Cloud Deployment in in EC2 instance in AWS

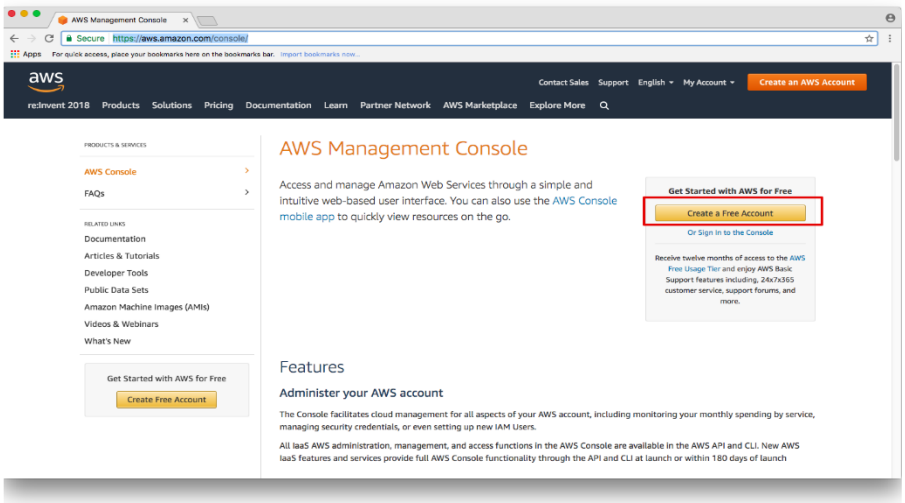
Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

Steps To Be Followed:

1. Create an AWS account

Signup using email

- Open AWS home page
- Choose Create an AWS Account
- In Root user email address, enter your email address, edit the AWS account name, and then Verify email address.
- Create a password as per the requirements

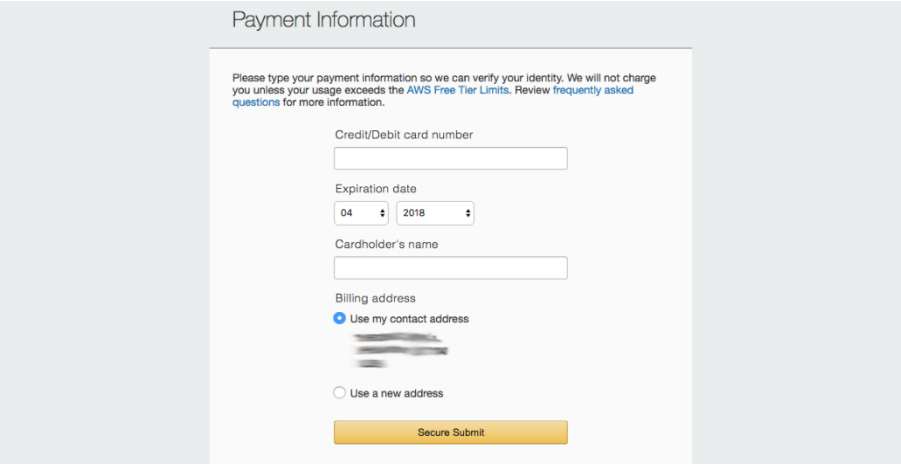


Add Contact Information

- Select Personal or Business
- Enter your personal or business information
- Read and accept AWS customer agreement
- Choose Continue

Add Payment method

- On the Billing information page, enter the information about your payment method
- Choose Verify and Add



Verify Phone number

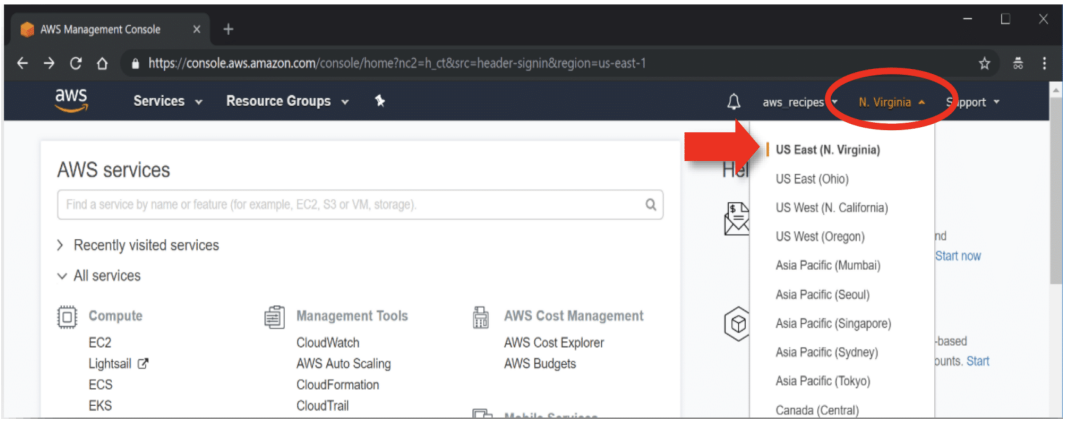
- On the Confirm your identity page, select a contact method to receive a verification code
- In a few moments, an automated system contacts you
- Enter the PIN you receive, and then choose Continue.

