Manage Modern Applications Using AWS Fargate

Abburi Venkata Praveen

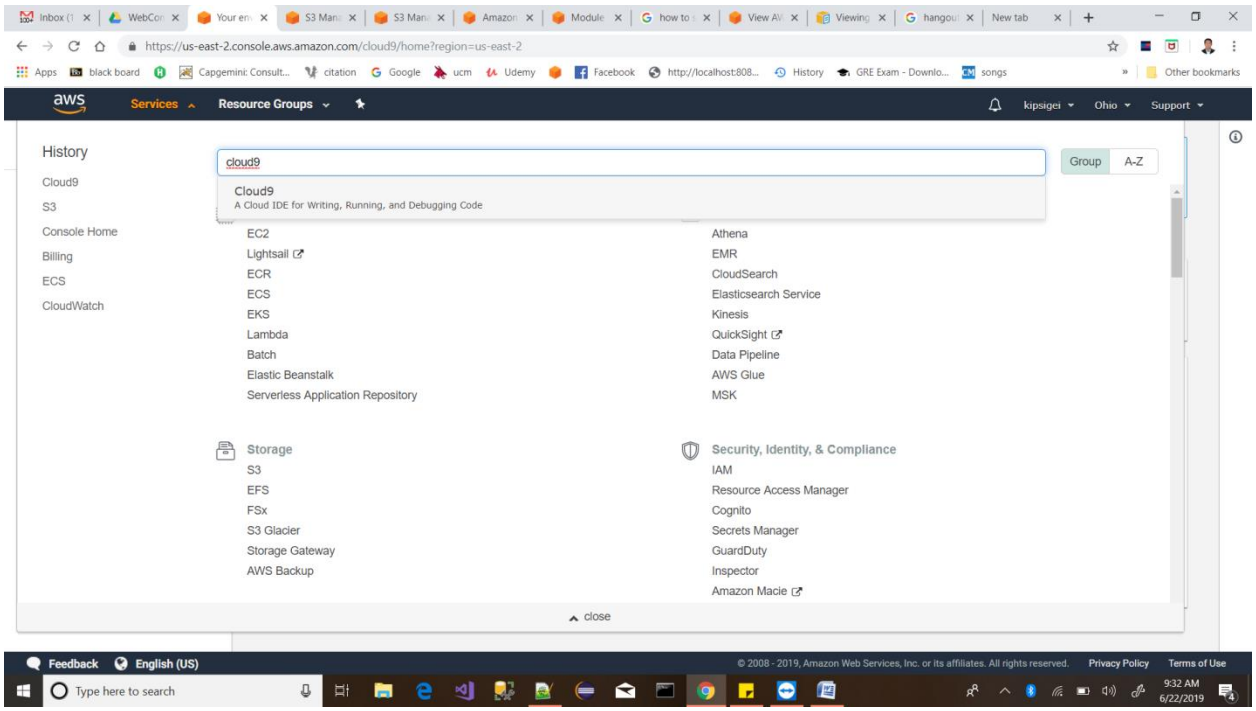
Vxa91760@ucmo.edu

700689176

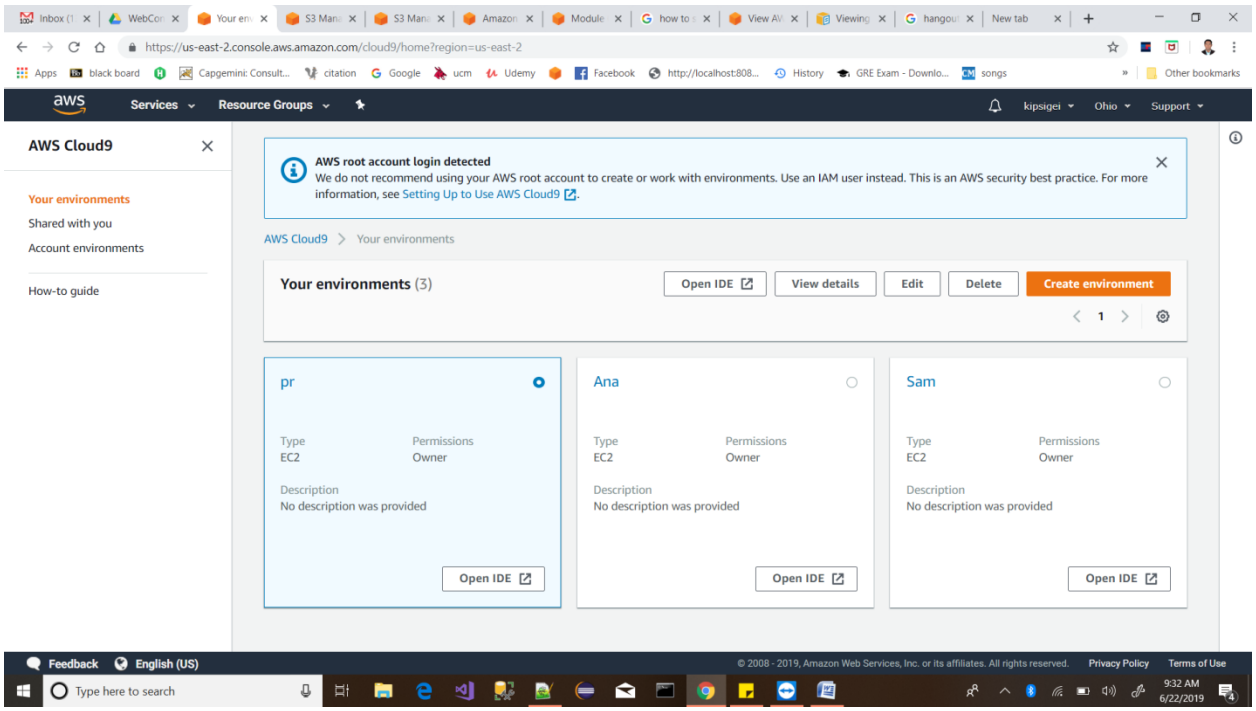
Step-1:

We need to setup a Cloud 9 IDE.

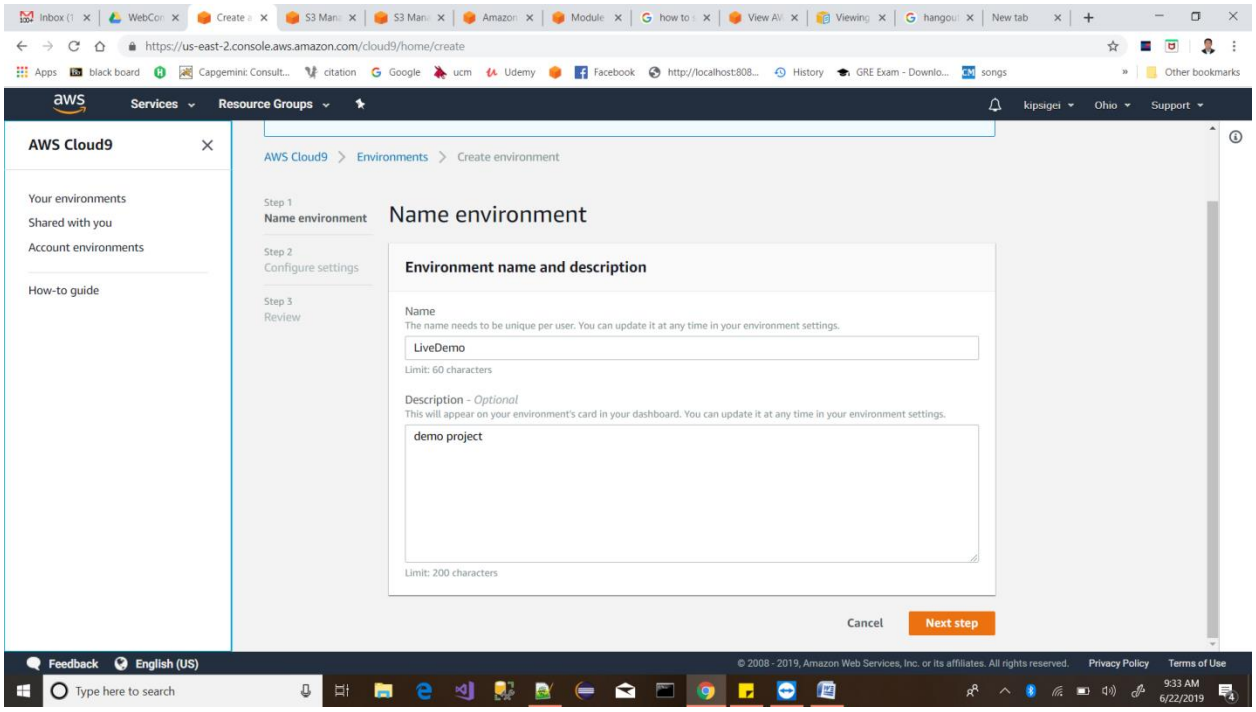
For the cloud 9 IDE we need to search in services and it will be displayed as below



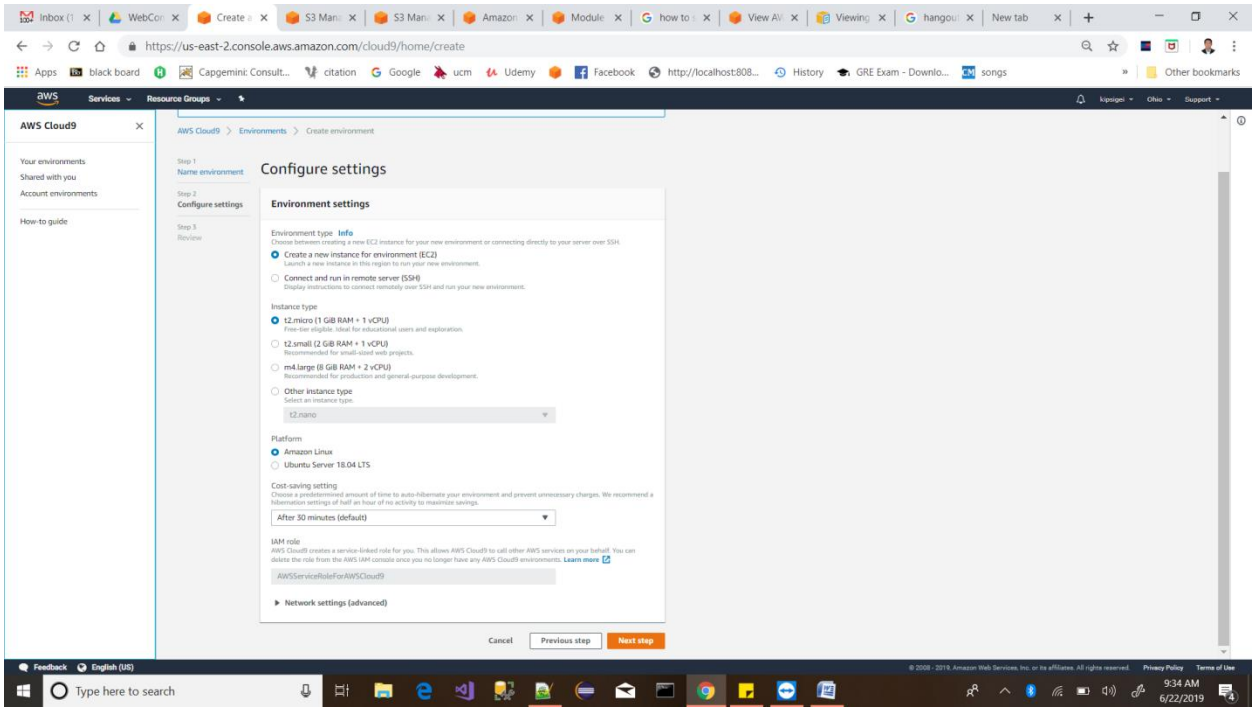
Step 2: Once we click on cloud9 IDE need to create an environment by selecting create environment. Here we can view the existing IDE's which we have created for testing purpose.



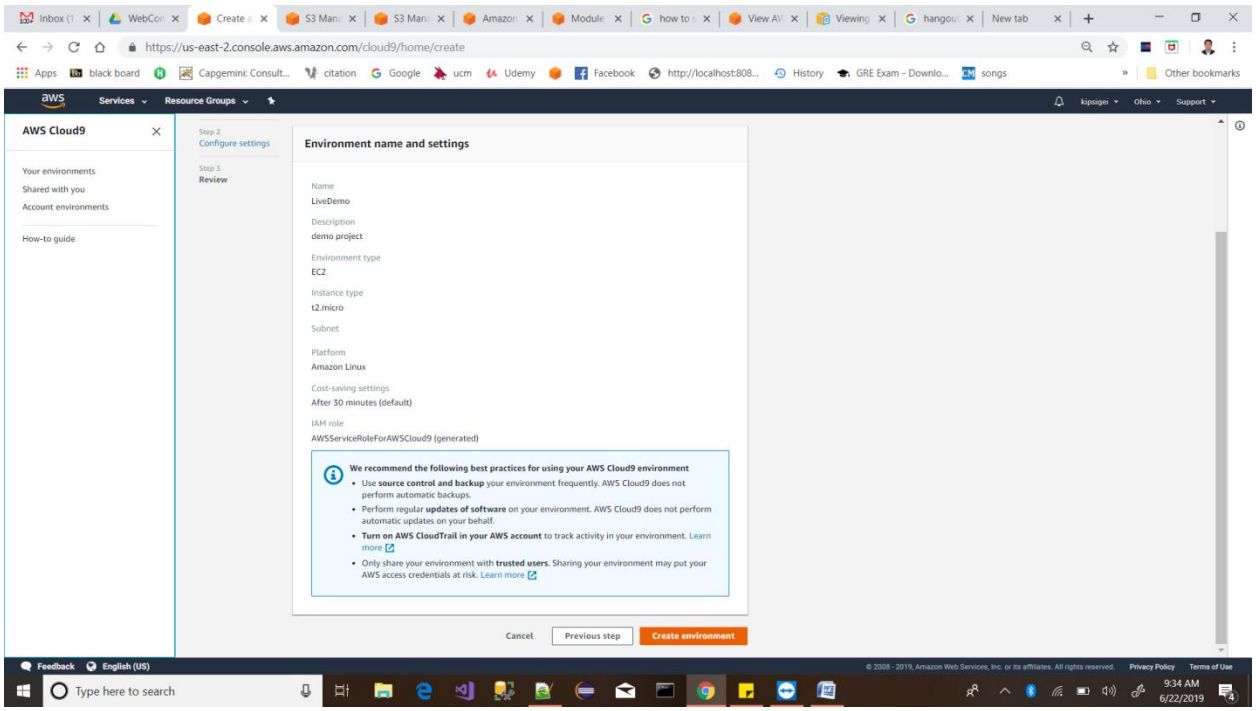
Step 3: Here we give the name of the environment



Step 4: Here we have the configuration ready for IDE.

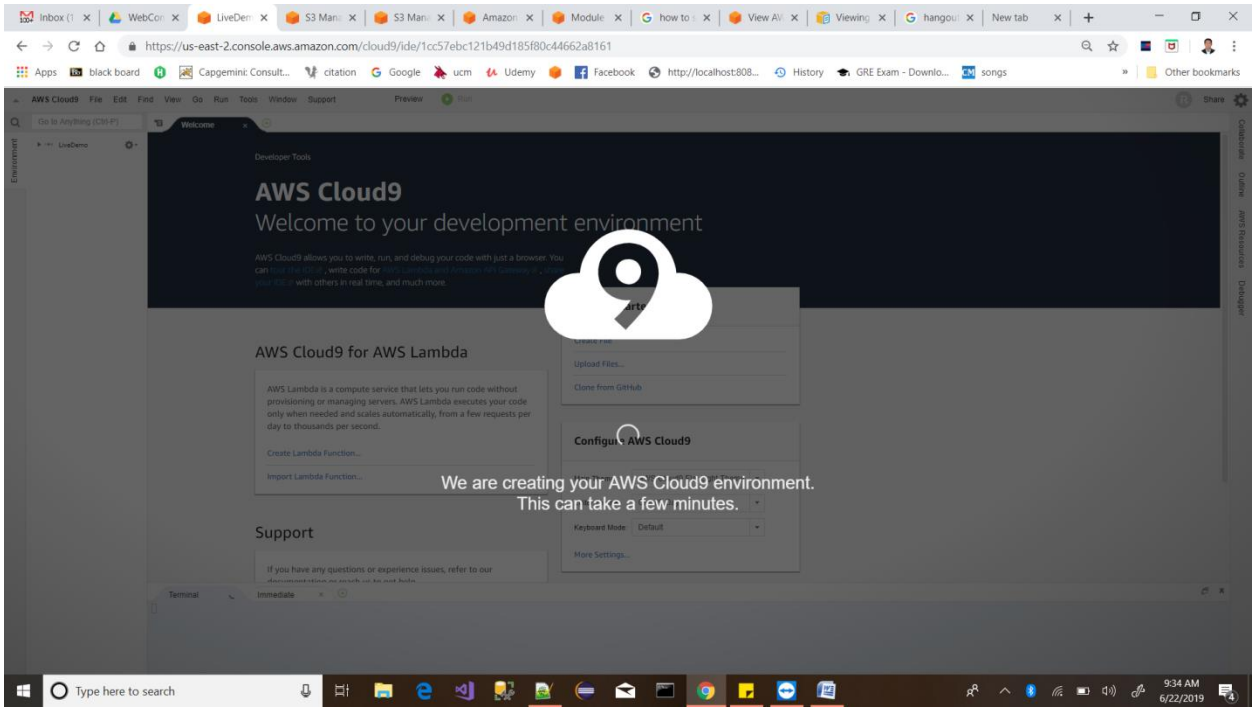


Here is the review of IDE Environment.

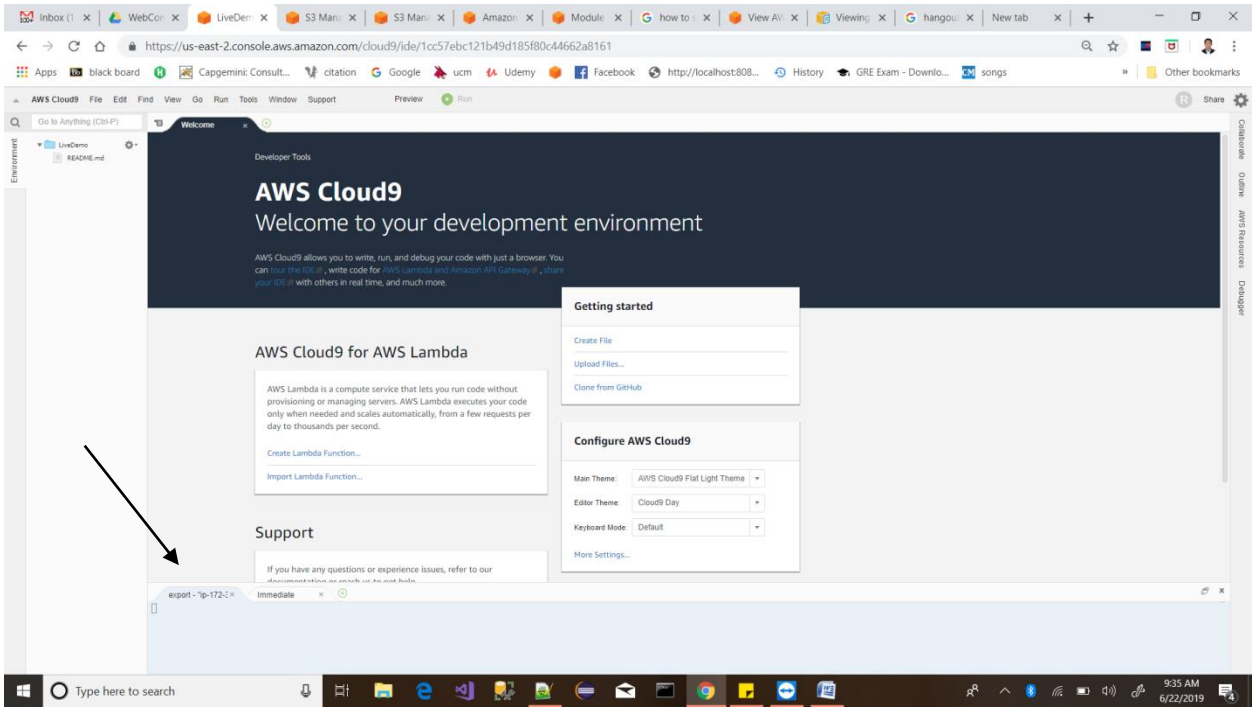


Step 5: continue to creation of IDE. This shows the environment creation step.

