



CI/CD series

# CI/CD with Jenkins





Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามช่างนาฏกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



Facebook somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

Home Posts Videos Photos

Like Following Share ...

+ Add a Button



# Agenda Day 1

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



# Agenda Day 2

1. Working with automated testing
2. Working with code quality
3. Automated deployment with Jenkins
4. Pipeline as a Code with Jenkins
5. Backup and Restore
6. Scalable Jenkins with Master/slave
7. Workshop



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



# Software development



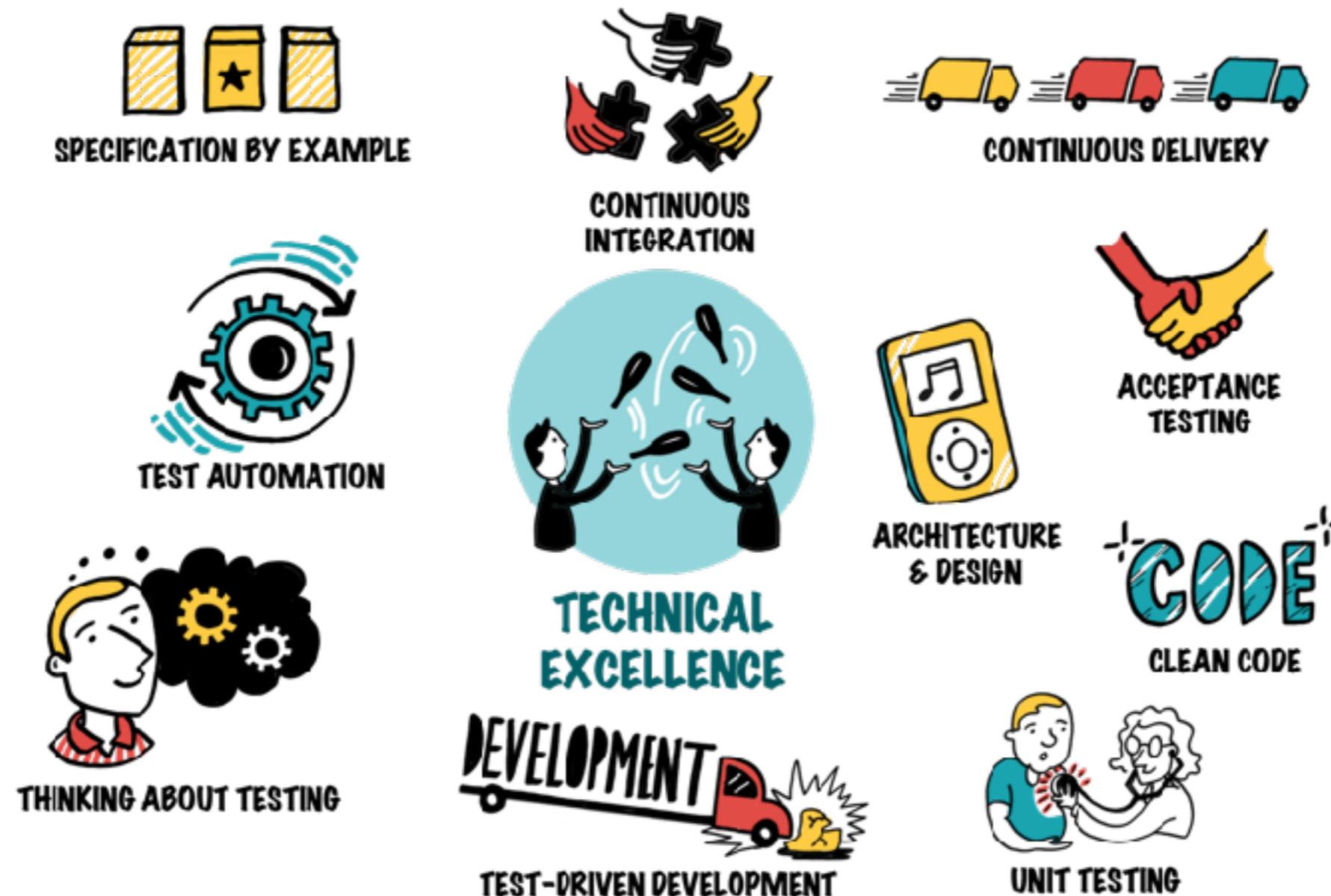
# **Quality vs. Quantity**



# Continuous Integration ?



# Technical Excellence



<http://less.works>

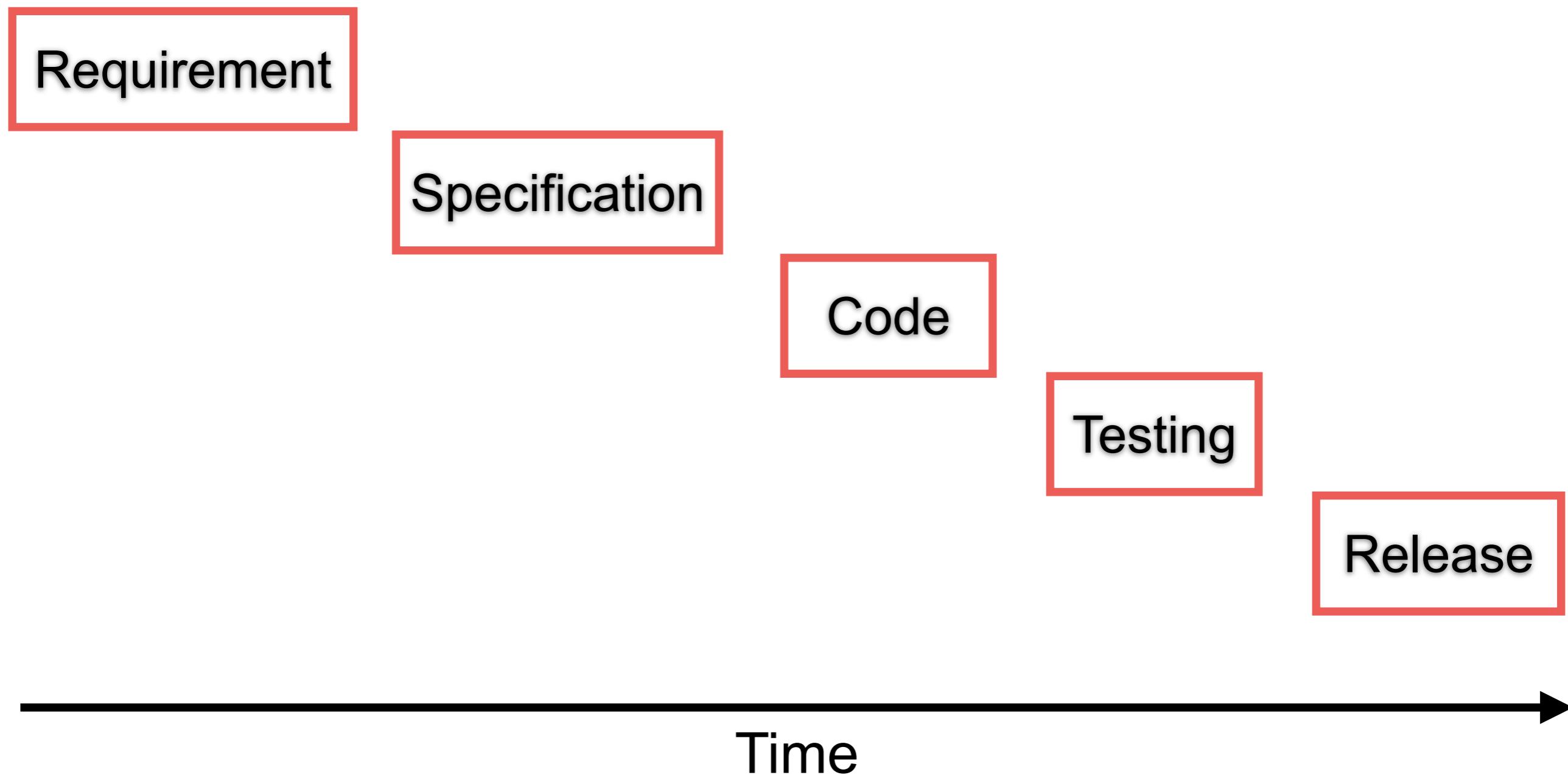
<https://less.works/less/technical-excellence>



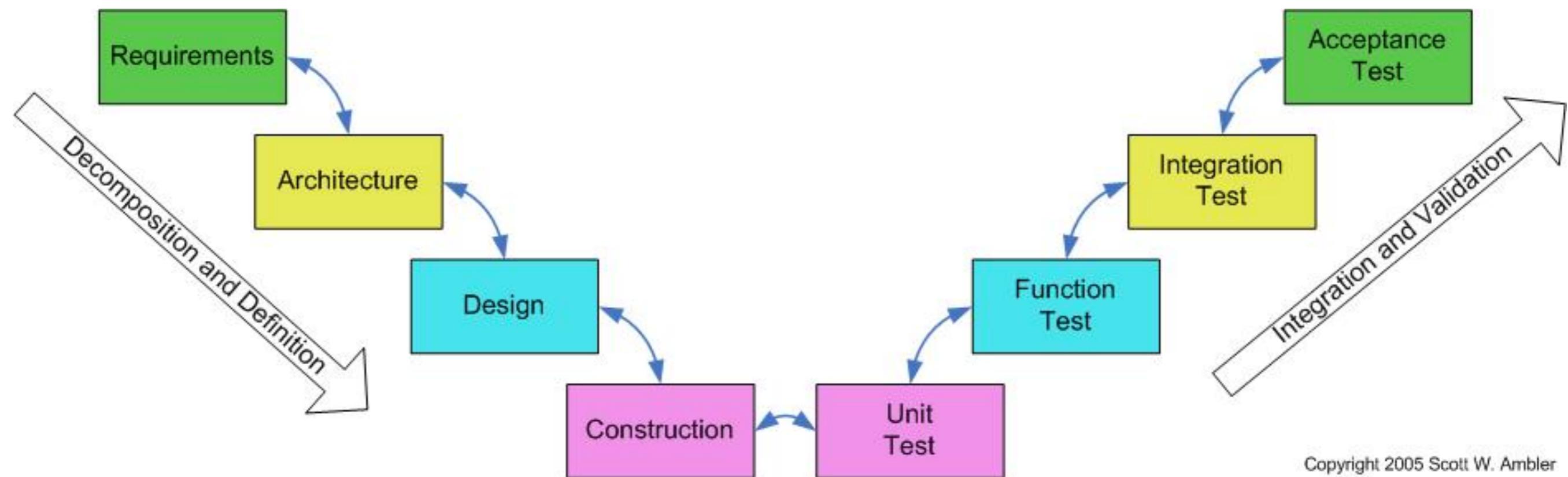
# Why CI/CD ?



