AWS Project : CI/CD on EC2 using github actions

Documentation

By : Abouchiba Mohamed Yassir

To see the files used in this project go to this link:
https://github.com/M-Yassir/AWS-Projects/upload/main/CICD_on_EC2

Introduction

Project Description

This project demonstrates the implementation of a CI/CD pipeline for deploying applications on an Amazon EC2 instance using GitHub Actions. The pipeline automates building, testing, and deploying code changes, ensuring rapid and reliable software delivery. By integrating GitHub Actions with AWS EC2, it highlights how DevOps practices can streamline workflows, improve deployment efficiency, and maintain high-quality standards.

Key Features

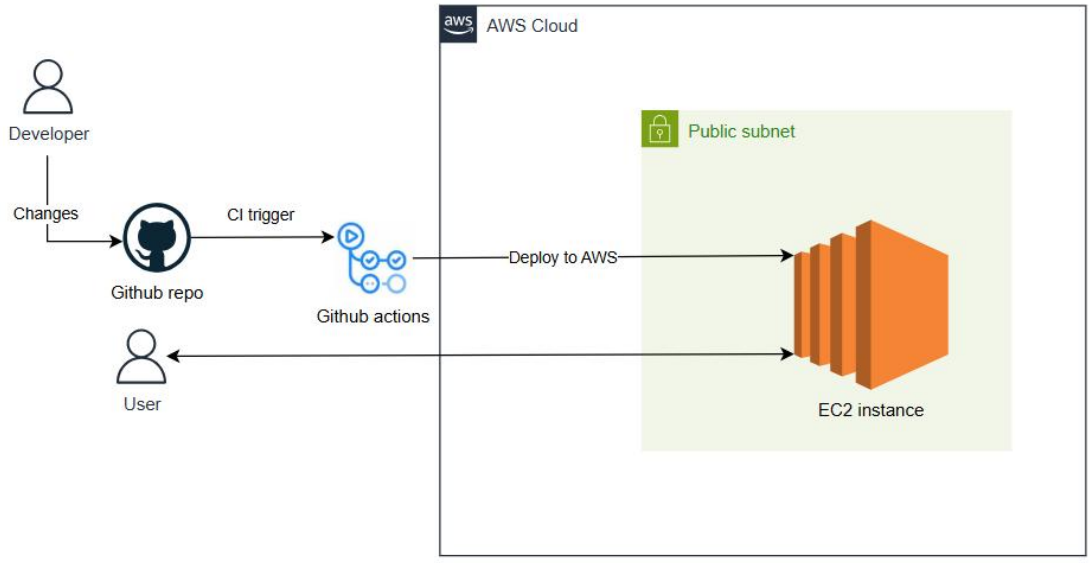
-Automated Builds and Testing: GitHub Actions triggers builds and tests on code pushes, ensuring reliability.

-Seamless Deployment: Automatically deploys to EC2 after successful builds, reducing manual errors.

-Version Control Integration: Manages code changes and deployments directly from GitHub.

-Cost-Effective Solution: Utilizes free or low-cost tools, ideal for small to medium projects.

AWS architecture diagram

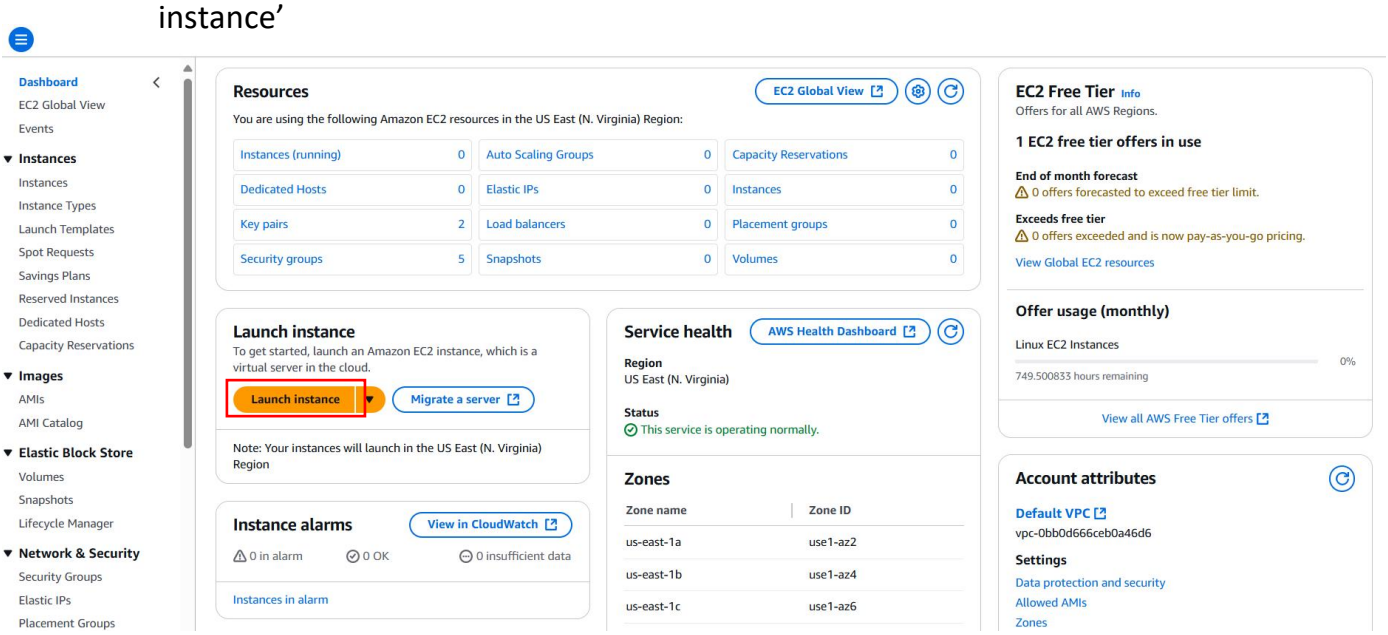


Step-by-Step implementation

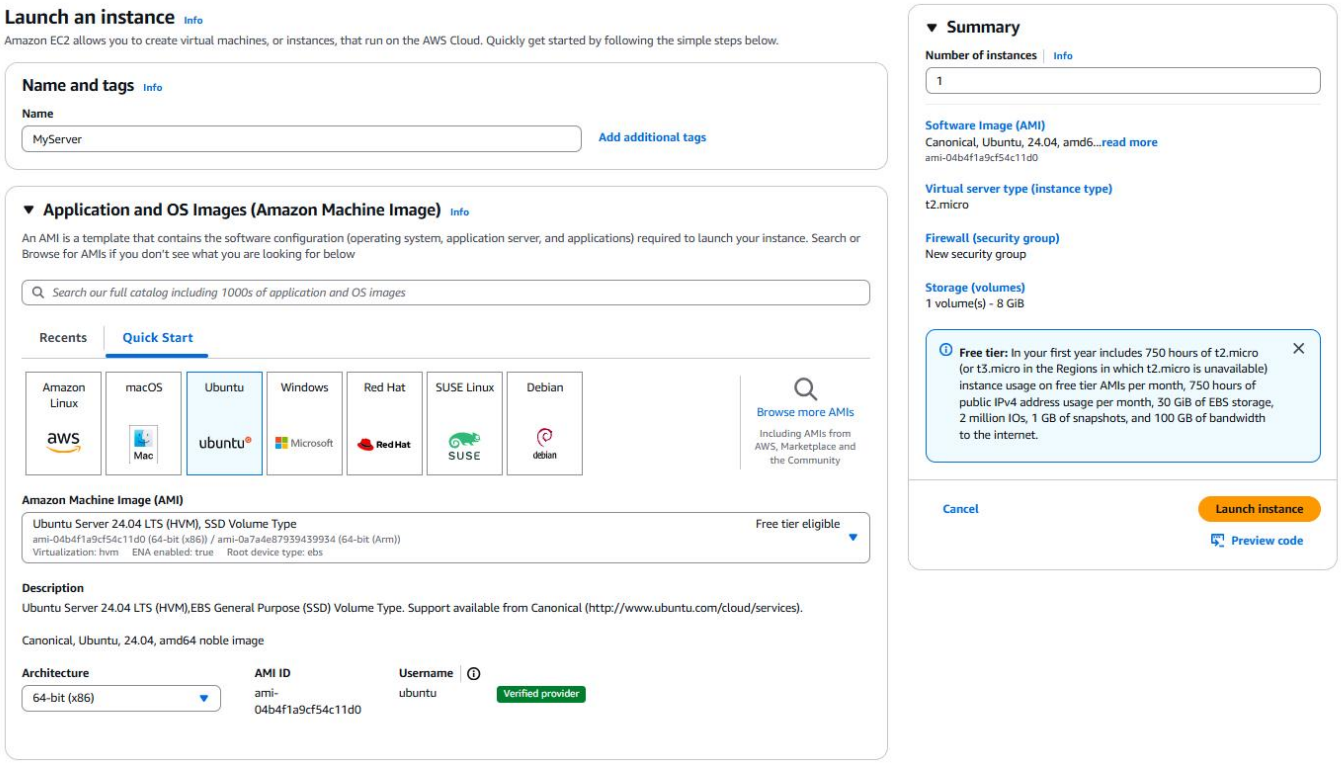
NB: - All the necessary code is on the repository given above.
- The creation of the index.html file isn't covered here as it isn't our topic.

