DevOps Writeup

# 1.0 Introduction

## 1.1 Lifecyclus

The lifecycle of an application:

* First we build the application
* What if the developer is done with development?
* What if the application runs in production
* What is in between, before and after?

Diagram

Description automatically generated

Flow of an application = From planning the building to planning for when the application is live

* What steps are in between?
* What has to happen during these steps

Diagram

Description automatically generated

## 1.2 Application flow

The flow of an application goes as followed:

* ~~Business requirement~~
* ~~Making of user stories~~
* ~~Creating wireframes and prototypes~~
* Development – creating new feature
* Development – Create Unit tests for new code
* Development – Install dependencies
* Ops - running test environment

QA – Responsible for tests

Ops – responsible for deploying application environment

Developer – Responsible for creating the application

Take feedback to new cycle

Wall of confused: A scenario in which development and Devops have fundamentally different thoughts

Developers wanting change while Devops want stability

Difference between functional and non-functional requirements:

* Functional requirements are:
  + Specification features
  + Prioritization of features
* Use automatic tests:
  + Automated functional tests
  + Integrated build process
* Aimed at business usage:
  + Bugs
  + evolution
* Non functional requirements are
  + Useful troubleshooting logs
  + Deployment
  + Undo-redo operations
* Non functional tests:
  + Automatic technical tests
  + Integrated build process
  + Continuous development
* Technical operation
  + performance and technical problems

Functional requirements describe what the system should do

Non functional requirements describe how the system works

# 2.0 Agile & Devops, The three ways

## 2.1 Agile & Devops

There are different concepts to take note of:

* Agile: Agile is that there is no one way to do something
* Waterfall: Maps out a project into distinct phases with each a beginning phase and an end phase
* Devops: Increases the organizations ability to deliver applications and services at high velocity
* Pipeline: Set of automated processes and tools used by developers s and Devops
* Flow: An end-to-end manufacturing chan of software to run lines of code in production
* Feedback: Modification or control of a process or system
* Swarming: When many team members as possible work on the same priority item
* Technical debt: The accumulation of sub optimal technical decision making made over the lifetime of an application

Why is Agile and Devops not the same:

* Devops is a practice bringing development and operations teams together
* Agile focusses on collaboration, customer feedback and small rapid release
* Devops focuses on constant testing and delivery
* Agile focusses on constant changes

A picture containing text, clipart

Description automatically generated

Different focus points within Devops for:

* developers: end user performance, analytics, code quality and code errors
* operations engineers: Availability, performance, user complains and performance analytics

## 2.2 The Three ways

The three ways:

* The first way: identify the component and make business idea a reality – Principles of flow
* The second way: Continuing to monitor, test and handle feedback  
  - Principles of feedback
* The third way: Continuous improvement, seeking to improve every process – Principles of continuous improvement

Principals of flow:

* work flowing from left to right
* Passing work through the different teams
* Each phase in Devops focusses on closing the loop between development and operations

Principals of feedback:

* Goes from right to left
* Providing feedback in a fast consistent way
* Quickly address and identify problems

Principals of continuous learning:

* Using the feedback to improve the process
* Implementing changes might be challenging in the organizational culture we work

A testing pyramid is a concept that groups software tests into three different categories:

* UI/exploratory tests
* Integration tests
* Unit tests

A positive is that you will have a nice way to test a negative is that it will take a lot of time

# 3.0 Source control

General concept:

* Source control: Practice of tracking and managing changes to code
* Git: Freely distributed source control system
* GitHub: Web based version-control platform for developers
* Flow: End to end manufacturing chan of software
* (Feature) Branch: Unique set of code changes with an unique name
* Merge: A way of putting a forked history back together again
* Merge conflict: When different changes are made on the same lines of a file
* Pull request: Lets you tell others about your changes you’ve pushed
* Commit: An individual change to a file
* Deploy: This is when you launch the code stored on a server of your own
* Main: The one where all changes eventually go back to
* Markdown (md): Plaintext format of writing structured documents
* Single source of truth: Storing all code, configuration and other digital assets where everyone can access them

Why is source control important and for what is it used:

* To maintain a single source of truth
* To facilitate collaboration
* To accelerate release velocity
* It allows multiple developers at once
* It stores your data so when a drive failure occurs your code isn’t gone

Basic GIT commands:

* Create a new branch  
  | git branch new  
  | git checkout -b new
* Move to other branch  
  | git checkout new
* See all branches  
  | git branch -a
* Push new branch to git  
  | git push origin new
* Pull all branches from git  
  | git fetch –all
* Merge a branch named new with the branch you are in (after merging you still need to pull)  
  | git merge new
* Deleting a branch  
  | git branch -d new

