NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Continuous Integration

Cloud Infrastructure Engineering

Nanyang Technological University & Skills Union - 2022/2023

Course Content

- Self Study Check In
- Continuous Integration
- Activity with Github Actions

Q1: What is Continuous Integration?

a. A practice that involves frequently integrating code changes into a shared repository
b. A practice that involves merging code changes only once every few weeks
c. A practice that involves developing code in isolation
d. A practice that involves developing code without version control

Q2: What are the benefits of Continuous Integration?

a. Improved code quality, slower feedback, slower time to market, and reduced collaboration between developers

b. Improved code quality, faster feedback, faster time to market, and improved collaboration between developers

c. Reduced code quality, faster feedback, slower time to market, and improved collaboration between developers

d. Reduced code quality, slower feedback, slower time to market, and reduced collaboration between developers

Q3: What are the key components of a Continuous Integration pipeline?

a. Version control, automated testing, manual testing, and manual deployment

b. Version control, automated testing, build automation, and deployment automation

c. Version control, manual testing, build automation, and manual deployment

d. Version control, manual testing, build automation, and deployment automation

Activity

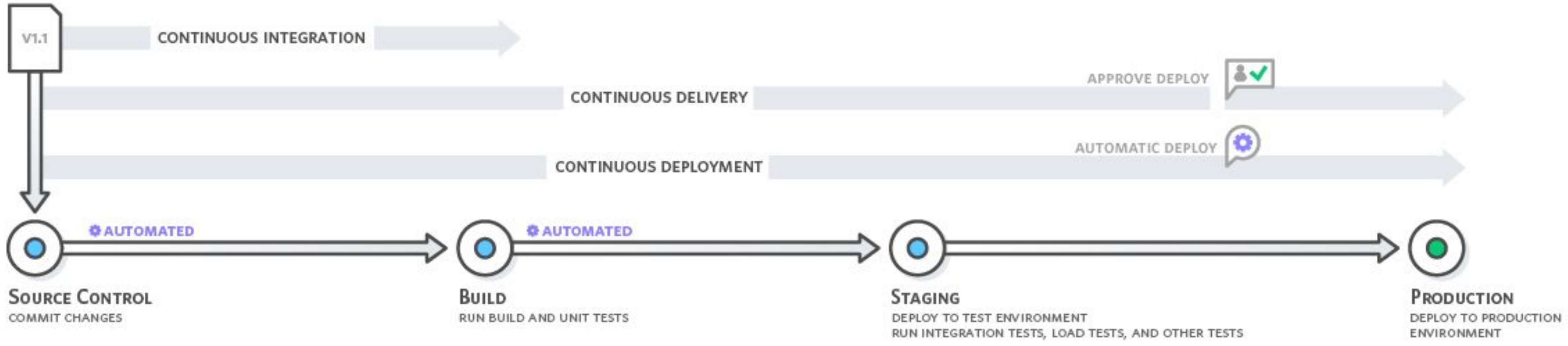
Instructor

- Ask to use AWS use single region for all learner for easier monitoring

What is Continuous Integration?

What is Continuous Integration?

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.



Why is Continuous Integration Needed? In the past, developers on a team might work in isolation for an extended period of time and only merge their changes to the master branch once their work was completed. This made merging code changes difficult and time-consuming, and also resulted in bugs accumulating for a long time without correction. These factors made it harder to deliver updates to customers quickly.

How does Continuous Integration Work? With continuous integration, developers frequently commit to a shared repository using a version control system such as Git. Prior to each commit, developers may choose to run local unit tests on their code as an extra verification layer before integrating. A continuous integration service automatically builds and runs unit tests on the new code changes to immediately surface any errors.