Step1: Creating a EC2 instance:

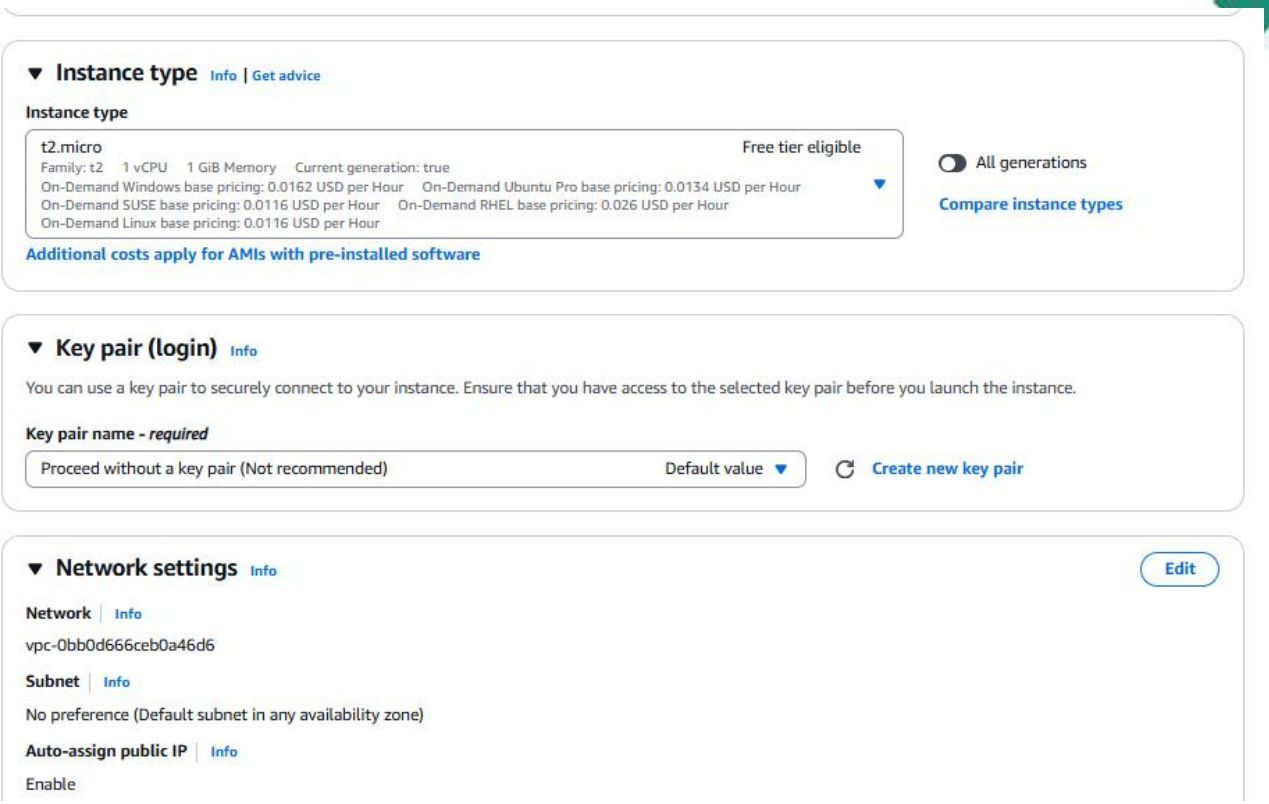
First, we go to the EC2 service for creating an instance and we click on 'launch instance'



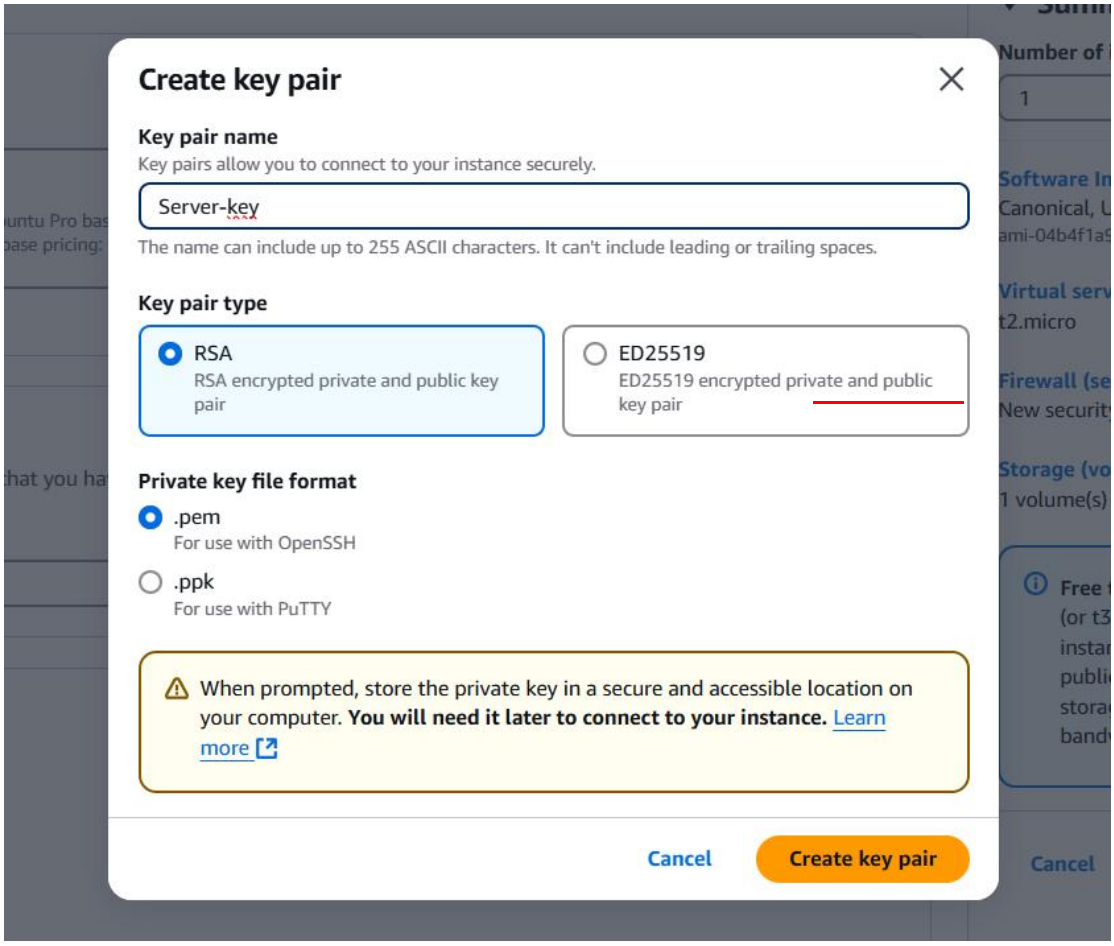
Then, we fill in the necessary informations like it's shown below



After scrolling down a little bit, an option for creating a new key pair will appear, we click on it:

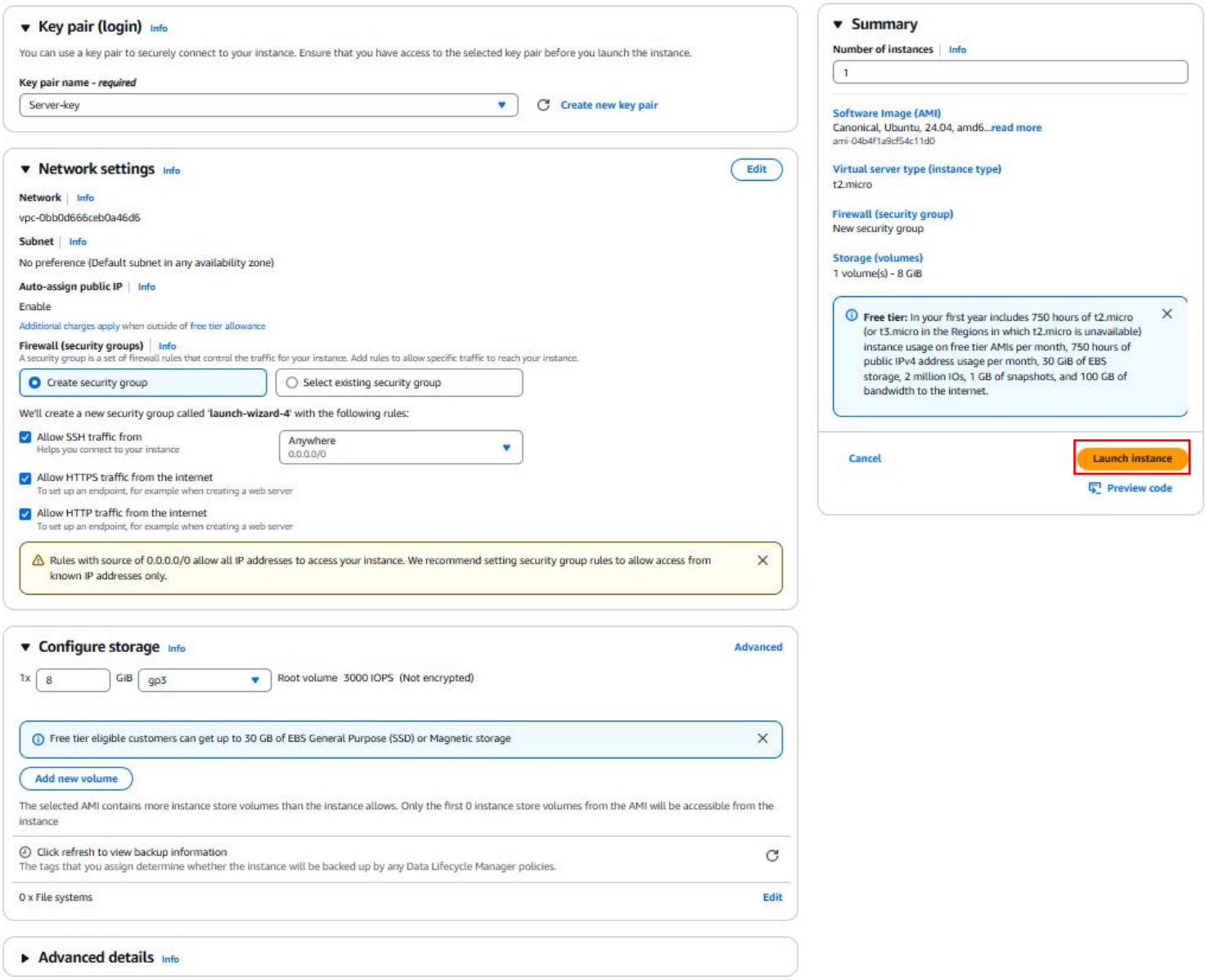


And we fill in the following informations:



After clicking on ‘create key pair’, a PEM file should be downloaded, we’ll need it later.

