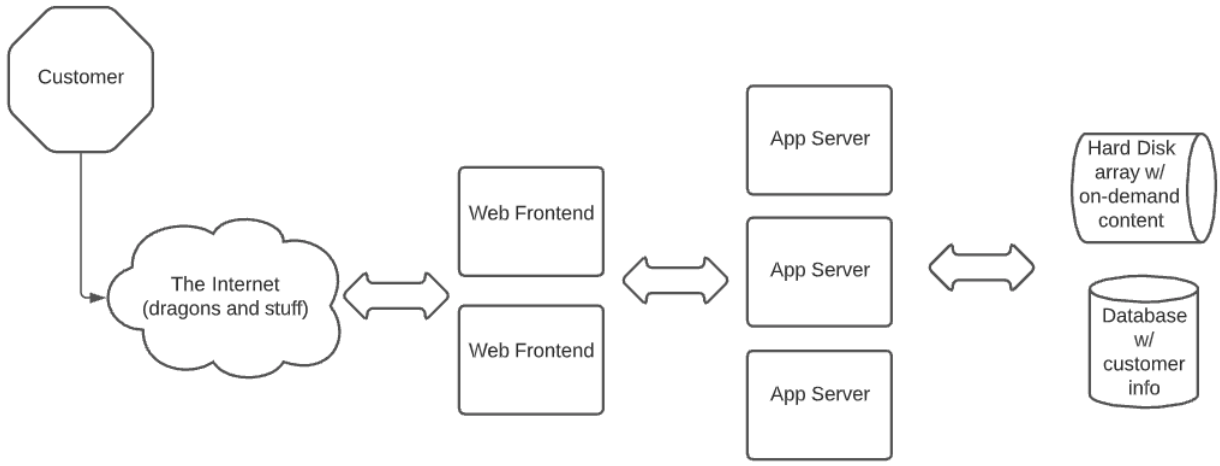
Final Project

Abstract → The document outlines a detailed solution to implement the technical plan (which was presented earlier) to migrate the architecture of CobraKai to AWS Cloud. The document provides strategies to resolve the prevalent issues in the on-premises architecture of cobra kai.

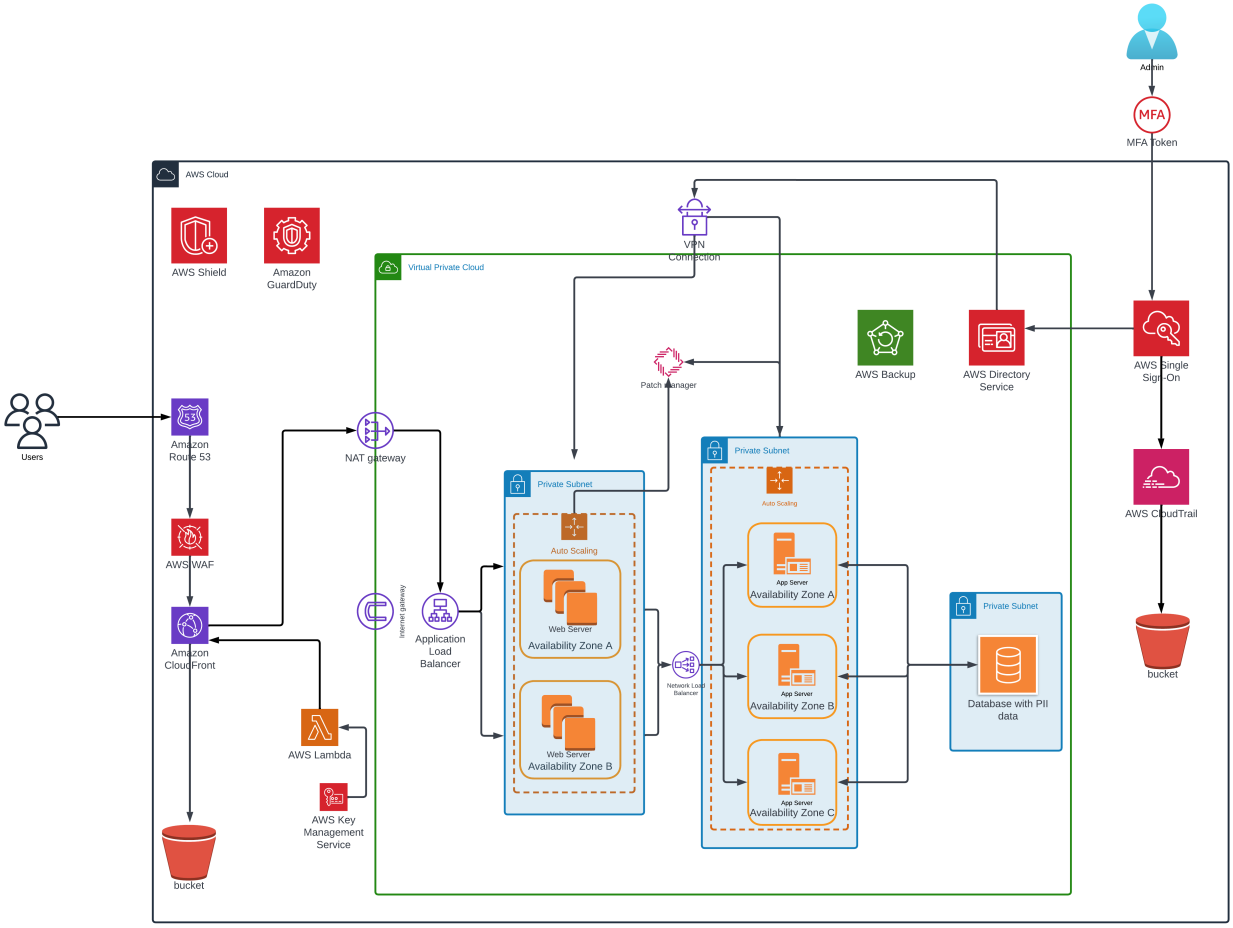
Current System →



Current Issues to Consider →

- Cobra Kai does not currently have a patching strategy
- Cobra Kai does not currently have a backup strategy
- Cobra Kai does not currently have an account permission strategy, every user has the ability to run privileged commands on the web server if they want to
- Their entire website infrastructure is highly vulnerable to DDoS, hardware failures, and human error. It runs in a closet for crying out loud
- The website has experienced DDoS attacks and compromise attempts they suspect comes from a rival dojo ran by Daniel LaRusso who with his deep pockets has become a persistent threat against Cobra Kai's l IT operations
- Customers have complained about slow streaming, downloads, and order processing
- Cobra Kai's platform is processing credit card data and also stores customer PII (name, phone, email, address, and additional details about the customer)

Proposed Architecture



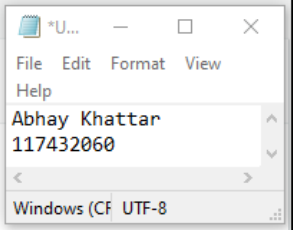
Technical Report

Configuring S3 bucket for image storage

- S3 is one for the building blocks of AWS, and is advertised as “infinitely scaling” storage
- In this scenario, we are using S3 to store the ova file which will be used to create the AMI (Amazon Machine Image), and consequently used to create the EC2 instances
- Following are some of the configurations of the S3 bucket →
  - Public access to the public has been blocked; this enables us to save the content of the bucket being compromised to any unauthorized user
  - Server Side Encryption has been enabled using the Amazon KMS (Amazon Key Management Service). This encrypts all the contents of the S3 bucket, when the data is at rest

Create bucket Info

Buckets are containers for data stored in S3. Learn more



General configuration

Bucket name

cobrakai-vm-117432060

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming

AWS Region

US East (N. Virginia) us-east-1

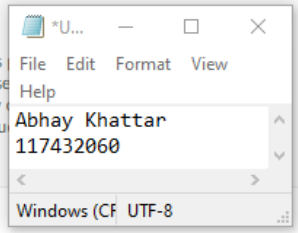
Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access
ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These se
and its access points. AWS recommends that you turn on Block all public access, but before applying any
applications will work correctly without public access. If you require some level of public access to this bu
customize the individual settings below to suit your specific storage use cases. Learn more



- [x] **Block *all* public access**

  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

  - [x] **Block public access to buckets and objects granted through *new* access control lists (ACLs)**

    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

  - [x] **Block public access to buckets and objects granted through *any* access control lists (ACLs)**

    S3 will ignore all ACLs that grant public access to buckets and objects.

  - [x] **Block public access to buckets and objects granted through *new* public bucket or access point policies**

    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

  - [x] **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**

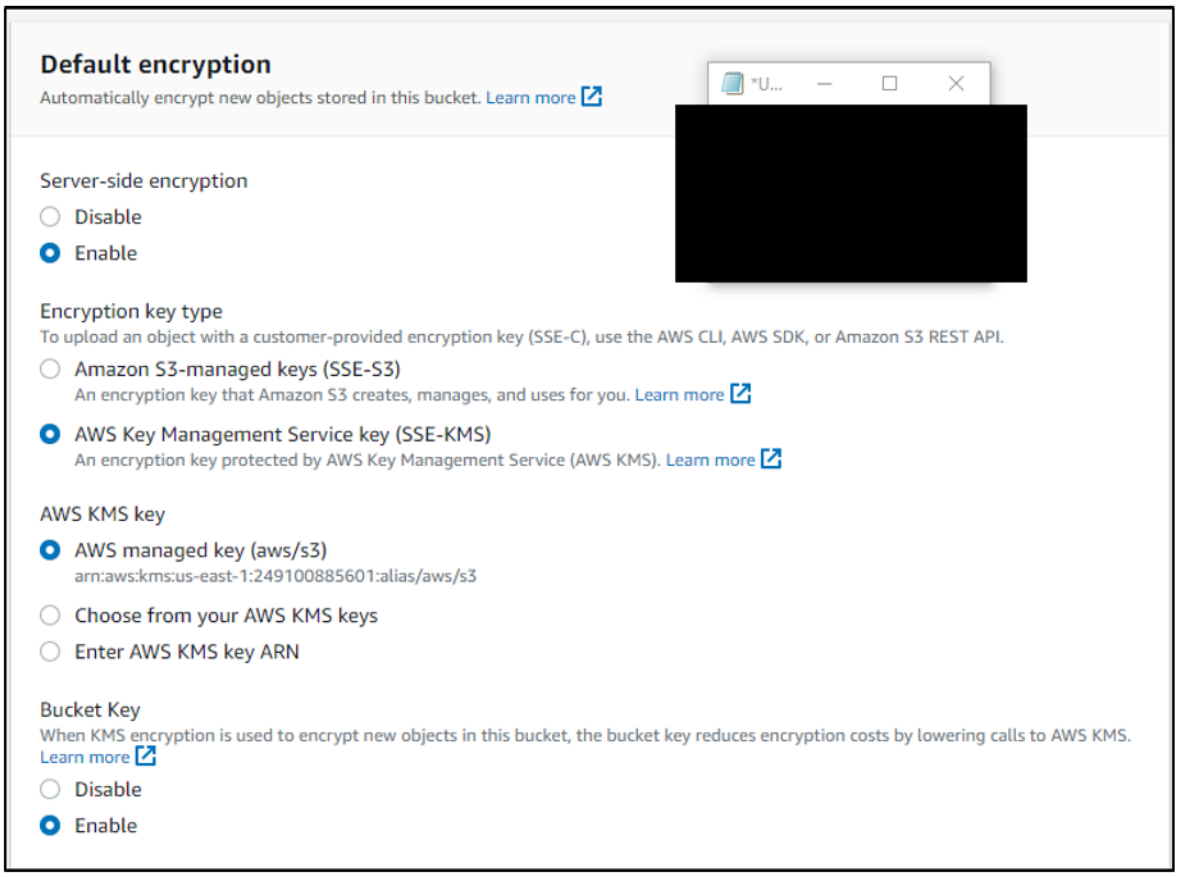
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning

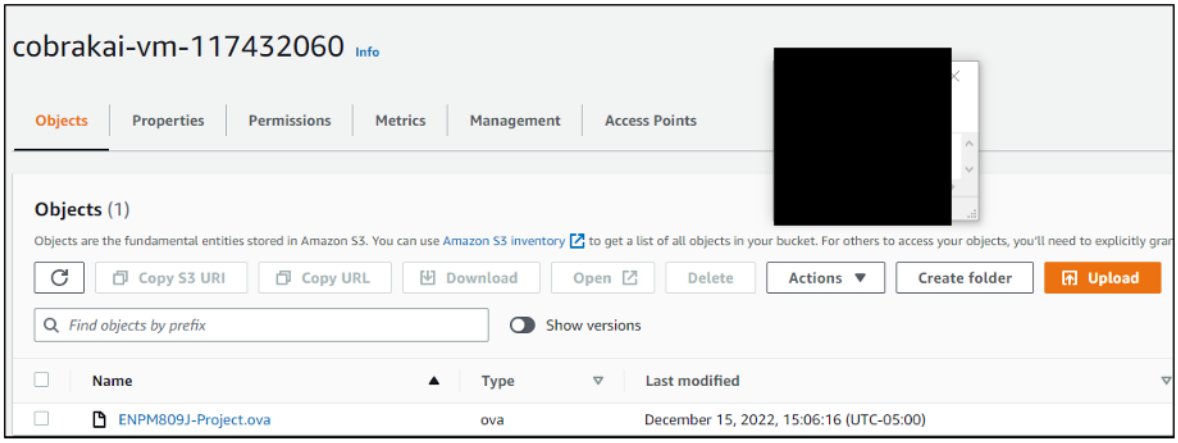
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

- (•) Disable
- ( ) Enable

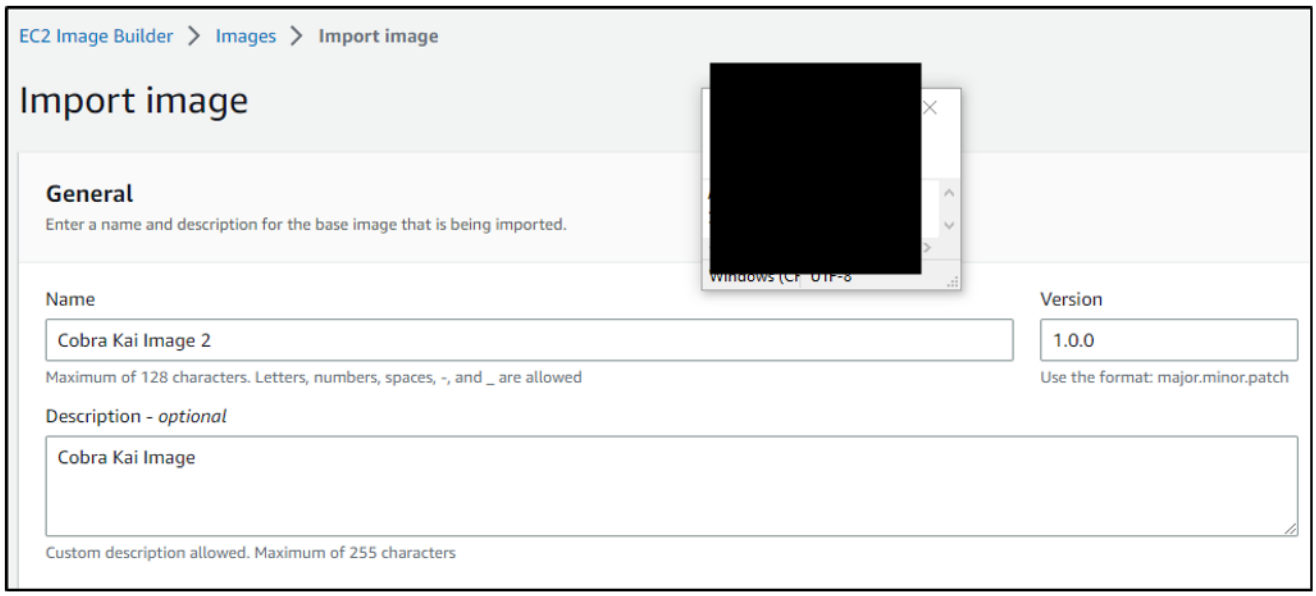


Upload the ova file to the S3 bucket

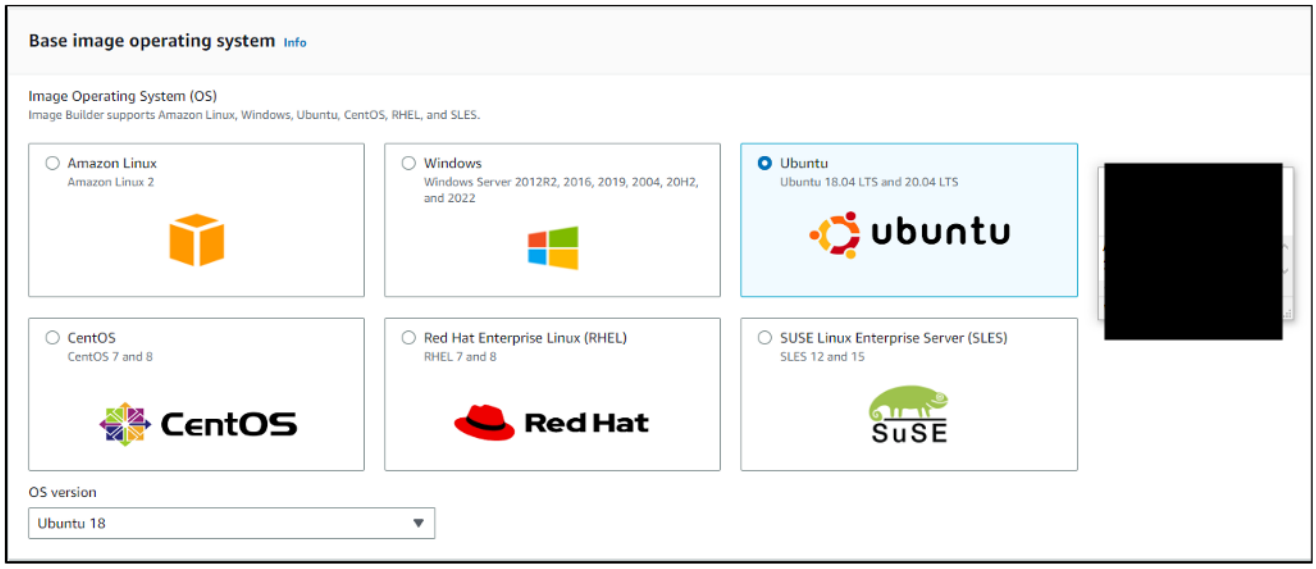


Creating AMI from ova file

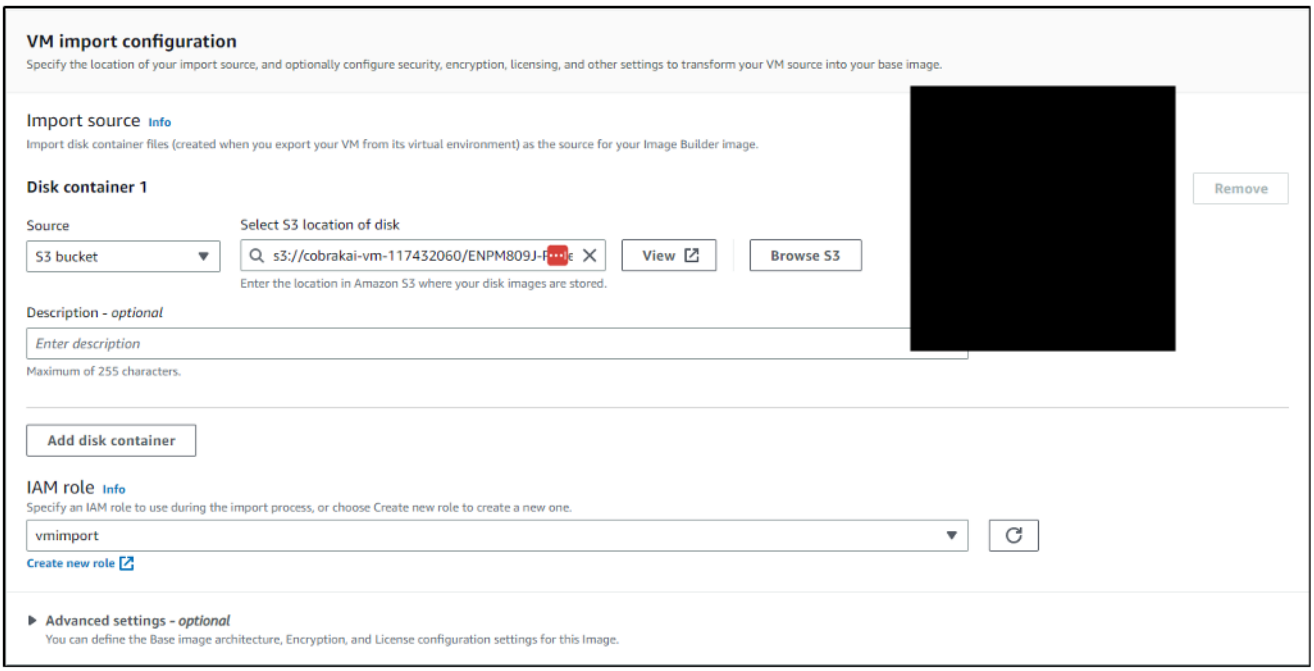
To create the AMI from the image, we need to create the image using the ova file in the S3 bucket, and we can do this using the **EC2 Image Builder**



