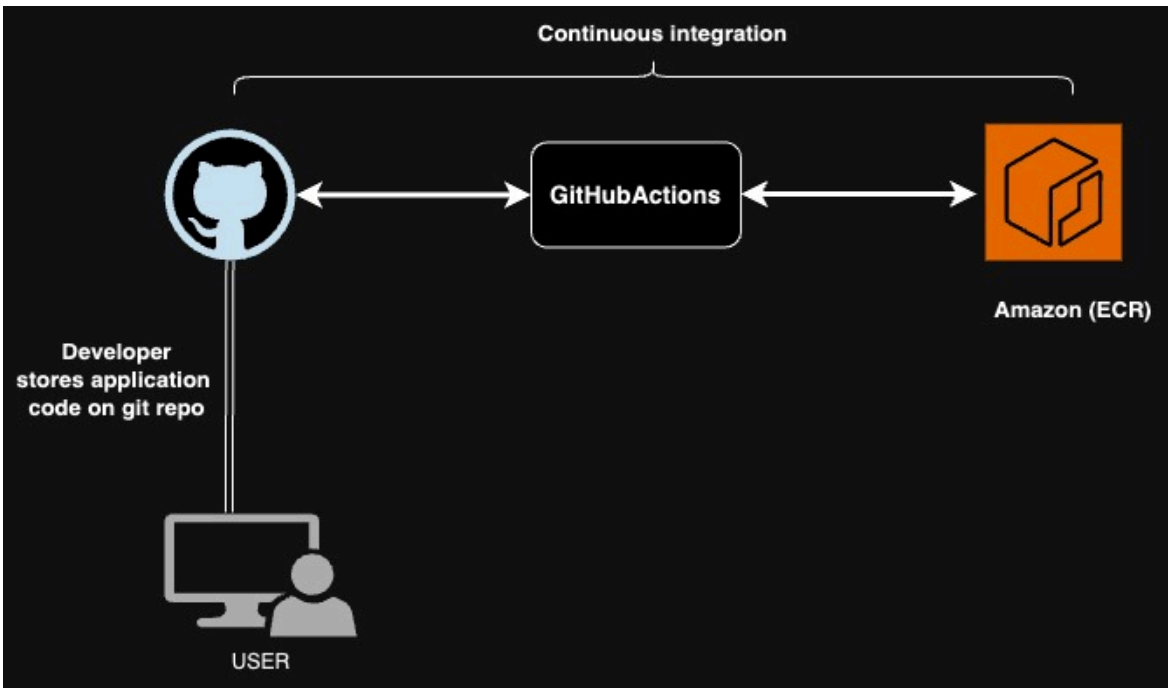
M.sai revanth

Implementation of continuous integration(CI) process using GitHub actions and pushing docker image to ECR

Overview:

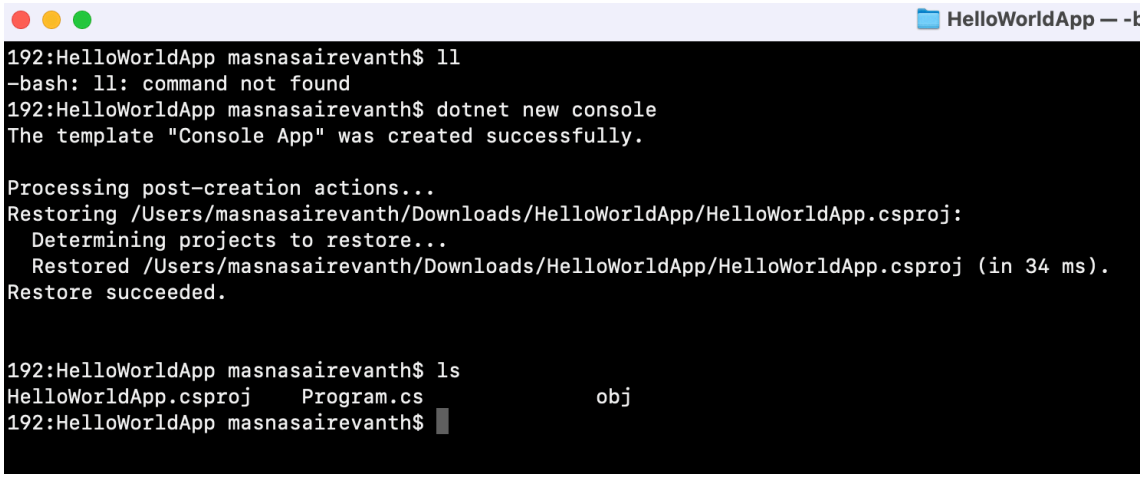
This project demonstrates the continuous integration(CI) process using GitHub Actions. I have created a DOTNET application on my local system, pushed the application to my GitHub repository and then created a GitHub Actions workflow file which contains a .YML file which creates a docker image and pushes it to amazon ECR(elastic container service). So when I make changes in the code and pushes to GitHub repository it automates the docker image creations and pushes to ECR.

