Cloud Security Final

Jayraj A. Vakil

UID: 119188361

Objectives

Using cloud and focusing on security aspects, redesign the Cobra Kai application.

Methodology Followed:

➢ Recommendation 1: Hardware Failure

As the whole infrastructure of Cobra Kai is running on a single rack of server, it is more prone to hardware failure as it is considered to be a single-point failure. If there is a hardware failure that results in the server being compromised, the whole business can be brought down. To overcome this issue, we will first need to migrate the whole on-premise infrastructure to Amazon Web Services (AWS).

First step is to create a S3 bucket to import the Virtual Machine (VM) image. For this follow the process below:

1. Go to AWS console and sign-in.
2. Now, go to S3 console and click on **click bucket**.
3. Now give the **bucket name** as you like and **select the region** you want your application to be hosted. Also enable **Bucket Versioning** and checkmark **Block All Public Access**.

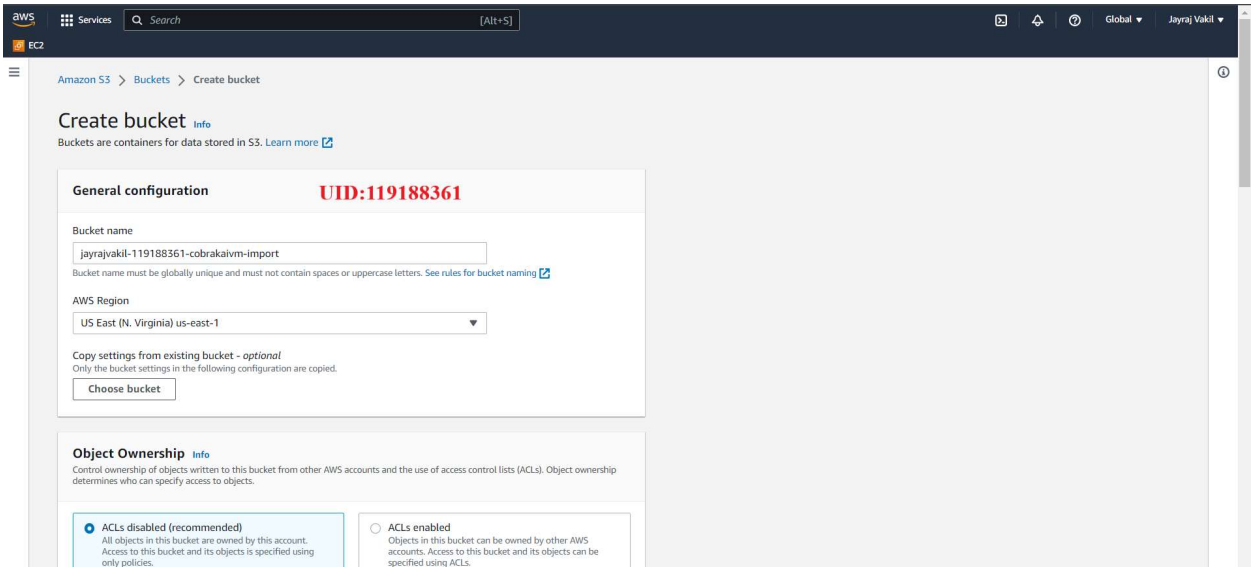

Figure 1. S3 bucket creation

4. After our bucket has been successfully created, add the .ova file which is our VM image to the S3 bucket by clicking on **Upload>Add Files**.

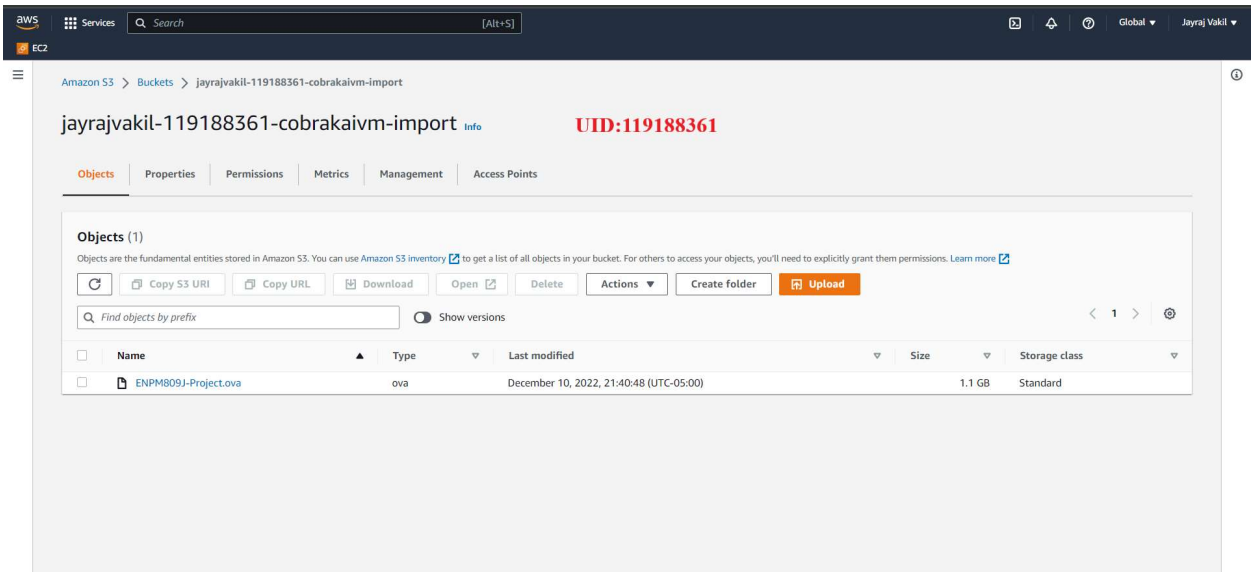

Figure 2. File uploaded to S3 bucket

5. Create an IAM role named **vmimport**. This role is created so that AWS will make operation on our behalf to import the VM file.

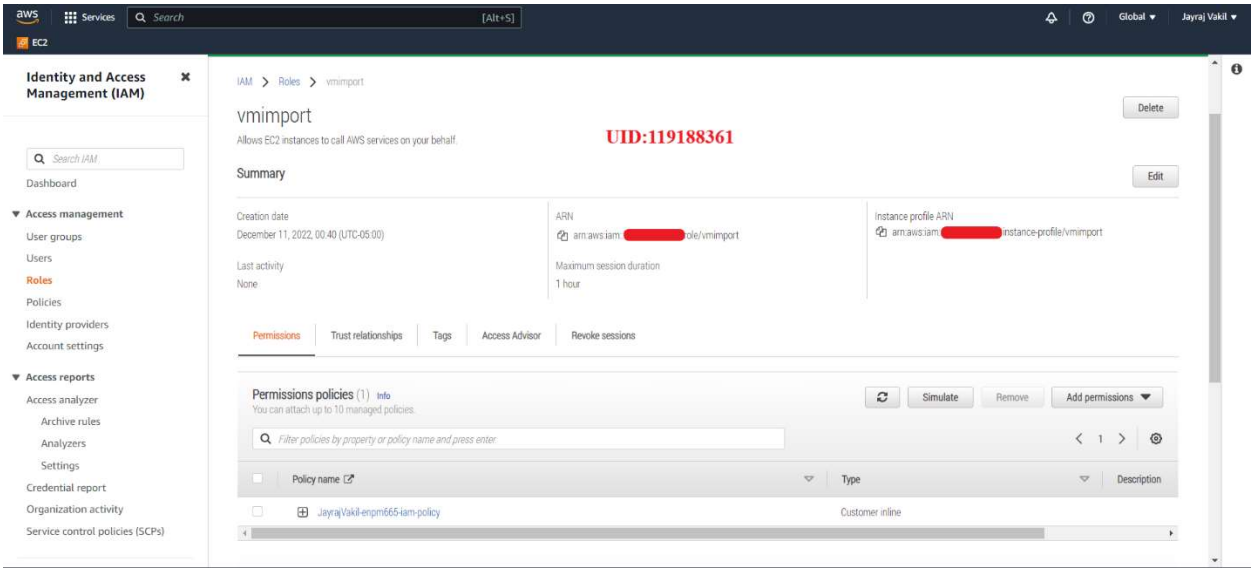

Figure 3. IAM role for VM import

6. After creating this role, go to **Permission tab>Add permission>Create inline policy**. Now paste the following text from the image below to the JSON tab editor. This policy is created so that the role gets sufficient access to the resources which in our case is the S3 bucket and to perform the operations required.

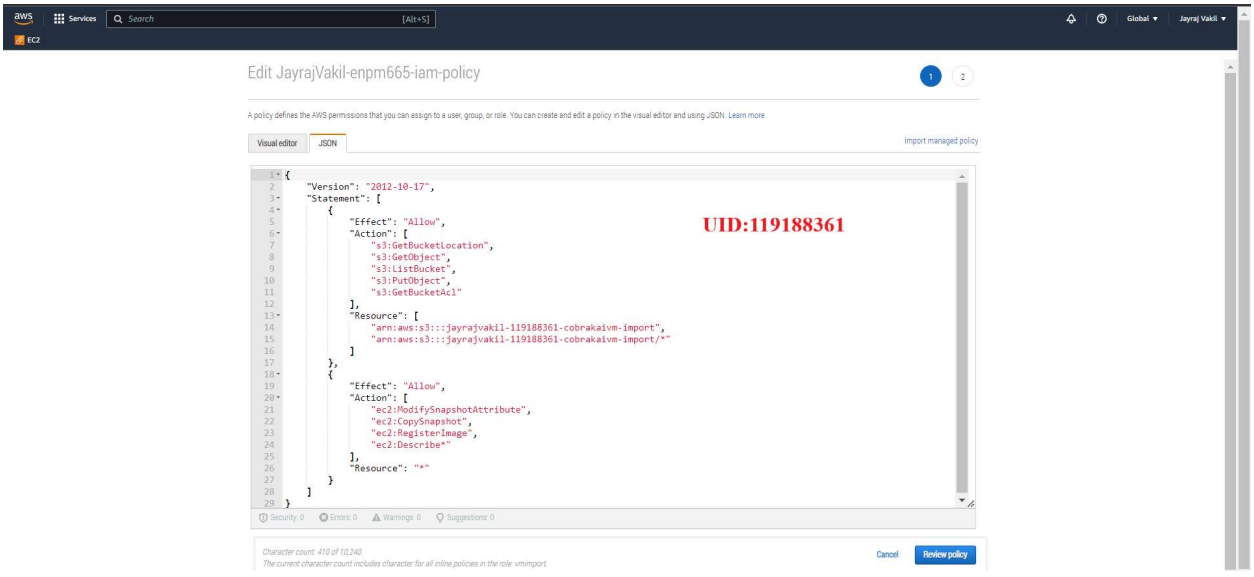

Figure 4. IAM policy for the role

7. Now, we need to create a **Trust relationship** between the role and entities the role can assume. Head over to the **Trust relationships** tab and enter the following as shown in the screenshot below:

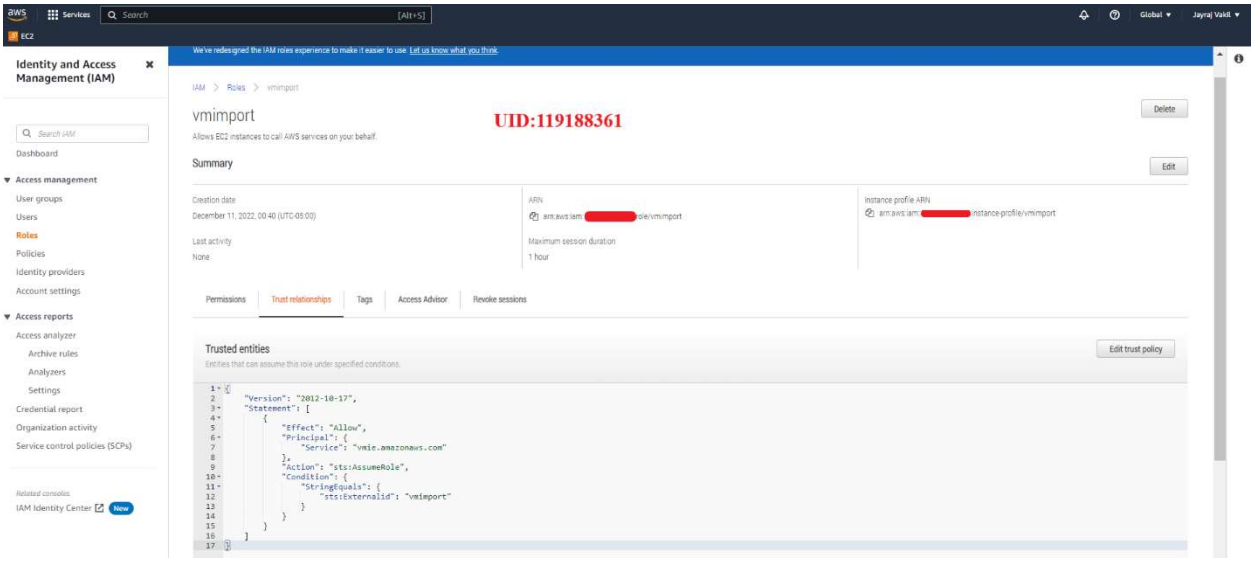

Figure 5. Trusted Entities

Now our setup is ready for S3 and IAM. We can proceed further to host the VM.

8. Now, on the local machine create a JSON file like **importami_JayrajVakil.json** and inside similarly type the following from the image below:
   Description field: Write anything meaningful
   Format: ova (as our on-premise server file is in ova format)
   S3 Bucket: Exact name as the S3 bucket created on AWS
   S3 Key: Exact filename of the ova file

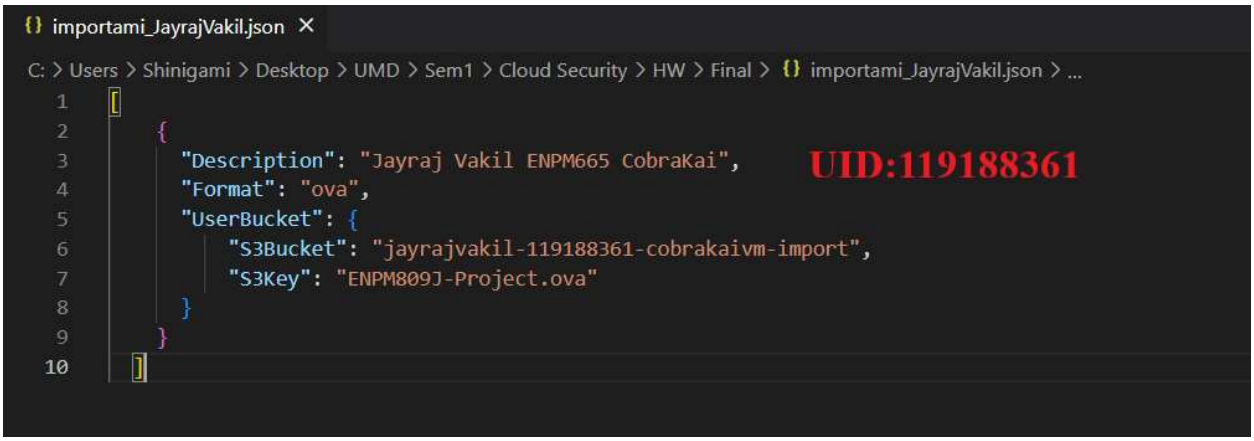

Figure 6. JSON file for AMI creation

9. Open the command import and make sure you have AWS CLI installed on it. Now, run the following commands to make an **Amazon Machine Image (AMI)** from our **.ova VM** file.
   The first command will start the import image process and second command will monitor the creation tasks. The AMI will be created when the status is **completed.**

   I. **aws ec2 import-image --description "Jayraj Vakil ENPM665 CobraKai" --disk-containers "file://importami_JayrajVakil.json"**
   II. **aws ec2 describe-import-image-tasks --import-task-ids import-ami-041f29311e77444f4**

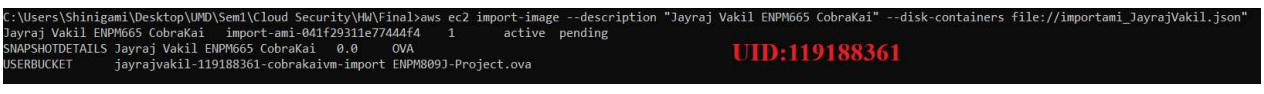

Figure 7. AMI creation (1)

Figure 8. AMI creation (2)

10. Go to **AWS console > EC2 > AMI**, there will be an AMI created. Select the AMI and click on **Launch instance from AMI**. This will launch an Elastic Cloud Compute (EC2) instance.

