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)

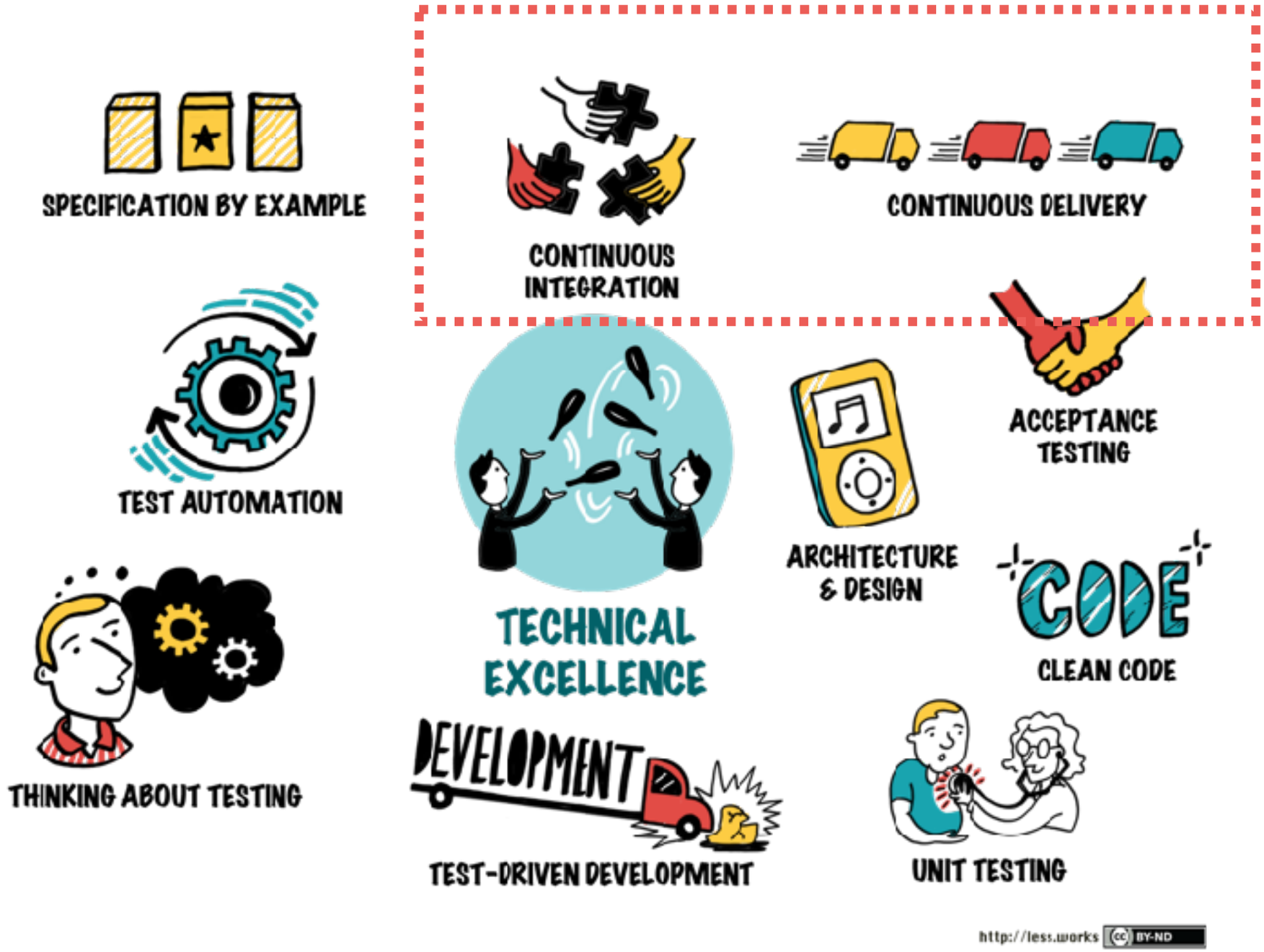
1. Pipeline as a Code with Jenkins
2. Best practice with declarative pipeline
3. Refactoring to shared libraries
4. Versioning and life cycle
5. Workshop

https://github.com/up1/course-ci-cd-with-jenkins

Software development

Continuous Integration ?

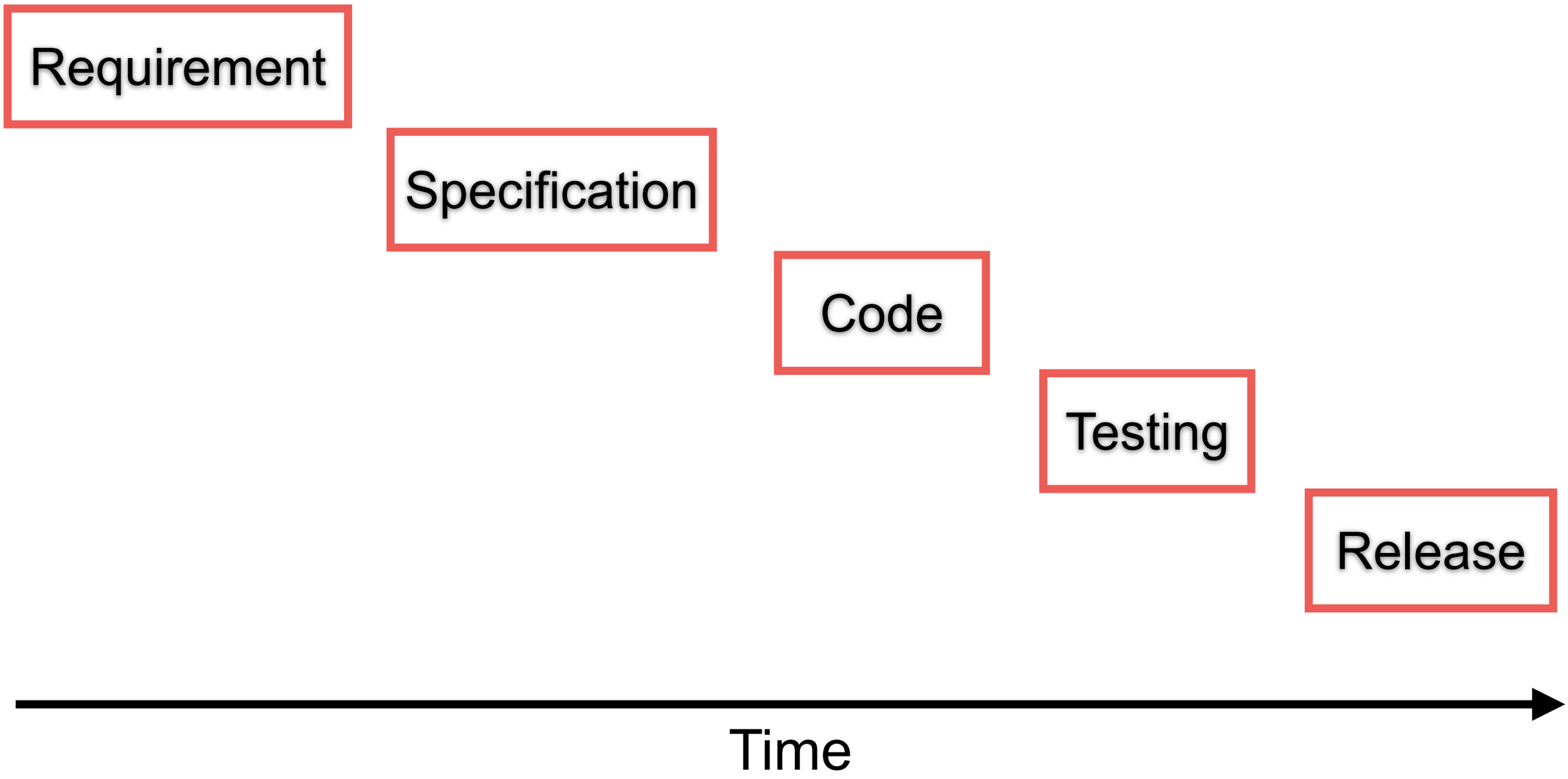
Technical Excellence



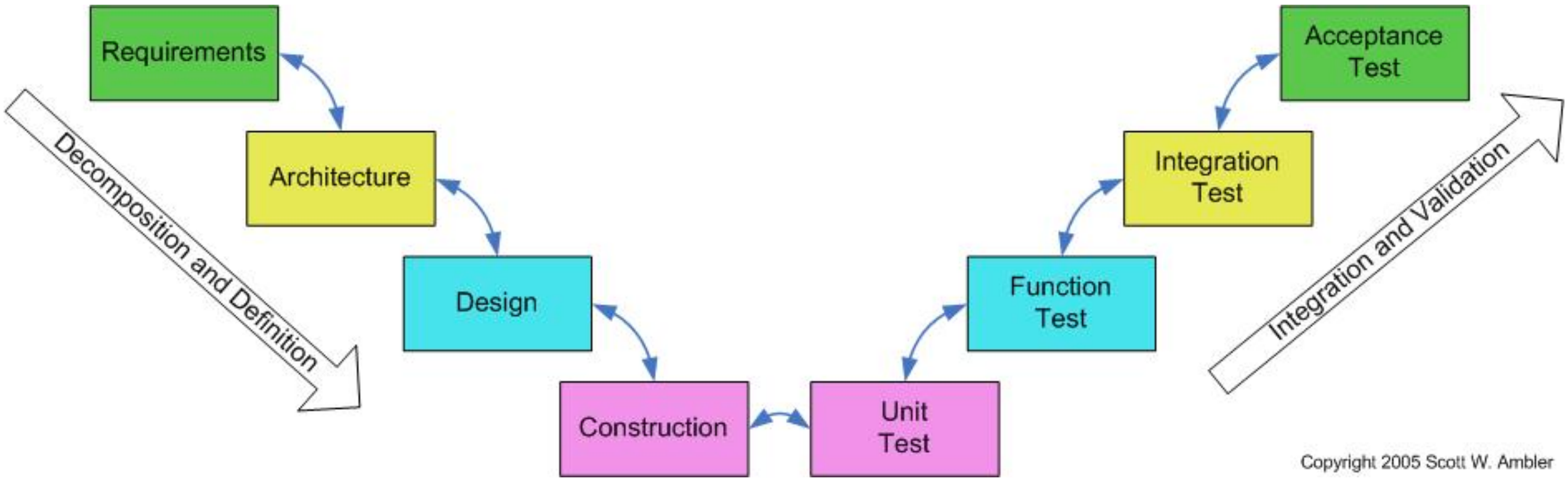
https://less.works/less/technical-excellence

Why CI/CD ?

