



CI/CD series

# CI/CD with Jenkins





# CI/CD with Jenkins



# Topics (1)

1. Concept of Continuous Integration
2. Concept of Continuous Delivery/Deployment
3. Practices of Continuous Integration
4. Build your CI/CD system with Jenkins
5. Design and build your pipeline
6. Workshop



# Topics (2)

1. Pipeline as a Code with Jenkins
2. Best practice with declarative pipeline
3. Refactoring to shared libraries
4. Versioning and life cycle
5. Workshop



**[https://github.com/up1/  
course-ci-cd-with-jenkins](https://github.com/up1/course-ci-cd-with-jenkins)**



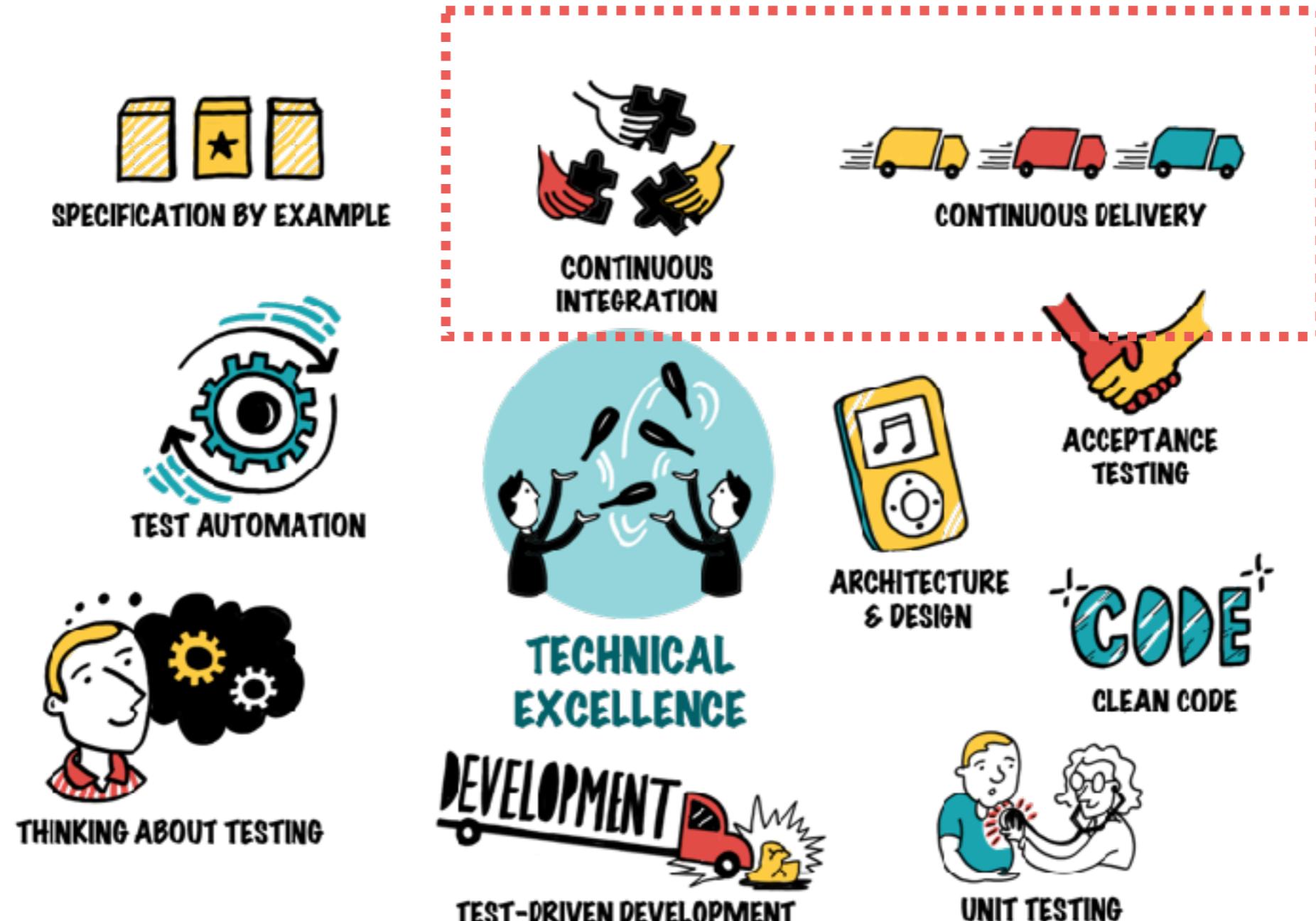
# Software development



# Continuous Integration ?



# Technical Excellence



<http://less.works>

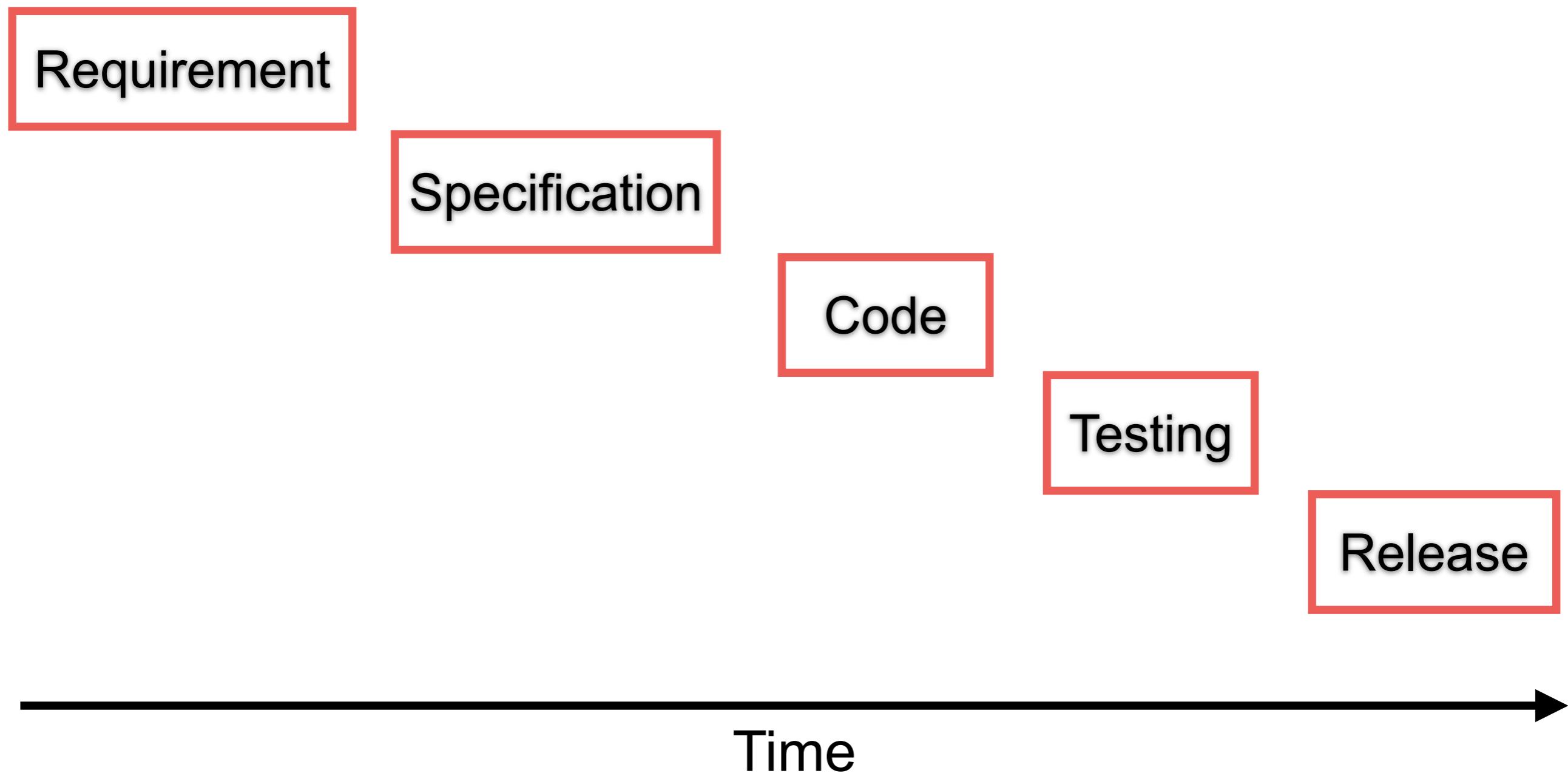
<https://less.works/less/technical-excellence>



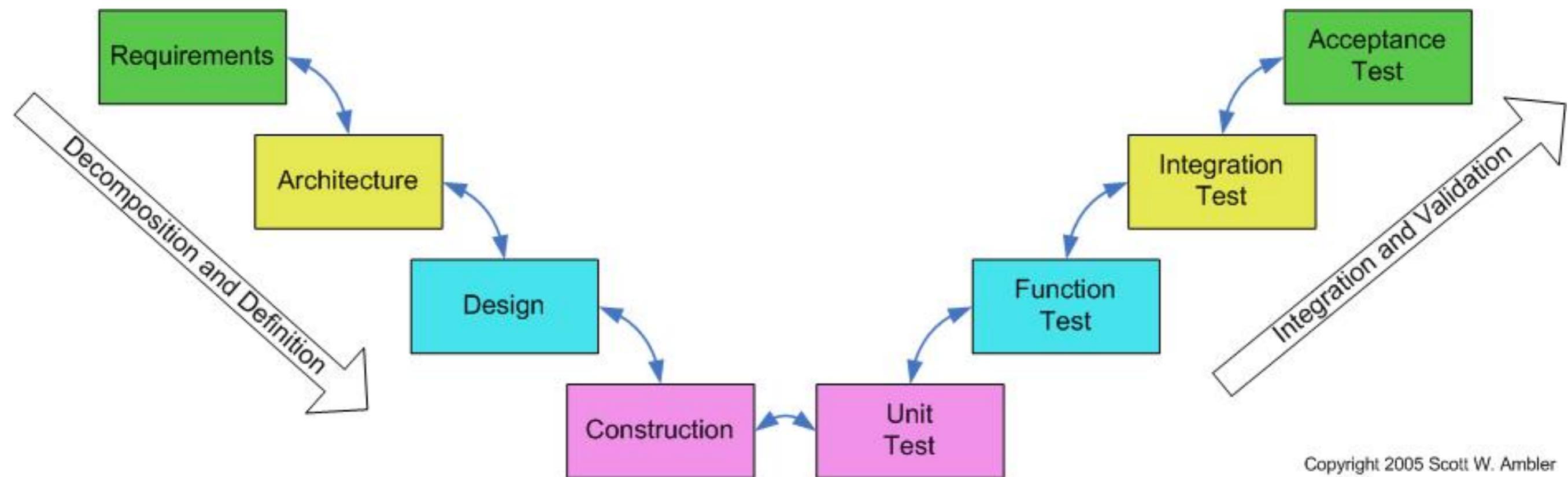
# Why CI/CD ?