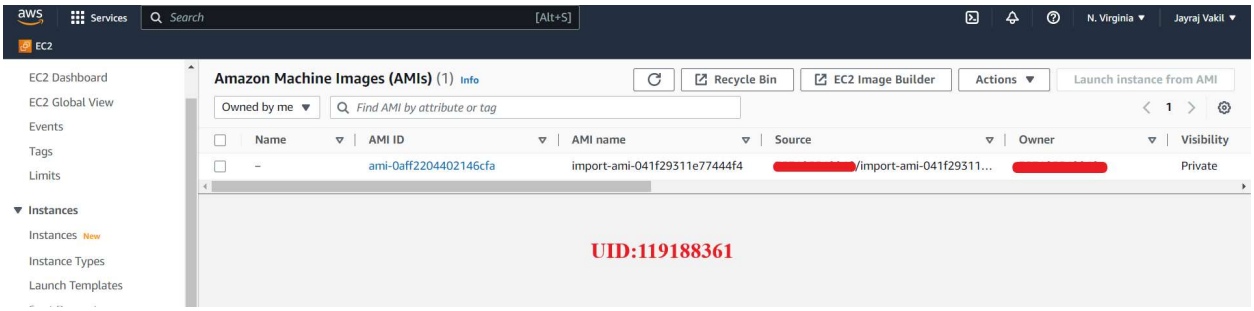

Figure 9. AMI created

11. Now, our server is up and running.

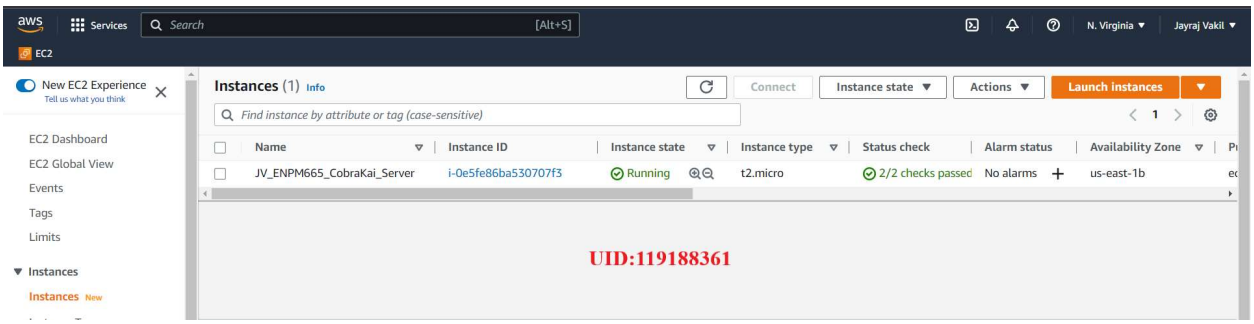

Figure 10. Cobra Kai server hosted

Here, our migration phase is finished.

12. Now, we will implement our load balancer to divide the requests. Head over to **EC2>Load Balancer>Create Application Load Balancer**.
13. Give the name of the load balancer whatever you want, Choose the scheme as **Internet-facing** as they will balance the requests of end-users and the address-type as **IPv4**.

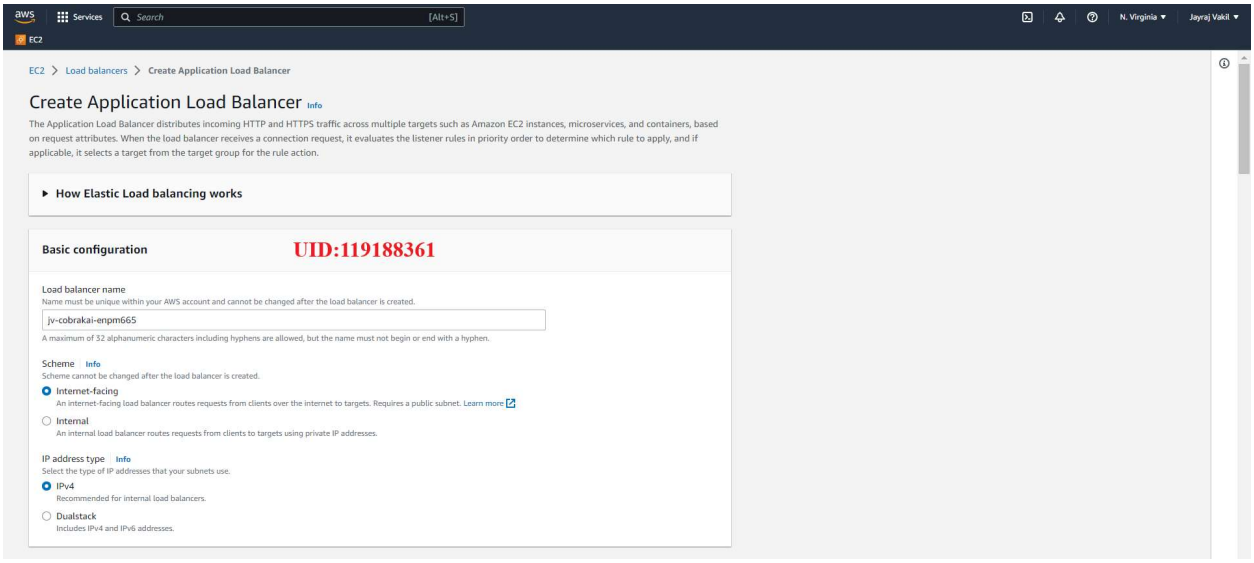

Figure 11. Load Balancer creation

14. For the Network mapping, select the Virtual Private Cloud (VPC) **same** as the **EC2 instance** created above. Now, select **2 Availability zones** and atleast **1 subnet** in each zone. This will be used by the load balancer to route the requests to these zones.

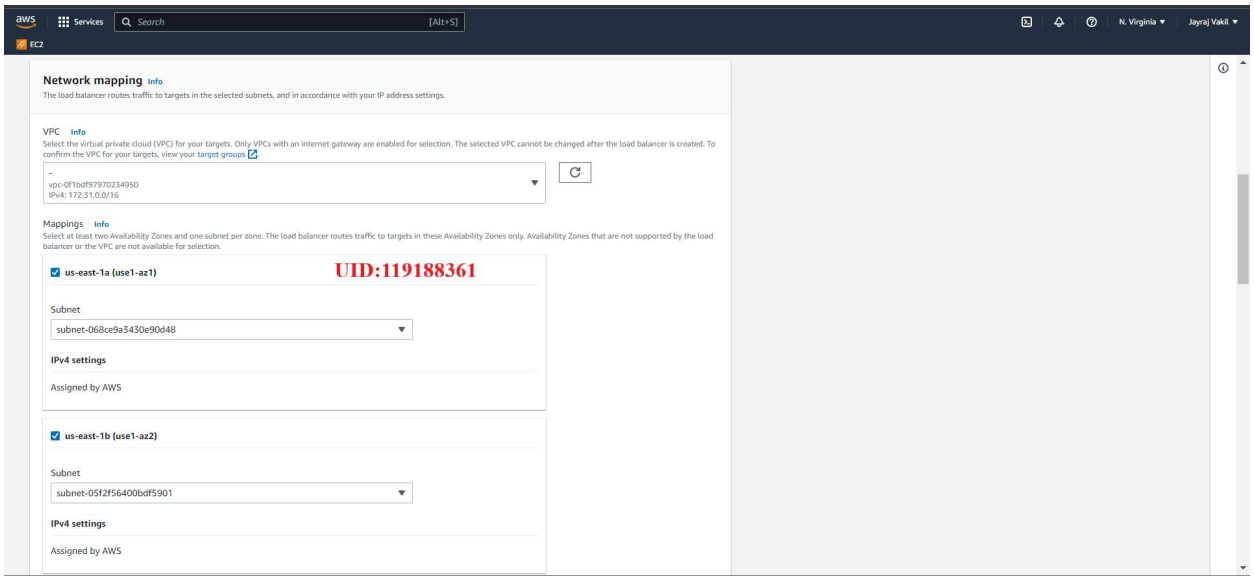

Figure 12. Network mapping of load balancer

15. Under the Security groups tab, click on **Create new security group** option and it will redirect you to **EC2>Security groups>Create new security group**. In this fill out the name as per your wish, select the VPC same as the **EC2 instance**.

16. Add an Inbound rule as **HTTP** type, **Anywhere-IPv4** in the source field and **0.0.0.0/0** IP address as CIDR block. This is done so that anyone can view and access the contents of the website. Also add the security group of the **EC2 instance** created.

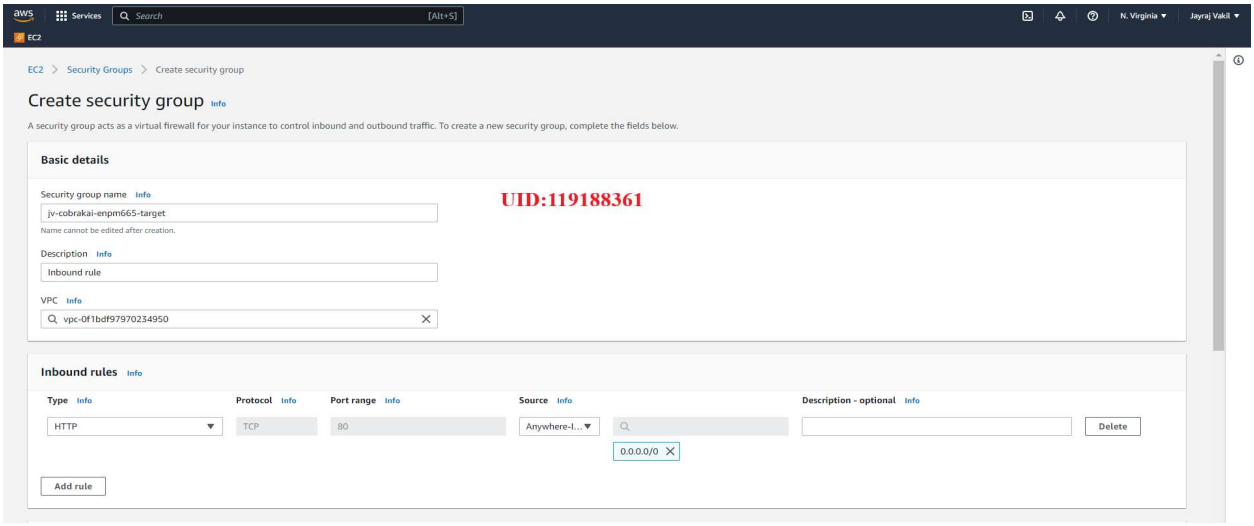

Figure 13. Security group for load balancer

17. For the Listeners and Routing tab, select the protocol as **HTTP** and port as **80** and now click on **Click target group**. This will now take you to the target group creation page.
18. Select **Instances,** enter your **Target group name**, select the protocol as **HTTP**, and port as **80,** VPC **same** as the **EC2 instance**, protocol version as **HTTP1** and then click on **next.**

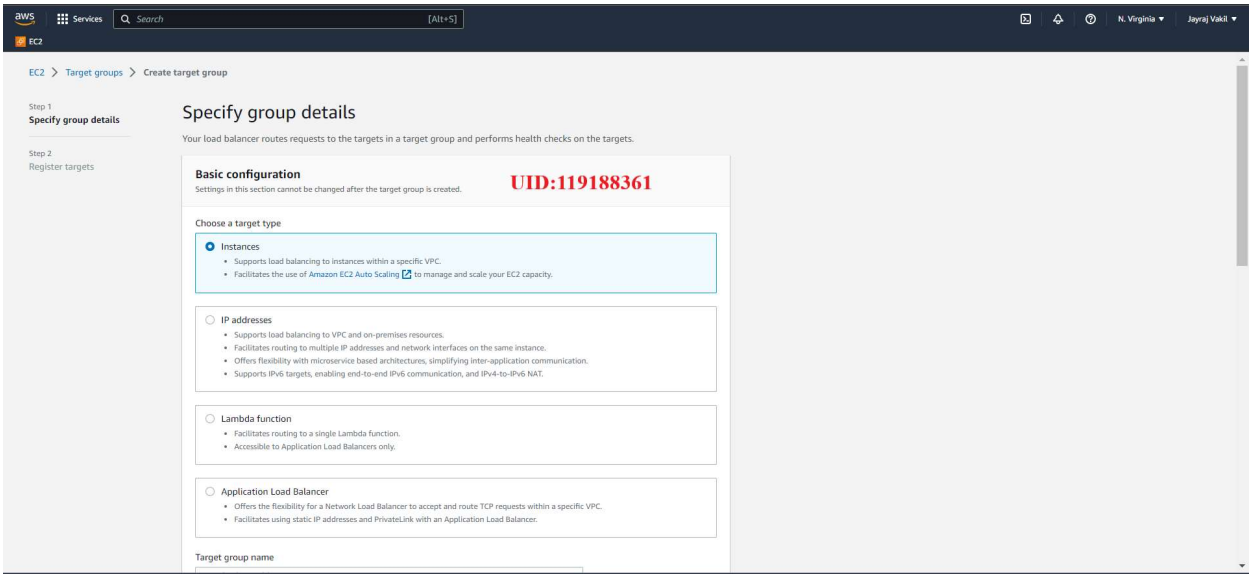

Figure 14. Target group configuration (1)

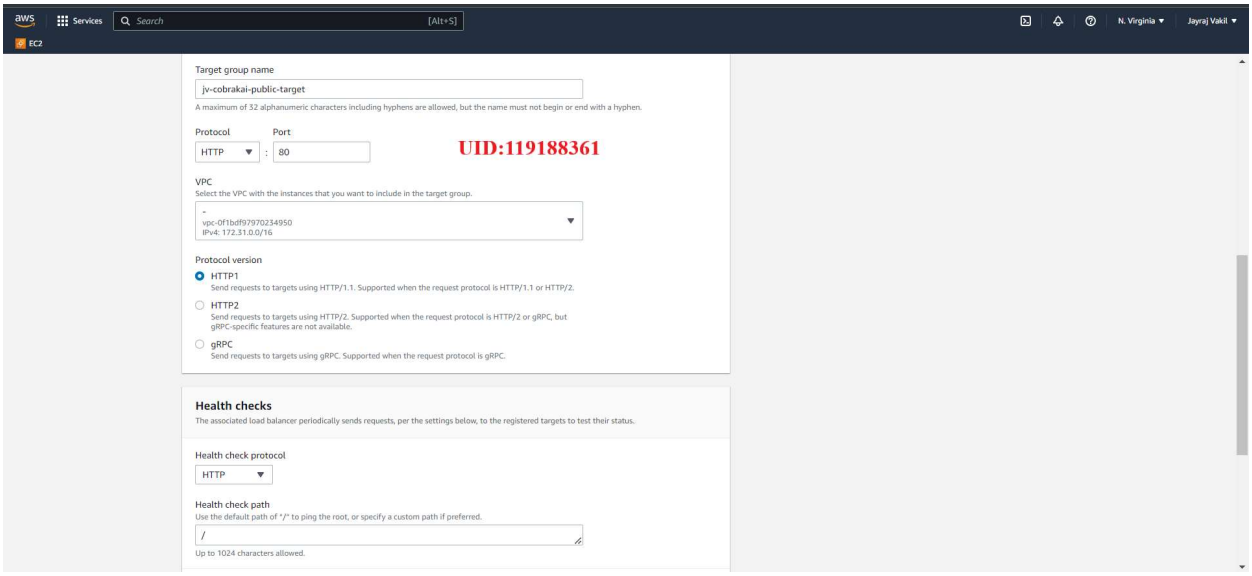

Figure 15. Target group configuration (2)

19. Now, we need to register the target(s) for our target group and that will be our **EC2 instance**.

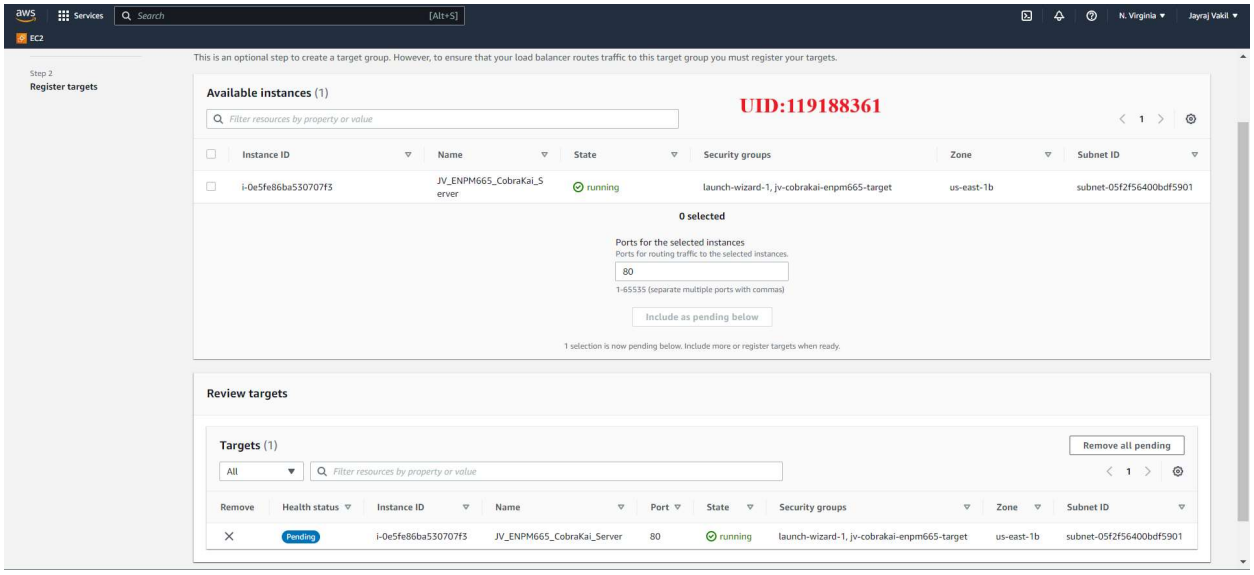

Figure 16. Register targets for the group

20. After this, click on **Create target group** and our target group is created and ready to be attached to the load balancer. After this check the summary and if finalized, click on **Create load balancer**.

