Final Migration Report

A. JAMES CLARK
SCHOOL OF ENGINEERING

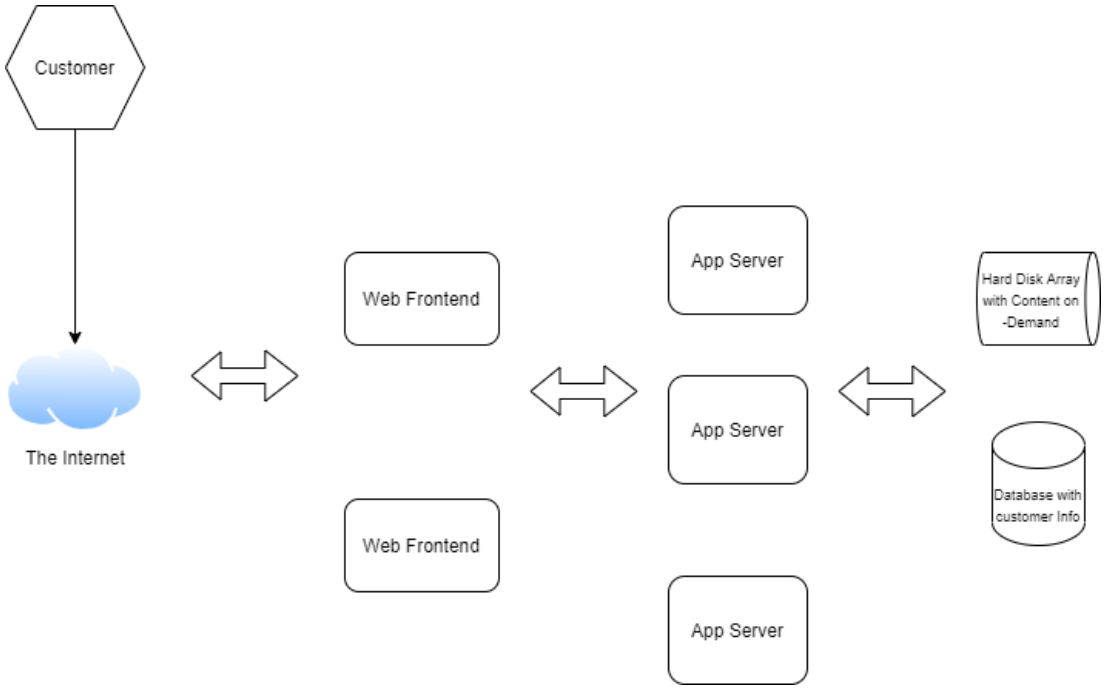
ENPM809J Cloud Security Fall 2020

Student Name: Shoumit Karnik
University ID: 116717496

EXECUTIVE SUMMARY

In this document the implementation plan, which will be adopted to redesign the Cobra Kai web streaming service to take advantage of the benefits of moving to the cloud, has been outlined. Some basic implementation steps for a rudimentary cloud structure and some technical examples of how the cloud security posture would look like have been documented below.

CURRENT SYSTEM ARCHITECTURE



PROPOSED SOLUTION ROADMAP

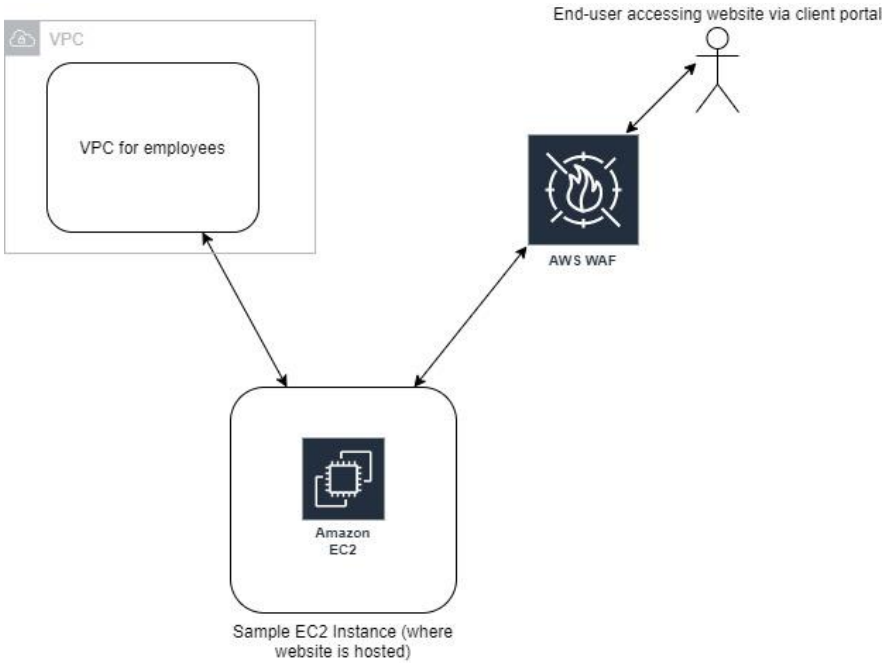
Some solution details have been provided subsequently. They include:

<u>Sample Security Group configurations:</u>

This part includes security group configurations, along with creating a VPC (Virtual Private Cloud) subnet for the employees.

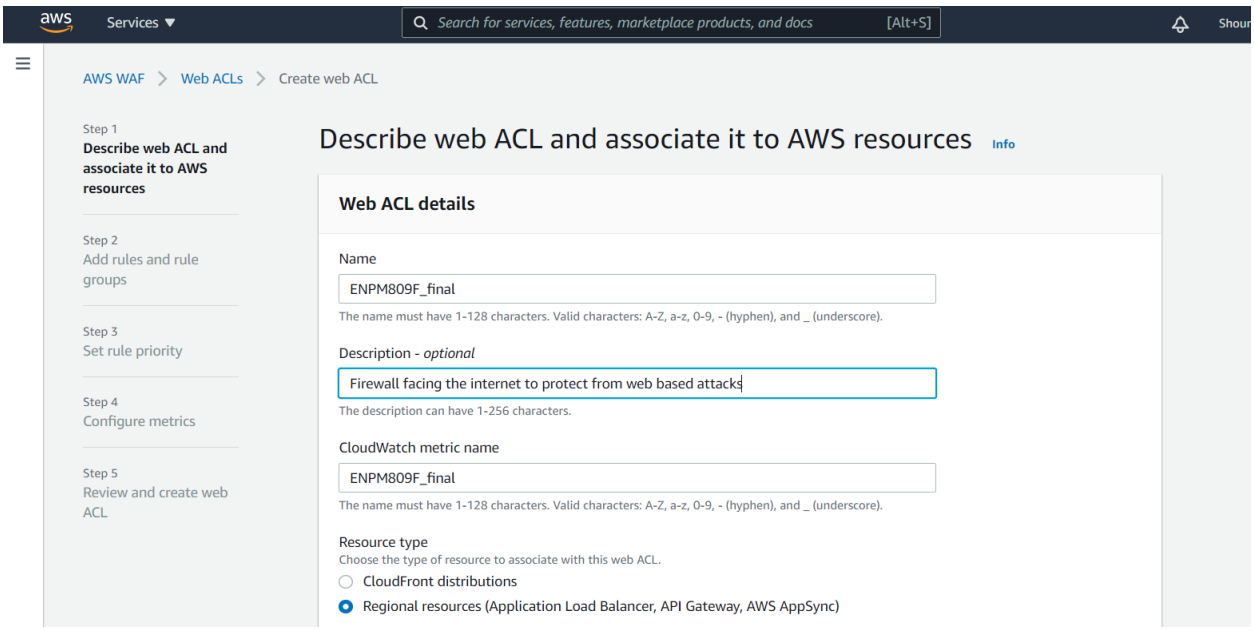
The EC2 instance where the website will be migrated to will also be protected from external attacks using a Web Application Firewall (AWS WAF). This specifically takes care of OWASP top 10 web attacks like XSS(Cross Site Scripting), XSRF(Cross-Site Request Forgery) etc. (Because the website manages credit card information and has been known to be vulnerable to SQL injection)

The EC2 instance will also have iptables (internal firewall) running on it, to provide an internal layer of security for the instance.

