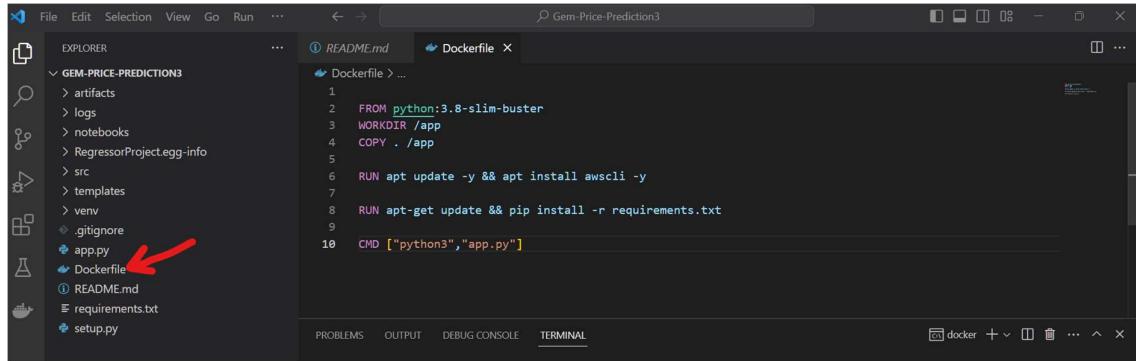


AWS Deployment using GitHub actions, ECR and EC2

1. Create a Dockerfile and define the docker container configurations:

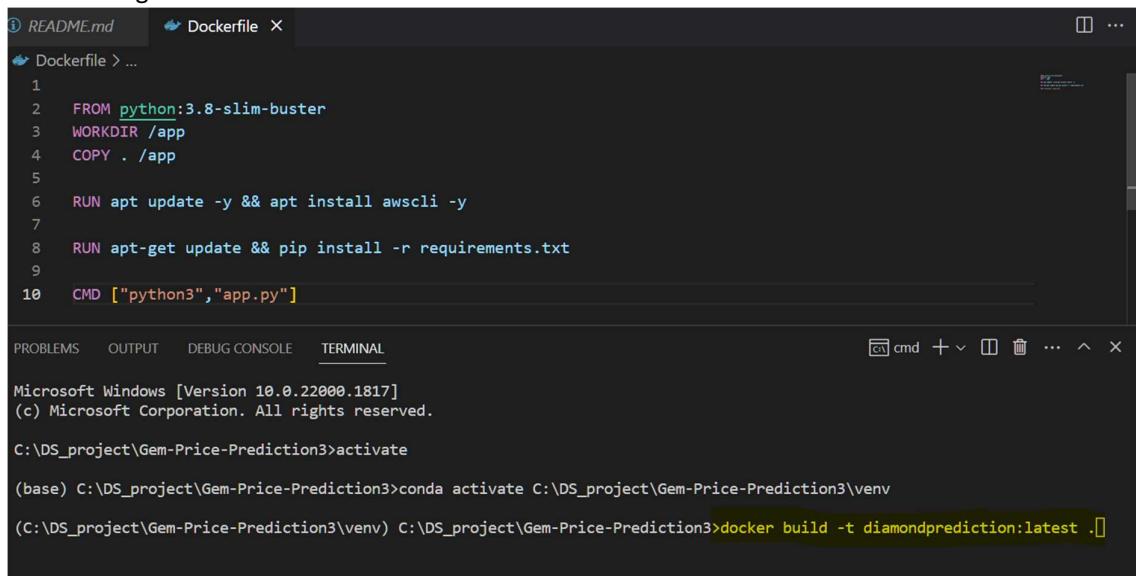


The screenshot shows the VS Code interface with the 'Gem-Price-Prediction3' project open. In the Explorer panel on the left, there is a file tree with several items: artifacts, logs, notebooks, RegressorProject.egg-info, src, templates, venv, .gitignore, app.py (which has a red arrow pointing to it), Dockerfile (which is selected and highlighted in blue), README.md, requirements.txt, and setup.py. In the center, the 'Dockerfile' tab is active, displaying the following Dockerfile content:

```
1 FROM python:3.8-slim-buster
2 WORKDIR /app
3 COPY . /app
4
5
6 RUN apt update -y && apt install awscli -y
7
8 RUN apt-get update && pip install -r requirements.txt
9
10 CMD ["python3","app.py"]
```

2. **Please note:** Step 2 and 3 are optional steps and can be skipped. This is because later we will be again building docker image and will push it to ECR repository and then this image will be pulled inside EC2 instance and will run as a containerized application or docker container. This entire job is mentioned inside our **workflow file** that is **.yaml** file.

Build Docker image using “`docker build -t diamondprediction:latest .`” command where “`diamondprediction`” is user defined image name and “`latest`” represents the tag or version of built image:



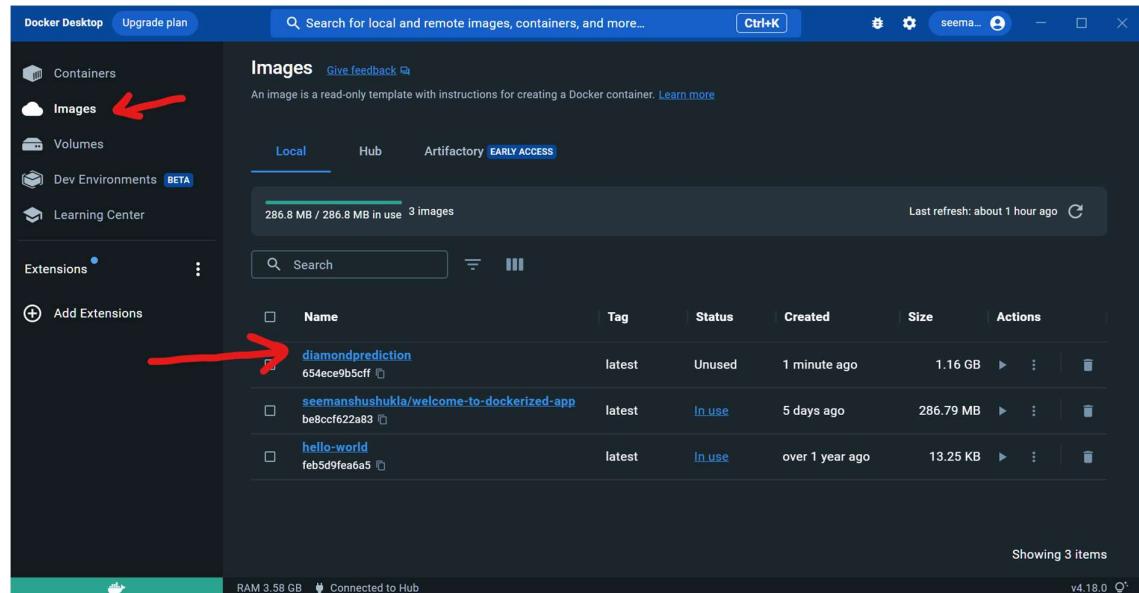
The screenshot shows the VS Code interface with the 'Gem-Price-Prediction3' project open. In the terminal tab at the bottom, the following command is being run and its output is shown:

```
Microsoft Windows [Version 10.0.22000.1817]
(c) Microsoft Corporation. All rights reserved.

C:\DS_project\Gem-Price-Prediction3>activate
(base) C:\DS_project\Gem-Price-Prediction3>conda activate C:\DS_project\Gem-Price-Prediction3\venv
(C:\DS_project\Gem-Price-Prediction3\venv) C:\DS_project\Gem-Price-Prediction3>docker build -t diamondprediction:latest .
```

Please note for working with docker one needs to install docker desktop application on their system. Please follow docker installation guide available on <https://www.docker.com/products/docker-desktop/>

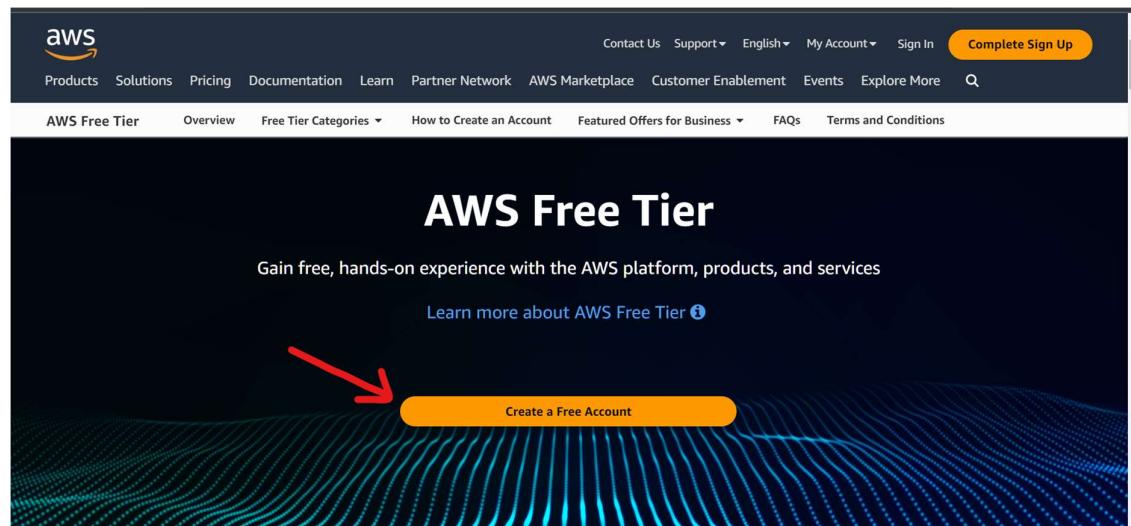
3. For verifying whether Docker image get created or not go to Docker desktop application and check for newly created docker image:



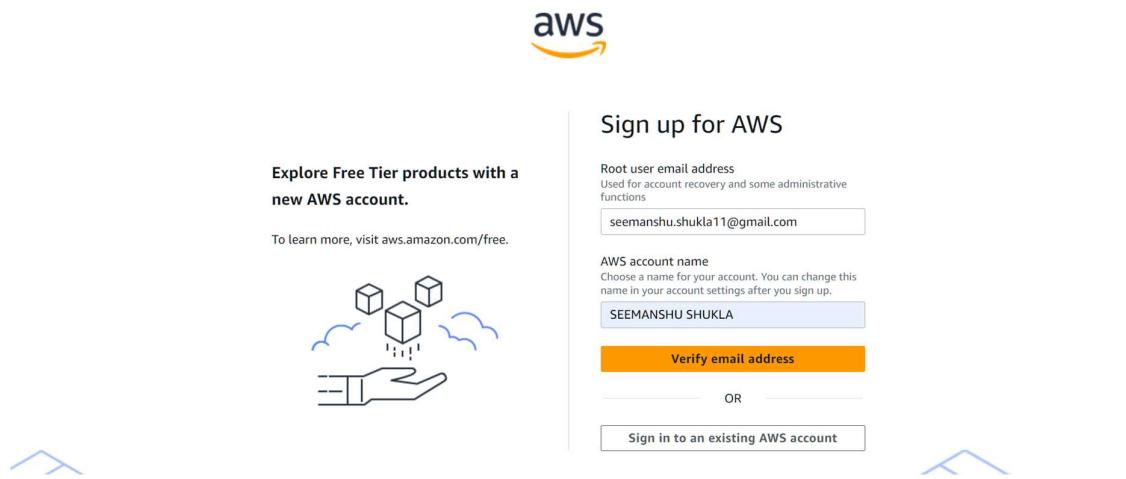
In above snip we can see that “*diamondprediction*” Docker image get created which is of size **1.16 GB**.

4. AWS account setup:

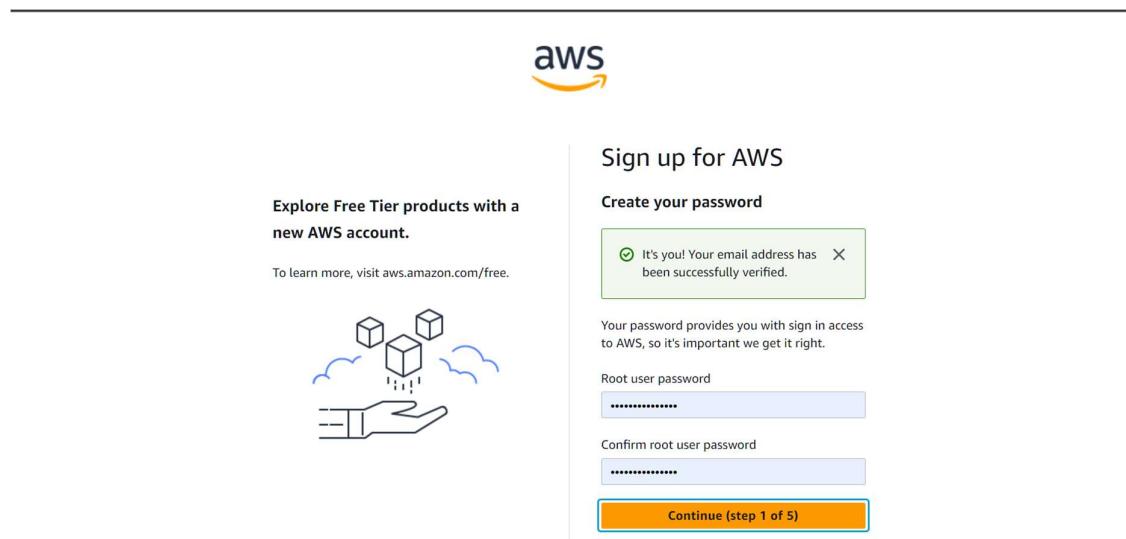
I. Go to AWS website and create a free account:



II. Enter details and click **Verify email address** and then verify the email using OTP sent to email address:



- III. Complete all the prompted 5 steps post which we will get redirected to Authentication page:



- IV. To verify our personal identity AWS will ask for debit or credit card. AWS will charge INR 2 for the same:

The screenshot shows a Paytm Payments Bank transaction page. On the left, under 'Merchant Details', it lists: Merchant Name (AMAZON), Date (Apr 27, 2023), Card Number (redacted), and Amount (₹2.00). On the right, under 'Authenticate Transaction', there is an 'OTP' section with a message: 'OTP successfully sent to your registered mobile number and email id.' Below it is an 'Enter OTP' input field containing '.....'. There are 'CANCEL' and 'SUBMIT' buttons at the bottom. A note at the bottom says: 'This screen will automatically time out after 5 minutes'.

- V. For learning purpose we will be selecting **Basic support – Free** plan . After this click **Complete sign up:**



Sign up for AWS

Select a support plan

Choose a support plan for your business or personal account. [Compare plans and pricing examples](#). You can change your plan anytime in the AWS Management Console.

<input checked="" type="radio"/> Basic support - Free <ul style="list-style-type: none"> Recommended for new users just getting started with AWS 24x7 self-service access to AWS resources For account and billing issues only Access to Personal Health Dashboard & Trusted Advisor 	<input type="radio"/> Developer support - From \$29/month <ul style="list-style-type: none"> Recommended for developers experimenting with AWS Email access to AWS Support during business hours 12 (business)-hour response times 	<input type="radio"/> Business support - From \$100/month <ul style="list-style-type: none"> Recommended for running production workloads on AWS 24x7 tech support via email, phone, and chat 1-hour response times Full set of Trusted Advisor best-practice recommendations 
--	---	---



Need Enterprise level support?

From \$15,000 a month you will receive 15-minute response times and concierge-style experience with an assigned Technical Account Manager. [Learn more](#)



Complete sign up

- VI. Please note that for the next 12 months, we will be provided with free access to all AWS services within the limits of the [Free Tier](#). Follow the free tier link to know more

Click on **GO to the AWS Management Console**



Congratulations

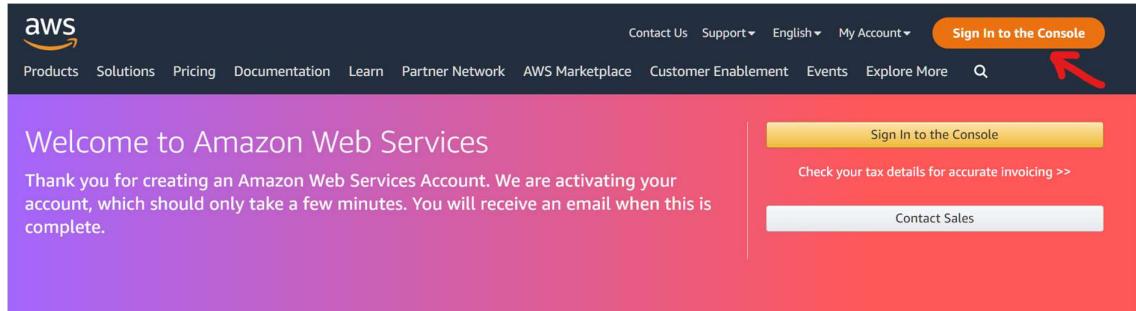
Thank you for signing up for AWS.

We are activating your account, which should only take a few minutes. You will receive an email when this is complete.

