**Name:**Dhanush Devaladakere Arvind

**UID:**119151556

**FINAL(COBRA KAI)**

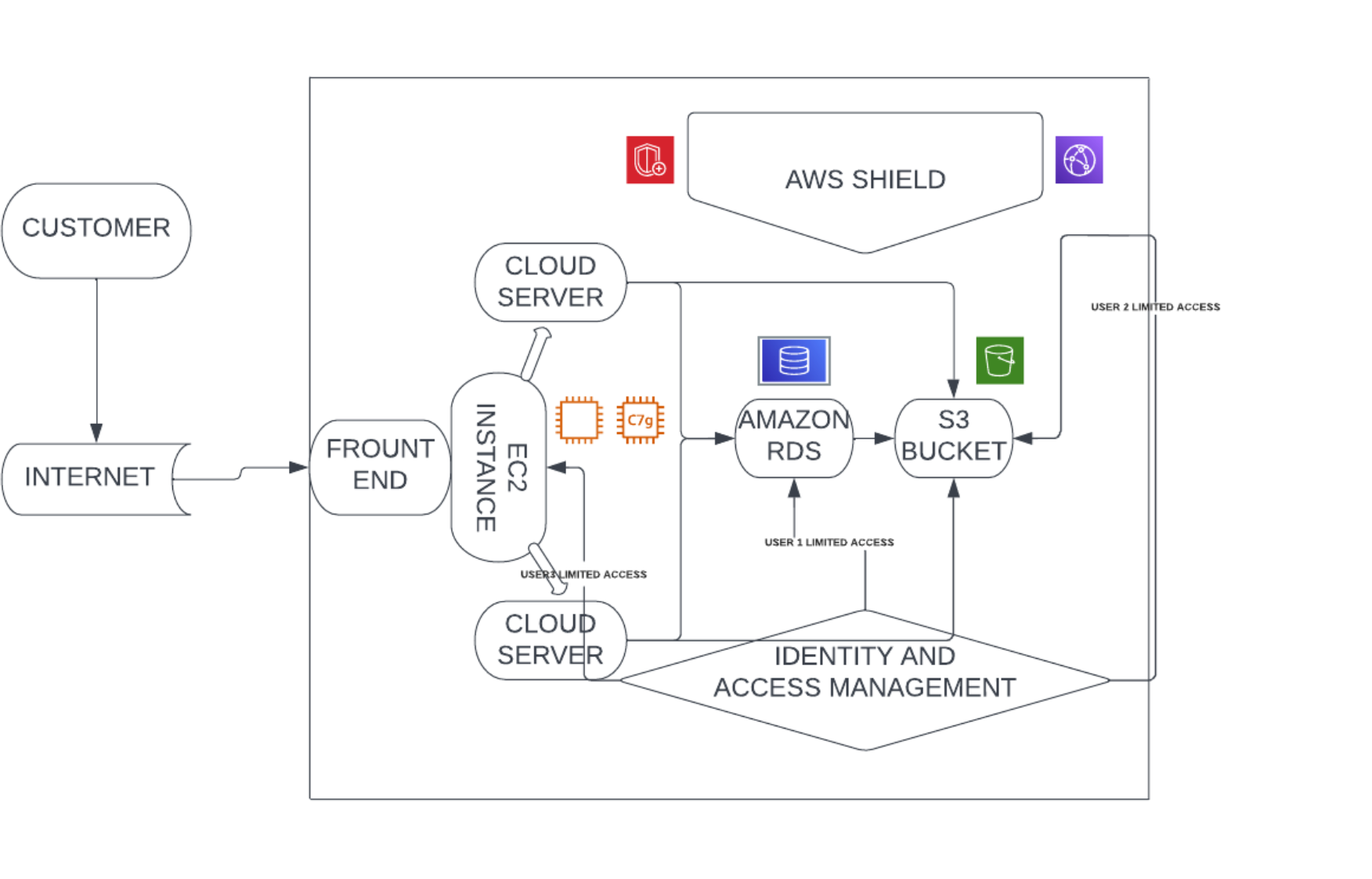
When it comes to the technical aspect of running a web application on the cloud.There are a lot of things to consider.One of the main things to consider is the budget. There are multiple ways to launch a web application on the cloud and the most common tool used to launch a website on the AWS cloud is the Amazon EC2.Most of the companies with a large website have many file’s(static or not)which help the website to run.In these cases the Amazon ec2 instances are choses over any other tool set in AWS.

Let’s first point out all the ways to launch a web application:

* AWS amplify
* Amazon S3
* AWS Lambda(serverless computing)
* Amazon EC2
* Amazon Lightsail

These are some of the ways an organization (company) can launch their website.For our case, we will be using the amazon ec2,assuming our Cobra Kai website is very large.