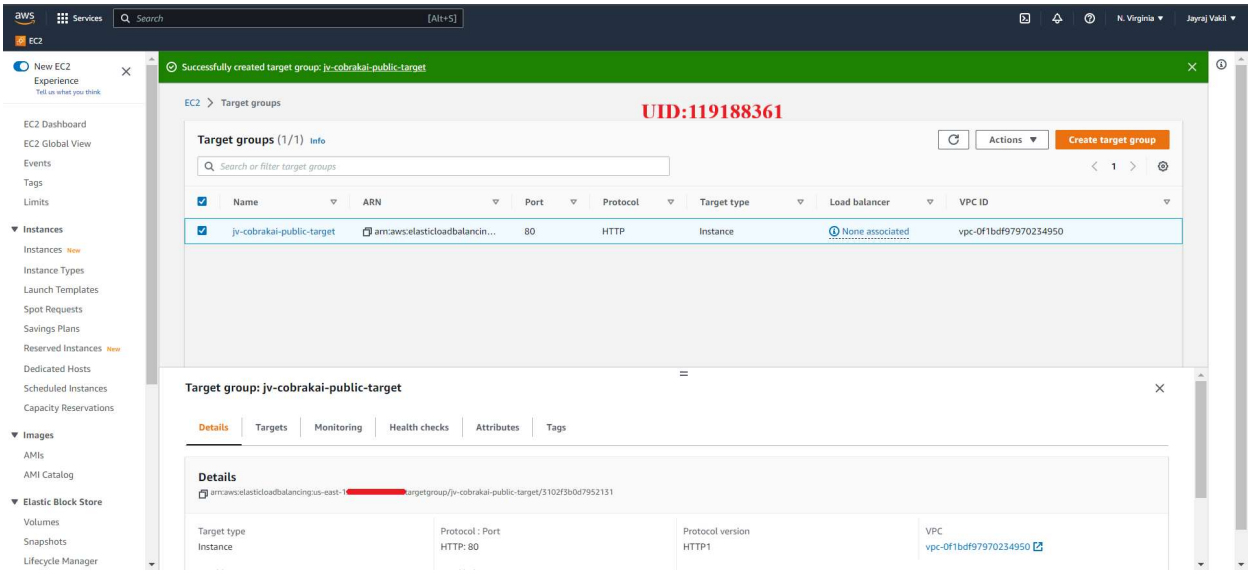

Figure 17. Target group created

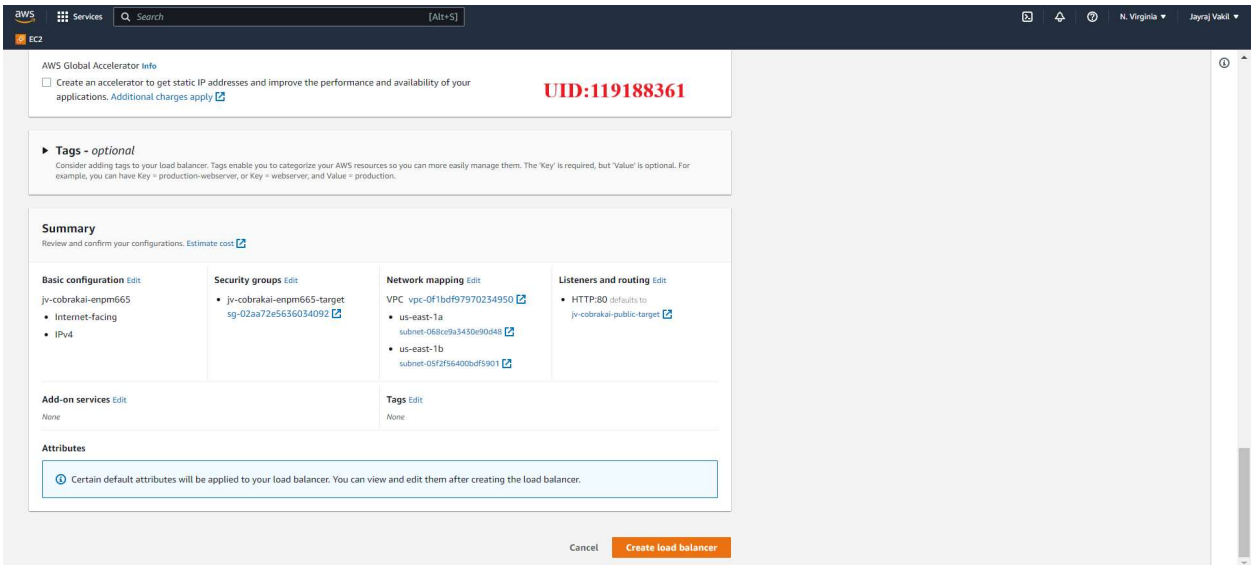

Figure 18. Summary of load balancer creation

21. Our load balancer has been created and is active.

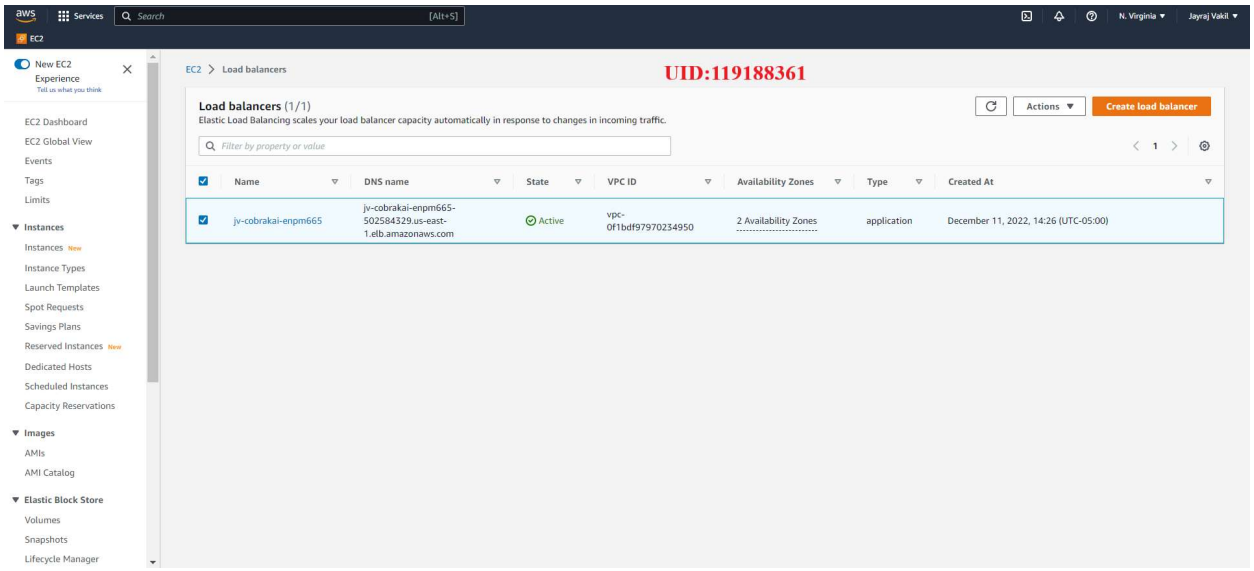

Figure 19. Load balancer active

22. Now, we can access the site using the Application Load Balancer (ALB) DNS name.

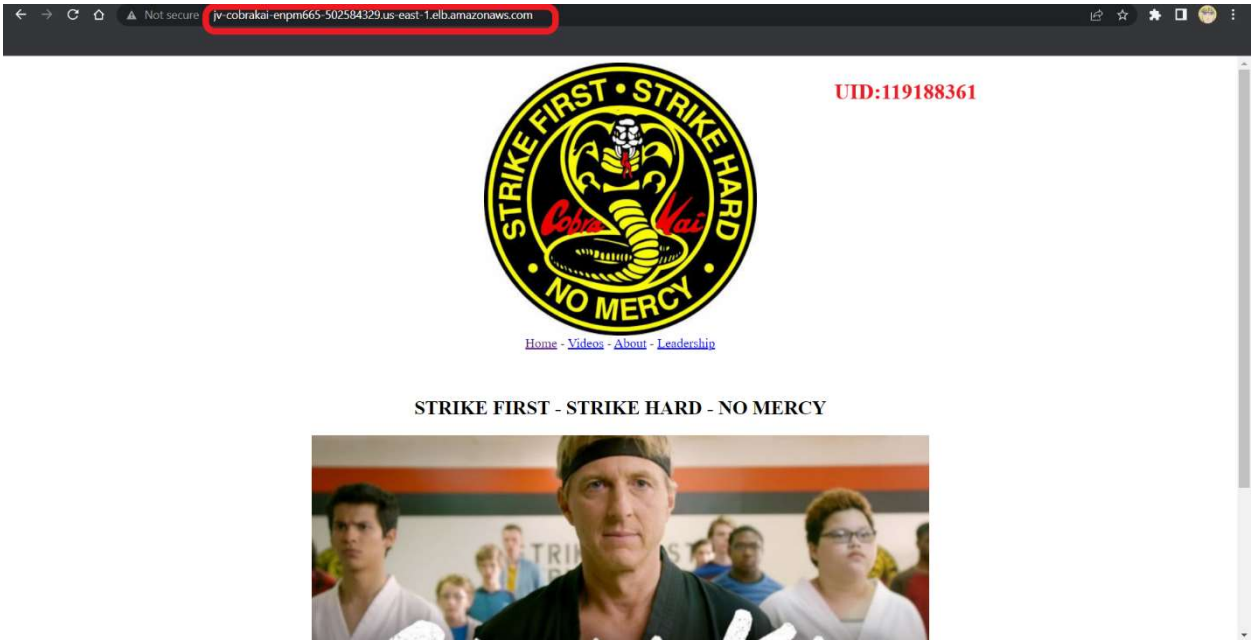

Figure 20. Website access using DNS of ALB

23. After our load balancer is active, we need to create auto-scaler so that when one of our server's health check status is **unhealthy**, the application will be scaled automatically. For this to be implemented, go to **EC2>Auto Scaling>Launch Configuration**. Now, give the name you want, select the **same AMI** we used for creating our **EC2 instance**, and select the **t2.micro** instance type.

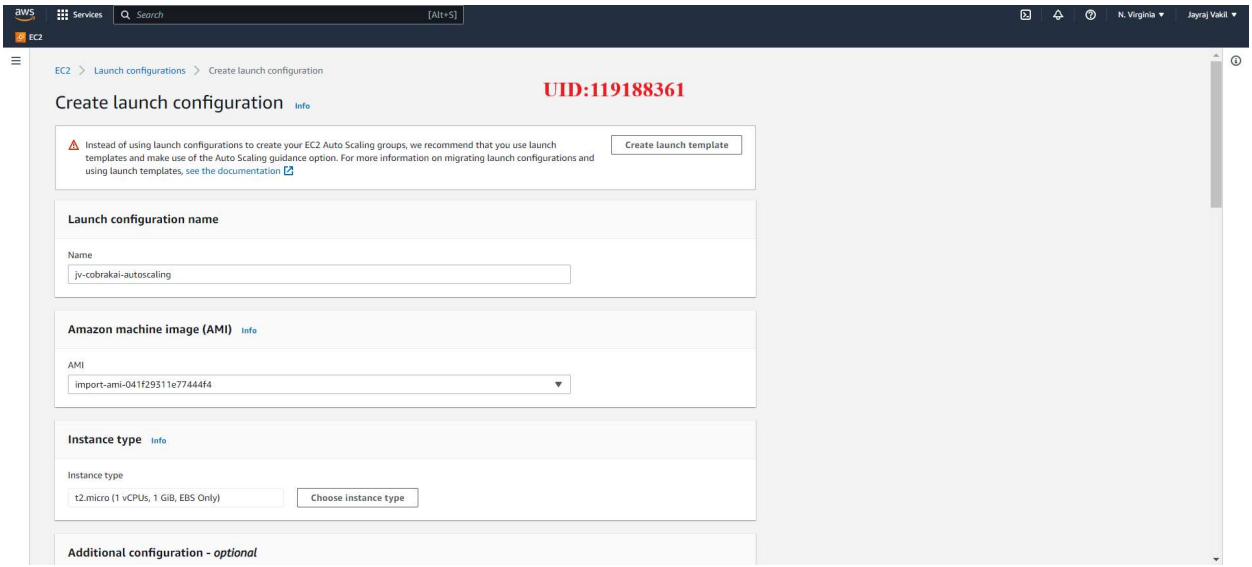

Figure 21. Launch configuration for Auto Scaling

24. In the security group tab, select the **2 security groups** which were used for the **EC2 instance**.

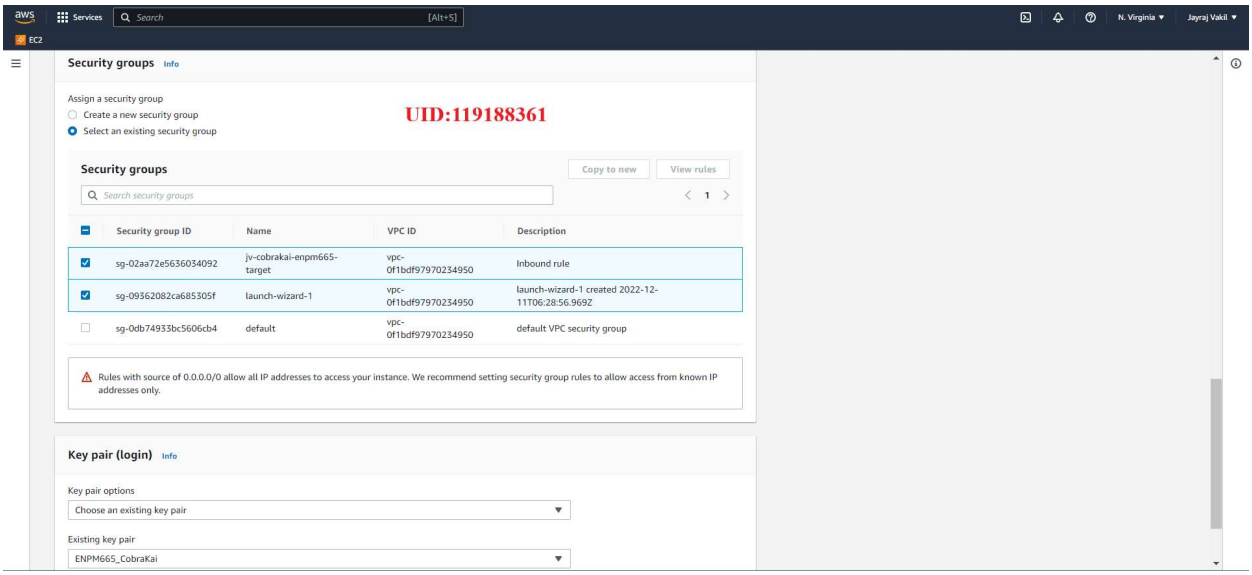

Figure 22. Security group for Auto Scaling

25. Now our launch configuration is created and we are ready to create the auto scaling group.

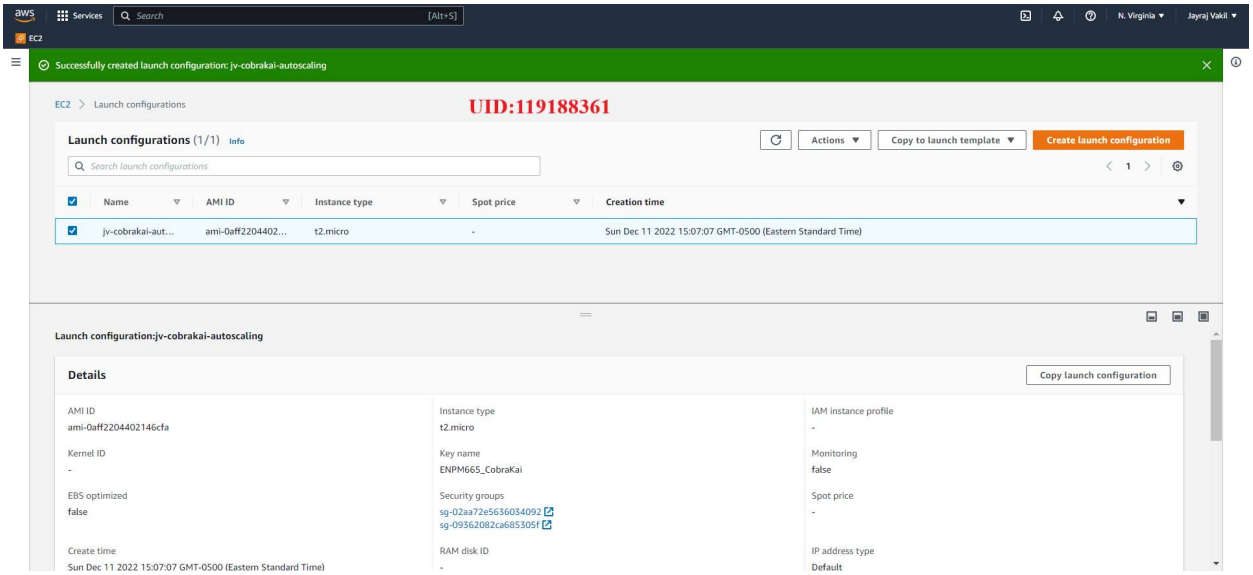

Figure 23. Launch configuration created successfully

26. Now navigate to **EC2>Auto Scaling>Auto Scaling Groups** and select the **Create an Auto Scaling group** button. Now give the group name as you wish, choose the **Launch configuration** template we created above and click on **Next**.