Architecture



Steps:

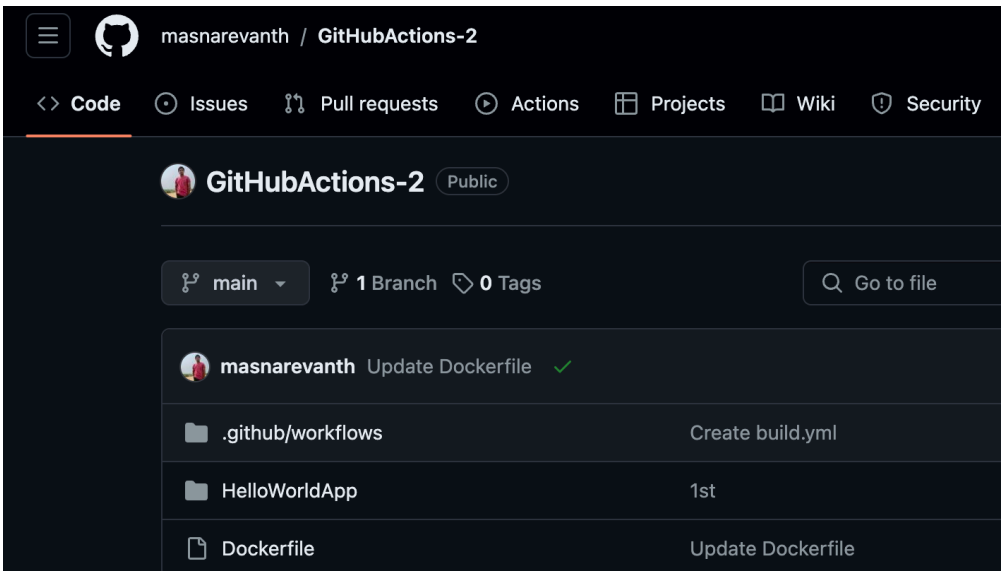
1. Let's create a .NET console application using command line first create a directory named HelloWorldApp, and run "DotNet new console" it creates a .csproj and .cs files.
2. It will displays the following output on the console.



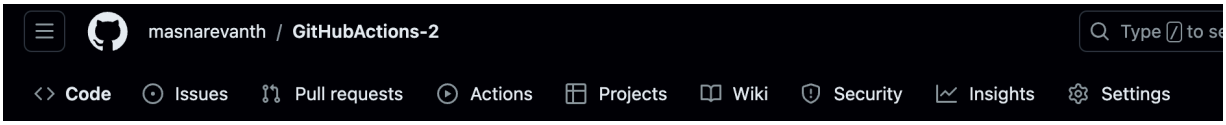
3. Open the Program.cd file end edit it edit it with the following.