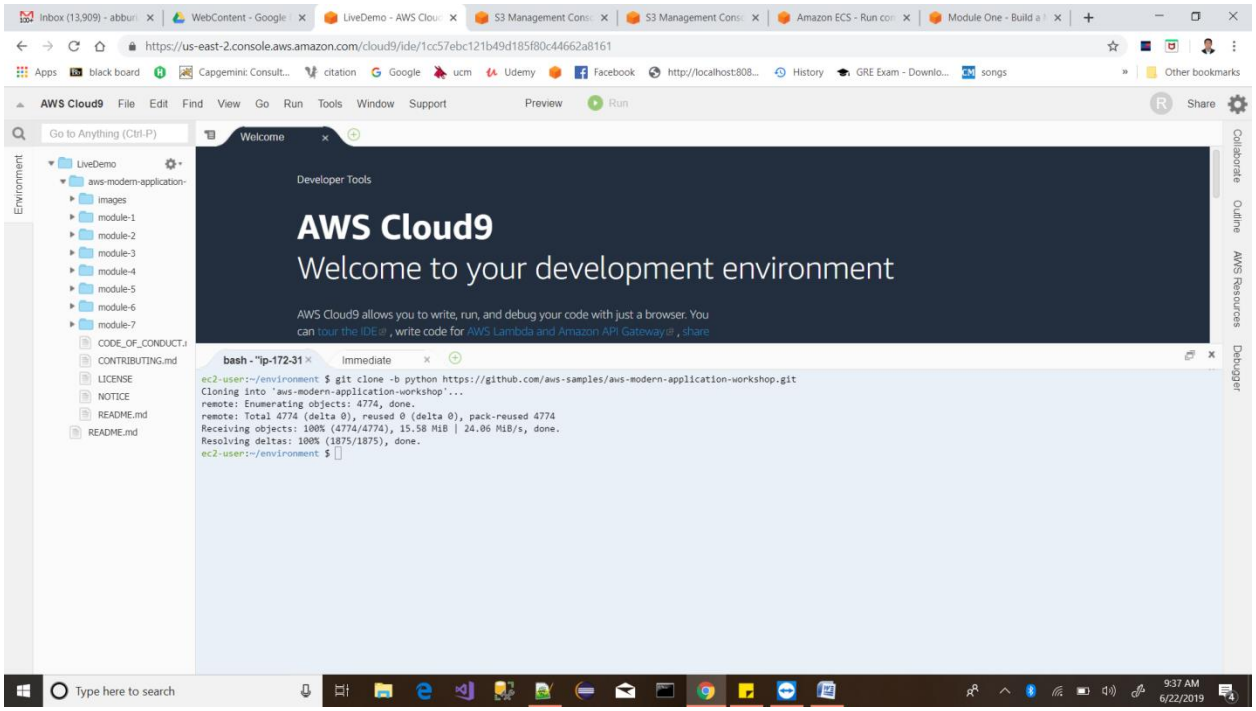
Usually it will take 3 minutes to set up the IDE.



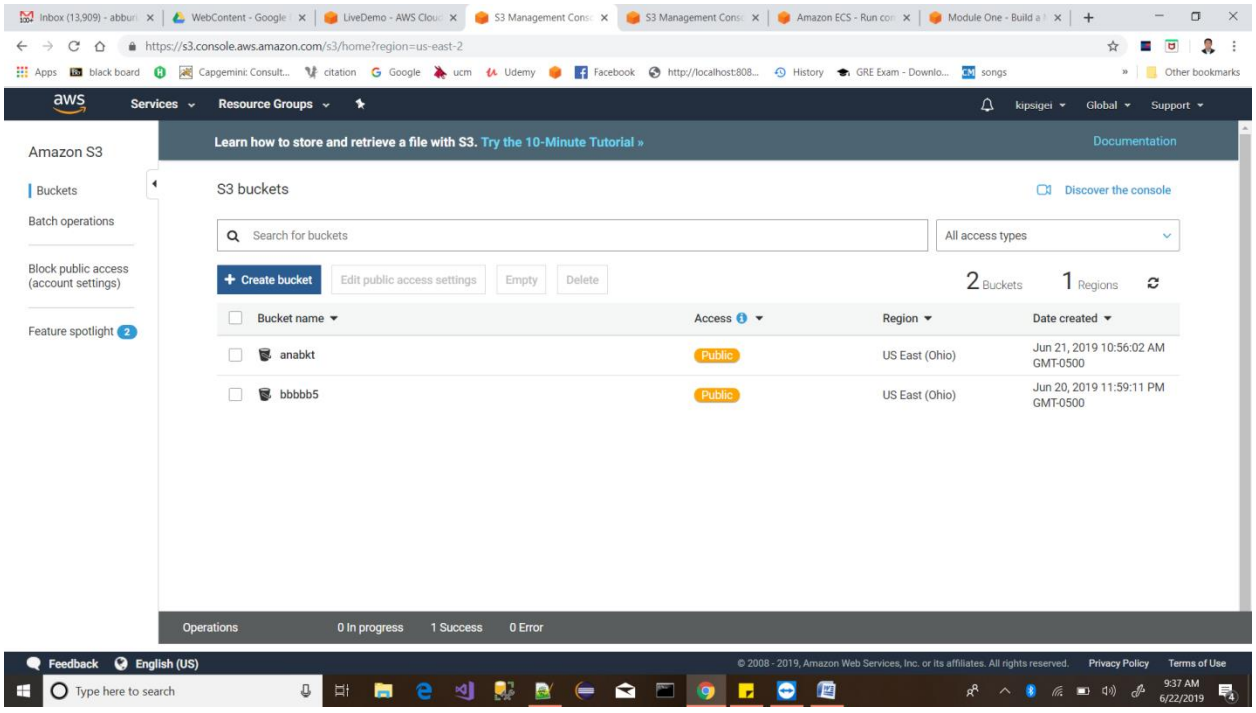
Step 6: Here is the cloud9 development Environment. The down arrow represent the command line interface for the cloud9 environment.



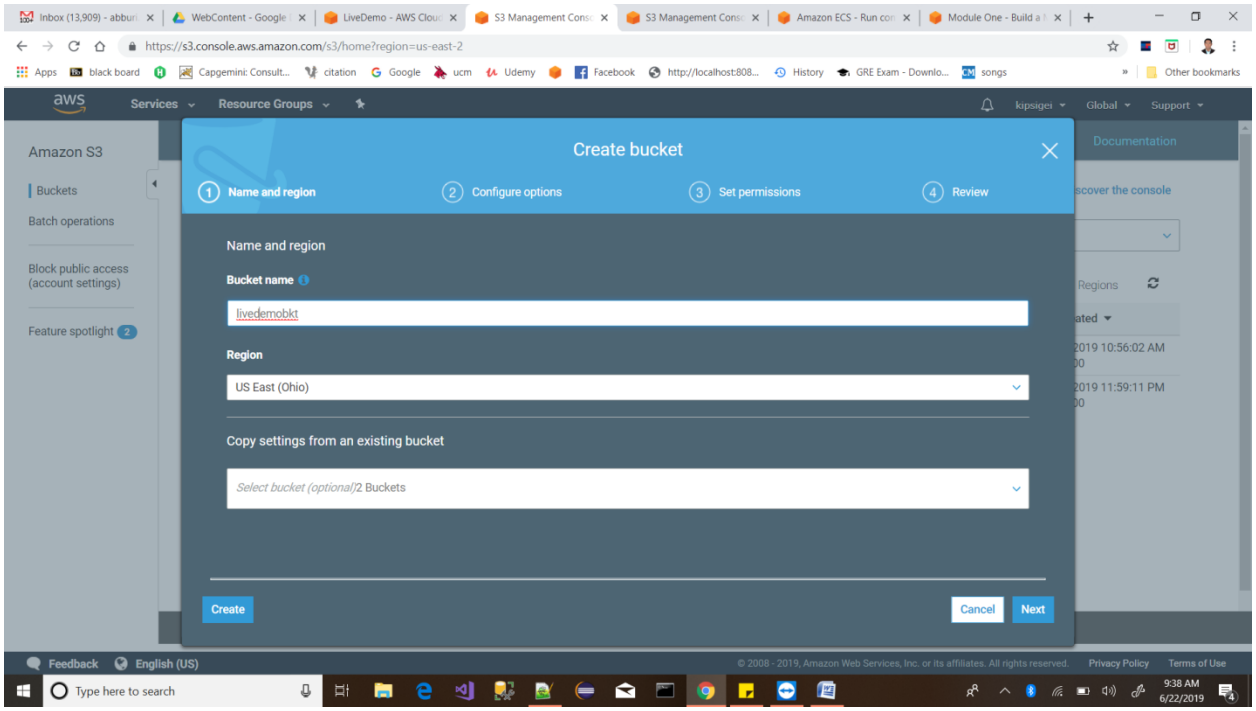
Step 7: First we will deploy the static website on s3 bucket for this one we are downloading the static web application package from github repository.



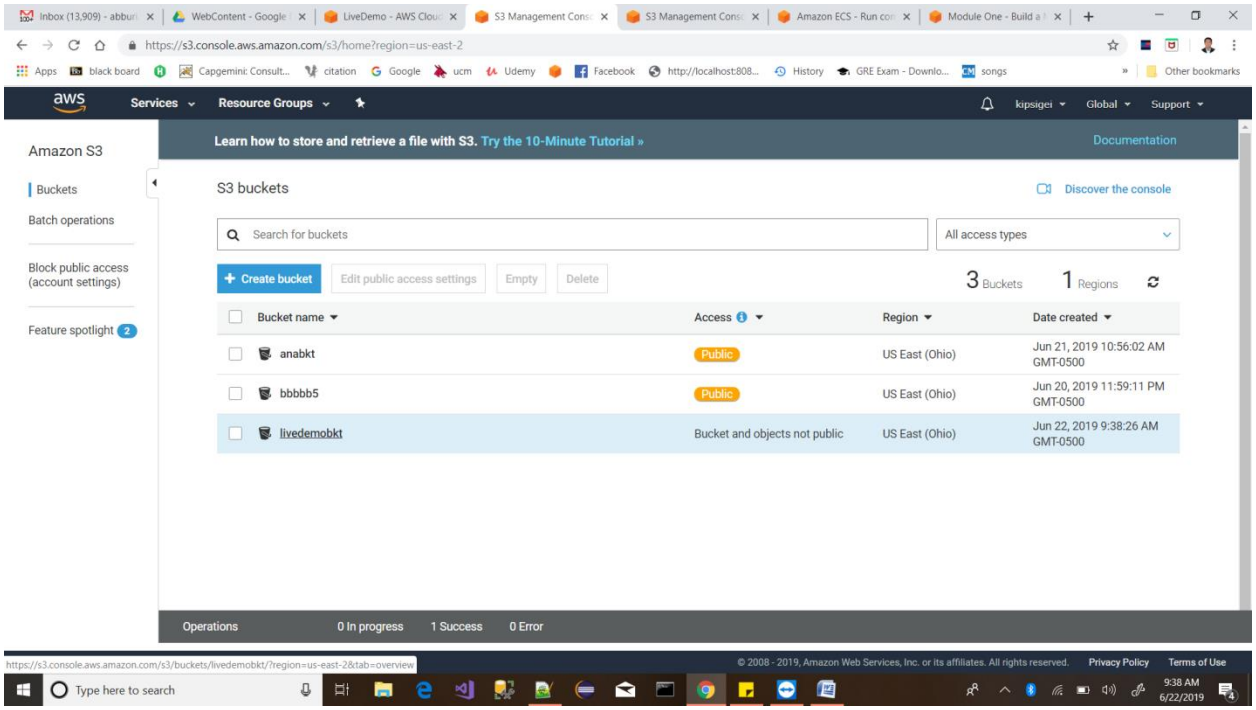
Step 8: Here we create a s3 bucket to store the application files and to host the files. Click on create bucket.



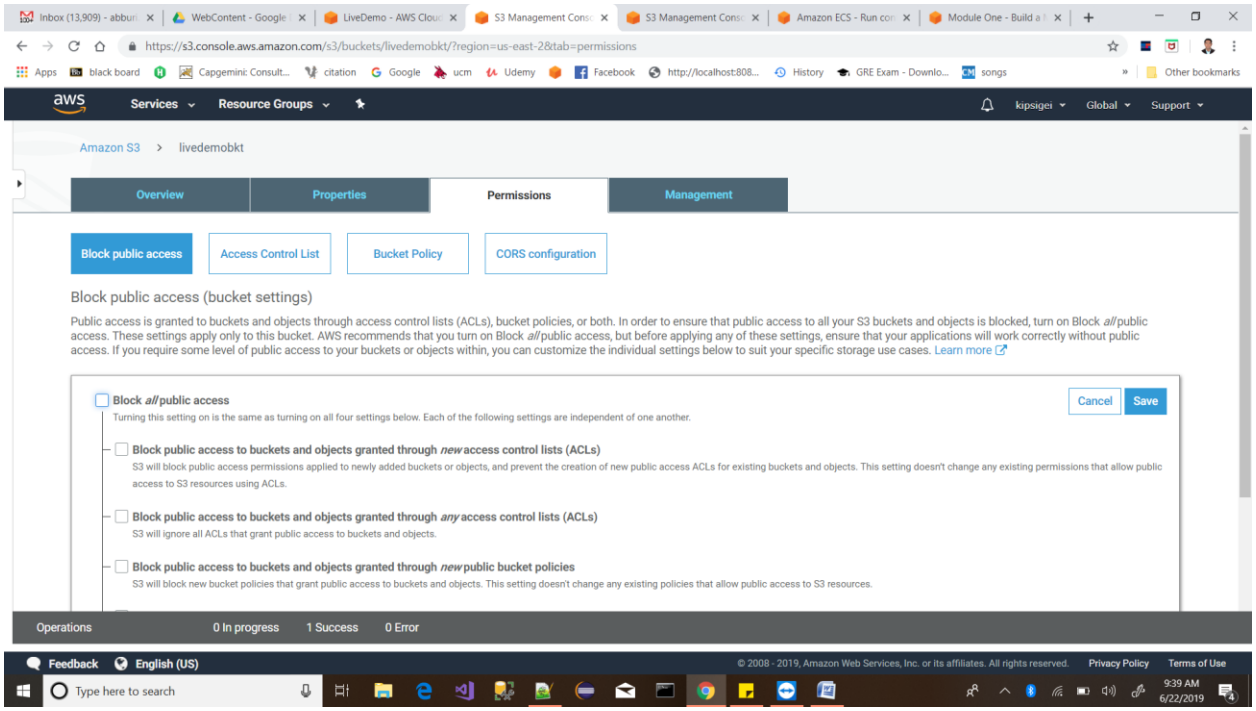
Step 9: Here we name the bucket as (Here I am giving **livedemobkt**) and click on create button. Here the bucket name is unique.



Here the s3 bucket has created.

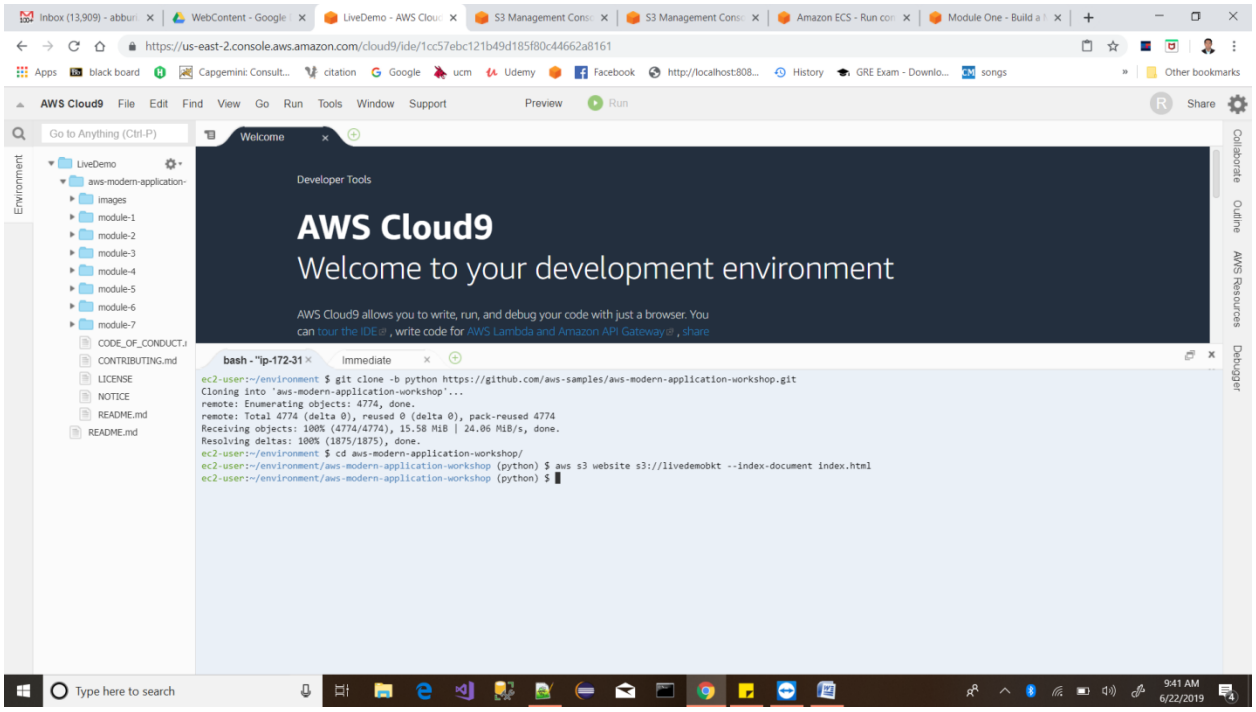


Step 10: We need to change the permission of the bucket to access publicly and to connect to cloud 9 IDE.