[Go to the AWS Management Console](#)



- VII. Click on **Sign in to the Console:**



The screenshot shows the AWS sign-in page. At the top, there's a dark navigation bar with the AWS logo, a search bar, and links for Contact Us, Support, English, My Account, and Sign In to the Console. A red arrow points to the "Sign In to the Console" button. Below the bar, a large purple banner says "Welcome to Amazon Web Services". It informs the user that their account is being activated and they'll receive an email when it's complete. To the right of the banner, there's a "Sign In to the Console" button, a "Check your tax details for accurate invoicing >>" link, and a "Contact Sales" button. At the bottom of the page, there's a "Personalize Your Experience" section with dropdown menus for "My role is:" and "I am interested in:", both currently set to "select role" and "select area".

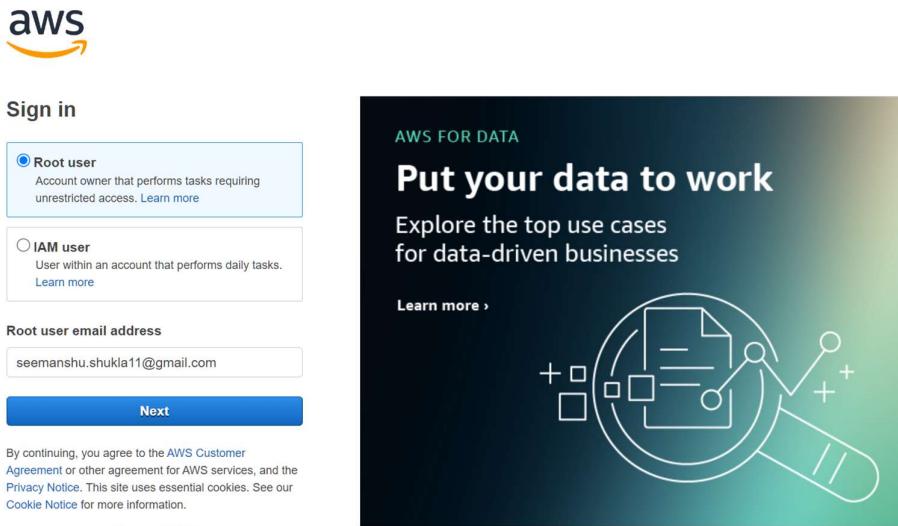
Personalize Your Experience

Fill in the blanks below to receive recommendations catered to your role and interests.

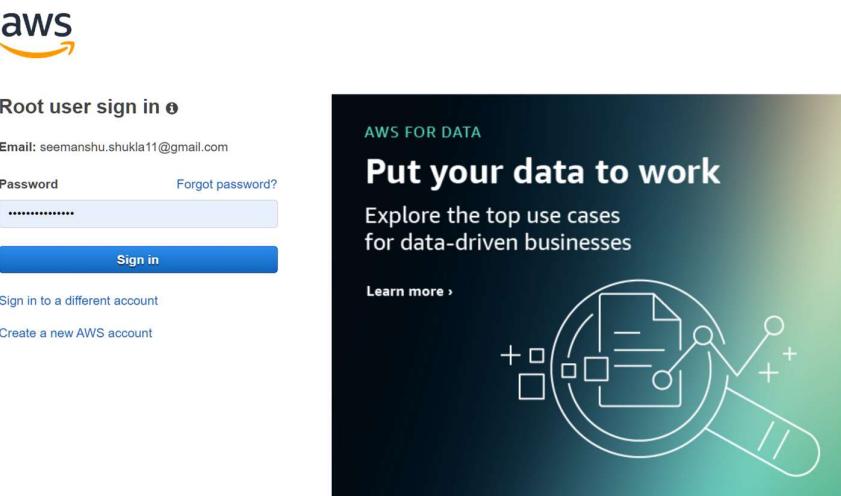
My role is: [select role](#)

I am interested in: [select area](#)

VIII. Select **Root user** and enter the email ID used while setting up the AWS account:



IX. Enter Password and click **Sign in** button:



5. IAM user Setup:

Once we are inside our AWS console we will start with creating an IAM user. IAM stands for **Identity Access Management**. Under this we as an IAM admin (since we are using a root account) will be creating a user who will be given some set of privileges using which this user can carry out the deployment job in AWS. This user will be set with some credentials that later will be used to configure it(user) while initiating the deployment to server.

I. Search for IAM service in the AWS console:

The screenshot shows the AWS console search results for 'IAM users'. The search bar at the top has 'IAM users' typed into it. Below the search bar, there is a sidebar with various service links. On the right, a list of services is shown, with 'IAM' being the first item. Under 'IAM', there is a 'Top features' section with tabs for 'Groups', 'Users' (which is highlighted with a red box), 'Roles', 'Policies', and 'Access Analyzer'. Other services listed include IAM Identity Center, Directory Service, and Cognito.

II. Add the user to whom we want to give access to ECR2 instance and ECR repository to carry out the deployment:

The screenshot shows the IAM 'Users' page. The left sidebar is titled 'Identity and Access Management (IAM)' and includes sections for 'Dashboard', 'Access management' (with 'User groups', 'Users' selected, and 'Roles', 'Policies', 'Identity providers', 'Account settings'), and 'Access reports'. The main right pane is titled 'Users (0) Info' and contains a message about IAM users. It features a search bar, a table header with columns for 'User name', 'Groups', 'Last activity', and 'MFA', and a note 'No resources to display'. In the top right corner of the main pane, there is a blue 'Add users' button with a red arrow pointing to it.

III. Enter the **User name** (this is user defined) and click **Next**:

The screenshot shows the 'Specify user details' step in the IAM 'Create user' wizard. The 'User name' field contains 'SeemanshuTest'. A note below it states: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)' and 'Provide user access to the AWS Management Console - optional'. Another note says: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more' with a link. At the bottom right, there are 'Cancel' and 'Next' buttons, with a red arrow pointing to the 'Next' button.

IV. Now we will be prompted with **Set permissions** page where we will defining the set of privileges to be given to the user for carrying out the deployment:

→Select **Attach policies directly**:

The screenshot shows the 'Set permissions' step in the IAM 'Create user' wizard. The 'Permissions options' section includes 'Add user to group', 'Copy permissions', and 'Attach policies directly' (which is selected). Below is a table titled 'Permissions policies (1080)' showing one policy: 'AccessAnalyzerServiceRolePol...' (AWS managed). There is a search bar and a pagination control.

→Follow the following steps to add necessary roles:

- 1)Search **amazonEC2Container** and hit enter
- 2)Select the check box that says full access
- 3)Use the cross(X) icon to remove this filter and search for **AmazonEC2**

Permissions policies (1/1080)
Choose one or more policies to attach to your new user.

Filter distributions by text, property or value 7 matches

Clear filters

Policy name	Type	Attached entities
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryFullAccess	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerRegistryPowerUser	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerRegistryReadOnly	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceAutoscaleRole	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceEventsRole	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceforEC2Role	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceRole	AWS man...	0

Permissions boundary - optional
Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. [Learn more](#)

→ Select the **amazonEC2FullAccess** and then remove the filter using cross(X) icon

Permissions policies (2/1080)
Choose one or more policies to attach to your new user.

Filter distributions by text, property or value 16 matches

Clear filters

Policy name	Type	Attached entities
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryFullAccess	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerRegistryPowerUser	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerRegistryReadOnly	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceAutoscaleRole	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceEventsRole	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceforEC2Role	AWS man...	0
<input type="checkbox"/> AmazonEC2ContainerServiceRole	AWS man...	0
<input checked="" type="checkbox"/> AmazonEC2FullAccess	AWS man...	0
<input type="checkbox"/> AmazonEC2readOnlyAccess	AWS man...	0

→ Scroll down the page and click **Next**:

The screenshot shows the 'Add user' wizard in the AWS IAM console. Step 4: Set permissions. A table lists various AWS managed policies. Below the table is a section titled 'Permissions boundary - optional' with a note about delegating permission management. At the bottom are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted in orange.

V. Review the details and click **Create user**. Here, we can clearly see the roles that were assigned to our user: SeemanshuTest:

The screenshot shows the 'Create user' wizard in the AWS IAM console. Step 5: Review and create. It displays 'User details' (User name: SeemanshuTest) and a 'Permissions summary' table showing two managed policies assigned: 'AmazonEC2ContainerRegistryFullAccess' and 'AmazonEC2FullAccess'. Below is a 'Tags - optional' section and a 'Create user' button at the bottom right.

VI. User is now created successfully. Open the highlighted newly created user:

The screenshot shows the AWS IAM service interface. A green success banner at the top right reads "User created successfully" with a checkmark icon. Below it, the main "Users" table lists one item: "SeemanshuTest". The "User name" column shows "SeemanshuTest", the "Groups" column shows "None", the "Last activity" column shows "Never", and the "MFA" and "Password a..." columns show "None". The left sidebar has sections for "Access management" (with "Users" selected), "Access reports", and "Metrics". The bottom navigation bar includes "CloudShell", "Feedback", "Language", "Privacy", "Terms", and "Cookie preferences".

VII. Go to Security Credentials:

The screenshot shows the detailed view for the user "SeemanshuTest". The "Summary" section displays ARN (arn:aws:iam:446896123821:user/SeemanshuTest), Console access (Disabled), and two Access keys. The "Permissions" tab is selected, showing "Permissions policies (2)" attached via Directly. Policies listed include "AmazonEC2ContainerRegistryFullAccess" and "AmazonEC2FullAccess", both of which are AWS managed policies. The left sidebar and bottom navigation bar are similar to the previous screenshot.

VIII. Now here, we will be creating an access key. This will be privileged key that will be basically used for carrying out our deployment by the newly created user.

Go to **Access keys** and click **Create access key** which will be having 3 sub parts:

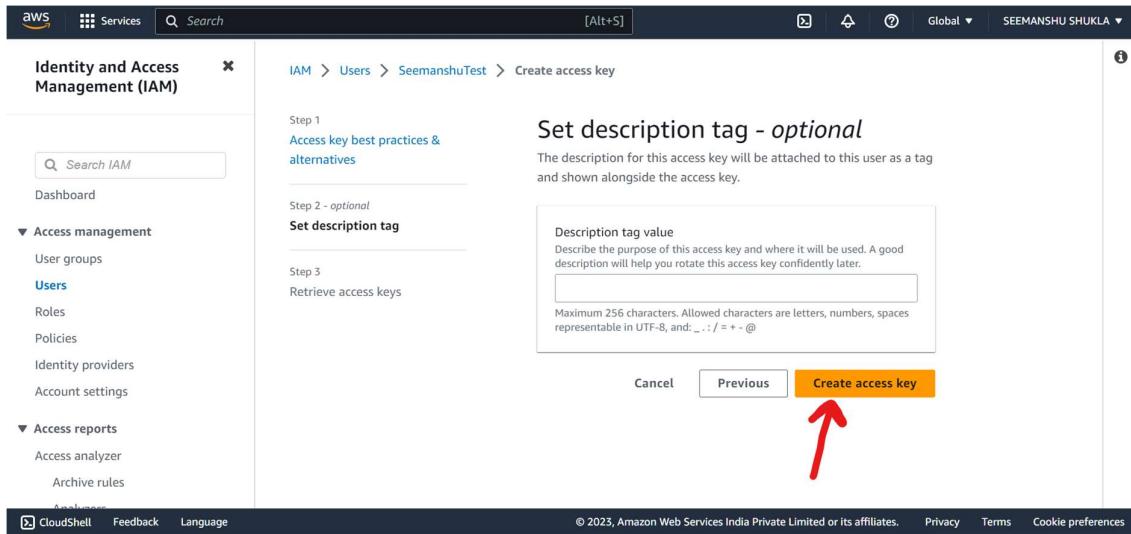
The screenshot shows the AWS IAM Multi-factor authentication (MFA) page. At the top, there's a table for MFA devices with columns for Device type, Identifier, and Created on. Below it, a message says 'No MFA devices. Assign an MFA device to improve the security of your AWS environment' with a 'Assign MFA device' button. The main content area is titled 'Access keys (0)' with a note about using access keys for programmatic calls. It includes a 'Create access key' button. A yellow box highlights this button. Below this, a message says 'As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials.' with a 'Create access key' button. At the bottom, there's a section for SSH public keys for AWS CodeCommit.

IX. Select CLI, radio box. Also, select the disclaimer check box at the bottom and click Next.

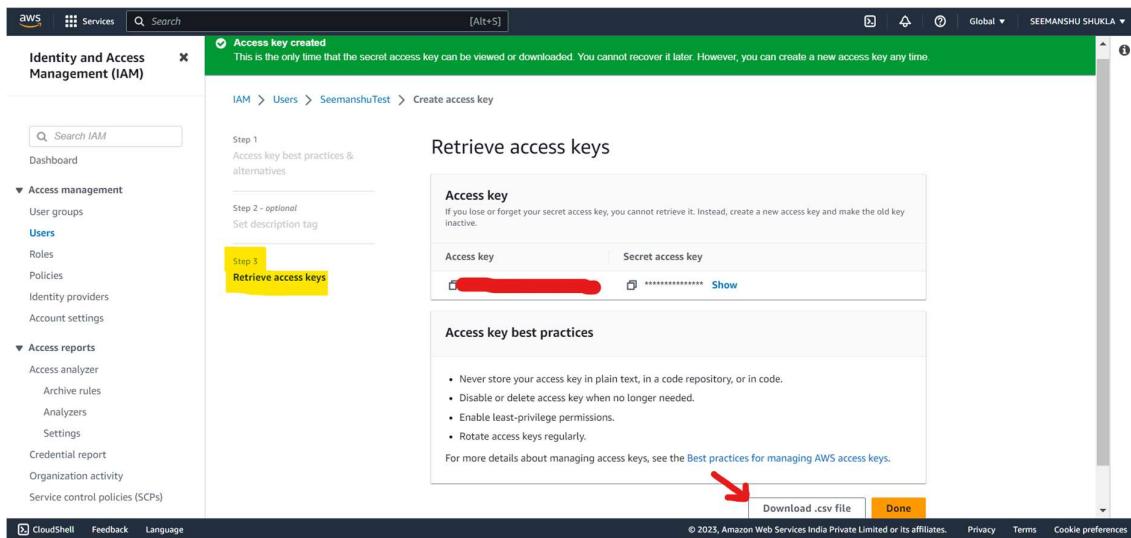
Recall that while creating Dockerfile we installed AWS CLI. It was installed for this purpose so that we can carry our deployment using access key in AWS CLI:

The screenshot shows the AWS IAM Access key creation wizard Step 2. It asks for a use case. Several options are listed with radio buttons: 'Command Line Interface (CLI)', 'Local code', 'Application running on an AWS compute service', 'Third-party service', 'Application running outside AWS', and 'Other'. The 'Command Line Interface (CLI)' option is selected. Below the options is a section titled 'Alternatives recommended' with two items: 'Use AWS CloudShell, a browser-based CLI, to run commands.' and 'Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center.' At the bottom, there's a checkbox 'I understand the above recommendation and want to proceed to create an access key.' followed by a 'Next' button. Red arrows point to the selected radio button for 'Command Line Interface (CLI)', the checked checkbox, and the 'Next' button.