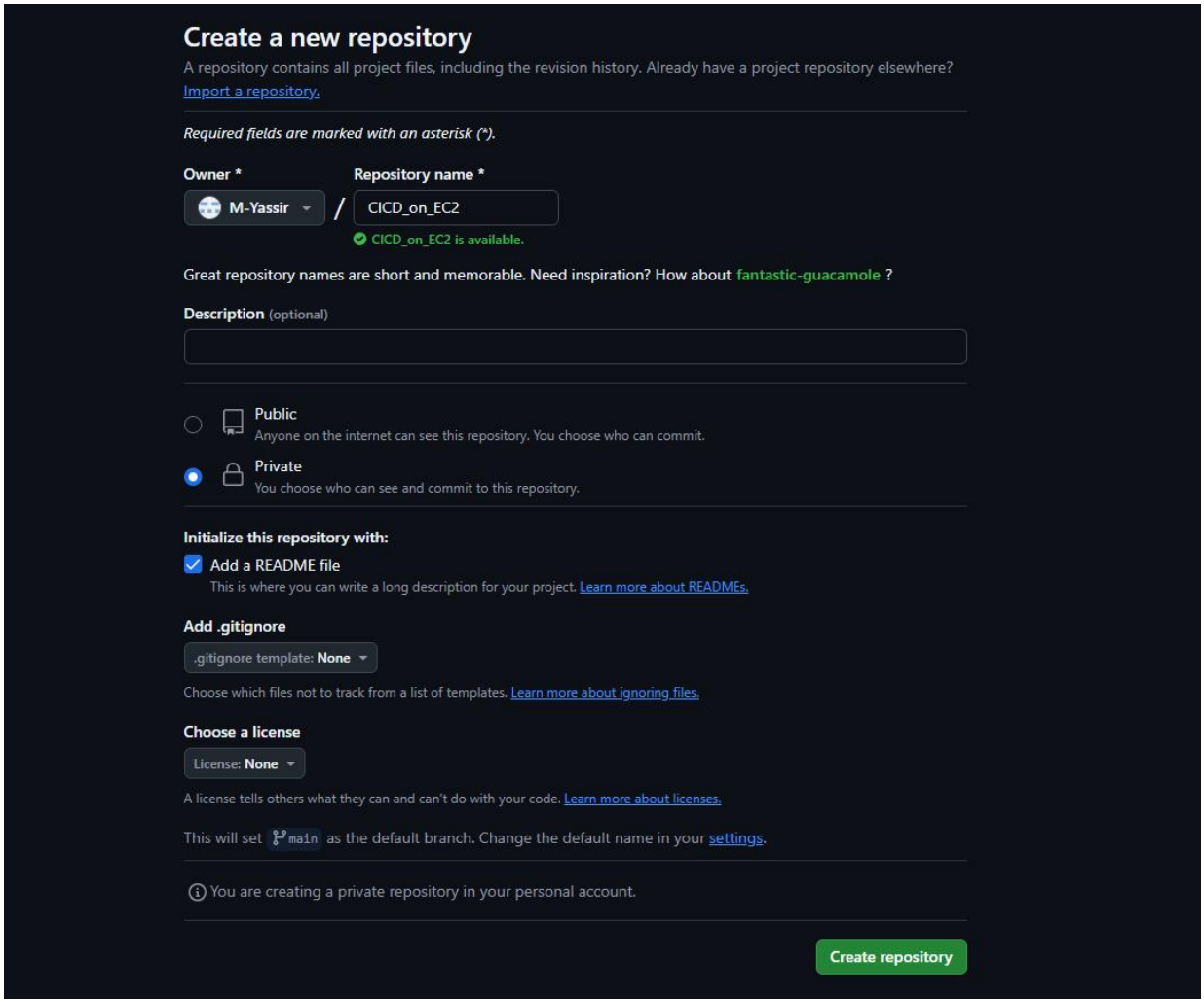
The final part should look like this:



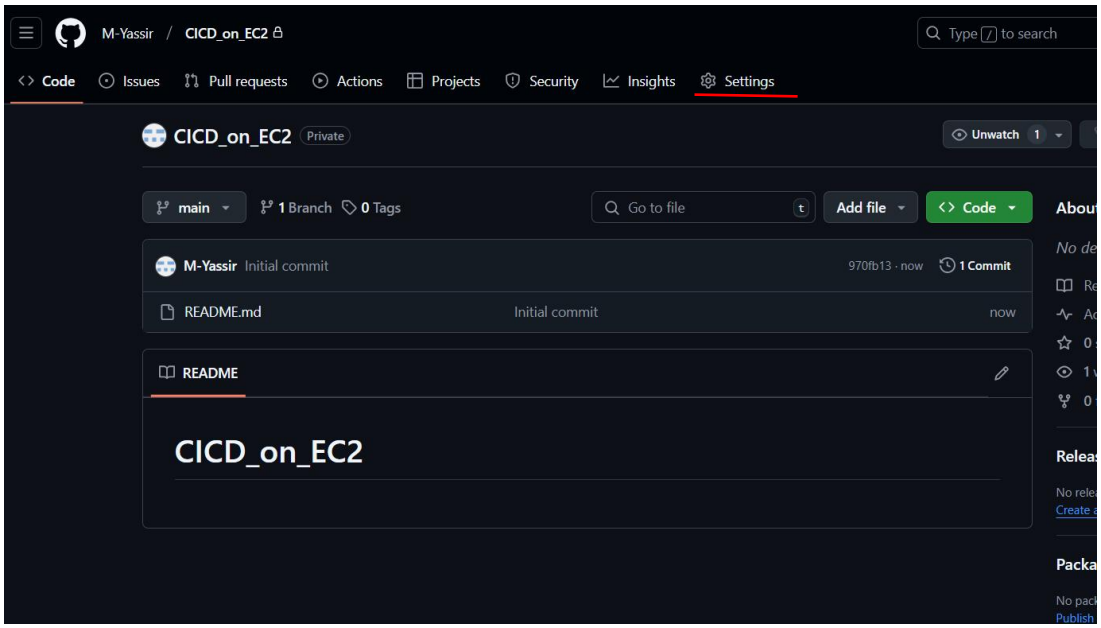
We click on ‘Launch instance’.
A message should appear indicating that the operation was successful.

Step2: Create a github repository and integrate the CICD pipeline:

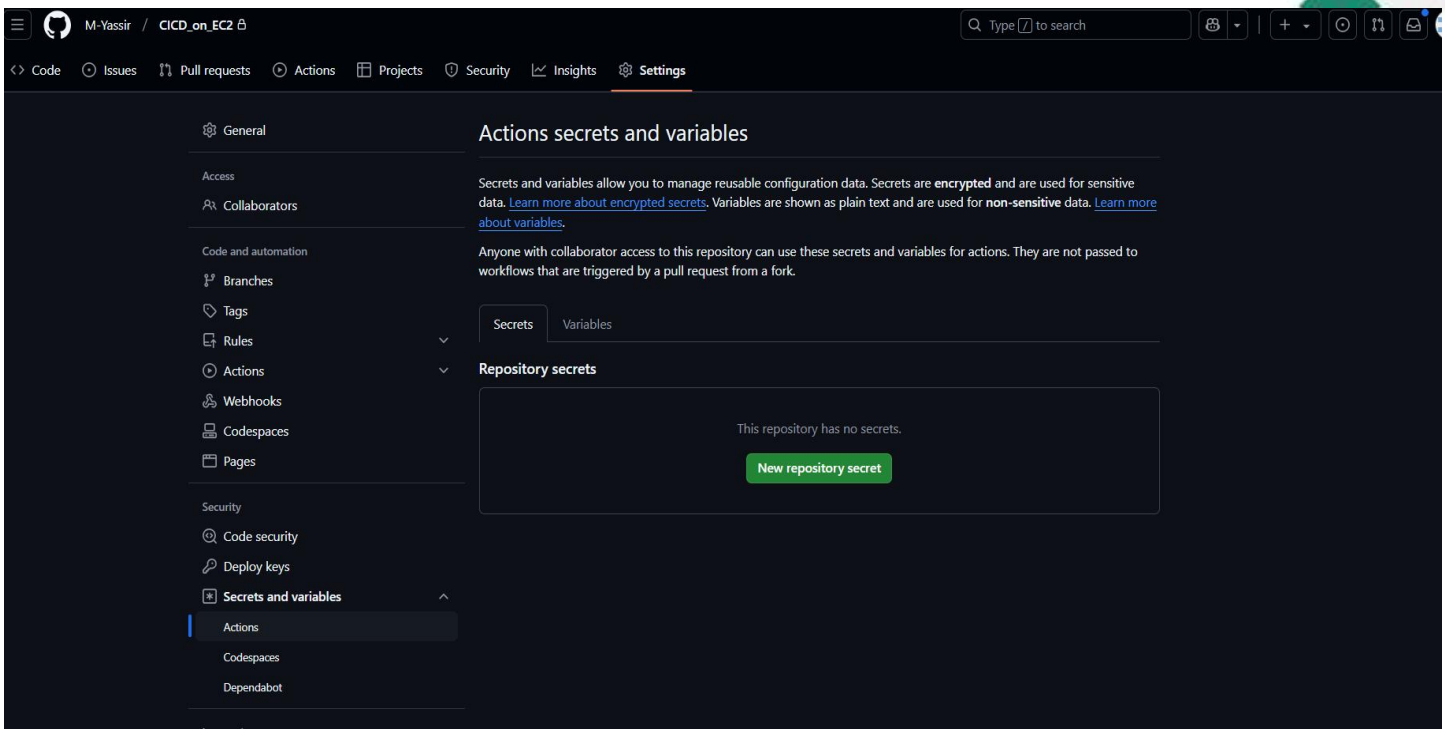
We create a new repository on github named ‘CICD_on_EC2’