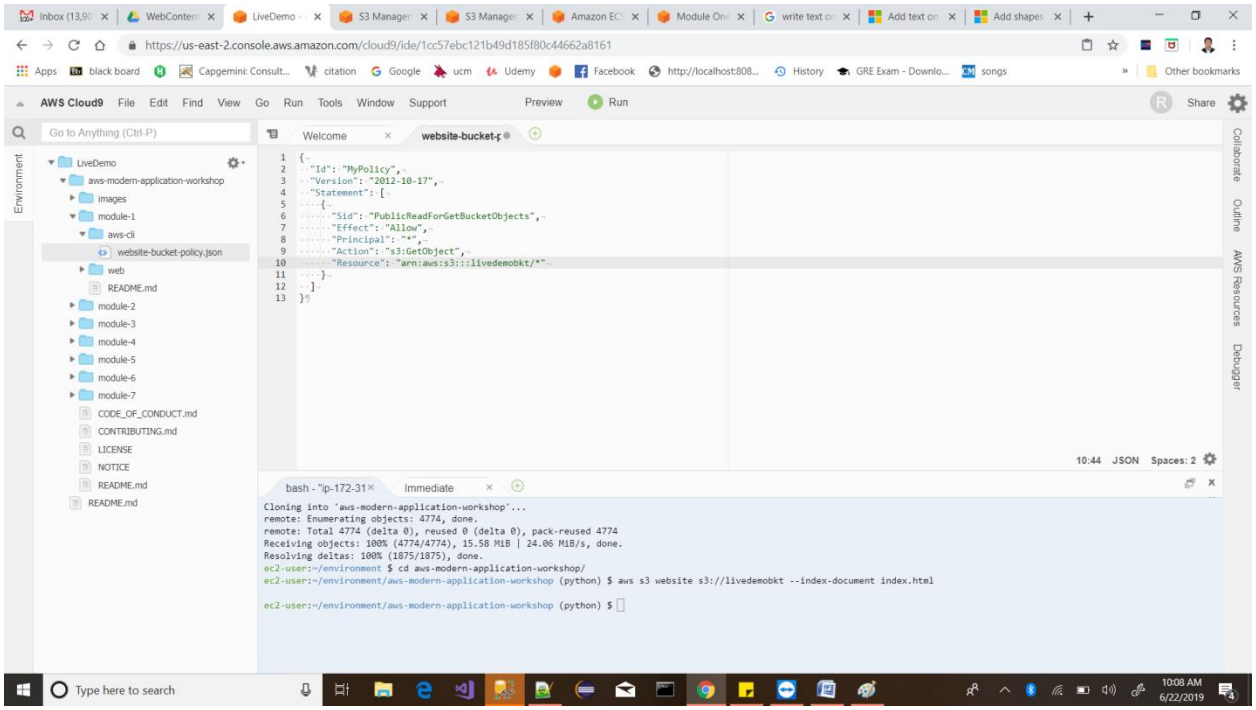


Step 11: To enable the bucket website hosting we set configuration option by running the following command as shown below.

aws s3 website s3://livedemobkt --index-document index.html

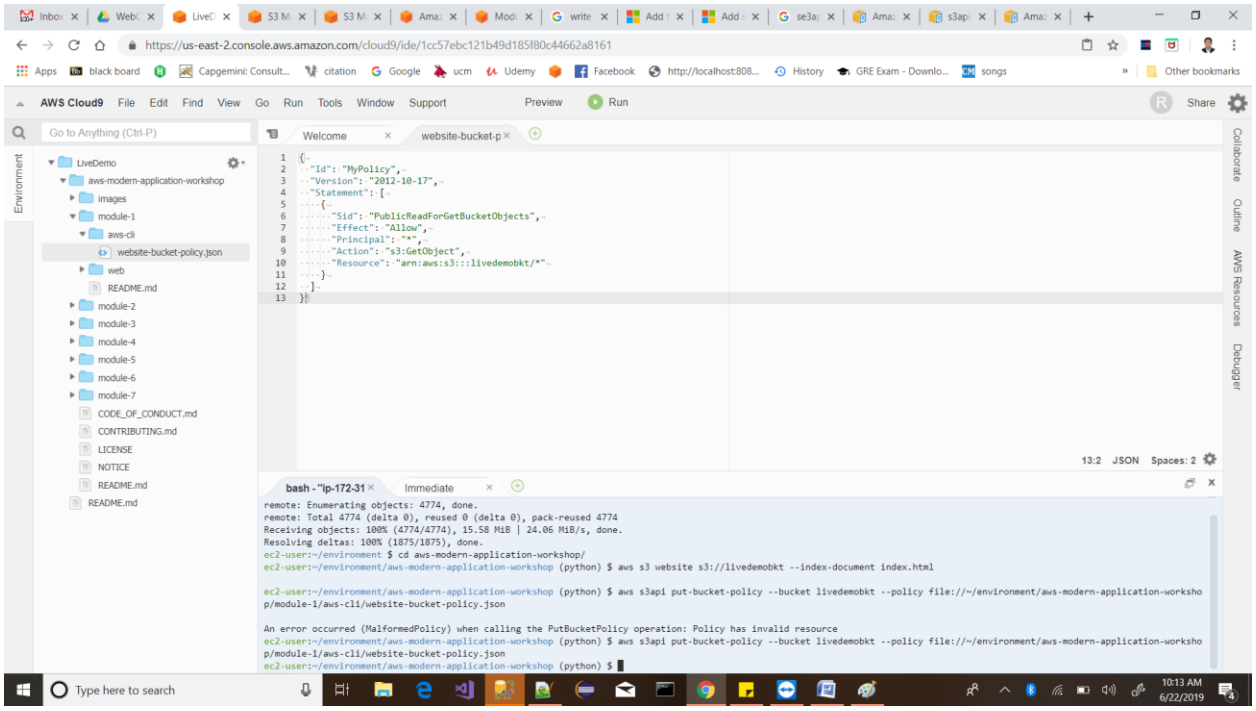


Step 13: The Policy Json file looks like below with the bucket name



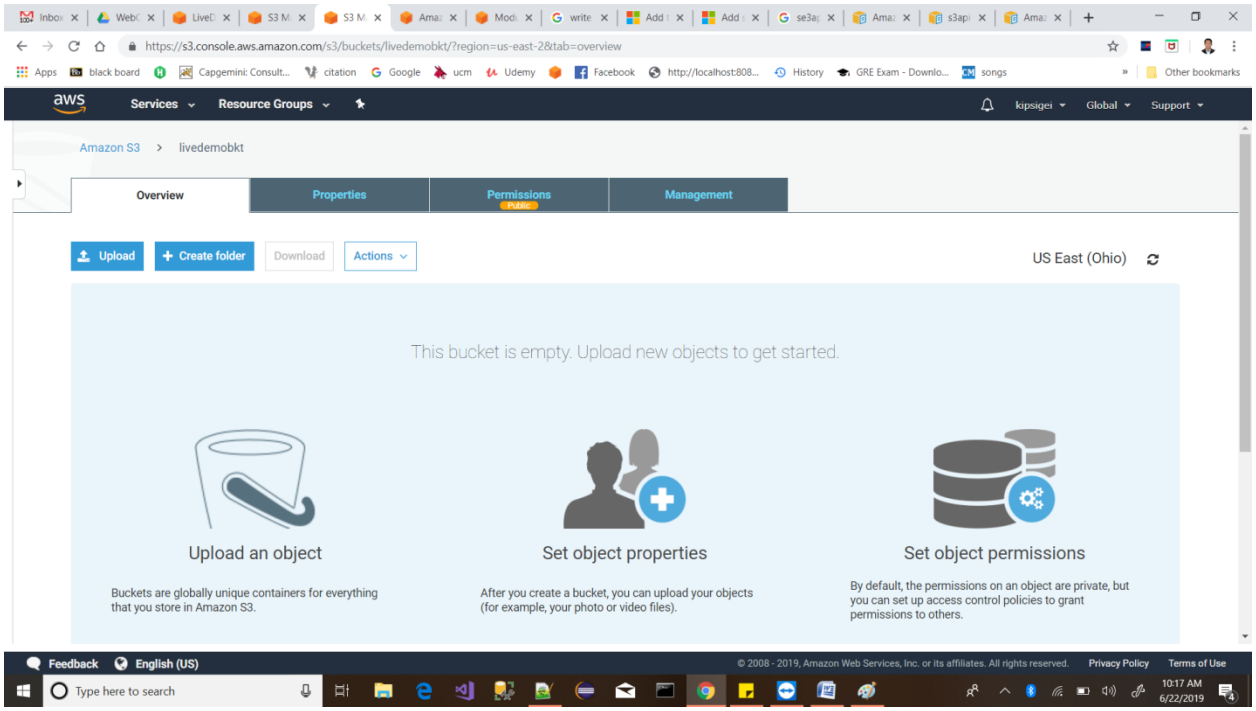
Step 14: Now we need to add bucket policy to our website.

aws s3api put-bucket-policy --bucket livedemobkt--policy file://~/environment/aws-modern-application-workshop/module-1/aws-cli/website-bucket-policy.json



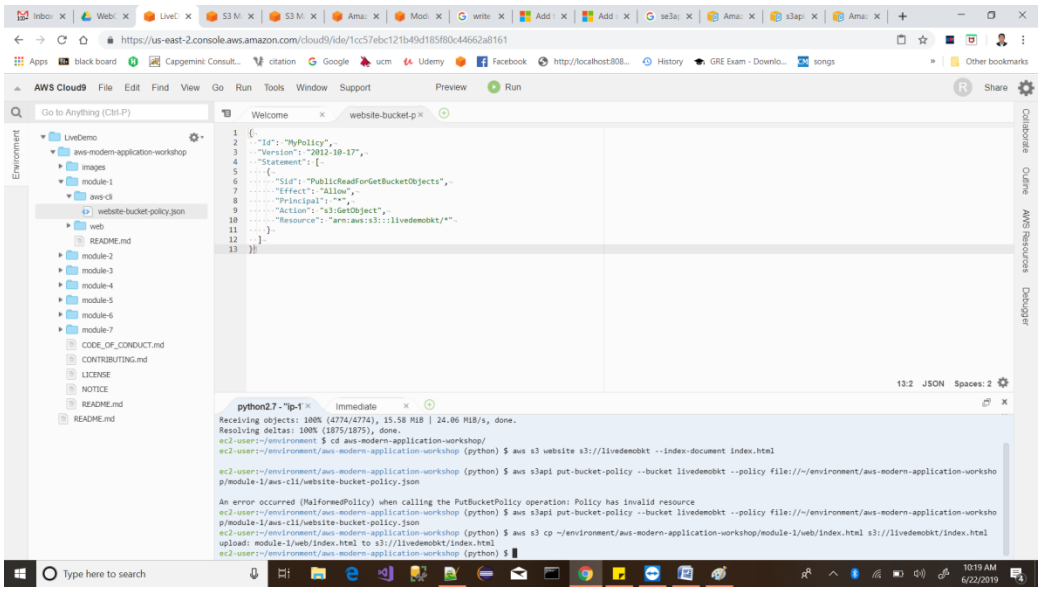
Step 15: Now we need copy the website content from cloud9 IDE to s3 bucket to access static application in s3 bucket.

If we observe the bucket files initially empty.

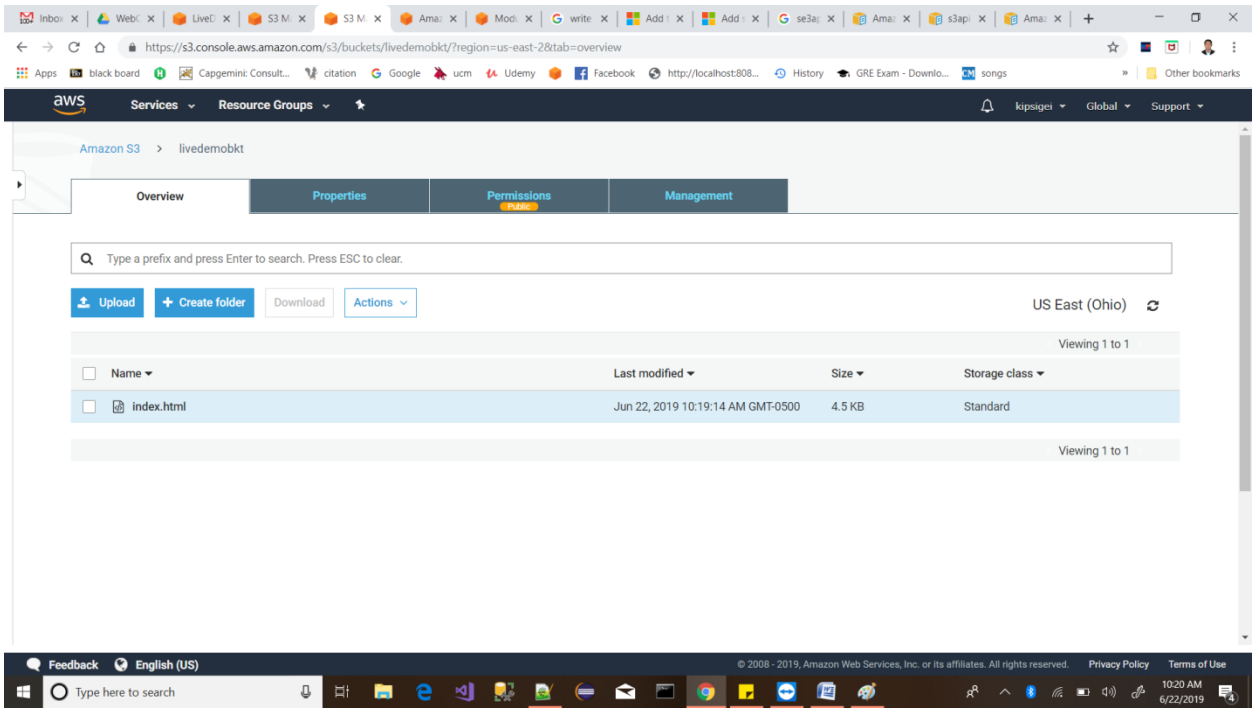


Step 16: Now we are copying the files from IDE to S3 bucket by running following command.

aws s3 cp~/environment/aws-modern-application-workshop/module-1/web/index.html s3://livedemobkt/index.html



Step 17: Now refresh the s3 bucket we can view the index.html file which we copied from the IDE.



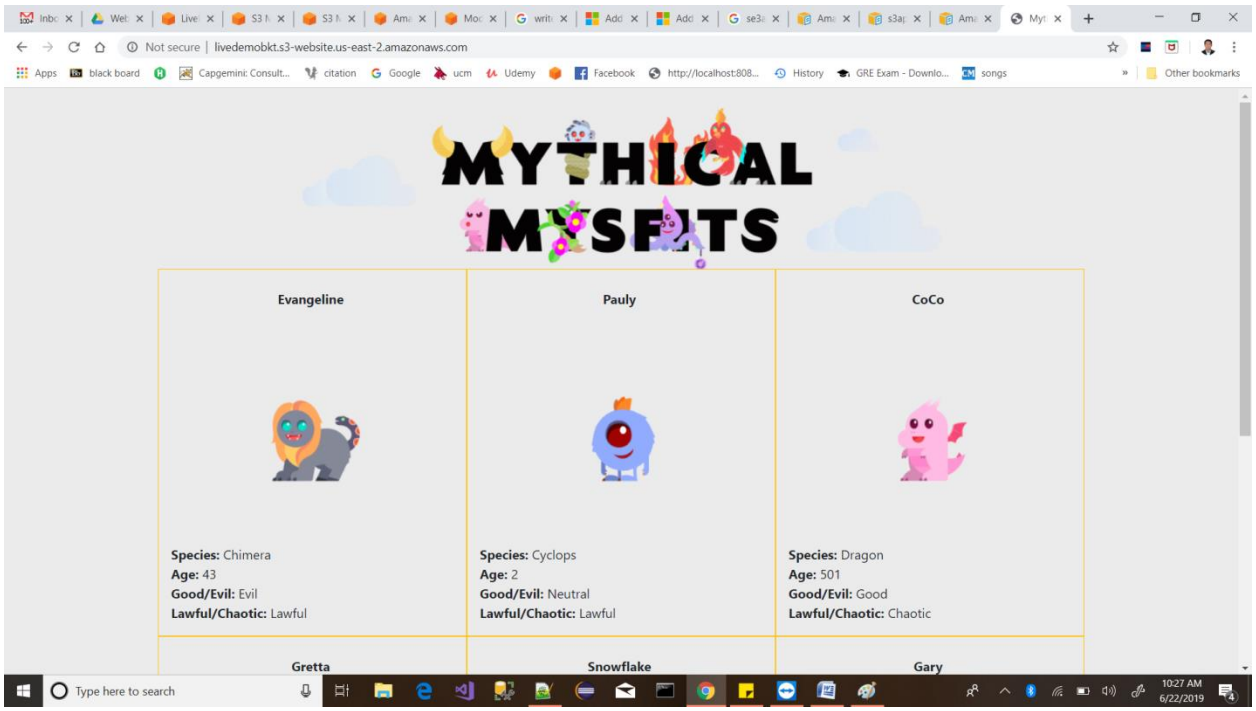
Step 18: Now we can run the application from s3 bucket.

Here we are using ohio (us-east-2) region

after replacing the bucket name and region from above above URL:

http://livedemobkt.s3-website.us-east-2.amazonaws.com

If we run the above URL in browser ,view the website as below.



Step 19: Now we are going to host a microservice using AWS Fargateserver.Before create a service ,we need setup an infrastructure environment with cloudFormationto build network infrastructure and vpc,iam roles forsecurity,NAT gateways.

Here will create a stack in cloud formation . The stack is collection of resources can managed as a single resources. by creating stack it will internally create & mange VPC,NAT gate ways,iam roles.

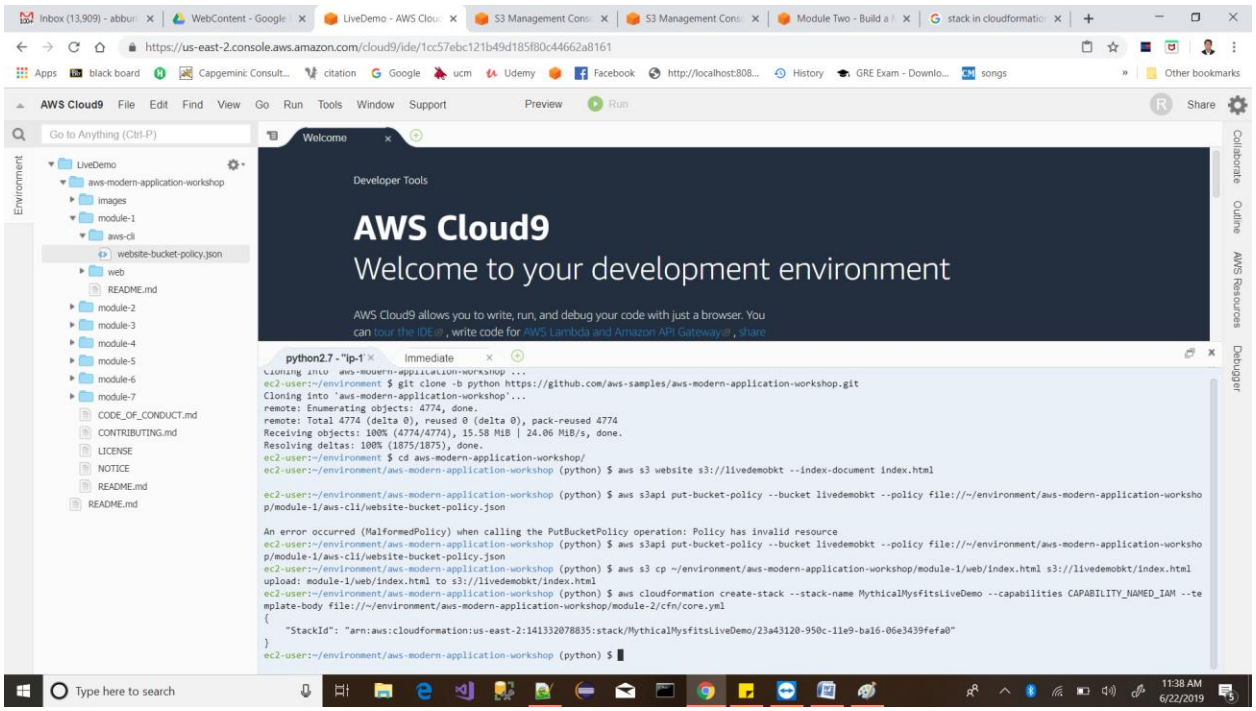
If we run the following command to create stack :

awscloudformation create-stack --stack-name MythicalMysfitsLiveDemo--capabilities CAPABILITY_NAMED_IAM --template-body file://~/environment/aws-modern-application-workshop/module-2/cfn/core.yml

Usually the creation of stack will take 10 minutes.