X. We can skip description part and click Create access key:

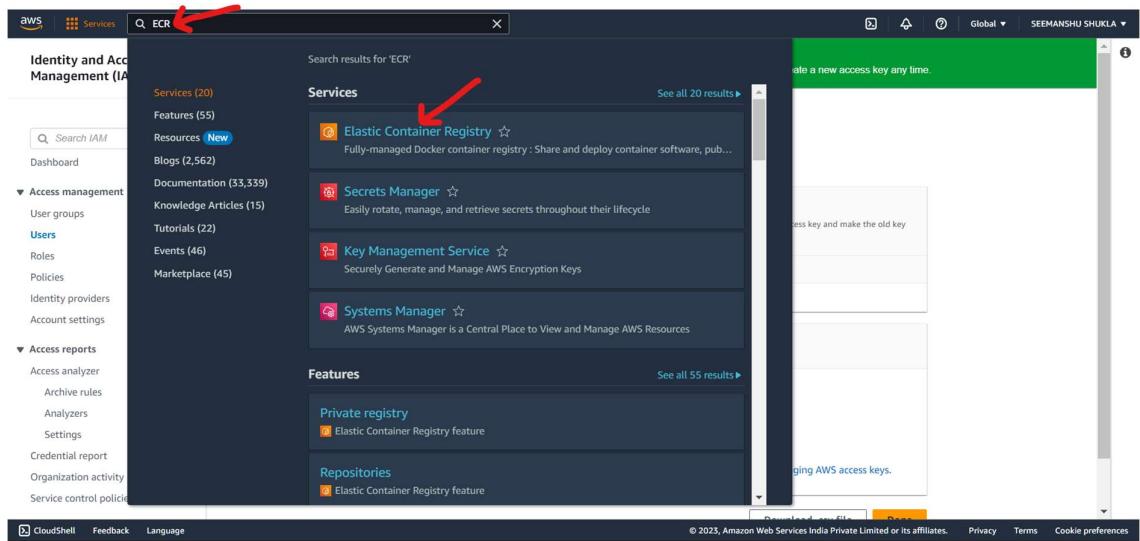


XI. Download access key as .csv file which later will be used to interact with EC2 instance and EC2 Repository via AWS CLI:

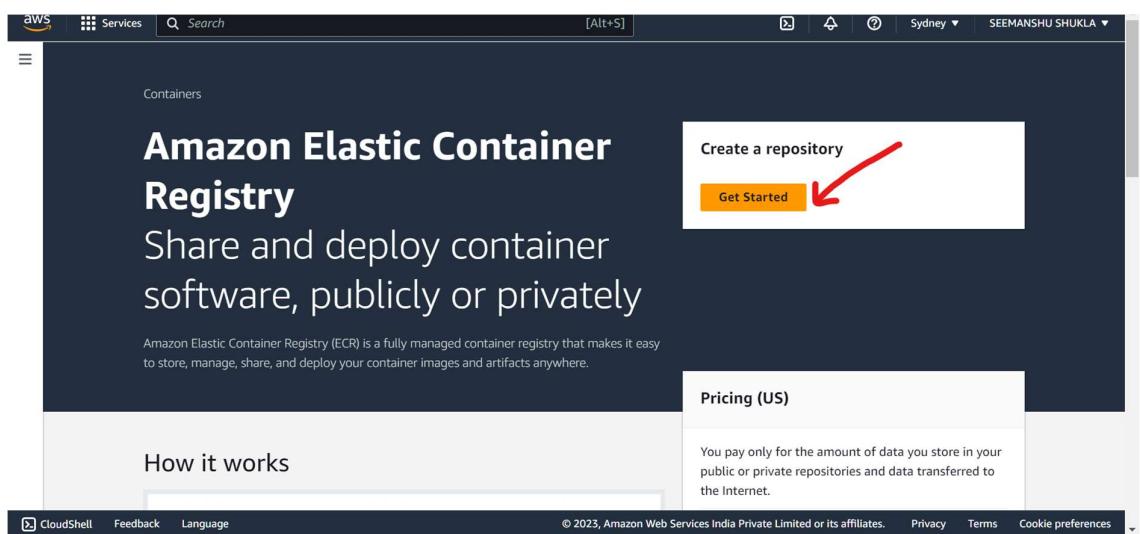


6. ECR (Elastic Container Repository) setup:

I. Search for ECR and open it. In ECR one can Share and deploy container software, publicly or privately:



II. Click Get Started:



III. Set Visibility as Private to make a private repository and set a repository name which is a user defined name:

The screenshot shows the 'Create repository' page in the AWS ECR console. In the 'General settings' section, the 'Visibility settings' dropdown is set to 'Private'. The 'Repository name' field contains '446896123821.dkr.ecr.ap-southeast-2.amazonaws.com/diamondpriceprediction'. The 'Tag immutability' section is collapsed. At the bottom, there are links for CloudShell, Feedback, Language, and a footer with copyright information and privacy terms.

IV. Scroll down and click **Create repository:**

The screenshot shows the continuation of the 'Create repository' page. It includes sections for 'Deprecation warning' (ScanOnPush configuration is deprecated), 'Scan on push' (disabled), and 'Encryption settings' (KMS encryption is disabled). A note states that KMS encryption settings cannot be changed after creation. At the bottom right, there is a red arrow pointing to the 'Create repository' button.

- V. Repository will get created. If we open this repository we will not find any image (docker image) since we are yet to build our docker image. The job for building docker image and pushing it to ECR is defined under .yaml workflow file. This file will be executed when we initiate our deployment.**

The screenshot shows the AWS ECR console with a green success banner at the top stating "Successfully created repository diamondpriceprediction". Below it, the "Private repositories (1)" section lists a single repository named "diamondpriceprediction". The table includes columns for Repository name, URI, Created at, Tag immutability, and Scan frequency. The repository details are: Name: diamondpriceprediction, URI: 446896123821.dkr.ecr.ap-southeast-2.amazonaws.com/diamondpriceprediction, Created at: 27 April 2023, 13:57:07 (UTC+05:5), Tag immutability: Disabled, Scan frequency: Manual.

The screenshot shows the AWS ECR console on the "diamondpriceprediction" repository details page. The left sidebar is expanded to show the "Images" section. The main content area shows the "Images (0)" section with a message "No images" and "No images to display".

VI. Go back and copy the URL for Private ECR repo.

Copied URL:

446896123821.dkr.ecr.ap-southeast-2.amazonaws.com/diamondpriceprediction

This url will be later used for setting secrete variable in GitHub in order to carry out deployment.

The screenshot shows the AWS ECR console. At the top, a green banner says "Successfully created repository diamondpriceprediction". Below it, the "Private" tab is selected under "Repositories". A table lists one private repository:

Repository name	URI	Created at	Tag immutability	Scan frequency
diamondpriceprediction	446896123821.dkr.ecr.ap-southeast-2.amazonaws.com/diamondpriceprediction	27 April 2023, 13:57:07 (UTC+05:5)	Disabled	Manual

A red arrow points to the URI column, and a tooltip says "Repository URI copied".

7. EC2 instance setup:

It is Virtual Server in the cloud (Linux based machine called Ubuntu will be used for current deployment).

I. Search for EC2 and open it.

The screenshot shows the AWS Services search results. The search bar at the top has "EC2" typed into it. The results are categorized under "Services". The "EC2" service is highlighted with a red arrow. Other services listed include EC2 Image Builder, Amazon Inspector, and AWS Firewall Manager.

II. Click Instances(**running**) option:

The screenshot shows the AWS EC2 Global View dashboard. On the left, a sidebar lists navigation options: New EC2 Experience (selected), EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations. A search bar at the top right contains the text "Search". The main content area is titled "Resources" and displays a summary of Amazon EC2 resources in the Asia Pacific (Sydney) Region. It includes tables for Instances (running), Dedicated Hosts, Instances, Load balancers, Security groups, and Volumes, all showing 0 items. It also lists Auto Scaling Groups, Elastic IPs, Key pairs, Placement groups, and Snapshots, all showing 0 items. A callout box provides information about easily sizing, configuring, and deploying Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. To the right, the "Account attributes" section lists supported platforms (VPC), default VPC (vpc-0f44160d3cdb09262), settings like EBS encryption and zones, and experimental features like EC2 Serial Console, Default credit specification, and Console experiments. At the bottom right, there's an "Explore AWS" section with a link to enable best price-performance.

You are using the following Amazon EC2 resources in the Asia Pacific (Sydney) Region:

Instances (running)	0	Auto Scaling Groups	0
Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	0
Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0
Volumes	0		

ⓘ Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. [Learn more](#)

Account attributes

Supported platforms

- VPC

Default VPC

vpc-0f44160d3cdb09262

Settings

EBS encryption

Zones

EC2 Serial Console

Default credit specification

Console experiments

Explore AWS

Enable Best Price-Performance with AWS Credit Options

III. Click **Launch instances**:

The screenshot shows the AWS EC2 Instances page. The top navigation bar includes the AWS logo, a 'Services' dropdown, a search bar, and account information for 'Sydney' and 'SEEMANSHU SHUKLA'. On the left, a sidebar under the 'Instances' heading lists various options: 'Instances' (selected), 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', and 'Capacity Reservations'. The main content area is titled 'Instances Info' and displays a table header with columns: Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. A search bar at the top says 'Find instance by attribute or tag (case-sensitive)' and a filter bar below it shows 'Instance state = running'. The table body is empty with the message 'No matching instances found'. At the top right of the main area, there are 'Actions' and 'Launch instances' buttons, with the 'Launch instances' button being highlighted by a red arrow.

IV. Scroll down and select the OS image – **Ubuntu** that is OS based on which server should be build. For learning purpose to make sure that we are not getting charged anything extra make sure to opt for **Free Tier** images. Select **Architecture** as **64 bit**.

The screenshot shows the AWS CloudFront console with the 'Origin' tab selected. The 'Origin Path' dropdown is open, displaying three options: '/index.html', '/path/*', and '/'. A red arrow points to the 'Path' dropdown menu.

V. Scroll down and select the **Instance type**. For learning purpose choosing **Free Tier** version of it

Instance type selects an instance type that meets your computing, memory, networking, or storage needs.

The screenshot shows the AWS Lambda console with the 'Code' tab selected. The 'Lambda@Edge' dropdown is open, displaying two options: 'Lambda@Edge' and 'Lambda'. A red arrow points to the 'Lambda@Edge' dropdown menu.

VI. Scroll down and click **Create new Key pair** which later can be used to securely connect with the instance.

The screenshot shows the AWS EC2 instance creation process. In the 'Key pair (login)' section, there is a dropdown menu labeled 'Select' and a button labeled 'Create new key pair'. A red arrow points to the 'Create new key pair' button.

VII. Enter the details as highlighted below and click **Create key pair**:

Create key pair

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn more](#)

Key pair name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair (Not supported for Windows instances)

Private key file format

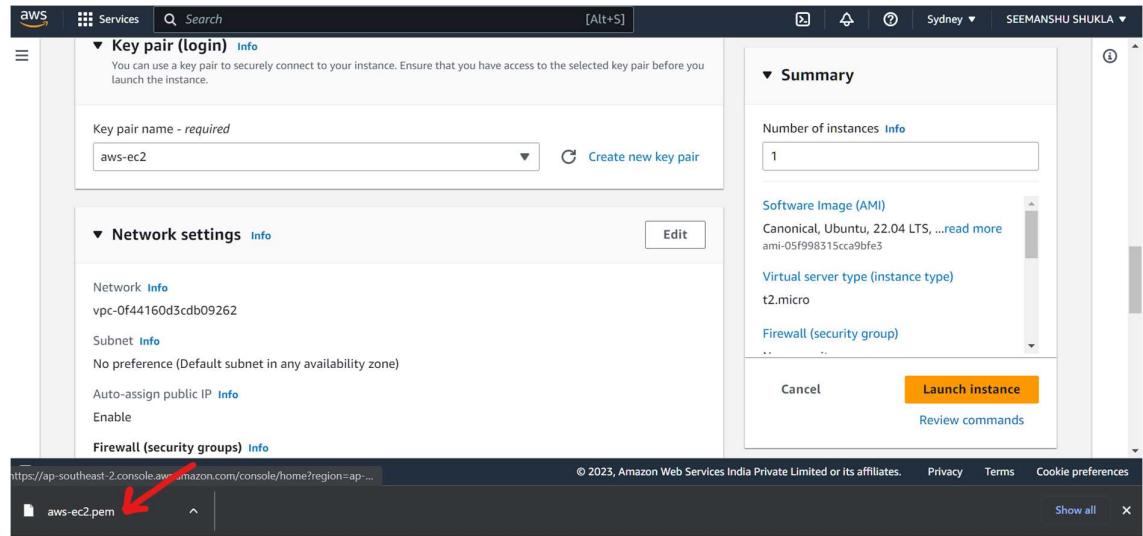
.pem

Create key pair

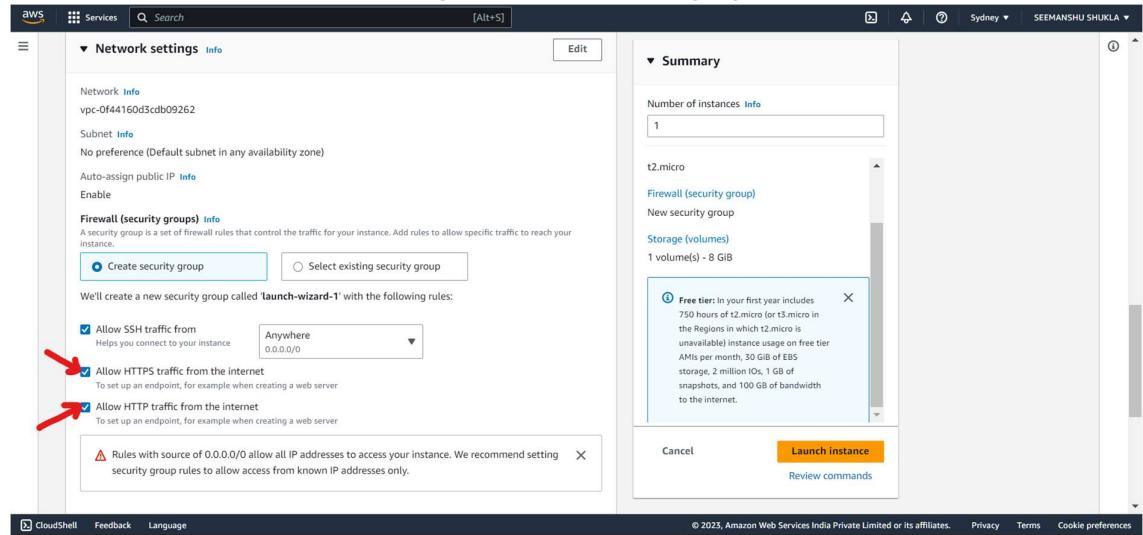
A large red arrow points from the 'Create key pair' button in the 'Create key pair' dialog to the 'Create key pair' button in the main EC2 instance creation wizard.

VIII. A file will be downloaded once key is successfully created. This downloaded file then need to be attached as to some 3rd party console using which we can establish a secure connection with our EC2 instance.

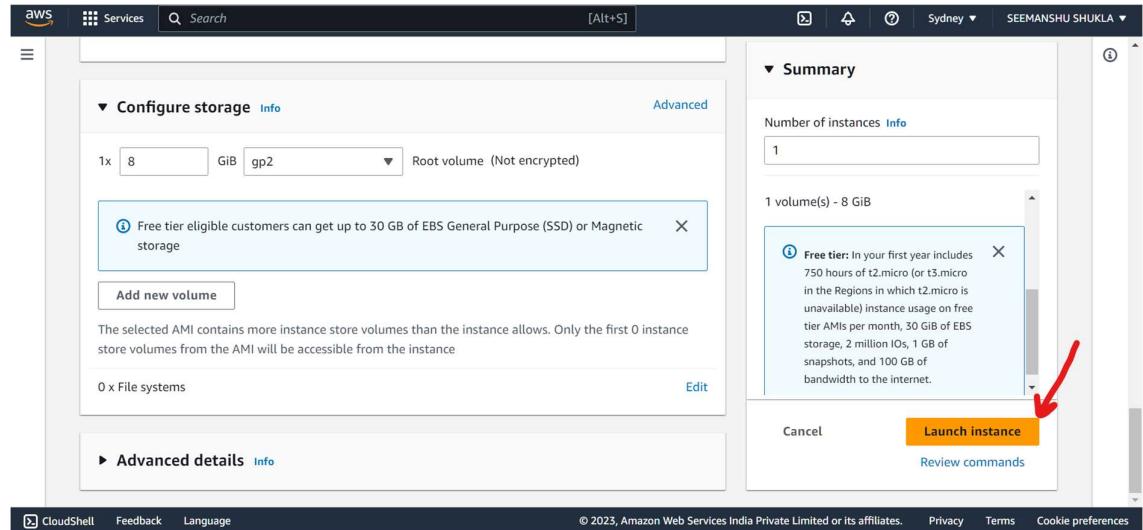
For E.g. MobaXterm is the 3rd part console based application that can be used to securely connect with our EC2 instance. For this while configuring instance over MobaXterm we will be asked to attach the secure key file, where we will be attaching the current .pem file.



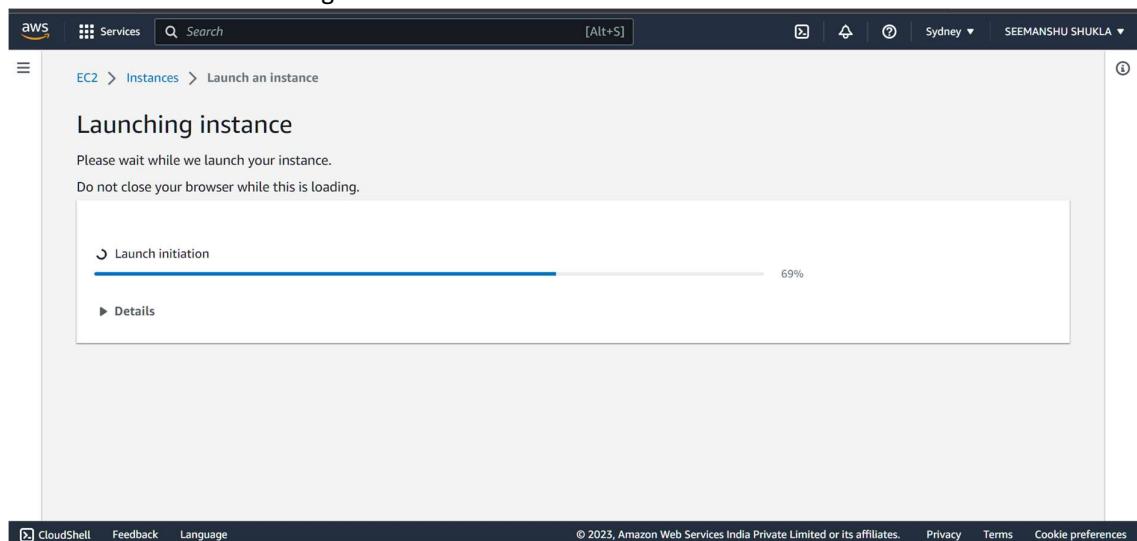
IX. Scroll down to Network settings and check all the highlighted check boxes:



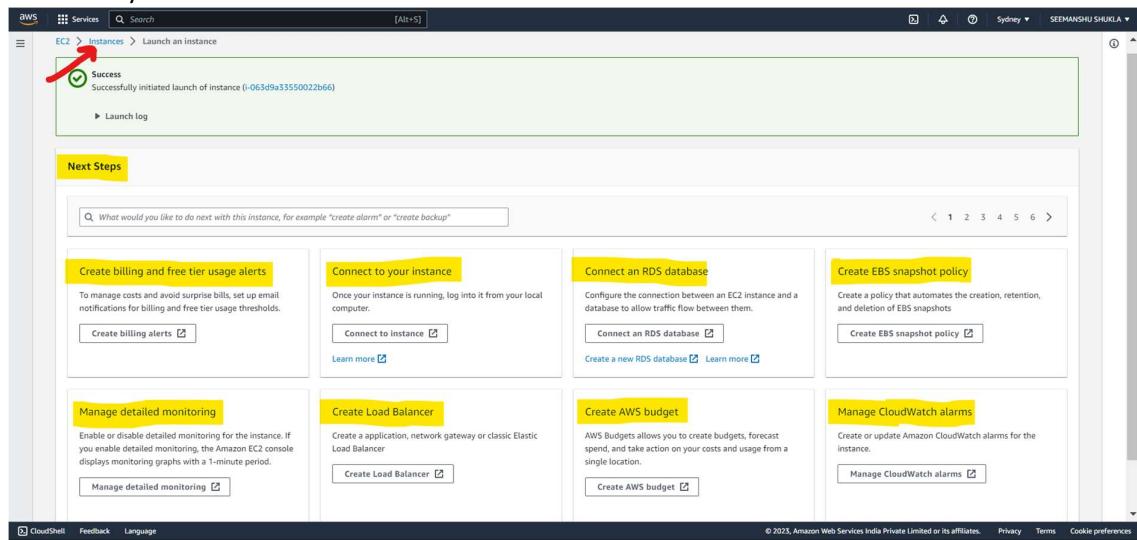
X. After this click Launch instance:



- XI. Launch initiated. Make sure that we are not closing this tab or browser while instance in launching.