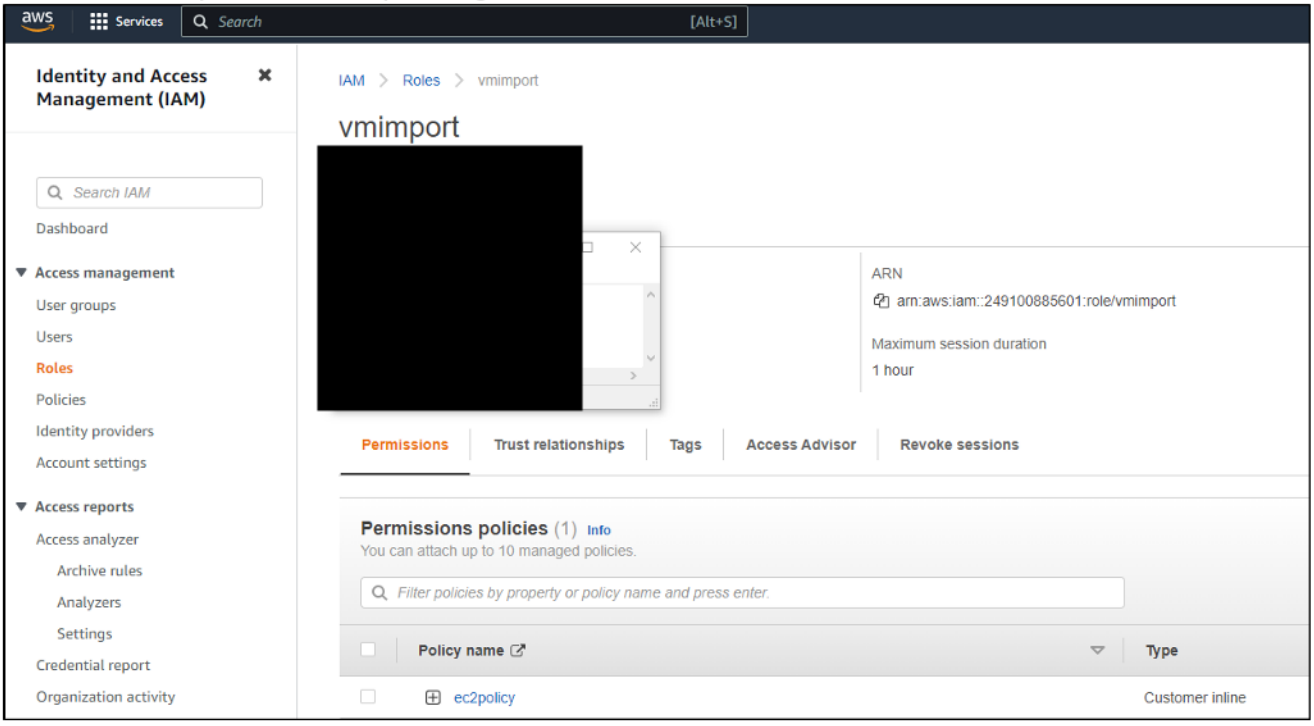
We choose the base OS as *Ubuntu 18*. This version of ubuntu is stable and because ubuntu is open source, it has been thoroughly investigated by the community for the vulnerabilities



We now configure the VM configuration, by selecting the S3 bucket with the ova file as the source



We have set the IAM role as *vmimport*[1] which is a new rule which has been defined keeping in mind the principle of least-privilege. This role only has access to the services which are required for importing the VM



Under the Permission Policies, we create a custom policy →

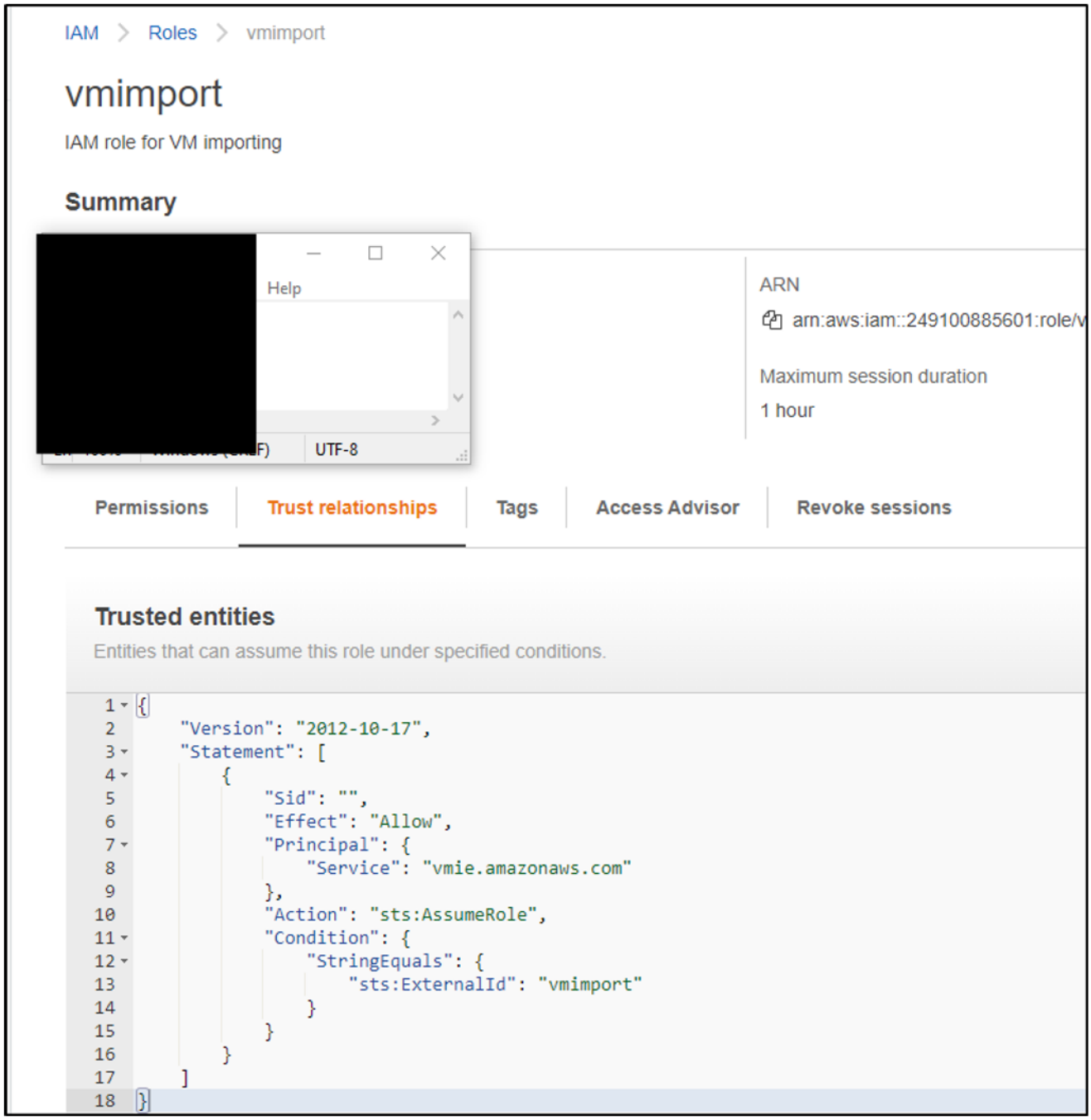
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::cobrakai-vm-117432060"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::cobrakai-vm-117432060/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifySnapshotAttribute",
                "ec2:CopySnapshot",
                "ec2:RegisterImage",
                "ec2:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

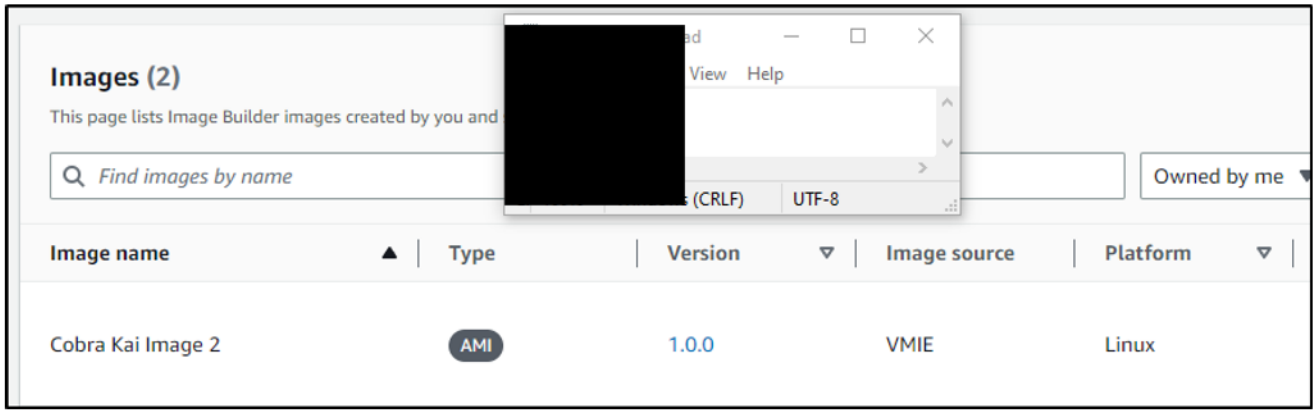
We also need to edit the *Trusted Relationships*, and enter the following policy in the *Trusted Entities*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "vmie.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:ExternalId": "vmimport"
                }
            }
        }
```