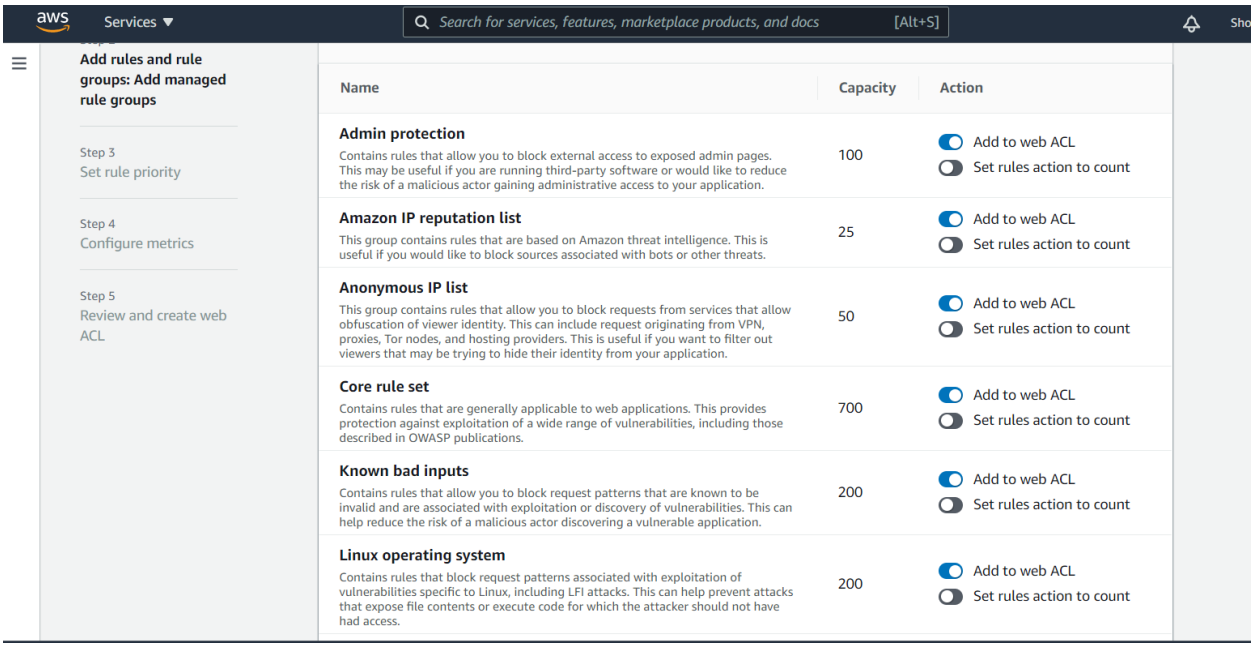


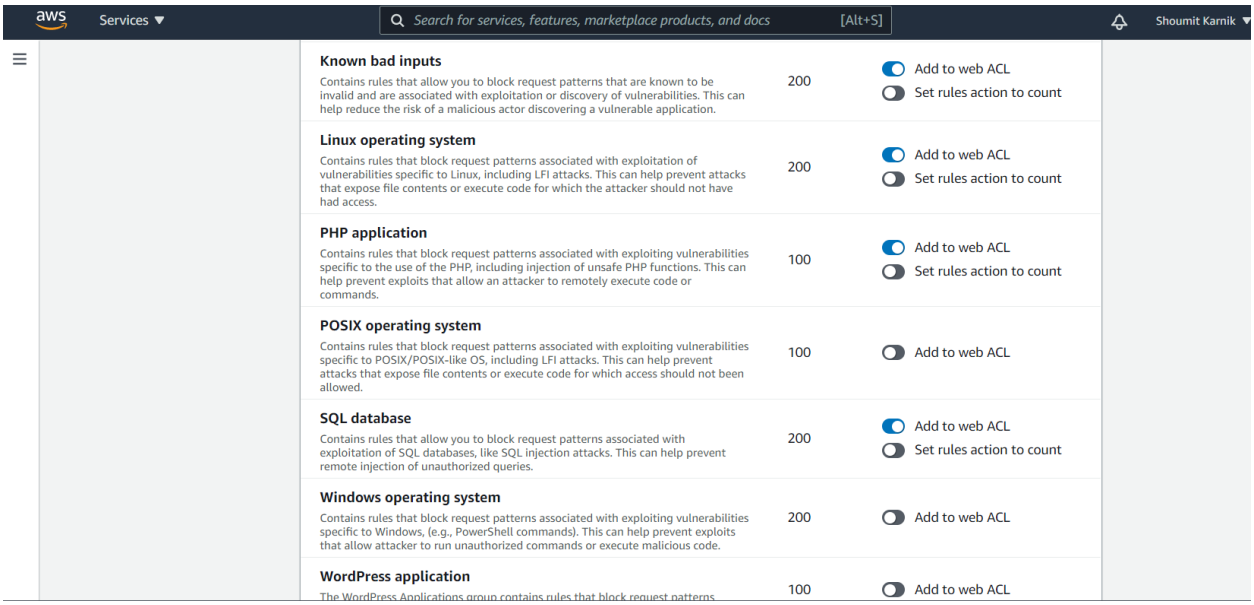
For AWS WAF settings :

A set of Web ACL rules will be used to setup AWS WAF.



The following rules will be selected to configure AWS WAF. Their priority can also be adjusted based on the security needs.





aws Services ▼ Search for services, features, marketplace products, and docs [Alt+S] Shoumit Karnik ▼

resources

Step 2
Add rules and rule groups

Step 3
Set rule priority

Step 4
Configure metrics

Step 5
Review and create web ACL

are reusable collections of rules. You can use managed rule groups offered by AWS and AWS Marketplace sellers. You can also write your own rules and use your own rule groups.

Rules

If a request matches a rule, take the corresponding action. The rules are prioritized in order they appear.

Edit | Delete | Add rules ▼

Name	Capacity	Action
AWS-AWSManagedRulesAdminProtectionRuleSet	100	Use rule actions
AWS-AWSManagedRulesAmazonIpReputationList	25	Use rule actions
AWS-AWSManagedRulesAnonymousIpList	50	Use rule actions
AWS-AWSManagedRulesCommonRuleSet	700	Use rule actions
AWS-AWSManagedRulesKnownBadInputsRuleSet	200	Use rule actions
AWS-AWSManagedRulesLinuxRuleSet	200	Use rule actions
AWS-AWSManagedRulesSQLiRuleSet	200	Use rule actions

Web ACL rule capacity units used

Web ACL rule capacity units used
The total capacity units used by the web ACL can't exceed 1500.
1475/1500 WCUs

Default web ACL action for requests that don't match any rules

Default action
- (●) Allow
- ( ) Block

WAF & Shield

- AWS WAF
  - Getting started
  - Web ACLs
  - IP sets
  - Regex pattern sets
  - Rule groups
  - AWS Marketplace
  - Switch to AWS WAF Classic
- AWS Shield
- AWS Firewall Manager

Introducing the new AWS WAF
We've improved the console and API experience and added AWS Managed Rules. Learn more.
Note: The previous version of AWS WAF is now named AWS WAF Classic. To access resources created with that version, switch to AWS WAF Classic.