Software Delivery Process



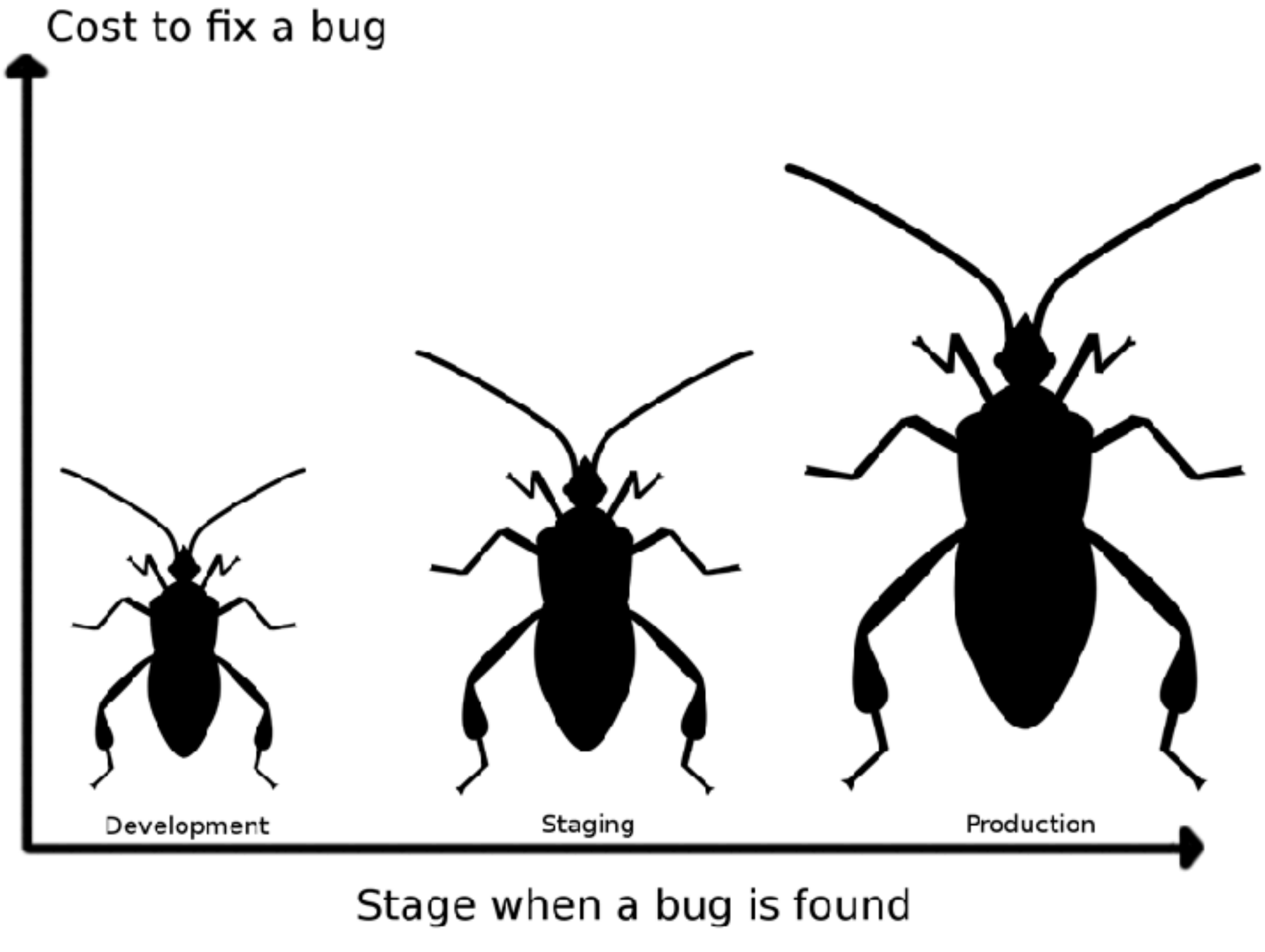
V Model

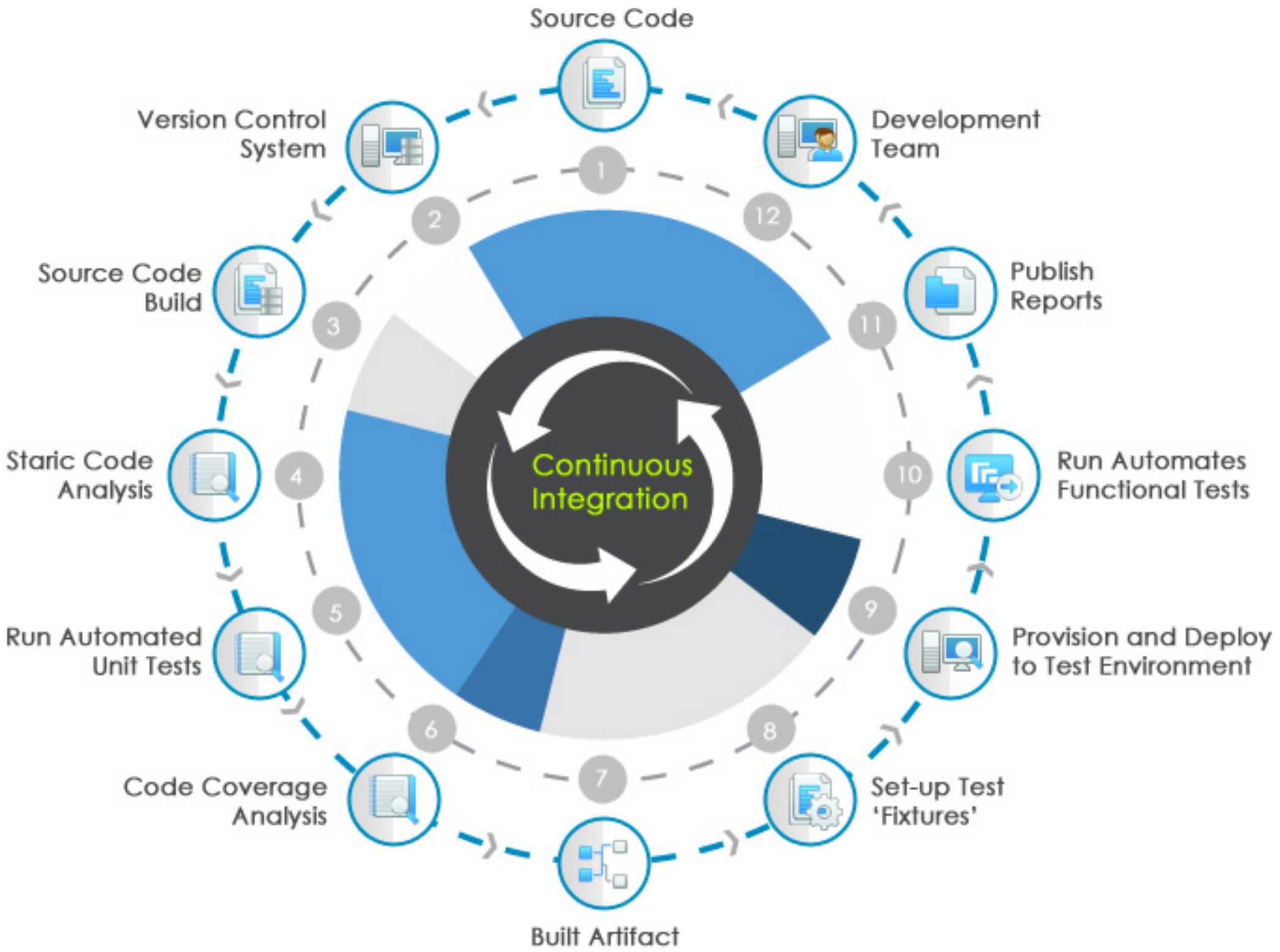


The cost of integration

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™

Visual Studio Team Foundation Server

Hudson

Travis

wercker

circleci

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

Continuous Integration

Discipline to **integrate frequently**

Continuous Integration