```
    ]
}
```



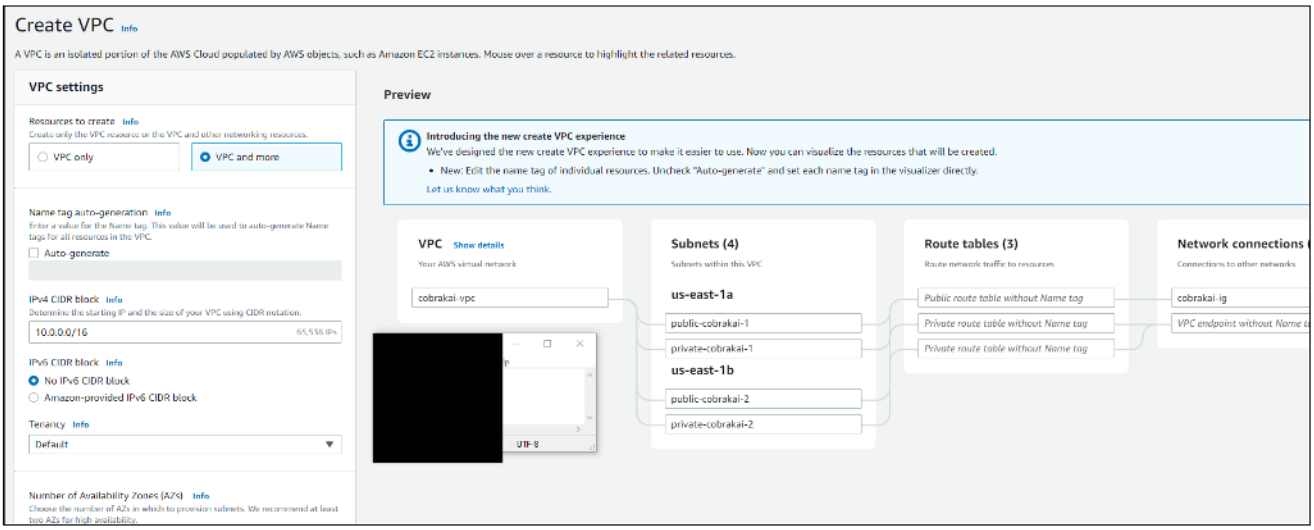
The image gets created, and can be used to create instances



Creating the Infrastructure

VPC

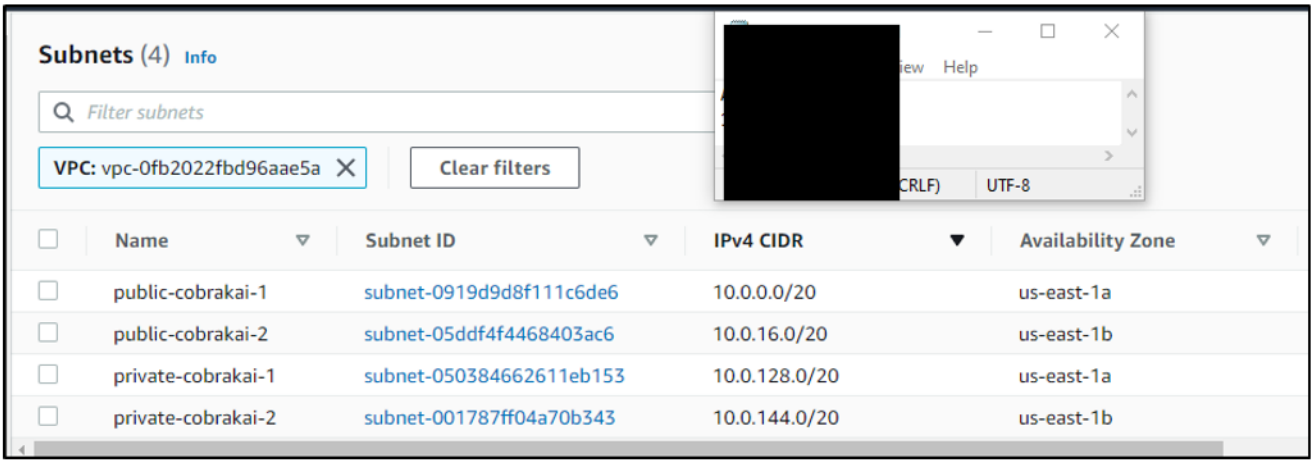
We start by creating the virtual private cloud where we will be placing all of our AWS infrastructures. This helps by creating a private network within the AWS Cloud to deploy our infrastructure.



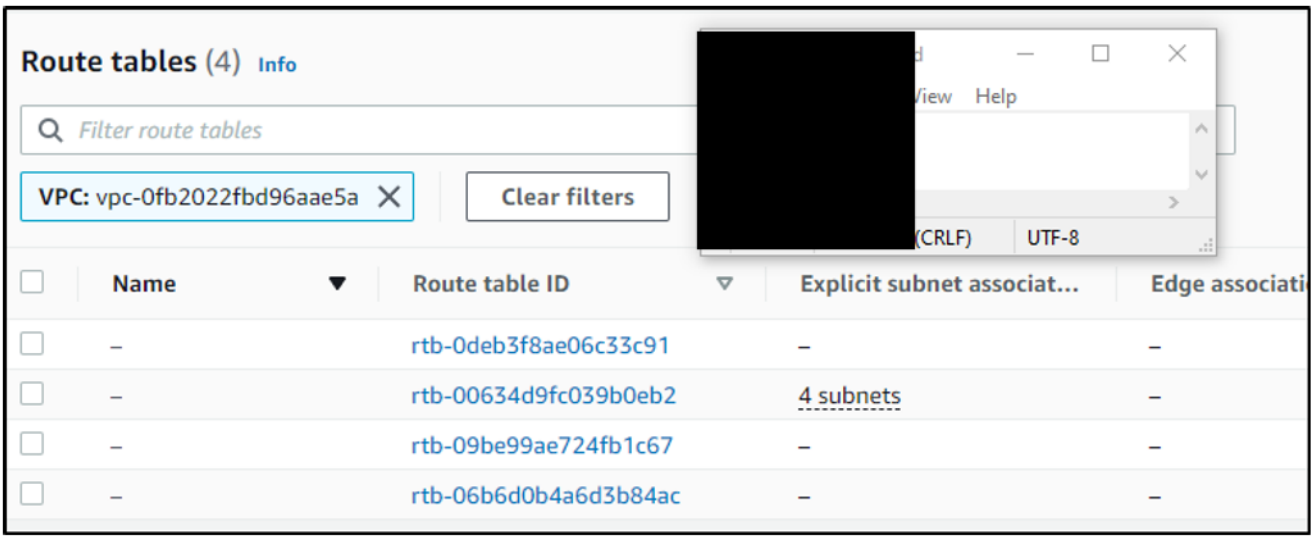
The VPC contains a total of Four subnets, one Internet gateway, and route tables. We create one public and one private subnet each in two different availability zones (US-east-1a, and US-east-1b)

The following are the IPv4 CIDR for the subnets →

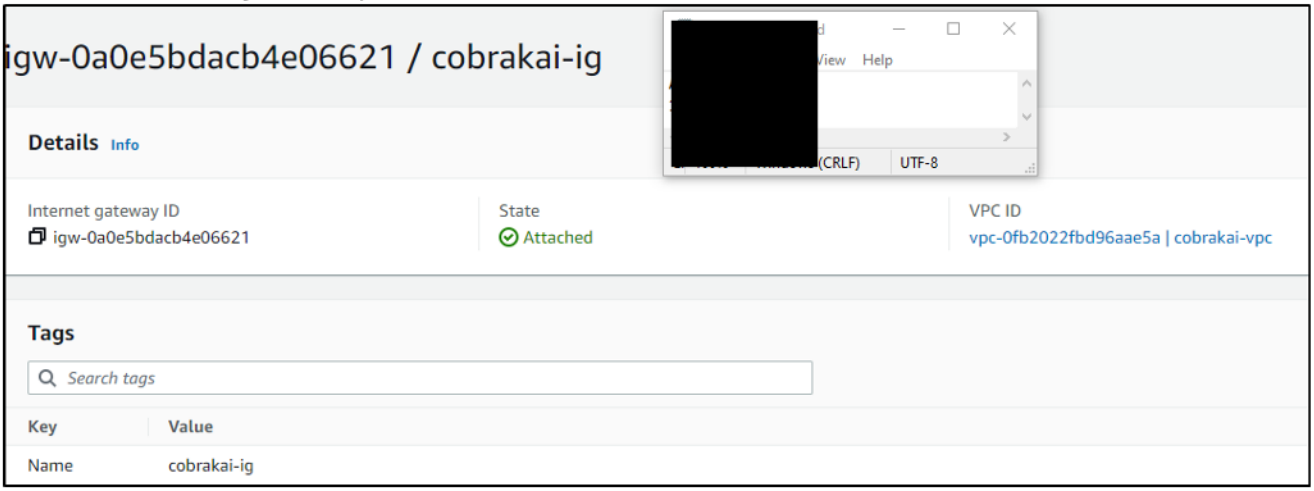
Name	IPv4 CIDR	Region
public-cobrakai-1	10.0.0.0/20	US-east-1a
private-cobrakai-1	10.0.16.0/20	US-east-1a
public-cobrakai-2	10.0.128.0/20	US-east-1b
private-cobrakai-2	10.0.144.0/20	US-east-1b



Route Tables →

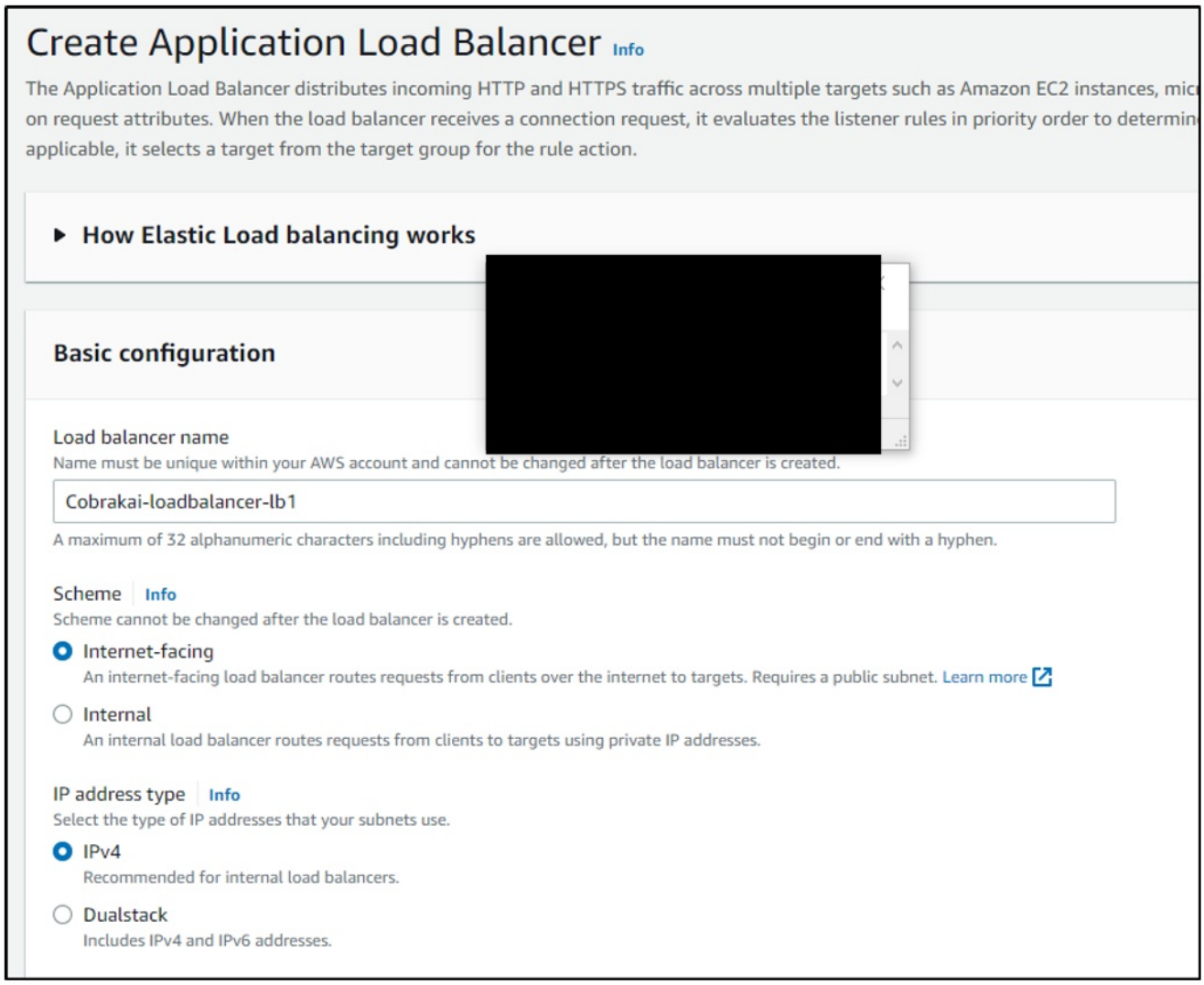


Internet Gateway → Helps our instances connect to the internet

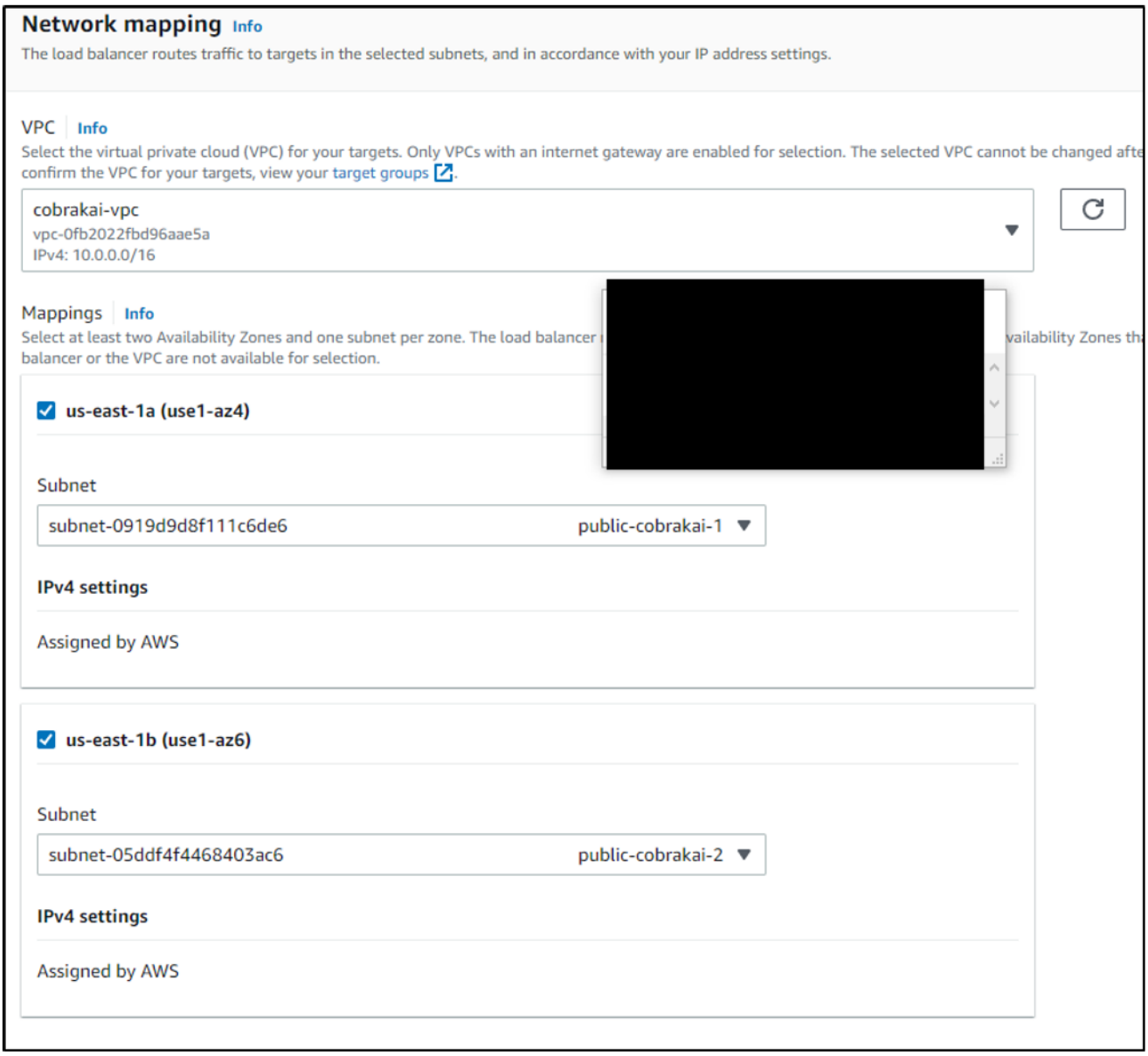


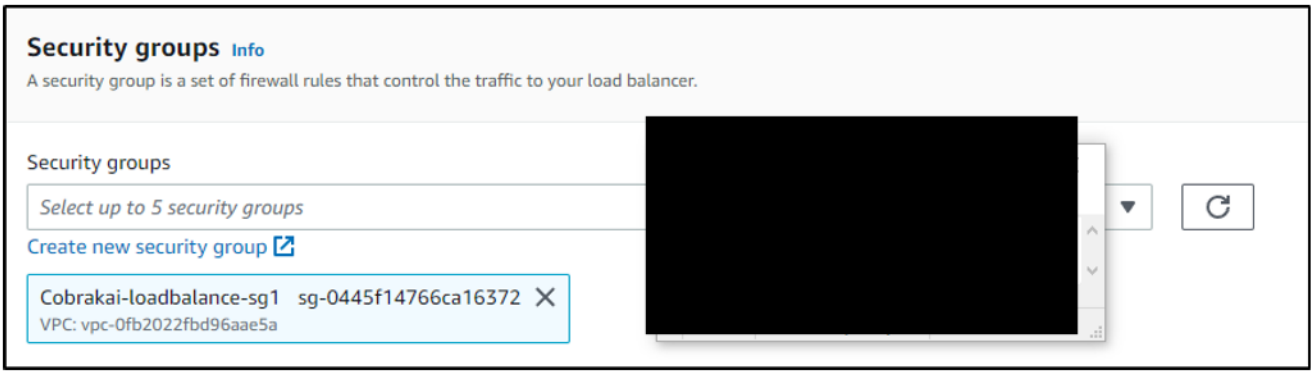
Elastic Load Balancer

We are using the Application Load Balancer (ALB) to helps distribute the internet traffic to multiple servers (EC2 Instances) . This will be the single point of access to the application (internet facing). The ALB works over Layer 7, as well as does regular health checks to the instances. This helps increase the availability of the application. We will integrate this with the *Auto Scaling Group*, and when the traffic increases the ASG would increase the number of instances, while the ALB will route the traffic automatically to help reduce the load on a single instance.



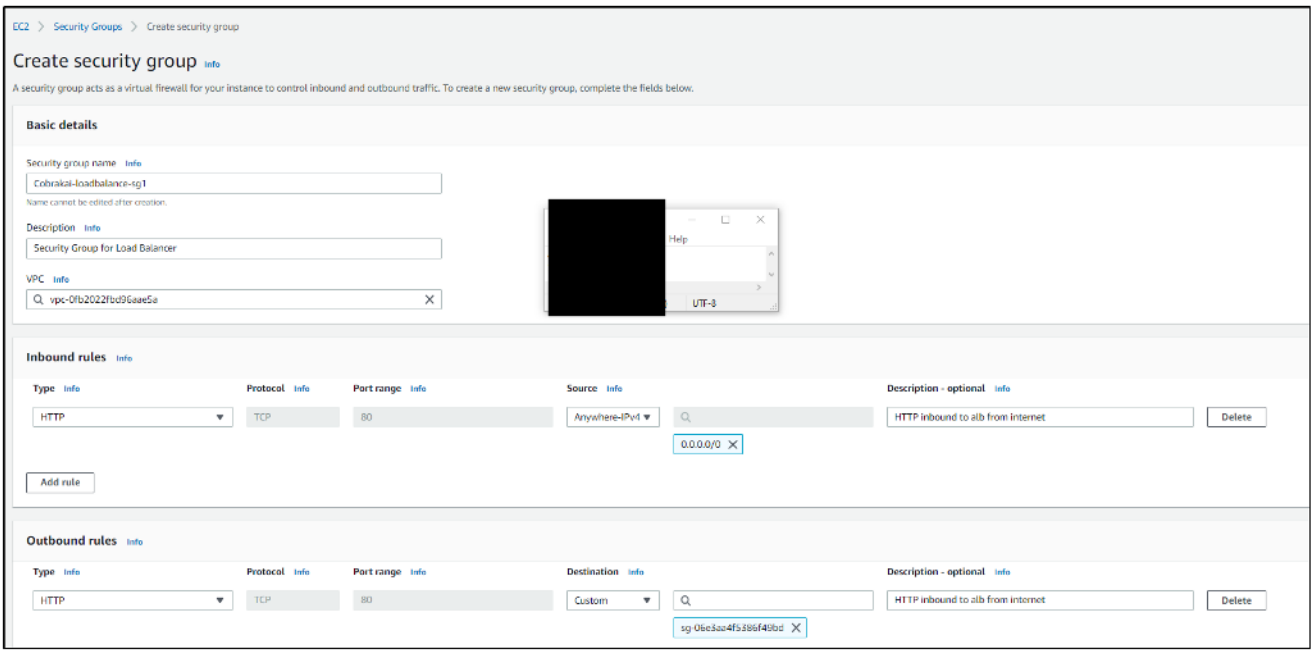
Setting the network configurations for the ALB. By putting the load balancer in atleast two regions, we are able to increase the availability of the application



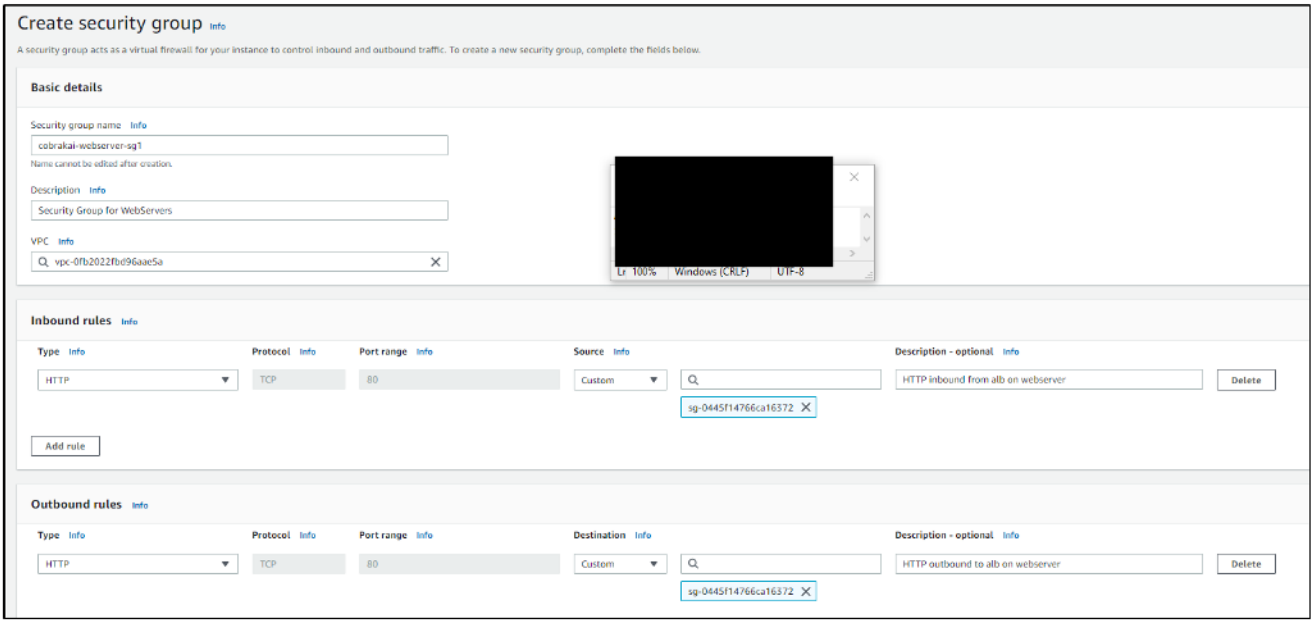


A **Security Group** acts as a Firewall to control the traffic to and from an EC2 instance. A SG has an *explicit Deny*, which means that if a rule does not allow for a certain traffic, it will automatically be denied.

Security Group for the Load Balancer → We allow traffic only on the HTTP port, and deny traffic to all the other ports

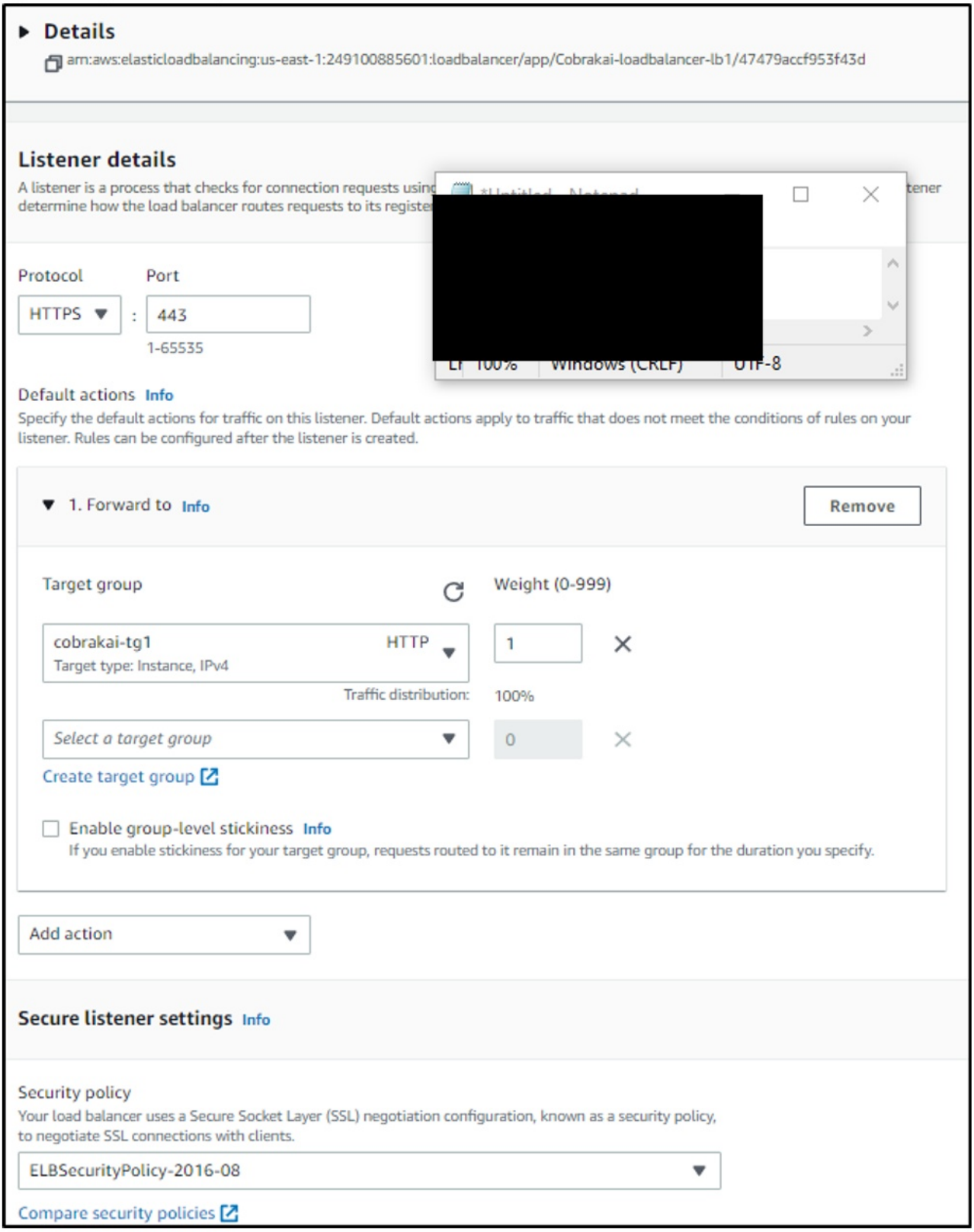


Security Group for the Web Server →



In this application, the traffic is currently being sent as HTTP request, i.e. it is not bein encrypted. This is a flaw in the application. We use the ALB to solve this vulnerability,

by having it list to port 443.

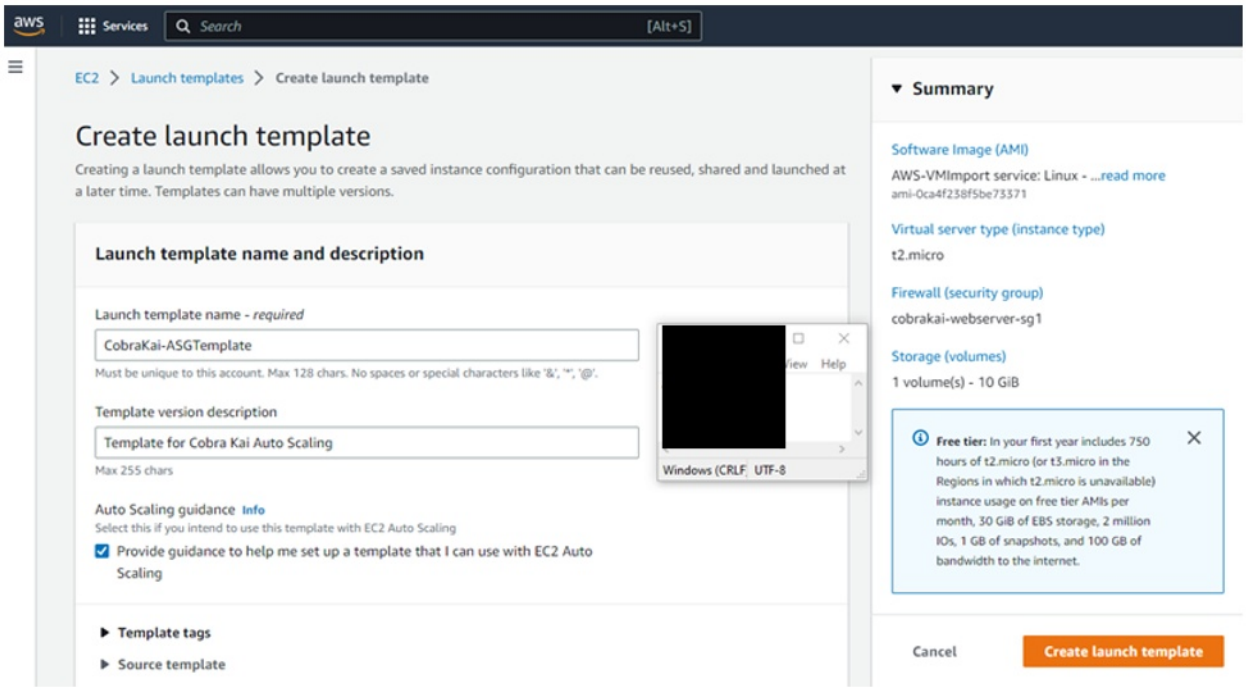


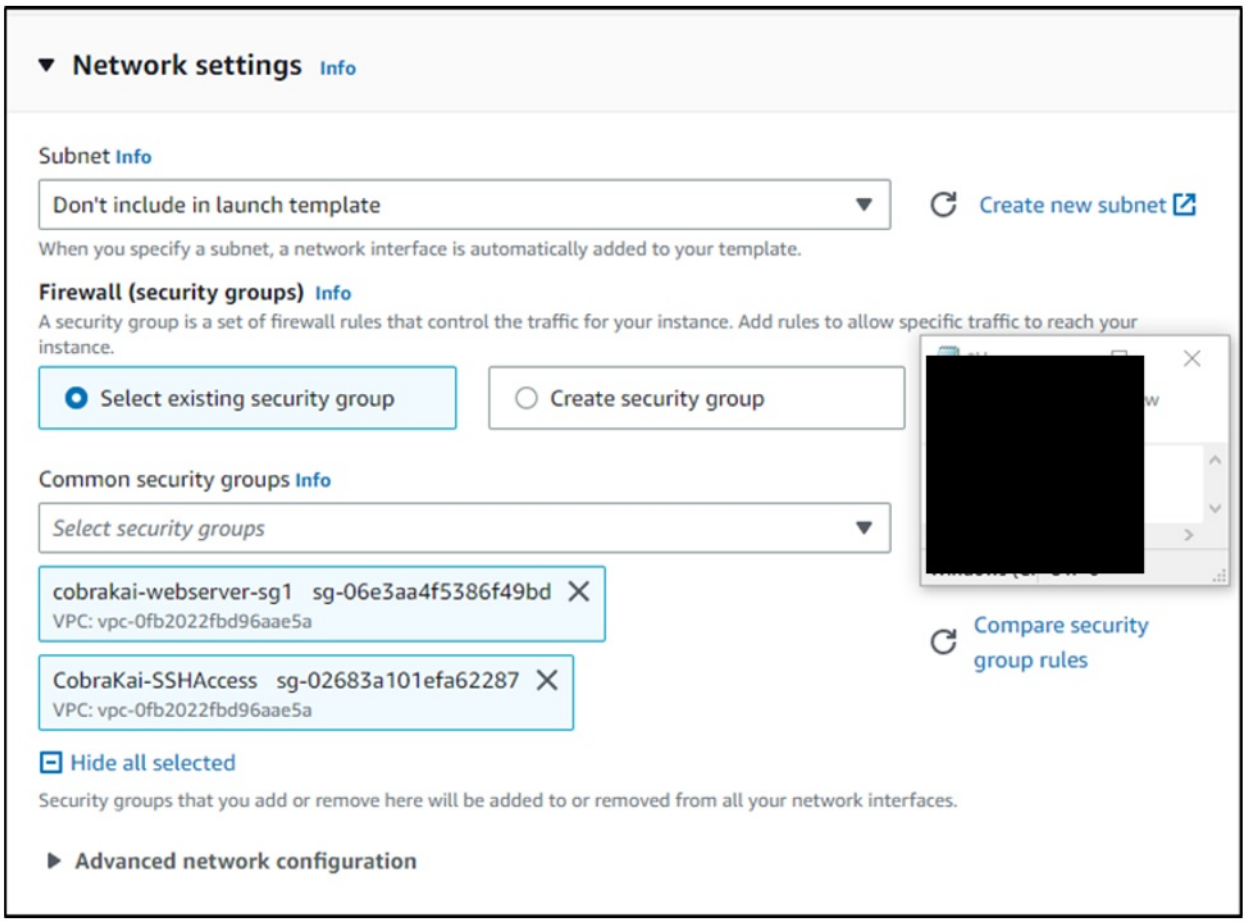
Auto Scaling Groups