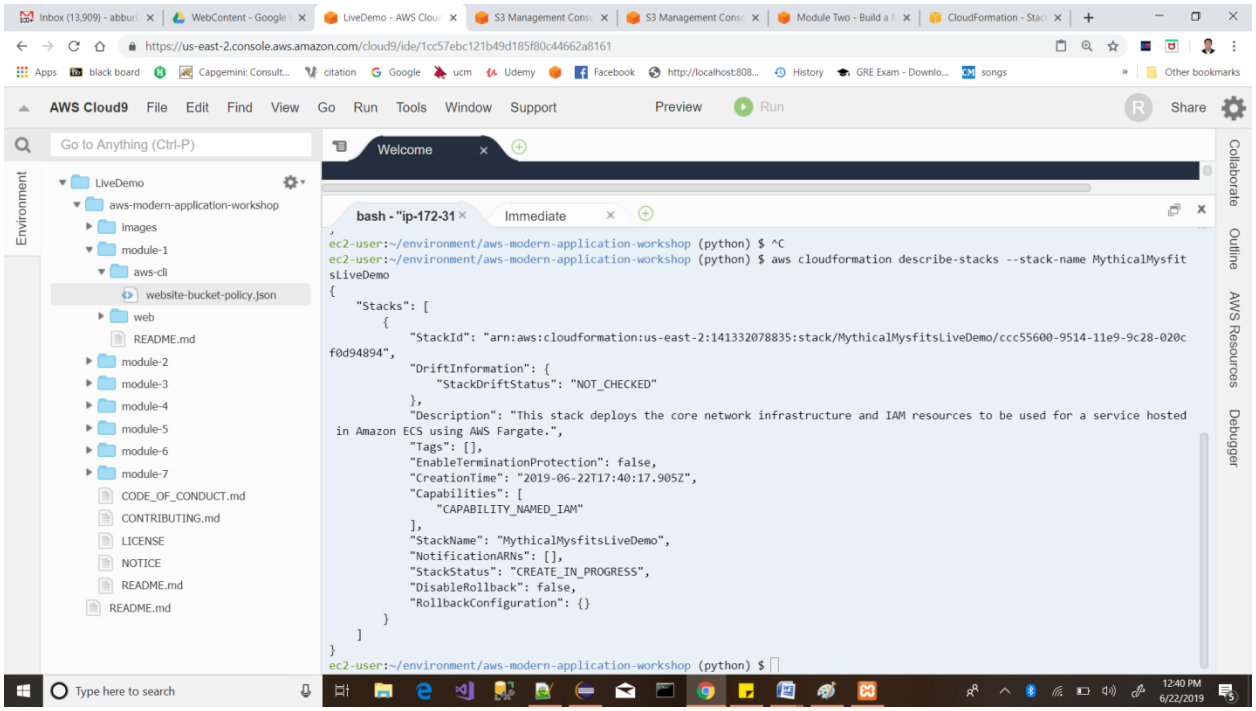
We can view the status of creation of stack.

awscloudformation describe-stacks --stack-name MythicalMysfitsLiveDemo

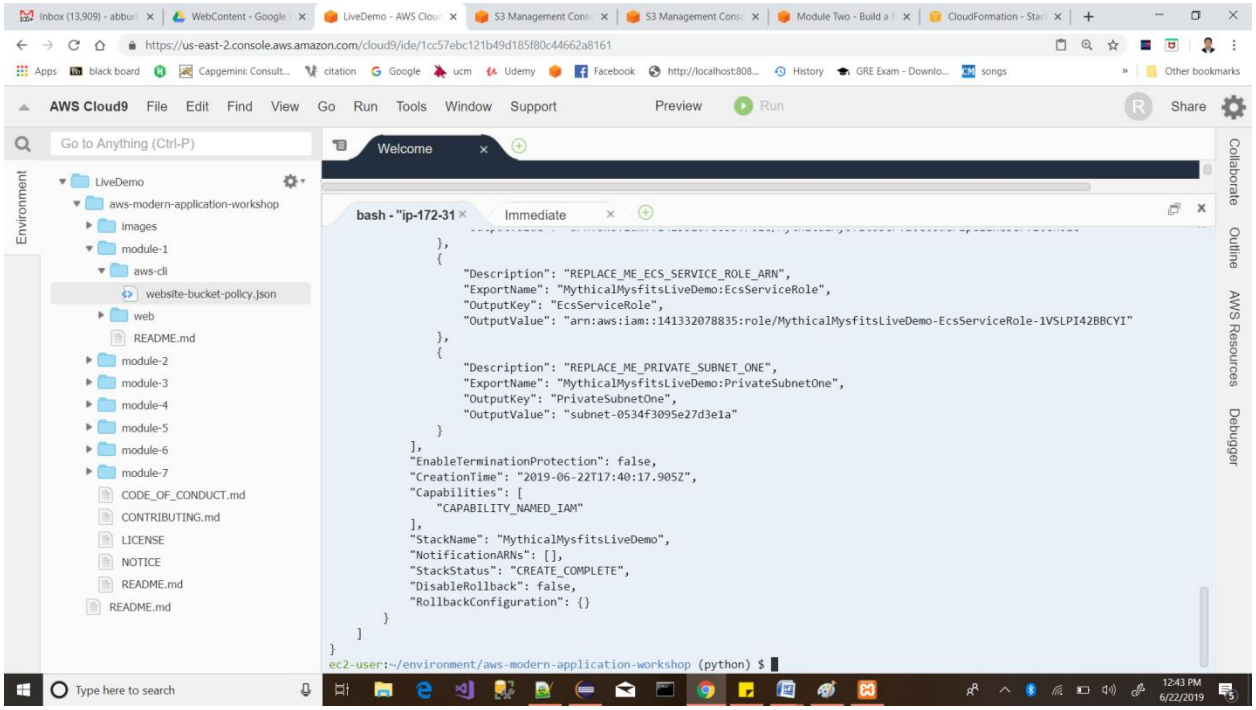


Step 20: Here we can check the status of stack creation with the following command.

awscloudformation describe-stacks --stack-name MythicalMysfitsLiveDemo



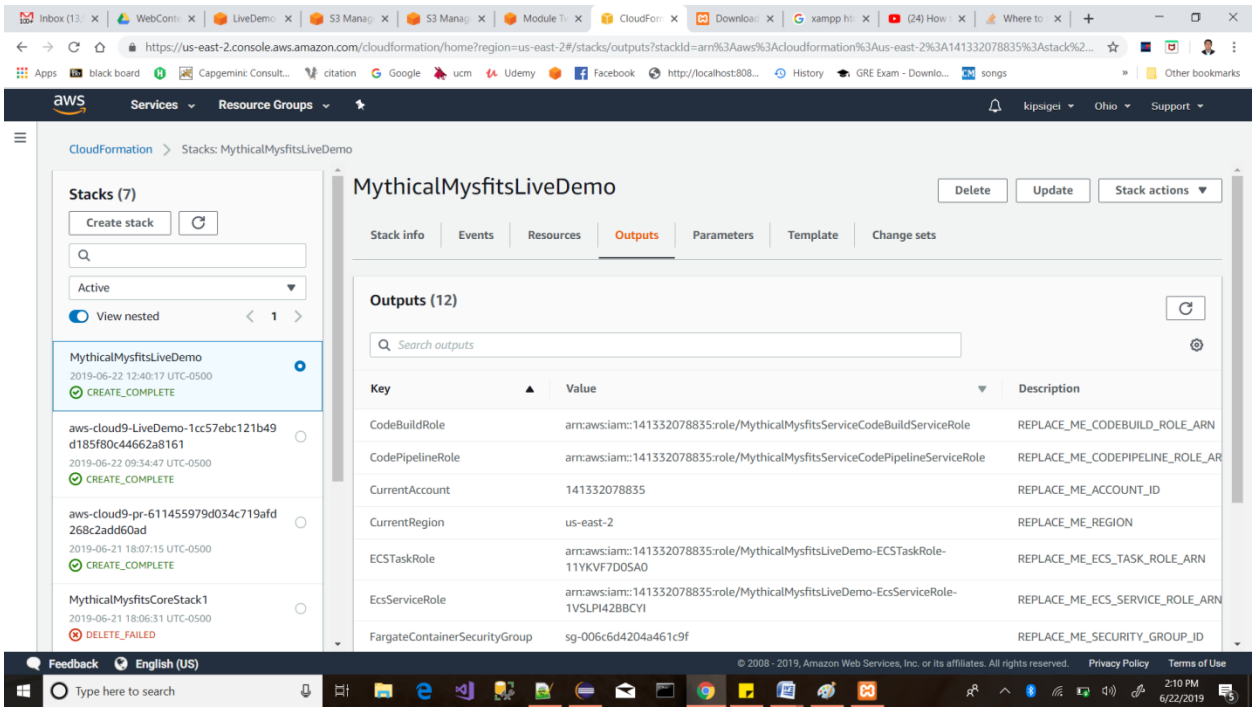
Step 21: We should wait until this step creation before proceeding to next step. We can run multiple times to check status of stack creation. Finally, after 10 minutes stack creation has completed.



Step 22: We can save these out data into json file for furthursteps by running the following command.

awscloudformation describe-stacks --stack-name MythicalMysfitsCoreStack> ~/environment/cloudformation-core-output.json

we can see these details in cloudformation click on output tab.

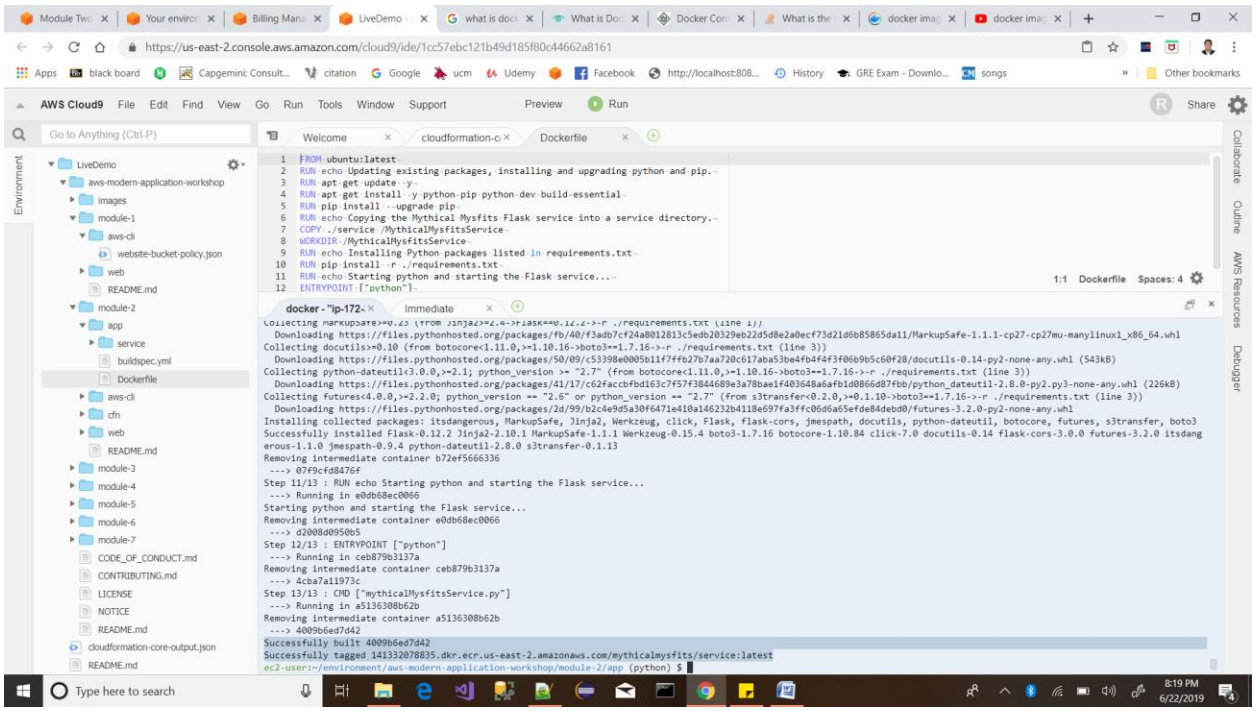


Step 23: Now we are going to build a service. Docker comes inbuilt with the cloud9 IDE instead of building a new docker locally we will create the image tag with the following command.

Change the path
cd~/environment/aws-modern-application-workshop/module-2/app

docker build . -t 141332078835.dkr.ecr.us-east-2.amazonaws.com/mythicalmysfits/service:latest

If we run the above command it will download and install required dependency packages for application service.



Successfully built 4009b6ed7d42

Successfully tagged 141332078835.dkr.ecr.us-east-2.amazonaws.com/mythicalmysfits/service:latest

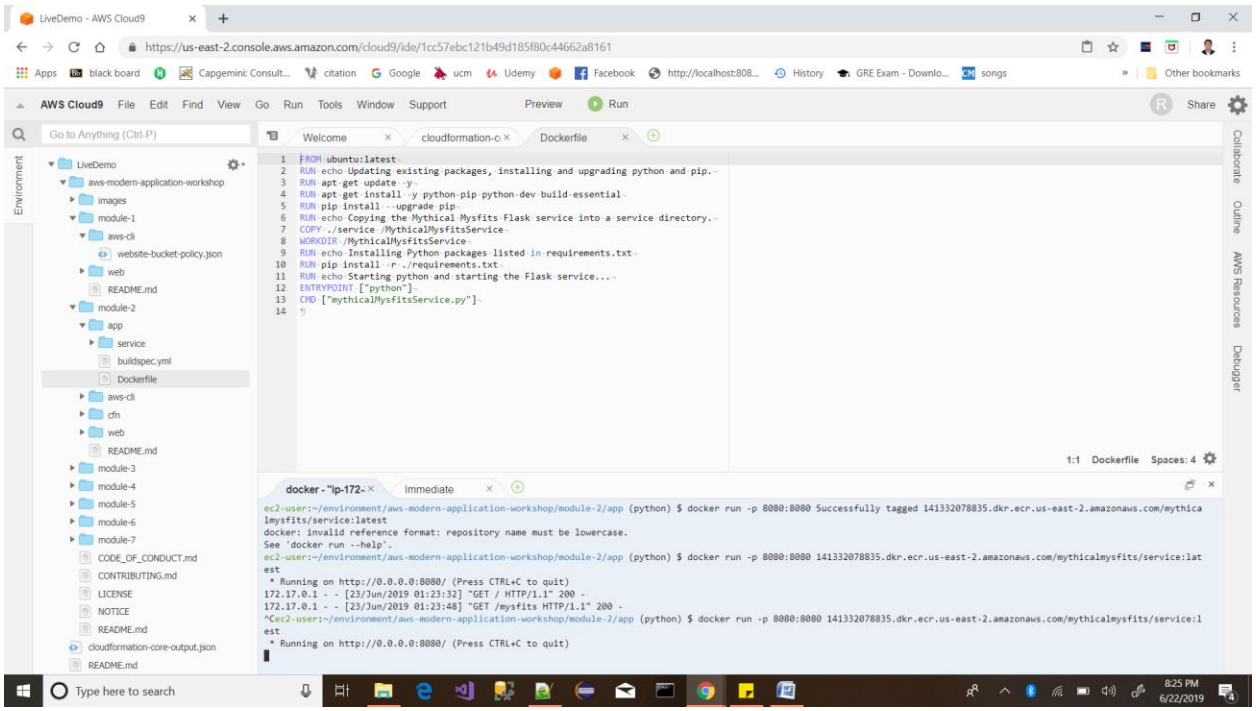
Now we can test this service locally by running following command

docker run -p 8080:8080 141332078835.dkr.ecr.us-east-2.amazonaws.com/mythicalmysfits/service:latest

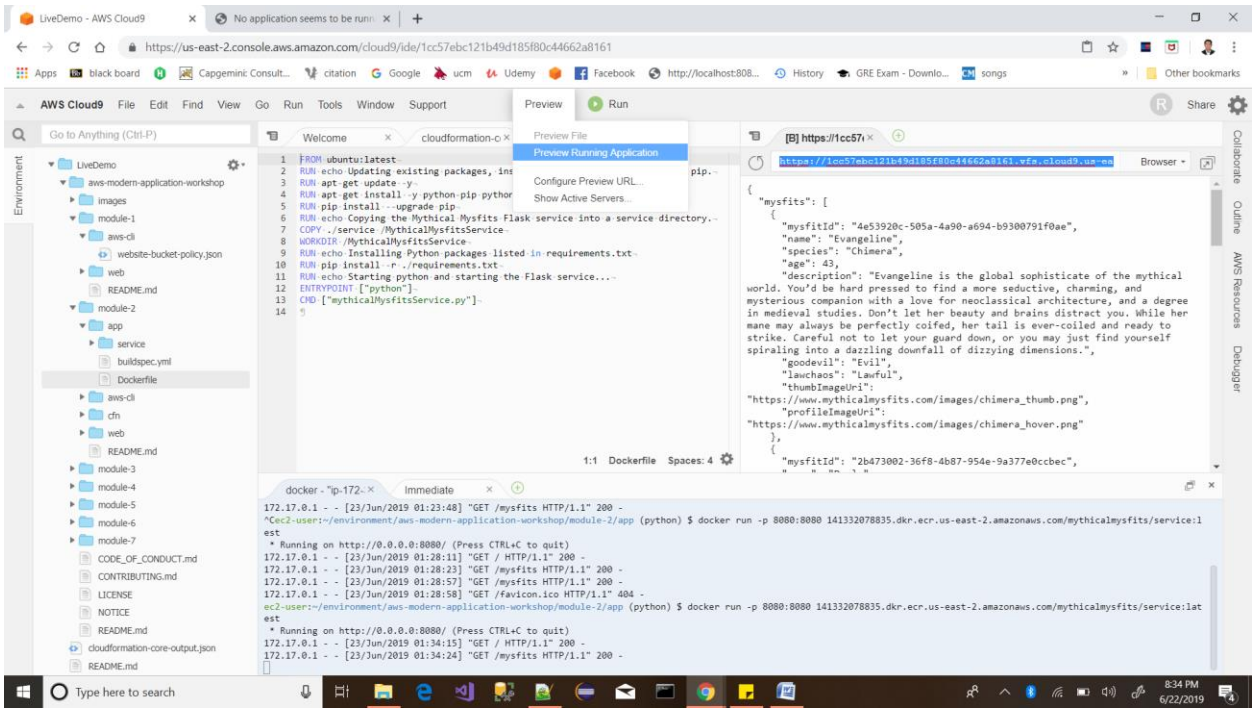
here we can view the service output with preview application option from cloud9.

