Docker & Continuous Integration

Speaker: David Power
Company: WP Engine

Topics

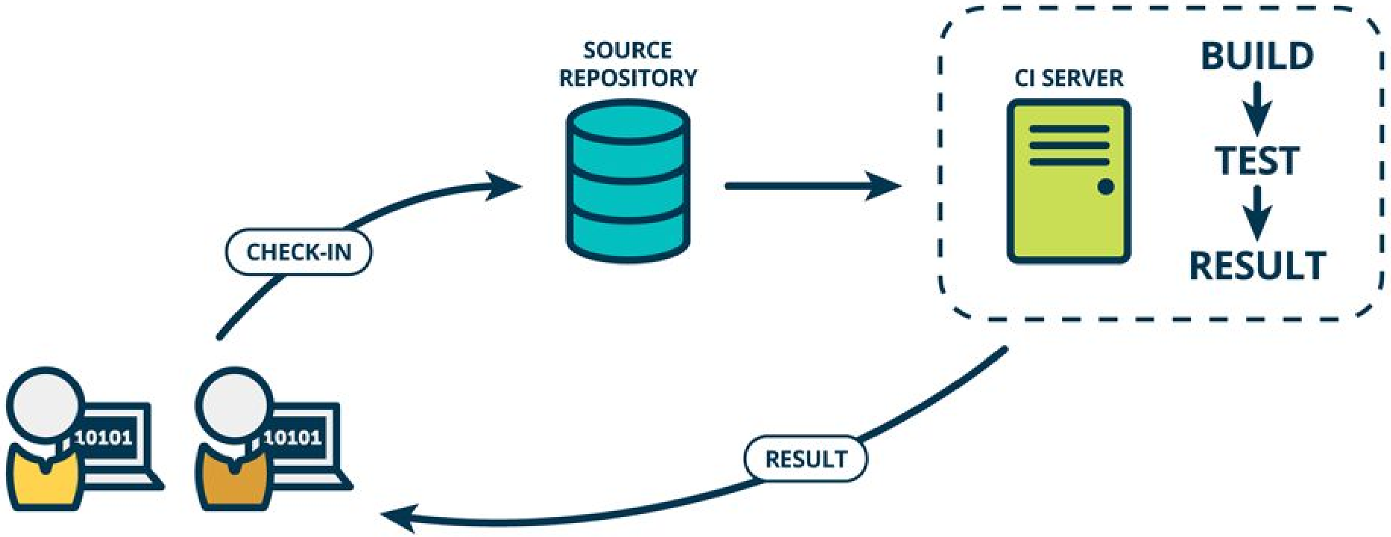
- Define Continues Integration
- The Good...
    - CI Tool - Jenkins (Demo)
- The Bad...
- A possible solution
    - Containerised it! (Demo)

What is continues Integration (CI)?

The continued merging of code to a *singular mainline branch*,

where code is built, tested and ***mingled*** with other code changes

The Good…



Source: https://blog.snap-ci.com/

The Good…

- Helps avoid nasty last minute merge conflicts - “*merge hell*”
- Should run against **branch** and **PR merge**
- Self testing; unit, smokes, integration, etc run per commit
    - Can act as a gate to complete merge in PR tool
- Reduce *Feedback Loop* duration
- *Continuous Integration Environment* should attempt to emulate production (if possible)
- Reduce PR review time

Demo 1: Installing Jenkins using Docker

Agenda:

1. When to find and download Jenkins Docker Image
   a. Setup - What’s involved?
2. Discuss demo project
3. View Failing Build
4. Discuss installation of software required to build and run project on CI Environment
5. View Passing build

The Bad...

Multiple projects with conflicting dependency versions makes CI complicated

Engineers must manage build nodes that can build projects with specific **dietary requirements**.

- Libraries and tool versions must match those the software was developed to

Simplification but...

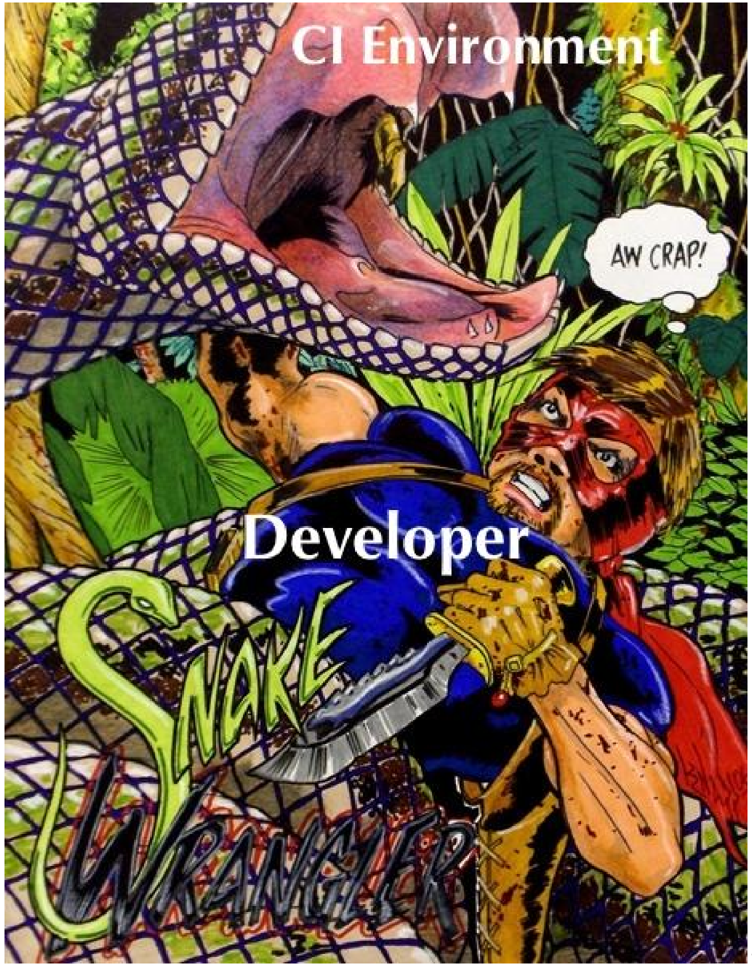
- Package managers will help but those package managers have their own dependencies, what about a compiler version?