# Software Delivery Process



# V Model

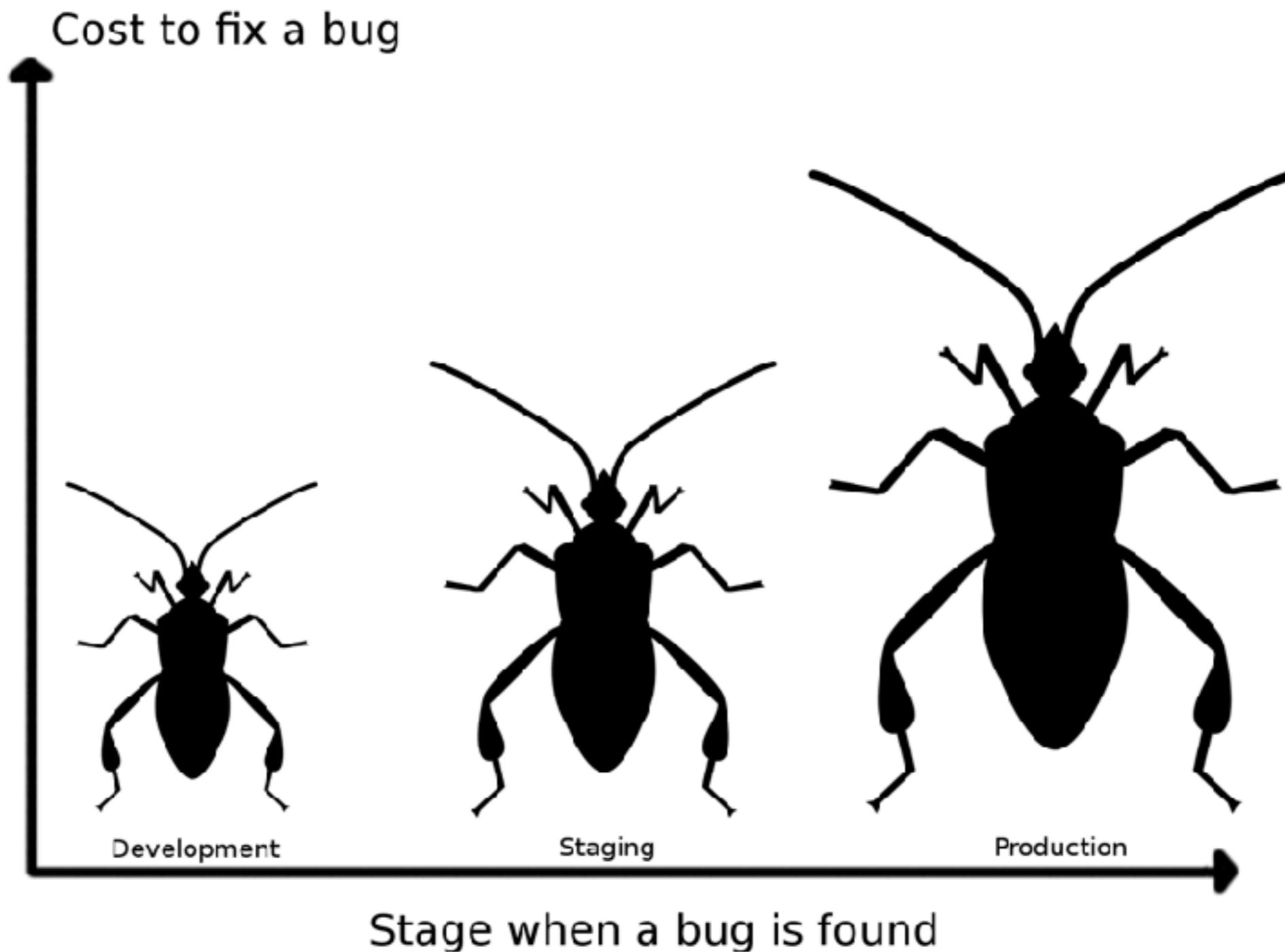


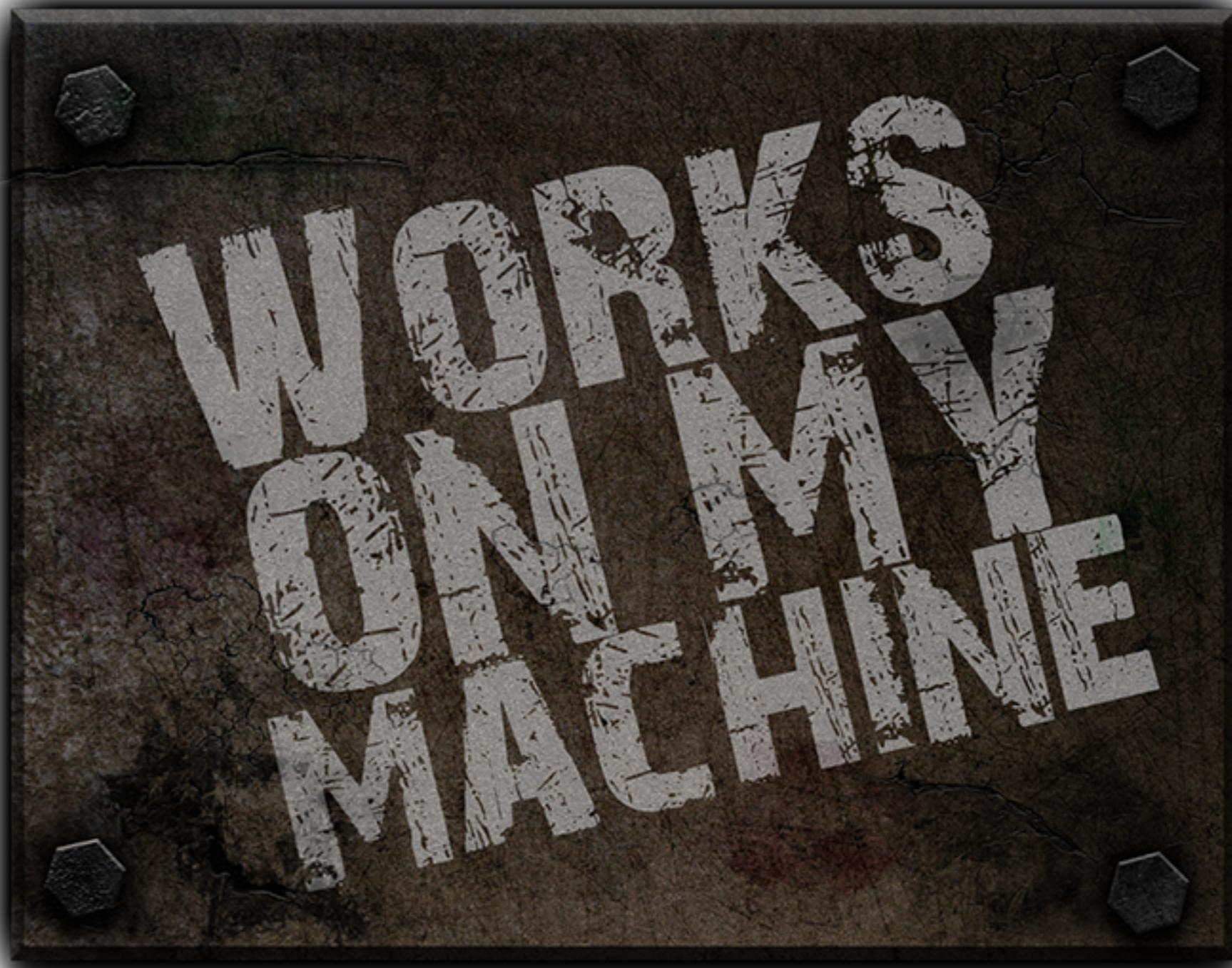
# The cost of integration

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

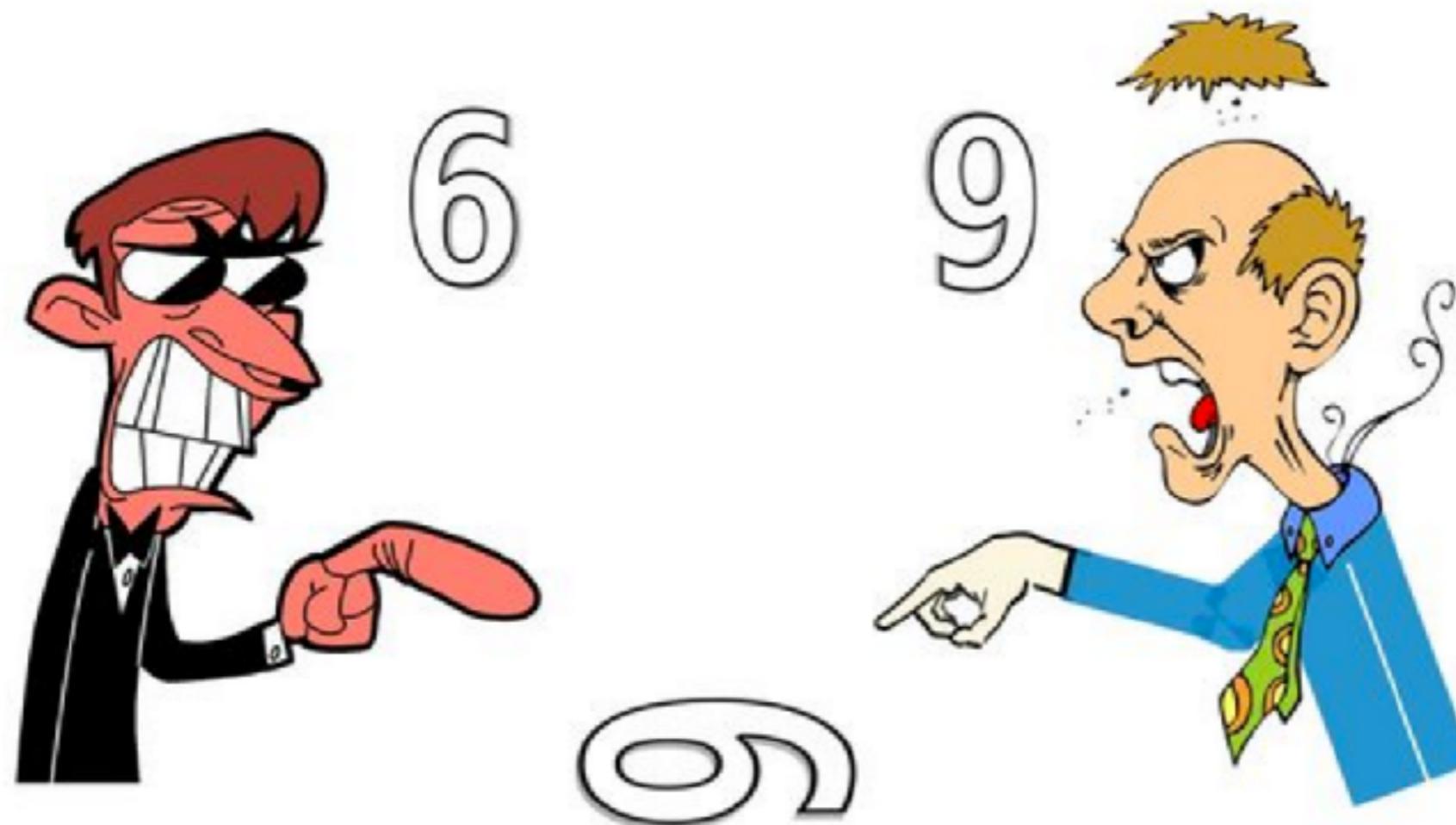


# The cost of integration





# Developer vs Tester





**TONIGHT WE TEST**

**IN PRODUCTION!!!**

memegenerator.net



# Testing process ?

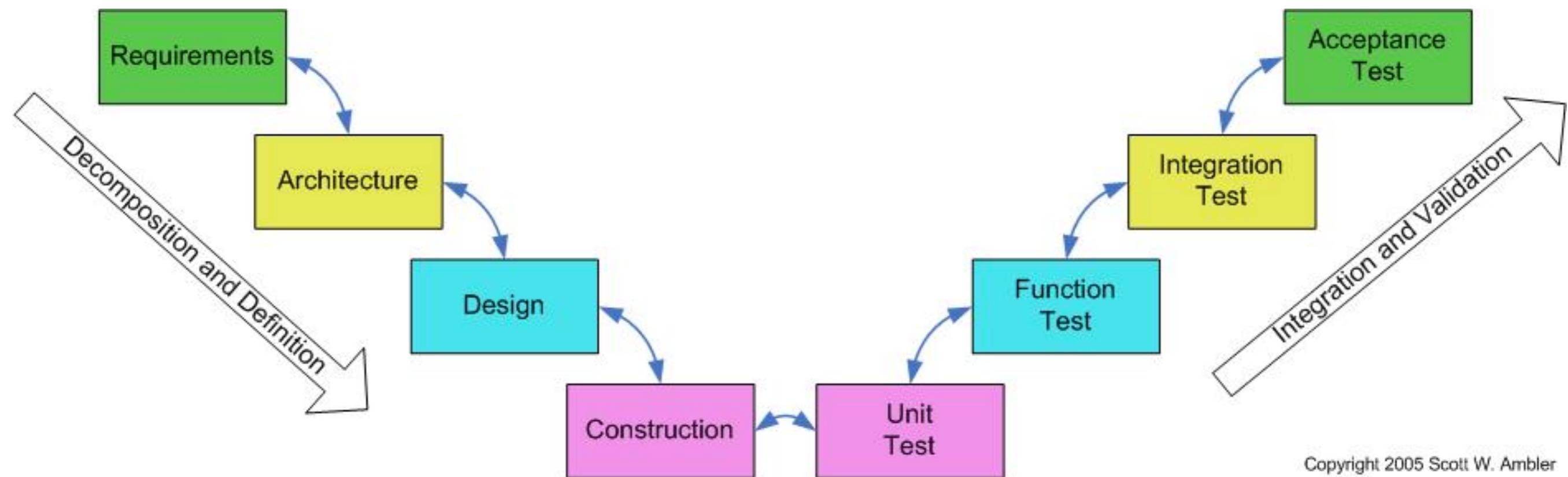
Test first

Test last

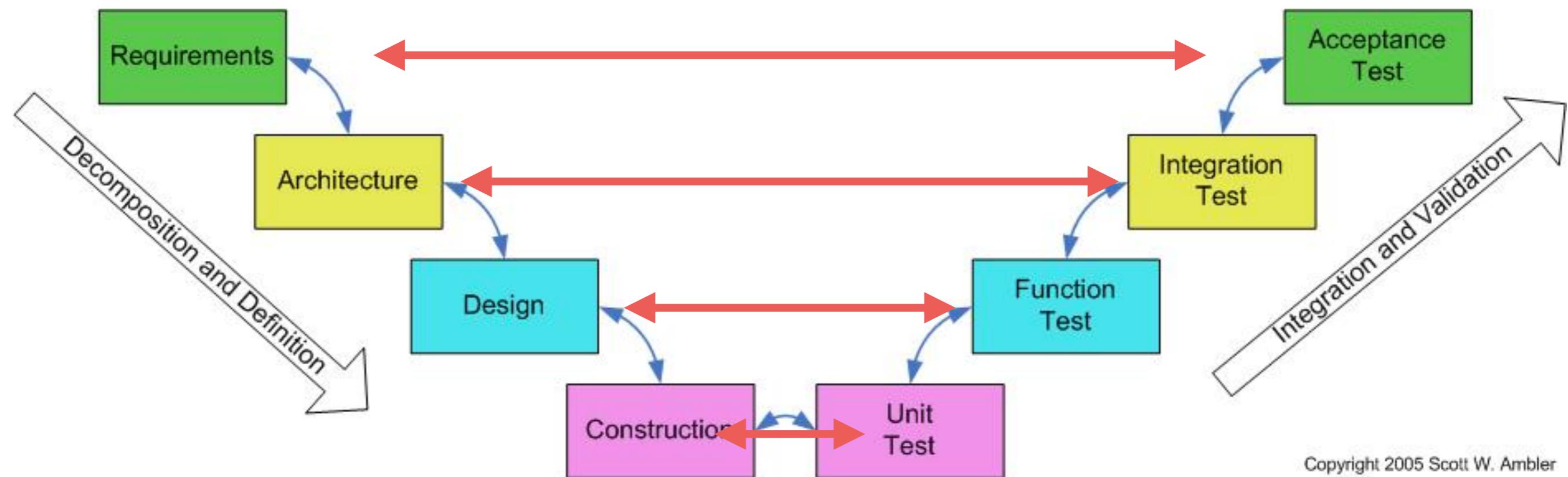
Test later



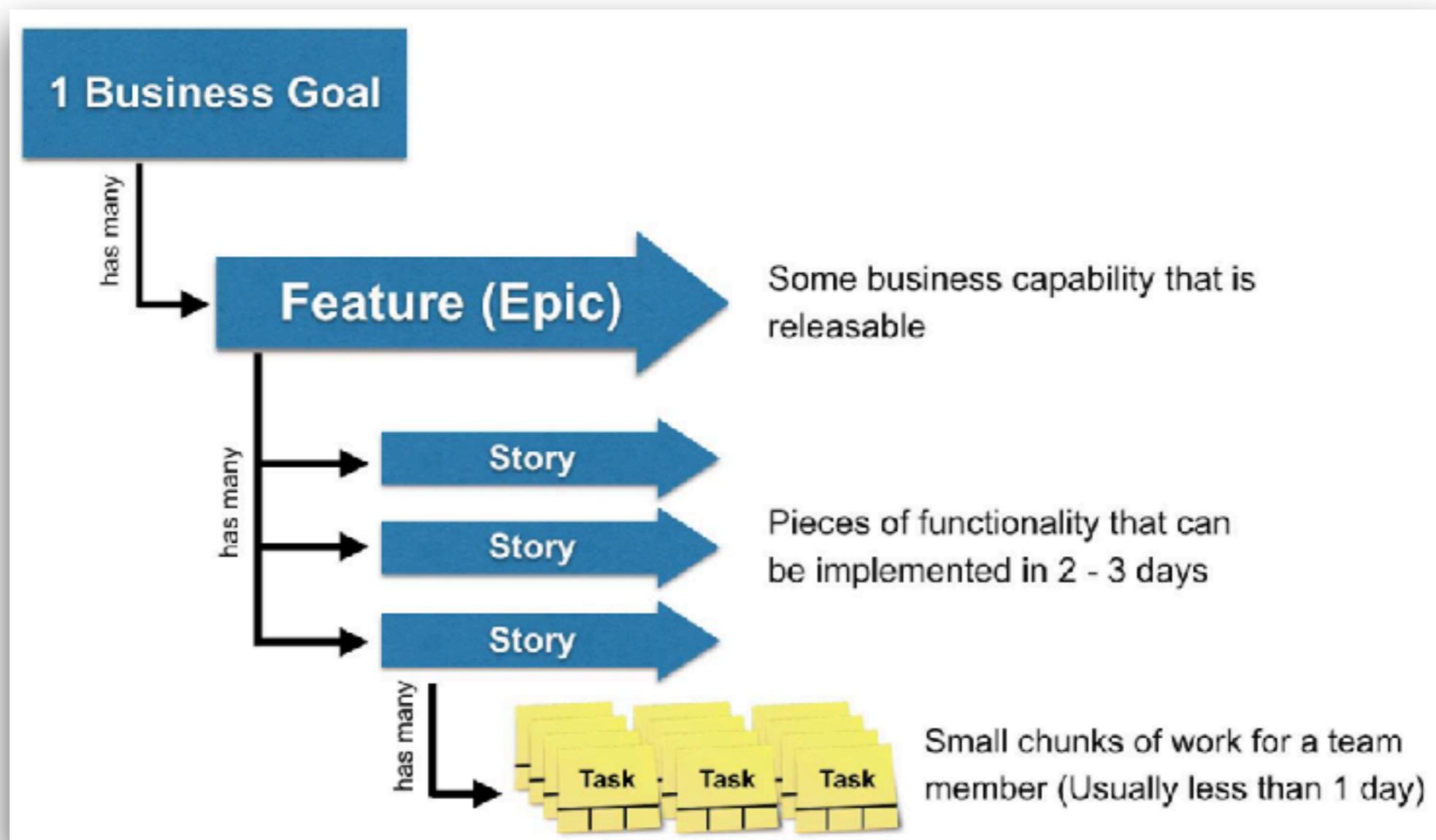
# V Model



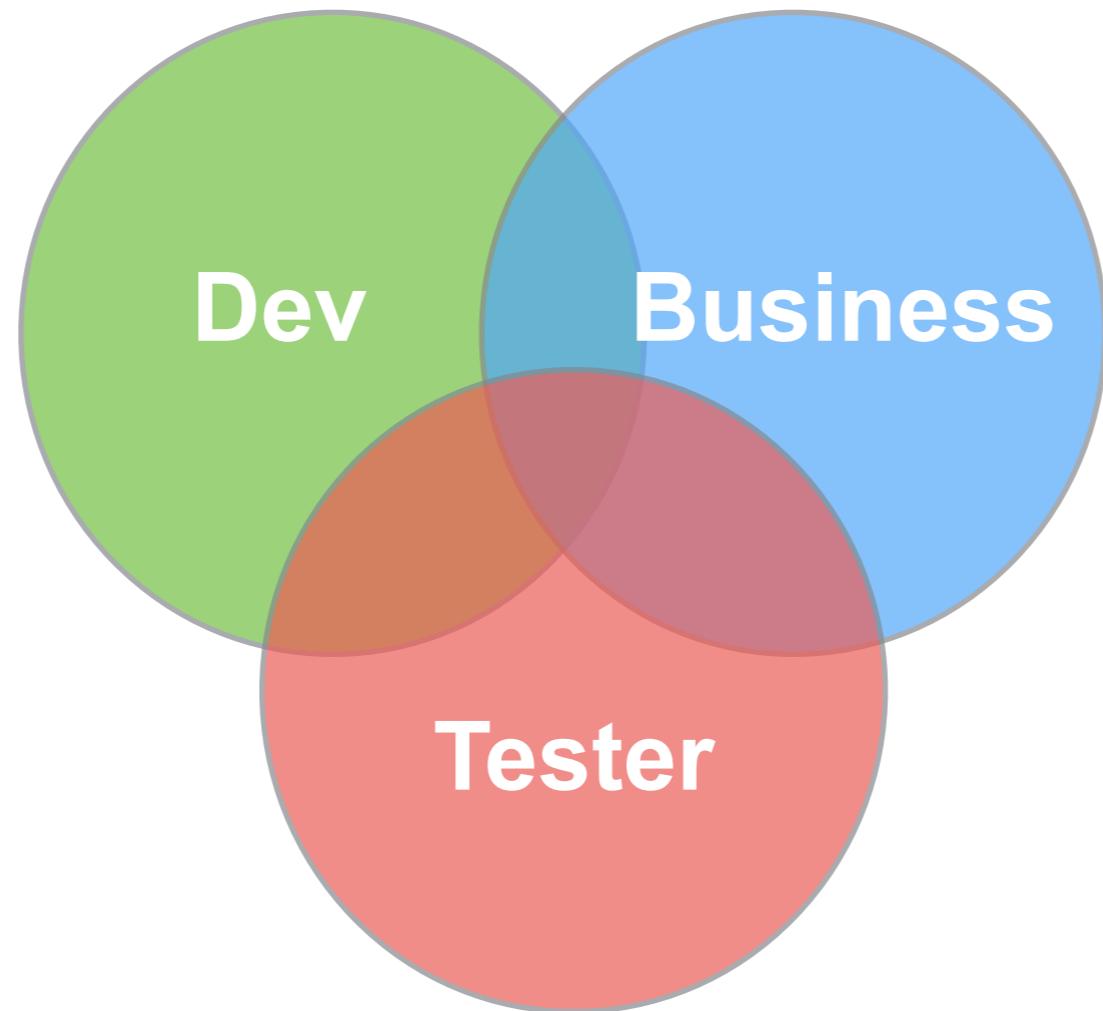
# V Model



# Start with business goal to tasks



# Power of Three



**Test case**  
= **Business Criteria**  
+ **Examples (real data)**



# **Quality process with Test ?**



# Iterative and incremental process

Feature 1

Time



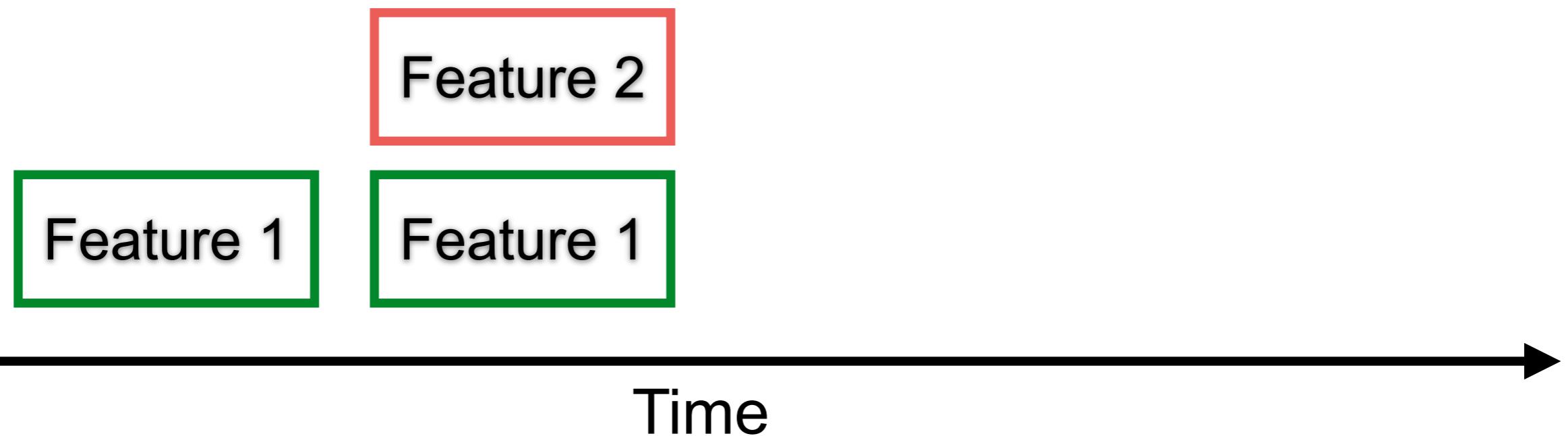
# Iterative and incremental process

Done = coded and tested



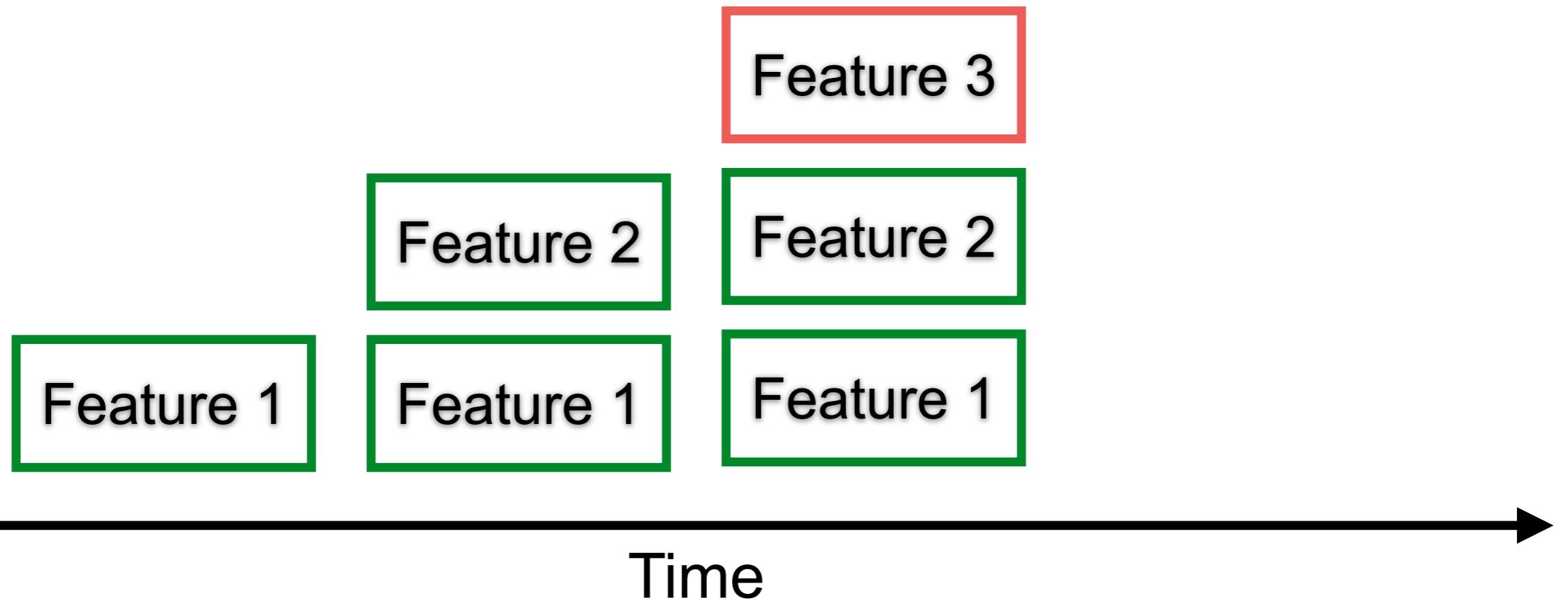
# Iterative and incremental process

Done = coded and tested



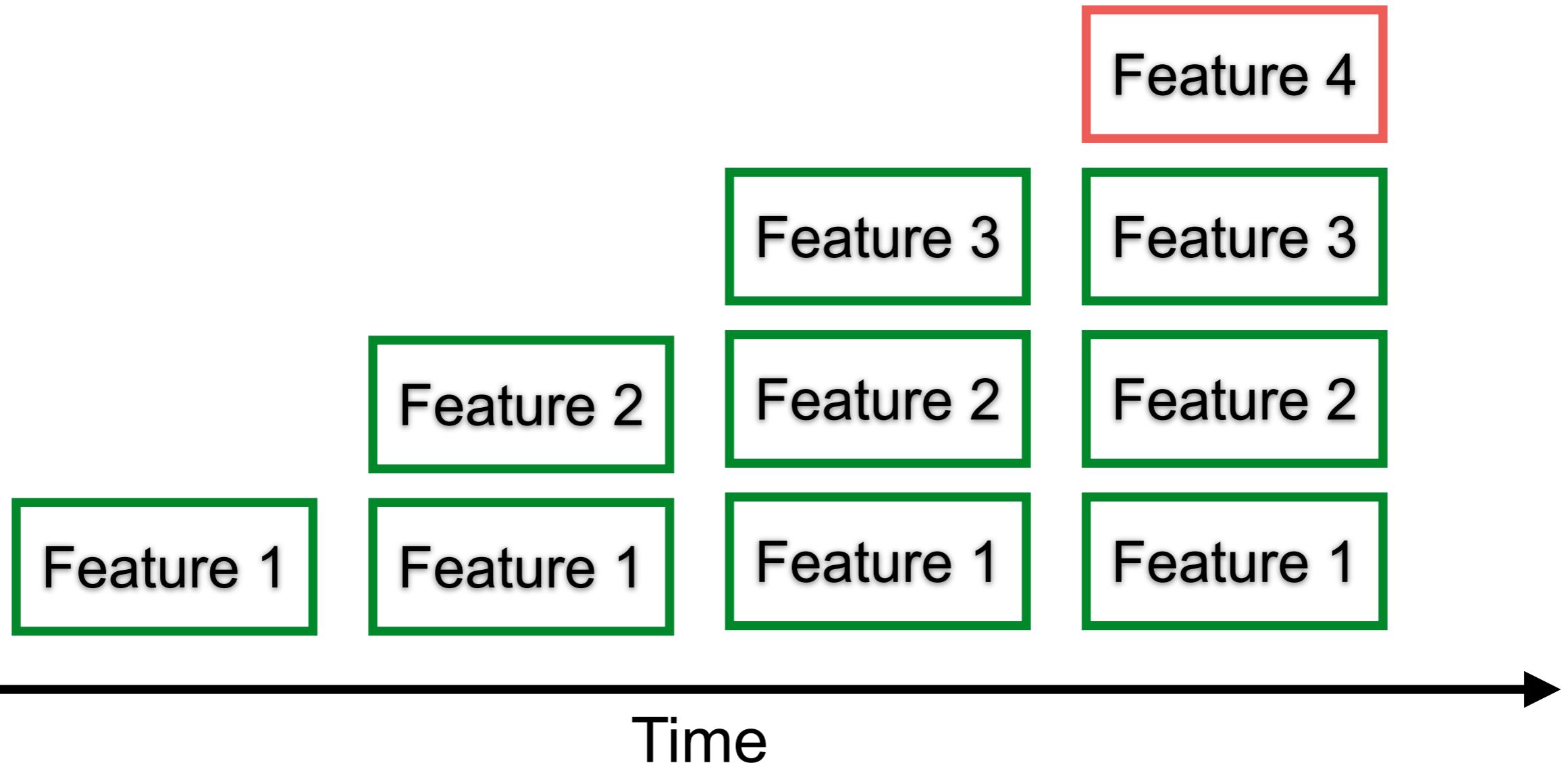
# Iterative and incremental process

Done = coded and tested



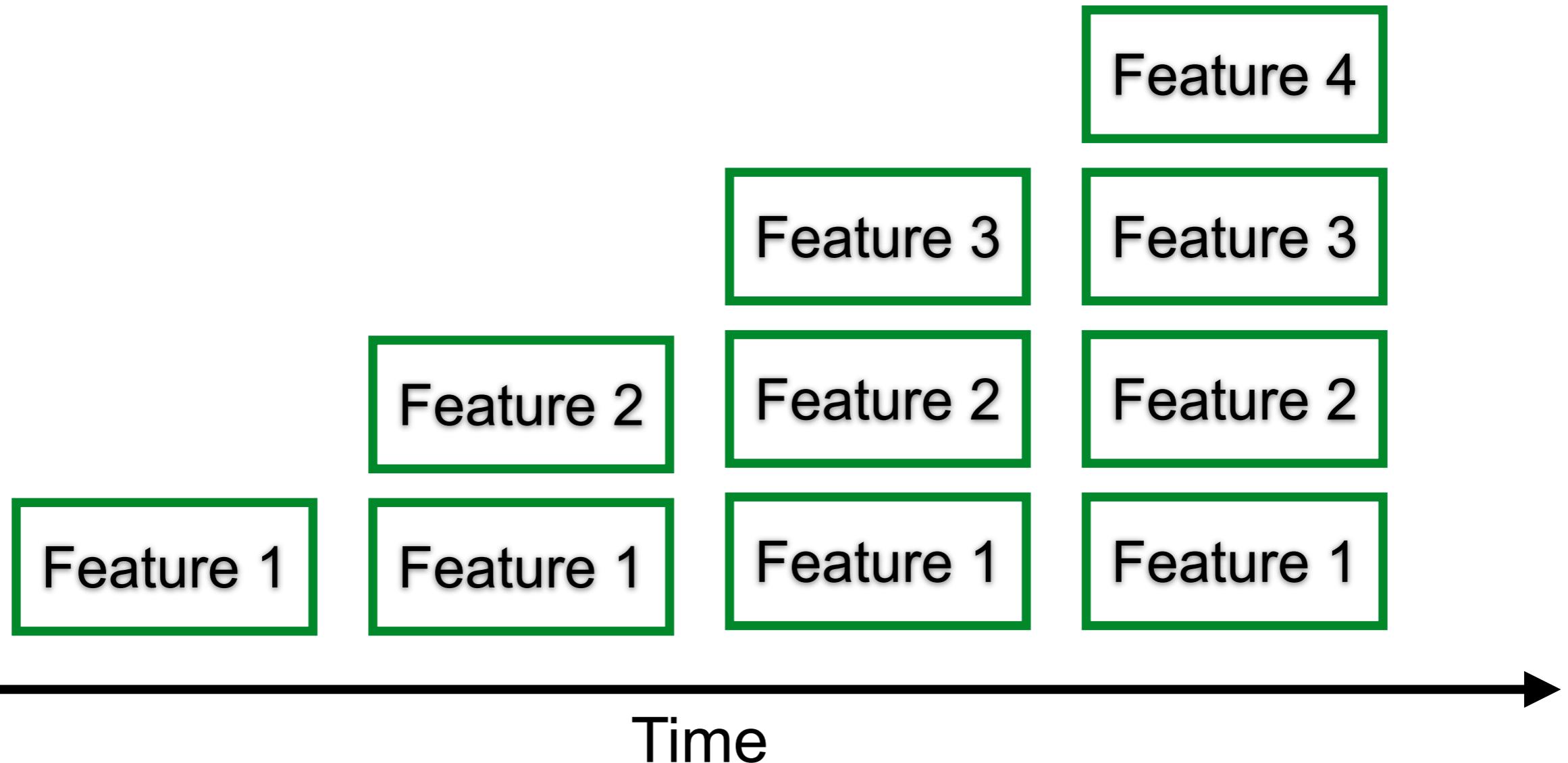
# Iterative and incremental process

Done = coded and tested



# Iterative and incremental process

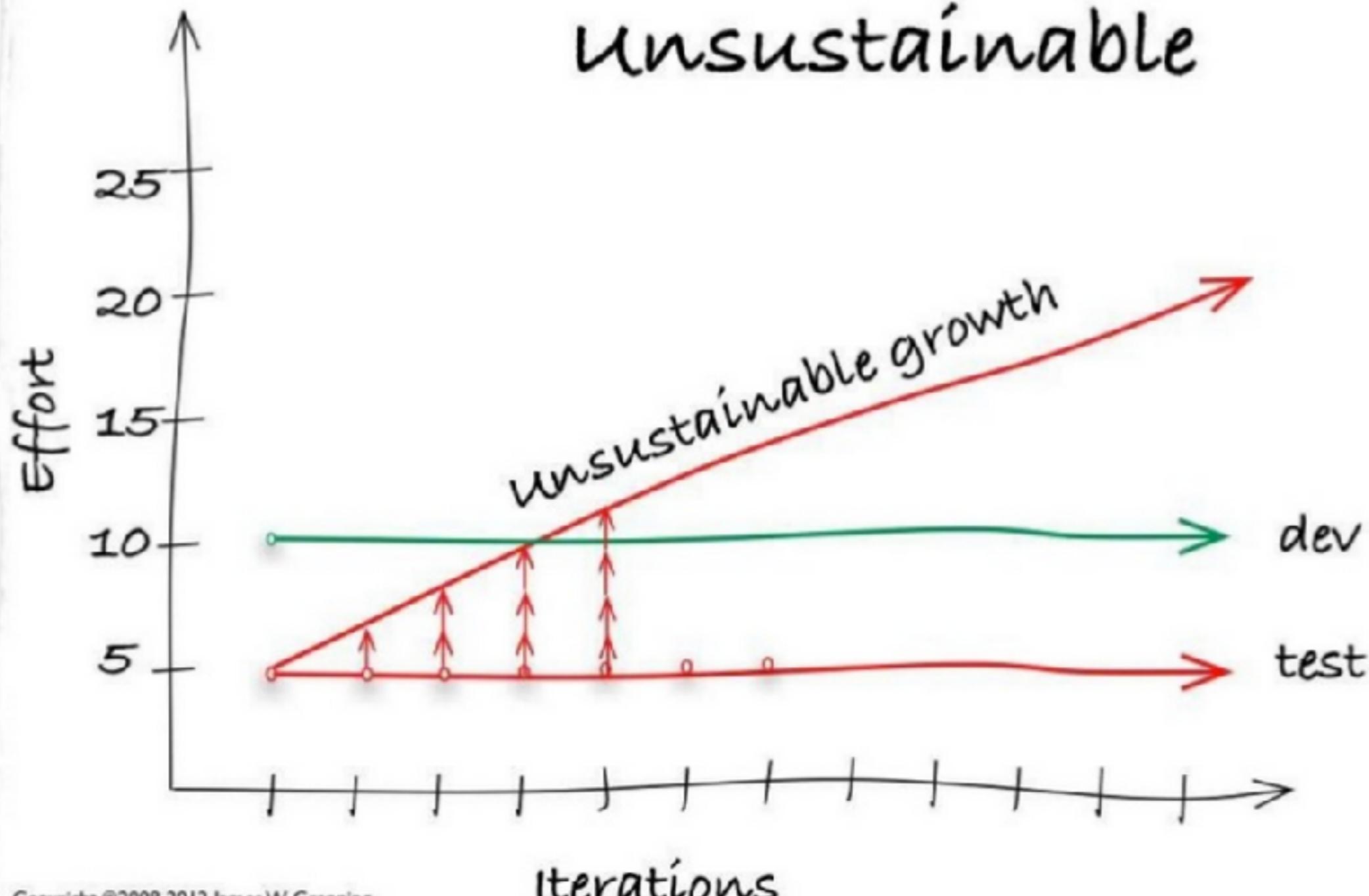
Done = coded and tested



# Manual testing ?



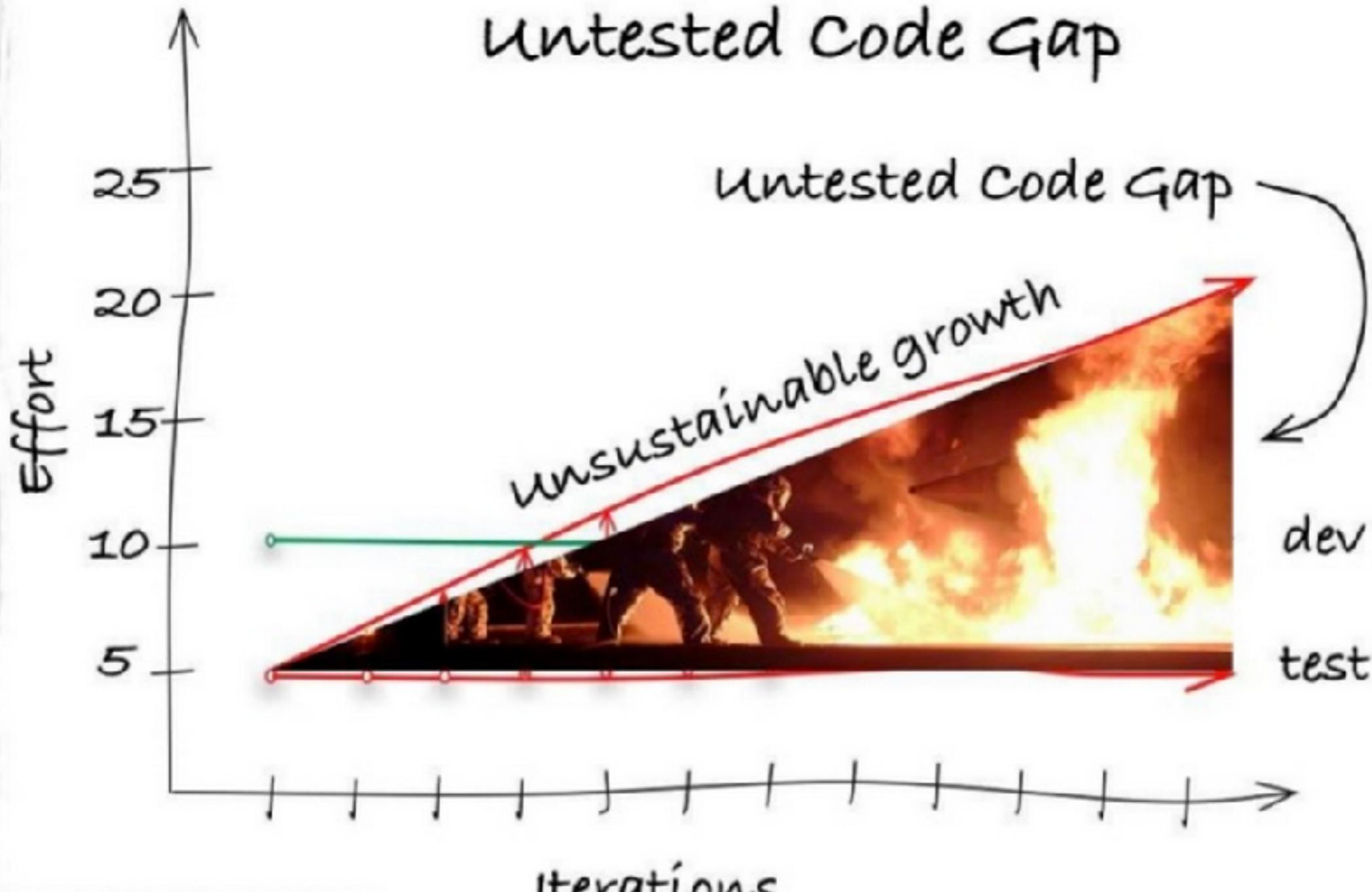
# Manual Test is unsustainable



Copyright ©2008-2012 James W. Grenning  
All Rights Reserved.



# Risk Accumulates in the Untested Code Gap



Copyright ©2008-2012 James W. Grenning  
All Rights Reserved.