**PREVIOUS ARCHITECTURE**

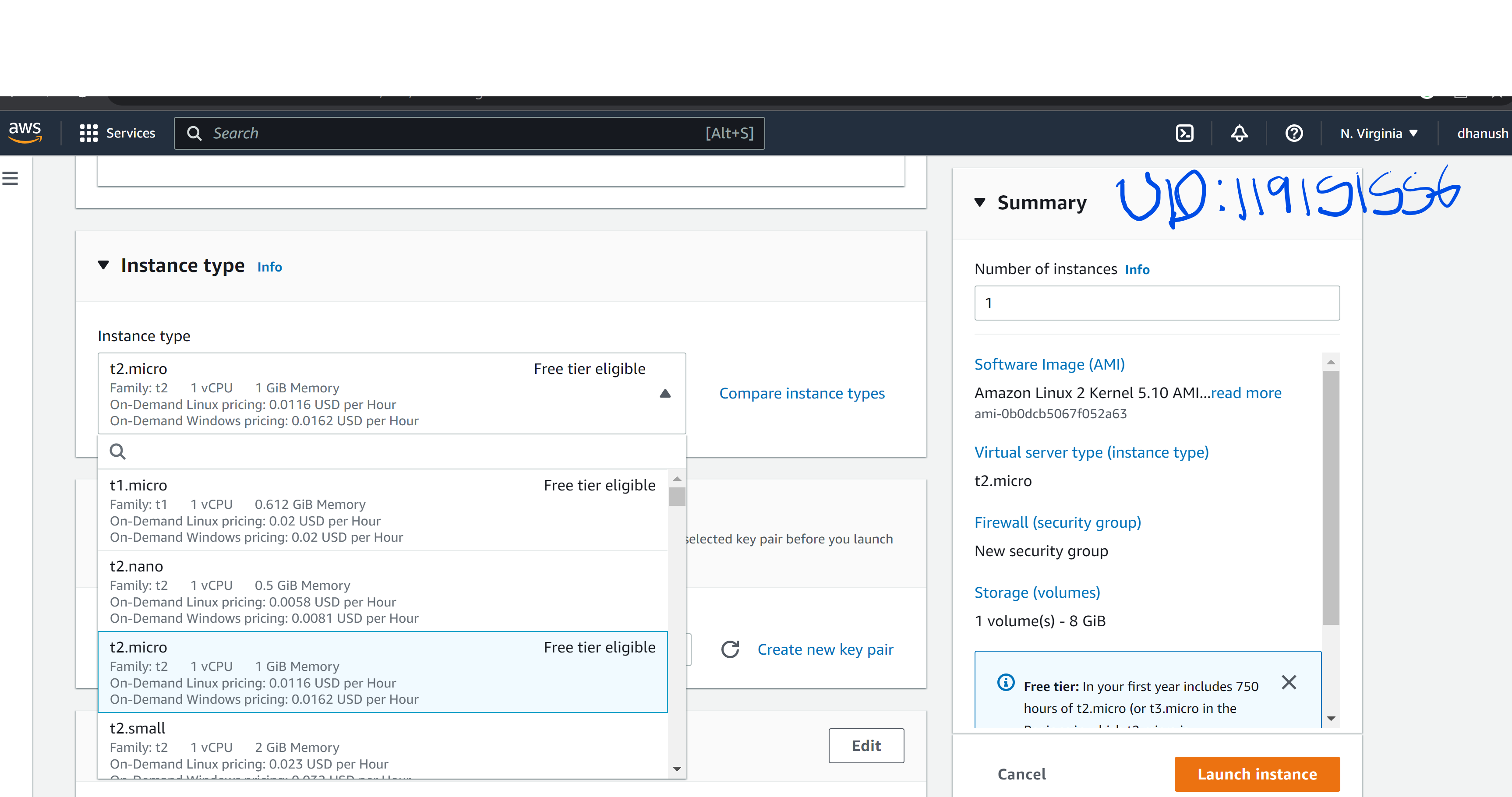
Fig 1.0.0

**DEPLOYING THE WEBSITE ON THE AWS CLOUD**

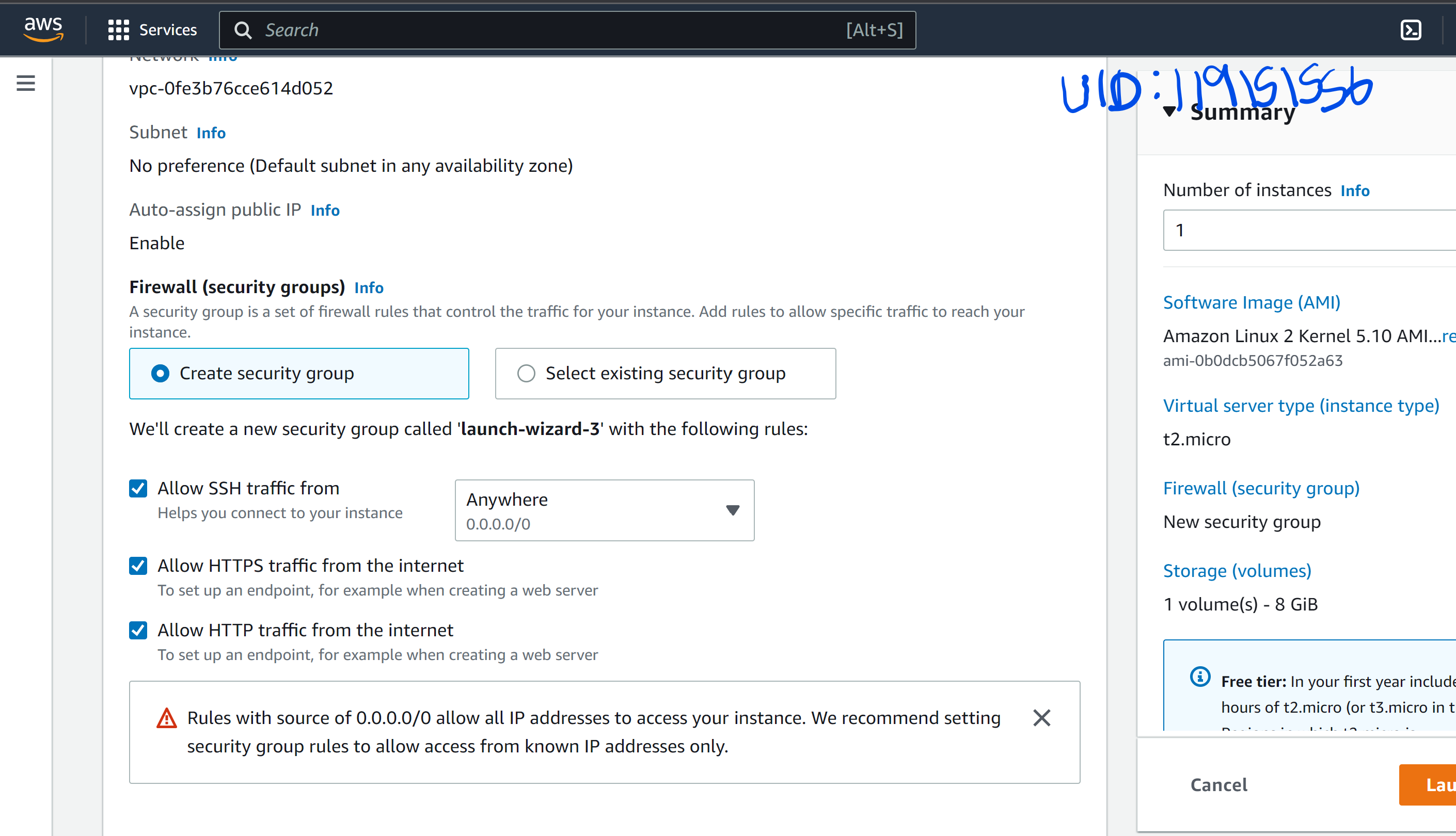
**Step’s:-**

First, I will demonstrate how to launch an instance and then make our website run on that particular instance.

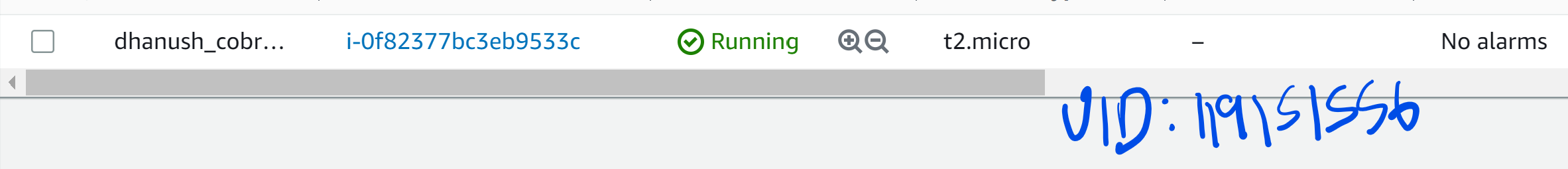
* The first step is to create an instance with any type(recommended to use the free tier type).



* While creating an instance,the options in the security groups section need to be changed from default.We need to allow http and https traffic requesting.This allows the public to access our instance.So enabling them are recommended.



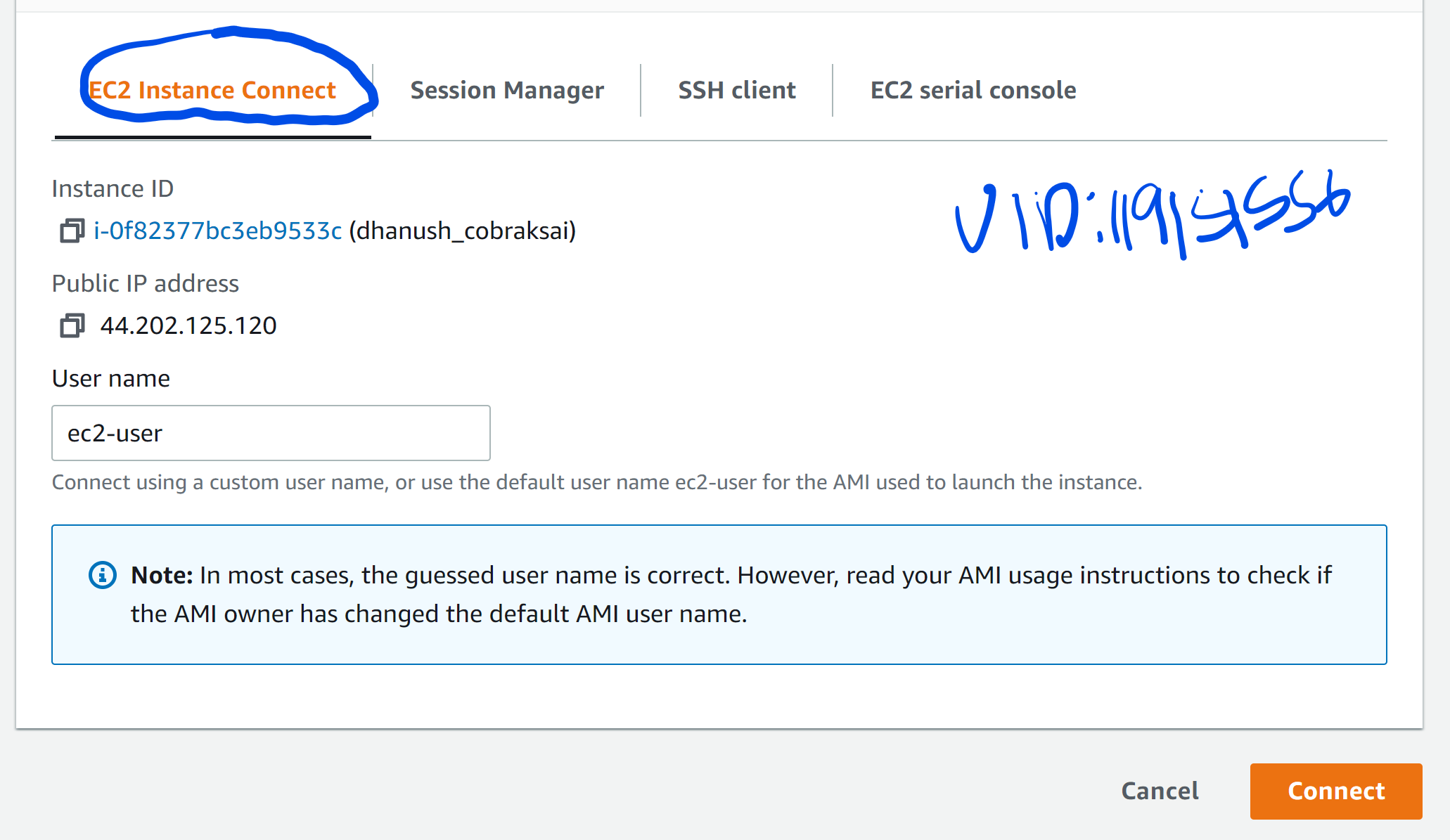
* The instance is running.