Select Plan Type

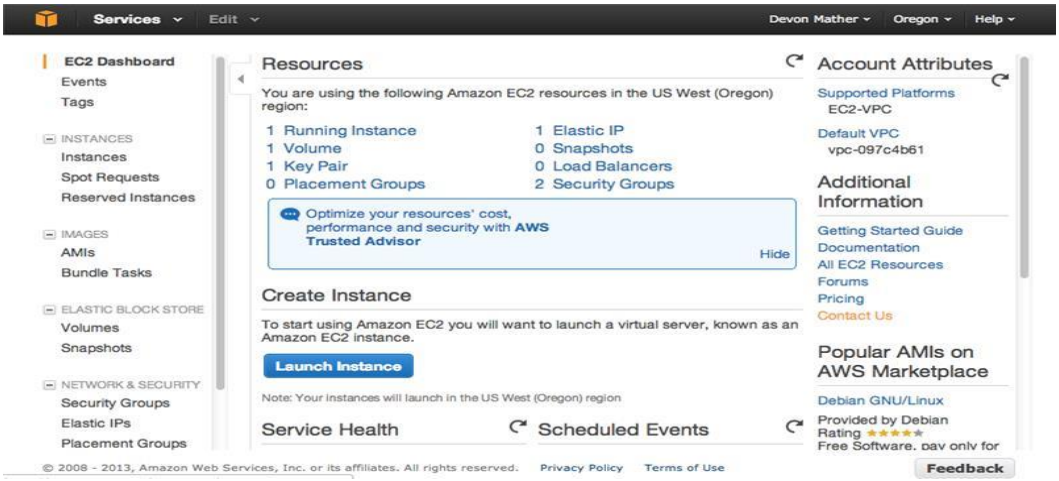
- On the Select a support plan page, choose one of the available Support plans
- Choose Complete sign up

2. Choosing a region

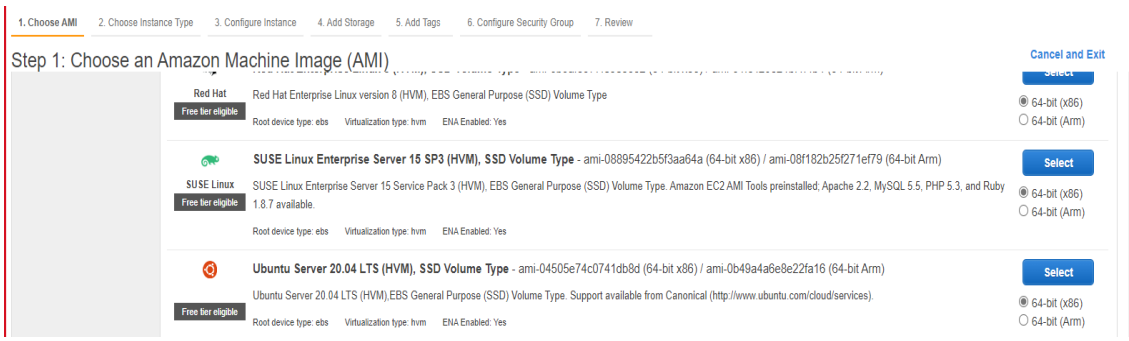
- Enter AWS management Console
- On the navigation bar, choose the name of the currently displayed Region. Then choose the Region to which you want to switch