Success
You successfully created the web ACL: ENPM809F_Final

AWS WAF > Web ACLs

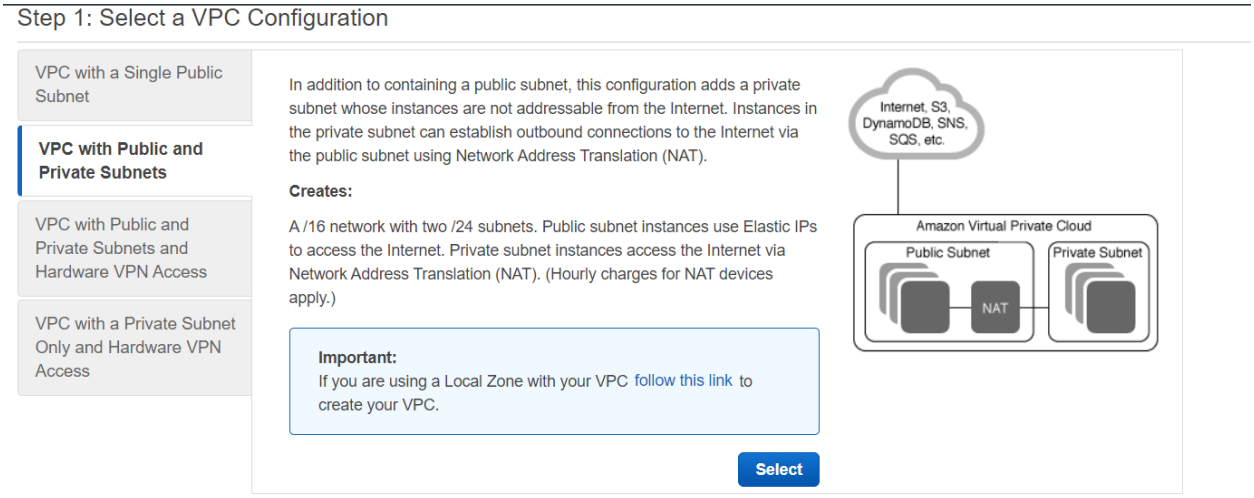
Web ACLs Info

US East (N. Virginia) ▼ | Copy ARN | Delete | Create web ACL

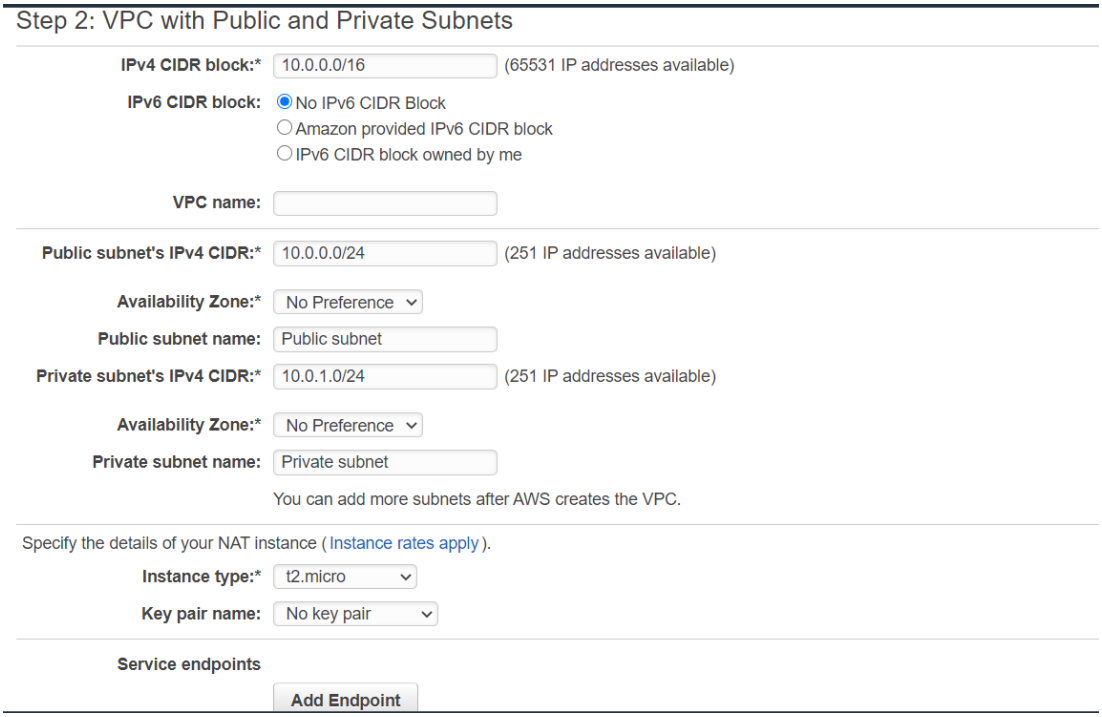
Find web ACLs

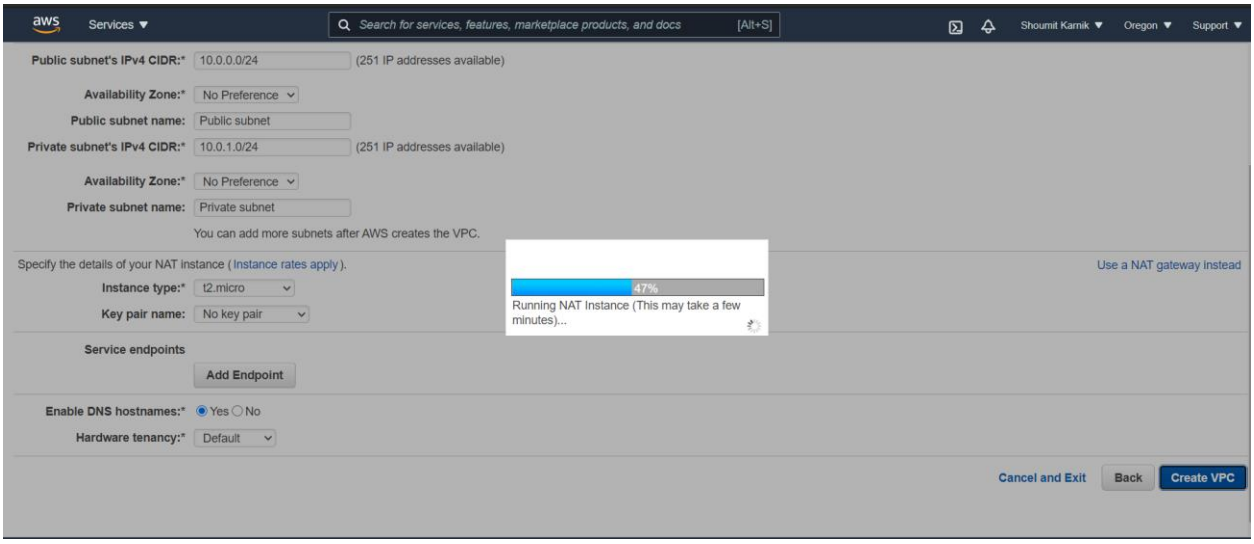
Name	Description	ID
ENPM809F_Final	-	f5780f8e-a87e-4e54-86c1-6bd38ae9ad3a

