# Developer Guide for Hackathon using Python Language - Sep 2023

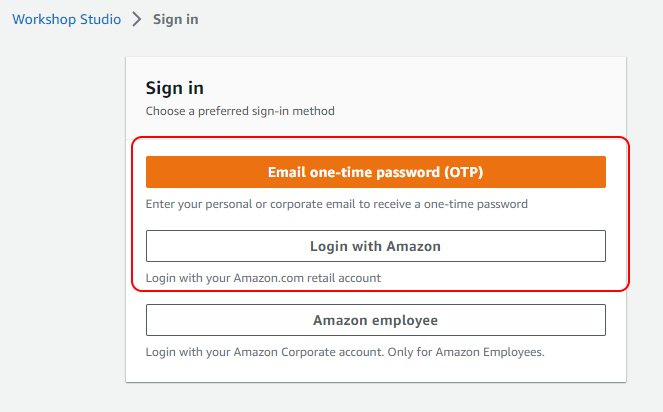
# Introduction:

This document explains step by step how to use Apprunner and other AWS Services during the Hackathon for CITI developers, Also this showing a Sample code using **Python Language** running on Apprunner .

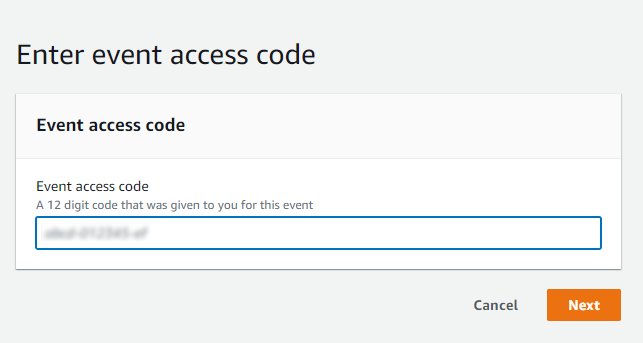
## Accessing AWS  environment for the Hackathon

Navigate to the link provided to you by the instructor. If you do not have a specific URL for this event, you can navigate to <https://catalog.us-east-1.prod.workshops.aws/join>and you will be prompted for an event code later.

You will need to login to the environment using one of the two indicated methods below.

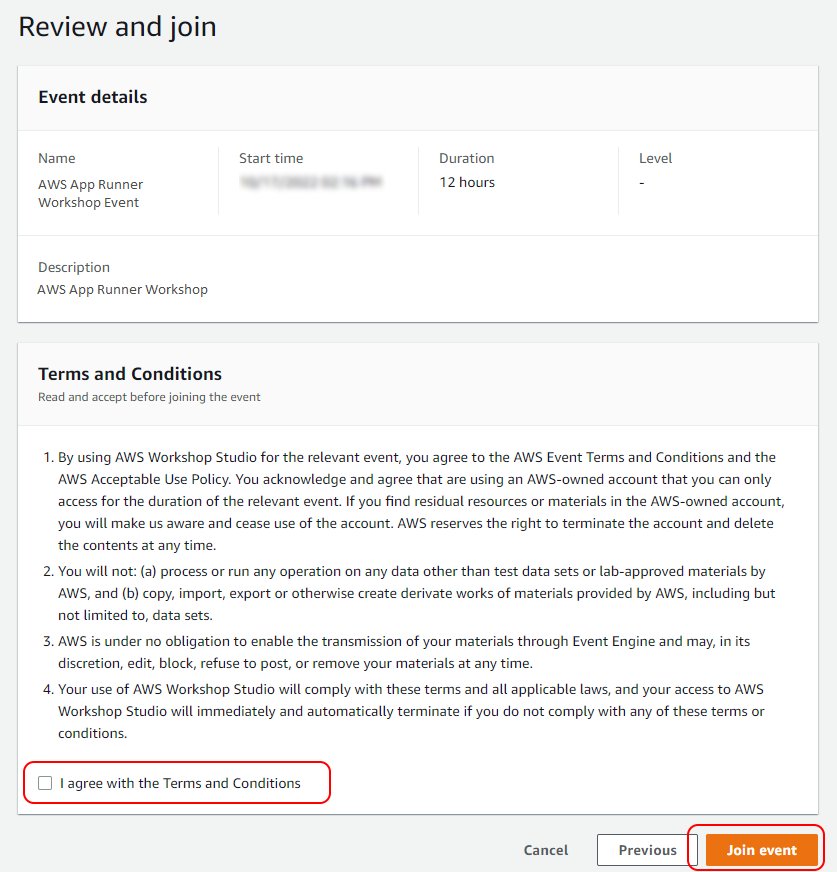


If you are prompted for an event code, enter the value supplied by the lab instructor.

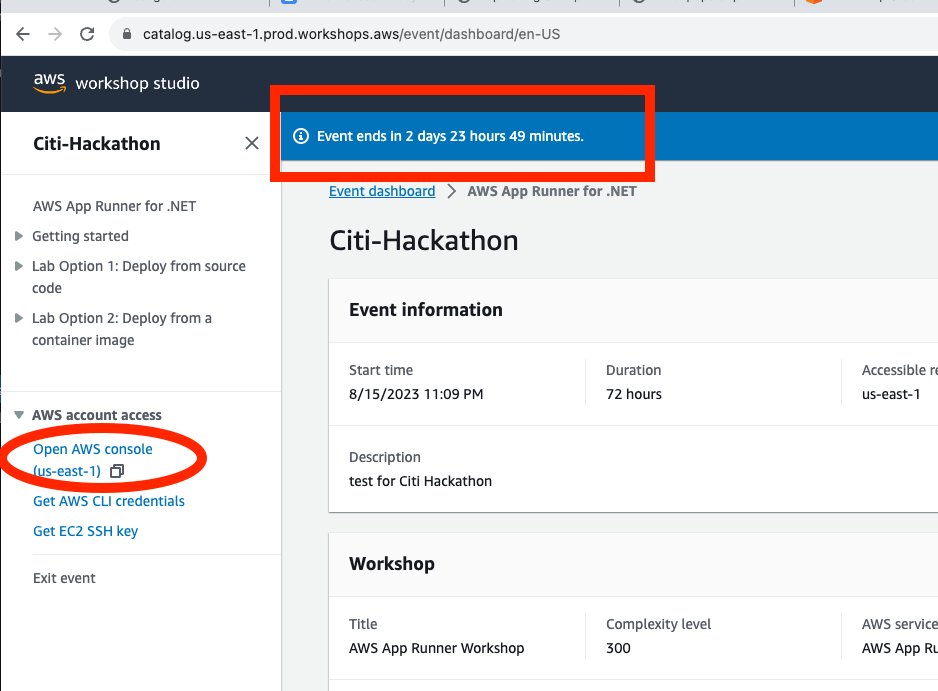


Finish the login process

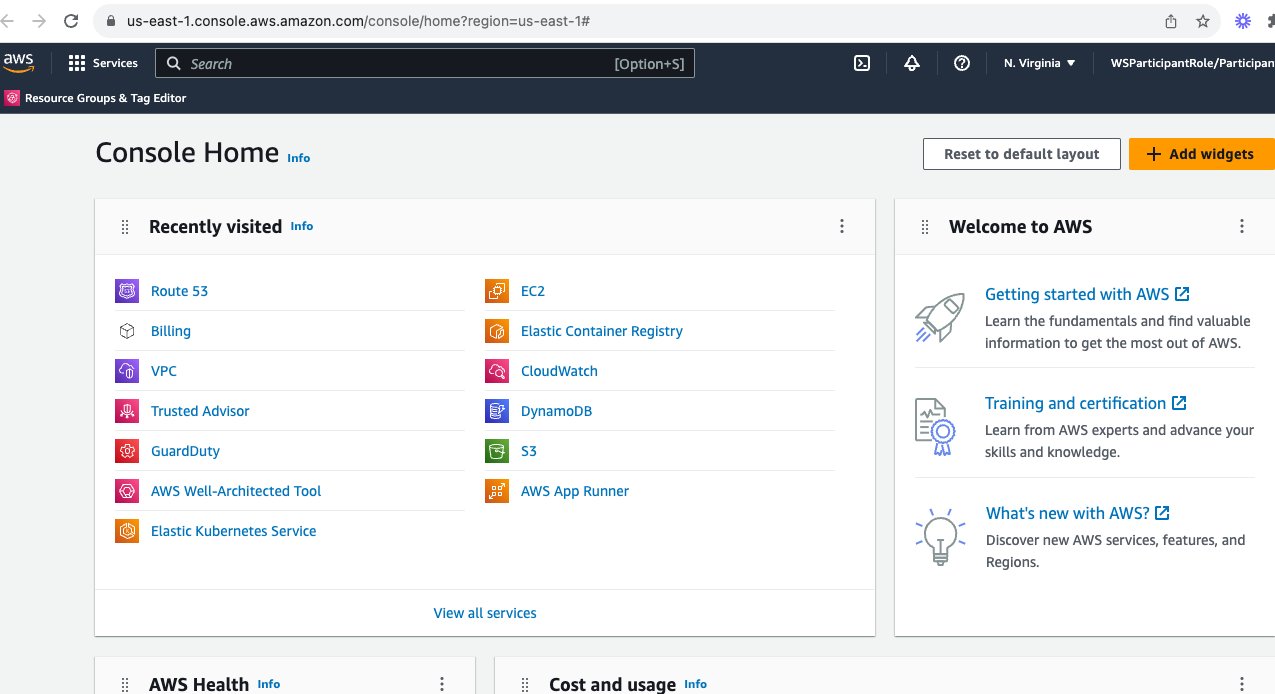
You will be prompted to accept the terms and conditions of the event. Read the page, and then select the **I agree with the Terms and Conditions** checkbox. Choose **Join event**.



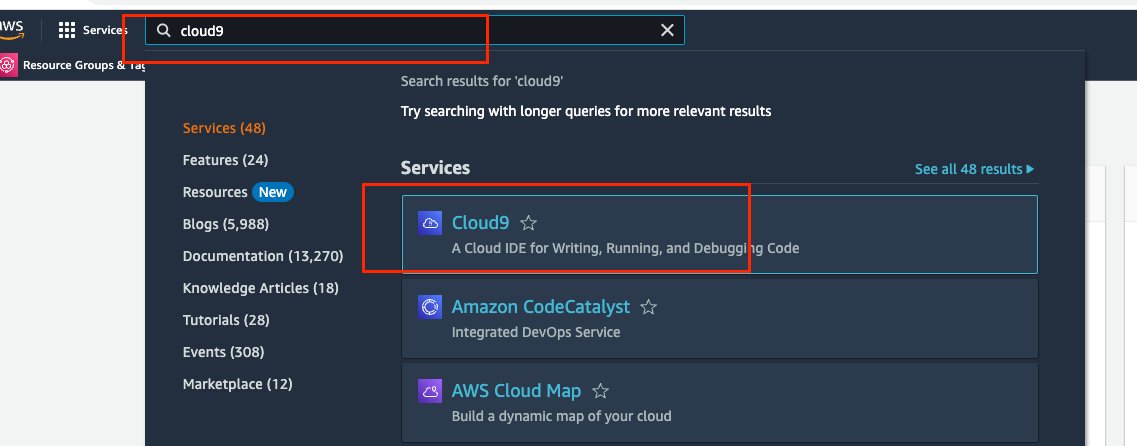
The workshop dashboard will be shown. At any time, to log into the AWS console, choose the **Open AWS Console** link in the **AWS account access menu**.