Benefits of Continuous Integration

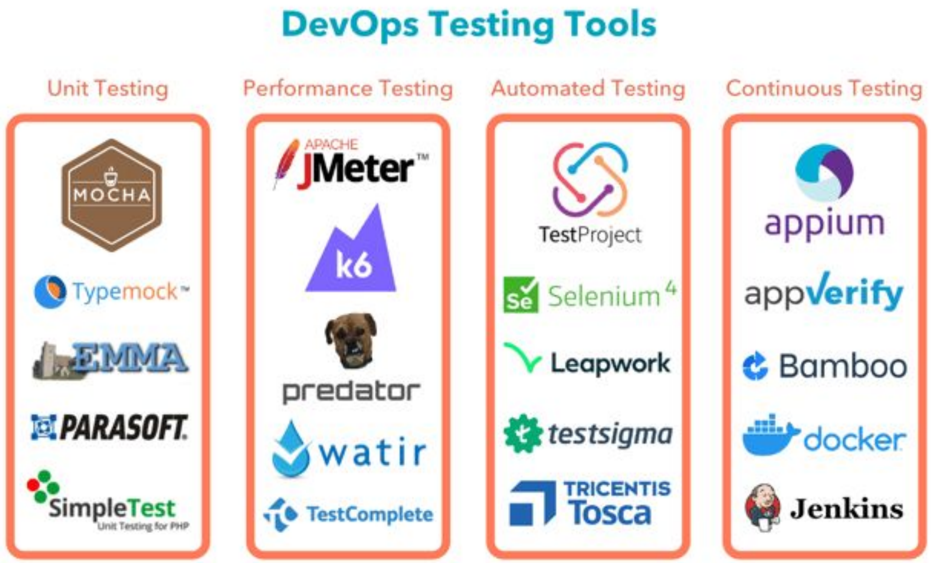
The benefits of Continuous Integration include:

- Improved code quality: CI enables developers to catch errors early in the development process, leading to better code quality and fewer bugs.
- Faster feedback: CI provides rapid feedback to developers, allowing them to quickly identify and fix issues.
- Faster time to market: CI enables teams to deliver high-quality software faster, reducing the time to market for new features.
- Improved collaboration: CI encourages collaboration between developers, enabling them to work together more effectively.

Key Components of a Continuous Integration Pipeline

The key components of a Continuous Integration pipeline include:

- Version control: Version control is essential for managing code changes and ensuring that everyone is working on the same codebase.
- **Automated testing: Automated testing is crucial for catching errors early in the development process. This includes unit tests, integration tests, and other forms of testing that can be run automatically as part of the build process.**
- Build automation: Build automation enables developers to quickly build and test their code, providing rapid feedback on any issues that arise.
- Deployment automation: Deployment automation enables developers to quickly deploy their code to production environments, reducing the time to market for new features.



DevOps Testing Tools

Unit Testing	Performance Testing	Automated Testing	Continuous Testing
MOCHA	APACHE JMeter™	TestProject	appium
Typemock™	k6	Selenium 4	appVerify
EMMA	predator	Leapwork	Bamboo
PARASOFT.	watir	testsigma	docker
SimpleTest Unit Testing for PHP	TestComplete	TRICENTIS Tosca	Jenkins

Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Activity
Implement CI with Terraform

Create new repository

Learners spend 5-10 mins to create new repository.

Activity

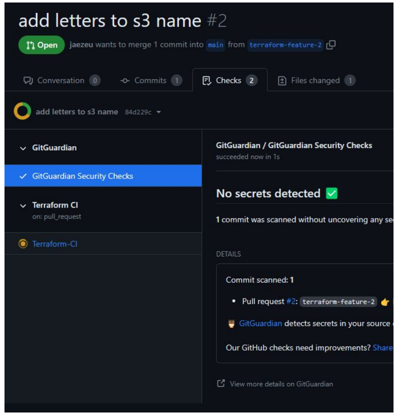
Learners try CI with Terraform 15-20 mins

- Create a .github/workflows directory in your repository on GitHub if this directory does not already exist.
- In the .github/workflows directory, create a file named terraform-ci.yml.
- Copy the YAML contents into the terraform-ci.yml (6m-cloud-3.10-continuous-integration/terraform-ci.yml at main · su-ntu-ctp/6m-cloud-3.10-continuous-integration (github.com))
- Push the workflow file into the main branch of the repository