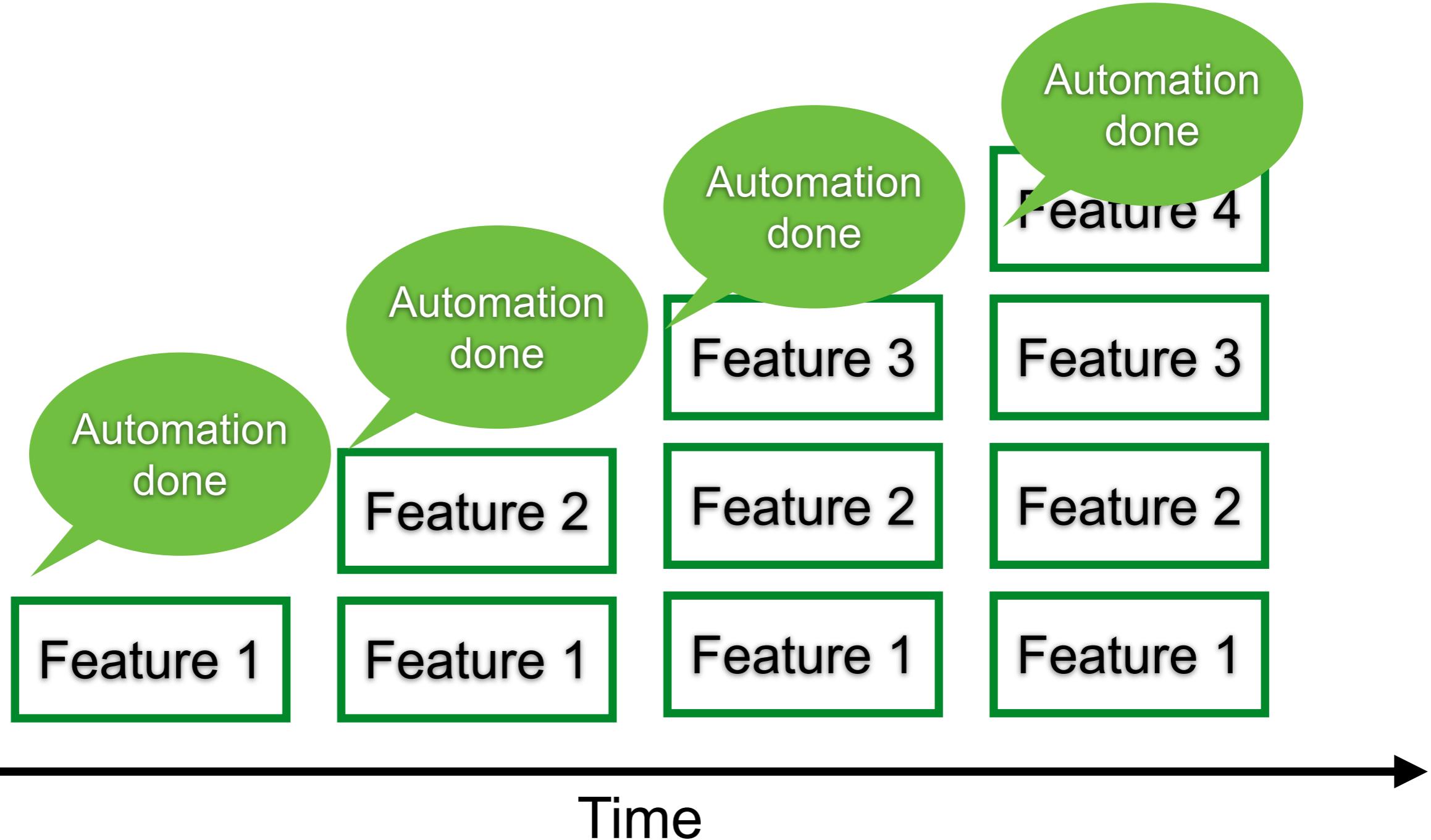


# We need automation testing

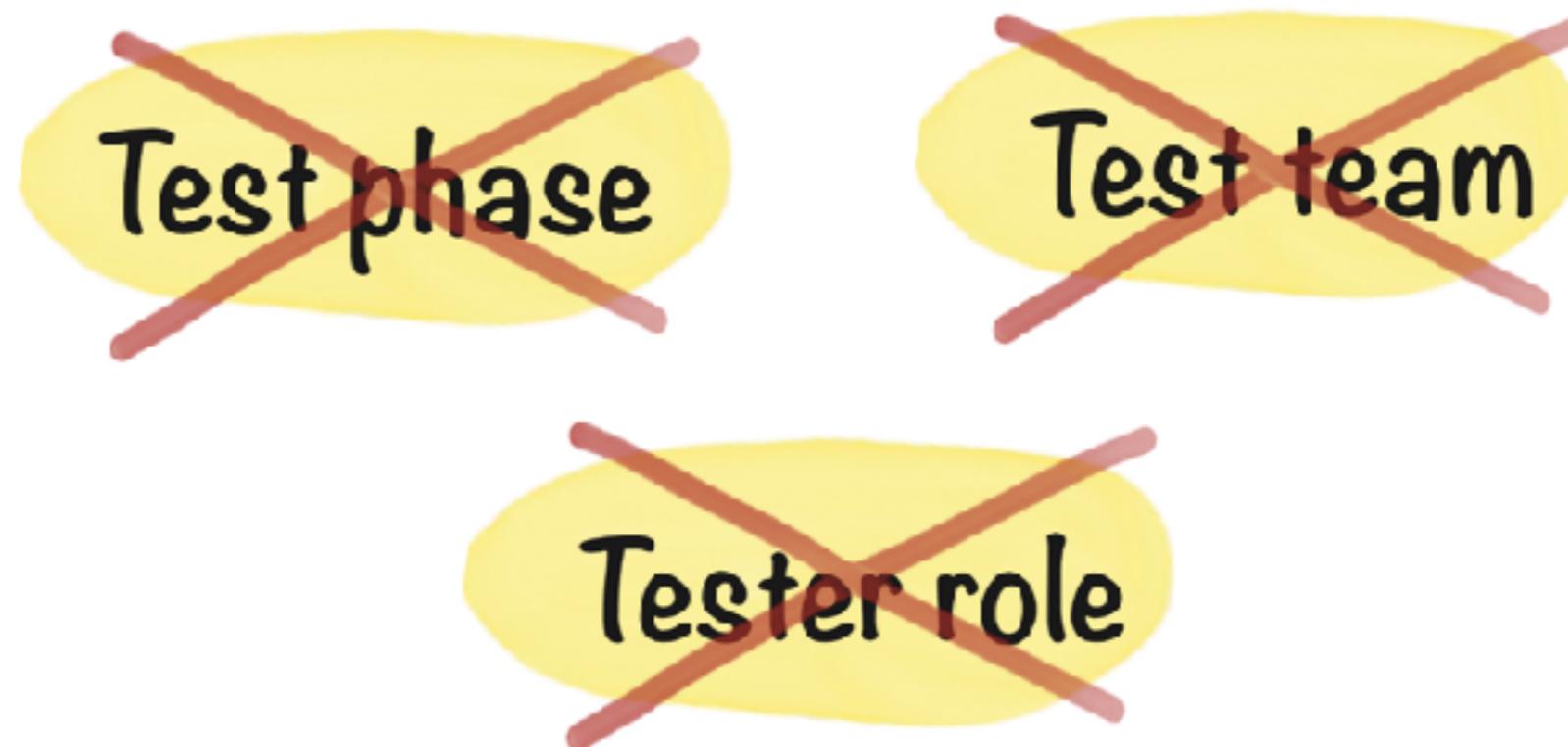


# Iterative and incremental process

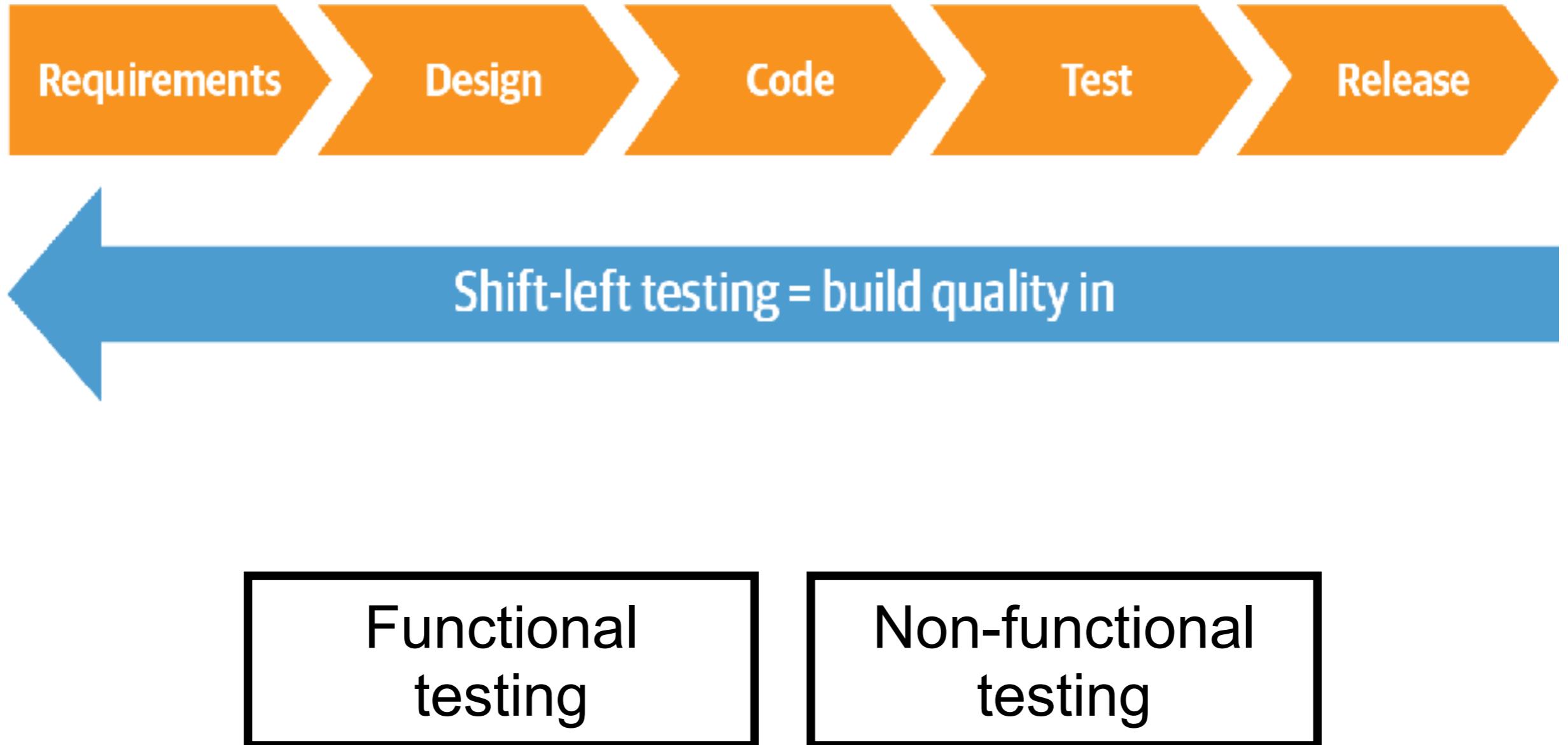
Done = coded and tested



# Test is activity



# Shift Left Testing



# Good Test

F.I.R.S.T + U

Fast

Isolate

Repeat

Self-verify

Timely

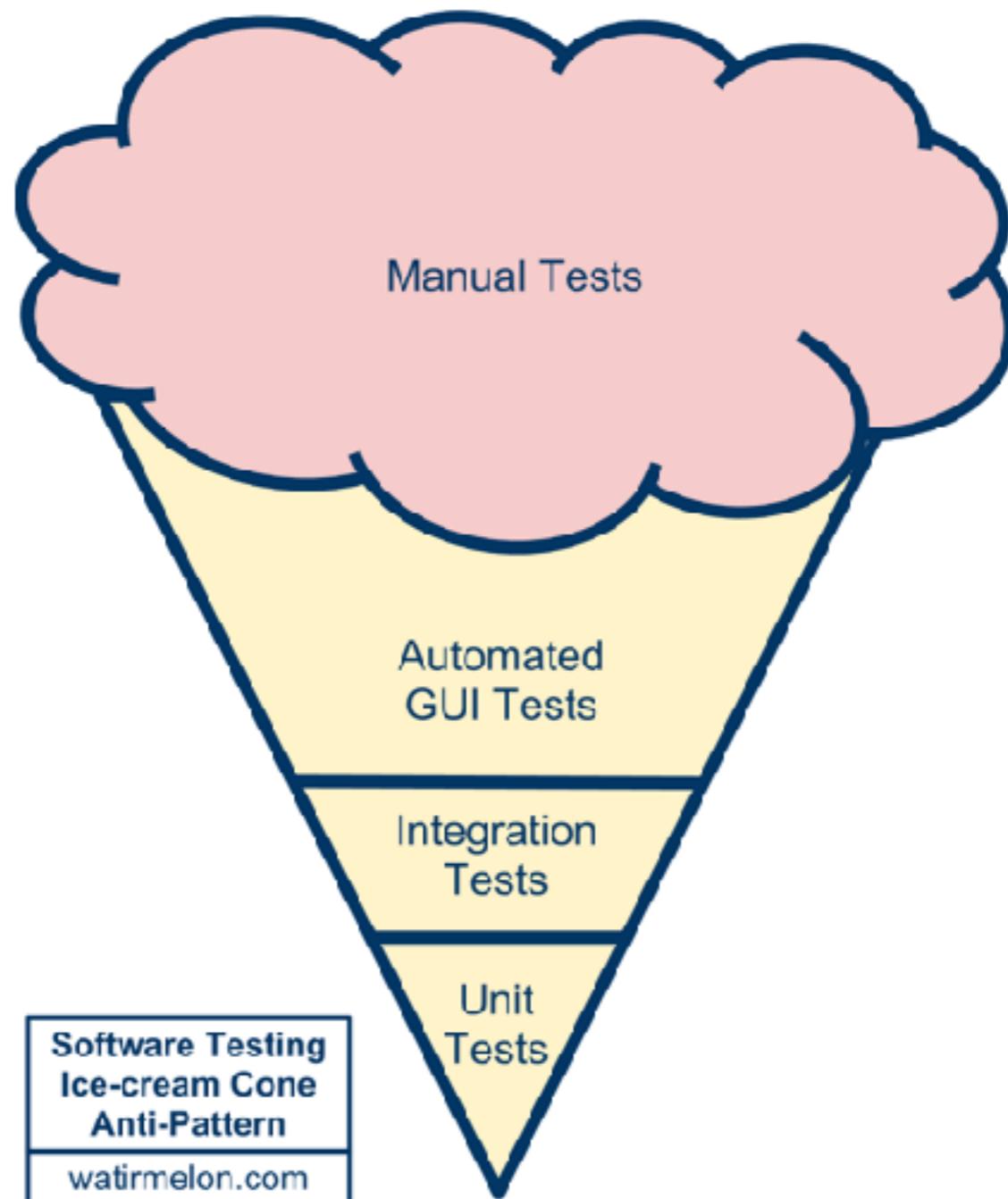
Understand



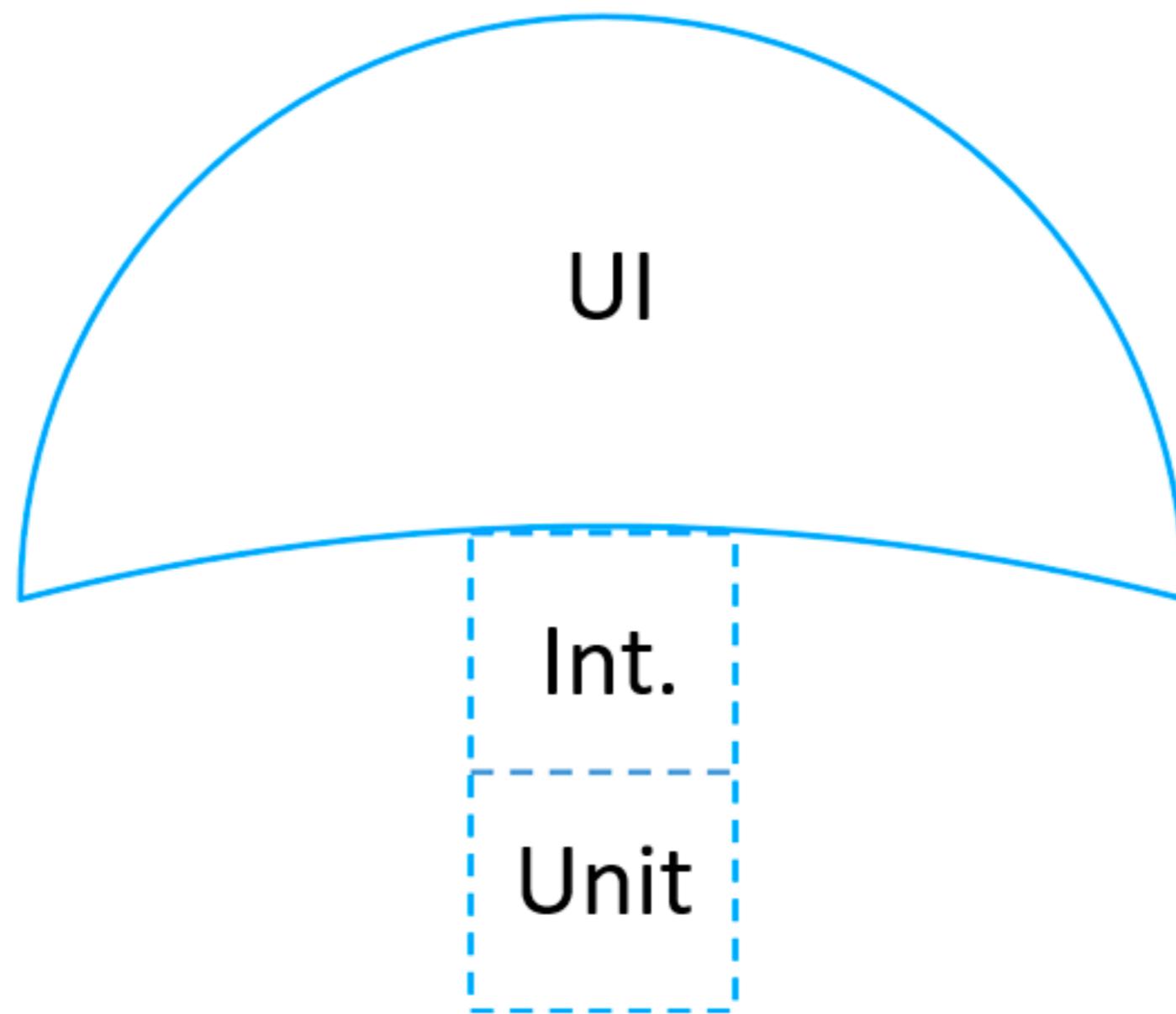
# Testing ?



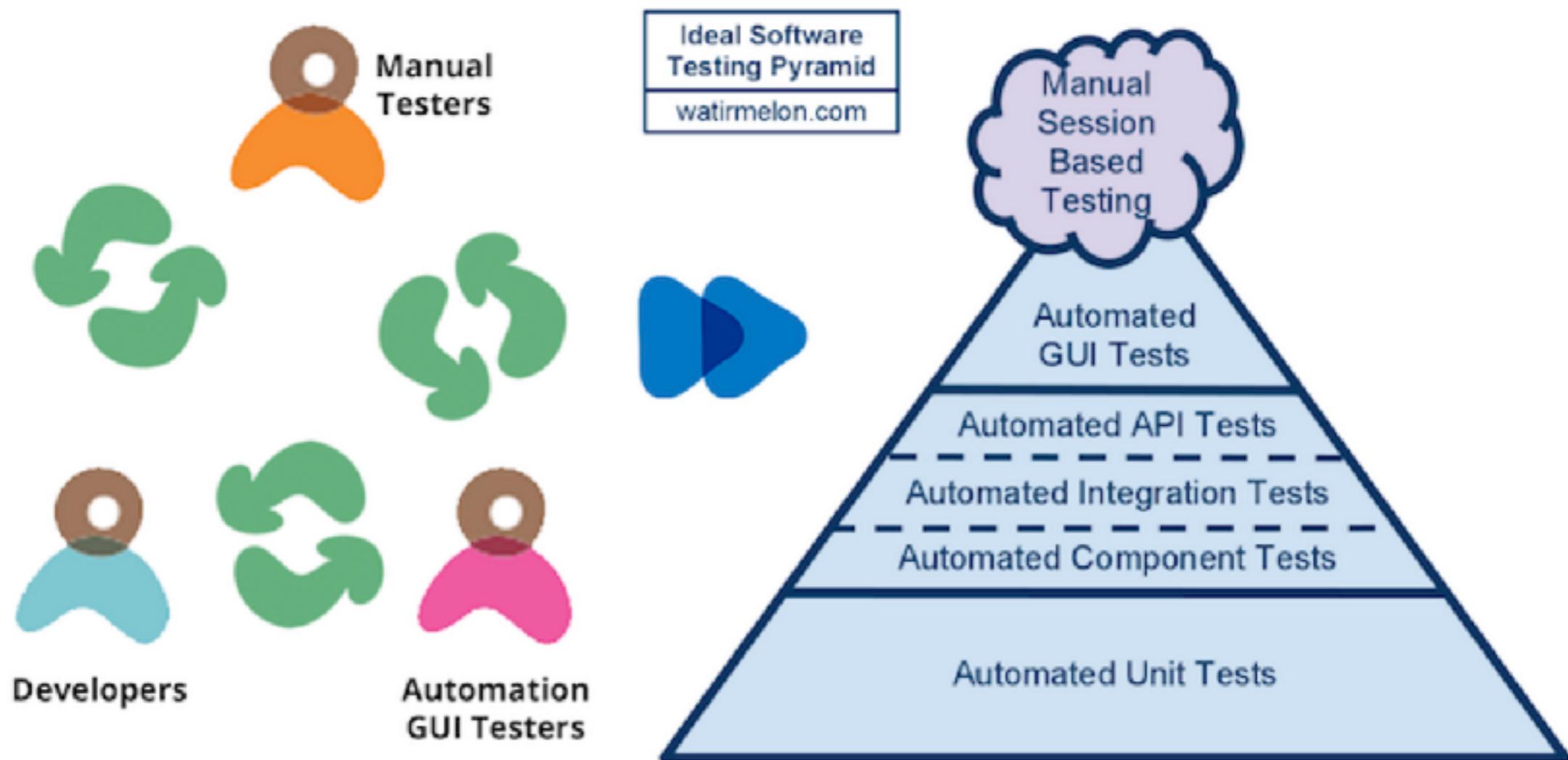
# Ice cream testing



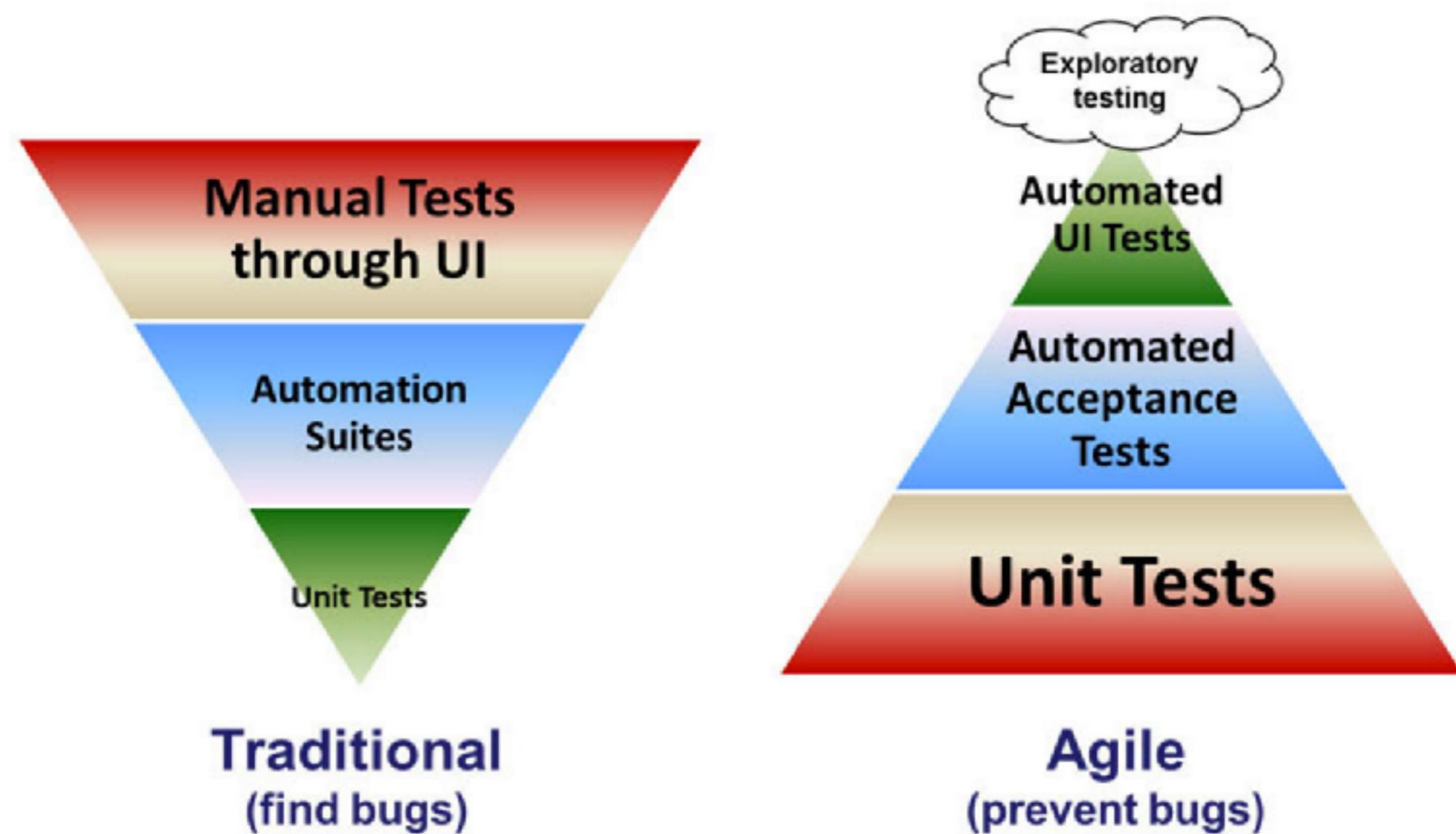
# Mushroom testing



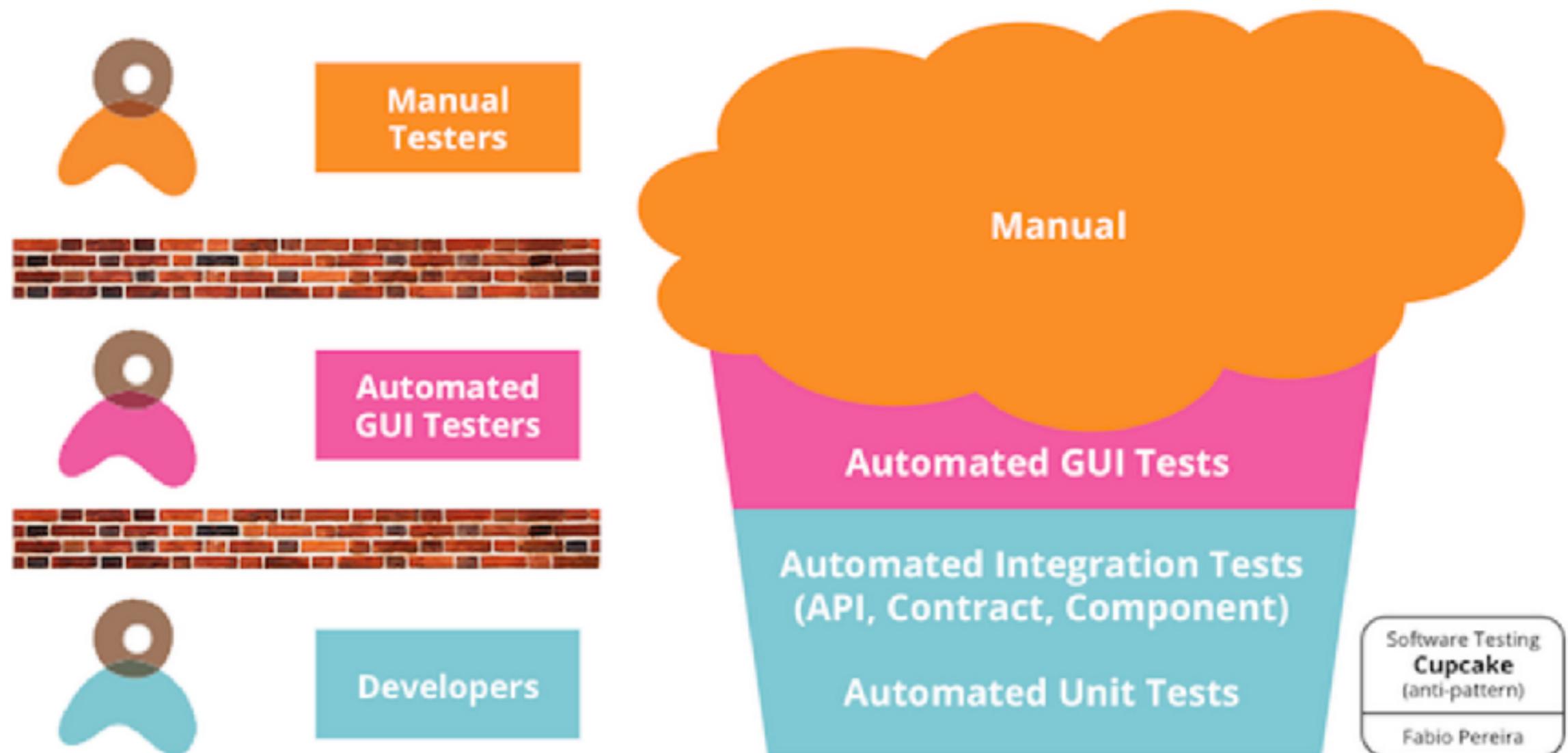
# Pyramid testing



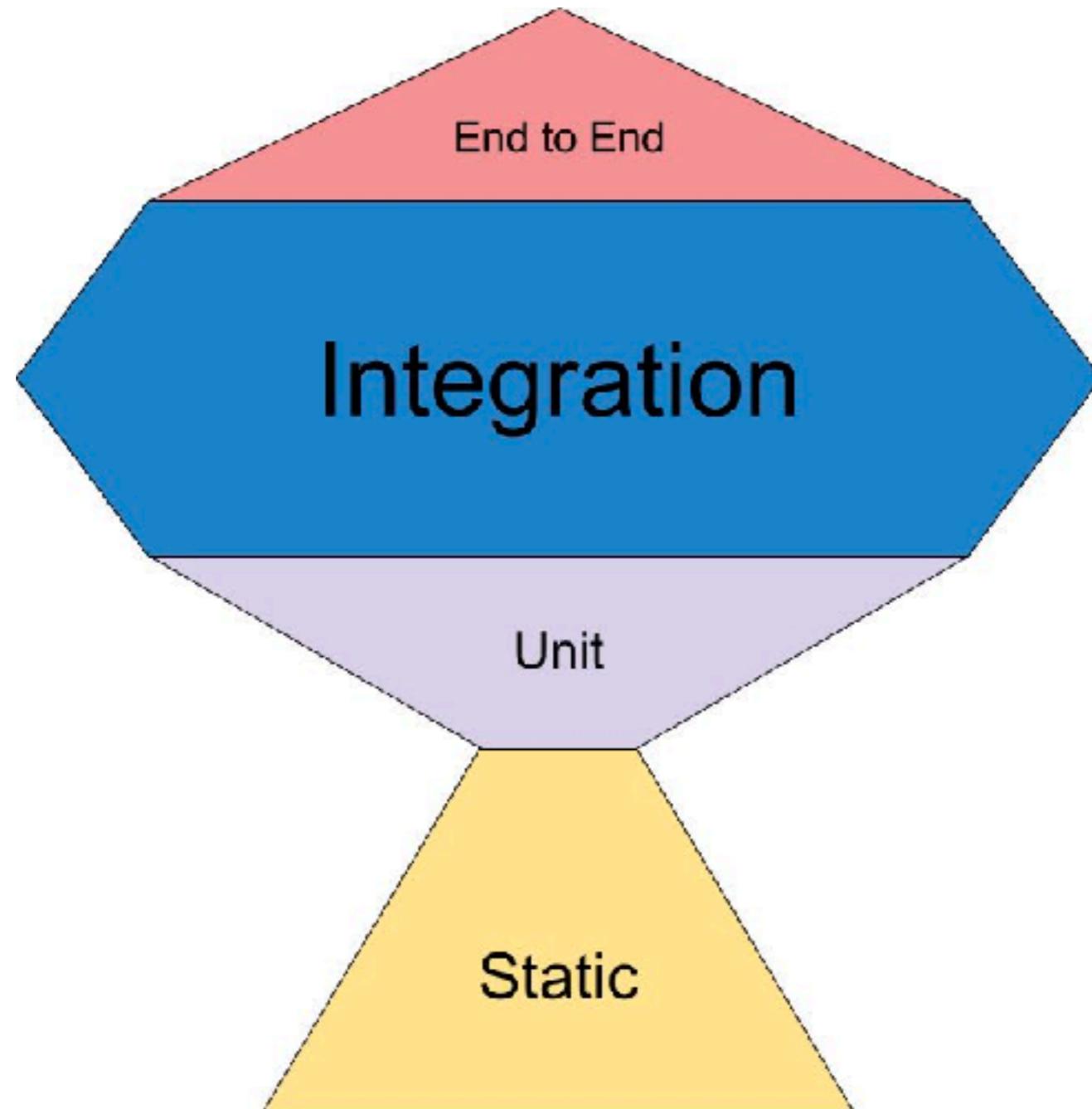
# Prevention over find bugs



# Cup cake testing



# Trophy testing



# Mind-set switch

**Instead of**

We are here to **find bug**

We are here to **ensure requirement are met**

We are here to **break the software**

**Think**

What can I do to help deliver the software  
successfully !!



# Remember !!

Test pyramid is a tool  
To talk about automation tasks  
How to prioritize and help to do automation ?  
Way to make **visible to the whole team**



# Testing strategies



# Test Strategy ?

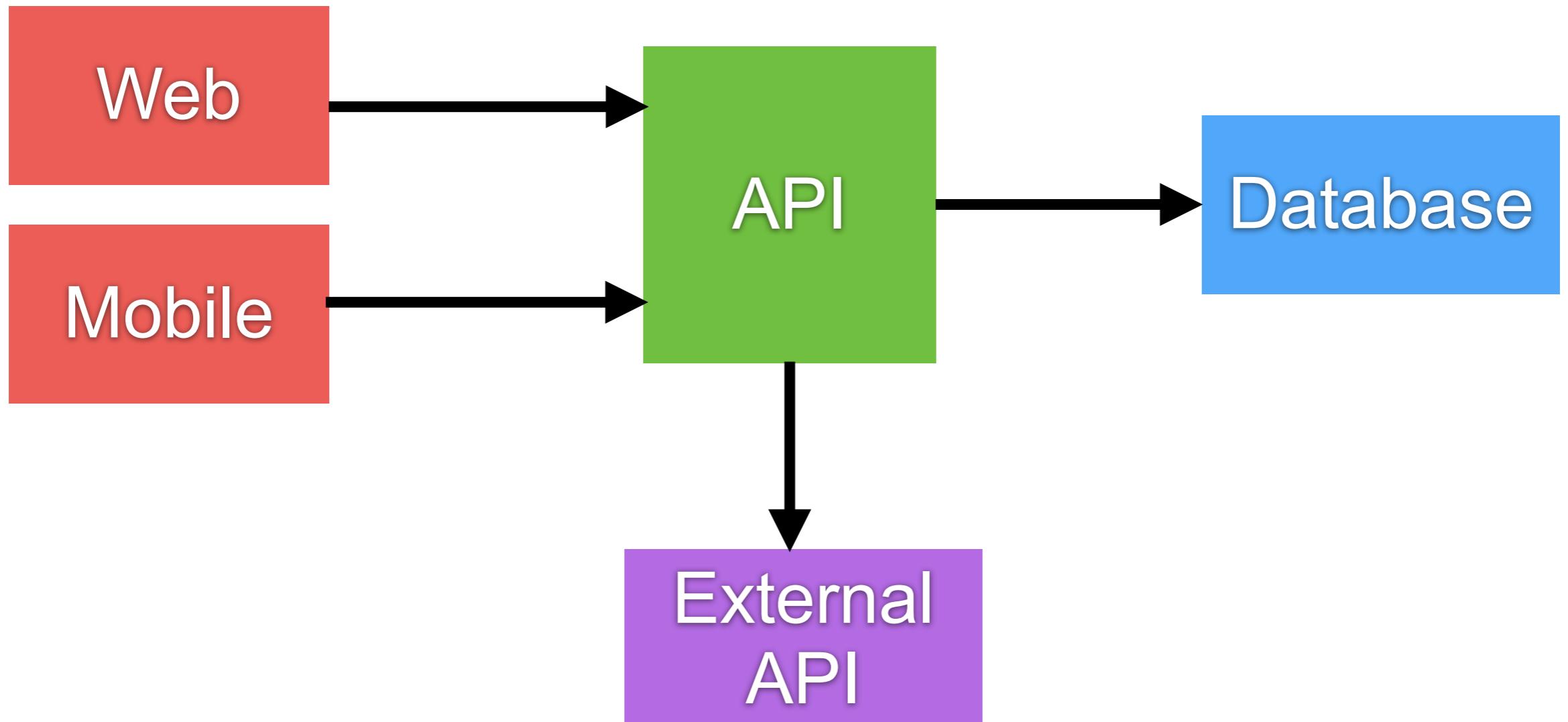
Unit  
Integration  
Service Component  
Contract  
End-to-end



# Start with your architecture



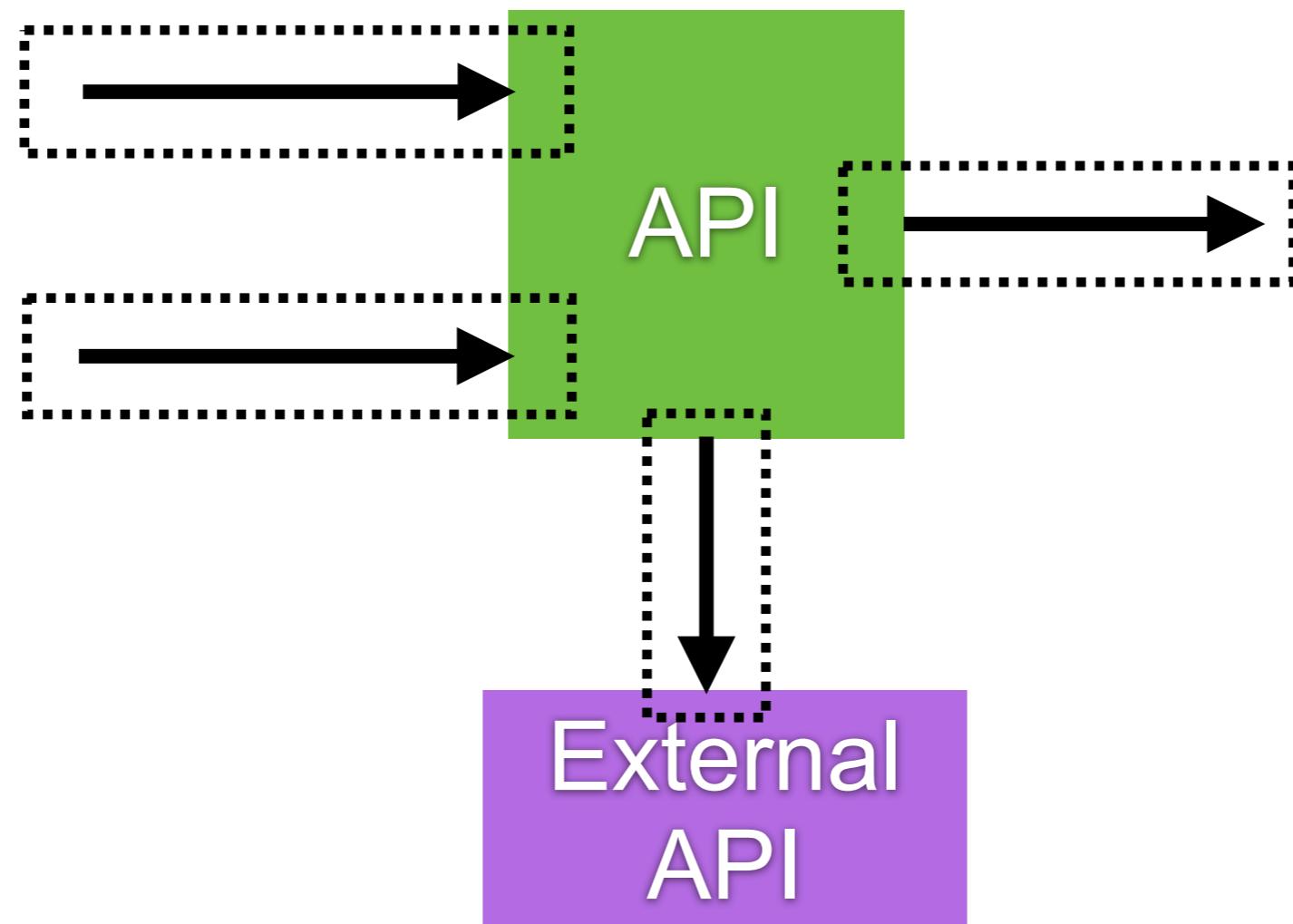
# Simple architecture



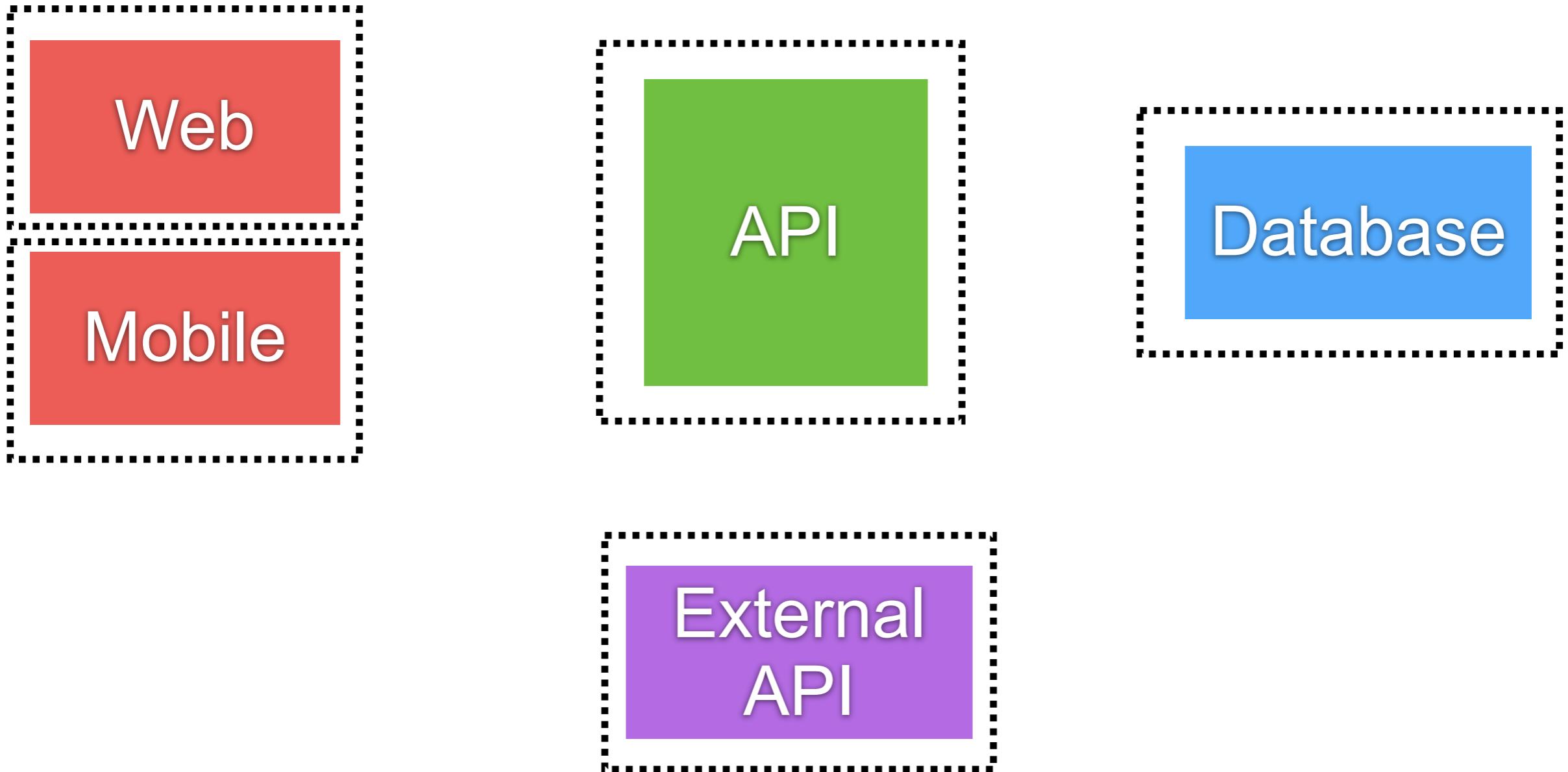
# Testing strategies ?