We need to create a Launch template to create the auto scaling group. We set the template with the following configurations →

- Firewall - Specify the security groups, so that only the ports which are required are open

- Health Check of the instances every 300 seconds. We can use the health dashboard to make sure the required number of instances are running
- We can configure the size for the group size and scaling policies. We have set them as the following -
  - Desired Capacity - 2
  - Minimum Capacity - 1
  - Maximum Capacity - 3
- These can be adjusted according to the traffic that might be expected
- We also set a notification system, which will notify the subscribers when an event occurs




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
Common security groups Info
Select security groups
cobrakai-webserver-sg1 sg-06e3aa4f5386f49bd
VPC: vpc-0fb2022fbd96aae5a
CobraKai-SSHAccess sg-02683a101efa62287
VPC: vpc-0fb2022fbd96aae5a
Compare security group rules
Hide all selected
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Advanced network configuration

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

- ○ No load balancer
  Traffic to your Auto Scaling group will not be fronted by a load balancer.
- ◉ Attach to an existing load balancer
  Choose from your existing load balancers.
- ○ Attach to a new load balancer
  Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

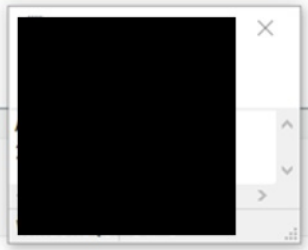
- ◉ Choose from your load balancer target groups
  This option allows you to attach Application, Network, or Gateway Load Balancers.
- ○ Choose from Classic Load Balancers

Existing load balancer target groups

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

cobrakai-tg1 | HTTP
Application Load Balancer: Cobrakai-loadbalancer-lb1



Health checks - *optional*

Health check type Info

EC2 Auto Scaling automatically replaces instances that fail health checks. If you enabled load balancing, you can enable ELB health checks in addition to the EC2 health checks that are always enabled.

☑ EC2 ☐ ELB

Health check grace period

The amount of time until EC2 Auto Scaling performs the first health check on new instances after they are put into service.

300 seconds

Configure group size and scaling policies Info

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

Group size - *optional* Info

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

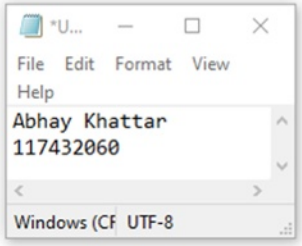
2

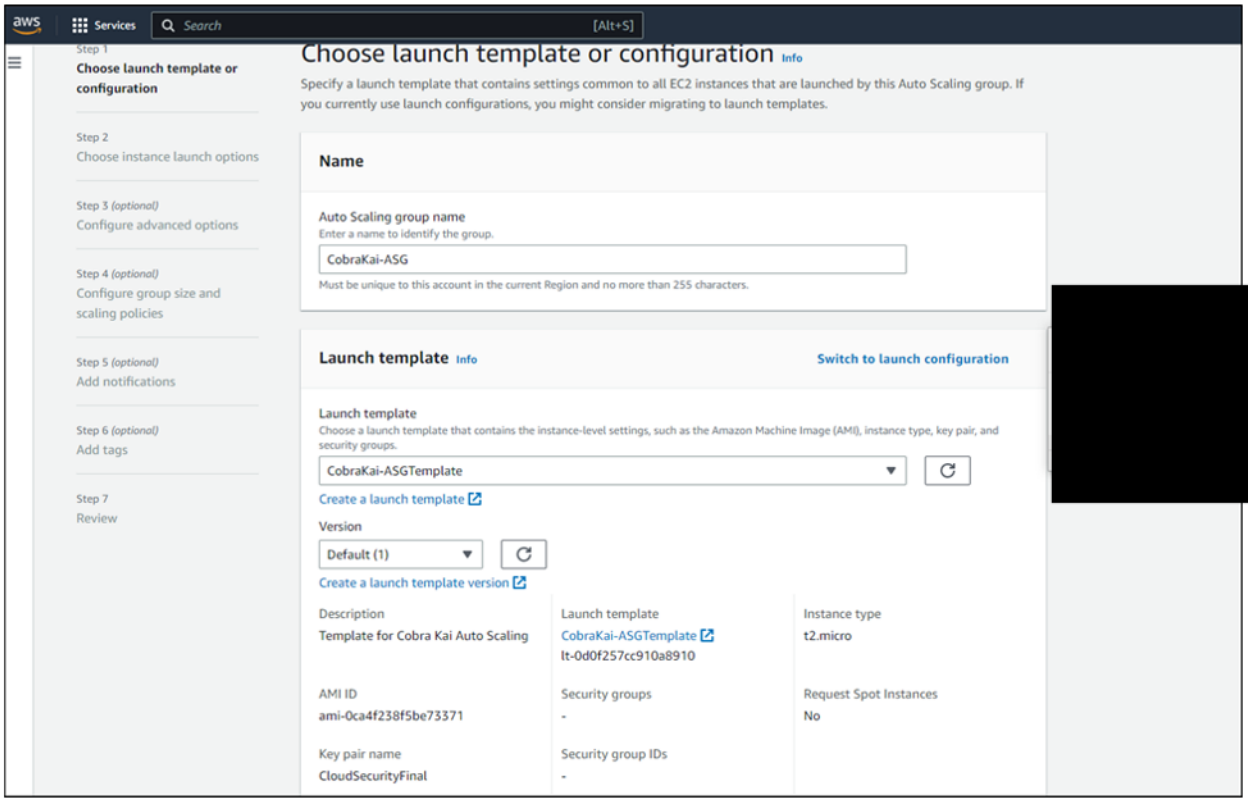
Minimum capacity

1

Maximum capacity

3

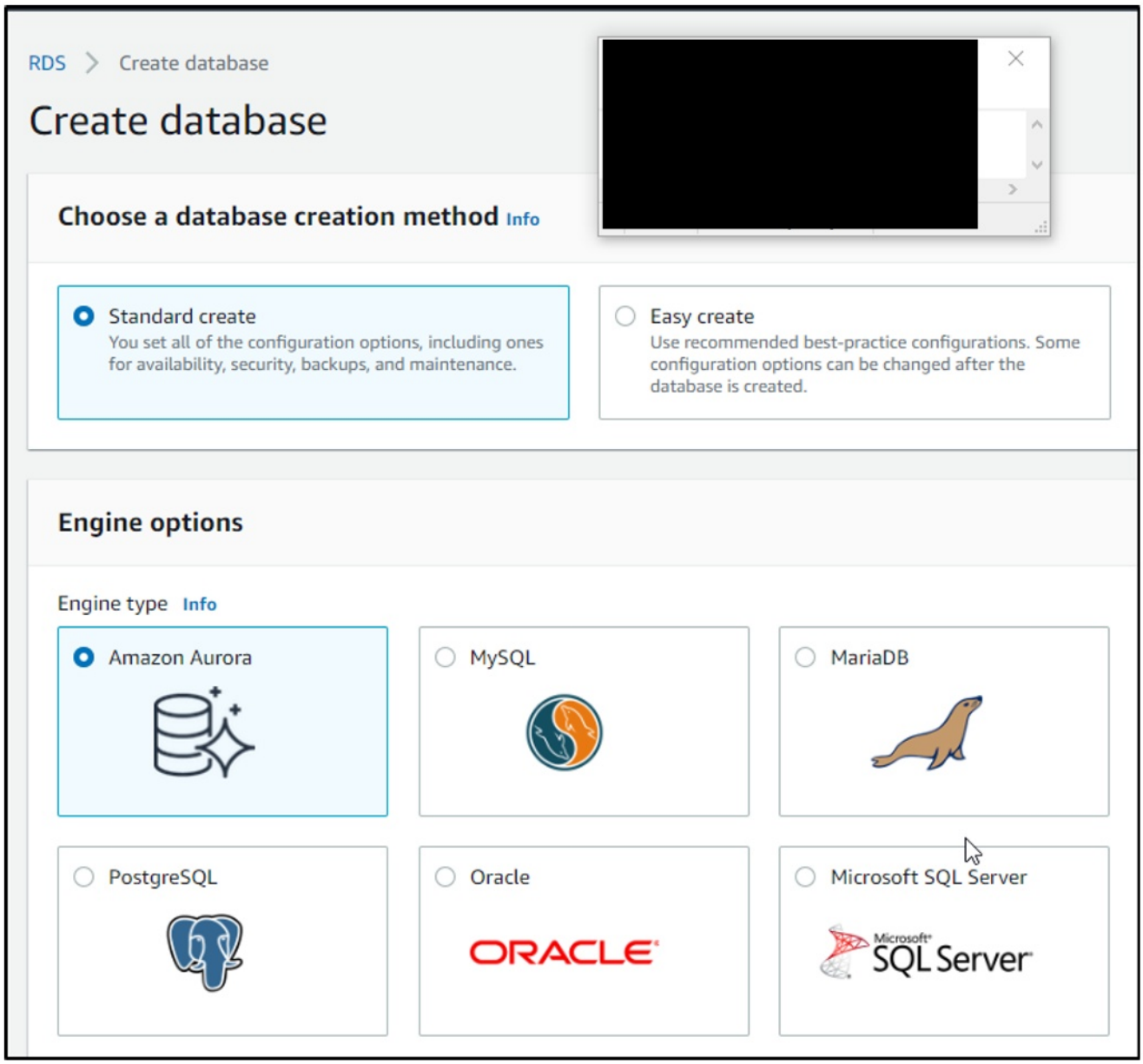


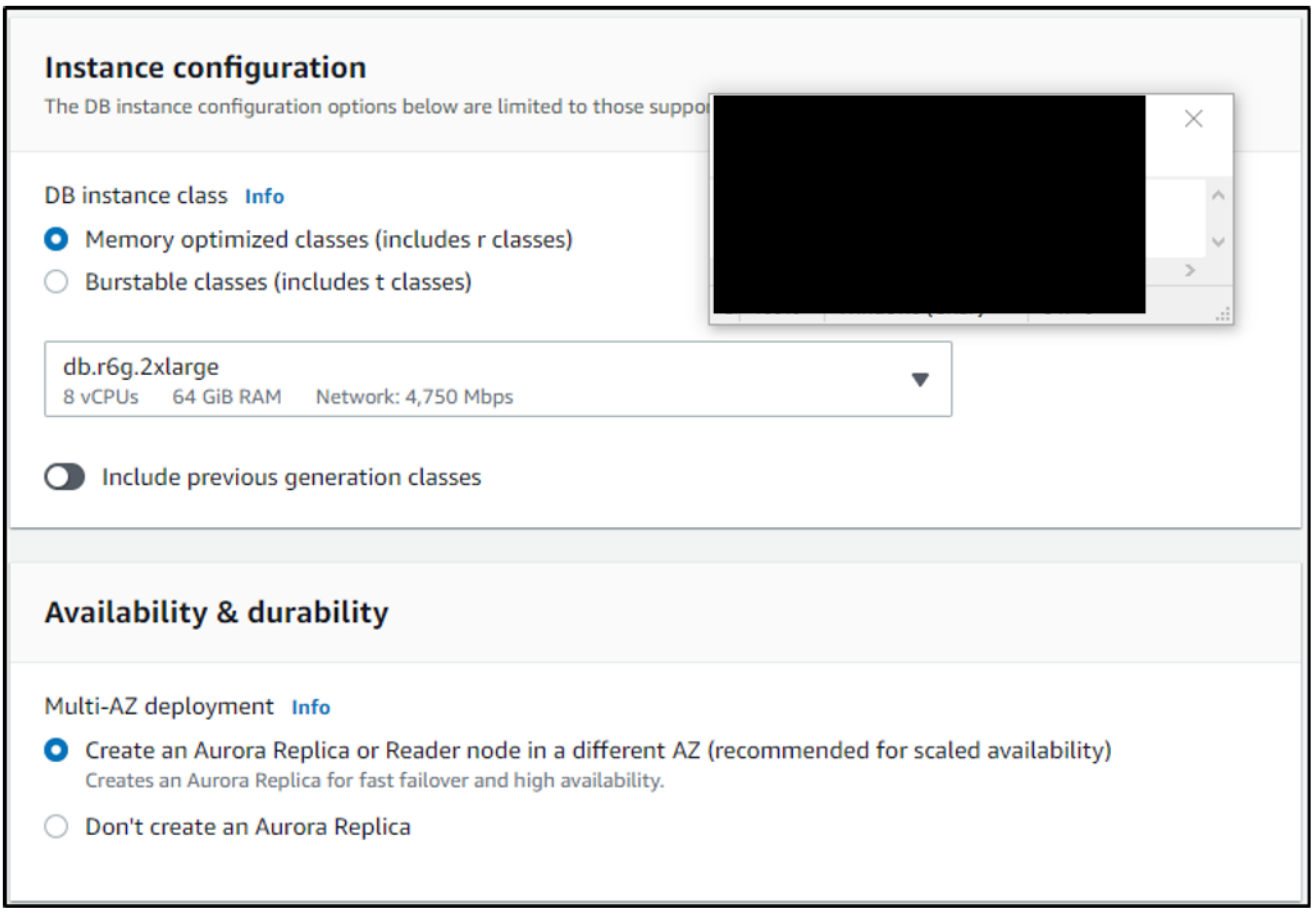


Setting up a Database

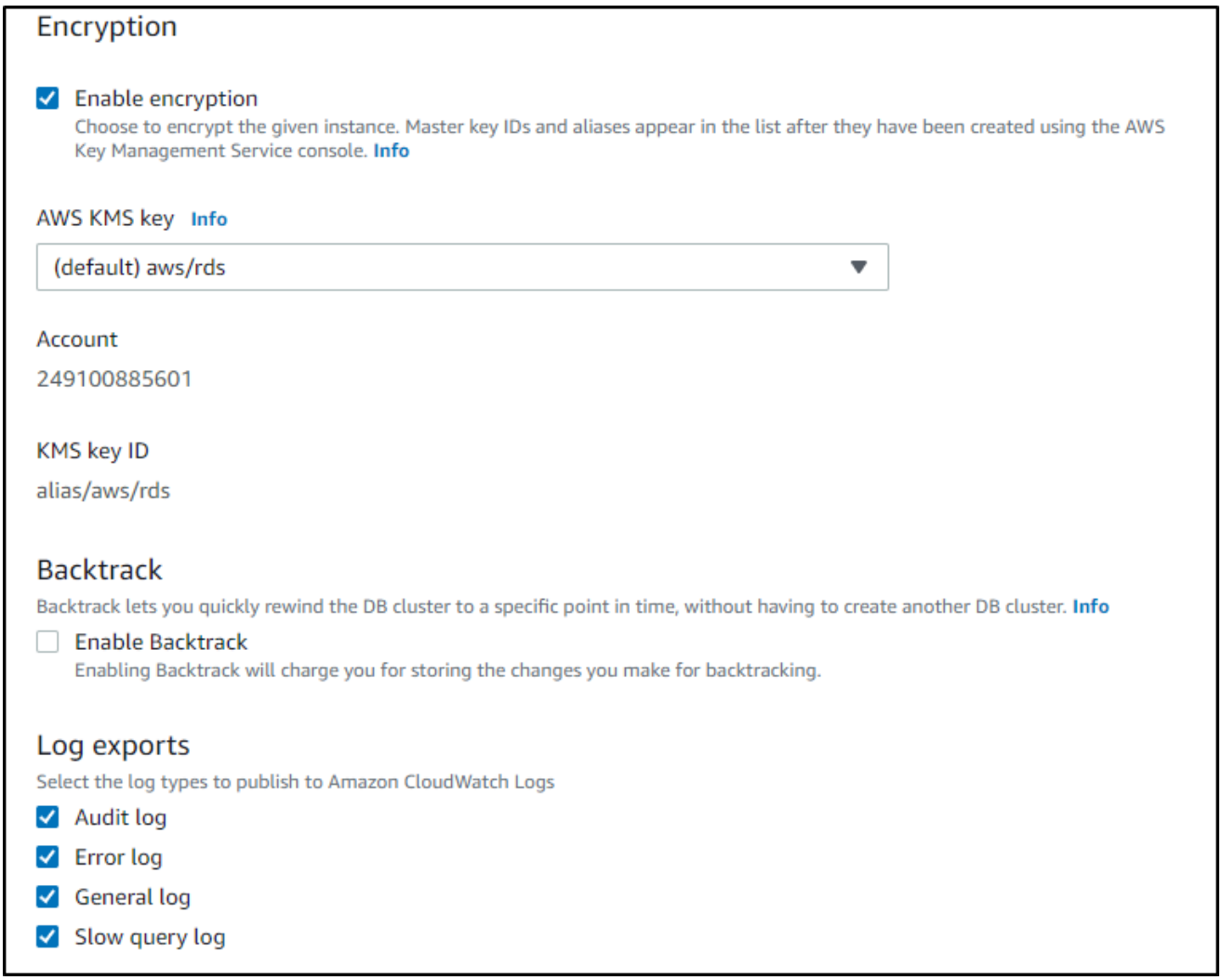