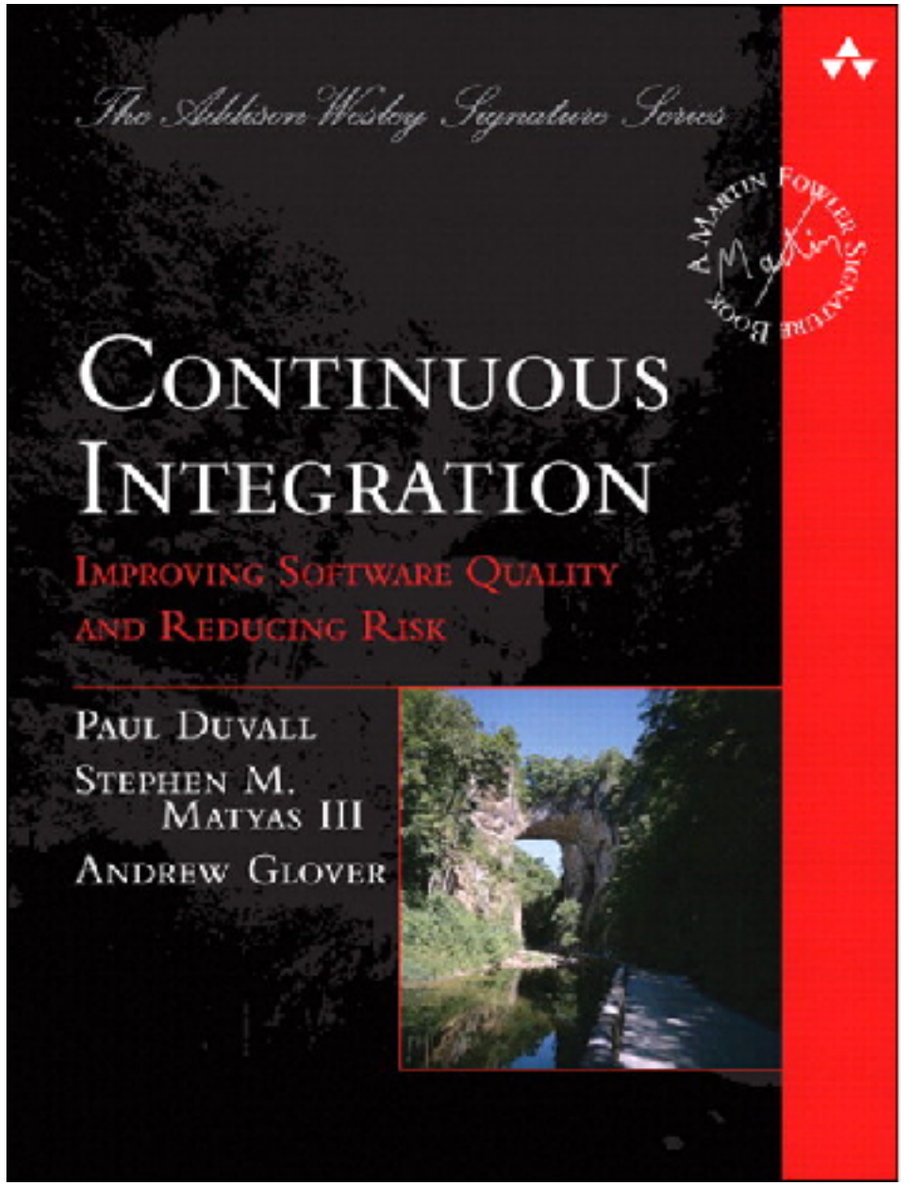
Strive to make **small change**

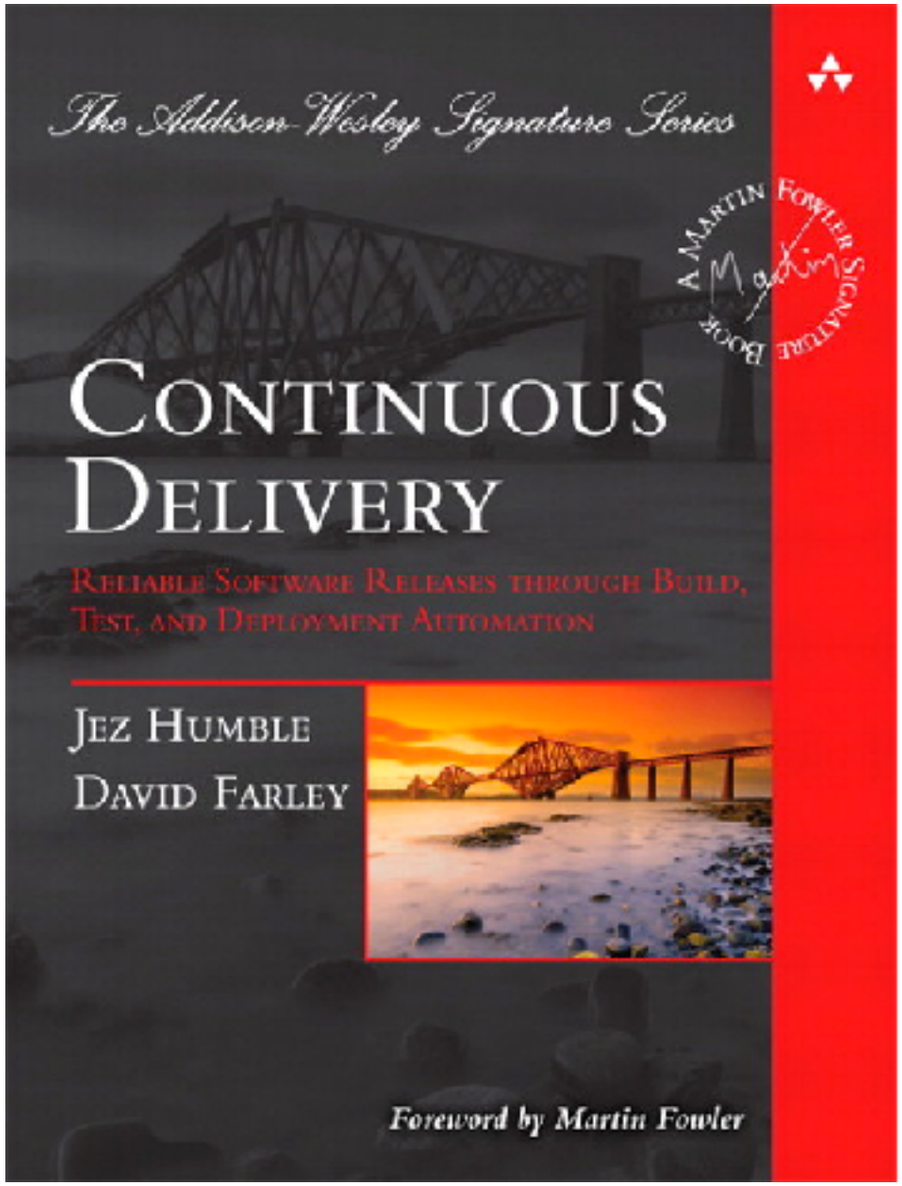
Continuous Integration

Strive for **fast feedback**

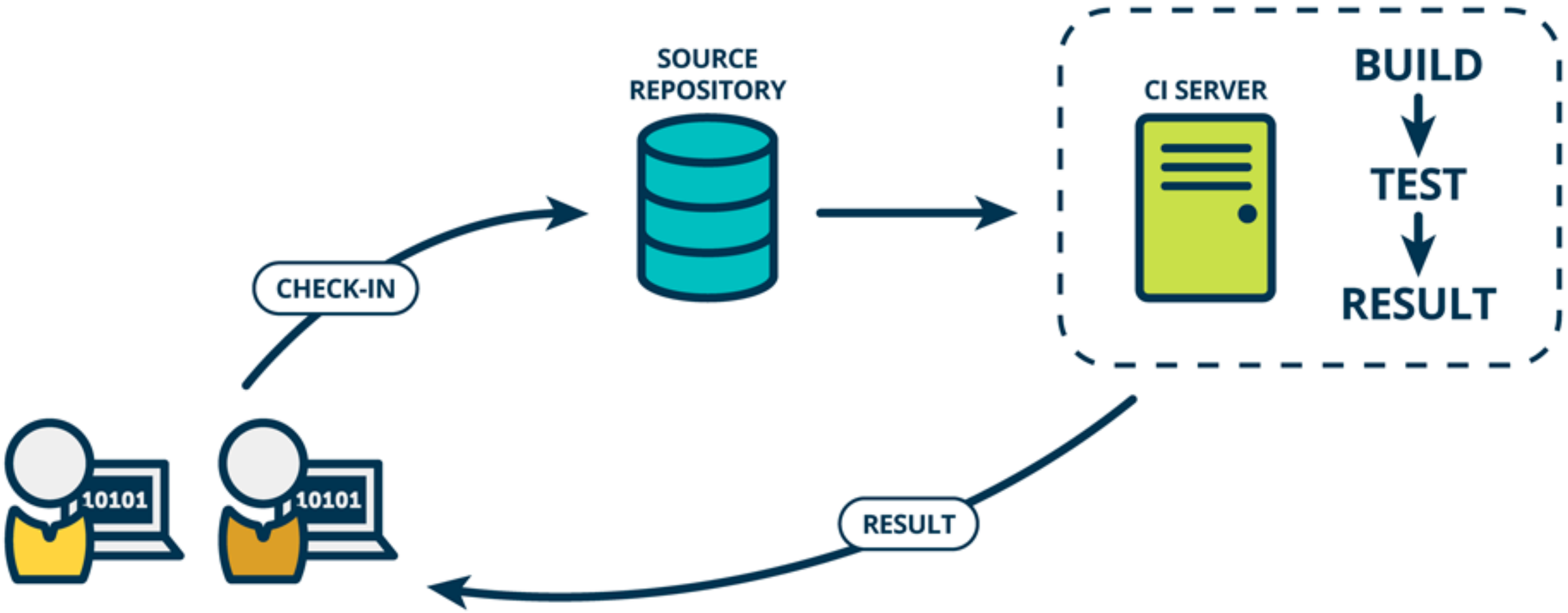
Practices of Continuous Integration

Improve quality and reduce risk

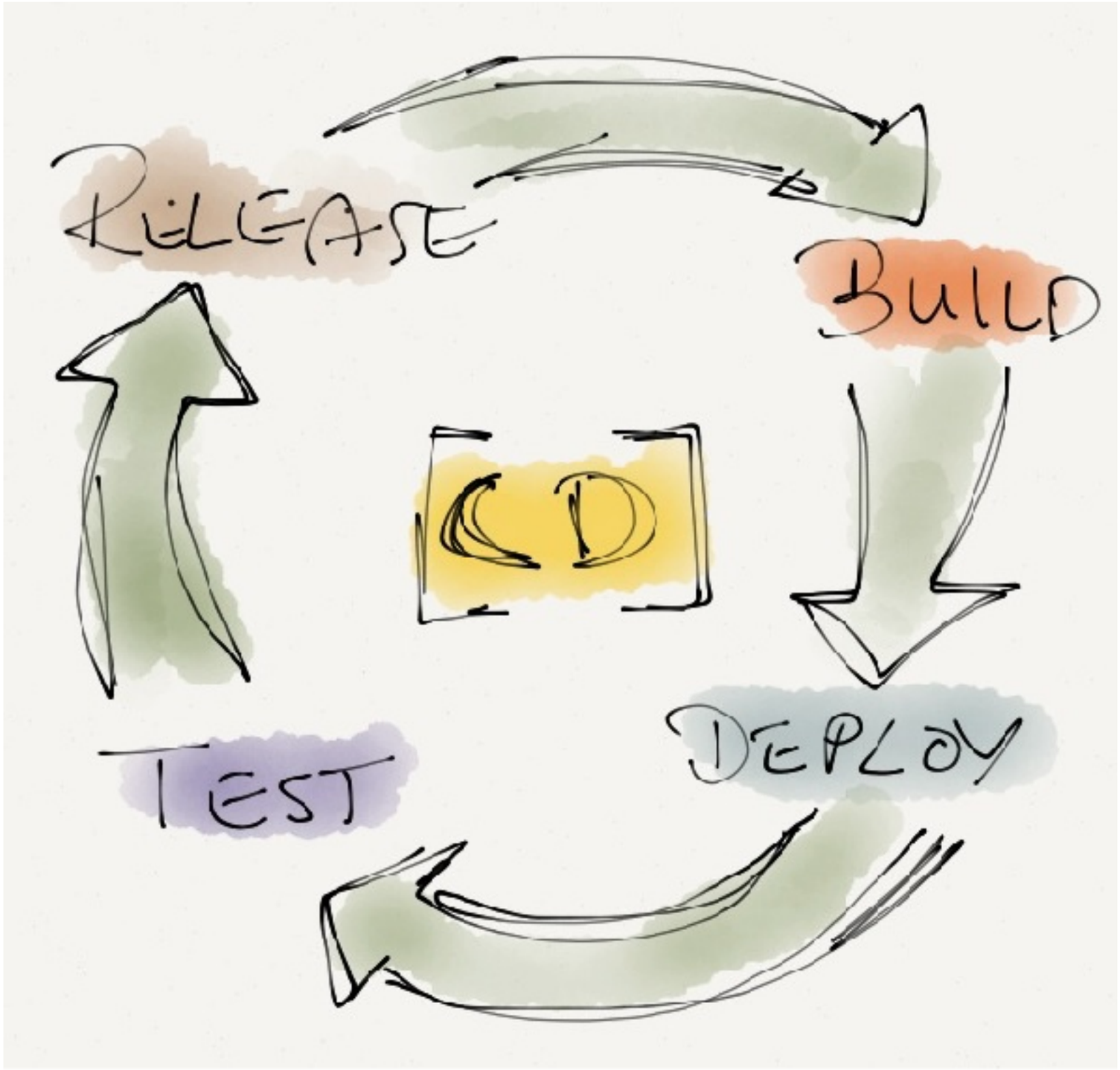




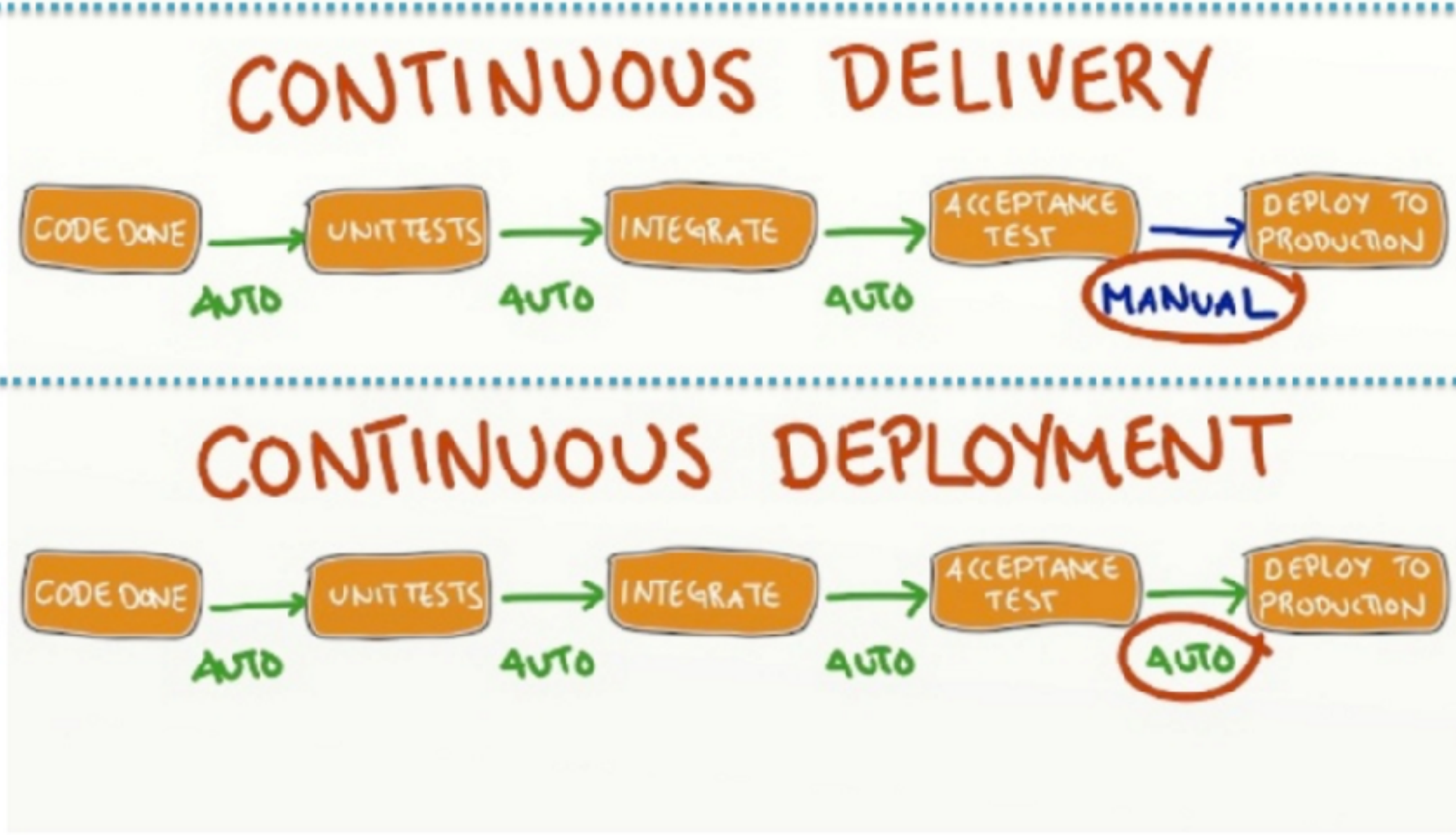
Continuous Integration



CD ?

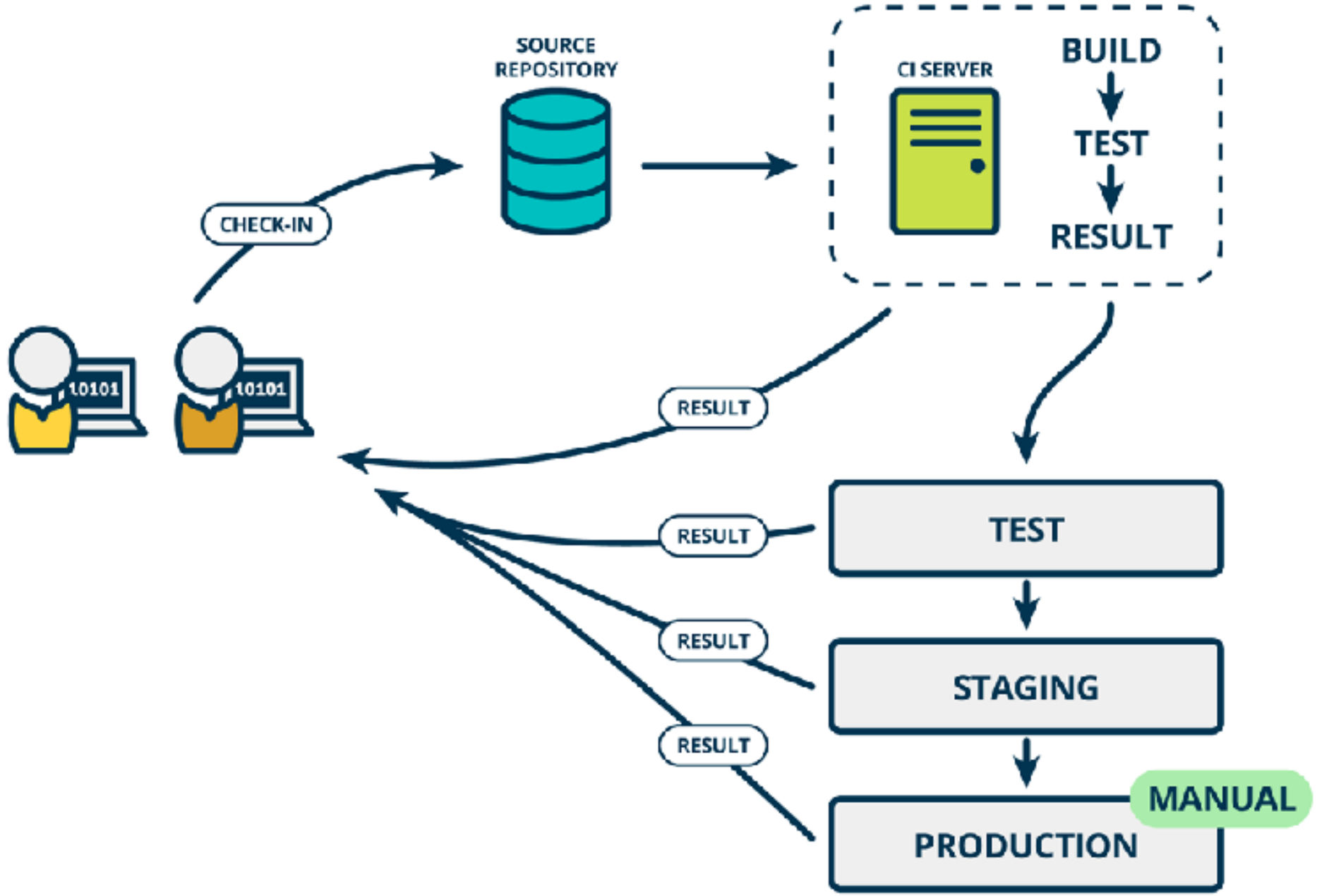


CD ?