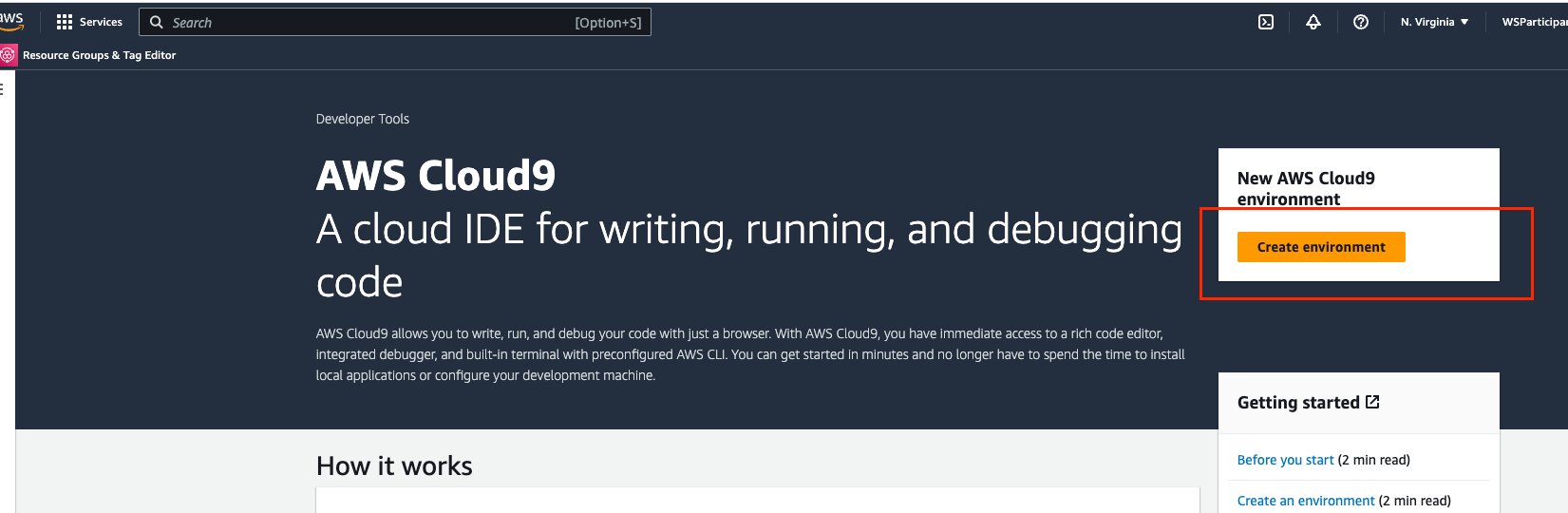


 Make sure you have access to AWS Services in the Console as shown below

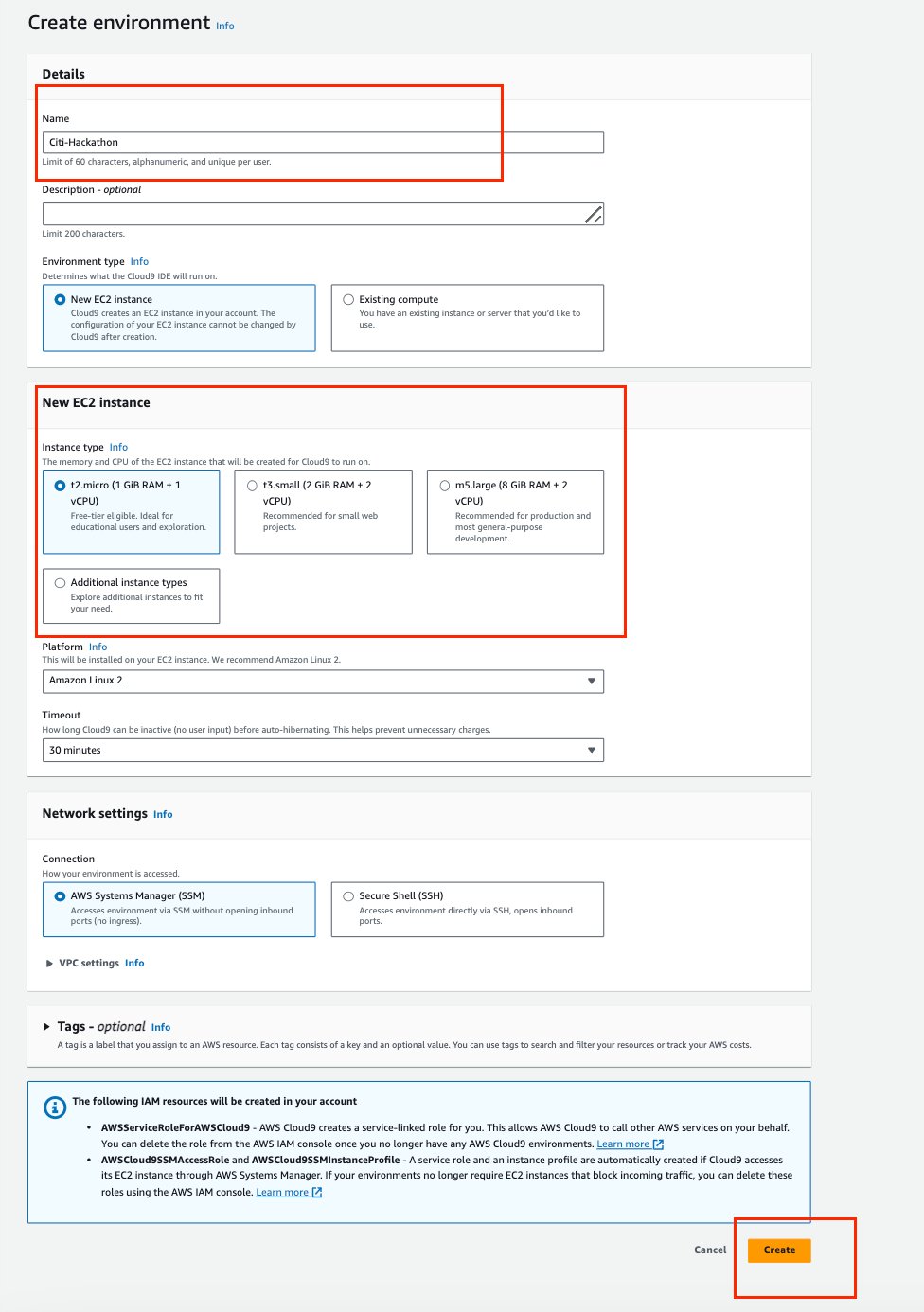


Create a Cloud 9 Environment which will help on executing the AWS CLI commands and running big jobs ( if needed ) without disturbance

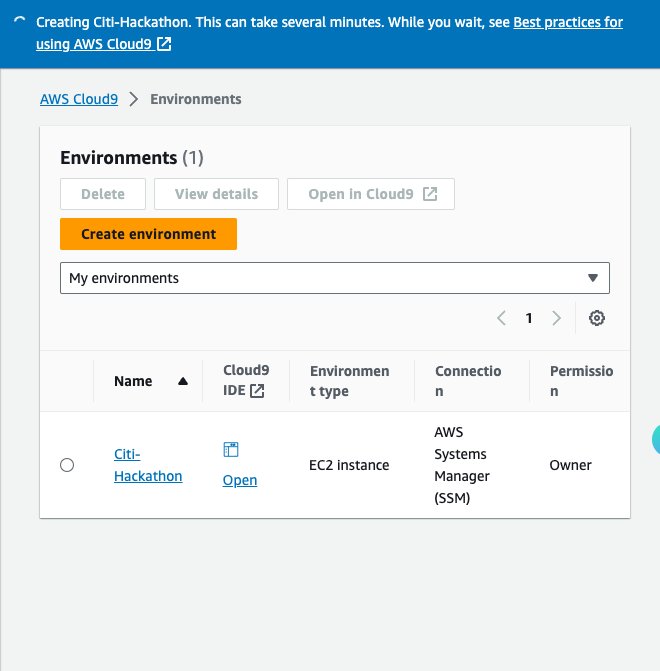




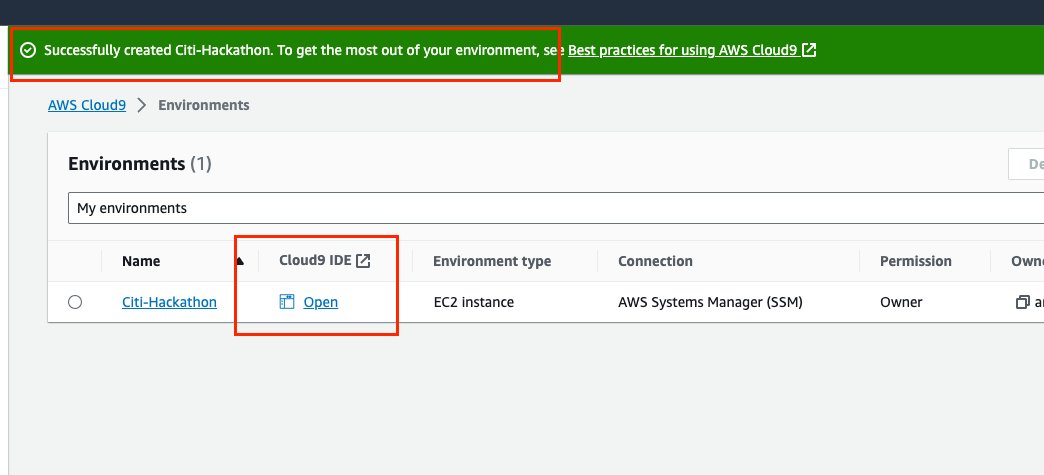
Give it a name and change the EC2 Instance size to be t3.small - Then Create the environment without changing any other configurations.



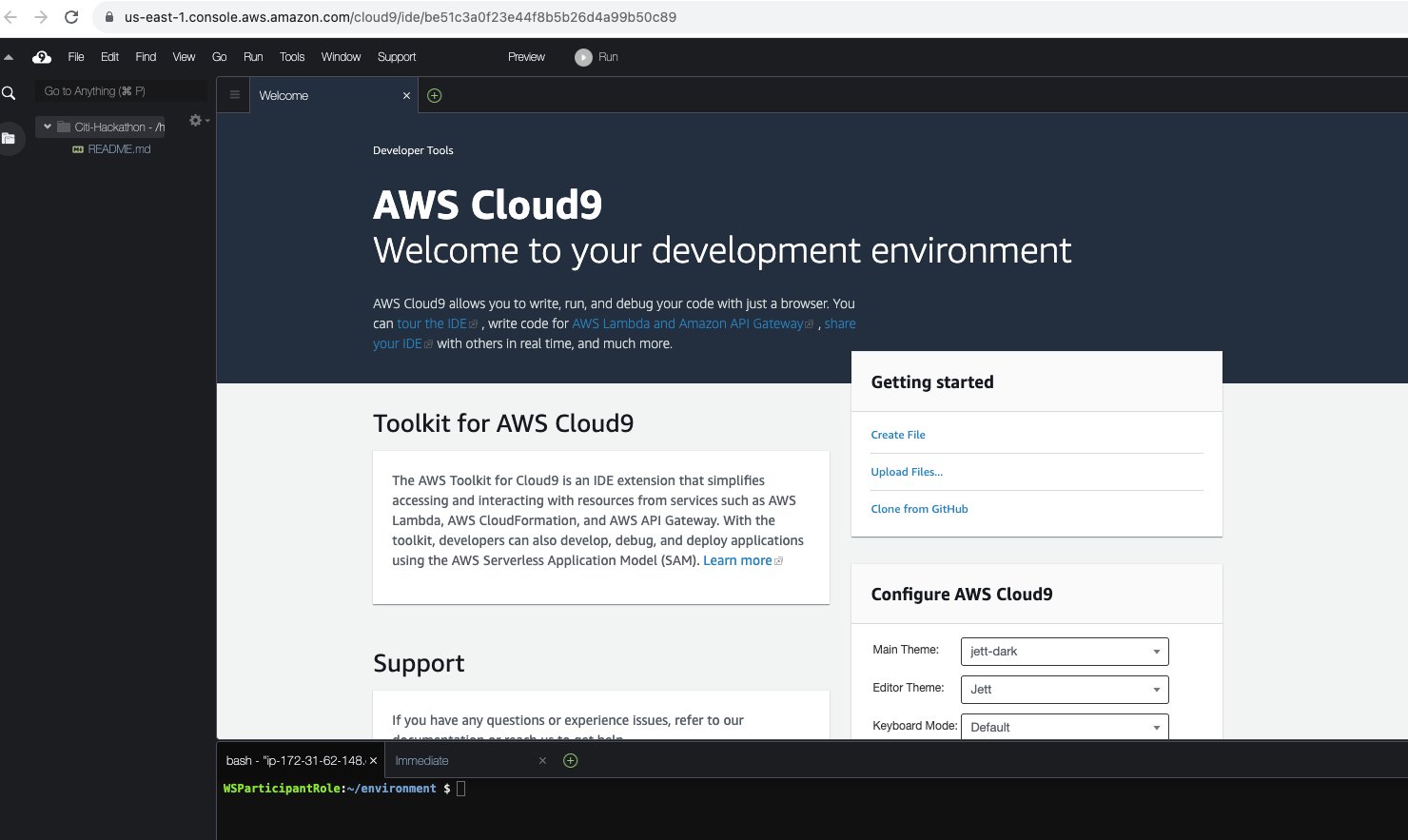
This can take couple of minutes to create your EC2 Instance for Cloud9



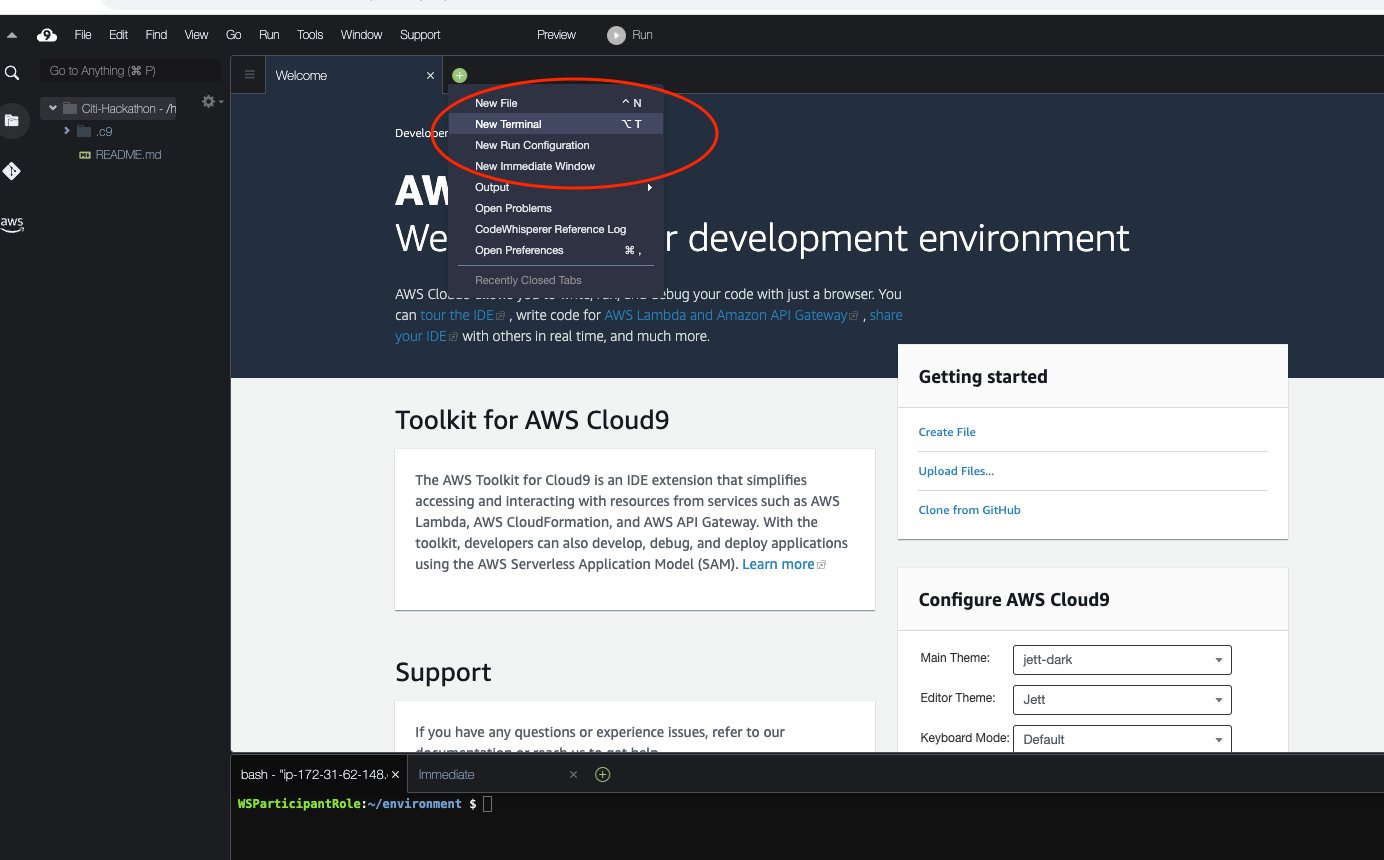
Once creation completed Successfully, Click Open :