Activity

- Upon completing previous steps, Create a feature branch for a terraform commit
- git checkout -b terraform-feature
- Then create a file named “main.tf” in the root directory of the repository with the following content. Just remember to change the bucket name to your own bucket name that is globally unique:

```
resource "aws_s3_bucket" "example" {

 bucket = "ENTER YOUR BUCKET NAME HERE"

 tags = {

  Environment = "Dev"

 }

}
```



add letters to s3 name #2

Open jaezeu wants to merge 1 commit into `main` from `terraform-feature-2`

add letters to s3 name 84d229c

GitGuardian / GitGuardian Security Checks
succeeded now in 1s

No secrets detected ✅

1 commit was scanned without uncovering any se

DETAILS

Commit scanned: 1

- Pull request #2: `terraform-feature-2` 👉

GitGuardian detects secrets in your source

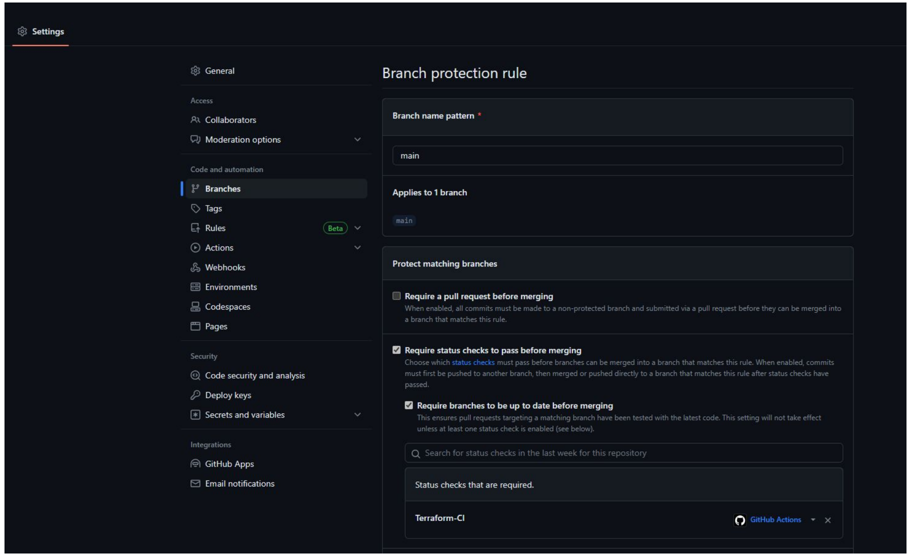
Our GitHub checks need improvements? Share

View more details on GitGuardian

- Then make a push using your feature branch (git push origin terraform-feature)
- Then create a pull request to the main branch then view the “Checks” tab on your pull request page

Activity

- Go to Github Settings and Enable Branch Protection on your main branch
- Select Enable status checks before merging and select the Terraform-CI job
- Once Enabled, change your s3 bucket name in your VS Code and make a commit again. Then observe your pull request page



Branch protection rule

Branch name pattern *

main

Applies to 1 branch

main

Protect matching branches

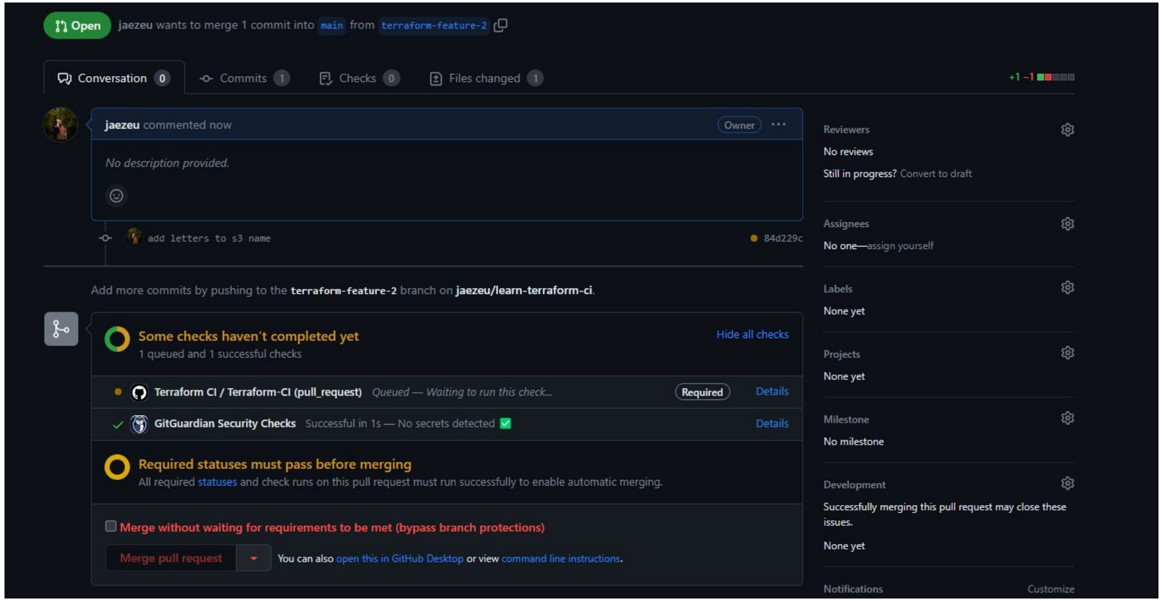
- [ ] Require a pull request before merging
  When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.
- [x] Require status checks to pass before merging
  Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.
  - [x] Require branches to be up to date before merging
    This ensures pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