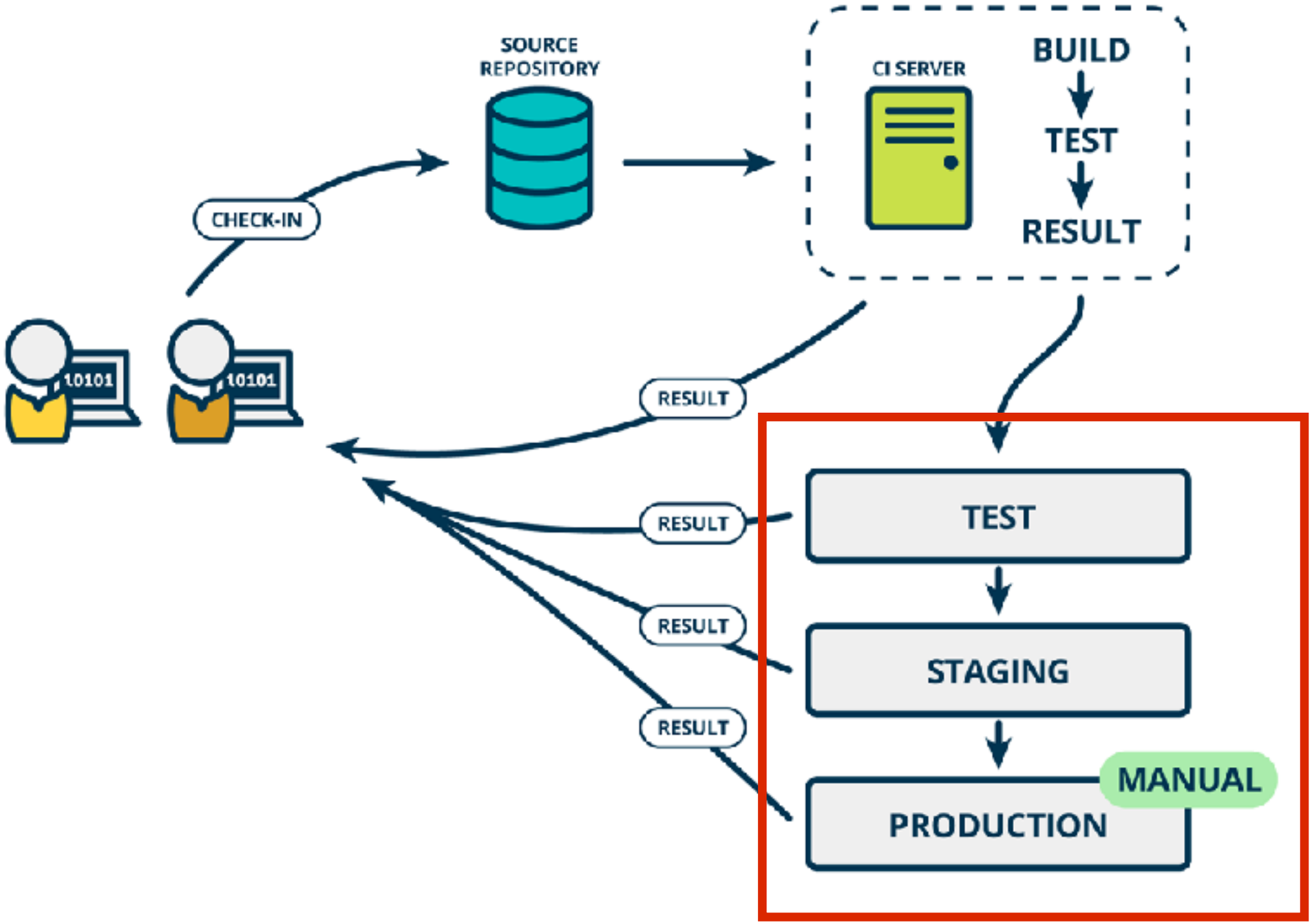


http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Continuous Delivery



Rise of DevOps



Continuous Integration

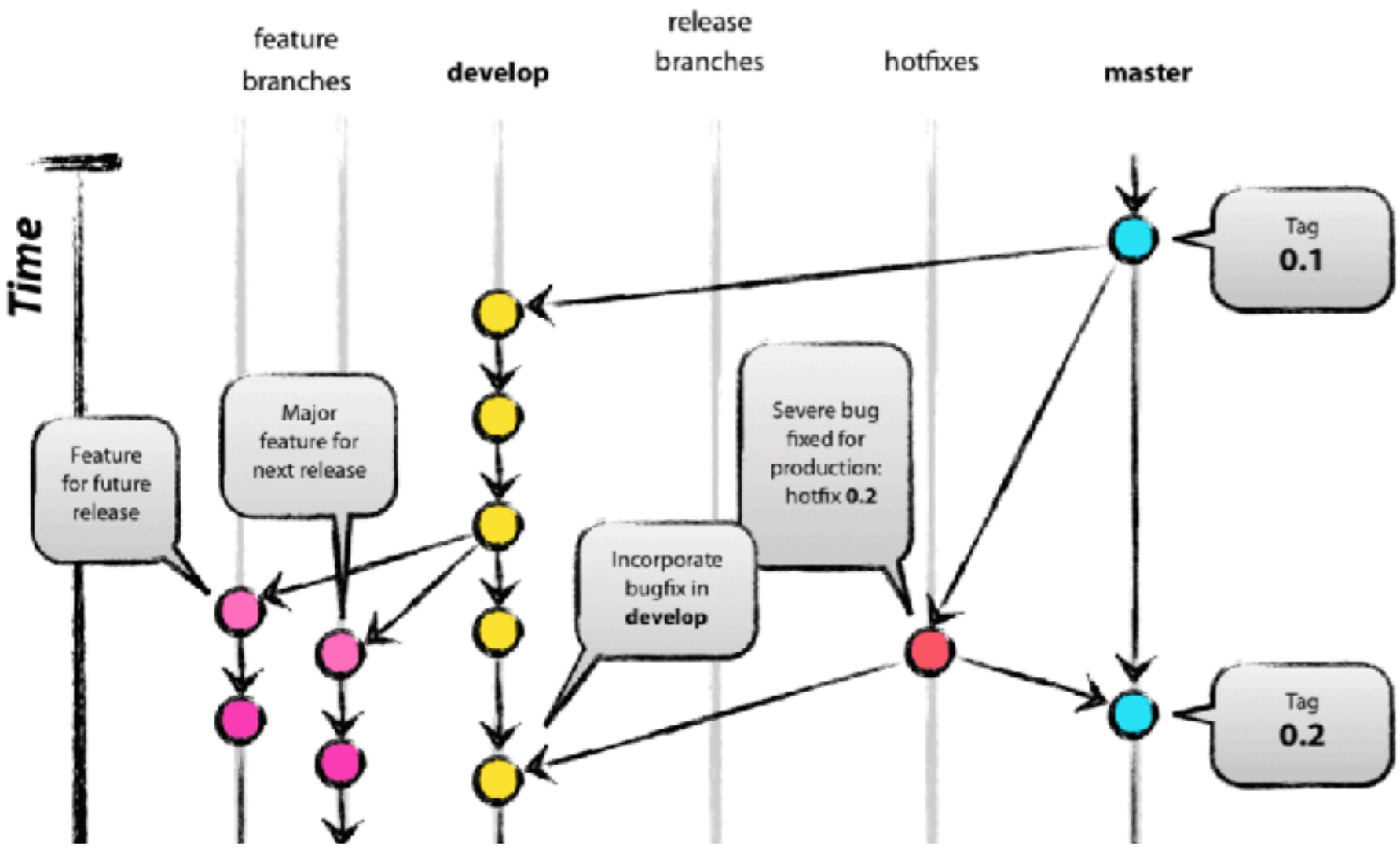
is a Software development **practices**

Practice 1

Maintain a single source repository

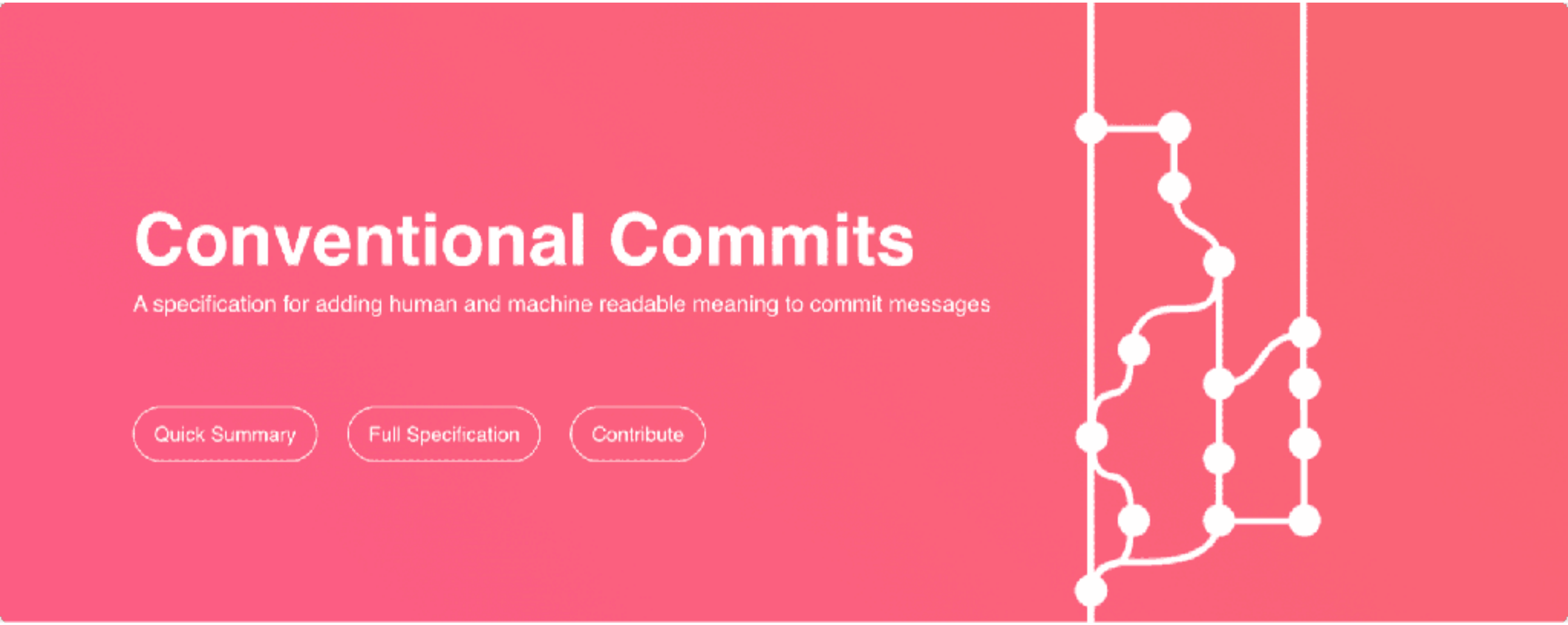
In general, you should store in source control everything you need to build anything