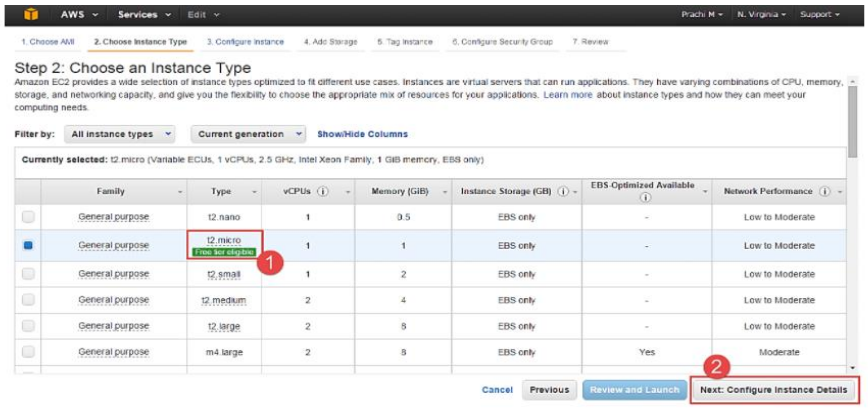
3. Search for EC2 on in search bar on navigation bar in the console
4. Launch an instance in EC2 dashboard



i. Choose an AMI of your choice (An AMI is an Amazon Machine Image. It is a template basically of an Operating System platform which you can use as a base to create your instance)



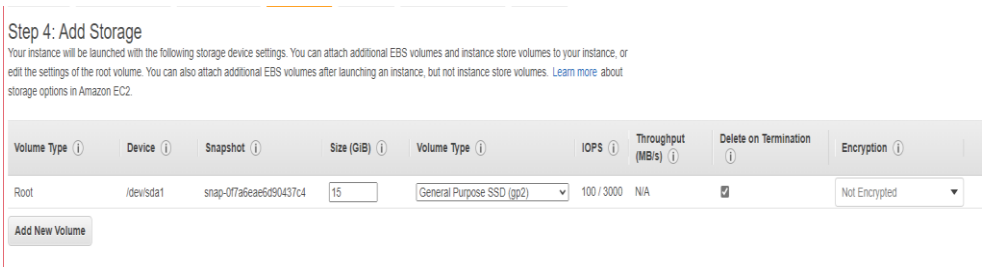
ii. Choose the type of instance you require based on your business needs



iii. Configure instance details

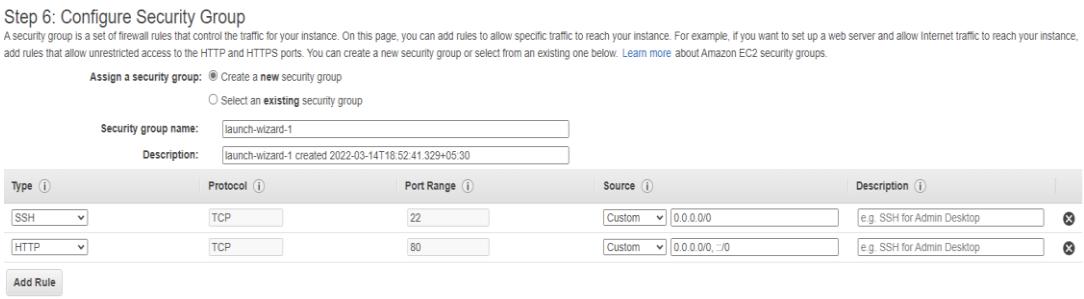
You can keep everything as default for now

iv. Add Storage



v. Tag instance
you can tag your instance with a key-value pair

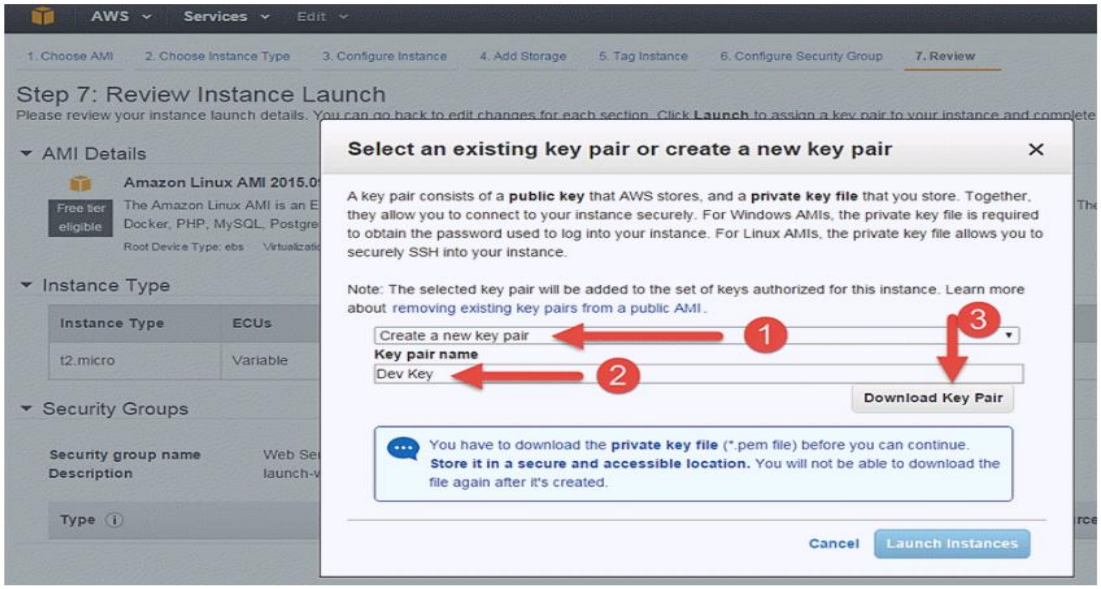
vi. Configure Security Groups
In this next step of configuring Security Groups, you can restrict traffic on your instance ports