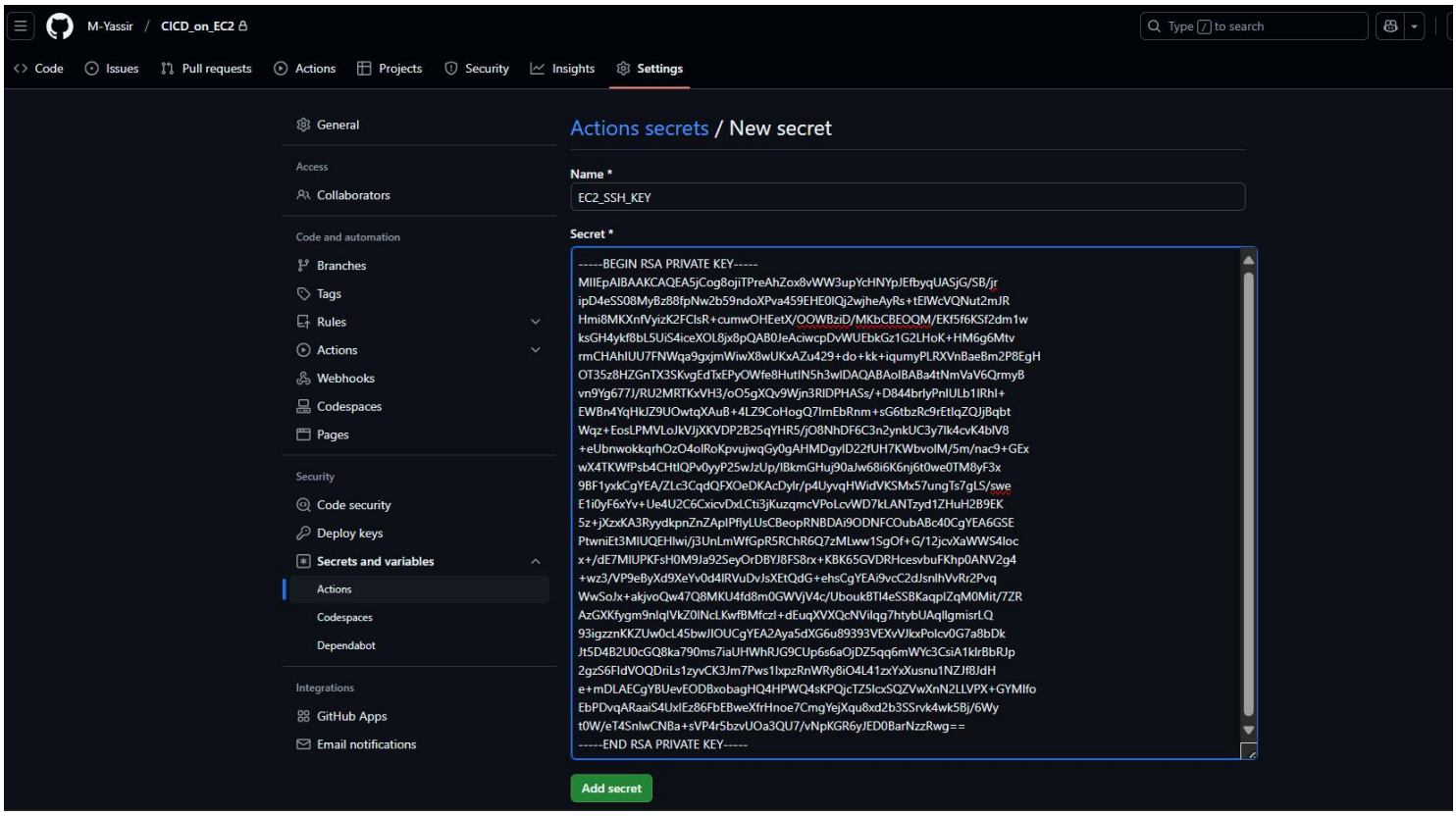
After clicking on ‘create repository’ we should get the following



We go to the settings tab on **Actions** in **Secrets and variables**, like it's shown below:

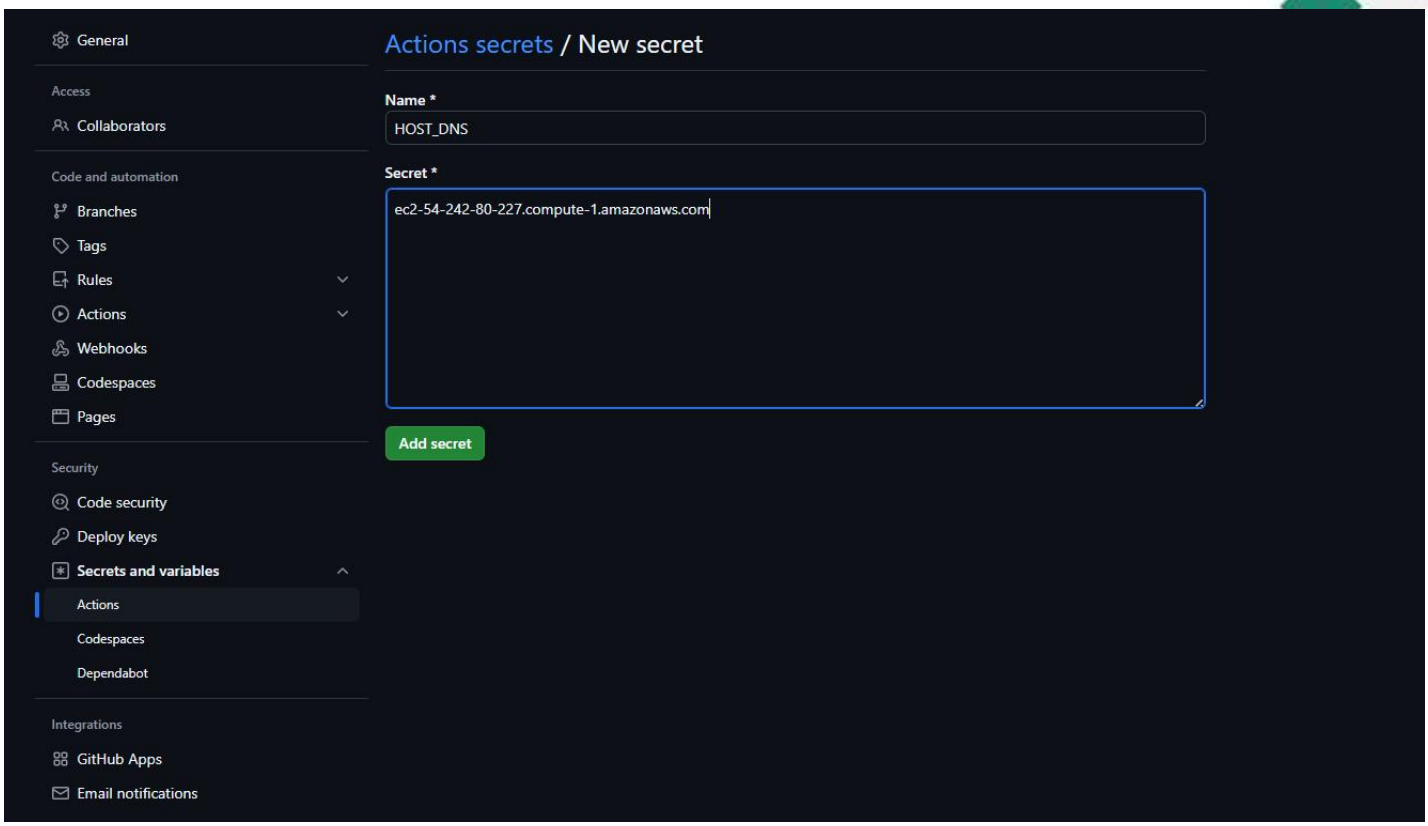


We click on 'New repository secret', we name it "EC2_SSH_KEY", we should open our previously downloaded PEM file, copy **all** of its content and paste it in 'secret*' as it's shown below:

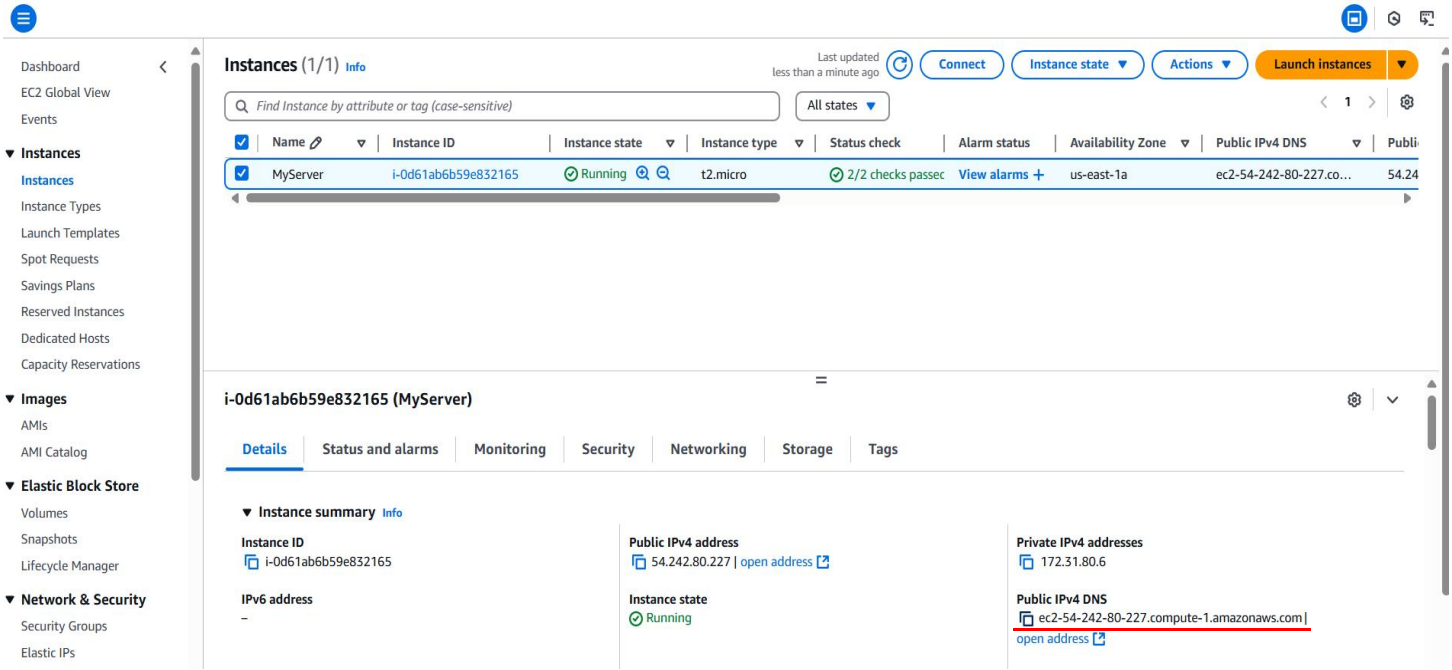


We click on 'Add secret'