For the database, we are using the **Amazon Aurora**, which is a proprietary technology from AWS. Amazon Aurora[2] is "cloud optimized" and claims 5x performance improvement over MySQL on RDS, and over 3x performance of the Postgres on RDS The size for the database increases automatically in increments on 10GB. Although it costs more than RDS but is more efficient.

Note → Aurora is not present in the free tier, so I did not deploy it in my environment, but I have added screenshots for all the steps


RDS
Create database
Create database
Choose a database creation method Info
Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.
Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.
Engine options
Engine type Info
Amazon Aurora
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server



Aurora has the capability of creating replicas in different AZ to increase the availability of the database. We also enable encryption using the AWS KMS, to encrypt the data at-rest. Aurora has in-built capability of auto scaling and multi region availability can be enabled as a functionality



We are collecting all the logs and exporting it to the Amazon CloudWatch.

IAM

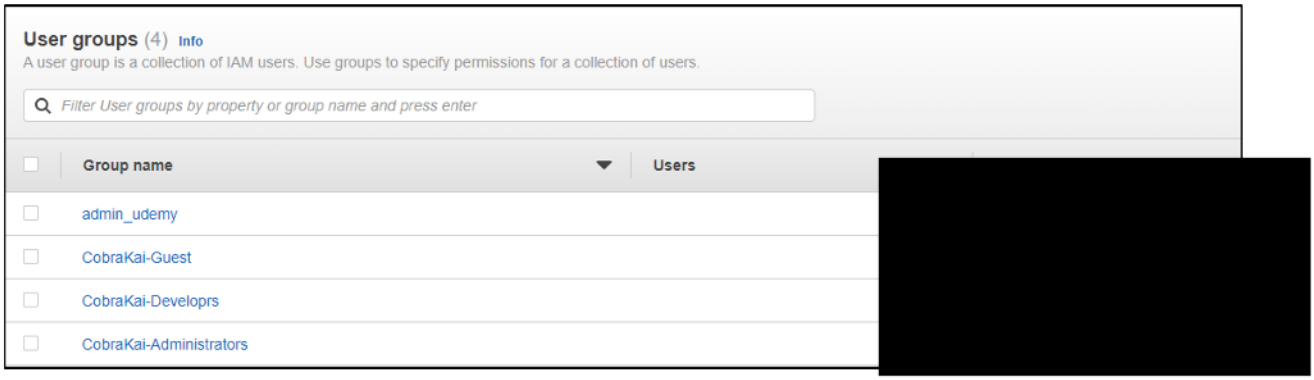
Note → It is better to use AWS Organization to manage the users, resources etc. for the organization. I was not able to implement it for my account as, it required buying a domain.

In accordance with the principles of Identity and access management, we set up different user groups which will help us give access to the users. We can add the users to their respective user groups, and give them the permissions accordingly. If we have multiple people who require the same access, instead of adding permissions individually, we can club them in a group and give all the users, the access that they need. We have the following three user groups -

1. CobraKai-Administrators → They maintain the entire application, and have access similar to that of the the root
2. CobraKai-Developrs → They develop and test new utilities, before adding them to the main application

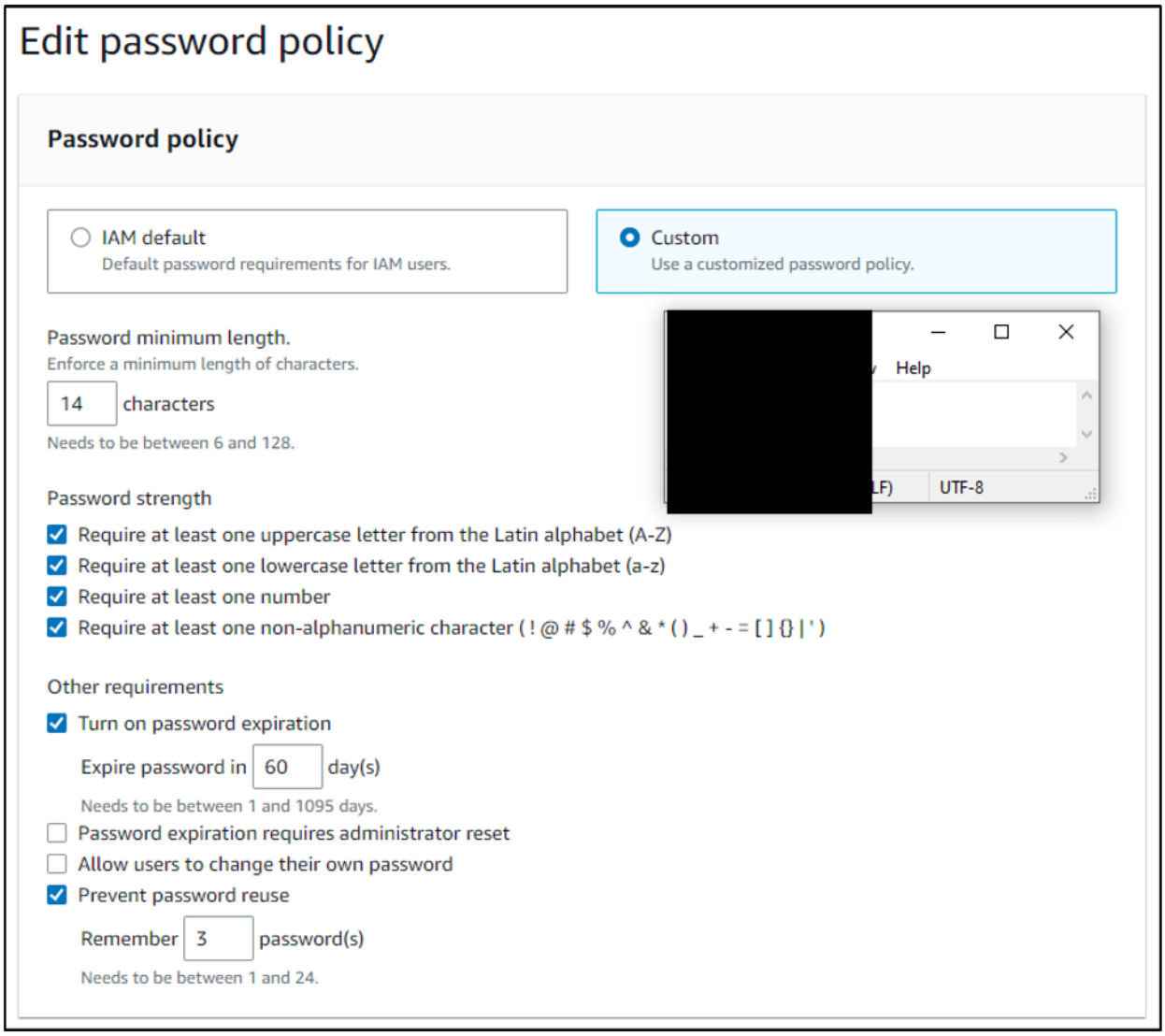
3. CobraKai-Guest

   These user groups have permissions in accordance with the **Principle of Least Privilege**, i.e. the minimum permissions which are required to complete the task

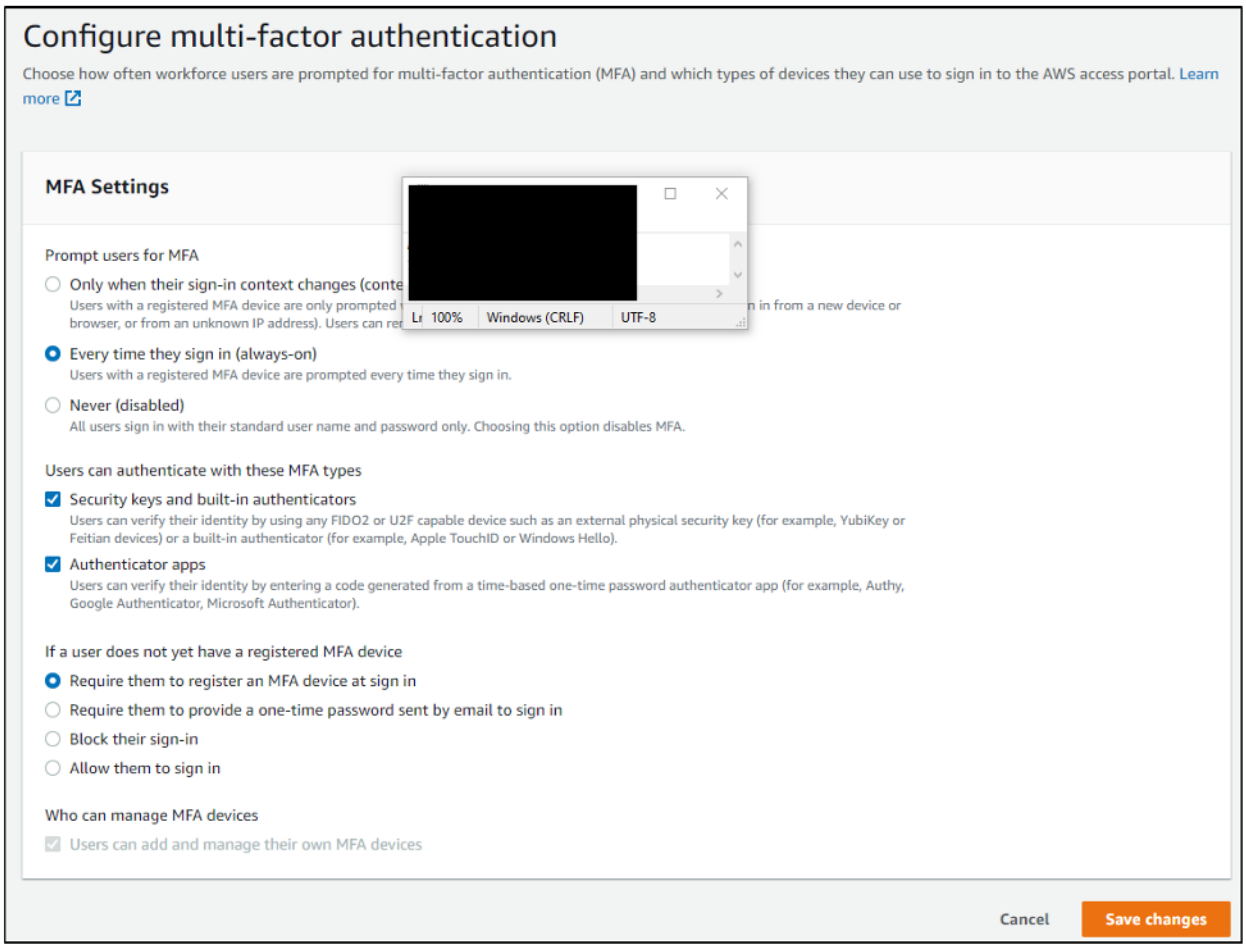


   Some other security measures which have been put in place are the following →

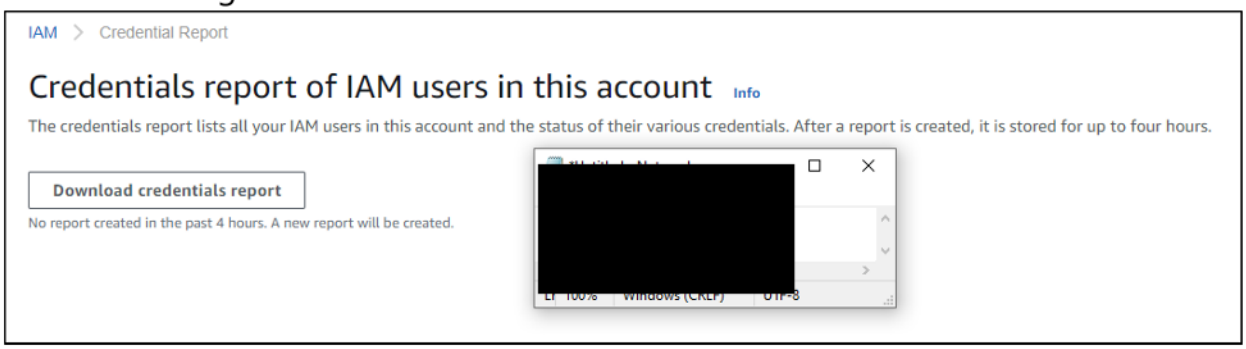
4. Password Policy → We have set up a custom password policy, which can help maintain the security standard of the organization



5. MultiFactor Authentication → We have enforced MultiFactor authentication. This helps by adding an extra layer of security when the user tries to login