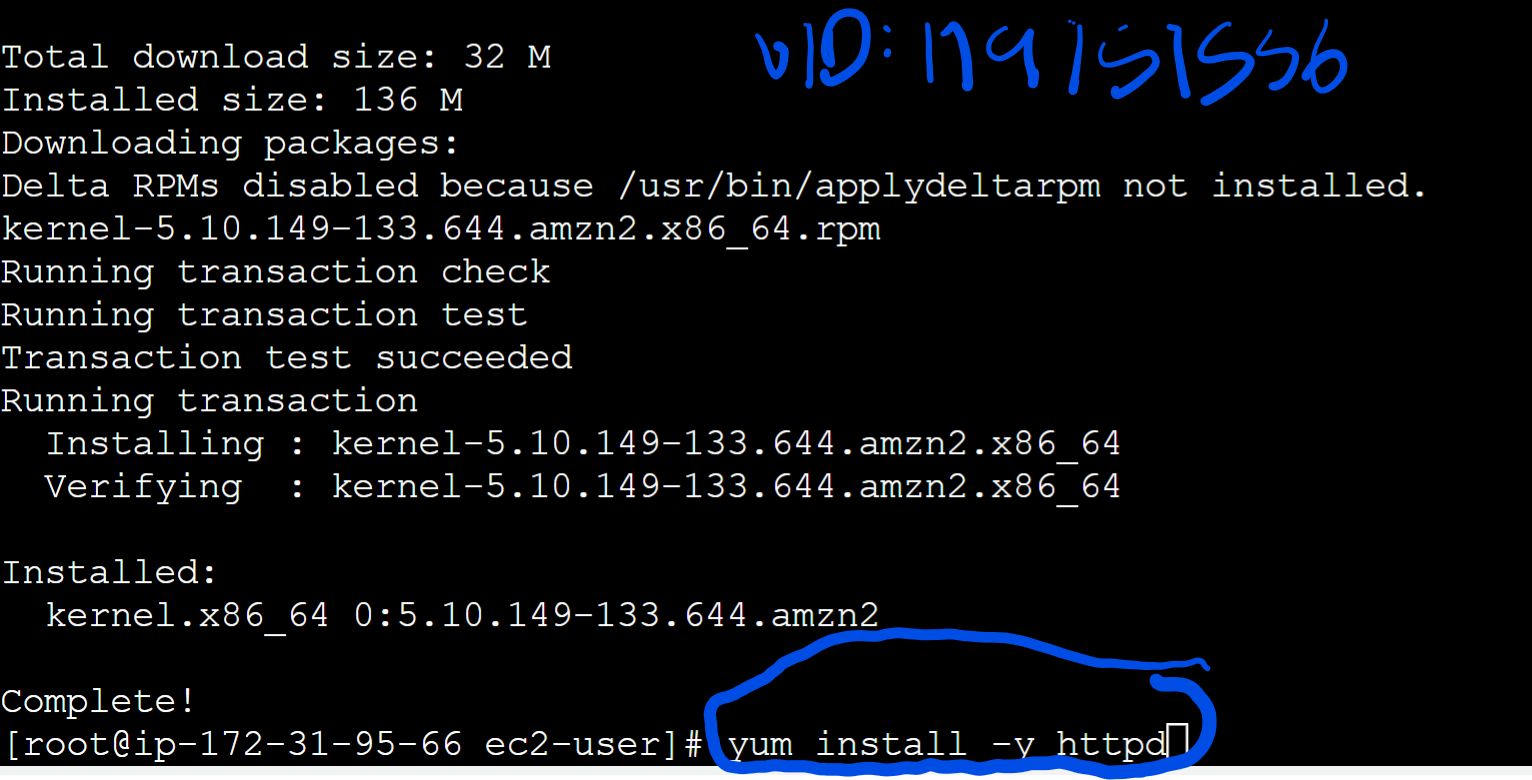
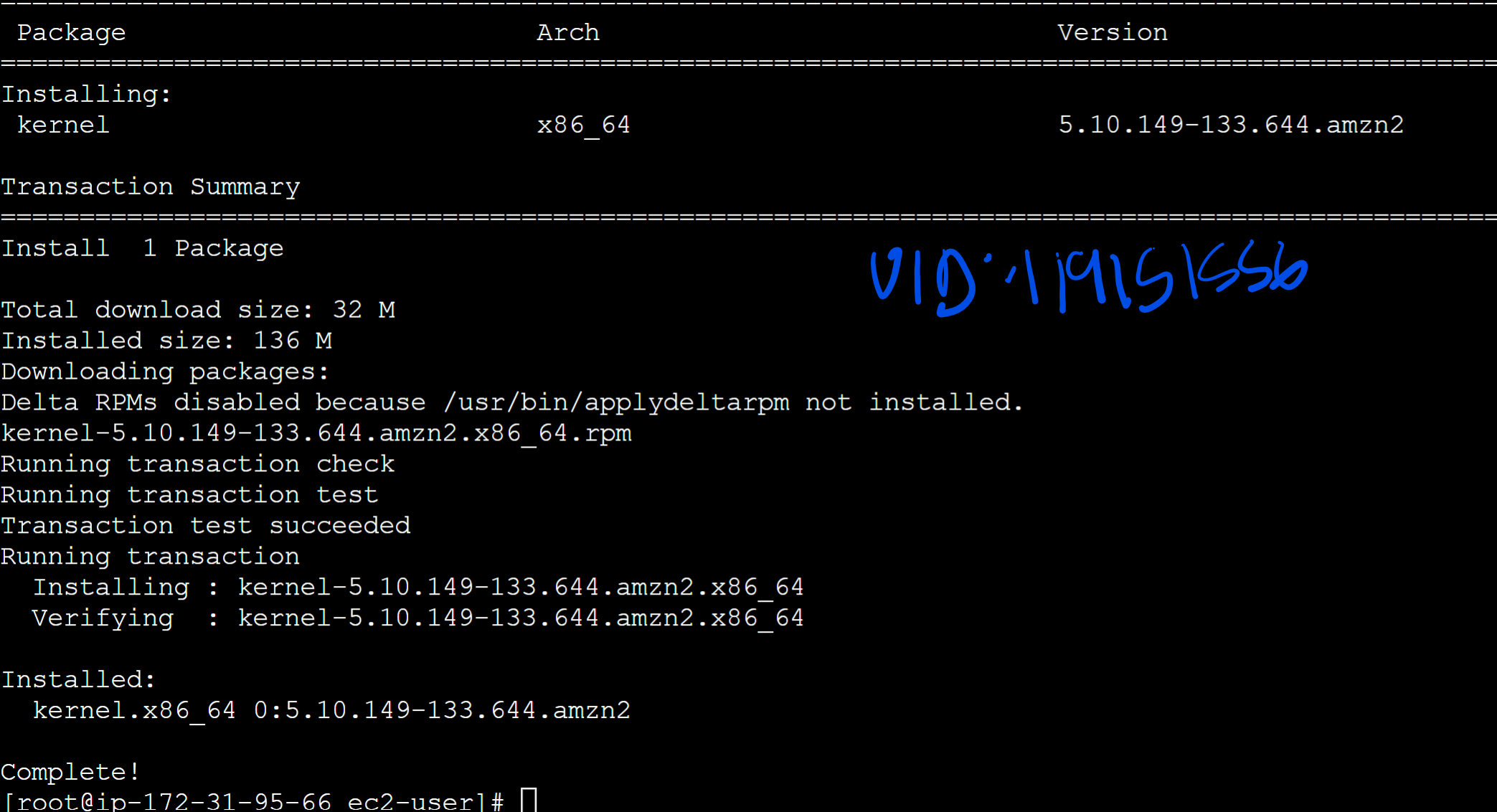
* There are multiple ways to connect to the instance,the recommended options are:-

-Through SSH client.

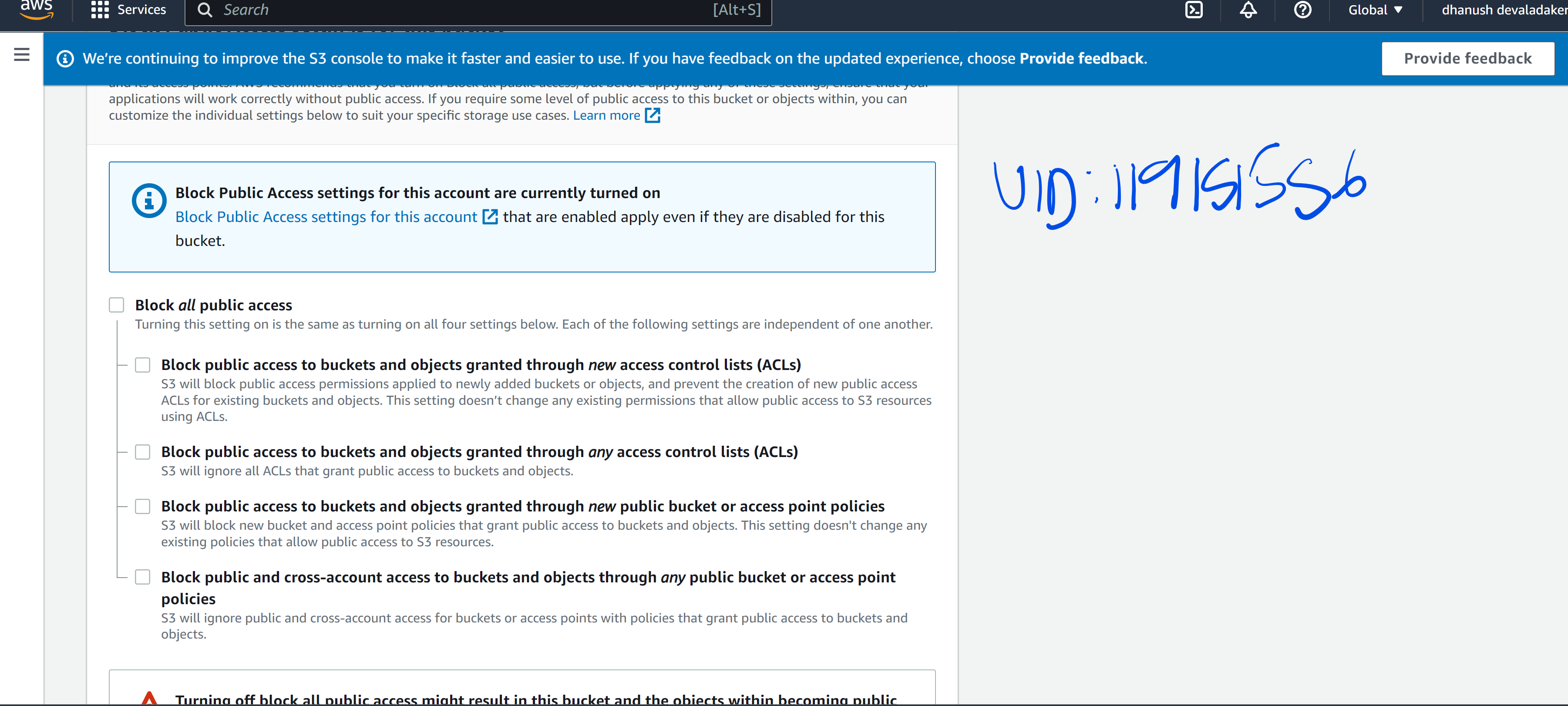
-Through ec3 instance connect. I used the ec2 instance connect



* Once it is launched,we need to update the ec2 instance and then install Apache to use the servers later.
* The command used to install Apache is “yum install httpd -y” or “sudo yum install httpd -y”. They are shown in the two screenshots below.



* Once it downloads, we can check the status of the server by inputting the command “systemctl status https”.It would be inactive as for the moment.
* The next step would be to create a S3 bucket.
* We need to go to the S3 section and go to buckets.While choosing the configuration options, we need to enable public access to our bucket.
* If not, the instance can’t access the files that we store in the bucket to run the application.