vii. Review Instances

- Review all our choices and parameters and go ahead to launch our instance
- You will be asked to create a key pair to login to you an instance. A key pair is a set of public-private keys.
- AWS stores the private key in the instance, and you are asked to download the private key

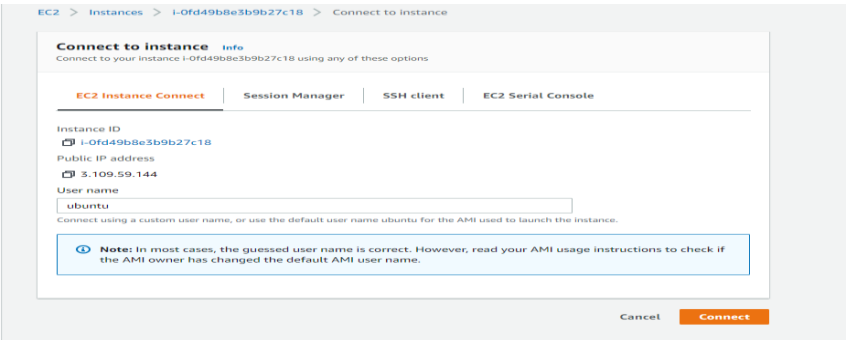
    1. Create a new key pair
    2. Give a name to your key
    3. Download and save it in your secured folder



viii. Download the key and launch the instance

5. Connect to Instance

- Click on EC2 instance connect and hit connect button



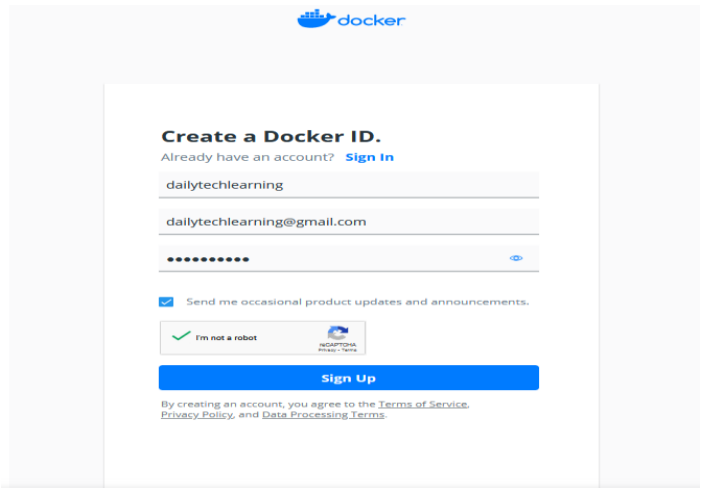
- Once connected, you will be provided with an ubuntu terminal to work on.

```
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Mar 14 13:45:50 UTC 2022

  System load:                      0.0
  Usage of /:                       23.5% of 14.48GB
  Memory usage:                     60%
  Swap usage:                       0%
  Processes:                        135
  Users logged in:                  0
  IPv4 address for br-ce91c7f2853f: 172.19.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            172.31.13.89

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

   https://ubuntu.com/aws/pro

40 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: Thu Mar 10 14:58:03 2022 from 13.233.177.0
ubuntu@ip-172-31-13-89:~$
```