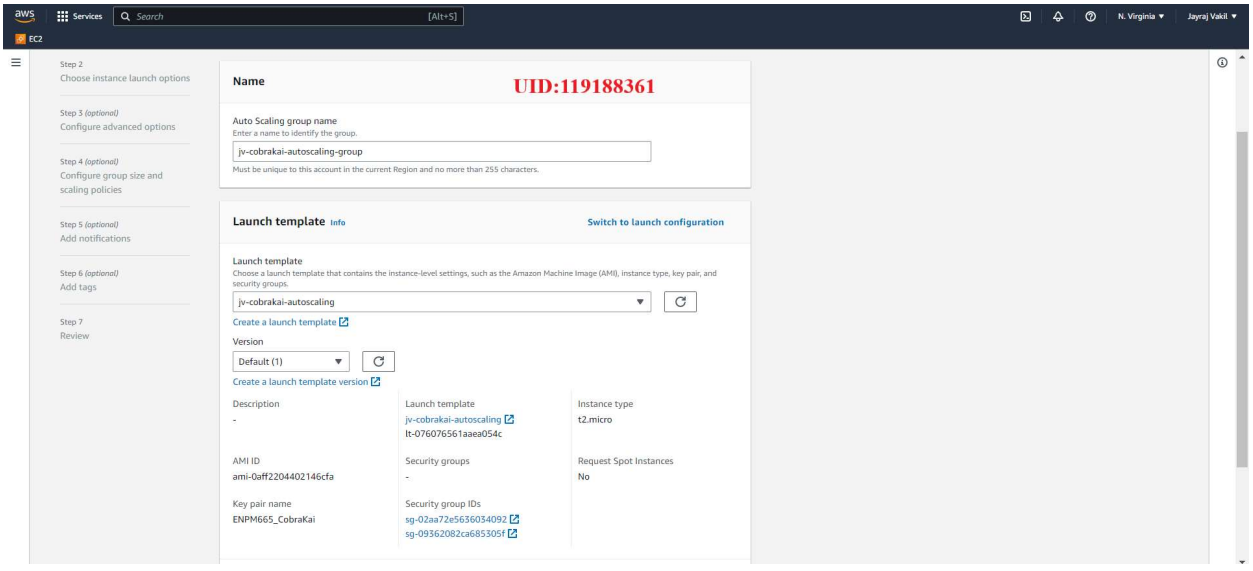

Figure 24. Auto Scaling configuration

27. Choose the VPC and subnets **same as EC2 instance** used for load balancer and click on **Next**.

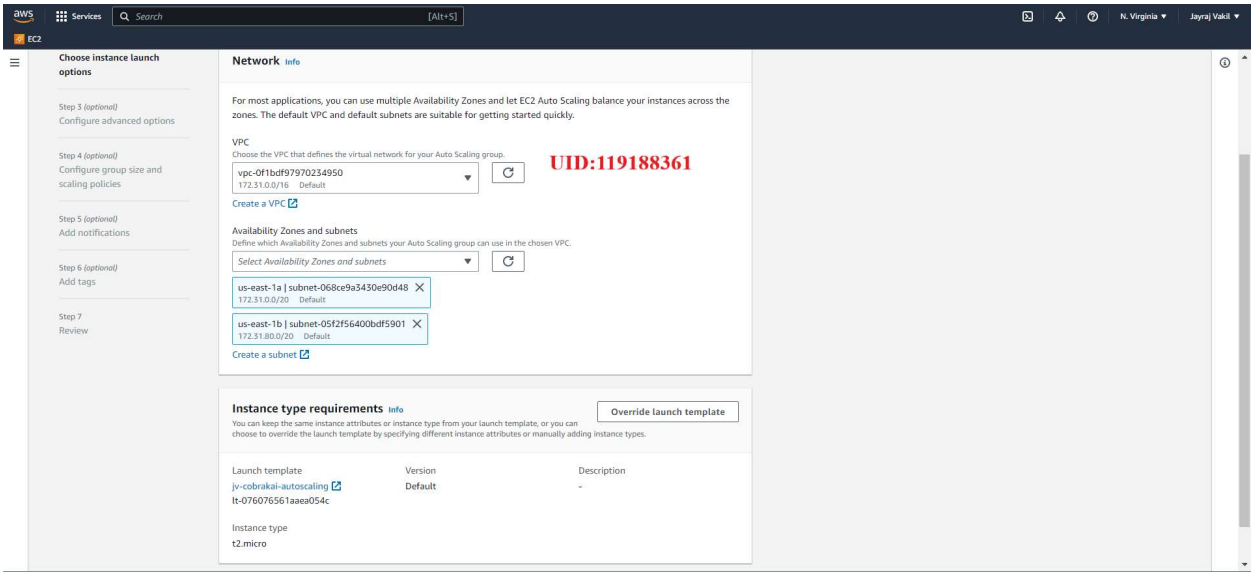

Figure 25. Auto Scaling network configuration

28. Now, in the advance configuration, we will attach our **existing load balancer** to the **auto scaling group** and the **target group we created above**.

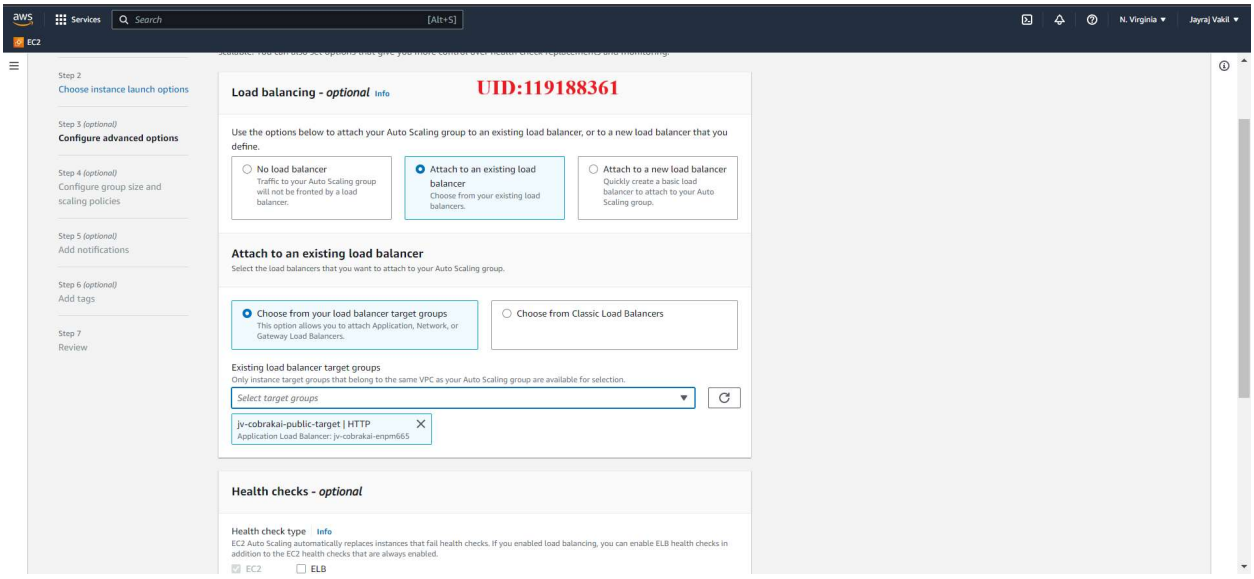

Figure 26. Attaching load balance

29. After we have created our Auto Scaler, it will automatically create an instance which will monitor everything and will scale it up or down depending on the need.

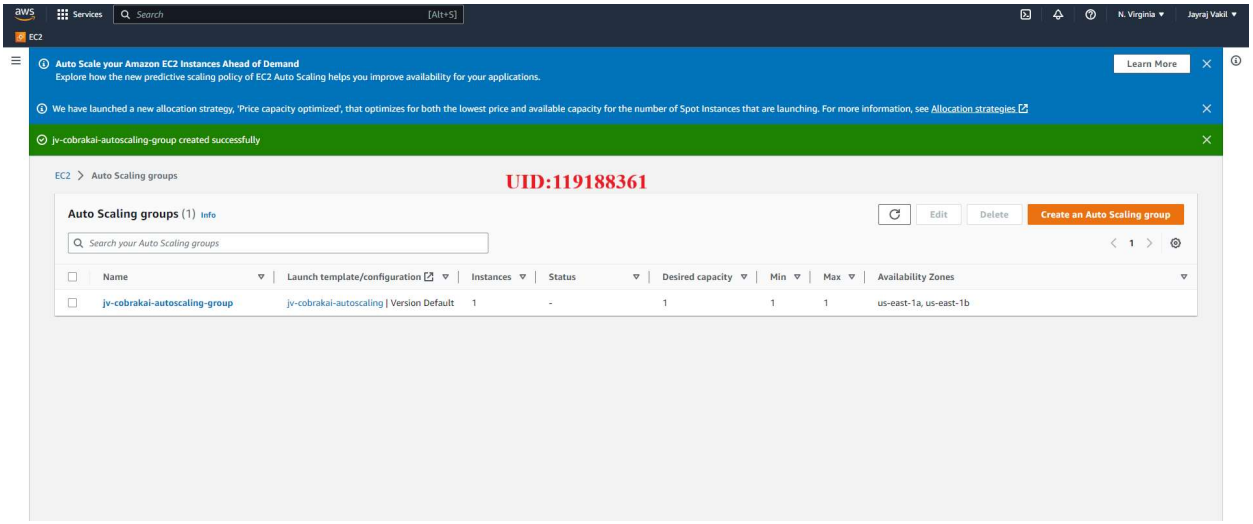

Figure 27. Successfully created Auto Scaler

Why this approach?

Using this approach, the VM will be migrated to the AWS in a secure way and only the person authorized would be able to perform migration. The process we are using is called “lift and shift” method, which is one of the most comfortable way through which the already setup on-premise server can be migrated without much changes.

After migrating we will have enough protection for preventing hardware failure and now to further strengthen it we will use load balancer and auto-scaler through which the loads will be divided among various availability zones. The function of load balancer is to send the requests to the server where the workload is low and redirect the request from one server to another if there is an overload on the former server. Auto scaling group will scale the resources up and down depending on the need of the server so that the architecture is cost-effective. This solves the problem of someone stealing the hard drives from the server, balancing the load, and hardware failures.

➢ Recommendation 2: Creating IAM users and groups

As there is no account permission strategy, the web server is highly vulnerable to attacks and can disrupt the whole business. For this, we need to implement accounts with the principle of least privilege. We will need to create Identity and Access Management (IAM) groups and users.

To implement IAM, we will follow the below steps:

1. Navigate to **IAM>User groups** and then click on **Create user group.**
2. Give the user group name as the roles they are supposed to perform. For e.g. **SystemAdmin-119188361** for system administrator as shown in the image below.

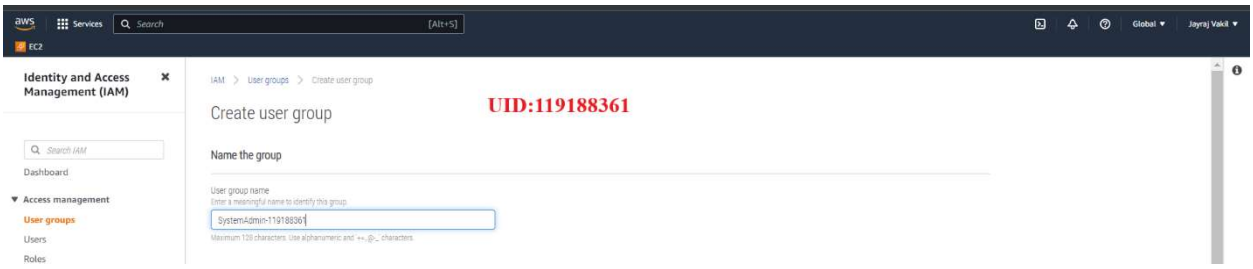

Figure 28. Create user group

3. Then attach the corresponding policies to the group and click on **Create group.**

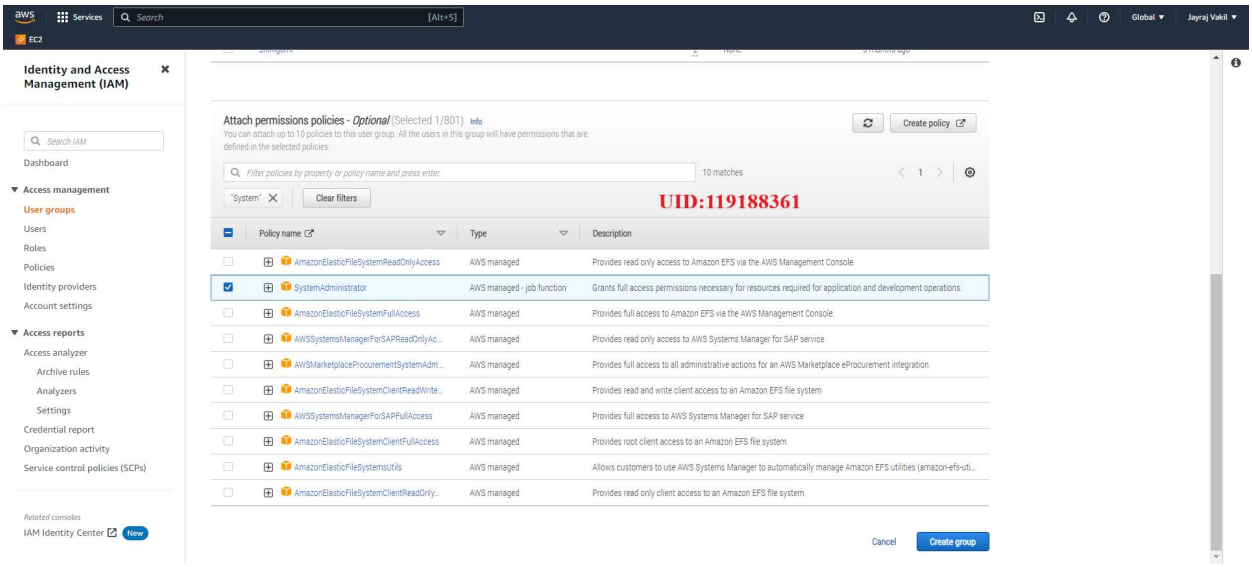

Figure 29. Attached policy

4. Now, we need to add the corresponding employees to the group created. For e.g. Bert will be assigned to the system administrator group.