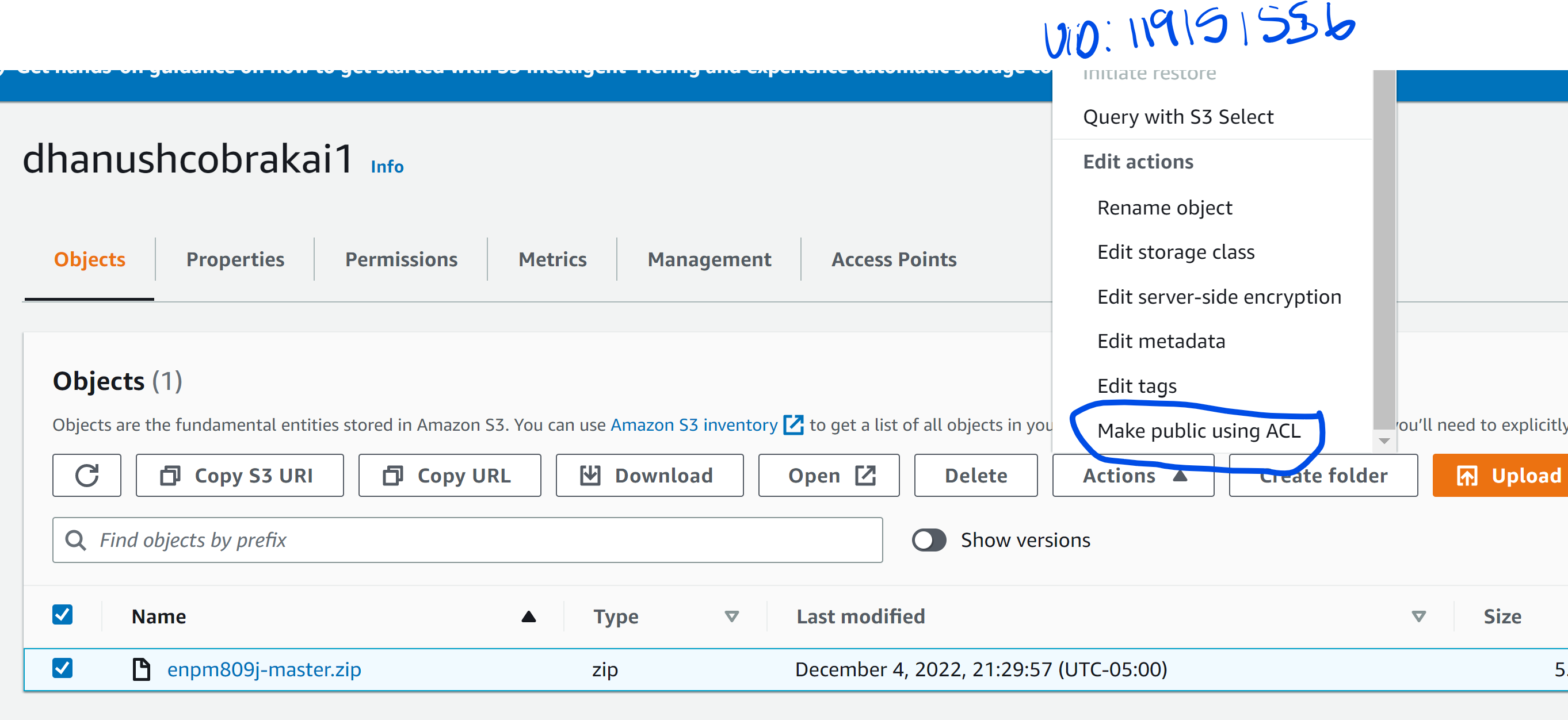
* Then the bucket is created.



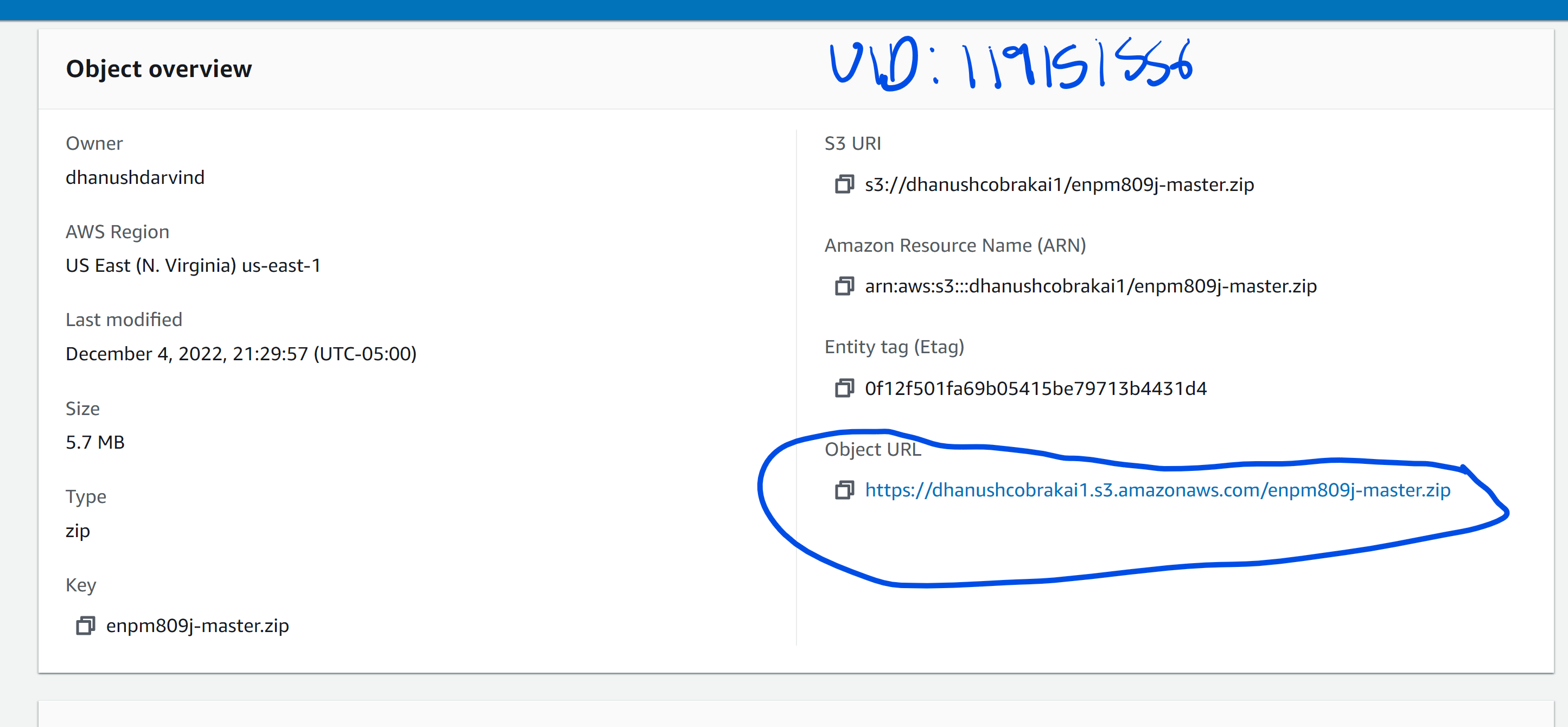
* Next would be to upload the web application files on to the bucket.In my case, I downloaded the zip file from Github through the link provided.Then I uploaded the files onto the S3 bucket I created.(Link:- <https://github.com/kts262/enpm809j>).
* The file is uploaded.



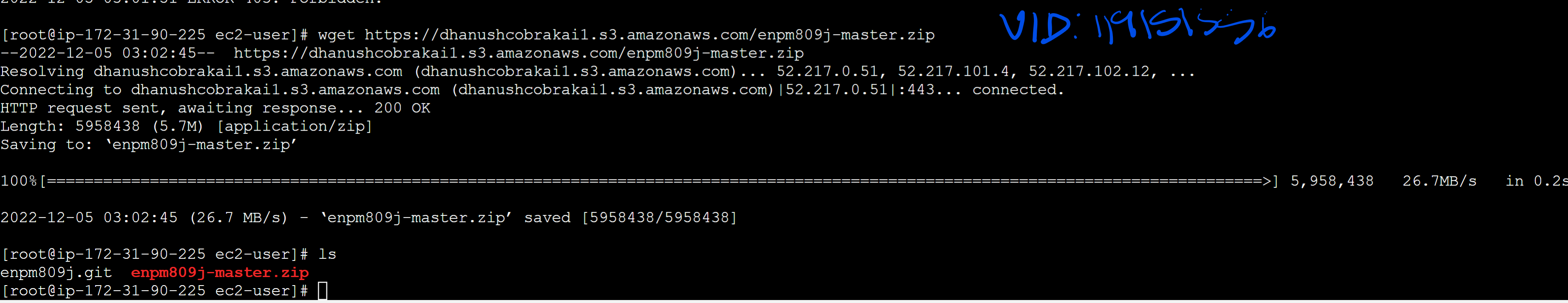
* Making the Zip file accessible to the public is an important step to retrieve the files from the bucket.



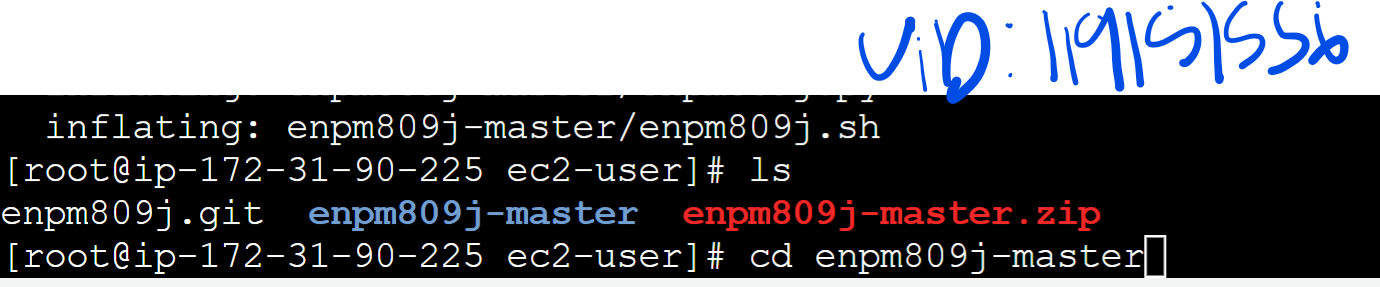
* The next step is to copy the object URl form our bucket and paste it on to the command line interface.



* Then we use the url to get all the files from the bucket and save it in the ec2-user as shown in the next screenshot.
* The command used is “wget URL”



* We got to unzip the file(unzip filename) and list all of them.The unzipped file should appear.



* Then enter the unzipped file using the command “cd”.All the web application files will br displayed.
* We need to then move all of them into the html folder ,where the hosting of the web application takes place.

