SKILLS | DevOps and Cloud Computing

Continuous Integration (CI) Fundamentals and Pipeline Setup

Objective

- Understand the importance and benefits of Continuous Integration.
- Learn about different CI tools and their advantages.
- Set up a basic CI pipeline for automated builds and tests.
- Gain hands-on experience configuring and running a simple CI workflow.

Explain what Continuous Integration is and why it is crucial in software development

Continuous Integration (CI)

Continuous Integration (CI) is a software development practice where developers regularly merge their code changes into a central repository. This process involves automated builds and tests to ensure the integrity of the codebase.

- CI is a key component of DevOps practices, aiming to streamline the software development lifecycle by providing rapid feedback on code changes

Key Components of CI:

- Frequent Integration
- Automated Builds
- Automated Testing
- Version Control Systems

Continuous Integration (CI)

Why is Continuous Integration Crucial in Software Development?

1. Reduces Risk and Increases Quality
2. Enhances Collaboration and Transparency
3. Speeds Up Development and Deployment
4. Supports Agile Development

Pop Quiz

Q. What is typically involved in the Continuous Integration process?

A

Only version control without testing.

B

Automated builds and tests triggered by code commits.

Pop Quiz

Q. What is typically involved in the Continuous Integration process?

A

Only version control without testing.

B

Automated builds and tests triggered by code commits.

PW SKILLS

Benefits of CI

Benefits of CI

Here are the key benefits of Continuous Integration:

- **Early Bug Detection:** Identifies issues early in the development cycle, reducing downstream problems.
- **Faster Feedback Loops:** Provides immediate feedback on code changes, enhancing developer productivity.
- **Improved Collaboration and Code Quality:** Enhances teamwork and ensures a stable, reliable codebase through regular integration and testing.

Discuss popular CI tools

Popular CI tools

Jenkins:

- Open-source: Free to use and modify.
- Highly Customizable: Supports a wide range of plugins for various tasks and integrations.
- Self-hosted: Can be run on-premises or in the cloud.

Travis CI:

- Cloud-based: Hosted solution, reducing infrastructure management.
- GitHub Integration: Seamlessly integrates with GitHub repositories, making it a popular choice for open-source projects.
- Multi-language Support: Supports a variety of programming languages.

Popular CI tools

CircleCI:

- Docker and Kubernetes Support: Optimized for containerized environments.
- Speed and Efficiency: Designed for fast build and deployment cycles.
- Cloud-based: Offers scalability and reliability in cloud environments.