6. User access keys for the root user is deactivated, and the root console access is also disabled
7. Credentials Reports can be a good way to check the status of credentials for the users in the organization



Creating a Backup Plan

We use AWS Backup to create a backup plan according to our needs. We make use of tags to identify the resources that we need to backup. We can customize the backup plan according to our need and according to the compliance requirement (if any).
In this scenario, we create a plan to backup all the resources in the account which will be retained for 1 year.

Create backup plan Info

Start options

Backup plan options Info

- (●) **Start with a template** — Create a Backup plan based on a template provided by AWS Backup.
- ( ) **Build a new plan** — Configure a new Backup plan from scratch.
- ( ) **Define a plan using JSON** — Modify the JSON expression of an existing backup plan or create a new expression.

Templates

Choose a template plan with existing rules.

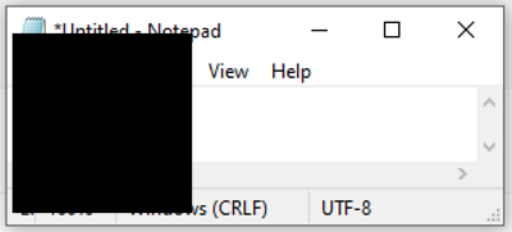
Daily-Monthly-1yr-Retention

Backup plan name

CobraKai

Backup plan name is case sensitive. Must contain from 1 to 50 alphanumeric or '-_.' characters.

Edit Backup rule



Backup rule configuration Info

Backup rule name

DailyBackups

Backup rule name is case sensitive. Must contain from 1 to 50 alphanumeric or '-_.' characters.

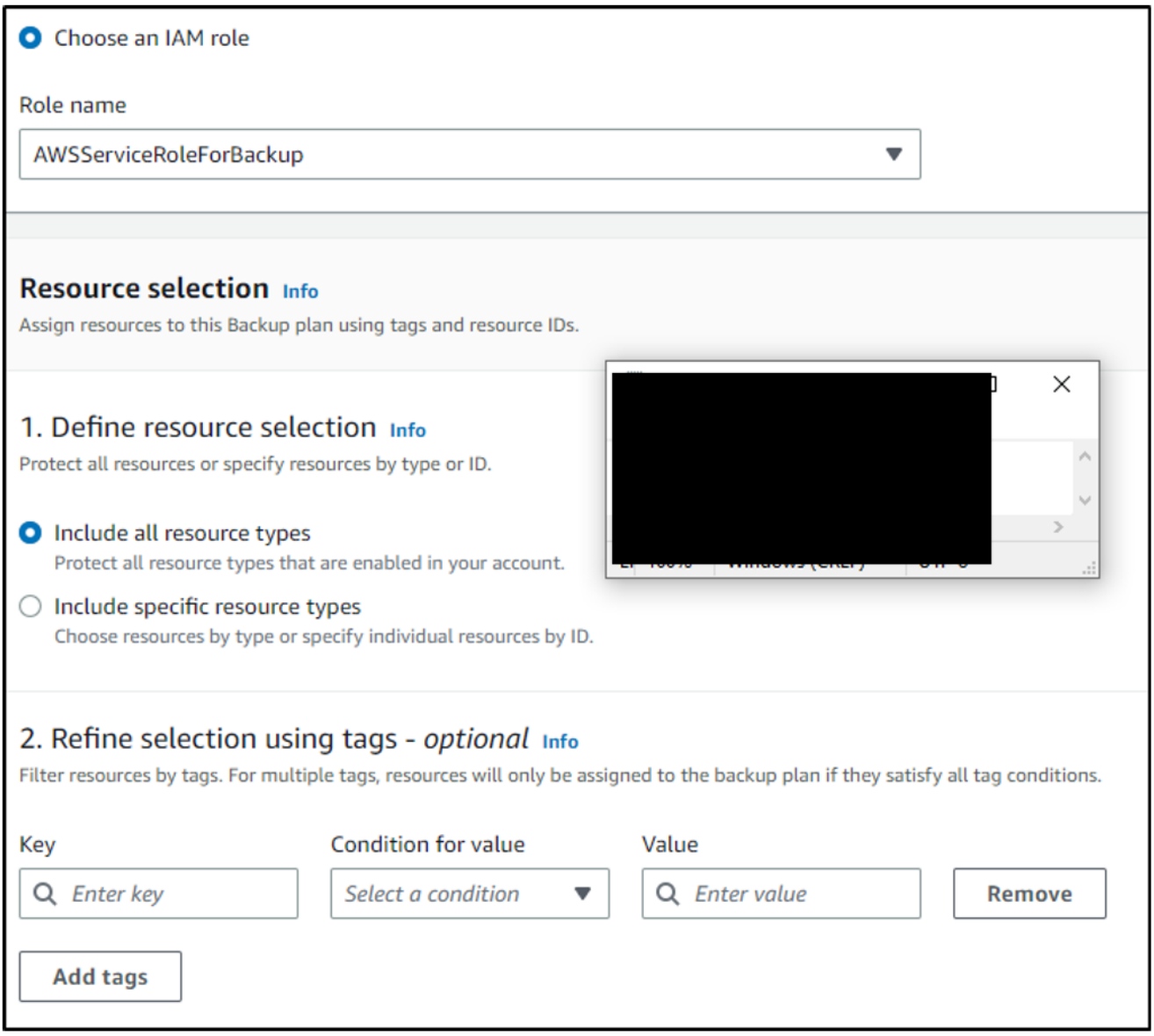
Backup vault Info

Default

Create new Backup vault

Backup frequency Info

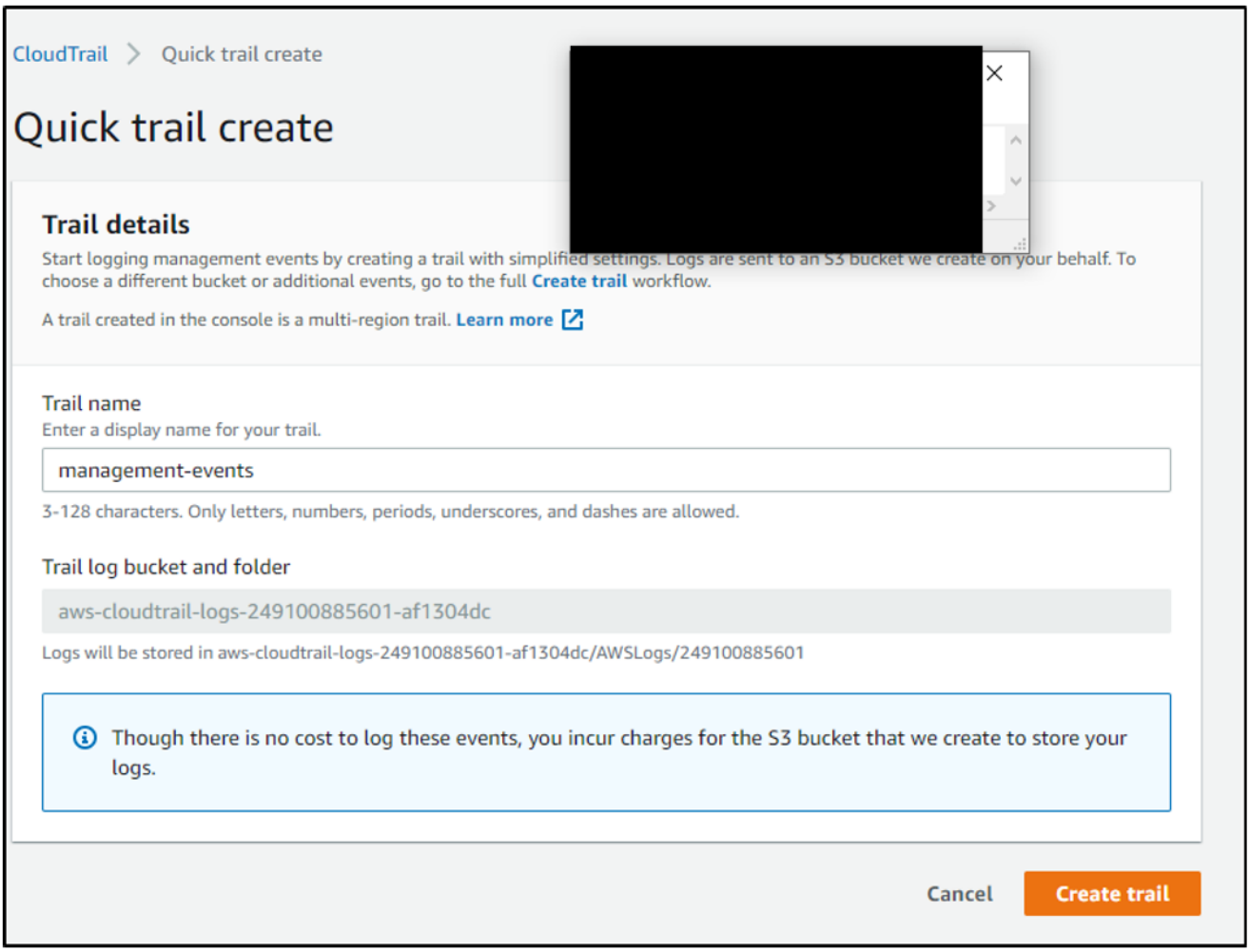
Daily

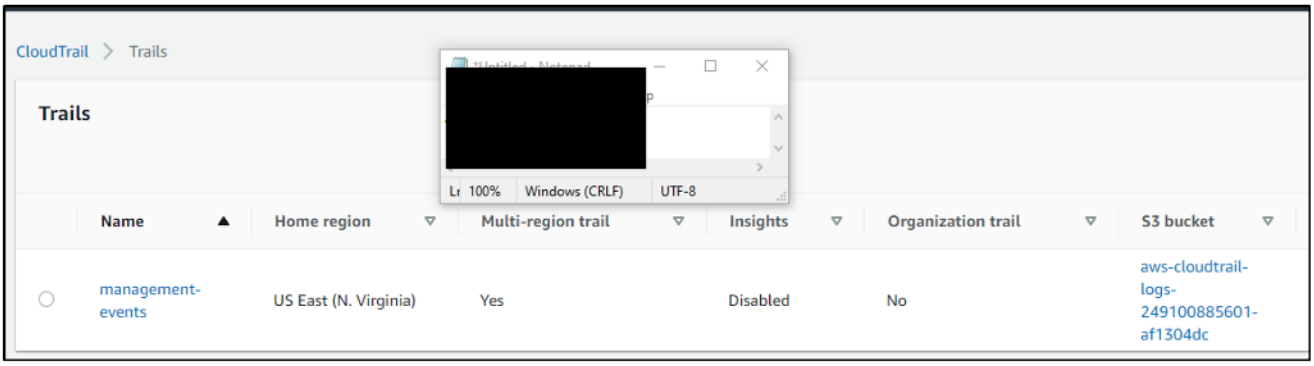


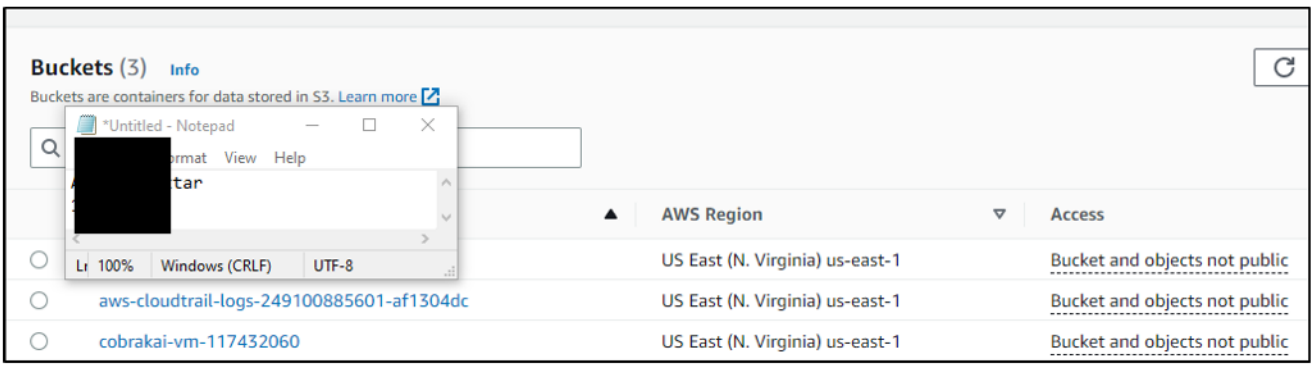
Other Services

CloudTrail

CloudTrail can be used to provide governance, compliance and audit for the AWS account. CloudTrail is enabled by default, and we have all the logs saved to an S3 bucket. We cannot edit or change the logs, and hence helps with governance and compliance during the audits. These logs also play a crucial role during the investigation of an incident.



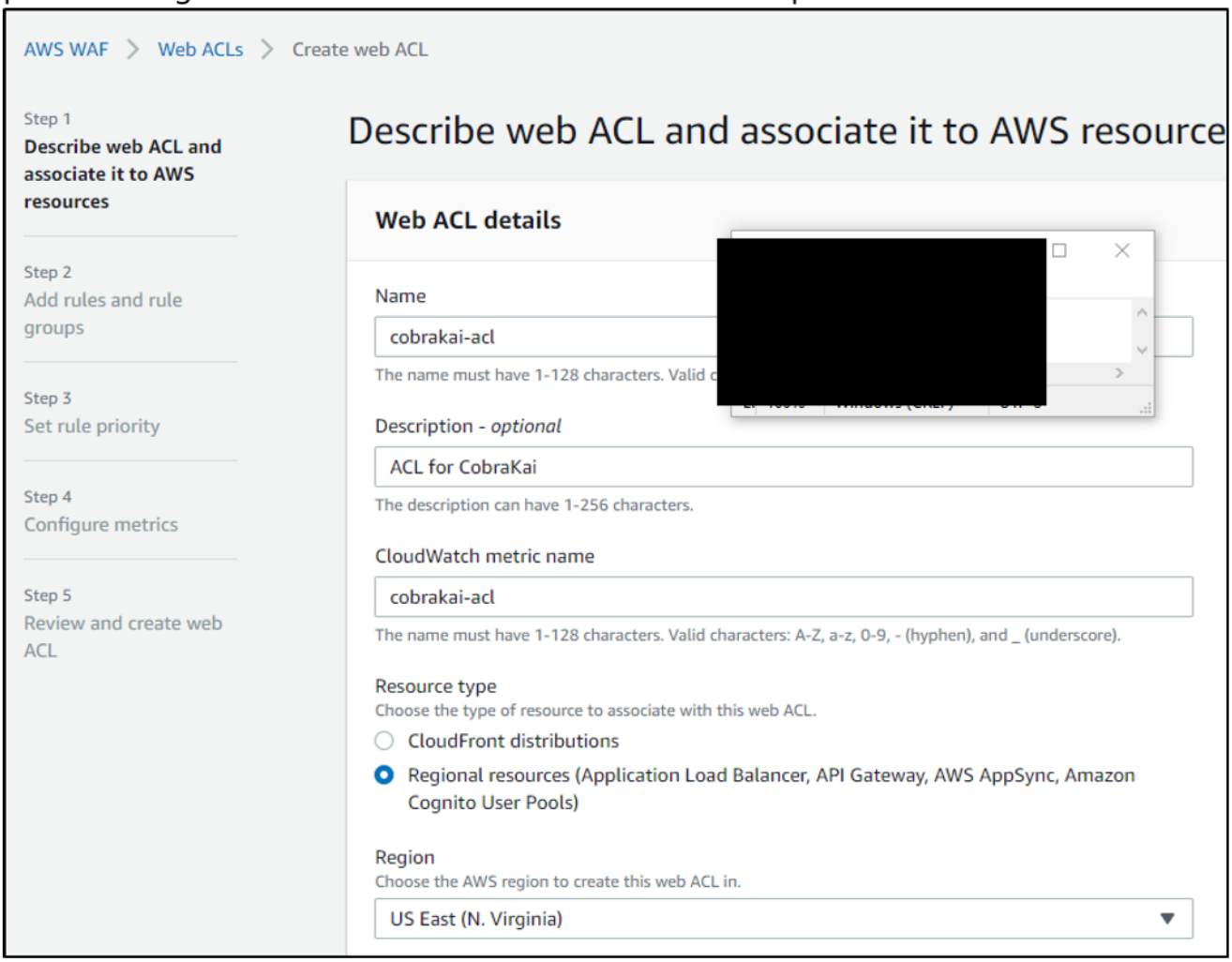


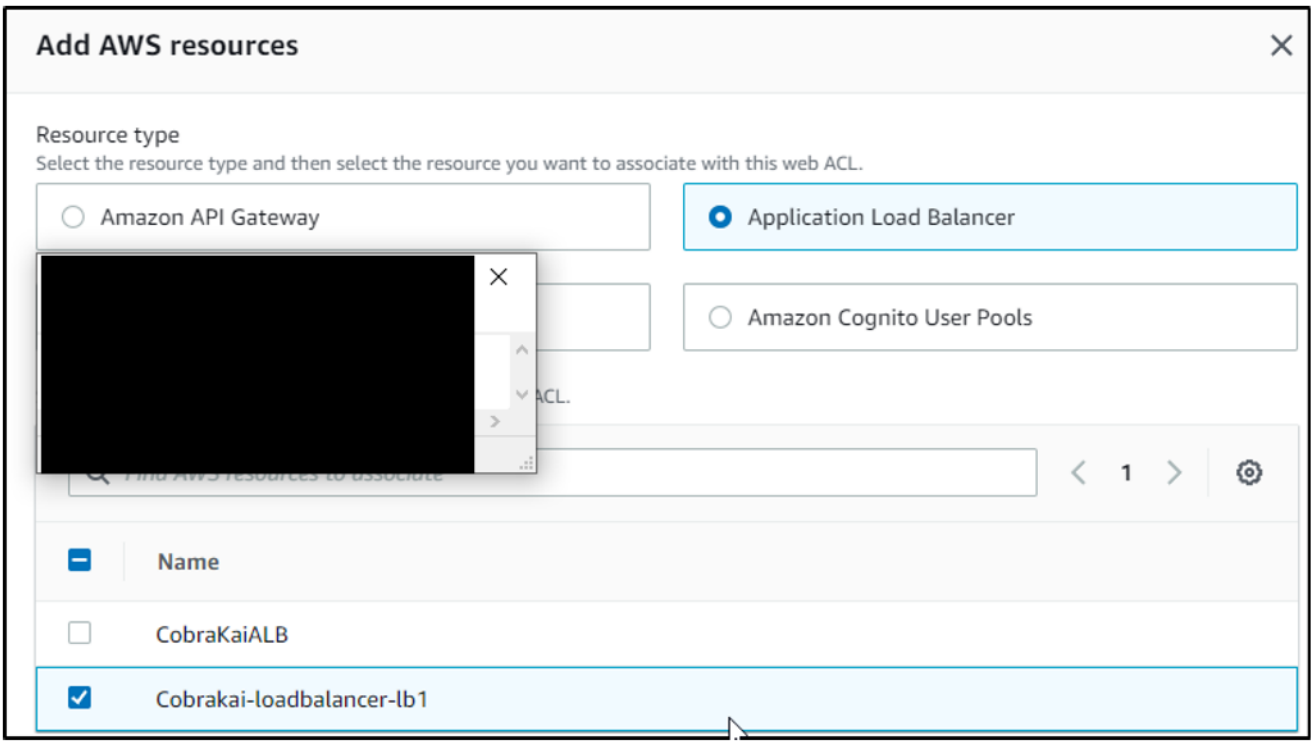


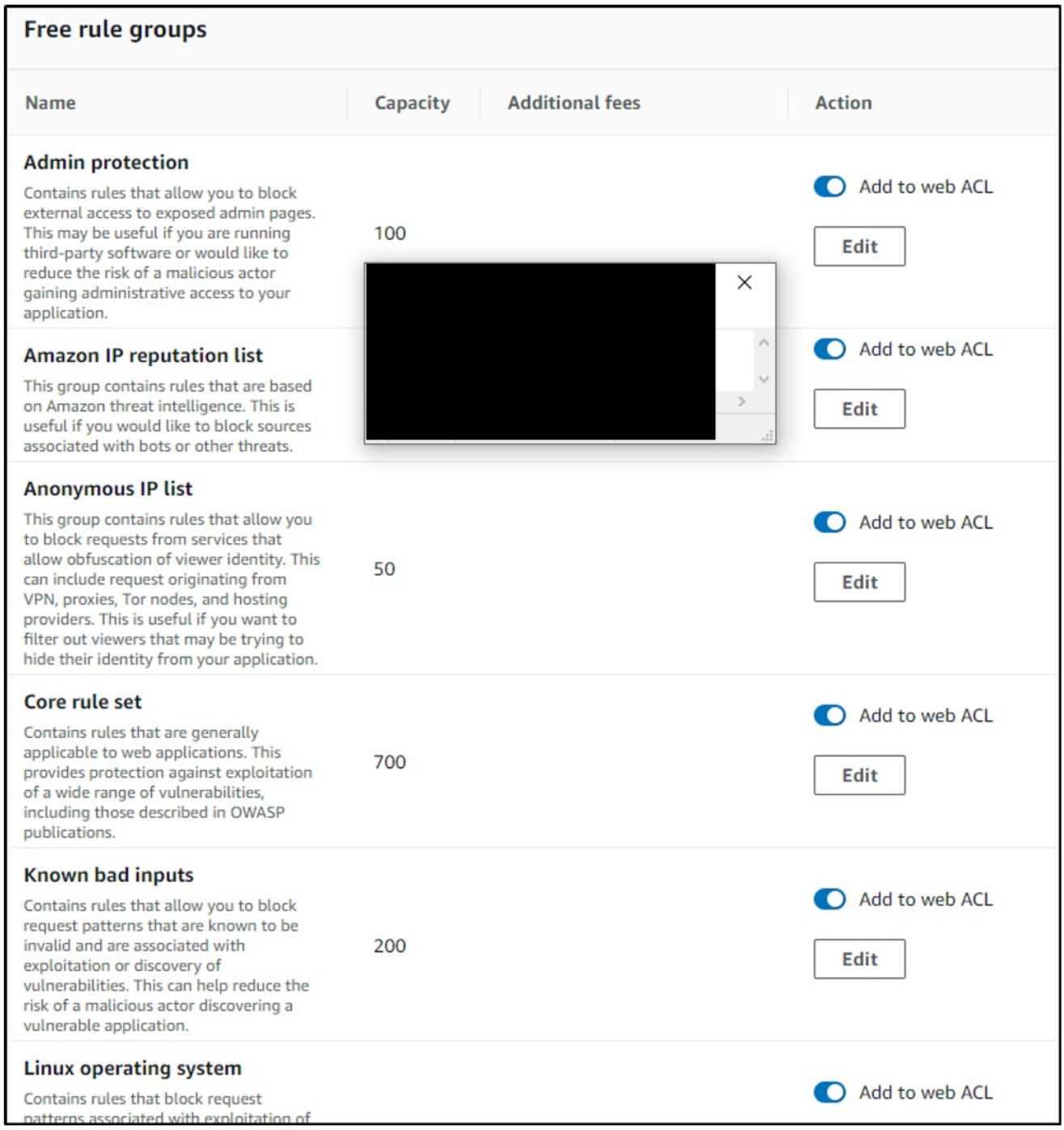
Web Application Firewall and Amazon Shield

The standard AWS Shield is enabled by default and protects againt DDOS attacks on the website and application.
The AWS WAF protects web application from common web exploits on layer 7 (HTTP). We enable the WAF on the Application Load Balancer because ALB is internet facing.

We enabled the free services which include rules such as admin protection, or protection against known bad IP address or malicious inputs




Add AWS resources
Resource type
Select the resource type and then select the resource you want to associate with this web ACL.
Amazon API Gateway
