Software Engineering II - Study Notes - Lecture 4

DevOps

- **Definition**: Practice of operations and development engineers working together throughout the entire service lifecycle.
- **Goal**: Increase the ability to deliver products faster with high quality.

Rapid Release Cycles

- **Modern software systems**: Often release several times a day.
- **Release pipelines**: How organizations deliver new content, encompassing integration, build, and deployment.
- **Efficient pipelines**: Crucial for modern software organizations.

Continuous Integration (CI)

- **Practice**: Developers frequently merge their code changes into a central repository.
- **Process**: Automated builds and tests are run after merging.
- **Components**:
  - Commit
  - Build
  - Test
  - Report

Benefits of CI

- Improve developer productivity.
- Find and address bugs quickly.
- Deliver updates faster.

Build System

- **Purpose**: Translate sources into deliverables automatically.
- **Why needed?**: Manual compilation is difficult for large scale systems.

Dependency Graph

- Build dependency graph visualizes the dependencies between different components/modules.

Build Tools

- Ant: Java library for automating Java applications' build processes.
- Maven: Build automation tool primarily for Java projects.
- Gradle: Build tool based on Ant and Maven concepts, using Groovy or Kotlin.
- PyBuilder: Build automation tool written in Python.

Makefile

- Defines dependencies and actions required for building a project.
- `make` command utilizes the Makefile to compile and link files based on dependencies.
- Incremental builds: Only rebuild targets that are out of date.

Continuous Integration (CI) in Practice

- Maintain a single source repository.
- Automate the build, and keep it fast.
- Fix broken builds immediately.
- Commit to the mainline every day.
- Every commit should build on an integration machine.
- Test in a clone of the production environment.
- Make the build self-testing.

Challenges with Nightly Builds

- Too infrequent and can result in difficulty identifying the cause of build failures.

CI Feedback Loop

- Run builds more frequently to match the pace of development.

Continuous Integration Tools

- On-premises: Jenkins, CruiseControl, Buildbot.
- Cloud-based: GitHub Actions, TravisCI, CloudBees, CircleCI.

GitHub Actions

- Triggered by events such as pull requests or issues.
- Contains jobs that run in sequence or parallel, inside a runner VM or container.
- Utilizes steps and actions for simplifying workflows.

GitHub Actions Workflow Example

```yaml
name: GitHub Actions Demo
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "Job triggered by a ${{ github.event_name }} event."
      ...
      - run: ls ${{ github.workspace }}
      - run: echo "Job status is ${{ job.status }}."
```

Test Automation in CI

- Integration with automated testing is essential.

- Quality Assurance (QA) teams should configure automated tests through the build system.

Static Analysis in CI

- Can be automated to scan code for issues during the build process.

References for Further Reading

- Continuous Delivery, by Jez Humble and David Farley (Chapter 3 and 6)
- GitHub Actions documentation: Events that trigger workflows
- TravisCI setup tutorial: Travis CI Tutorial

Note for Exam Preparation: Understand the interaction between build systems and their clients (e.g., Quality Assurance Personnel, Static analysis tools), the importance of incremental builds, and be knowledgeable about configuring CI tools like GitHub Actions and TravisCI.

Class Notes

Comp 4350 Software Engineering II Lecture 4

Dr. Shaowei Wang

The rapid release cycle of modern software systems

Often release several times in one day!

Release pipelines:

How organizations deliver new content

1. ### Integration
    -
    -
    -
2. Build
3. Deployment

Modern software organizations rely on an efficient and robust release pipeline!

DevOps

DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support, to increase organization's ability to deliver products faster, while ensuring high quality

Continuous integration

Continuous integration is a DevOps software development practice where developers frequently merge their code changes into a central repository, after which automated builds and tests are run on the branch.

Continuous Integration (CI)

- Commit
- Build
- Test
- Report
- Commit 9719cf0 was successfully integrated
- Commit 9719cf0
- .java
- .xml

Continuous integration benefits

Improve developer productivity Find and address bugs quickly Deliver updates faster

Source code

Deliverable What is a build system? Modern software organizations rely on an efficient and robust build system!

Why we need a build system?

.gcc -c random.c .gcc -c input.c .gcc -c main.c Link them together to complete the program

It’s extreme hard to build large system manually

Jetty dependency tree

Build tools

Ant Maven Gradle PyBuilder

How does build system work?

Define the dependency Define the action

make to the rescue

Step 1 - Expressing dependencies Step 2 - Writing recipes

This is a Makefile Step 1 - Expressing dependencies Target - An output file Dependencies - Targets that must be built prior to building this one

make command reads the makefile

$ make

Should we rebuild the entire system?

make Incremental Builds!

$ vim random.c ... $ make

Only the out-of-date targets are rebuilt!

First of all – build the dependency graph

program : random.o input.o main.o Target - An output file

Only random.c is modified

Random.o is older than random.c => receipt is executed Input.o is newer than input.c => nothing happens Main.o is newer than main.c => nothing happens Program is older than random.o => receipt is executed

Ant build file

[Example of Ant build file]

Build system interactions:

Quality Assurance Personnel

Build system interactions:

Static analysis

Build system interactions:

Code review environments

Build system interactions:

Nightly builds

Martin Fowler's practice on CI

Maintain a Single Source Repository. Make it Easy for Anyone to Get the Latest Executable Automate the Build Keep the Build Fast Fix Broken Builds Immediately Everyone Commits To the Mainline Every Day Every Commit Should Build the Mainline on an Integration Machine Test in a Clone of the Production Environment Make Your Build Self-Testing

Continuous integration tools

On-premises:

- Jenkins
- CruiseControl
- Buildbot

Cloud-based:

- GitHub Action

- TravisCI
- CloudBees
- CircleCI

On-premises VS Could-based

Github Actions

GitHub Actions workflow

[Description of GitHub Actions workflow]

Event

[Link to GitHub Actions Events]

Action

Set up GtiHub Action for a GitHub project

Tutorial [Link to the GitHub Actions Quickstart] Github repo [Link to the GitHub repository]

Reference

Chapter 3 and 6, Continuous Delivery, Jez Humble and David Farley [Reference to an article]

Set up TravisCI for a GitHub project

[Steps to set up TravisCI for a GitHub project]