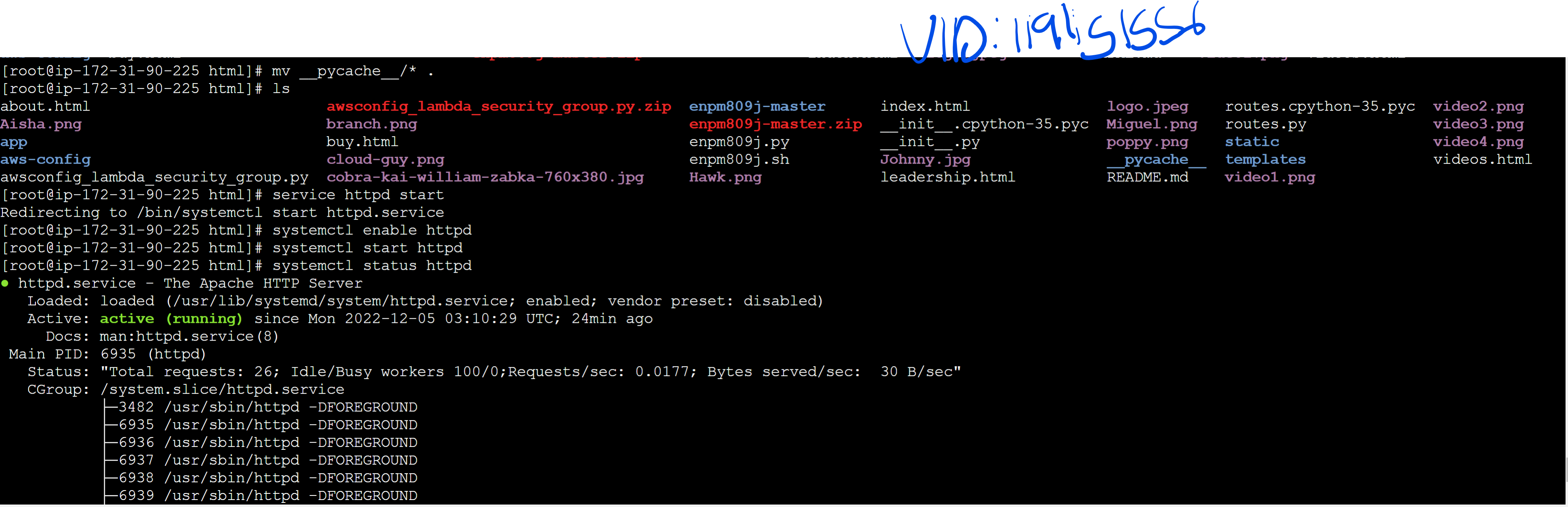


* In this case ,there were many other directories in the unzipped file,we need to get all those files and place them on to the html directory directly.(command:mv filename/\* .).
* We also need to make sure the httpd service is enabled and we manually need to start the server.
* The commands used are:-

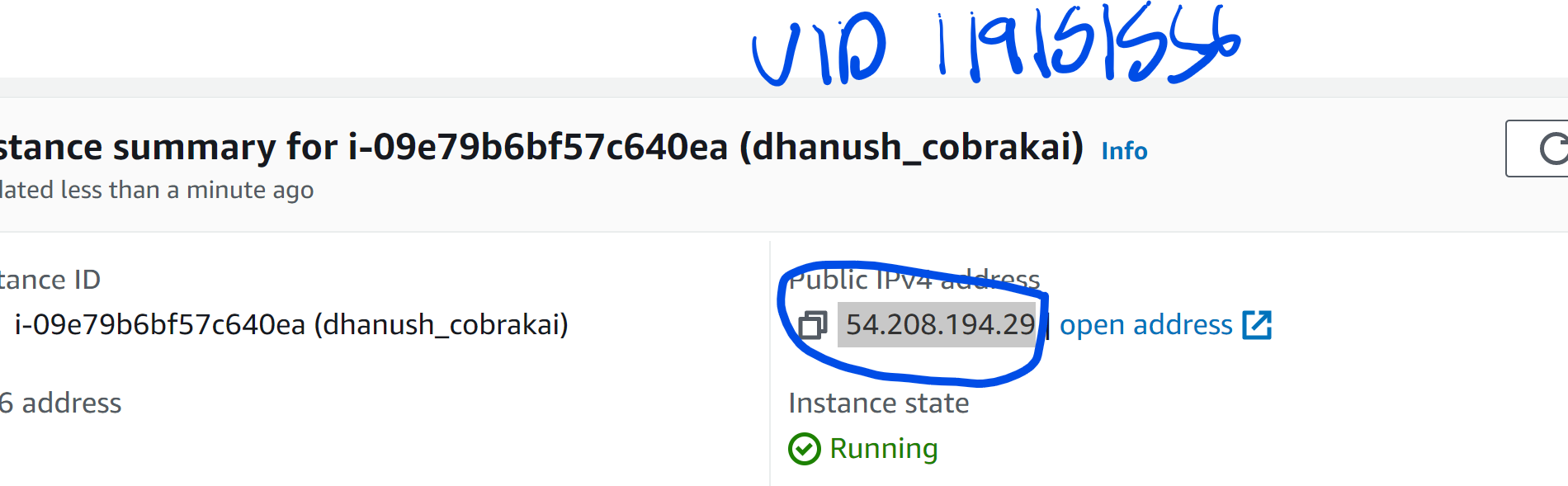
-systemvctl enable httpd

-systemctl start httpd or start httpd.service

* As you can see from the screenshot below, the server is active.



* Then we copy the public ip address of the instance,paste it on to a new tab to make sure the application stored in the S3 bucket we created, is running on our instance.

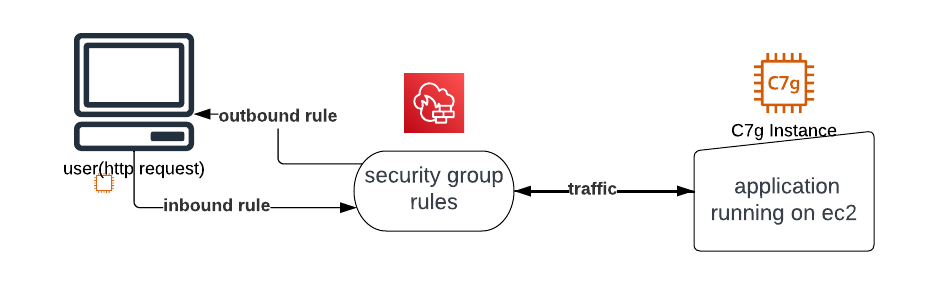


* Yes, the website is now hosted on AWS cloud using the instance we created.



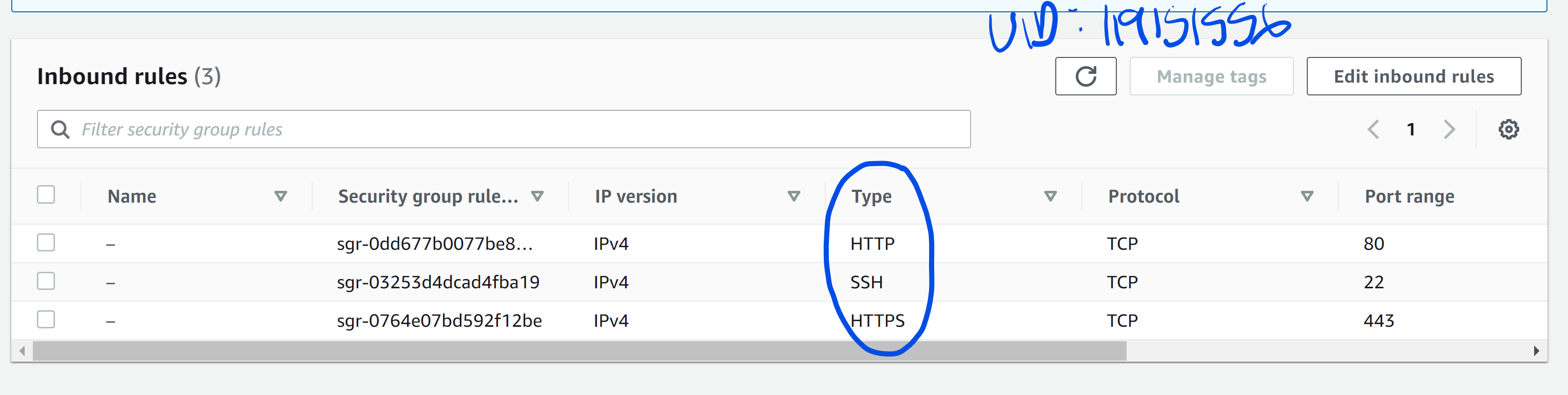
**SECURITY GROUPS**

The security groups play a major role in AWS.It is responsible for all the traffic that is allowed in the ec2 instance and all the traffic that leaves the instance.These are called inbound rules and outbound rules as shown in the diagram(1.0.1) below.In this example ,the user is sending a request to the application running on the instance and the request is of the type “http” .If the inbound rules include allowing the “http” requests to enter, then the client won’t have any problem accessing the web application on that particular ec2 instance.

fig 1.0.1

The instance I created had these three inbound rules shown in the screenshot below,these allowed inbound traffic which were http,https or ssh type at port number 80,443 and 22.

The outbound traffic was set to “all traffic”.



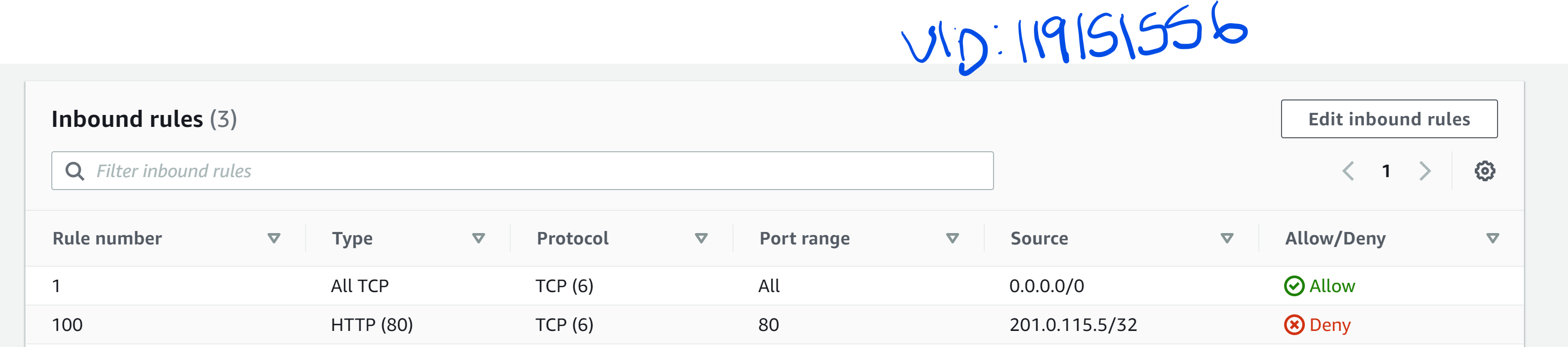
When choosing the rules in security groups for an instance,the user has all the power to change the kind of traffic that enters and exits the instance at anytime.The default ip address range is set to all(0.0.0.0).We can also restrict the range of ip addresses that can access our ec2 instance using security groups.If a ip address or a range of ip addresses seem suspicious,it can be ruled out from the addresses that can actually access the application on the instance.