Search for status checks in the last week for this repository

Status checks that are required.

Terraform-CI — GitHub Actions

Activity



Open jaezeu wants to merge 1 commit into `main` from `terraform-feature-2`

Conversation 0 · Commits 1 · Checks 0 · Files changed 1 · +1 −1

jaezeu commented now · Owner

No description provided.

`add letters to s3 name` · 84d229c

Add more commits by pushing to the **`terraform-feature-2`** branch on **jaezeu/learn-terraform-ci**.

Some checks haven't completed yet
1 queued and 1 successful checks · Hide all checks

- Terraform CI / Terraform-CI (pull_request) *Queued — Waiting to run this check...* · Required · Details
- ✓ GitGuardian Security Checks · Successful in 1s — No secrets detected ✅ · Details

Required statuses must pass before merging
All required statuses and check runs on this pull request must run successfully to enable automatic merging.

☐ Merge without waiting for requirements to be met (bypass branch protections)

Merge pull request · You can also open this in GitHub Desktop or view command line instructions.

Reviewers	No reviews Still in progress? Convert to draft
Assignees	No one—assign yourself
Labels	None yet
Projects	None yet
Milestone	No milestone
Development	Successfully merging this pull request may close these issues. None yet
Notifications	Customize

Activity
Add Unit Testing to Existing Code

Create new repository

Learners spend 20-30 mins to create new repository with the previous file in it. Follow:

- https://github.com/su-ntu-ctp/6m-cloud-3.4-cloud-native-application-containerization-i/tree/main/hello-node-app

Activity

Instructor add unit testing using jest/ supertest for the existing code

- install jest
  - npm install jest --save-dev
- add test folder and file under “__tests__/index.test.js”

Add index.test.js file under __tests__ folder

```
// index.test.js (Jest test file)

const request = require('supertest');
const app = require('../index'); // Assuming the Express app code is in a file named 'index.js'

// Jest test cases
describe('Test API Endpoints', () => {
  // Test / endpoint
  it('should return "Hello from Node!"', async () => {
    const response = await request(app).get('/');
    expect(response.status).toEqual(200);
    expect(response.text).toEqual('Hello from Node!');
  });

  // Test /test endpoint
  it('should return "Hello from /test Node!"', async () => {
    const response = await request(app).get('/test');
    expect(response.status).toEqual(200);
    expect(response.text).toEqual('Hello from /test Node!');
  });
});
```

Update package.json to have test script

```
{
  "name": "simple-node-application",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "jest --forceExit"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  },
  "devDependencies": {
    "supertest": "^6.3.3"
  }
}
```

Activity

Learners spend 10-20 to run unit test

Activity

Add Unit Testing as part of the Jobs/Actions on the GitHub Actions Workflow component

Activity

Instructor add unit testing as part of the Jobs/Actions on the GitHub Actions Workflow component

Activity

Spend 5-10 mins for Learners to add unit testing as part of the Jobs/Actions on the GitHub Actions Workflow component

Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that you created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that you created.
- Check the AWS account after learner clean up.

Questions?