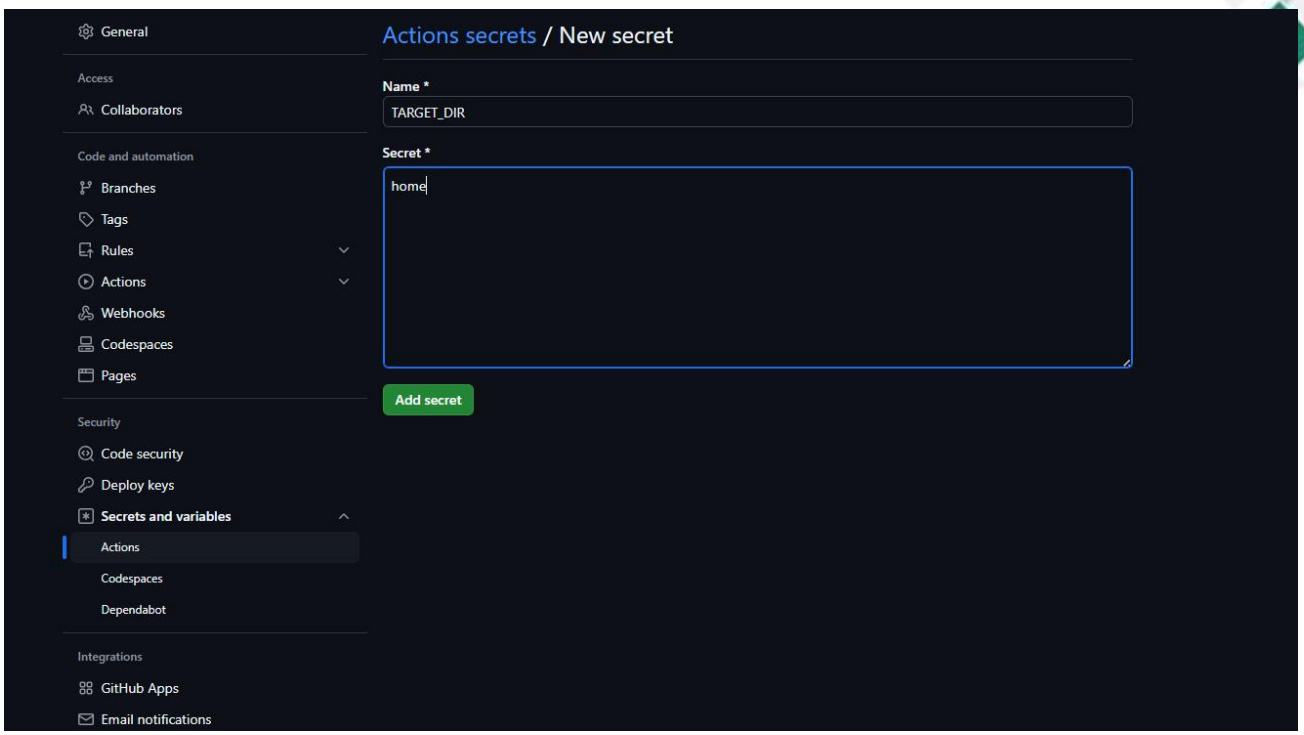
The same steps should be made when creating the other secrets, for the host DNS we should get this



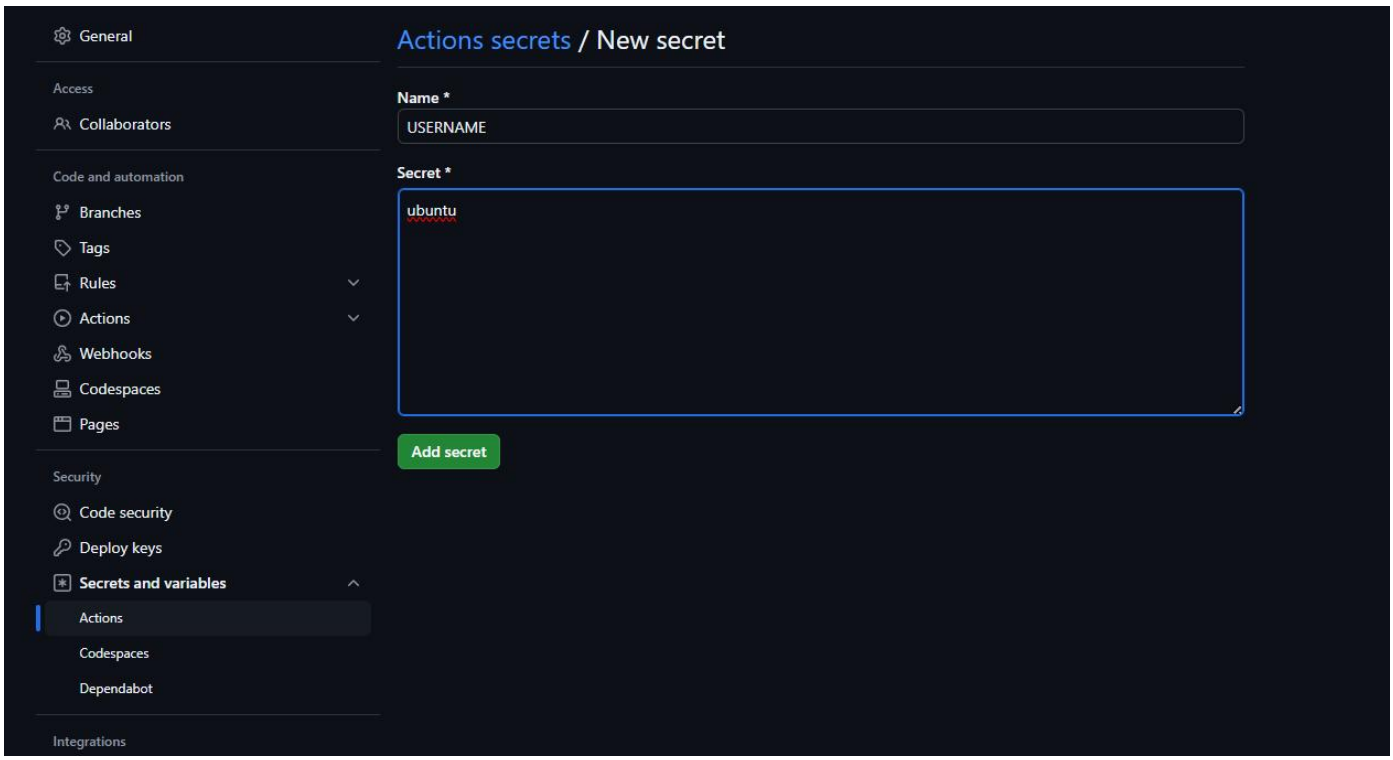
The 'Secret*' (Public IPv4 DNS) value should be got on the **Details** when selecting the current EC2 instance in the **instances** tab like it is shown here



for target directory we should do this:

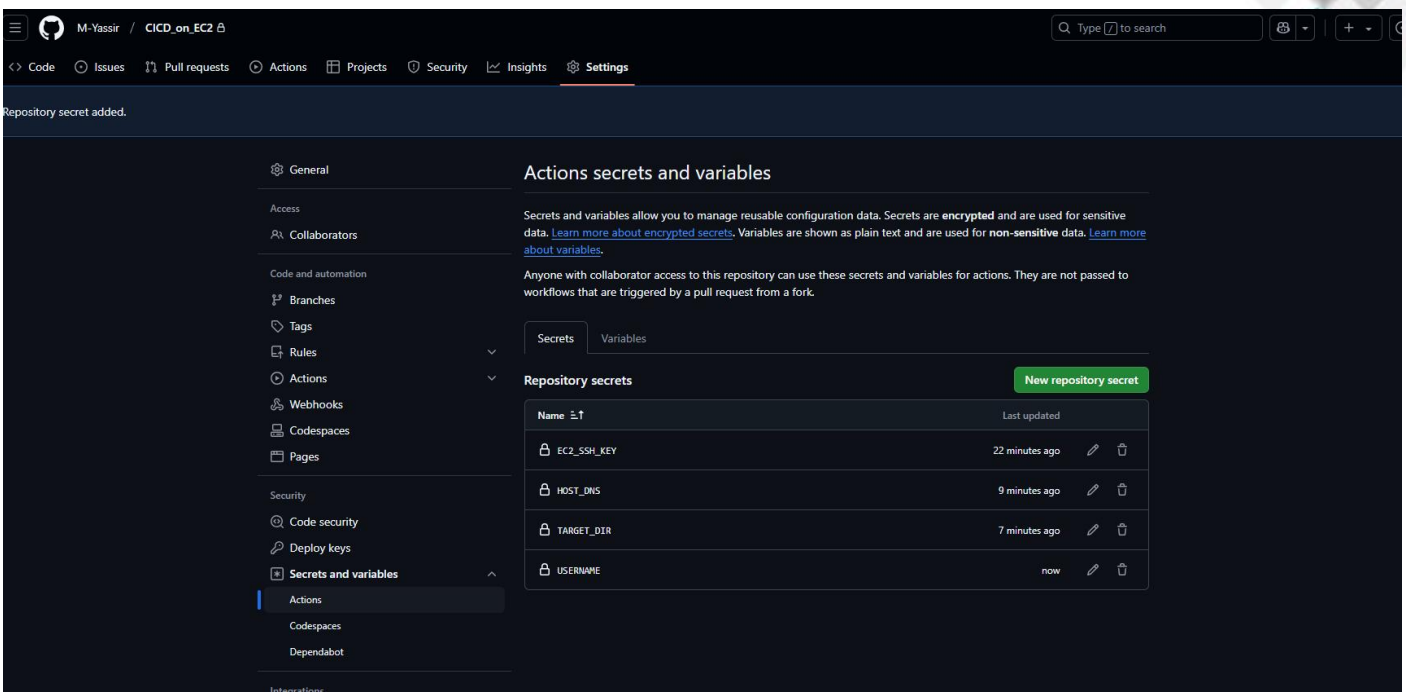


and for username as the default name of an ubuntu EC2 instance is 'ubuntu' we should do this:

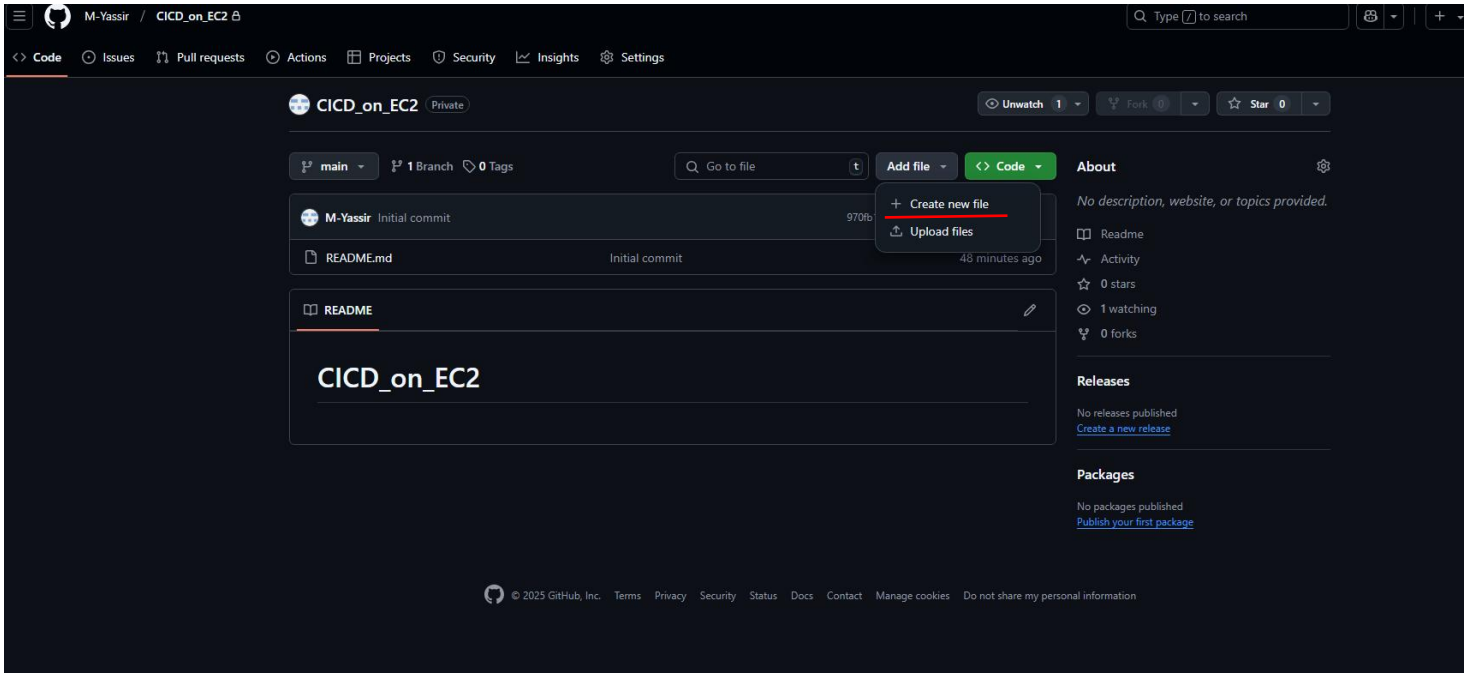


NB: In order to avoid unexpected errors, all the fields should be the same as above

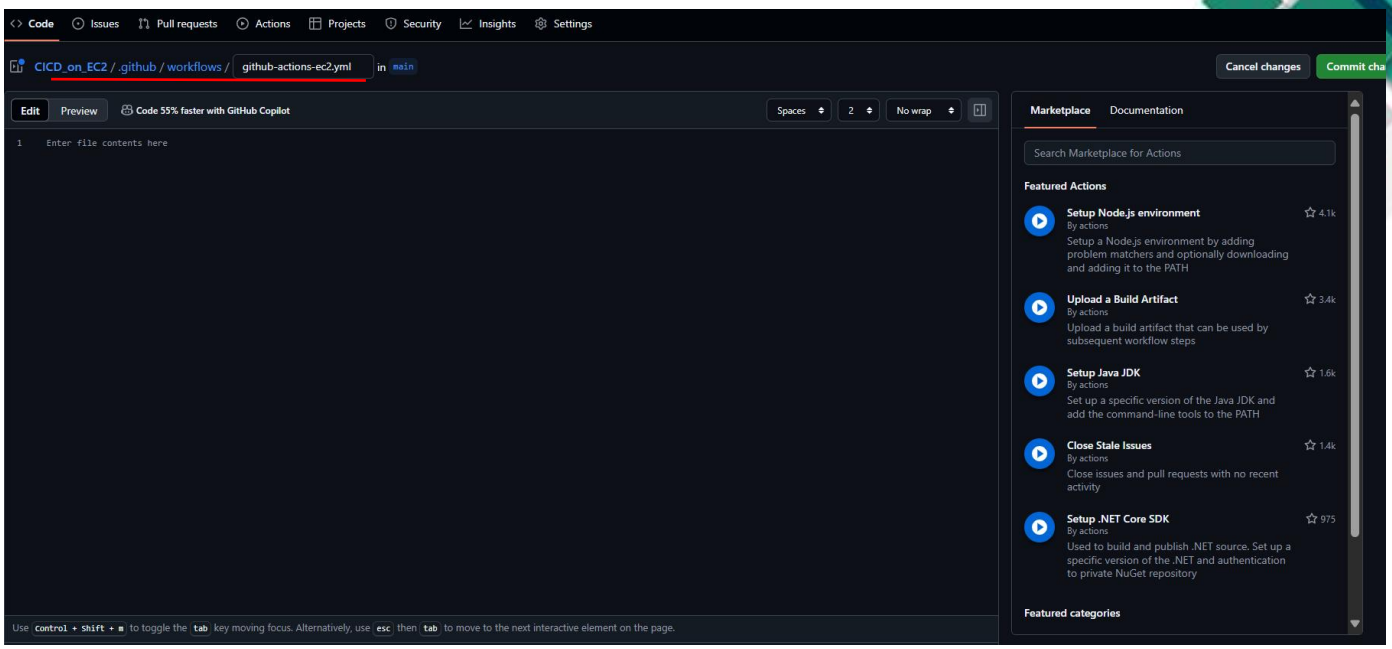
at the end, we should get something like this



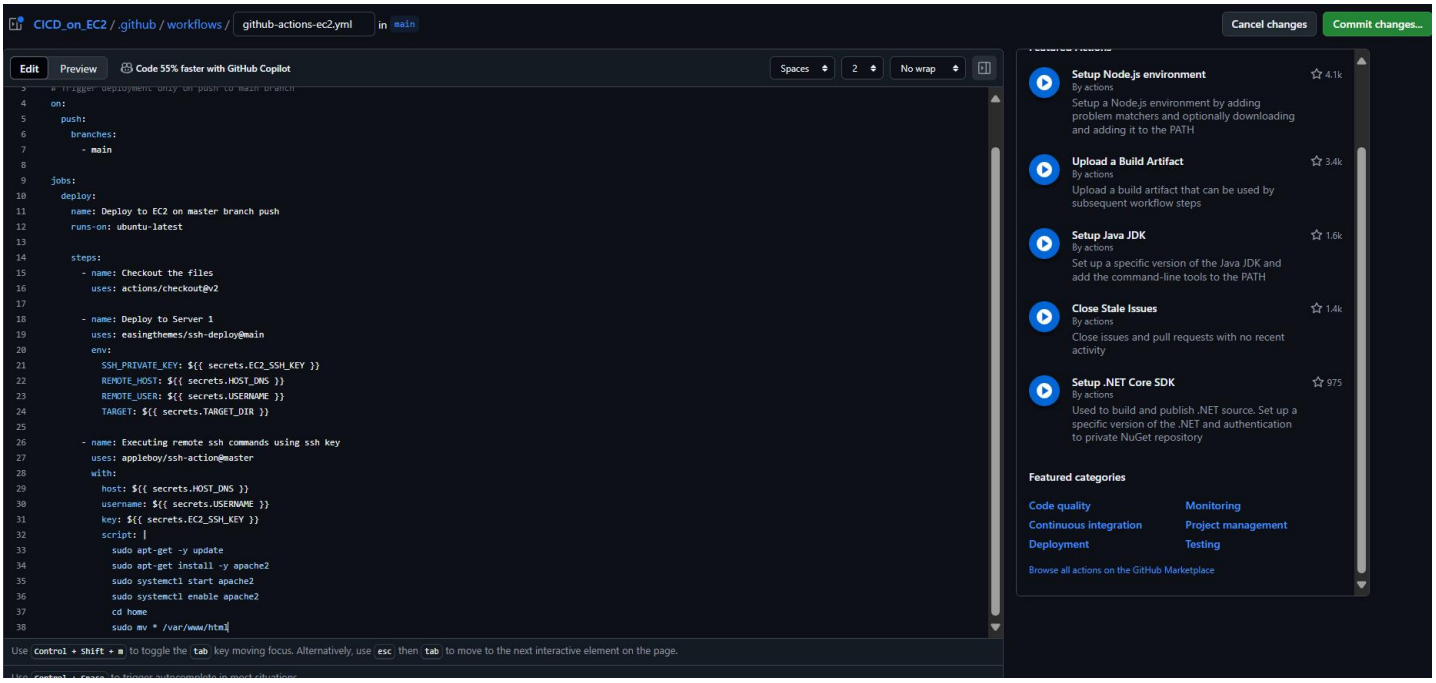
Then, we go to the Code tab, click on ‘Add file’ and ‘Create new file’