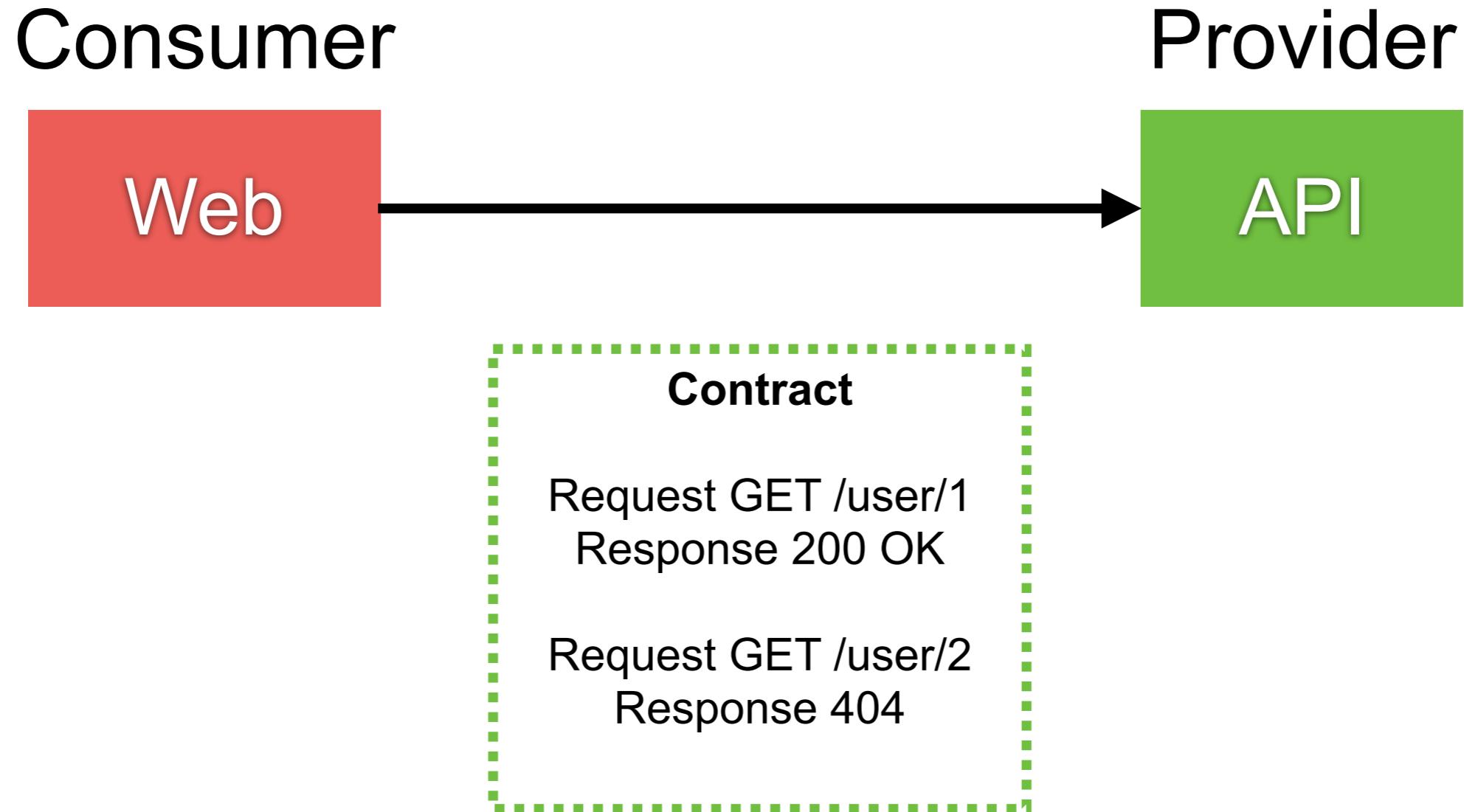
# Integration testing



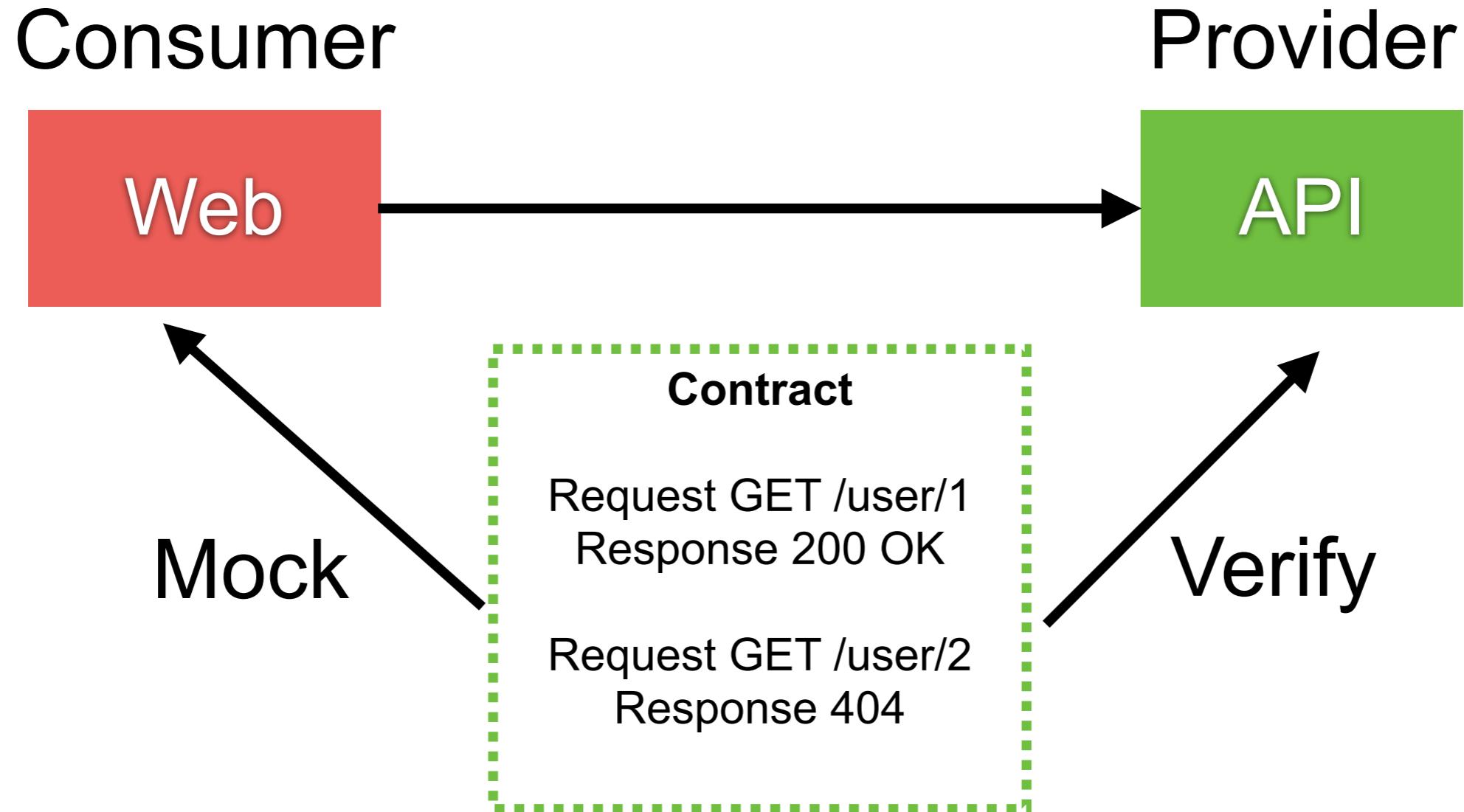
# Service component testing



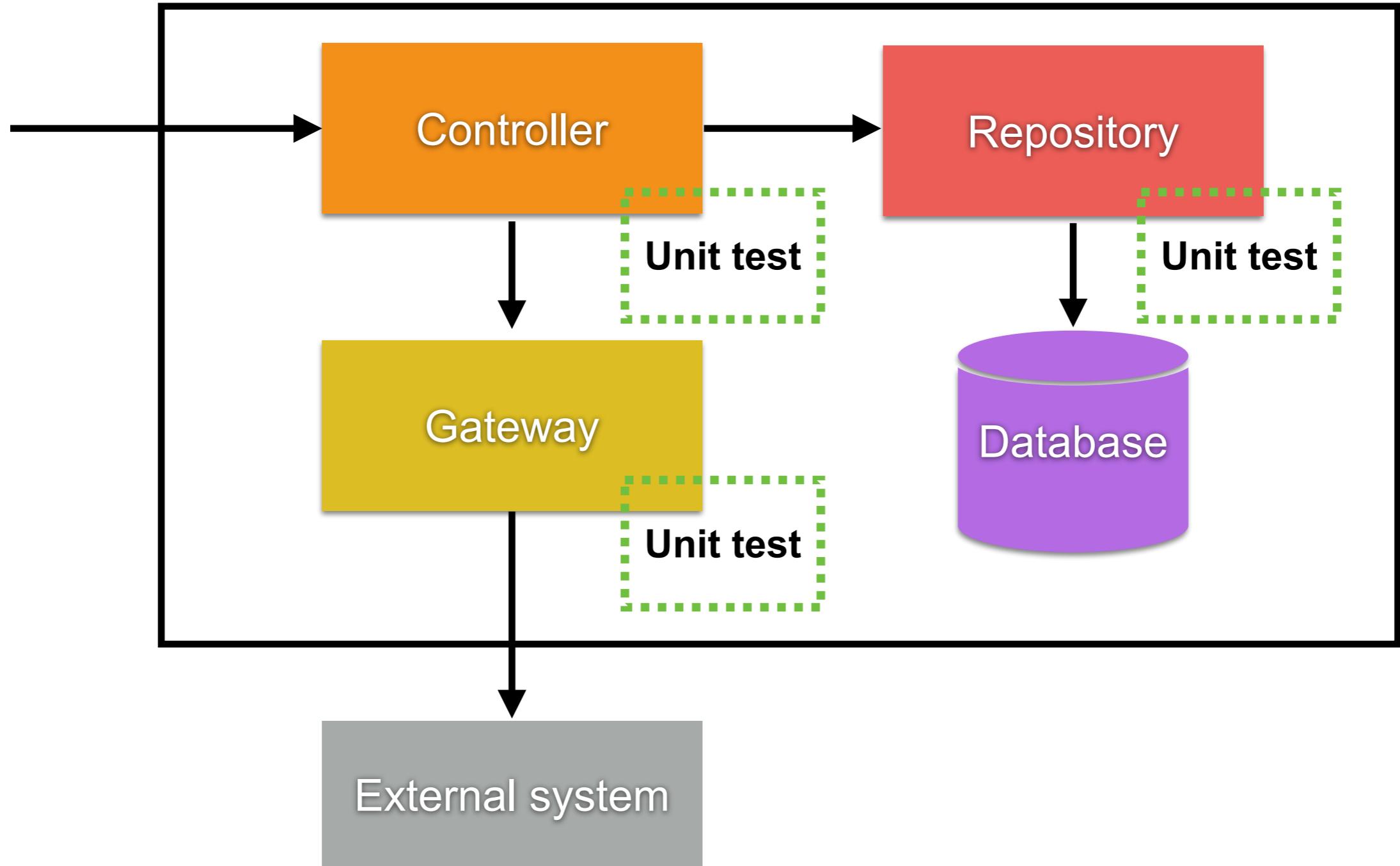
# Contract testing (1)



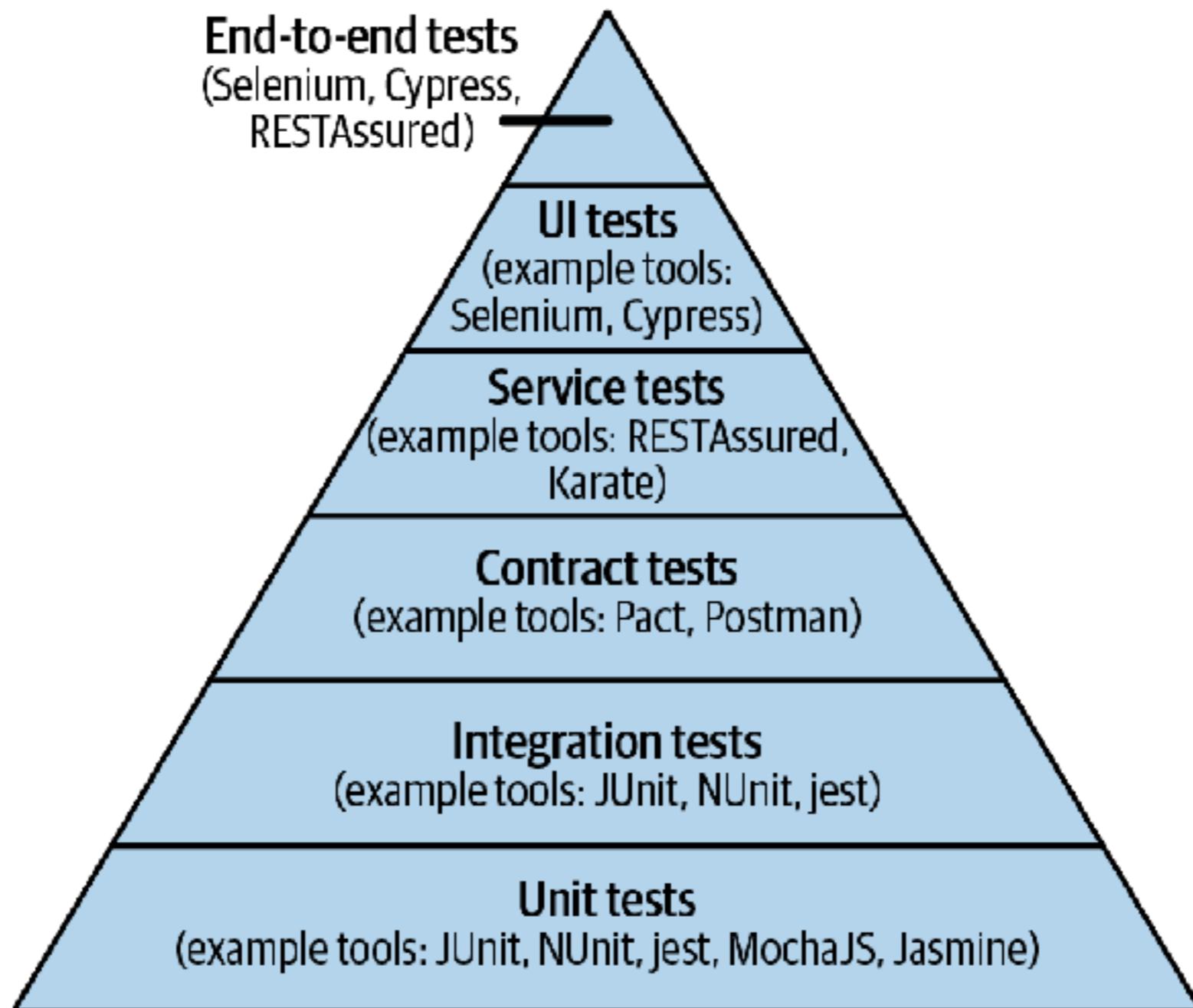
# Contract testing (2)



# Unit testing



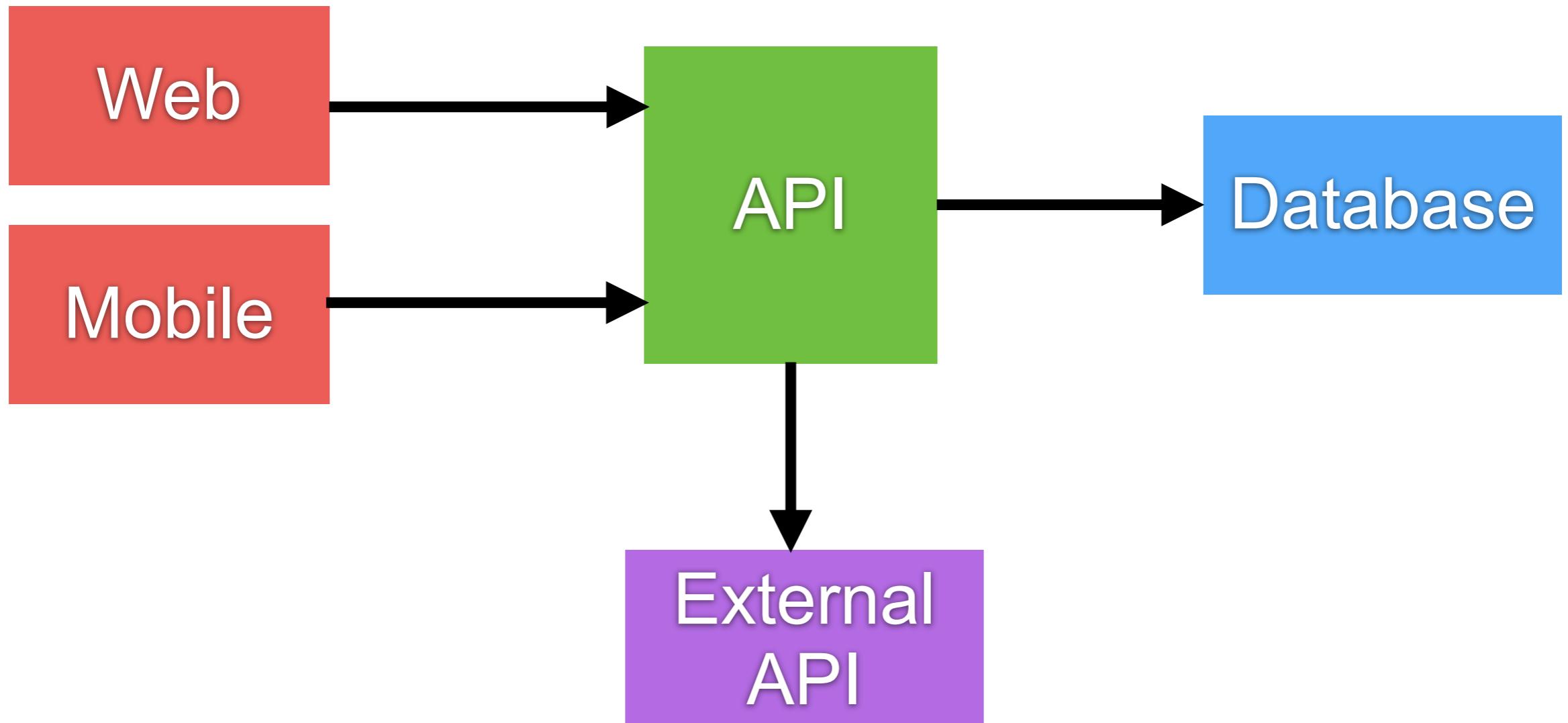
# Number of your test cases ?



# Testing tools and framework ?

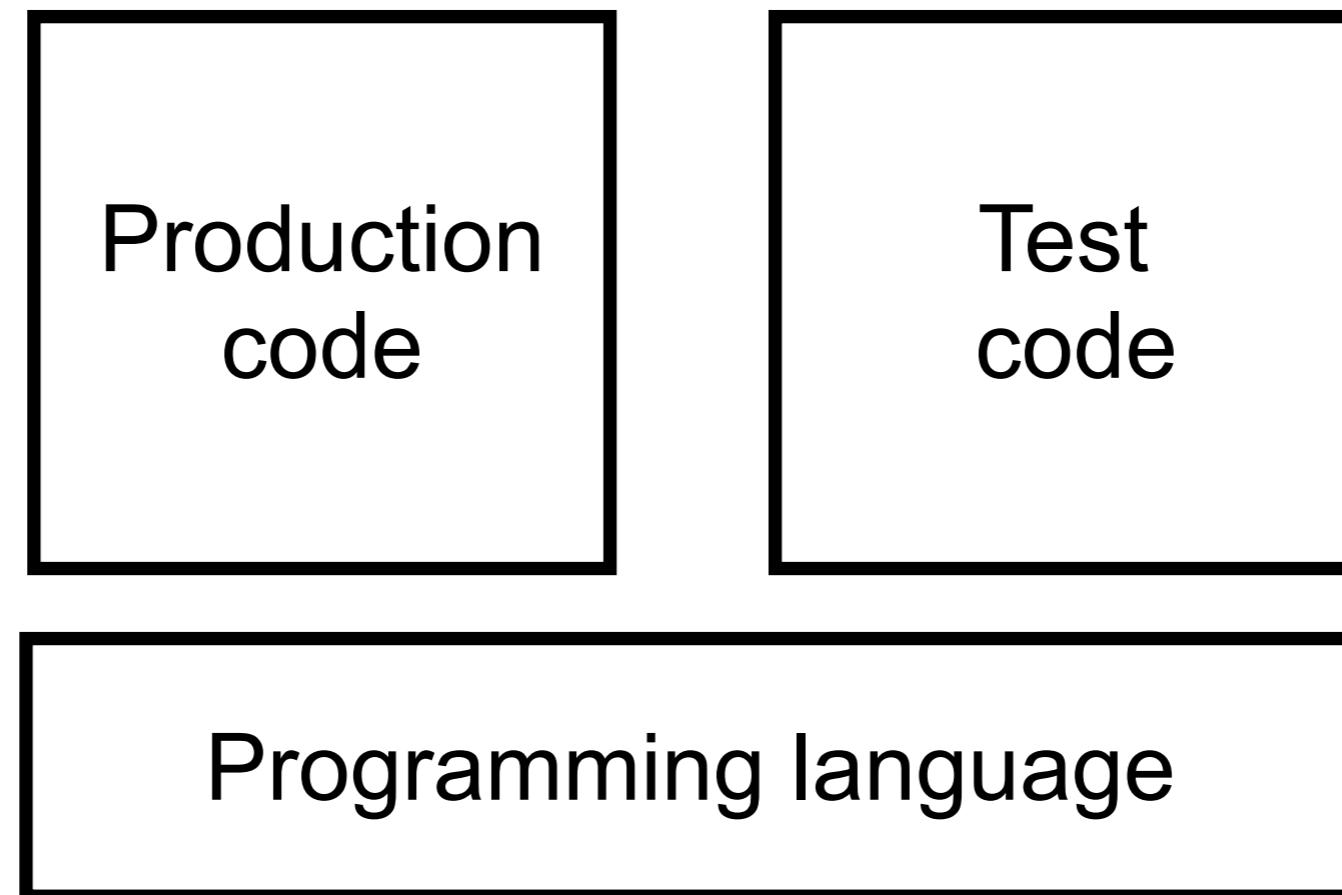


# Testing tools ?



# Testing ?

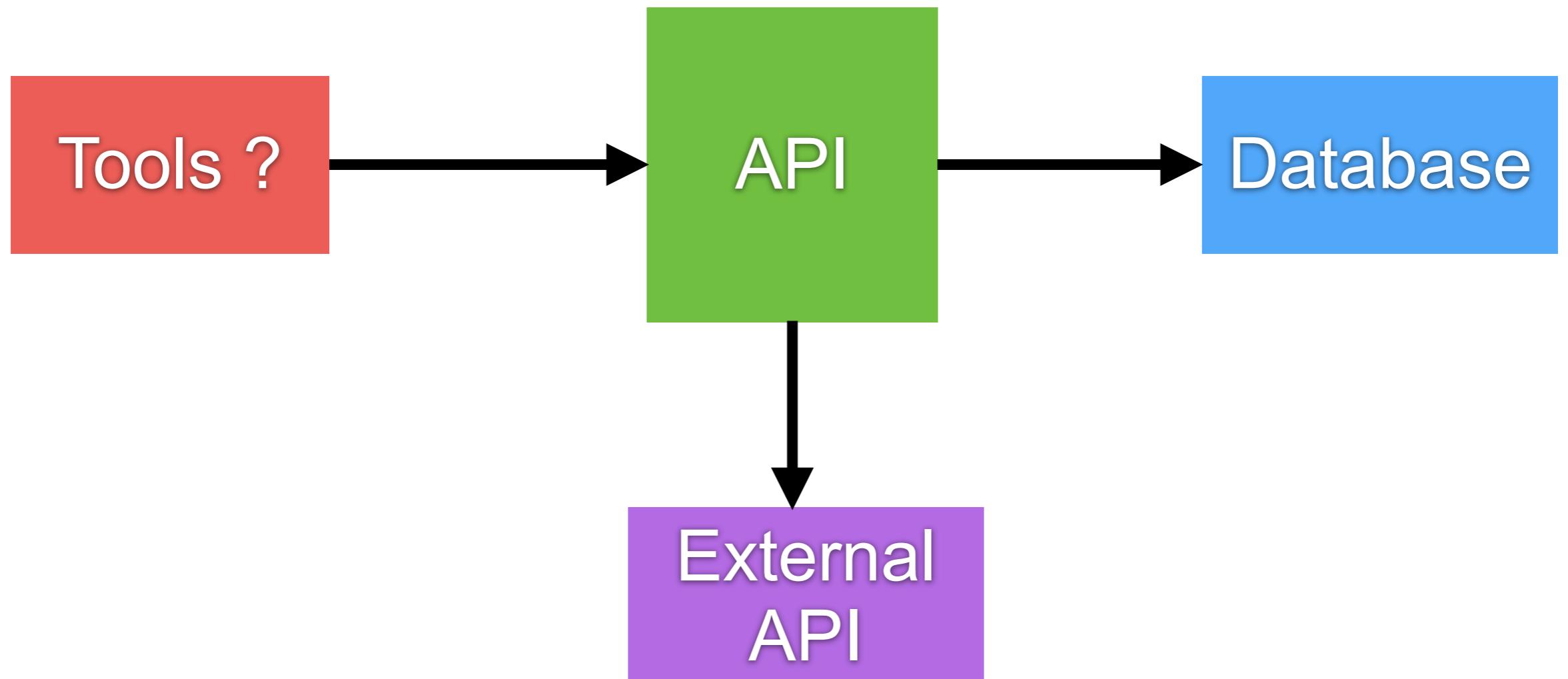
Write test cases with programming language