Application Load Balancer
Amazon Cognito User Pools
ACL.
1
Name
CobraKaiALB
Cobrakai-loadbalancer-lb1



CloudFront

As cobrakai would be required to supply videos on demand, we can use the AWS CloudFront which is the content delivery functionality of AWS. “CloudFront is a web service which speeds up the distribution of the web content to the users. CloudFront helps with an increased reliability and availability as the copies of the files and already cached in multiple edge locations around the world, and hence it is served instantly.”[3] We can add the SSL certificates for secure connections, add the WAF to cloudfront to protect it from common web exploits.

CloudFront > Distributions > create

Create distribution

Origin

Origin domain
Choose an AWS origin, or enter your origin's domain name.

cobrakai-vm-117432060.s3.us-east-1.amazonaws.com

Origin path - *optional* Info
Enter a URL path to append to the origin domain name for origin requests.

Enter the origin path

Name
Enter a name for this origin.

cobrakai-vm-117432060.s3.us-east-1.amazonaws.com

Origin access Info

- (•) Public
  Bucket must allow public access.
- ( ) Origin access control settings (recommended)
  Bucket can restrict access to only CloudFront.
- ( ) Legacy access identities
  Use a CloudFront origin access identity (OAI) to access the S3 bucket.

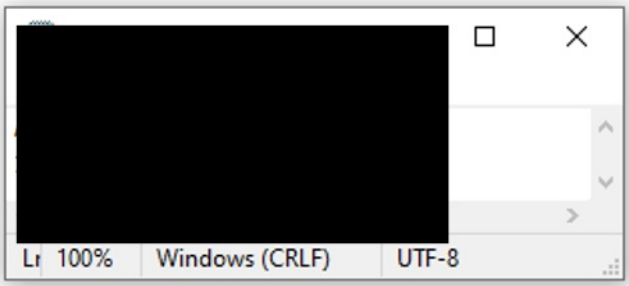
Compress objects automatically Info

- ( ) No
- (•) Yes

Viewer

Viewer protocol policy

- ( ) HTTP and HTTPS
- (•) Redirect HTTP to HTTPS
- ( ) HTTPS only

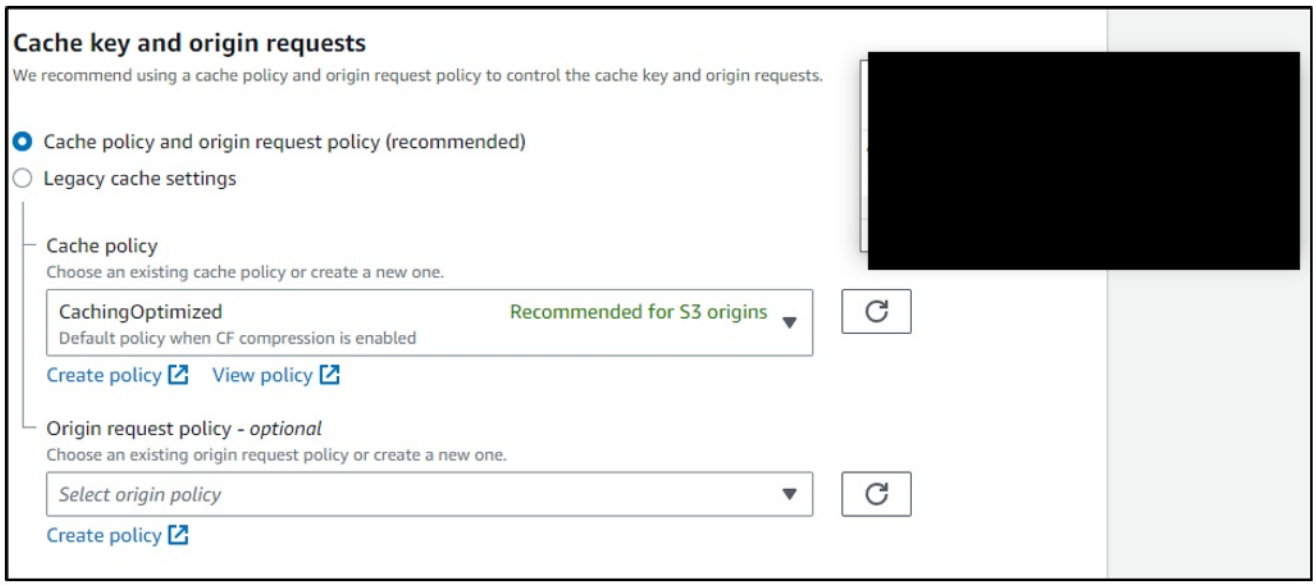


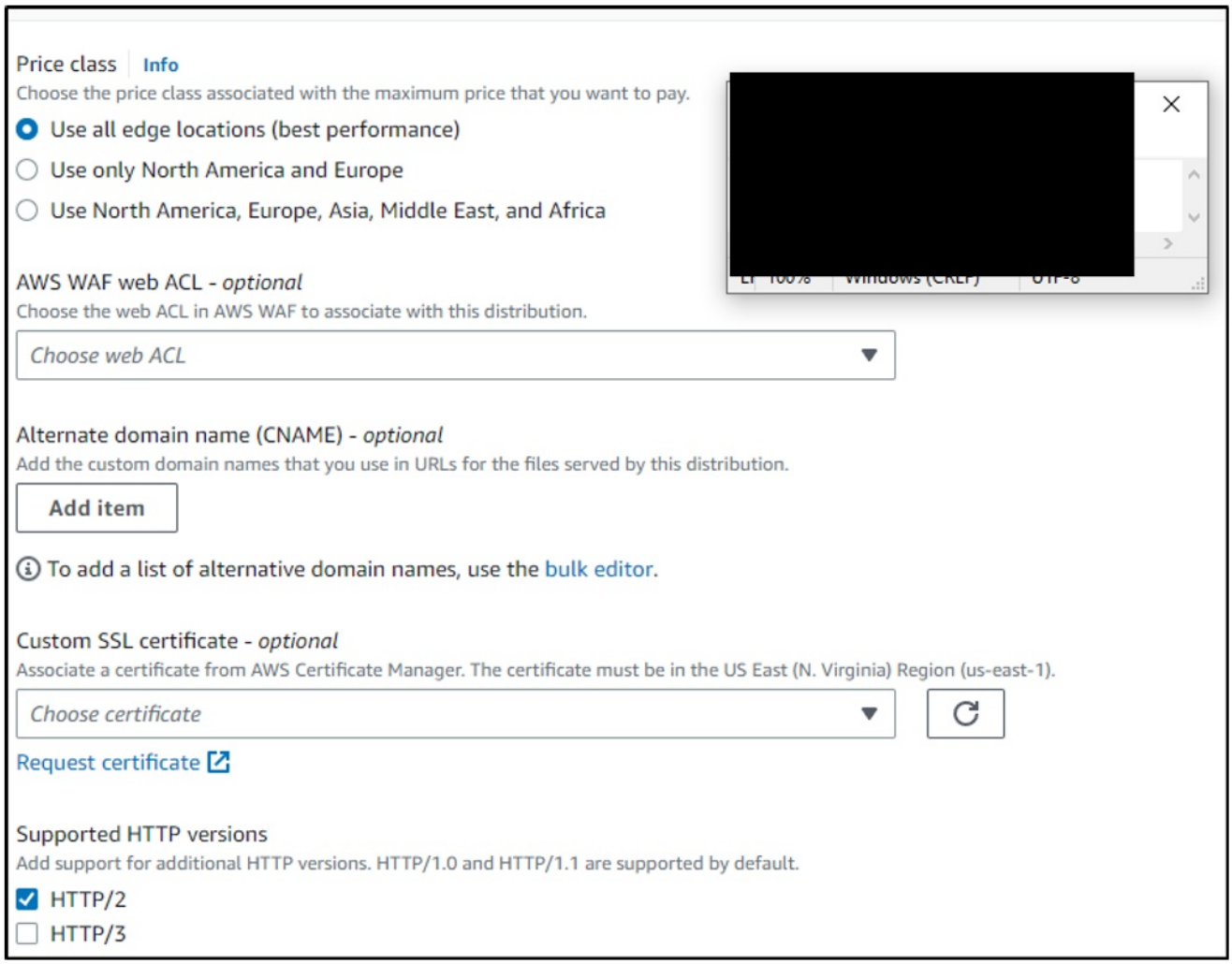
Allowed HTTP methods

- (•) GET, HEAD
- ( ) GET, HEAD, OPTIONS
- ( ) GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Restrict viewer access
If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.

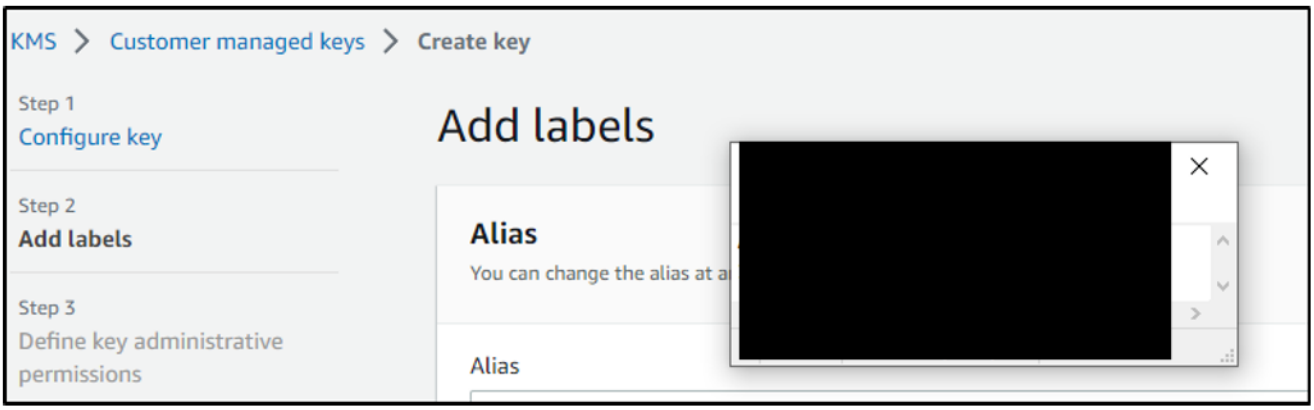
- (•) No
- ( ) Yes



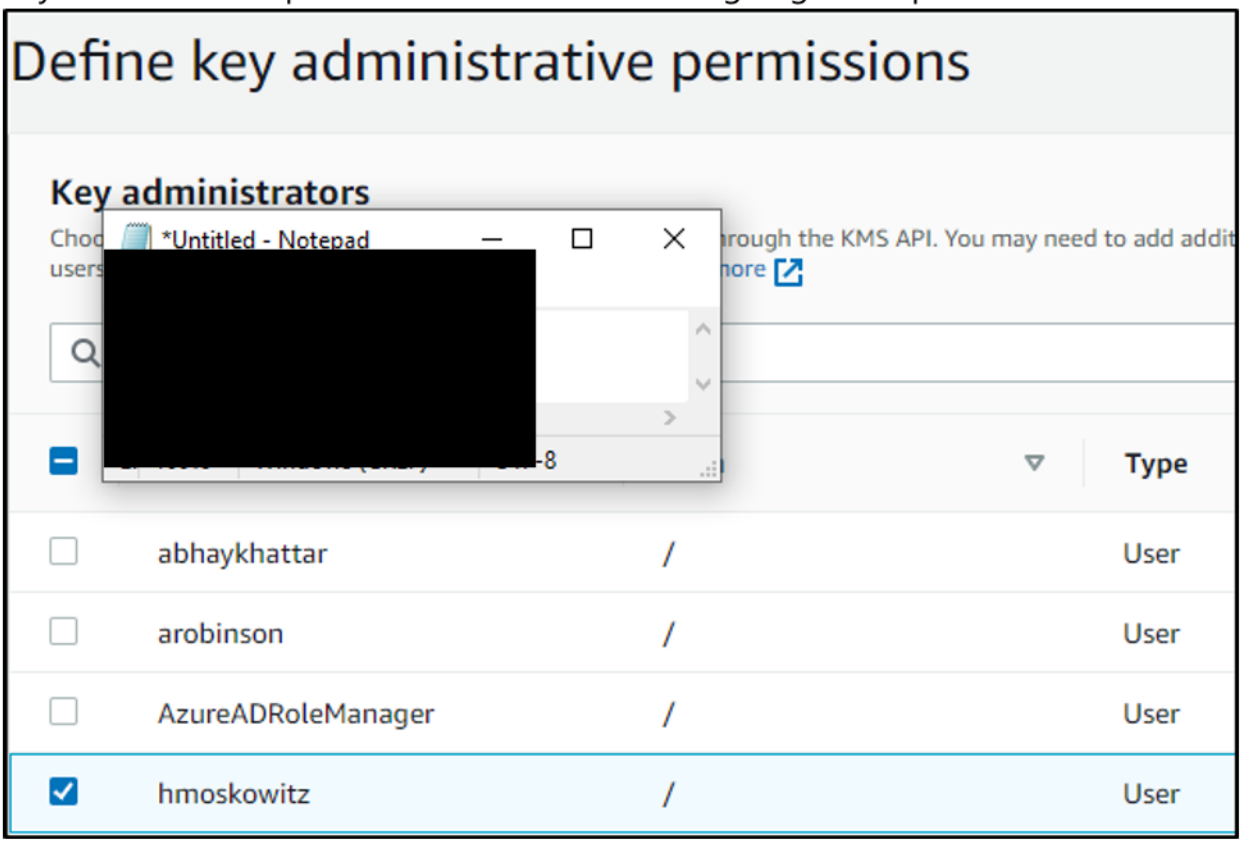


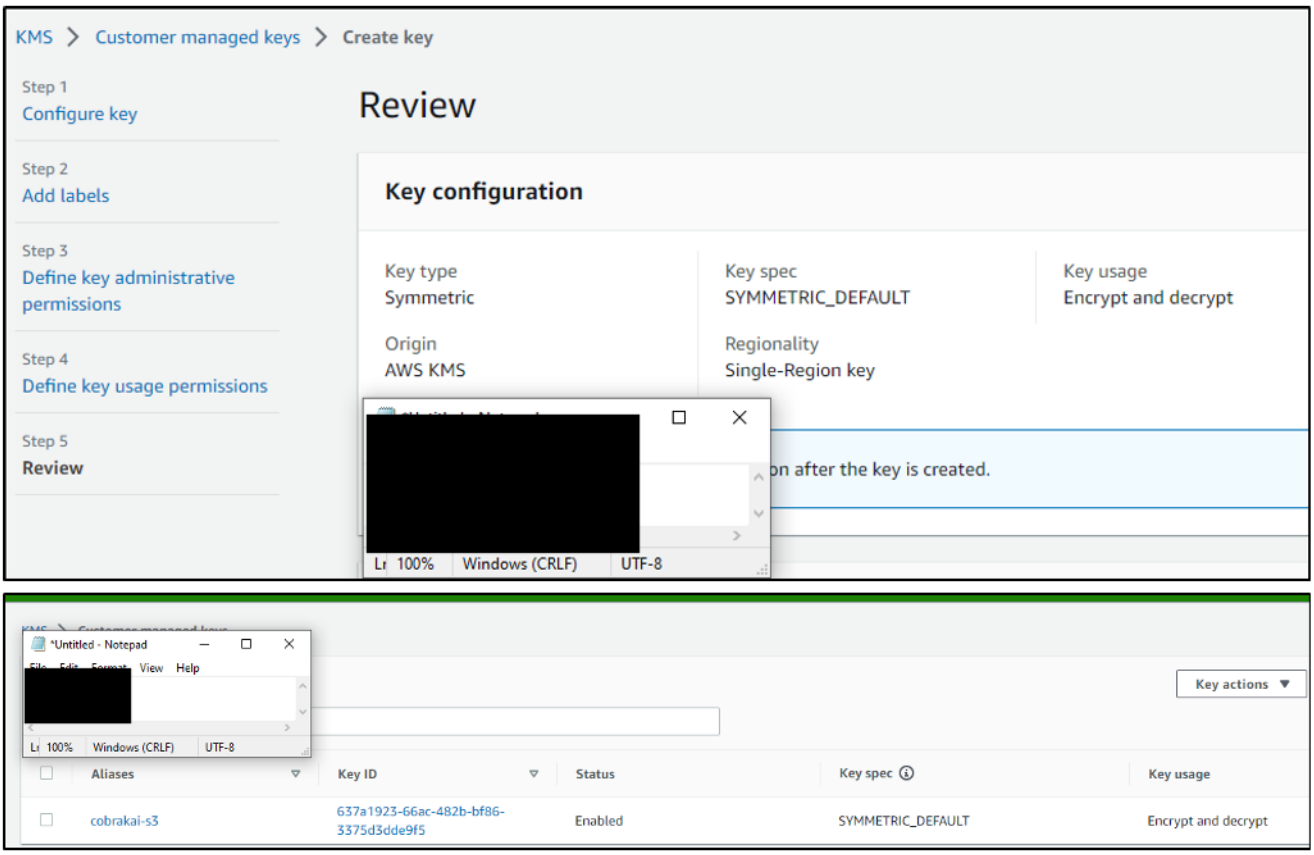
AWS Key Management Service (KMS)

KMS is an AWS product which helps with the management of the encryption keys. Using these keys we are able to encrypt the backups, S3, the databases, etc.



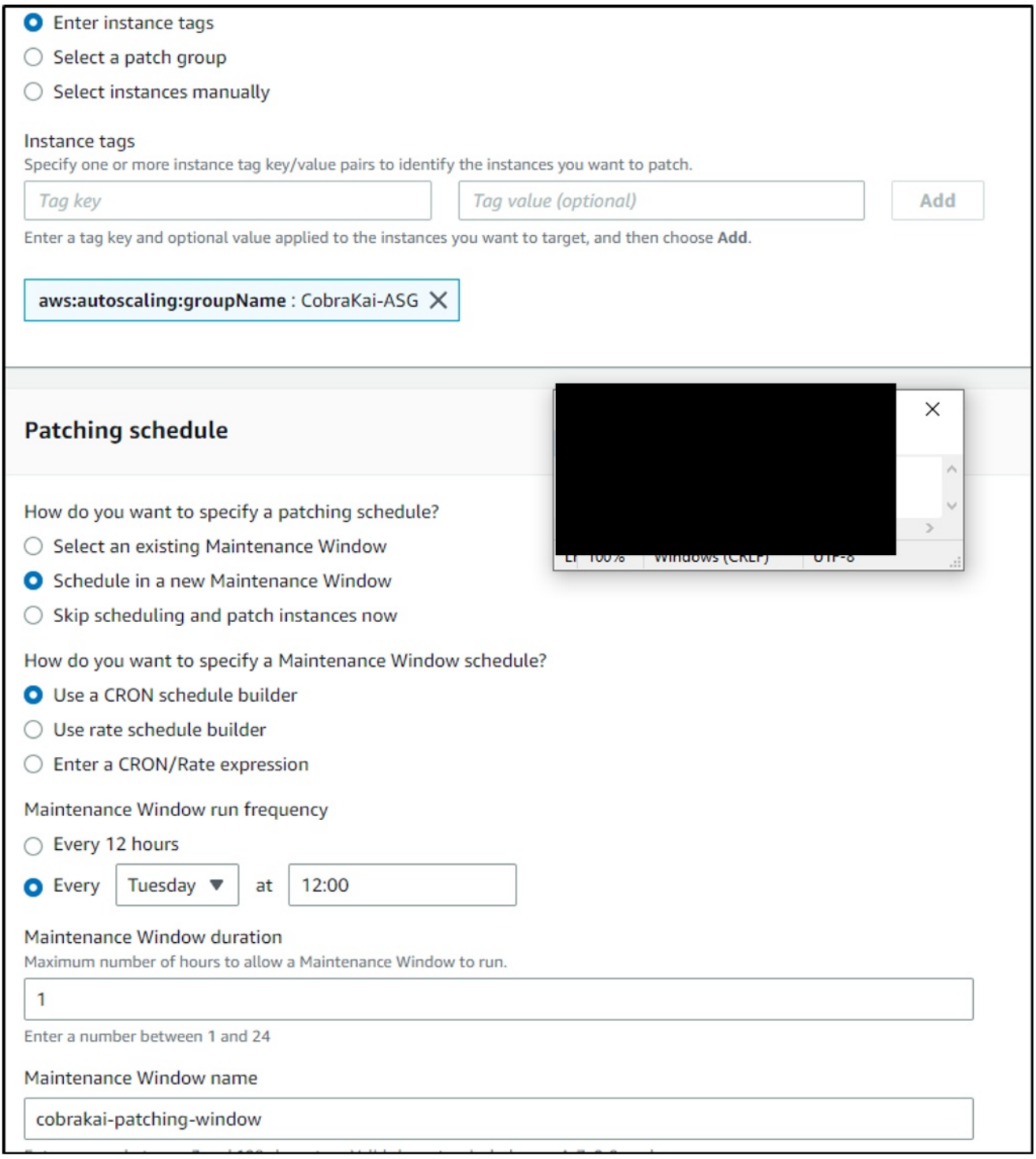
While setting KMS for the first time, we will have to identify the user who would have key administrative permissions. Here we are assigning those permissions to Hawk



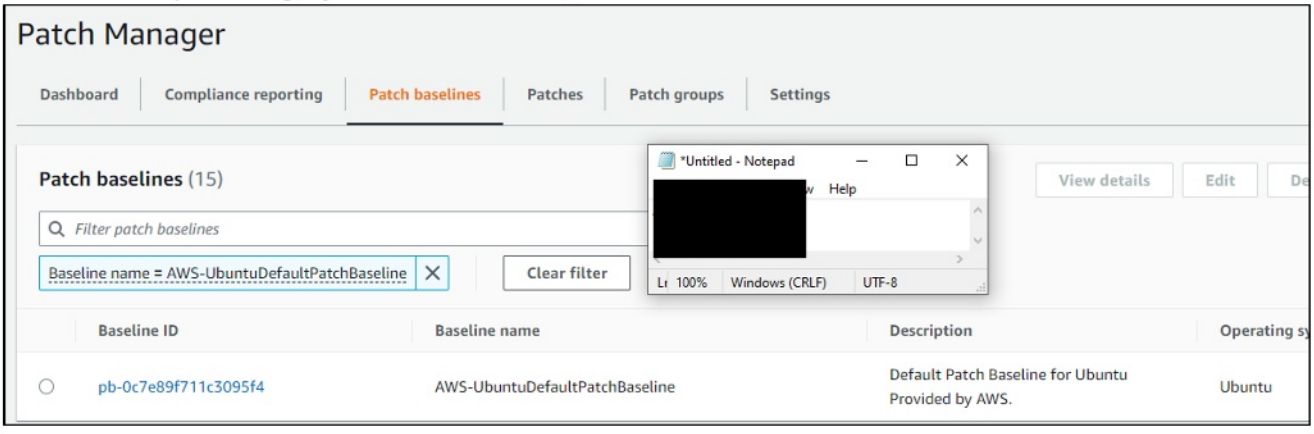


AWS System Manager

We use the AWS System Manager to create a Patching Schedule. It is essential to have a patching strategy so that we are able to keep the systems up-to-date with the latest patches and in compliance with the security standards.