Git Branching Model



https://nvie.com/posts/a-successful-git-branching-model/

Conventional Commits

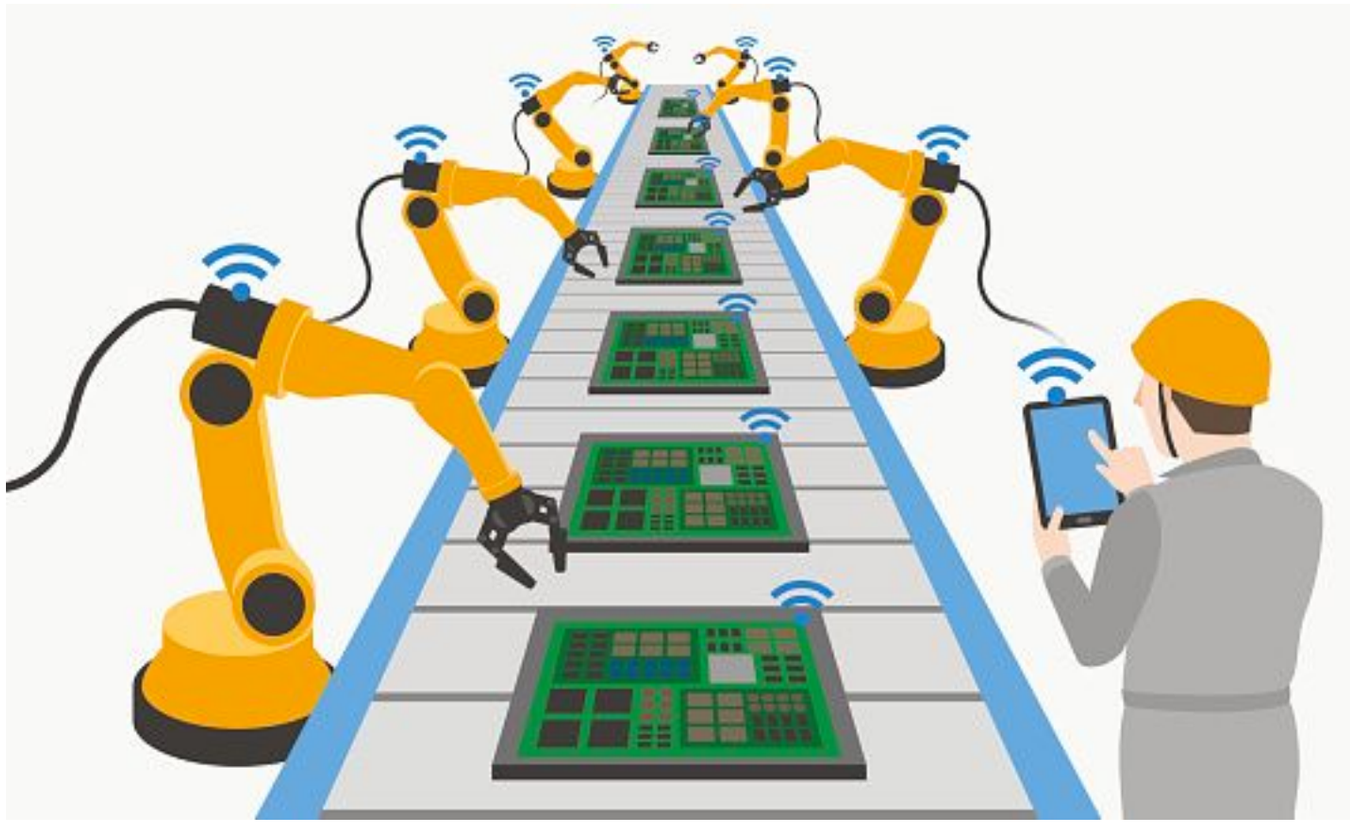


https://www.conventionalcommits.org/en/v1.0.0/

Practice 2

Automated the build

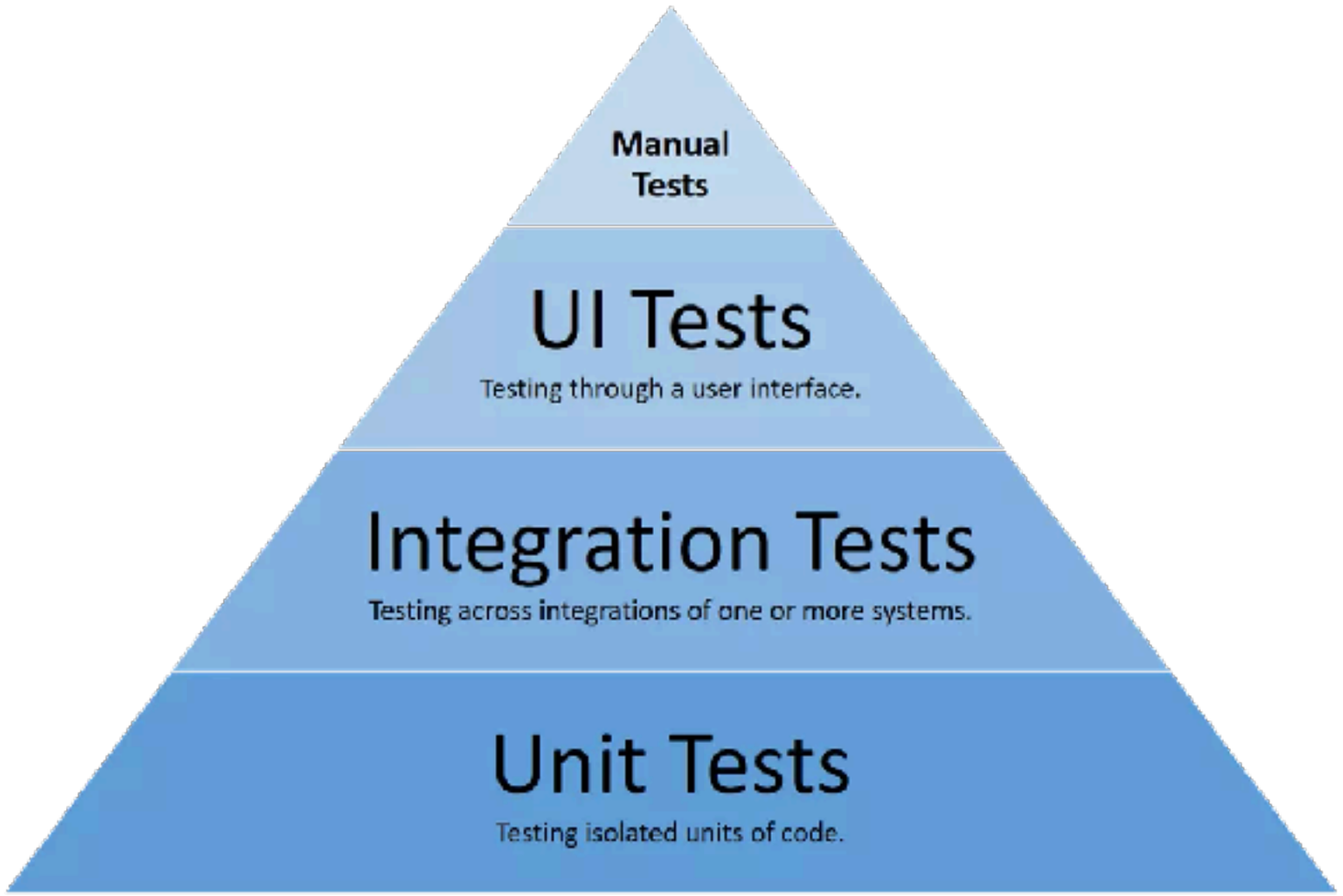
Automated environment for builds



Practice 3

Make your build **self-testing**

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

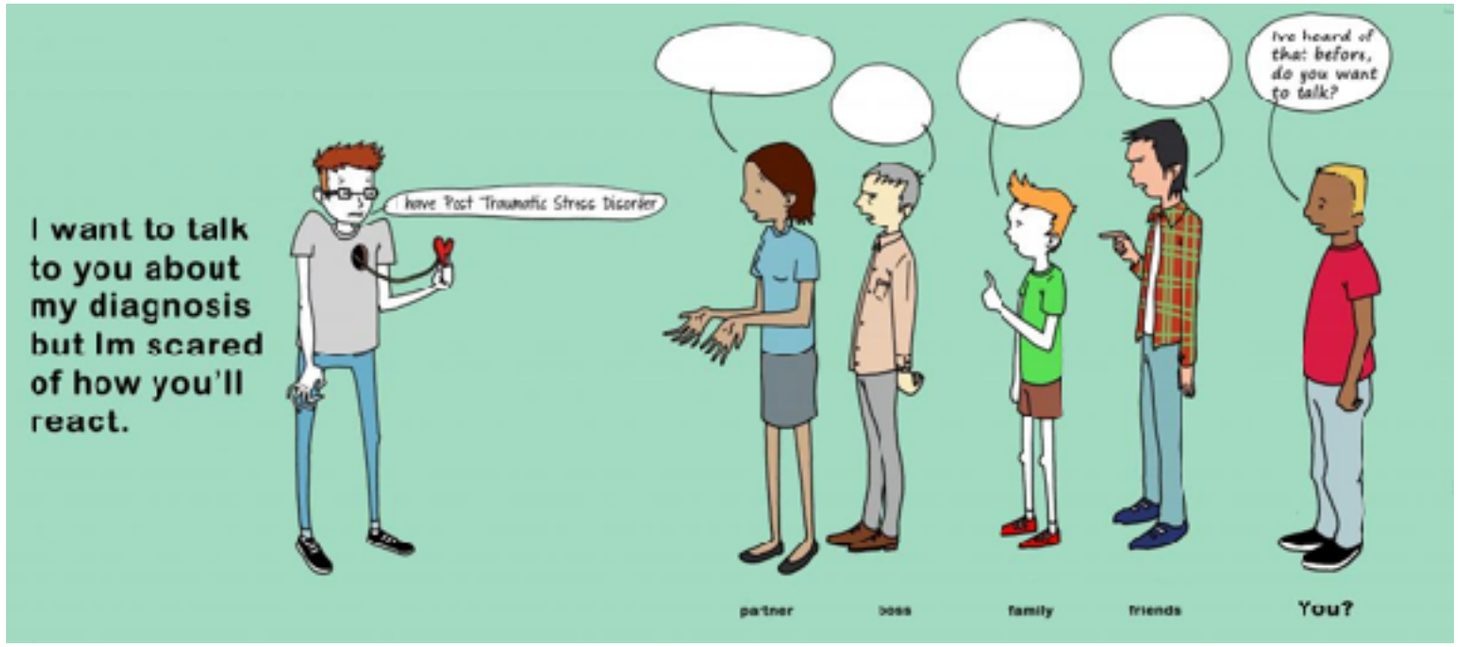


Practice 4

Everyone **commits** to the mainline everyday

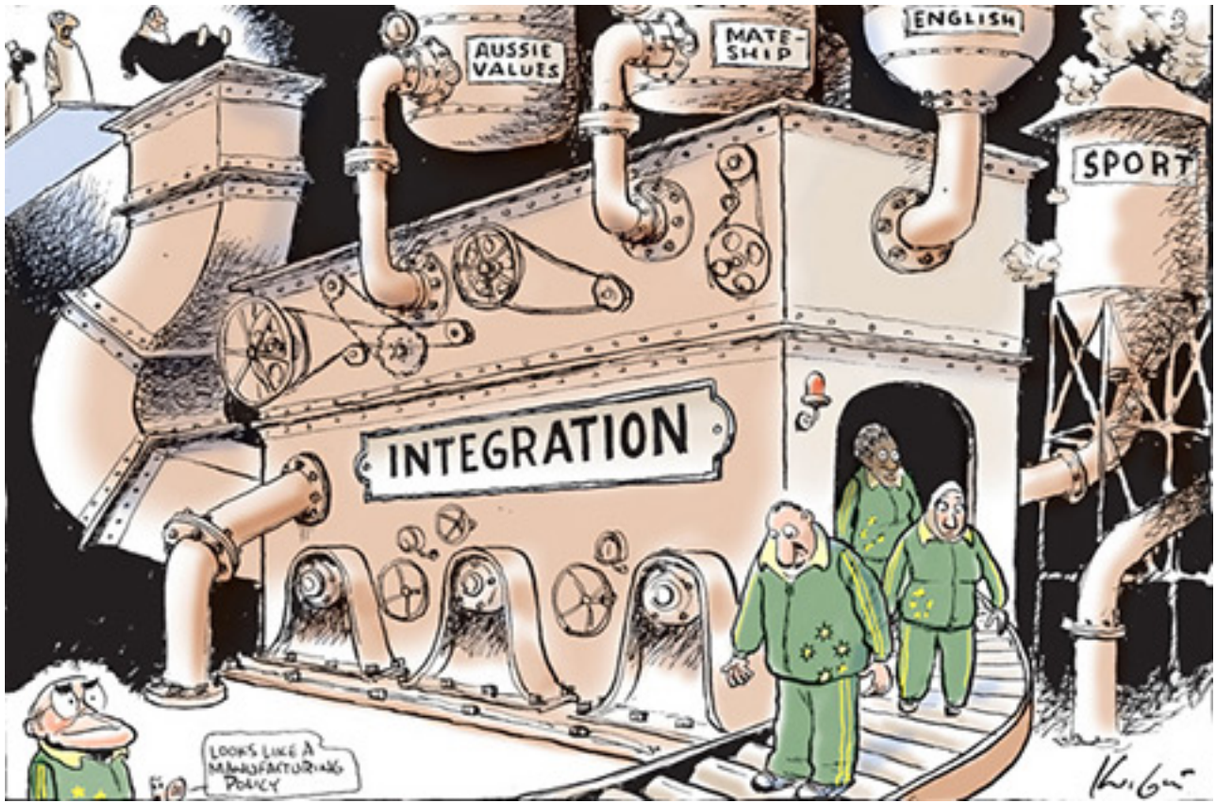
Integration is about **communication**

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

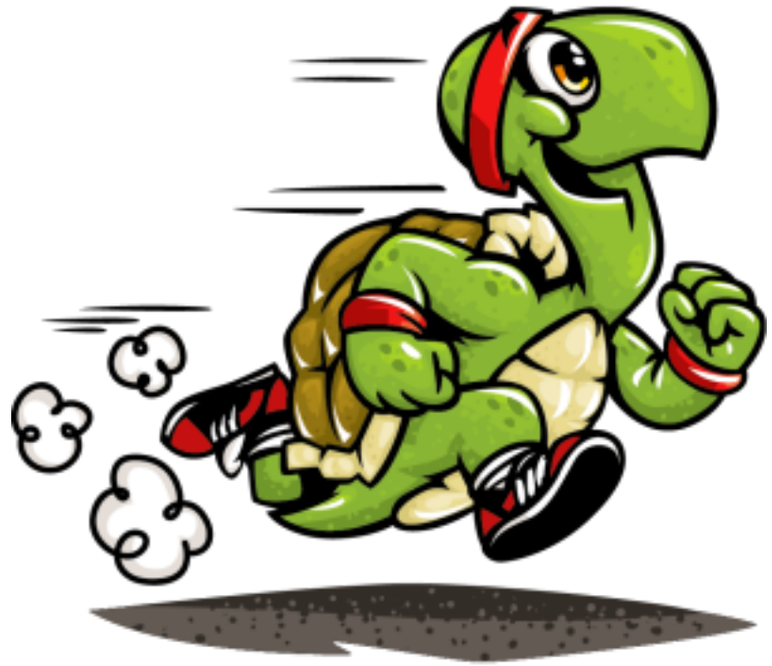
Fix broken builds **immediately**

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

Practice 7

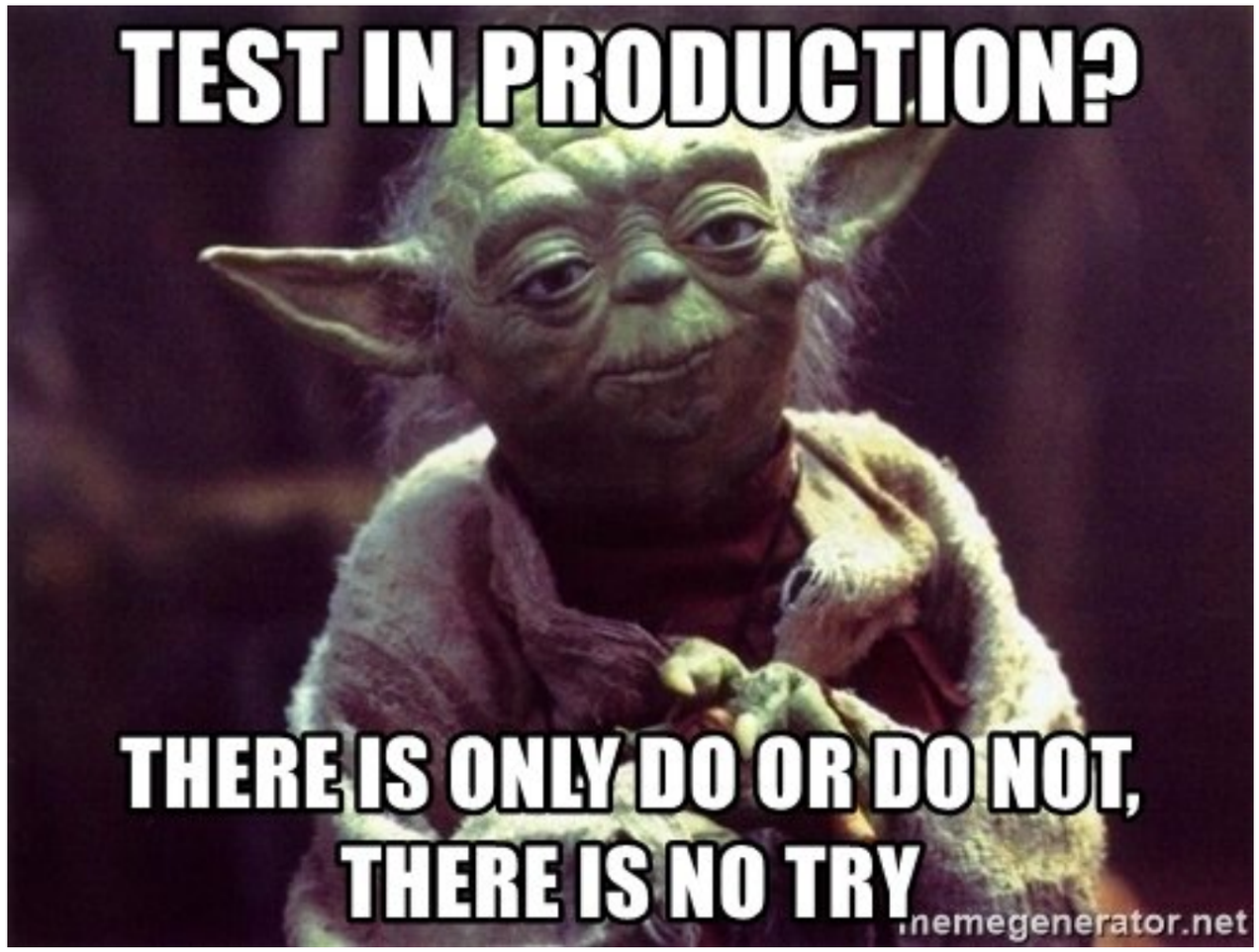
Keep the build **fast**

Continuous Integration is to provide rapid feedback



Practice 8

Test in clone of the **Production** environment



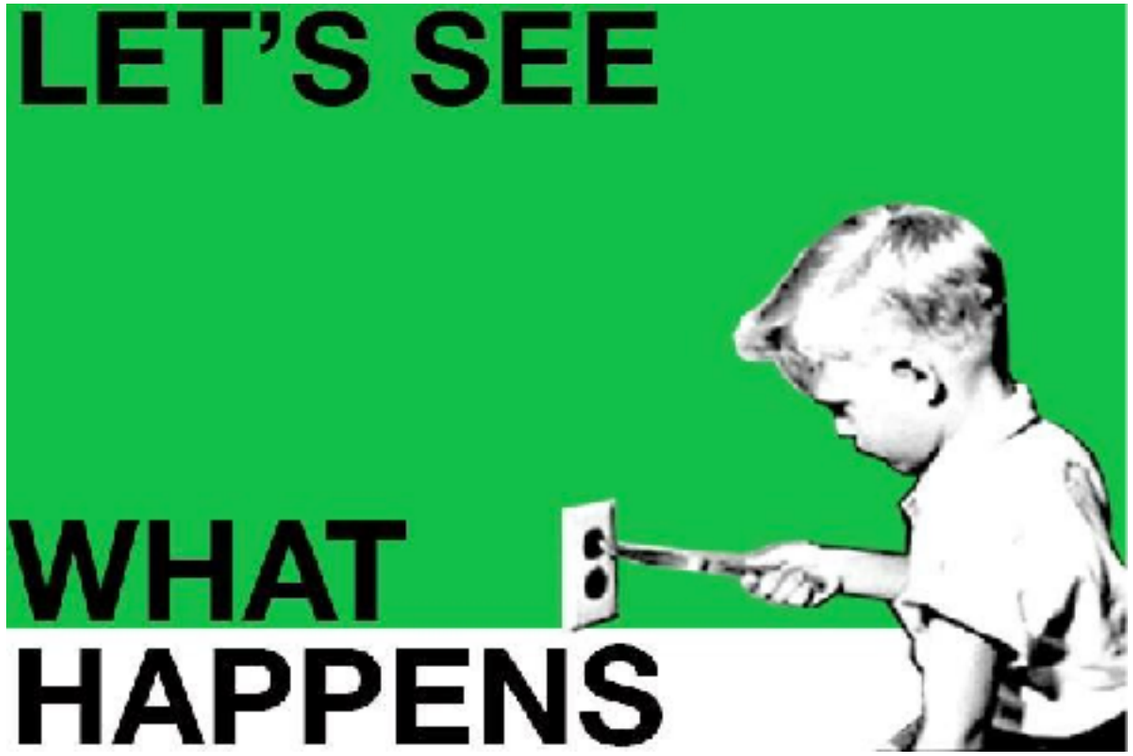
Practice 9

Make it easy for anyone to get
the latest executable

Make sure well known place where people can find