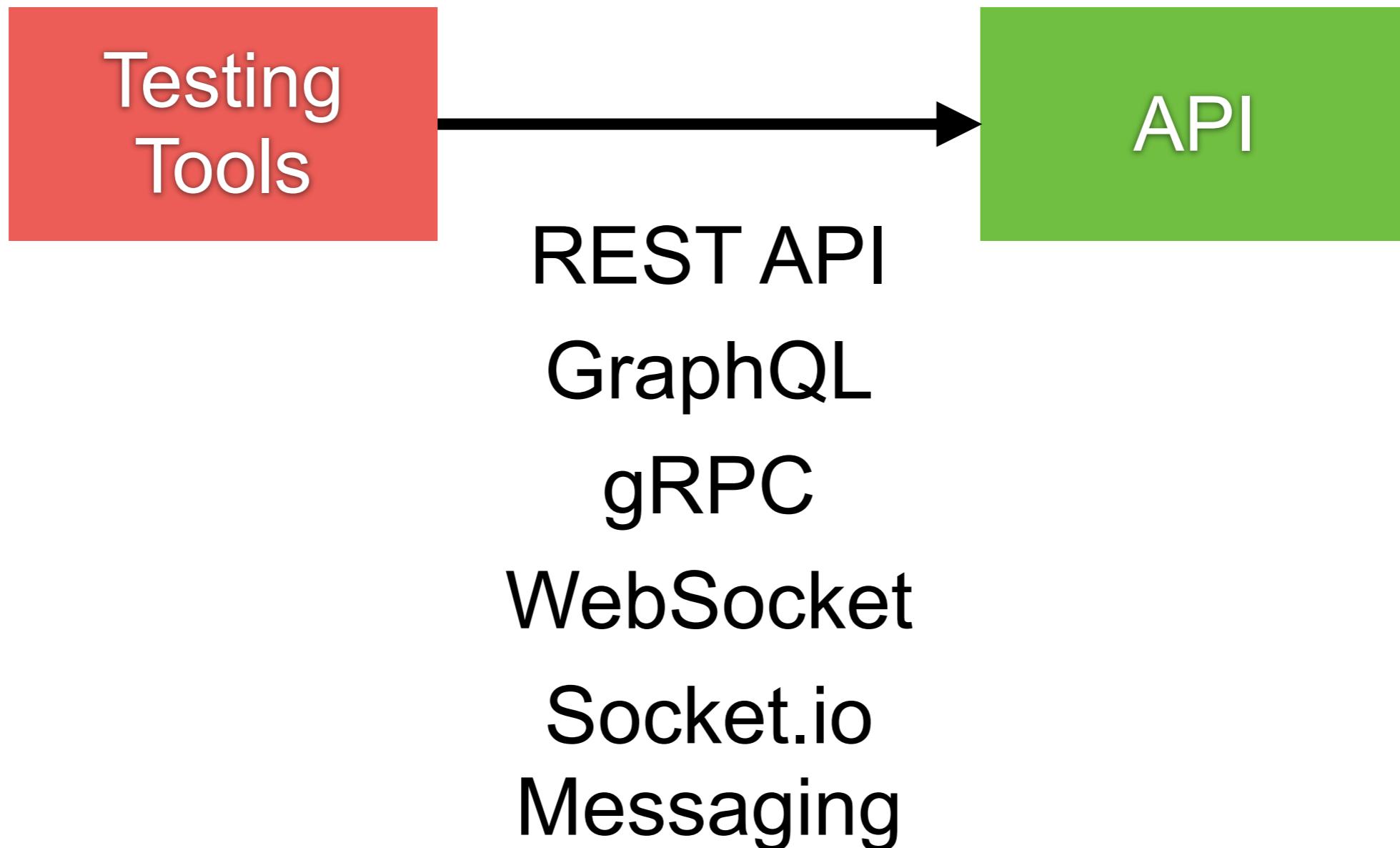
# Testing for API



# API Testing ?



# Testing for API



# Testing for API



POSTMAN



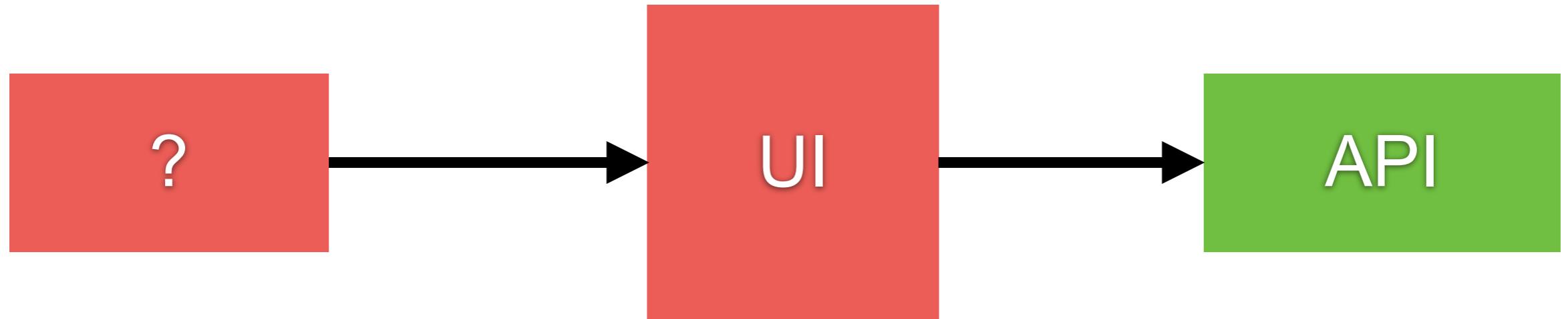
bruno



# Testing for Web UI



# UI Testing ?



# External testing for UI

Write test script and coding



Robot Framework



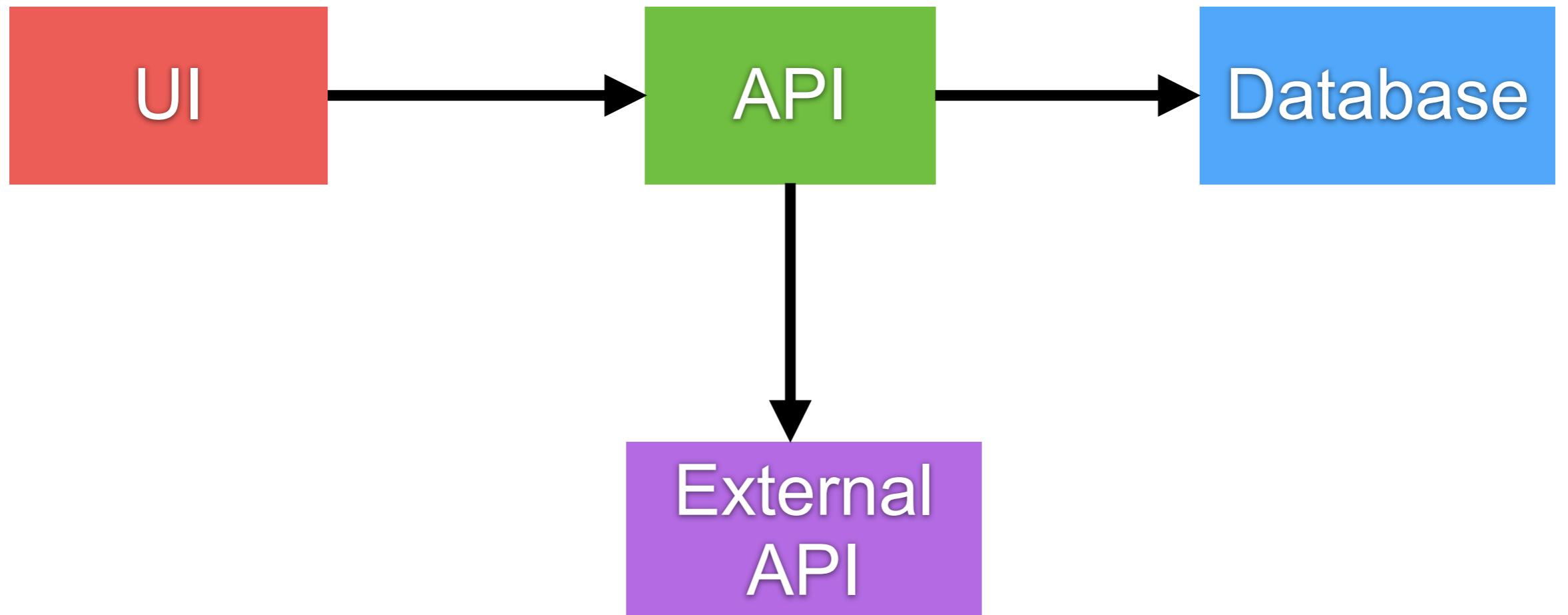
Playwright



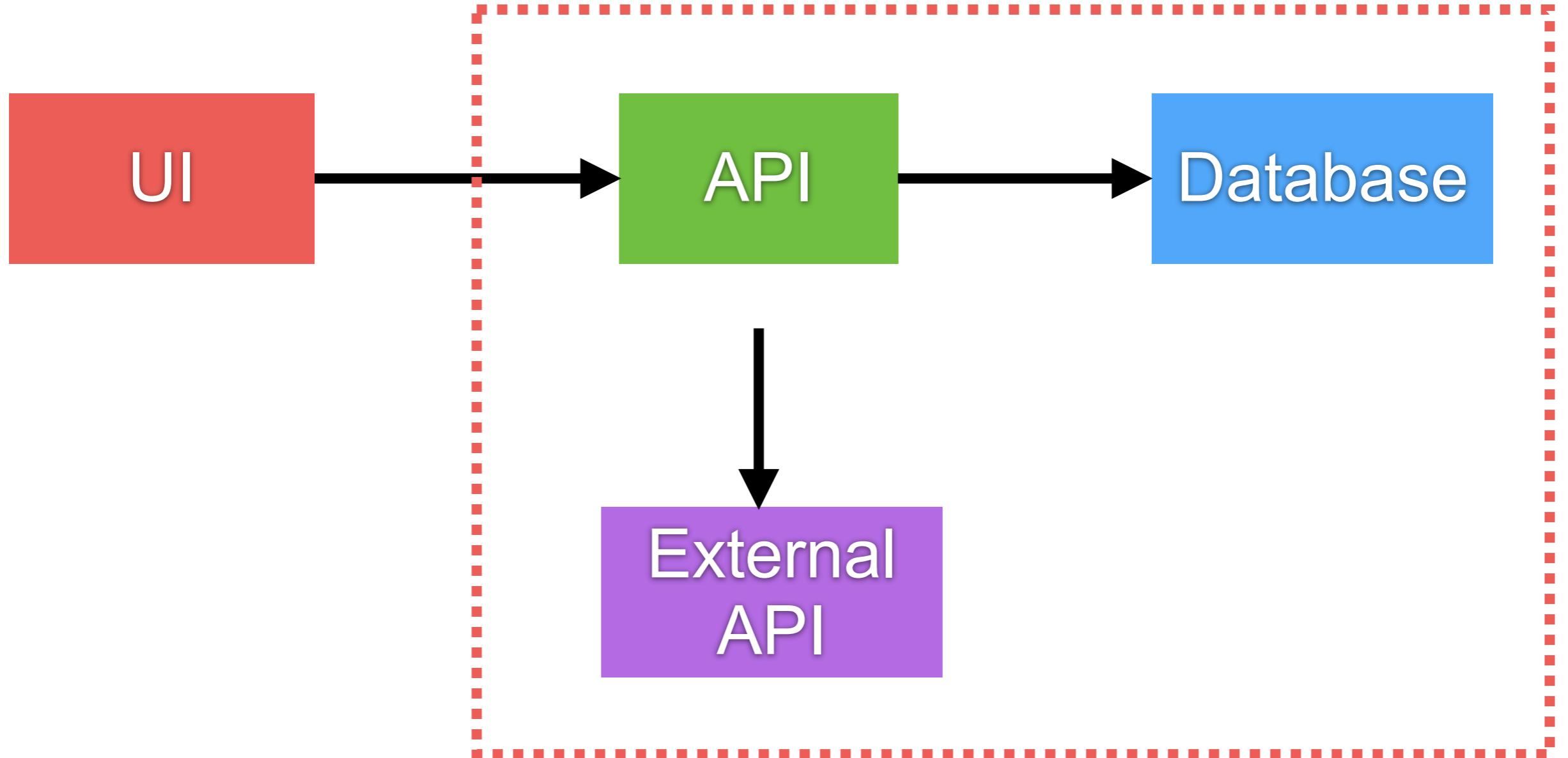
# Manage dependencies



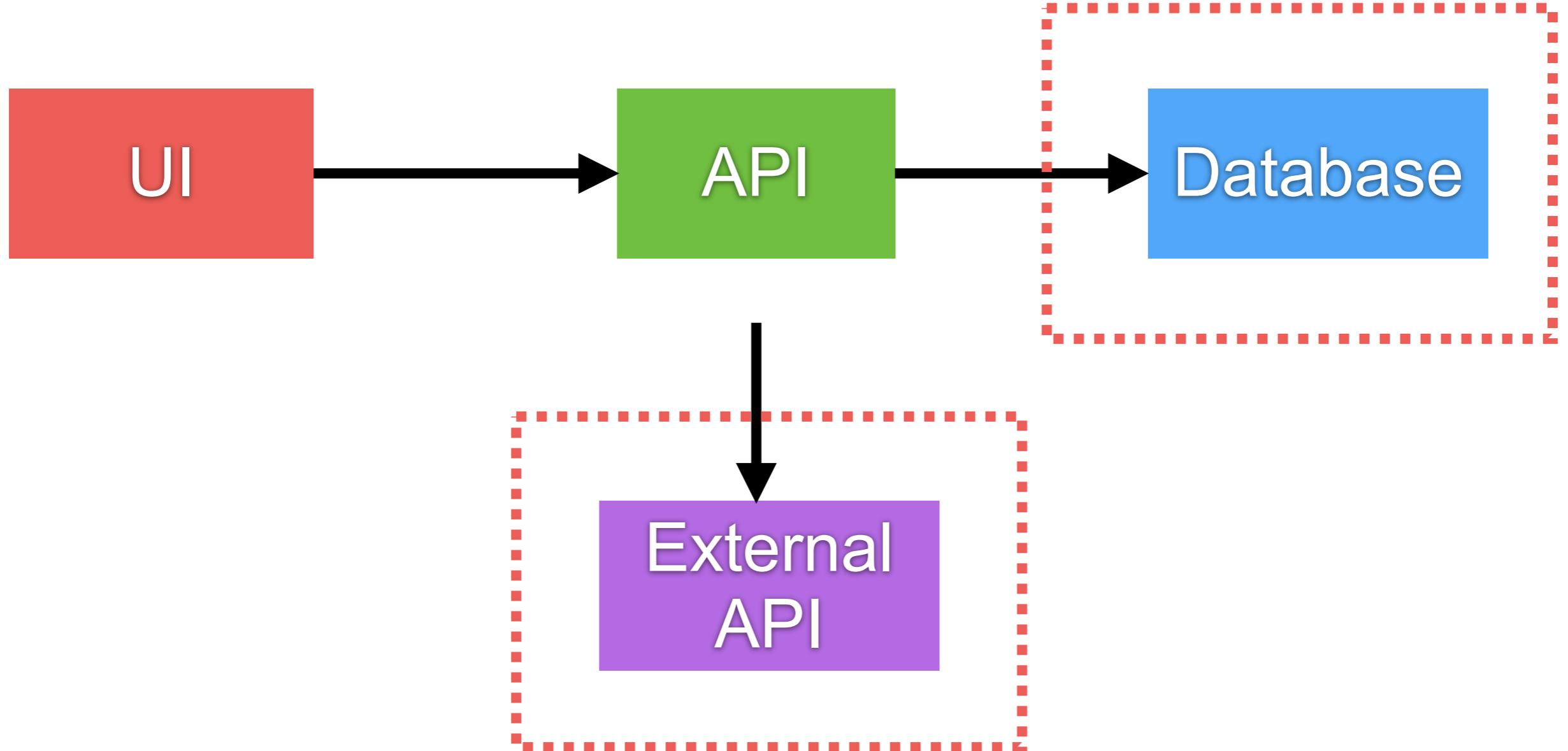
# Manage dependencies ?



# UI Perspective



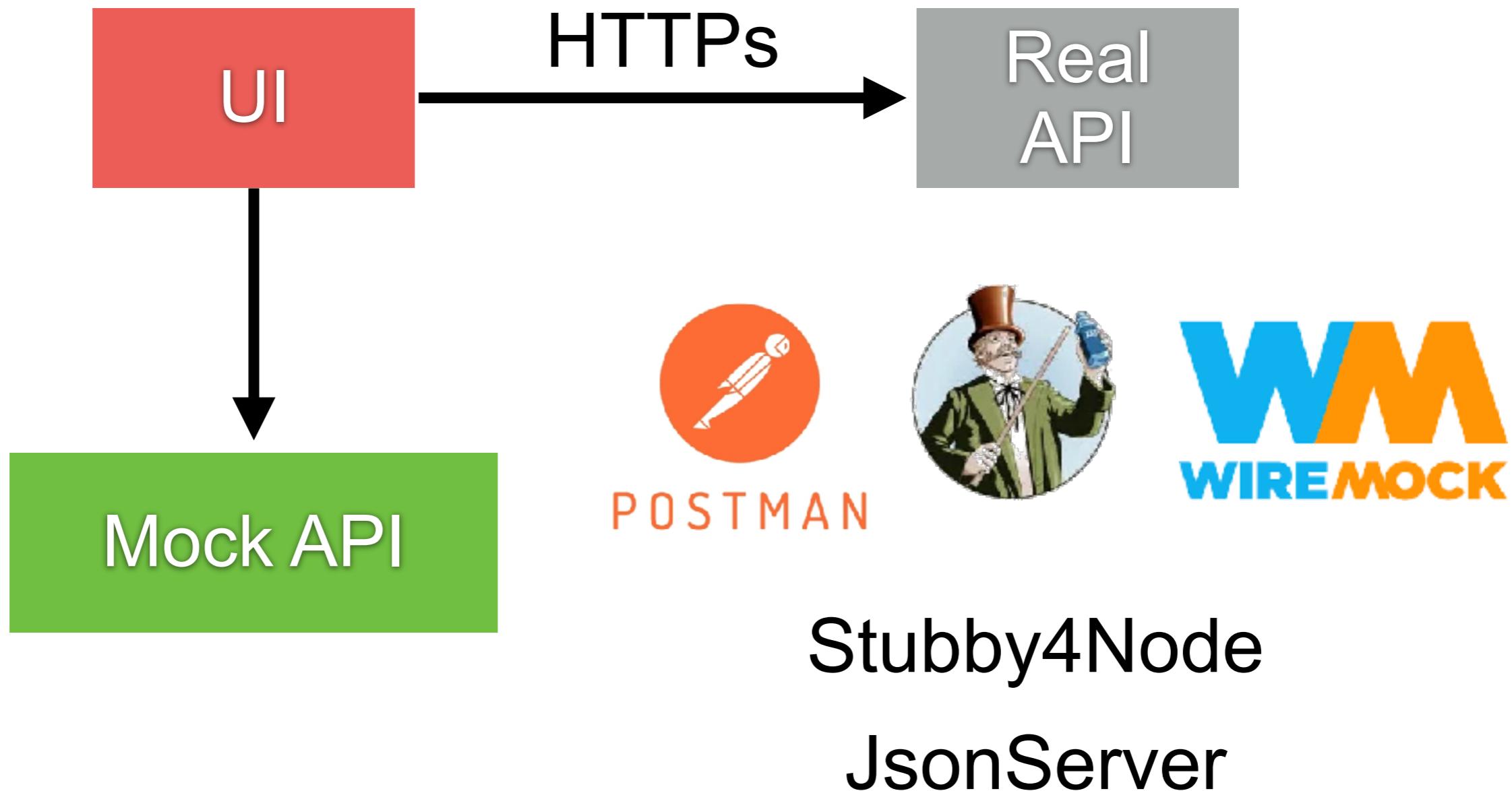
# API Perspective



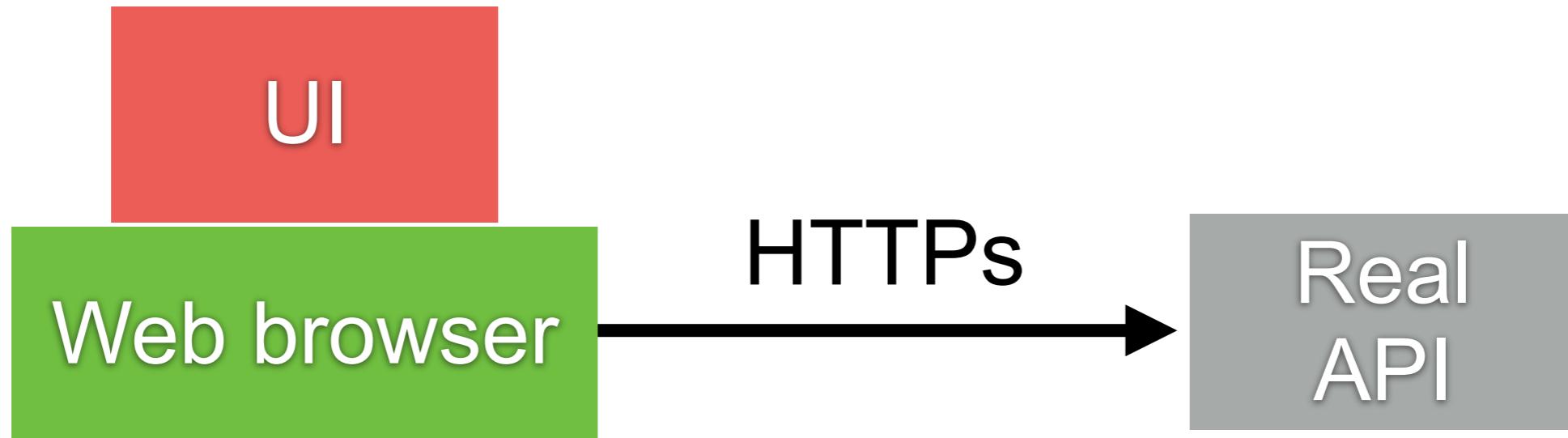
# Manage dependencies of UI



# External Mock API Server



# Intercept in Web browser

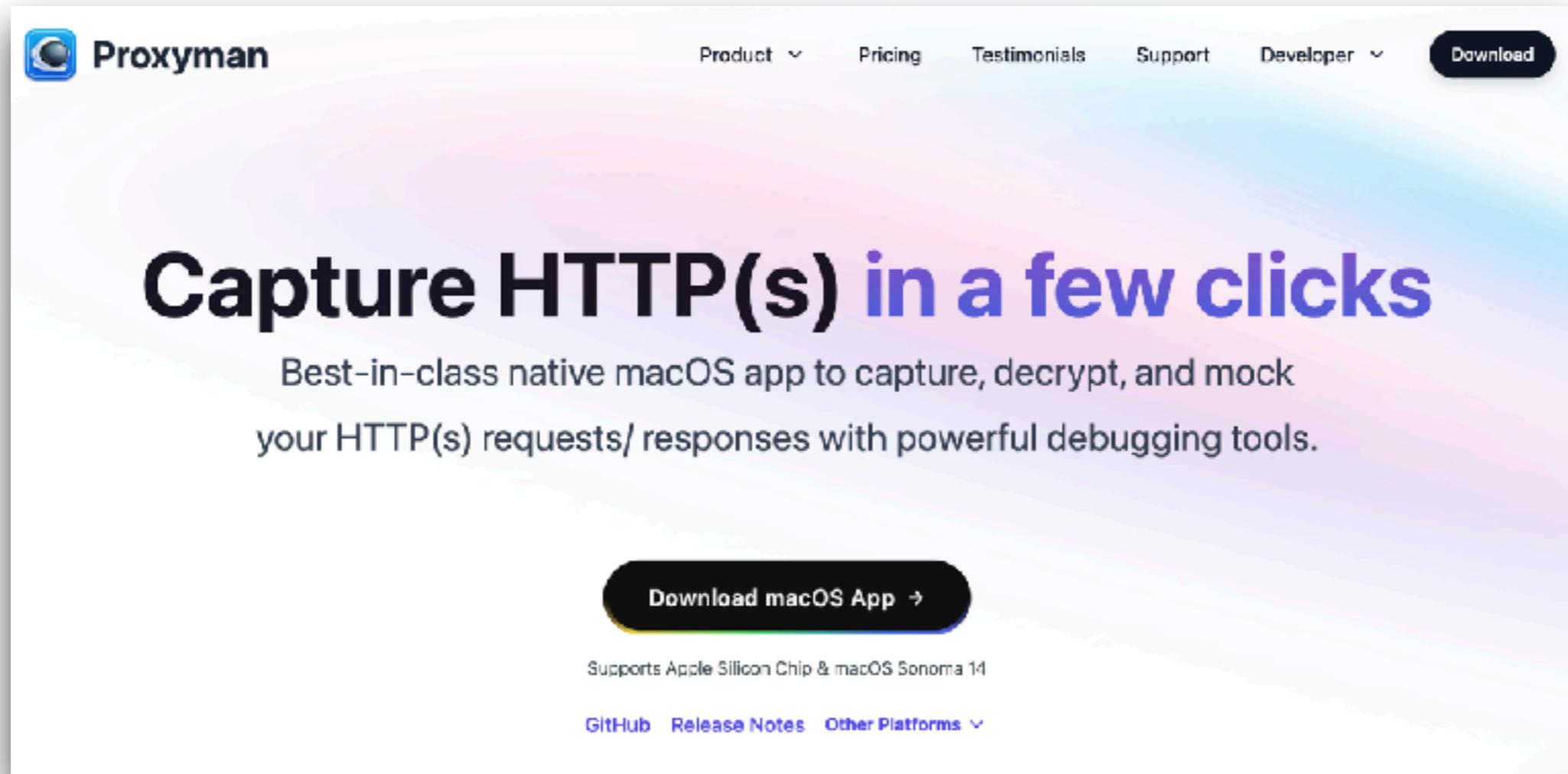


**Playwright**



# Proxy Man

## Capture HTTP request !!

A screenshot of the Proxyman website. The header features the "Proxyman" logo with a blue square icon containing a white circle and a checkmark. The main navigation menu includes "Product", "Pricing", "Testimonials", "Support", "Developer", and a "Download" button. Below the menu, a large call-to-action text reads "Capture HTTP(s) in a few clicks". A subtext below it states: "Best-in-class native macOS app to capture, decrypt, and mock your HTTP(s) requests/ responses with powerful debugging tools." At the bottom of the main content area is a "Download macOS App" button with a yellow gradient effect. Smaller text at the bottom indicates support for "Apple Silicon Chip & macOS Sonoma 14" and links to "GitHub", "Release Notes", and "Other Platforms".

Proxyman

Product Pricing Testimonials Support Developer Download

## Capture HTTP(s) in a few clicks

Best-in-class native macOS app to capture, decrypt, and mock your HTTP(s) requests/ responses with powerful debugging tools.

Download macOS App →

Supports Apple Silicon Chip & macOS Sonoma 14

[GitHub](#) [Release Notes](#) [Other Platforms](#)

<https://proxym.io/>

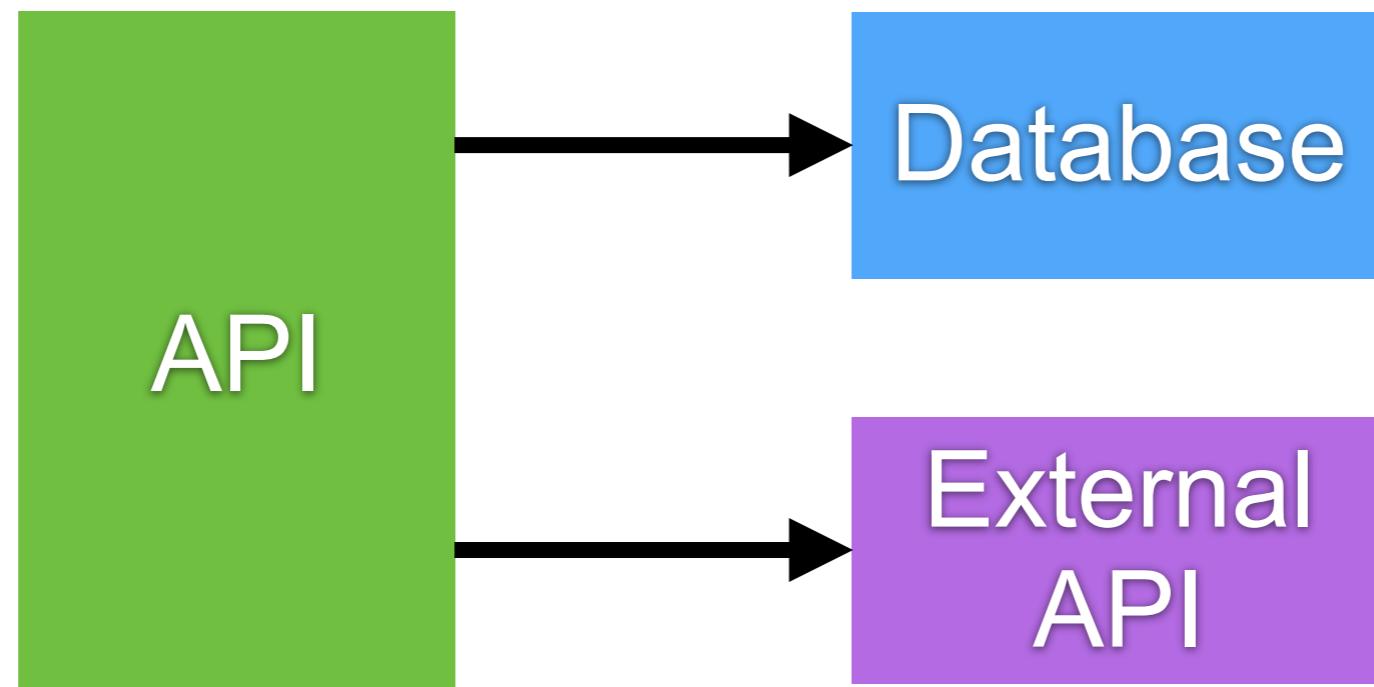


# Manage dependencies of API



# Manage dependencies of API

Database  
External API



# Database

In-memory database  
Container-based  
Real database

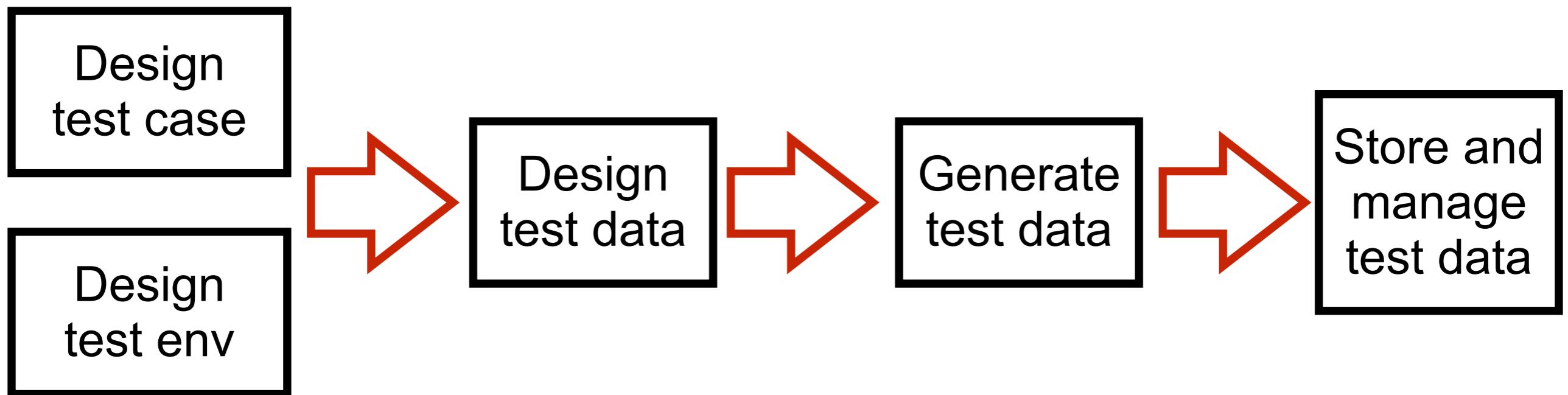
Tables

Data for test

Database migration tools



# Design test data ?



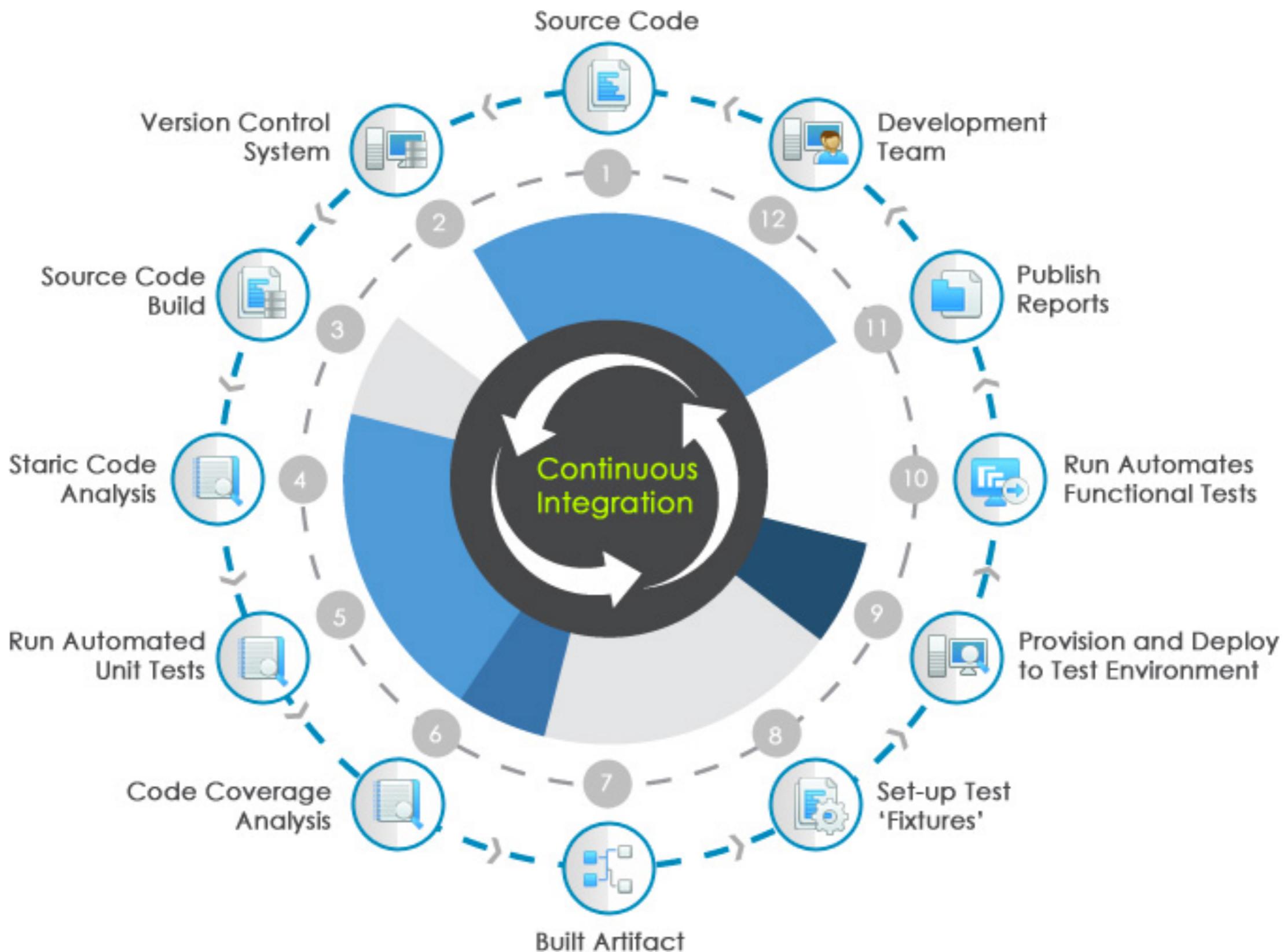
# Delivery pipeline/process ?