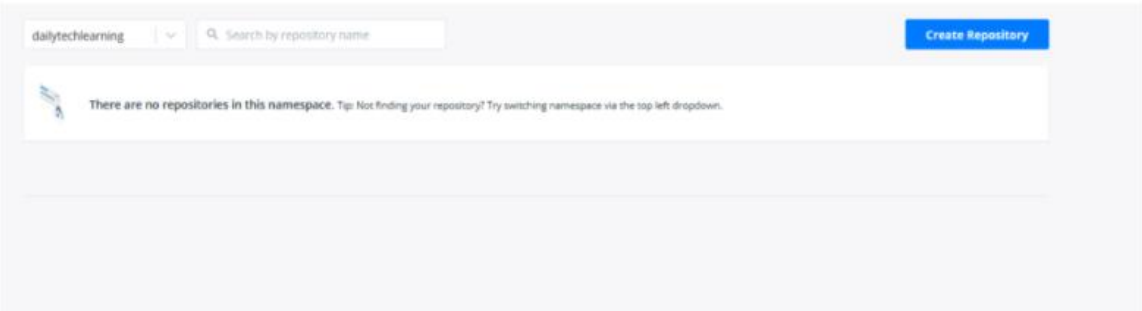
6. Installation of tools

- Check whether git, docker, docker-compose is installed. If not install the same with respective command lines
- For docker installation refer [Install Docker Engine on Ubuntu | Docker Documentation]
- For docker compose installation refer [How to Install Docker Compose on Ubuntu 20.04 {Step-by-Step Guide} (phoenixnap.com)]

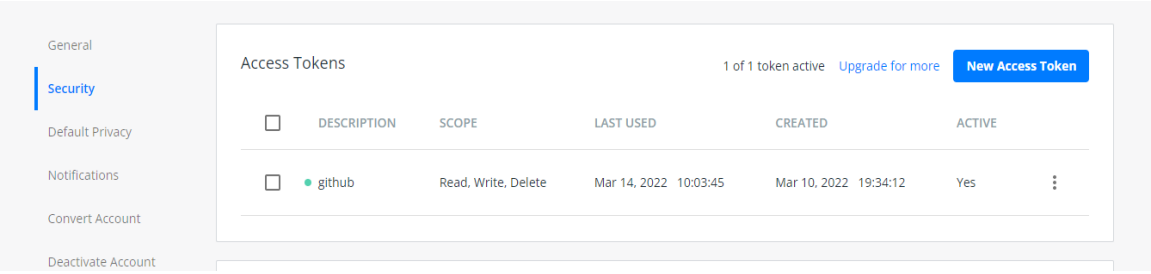
7. Create Docker Hub account



- Select the desired plan and then can create repository where you need to publish images

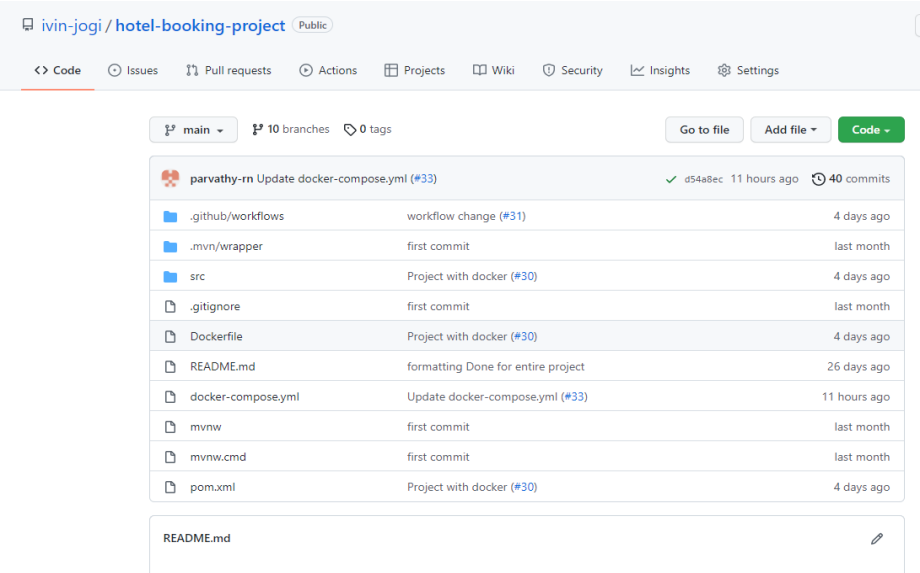


- Generate an access token in security in account setting of user profile



8. Update Git Repository

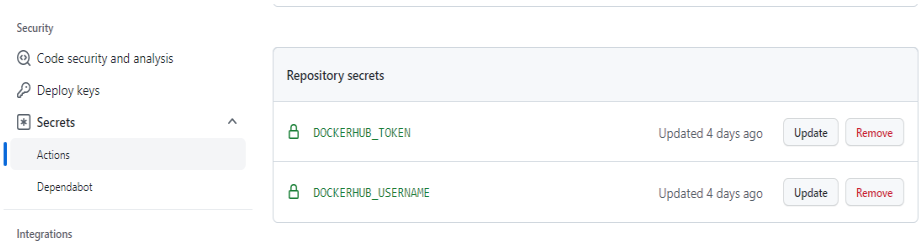
- Make your corresponding git repository updated by raising required pull request for every change and merge with master branch



