Linux Users Group @ UT DALLAS
Continuous Integration / Continuous Deployment

2017-10-21
Overview

● Introduction to CI
● GitHooks
● External Tools
  ○ travis-ci
  ○ Jenkins
  ○ Buildbot
Introduction to CI/CD
“Traditional” Development

1. Write code
2. Occasionally test
3. Eventually Deploy
What’s wrong with this model?

- People have to do everything **manually**
- **Individuals** are responsible for testing
- Deployment is not **automatic**
- **Disconnect** between development, testing and deployment
- No **global** view
- People make mistakes
- People get lazy or forget
Continuous Integration

Write → Test Automatically → Global Dashboard
Continuous Delivery

Write → Test Automatically → Global Dashboard

Deployment Pipeline
What kind of tests?

- **Code linting**
  - Enforcing *style* guidelines and best practices

- **Build tests**
  - Ensuring code *compiles* correctly

- **Unit tests**
  - Verifying *proper operation* of individual units of code

- **Integration tests**
  - Validating software modules as a *group*
GitHooks
A brief overview of Git

Developers write code, creating snapshots called “commits”

One of more developers push and pull their code in a central repository
Git “Hooks”

Git hooks are scripts that run automatically every time an event occurs in a Git repository.

- What kinds of events?
  - Pre/post commit, ...
Demo!
External Tools
Travis CI

- Integrates well with GitHub
- Free for Open-Source projects
- Reads from a .travis.yml file in your repo
  - Easy to set-up
Demo!
Jenkins

- **Self-hosted automation server**
- Define **hooks, tasks, and pipelines**
- Extensible and configurable with **plugins**
- More **integrations** than other solutions
Demo! (If there’s time + interest)
Buildbot

- **Self-hosted job scheduling system**
- More of a **framework** than a ready-to-use app
- Better suited for **complex** release tasks
- Configurable via **Python**
Questions?