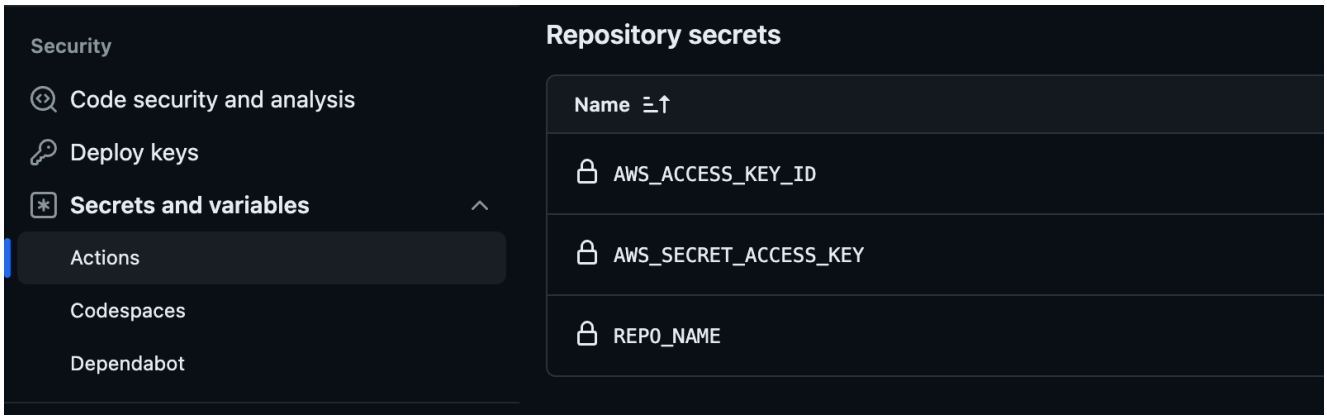
```
using System;

namespace HelloWorldApp
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("hello-world");
      }
   }
}
```

4. In the same folder create a DockerFile, this file contains the steps to create docker image.
5. Push the local code to your GitHub repository.
   - Git init
   - Git remote add origin <url of git repo>
   - Git branch -M main
   - Git add .
   - Git commit -m “commit msg”
   - Git push origin main
6. In general the git workflow look like this.
   - Git add . && git commit -m “msg” && git push origin main

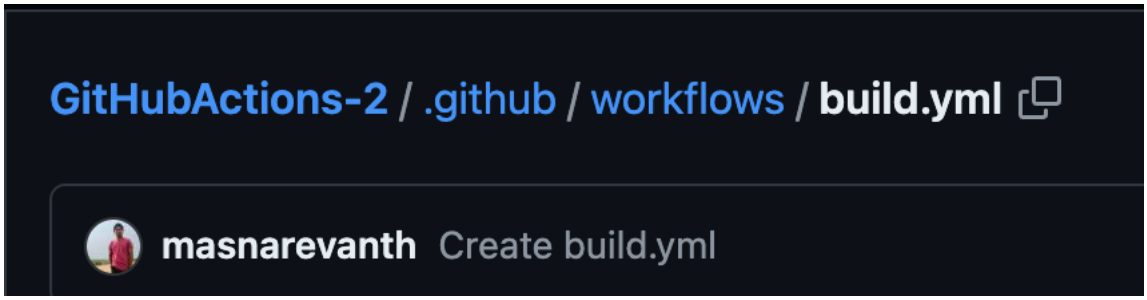


7. Go to GitHub Actions and create a workflow file.
8. Workflow file is a yaml file which contains the required steps to create docker image and push that image to amazon ECR.
9. You need to write a yaml file which consists of steps.
10. Setup the AWS credentials in your git repository setting under secrets and variables.
11. Give your credentials like:
    - AWS_ACCESS_KEY_ID
    - AWS_SECRET_ACCESS_KEY
    - REPO_NAME

