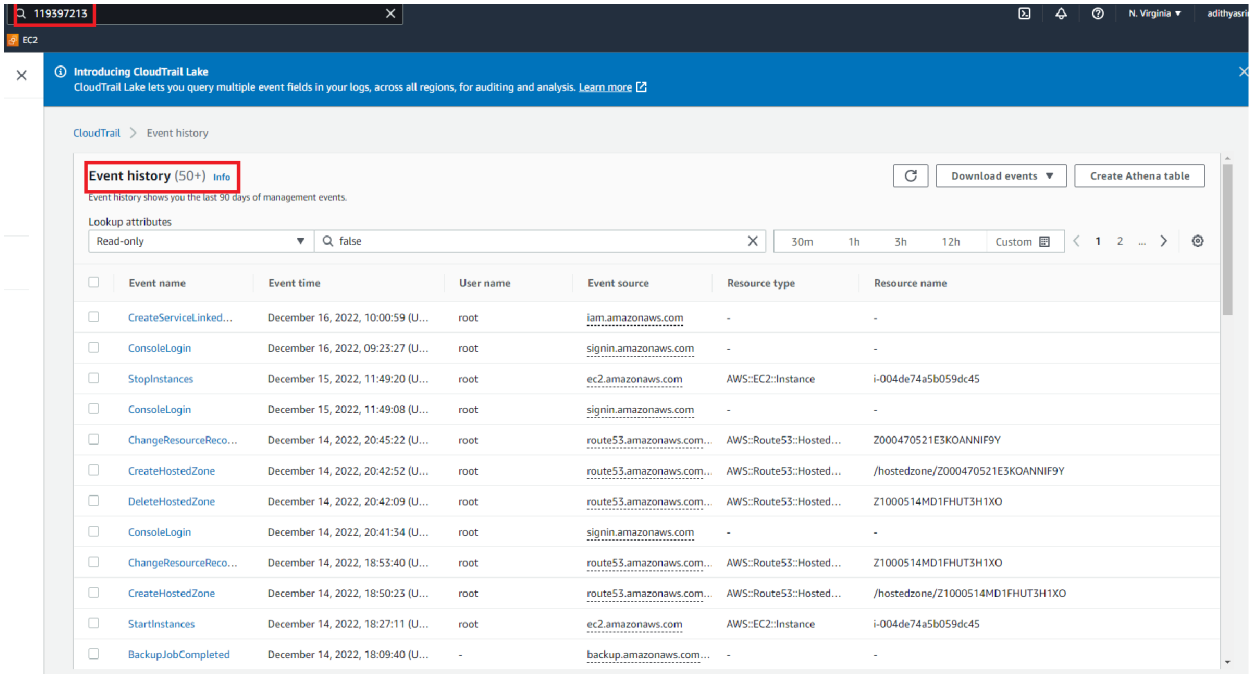

## 6) Cloud Trail:

With the help of AWS CloudTrail, the CobraKai team can govern,enable operational and risk audit, and comply with your AWS account. All the usage is aws services are recorded for this purpose. This can be viewed under the events history tab.

So we created a new trail for this purpose. We add an additional SNS notification system for this so that the team can keep track of various events happening.