# Software Delivery Process



# V Model

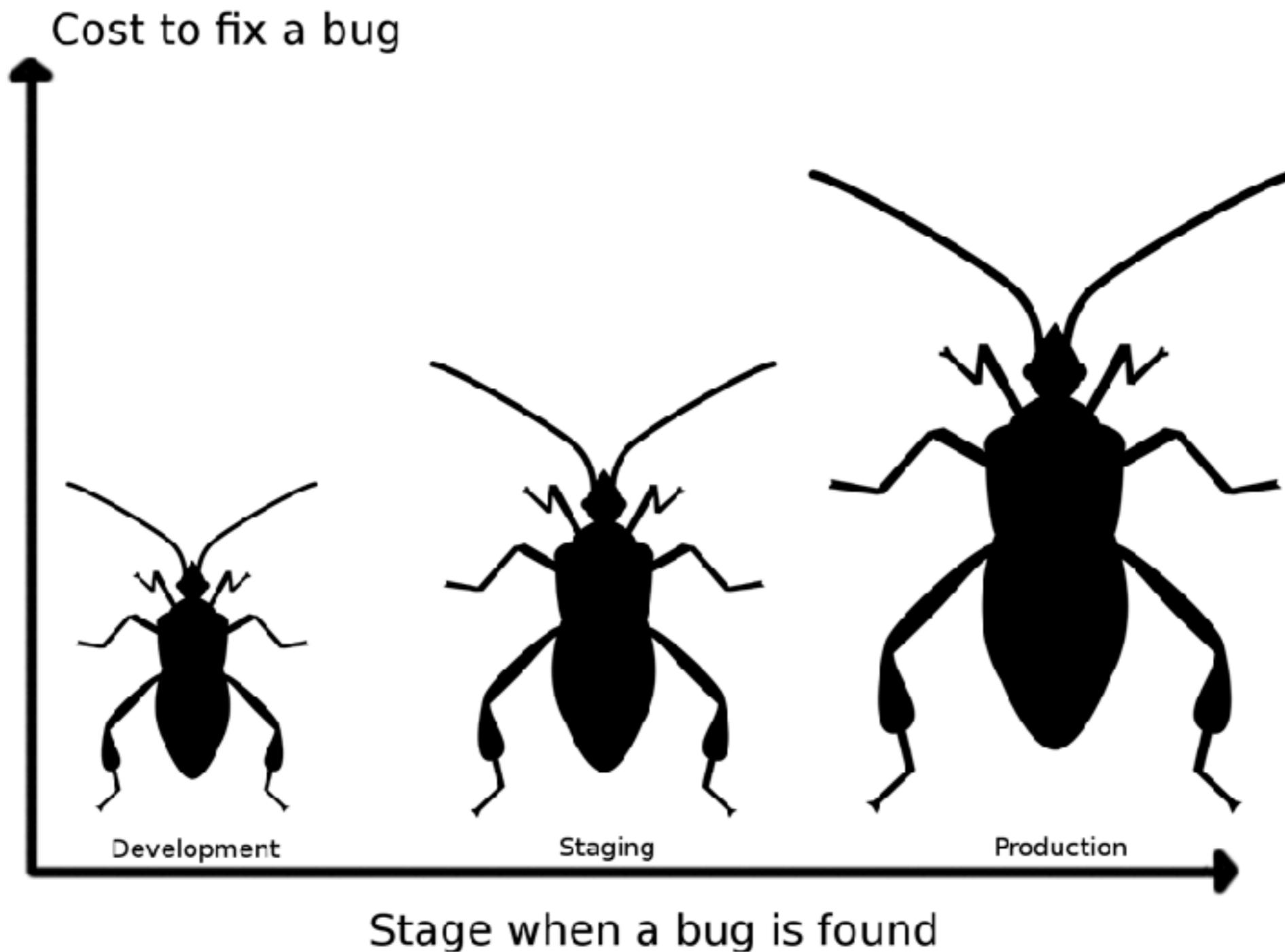


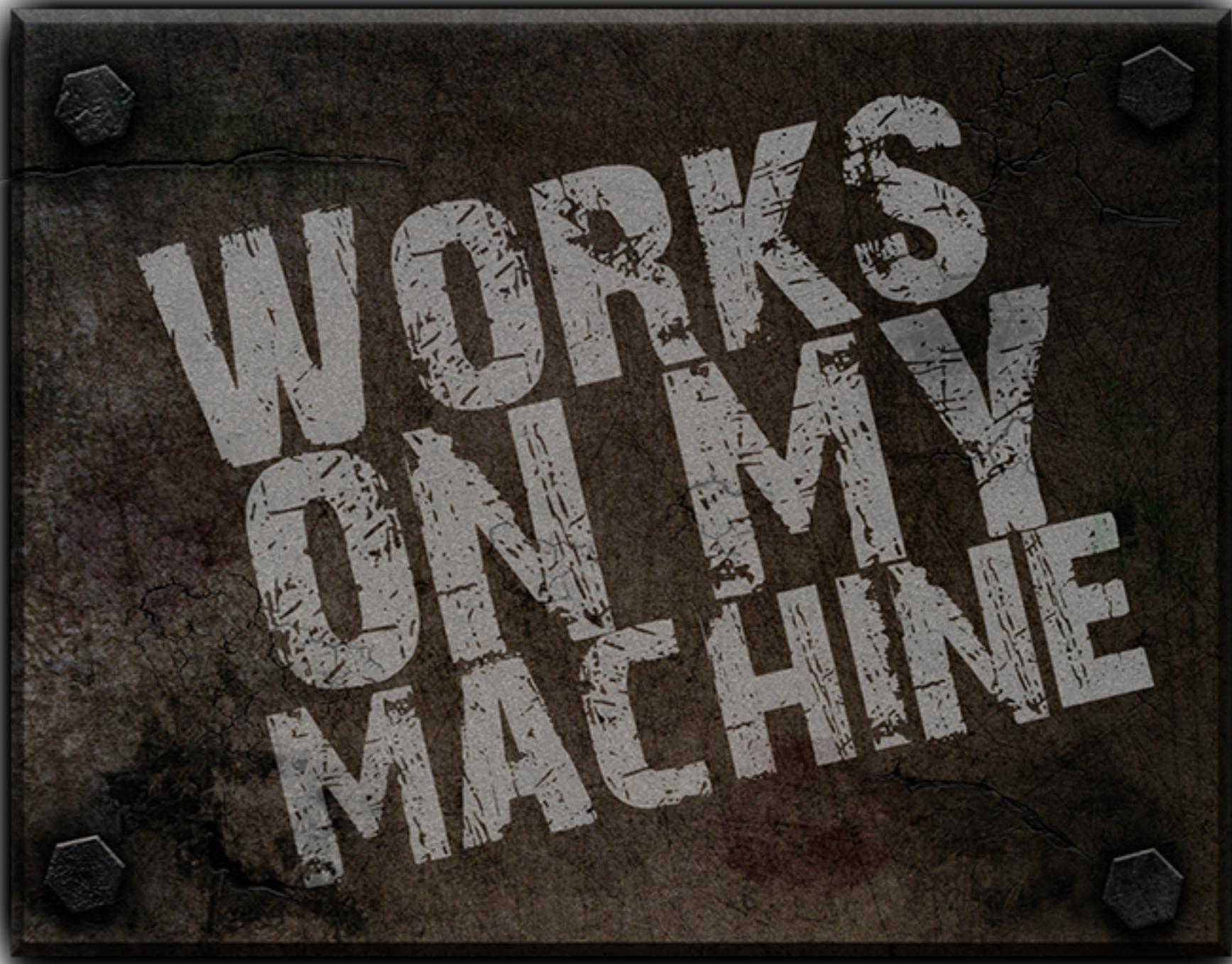
# The cost of integration

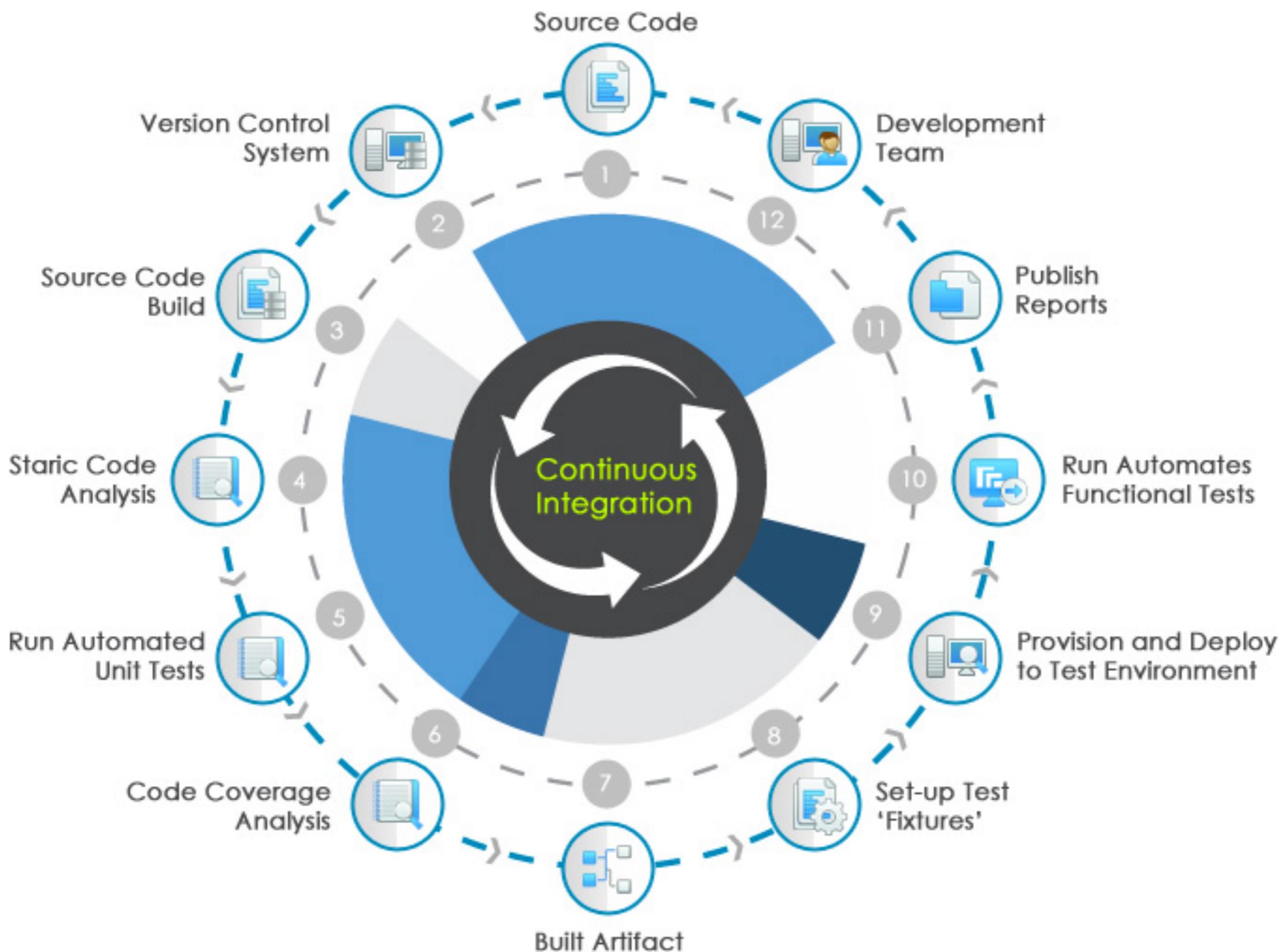
1. Merging the code
2. Duplicate changes
3. Test again again !!
4. Fixing bugs
5. Impact on stability



# The cost of integration









Jenkins

Bamboo



TeamCity

> go<sup>TM</sup>



Hudson





Jenkins

Bamboo

CI is about what people do  
not about what tools they use



Visual Studio



Team Foundation Server

Hudson



Travis

wercker

circleci



# Continuous Integration

Discipline to integrate frequently



# Continuous Integration

Strive to make **small change**



# Continuous Integration

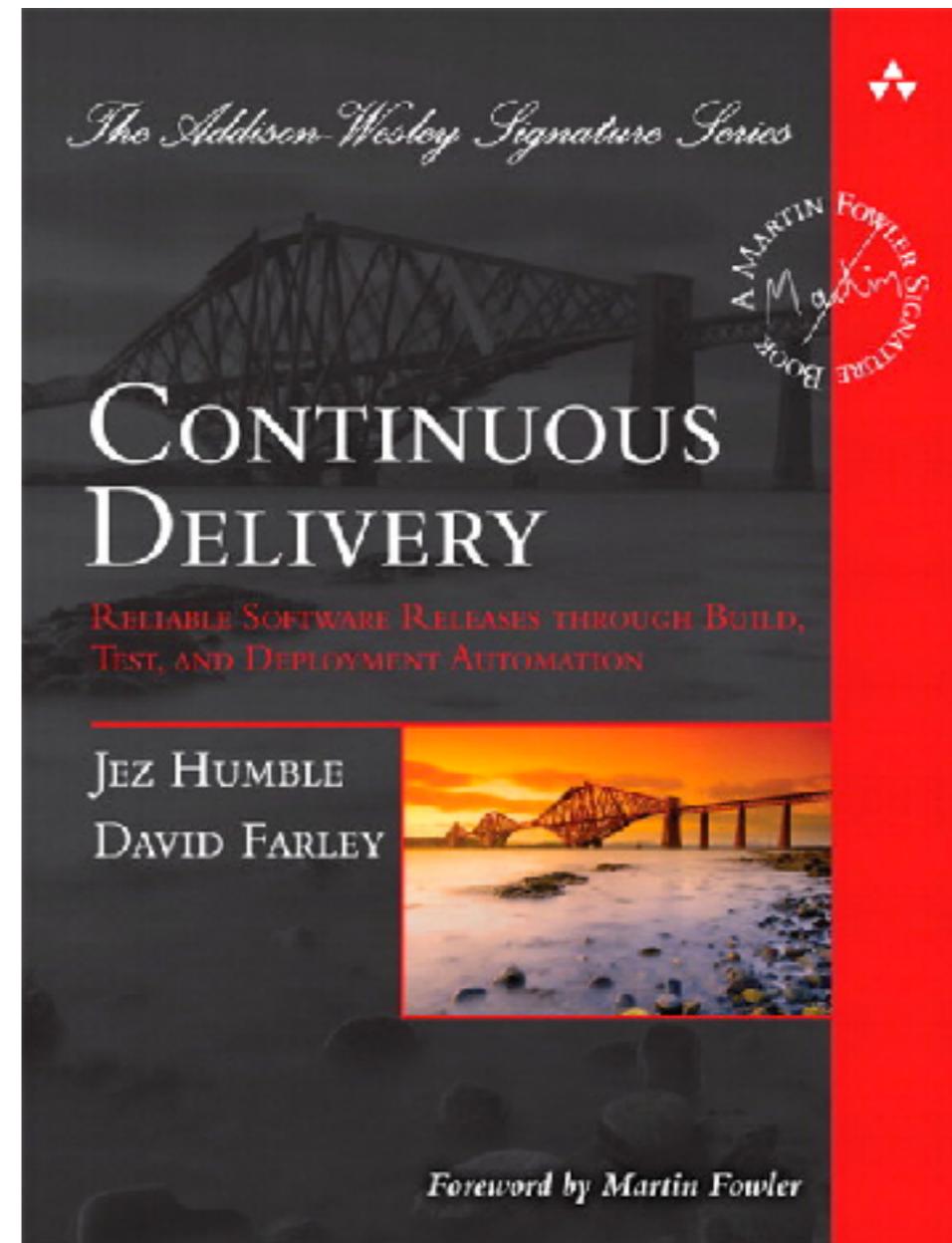
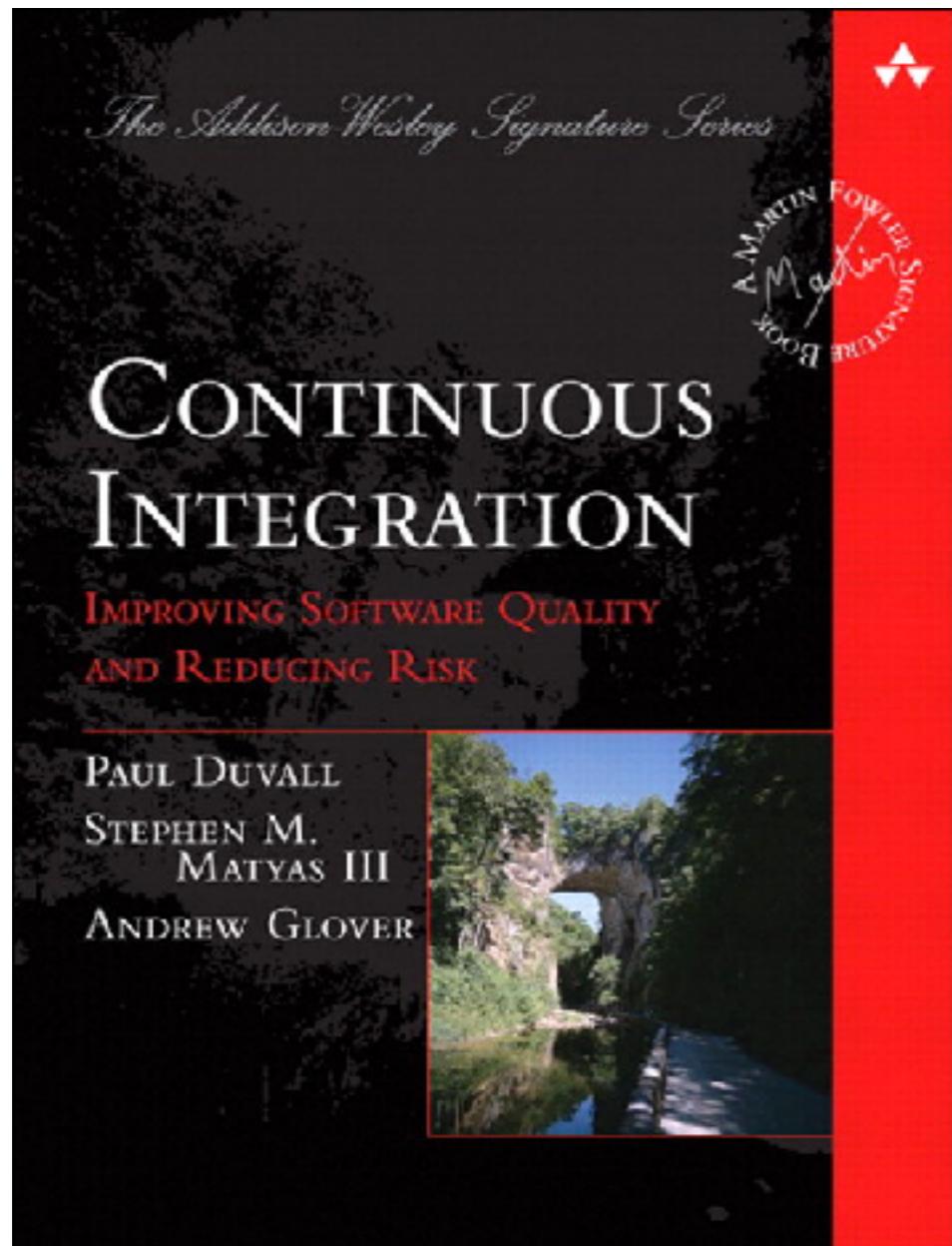
Strive for **fast feedback**