- XII. Now one ideally needs to perform all the mentioned Next steps. Before that lets check our newly created instance by clicking on **instance** option. Also, if we scroll down a bit there is an option with the name **View all instances** using which as well anyone can view all the instances



- XIII. Now, we can see our instance with all the details in tabular form.

Once **Instance state** and **Status check** is turned into **Green** then it means that our EC2 instance has been set successfully and is now up and running.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, and Instances. Under Instances, there are sub-options for Instances, Instance Types, Launch Templates, and Spot Requests. The main content area displays a table of instances. One instance is listed: Name: diamondprice..., Instance ID: i-063d9a33550022b66, Instance state: Running, Instance type: t2.micro, Status check: 2/2 checks passed, and Alarm status: No alarms. A yellow box highlights the Instance ID 'i-063d9a33550022b66'.

XIV. Click on the **Instance ID** as highlighted below:

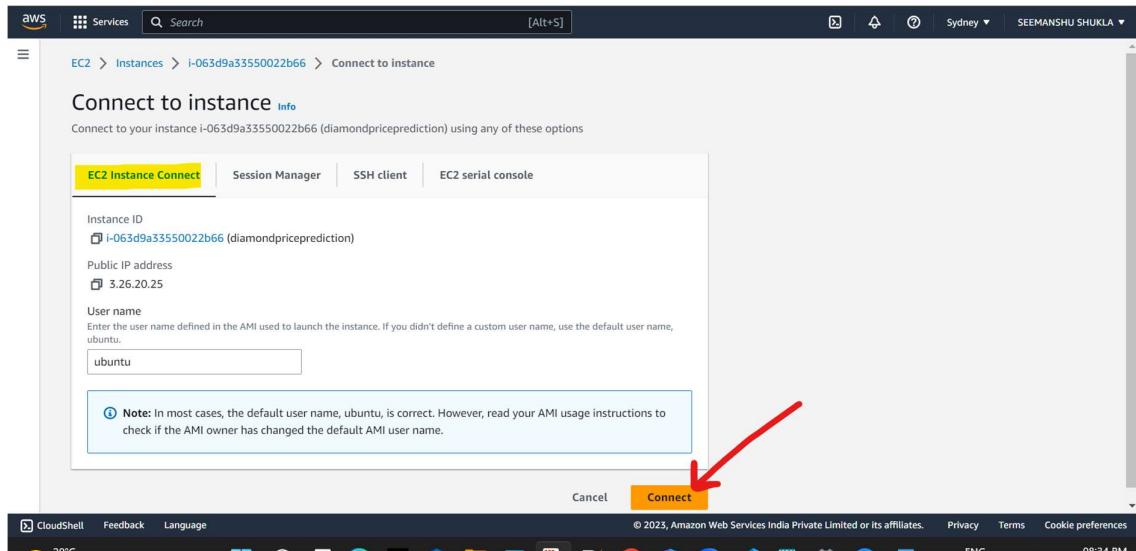
This screenshot is identical to the one above, showing the AWS EC2 Instances page. The Instance ID 'i-063d9a33550022b66' is highlighted with a yellow box to indicate where the user should click.

XV. Use **Connect** option to connect with the instance (server):

The screenshot shows the Instance summary for the selected instance. The top navigation bar includes the AWS logo, Services, Search, and account information. Below it, the EC2 Instances page shows the instance details. The 'Connect' button in the top right corner of the instance summary box is highlighted with a yellow box.

Instance summary for i-063d9a33550022b66 (diamondpriceprediction) Info		
Updated less than a minute ago		
View log file Connect Instance state Actions		
Instance ID i-063d9a33550022b66 (diamondpriceprediction)	Public IPv4 address 3.26.20.25 open address	Private IPv4 addresses 172.31.25.149
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-26-20-25.ap-southeast-2.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-25-149.ap-southeast-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-25-149.ap-southeast-2.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations.
Auto-assigned IP address 3.26.20.25 [Public IP]	VPC ID vpc-0f44160d3cdb09262	

XVI. Under **EC2 instance Connect** click **Connect** option:



- XVII. Now our instance/server will be launched which will be a linux based machine. We can access this machine using opened command prompt. Here we will be doing all the necessary configurations and installations. This command prompt is of EC2 instance that we have just created:

```

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-25-149:~$ i-063d9a33550022b66 (diamondpriceprediction)
Public IPs: 3.26.20.25 Private IPs: 172.31.25.149

```

8. Docker Setup In EC2 commands to be Executed:

Run all the below commands in EC2 instance via cmd CLI. Also, please make sure to run below command in the sequential order.

#optional

sudo apt-get update -y

sudo apt-get upgrade

#required

curl -fsSL <https://get.docker.com> -o get-docker.sh

“for installing docker on EC2”

```
sudo sh get-docker.sh
```

```
sudo usermod -aG docker ubuntu
```

“changing to user mode so that we don’t need to write sudo cmd again and again”

```
newgrp docker
```

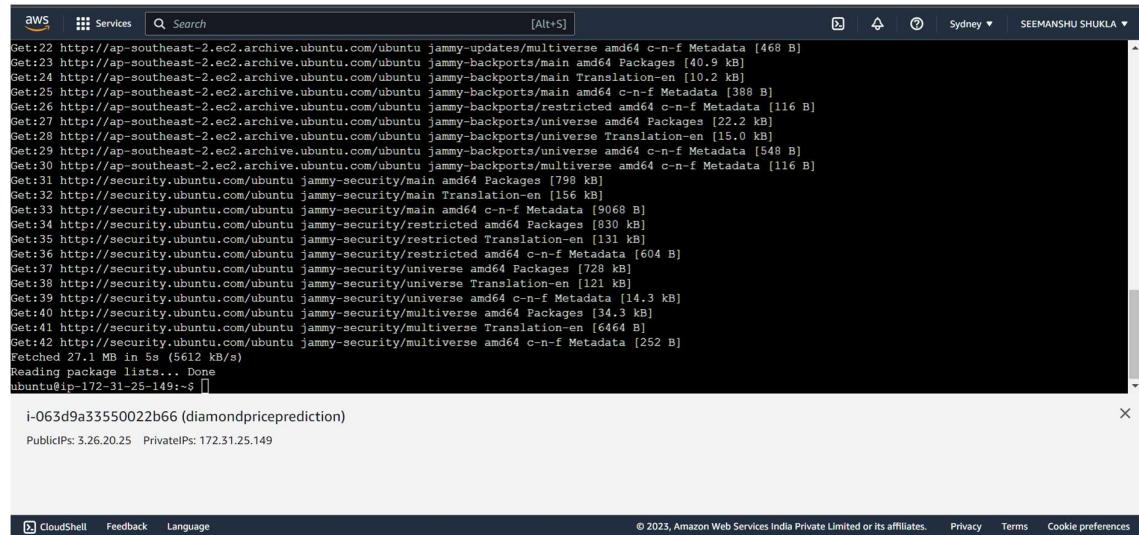
“creating a new group with name docker”

☞ Docker Setup In EC2 commands to be Executed

```
#optional  
  
sudo apt-get update -y  
  
sudo apt-get upgrade  
  
#required  
  
curl -fsSL https://get.docker.com -o get-docker.sh  
  
sudo sh get-docker.sh  
  
sudo usermod -aG docker ubuntu  
  
newgrp docker
```

Now, we will execute above commands in sequential manner in our EC2 instance command prompt.

→First cmd:



```
AWS Services Search [Alt+S] Sydney SEEMANSHU SHUKLA  
Get:22 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [468 B]  
Get:23 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [40.9 kB]  
Get:24 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [10.2 kB]  
Get:25 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]  
Get:26 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]  
Get:27 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [22.2 kB]  
Get:28 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [15.0 kB]  
Get:29 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [549 B]  
Get:30 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]  
Get:31 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [798 kB]  
Get:32 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [156 kB]  
Get:33 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [9068 B]  
Get:34 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [830 kB]  
Get:35 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [131 kB]  
Get:36 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [604 B]  
Get:37 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [728 kB]  
Get:38 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [121 kB]  
Get:39 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [14.3 kB]  
Get:40 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [34.3 kB]  
Get:41 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [6464 B]  
Get:42 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [252 B]  
Fetched 27.1 MB in 5s (5612 kB/s)  
Reading package lists... Done  
ubuntu@ip-172-31-25-149:~$  
  
i-063d9a33550022b66 (diamondpriceprediction)  
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149
```

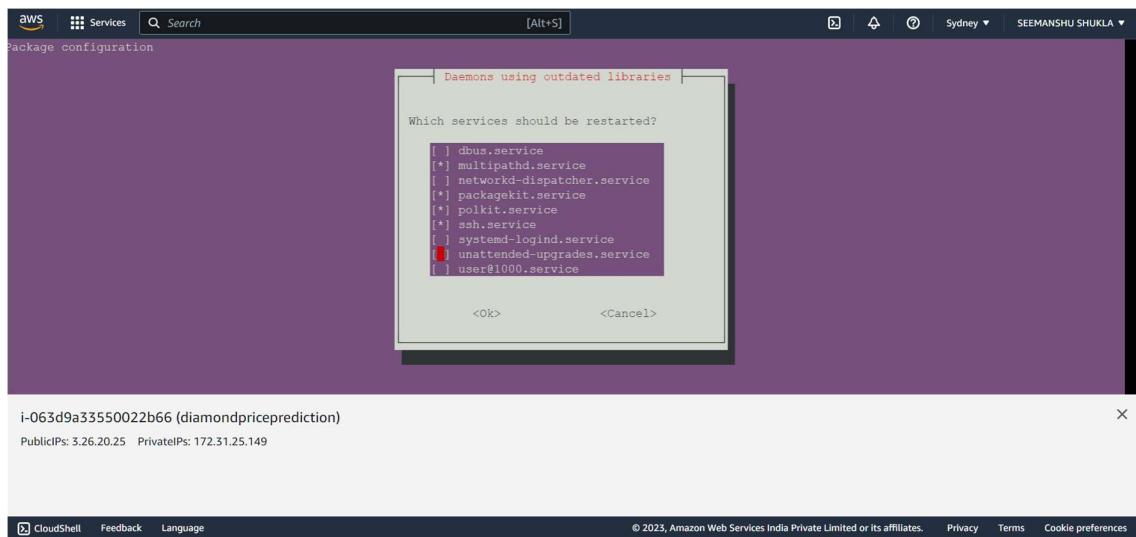
→Second cmd:

Give “Y” permission when asked Do you want to continue?

```
Get:36 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [604 B]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [728 kB]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [121 kB]
Get:39 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [14.3 kB]
Get:40 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [34.3 kB]
Get:41 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [6464 B]
Get:42 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [252 B]
Fetched 27.1 MiB in 5s (5612 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-25-149:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  linux-aws linux-headers-aws linux-image-aws
The following packages will be upgraded:
  aptor apt-utils bind9-dnsutils bind9-host bind9-libs cloud-init distro-info-data libapt-pkg6.0 libldap2-2.5-0 libldap-common libnss-systemd
  libpam-systemd libssl13 libsystemd libudev libxml2 openssl python3-apport python3-problem-report python3-tz sudo systemd systemd-sysv tzdata
  ubuntu-advantage-tools udev vim vim-common vim-runtime vim-tiny xxd
32 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 26.4 MB of archives.
After this operation, 358 kB of additional disk space will be used.
Do you want to continue? [Y/n] [ ]
```

i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

If prompted to this page then simply close the instance by closing the browser tab where instance is opened and open the instance again as we did earlier and run Third cmd.



→Third cmd:

```
aws Services Search [Alt+S] Sydney SEEMANSHU SHUKLA

System load: 0.02880859375 Processes: 101
Usage of /: 23.3% of 7.57GB Users logged in: 0
Memory usage: 24% IPv4 address for eth0: 172.31.25.149
Swap usage: 0%

* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

38 updates can be applied immediately.
20 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Thu Apr 27 15:06:35 2023 from 13.239.158.5
ubuntu@ip-172-31-25-149:~$ curl -fsSL https://get.docker.com -o get-docker.sh
ubuntu@ip-172-31-25-149:~$ [REDACTED]
```

i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

→Fourth cmd:

```
aws Services Search [Alt+S] Sydney SEEMANSHU SHUKLA

docker-init:
Version: 0.19.0
GitCommit: de40ad0

=====
To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

dockerd-rootless-setuptool.sh install
Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/
WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/
```

[REDACTED]
ubuntu@ip-172-31-25-149:~\$ [REDACTED]

i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

→Fifth cmd:

aws Services Q Search [Alt+S] Sydney SEEMANSHU SHUKLA

Version: 0.19.0
GitCommit: de40ad0

```
-----
```

To run Docker as a non-privileged user, consider setting up the Docker daemon in rootless mode for your user:

```
dockerd-rootless-setuptool.sh install
```

Visit <https://docs.docker.com/go/rootless/> to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root users access, refer to <https://docs.docker.com/go/daemon-access/>

```
WARNING: Access to the remote API on a privileged Docker daemon is equivalent to root access on the host. Refer to the 'Docker daemon attack surface' documentation for details: https://docs.docker.com/go/attack-surface/
```

```
-----
```

ubuntu@ip-172-31-25-149:~\$ sudo usermod -aG docker ubuntu

ubuntu@ip-172-31-25-149:~\$ []

i-063d9a33550022b66 (diamondpriceprediction)

Public IPs: 5.26.20.25 Private IPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

→Sixth cmd:

aws Services Q Search [Alt+S] Sydney SEEMANSHU SHUKLA

GitCommit: de40ad0

```
-----
```

To run Docker as a non-privileged user, consider setting up the Docker daemon in rootless mode for your user:

```
dockerd-rootless-setuptool.sh install
```

Visit <https://docs.docker.com/go/rootless/> to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root users access, refer to <https://docs.docker.com/go/daemon-access/>

```
WARNING: Access to the remote API on a privileged Docker daemon is equivalent to root access on the host. Refer to the 'Docker daemon attack surface' documentation for details: https://docs.docker.com/go/attack-surface/
```

```
-----
```

ubuntu@ip-172-31-25-149:~\$ sudo usermod -aG docker ubuntu

ubuntu@ip-172-31-25-149:~\$ newgrp docker

ubuntu@ip-172-31-25-149:~\$ []

i-063d9a33550022b66 (diamondpriceprediction)