   Navigate to **IAM>Users** and click **Add Users** and for the credential type choose **Access key – programmatic access.**

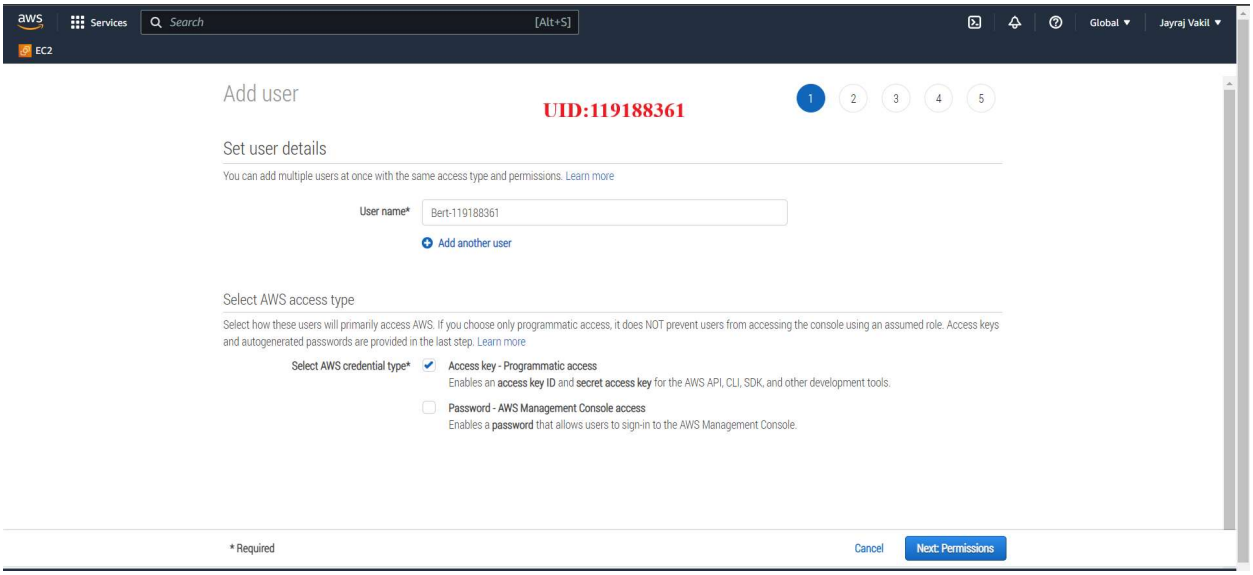

Figure 30. Add user

5. Add the user to the group and then click on **Add user** and similarly we will add myself and Aisha to the group.

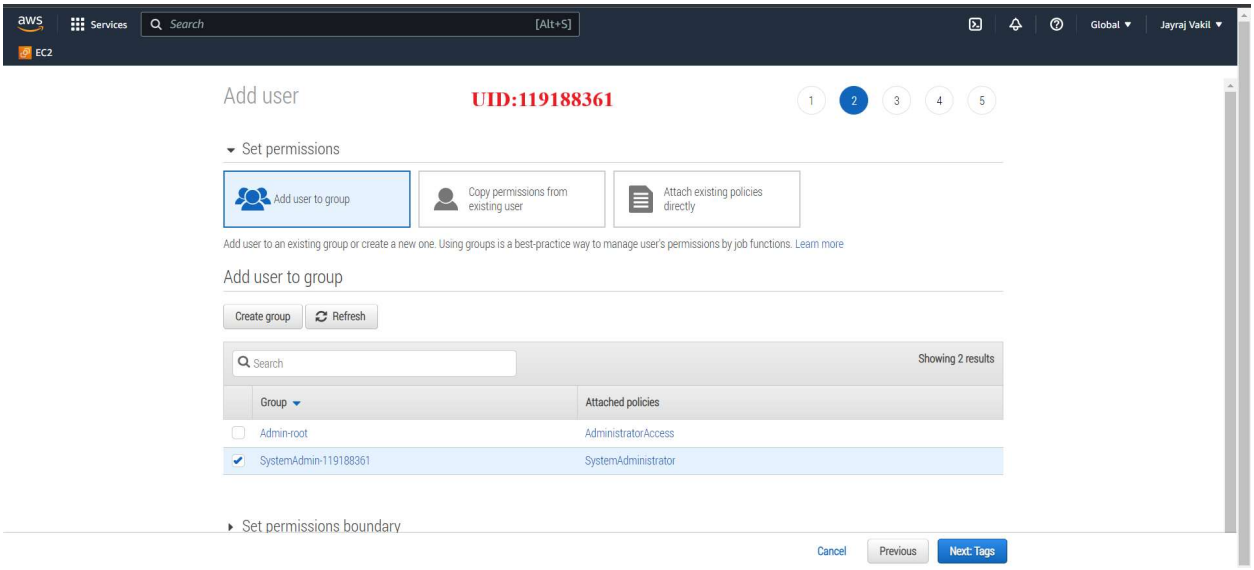

Figure 31. Adding user to the group

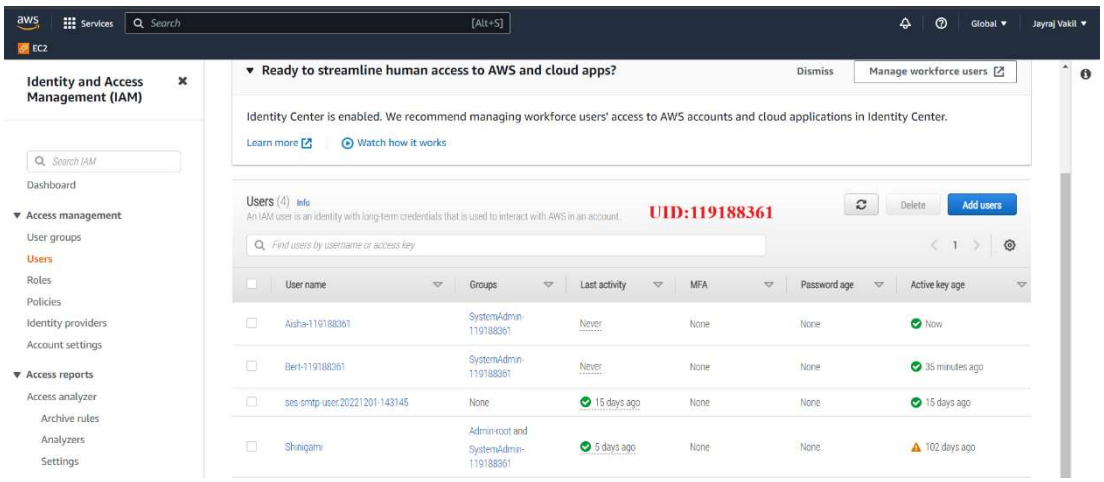

Figure 32. User added to the group

6. Similarly, we will create the following groups:

Group	Policies	Users (employees)
Development	CodeBuildAccess	Eli, Demetri
Production	DeploymentToolkit	Demetri
Audit	SecurityAudit	Miguel
Security	WAFFullAccess	Myself, Aisha
Root	Root	Aisha, Johnny

Why this tool?

IAM will be really useful in implementing principle of least privilege. Through this tool, employees will only have access to their corresponding resources and not everyone will be able to run privileged commands which will help Cobra Kai to stop giving full authorized access to everyone. This will improve security to a greater extent than what was previously implemented.

➢ Recommendation 3: Backing up the servers

Currently, Cobra Kai doesn't have a single backing-up strategy for its infrastructure. So, if there is some malfunction in the server and it goes haywire, the whole business can be gone in seconds. For this problem, AWS provides its own backup service known as AWS Backup.

For this backup service, we will create a backup plan by following these steps:

1. Go to **AWS console>AWS Backup>Create Backup Plan**.
2. For ease of use, we will use a template provided by AWS. Enter the name you want for the backup. We are using the template **Daily-Weekly-Monthly-5yr-Retention**. Now for automatic backup of servers we need to add a tag. Our custom tag name is **JV-Backup** and value for it is **True**.

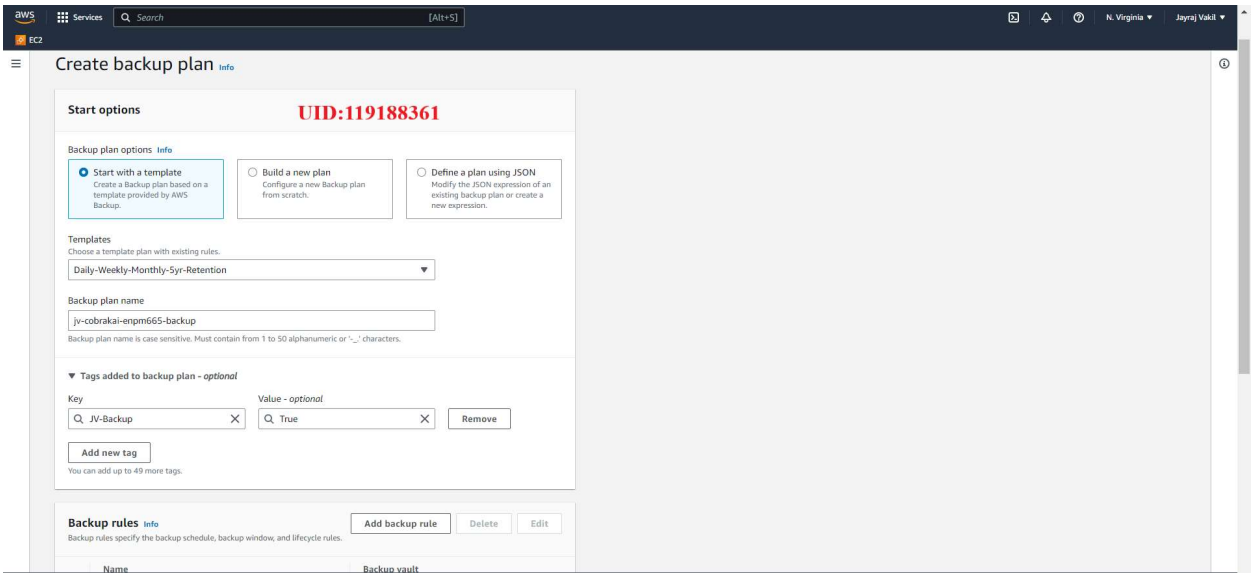

Figure 33. Backup plan creation

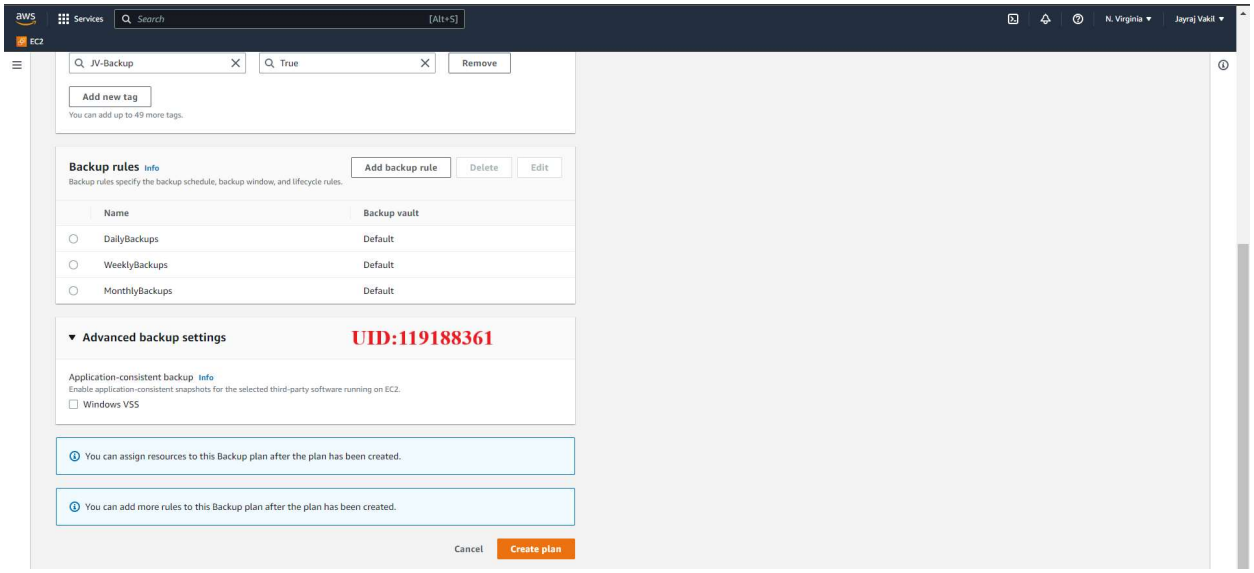

Figure 34. Backup rules

3. For the backup rules, we will edit the backup rules according to our requirements.

	Cold storage transition	Retention period
DailyBackups	14 days	45 days
WeeklyBackups	4 weeks	26 weeks
MonthlyBackups	1 month	5 years

The WeeklyBackups will take place on **every Saturday** of the week during the default time window.

The MonthlyBackups will take place on **1st date of every month** during the default time window.

Cold storages are storages where the data which is infrequently used or the data which is archived will be stored. Retention period is the amount of time the data will be kept before deleting.

4. Finally assign the resources to this backup by refining the results with our **key-value** pair tags.

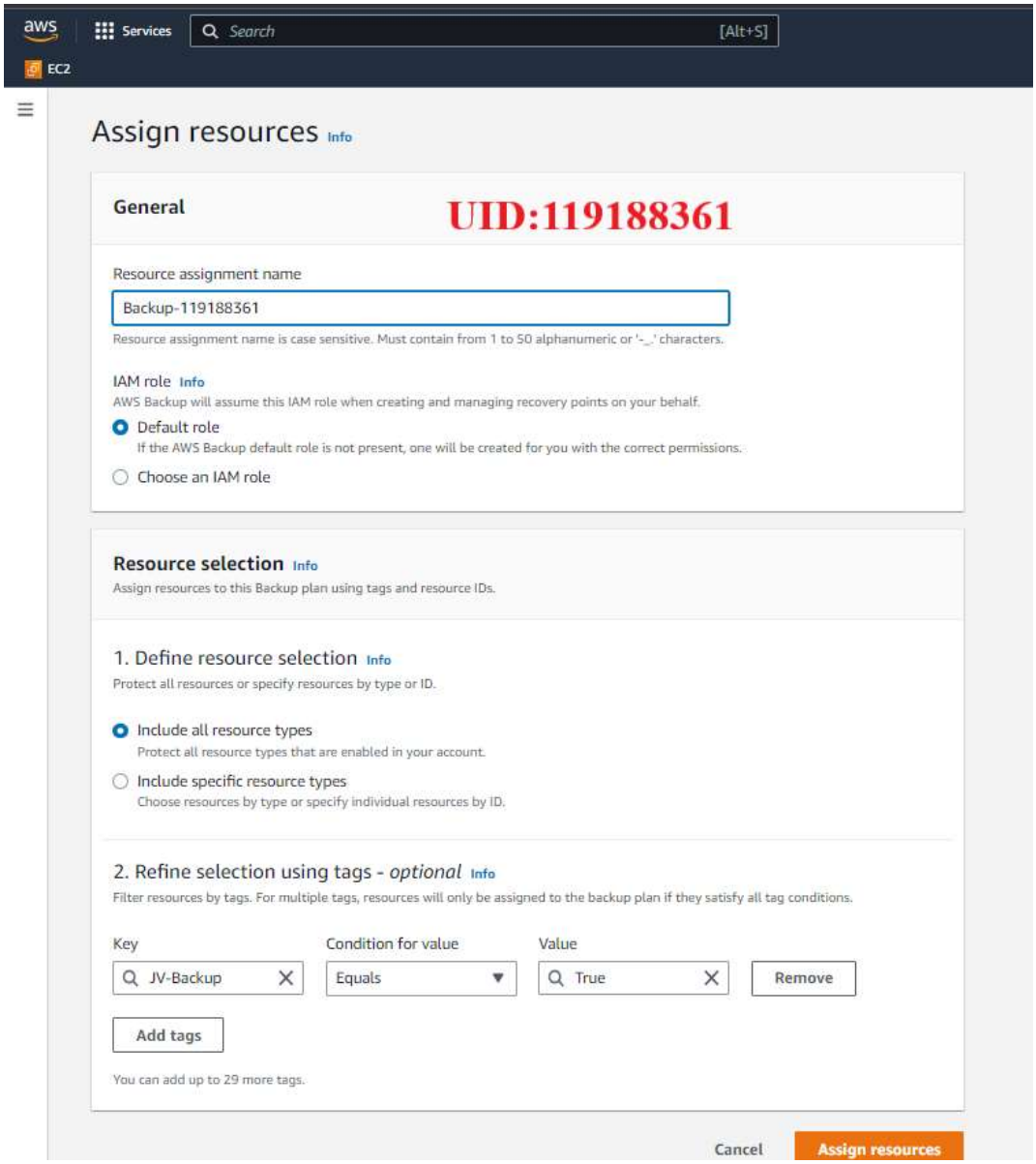

Figure 35. Assigning resources

Why this tool?

AWS Backup is a centralized backup service and it's a completely managed service. It will also protect data automatically. It can also automate backups. It provides a console to monitor backup policies as well as activity. It also abides by compliance requirements. Therefore, it can be used as an end-to-end solution.

➢ Recommendation 4: Patching up the servers

Right now, Cobra Kai doesn't have a single patching strategy on their server and they might be running their website on some outdated software/operating system. This might be one of the reasons that their website is highly vulnerable to attacks. To overcome this issue, AWS has Patch Manager which is a part of AWS Systems Manager.

Follow the steps below to implement the patching for the AWS:

1. Go to **AWS Console>AWS System Manager>Patch Manager**. Now, click on **Configure Patching**.
2. We will select the option **Enter Instance tags** to patch the instances and enter our tag as **Key = jv-119188361-patch** and **Value = True**.
3. We will patch our instances at a scheduled maintenance time using **CRON Schedule Builder** and our scheduled day and time will be on **Sunday** at **12:00 PM** and select the patch operation type to **Scan and Install**. Now hit on **Configure Patching button**.