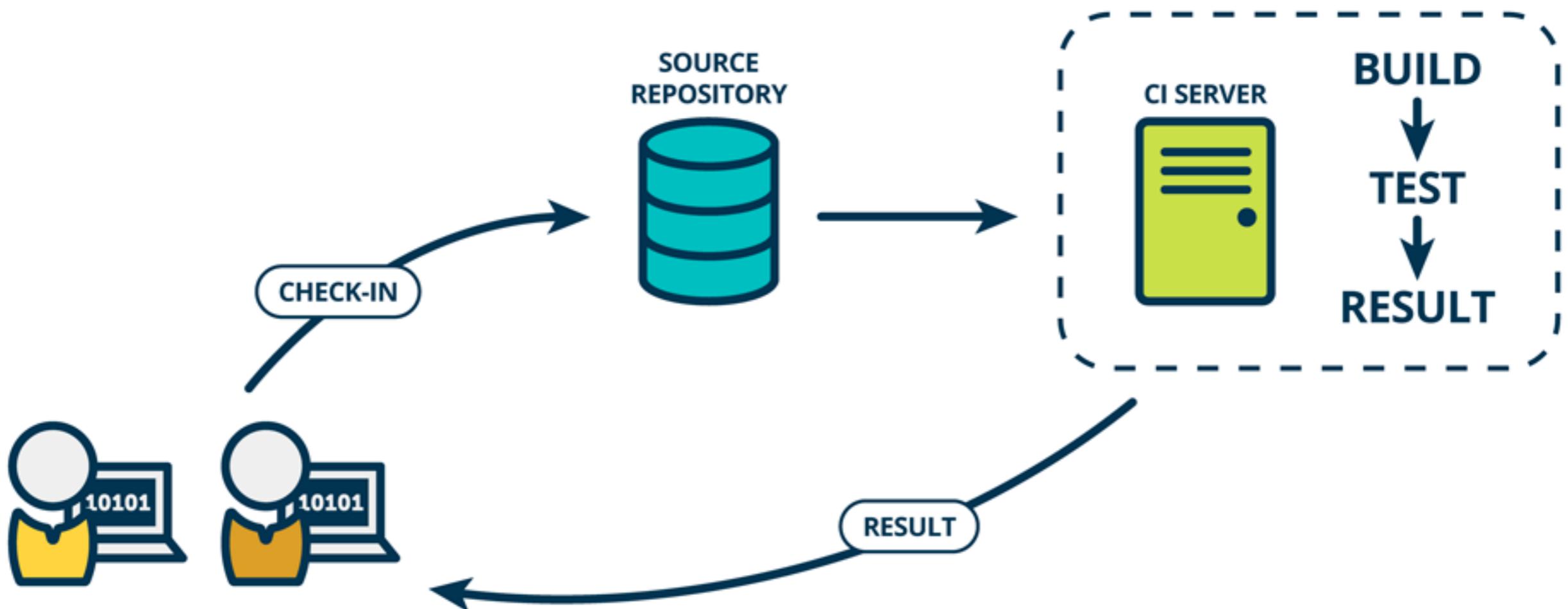
# **Practices of Continuous Integration**