Public IPs: 5.26.20.25 Private IPs: 172.31.25.149

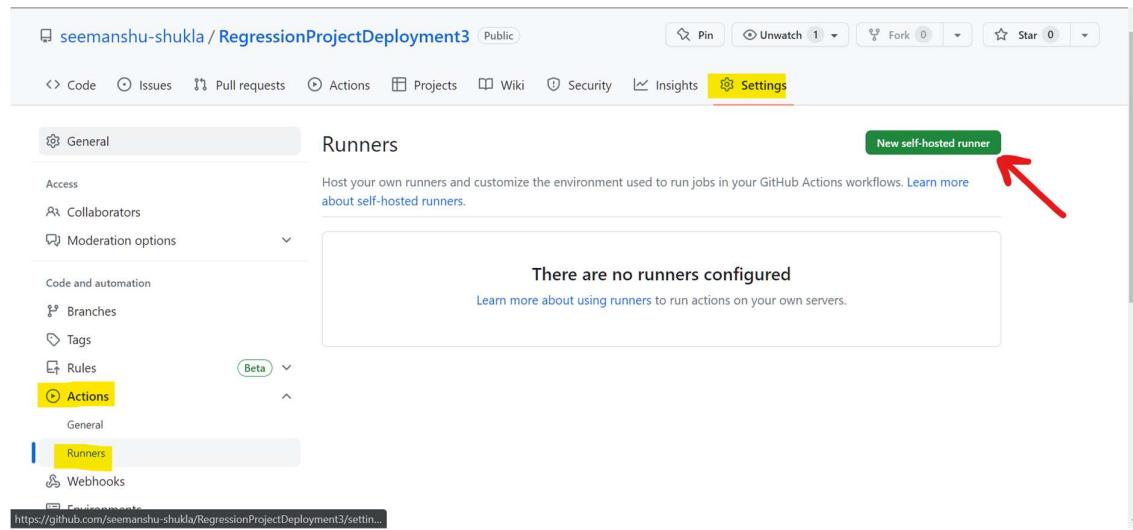
CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

9. Self-hosted runner setup:

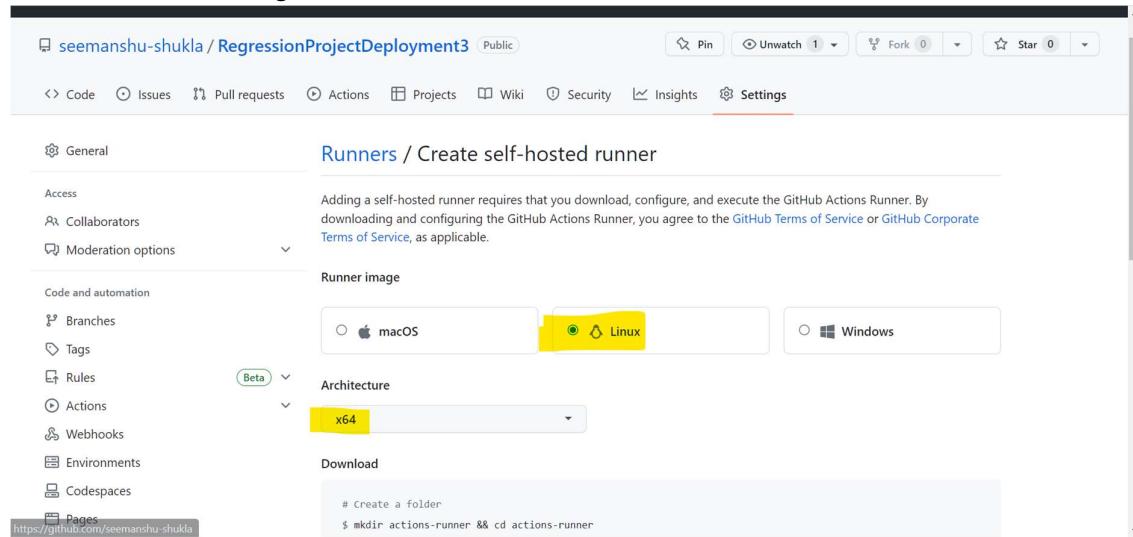
So when we push any new changes to our GitHub repository then our deployment should automatically listen it and push those changes to the existing deployment. To make this happen we use self-hosted runners.

- #### I. Let's create a runner in GitHub: Traverse as stated below

Open GitHub repo where application's codebase is pushed > Settings > Actions > Runners > and then click **New self-hosted runner**.



- II. Since, our server(EC2 instance) is Linux (64 bit) based we will selecting Linux (64 bit) as a runner image:



- III. Scroll down and code template thus generated will be required to be executed in the server instance (EC2) post which our self-hosted runner will start listening to all the changes that comes into GitHub and will push it to our server(EC2 instance where app will be deployed).

The screenshot shows the GitHub Actions setup interface. On the left, there's a sidebar with options like Environments, Codespaces, Pages, Security, Code security and analysis, Deploy keys, Secrets and variables (which is expanded), Integrations, GitHub Apps, and Email notifications. The main area has three sections: 'Download' (with shell commands for creating a folder, downloading the runner package, validating the hash, and extracting it), 'Configure' (with shell commands for creating the runner and starting the configuration experience), and 'Using your self-hosted runner' (with a YAML example for a workflow file).

Run above commands in sequential order inside server instance via EC2 cmd prompt.

IV. For Download:

The screenshot shows an AWS CloudShell terminal window. The user has run the following commands sequentially:

```

ubuntu@ip-172-31-25-149:~$ mkdir actions-runner && cd actions-runner
ubuntu@ip-172-31-25-149:~/actions-runner$ curl -o actions-runner-linux-x64-2.303.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.303.0/actions-runner-linux-x64-2.303.0.tar.gz
# Total    % Received % Xferd  Average Speed   Time   Time  Current
#          %          %          %       Dload  Upload  Total  Spent  Left  Speed
0      0  0     0  0      0  0:--:-- --:--:--:--:--:-- 0
100 137M 100 137M 0  49.0M  0 0:00:02 0:00:02 66.3M
ubuntu@ip-172-31-25-149:~/actions-runner$ echo "e4a9fb7269c1a156eb5d5369232d0cd62e06bec2fd2b321600e85ac914a9cc73" actions-runner-linux-x64-2.303.0.tar.gz | sha256sum -a 256 -c
actions-runner-linux-x64-2.303.0.tar.gz: OK
ubuntu@ip-172-31-25-149:~/actions-runner$ tar xzf ./actions-runner-linux-x64-2.303.0.tar.gz
ubuntu@ip-172-31-25-149:~/actions-runner$ []

```

Red arrows point to the first four commands in the sequence, highlighting the download and verification steps.

V. For Configure:

→ After running first configure cmd was getting this error:

A screenshot of a terminal window titled "aws Services". The command run was "tar xf ./actions-runner-linu...tar.gz". The output shows a "Self-hosted runner registration" section and an "# Authentication" section. A red arrow points to the error message at the bottom: "Http response code: NotFound from 'POST https://api.github.com/actions/runners-registration' (Request Id: ABEE:3B16:5427C0:590E0F:644AC2C4) {"message": "Not Found", "documentation_url": "https://docs.github.com/rest"} Response status code does not indicate success: 404 (Not Found)." The user "SEEMANSHU SHUKLA" is visible in the top right.

→ As a resolution have just reloaded the GitHub runner page from where we generated runner code template for Linux based machine.

→ After successfully running the **first cmd** for **Configure**:

A screenshot of a terminal window titled "aws Services". The command run was "tar xf ./config.sh --url https://github.com/seemanshu-shukla/RegressionProjectDeployment3 --token APTXB65E4HJPCLLKMJIYWT...". A red arrow points to the URL parameter in the command. The output shows a "Self-hosted runner registration" section and an "# Authentication" section. A red arrow points to the "Connected to GitHub" message. Below it, a prompt asks "Enter the name of the runner group to add this runner to: [press Enter for Default]". The user "SEEMANSHU SHUKLA" is visible in the top right.

In above hit enter to pass default runner group name.

→ “self-hosted” is the runner name that we usually given to a runner and then hit Enter:

AWS CloudShell terminal window showing the process of registering a self-hosted runner. The terminal output includes:

```
ubuntu@ip-172-31-25-149:~/actions-runner$ ./config.sh --url https://github.com/seemanshu-shukla/RegressionProjectDeployment3 --token APTXB643SH4M4KU6DQHJ33TEJLIVU
Self-hosted runner registration

# Authentication
Connected to GitHub
# Runner Registration
Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for ip-172-31-25-149] self-hosted
```

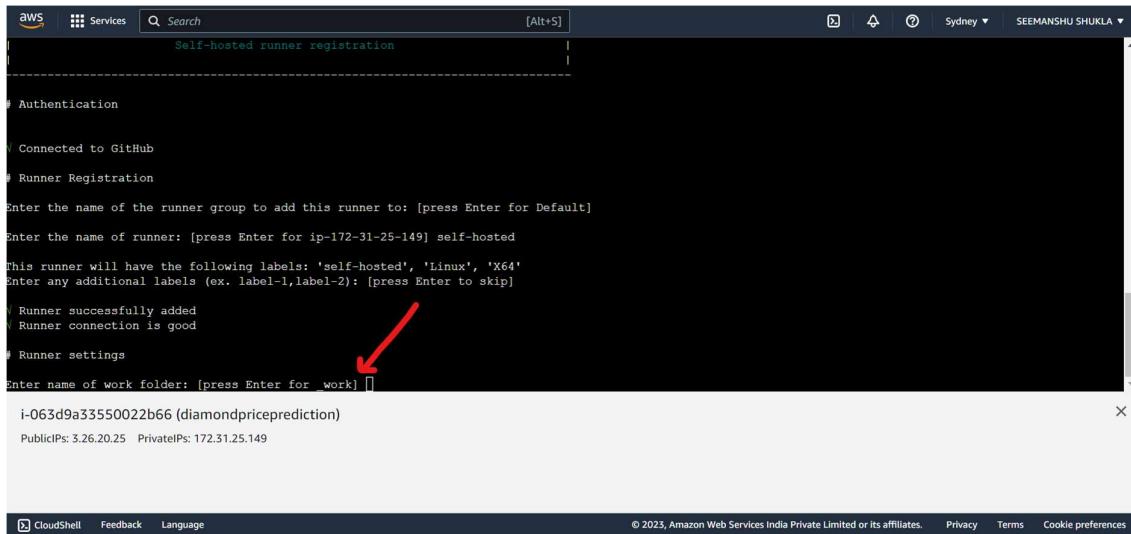
→ Configuration/labels of runner getting displayed. After this hit **Enter** to skip the step of adding any additional labels:

AWS CloudShell terminal window showing the configuration of a runner. The terminal output includes:

```
Self-hosted runner registration

# Authentication
Connected to GitHub
# Runner Registration
Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for ip-172-31-25-149] self-hosted
This runner will have the following labels: 'self-hosted', 'Linux', 'X64' ←
Enter any additional labels (ex. label-1,label-2): [press Enter to skip] []
i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149
```

→ To pass the default work folder name “_work” press **Enter**:



```
aws Services Search [Alt+S] Sydney SEEMANSHU SHUKLA ▾
Self-hosted runner registration

# Authentication

Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for ip-172-31-25-149] self-hosted
This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

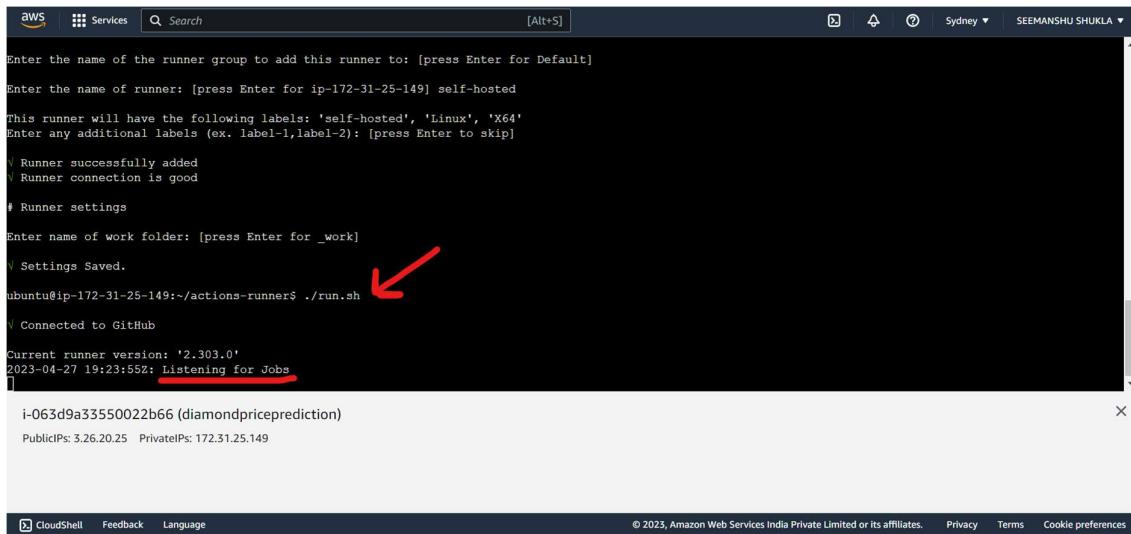
\ Runner successfully added
\ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work] ↴
i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
```

→ Finally, use cmd “`./run.sh`” to run the runner. Observe we get a message “**Listening to Jobs**” which means that the runner is now running and actively listening for new jobs(new push to GitHub repo). In case it listens the new job then it will push it to the server. This is how runner is used to achieve CD(Continuous Deployment).



```
aws Services Search [Alt+S] Sydney SEEMANSHU SHUKLA ▾
Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for ip-172-31-25-149] self-hosted
This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

\ Runner successfully added
\ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]
\ Settings Saved.

ubuntu@ip-172-31-25-149:~/actions-runner$ ./run.sh ↴
2023-04-27 19:23:55Z: Listening for Jobs
i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
```

VI. Code under Using your self-hosted runner:

Runs on: self-hosted

This will be used in yaml file placed under .github/workflows folder.

VII. Now, in GitHub if we again go to Settings>Actions>Runners then we can see our runner that we just created. Currently, it is in **Ideal state** because no new changes are pushed to the GitHub repository. Moment when some changes are made runner will become **Active** post which it will push the changes to server(EC2 instance). This is how we are achieving continuous deployment (CD) in an attempt to create CI-CD pipeline.

The screenshot shows the GitHub Settings page for a repository named 'seemannshu-shukla / RegressionProjectDeployment3'. The 'Actions' tab is selected. Under the 'Runners' section, there is a table with one row. The row contains the status 'self-hosted', 'idle', and a three-dot menu icon. A red arrow points to the 'idle' status indicator.

	Runners	Status	...
<input type="checkbox"/> self-hosted	idle		

10. Setup GitHub secrets:

Example-

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_REGION = ap-southeast-2
AWS_ECR_LOGIN_URL= demo>> 566373416292.dkr.ecr.ap-south-1.amazonaws.com
ECR_REPOSITORY_NAME = simple-app
```

Setup github secrets:

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_REGION = us-east-1
AWS_ECR_LOGIN_URI = demo>> 566373416292.dkr.ecr.ap-south-1.amazonaws.com
ECR_REPOSITORY_NAME = simple-app
```

GitHub is the **deployment provider**. In order to make this deployment work we will need to configure our GitHub with some secrets. Also, please note that the deployment will be happening by using **jobs** mentioned under **.yaml workflow** file which will be using these GitHub secrets as well.

For creating these secrets traverse through the below path:

GitHub>Settings>Secrets and variables>Actions then click **New repository secret**

The screenshot shows the GitHub Settings interface for a repository. The left sidebar has sections like General, Access, Collaborators, Moderation options, Code and automation, Security, and Actions. The Actions section is expanded, showing sub-options like Branches, Tags, Rules (Beta), Actions (selected), Webhooks, Environments, Codespaces, and Pages. The main content area is titled 'Actions secrets and variables'. It contains two tabs: 'Secrets' (selected) and 'Variables'. Under 'Secrets', there are two sections: 'Environment secrets' (Manage environments) and 'Repository secrets'. Both sections show a message: 'There are no secrets for this repository's environments.' and 'There are no secrets for this repository.' respectively. At the bottom right of the main content area, there is a prominent 'New repository secret' button, which is highlighted with a red arrow.

A) Setting up **AWS_ACCESS_KEY_ID** secret:

→ Value for this secret variable can be found in the **.csv file** that we downloaded after creating **IAM user “SeemanshuTest”** in Step 5:

The screenshot shows an Excel spreadsheet titled 'SeemanshuTest_accessKeys.csv'. The table has two rows of data. Row 1 contains two cells: 'Access key ID' in column A and 'Secret access key' in column B. Row 2 contains two cells, both of which are heavily redacted with a large red brush stroke. The Excel ribbon at the top shows the Home tab selected. The table is located in the range A1:B2.

Access key ID	Secret access key
2	[redacted]

→ Enter secret name as highlighted below and secret value copied from “**Access key ID**” column from .csv file and click **Add secret**:

The screenshot shows the GitHub Actions secrets creation interface. On the left, there's a sidebar with various repository settings like General, Access, Collaborators, and Moderation options. The main area is titled "Actions secrets / New secret". It has fields for "Name *" (containing "AWS_ACCESS_KEY_ID") and "Secret *" (containing a long string of characters). At the bottom right of this form is a green "Add secret" button, which is highlighted with a red arrow.

B) Setting up AWS_SECRET_ACCESS_KEY secret:

→ Below we can see that the secret that we set in above step is now reflecting under **Repository secrets** section. Click on **New repository secret**:

The screenshot shows the GitHub Repository secrets page. The left sidebar includes sections for General, Access, Collaborators, and Moderation options, along with Code and automation, Security, and Pages. The main content area is titled "Actions secrets and variables". It features tabs for "Secrets" (selected) and "Variables". A green "New repository secret" button is located at the top right of this section. Below it is a "Repository secrets" section containing a single entry: "AWS_ACCESS_KEY_ID" with a status of "Updated now" and edit/delete icons. A red arrow points to the "New repository secret" button.