A good git commit message contains clear feedback of changes

The git flow goes as followed:

* A developer creates a feature in a separate branch
* Developer pushed the branch to central repo
* Developer creates a pull request via Github
* The rest of the team goes over the code, gives feedback etc
* The responsible of the repo merges the changes to the main branch

A branching strategy defines how a team uses branches to achieve a level of concurrent development

Github flow is how we work with GitHub  
Diagram

Description automatically generated

We use main for deployment and develop to merge our changes

# 4.0 Flows -CI & CD concepts

The concepts:

* Pipeline: Ensures all code in version control is built and tested
* Continuous integration: Automating the integration of code changes from multiple contributors
* Continuous delivery: Ability to get changes of all types
* Build: The stage where the application is compiled
* Artifact: A by-product of the software development process
* Compiling code: transform source code to machine code
* build once deploy anywhere: an agile approach to creating new applications
* vendor agnostic: an organization that is open to all viable and established solutions
* vendor centric:
* Jenkins: Used to reliably build, test and deploy software

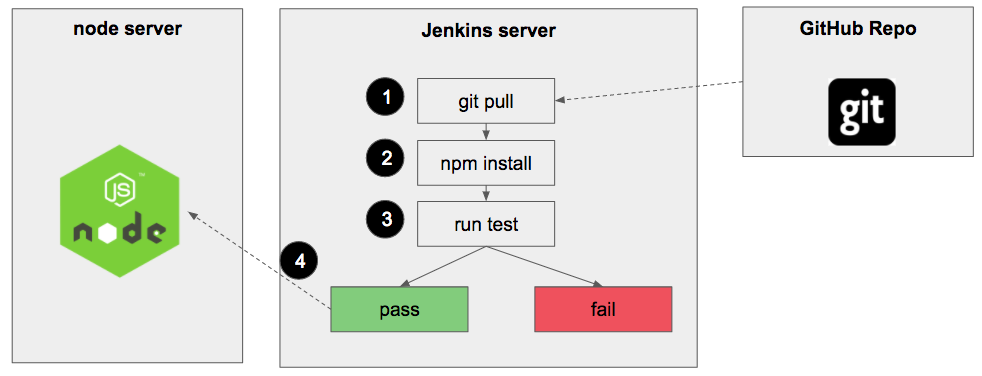
Phases in continuous integration:

* Unit test
* Code analysis
* test coverage

Stages in CI pipeline:

* Build
* Test
* Release
* Deploy

Global view on a CI pipeline in NodeJS



Different code testing:

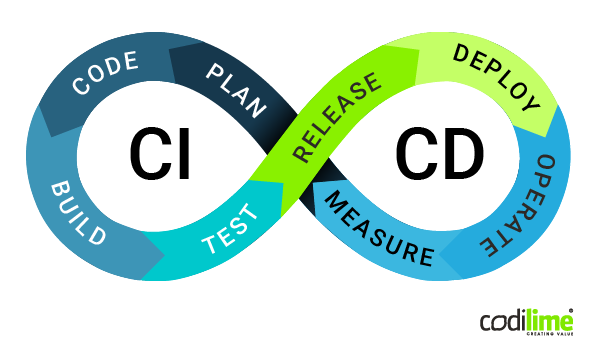
* Unit tests, automized tests

Reusing artifacts is essential for reusing your code and improving the build process

Phases in continuous delivery:

* Build
* Unit test
* Code Quality analysis
* Integration test
* Acceptance test
* Store versioned Binary Artifact

Phases of CD pipeline



Global view on a CD pipeline in NodeJS

Diagram

Description automatically generated

CI is simpler, its about using the right tools.  
CD is more complicated and is about processes that have to happen after code is integrated

We use this to integrate code into a mainline code base

**Jenkins helps in automate building and testing systems to the integration work**

# 5.0 CI in Jenkins

Define concepts

* Pipeline: Ensures all code in version control is built and tested
* Continuous integration: Automating the integration of code changes from multiple contributors
* Continuous delivery: Ability to get changes of all types
* Infrastructure as code: Being able to reproduce and/or restore a full infrastructure based on recipes and automation
* Stage view: Progress of each stage
* Build attempt: Jenkins tries to build the application
* Groovy: A language that offers the ability to do what Java can
* Pipeline script:
* Stages:
* Steps
* Agents:
* Post run steps:
* Workspace:

# 6.0 DTAP & environments on demand

# 7.0 Flows & feedback – Integrated testing

# 8.0 Flows & feedback – Monitoring and reporting

# 9.0 Integration exercise