Before adding file contents, we should add the following name “.github/workflows/github-actions-ec2.yml” to the title “CICD_on_EC2”

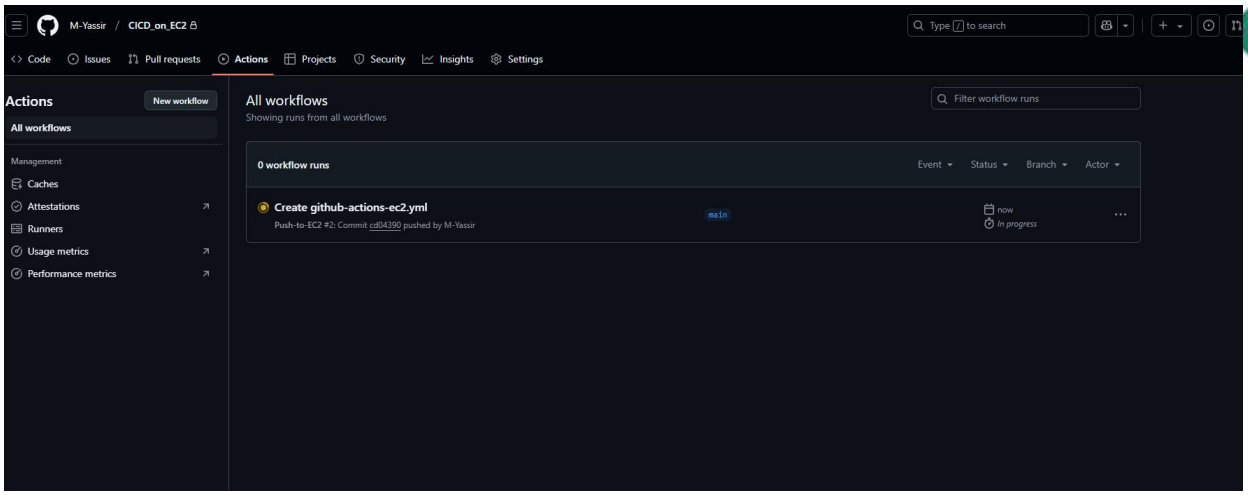


Then, we should type the following code in it (the code is on the repository)

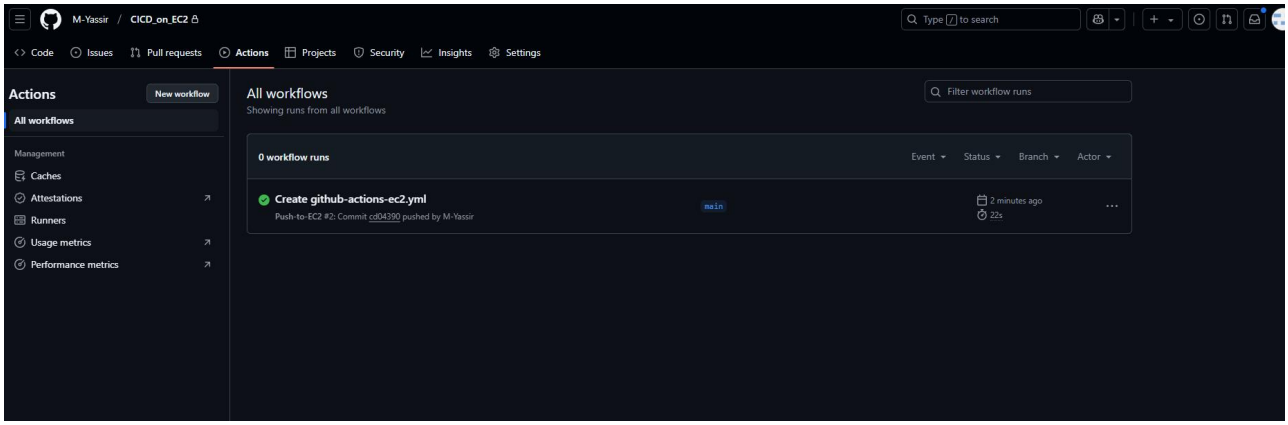


We click on “Commit changes”

We go to the ‘Actions’ tab, we should see this

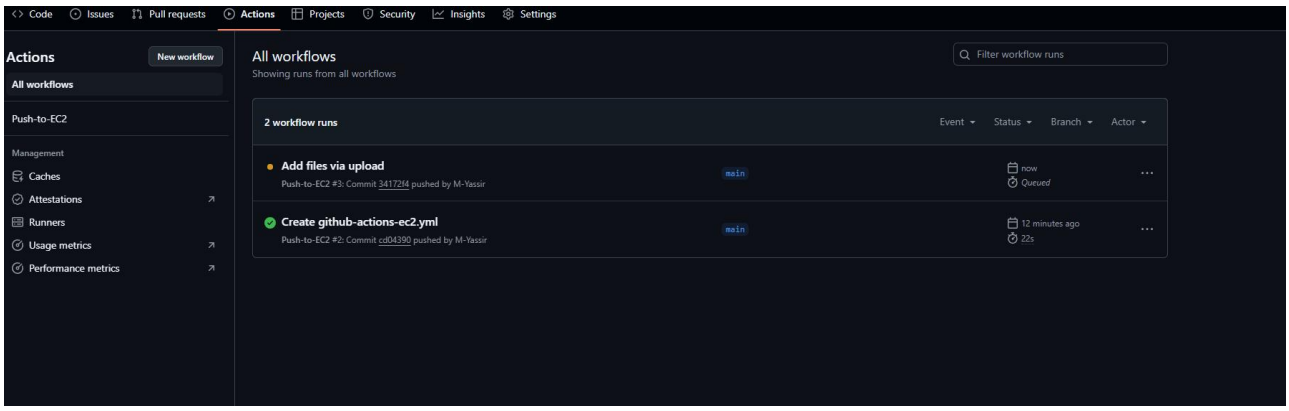


Later, it should be like this

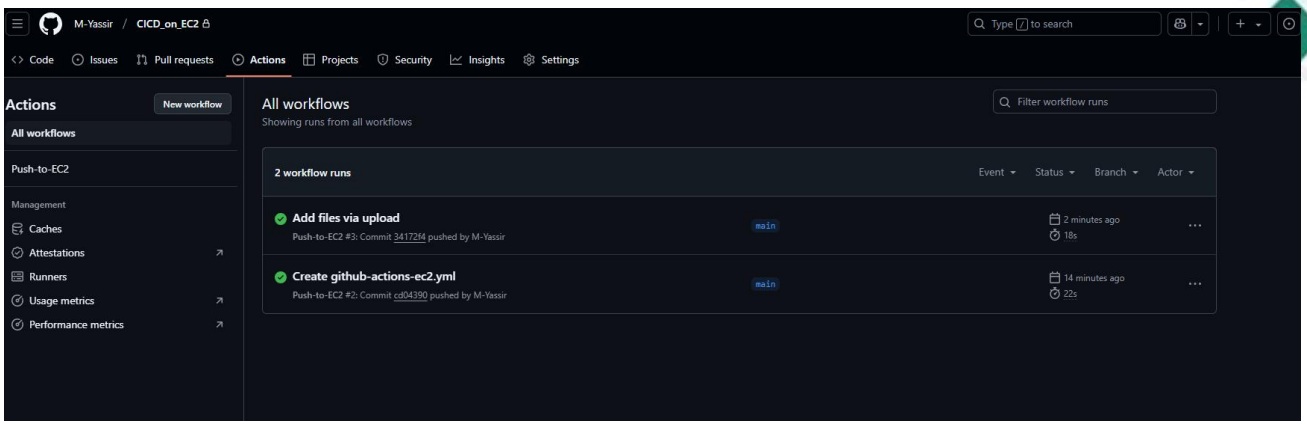


meaning that the operation was successful.