Or we can hit in web browser with the following URL

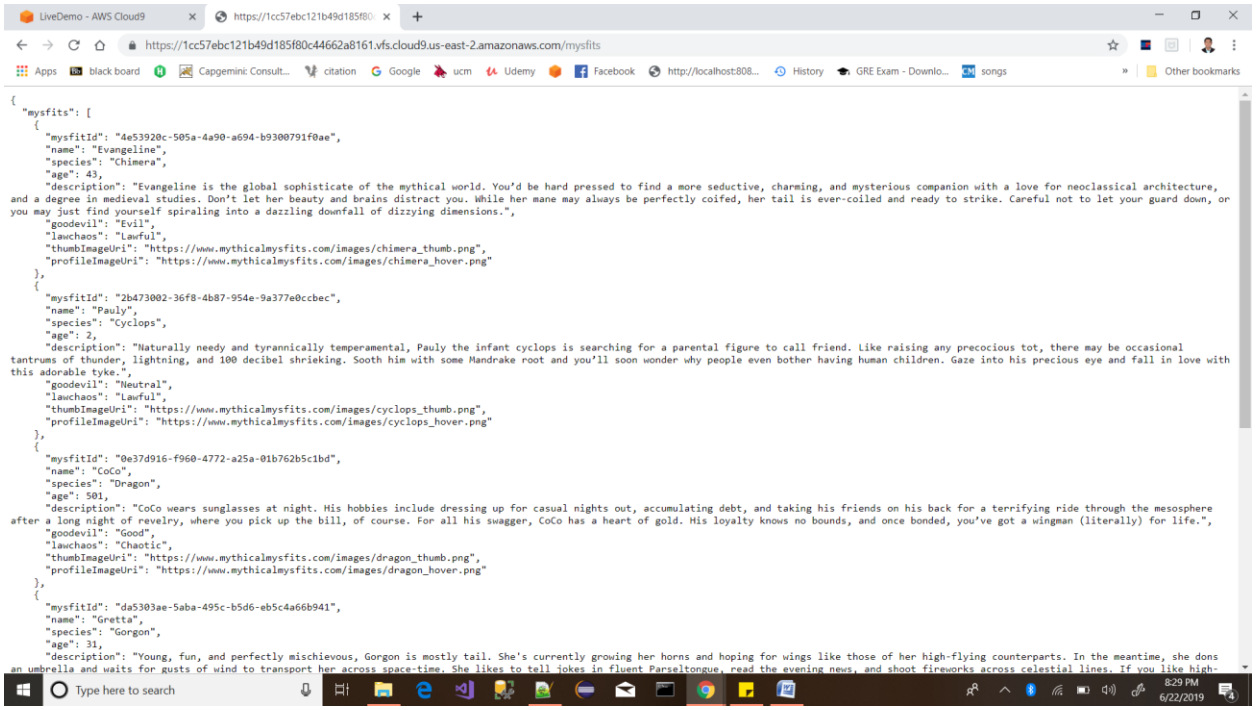
https://1cc57ebc121b49d185f80c44662a8161.vfs.cloud9.us-east-2.amazonaws.com/mysfits



Step 24: We can see the running service from IDE browser

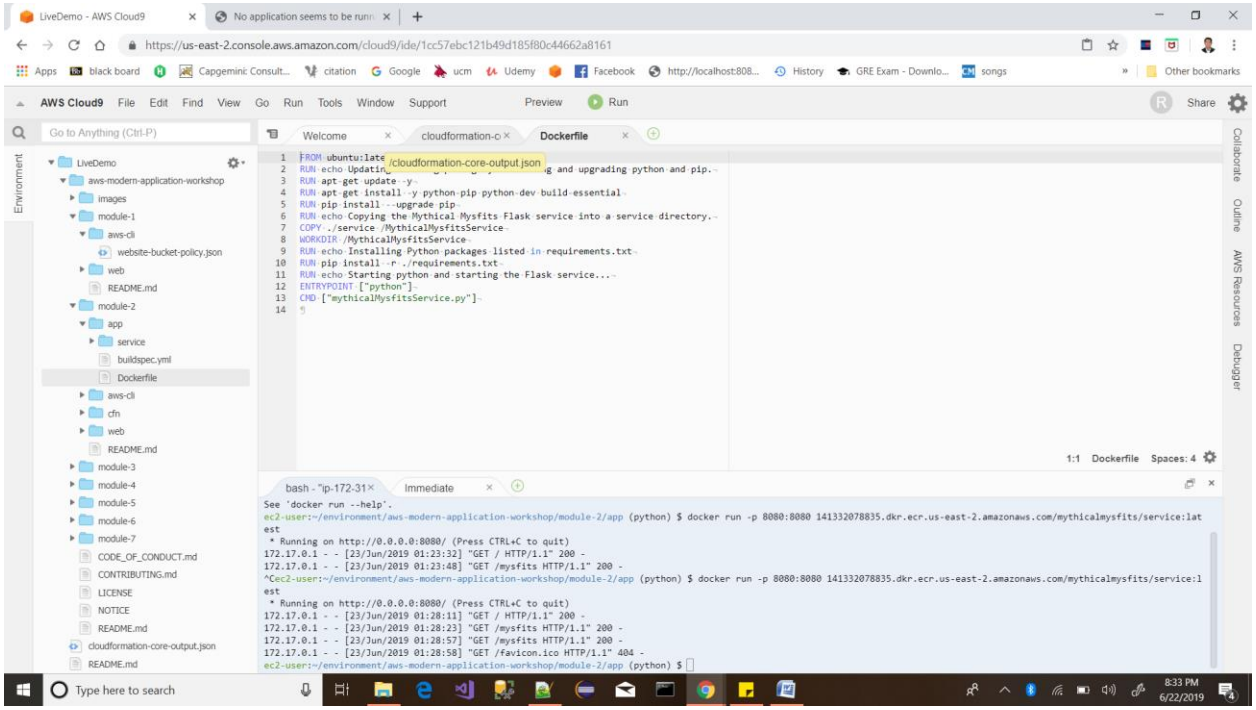


Step 25: The service output from docker image from browser view.



Step 26: We can observe this responsefrom the service that returns the JSON document stored at `/aws-modern-application-workshop/module-2/app/service/mysfits-response.json`

Once testing is done, we can press ctrl+c to stop the docker service.

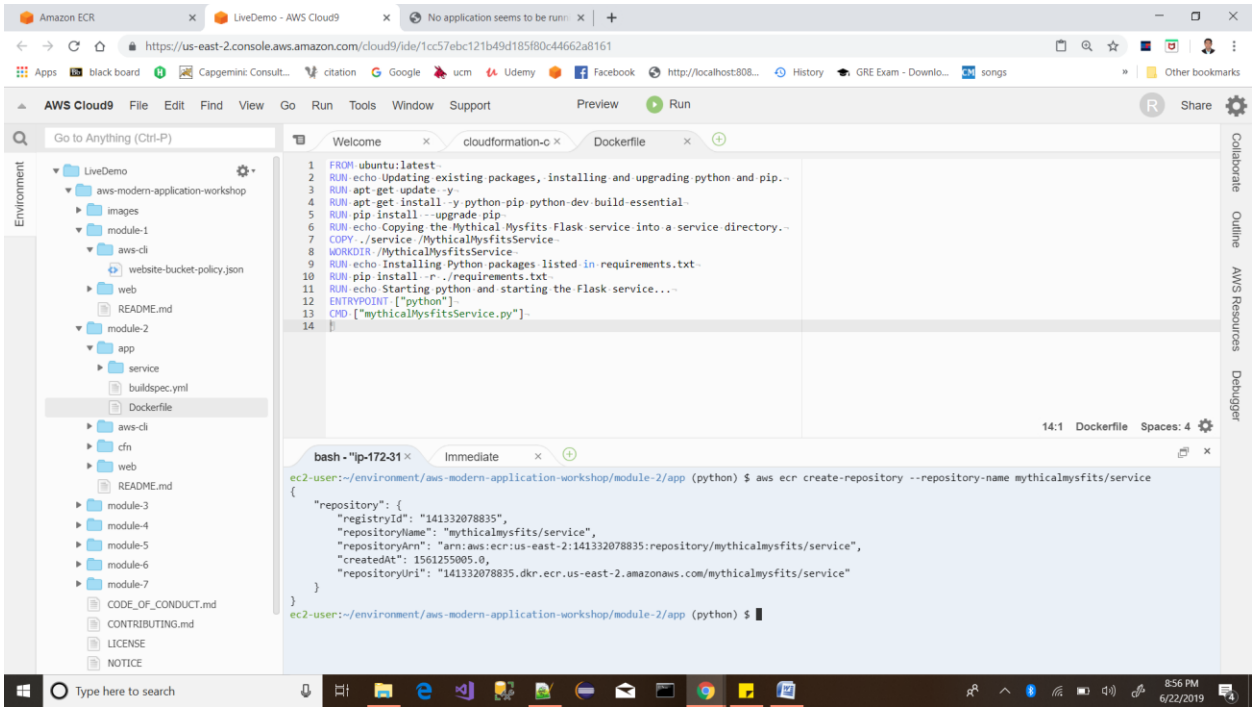


Step 27: Now we need to push this docker image into container image repository in Amazon elastic container registry (ECR).

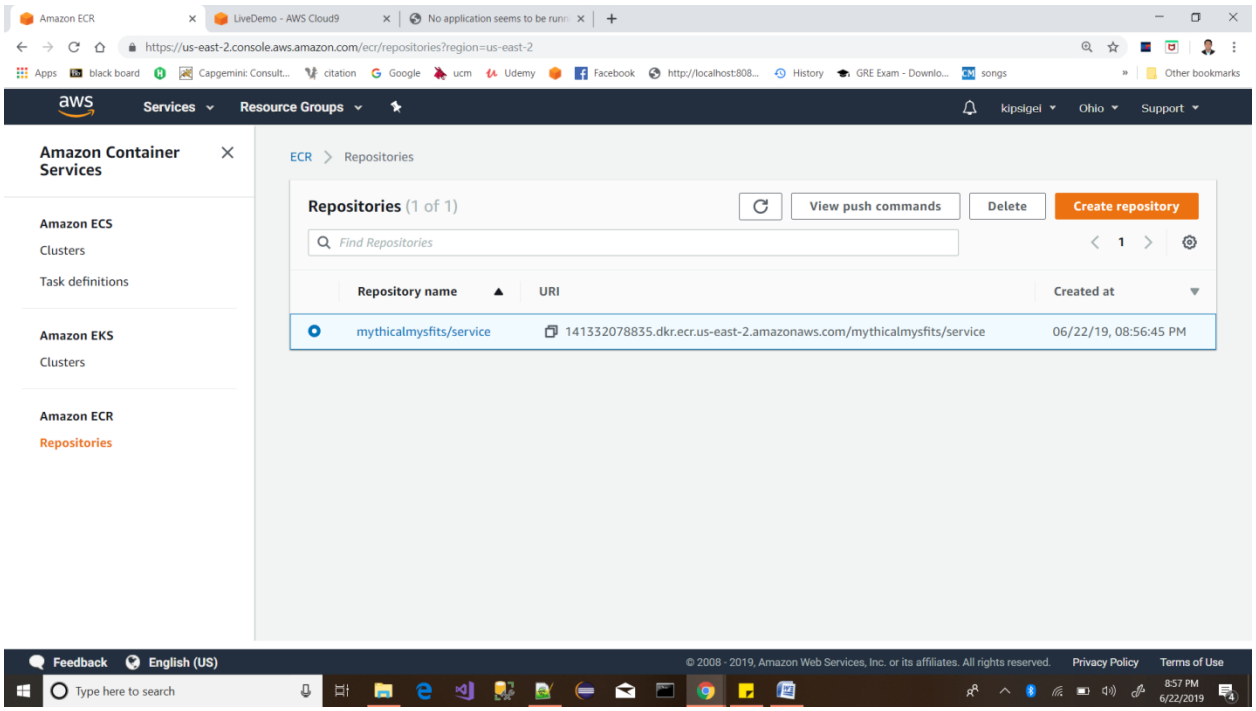
Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.

By running the following command, creates a new repository in the default AWS ECR registry created for your account.

awsecr create-repository --repository-name mythicalmysfits/service



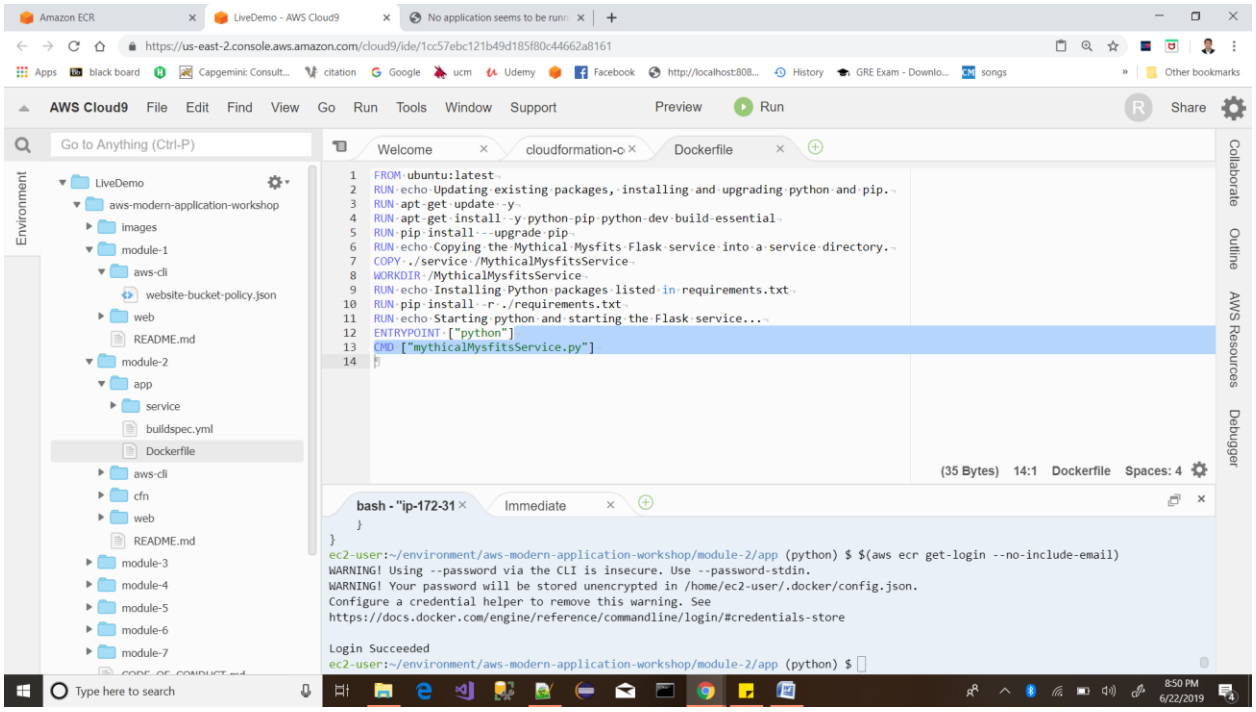
Step 28: After running the above command repository has been created and we can see the created repository in aws ECR.



Step 29: Now we need to push the docker image into repository with the following command.

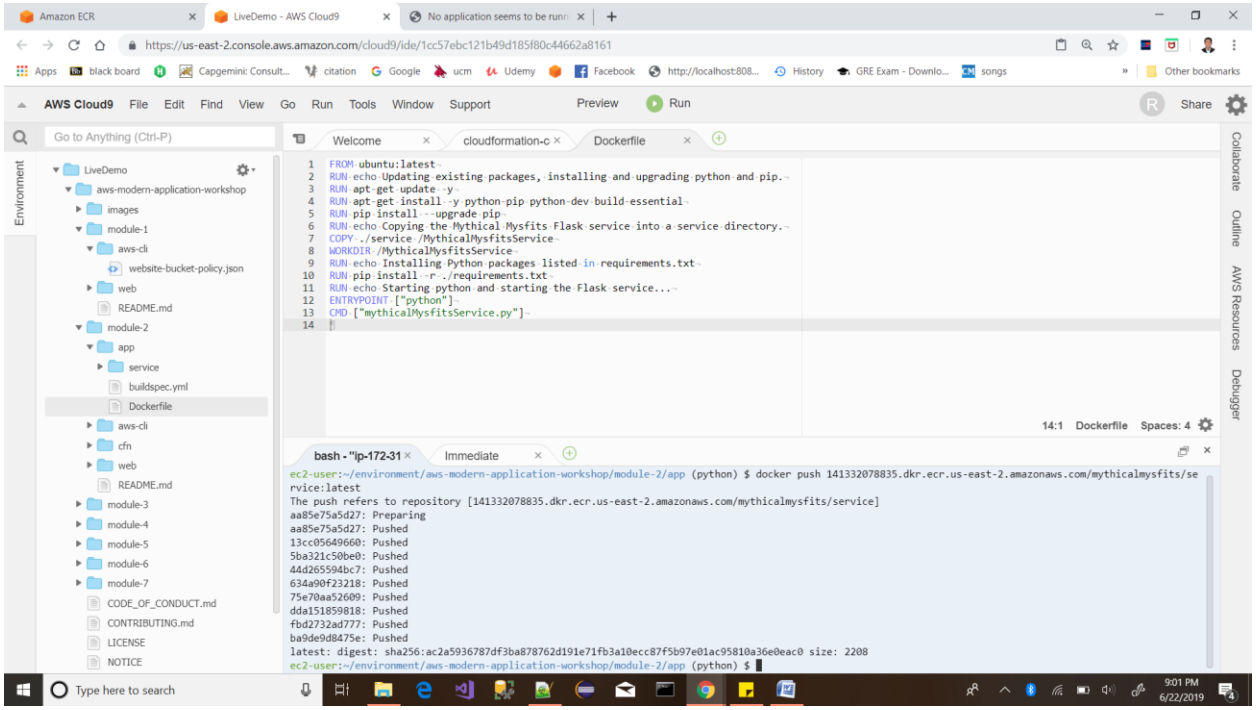
To push image file into Repository we need authentication to with docker by executing following command.