Code

Testing

Deploy







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

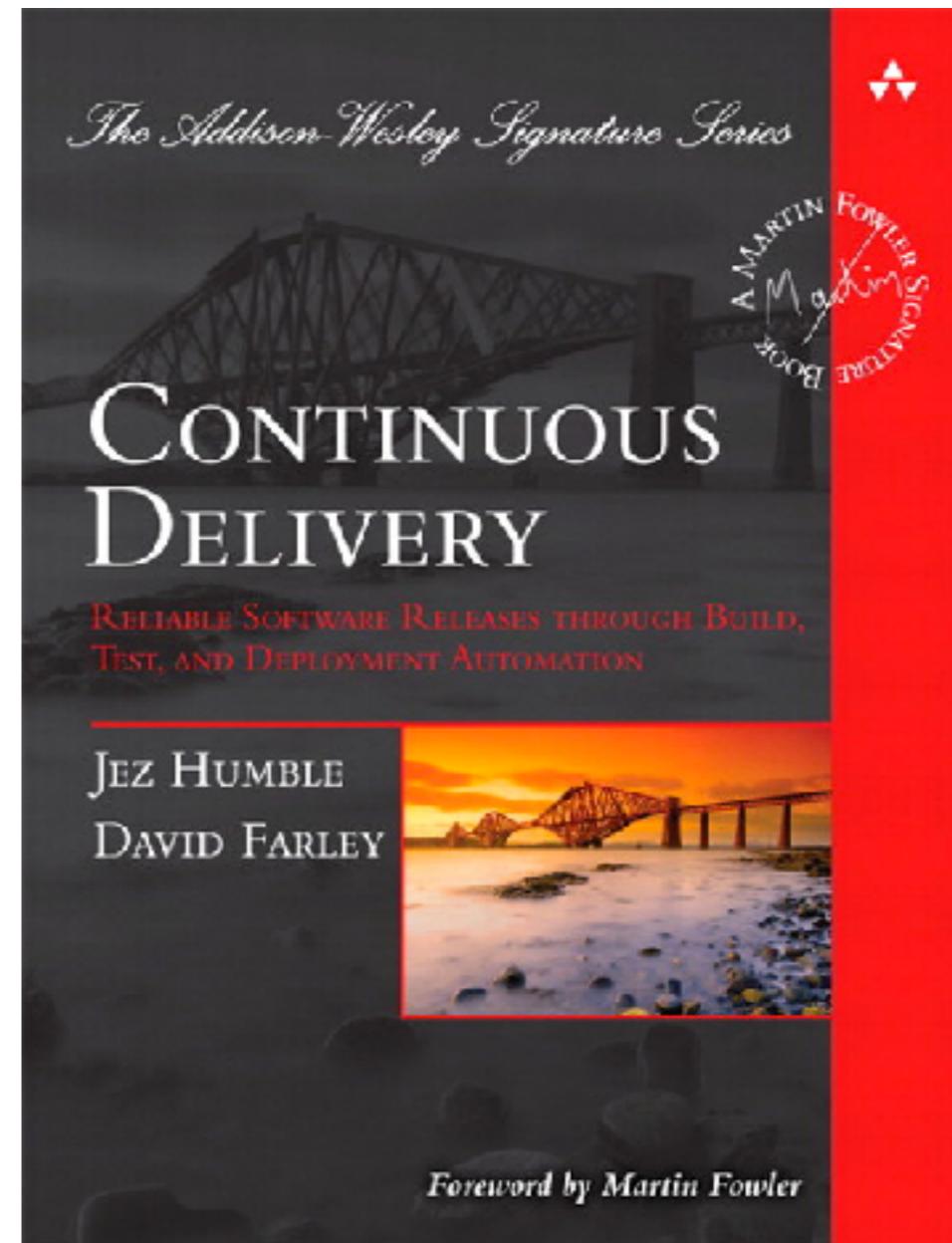
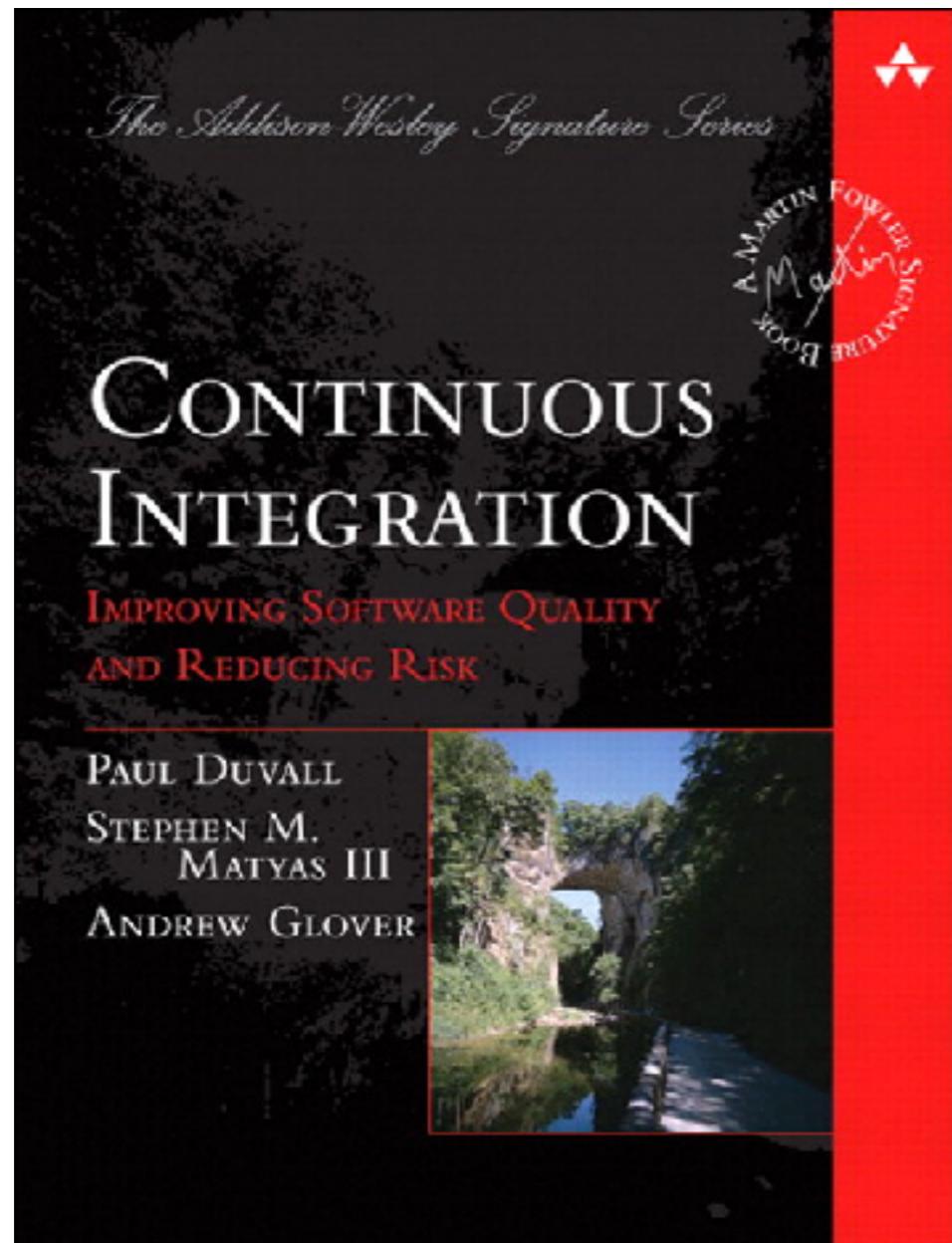
Strive for **fast feedback**



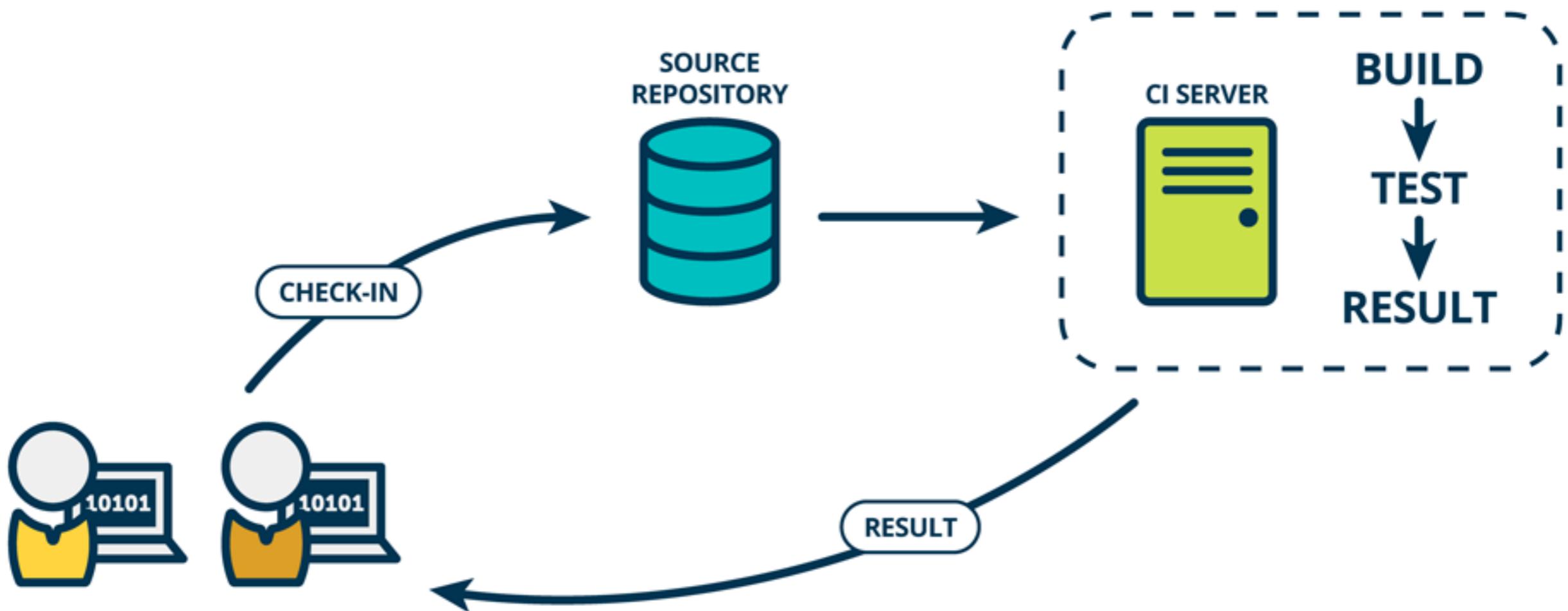
# **Practices of Continuous Integration**



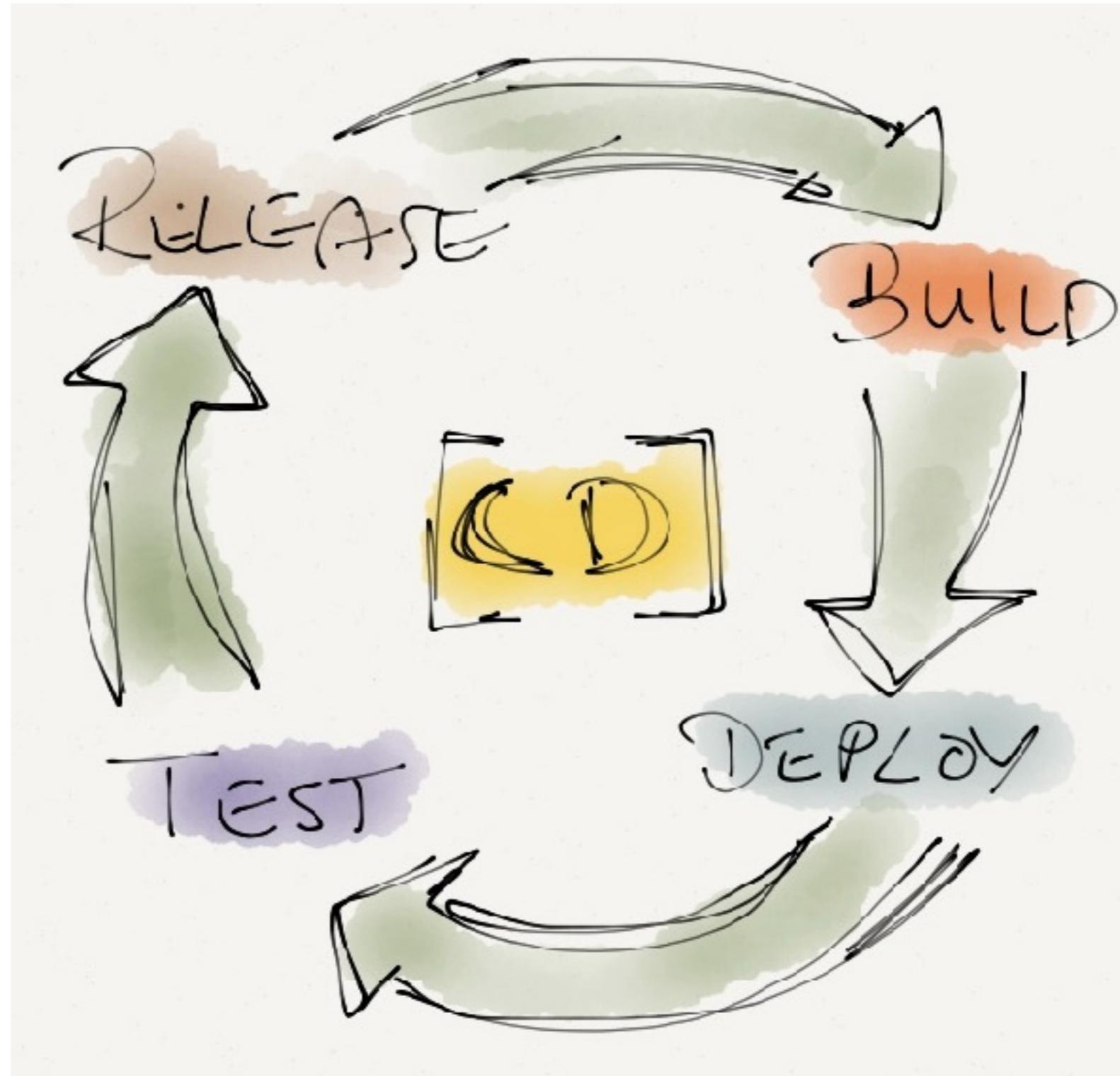
# Improve quality and reduce risk



# Continuous Integration



# CD ?



# CD ?

## CONTINUOUS DELIVERY



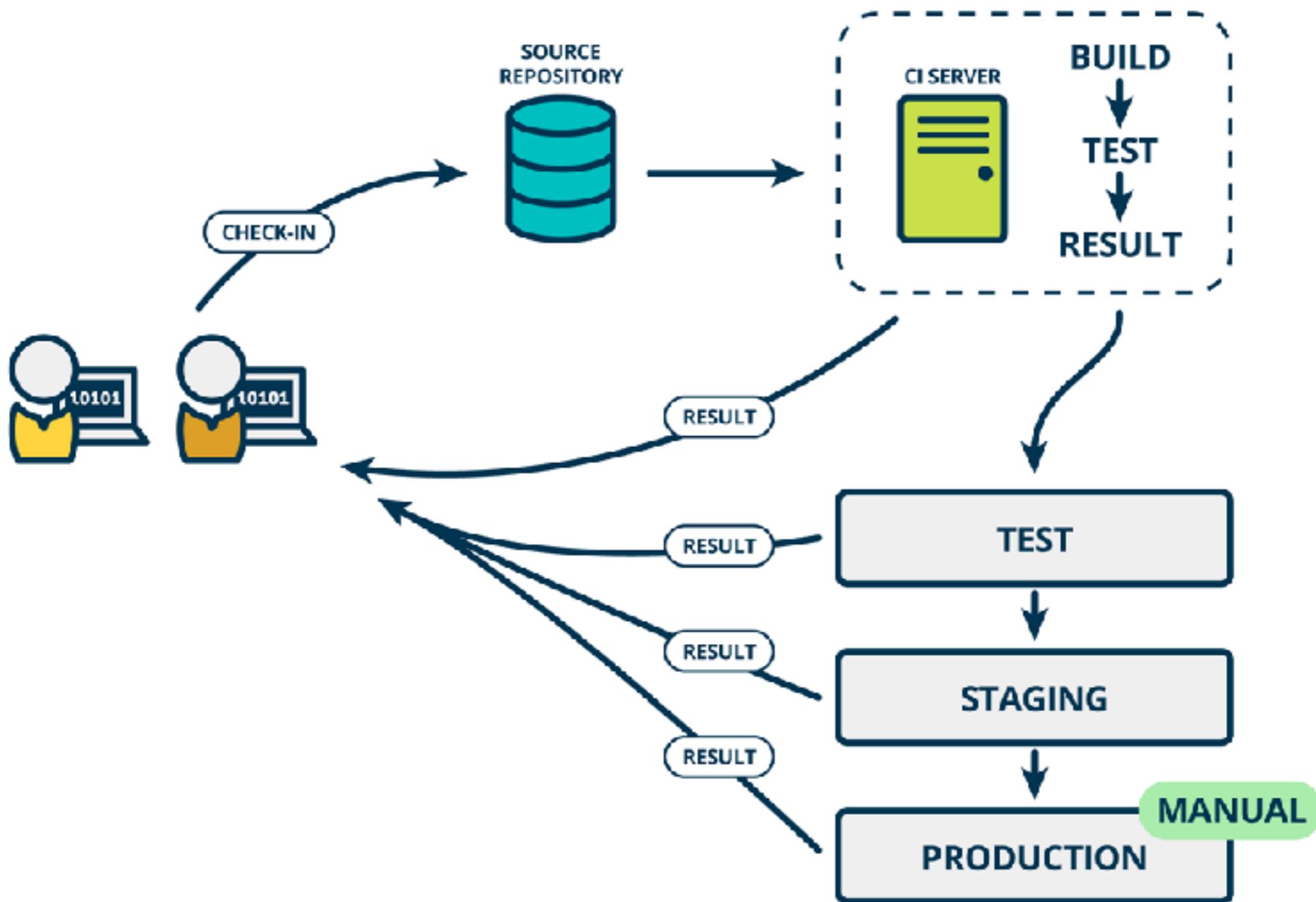
## CONTINUOUS DEPLOYMENT



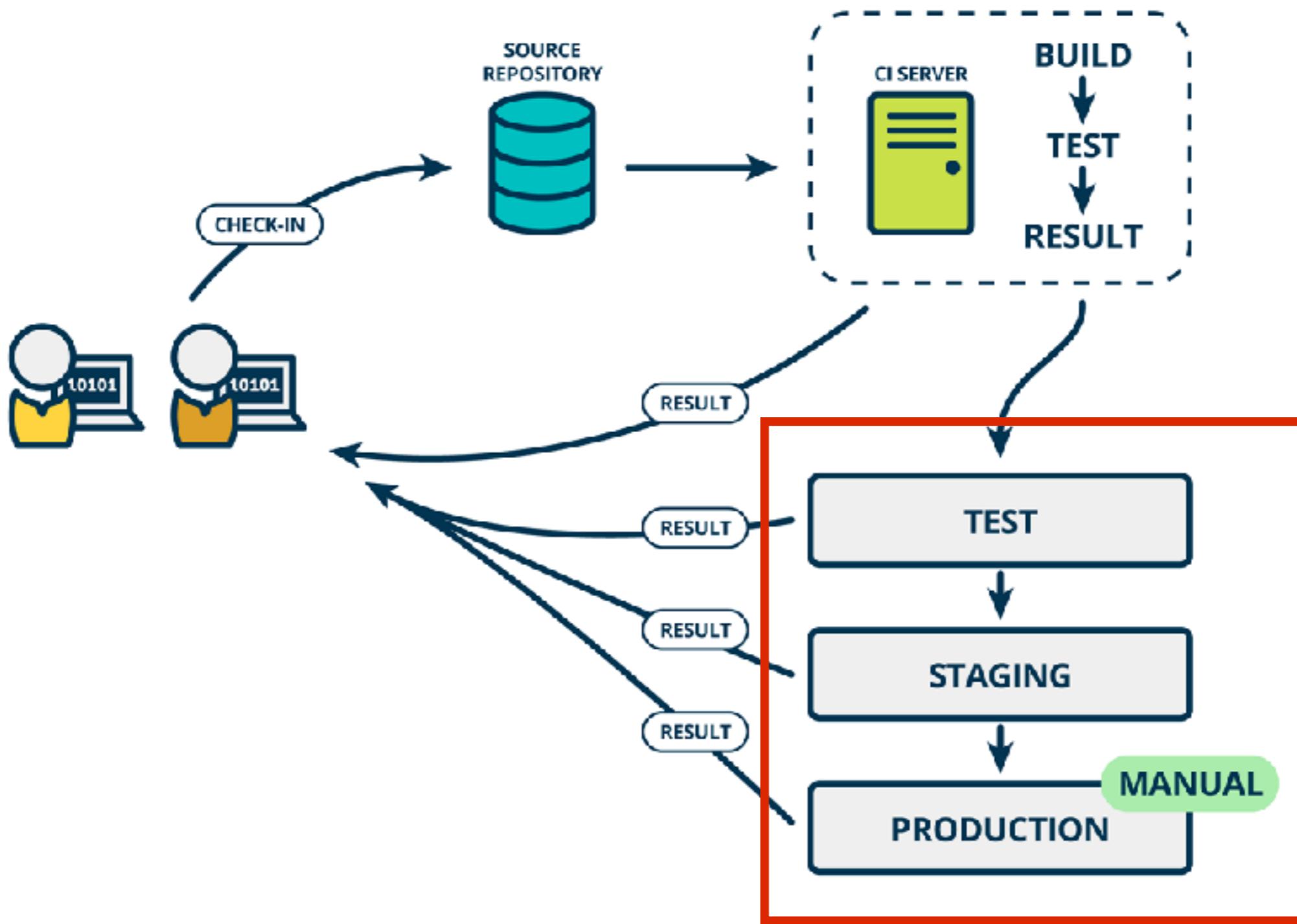
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



# Continuous Delivery



# Rise of DevOps



# **Continuous Integration**

**is a Software development practices**



# Practice 1

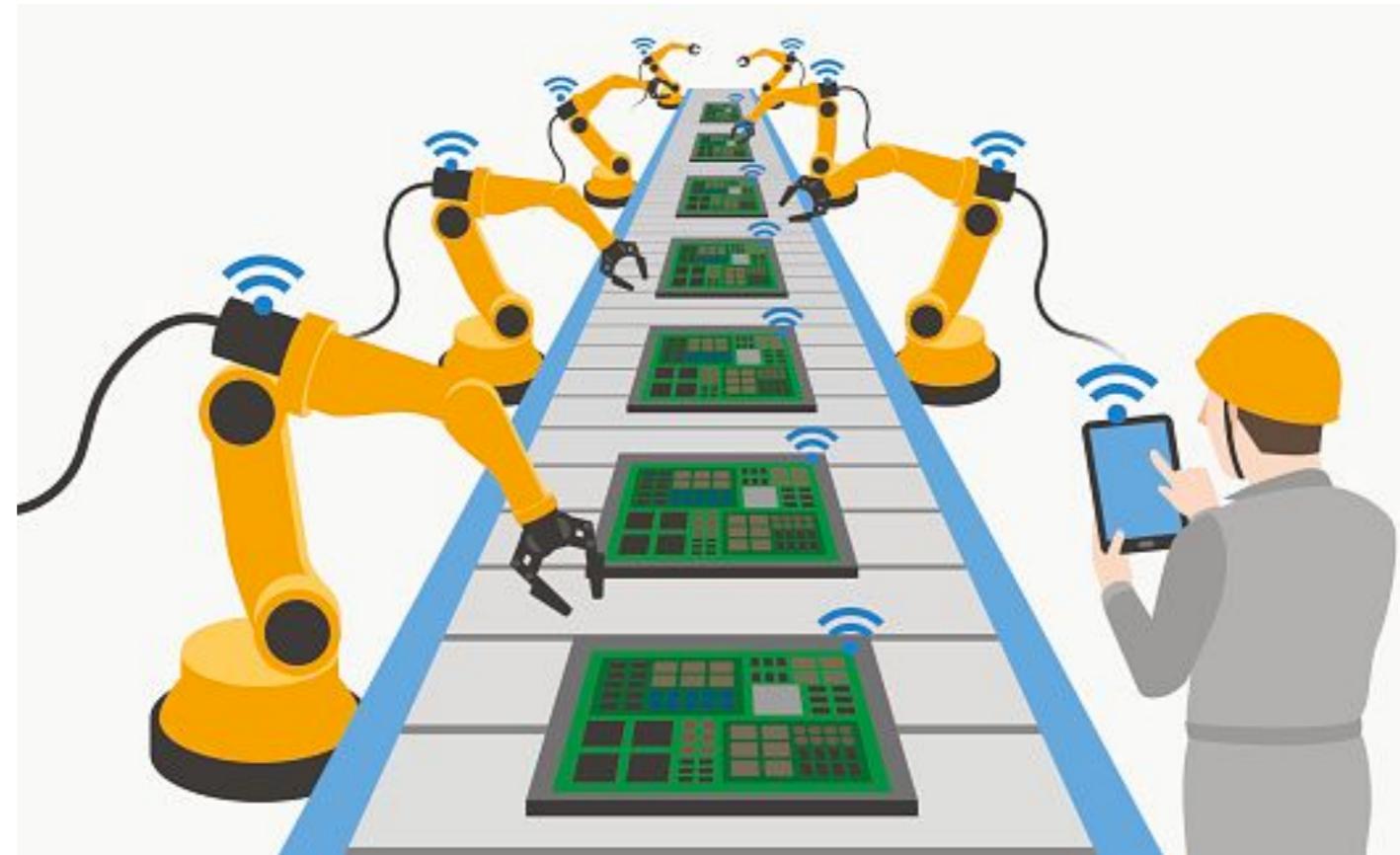
Maintain a single source repository

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



# Practice 2

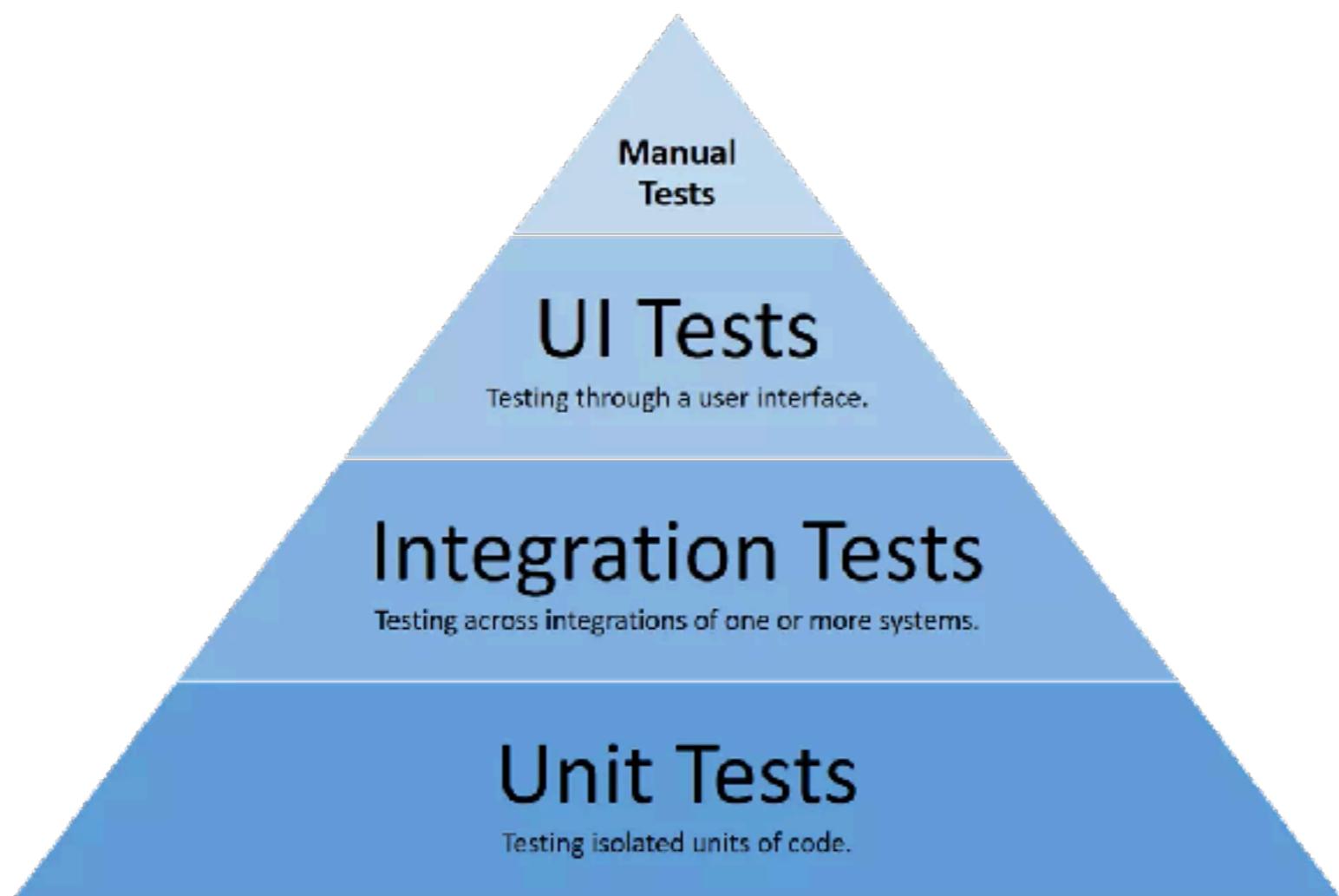
Automated the build  
Automated environment for builds



# Practice 3

Make your build **self-testing**

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

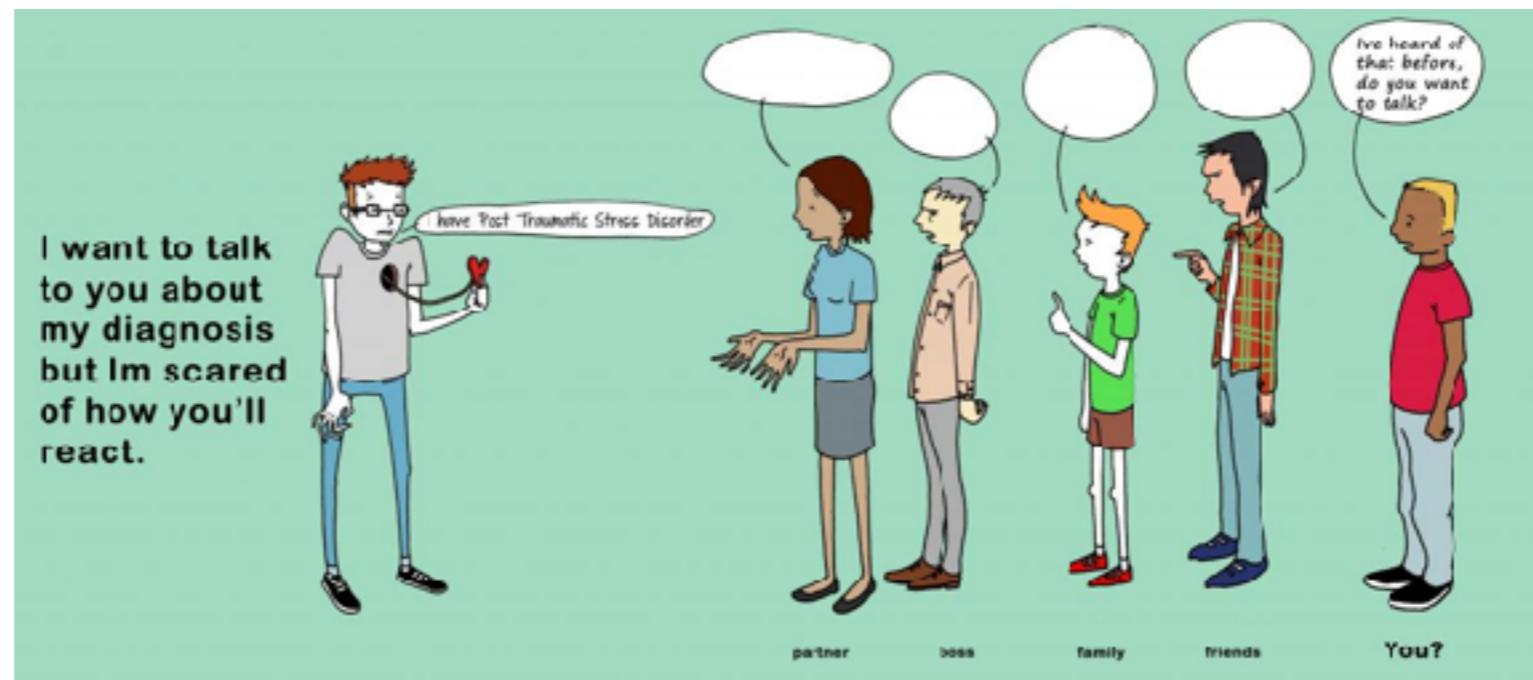


# Practice 4

**Everyone commits to the mainline everyday**

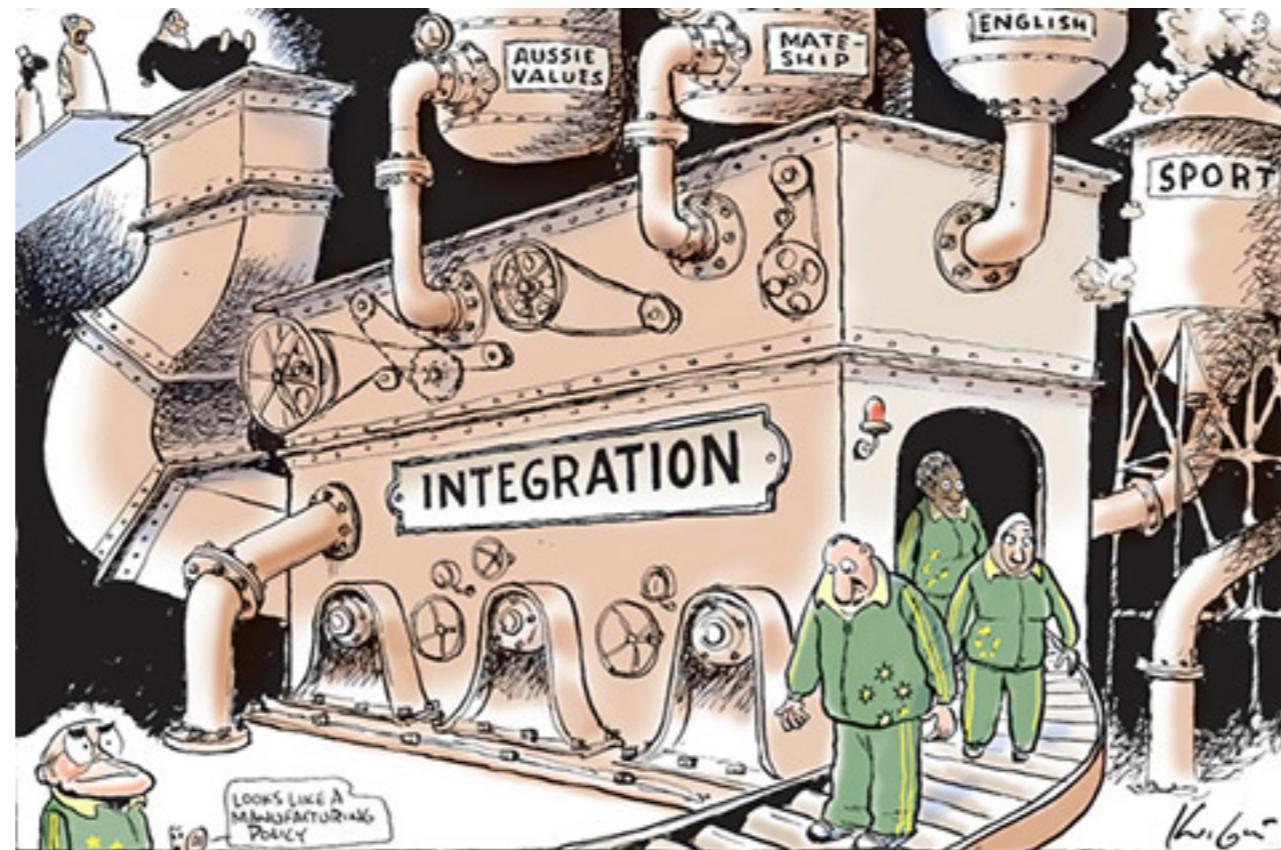
**Integration is about communication**

**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

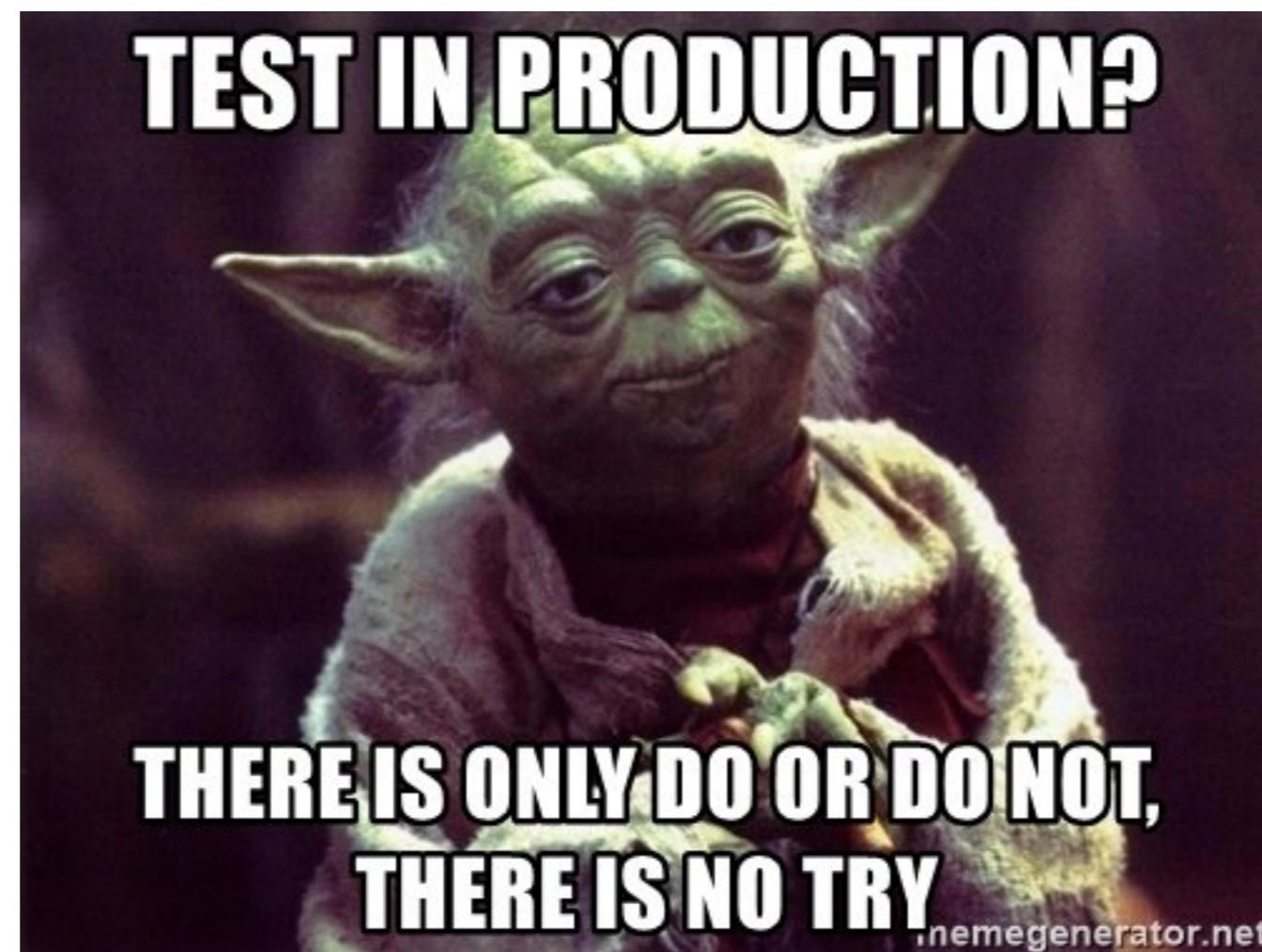
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