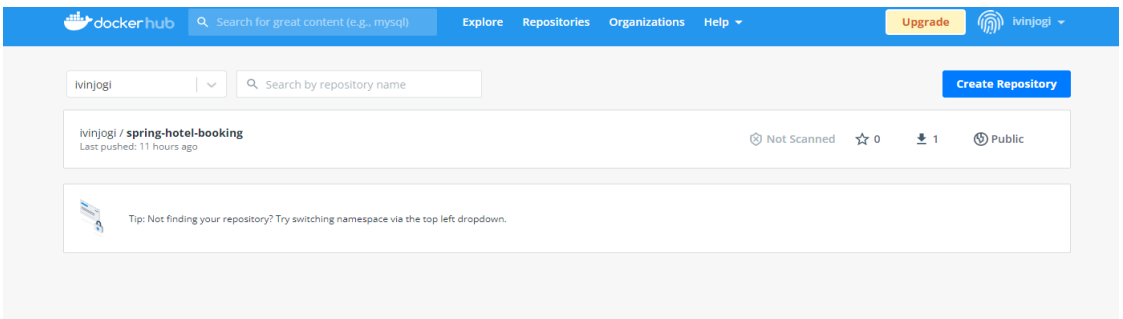
- Add the access token generated in docker hub along with username in the workflow of your project code

```yaml
    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 8
      uses: actions/setup-java@v2
      with:
        java-version: '8'
        distribution: 'adopt'
        cache: maven
    - name: Build with Maven
      run: mvn -B package -DskipTests --file pom.xml
    - name: Build the Docker image
      run: docker build . --file Dockerfile --tag $IMAGE_NAME

    - name: Login to DockerHub
      uses: docker/login-action@v1
      with:
        username: ${{ secrets.DOCKERHUB_USERNAME }}
        password: ${{ secrets.DOCKERHUB_TOKEN }}

    - name: Push image to GitHub Container Registry
      run: |
        IMAGE_ID=${{ secrets.DOCKERHUB_USERNAME }}/$IMAGE_NAME
        # Change all uppercase to lowercase
        IMAGE_ID=$(echo $IMAGE_ID | tr '[A-Z]' '[a-z]')
        # Strip git ref prefix from version
        VERSION=$(echo "${{ github.ref }}" | sed -e 's,.*/\(.*\),\1,')
        # Strip "v" prefix from tag name
        [[ "${{ github.ref }}" == "refs/tags/"* ]] && VERSION=$(echo $VERSION | sed -e 's/^v//')
        # Use Docker `latest` tag convention
        [ "$VERSION" == "master" ] && VERSION=latest
        echo IMAGE_ID=$IMAGE_ID
        echo VERSION=$VERSION
        docker tag $IMAGE_NAME $IMAGE_ID:$VERSION
        docker push $IMAGE_ID:$VERSION
```

- Define Image name as the name of our docker image
- Also save those details in secrets in repository settings in git hub



- Raise PR for the changes made
- This is done so that any pulls made in git hub would reflect in docker hub
- Thus, a new repository for our application would be generated in docker hub



9. Clone the project

- Clone the project into ubuntu terminal from git using git clone command
- After fetching the whole project get into the project folder (note: you can check it by using Linux command: ls)
- Make sure you get into right folder, so that you can execute commands successfully[check it by typing 'pwd']
- If any changes to be made further, it could be done in git and raise pr for the same and pull the changes from the terminal using git pull command

```
ubuntu@ip-172-31-13-89:~$ cd hotel-booking-project/
ubuntu@ip-172-31-13-89:~/hotel-booking-project$ git pull
remote: Enumerating objects: 65, done.
remote: Counting objects: 100% (65/65), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 46 (delta 18), reused 21 (delta 9), pack-reused 0
Unpacking objects: 100% (46/46), 7.72 KiB | 790.00 KiB/s, done.
From https://github.com/ivin-jogi/hotel-booking-project
   8923fcb..11c7821  main             -> origin/main
 * [new branch]      ivin-jogi-patch-3 -> origin/ivin-jogi-patch-3
 * [new branch]      newcode           -> origin/newcode
Updating 8923fcb..11c7821
Fast-forward
 .github/workflows/maven.yml                                          | 33 ++++++++++++++++++++++++++++++++---
 Dockerfile                                                           | 11 +++++++++++
 docker-compose.yml                                                   | 24 ++++++++++++++++++++++
 pom.xml                                                              |  1 +
 src/main/java/com/ibs/litmusproject/hotelbooking/service/SearchDetailsService.java | 24 ++++++++++++++---------
 src/main/resources/application.properties                            |  6 +++---
 src/test/resources/applicationtest.properties                        |  2 +-
 7 files changed, 84 insertions(+), 17 deletions(-)
 create mode 100644 Dockerfile
 create mode 100644 docker-compose.yml
ubuntu@ip-172-31-13-89:~/hotel-booking-project$
```