For Employee VPC settings the following steps will be followed:

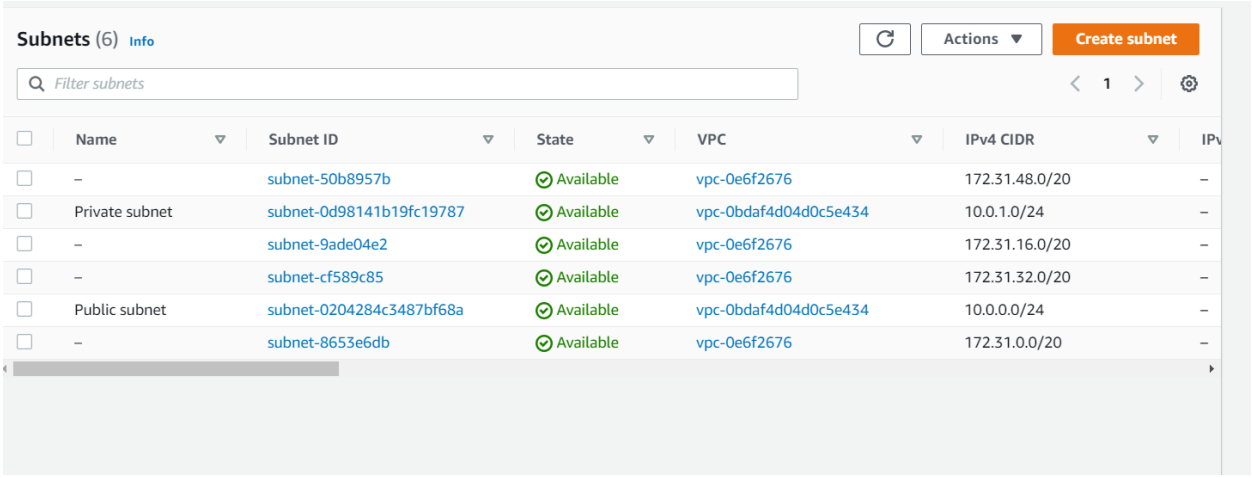


A VPC with Public and Private Subnets will be selected because we do not need Employee instances accessible from the internet, hence they will be included in the private subnet.

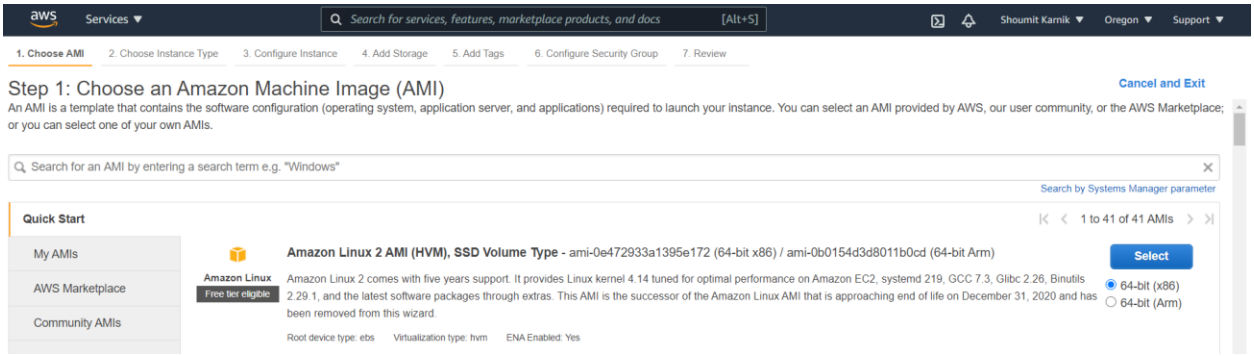




The IP address ranges of the subnets created are:

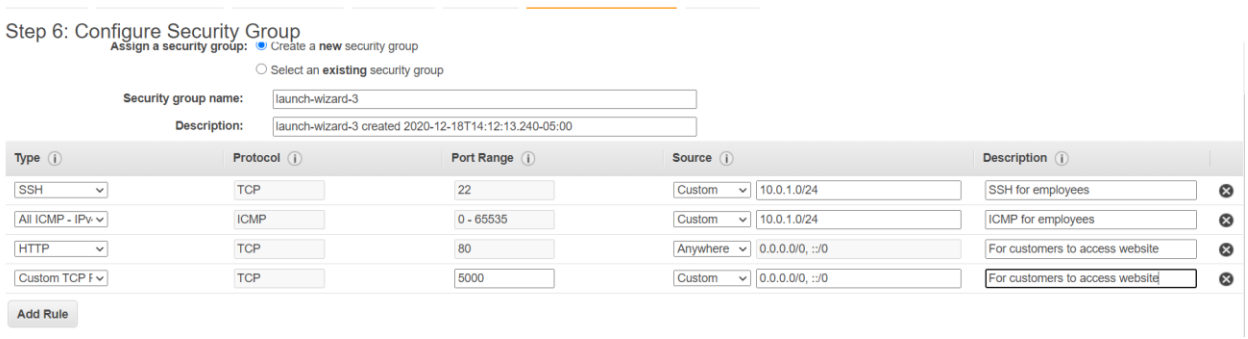


Next an instance is to be created for the website to be migrated to:



For the security group configurations, the considerations that have been taken into account are as follows:

- The website will be facing the internet, hence it will be open to all and hence the IP address configuration would be 0.0.0.0/24 for ports 80 and 5000.
- The employee private subnet will be used for the employee network. They will be given some special port access privileges like SSH and ICMP. This would help them with:
    - Accessing instances for changing/modifying code
    - Performing network diagnostics in case of system failures



For the internal EC2 instance security, as an added layer of security, iptables can be installed and configured on the instance where the website is running.

On the EC2 instance after SSH access, the following commands will be run:

sudo yum install iptables-services

sudo iptables -nvL

These commands install iptables and give a current state of the iptables firewall.

```
[ec2-user@ip-172-31-36-161 ~]$ sudo yum install iptables-services
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package iptables-services.x86_64 0:1.8.4-10.amzn2.1.2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package              Arch       Version                 Repository      Size
================================================================================
Installing:
 iptables-services    x86_64     1.8.4-10.amzn2.1.2      amzn2-core      58 k

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

Total download size: 58 k
Installed size: 24 k
Is this ok [y/d/N]:
```

The rules added here will be as follows:

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

iptables -A INPUT -p tcp --dport 5000 -j ACCEPT

iptables -A INPUT -p icmp -s 10.0.1.0/24 -j ACCEPT

iptables -R INPUT 1 -p tcp -s 10.0.1.0/24 --dport 22 -j ACCEPT

```
[ec2-user@ip-172-31-36-161 ~]$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEP
T
[ec2-user@ip-172-31-36-161 ~]$ sudo iptables -A INPUT -p tcp --dport 5000 -j ACC
EPT
[ec2-user@ip-172-31-36-161 ~]$ sudo iptables -A INPUT -p icmp -s 10.0.1.0/24 -j
ACCEPT
[ec2-user@ip-172-31-36-161 ~]$ sudo iptables -R INPUT 1 -p tcp -s 10.0.1.0/24 --
dport 22 -j ACCEPT
```

The rules have been updated and specific IP ranges have been let in for SSH and ICMP and all traffic is allowed for port 80 and 5000, since the website is running on these two ports.