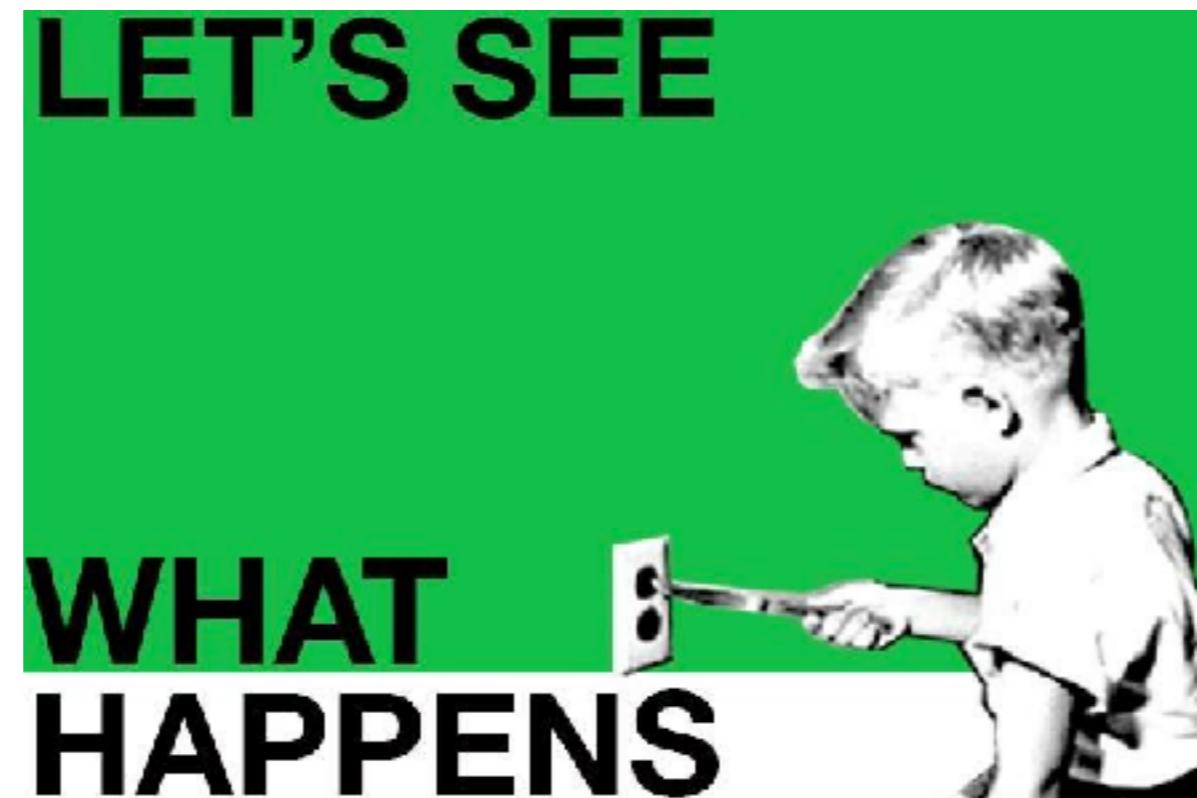
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



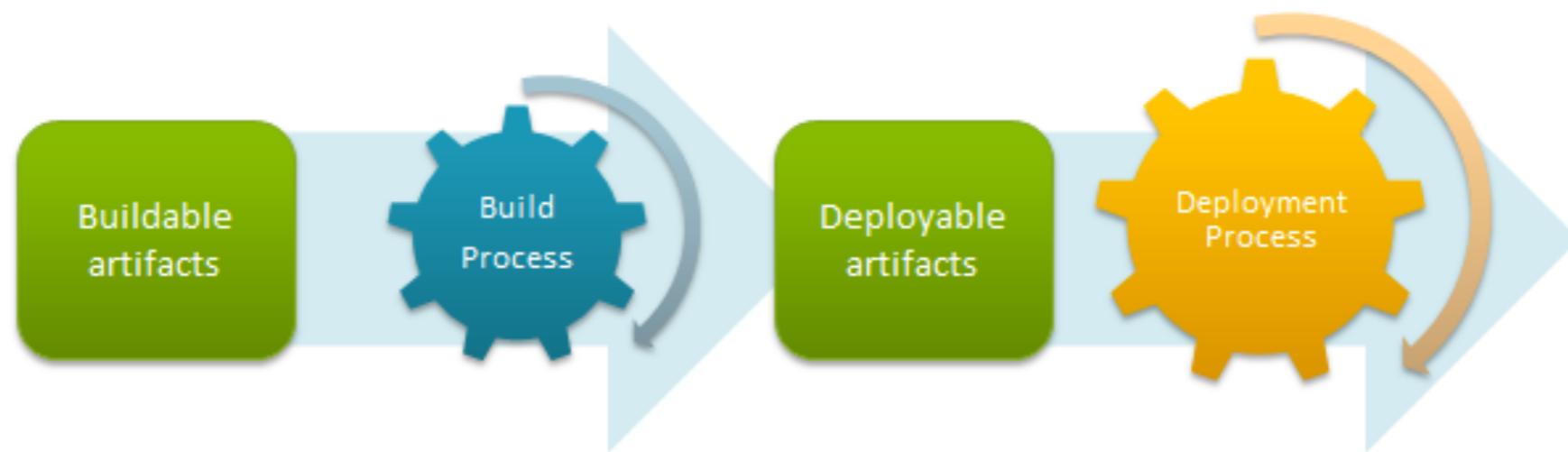
# Practice 10

**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



# Continuous Delivery

Use version control for all production artifacts

Automate your deployment process

Implement continuous integration (CI)

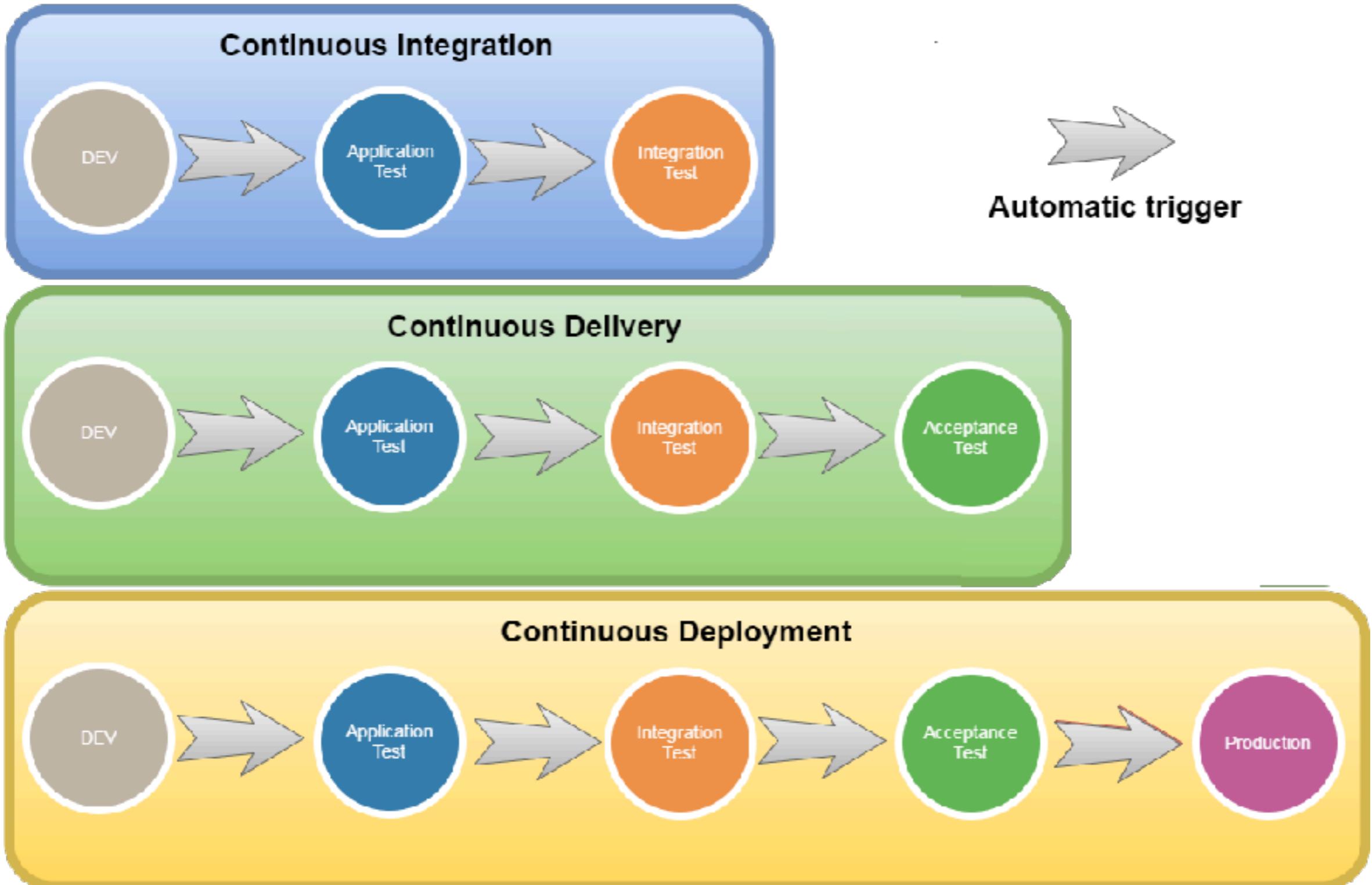
Use trunk-based development methods

Implement test automation

Support test data management

Integrate security into software development process





# **How to achieve the CI ?**



# 1. Use good version control

Local  
Centralize  
Distributed



VSS = A brown粪便 emoji with three white steam or smoke-like lines above it, representing a bad version control system.

JUST SAY NO!



## 2. Choose Branch strategy

Main only

Development isolation

Feature isolation

Release isolation

Service and Release isolation

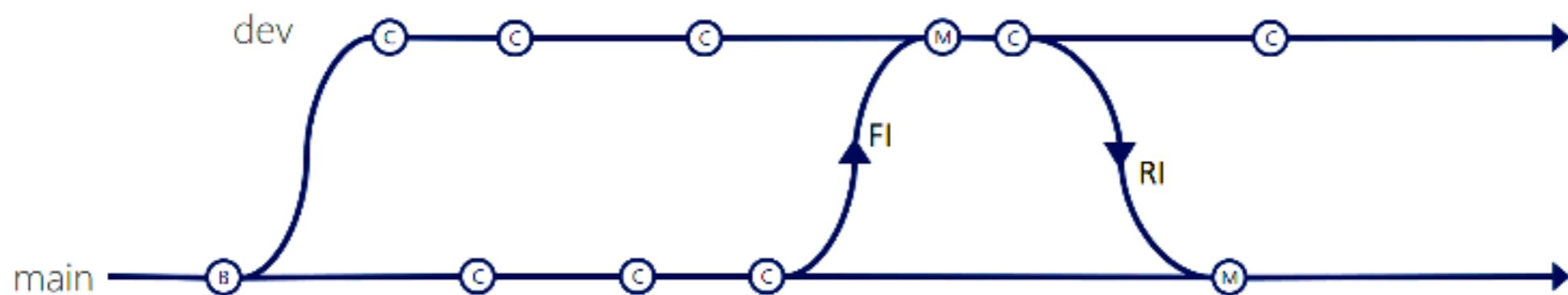
Service, Hotfix and Release isolation



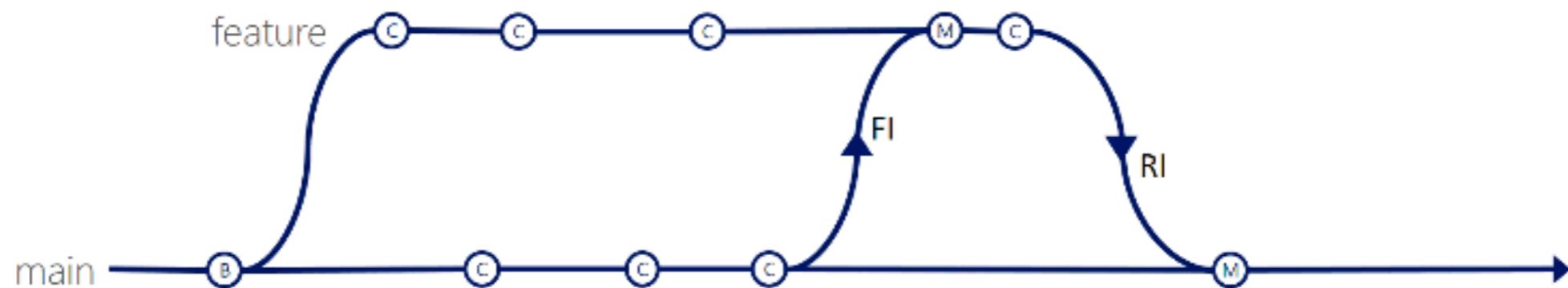
# Main only



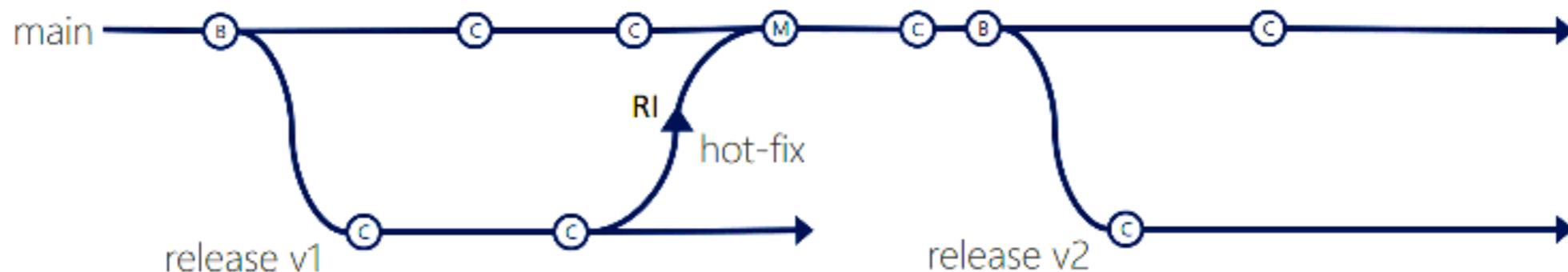
# Development isolation



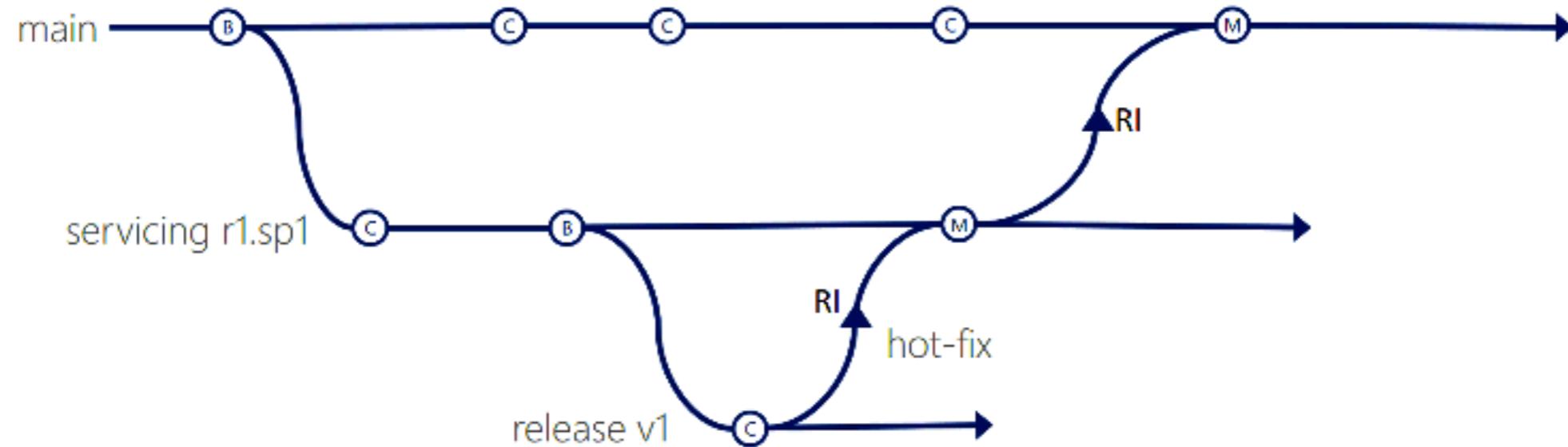
# Feature isolation



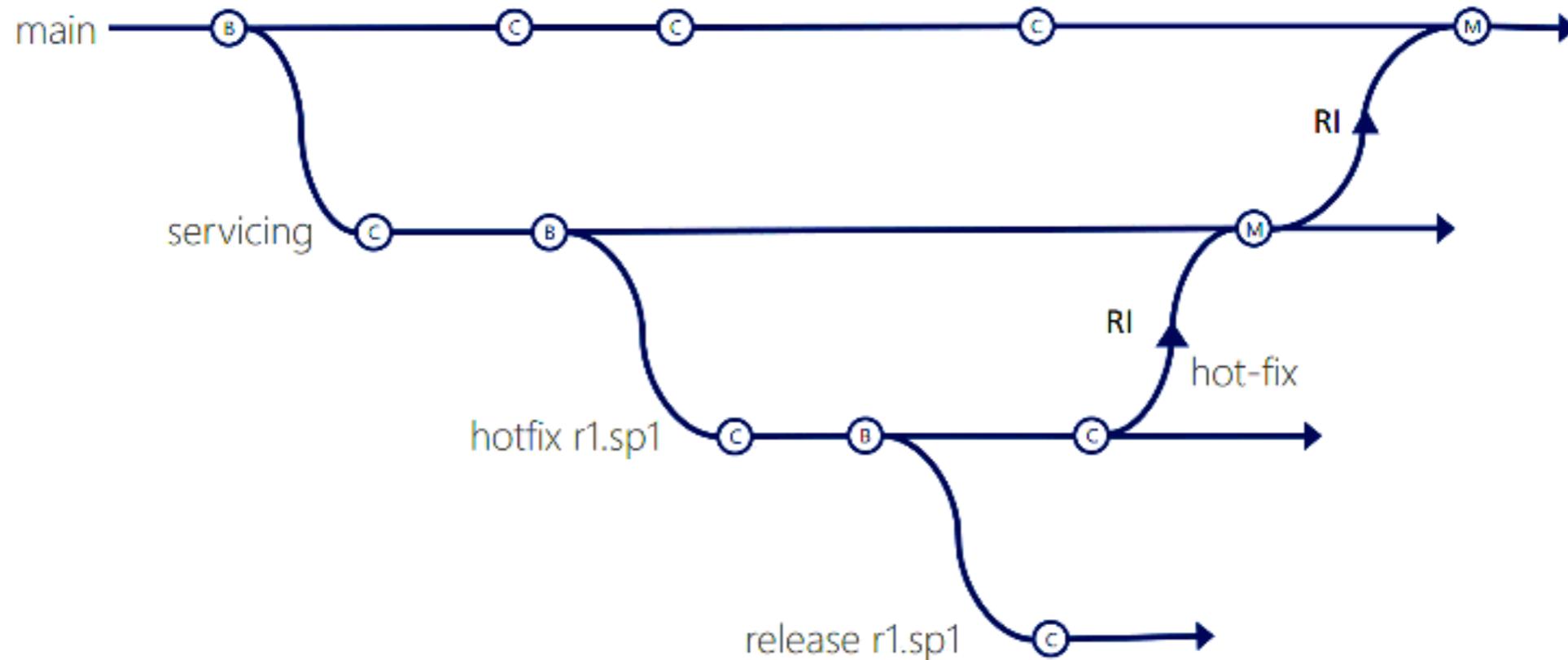
# Release isolation



# Service and Release isolation



# Service, Hotfix, Release isolation



# Validate, Validate and Validate

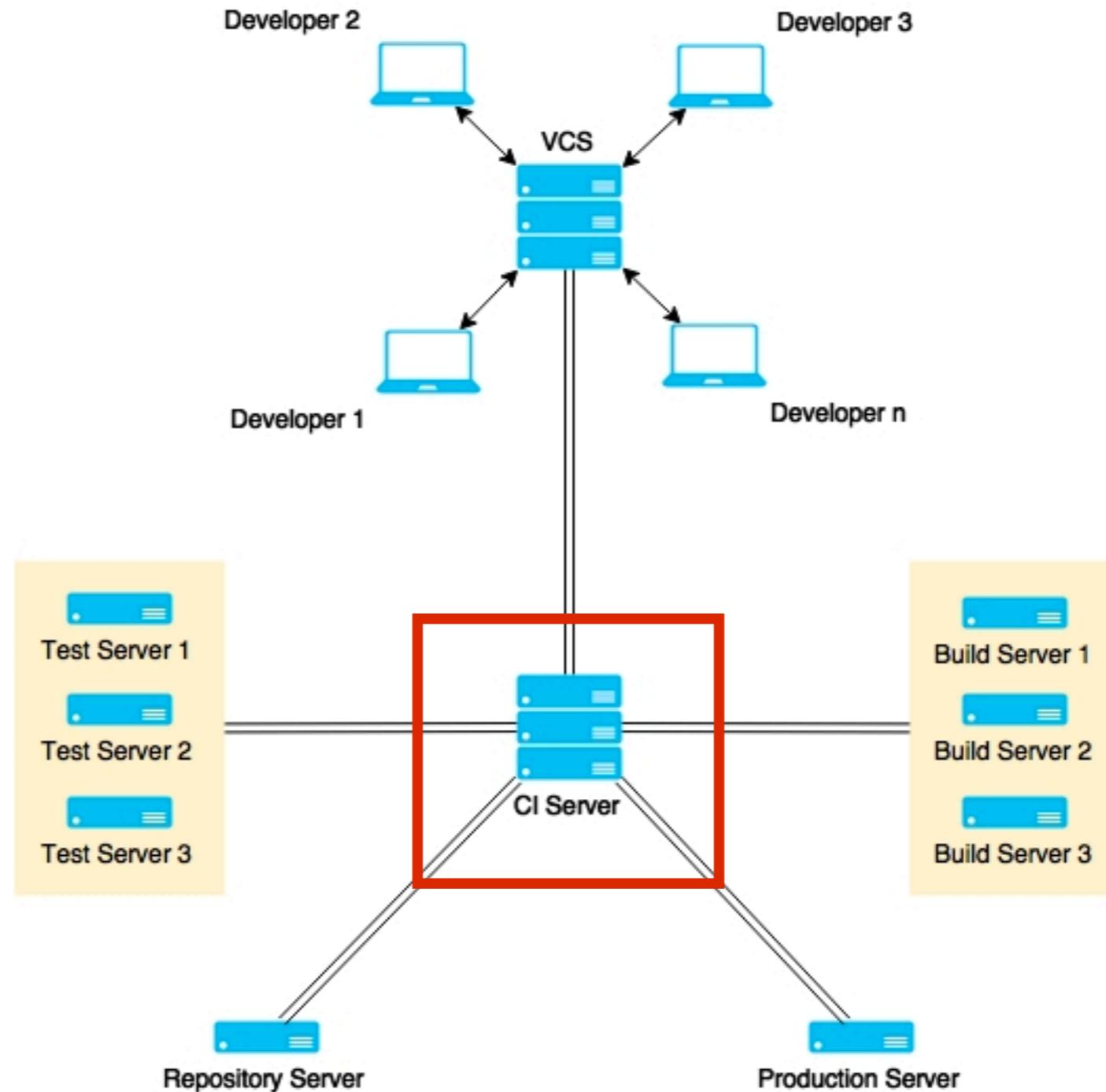


# Suggestion

Keeping branches short-lived, merge conflicts are  
keep to as few as possible



# 3. Use good CI tool





Jenkins

Bamboo



TeamCity

> go<sup>TM</sup>



Hudson



# 4. Use good build tool

- Javascript
  - Gulp, Grunt, Brocolli



- C#/.NET
  - Nant, MSBuild



- Java/JVM
  - Ant, Maven, Gradle, SBT, Leiningen



sbt gradle



# More ...

Use static code analysis  
Automated testing  
Automated deployment  
**People discipline/habit**



**“Behind every successful agile  
project, there is a  
Continuous Integration Server”**



# 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 logo features a cartoon character with a bald head, wearing a blue suit jacket, a red bow tie, and a white shirt. The character is also wearing headphones and holding a white coffee cup. The background of the logo is a red circle.

**Jenkins**

**Build great things at any scale**

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

[Documentation](#) [Download](#)

<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
```



# Change port of Jenkins

```
$java -jar jenkins.war --httpPort=<port>
```



# Open in browser

<http://localhost:8080>

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/Users/somkiat/data/slide/ci-cd/swpark/software/keep/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue



# Copy password from console

```
*****  
*****  
*****
```

Jenkins initial setup is required. An admin user has been created.

Please use the following password to proceed to installation:

a4b3a5231b8048419192d0c5afd3fce8

This may also be found at: /Users/somkiat/data/slide/ci-cd/swp/initialAdminPassword

```
*****  
*****  
*****
```



# Customize plugins

Getting Started



## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.46.3



# Waiting

## Getting Started

## Getting Started

<input type="radio"/> Folders Plugin	<input type="radio"/> OWASP Markup Formatter Plugin	<input type="radio"/> build timeout plugin	<input type="radio"/> Credentials Binding Plugin
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup Plugin	<input type="radio"/> Ant Plugin	<input type="radio"/> Gradle Plugin
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Organization Folder Plugin	<input type="radio"/> Pipeline: Stage View Plugin	<input type="radio"/> Git plugin
<input type="radio"/> Subversion Plug-in	<input type="radio"/> SSH Slaves plugin	<input type="radio"/> Matrix Authorization Strategy Plugin	<input type="radio"/> PAM Authentication plugin
<input type="radio"/> LDAP Plugin	<input type="radio"/> Email Extension Plugin	<input type="radio"/> Mailer Plugin	

\*\* - required dependency

Jenkins 2.46.3



# Success

Getting Started

## Installation Failures

Some plugins failed to install properly, you may retry installing them or continue with

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ build timeout plugin	✓ Credentials Binding Plugin
✓ Timestamper	✓ Workspace Cleanup Plugin	✓ Ant Plugin	✓ Gradle Plugin
✓ Pipeline	✓ GitHub Organization Folder Plugin	✓ Pipeline: Stage View Plugin	✓ Git plugin
✓ Subversion Plug-in	✓ SSH Slaves plugin	✓ Matrix Authorization Strategy Plugin	✓ PAM Authentication plugin
✓ LDAP Plugin	✓ Email Extension Plugin	✓ Mailer Plugin	

Jenkins 2.46.3

[Continue](#)

[Retry](#)



# Create a new user

Getting Started

## Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.46.3

[Continue as admin](#)

[Save and Finish](#)



# Ready to use

Getting Started

## Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

---

Jenkins 2.46.3



# Welcome to Jenkins

Jenkins  somkiat | log out

ENABLE AUTO REFRESH

New Item  add description 

People 

Build History 

Manage Jenkins 

My Views 

Credentials 

Build Queue  -  
No builds in the queue.

Build Executor Status  -  
1 Idle  
2 Idle

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

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



# About Jenkins's HOME

Default of **JENKINS\_HOME** is  
`<path of user>/jenkins`



# About Jenkins's HOME

## Data in JENKINS\_HOME

```
/Users/somkiat/.jenkins
├── config.xml
├── failed-boot-attempts.txt
├── hudson.model.UpdateCenter.xml
├── jenkins.CLI.xml
├── jenkins.install.UpgradeWizard.state
├── jobs
├── logs
├── nodeMonitors.xml
├── nodes
├── plugins
├── queue.xml
├── queue.xml.bak
├── secret.key
├── secret.key.not-so-secret
├── secrets
├── updates
├── userContent
├── users
└── war
```



# About Jenkins's HOME

File and Folder name	Description
config.xml	All about configuration
jobs	Keep all jobs/project
plugins	Keep all plugins
nodes	Keep all nodes
logs	Keep all logs



# Change Jenkins's HOME

## For Windows