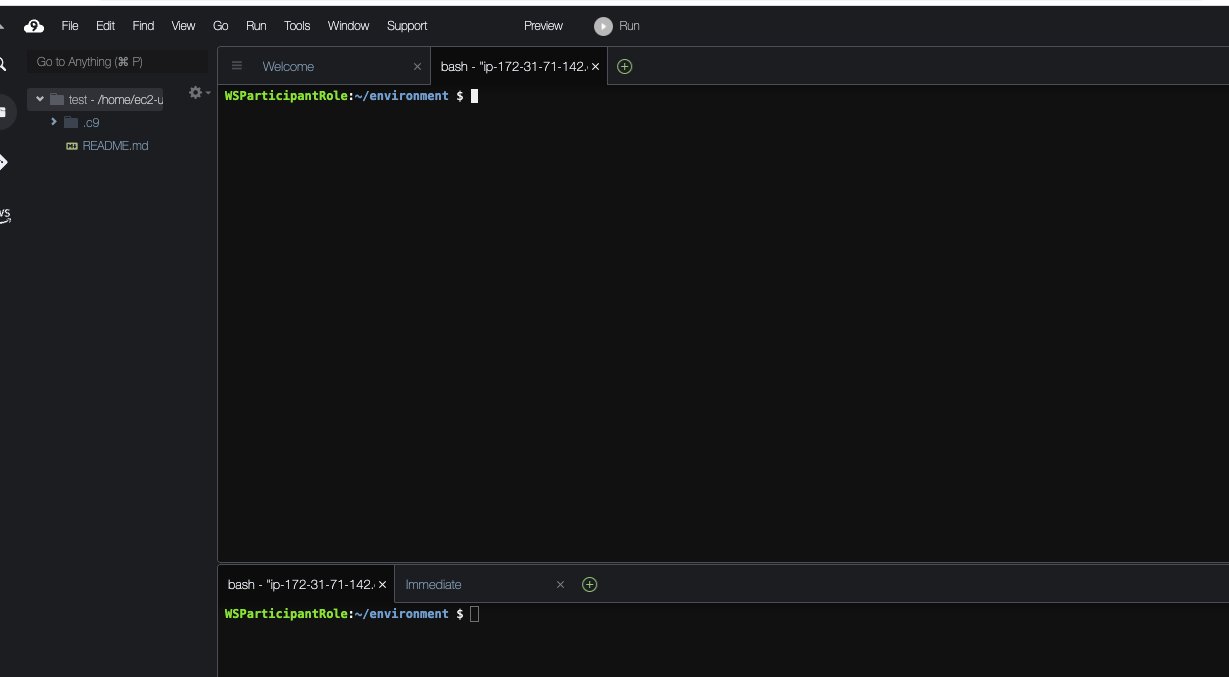


 This is the Terminal of your Cloud9 Environment ( Ec2 Instance )

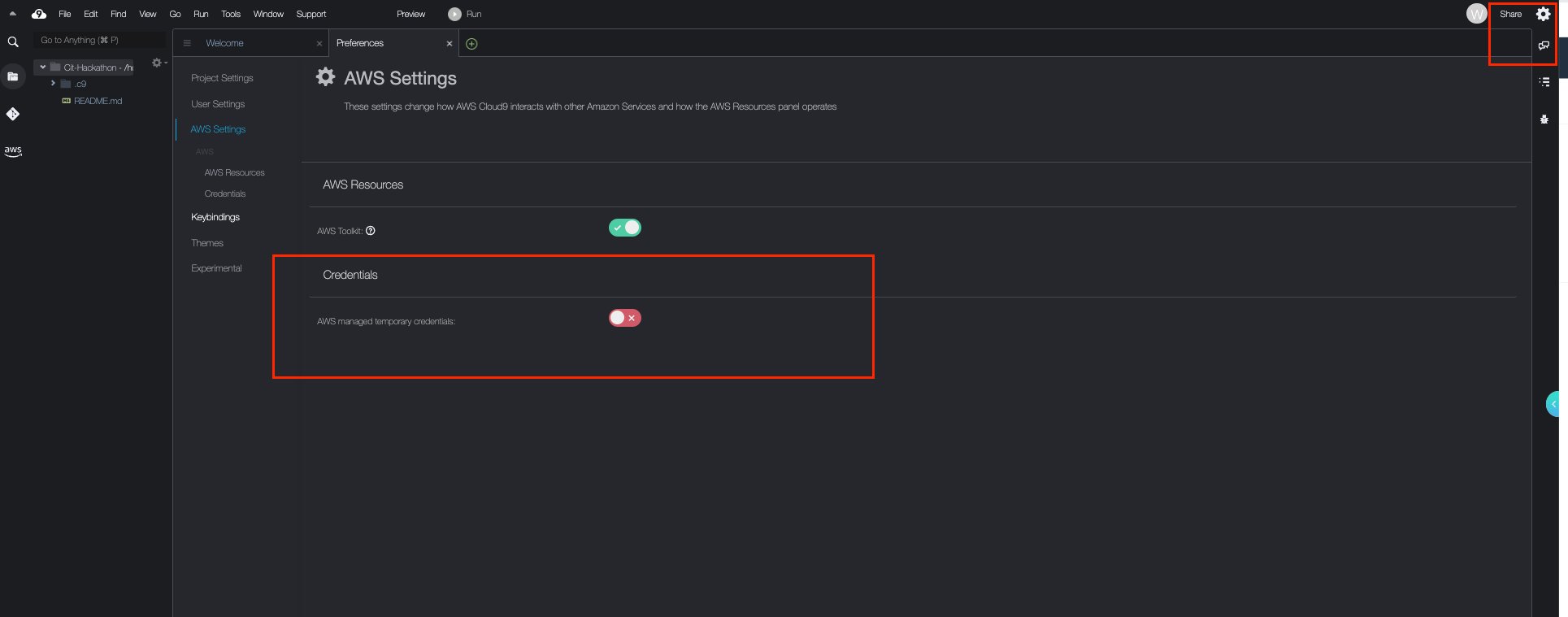


You can open a new terminal using the  green “+” sign as shown below:





you need to disable the AWS Temporary credential from the Cloud9 machine from the machine Setting , you can find setting button on the up right corner then make sure AWS managed temporary credential is disabled as shown below:

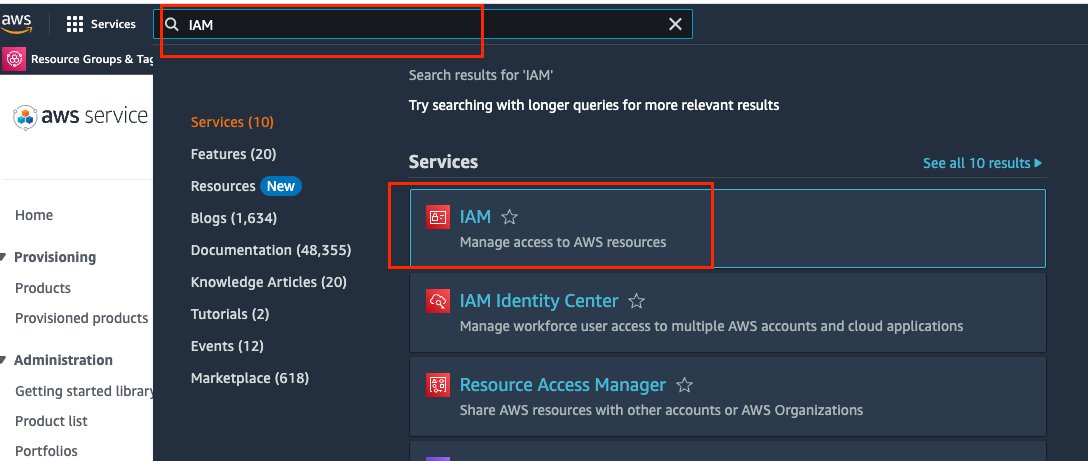


Now, We need to assign a proper IAM Permission to Cloud9 EC2 instance to be able to execute AWS CLI commands on other AWS Services like DynamoDB and S3

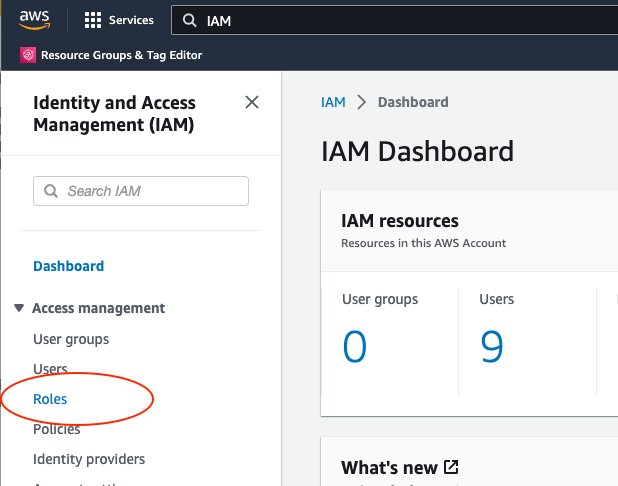
First you need to create Admin Role to attach it to Cloud9 :

**Please Note: for the Demo purposes we choose the Full Access permissions for Cloud9 Ec2 instance but best practice here is to always give the least privilege permissions .**

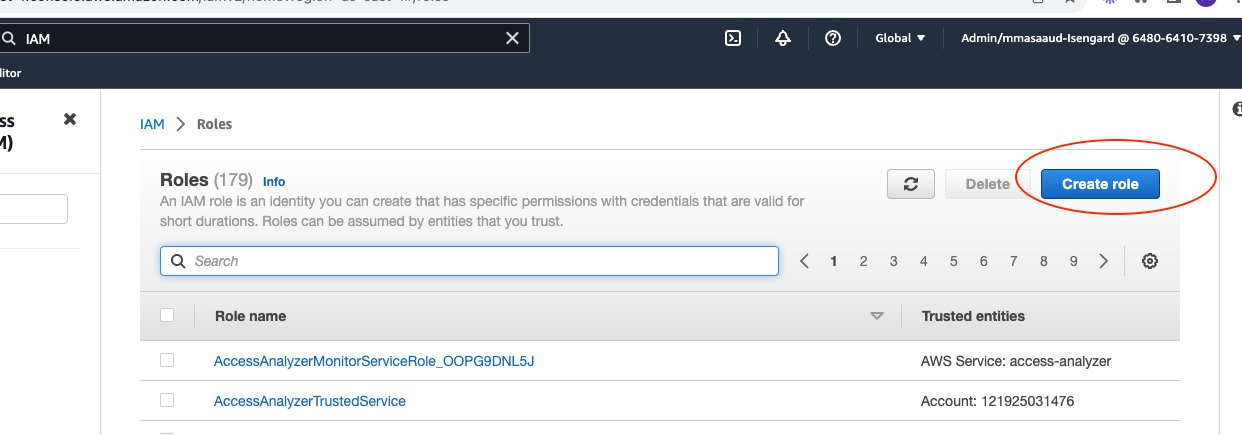
Go back to AWS console and Search for IAM on the Service search bar as below:

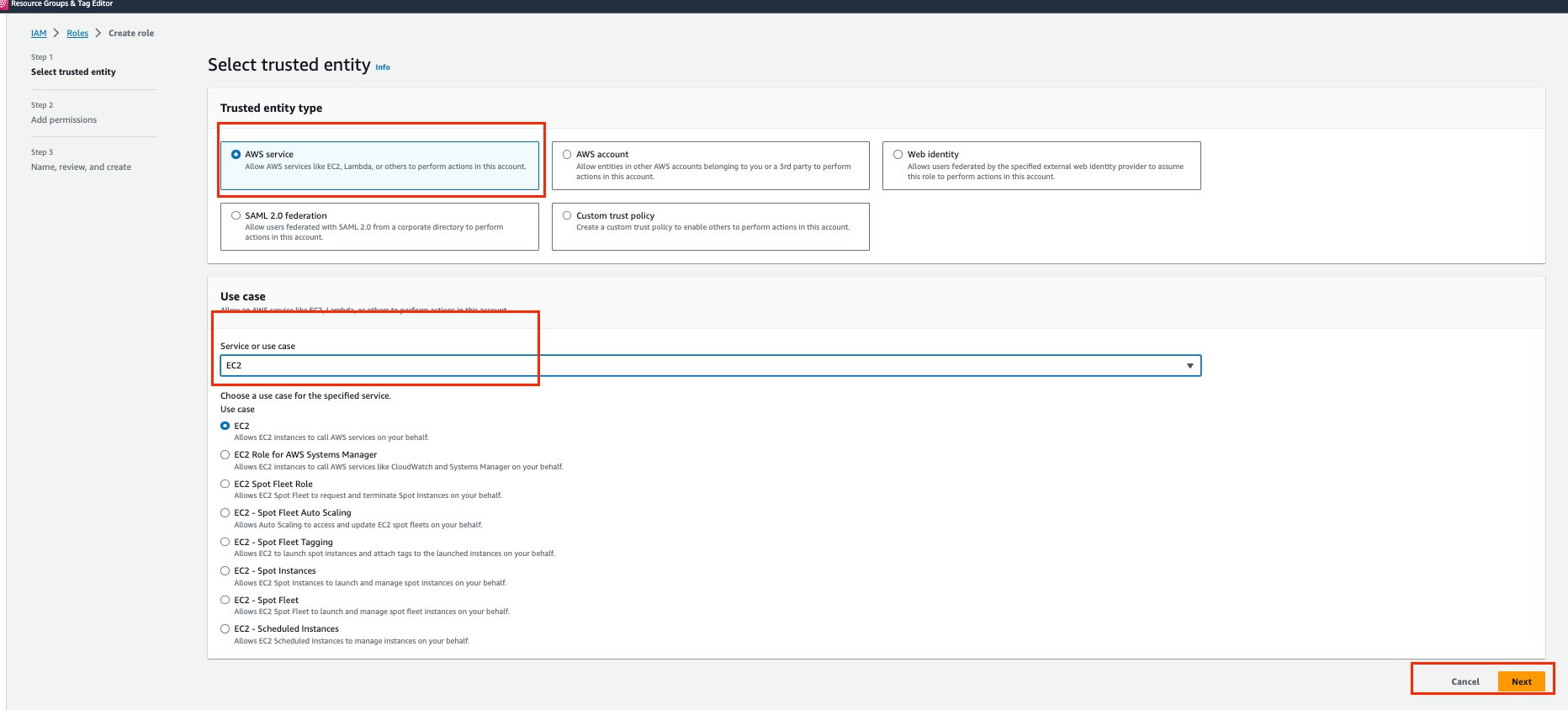


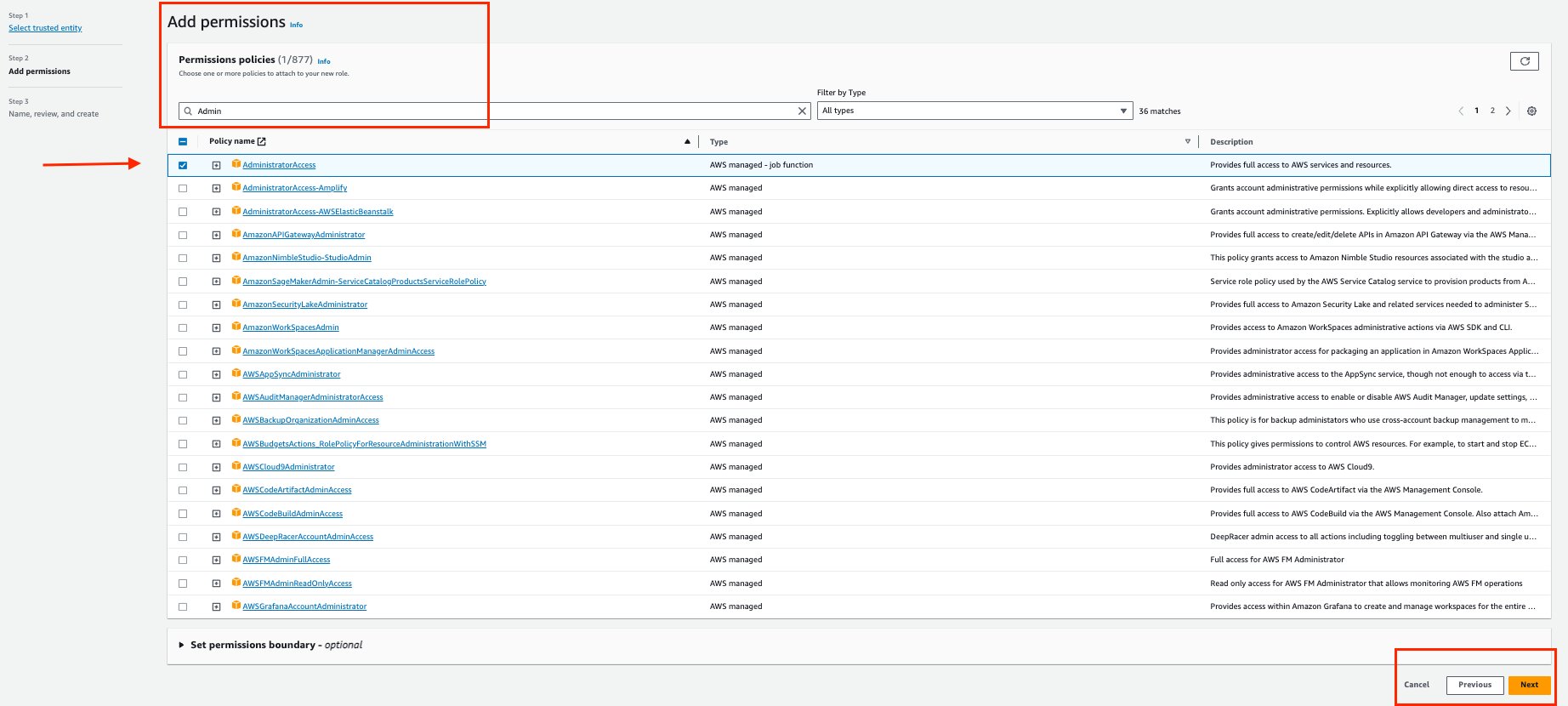
Go to Roles on the left side:



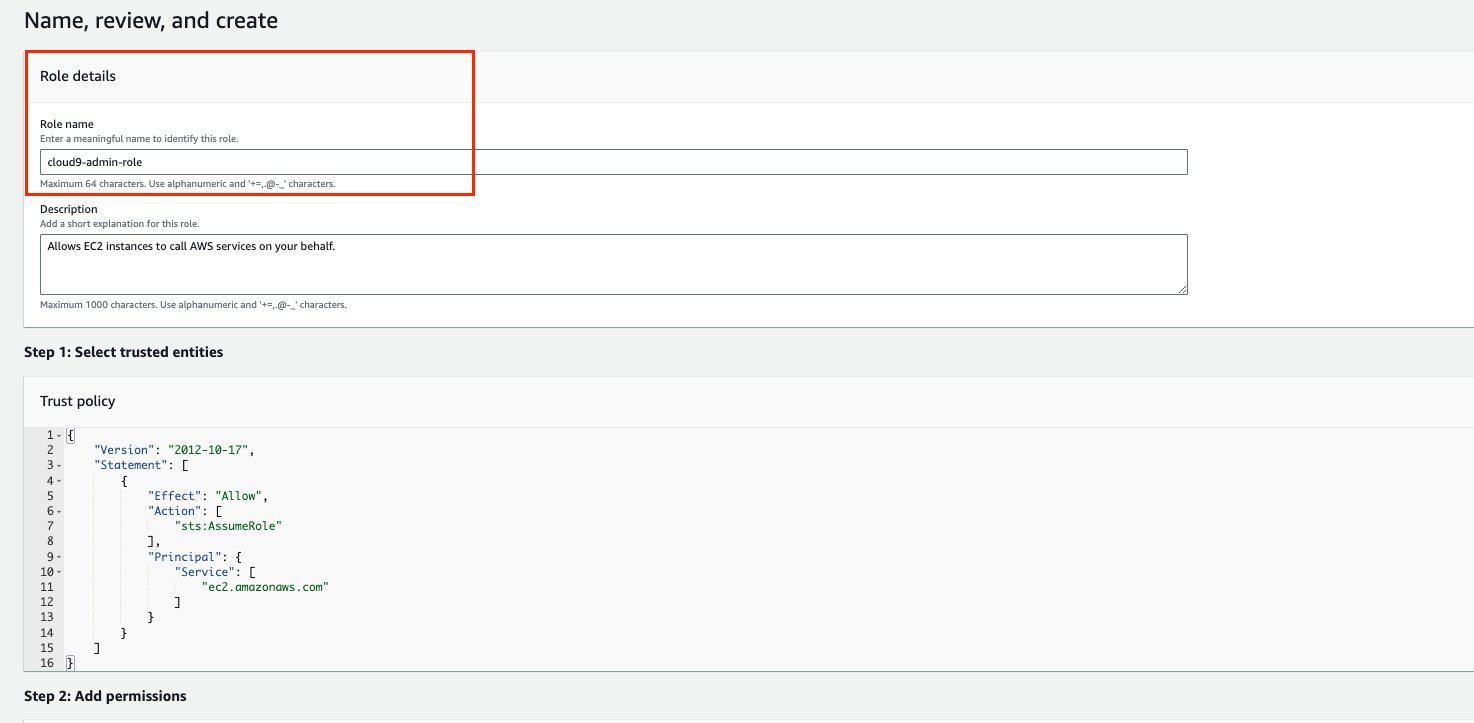
Then create a new role which will be attached to EC2 Instance:

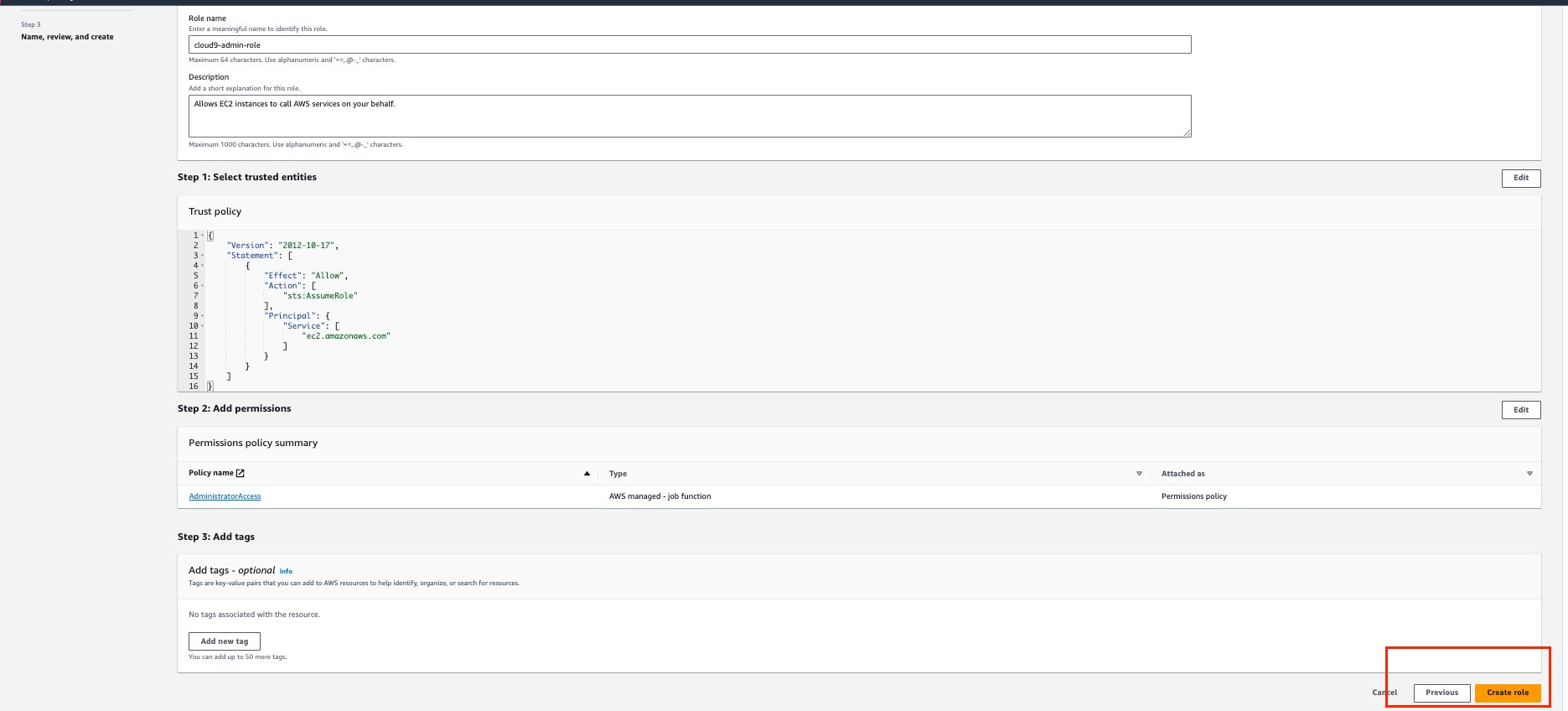




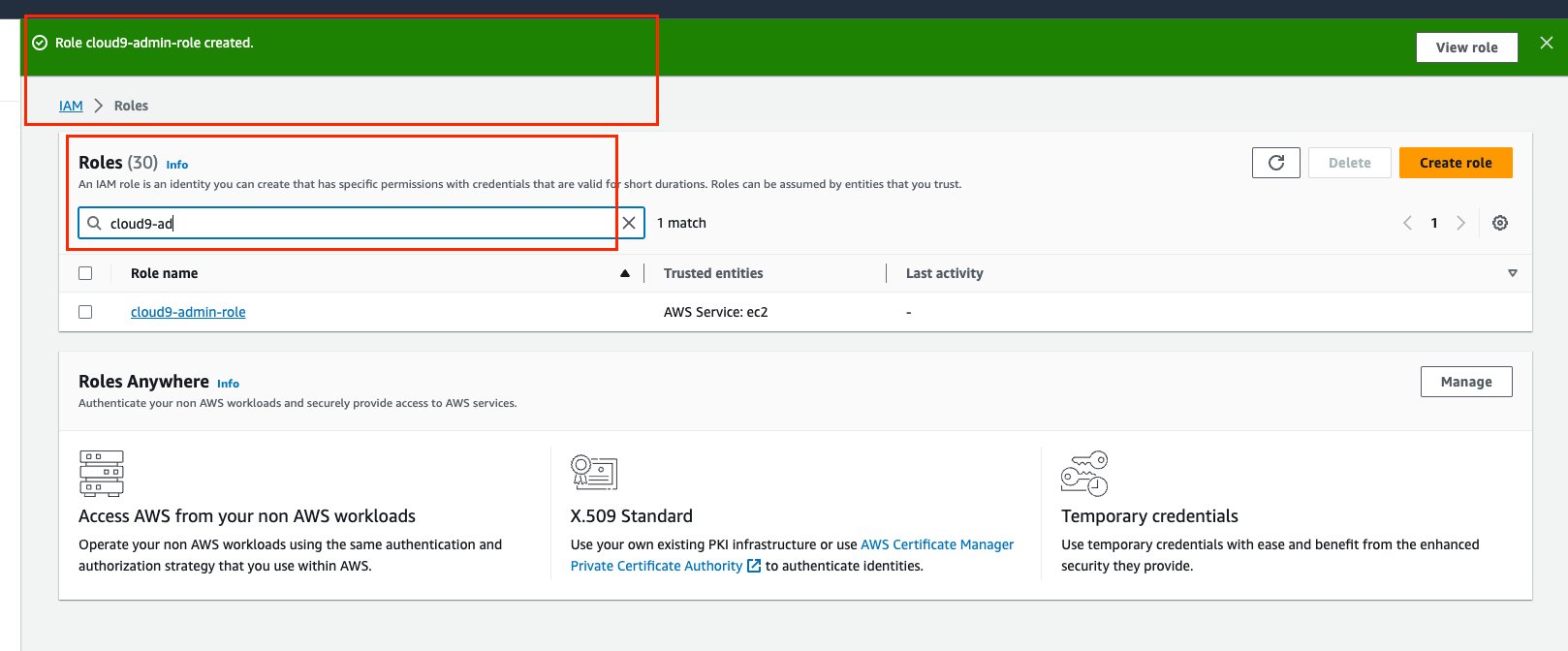


give a Role Name ( Please keep a record of that Role name as we need while assigning it to the Ec2 Instance for Cloud9 :



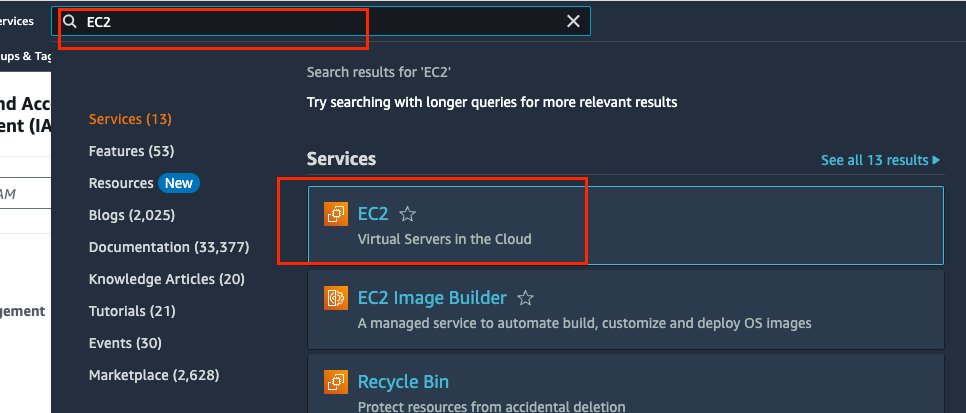


To validate role creation please type in role name created on Search and make sure it got created Successfully

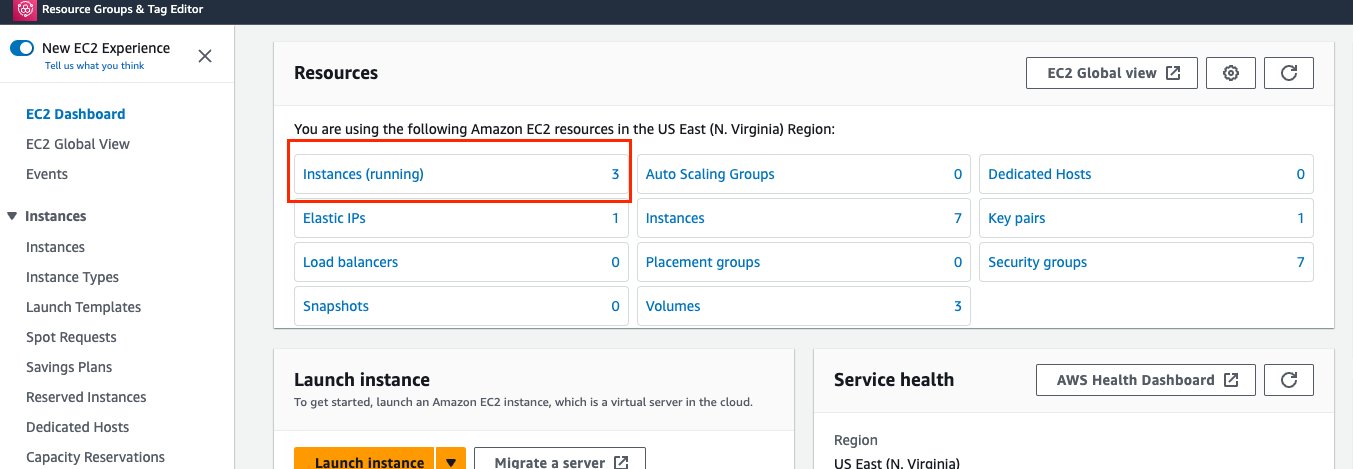


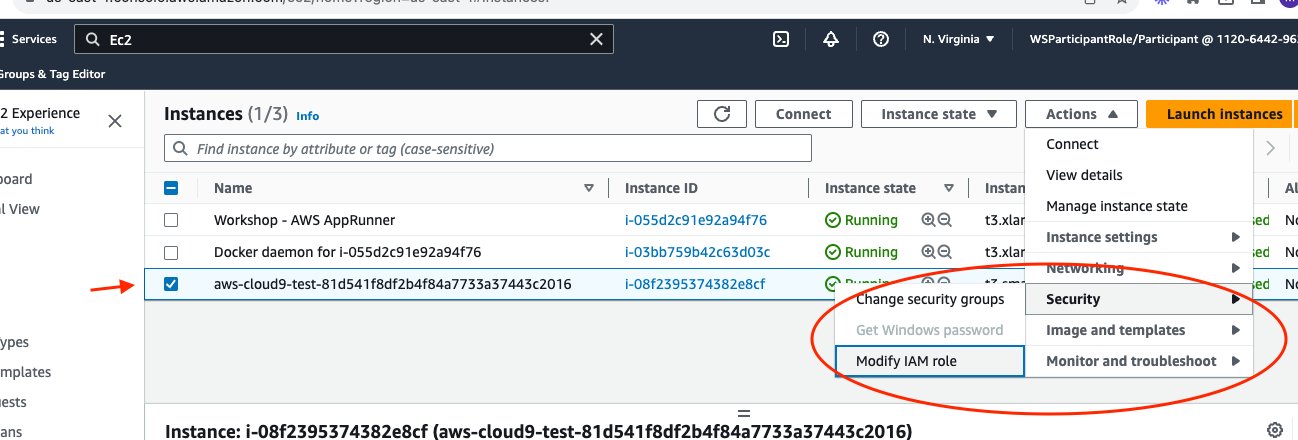
After creating the IAM Role you will need to attach it to the EC2 instance to enable Cloud9 to execute any AWS command needed based on the permissions you gave on that Role:

back to AWS Console and Look for EC2 as shown below :

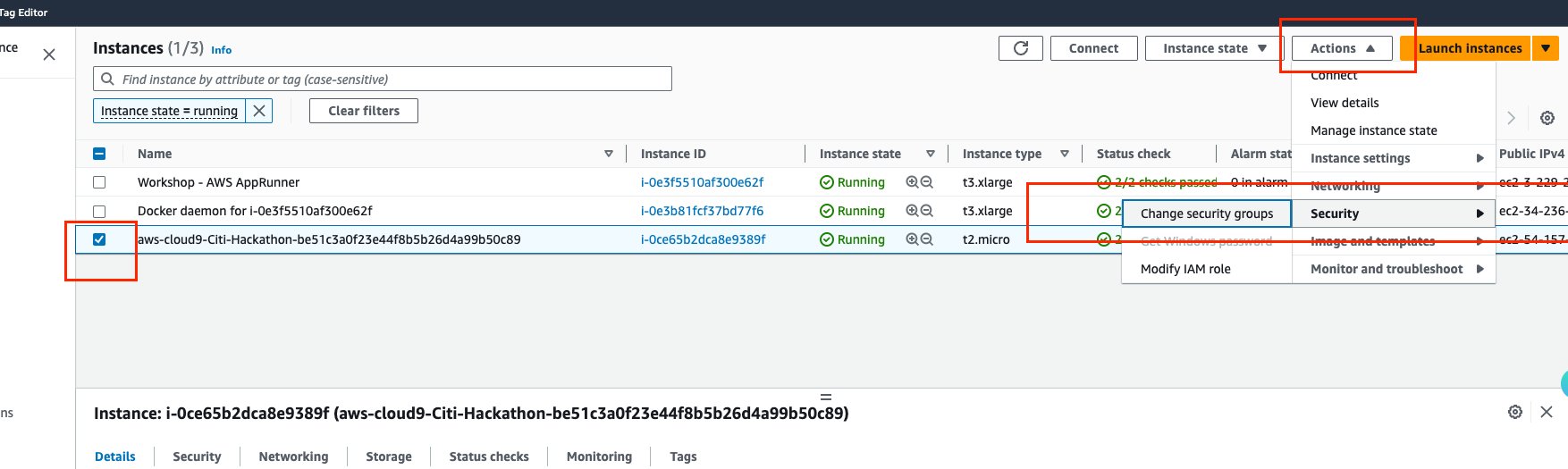


Click on the “ Instance Running”

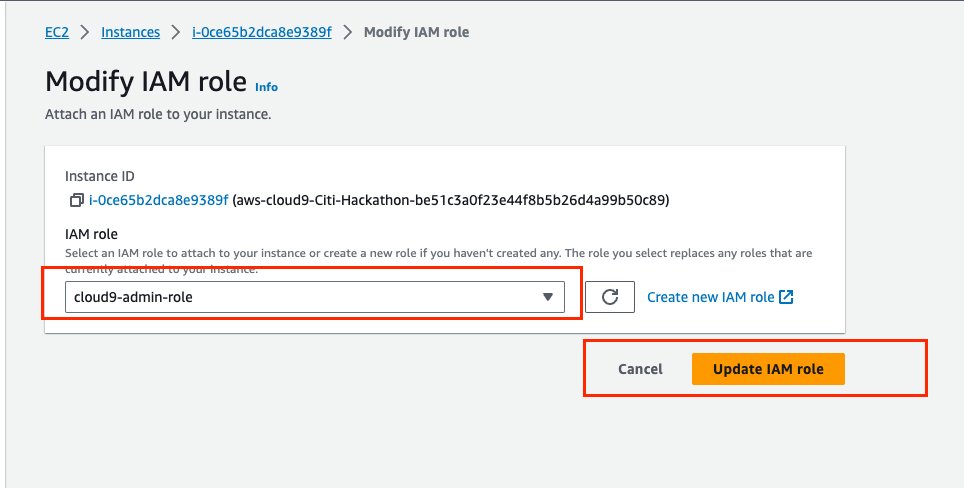




check the box for the EC2 instance related to Cloud9 then go to Security > Modify IAM Policy :



Select the newly created admin role and click Update IAM Role them Click on Update IAM Role :



After assigning the Proper Permissions to the Cloud9 machine, you can now you can execute AWS commands from Cloud9 environment  as shown below :

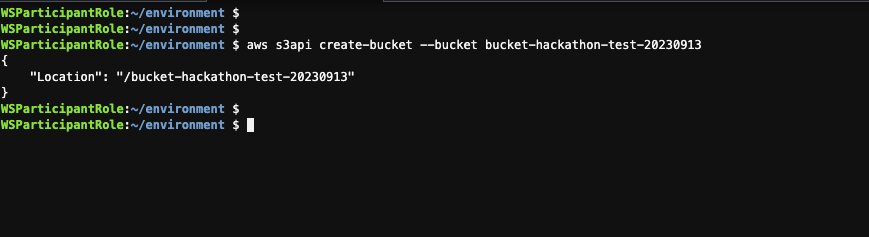
**1- Create S3 Bucket**

give example from random number to actual number , open up word or notepad and past that there to copy it on AWS CLI

on Cloud9

**S3 bucket name must be Globally unique, so please make sure to add random number or date to make it unique:**

aws s3api create-bucket —bucket bucket-hackathon-test-<random number>



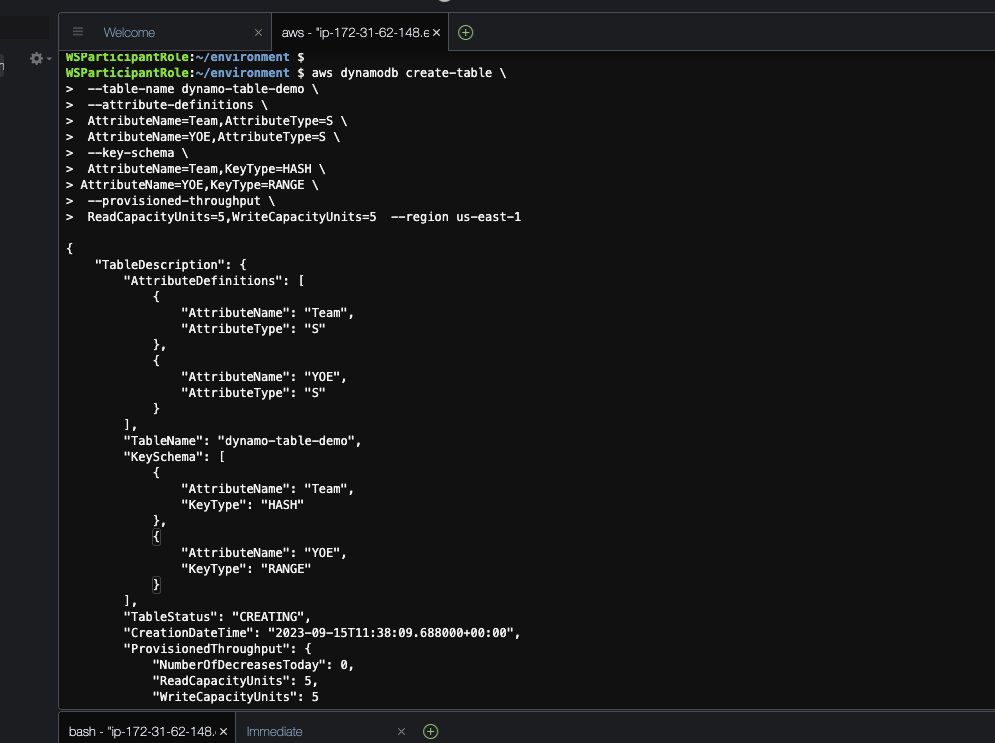
**2- Create dynamoDB table**

For my Sample Code I’m using DynamoDB for my no-SQL Database, here is the command to  create the table :