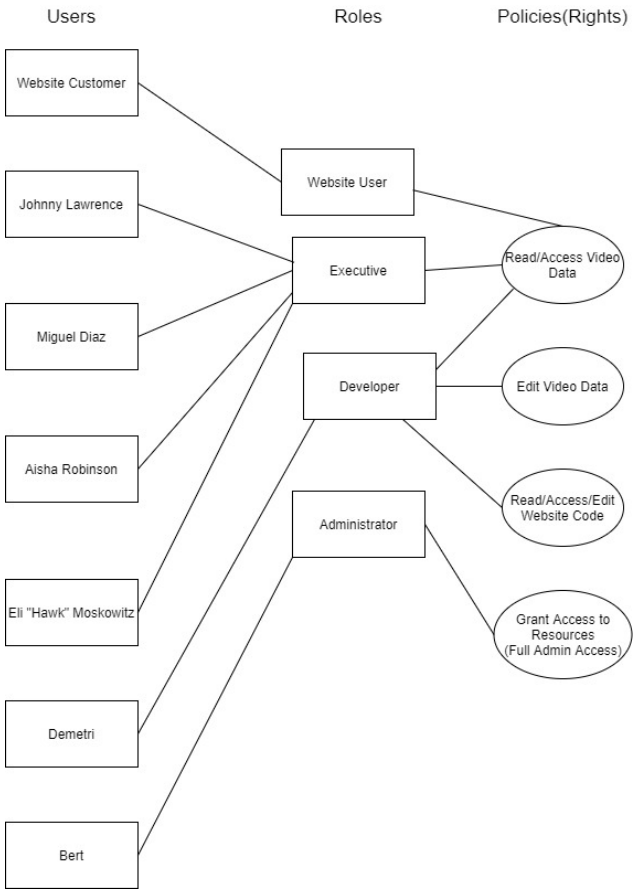
```
[ec2-user@ip-172-31-36-161 ~]$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 15 packets, 1084 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     tcp  --  *      *       10.0.1.0/24          0.0.0.0/0
         tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
         tcp dpt:5000
    0     0 ACCEPT     icmp --  *      *       10.0.1.0/24          0.0.0.0/0


Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 9 packets, 2028 bytes)
 pkts bytes target     prot opt in     out     source               destination

[ec2-user@ip-172-31-36-161 ~]$
```

Note that these rules were based off the data provided in the sample VM and the data provided in the document:

```
enpm809j@ubuntu:~$ sudo netstat -tulpn | grep LISTEN
[sudo] password for enpm809j:
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN      862/python3
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      871/mysqld
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      884/nginx -g daemon
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      851/sshd
tcp6       0      0 :::80                   :::*                    LISTEN      884/nginx -g daemon
tcp6       0      0 :::22                   :::*                    LISTEN      851/sshd
```

IAM policy/role configurations:

We have the following organizational structure:



These roles were outlined in the initial plan provided and we will be building our roles and policies based on that plan.

The AWS Identity and Access Management console will be used.

Various employee roles will be associated to our instance. Their consfiguration settings have been given below. The Website user role for the customers has not been configured, since they will be directly accessing the website. This will be done using a temporary key being provided to them from the EC2 instance and then later they can get read access to a bucket which will store the data.

Roles:

Three employee roles which would be created are:

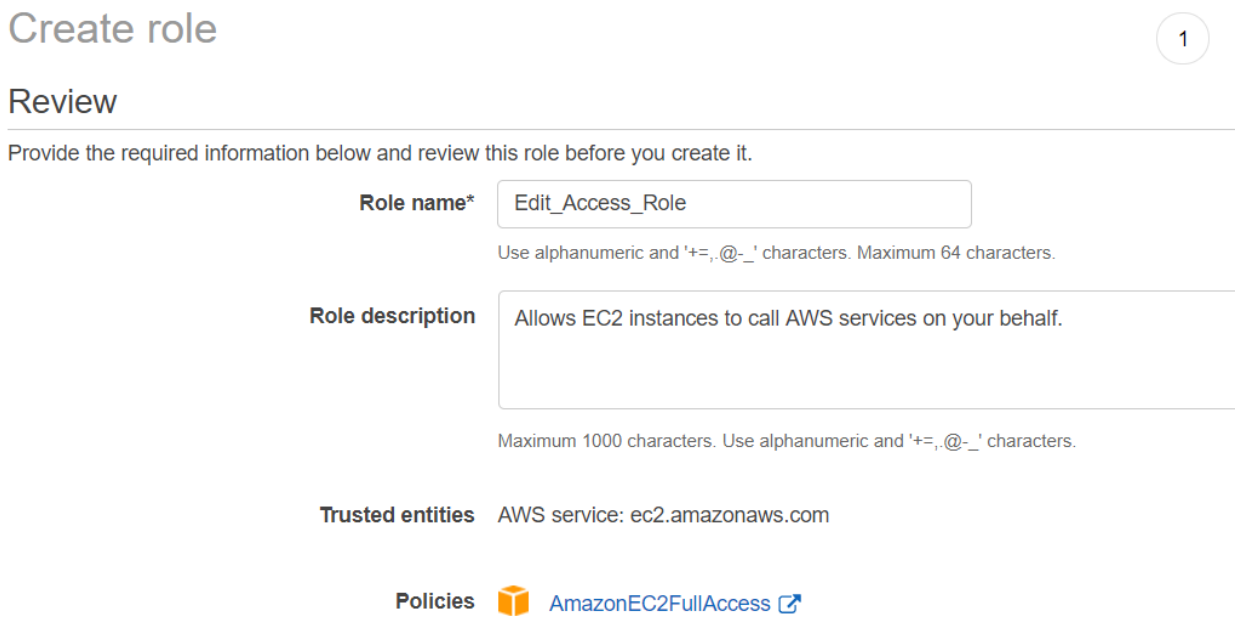
Edit_Access_Role – Given to developers to fully access the EC2 instance and make changes to code or data.

Org_Admin Role – Given to the system administrator (Bert). This enables them to add/delete new users and make changes to the configurations of the system.

Read_Access_Role – Given to high level executives of the company, who only need to view the data or code but should not be given edit access

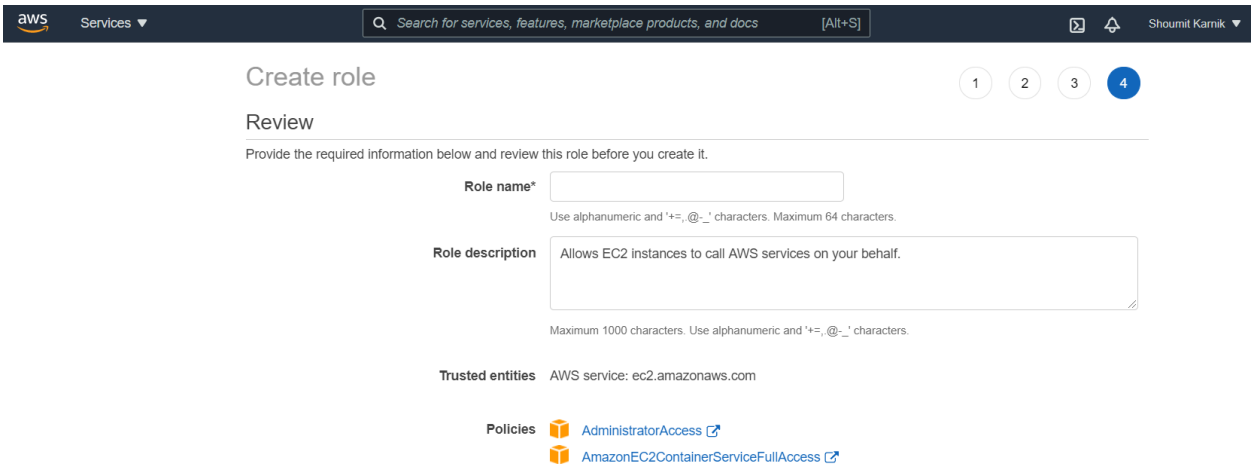
The permission policies attached to each role are as follows:

Edit_Access_Role :



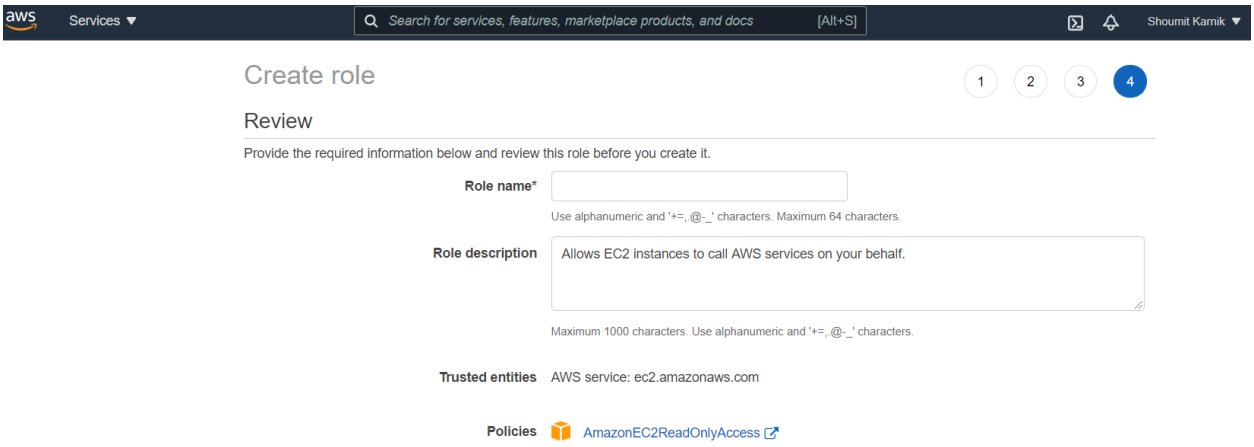
Policies attached: AmazonEC2FullAccess

Org_Admin Role:



Policies attached: AdministratorAccess, AmazonEC2ContainerServiceFullAccess

Read_Access_Role:

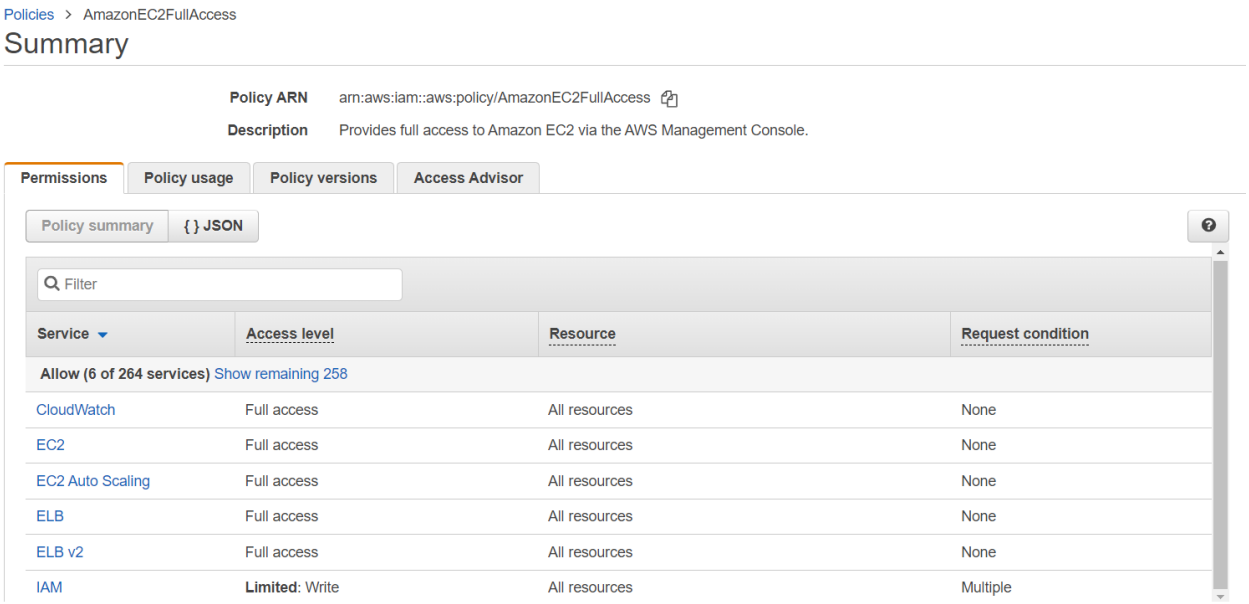


Policies attached: AmazonEC2ReadOnlyAccess

Policies:

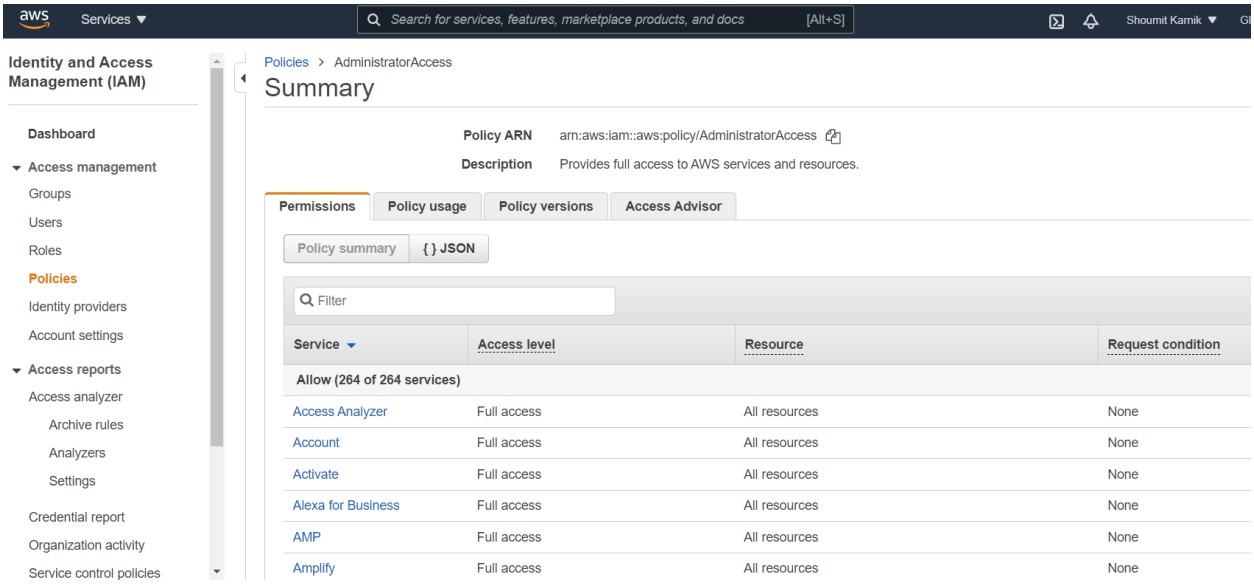
The policy details are as follows :

