CI/CD with Jenkins

Automating build, test, and deploy on-premise or in the cloud

Christopher Thielen, Evan Katsuranis. December 8, 2020.

Introduction to Jenkins

What is Jenkins?

The Gold Standard

- Jenkins is a **self-hosted** CI/CD tool which runs anywhere (Linux, Windows, macOS, VM in the cloud, etc).
- Jenkins can be run solo or in clusters for larger loads (e.g. E2E test suites).
- It is the oldest and most widely used continuous delivery tool to date
- Jenkins performs tasks organized into jobs. Jobs run through triggers, e.g. manual, API, repository push, another job etc.
- Tasks can be configured in Jenkins directly or through a pipelines file (`Jenkinsfile`) stored in a project's source code repository.

Evolution of Jenkins

Historical? Modern?

- Jenkins (formerly Hudson) has been around since 2004, built by a developer who worked at Sun to automatically run his tests before deploying his code.
  - Tired of incurring the wrath of his team every time he broke the build.
- Notably, Jenkins predates the DevOps movement, so is it still a good choice today?

Yes

Evolution of Jenkins

Historical? Modern?

- 2016 - Version 2.0 - Pipelines became standard
  - Job tasks can be version controlled with source code
- 2018 - JCasC
  - Jenkins can be configured via XML and scripts
- Continues to evolve: LTS releases and weekly releases

Why use Jenkins

Historical? Modern?

- Can’t use a cloud tool due to security requirements.
- Portability, Jenkins will run in any environment.
- Developer familiarity (15 million Jenkins Developers, 68% of CI done with Jenkins according to Business Wire in 2018.)

Demo

The Net Effect of CI/CD

The Net Effect of CI/CD

- Recall that Jenkins began as a way for a developer to not break builds and reduce friction with the team
- CI/CD is a standard which improves team/cross-team collaboration:
  - Developers: Avoid breaking builds
  - Product Owners: Encourage a testing culture with requirements traceability
  - Security: Automated security scanning
- Reduction of friction improves agility, delivers features faster, keeps users happy

A Story of Adopting CI/CD

A Story of Adopting CI/CD

Pre-CI/CD

- Pre-CI/CD Practices:
  - Team manually builds software.
  - Team manually sets up and deploys to each environment.
    - Test/Stage/Prod environments may drift if someone neglects an update.
  - Any re-deployment or rollbacks are performed manually.
    - Team may or may not have an artifact repository to restore previous deployments if needed.

A Story of Adopting CI/CD

Automate builds, automate test results

- Team spins up a CI/CD tool to automatically build artifacts
- Now there is an artifact repository where the team can deploy in a unified but manual process to deploy both new releases and rollbacks
- Later, team configures CI/CD tool to run tests on each commit

A Story of Adopting CI/CD

Push-button deployments

- Team configures CI/CD tool to connect to various servers (SSH keys, etc.)
- Team enables auto-deploy to a test environment, speeding up QA process
- With the buy-in from management, team configures a push-button approach to deploying those artifacts to other environments but still does not deploy to production automatically

A Story of Adopting CI/CD

Improvement with Metrics

- The team starts collecting metrics on its various environments and develops a baseline for the success of an automated deployment
- With this the team is ready to convince management to enable automatic deployments to production
- Those automatic deployments can then be measured for success automatically against the pre-defined baseline

A Story of Adopting CI/CD

CI/CD Nirvana

- Each push to the main repository is automatically run against a battery of tests. Upon passing, the changes are deployed to production.
- Production metrics are automatically monitored to determine the quality of the change and determine if a rollback is necessary (page load time, error rate, etc.)
    - If necessary, a rollback is automatically performed based on the metrics.
- Team adopts feature flagging so completely new features or massive changes happen behind-the-scenes but still in production.

Thanks