Improve your Development Lifecycle with CI/CD

TechConnect Conference March 2019

Maxime Deravet

Agenda

1. Who i am/What I do
2. Overview of What CI/CD is
3. Pipelines
4. CI/CD as code
5. Tooling

Who am I

- Solution architect with the Enterprise Document Management team within UW-IT IM
- Worked as Software engineer for close to 10 years
- Tired dad of a one year old

Enterprise Document Management

We are helping departments to create and manage electronic documents and automate their business processes.

Responsible for:

- Document Management
- eSignatures

Working with departments on campus like:

- Facilities
- Office of Student Financial Aid
- Procurement services
- ...

Enterprise Document Management tech stack

- About 14 rest APIs written in Java
- 4 Single page web applications
- ~ 12 serverless lambda functions, written with Python or NodeJS
- All automatticaly deployed on AWS through CI/CD

We couldn't manage deployement of all these apps manually

What is CI/CD ?

CI/CD or CICD may refer to the combined practices of continuous integration and continuous delivery[1]

[1]https://en.wikipedia.org/wiki/CI/CD

What is Continuous Integration ?

Practice where a developers commit code to the repository often, possibly multiple times a day.

Each of these commit are validated

validation could be :

validation could be :

- code builds

validation could be :

- code builds
- unit tests aren't failing

validation could be :

- code builds
- unit tests aren't failing
- code coverage is > x%

validation could be :

- code builds
- unit tests aren't failing
- code coverage is > x%
- commit follow specific code style

validation could be :

- code builds
- unit tests aren't failing
- code coverage is > x%
- commit follow specific code style
- commit has been signed

validation could be :

- code builds
- unit tests aren't failing
- code coverage is > x%
- commit follow specific code style
- commit has been signed
- ...

The goal is to detect errors quickly and to be able to easily pinpoint them

What is CD ?

Continuous deployement

Continuous deployment (CD) is a software engineering approach in which software functionalities are delivered frequently through automated deployments[1]

[1]https://en.wikipedia.org/wiki/Continuous_deployment

Continuous deployement vs Continuous delivery

Continuous delivery is the ability to deliver software that can be deployed at any time through manual releases; this is in contrast to continuous deployment which uses automated deployments[1]

[1]https://en.wikipedia.org/wiki/Continuous_delivery

The goal is to remove as much human interaction as possible

The goal is to remove as much human interaction as possible

Because :

- We make mistakes

The goal is to remove as much human interaction as possible

Because :

- We make mistakes
- We forget how to build/deploy

The goal is to remove as much human interaction as possible

Because :

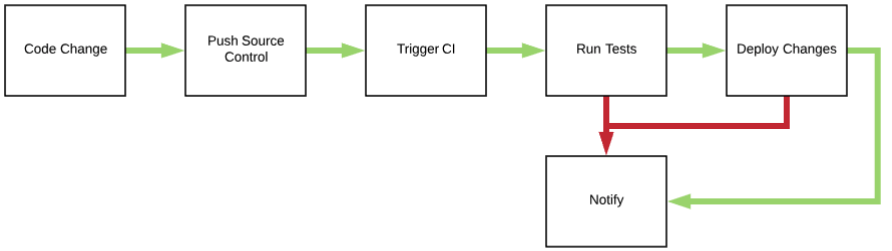
- We make mistakes
- We forget how to build/deploy
- We don't have time

The goal is to remove as much human interaction as possible

Because :

- We make mistakes
- We forget how to build/deploy
- We don't have time
- Computers are better than us at being consistent

CI/CD Pipeline



Where do I start ?

Start small, build from there

- Is my code building ?

Where do I start ?

Start small, build from there

- Is my code building ?
- Add Unit tests ?

Where do I start ?

Start small, build from there

- Is my code building ?
- Add Unit tests ?
- Add end to end tests ?

Where do I start ?

Start small, build from there

- Is my code building ?
- Add Unit tests ?
- Add end to end tests ?
- Check code coverage ?

You don't need to go full continuous deployement to get some benefits of CI/CD !

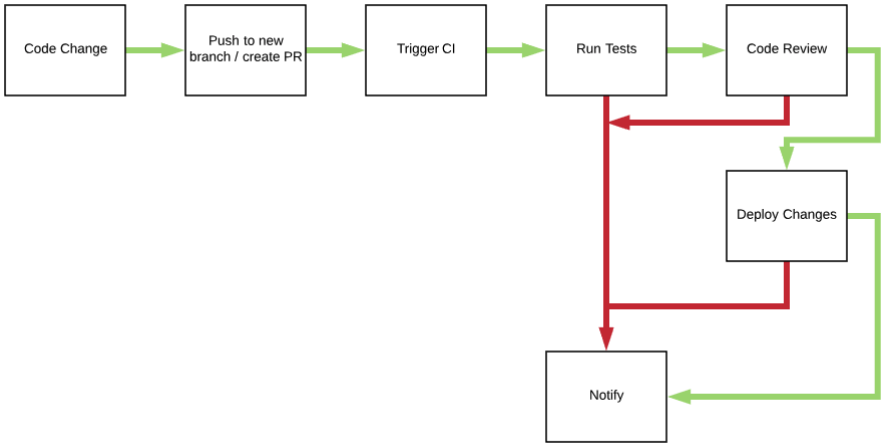
What about code reviews ?