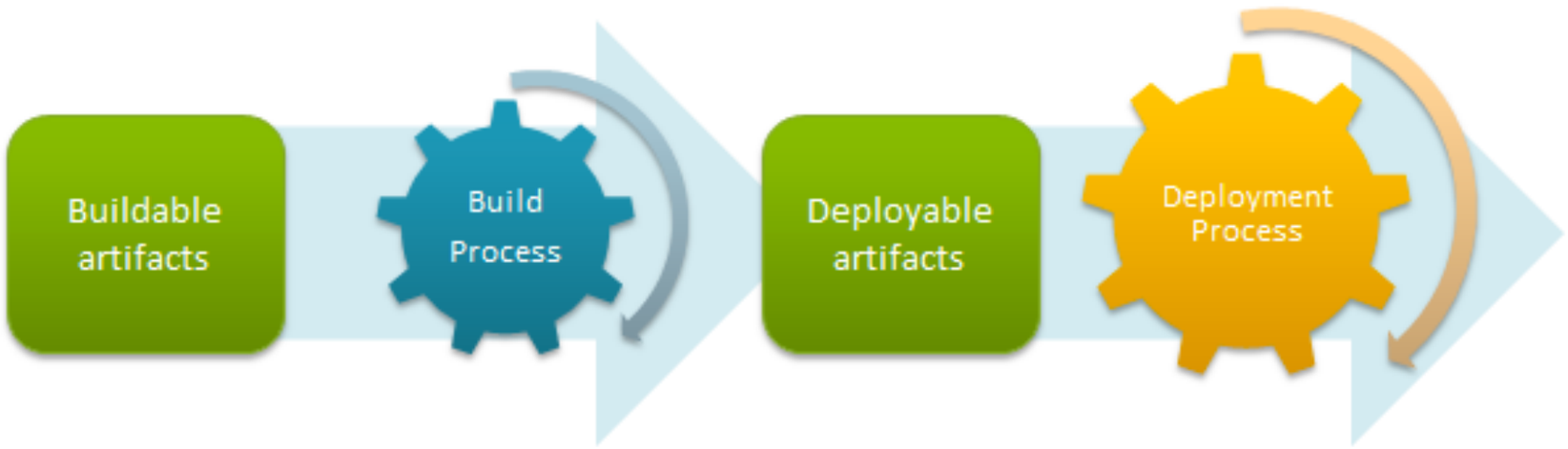
Practice 10

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

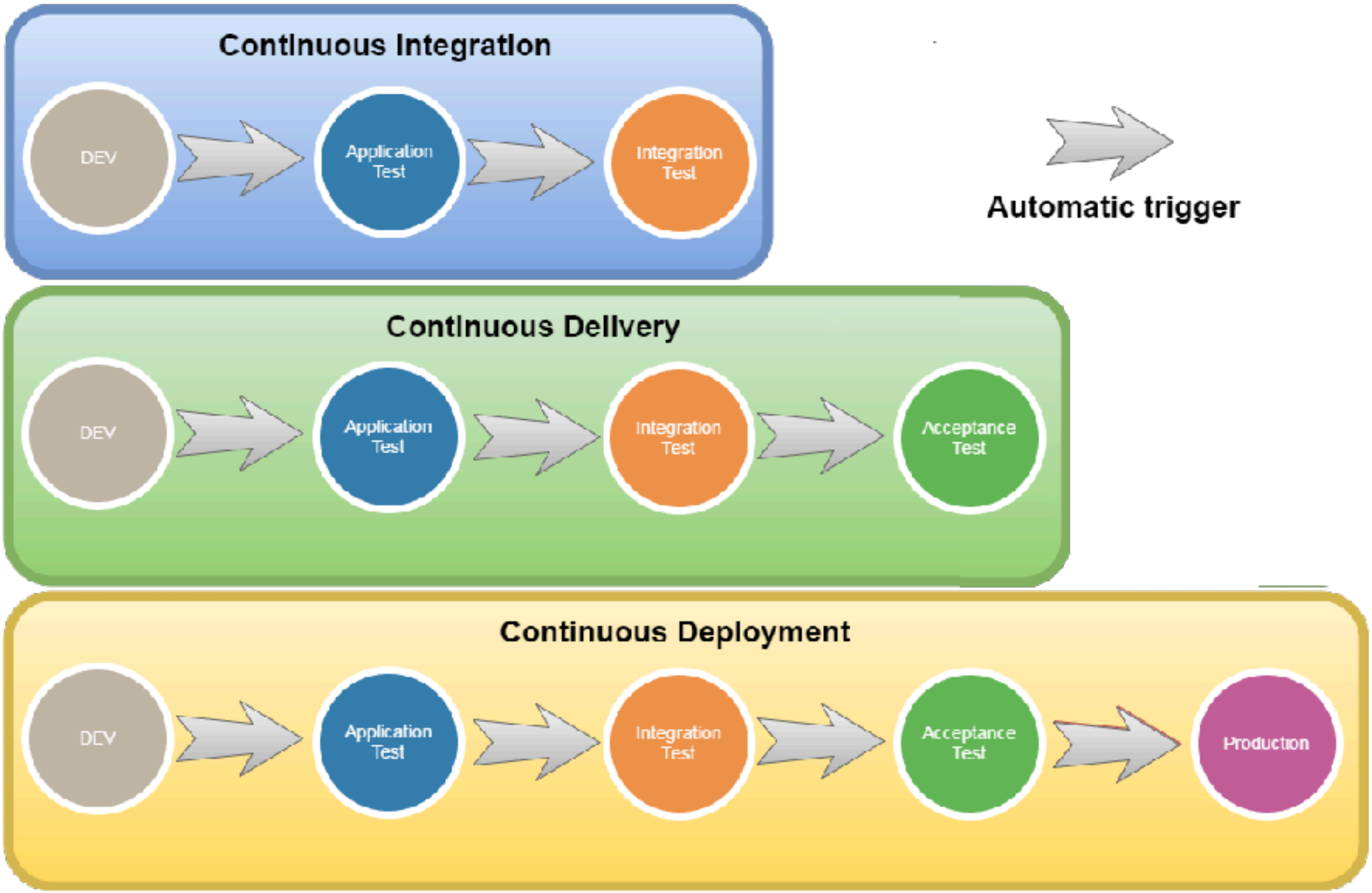


Practice 11

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 · cloudbees · OSL OPEN SOURCE LAB

rackspace 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

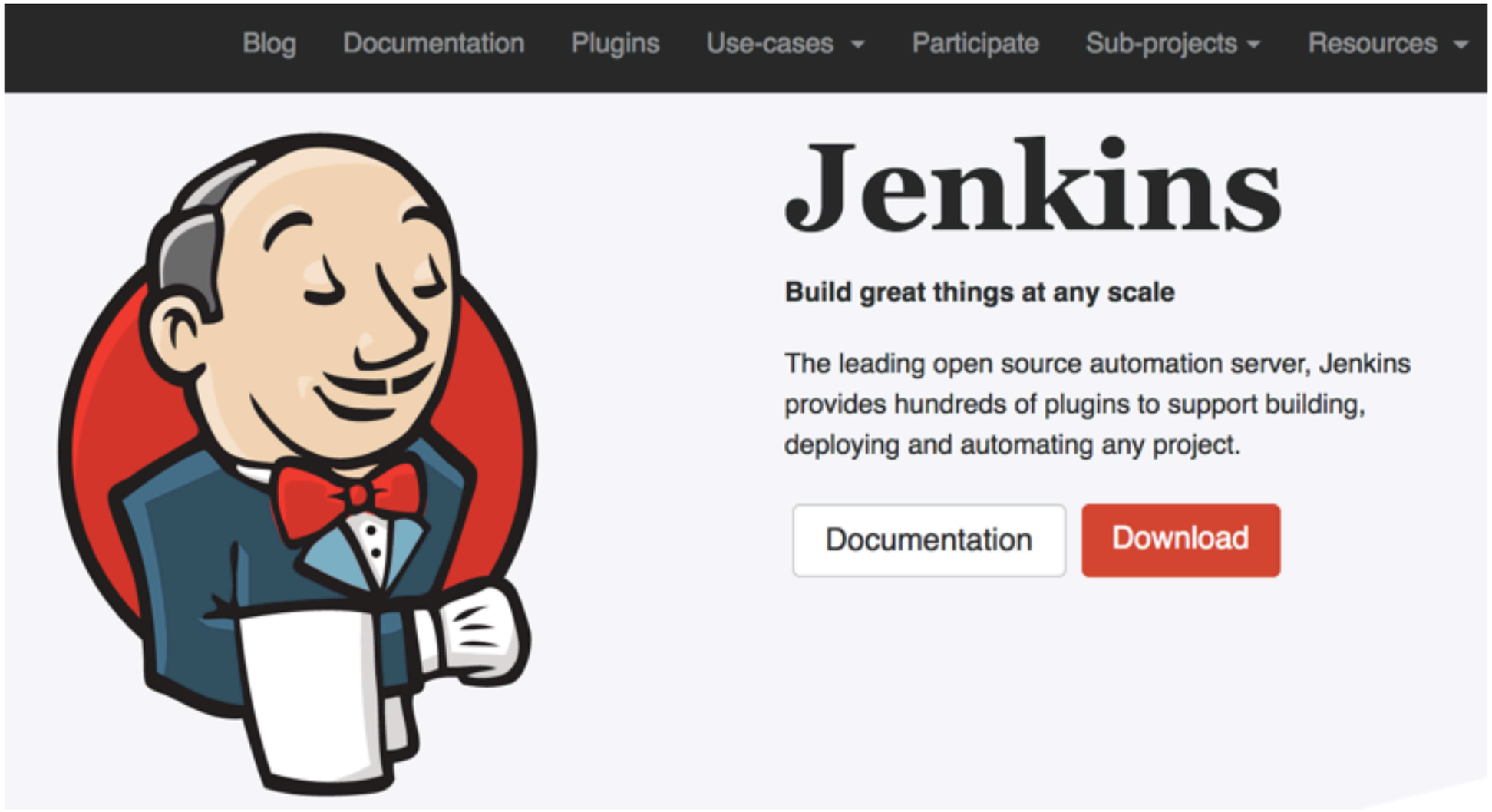
JBoss

Websphere

WebLogic

Glassfish

Download Jenkins



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

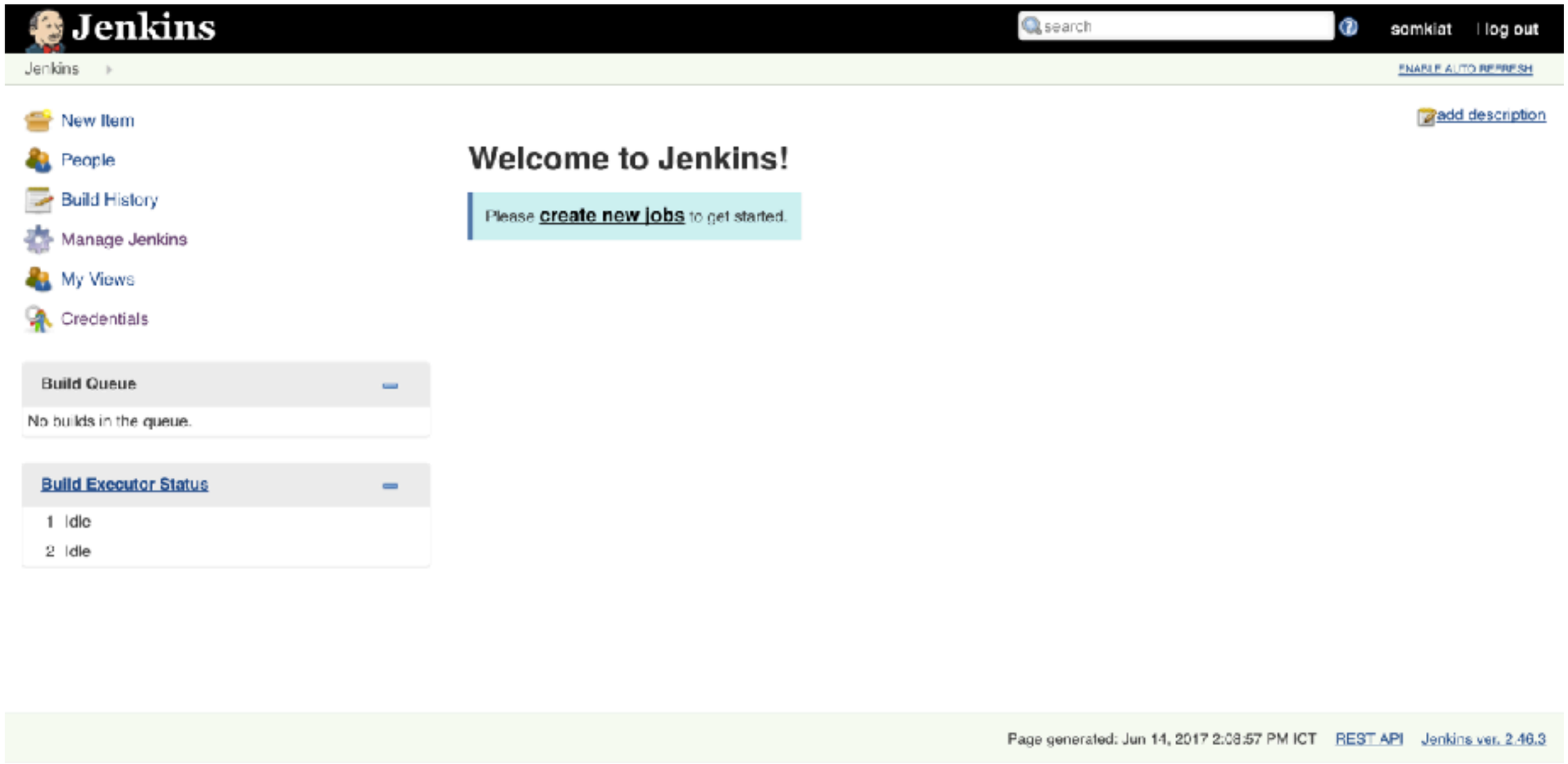
Default port of server is **8080**

```
 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
gine v4.0 running: controlPort=disabled
 jenkins.InitReactorRunner$1 onAttained
ion
 jenkins.InitReactorRunner$1 onAttained

 jenkins.InitReactorRunner$1 onAttained
```

Welcome to Jenkins



Finds Jenkins's plugin

https://plugins.jenkins.io/

Let’s create pipeline with Jenkins

Types of Jenkins Projects

Freestyle project

Pipeline project

Multi-configuration

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

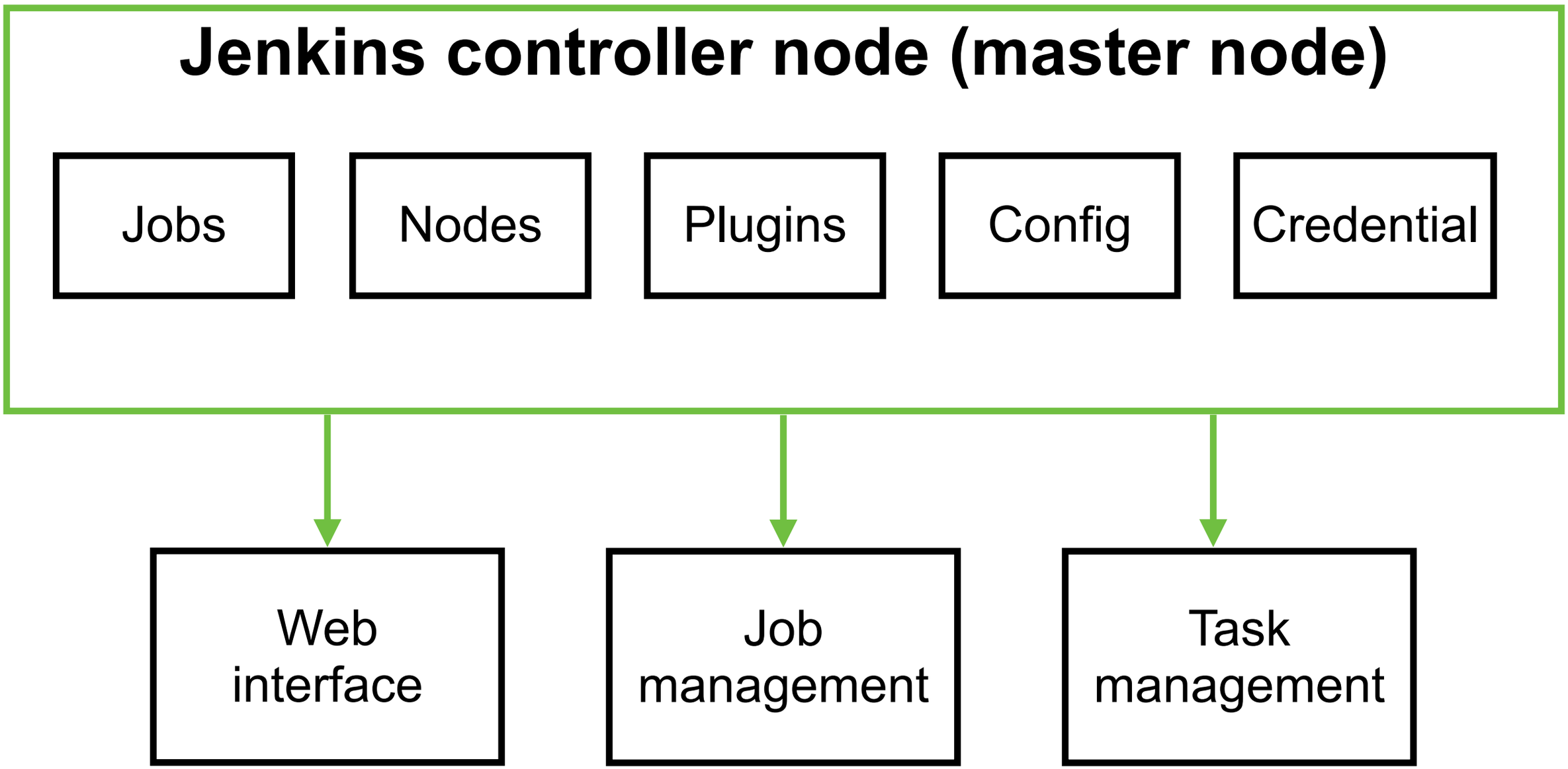
Nodes

Machines where Jenkins executes jobs. (Single to multi)