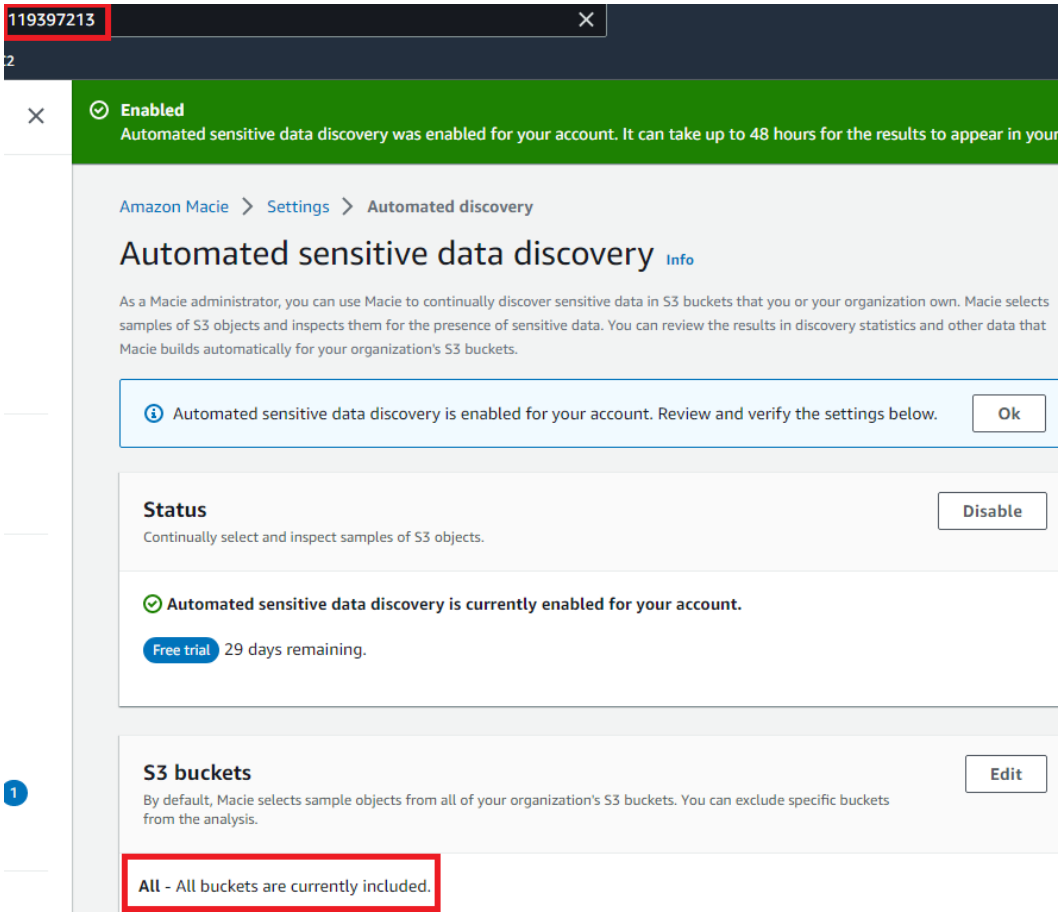



We could see events history:



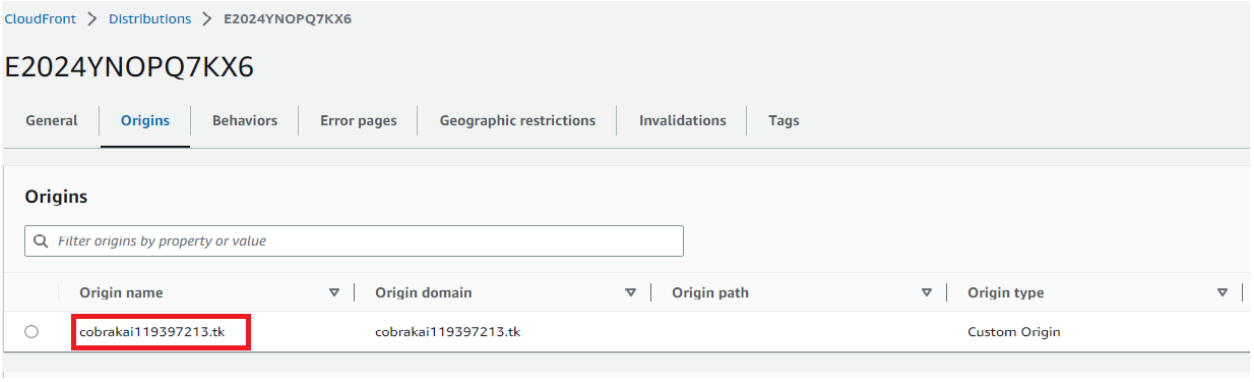

## 7) AWS Maice:

AWS Macie detects and protects sensitive data using machine learning (ML) and pattern matching.The team can use this to identify the sensitive data and make sure to protect them as well as comply with the compliance requirements. It automatically generates S3 buckets with bucket level insights.

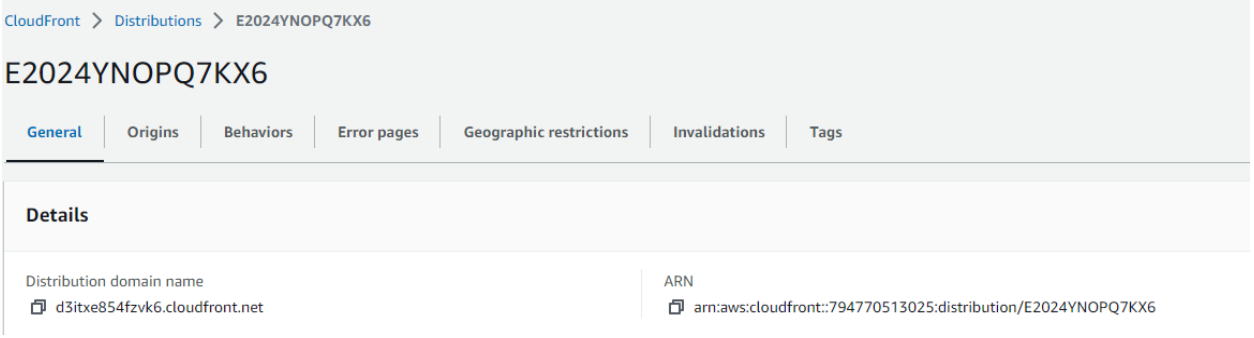



## 8) Cloud front:

This is a content delivery network which helps the end users to have a very good user experience by caching the content and minimizing the latency. If the requested content is present in the edge location, then cloudfront instantly provides it, otherwise it fetches the content from the web server behind the edgelocation and provides it.



We enable cloud front for this account by adding the registered domain name .




## 9) AWS Shield:

AWS Shield standard protects the AWS infrastructure from different types of Distributed Denial of Service attacks. So the CobraiKai team could use this to protect their resources from Daniel LaRusso's attempts to stage a DDoS attack against the platform. It is automatically enabled, if the cobraKai team feels they want further security there is always AWS shield Advanced which is resilient against 99% of DDOS attacks.