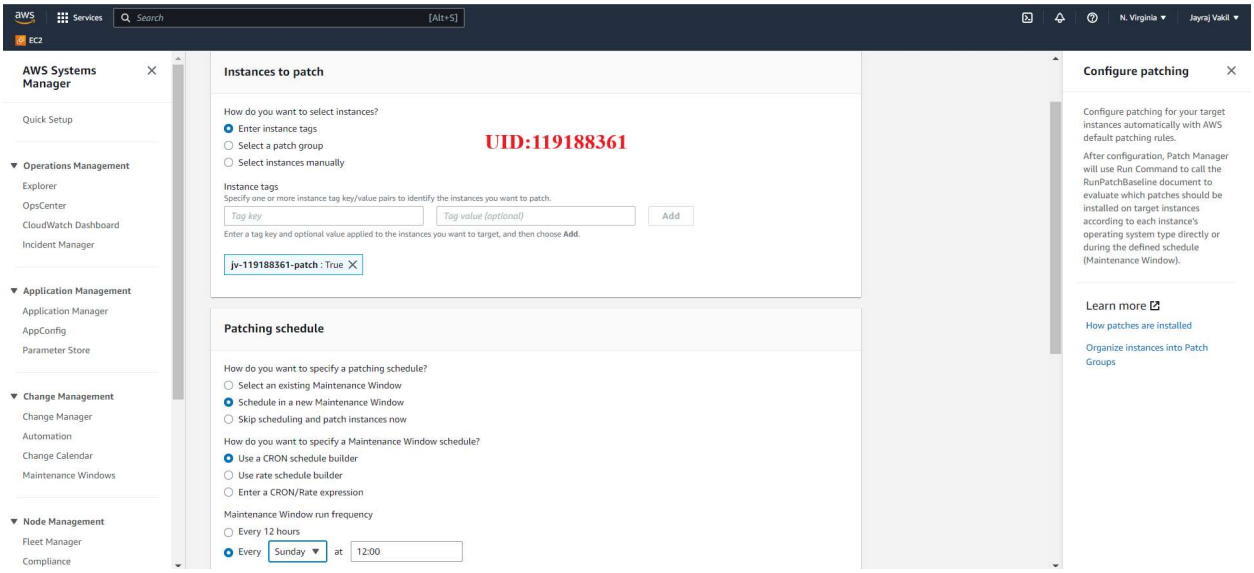

Figure 36. Configure Patching of Instances

4. Now, after getting redirected to the Patch Manager dashboard it is time to create a baseline for patching. Go to **Patch Manager>Create Patch Baseline**.
5. Give the name you want, select the operating system as **Amazon Linux**.
6. For the rules, select **All** for products, severity and classification. Also set the **Compliance reporting** option to **High**. After this, hit on **Create patch baseline**.

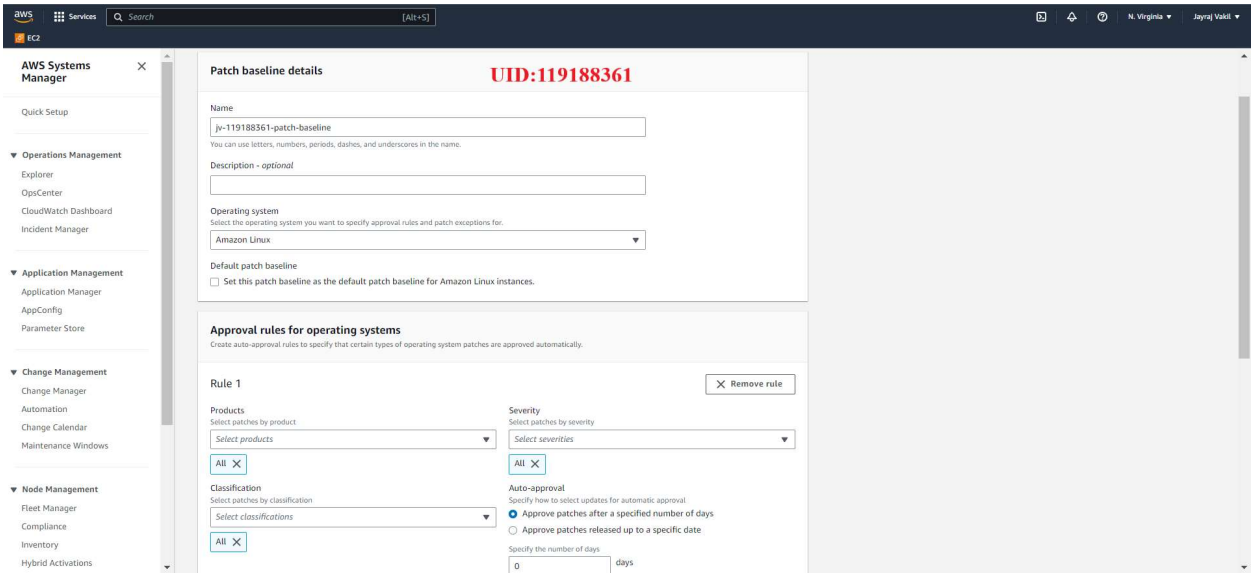

Figure 37. Patch baseline configured

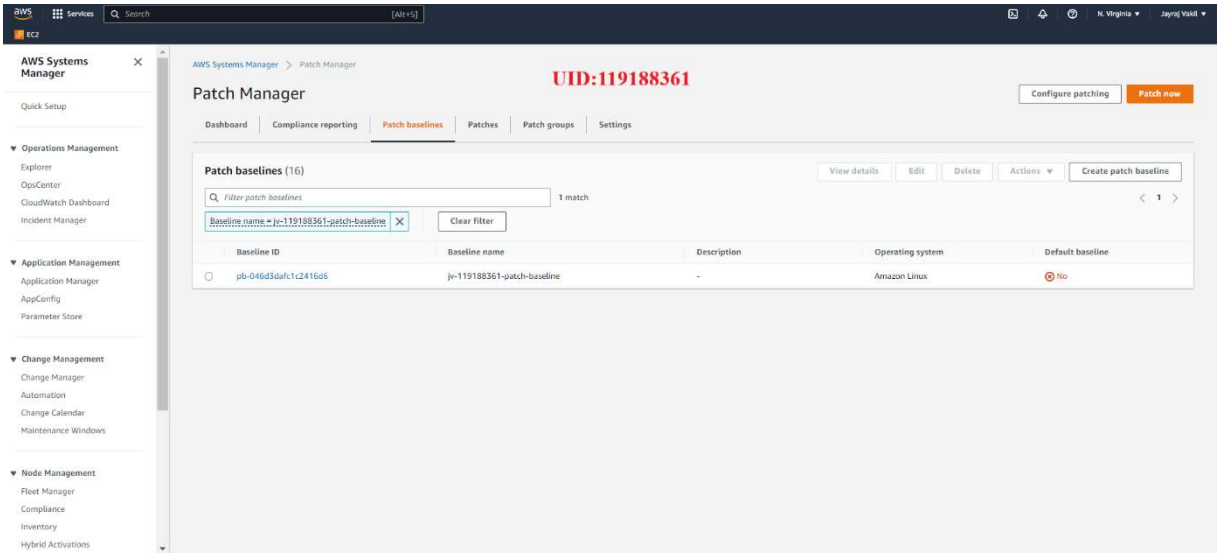

Figure 38. Patch baseline successfully created

Why this tool?

The process of patching is automated with regard to security updates as well as other updates. Patch Manager uses some baselines for its patching called patch baselines. The patch baselines can be predefined as well as custom-built. Patch Manager first scans all the instances and searches for all the missing patches and then proceeds to install them. They can be installed in large groups by using Tags or can be individually installed. On

top of this, if there is a vulnerability that requires immediate action then AWS patch manager provides **Patch Now** option to patch all the instances affected immediately.

Some of the useful features of this Patch Manager are auto-approving patches, patch scheduling, reporting compliance while scanning, and storing the report directly to the S3 bucket. Also, as it is a part of the AWS family, it integrates well with IAM, CloudTrail, and other services to give a secure patching experience.

➢ Recommendation 5: DDoS Prevention

DDoS attacks are usually performed to disrupt the network services with unnecessary requests to the server than they can handle. The current server of Cobra Kai is found to be vulnerable to DDoS attacks. They are suspecting that their competition who is Daniel LaRusso is staging DDoS attacks.

To mitigate this threat, AWS Shield is already implemented by default and it's free. To further strengthen the infrastructure against DDoS, we will implement AWS Web Application Firewall (WAF) through Web Access Control Lists (ACLs).

Follow the below steps to implement WAF:

1. Headover to **AWS WAF and Shield>Web ACLs** and then click on **Create web ACL.**
2. Now, enter the name as you like, as well as the description. Select **Regional resources** as we are using ALB to forward the requests and we need to protect ALB, select the region where the ALB is hosted and finally add our **ALB** to the associated AWS resources.

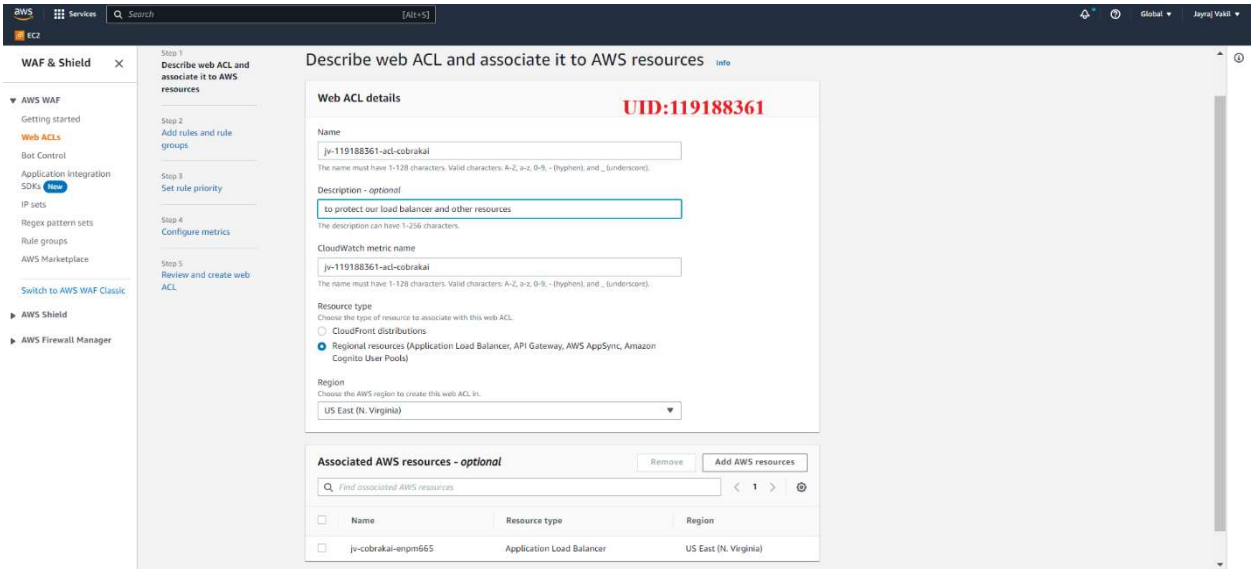

Figure 39. Defining web ACLs

3. We will add the required the AWS managed rules for our ACL and below are the rules we have added in order of priority. Whenever, an action is matched with the rule, it will take the action to block the request.

   The rules we have implemented in our ACLs are

   - To protect admin from malicious users trying to gain admin access.
   - Amazon IP list which is based on AWS's own threat intelligence.
   - Anonymous IP will block requests from unknown identities like Tor nodes, proxies or someone who hides their identity.
   - Core rules are used for applications to protect them from a wide spectrum of vulnerabilities found.
   - Known bad inputs works on Regex patterns.
   - Linux OS to protect our Linux instance from its associated web attacks.
   - SQL database rule to protect from SQL based attacks.

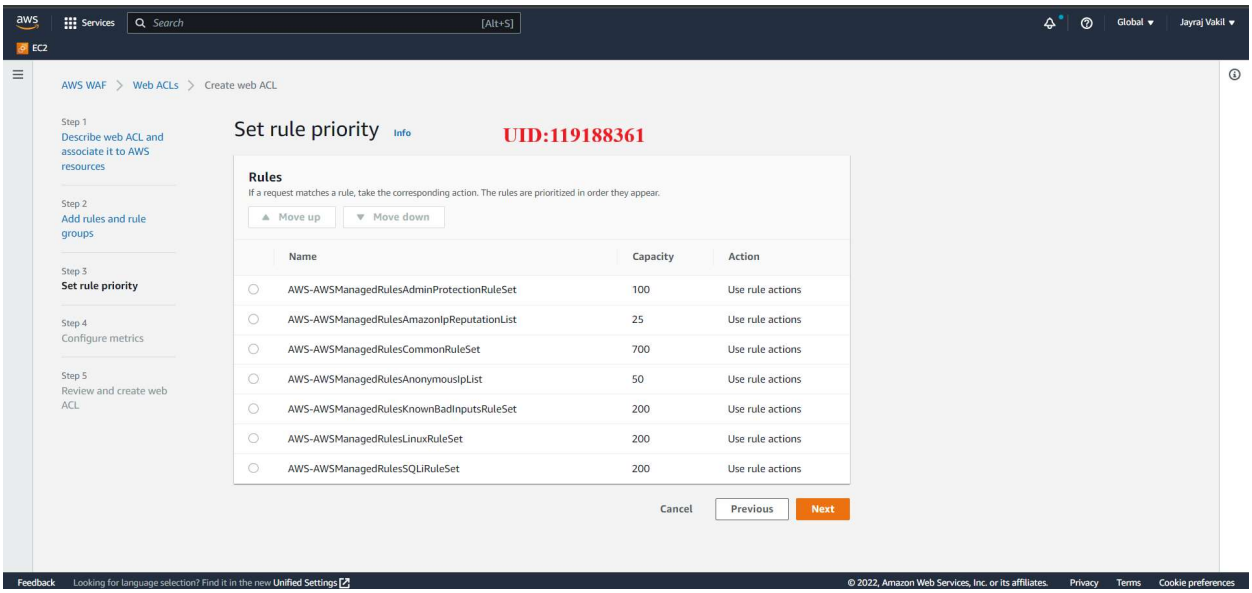

Figure 40. Rule Priority for ACL

4. After this review the ACLs and hit on **Create** button.

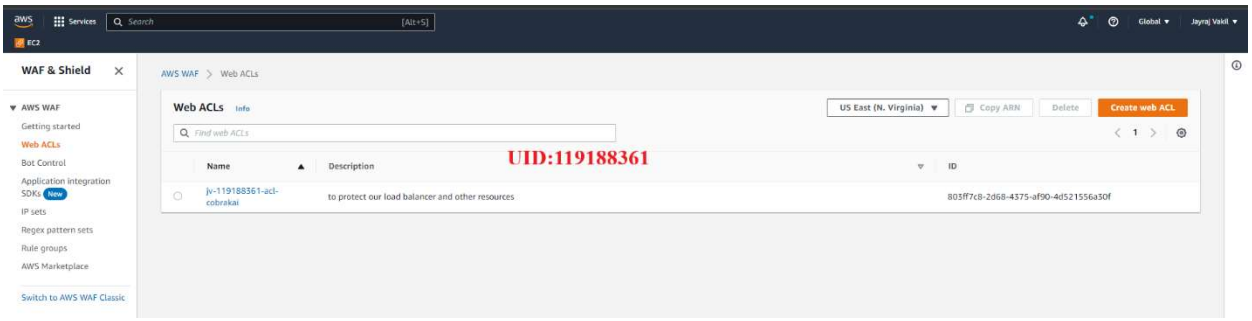

Figure 41. Web ACLs created

Why this tool?

We have already used load balancers and auto scaling group to redirect most of the requests. But still seeing the rivalry with LaRusso, we cannot be sure when an uncertain DDoS attack might happen. So, we are using AWS Shield and WAF. The free version of Shield doesn't provide much except standardized protection against DDoS attacks. Therefore, we are implementing ACLs to tighten the security.

➢ Recommendation 6: Slow streaming, downloading and order processing

Cobra Kai users were complaining about the slow streaming, downloading, and also order processing. Now the server cannot hold the incoming users due to exponential increase in the userbase. To overcome this issue, we will use AWS CloudFront and Route53.