$(awsecr get-login --no-include-email)

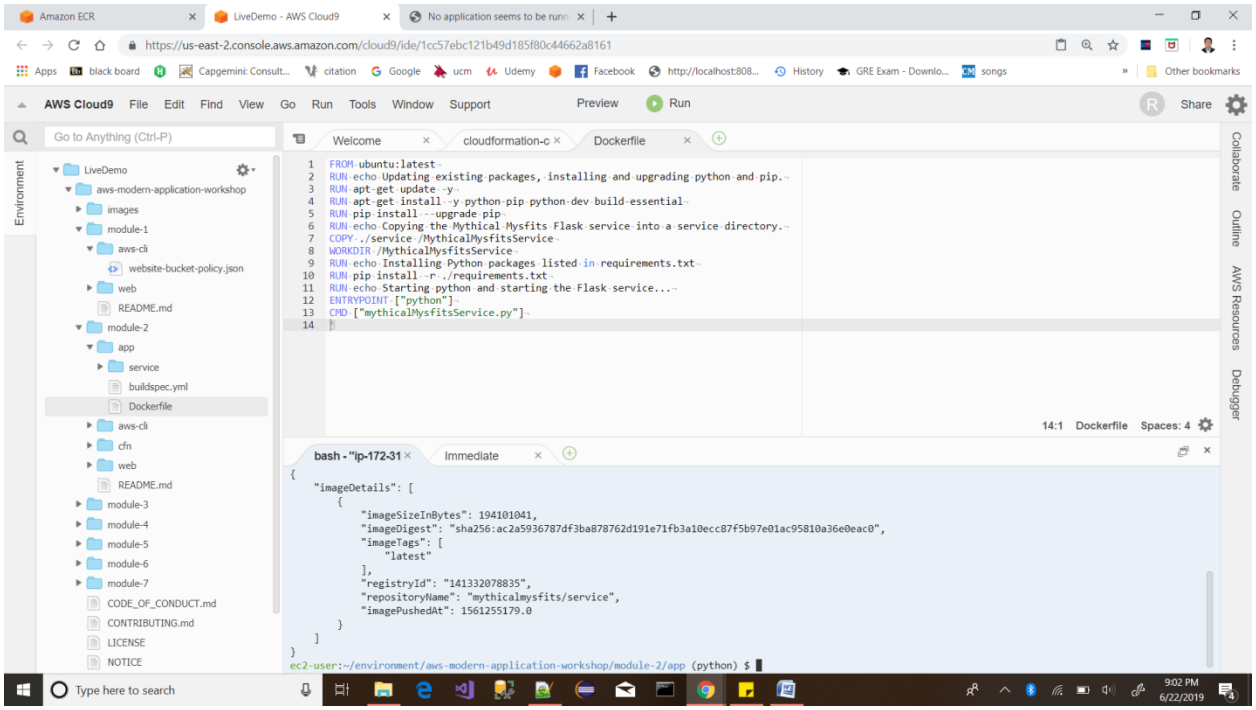


Step 30: Now we are pushing the docker image to ECR repository.

docker push 141332078835.dkr.ecr.us-east-2.amazonaws.com/mythicalmysfits/service:latest



Step 31: Now we can view the docker image in ECR Repository.



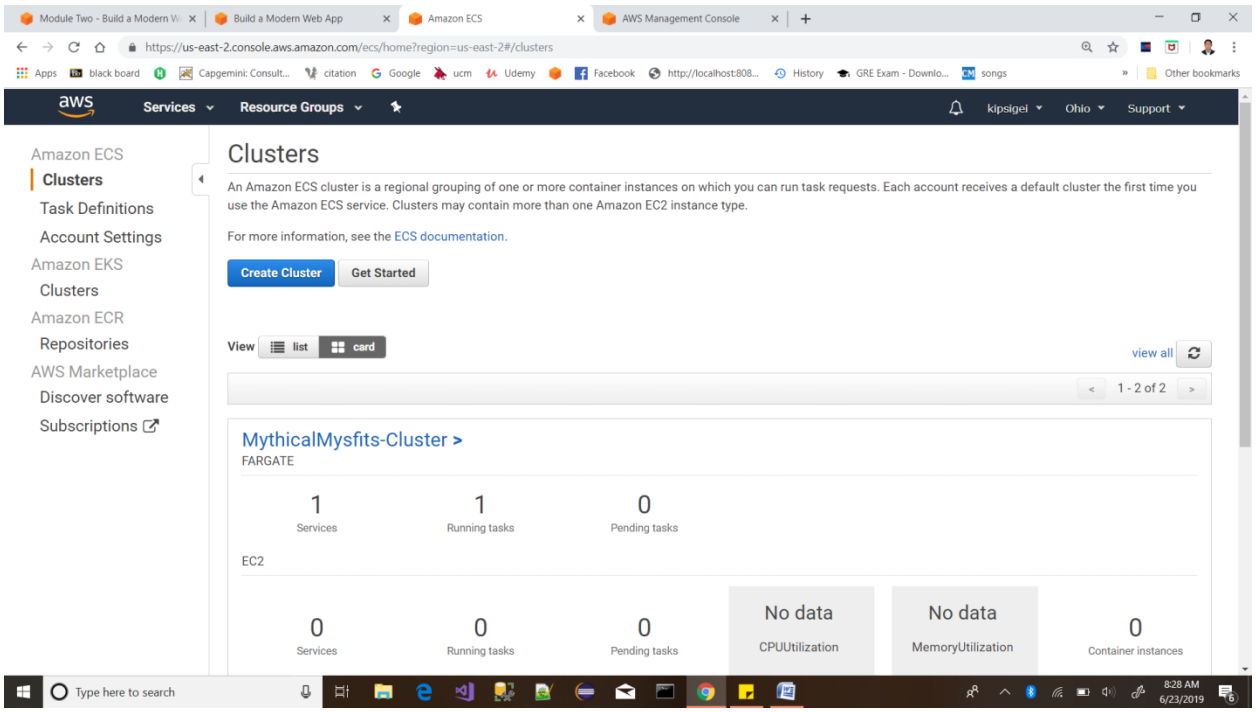
Step 32: Now service available in local. We are going to deploy the service hosted on Amazon ECS using AWS Fargate and will make the service available to publicly behind Network load balancer.

First, we need to create cluster in ECS. This represents the cluster of “servers” that your service containers will be deployed to. Servers is in "quotations" because you will be using AWS Fargate. Fargate allows us to specify that containers be deployed to a cluster without having to provision or manage any servers yourself.

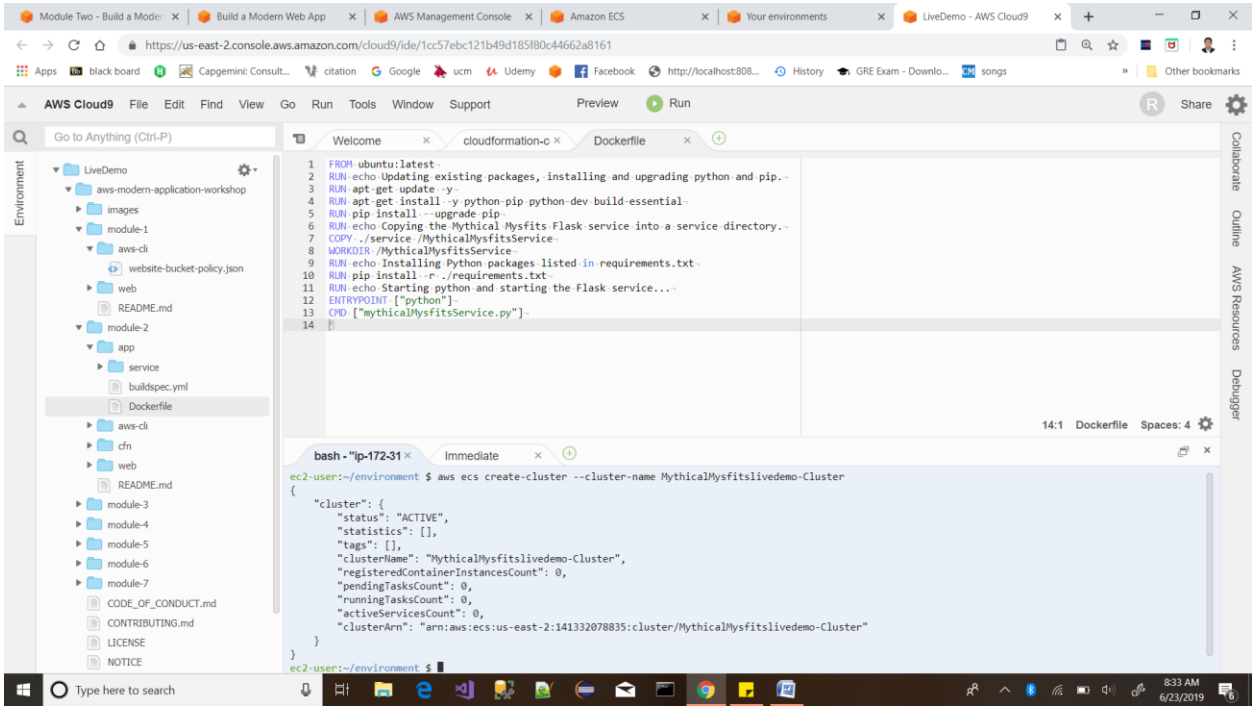
To create a cluster, we need to execute below command.

awsecs create-cluster --cluster-name MythicalMysfitslivedemo-Cluster

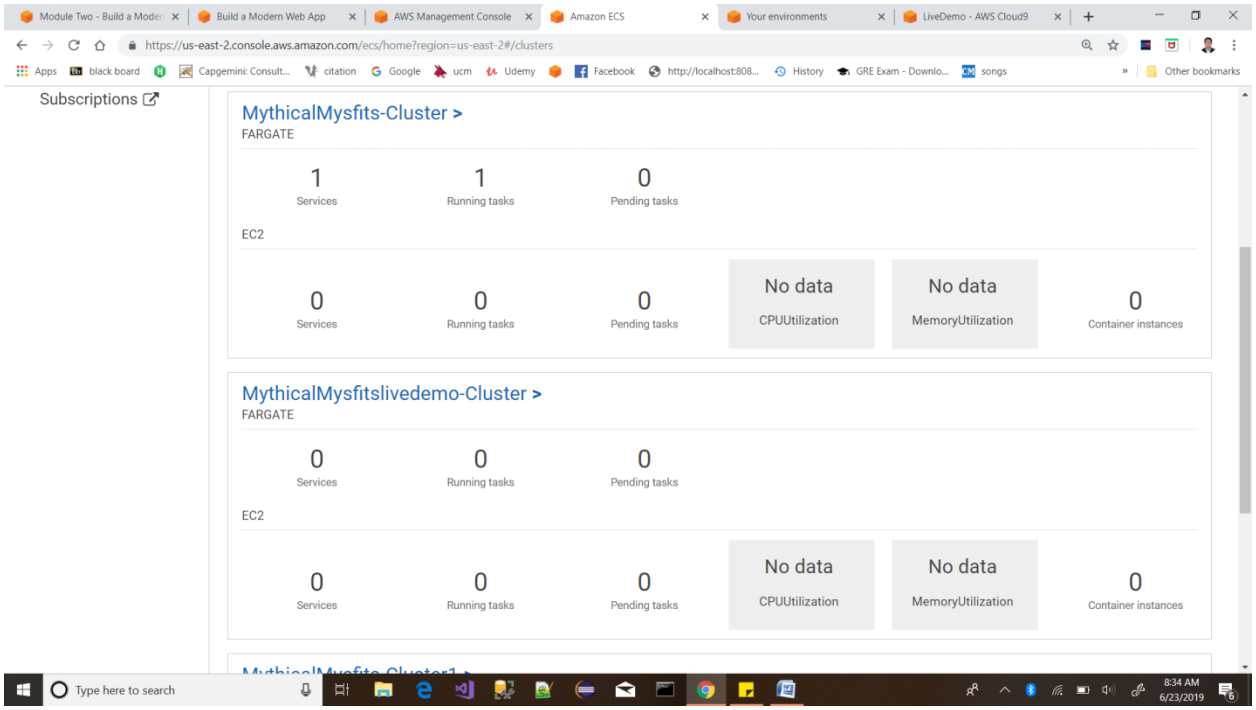
if we observe the before running command , The ECS console shows available clusters and services.



Step 33: Now we are executing the command to create the cluster

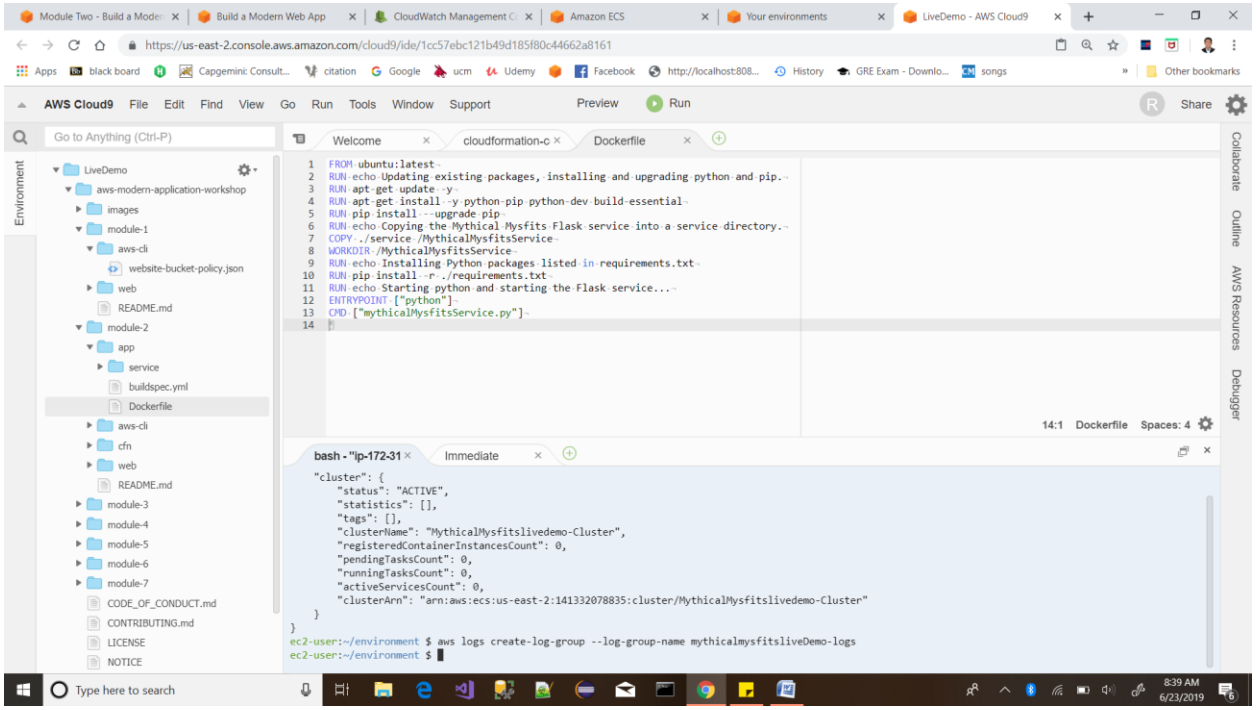


Step 34: Now we can see that the cluster is created in ECS console.

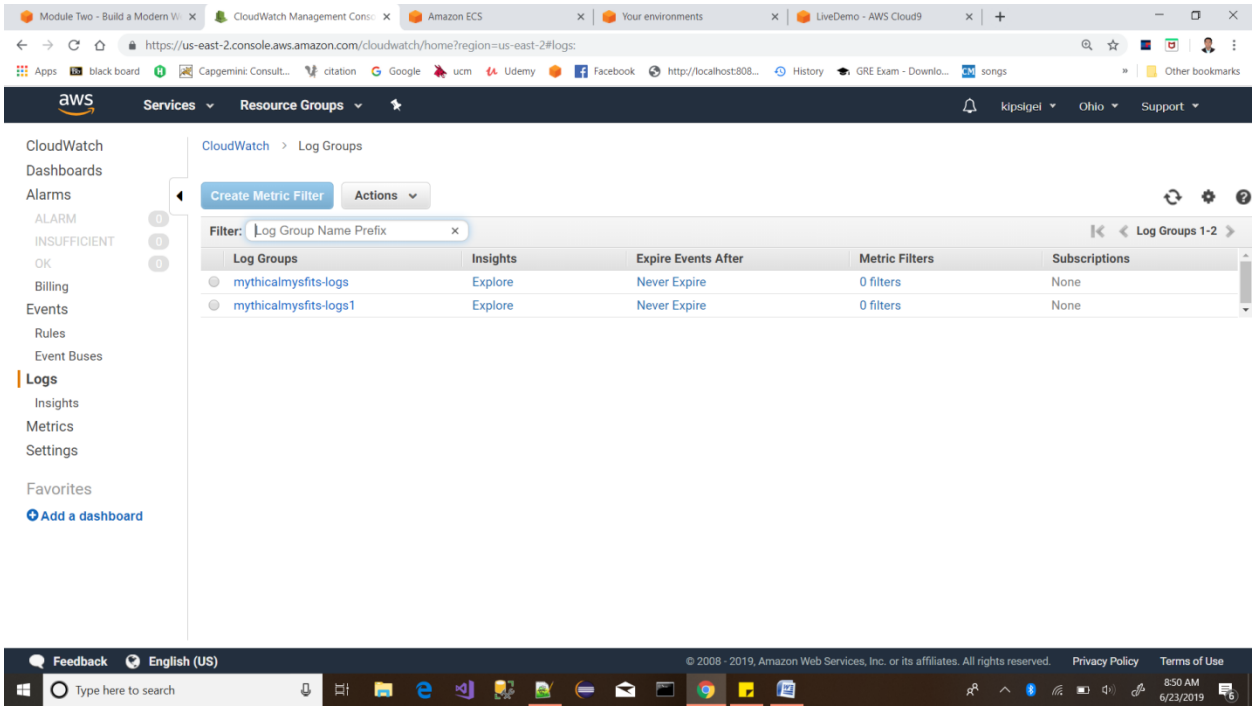


Step 35: Next we need to create the logs for the serviceThat we are configuring with AWS cloudWatch Logs. It is a collection of logs and we can monitor the system behavior in graph format.Execute the following command.

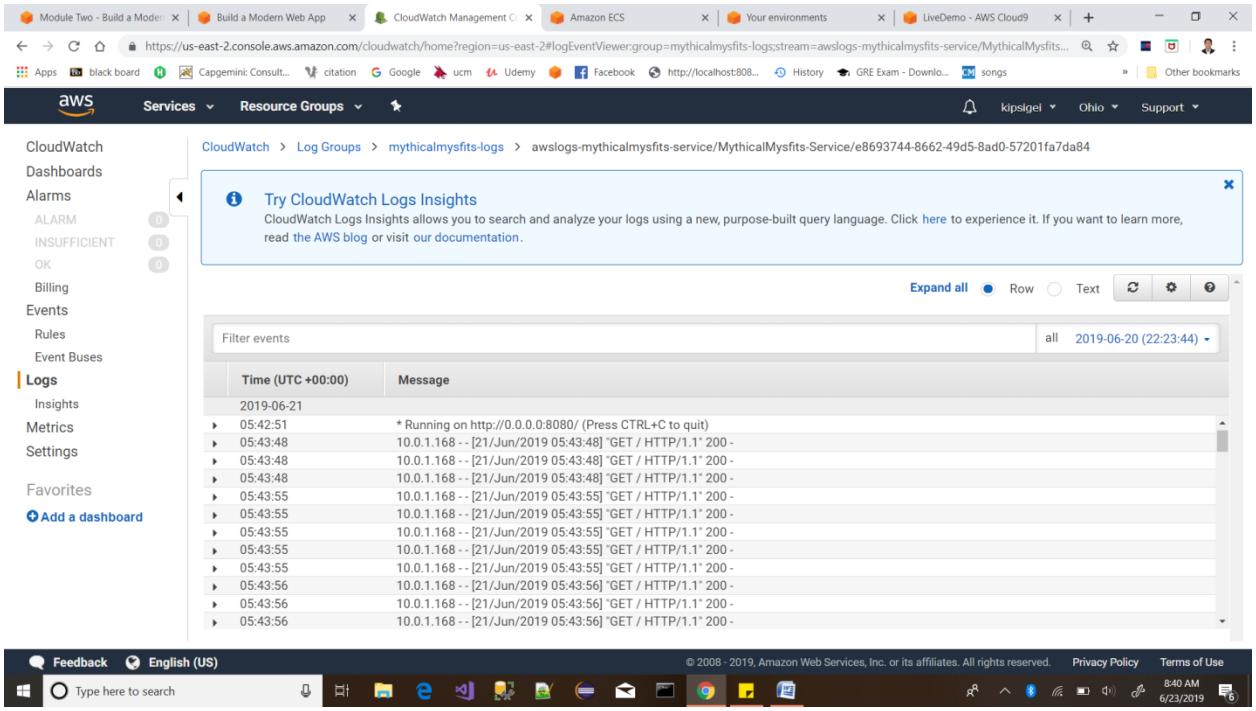
aws logs create-log-group --log-group-name mythicalmysfits-logs



Step 36: Now we can see the cloud watch logs file in AWS cloud watch console.