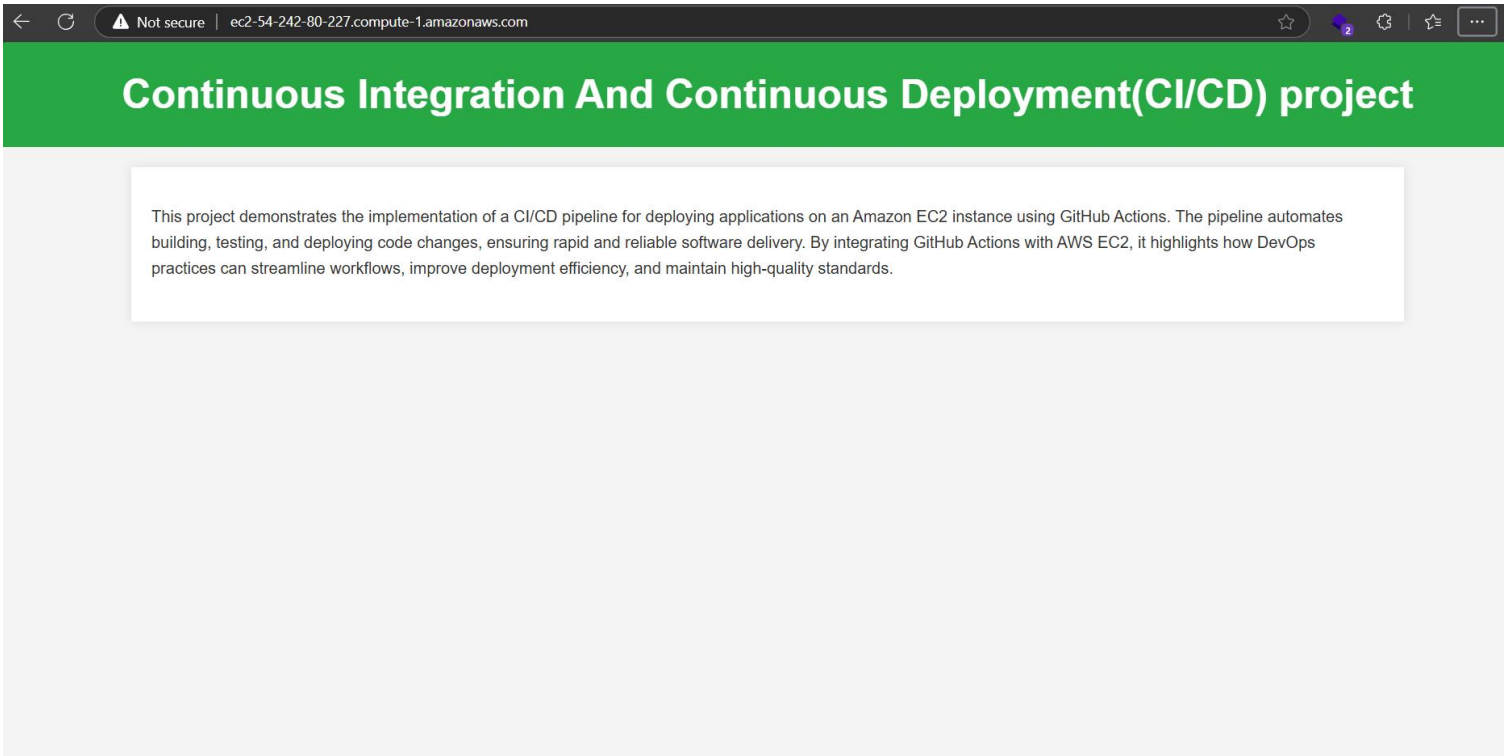
Let’s try it out, we’re going to add an index.html file to this repo (the code is on the repository), we should get this on the actions tab



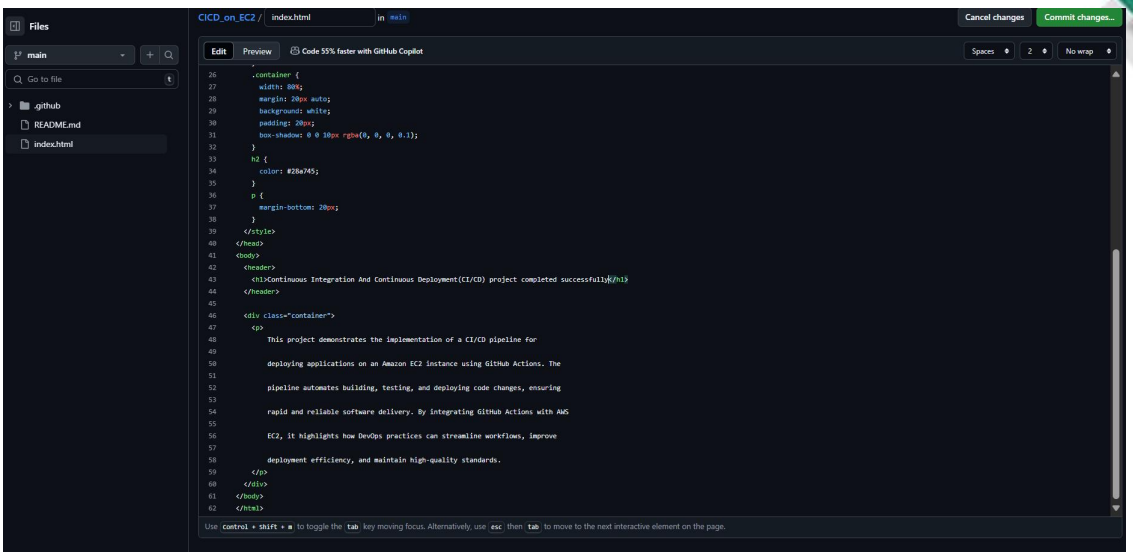
Later we should get this



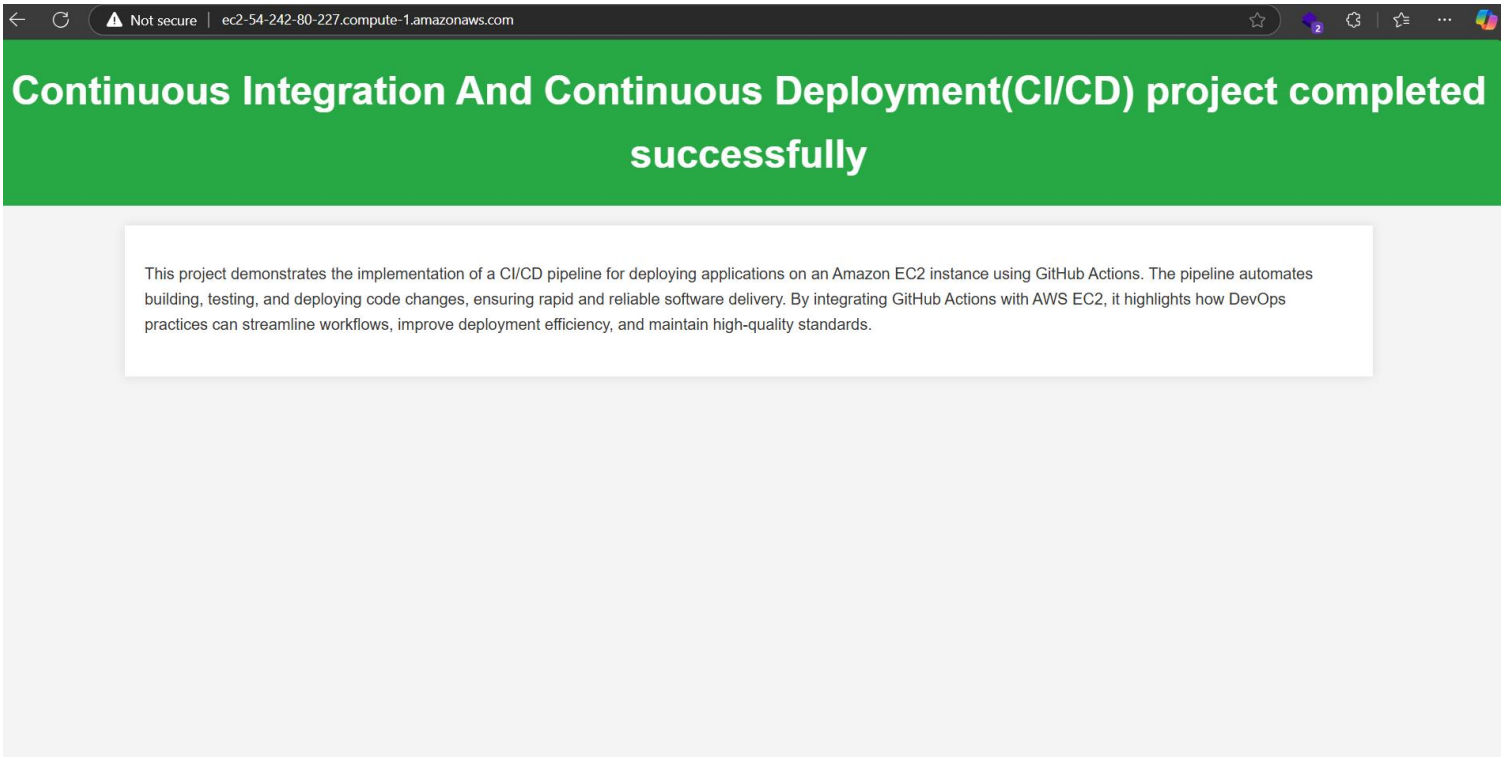
Let's go to the **Public IPv4 DNS** of the instance



and it works.
Let's go now to modify the header of the index.html file on github (add "completed successfully" to the header)



By commiting changes and waiting till it finishes the update and revisit the public ipv4 DNS of that instance, we should get this



and it also works.

Conclusion

This project has been an insightful journey into building a Continuous Integration and Continuous Deployment (CI/CD) pipeline for an EC2-based web application using GitHub Actions. By leveraging GitHub Actions for automation and deploying directly to an EC2 instance, I successfully implemented an efficient and reliable deployment process for managing application updates.

Challenges Faced & Lessons Learned:

1) GitHub Actions Workflow Configuration

One of the major lessons was configuring the GitHub Actions workflow to ensure seamless deployment.
I learned how to structure multi-step workflows, manage secrets securely, and ensure proper communication with the EC2 instance.

2)Monitoring Deployment Success

Initially, verifying whether deployments were successful on the EC2 instance was not straightforward.
I learned how to incorporate logging and verification steps to ensure that the latest code was properly deployed and accessible.

3)SSH Key Authentication & Security

Managing secure connections to the EC2 instance via SSH required proper handling of keys and permissions.
I learned how to store and use SSH keys safely in GitHub Secrets and automate secure deployments without exposing sensitive information.

Final Takeaways:

Through this project, I gained hands-on experience in CI/CD pipeline development and learned how GitHub Actions and EC2 work together to create a streamlined deployment process. Some key skills I developed include:
✅ Automating deployments to EC2 using GitHub Actions
✅ Configuring multi-step workflows with secure secret management
✅ Verifying deployment success through automated logging
✅ Managing SSH-based deployments for cloud environments

This project has deepened my understanding of CI/CD pipelines for cloud-based environments and reinforced the importance of automation and continuous delivery. Moving forward, I am excited to apply these skills to build even more robust and scalable deployment pipelines.