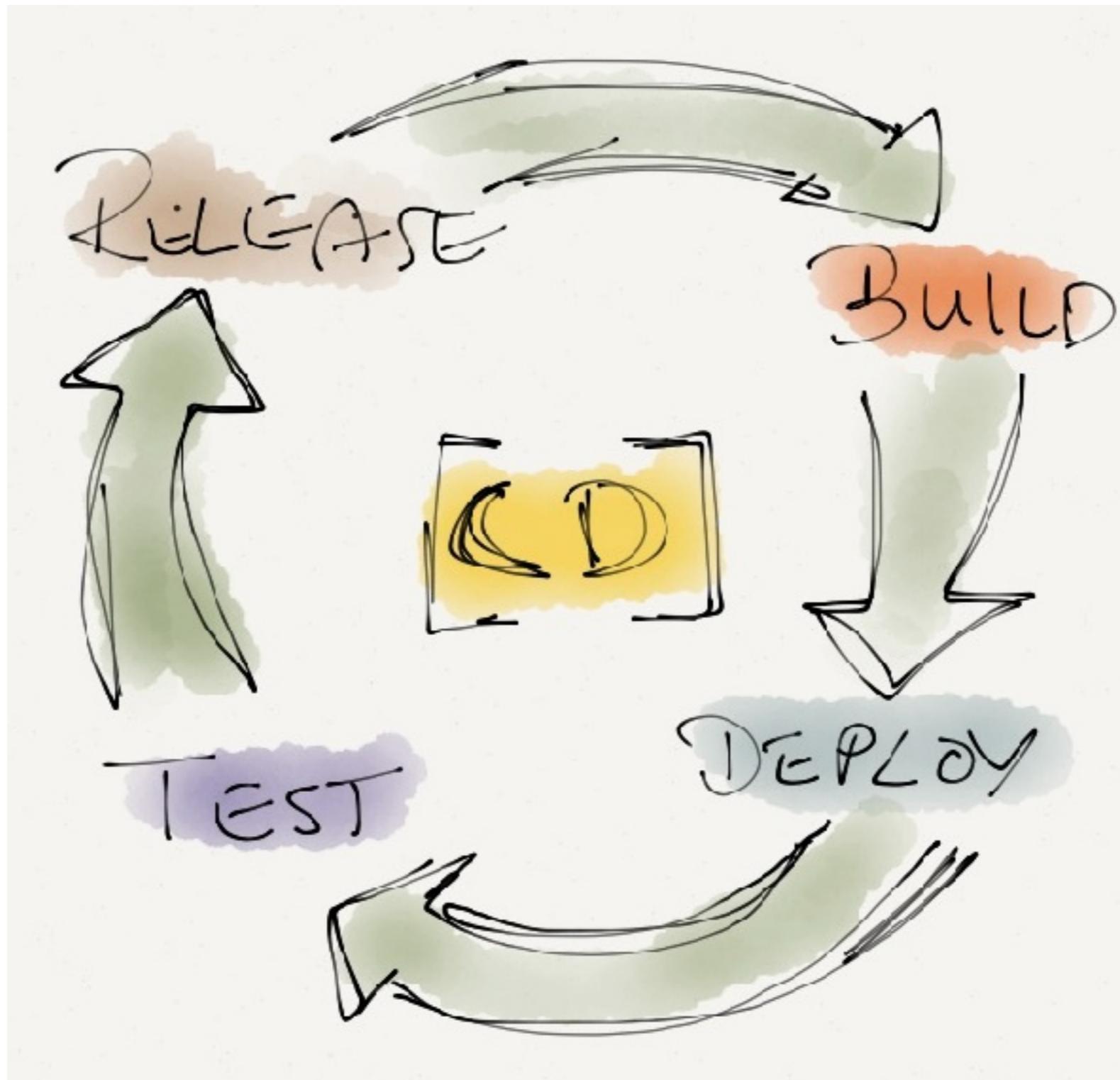
# Improve quality and reduce risk



# Continuous Integration



# CD ?



# CD ?

## CONTINUOUS DELIVERY



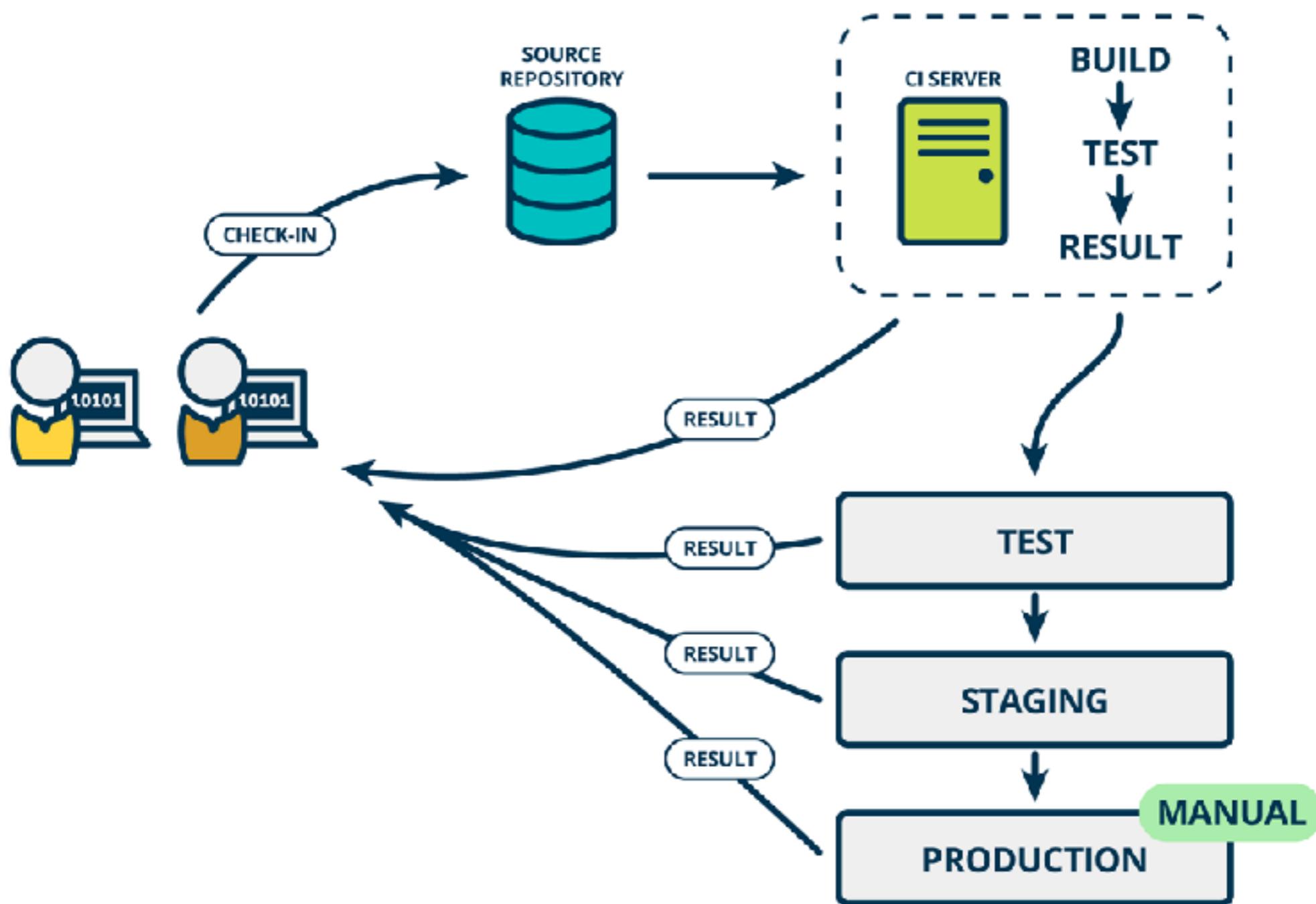
## CONTINUOUS DEPLOYMENT



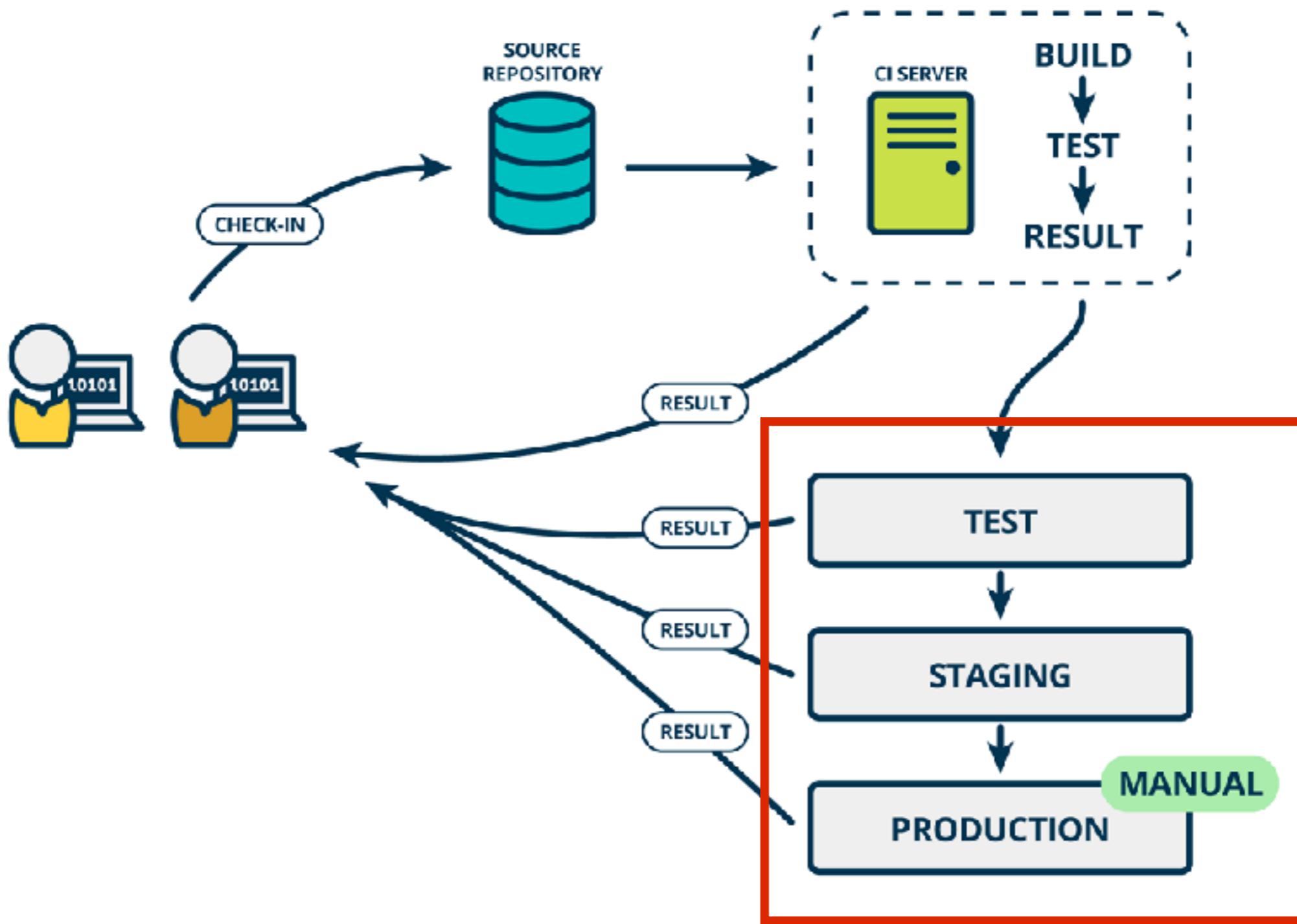
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



# Continuous Delivery



# Rise of DevOps



# **Continuous Integration**

## **is a Software development practices**



# Practice 1

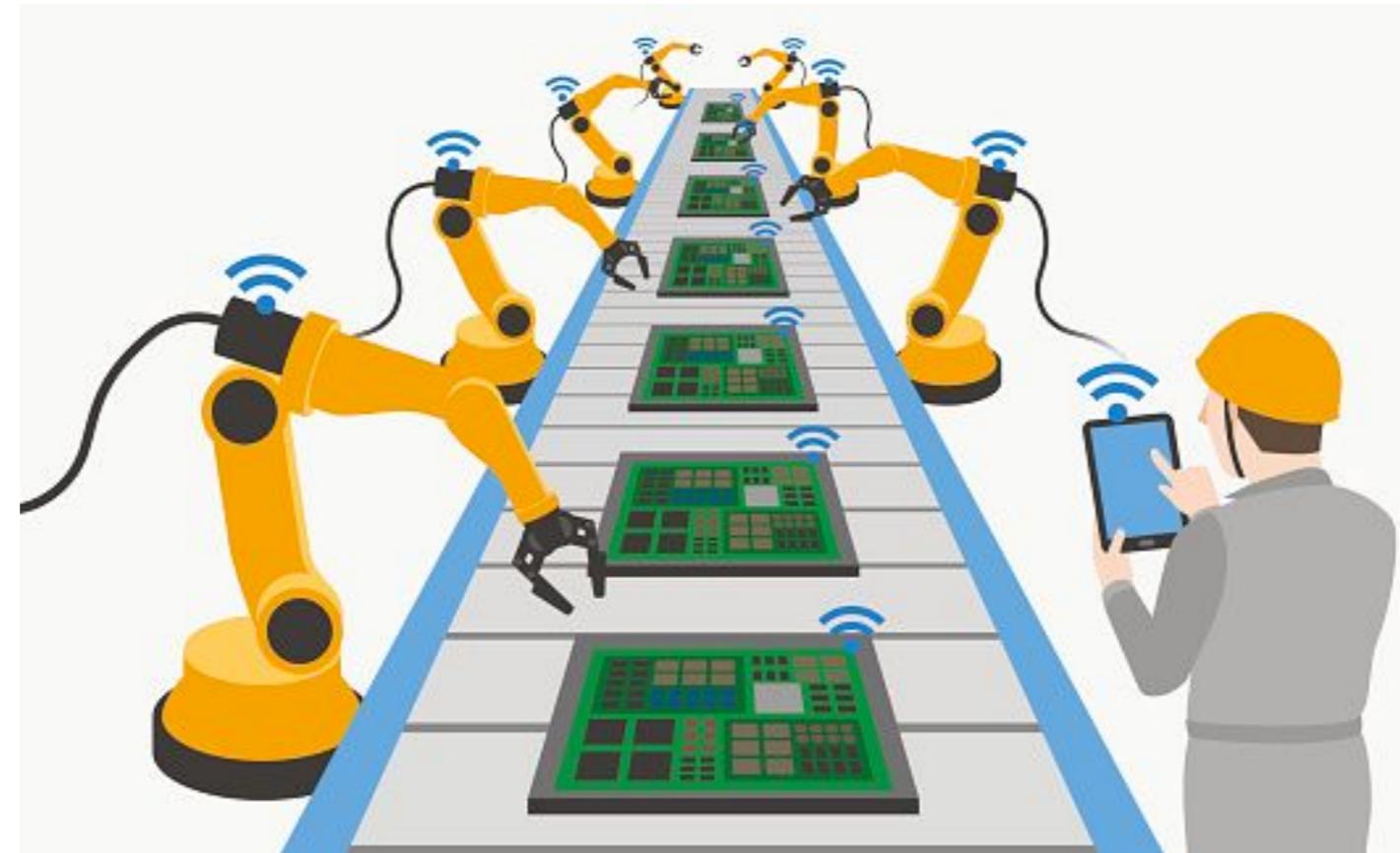
Maintain a single source repository

In general, you should store in source control  
everything you need to build anything



# Practice 2

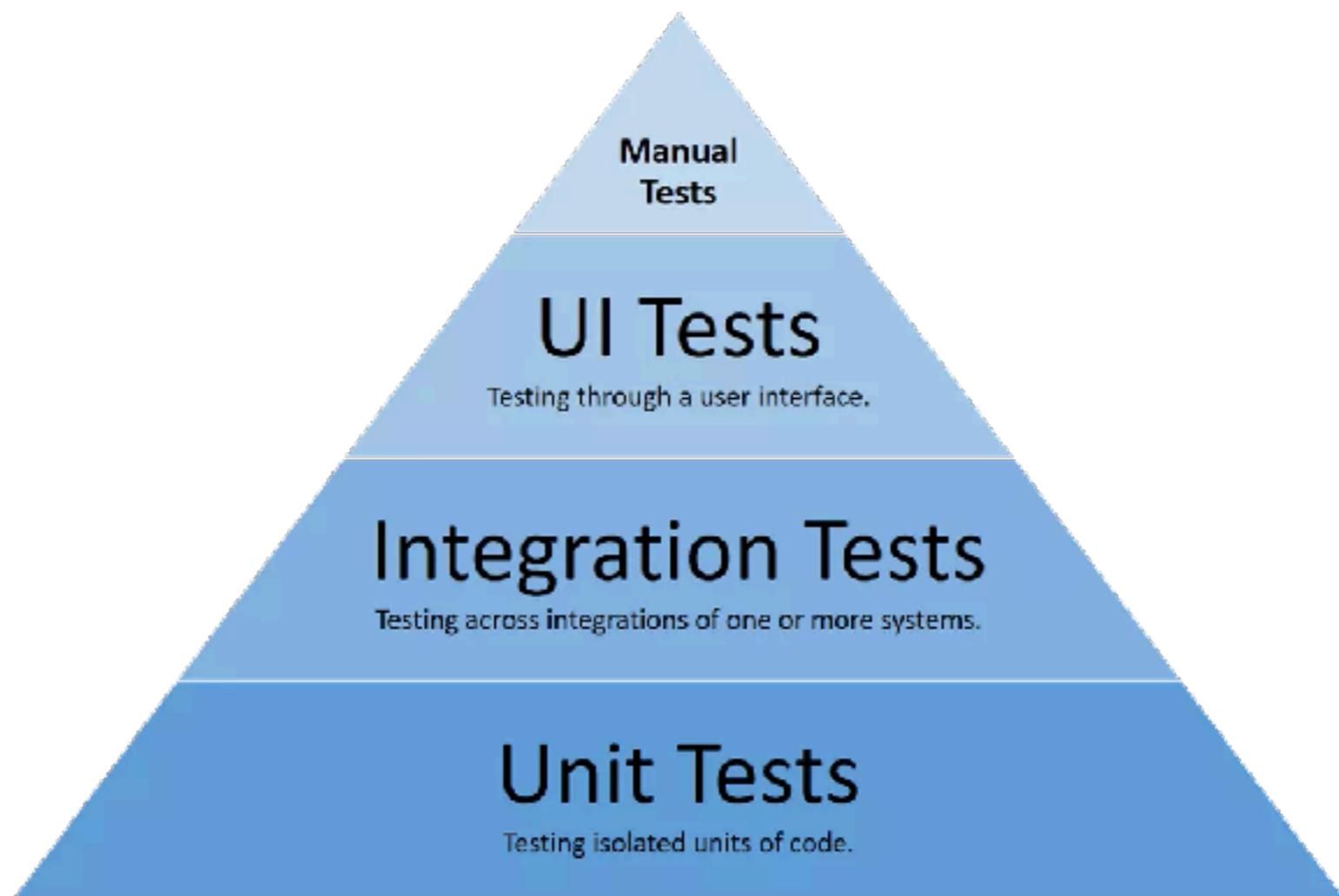
Automated the build  
Automated environment for builds



# Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**

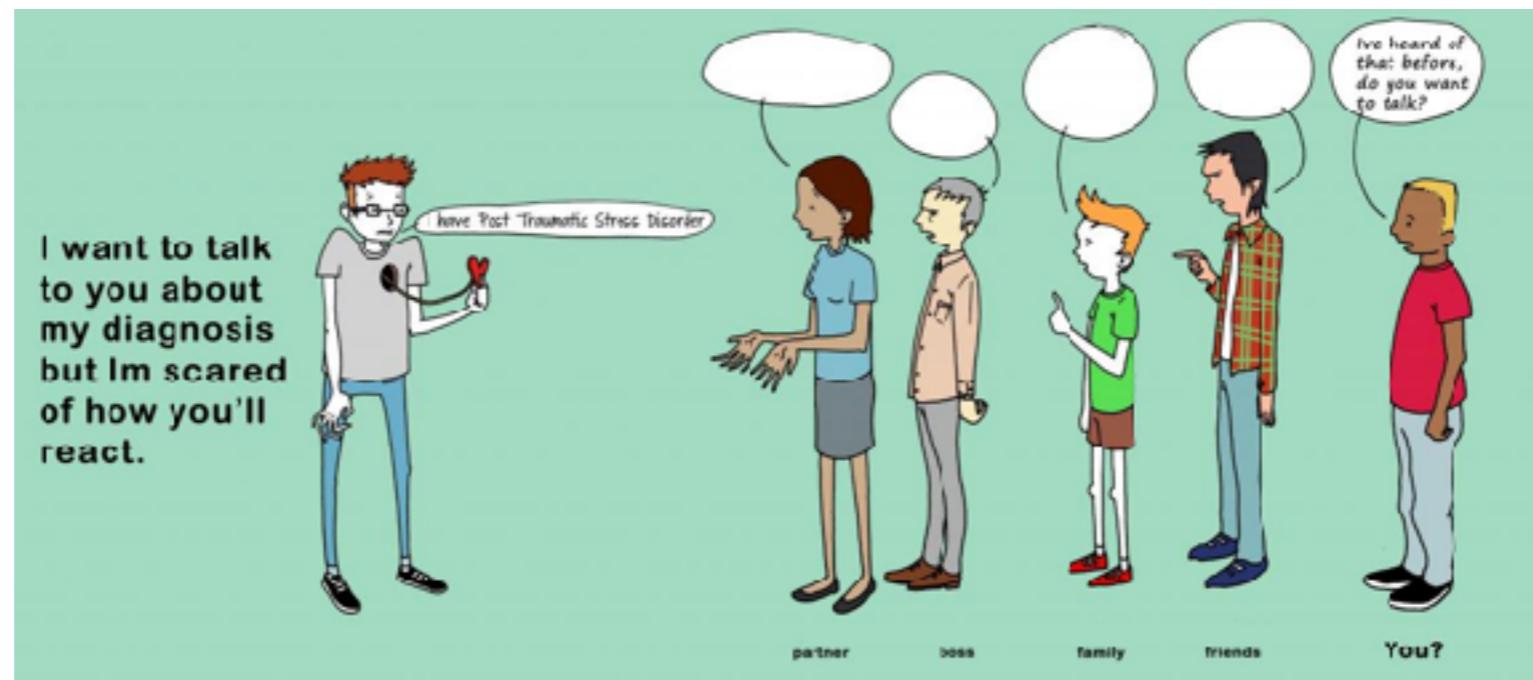


# Practice 4

**Everyone commits to the mainline everyday**

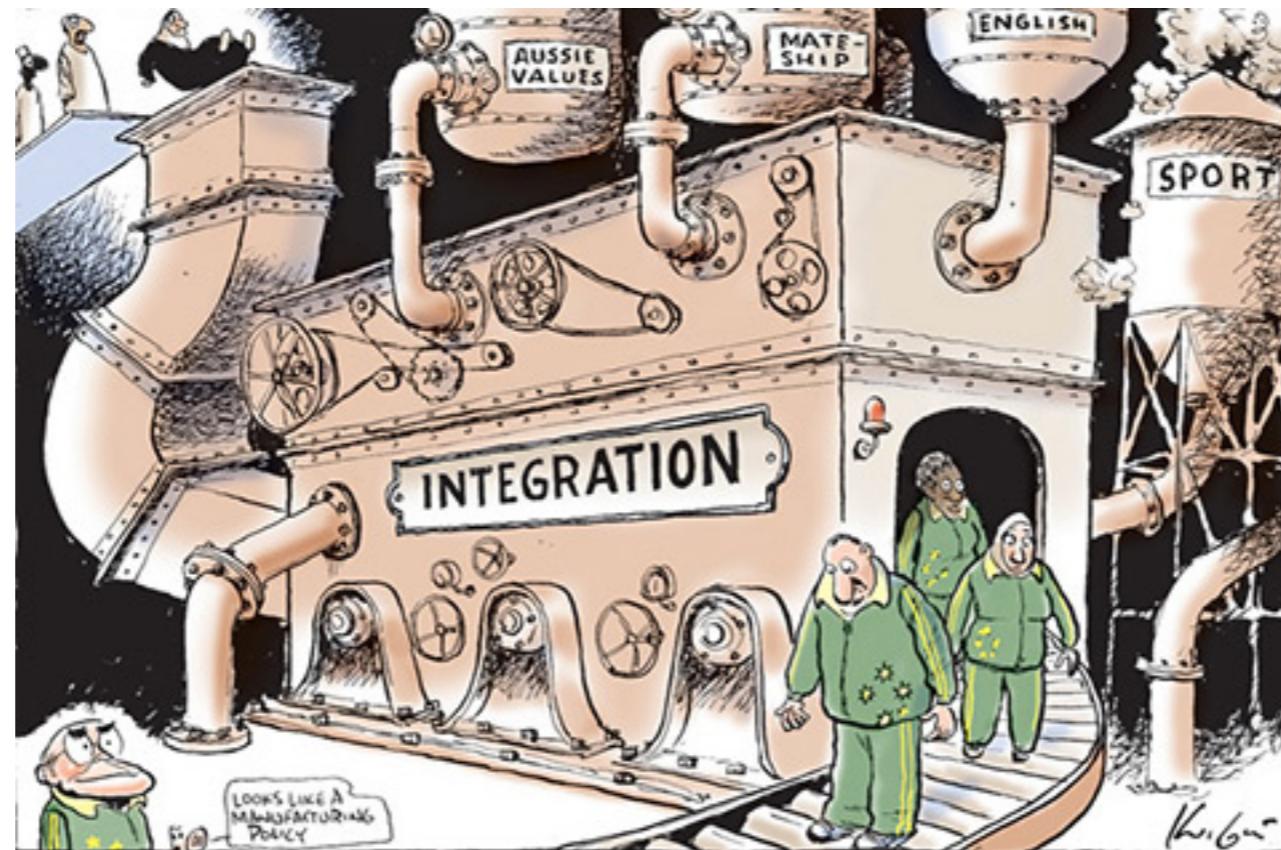
**Integration is about communication**

**Integration allows developers to tell other developers**



# Practice 5

Every commits should build the mainline on an  
**Integration machine**



# Nightly build is not enough for Continuous Integration



# Practice 6

**Fix broken builds immediately**

**“Nobody has a higher priority task than  
fixing the build”**



# Practice 7

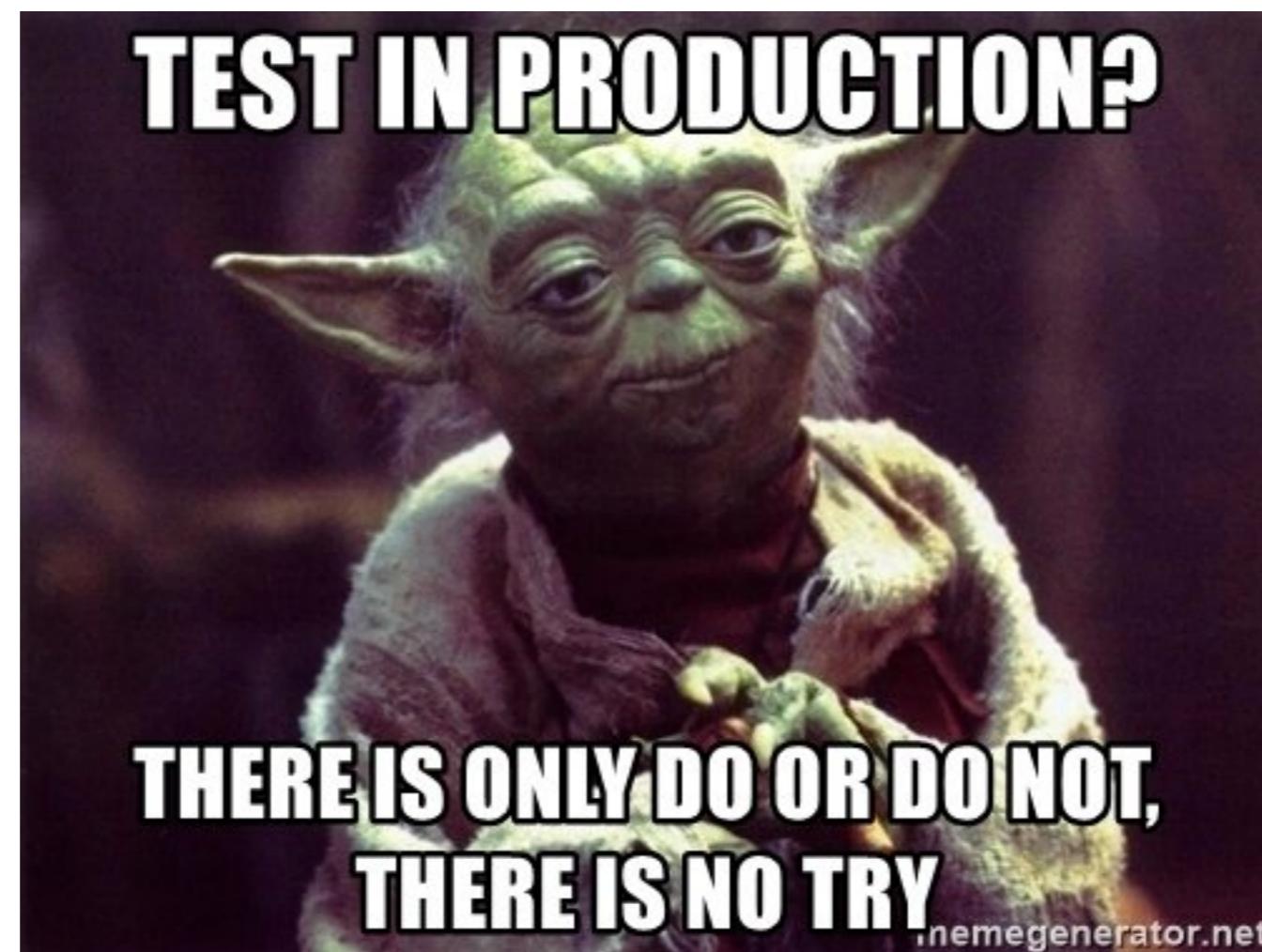
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

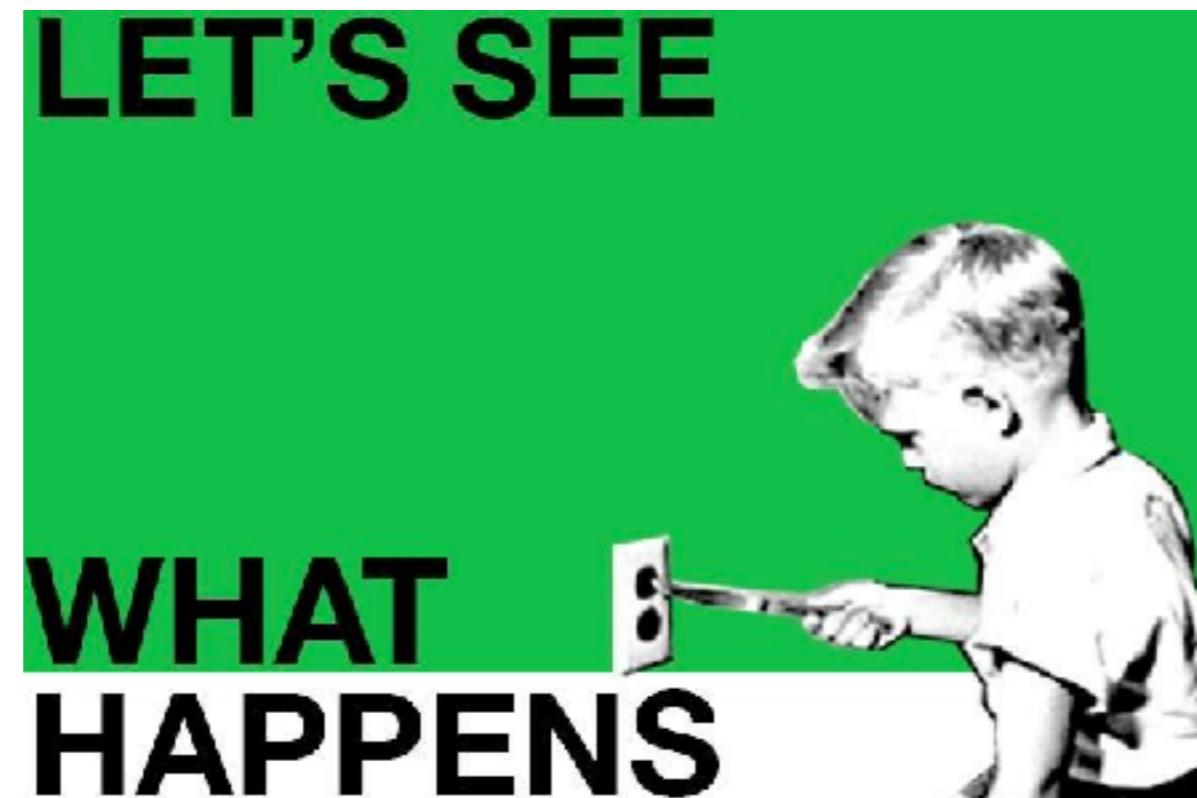
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



