Get into Continuous Integration with CircleCI

Andres Arias
@codefactorycr
aariasc@gmail.com

Thanks to our Sponsors!

Partners

Premier

Logo:

Agenda

- What is Continuous Integration (CI)?
- What is not CI?
- Why do we need it?
- Some concepts
- Why CircleCI?
- Hands on examples

Slides and code samples available in https://github.com/aariasc/utahgeek-circleci

About me

- Coffee Drinker
- Crazy cat guy
- Software Engineer since 2002
- Sr. DevOps Engineer @ Powerschool
- Born in Costa Rica! Living in SLC since 2013

What is Continuous Integration (CI)?

Software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including tests) to detect integration errors as quickly as possible.

- *Martin Fowler*

Build code better, faster, cheaper!

What is not CI?

- Only nightly/scheduled builds
- Scheduled (and painful) integration points
- Building only with an IDE
- Developer or Feature branches that live forever

Why do we need it?

It’s all about reducing risks!

- Fixing bugs late is expensive
  - The sooner they are found, the easier and cheaper to fix
- Lack of team unity (e.g. not being aware what others are doing)
  - API changes, library upgrades, etc
- No project visibility
  - Are tests failing? Do we have any automated tests?
- No deployable version
  - But it works in my machine!!

Remember: better, faster, cheaper

Some Concepts: A Build

You need to define what is a **build** for you. Depends on your language, tools and needs, and will change with time

Important: Compilation != Build

A build should include at least:

- Compilation
- Test Execution
- Nice to have: Code Inspection, Automated Deployment, Document Generation, DB Updates

But what if my code is not compiled??

This is not a excuse! You could even have CI in a static HTML project (example later)

Options available:

- Static Code Analysis
- Code Linting
- Code Sniffing Tools (e.g. mess detectors, copy-paste detectors, enforce code styles, etc)
- Unit Tests

Some Concepts: A build script

After defining what is a build for you, now create a script to automate its creation and make it **repeatable**

Can be a small shell script, or use tools like npm, grunt, gulp, maven, gradle, msbuild, etc

This script has to live with the code (in VC) and available for all developers

Developers **must** be able to run it locally

Don't rely only in the IDE to create your build

CI tools will automate the execution of this script, they won't create it for you (at least CircleCI won't)

Why CircleCI?

- Docker-based CI environment as a service. No need to set up a build machine
- Free package with 1 container! Perfect to get started
- Every build will use a clean docker container, so no weird dependencies
- Fully customizable workflows: from a simple build, to a more complex scenario including deploying to production
- Seamless integration with Github and Bitbucket
- Workflow configuration is handled as a YAML file, which lives in the same code repo
- Every commit will trigger a new build. No human interaction required
- Traceability: you can see what was in the commit that triggered that build
- Sorry - no Windows support in the meantime

https://circleci.com

Examples

Thank you!

Happy coding!