- Make sure the image name in docker-compose.yml file is changed to repository image name in docker hub

10. Pull docker hub image

- Pull the docker hub image using the command
  docker pull image_name

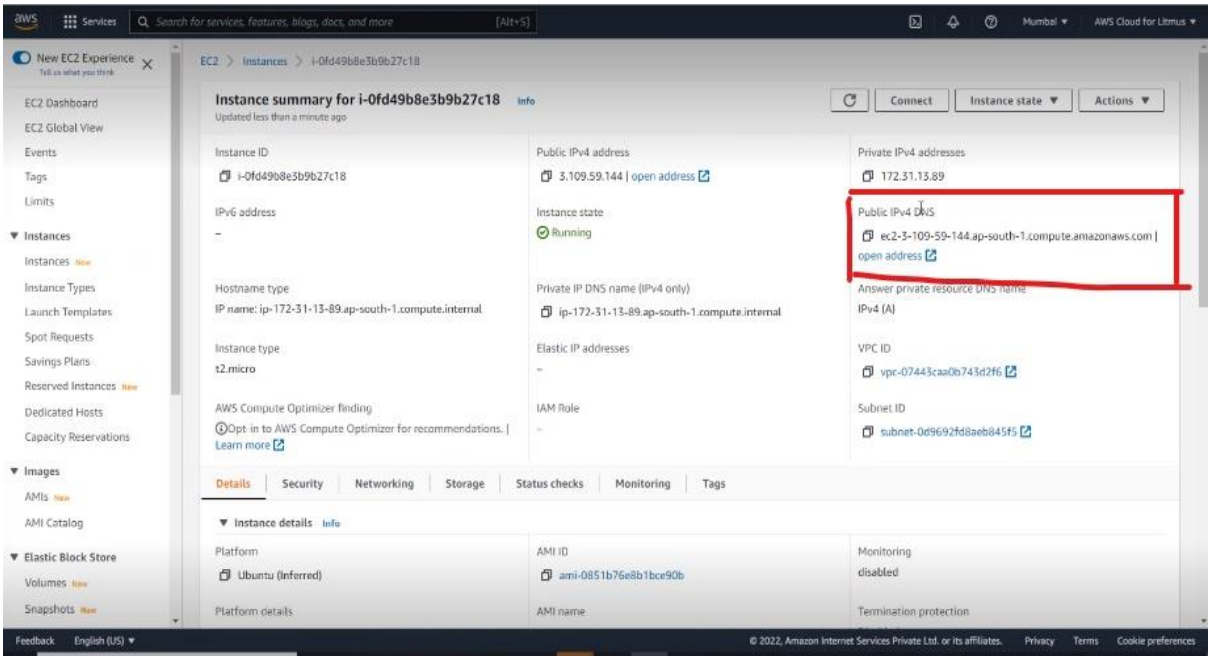
```
ubuntu@ip-172-31-13-89:~/hotel-booking-project$ docker pull ivinjogi/spring-hotel-booking:main
main: Pulling from ivinjogi/spring-hotel-booking
e4d61adff207: Already exists
4ff1945c672b: Already exists
ff5b10aec998: Already exists
12de8c754e45: Already exists
4848edf44506: Already exists
612ca5886be9: Already exists
3c8418aa597a: Already exists
83c81bef9540: Downloading [==============================>                    ]  24.5MB/37.95MB
```