**Note:**

**Please make sure to use the right region name while excusing that command below, for example I’m using us-east-1 so the command should be like below :**

aws dynamodb create-table \  
 --table-name dynamo-table-demo \  
 --attribute-definitions \  
 AttributeName=Team,AttributeType=S \  
 AttributeName=YOE,AttributeType=S \  
 --key-schema \  
 AttributeName=Team,KeyType=HASH \  
AttributeName=YOE,KeyType=RANGE \  
 --provisioned-throughput \  
 ReadCapacityUnits=5,WriteCapacityUnits=5  --region us-east-1

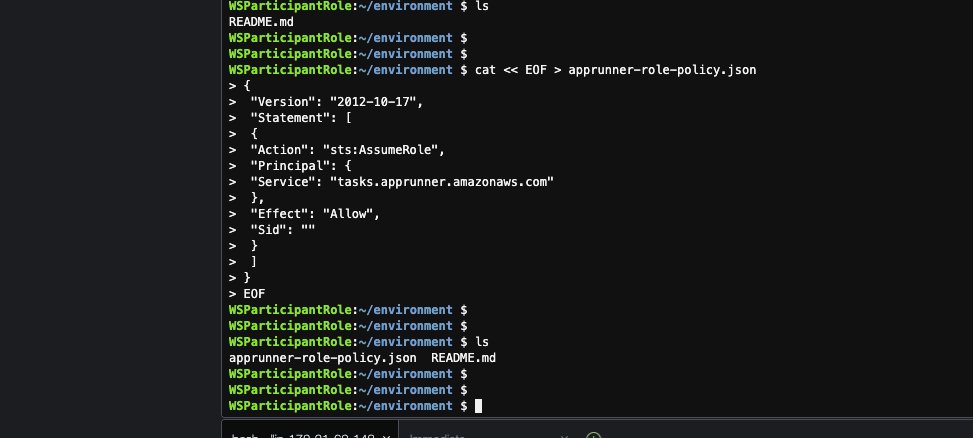


**3- Create IAM Role for app Runner:**

**Please Note: for the Demo purposes we choose the Full Access permissions to DynamoDB , best practices is to give the least privilege permissions for AppRunner which enable it to execute the code on the targeted services.**

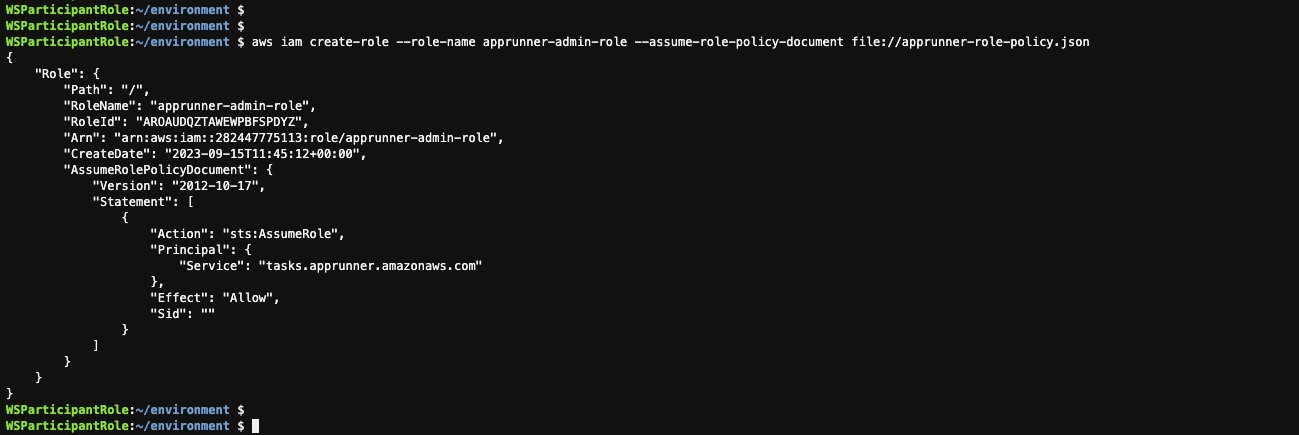
Finally, We need to assign IAM Role for Apprunner Service to enable the service communicate with S3 and DynamoDB while running the code, please follow the steps below so Apprunner can communicate with your Github account and with other AWS Service :

cat << EOF > apprunner-role-policy.json  
{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Action": "sts:AssumeRole",  
 "Principal": {  
 "Service": "tasks.apprunner.amazonaws.com"  
 },  
 "Effect": "Allow",  
 "Sid": ""  
 }  
 ]  
}  
EOF



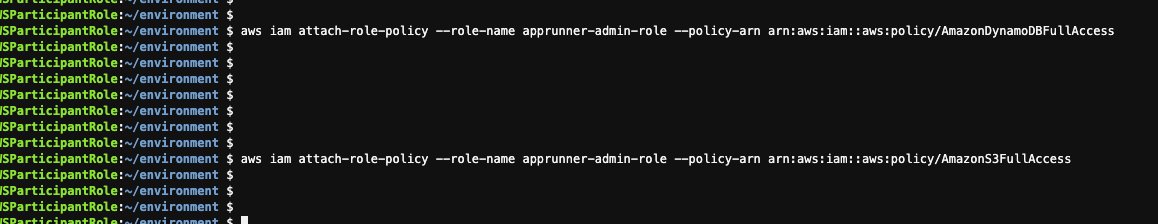
**Create IAM Assume Role for apprunner to enable it communicate with Github and other API as needed :**

aws iam create-role --role-name apprunner-role --assume-role-policy-document [file://apprunner-role-policy.json](file:///apprunner-role-policy.json)

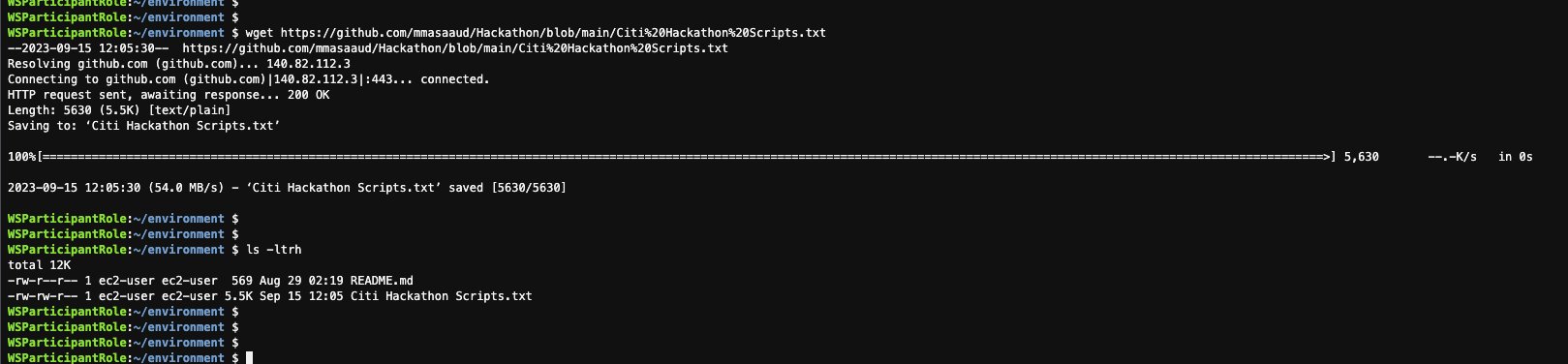


**Give also permissions to APprunner to communicate with DynamoDB and S3 as We need it to run the code there**

aws iam attach-role-policy --role-name apprunner-role --policy-arn arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess  
  
aws iam attach-role-policy --role-name apprunner-role --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

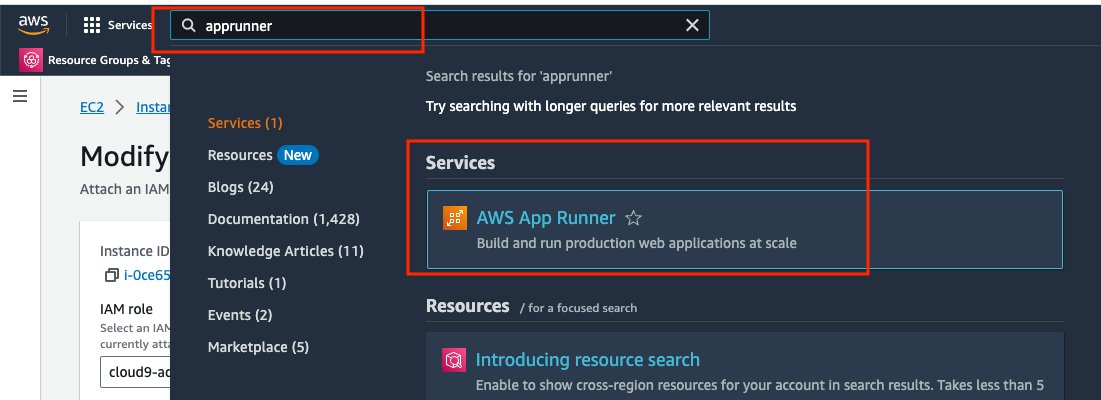


**Optional Step** You can also download all those commands executed previously from Github using the command below that might easier for copy/paste commands from there directly:

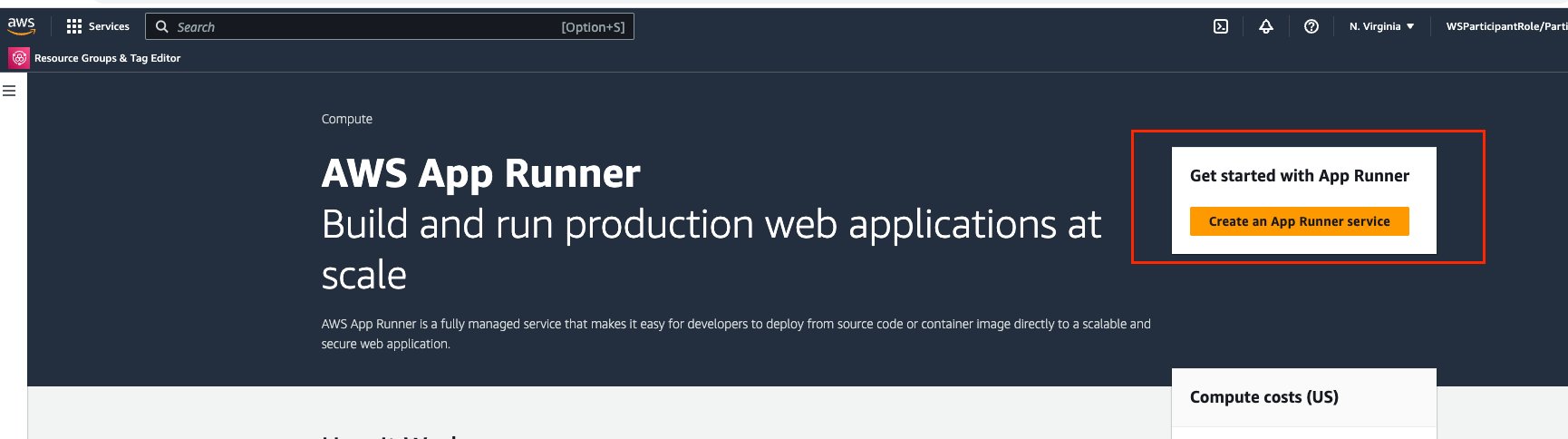


**AppRunner Code execution :**

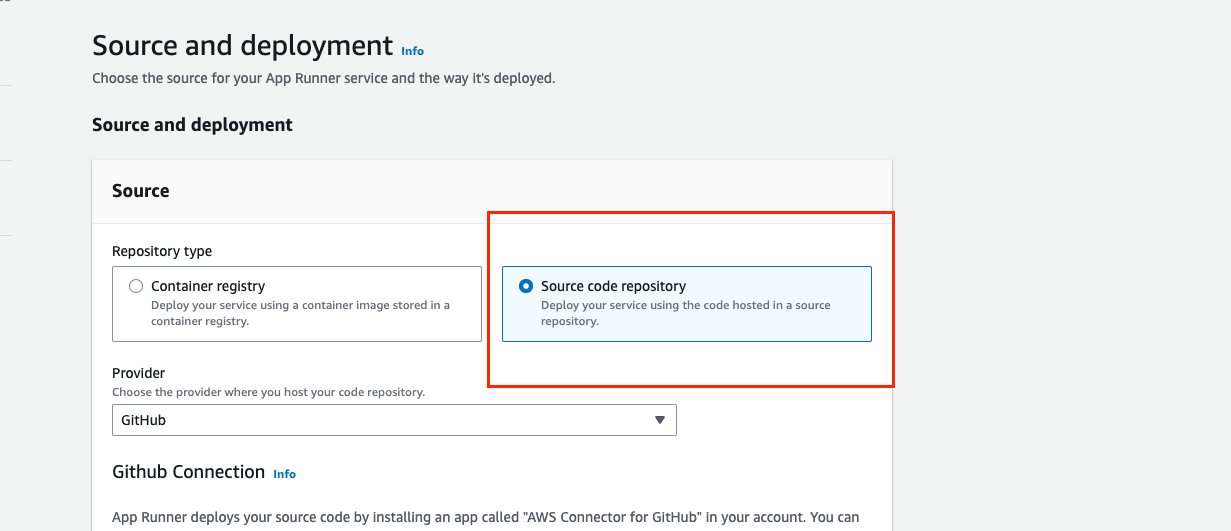
Back to AWS Console and look for Apprunner Service on the search bar :