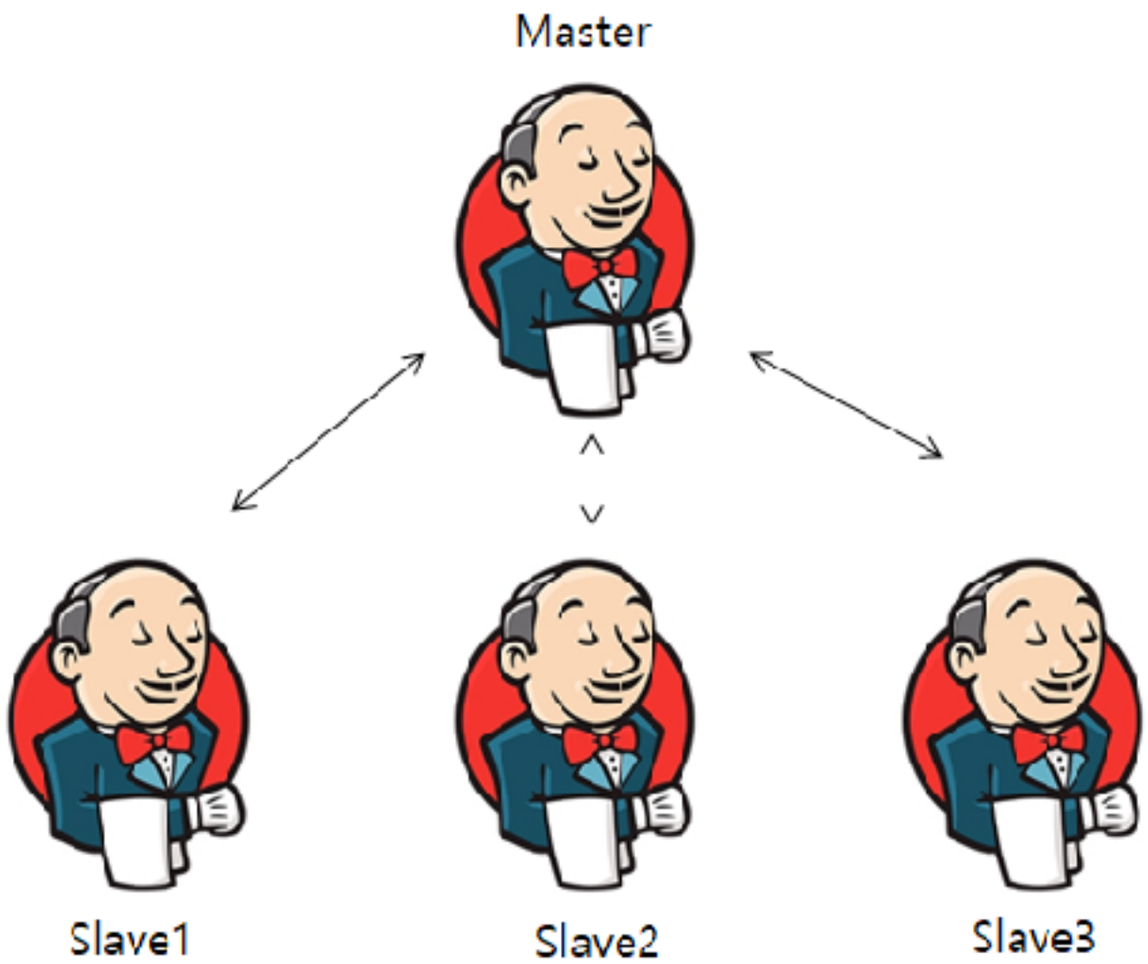
Plugins

Extend Jenkins’s functionality, working with tools and technology

Jenkins Architecture

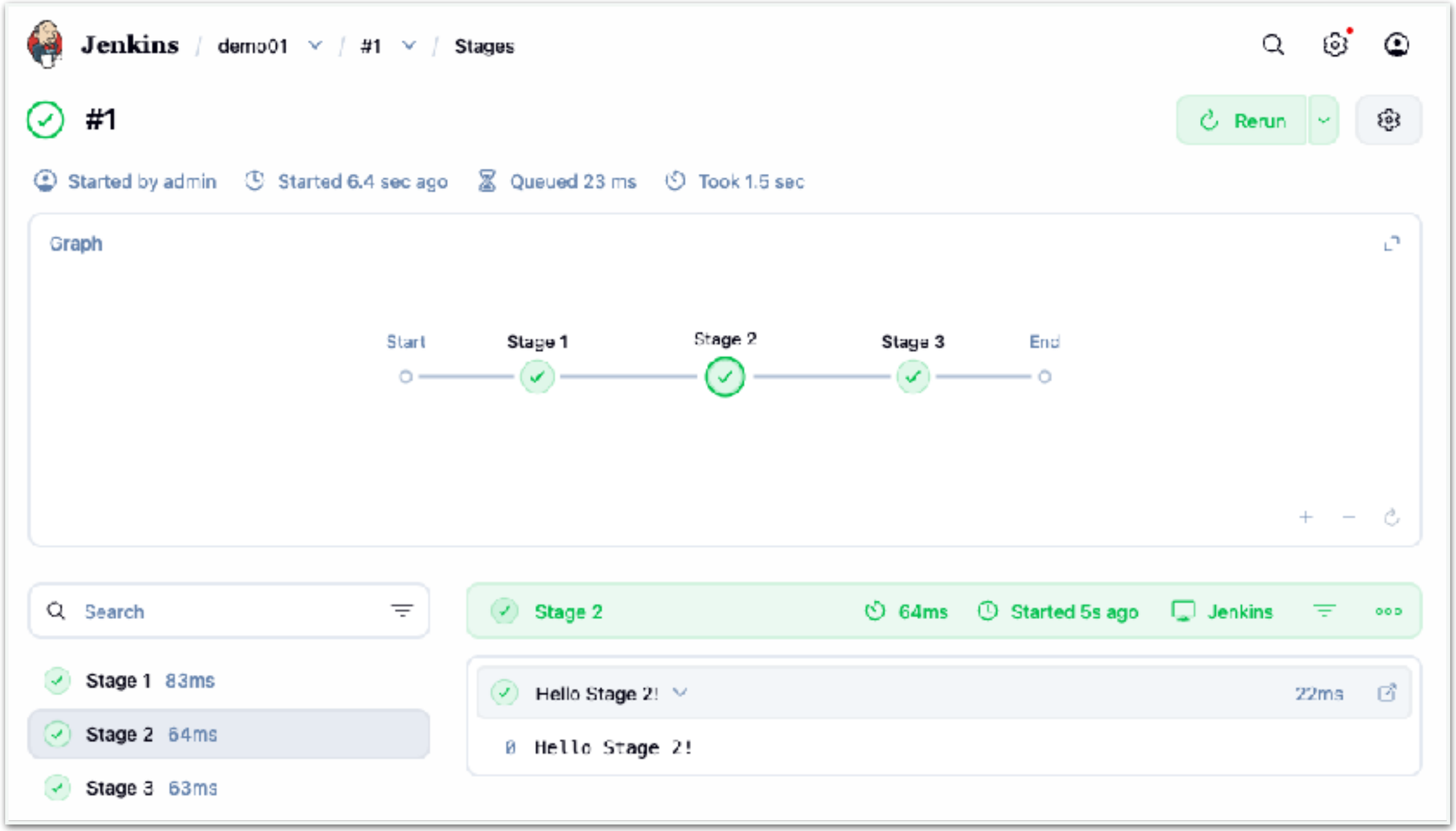


Scaling

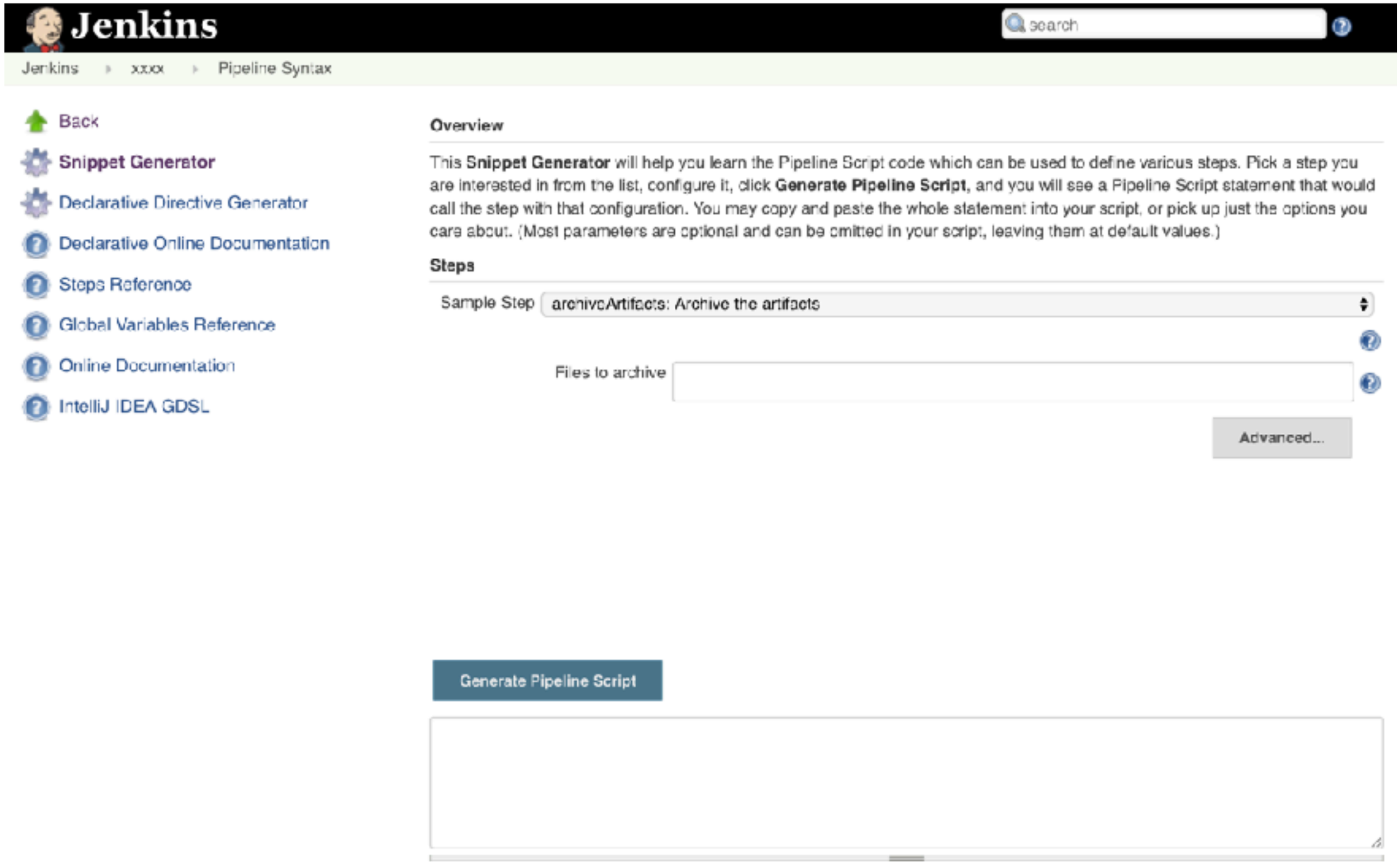


https://www.jenkins.io/doc/book/scaling/

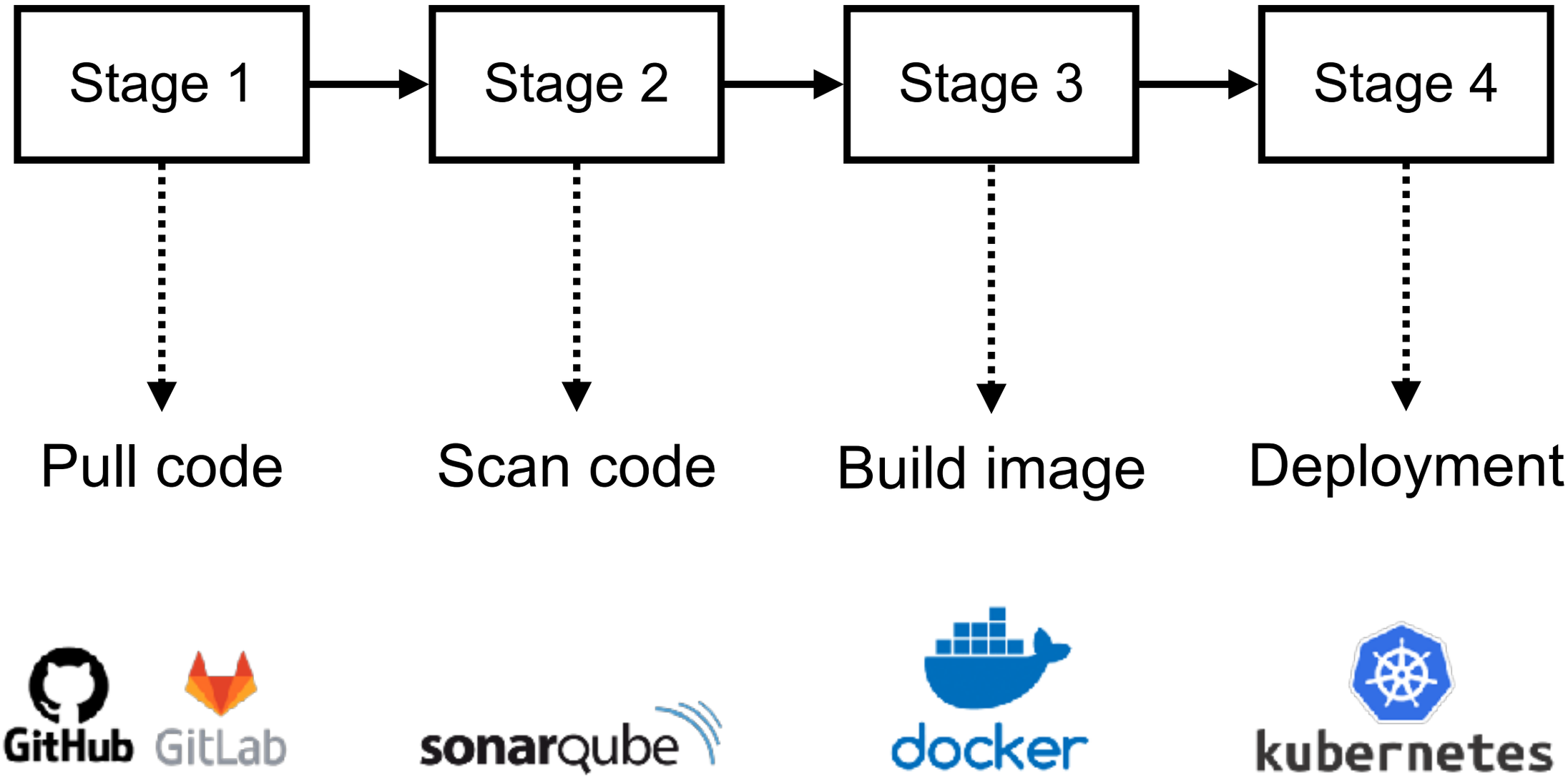
Build pipeline



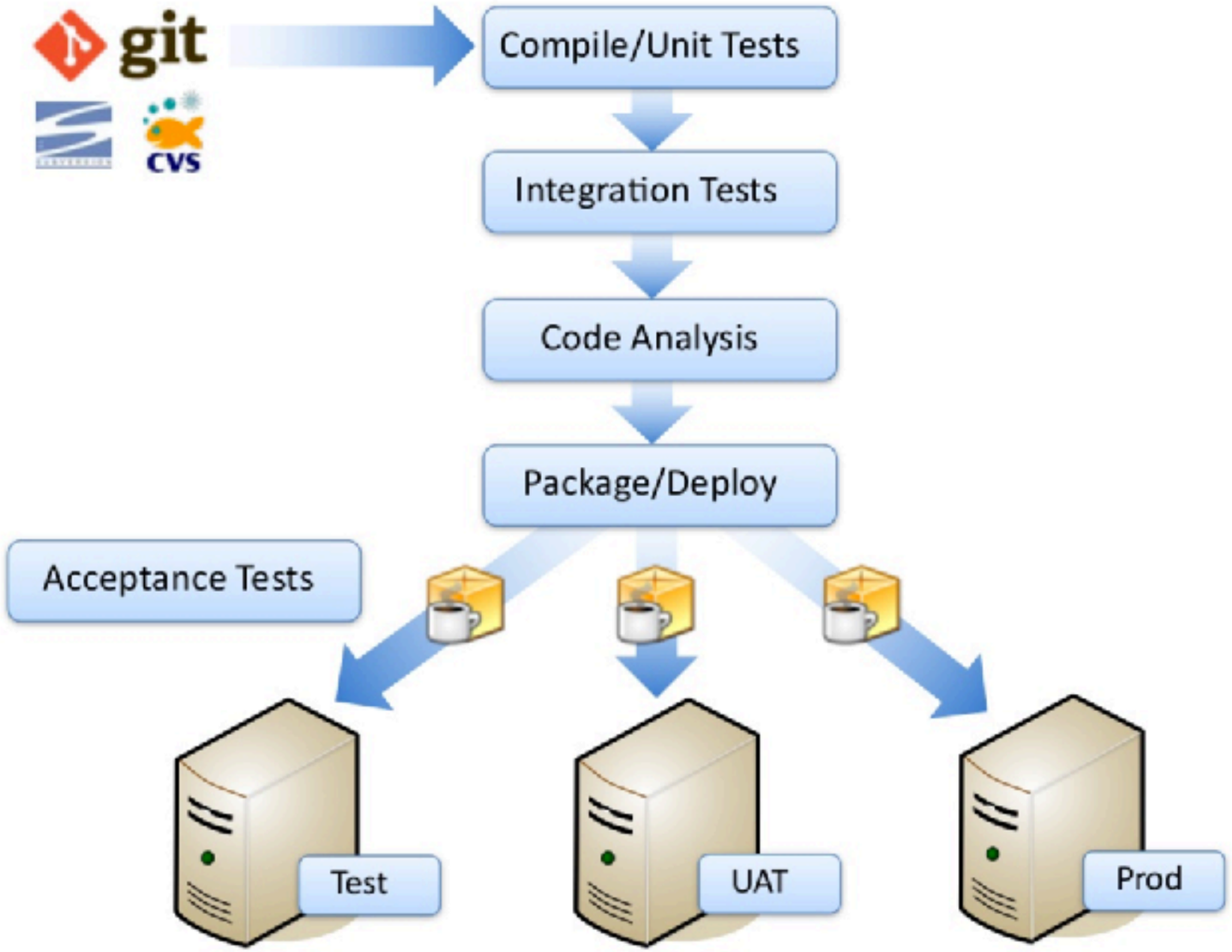
Pipeline syntax



Design your pipeline



Pipeline of this project



List of Tools

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

OWASP Top 10 CI/CD

Top 10
CI/CD
Security
Risks

CICD-SEC-1 **Insufficient Flow Control Mechanisms**

CICD-SEC-2 **Inadequate Identity and Access Management**

CICD-SEC-3 **Dependency Chain Abuse**

CICD-SEC-4 **Poisoned Pipeline Execution (PPE)**

CICD-SEC-5 **Insufficient PBAC (Pipeline-Based Access Controls)**

CICD-SEC-6 **Insufficient Credential Hygiene**