For the purpose of patching, we identify the machines using tags. According to the current configuration, we are patching machines which are in an autoscaling group, and have the group name as `CobrKai-ASG` on every Tuesday at 12 noon.

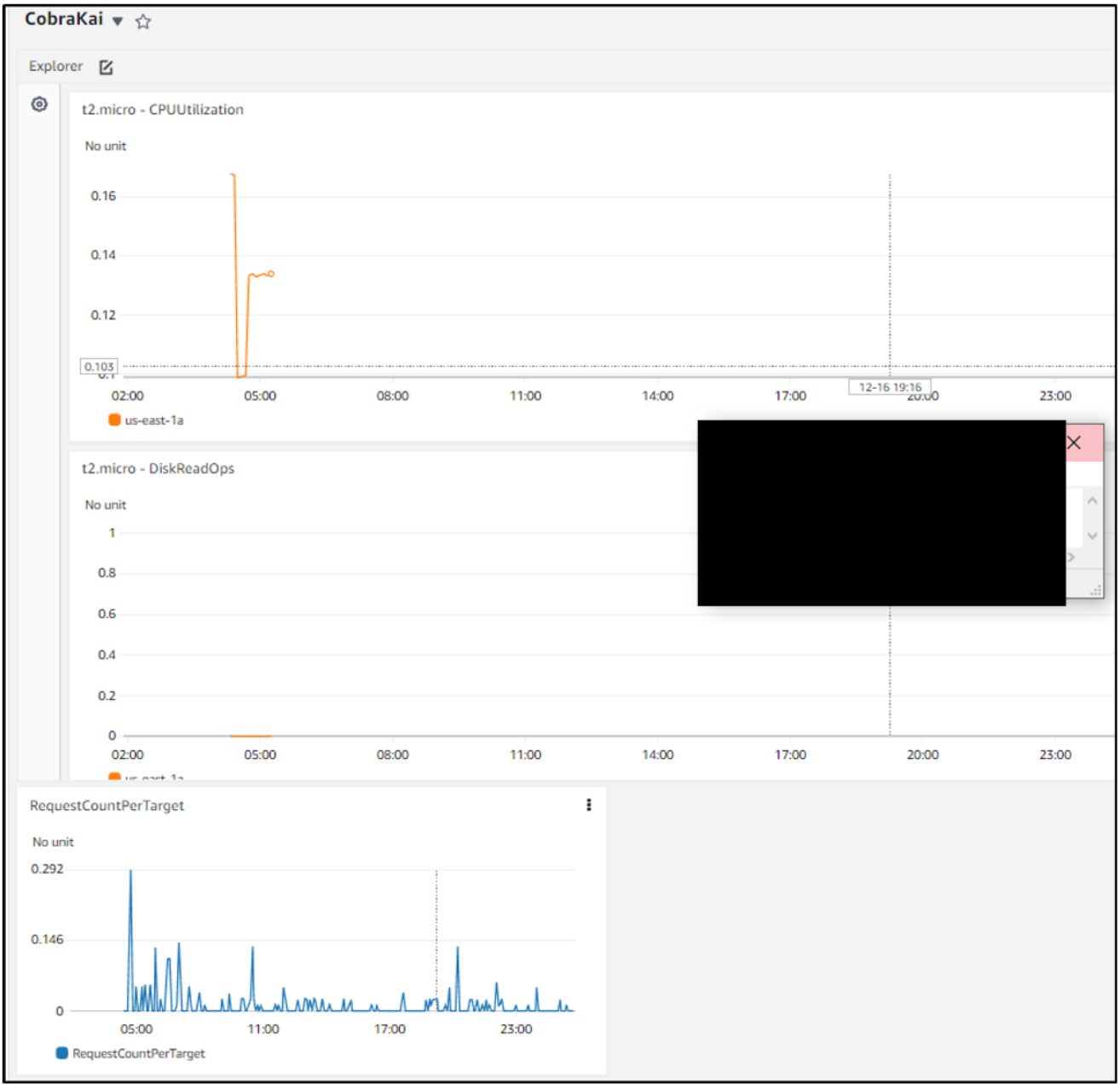


We set the baseline as the AWS-UbuntuDefaultPatchBaseline which is provided by AWS (our base operating system is Ubuntu)



CloudWatch

We can use CloudWatch to create dashboards and monitor all the resources in AWS. CloudWatch collects and visualizes real-time logs, metrics, and event data in automated dashboards to streamline your infrastructure and application maintenance. [4]

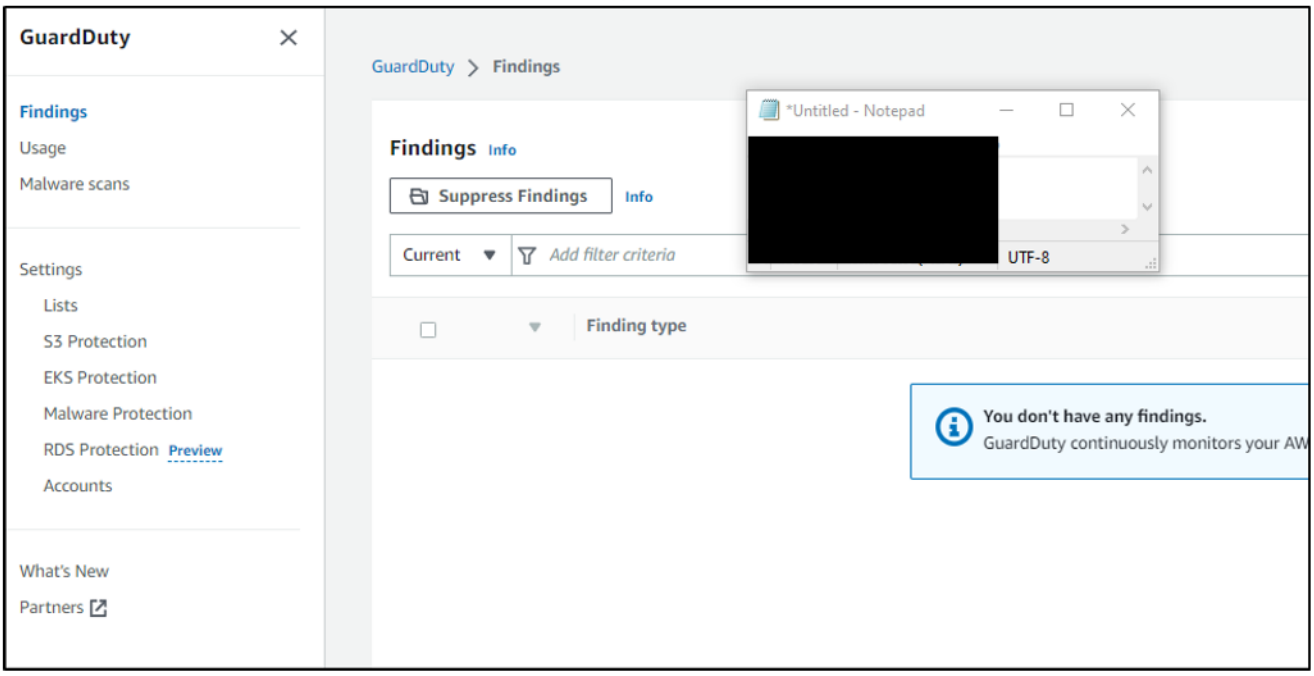


CloudWatch makes it easier to monitor the resources by using customizable dashboards, and alerts.

AWS Guard Duty

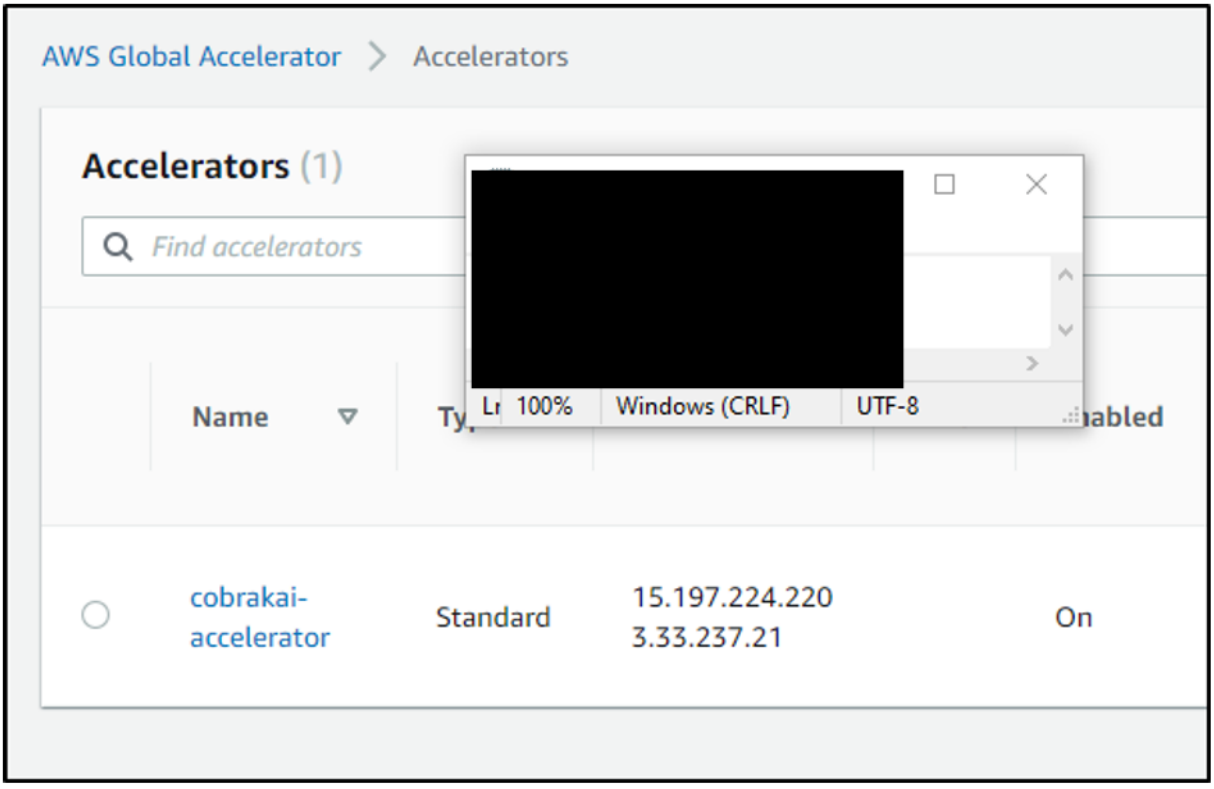
GuardDuty monitors the AWS resources (AWS accounts, instance, storage, database, etc.) for potential threats. GuardDuty uses Machine Learning algorithms for threat

detection and is also capable of threats by initiating automated response.[5]



Global Accelerator

Global Accelerator is used to increase the speed for upload and download of resources to and from the cloud. It uses the end points and amazon's own network for the transfer of data



PCI DSS Compliance

Security has to be integrated at every layer/step in order to be PCI DSS compliant. AWS Architect provides a list of the best practices to be used while configuring services in

AWS in order to be PCI DSS compliant. The following are steps which we have integrated as part of security in the architecture →

- Creation of **Virtual Private Cloud** ensures that the resources are isolated
- Using **Security Groups and Network Access Control List (NACL)** to create rules and allow/deny the traffic and access
- **Encryption of Data** has been at every step
- Enforcing strong Identity Access and Management
    - Enforcing a strong password policy, to mitigate password bruteforce and password reuse
    - Using a Role based approach, and implementing the **principle of least privilege**
    - Enforcing MultiFactor authentication
- Weekly patching for all the systems, so that the systems are up-to-date with the latest patches against vulnerabilities
- Using CloudWatch to monitor the resources
- Using CloudTrail to gather logs and event information which can be used for auditing and incident response
- Using AWS GuardDuty to identify and respond to potential threats

---

References

1. https://aws.amazon.com/premiumsupport/knowledge-center/ec2-export-vm-using-import-export/
2. https://aws.amazon.com/rds/aurora/
3. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html
4. https://aws.amazon.com/cloudwatch/
5. https://aws.amazon.com/guardduty/