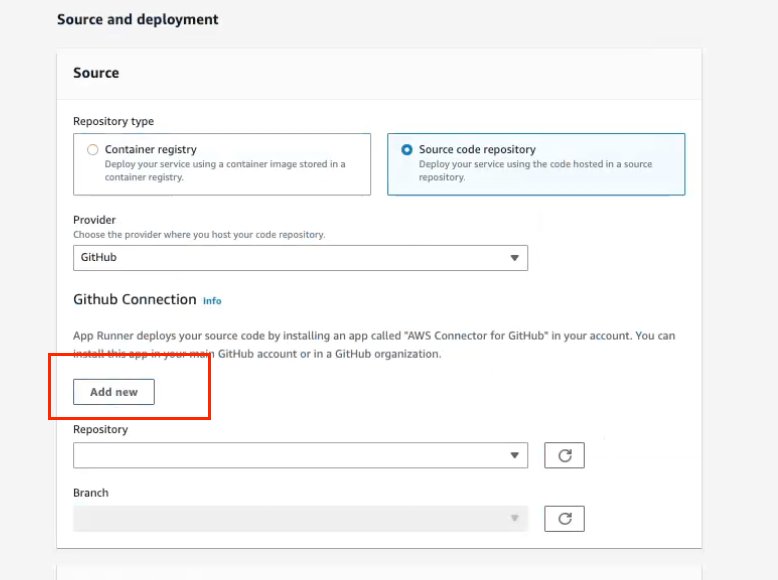
Click on create a new Service :

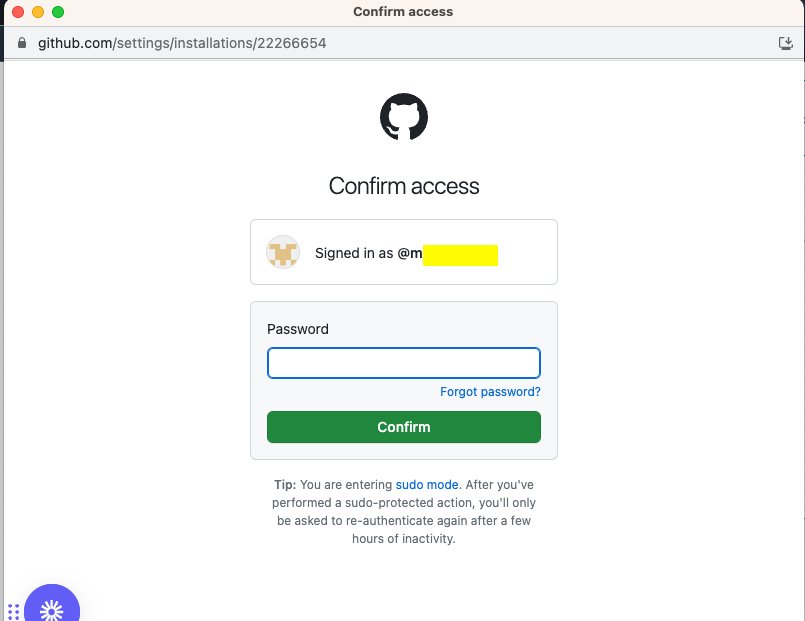


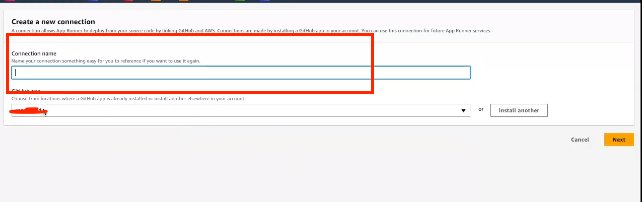
Choose your repository type either you can use the source code directly from your Github account or you can use it from a container image which you uploaded already to Amazon ECR, for the sample code I’m using Source Code repository directly from Github as shown below



Then you need to give your Github credentials to Apprunner so it can access your repositories from Github as shown below:







**For this Sample application repository, please use the Github link on the reference section and fork the repository into your personal account so you can create the Apprunner Service using that forked Repo.**

After Choosing the required repository and the right branch, you would need to choose the Deployment Setting as shown below, you have two option:

**1- Manual:**

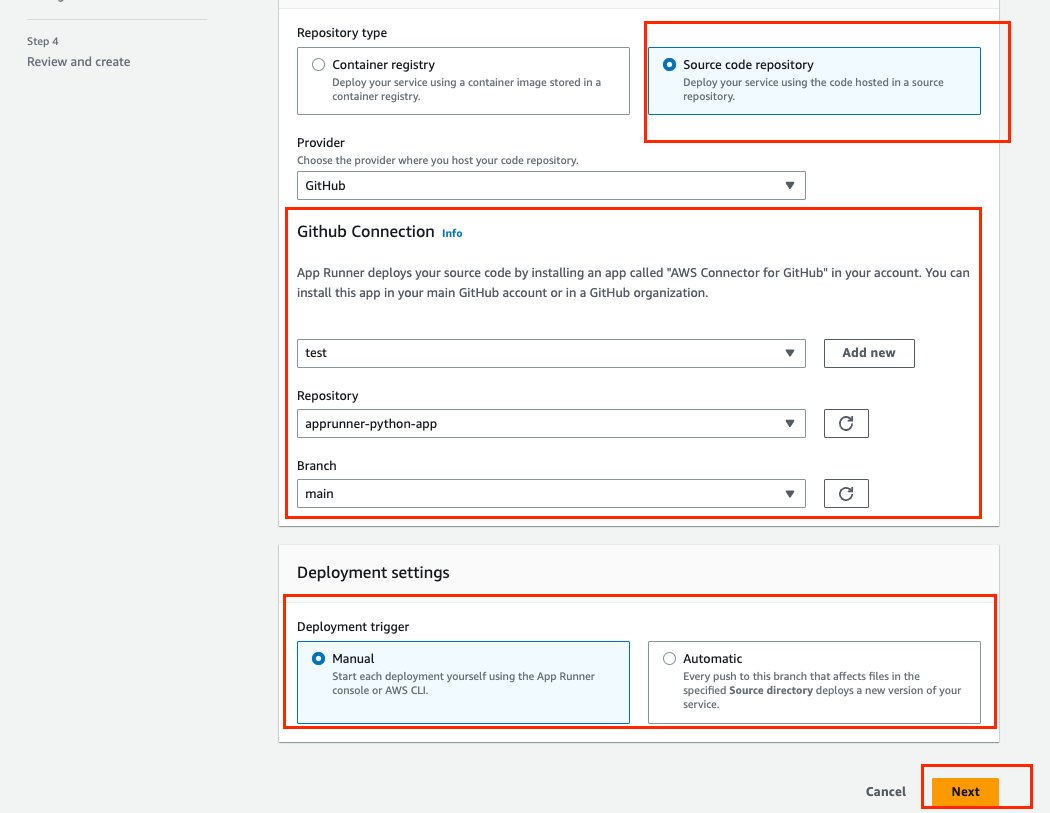
Which will not reflect any changes happening on your repository unless you explicitly restart your service after you commit your code changes on Github.

**2- Automatic:**

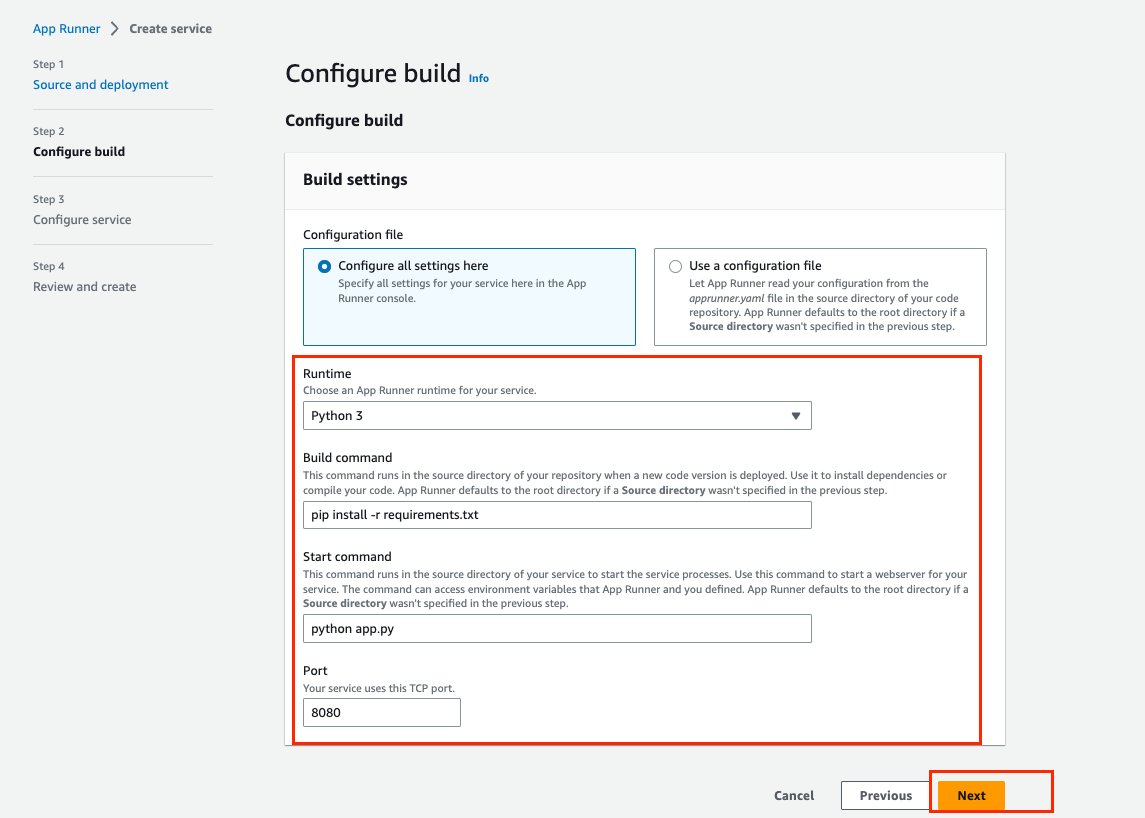
This will instantly restart your running sevice once any changes committed on Github repository without any user interaction.

For more information please check out Deployment methods details. here:

<https://docs.aws.amazon.com/apprunner/latest/dg/manage-deploy.html#:~:text=Deployment-,methods,-App%20Runner%20provides>



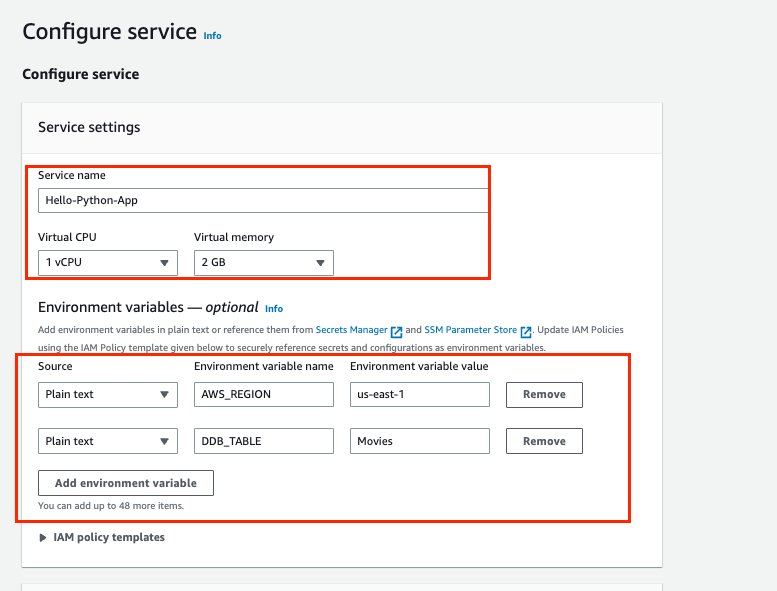
Next step is to choose your Runtime configuration, Build Command, Start Command and port number based on the language used in your code, I’m using Java for that Sample code so my configurations like below:



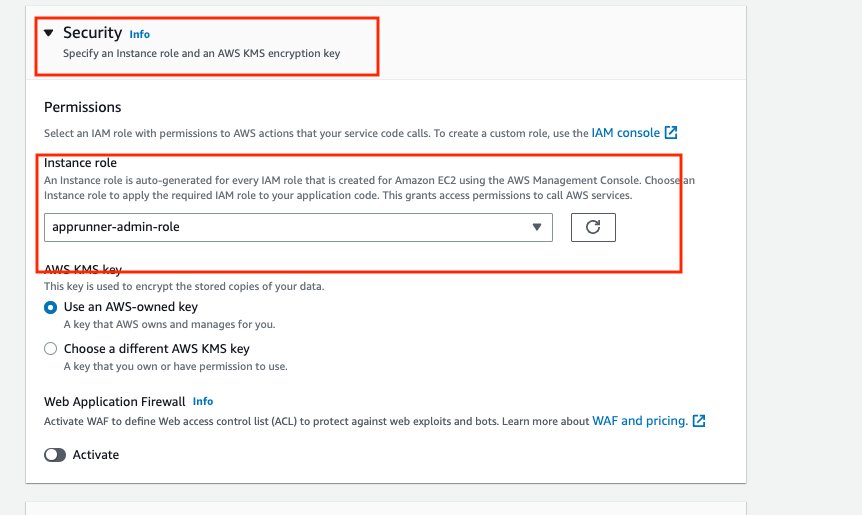
Give a service name and make sure it would have enough CPU and Memory to start your application, one of the main features of using Apprunner that it takes care of compute resources scalability without any manual interaction from the user so even if you have spikes happening apprunner will scale the resources accordingly based on the number of HTTP/HTTPS request received

For my Sample Code I need to set couple of  Environment variables so click on Add environment variable and add the following two environment variables:

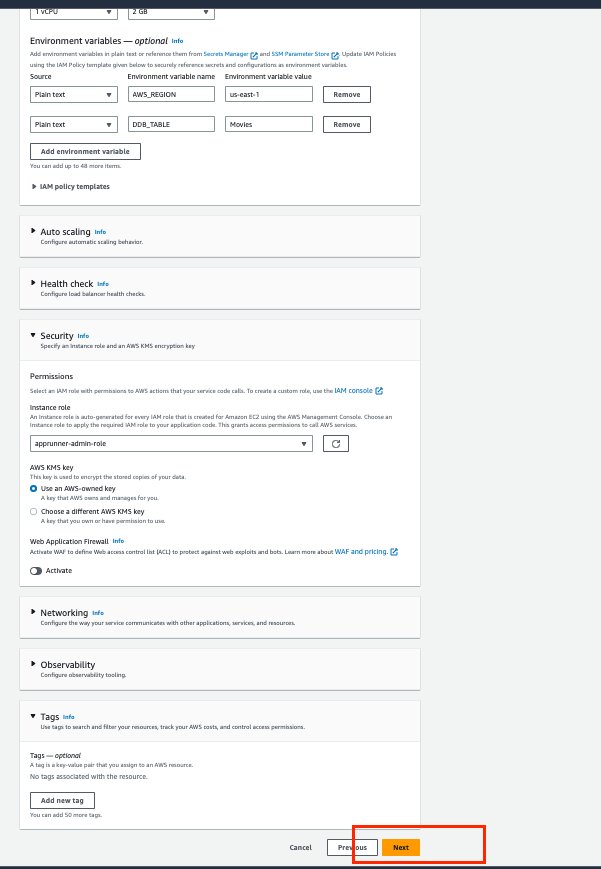
|  |  |  |
| --- | --- | --- |
|  | Key | Value |
| 1 | AWS\_REGION | AWS Region ID (Eg: us-east-1 ) |
| 2 | DDB\_TABLE | Movies |

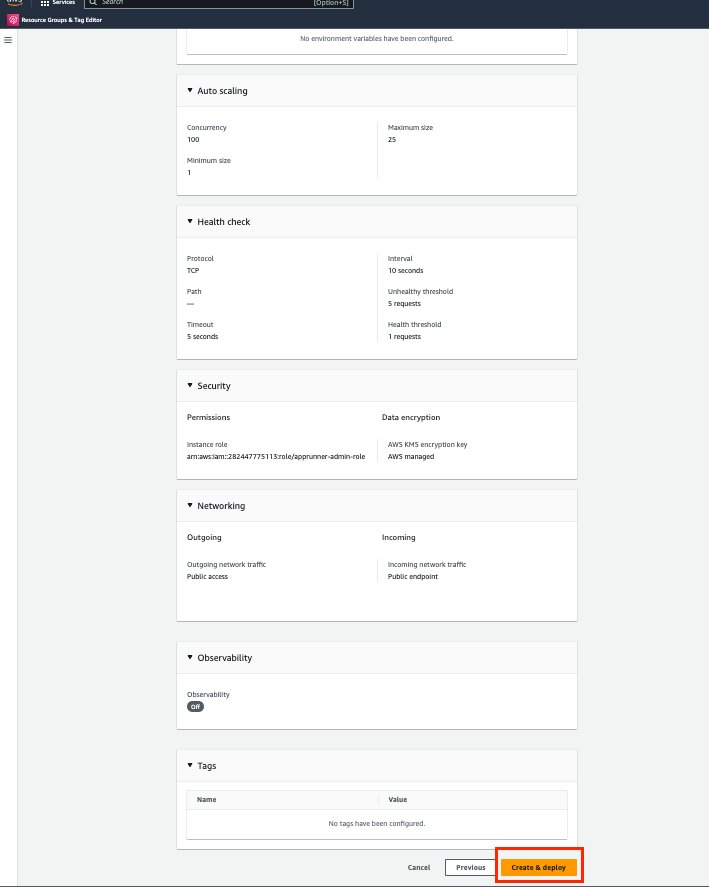


Scroll down on the same page to the Security Section and make sure to choose the IAM Security Role we created in Cloud9 command line for Apprunner to give it access to S3 and DynamoDB



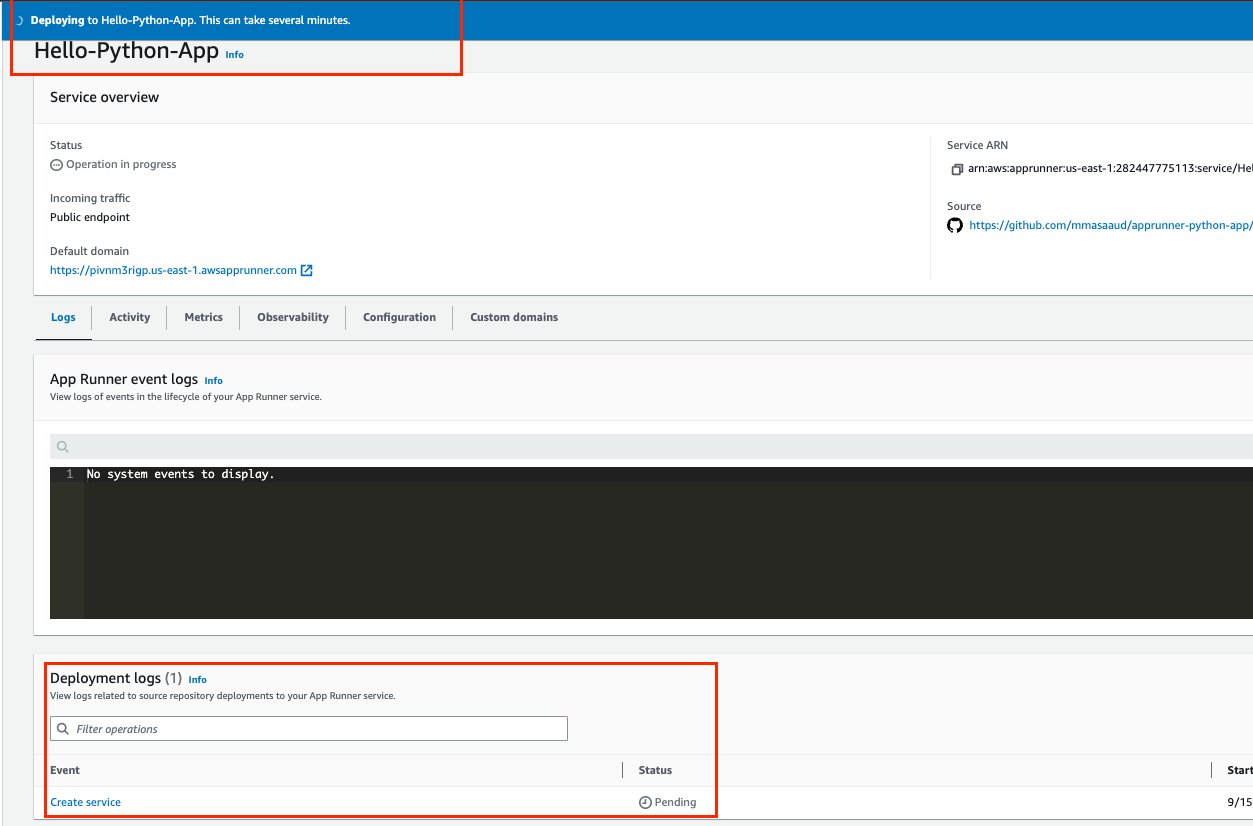
Finally, we will have to review the create the service if everything looks good as explained previously :

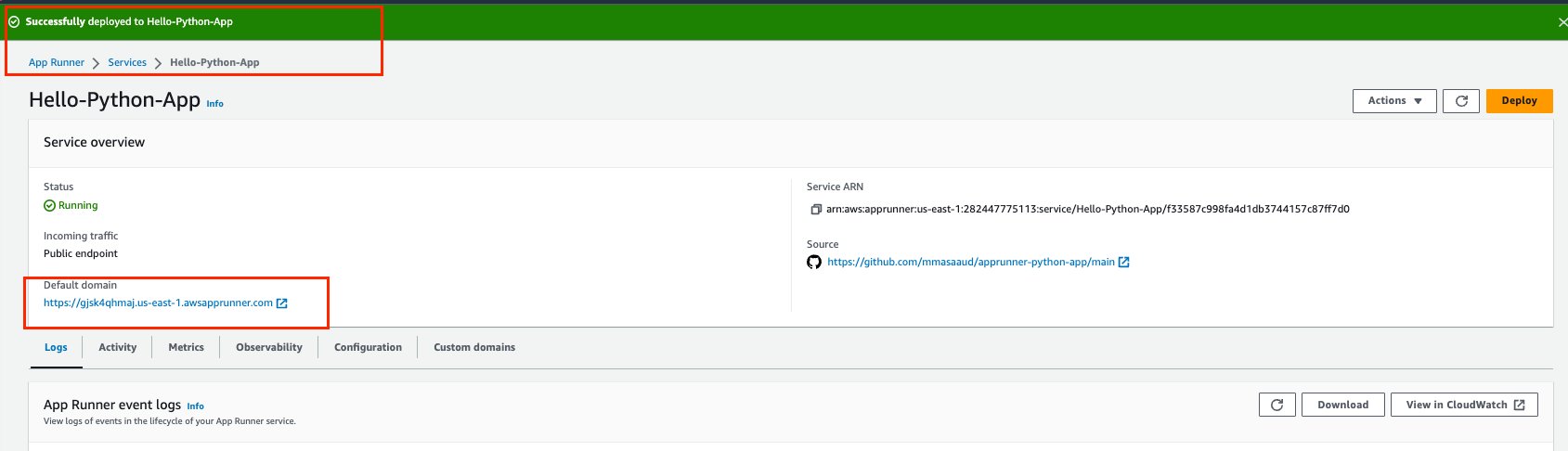


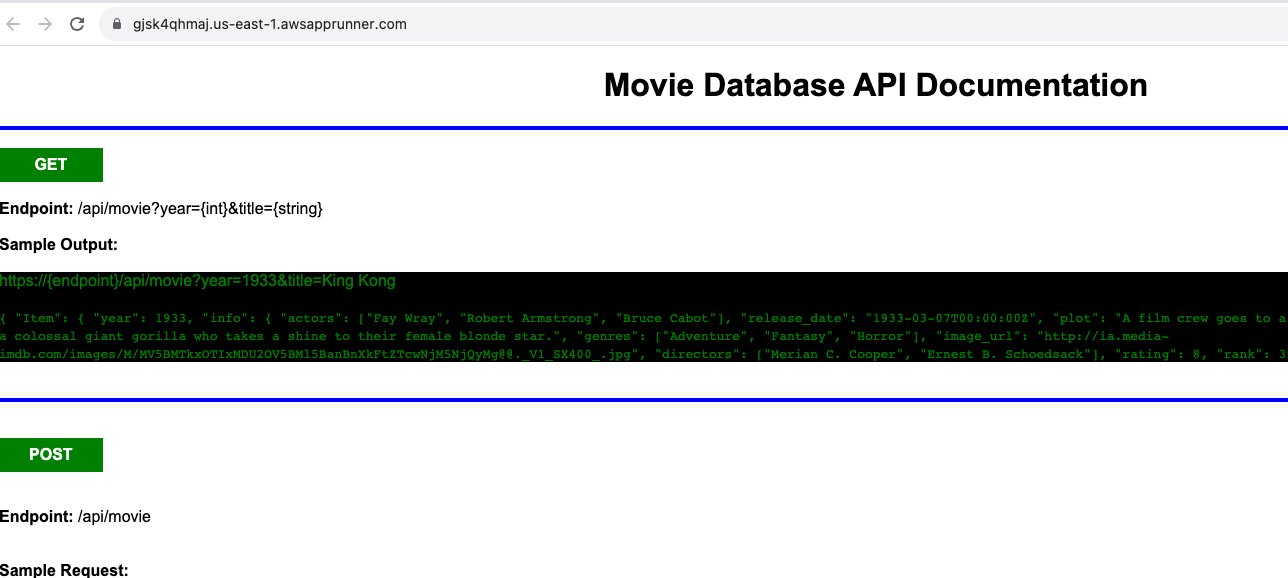


This will take between 5-7 minute to create the service, you can also check the deployment log while deploying is taking place :

After few minutes, service created Successfully and you can access if from the default domain link, which will be created automatically as a part of service creation :







Importnat Links :

1- AWS SDK for Python :

<https://aws.amazon.com/developer/language/python/>

2- Github link for Sample Python app :

<https://github.com/mmasaaud/apprunner-python-app>

3- DynamoDB document :

<https://us-east-2.console.aws.amazon.com/dynamodbv2/home?region=us-east-2#service>

4- S3 document :

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>