Step 37: By selecting a specificlog group we can see the logs of service. Below screen shot is sample of existing service. But the newly created one logs are empty.



Step 38: Now we have created cluster and a log group where our container logs will be pushed to and we're ready to register an ECS task definition. A task in ECS is a set of container images

that should be scheduled together. A task definition declares that set of containers and the resources and configuration those containers require. You will use the AWS CLI to create a new task definition for how your new container image should be scheduled to the ECS cluster we just created.

A JSON file has been provided that will serve as the input to the CLI command.

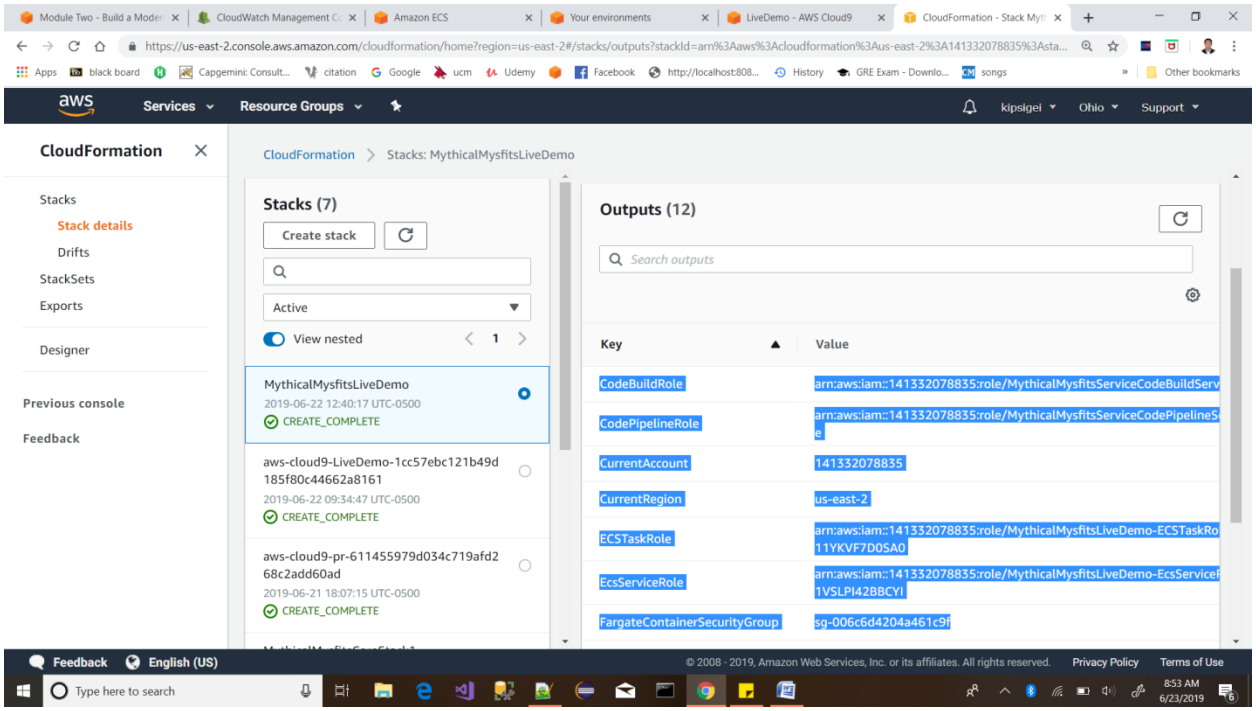
Open ~/environment/aws-modern-application-workshop/module-2/aws-cli/task-definition.json in the IDE.

Here we need replace the image tags in the above file and we need to run the following command.

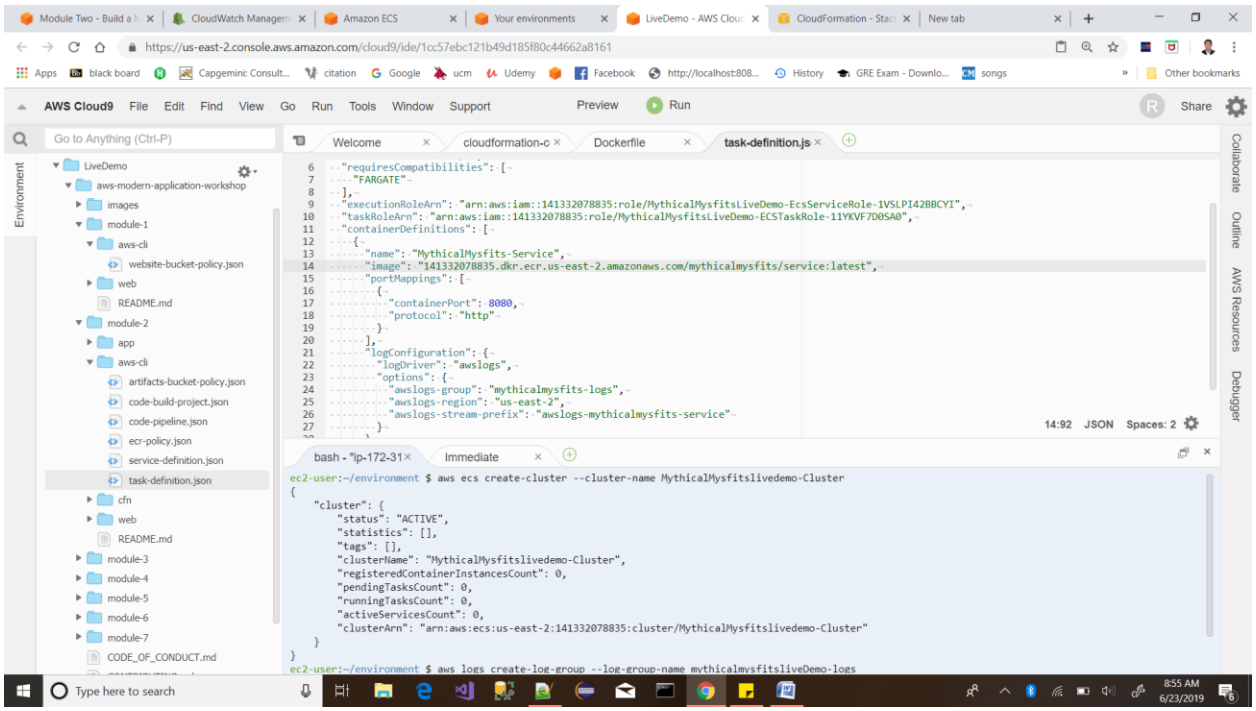
awsecs register-task-definition --cli-input-json file://~/environment/aws-modern-application-workshop/module-2/aws-cli/task-definition.json

Step 39:

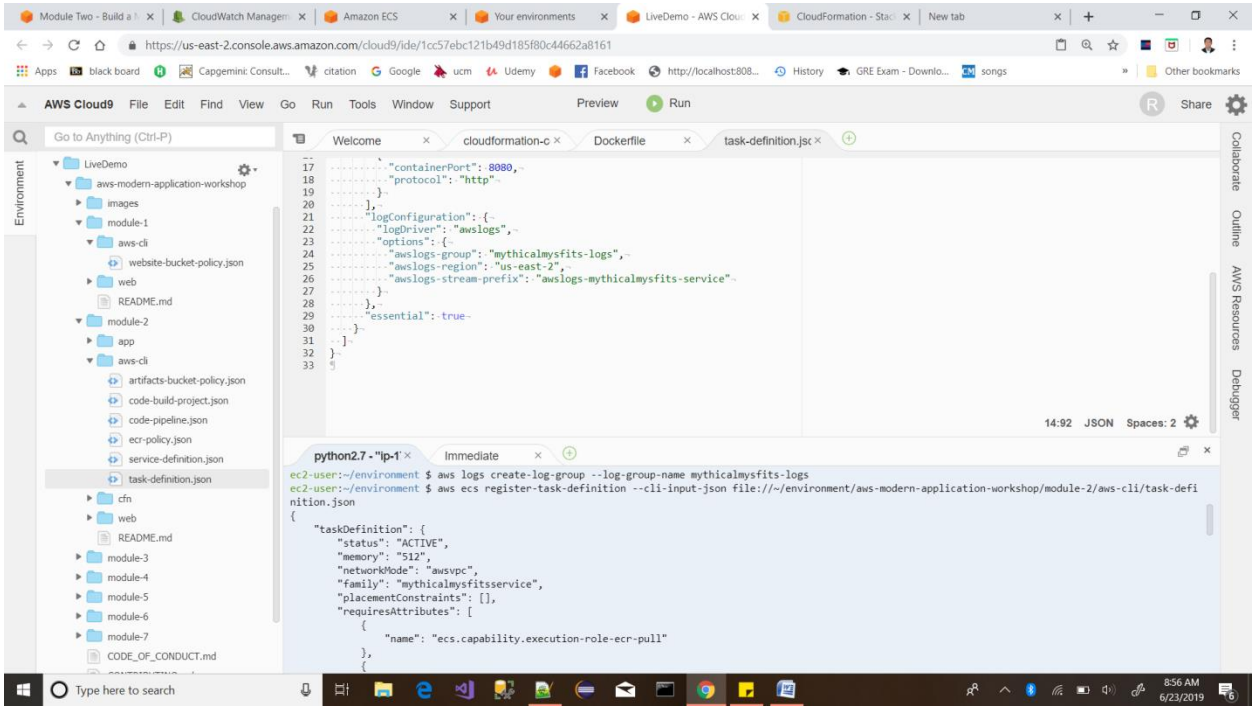
These values we can see in cloud formation template



Step 40: The task definition file has shown as below



Step 41: Now we are executing the below command to register an ECS Task definition.



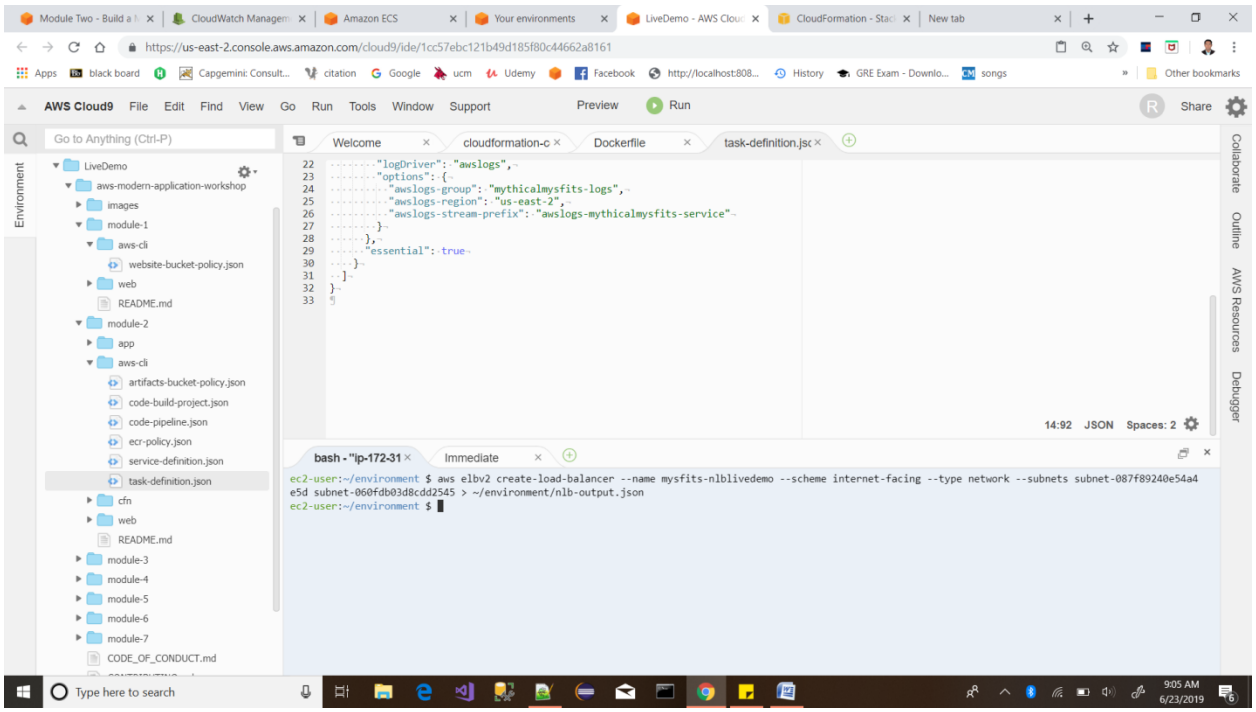
Step 42: Now we need to create the load balancer to fargate service.

If we create a Network load balancer with public access any one can access the website or services from outside the network.

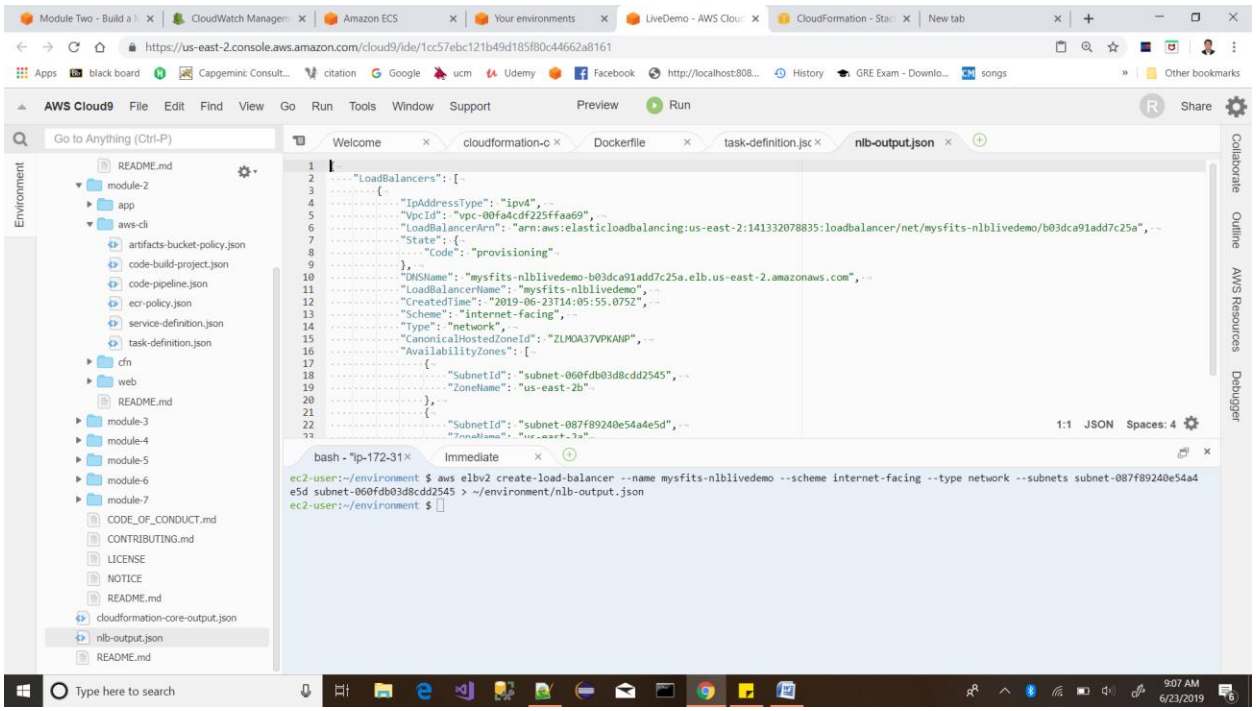
To provision a NLB we need to execute following command.

aws elbv2 create-load-balancer --name mysfits-nlblivedemo --scheme internet-facing --type network --subnets subnet-087f89240e54a4e5d subnet-060fdb03d8cdd2545 > ~/environment/nlb-output.json

After executing the command we can see the nlb-output.json file in the environment.In this file we can se the DNS name and Loadbalncer ARN names



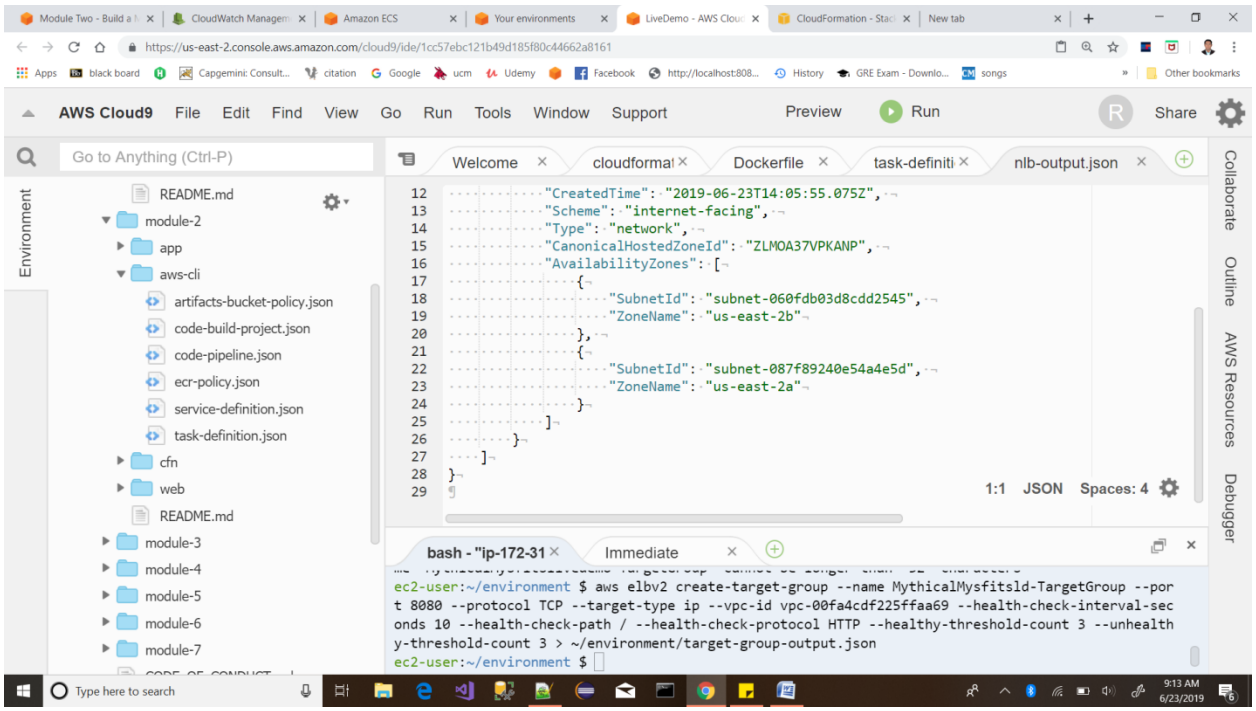
Step 43: After executing the command we can see the nlb-output.json file in the environment.In this file we can se the DNS name and Loadbalncer ARN names



Step 44: Next, use the CLI to create an NLB target group. A target group allows AWS resources to register themselves as targets for requests that the load balancer receives to forward.