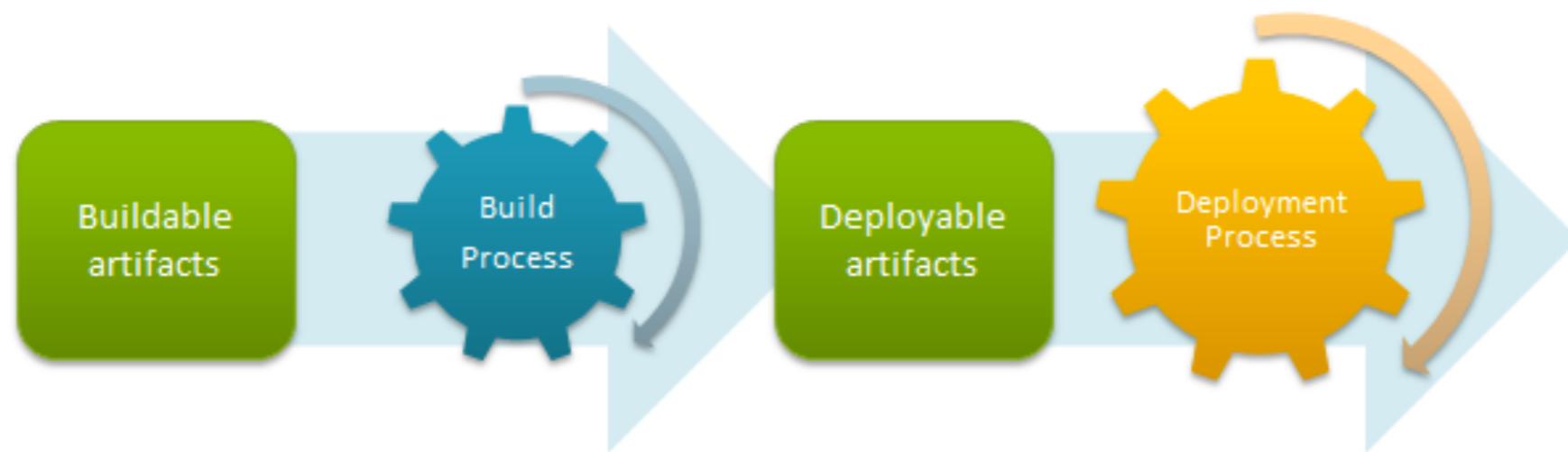
# Practice 10

**Everyone** can see what's happening  
**Easier** to see the state of the system and changes  
Show the good information



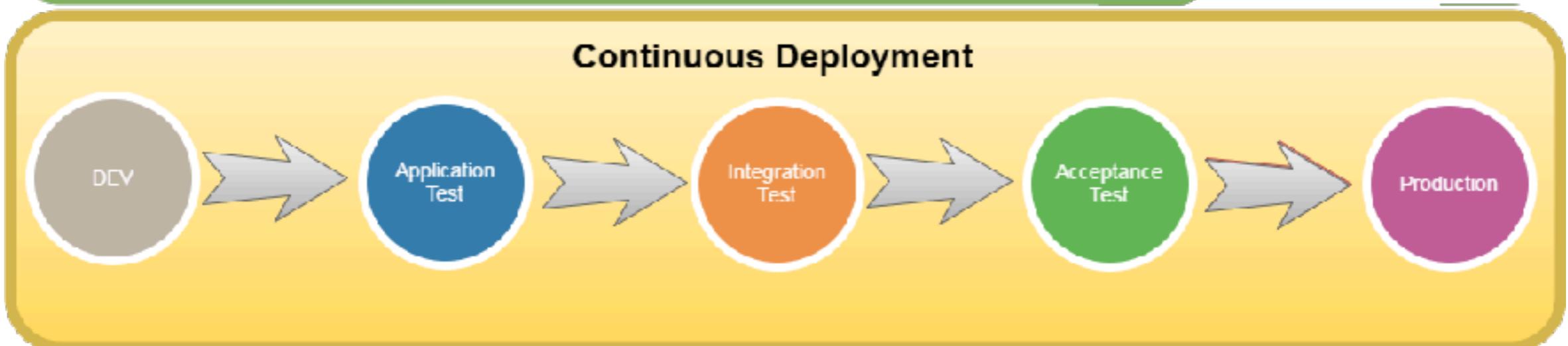
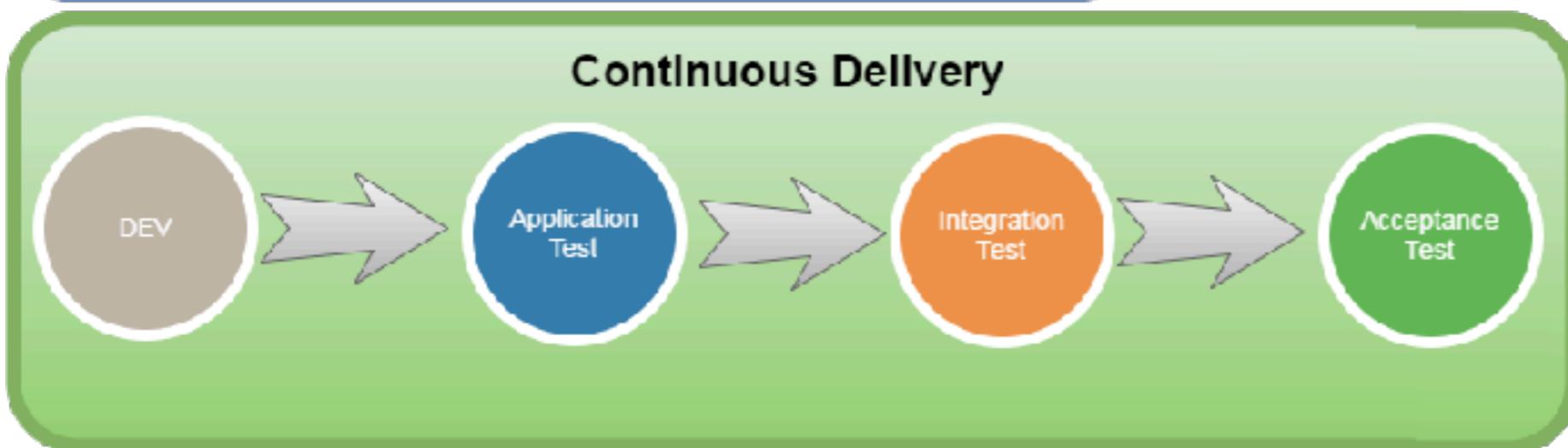
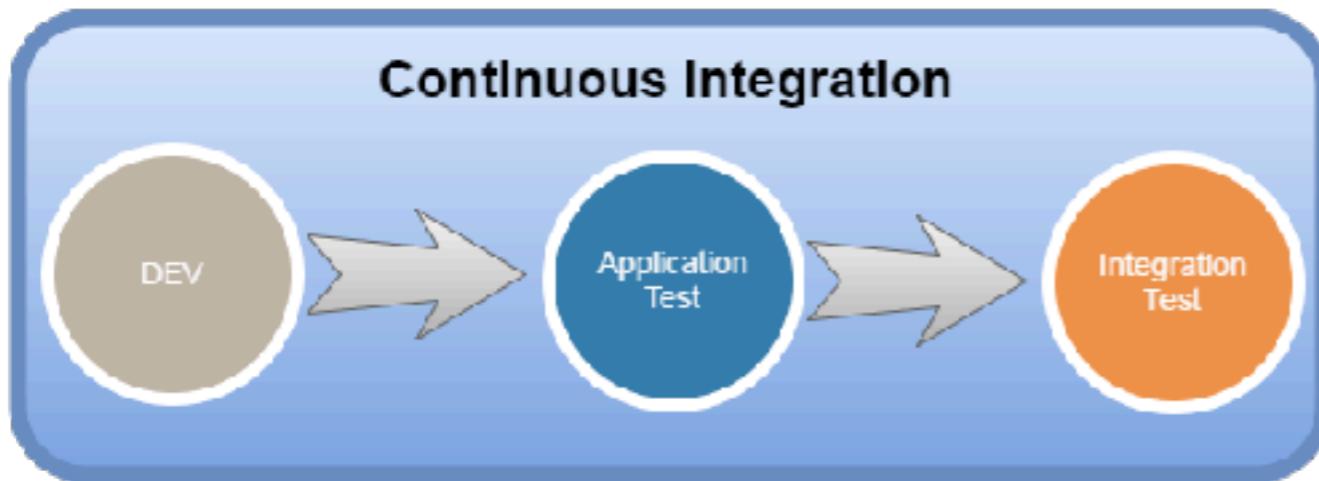
# Practice 11

## Automated deployment



# Continuous Delivery





# Let's start with Jenkins



Application and framework to manage and monitor  
the executable of **repeated tasks**



# Jenkins

<https://jenkins.io/>



# Why Jenkins ?

Easy !!

Extensible

Scalable

Opensource

Large community

**Lot of plugins**



# Who use Jenkins ?

We thank the following organizations for their major commitments to support the Jenkins project.



Microsoft



redhat.

We thank the following organizations for their support of the Jenkins project through free and/or open source licensing programs.

Atlassian

Datadog

JFrog

Mac Cloud

PagerDuty

XMission

<https://jenkins.io/>



# Hardware requirements

For Jenkins server

RAM +2GB

More CPU

More Disk



# Installation



# Jenkins in containers

Apache Tomcat

Jetty

JBoss

Websphere

WebLogic

Glassfish



# Download Jenkins



The Jenkins website homepage. At the top is a dark navigation bar with links: Blog, Documentation, Plugins, Use-cases ▾, Participate, Sub-projects ▾, and Resources ▾. The main title "Jenkins" is in large, bold, black font. Below it is the tagline "Build great things at any scale". A descriptive paragraph follows: "The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project." At the bottom right are two buttons: "Documentation" (white background) and "Download" (red background).

<https://jenkins.io/>



# Use Long Term Support (LTS)

## Getting started with Jenkins

The Jenkins project produces two release lines, LTS and weekly. Depending on your organization's needs, one may be preferred over the other.

Both release lines are distributed as `.war` files, native packages, installers, and Docker containers.

### Long-term Support (LTS)

LTS (Long-Term Support) releases are chosen every 12 weeks from the stream of regular releases as the stable release for that time period. [Learn more...](#)

[Changelog](#) | [Upgrade Guide](#) | [Past Releases](#)

[Deploy Jenkins 2.46.3](#)

 [Deploy to Azure](#)

[Download Jenkins 2.46.3 for:](#)

Docker

FreeBSD

### Weekly

A new release is produced weekly to deliver bug fixes and features to users and plugin developers.

[Changelog](#) | [Past Releases](#)

[Download Jenkins 2.65 for:](#)

Arch Linux

Docker

FreeBSD

Gentoo



# Start Jenkins

\$java -jar jenkins.war

Default port of server is **8080**

```
org.eclipse.jetty.server.AbstractConnector doStart
ector@3e2fc448{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}
org.eclipse.jetty.server.Server doStart
```

```
winstone.Logger logInternal
Engine v4.0 running: controlPort=disabled
jenkins.InitReactorRunner$1 onAttained
:ion
jenkins.InitReactorRunner$1 onAttained
jenkins.InitReactorRunner$1 onAttained
```



# Welcome to Jenkins

Jenkins  somkiat | log out

ENABLE AUTO REFRESH

New Item 

People 

Build History 

Manage Jenkins 

My Views 

Credentials 

Build Queue  -

No builds in the queue.

Build Executor Status  -

1 Idle

2 Idle

search 

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Page generated: Jun 14, 2017 2:08:57 PM ICT [REST API](#) [Jenkins ver. 2.46.3](#)



# Finds Jenkins's plugin

The screenshot shows the Jenkins Plugins Index page. The top navigation bar includes links for Jenkins, Blog, Documentation, Plugins (which is underlined in red), Use-cases, Participate, Sub-projects, and Resources. The main content area has a blue background. On the left, there is a circular icon featuring a cartoon character wearing a red and white suit, possibly a superhero or robot, with arms crossed. To the right of the icon, the word "Plugins Index" is written in large, bold, white letters. Below that, a smaller text says "Discover the 1000+ community contributed Jenkins plugins to support building, deploying and automating any project." At the bottom right, there is a search bar with the placeholder "Find plugins..." and a magnifying glass icon.

<https://plugins.jenkins.io/>



# **Let's create pipeline with Jenkins**



# Types of Jenkins Projects

Freestyle  
project

Pipeline  
project

Multi-config

Multi-branch



# Pipeline as a Code

## Declarative Pipeline Scripted Pipeline

Structured

Better error  
reporting

Readable

<https://jenkins.io/doc/book/pipeline/>



# Create a pipeline

**Jenkinsfile** in version control  
Working in Jenkins's UI

Stages

Nodes or  
Agents

Plugins



# Basic Structure of Pipeline



```
pipeline {
    agent any
    stages {
        stage('Stage 1') {
            steps {
                echo 'Hello Stage 1!'
            }
        }
        stage('Stage 2') {
            steps {
                echo 'Hello Stage 2!'
            }
        }
        stage('Stage 3') {
            steps {
                echo 'Hello Stage 3!'
            }
        }
    }
}
```



# Basic components

## Agent

Specified the execution environment for pipeline (machine)

## Stages (stage)

Structuring pipeline into different phases

## Steps (step)

Individual commands or actions to be executed in each stage



# More Syntaxs

Environment

Post

Input

When

Parallel

Parameter

Script block

Credential

(Un)Stash



# Jenkins Core Concepts

## Stages

Groups related jobs into phases like build, test and deploy

## Nodes

Machines where Jenkins executes jobs. (Single to multi)