First, we will implement Route53 by following the below steps:

1. Search for **Route53** in the search box and navigate to the dashboard then click on **Hosted zones>Create hosted zone**.
2. After that give the domain name you want and the description, then select the type as **Public hosted zone** as it will be the Domain Name Service (DNS) for the end-users and click on **Create hosted zone**.
3. Now, click on **Create record,** then enter the record name, select the record type as **A record**, and value as the **IP address of our Application load balancer**, finally select routing policy as **Simple.**

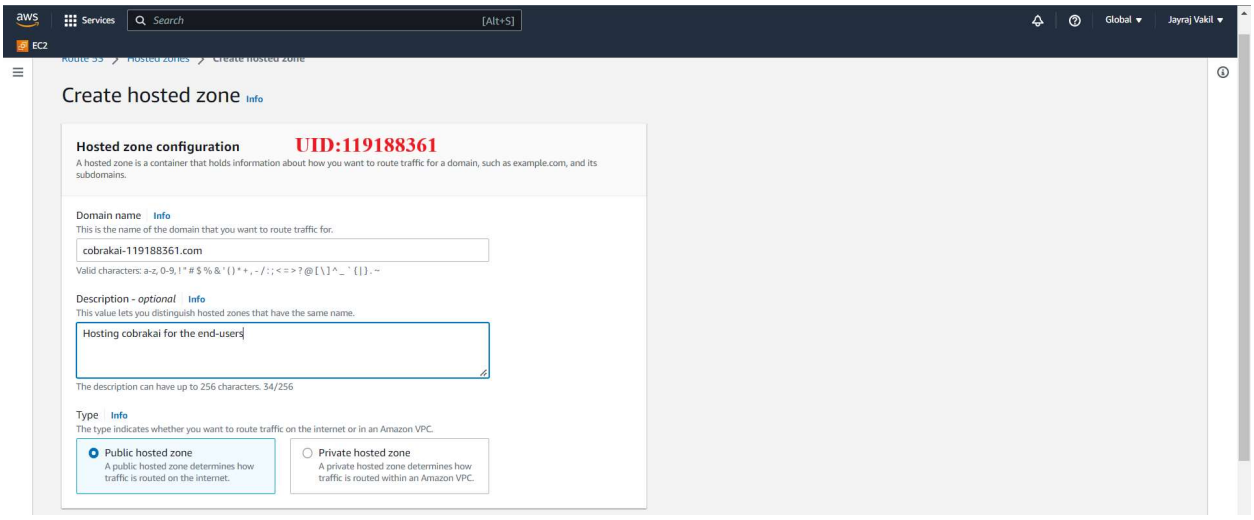

Figure 42. Route53 configuration

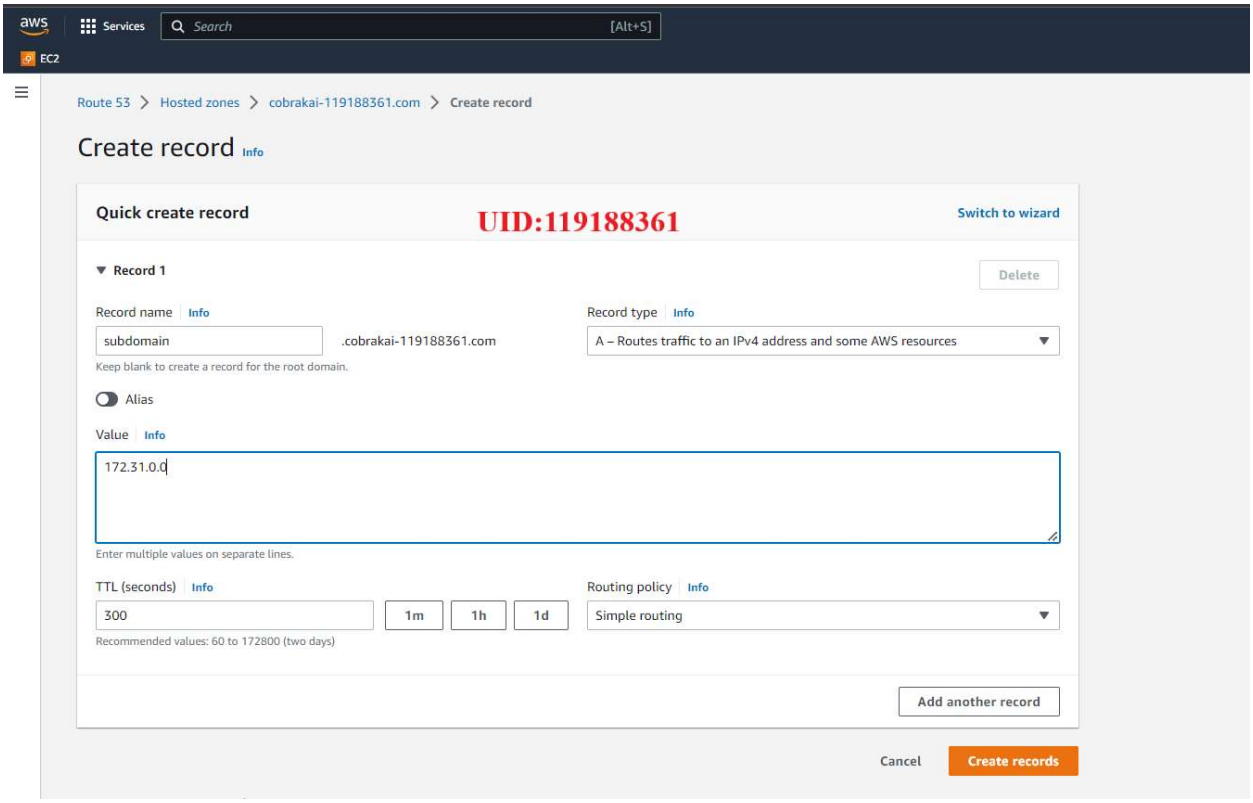

Figure 43. Create record for Route53

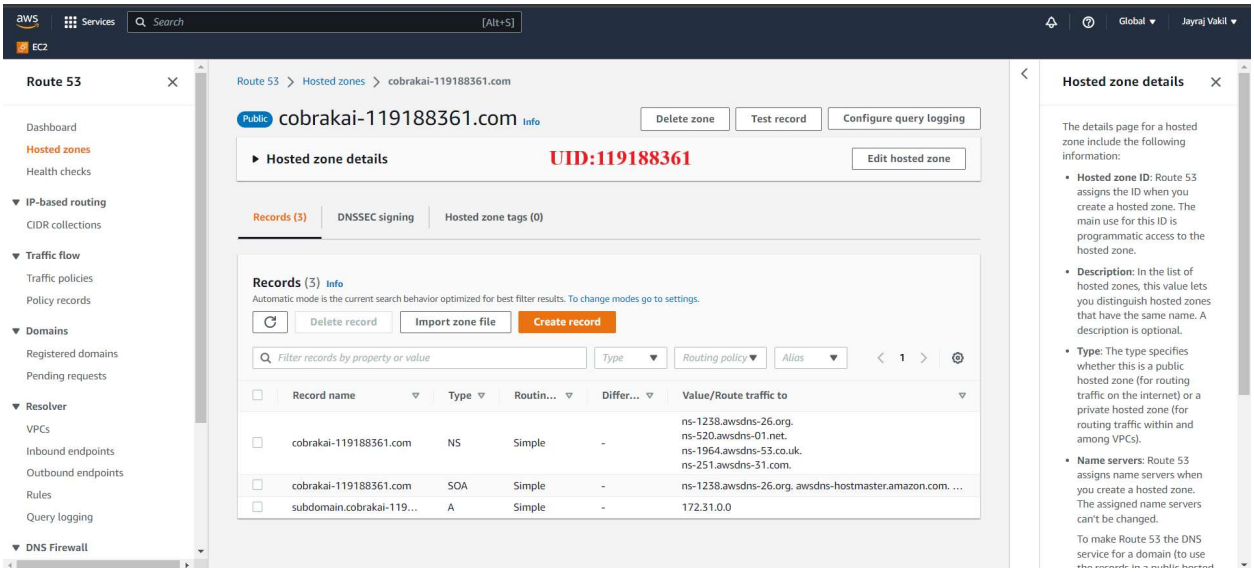

Figure 44. Route53 zone hosted

To implement CloudFront, we will follow the below steps:

1. Navigate to **CloudFront** using the search box and select **Create distribution** button.
2. Enter the origin domain name as our **S3 bucket's domain name** and give the name as you wish**,** keep the origin access as **Public**.

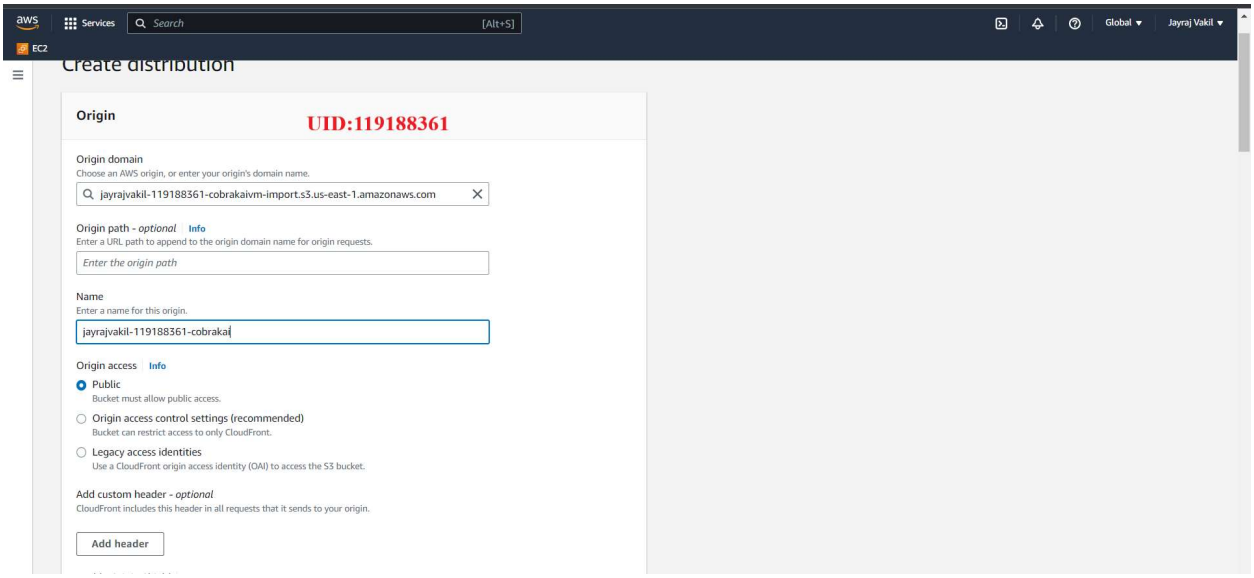

Figure 45. CloudFront distribution creation

3. Select **HTTP and HTTPS** as the viewer protocol policy and keep the rest as it is.
4. In the **Custom SSL Certificate** field, click **Request certificate** and follow the same as shown in the image below.

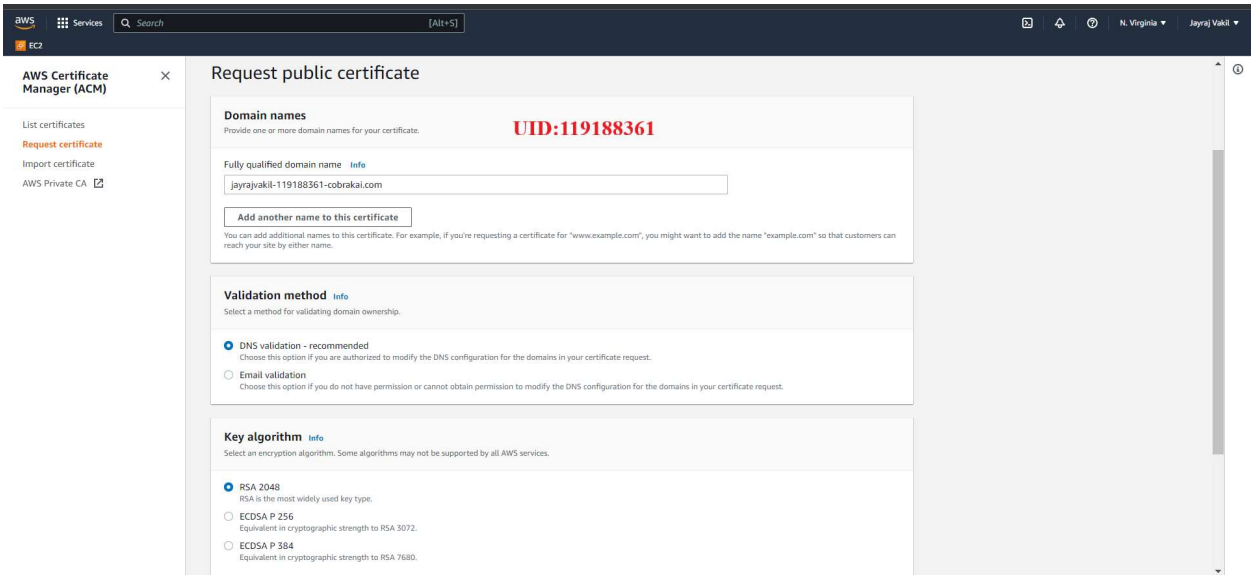

Figure 46. Request public certificate

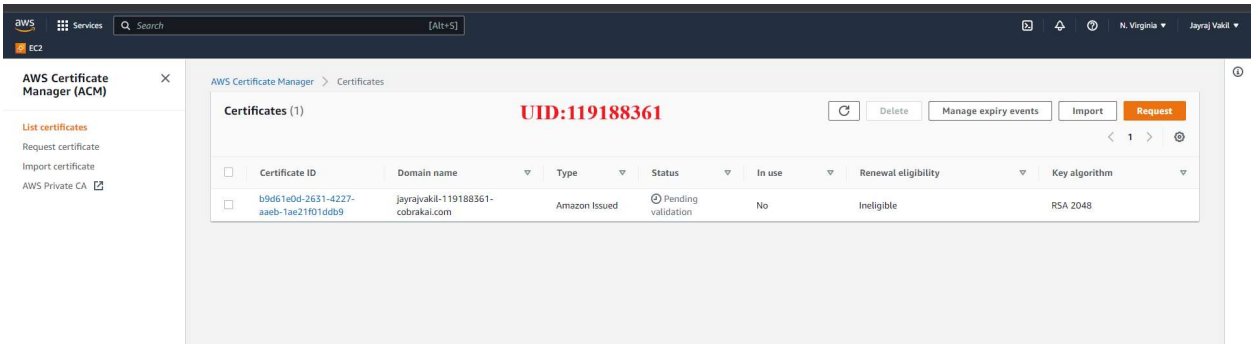

Figure 47. Certicate created

5. After this, select the certificate created above and then click on **Create distribution** button.

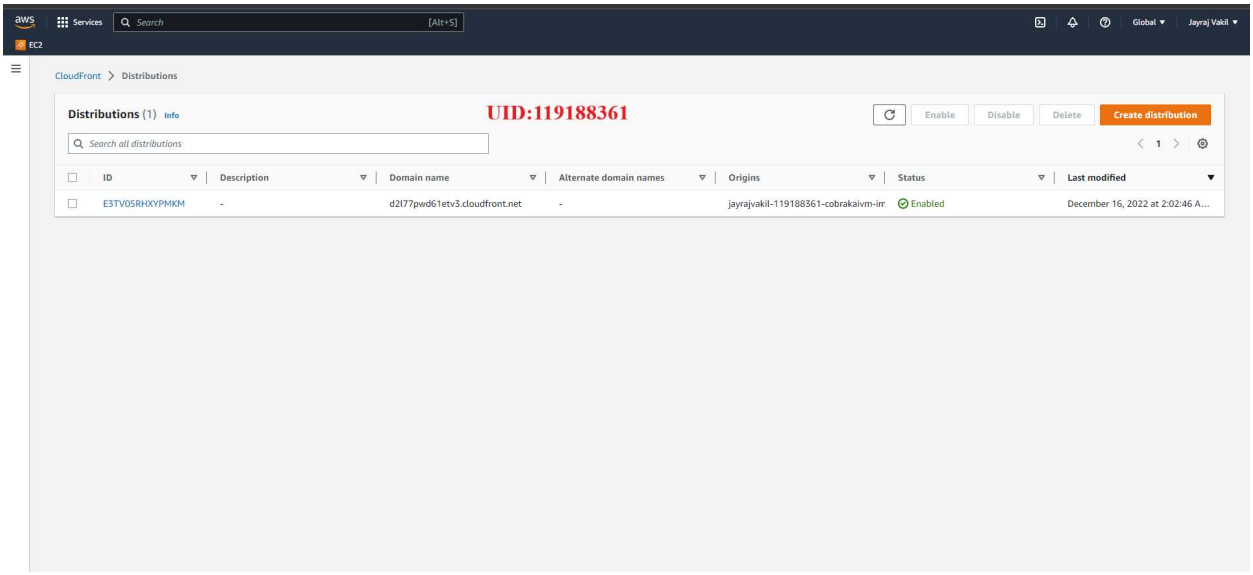

Figure 48. CloudFront distribution created successfully

Why this tool?

Route53 is a DNS which will be the front face of our public facing servers and will route the traffic requests to the load balancer. The record created here will direct point to our load balancer through the CloudFront distribution. The CloudFront distribution will be handling the static content of the Cobra Kai website to reduce the load on the server which will improve the slow-streaming, downloading issues the users face by giving users low latency. Also, the viewer access can be restricted right through CloudFront which can help in implementing geo-restriction. Both these tools are in compliance with the security standards such as HIPAA, DSS, and PCI.