CICD-SEC-7 **Insecure System Configuration**

CICD-SEC-8 **Ungoverned Usage of 3rd Party Services**

CICD-SEC-9 **Improper Artifact Integrity Validation**

CICD-SEC-10 **Insufficient Logging and Visibility**

OWASP

https://owasp.org/www-project-top-10-ci-cd-security-risks/

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

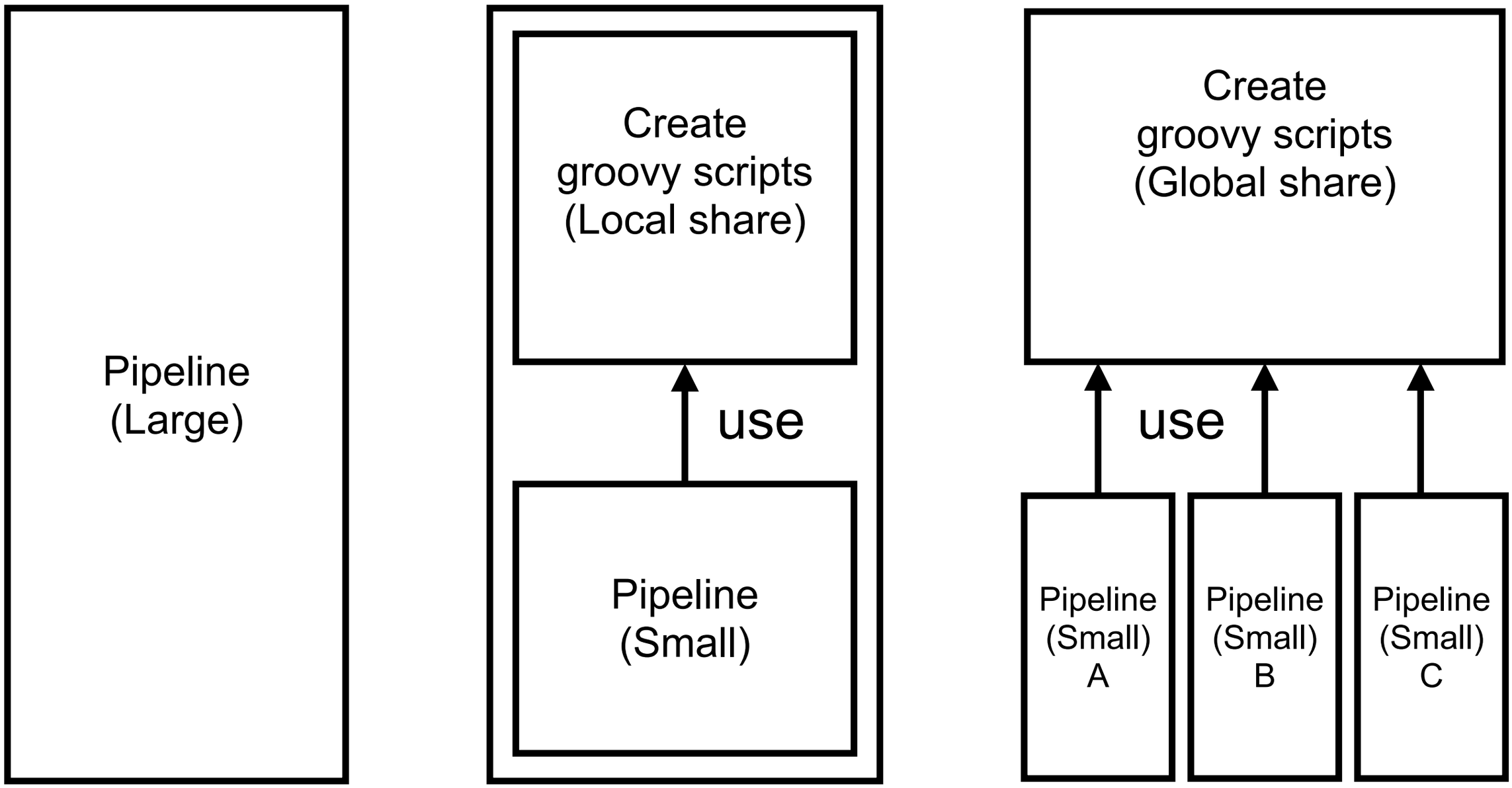
Example pipeline

```
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 ?



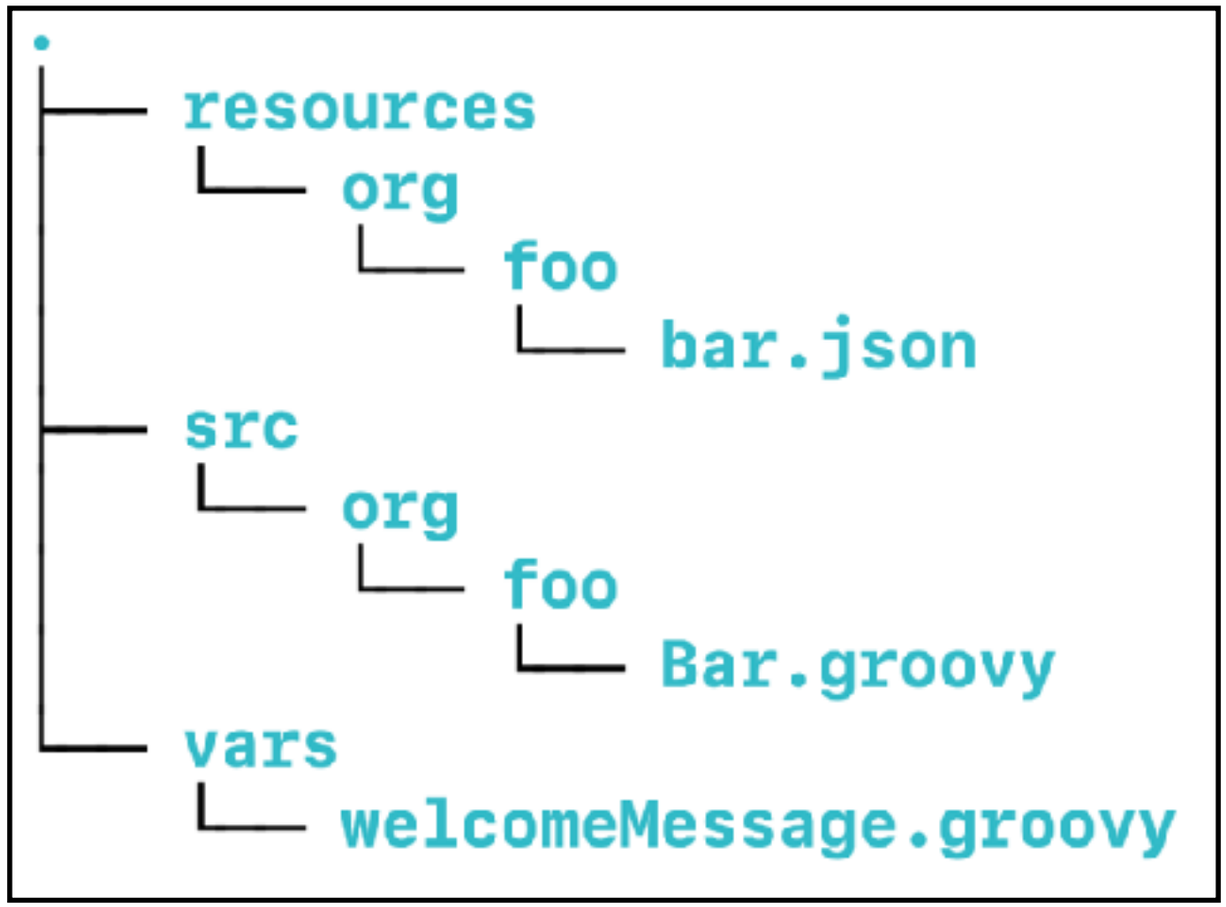
https://github.com/up1/demo-pipeline-nodejs/wiki/Workshop-with-pipeline

Steps to shared library

Repository setup (project structure)
Jenkins configuration with global pipeline library
Write custom script with groovy
Integrate in pipeline !!

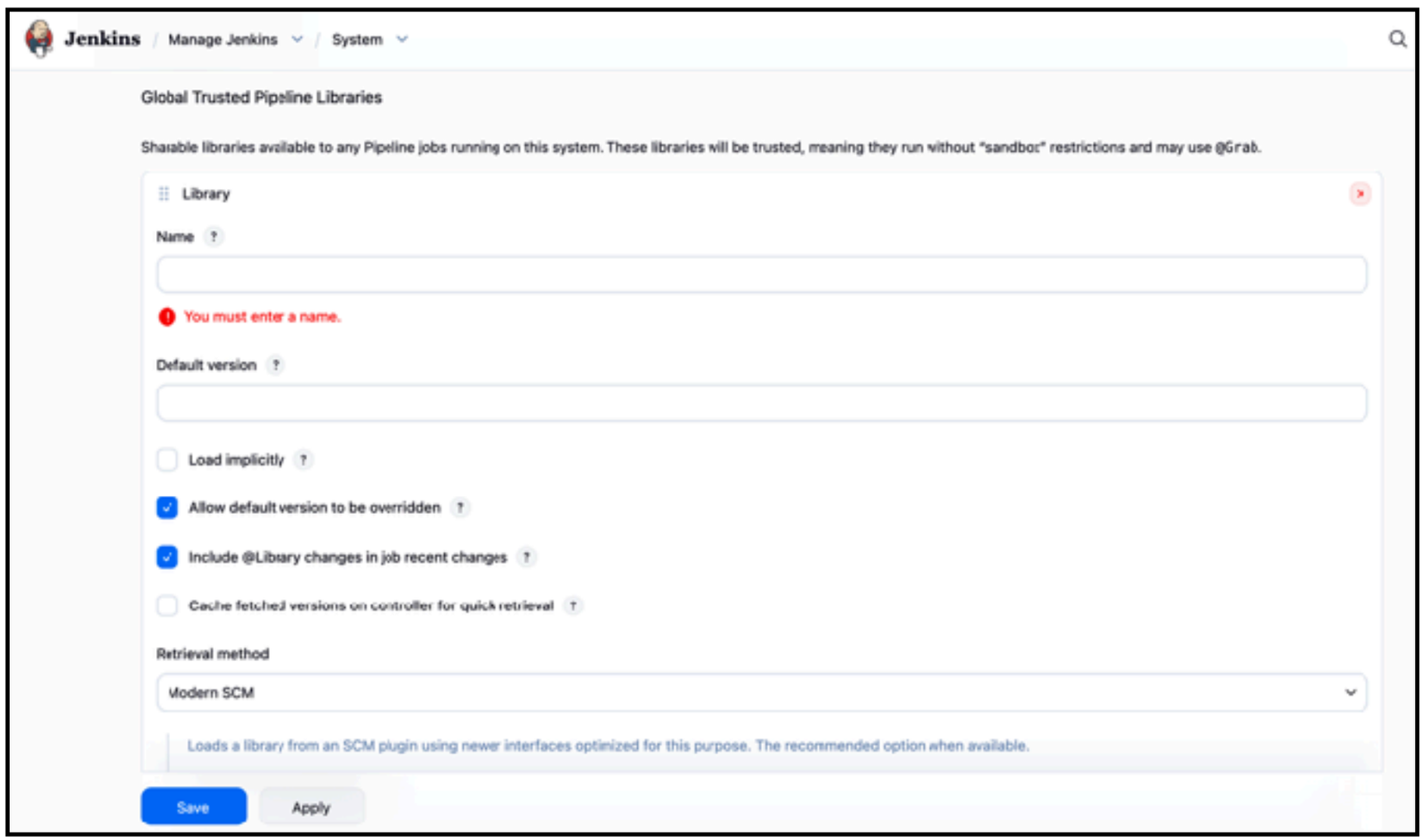
1. Create repository

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