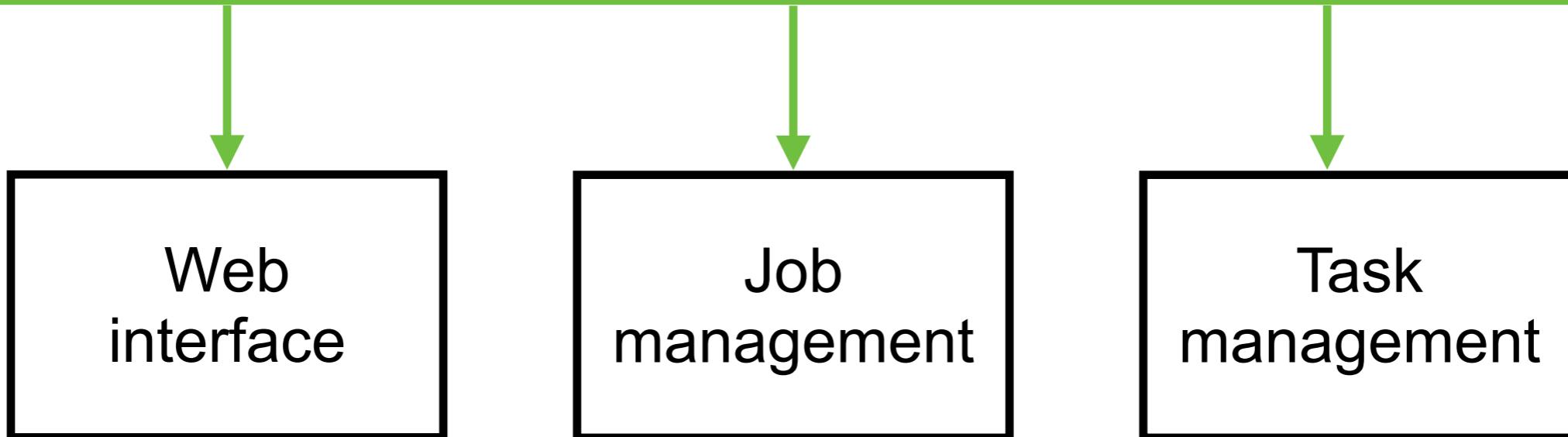
## Plugins

Extend Jenkins's functionality, working with tools and technology

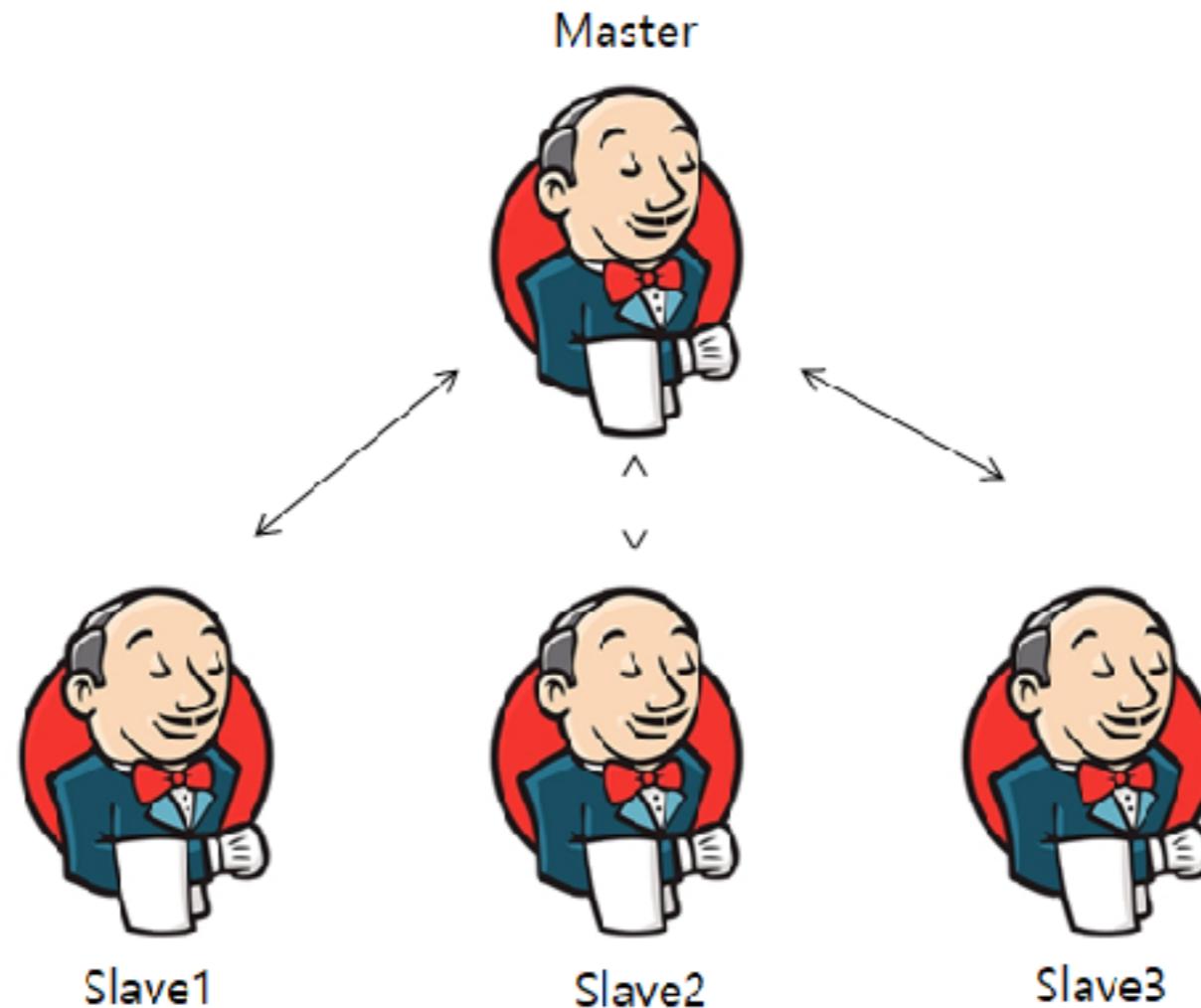


# Jenkins Architecture

## Jenkins controller node (master node)



# Scaling



<https://www.jenkins.io/doc/book/scaling/>



# Build pipeline

The screenshot shows the Jenkins Pipeline interface for a build named #1. The top navigation bar includes the Jenkins logo, the project name "demo01", the build number "#1", and a "Stages" link. To the right are search, refresh, and settings icons.

The main area displays the build status as "Success" (#1). Below it, details show the build was started by "admin" 6.4 seconds ago, queued for 23 ms, and took 1.5 sec. A "Graph" section shows a linear flow from "Start" to "Stage 1", "Stage 2", "Stage 3", and "End", with each stage marked as successful (green checkmark).

At the bottom, a "Search" field is present. Below it, a summary table lists the stages:

Stage	Duration	Started At	Run By	Actions
Stage 1	83ms	6s ago	Jenkins	<span>View</span>
Stage 2	64ms	5s ago	Jenkins	<span>View</span>
Stage 3	63ms	4s ago	Jenkins	<span>View</span>

Details for Stage 2 are expanded, showing the output "Hello Stage 2!" and a timestamp of 22ms. The Jenkins logo is located at the bottom left of the slide.



# Pipeline syntax

Jenkins search ?

Jenkins > xxxx > Pipeline Syntax

Back Snippet Generator Declarative Directive Generator Declarative Online Documentation Steps Reference Global Variables Reference Online Documentation IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step archiveArtifacts: Archive the artifacts

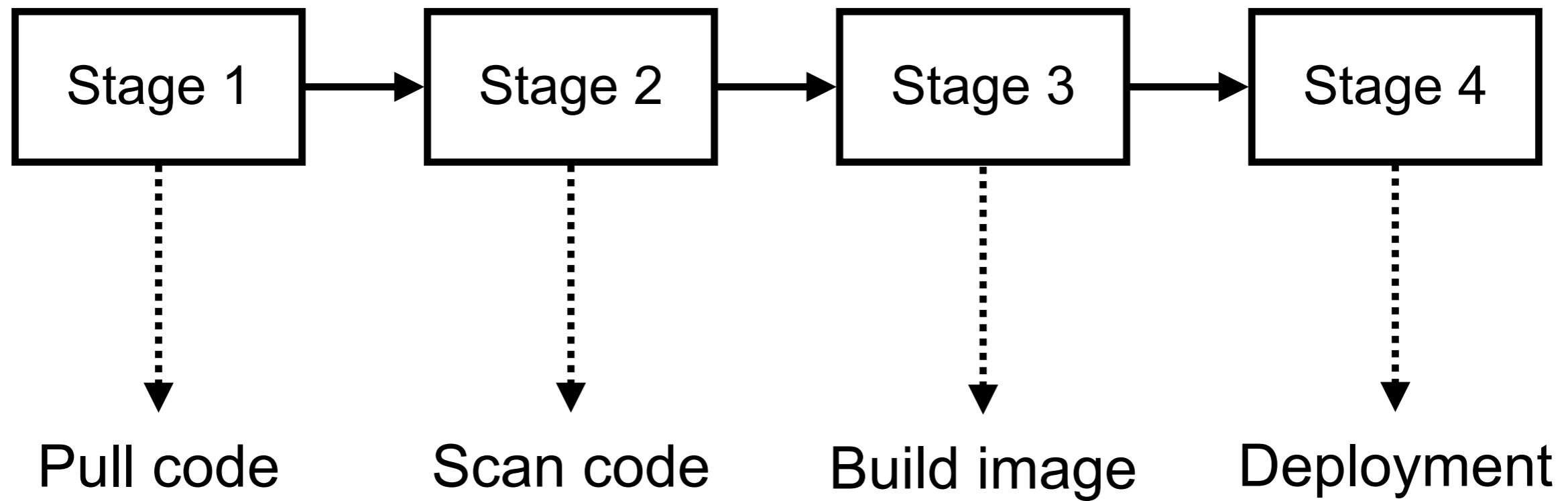
Files to archive

Advanced...

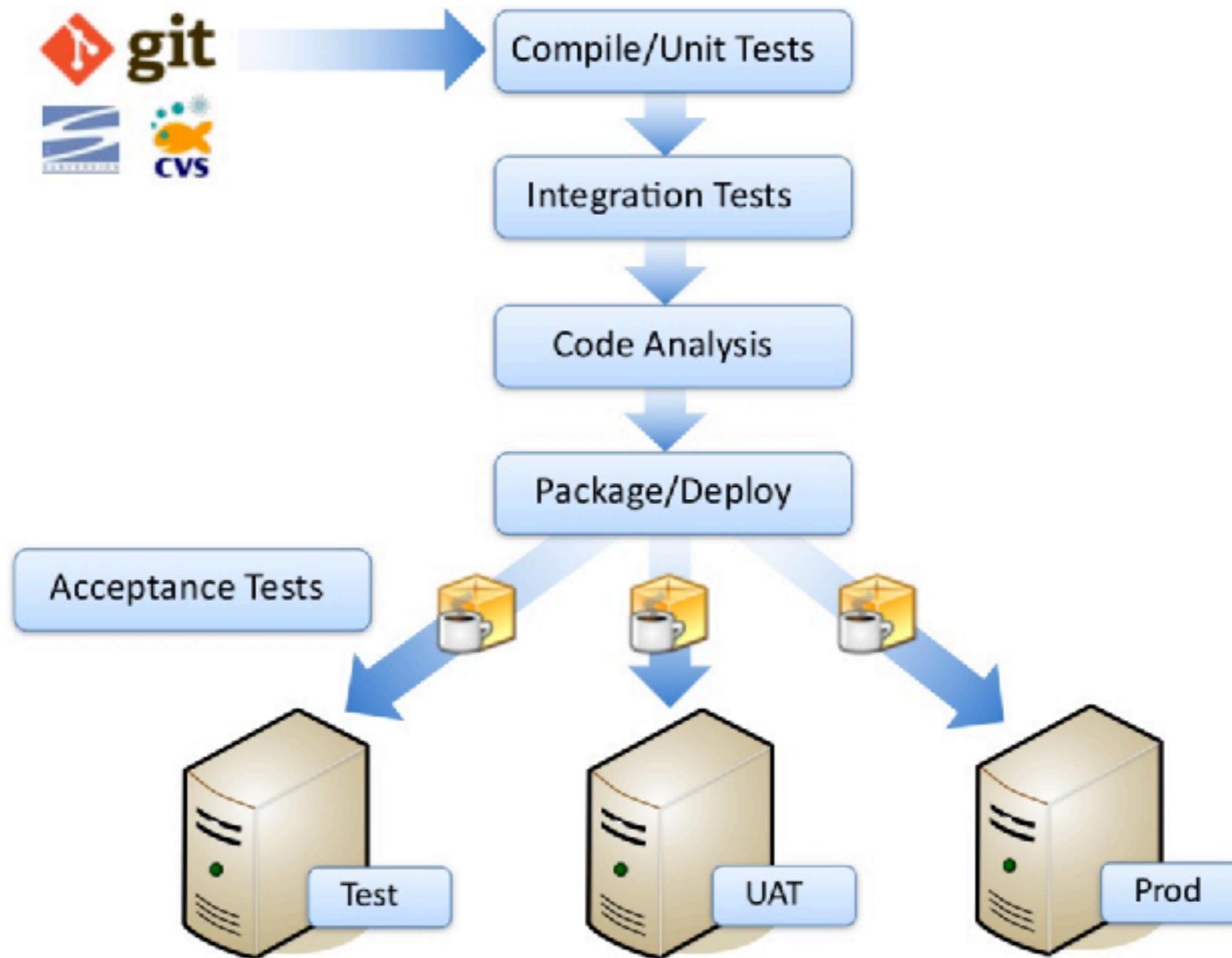
Generate Pipeline Script



# Design your pipeline



# Pipeline of this project



# List of Tools

Technology	Description
<b>Java</b>	Main programming language
<b>Apache Maven</b>	Build tool
<b>JUnit</b>	Unit and integration test tool
<b>Cobertura</b>	Code coverage tool
<b>Robotframework</b>	Acceptance test tool
<b>Git</b>	Version Control System
<b>SonarQube</b>	Static code analysis tool
<b>JFrog Artifactory</b>	Keep artifact files
<b>Jenkins</b>	Continuous Integration Server



