CI/CD with github actions to ECS

Continuous integration (CI)

Continuous integration is the practice of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when you commit or merge them, and automatically kicking off a build. With continuous integration, errors and security issues can be identified and fixed more easily, and much earlier in the software development lifecycle.

continuous delivery/continuous deployment (CD)

Continuous delivery is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

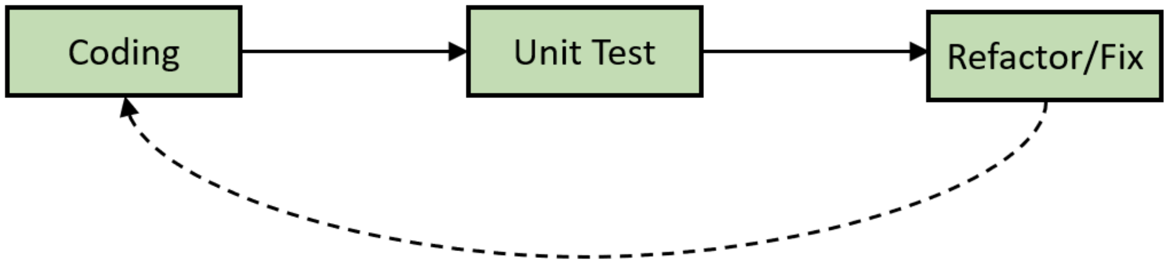
CD is the next step of CI. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

CI/CD allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.

Unit tests

"Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended"

Essentially, a unit test is a method that instantiates a small portion of our application and verifies its behavior independently from other parts. A typical unit test contains 3 phases: First, it initializes a small piece of an application it wants to test (also known as the system under test, or SUT), then it applies some stimulus to the system under test (usually by calling a method on it), and finally, it observes the resulting behavior. If the observed behavior is consistent with the expectations, the unit test passes, otherwise, it fails, indicating that there is a problem somewhere in the system under test.



Basic Unit Test Life Cycle

Pytest

Pytest is possibly the most widely used Python testing framework around - this means it has a large community to support you whenever you get stuck. It's an open-source framework that enables developers to write simple, compact test suites while supporting unit testing, functional testing, and API testing.

- In Pytest, we can use the fixture function as an input parameter of the test function, and that input parameter is already the return object.
- We have to indicate that the function is a fixture with @pytest.fixture. These specific Python decorations let us know that the next method is a pytest fixture.

Github actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

GitHub Actions goes beyond just DevOps and lets you run workflows when other events happen in your repository. For example, you can run a workflow to automatically add the appropriate labels whenever someone creates a new issue in your repository.

GitHub provides Linux, Windows, and macOS virtual machines to run your workflows, or you can host your own self-hosted runners in your own data center or cloud infrastructure.

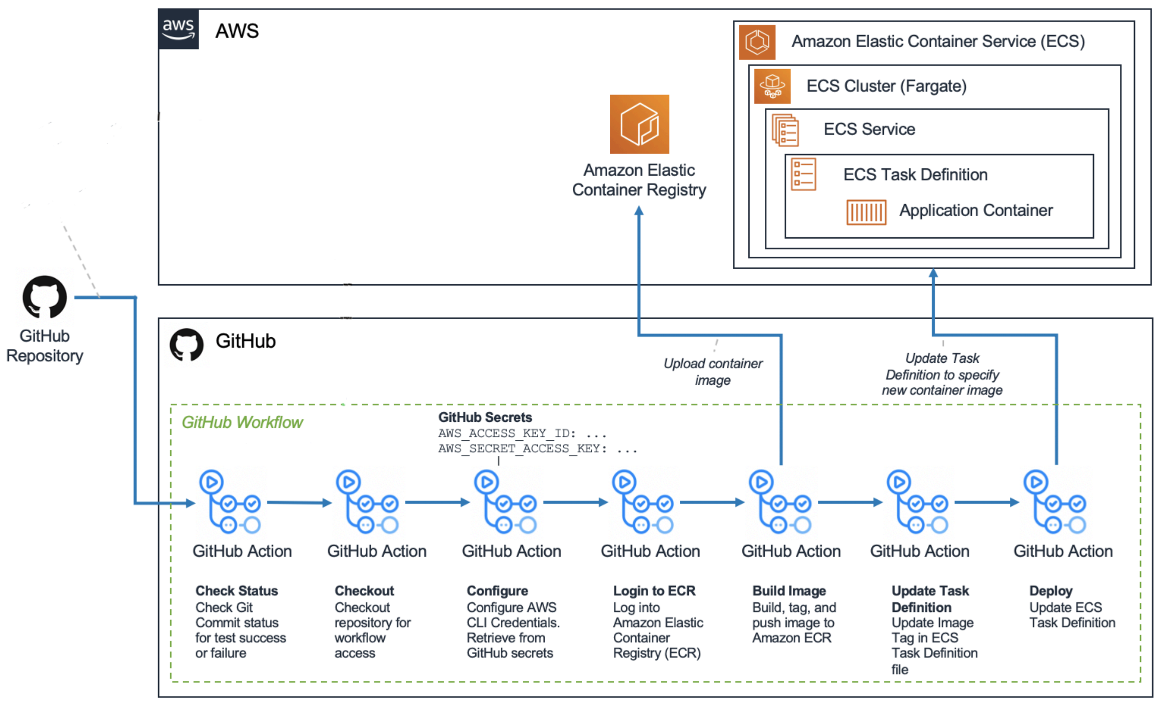
The components of GitHub Actions

- You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.

- **Workflows** A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.Workflows are defined in the .github/workflows directory in a repository, and a repository can have multiple workflows, each of which can perform a different set of tasks. For example, you can have one workflow to build and test pull requests, another workflow to deploy your application every time a release is created, and still another workflow that adds a label every time someone opens a new issue.

- **Events** An event is a specific activity in a repository that triggers a workflow run. For example, activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository. You can also trigger a workflow run on a schedule, by posting to a REST API, or manually.

- **Jobs** A job is a set of steps in a workflow that execute on the same runner. Each step is either a shell script that will be executed, or an action that will be run. Steps are executed in order and are dependent on each other. Since each step is executed on the same runner, you can share data from one step to another. For example, you can have a step that builds your application followed by a step that tests the application that was built.


AWS
Amazon Elastic Container Service (ECS)
ECS Cluster (Fargate)
ECS Service
ECS Task Definition
Application Container
Amazon Elastic Container Registry
GitHub Repository
GitHub
Upload container image
Update Task Definition to specify new container image
GitHub Workflow
GitHub Secrets
GitHub Action
Check Status
Check Git Commit status for test success or failure
GitHub Action
Checkout
Checkout repository for workflow access
GitHub Action
Configure
Configure AWS CLI Credentials. Retrieve from GitHub secrets
GitHub Action
Login to ECR
Log into Amazon Elastic Container Registry (ECR)
GitHub Action
Build Image
Build, tag, and push image to Amazon ECR
GitHub Action
Update Task Definition
Update Image Tag in ECS Task Definition file
GitHub Action
Deploy
Update ECS Task Definition

```
AWS_ACCESS_KEY_ID: ...
AWS_SECRET_ACCESS_KEY: ...
```