→ Enter secret name as highlighted below and secret value copied from “**Secret access key**” column from .csv file and click **Add secret**:

The screenshot shows the GitHub Actions secrets configuration page for a repository named "RegressionProjectDeployment3". The "General" tab is selected. In the "Name" field, the value "AWS_SECRET_ACCESS_KEY" is highlighted with a yellow background. In the "Secret" field, the value "GV[REDACTED]t[REDACTED]B" is shown. A red arrow points to the green "Add secret" button at the bottom of the form.

C) Setting up secret **AWS_REGION** with value **ap-southeast-2**:

→ To find AWS_REGION go to AWS console and on top we get an option to check AWS_REGION. In our case our instance is available in **Sydney** region which is represented as **ap-southeast-2**:

The screenshot shows the AWS Lambda console interface. On the right, a dropdown menu is open, showing various AWS regions. The "Sydney" option is highlighted with a red background and a red arrow points to it. The dropdown menu includes options like "us-east-1", "us-east-2", "us-west-1", "us-west-2", "ap-south-1", "ap-northeast-3", "ap-northeast-2", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ca-central-1", "eu-central-1", "eu-west-1", and "eu-west-2".

→ Below we can see that the secret that we set in above step is now reflecting under **Repository secrets** section. Click on **New repository secret**:

The screenshot shows the GitHub repository settings page for a repository named 'seemanshu-shukla / RegressionProjectDeployment'. The left sidebar has sections like Access, Collaborators, Moderation options, Code and automation, Security, and Actions. Under 'Actions', 'Secrets and variables' is selected. In the main area, there's a 'Repository secrets' section with two entries: 'AWS_ACCESS_KEY_ID' (updated 7 minutes ago) and 'AWS_SECRET_ACCESS_KEY' (updated 1 minute ago). At the bottom right of this section is a green 'New repository secret' button, which is highlighted with a red arrow.

→ Enter the highlighted secret name and secret value based on AWS region(as explained above) and click **Add secret**:

The screenshot shows the 'Actions secrets / New secret' form. It has fields for 'Name *' (containing 'AWS_REGION') and 'Secret *' (containing 'ap-southeast-2'). At the bottom right is a green 'Add secret' button, which is highlighted with a red arrow.

D) Setting up AWS_ECR_LOGIN_URI secret:

Recall In **Step 2** while creating **ECR** repository we copied a **URL**. This URL will be used as secret value here. In our case copied URL is given below:

446896123821.dkr.ecr.ap-southeast-2.amazonaws.com/diamondpriceprediction
In above link "**diamondpriceprediction**" represents ECR repository name which will be used while setting **ECR_REPOSITORY_NAME** secrete variable.

Please Note:

For AWS_ECR_LOGIN_URI we will remove ECR repository name and then pass the resultant link.

That is we will pass: **446896123821.dkr.ecr.ap-southeast-2.amazonaws.com**

→ Below we can see that the secret that we set in above step is now reflecting under **Repository secrets** section. Click on **New repository secret**:

The screenshot shows the GitHub repository settings page for a repository named 'RegressionProjectDeployment3'. On the left, there's a sidebar with sections like 'Code and automation', 'Security', and 'Integrations'. The 'Code and automation' section is expanded, showing 'Branches', 'Tags', 'Rules' (marked as Beta), 'Actions', 'Webhooks', 'Environments', 'Codespaces', and 'Pages'. The 'Actions' section is selected. In the main area, there are two tabs: 'Secrets' (which is selected) and 'Variables'. Under 'Secrets', there are two sections: 'Environment secrets' (which is empty) and 'Repository secrets'. The 'Repository secrets' section contains three entries: 'AWS_ACCESS_KEY_ID' (updated 20 minutes ago), 'AWS_REGION' (updated now), and 'AWS_SECRET_ACCESS_KEY' (updated 13 minutes ago). At the top right of the 'Secrets' tab, there is a green button labeled 'New repository secret' with a red arrow pointing to it.

→ Pass name and secret as highlighted below and then click **Add secret**:

The screenshot shows the 'Actions secrets / New secret' configuration page. The 'Name' field is filled with 'AWS_ECR_LOGIN_URI' and has a yellow highlight box around it. The 'Secret' field is filled with '446896123821.dkr.ecr.ap-southeast-2.amazonaws.com' and also has a yellow highlight box around it. At the bottom of the form, there is a green 'Add secret' button.

E) Setting up the **ECR_REPOSITORY_NAME** secret:

In the URL used above that last substring followed by "/" is **ECR_REPOSITORY_NAME**, that is "**diamondpriceprediction**"

→ Below we can see that the secret that we set in above step is now reflecting under **Repository secrets** section. Click on **New repository secret**:

The screenshot shows the GitHub repository settings page for a repository named 'RegressionProjectDeployment3'. The left sidebar has sections like Access, Collaborators, Moderation options, Code and automation, Security, and Actions. Under 'Actions', the 'Secrets and variables' option is selected. In the main area, there are two sections: 'Environment secrets' (empty) and 'Repository secrets'. Under 'Repository secrets', there are two entries: 'AWS_ACCESS_KEY_ID' (updated yesterday) and 'AWS_ECR_LOGIN_URI' (updated now). A red arrow points to the green 'New repository secret' button at the top right of the 'Repository secrets' section.

→ Enter the below highlighted variable name and secrete. In my case ECR name is **diamondpriceprediction**. After this click **Add secret**:

The screenshot shows the 'Actions secrets / New secret' configuration page. The 'Name *' field contains 'ECR_REPOSITORY_NAME' and the 'Secret *' field contains 'diamondpriceprediction'. A red arrow points to the green 'Add secret' button at the bottom.

→ In the below snip we can see that our secret variable **ECR_REPOSITORY_NAME** is now reflecting under **Repository secrets** section meaning that it has been created successfully:

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with options like Tags, Rules, Actions, Webhooks, Environments, Codespaces, and Pages. Under the Actions section, there's a 'Secrets and variables' subsection with an 'Actions' button. The main area is titled 'Repository secrets' and lists several environment variables:

Variable	Last Updated	Action Buttons
AWS_ACCESS_KEY_ID	Updated 41 minutes ago	
AWS_ECR_LOGIN_URL	Updated 9 minutes ago	
AWS_REGION	Updated 22 minutes ago	
AWS_SECRET_ACCESS_KEY	Updated 35 minutes ago	
ECR_REPOSITORY_NAME	Updated now	

11. Starting the deployment and visualizing it inside GitHub:

Make some changes like add an extra space in app.py and push it to GitHub after which **self-hosted Runner** will start listening for the **jobs** as defined under **.yaml workflow file** after which deployment will get initiated in CICD fashion as defined under .yaml file. As we initiate the deployment “**actions/checkout@v3**” (as defined under .yaml) will scan the entire GitHub repository and will look for the latest changes. As this is the first time runner is getting activated hence, it will push entire app in the server. Later based on changes made in codebase it will only push the latest changes to the server deployment:

The screenshot shows a VS Code interface with the following details:

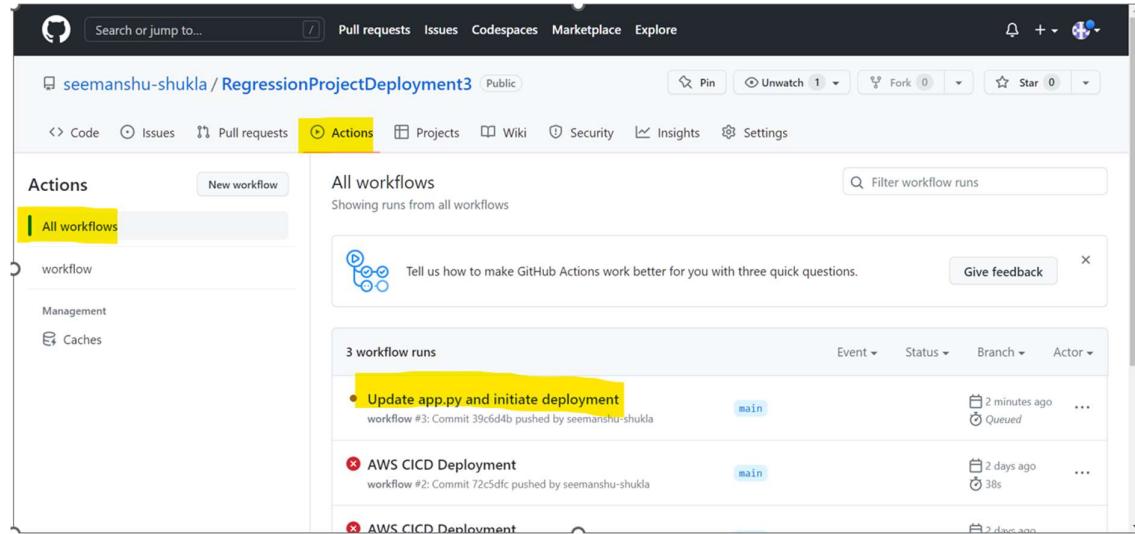
- EXPLORER**: Shows the project structure for 'GEM-PRICE-PREDICTION3' containing files like .github\workflows\main.yaml, Dockerfile, app.py, Dockerfile, README.md, requirements.txt, and setup.py.
- TERMINAL**: Shows the command line output of a git push operation:


```
1 file changed, 2 insertions(+), 1 deletion(-)

(c:\DS_project\Gem-Price-Prediction3\venv) C:\DS_project\Gem-Price-Prediction3>git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 336 bytes | 336.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/seemanshu-shukla/RegressionProjectDeployment3.git
  72c5dfe..39c6d4b main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

In above snip observe **M** adjacent to app.py file in Vscode which represents modified as we have added extra space just to initiate the deployment.

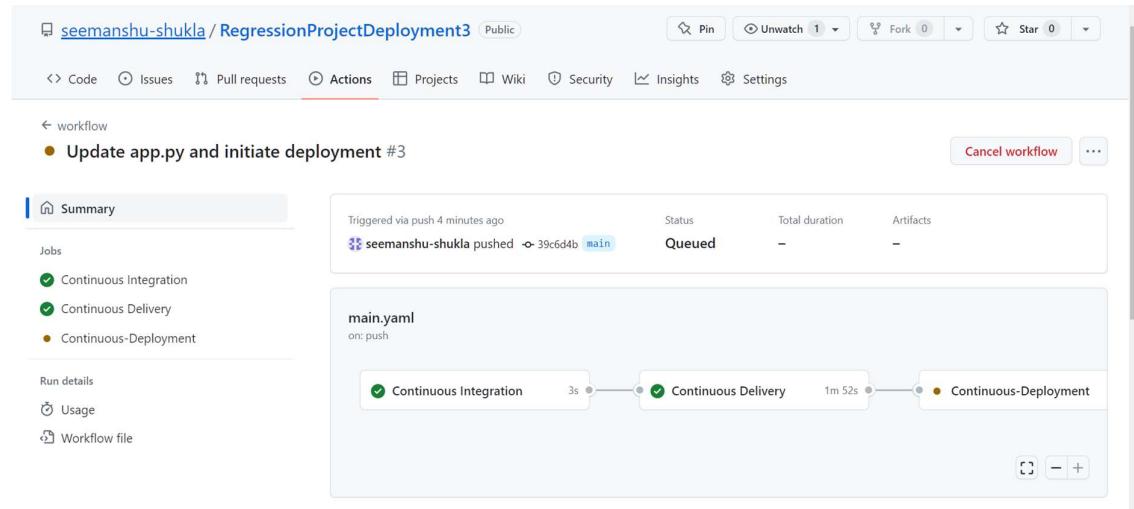
- As our changes are successfully committed to GitHub then observe that under **Actions** a new workflow is created:



The screenshot shows the GitHub Actions page for the repository "seemanshu-shukla / RegressionProjectDeployment3". The "Actions" tab is selected. On the left, there's a sidebar with "Actions", "New workflow", "All workflows" (which is highlighted), "workflow", "Management", and "Caches". The main area shows "All workflows" with a search bar "Filter workflow runs". Below it, there's a feedback card "Tell us how to make GitHub Actions work better for you with three quick questions." and a "Give feedback" button. The "3 workflow runs" section lists three entries:

- **Update app.py and initiate deployment** (Workflow #3: Commit 39c6d4b pushed by seemanshu-shukla) - Status: main, 2 minutes ago, Queued
- ✗ **AWS CICD Deployment** (Workflow #2: Commit 72c5dfc pushed by seemanshu-shukla) - Status: main, 2 days ago, 38s
- ✗ **AWS CICD Deployment** (Workflow #1: Commit 39c6d4b pushed by seemanshu-shukla) - Status: main, 2 days ago

- If we open this workflow we can visualize our entire deployment workflow happening in CICD fashion as defined under the **workflow file** that is **.yaml** file:



The screenshot shows the details of the workflow "Update app.py and initiate deployment" (Workflow #3). The top navigation bar includes "Code", "Issues", "Pull requests", "Actions", "Projects", "Wiki", "Security", "Insights", and "Settings". The "Actions" tab is selected. The workflow summary shows it was triggered by a push 4 minutes ago, pushed by "seemanshu-shukla" to branch "main", and is currently "Queued". It also shows "Total duration" and "Artifacts" as "-". The "main.yaml" file is displayed with its content:

```
on: push
```

Below the file content, a diagram visualizes the workflow steps:

```
graph LR; CI[Continuous Integration] --> CD[Continuous Delivery]; CD --> CD2[Continuous Deployment]
```

The steps are labeled "Continuous Integration", "Continuous Delivery", and "Continuous Deployment". The "Continuous Integration" step took 3s, "Continuous Delivery" took 1m 52s, and "Continuous Deployment" is ongoing.

→ Click on **Continuous Integration** to visualize it:

The screenshot shows a GitHub repository page for 'seemanshu-shukla / RegressionProjectDeployment3'. A workflow named 'Update app.py and initiate deployment #3' is displayed. On the left, a sidebar lists 'Jobs' such as 'Continuous Integration', 'Continuous Delivery', and 'Continuous-Deployment'. The 'Continuous Integration' job is selected and expanded, showing a list of steps: 'Set up job', 'Checkout Code', 'Lint code', 'Run unit tests', 'Post Checkout Code', and 'Complete job', all of which have been completed successfully.

→ Click on **Continuous Delivery** to visualize it:

The screenshot shows the same GitHub repository page. The 'Continuous Delivery' job is selected and expanded, showing a more complex list of steps: 'Set up job', 'Checkout Code', 'Install Utilities', 'Configure AWS credentials', 'Login to Amazon ECR', 'Build, tag, and push image to Amazon ECR' (which took 1m 32s), 'Post Login to Amazon ECR', 'Post Configure AWS credentials', 'Post Checkout Code', and 'Complete job'. Most steps are marked as completed, except for the build step which is currently in progress.

- Click on **Continuous-Deployment** to visualize it. Below we can clearly see we are getting a message that "***Waiting for a runner to pick up this job...***"

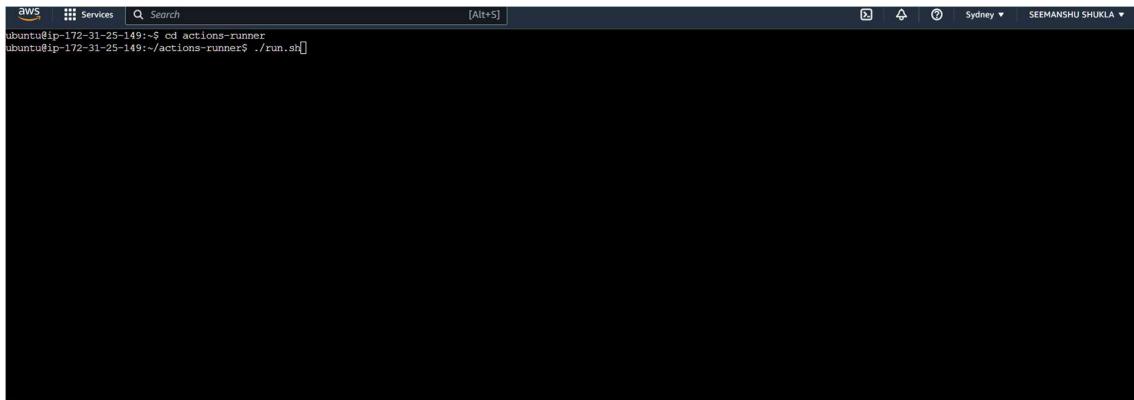
The screenshot shows a GitHub Actions workflow summary for a repository named 'seemanshu-shukla / RegressionProjectDeployment3'. The workflow has three steps: 'Continuous Integration', 'Continuous Delivery', and 'Continuous-Deployment'. The 'Continuous-Deployment' step is currently active, showing a message: 'Requested labels: self-hosted Job defined at: seemanshu-shukla/RegressionProjectDeployment3/.github/workflows/main.yaml@refs/heads/main Waiting for a runner to pick up this job...'. The workflow was started 6m 44s ago.

- On further investigating found that our **Runner** was in the **Offline** state. This is the reason why we were getting the message that "***Waiting for a runner to pick up this job...***". In my case after creating and running the runner I started the deployment on next day. So, may be due to inactivity my runner's object state changed to Offline.

The screenshot shows the GitHub Actions settings page for the same repository. Under the 'Runners' section, there is a table listing a single runner: 'self-hosted' (self-hosted, Linux, X64). The status of this runner is 'Offline'. There is a green button labeled 'New self-hosted runner' available for creating a new runner.

Please note: Before initiating deployment make sure that runner is not in Offline state otherwise deployment will stop/halt at Continuous-Deployment stage.

- As a resolution we again runed the runner inside EC2 instance as we did earlier while setting up runner in EC2 in **Step 9**. For this first we changed the directory to **actions-runner** using “**cd actions-runner**” command and then used “**./run.sh**” command to run the runner.

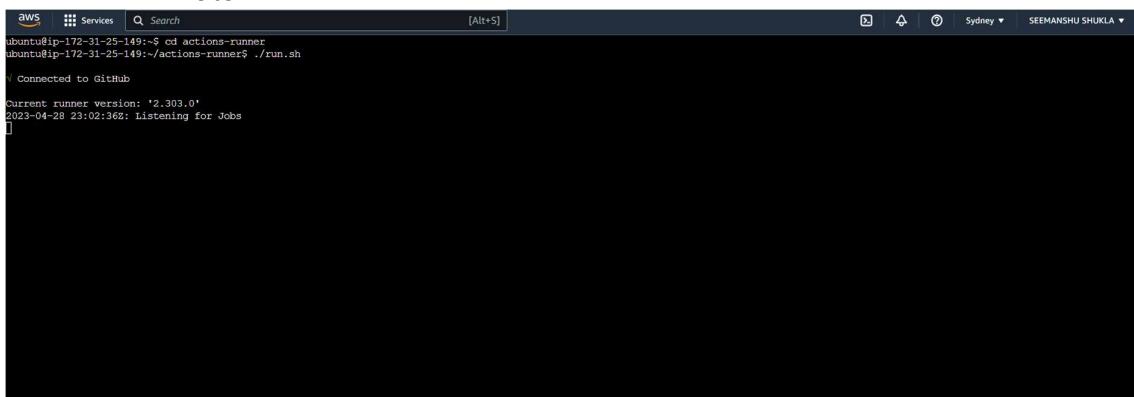