Why is code review important?

It'll let you catch that kind of bugs

It's a good way to do knowledge sharing

Include it in your pipeline with pull requests



What EDM is doing with CI/CD

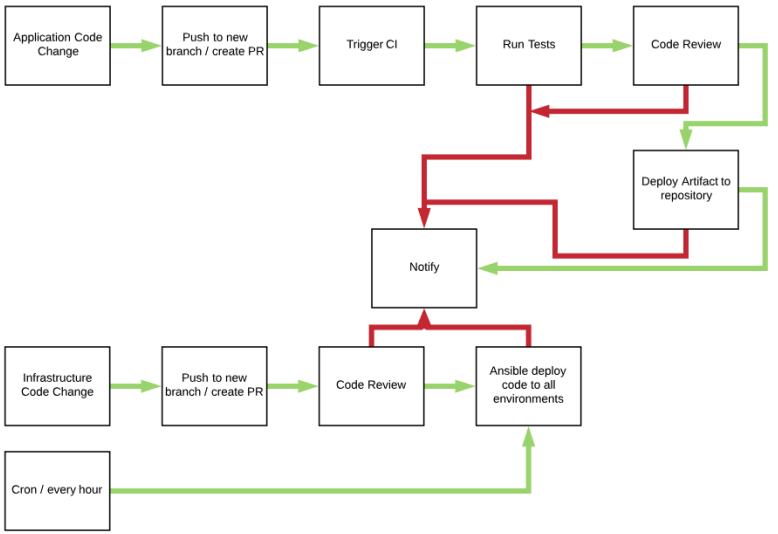
We wanted to easily gate keep what version is deployed in which environment

We wanted to have a single "entrypoint" for deployment

We are currently going the continuous delivery way

With a separated pipeline for deployment

EDM pipeline overview



Tools

CI/CD : Probably too many to list them alls

A few big names :

- Bamboo
- Jenkins
- Travis CI
- Azure Pipelines
- Gitlab CI/CD

CI/CD tools : A shift in philosophy

"Older" philosophy

- Manual configurations of your jobs
- Need to have build agents that have the capability you needs
  - is Java 8 installed ? What if I need Java 11
- Difficult to try a new feature in a new branch without breaking your CI for your master branch
- No easy way to rollback changes
- Overall a lot of maintenance and knowledge of the tool needed

Example: Bamboo

Now: Configuration as code

Travis CI :

.travis.yml

```
language: java
```

This will make travis run `mvn install` and `mvn test` on every commit

Wanna try java 11, open a new PR

.travis.yml

```
language: java
jdk: openjdk11
```

Example

Need to install a specific build tool ?

.travis.yml

```
language: node_js
node_js: lts/*

before_install:
  - curl -o- -L https://yarnpkg.com/install.sh | bash -s -- --version 1.3.2
```

Multi stage example

These examples were for Travis CI but most modern CI/CD tools support configuration as code

Azure pipelines

azure-pipelines.yml

```
steps:
- task: Maven@3
  inputs:
    mavenPomFile: 'pom.xml'
    jdkVersionOption: '1.11'
    goals: 'package'
```

Gitlab CI ?

.gitlab-ci.yml

```
image: maven:latest

build:
  stage: build
  script:
    - mvn $MAVEN_CLI_OPTS compile

test:
  stage: test
  script:
    - mvn $MAVEN_CLI_OPTS test
```

The syntax is slightly different between all of these but the general idea is the same :

Your CI/CD pipeline is configured with code and lives in your application repository

Really powerfull because the people who built the code know how to run/test it

I configured CI, I wrote some unit tests, now what ?

Write more tests !

Unit tests are great but they are sometimes not sufficient

They only test an individual unit of your code

[1]

End to End (E2E) tests ?

The complexity is :

"How do I test something that needs to be deployed to be testable without breaking my whole environment if there is a bug?"

[1]

EDM's approach to E2E :

Linked build jobs:

E2E tests are triggered after every successful infrastructure build

Slow feedback loop and code is still deployed

Need to rollback to recover

Experimentation with mock E2E

For our latest SPA UI, we are experimenting with mocking APIs used by the UI

- Expected JSON response from APIs are defined in the UI repository.
- Before running the tests, a container is started to serve these JSON response for specific path
- E2E tests are triggered

This allow us to run the tests as part of our CI pipeline

Other tools

Automated dependency update : Dependabot

Dependabot will create pull requests when it detect new version of your dependencies

If you have good tests, this is a life saver

Code review and code analytics : Codacy

Dependency vulnerability management : Snyk

How much does it cost ?

Travis CI, Dependabot, Codacy, Snyk are free when you build open sourced repos

Pricing for private projects:

- Travis CI : $129/month for 2 concurrent jobs
- Dependabot : $15/month for up to 5 private repos
- Codacy : Free for teams up to 4 users. $15/user/month after that
- Snyk : Free for up to 200 tests on private projects. $599/month after that

Github ?

Questions ?

http://bit.ly/cicd-techconnect-2019