This is a very important tool in the AWS cloud platform, as these days many ip addresses are suspicious and the instance can be prone to all kinds of malicious Ddos attacks coming from these suspicious ip addresses.

The security groups can be considered as a firewall to the ec2 instance.When it comes to the Cobra Kai’s application, most of the inbound traffic to the instance is http and https.These inbound traffic that comes from the client will be allowed to access our ec2 instance unless a particular ip address is restricted.**Security groups can be considered as a extra measure of security for web applications running on an ec2 instance.**

**NACL(NETWORK ACCESS CONTROL LIST)**

The network access control list is very similar to security groups,but in a broader aspect. The security groups are for the instances running on the AWS, whereas the NACL is for controlling the traffic inside of the vpc(Virtual private cloud) on the AWS. The subnet in the VPC can have multiple AWS resources like aws amplify,aws S3 or aws ec2 instance with multiple ip addresses.This subnet can be connected to the internet, databases the user is running and also other VPC’s.It has similar rules like the security groups inbound rules and the outbound rules which let traffic in and out into the aws instance,but here only to the VPC.



The screen shot above gives a idea about how the rules are set.In this the rules are for the traffic coming into the subnet,that is the inbound rules.

-Rule 1:- It allows all TCP connections coming from any address onto all ports.

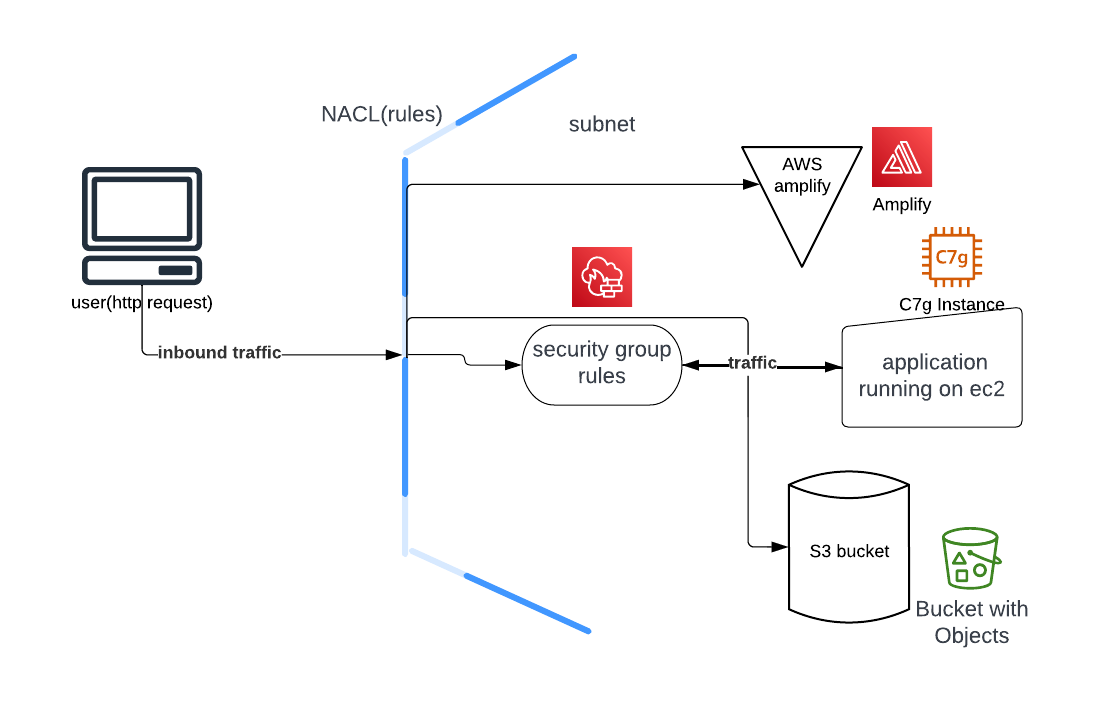
-Rule 100:-It does not allow a http connection coming from one particular ip address as shown in the screen shot above(say Daniel Larussos know IP address),maybe because it appeared suspicious in previous cases.In a similar way there can be multiple rules that can be set up for the traffic that in incoming as well as for the traffic that is outgoing.

**EXAMPLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| RULE NUMBER | TYPE | SOURCE | ALLOW/DENY |
| 1 | HTTP(80) | 214.54.0.16/23 | ALLOW |
| 23 | HTTPS(80) | 205.0.25.128 | DENY |
| 18 | ALL TCP | 198.170.25.0 | DENY |
| 15 | ALL TCP | 0.0.0.0 | ALLOW |

The following example tells us how the user can deny and allow traffic to enter the subnet according to his own understanding.

Here the last three rules are simple,but first one seems unnecessary .We see that rule 1 comes under rule 15,but why is that mentioned twice on the table.This is because,even if the user decides to remove the 15th rule later on,the http request from 214.54.0.16 should always be allowed irrespective of other rules on the table.

Fig 1.0.2

From the diagram 1.0.2 we can see the security measure is drastically increased by controlling the type of traffic that flows into the subnet as well as the traffic that goes out of it.With NACL and security groups ,the security for traffic flow couldn’t get any better.

**AWS WAF(WEB APPLICATION FIREWALL)**

The AWS WAF is a firewall for all web applications that are running on the AWS,this is a paid entity inside of AWS.This adds more to the present security methods that protect the web applications that are on the AWS.The firewall is deployed mostly along with AWS cloudfront which distributes the content(data) faster on to the internet.

In the AWS WAF, all the requests that are coming to the subnet will be monitored closely and will drop the packets which seem suspicious.This totally depends on how the user has configured the firewall manager.Some of the functions can be automated as well.This not only adds to the present security that are already in place(security groups and NACL),it also helps preventing against any kind of attack on the web application.

The updates in the WAF are really quick.The WAF has a bunch of rules that help the user inspect every part of the request that comes from the web with minimal effect on latency.The AWS WAF is very sophisticated ,it allows you to inspect every header of the packet that is coming in through the firewall or leaving the subnet.

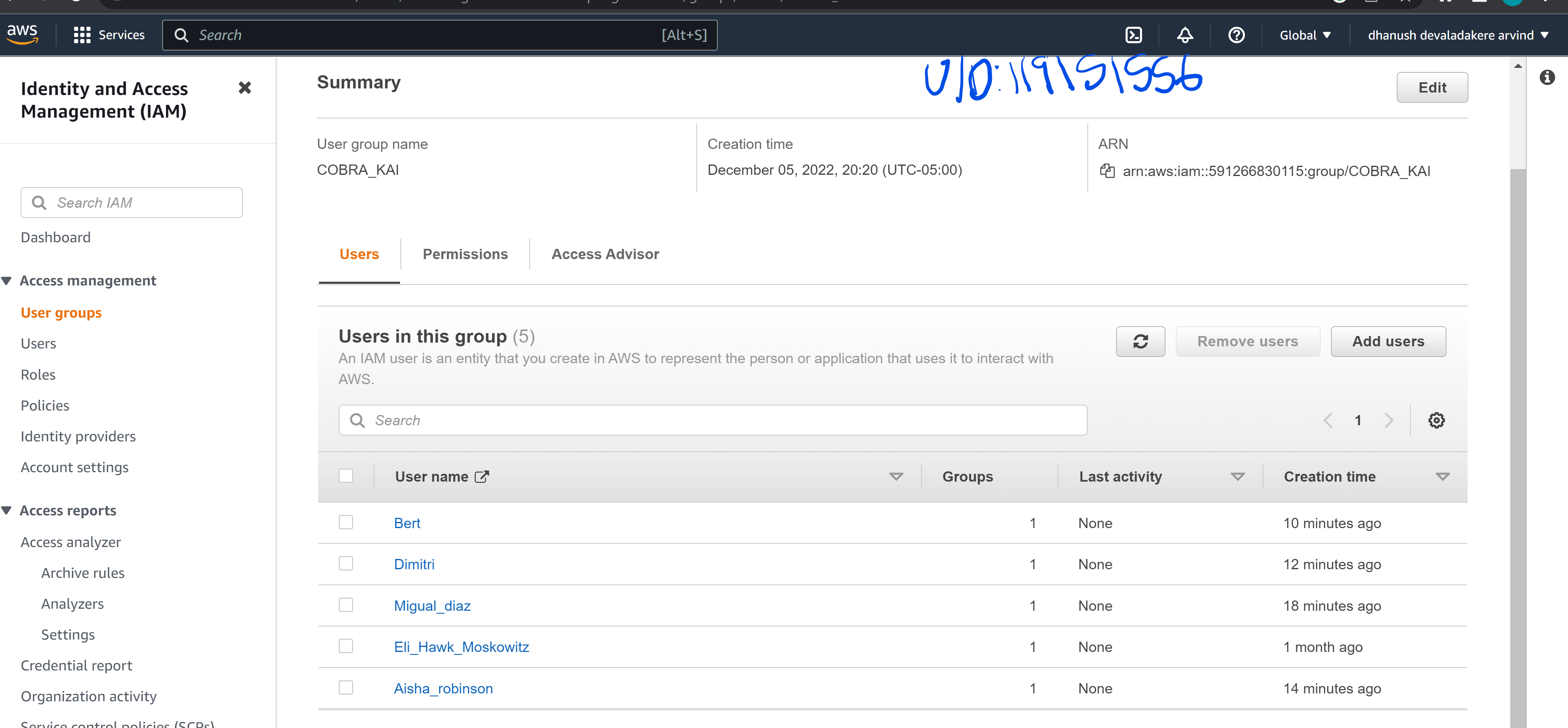
The AWS firewall manager not only helps to configure rules related to firewall with one account but multiple accounts at the same time.

**IAM(IDENTITY AND ACCESS MANAGEMENT)**

Identity and access management is vital when it comes to running a massive web application on the cloud.In simple terms IAM is where individual personal in an organization or a company is given certain permissions to access different parts of the AWS tools where their application is running.This could be to make changes,modify certain restrictions or a bunch of operations a person can make with the limited access they have in the organization.