AmazonEC2FullAccess:

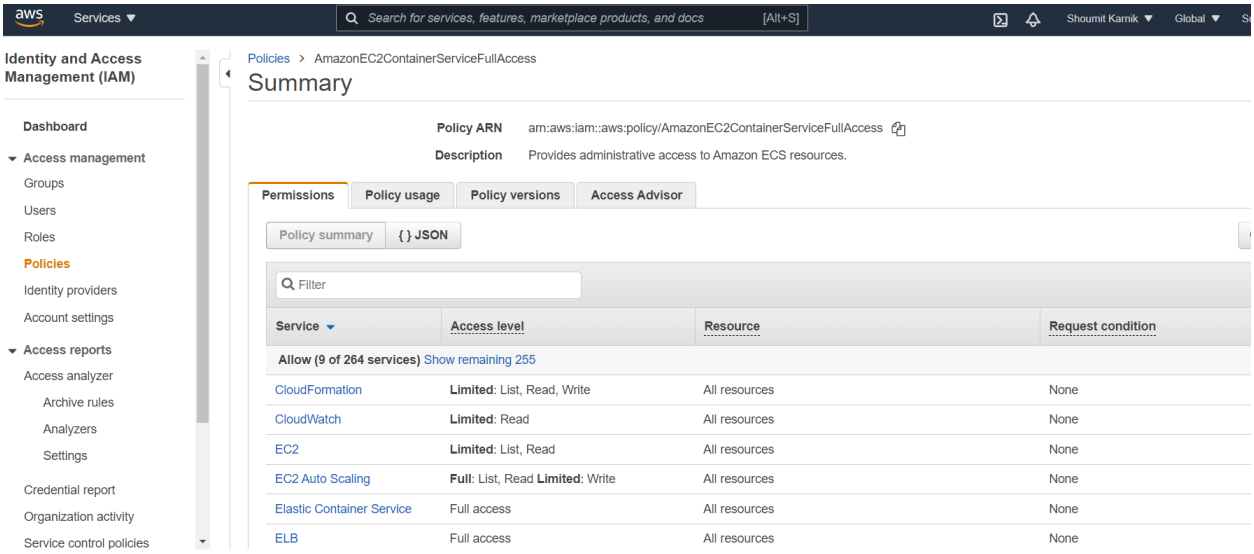


This policy will be given to the developer. It ensures that the developer can make necessary changes to the EC2 instance but cannot override the Administrator privileges.

AdministratorAccess:

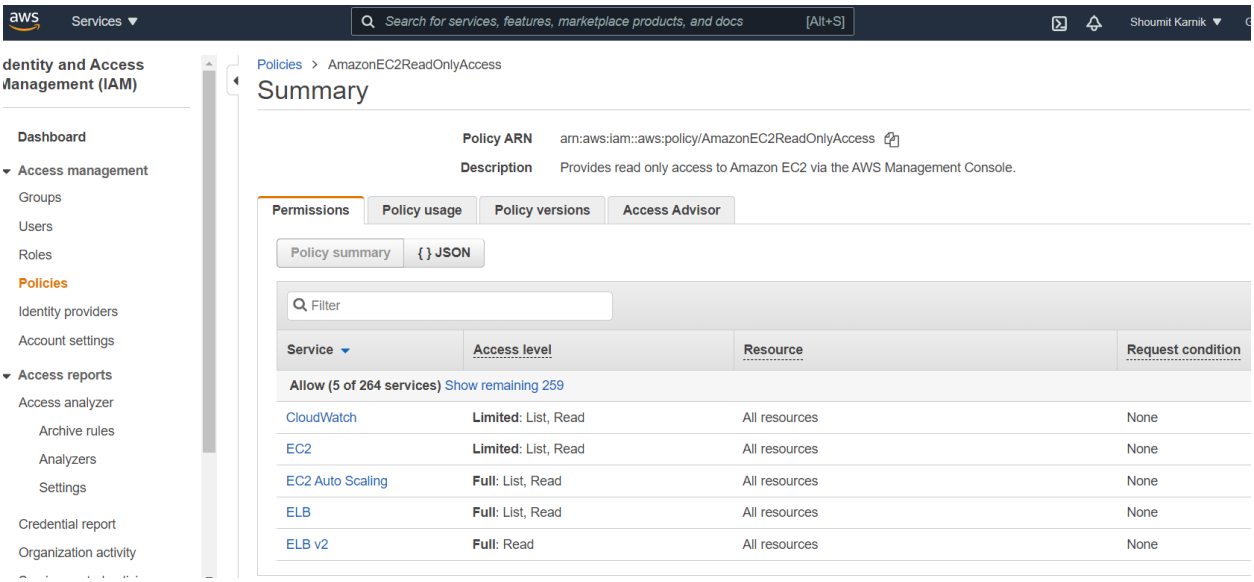


AmazonEC2ContainerServiceFullAccess:



It is necessary to give the administrator full access to these resources, since they will be fully responsible for managing services like configurations, IAM etc. Hence the organization's amazon account's Administrator role needs to be assigned to them.

AmazonEC2ReadOnlyAccess:



This role helps the executive staff to just have read permissions to the instance so that they can access the data and make sure that everything is set correctly. If they need any configuration or development changes, they should be contacting the respective staff.

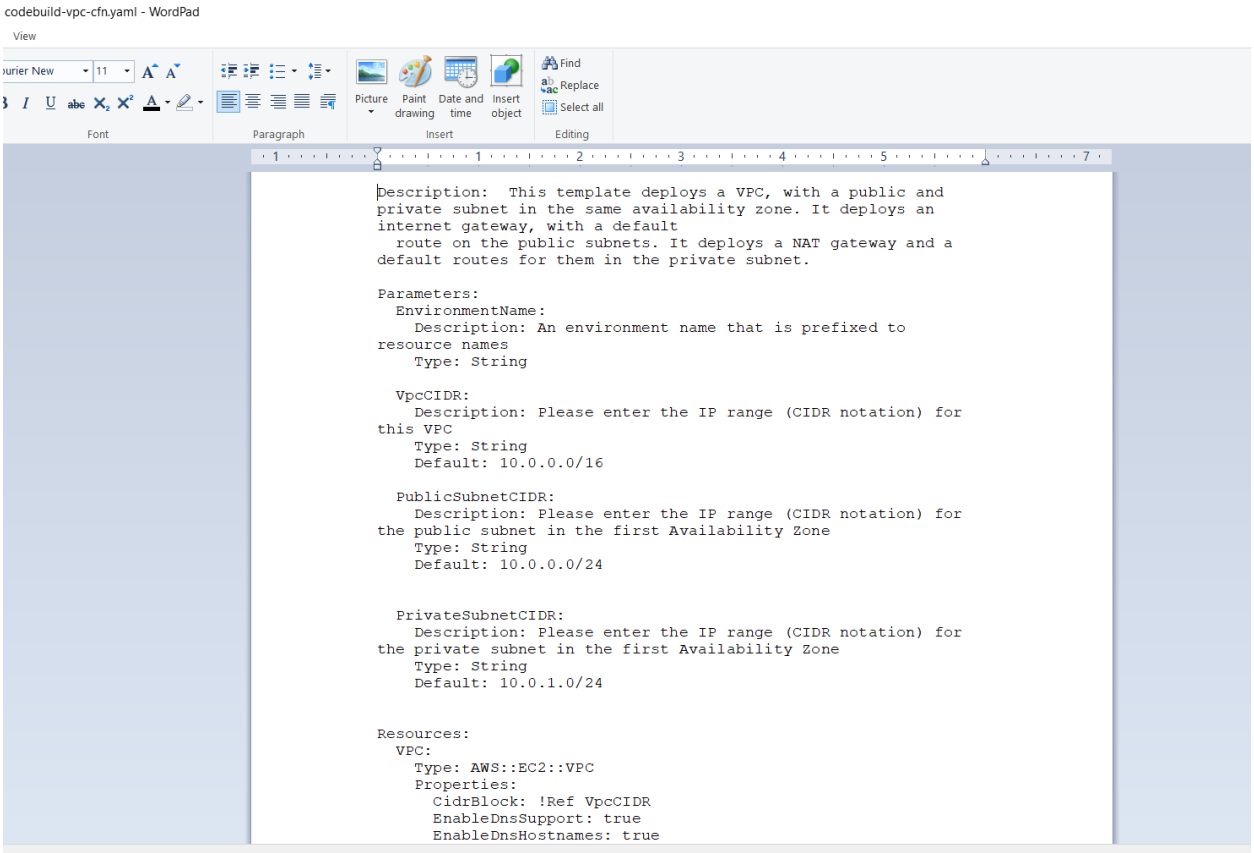
The users listed above can be assigned the aforementioned roles, hence giving them the required access without providing more access than required.