11. Run Application

- Run the application using docker compose up command

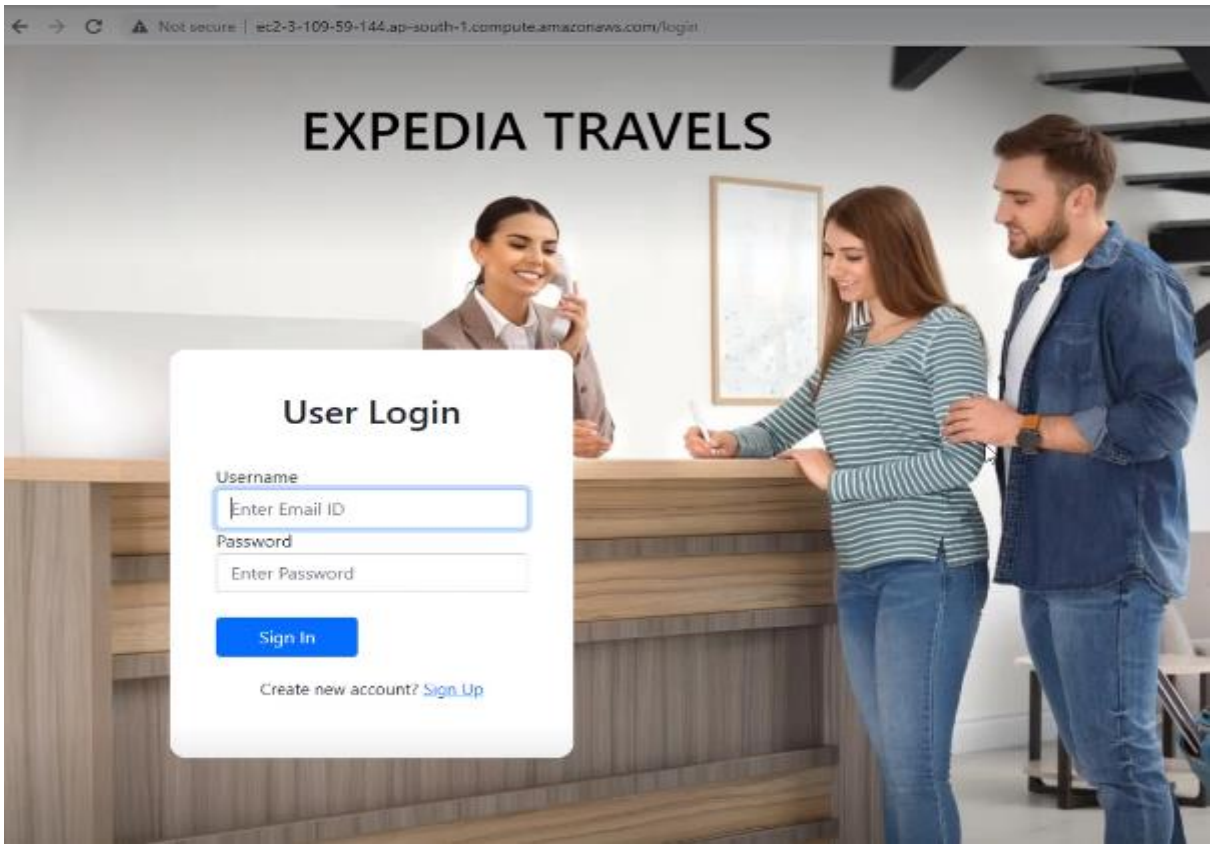
```
ubuntu@ip-172-31-13-89:~/hotel-booking-project$ docker compose up
[+] Running 3/3
 ⠿ Network hotel-booking-project_default  Created
 ⠿ Container db                           Created
 ⠿ Container hotelbookingapp              Created
Attaching to db, hotelbookingapp
db              | The files belonging to this database system will be owned by user "postgres".
db              | This user must also own the server process.
db              |
db              | The database cluster will be initialized with locale "en_US.utf8".
db              | The default database encoding has accordingly been set to "UTF8".
db              | The default text search configuration will be set to "english".
db              |
db              | Data page checksums are disabled.
db              |
db              | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db              | creating subdirectories ... ok
db              | selecting dynamic shared memory implementation ... posix
db              | selecting default max_connections ... 100
db              | selecting default shared_buffers ... 128MB
db              | selecting default time zone ... UTC
db              | creating configuration files ... ok
db              | running bootstrap script ... ok
db              | performing post-bootstrap initialization ... sh: locale: not found
db              | 2022-03-10 14:48:53.836 UTC [30] WARNING:  no usable system locales were found
db              | ok
```

- To stop running to make changes use docker compose down if needed [to make changes directly type command, vi file_name. Then hit 'i' to insert changes. Once changes made, type, :qw to save and exit]
- After successful run, in the instance that is running, you will find a public DNS link



12. Hit the url in browser

- Copy the link and browse it using a browser
- Expected outcome is the successful deployment of your application
- You can access your application using that link as long as the instance is running



..... THANK YOU .....