When it comes to the Cobra Kai application. A user group is created with a bunch of users who have limited access i.e, have access only in their area of expertise.The users created in our case are:-

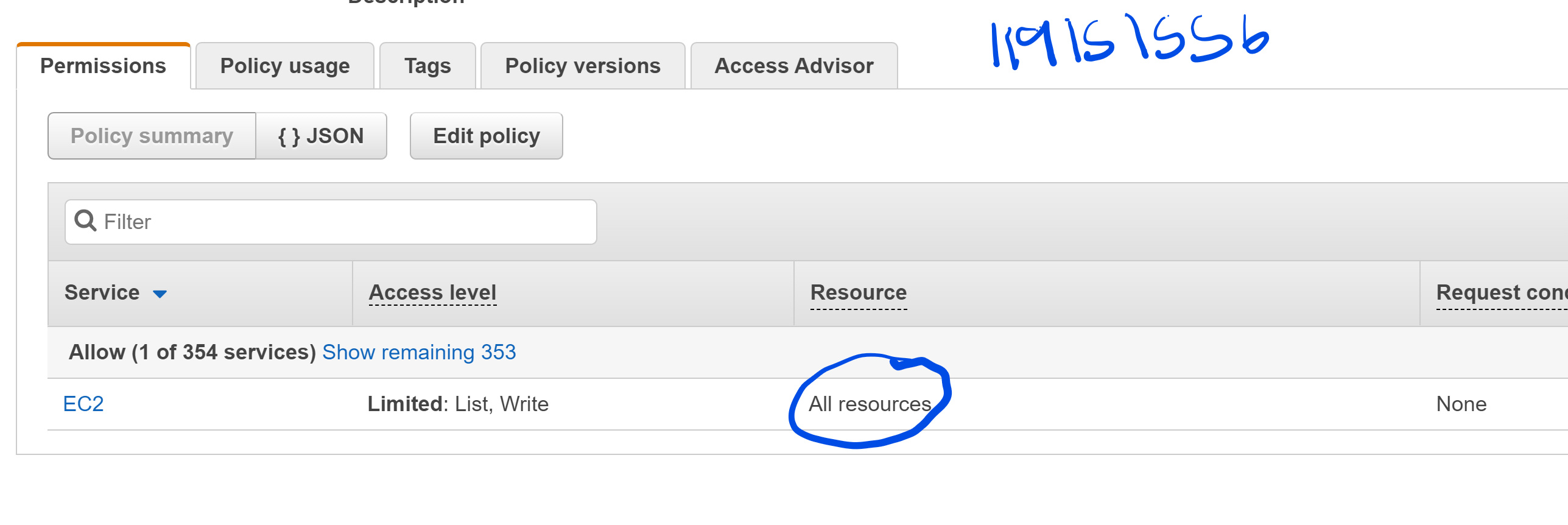
* Eli “Hawk” Moskowitz – Chief Information Officer.
* Miguel Diaz – Chief Operating Officer.
* Aisha Robinson – Chief Information Security Officer.
* Demetri – Web developer.
* Bert - System administrator.



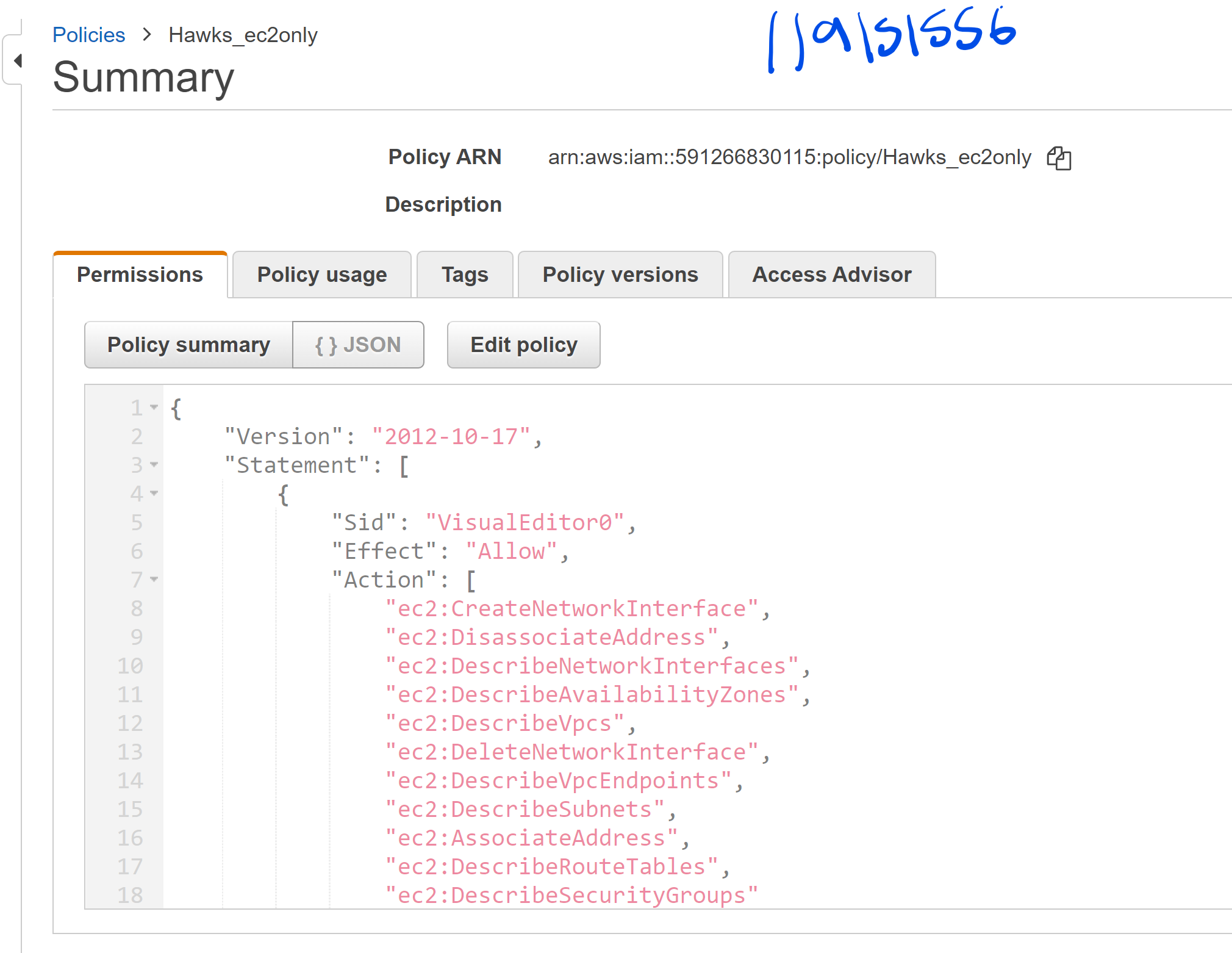
The user group created contains all the names that are mentioned above with different policies attached to them.

**Example 1:**

The screen shot below shows the custom policy that is attached to the user “Eli\_Hawk\_Moskowitz”. This gives him complete access to the EC2 instance where the web application is running.In this example ,he would be allowed to change security group policies only with regards to the EC2 instance.

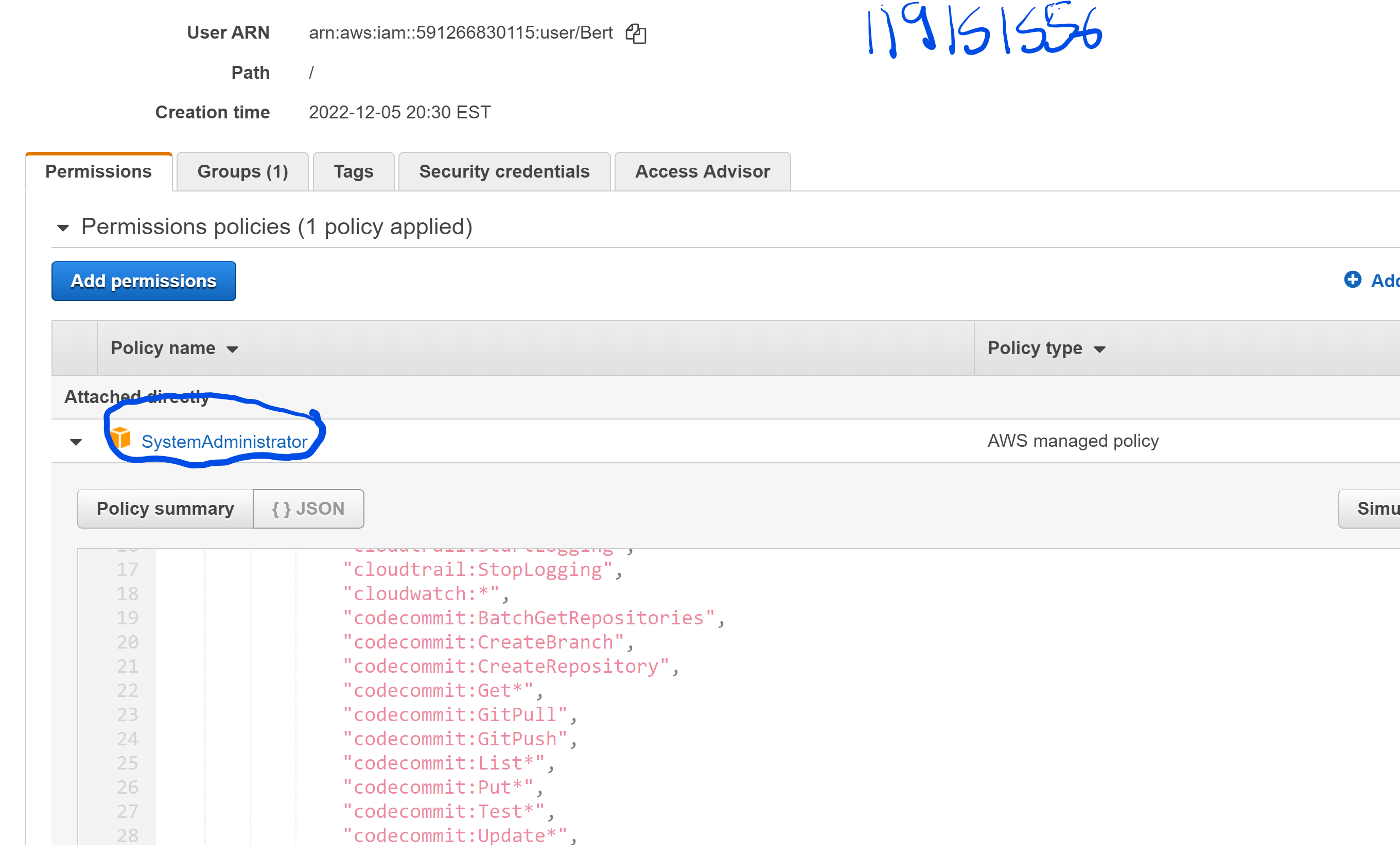


The next screenshot shows the JSON file with lists all the permission the user has and could access in the AWS.