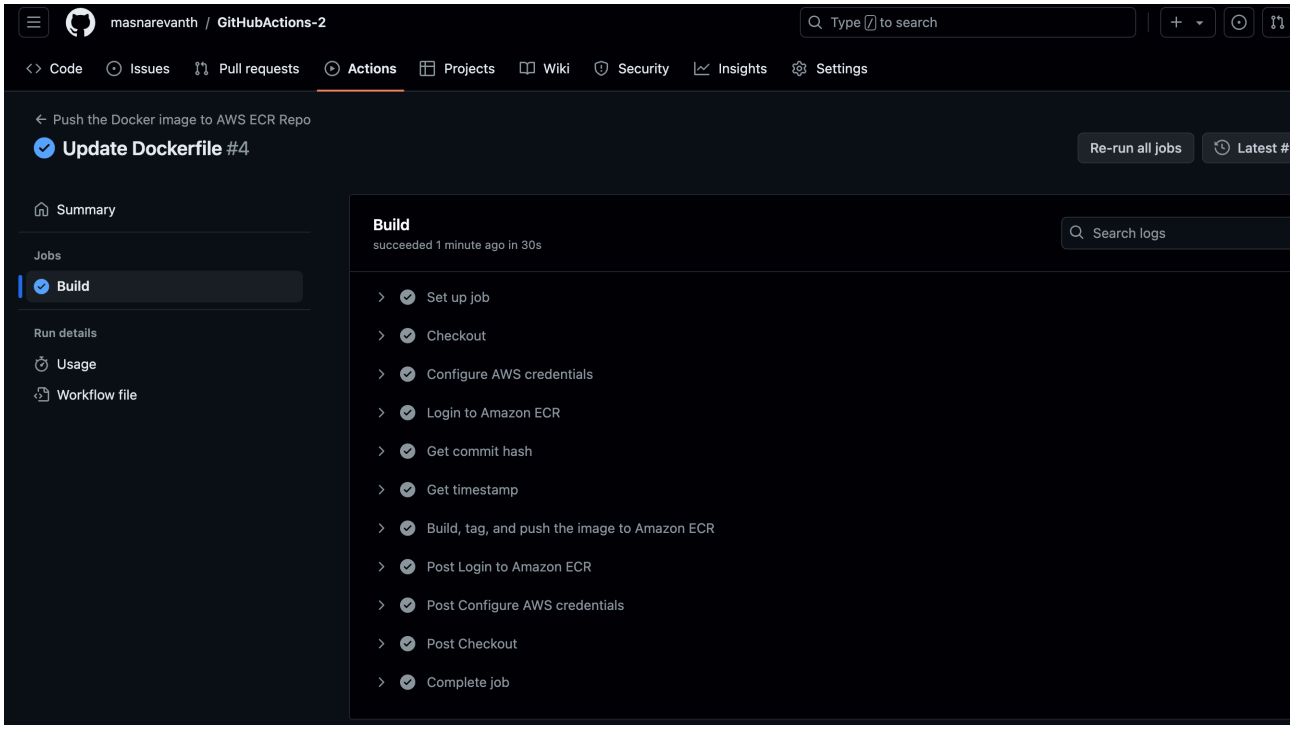




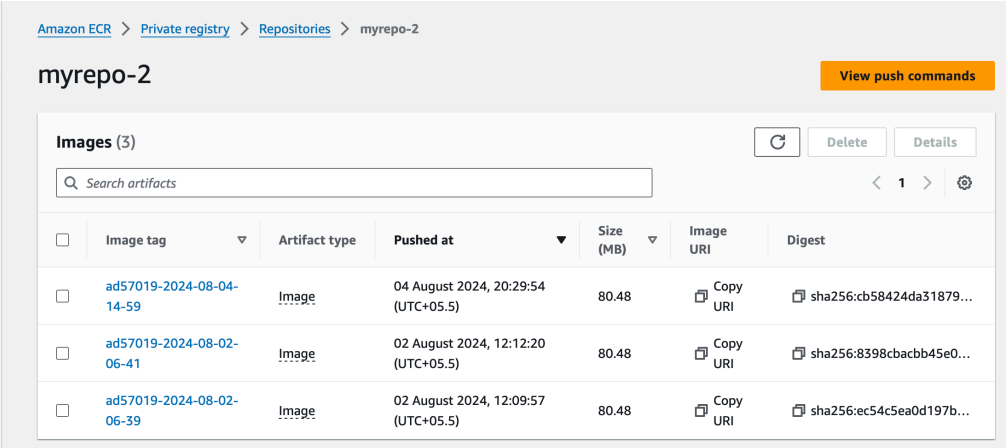
12. Write your yaml file.



13. Everything is set perfect now lets see the actions tab, if everything is properly configured your docker image is generated and pushed to your ECR repo.



14. Your .yml needs to be go through these steps successfully.
15. Using your AWS credentials it login to your account and creates an image.



16. See whenever you make changes in code on your local repo and pushes it to git repo it automatically creates docker image and pushes to ECR repo. Your image will stored on ECR.

Conclusion:

Hence, we have successfully implemented the continuous integration CI process in the project.this makes the developer work more handy.

References:

- GitHub official documentation.
- Online articles.