Sample CloudFormation Templates:

Some very simple sample CloudFormation templates in YAML(YAML Ain't Markup Language) format have been outlined for the migration task. Note that this only has automated the security groups and the IAM roles for now.

For VPC automation:

CloudFormation YAML:



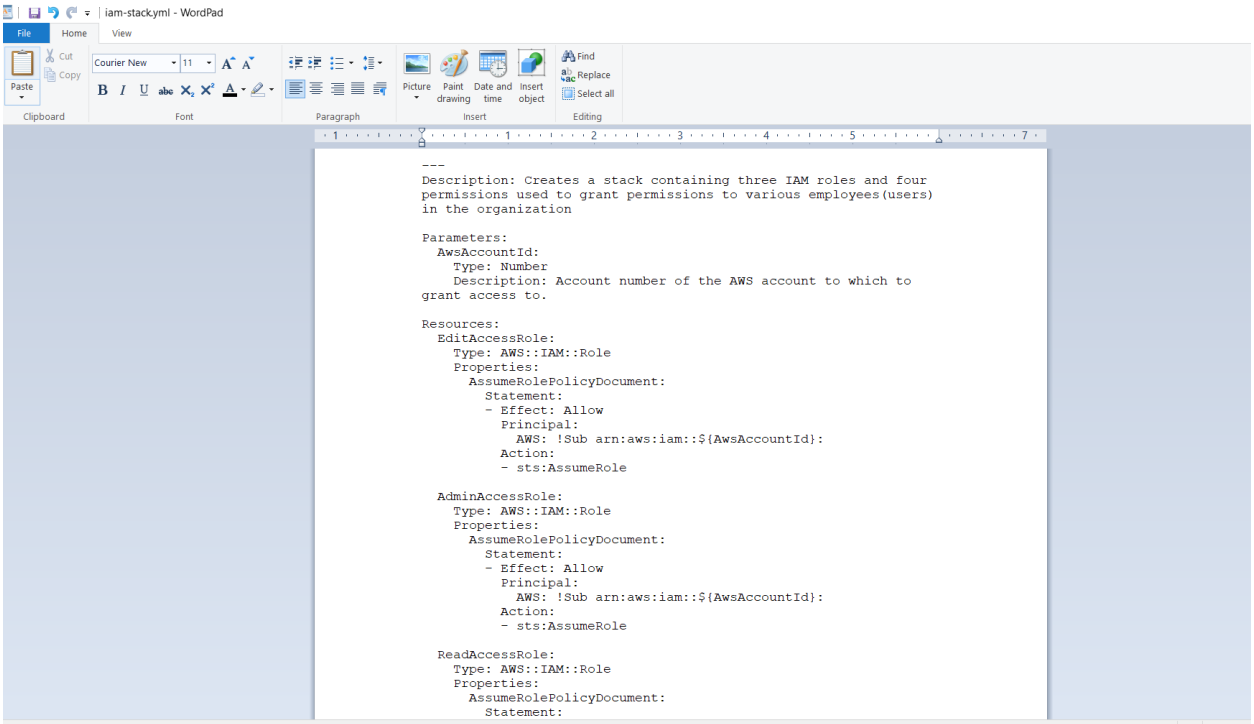

```
Description:  This template deploys a VPC, with a public and
private subnet in the same availability zone. It deploys an
internet gateway, with a default
  route on the public subnets. It deploys a NAT gateway and a
default routes for them in the private subnet.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to
resource names
    Type: String

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for
this VPC
    Type: String
    Default: 10.0.0.0/16

  PublicSubnetCIDR:
    Description: Please enter the IP range (CIDR notation) for
the public subnet in the first Availability Zone
    Type: String
    Default: 10.0.0.0/24


  PrivateSubnetCIDR:
    Description: Please enter the IP range (CIDR notation) for
the private subnet in the first Availability Zone
    Type: String
    Default: 10.0.1.0/24


Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
```

This file deploys the VPC and the security groups discussed above. It is in YAML format. It has been attached as a separate file. I have used the file given in the reference and modified it to fit the purpose of the project. [Ref.]

Note: The EC2 instance creation and iptables configuration still needs to be done manually.

For IAM Role Automation:

CloudFormation YML:



```
---
Description: Creates a stack containing three IAM roles and four
permissions used to grant permissions to various employees(users)
in the organization

Parameters:
  AwsAccountId:
    Type: Number
    Description: Account number of the AWS account to which to
grant access to.

Resources:
  EditAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            AWS: !Sub arn:aws:iam::${AwsAccountId}:
          Action:
          - sts:AssumeRole

  AdminAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            AWS: !Sub arn:aws:iam::${AwsAccountId}:
          Action:
          - sts:AssumeRole

  ReadAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
```

This YML file can be used for the automated creation of the IAM roles and policies that were seen in the IAM policy/role configurations section. This file has also been attached as a separate file. The code has been taken from the various JSON (JavaScript Object Notation) files of the different policies and roles specified. The Account ID spaces however have been left blank so that Account credentials cannot be leaked. They should be filled in with the organization's Amazon account credentials and permissions.

These would be the initial cloud migration steps strictly from a security standpoint. These have been developed using a free AWS account. (Except for AWS WAF, it costed 0.6$)

FUTURE SCOPE

Given more time and access to more resources in AWS which are mostly paid services the following can be done:

- A more detailed approach for the AWS CloudFormation Templates
- A server(NGINX) was seen in the code. A separate node can be created for that in AWS Console using Cloud Migration Hub.
- The system is DDoS(Distributed Denial of Service) resilient because of the AWS WAF right now, but to make it more efficient a distributed set of nodes and EC2 instances will be required as mentioned in initial document.
- Right now, the system does not use rolling keys and it is assumed that everyone will have their own key to access the system. This can be made more secure with the use of Amazon KeyStore as mentioned in the previous migration proposal.