**Example 2:**

Let’s take another example,that is “Bert”.His role in the organization is to be a system administrator.The policy attacked to him is shown in the screenshot below.



The role as a system administrator gives him access to multiple resources in the AWS cloud.The policy attached is a build in “already existing” policy which is named as “SystemAdministrator”.

Using the IAM tool in AWS, the owner can give his/her teammates access to number of resources and also can restrict them from accessing tools they are not supposed to.This itself is a security measure on its own.

**CREDIT CARD PROCESSING**

When it comes to credit card processing,we need high computational power for fast and reliable transactions from the customers.There are multiple ways to speed up processing with different kinds of EC2 instances with varieties of configurations.Out of all these instances it is important to decide which instance provides the best computational power , as well as fast processing. The instance we should be looking for is under these two sections:

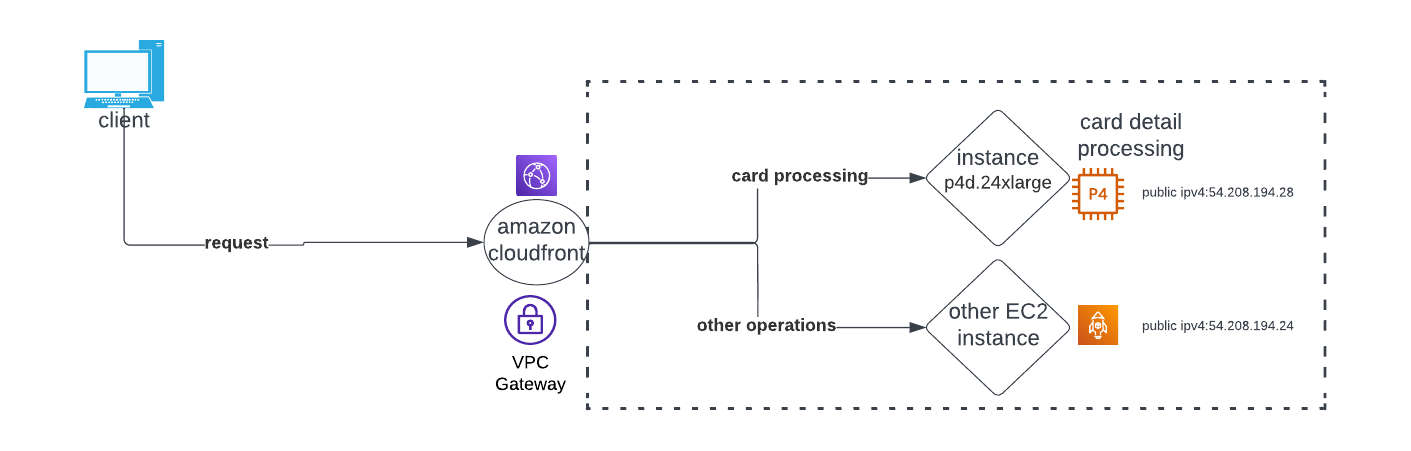
* Accelerated computing.
* Computing optimized.

In each of these categories,there are multiple instances with different configurations.Some of them are the latest versions, and it is advisable to choose from the latest versions.

**Accelerated computing:-**

Let’s take the latest instance in this category, which is the **P4.**

* GPU-8.
* vCPUs-96.
* GPU Memory-320 GB.
* Network Bandwidth-400 ENA and 400 EFA.
* GPU peer to peer-600 GB/s

These configurations are enough to process multiple credit cards at a time.This was one of the major setbacks when it came to the previous model of the Cobra Kai application.This also comes with a cost,and should be prioritized with regards to the budget. Fig 1.0.3

**CLOUDFORMATION TEMPLATES AND AWS SHIELD ADVANCED(EXTRA SECURITY)**

Cloudformation templates is the last step of the implementation.After migrating all the parts of the web application on to the cloud.By using the cloud formation templates, we can see all the resources that are running on the AWS and also see how all of them are related to get an idea of how the application works on the cloud.The user doesn’t have to go through each and every configuration of the individual resource,the cloud formation templates will take care of that.

The cloud formation templates can be used to create a cloud formation stack.This is nothing but all the resources used by the application on the cloud,which are put together as a stack.These are user friendly to create and also can be deleted easily, once the setting up is done or whenever the user chooses to delete them.This lets the user to focus more on the web application and not on the tools that are used to run them.The basic idea of the cloud formation is to make all the resources that help the user run the application to appear as a single unit using the stack.

**AWS SHIELD ADVANCED**

The AWS shield advanced is the upgrade when compared to the standard AWS shield which comes at a no cost for all the users that are using the AWS cloud.This obviously has added perks to it ,since it’s a paid entity in the AWS cloud.

The AWS shield standard version only protects the application on the layer 3 and layer 4 part of the infrastructure. Whereas the AWS shield advanced protects the application from sophisticated Ddos attacks from layer 3 to layer 7 of the application.Since Cobra Kai was vulnerable to a lot of Ddos attacks since the beginning, it is recommended to shift from the standard version of the AWS shield to the advanced version which is very helpful, as security is our major concern when it comes to this application.

One of the perks of the AWS shield advanced is that it gives you access to AWS SRT(AWS shield response team ) 24/7.It monitors for threats and gives the user real time visibility about the attacks that’s taken place and the ones that might happen.AWS shield advanced gives a very tailored protection to the application along with the security group rules, NACL configurations and also the AWS WAF.

**REDESIGNED ARCHITECTURE OF THE COBRA KAI APPLICATION**

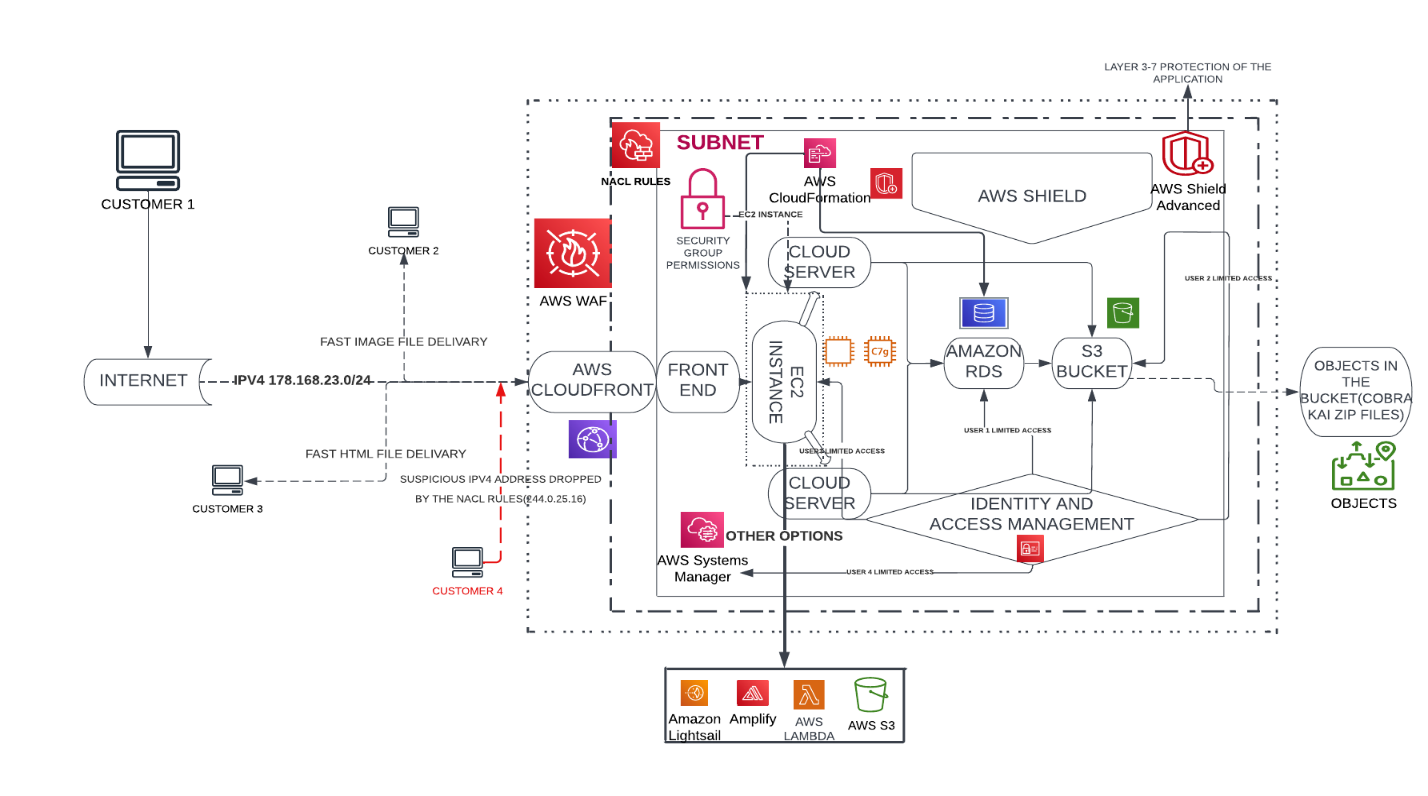


Fig 1.0.4

This architecture (fig 1.0.4) gives the cobra kai application enhanced security measures when compared to the last one. (NOTE- The above architecture is strictly based on the Cobra Kai’s application that I created.New tools and entities that I mentioned above like NACL,security groups, AWS WAF,AWS shield advanced and more IAM roles have been implemented).