➢ Recommendation 7: Processing credit cards and storing customer PII

Currently, Cobra Kai just processes credit cards and stores customer data without any protection to the data. If by any chance there is a data breach of Cobra Kai, all the personal and sensitive information of the users will be leaked and this would sully the reputation of Cobra Kai. To overcome this issue, we will store the credit card information and Personal Identifiable Information (PII) in our database securely. For this we will use AWS Relational Database Service (RDS).

Follow the steps below to implement the database for the Cobra Kai:

1. Search for **RDS** in the search bar of AWS. Navigate to RDS and click on **Create Database** button and select **Standard create**, select **PostgreSQL** for database.

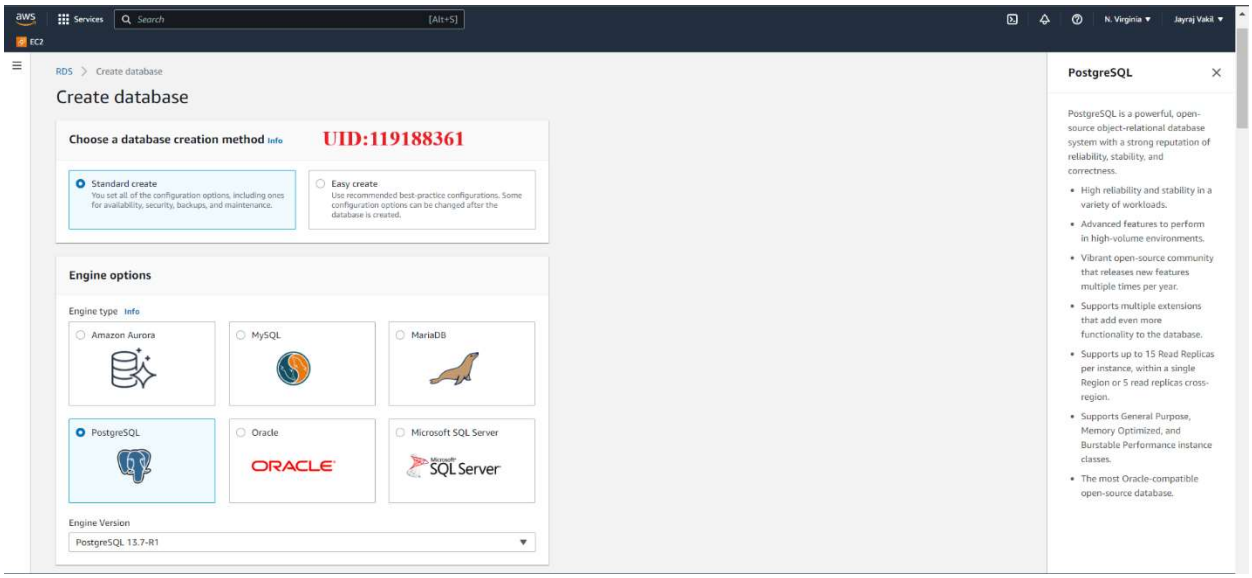

Figure 49. Creating Database

2. Choose the **Dev/Test** option for template as it is cost-effective and we won't be requiring the database as often. So, there is no need to select production option. For the availability and durability, we will choose **Multi-AZ DB Cluster**. We are selecting this option because it will be creating readable standby instances in other availability zone along with the primary instance so that even in case of anything going wrong, we can still access the database through other availability zone.

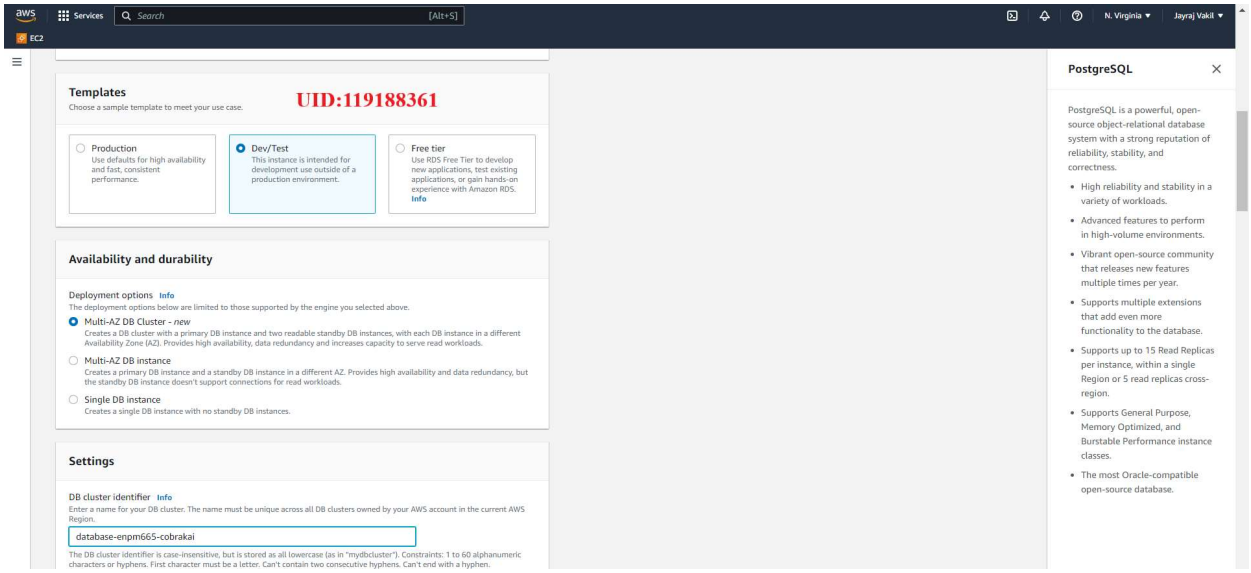

Figure 50. Template and other settings for database

3. Now configure the other settings such as database cluster name, database username, database password, instance class, storage size as you wish.
4. For the connectivity, we will connect this to our **EC2 instance** which will then automatically fill out the other fields such as VPC, subnets and then a new VPC security group will be attached to it after creation. Also check that the port 3306 or 5432 is open inside the security group for communication.
5. Now select **Additional Configuration** tab, inside which checkmark **enable auto backups,** keep the **backup retention period** to the **number of days you desire**, and also **enable encryption** and choose the **default aws/rds key**. On top of it, export the logs too. Finally **enable delete protection**, due to this no one will be able to delete the database. After this hit on **Create database button** and our database will be created.

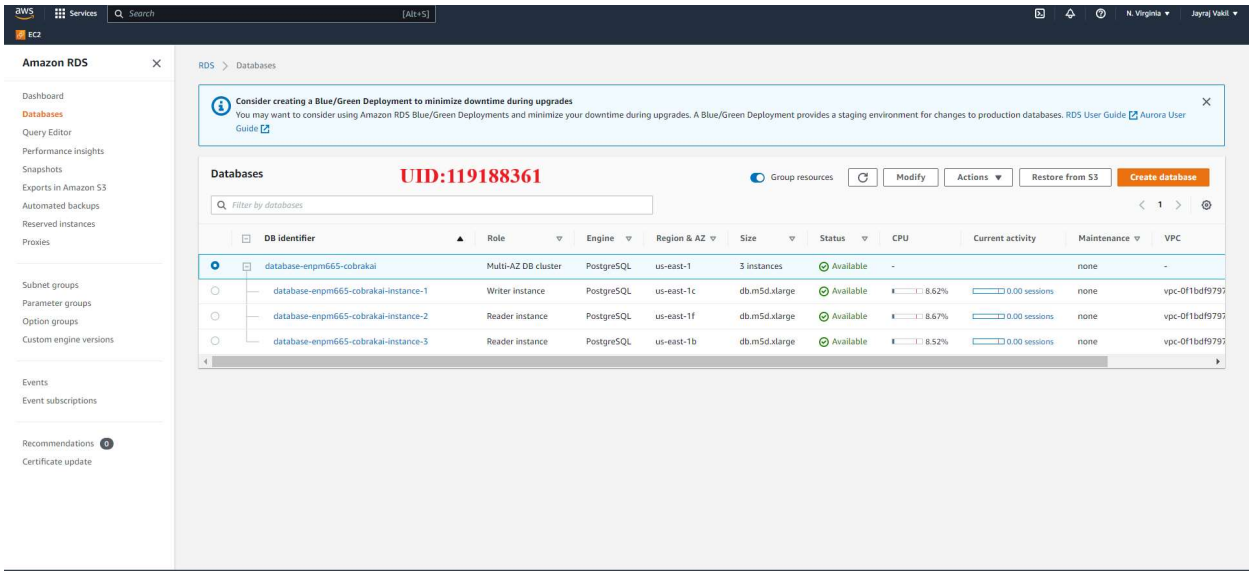

Figure 51. Database created

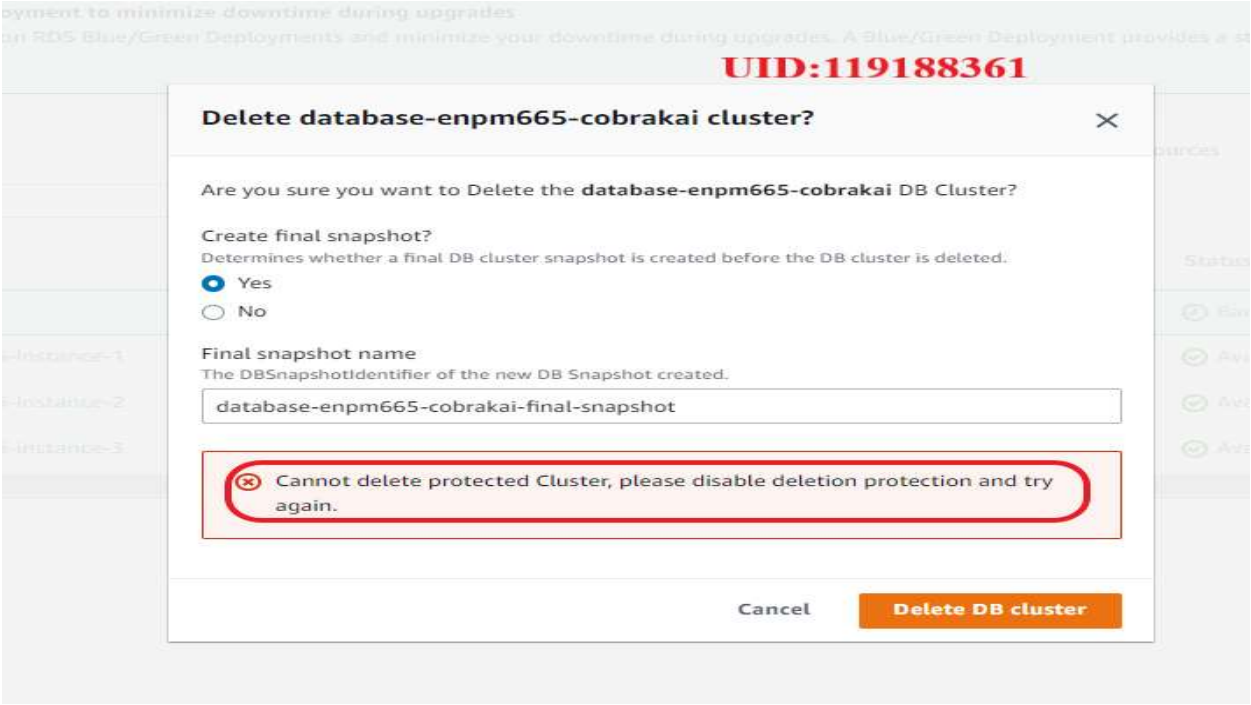

Figure 52. Cannot delete database due to protection

Why this tool?

Our credit card data and customer PII will be stored in a Relational Database Service (RDS) instance. RDS allows us to easily scale the storage, scale computation, automated backups, pay-per-use policy, and allows encryption. The data will be available in multiple zones. The encryption is handled by AWS Key Management Service (KMS). KMS encrypts sensitive data by leveraging what is known as Hardware Security Module (HSM). AWS managed keys are the most efficient way to encrypt without involving much complexity.

KMS has been certified by compliance regimes. It is PCI DSS level 1 compliant, FIPS 140-2, FedRAMP, HIPAA, DoD CC SRG, and ISO.

➢ Improved Architecture

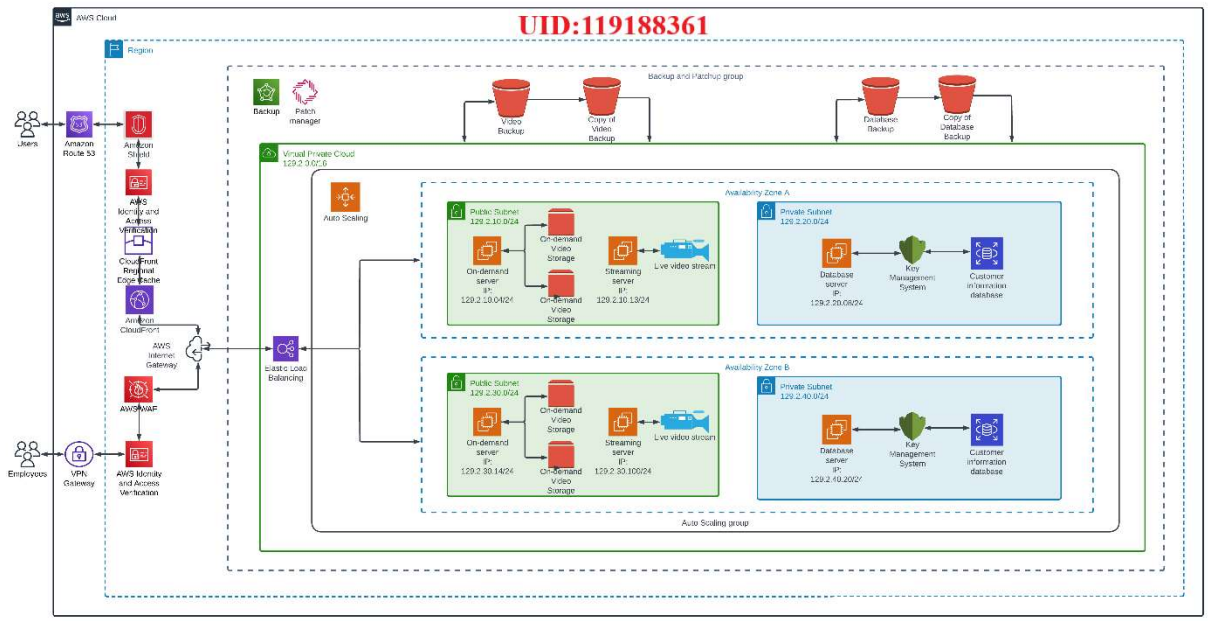

Figure 53. Cloud Architecture

Conclusion:

Through AWS, we have implemented various strategies and tools to secure the infrastructure. We have implemented the following to overcome security loopholes in the previous infrastructure:

1. AWS Patch Manager to patch the resources on a scheduled maintenance window.
2. AWS Backup to backup the resources incase of a disaster happening to any of the critical resources.
3. AWS IAM to implement account permission strategy so that only authorized users can access the resources
4. We have migrated the whole on-premise infrastructure to AWS cloud to mitigate the hardware failure issues as well as implemented application load balancer and auto scaler which will redirect the request to other availability zone in case of heavy workload or server failure.
5. To protect against DDoS attack, we have implemented AWS Shield and Web ACLs through WAF. On top of it, the application load balancer and auto scaling will also help.
6. To resolve the issue of slow streaming and downloading, we have implemented CloudFront as well as Route53.
7. For storing PII and credit card information, we will use AWS RDS to store the data and encrypt the database fields.

All the tools we used and the way we used comply with the security standards.

References:

https://umd.instructure.com/courses/1336126/assignments/6164203

https://docs.aws.amazon.com/mgn/latest/ug/launch-cutover-gs.html

https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use.html

https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

https://aws.amazon.com/route53/

https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/#:~:text=Refactoring%20%2F%20Re%2Darchitecting%20%E2%80%94%20Re,in%20the%20application's%20existing%20environment.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volumes.html#backup-benefit

https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html

https://aws.amazon.com/backup-restore/use-cases/

https://docs.aws.amazon.com/whitepapers/latest/aws-operational-resilience/how-aws-maintains-operational-resilience-and-continuity-of-service.html

https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/rel_withstand_component_failures_failover2good.html

https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html

https://aws.amazon.com/blogs/aws/aws-shield-protect-your-applications-from-ddos-attacks/

https://www.whizlabs.com/blog/how-does-aws-cloudfront-work/

https://aws.amazon.com/kms/features/?pg=ln&sec=c/#AWS_Service_Integration

https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-import-snapshot.html#start-import-task

https://docs.aws.amazon.com/vm-import/latest/userguide/what-is-vmimport.html

https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html