Pop Quiz

Q. Which CI tool is known for its tight integration with GitHub?

A

Travis CI

B

CircleCI

Pop Quiz

Q. Which CI tool is known for its tight integration with GitHub?

A

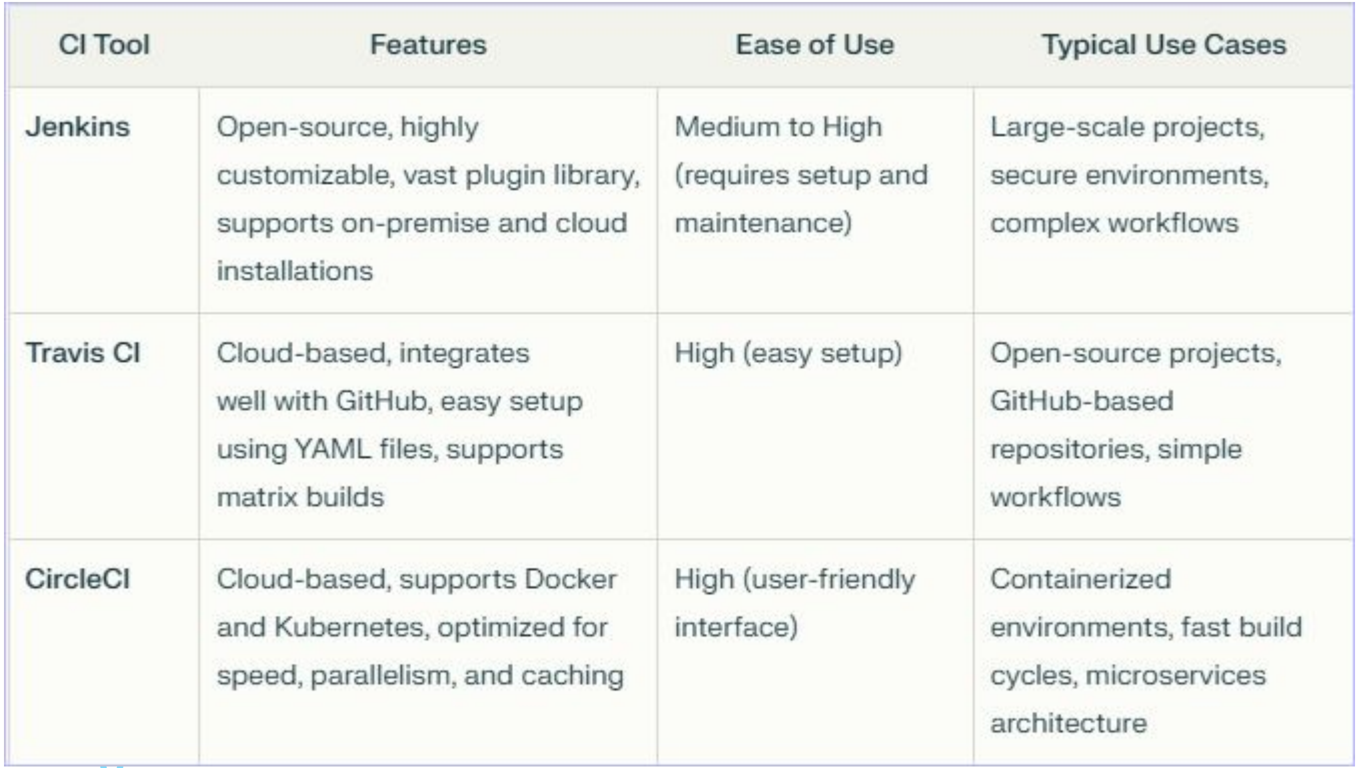
Travis CI

B

CircleCI

Compare features, ease of use, and typical use cases

Let’s compare



CI Tool	Features	Ease of Use	Typical Use Cases
Jenkins	Open-source, highly customizable, vast plugin library, supports on-premise and cloud installations	Medium to High (requires setup and maintenance)	Large-scale projects, secure environments, complex workflows
Travis CI	Cloud-based, integrates well with GitHub, easy setup using YAML files, supports matrix builds	High (easy setup)	Open-source projects, GitHub-based repositories, simple workflows
CircleCI	Cloud-based, supports Docker and Kubernetes, optimized for speed, parallelism, and caching	High (user-friendly interface)	Containerized environments, fast build cycles, microservices architecture

Take A 5-Minute **Break!**

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

Explain the CI pipeline stages

CI Pipeline stages

Build:

- Purpose: Compile the code and check for syntax errors.
- Process: This stage involves taking the source code from a version control system and converting it into an executable format. It ensures that the code can be compiled without errors.

Test:

- Purpose: Run unit tests to validate functionality.
- Process: Automated tests are executed to verify that the code behaves as expected. This includes unit tests, integration tests, and sometimes UI tests.

CI Pipeline stages

Deploy (Optional):

- Purpose: Automate deployment in later CI/CD stages.
- Process: While deployment is typically part of the Continuous Deployment (CD) phase, some CI pipelines may include automated deployment to staging or production environments if the tests pass. However, in a strict CI context, deployment is not always included.

Pop Quiz

Q. What is the sequence of stages in a typical CI pipeline??

A

Build, Test, Deploy

B

Deploy, Build, Test

Pop Quiz

Q. What is the sequence of stages in a typical CI pipeline??

A

Build, Test, Deploy

B

Deploy, Build, Test

Demonstrate setting up a simple CI workflow

Let’s do it

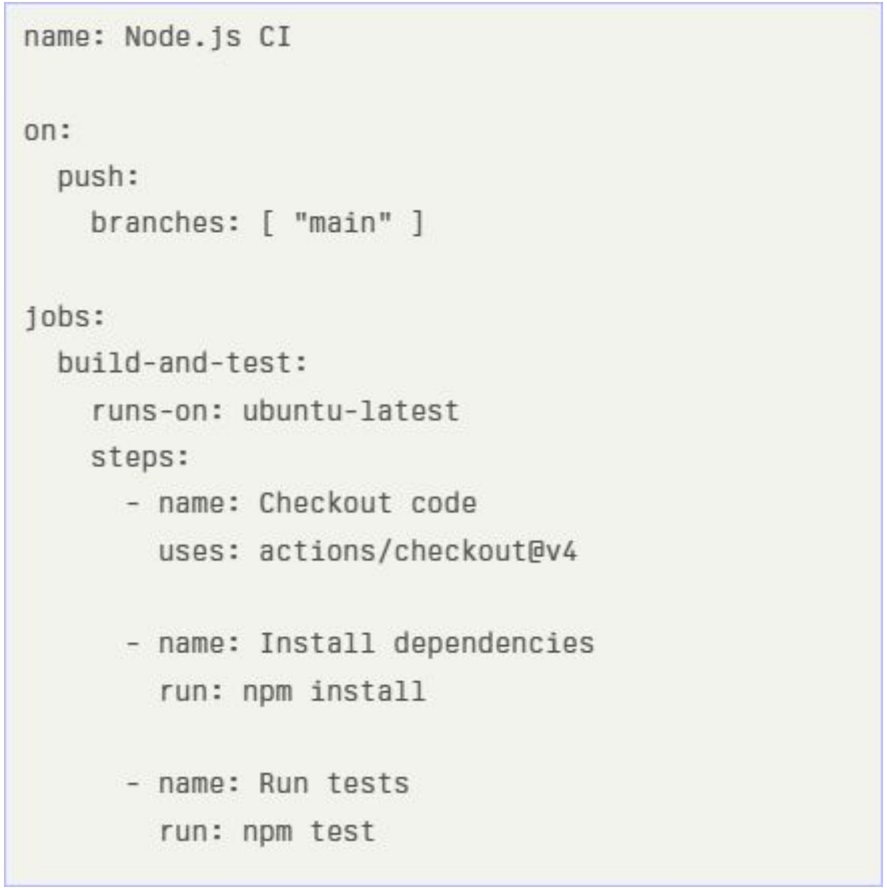
Step 1: Create a Repository with a Sample Project

- Create a new repository on GitHub.
- Add a sample project (e.g., a simple Node.js app) to the repository.

Step 2: Add a Configuration File

- Create a new directory named .github/workflows in your repository.
- Create a YAML file (e.g., ci.yml) inside this directory.

Example ci.yml file for a Node.js project:



```
name: Node.js CI

on:
  push:
    branches: [ "main" ]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Install dependencies
        run: npm install

      - name: Run tests
        run: npm test
```

Let’s do it

Step 3: Trigger an Automated Build and Test Process

- Commit and push the changes to your repository.
- Navigate to the "Actions" tab in your repository on GitHub.
- View the workflow run to see the automated build and test process.

This setup will trigger a CI workflow every time you push changes to the main branch, automating the build and test process for your project.

PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this session
- Review your lecture notes
- Note down questions and queries regarding this session and consult the teaching assistants

Thanks

!