# **Introduction to Shared Libraries**



# Shared Libraries

Collection of Groovy scripts

Reusable functions (steps) in pipeline

Common tasks, concise and readable

Reduce duplication, inconsistency



# Example pipeline

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build') {  
            steps {  
                sh 'npm install'  
                sh 'npm run build'  
            }  
        }  
        stage('Unit Test') {  
            steps {  
                sh 'npm test'  
            }  
        }  
        stage('Deploy') {  
            steps {  
                script {  
                    if (env.BRANCH_NAME == 'main') {  
                        sh 'echo Deploying to production...'  
                    } else {  
                        sh 'echo Deploying to staging...'  
                    }  
                }  
            }  
        }  
    }  
}
```

Build nodes app



# Example pipeline

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build') {  
            steps {  
                sh 'npm install'  
                sh 'npm run build'  
            }  
        }  
        stage('Unit Test') {  
            steps {  
                sh 'npm test'  
            }  
        }  
        stage('Deploy') {  
            steps {  
                script {  
                    if (env.BRANCH_NAME == 'main') {  
                        sh 'echo Deploying to production...'  
                    } else {  
                        sh 'echo Deploying to staging...'  
                    }  
                }  
            }  
        }  
    }  
}
```

Run test



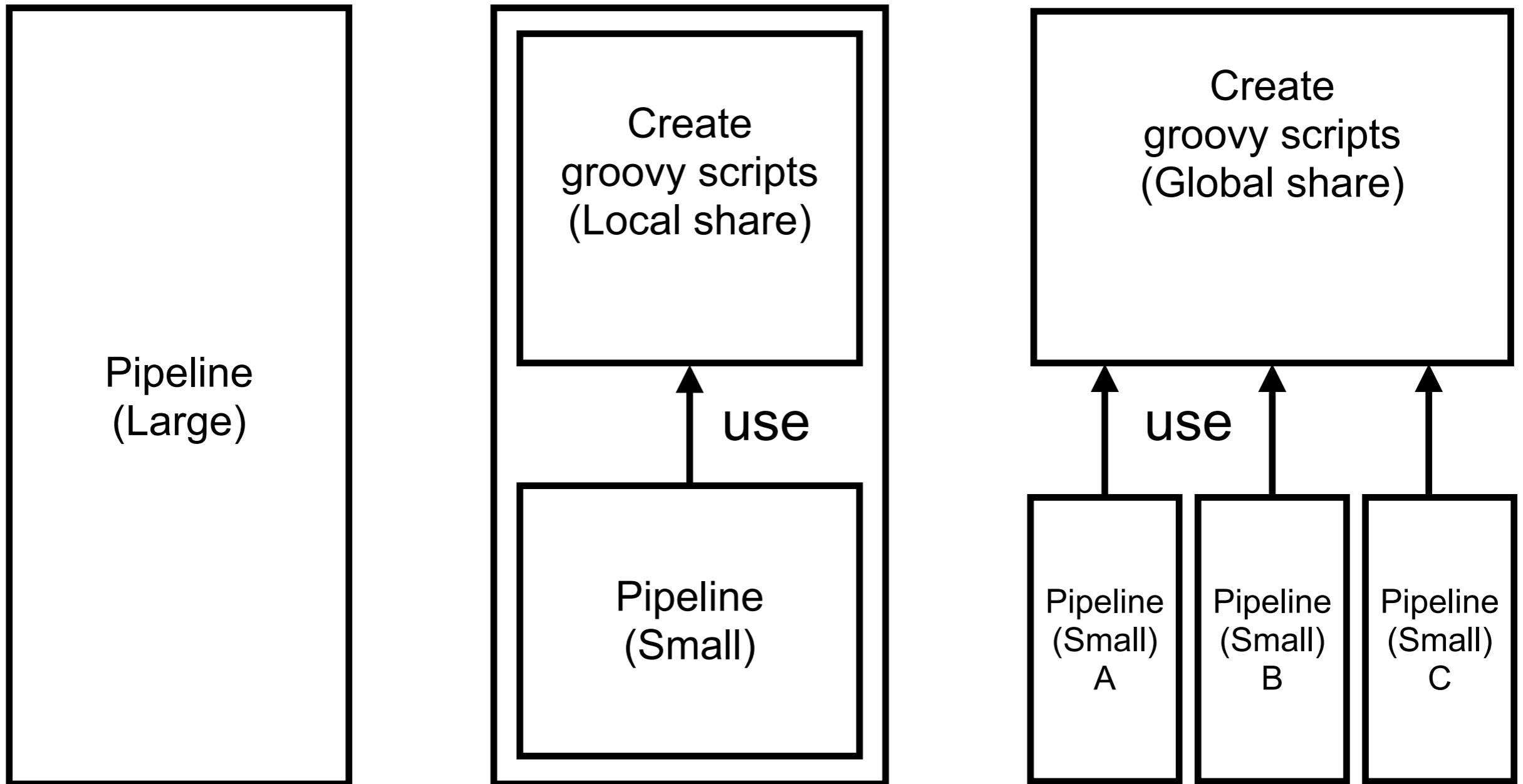
# Example pipeline

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build') {  
            steps {  
                sh 'npm install'  
                sh 'npm run build'  
            }  
        }  
        stage('Unit Test') {  
            steps {  
                sh 'npm test'  
            }  
        }  
        stage('Deploy') {  
            steps {  
                script {  
                    if (env.BRANCH_NAME == 'main') {  
                        sh 'echo Deploying to production...'  
                    } else {  
                        sh 'echo Deploying to staging...'  
                    }  
                }  
            }  
        }  
    }  
}
```

Deploy application



# How to create reuse scripts ?



<https://github.com/up1/demo-pipeline-nodejs/wiki/Workshop-with-pipeline>



# Steps to shared library

Repository setup (project structure)

Jenkins configuration with global pipeline library

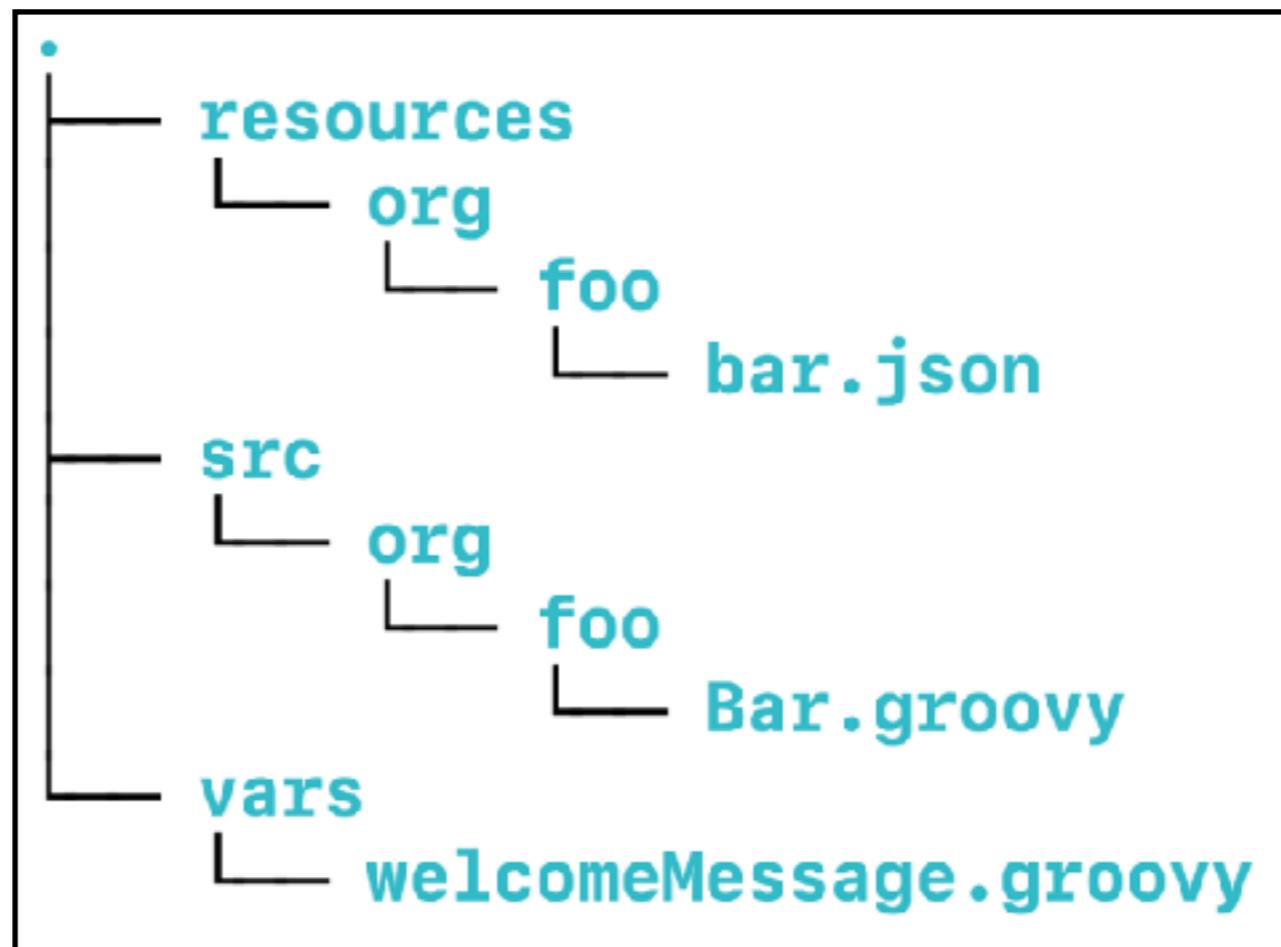
**Write custom script with groovy**

Integrate in pipeline !!



# 1. Create repository

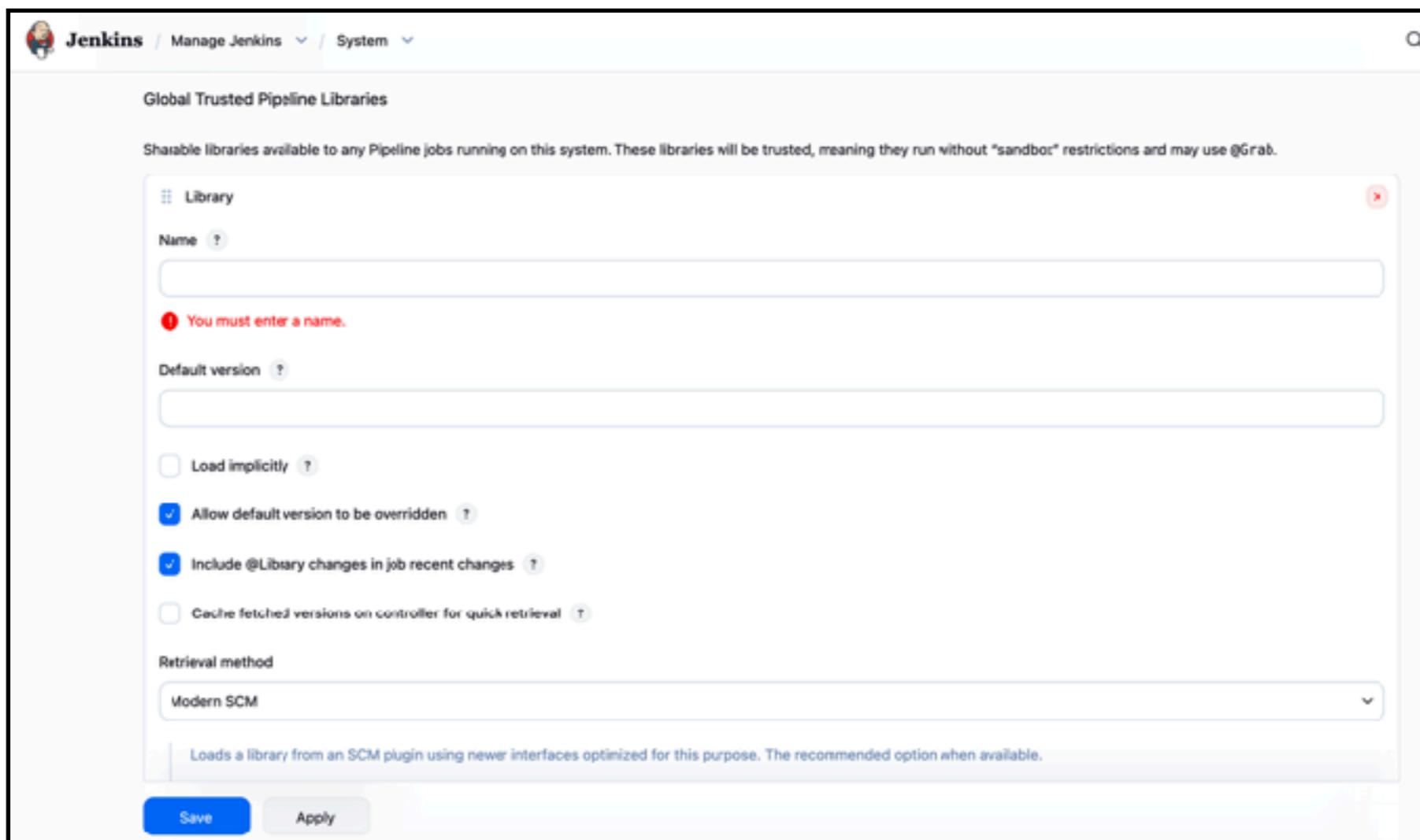
Dedicate repository to keep your shared library  
Easy to organize, maintain and tracking



# 2. Global pipeline library

Setting -> System -> Global pipeline library

Config Jenkins, how to access the share library code



# 3. Write groovy script !!



# **Workshop**

# **Pipeline as a Code**



# Workshops

Learn pipeline syntax

Stages and steps

Sequential and parallel process

Post processing

Conditional

Shared library with groovy script

Best practices



# Q/A