The Bad...

Time is wasted trying to wrangle the CI Anaconda

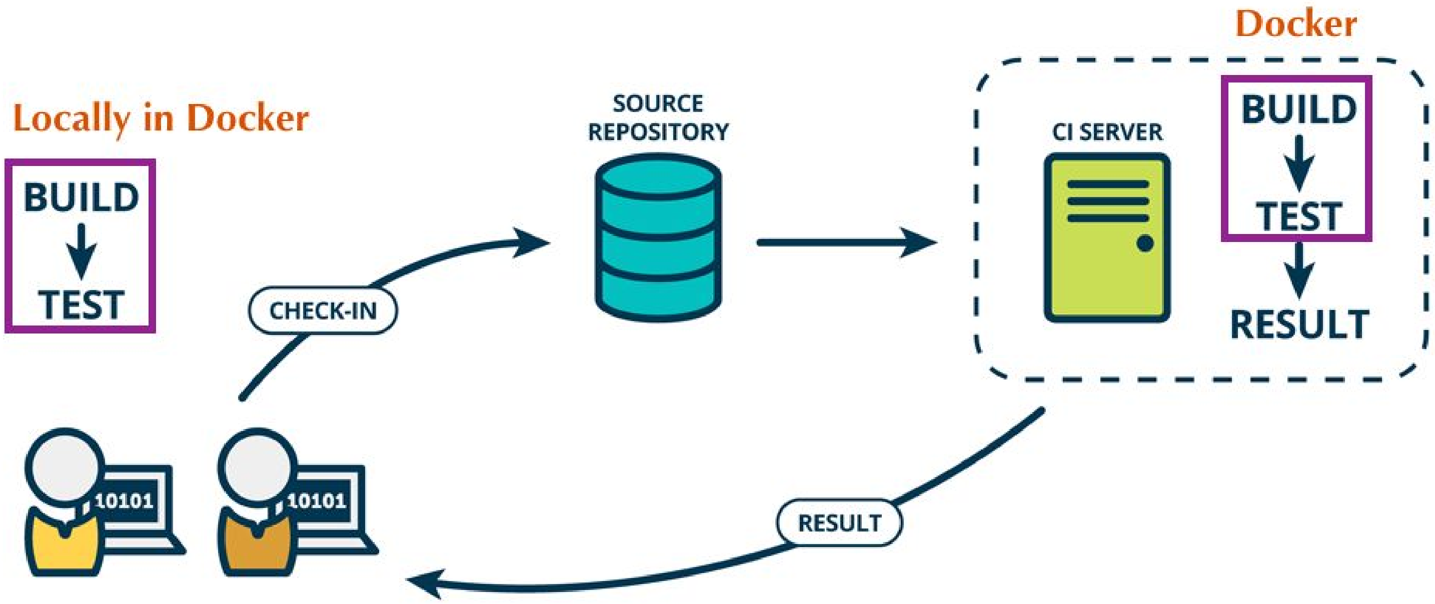
Updates to packages that make your builds pass might cause other project builds to fail

Wouldn't it be easier if those dependencies didn't exist in Jenkins?



Artist: Nelson Knapp

A Possible Solution



A Possible Solution

Containers the Build and Test stages

- CI server needs Docker installed, that’s it!
- **Portability**, Enable developers to run the CI build and test stages locally
    - Debug failures - *why on earth is that test failing?!*
- **Tightly couple** the project dependencies to the project in code
    - CI environment dependency lag is no longer an issue
        - *No need to remember to update jenkins*

Demo 2: Containerise `Build & Test’

Agenda:

1. Checkout example project
2. Demonstrate that dependencies are not in Jenkins
3. Run *Build & Test* stage **locally**
4. Start build on Jenkins by pushing changes

A Possible Solution

Can you go too far?

Yes!

- Containerising the entire CI environment shifts the dependency issue discussed previously into the container
    - What happens if we update the version of Jenkins or one of its plugins?

But it isn’t perfect!

Problem

- Leftover, running containers and images need to be managed
  - can consume all server resources

Solution

- CI Pipelines should try clean-up after themselves
- Delete stale images and running containers with Cron jobs

Questions?

Couldn’t think of one?

Ask on our Slack channel ***Docker Limerick***

Thank you