For creating target group need to execute following command.

aws elbv2 create-target-group --name MythicalMysfitsld-TargetGroup --port 8080 --protocol TCP --target-type ip --vpc-id vpc-00fa4cdf225ffaa69 --health-check-interval-seconds 10 --health-check-path / --health-check-protocol HTTP --healthy-threshold-count 3 --unhealthy-threshold-count 3 > ~/environment/target-group-output.json



Step 45: Now target-group-output.json file has created this consist of target group ARN Values and vpv id as follows:

"TargetGroups": [

{

"HealthCheckPath": "/",

"HealthCheckIntervalSeconds": 10,

"VpcId": "vpc-00fa4cdf225ffaa69",

"Protocol": "TCP",

"HealthCheckTimeoutSeconds": 6,

"TargetType": "ip",

"HealthCheckProtocol": "HTTP",

"Matcher": {

"HttpCode": "200-399"

},

"UnhealthyThresholdCount": 3,

"HealthyThresholdCount": 3,

"TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-2:141332078835:targetgroup/MythicalMysfitsld-TargetGroup/97b6288f0abc266d",

"HealthCheckEnabled": true,

"HealthCheckPort": "traffic-port",

"Port": 8080,

"TargetGroupName": "MythicalMysfitsld-TargetGroup"

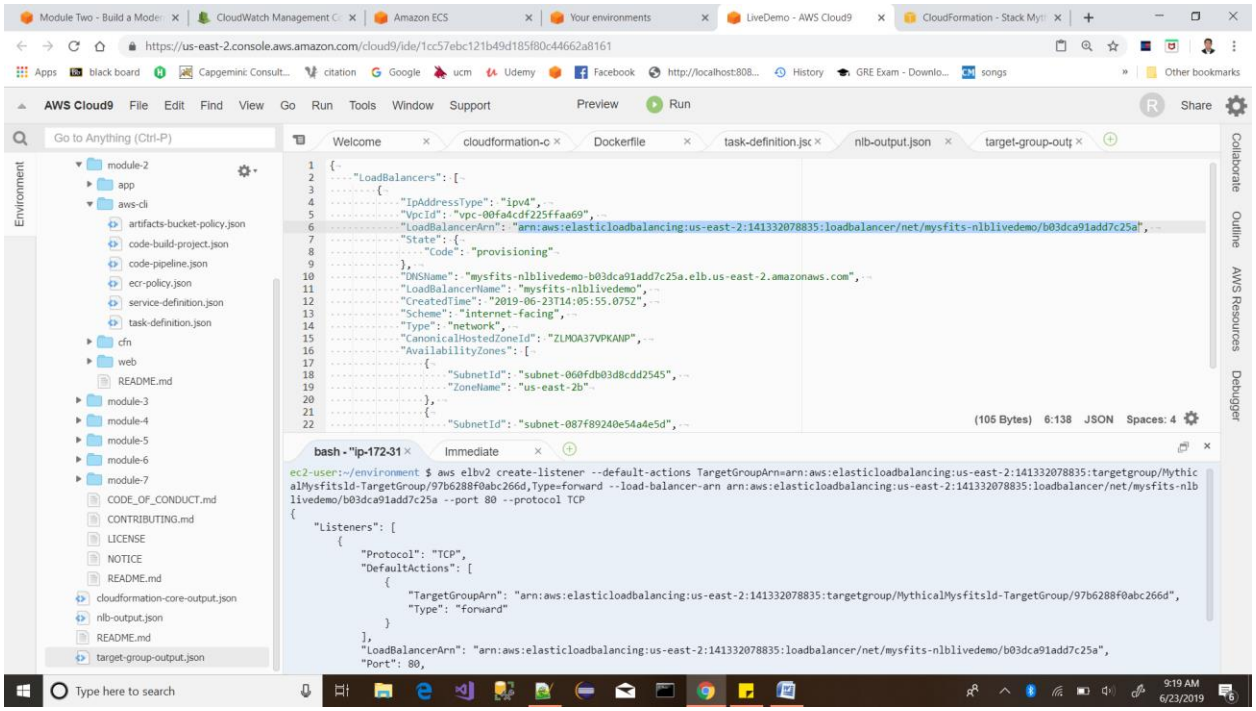
}

Step

Now we need to make this loadbalncer in Listening state:

In the following command need to replace the target group ARN whilch is there in target-group-output.json and Network Loadbalancer ARN which is aviable at nlb-output.json .

aws elbv2 create-listener --default-actions TargetGroupArn=arn:aws:elasticloadbalancing:us-east-2:141332078835:targetgroup/MythicalMysfitsld-TargetGroup/97b6288f0abc266d,Type=forward --load-balancer-arn arn:aws:elasticloadbalancing:us-east-2:141332078835:loadbalancer/net/mysfits-nlblivedemo/b03dca91add7c25a --port 80 --protocol TCP

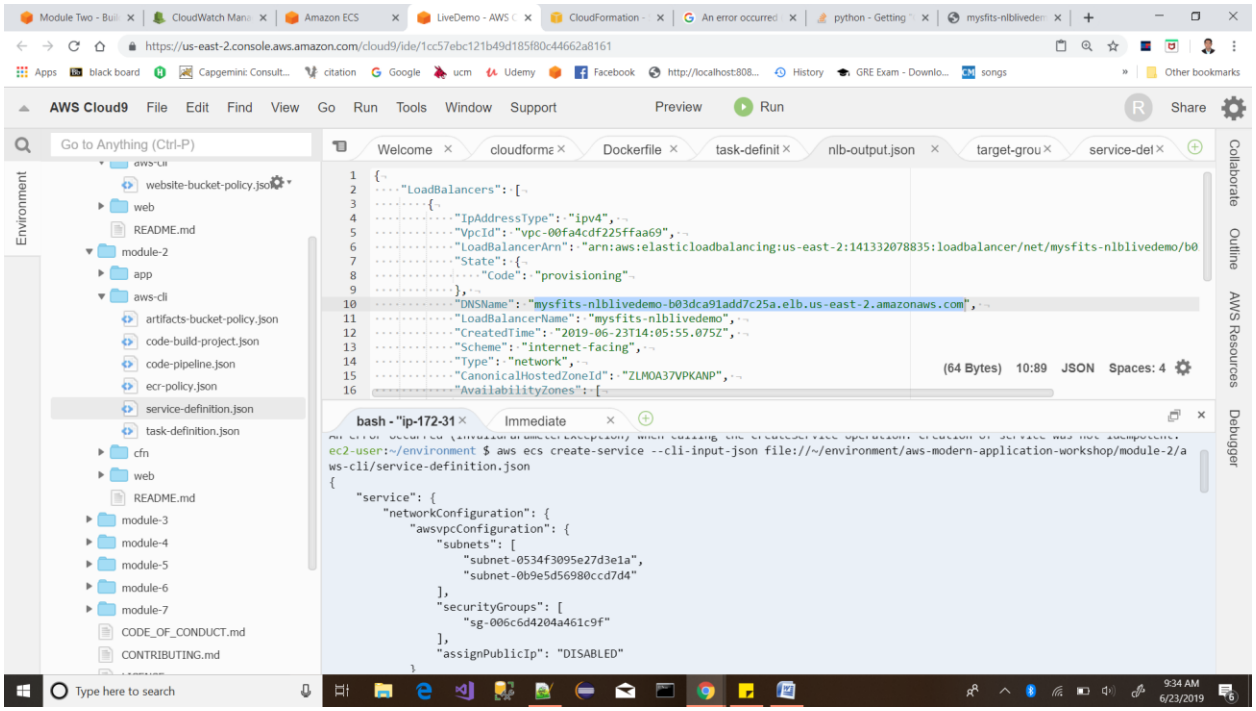


Step 46:

Grant the permission to the service with iam role.

```
awsiam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

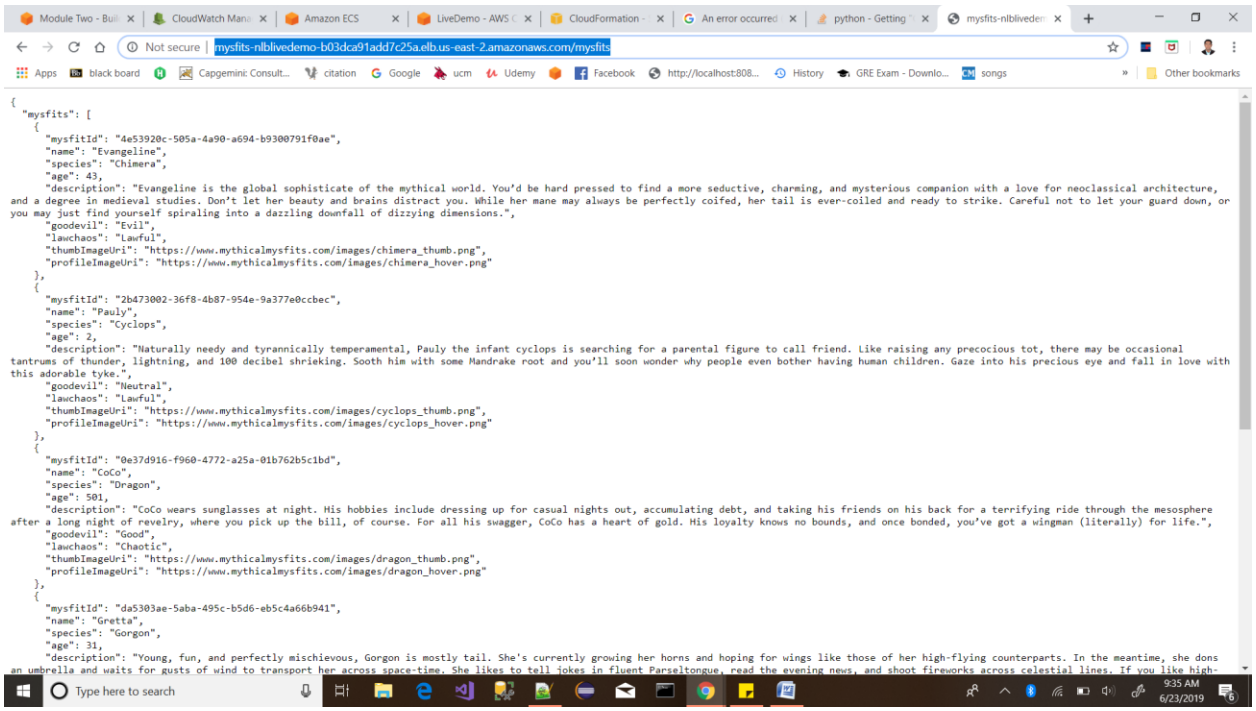
NLB has created and configured.now create the ECS service to be created and that service should be launched with AWS Fargate by running the following command.



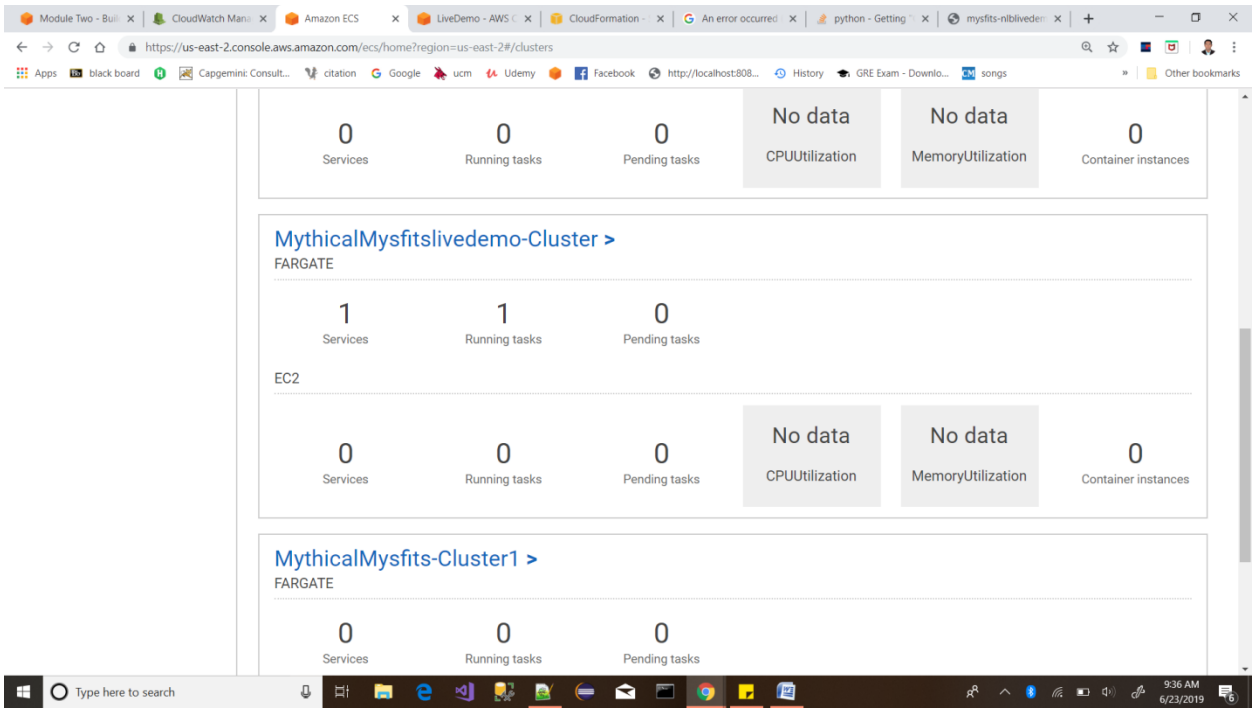
Step 47: Now we can test the service with DNS configured for the service

http://mysfits-nlblivedemo-b03dca91add7c25a.elb.us-east-2.amazonaws.com/mysfits

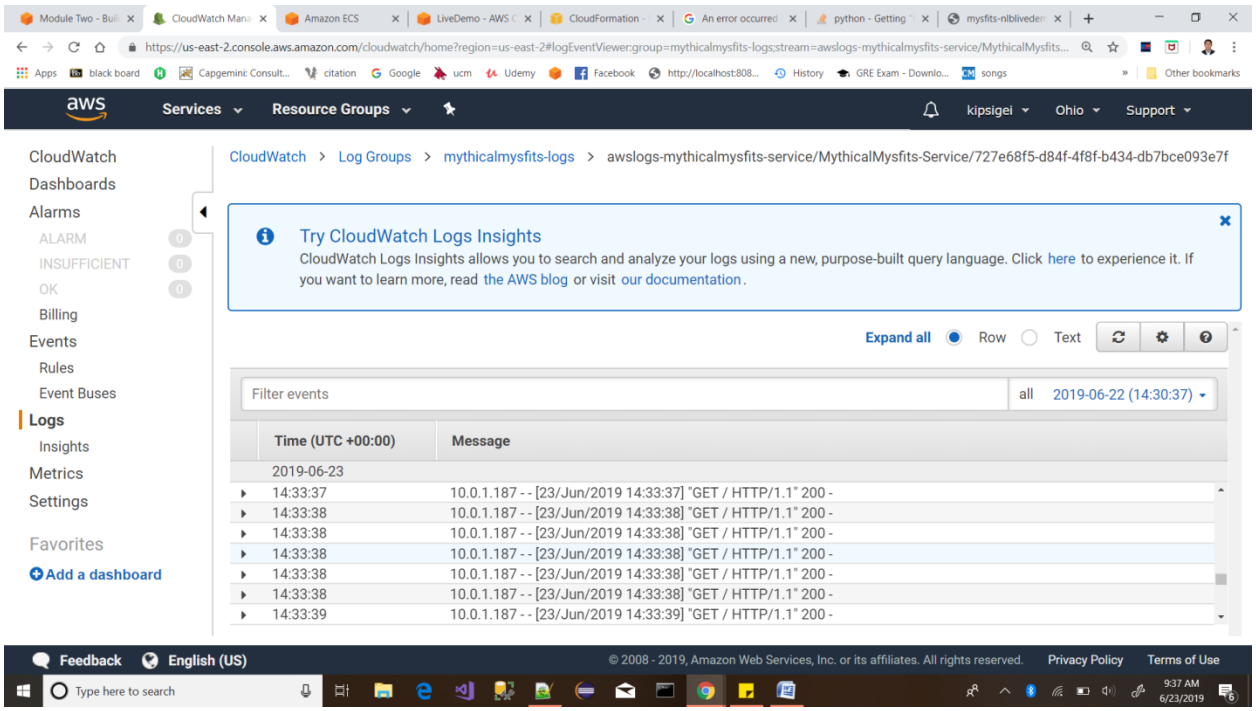
we can see the response from server and the service output displayed in browser.



Step 48: Now we can see the service available in the cluster



Step 49: We can observe the logs of service.


Module Two - Buil
CloudWatch Mana
Amazon ECS
LiveDemo - AWS
CloudFormation
An error occurred
python - Getting
mysfits-nlbliveden
https://us-east-2.console.aws.amazon.com/cloudwatch/home?region=us-east-2#logEventViewer:group=mythicalmysfits-logs;stream=awslogs-mythicalmysfits-service/MythicalMysfits...
Apps
black board
Capgemini: Consult...
citation
Google
ucm
Udemy
Facebook
http://localhost:808...
History
GRE Exam - Downlo...
songs
Other bookmarks
aws
Services
Resource Groups
kipsigei
Ohio
Support
CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 0
OK 0
Billing
Events
Rules
Event Buses
Logs
Insights
Metrics
Settings
Favorites
Add a dashboard
CloudWatch > Log Groups > mythicalmysfits-logs > awslogs-mythicalmysfits-service/MythicalMysfits-Service/727e68f5-d84f-4f8f-b434-db7bce093e7f
Try CloudWatch Logs Insights
CloudWatch Logs Insights allows you to search and analyze your logs using a new, purpose-built query language. Click here to experience it. If you want to learn more, read the AWS blog or visit our documentation.
Expand all
Row
Text
Filter events
all
2019-06-22 (14:30:37)
Time (UTC +00:00)
Message
2019-06-23
14:33:37 10.0.1.187 - - [23/Jun/2019 14:33:37] "GET / HTTP/1.1" 200 -
14:33:38 10.0.1.187 - - [23/Jun/2019 14:33:38] "GET / HTTP/1.1" 200 -
14:33:38 10.0.1.187 - - [23/Jun/2019 14:33:38] "GET / HTTP/1.1" 200 -
14:33:38 10.0.1.187 - - [23/Jun/2019 14:33:38] "GET / HTTP/1.1" 200 -
14:33:38 10.0.1.187 - - [23/Jun/2019 14:33:38] "GET / HTTP/1.1" 200 -
14:33:38 10.0.1.187 - - [23/Jun/2019 14:33:38] "GET / HTTP/1.1" 200 -
14:33:39 10.0.1.187 - - [23/Jun/2019 14:33:39] "GET / HTTP/1.1" 200 -
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search
9:37 AM
6/23/2019