```
aws Services Search [Alt+S] Sydney SEEMANSHU SHUKLA
ubuntu@ip-172-31-25-149:~$ cd actions-runner
ubuntu@ip-172-31-25-149:~/actions-runner$ ./run.sh

i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
```

- We can see below that now we are getting a message that “***Listening for Jobs***”



```
aws Services Search [Alt+S] Sydney SEEMANSHU SHUKLA
ubuntu@ip-172-31-25-149:~$ cd actions-runner
ubuntu@ip-172-31-25-149:~/actions-runner$ ./run.sh

Connected to GitHub
Current runner version: '2.303.0'
2023-04-28 23:02:36Z: Listening for Jobs

i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
```

→ To verify, let's now check the state of our Runner object. Now we can see that it is changed from **Offline** to **Ideal**. Therefore, now we are good to resume our deployment.

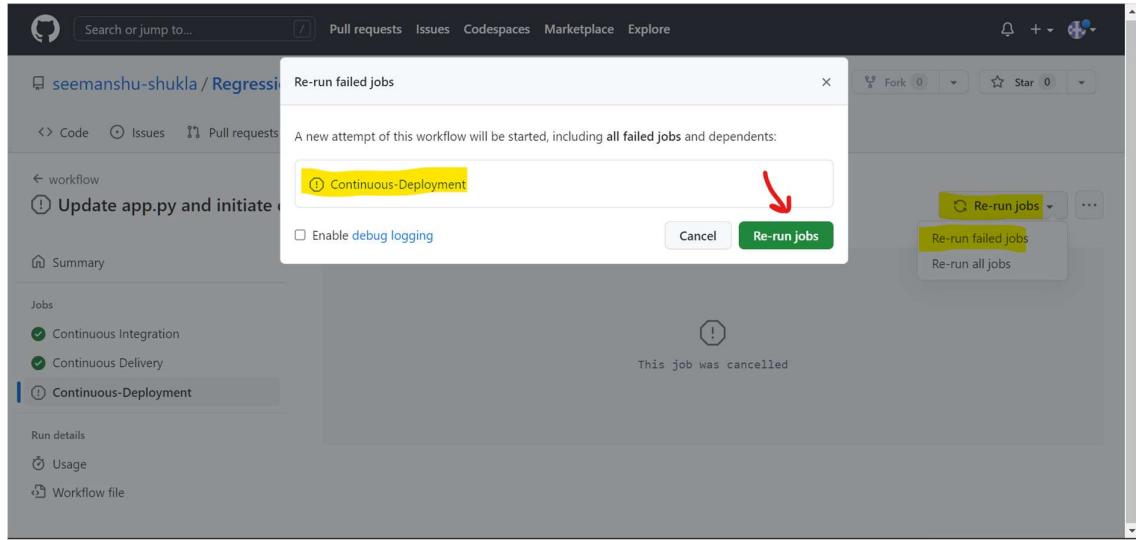
The screenshot shows the GitHub Settings page for the repository "seemanshu-shukla / RegressionProjectDeployment3". The left sidebar has a "Runners" tab selected. The main area is titled "Runners" and contains a table with one row:

Runners	Status
self-hosted (self-hosted) (Linux) (X64)	Idle

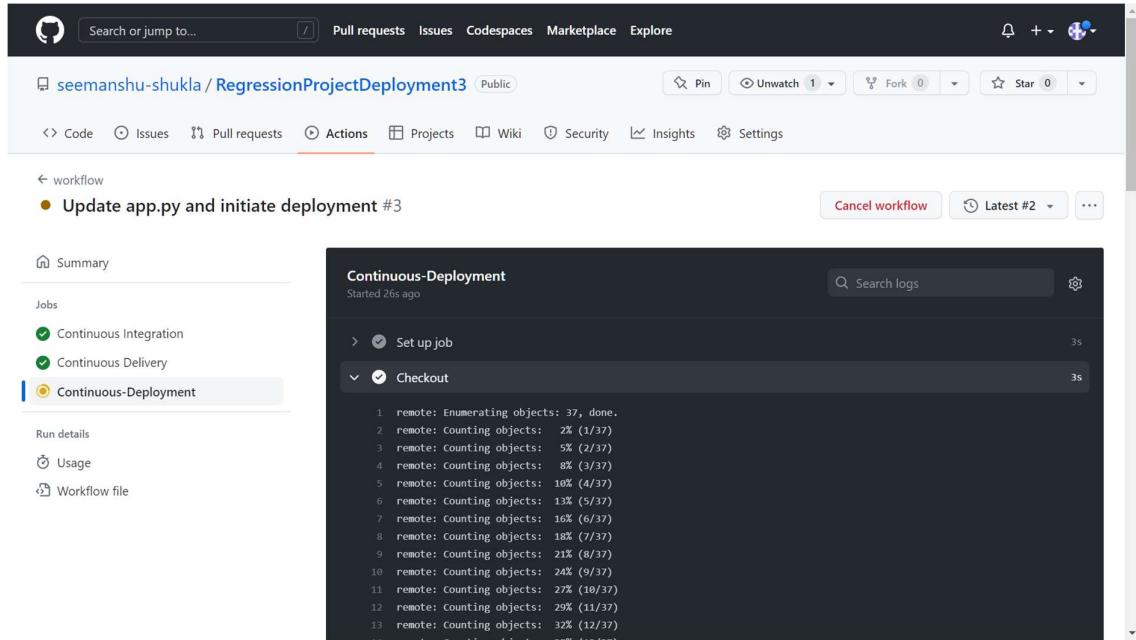
→ Now, we will go back to our workflow and re-run the failed jobs to complete the deployment. Click **Re-run jobs** drop down:

The screenshot shows the GitHub Actions workflow summary for the repository "seemanshu-shukla / RegressionProjectDeployment3". The workflow is named "Update app.py and initiate deployment #3". The "Continuous-Deployment" job is listed as failed, indicated by a red exclamation mark icon. A red arrow points to the "Re-run jobs" dropdown menu next to the job status.

→ Re-run the failed job by clicking the **Re-run jobs**:



→ Now, we can clearly visualize **Continuous-Deployment** in progress state:



- Once, a green check mark is added adjacent to **Continuous-Deployment** then it signifies that it is implemented successfully.

- Also, if we go back to our EC2 instance we can see the message "**Job Continuous-Deployment completed with result: Succeeded**".

```

aws Services Search [Alt+S]
ubuntu@ip-172-31-25-149:~ cd actions-runner
ubuntu@ip-172-31-25-149:~/actions-runner$ ./run.sh
Connected to GitHub
Current runner version: '2.303.0'
2023-04-28 23:02:36Z: Listening for Jobs
2023-04-28 23:07:17Z: Running job: Continuous-Deployment
2023-04-28 23:08:01Z: Job Continuous-deployment completed with result: Succeeded
[...]

```

i-063d9a33550022b66 (diamondpriceprediction)
PublicIPs: 3.26.20.25 PrivateIPs: 172.31.25.149

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

12. Some additional changes:

- In workflow .yaml file we mentioned port as 8080:8080 (docker container port and instance port mapping) while running the docker image but nowhere it was configured.
So, we need to configure this port at EC2 instance as well as application(app.py) level which later will run as a docker container.

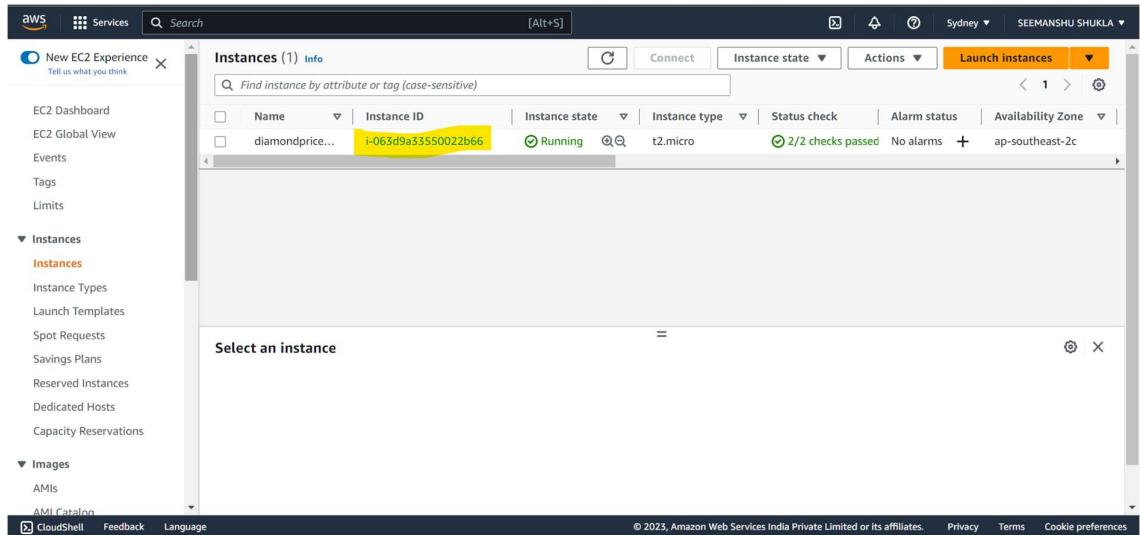
.yaml file:

```
github.com/seemanshu-shukla/RegressionProjectDeployment3/blob/main/github/workflows/main.yaml
```

```
70      - name: Checkout
71        uses: actions/checkout@v3
72
73      - name: Configure AWS credentials
74        uses: aws-actions/configure-aws-credentials@v1
75        with:
76          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
77          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
78          aws-region: ${{ secrets.AWS_REGION }}
79
80      - name: Login to Amazon ECR
81        id: login-ecr
82        uses: aws-actions/amazon-ecr-login@v1
83
84
85      - name: Pull latest images
86        run: |
87          docker pull ${secrets.AWS_ECR_LOGIN_URI}/${secrets.ECR_REPOSITORY_NAME}:latest
88
89      # - name: Stop and remove container if running
90      #   run: |
91      #     docker ps -q --filter "name=mltest" | grep -q . && docker stop mltest && docker rm -fv mltest
92
93      - name: Run Docker Image to serve users
94        run: |
95          docker run -d -p 8080:8080 --ipc="host" --name=mltest -e 'AWS_ACCESS_KEY_ID=${secrets.AWS_ACCESS_KEY_ID}' -e 'AWS_SECRET_ACCESS_KEY=${secrets.AWS_SECRET_ACCE
96      - name: Clean previous images and containers
97        run: |
98          docker system prune -f
```

- If we run our deployed application using public domain of EC2 instance without configuring 8080 port at both EC2 instance and container/app level then we will be getting an error. To eliminate this error we need to configure 8080 port at both the mentioned levels.

- Going to EC2 instance and under Instances clicking highlighted Instance ID:



→ Copying the **Public IPv4 address** and opening it in the new tab of browser:

The screenshot shows the AWS EC2 Instances page for an instance named 'diamondpriceprediction'. The instance ID is i-063d9a33550022b66. Key details include:

- Public IPv4 address:** 3.26.20.25 [open address]
- Private IPv4 address:** 172.31.25.149
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-25-149.ap-southeast-2.compute.internal
- Instance type:** t2.micro
- VPC ID:** vpc-0f44160d3cdb09262
- Subnet ID:** subnet-0e99c35b09557a214

→ Getting an error stating "*This site can't be reached*".

The screenshot shows a web browser window with the URL 3.26.20.25. The page displays an error message:

This site can't be reached
3.26.20.25 refused to connect.
Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED

Buttons at the bottom: Reload and Details.

→ To counter the above issue as discussed we will **Firstly, configure the 8080 port in EC2 instance**. Open the instance where application is deployed and open **Security groups** under **Security** section.

Instance summary for i-063d9a33550022b66 (diamondpriceprediction) [Info](#)

Updated less than a minute ago

Instance ID	i-063d9a33550022b66 (diamondpriceprediction)	Public IPv4 address	3.26.20.25 open address	Private IPv4 addresses	172.31.25.149
IPv6 address	-	Instance state	Running	Public IPv4 DNS	ec2-3-26-20-25.ap-southeast-2.compute.amazonaws.com open address
Hostname type	IP name: ip-172-31-25-149.ap-southeast-2.compute.internal	Private IP DNS name (IPv4 only)	ip-172-31-25-149.ap-southeast-2.compute.internal	Elastic IP addresses	-
Answer private resource DNS name	IPv4 (A)	Instance type	t2.micro	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address	3.26.20.25 [Public IP]	VPC ID	vpc-0f44160d3cd09262	Auto Scaling Group name	-
IAM Role	-	Subnet ID	subnet-0e99c35b09557a214		
IMDSv2	Optional				

[Details](#) **Security** [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

Security details

IAM Role	Owner ID	Launch time
-	446896123821	Thu Apr 27 2023 20:59 GMT+0530 (India Standard Time)

Security groups

[sg-0a87b7fc500606ff5 \(launch-wizard-1\)](#)

Inbound rules

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

→ Click Edit inbound rules:

EC2 > Security Groups > sg-0a87b7fc500606ff5 - launch-wizard-1

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0a87b7fc500606ff5	launch-wizard-1 created 2023-04-27T14:04:50.803Z	vpc-0f44160d3cd09262
Owner	Inbound rules count	Outbound rules count	
446896123821	3 Permission entries	1 Permission entry	

[Inbound rules](#) [Outbound rules](#) [Tags](#)

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

Inbound rules (3)

Name	Security group rule...	IP version	Type	Protocol	Port range

[Edit inbound rules](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

→ Click Add rule:

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules	Type	Protocol	Port range	Source	Description - optional
sgr-098fd995a2e5d6f65	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0e0bab79e4bedda90	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-01faf84d70300e060	SSH	TCP	22	Custom	0.0.0.0/0

Add rule

Cancel Preview changes Save rules

→ Enter the highlighted details and click **Save rules**:

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules	Type	Protocol	Port range	Source	Description - optional
sgr-098fd995a2e5d6f65	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0e0bab79e4bedda90	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-01faf84d70300e060	SSH	TCP	22	Custom	0.0.0.0/0
-	Custom TCP	TCP	8080	Anywhere	0.0.0.0/0

Add rule

Cancel Preview changes Save rules

→ Inbound rule added successfully. This means that port 8080 has been successfully configured at EC2 instance level.

The screenshot shows the AWS EC2 Security Groups page. A green banner at the top states: "Inbound security group rules successfully modified on security group (sg-0a87b7fc500606ff5 | launch-wizard-1)". A red arrow points from this banner to the title of the main content area, "sg-0a87b7fc500606ff5 - launch-wizard-1". Below the banner, there's a "Details" section with fields like Security group name (launch-wizard-1), Security group ID (sg-0a87b7fc500606ff5), Description (launch-wizard-1 created 2023-04-27T14:04:50.803Z), VPC ID (vpc-0f44160d3cdb09262), Owner (446896123821), Inbound rules count (4 Permission entries), and Outbound rules count (1 Permission entry). Below the details, there are tabs for "Inbound rules", "Outbound rules", and "Tags". A message in the center says, "You can now check network connectivity with Reachability Analyzer", with a "Run Reachability Analyzer" button.

→ **Secondly, we will configure 8080 port at app.py level** which will be running as a docker container inside the EC2 instance (server). Changes done at app.py level will also need to be pushed to the GitHub and from GitHub to server via runner.

Since, app.py will run as a docker container so app.run(host="0.0.0.0", port=8080) will signify that run this application inside docker container's local host using 8080 port number.

Please note, that our docker image for the entire application (app.py and all other dependencies like setup.py, src, artifacts etc.) is pushed inside AWS ECR private repository. From here, this docker image is getting pulled inside EC2 instance where this image is running as dockized or containerized application.

The screenshot shows the VS Code code editor with a Python file named "app.py" open. The code contains the following snippet:

```

38
39     return render_template('results.html', final_result=results)
40
41
42
43
44
45
46 if __name__ == "__main__":
47     #app.run(host='0.0.0.0', debug=True)
48     app.run(host='0.0.0.0', port=8080)

```

A red arrow points to the line "app.run(host='0.0.0.0', port=8080)". The code editor interface includes the Explorer, Editor, and Terminal panes. The terminal shows the command "PS C:\DS_project\Gem-Price-Prediction3>".

13. Accessing the deployed application:

→ Go to EC2 instance and click on Instance ID under Instances:

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store. The main content area has a search bar at the top. Below it is a table titled 'Instances (1) Info' with one row. The row contains columns for Name (diamondprice...), Instance ID (i-063d9a33550022b66), Instance state (Running), Instance type (t2.micro), Status check (2/2 checks passed), Alarm status (No alarms), Availability Zone (ap-southeast-2c), and Public IPv4 DNS (ec2-3-26-20-25.ap-sout...). A modal window titled 'Select an instance' is open at the bottom, showing the same instance details.

→ Copy the Public IPv4 address by using the copy icon:

The screenshot shows the 'Instance summary for i-063d9a33550022b66 (diamondpriceprediction)' page. The left sidebar is identical to the previous screenshot. The main content area displays detailed information about the instance. Under the 'Details' tab, the 'Public IPv4 address' field is highlighted with a blue border, indicating it's selected for copying. Other fields shown include Instance ID (i-063d9a33550022b66), IP6 address (-), Hostname type (IP name: ip-172-31-25-149.ap-southeast-2.compute.internal), Answer private resource DNS name (IPV4 (A)), Auto-assigned IP address (3.26.20.25 [Public IP]), IAM Role (-), IMDSv2 (Optional), Instance state (Running), VPC ID (vpc-0f44160d5cd09262), Subnet ID (subnet-0e99c35b09557a214), Private IPv4 addresses (172.31.25.149), Public IPv4 DNS (ec2-3-26-20-25.ap-southeast-2.compute.amazonaws.com), and Auto Scaling Group name (-). At the bottom, there are tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags.

→ Once URL is copied successfully then a green check mark with green text will appear stating the **Public IPv4 address copied**.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances, Images, and Elastic Block Store. The main area displays an instance summary for an instance named 'diamondpriceprediction'. The summary includes details such as Instance ID (i-063d9a33550022b66), IPv4 address (3.26.20.25), Instance state (Running), Hostname type (IP name: ip-172-31-25-149.ap-southeast-2.compute.internal), Answer private resource DNS name (IPv4 (A)), Auto-assigned IP address (3.26.20.25 [Public IP]), IAM Role (None), IMDSv2 (Optional), Private IP DNS name (ip-172-31-25-149.ap-southeast-2.compute.internal), Instance type (t2.micro), VPC ID (vpc-0f44160d3cdb09262), Subnet ID (subnet-0e99c35b09557a214), and Auto Scaling Group name (None). At the bottom, there are tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. A tooltip 'Public IPv4 address copied' with a green checkmark is overlaid on the Public IP address field.

→ Open a new tab in the current browser and paste the copied URL as shown below:

The screenshot shows a web browser window with a single tab open at the URL 3.26.20.25:8080. The page content is a simple welcome message: "Welcome!" followed by "This is AI Powered Platform Where You Can Predict The Price of Your Gem Stones".

➔ User “/predict” to get re-directed to predict web page where we can provide the input based on which prediction on price will be made.

Not secure | 3.26.20.25:8080/predict

Carat:

Depth:

Table:

x:

y:

z:

Cut: Fair

Color: D

Clarity: If1

Submit

➔ Enter the values and click **Submit**:

Not secure | 3.26.20.25:8080/predict

Carat: 23

Depth: 24

Table: 34

x: 11

y: 17

z: 16

Cut: Fair

Color: D

Clarity: If1

Submit

- ➔ Getting redirected to results web page that is depicting the predicted price of the gem stone based on input features:



14. Deleting deployment:

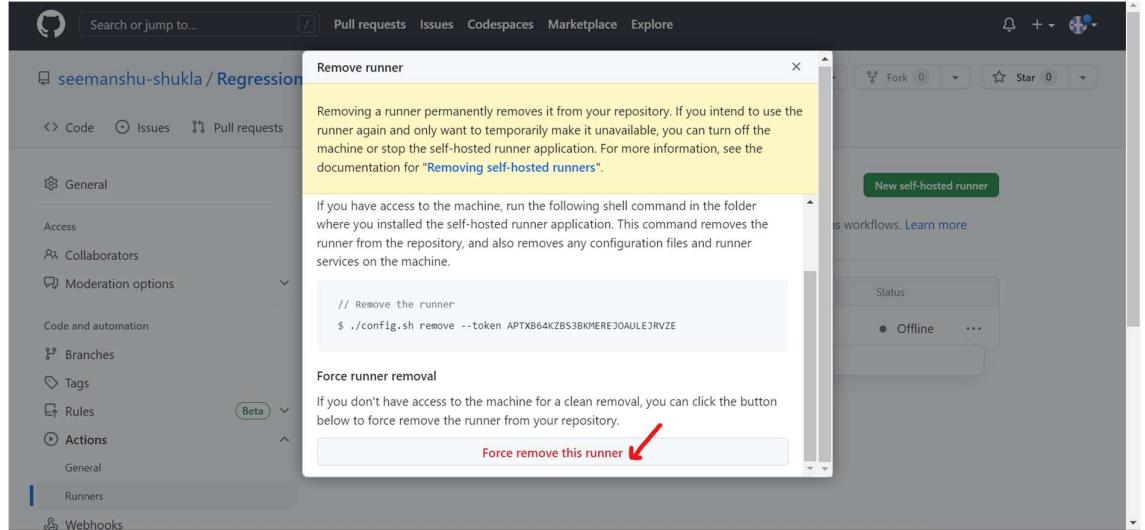
Since, this deployment was done just for learning purpose we will be deleting this deployment so that AWS does not charge us anything extra.

A. Delete Runner in GitHub:

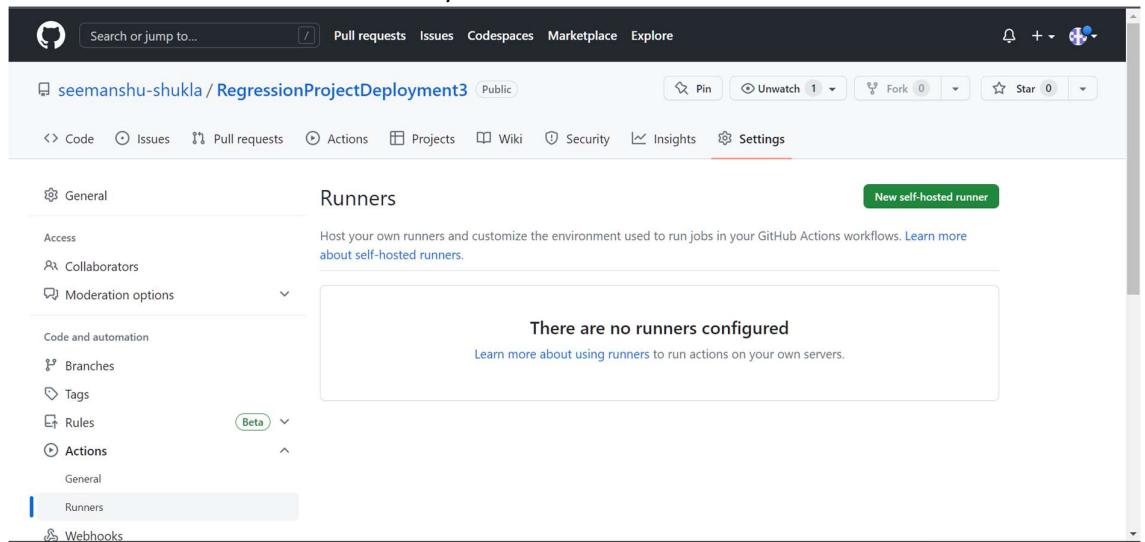
- ➔ Go to GitHub > Settings > Actions > Runners and select the “...” icon and click Remove runner

A screenshot of the GitHub Actions Runners settings page. The left sidebar shows navigation options like General, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, General, Runners, and Webhooks. The main area is titled "Runners" and contains instructions: "Host your own runners and customize the environment used to run jobs in your GitHub Actions workflows. Learn more about self-hosted runners." Below this is a table with one row: "self-hosted" (self-hosted, Linux, X64), Status (Offline), and a red "Remove runner" button.

→ In the following screen click **Force remove this runner**:

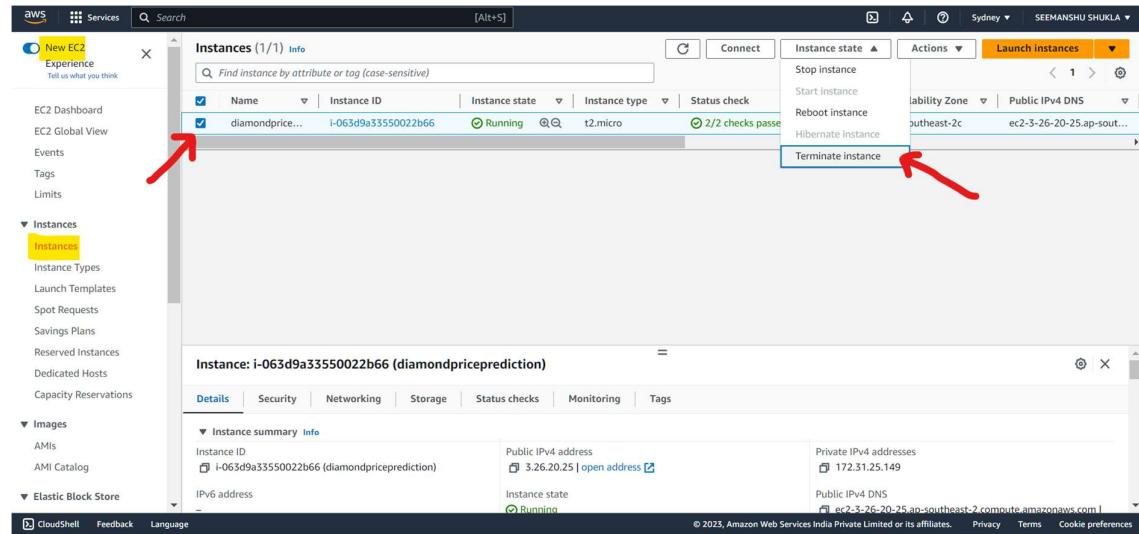


→ Runner deleted successfully:

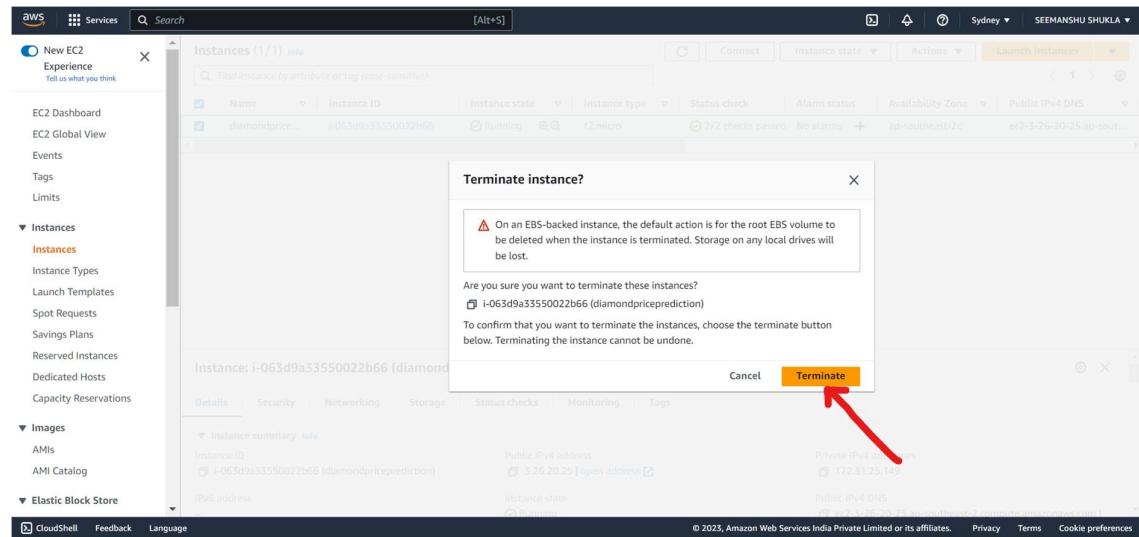


B. Deleting the EC2 instance:

→ Go to **EC2 instance** > select check box for running instance > on top click “**Instance state**” drop down and select **Terminate instance**:



→ Click Terminate:



→ Post this the Instance state will be changed from Running to Terminated.

C. Deleting ECR(Elastic Container Repository):

→ Open ECR and under **Repositories** select the check box adjacent to repository that needs to be deleted and then on top click the **delete** option:

The screenshot shows the Amazon ECR Repositories page. On the left, there's a sidebar with options like 'Private registry', 'Public registry', and 'Repositories'. The 'Repositories' option is highlighted with a yellow box and has a red arrow pointing to its checkbox. In the main area, there's a table titled 'Private repositories (1 of 1)'. The table has columns for 'Repository name', 'URI', 'Created at', 'Tag immutability', 'Scan frequency', 'Encryption type', and 'Pull through cache'. One row is visible for 'diamondpriceprediction'. At the top right of the table, there are buttons for 'View push commands', 'Delete', 'Actions', and 'Create repository'. A red arrow points to the 'Delete' button.

→ As instructed type **delete** and click **Delete button**:

The screenshot shows a confirmation dialog box. The title is 'Delete diamondpriceprediction'. The message inside says, 'Are you sure you want to delete diamondpriceprediction? Your images will also be deleted.' Below that, it says, 'To confirm deletion, type **delete** in the field.' There is a text input field containing the word 'delete'. At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Delete', with a red arrow pointing to the 'Delete' button.

→ ECR repository is deleted successfully:

The screenshot shows the AWS ECR console. The top navigation bar includes the AWS logo, services dropdown, search bar, and account information for Sydney and Seemanshu Shukla. The main content area has a green header bar with the text "Successfully deleted repository diamondpriceprediction". Below this, the breadcrumb navigation shows "Amazon ECR > Repositories". A tab bar at the top of the main content area has "Private" selected. The main section is titled "Private repositories" and contains a search bar and a table with columns: Repository name, URI, Created at, Tag immutability, Scan frequency, Encryption type, and Pull through cache. The table displays the message "No repositories" and "No repositories were found". At the bottom of the page, there are links for CloudShell, Feedback, Language, and copyright information for 2023.

- D. Deleting the IAM user(secrete key that got generated as .csv file which later was used to define our secrete variables):

→ Go to IAM and under **Users** select check box adjacent to the user that needs to be deleted and click **Delete** option on top:

The screenshot shows the AWS IAM console. The top navigation bar includes the AWS logo, services dropdown, search bar, and account information for Global and Seemanshu Shukla. The main content area has a yellow box around the "Identity and Access Management (IAM)" service name. The left sidebar shows "Access management" with "User groups" and "Users" selected, indicated by a yellow box and a red arrow pointing to it. The main content area is titled "Users (Selected 1/1) Info" and contains a table with columns: User name, Groups, Last activity, MFA, Password age, and Active key age. One row is selected, showing "SeemanshuTest" with "None" in all other columns. At the top right of the table, there are "Delete" and "Add users" buttons, with a red arrow pointing to the "Delete" button. The bottom of the page includes links for CloudShell, Feedback, Language, and copyright information for 2023.

➔ IAM user is deleted successfully:

The screenshot shows the AWS Identity and Access Management (IAM) service interface. In the top navigation bar, there is a green banner message: "User SeemanshuTest deleted." Below this, the main content area is titled "Users (0) info". A sub-header says, "An IAM user is an identity with long-term credentials that is used to interact with AWS in an account." There is a search bar labeled "Find users by username or access key". To the right of the search bar are buttons for "Delete" and "Add users". Below the search bar is a table header with columns: "User name", "Groups", "Last activity", "MFA", "Password age", and "Active key age". A note at the bottom of the table area states "No resources to display". On the left side of the screen, there is a sidebar with several sections: "Identity and Access Management (IAM)", "Dashboard", "Access management" (which includes "User groups", "Users", "Roles", "Policies", "Identity providers", and "Account settings"), "Access reports" (which includes "Access analyzer", "Archive rules", "Analyzers", "Settings", "Credential report", "Organization activity", and "Service control policies (SCPs)"), and "CloudShell", "Feedback", and "Language" buttons at the bottom.