```
set JENKINS_HOME=<your path>
```

## For Linux/Mac

```
export JENKINS_HOME=<your path>
```

try to restart Jenkins ...



# Finds Jenkins's plugin



Jenkins

Blog Documentation Plugins Use-cases Participate Sub-projects Resources

# Plugins Index

Discover the 1000+ community contributed Jenkins plugins to support building, deploying and automating any project.

Browse Find plugins... 

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



# Try to install a first plugin

Choose Available tab and select a plugin

The screenshot shows a Jenkins plugin management interface. At the top, there are tabs: 'Updates' (disabled), 'Available' (selected and highlighted with a red border), 'Installed' (disabled), and 'Advanced' (disabled). Below the tabs is a search bar labeled 'Filter:' with a magnifying glass icon. The main area displays a list of available plugins under the heading '.NET Development'. Each plugin entry includes a checkbox, the plugin name, a brief description, and its version number. The plugins listed are: CCM Plug-in (version 3.1), FxCop Runner plugin (version 1.1), MSBuild Plugin (version 1.27), MSTest plugin (version 0.19), MSTestRunner plugin (version 1.3.0), NAnt Plugin (version 1.4.3), NCover.plugin (version 0.3), PowerShell plugin (version 1.3), Violation Comments to Bitbucket Server Plugin (version 1.50), and Violations plugin (version 0.7.11). At the bottom of the page, there are three buttons: 'Install without restart' (disabled), 'Download now and install after restart' (highlighted with a red border), and 'Check now'.

Name	Version
CCM Plug-in	3.1
This plug-in generates the trend report for CCM, an open source static code analysis program.	
FxCop Runner plugin	1.1
MSBuild Plugin	1.27
MSTest plugin	0.19
Generates test reports for MSTest.	
MSTestRunner plugin	1.3.0
NAnt Plugin	1.4.3
NCover.plugin	0.3
PowerShell plugin	1.3
Violation Comments to Bitbucket Server Plugin	1.50
Finds violations reported by code analyzers and comments Bitbucket Server (or Stash) pull requests (or commits) with them.	
Violations plugin	0.7.11

Install without restart   Download now and install after restart   Check now

Update information obtained: 9 hr 37 min ago



# Manage Nodes

Add, remove, status of nodes

Jenkins

Jenkins > Nodes >

ENABLE AUTO REFRESH

Back to Dashboard

Manage Jenkins

New Node

Configure

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

search

S Name ↓ Architecture Clock Difference Free Disk Space Free Swap Space Free Temp Space Response Time

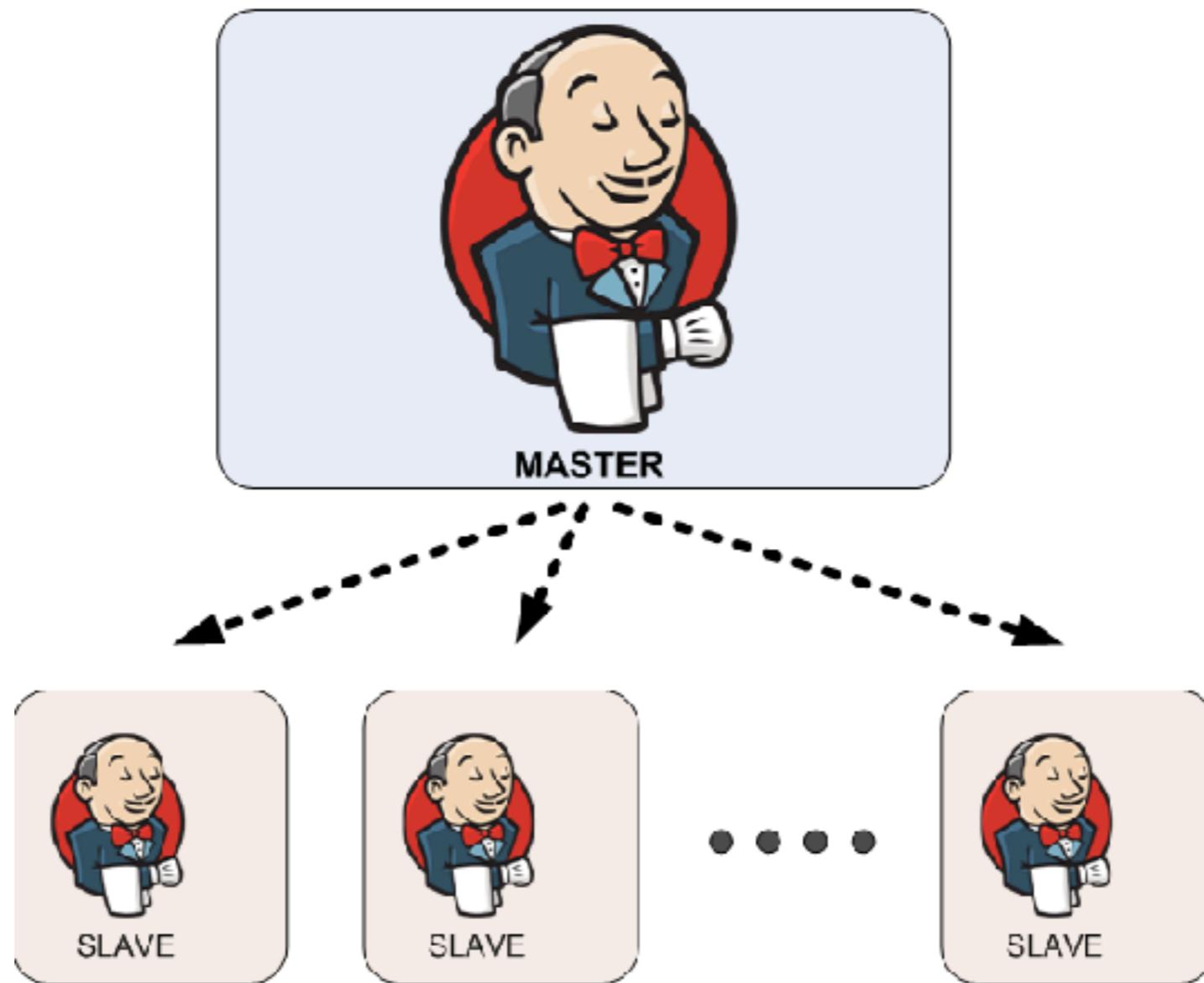
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Mac OS X (x86_64)	In sync	5.73 GB	463.00 MB	5.73 GB	0ms
	Data obtained	36 min	36 min	36 min	36 min	36 min	36 min

Refresh status



# Manage Nodes

Master-slave concept to scale Jenkins



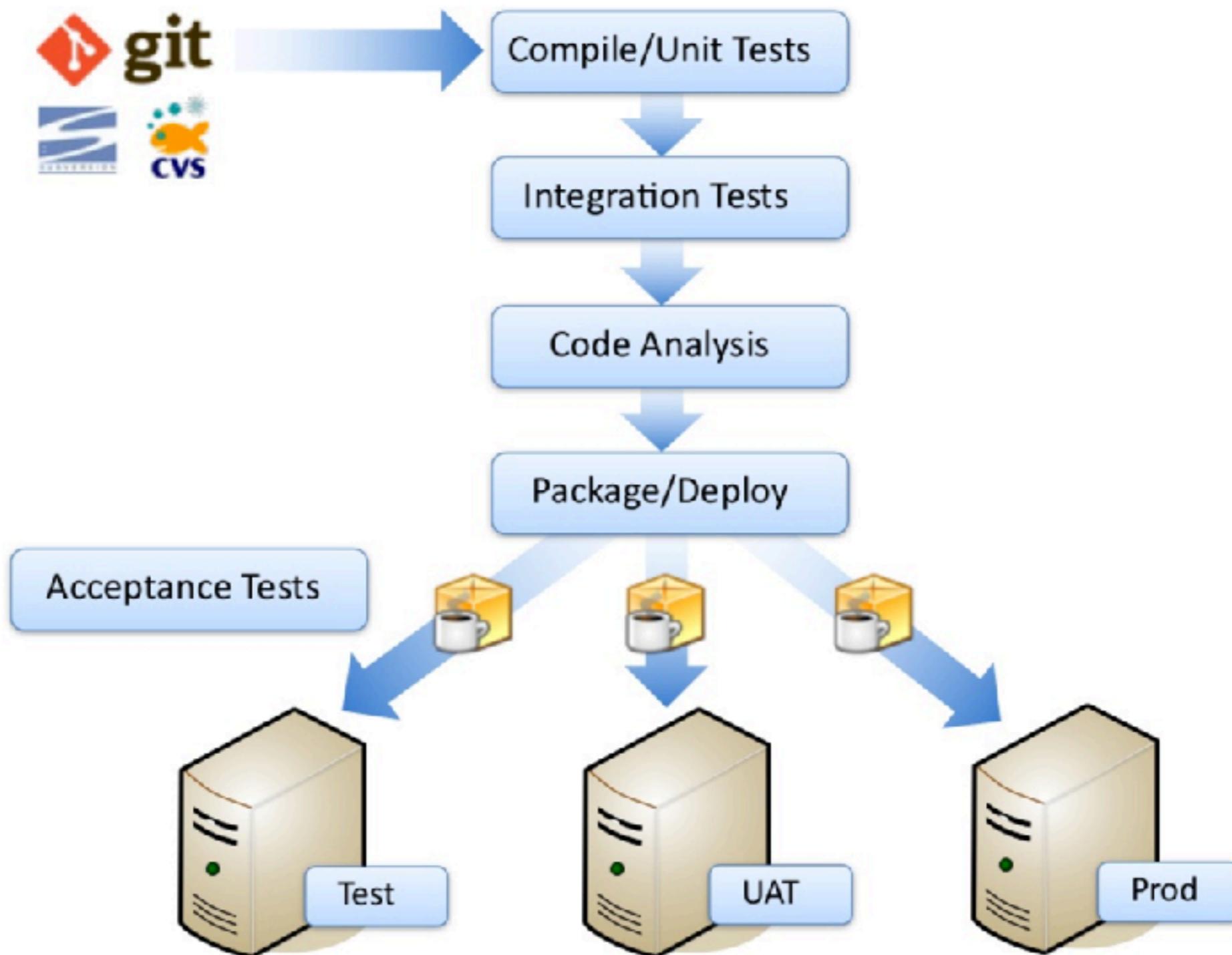
# Create a first Job



# Workshop with Java Web Application

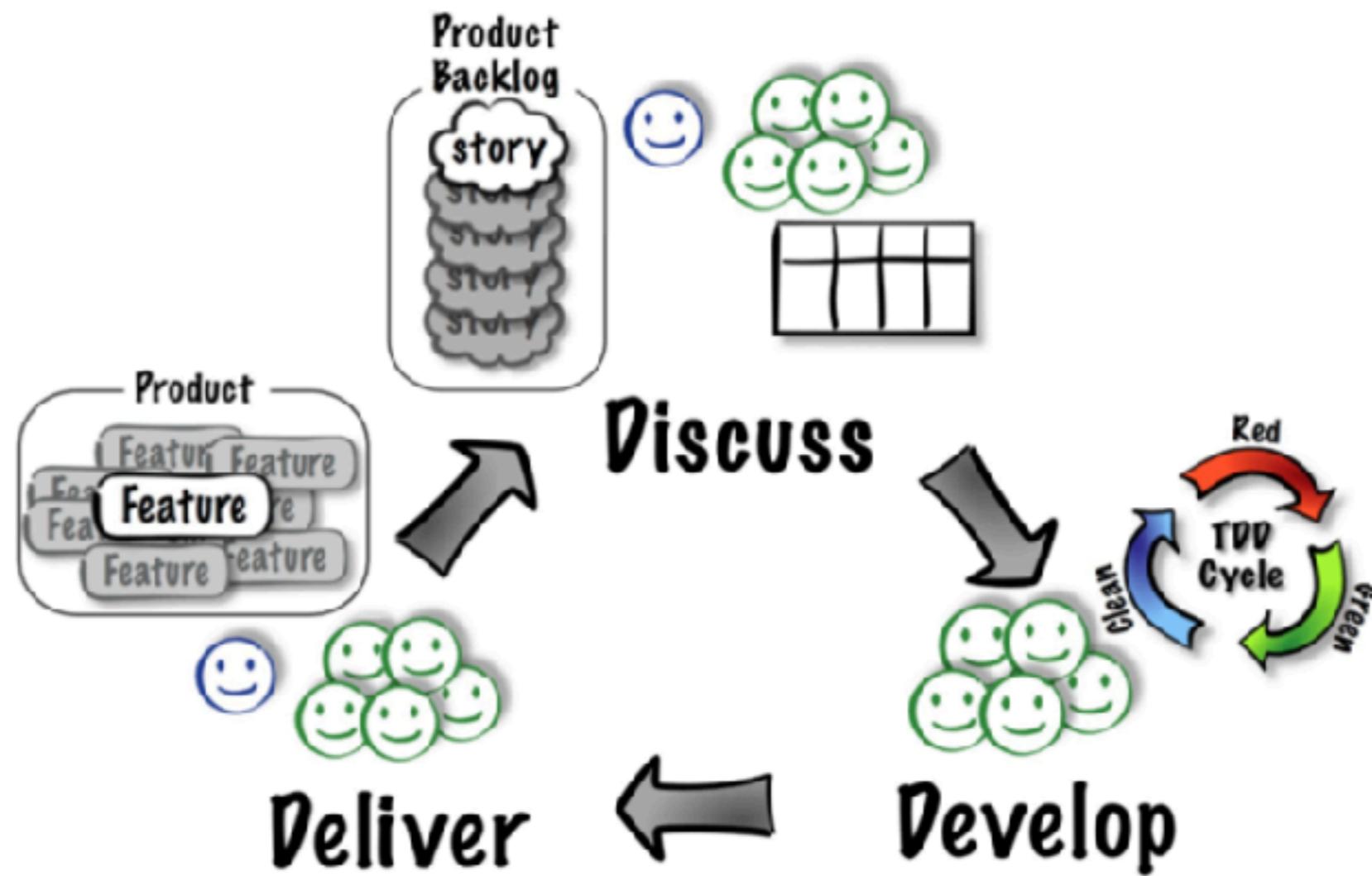


# Pipeline of this project



# Suggestion

“Must setup/configuration CI server before the first feature”



# Tools for development

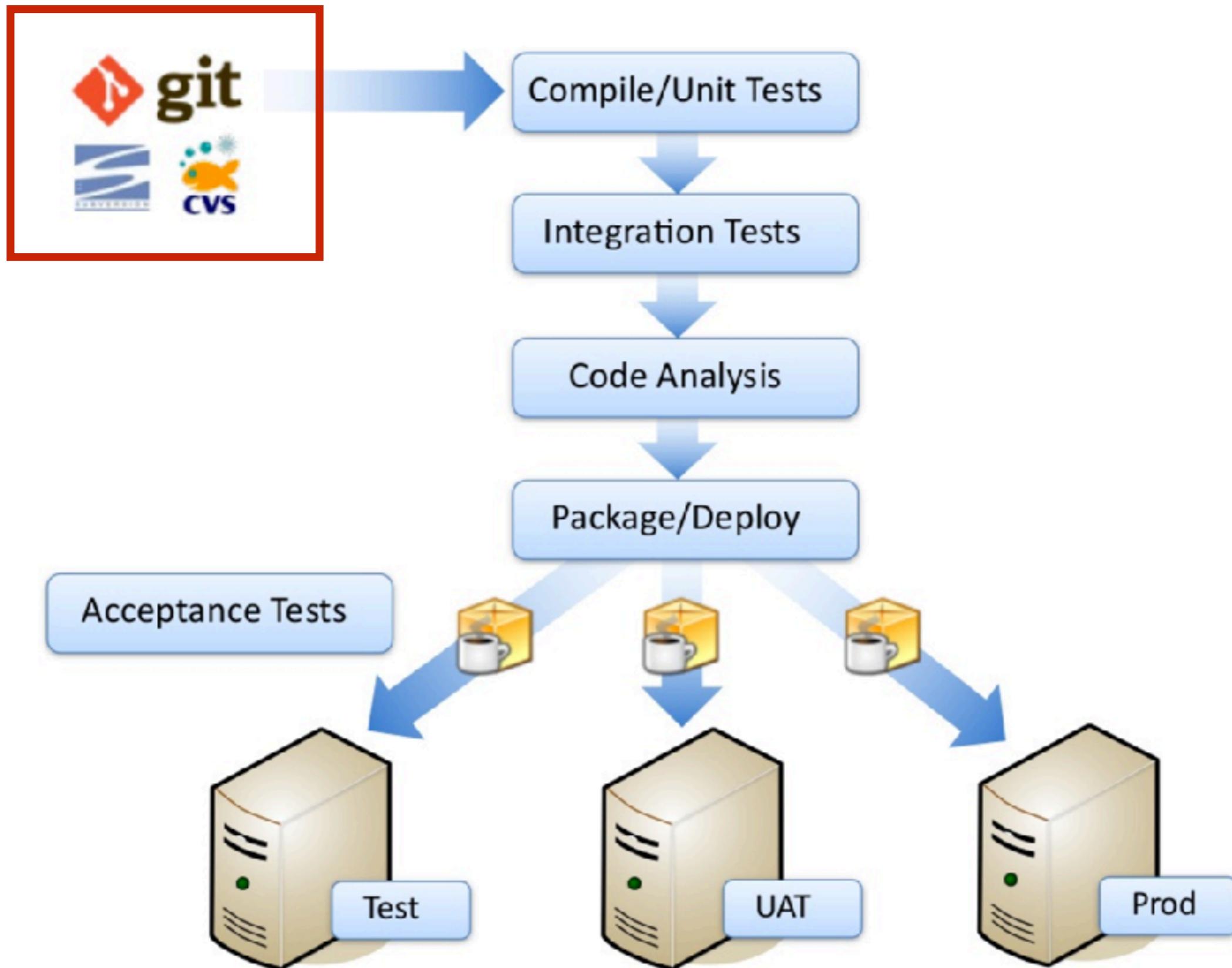
Technology	Description
<b>Java</b>	Main programming language
<b>Apache Maven</b>	Build tool
<b>JUnit</b>	Unit and integration test tool
<b>Cobertura</b>	Code coverage tool
<b>Robotframework</b>	Acceptance test tool
<b>Apache Tomcat</b>	Java Web Server
<b>Git</b>	Version Control System
<b>SonarQube</b>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

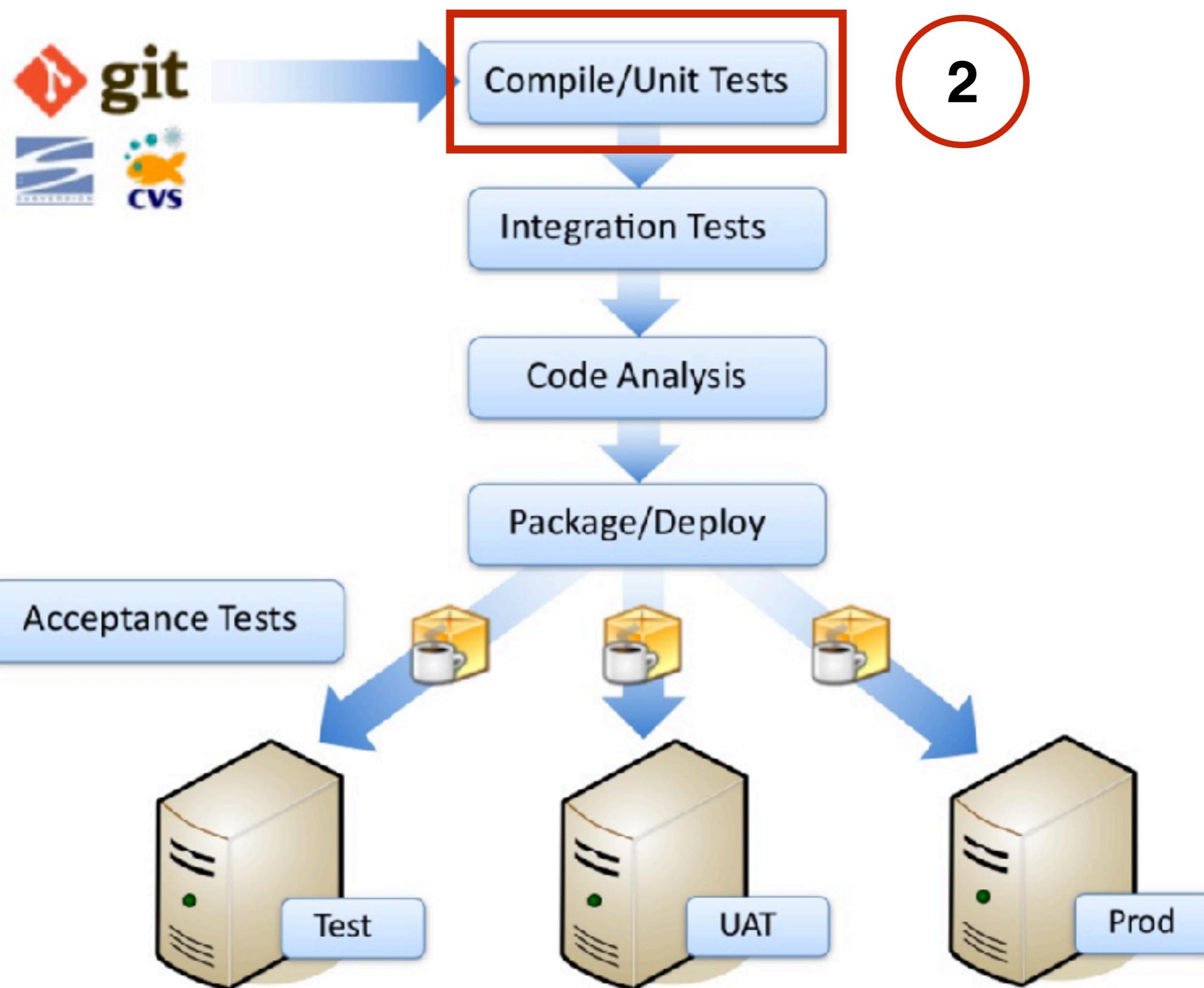


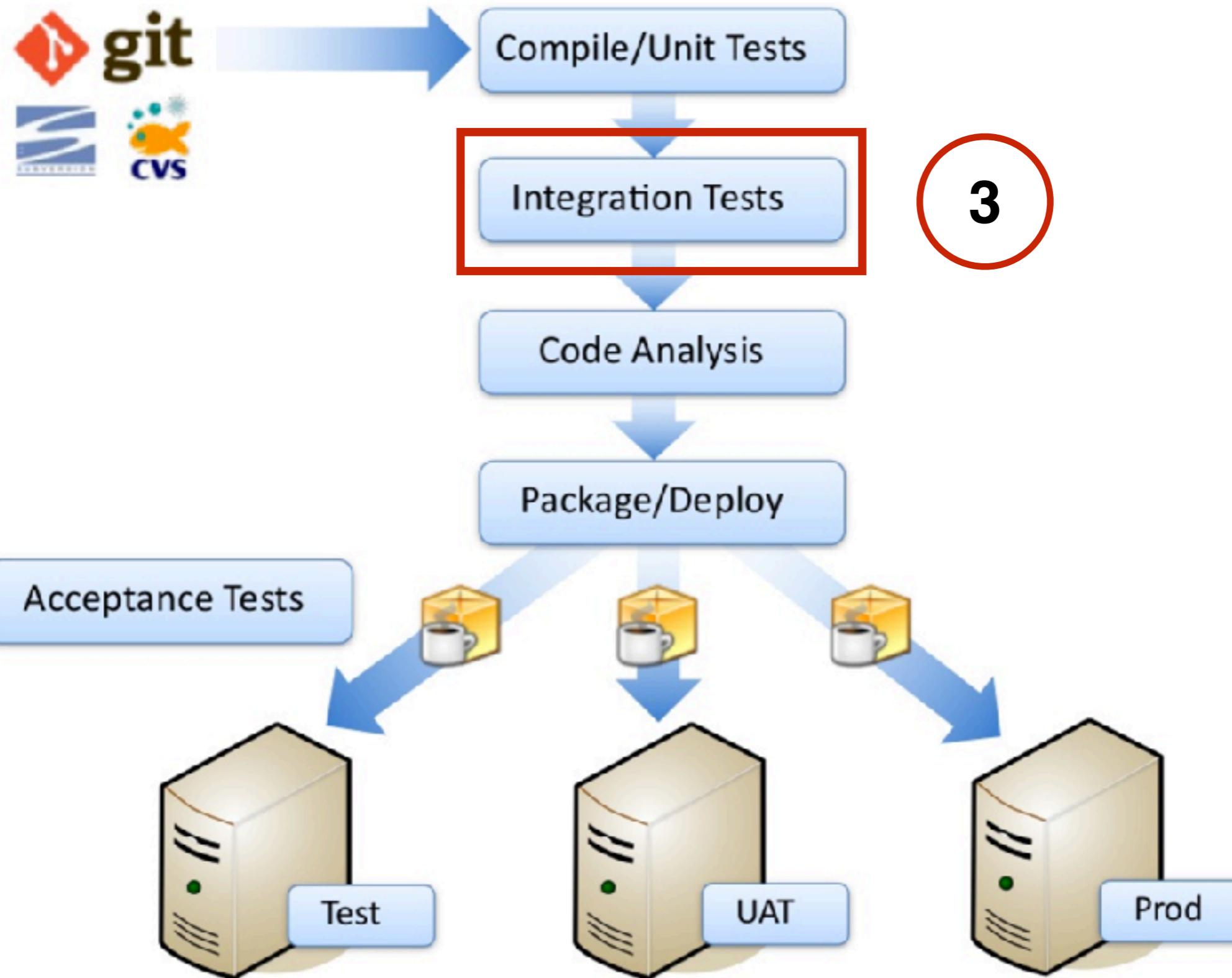
# **Let's create pipeline with Jenkins**



1

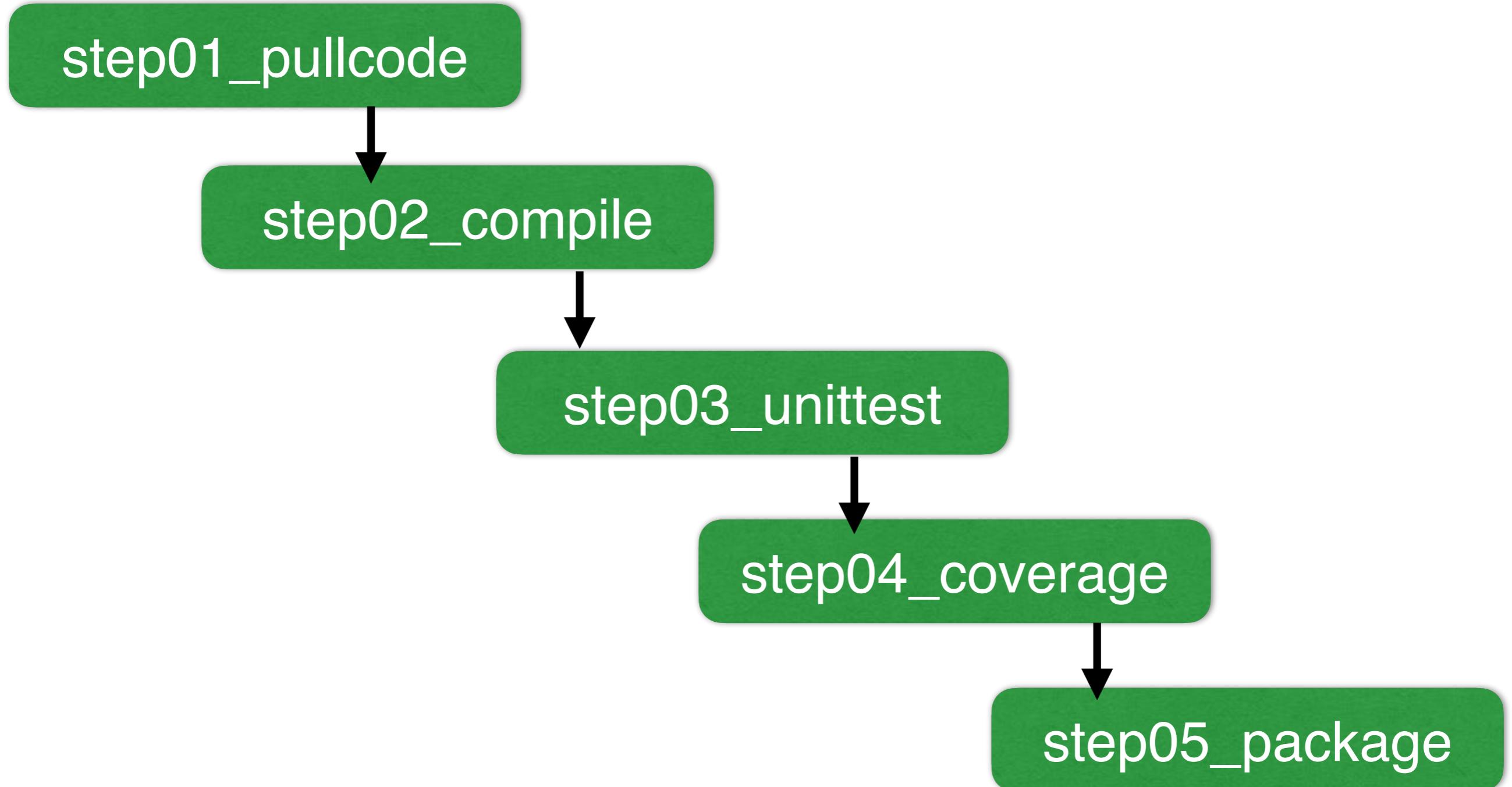


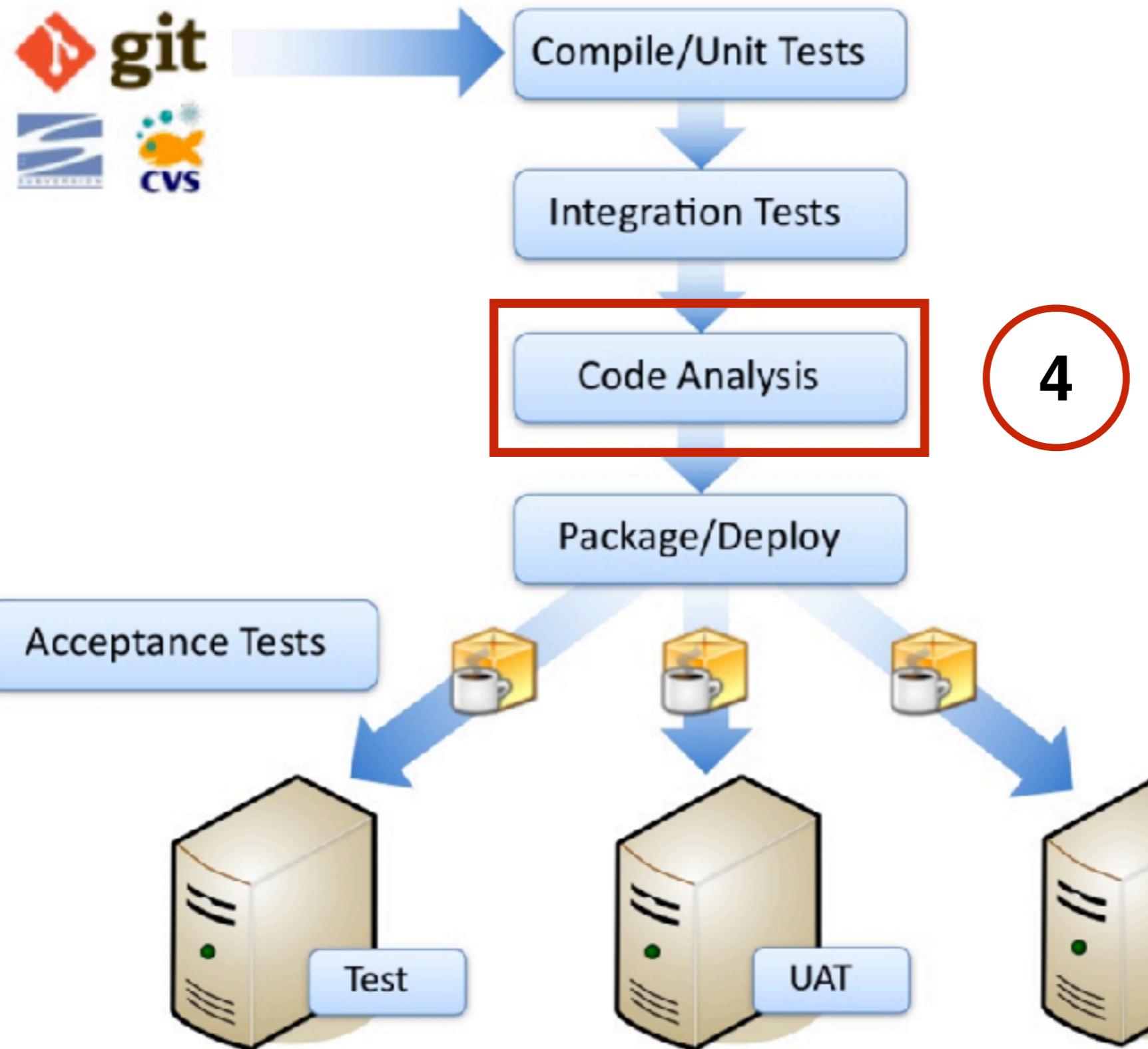


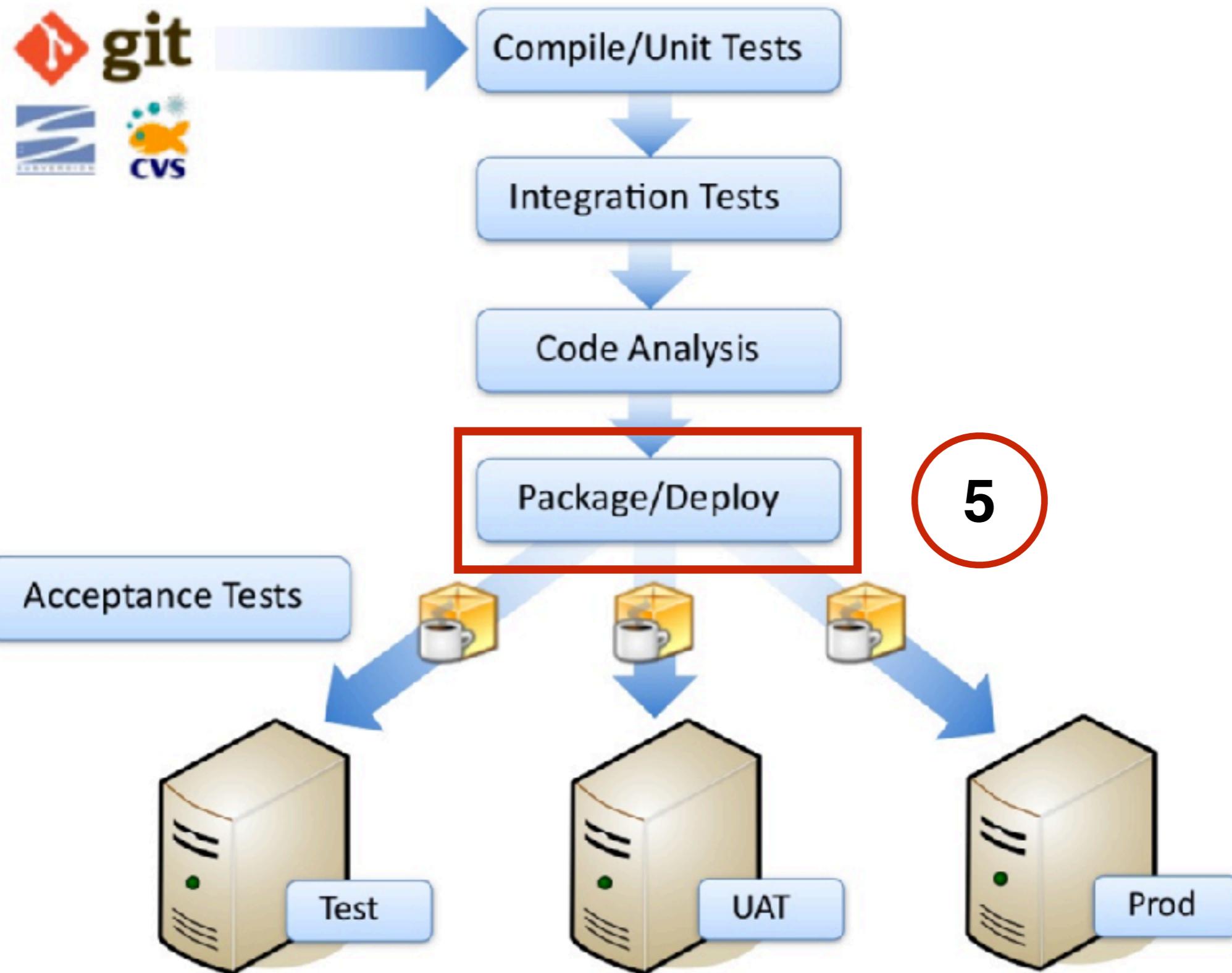


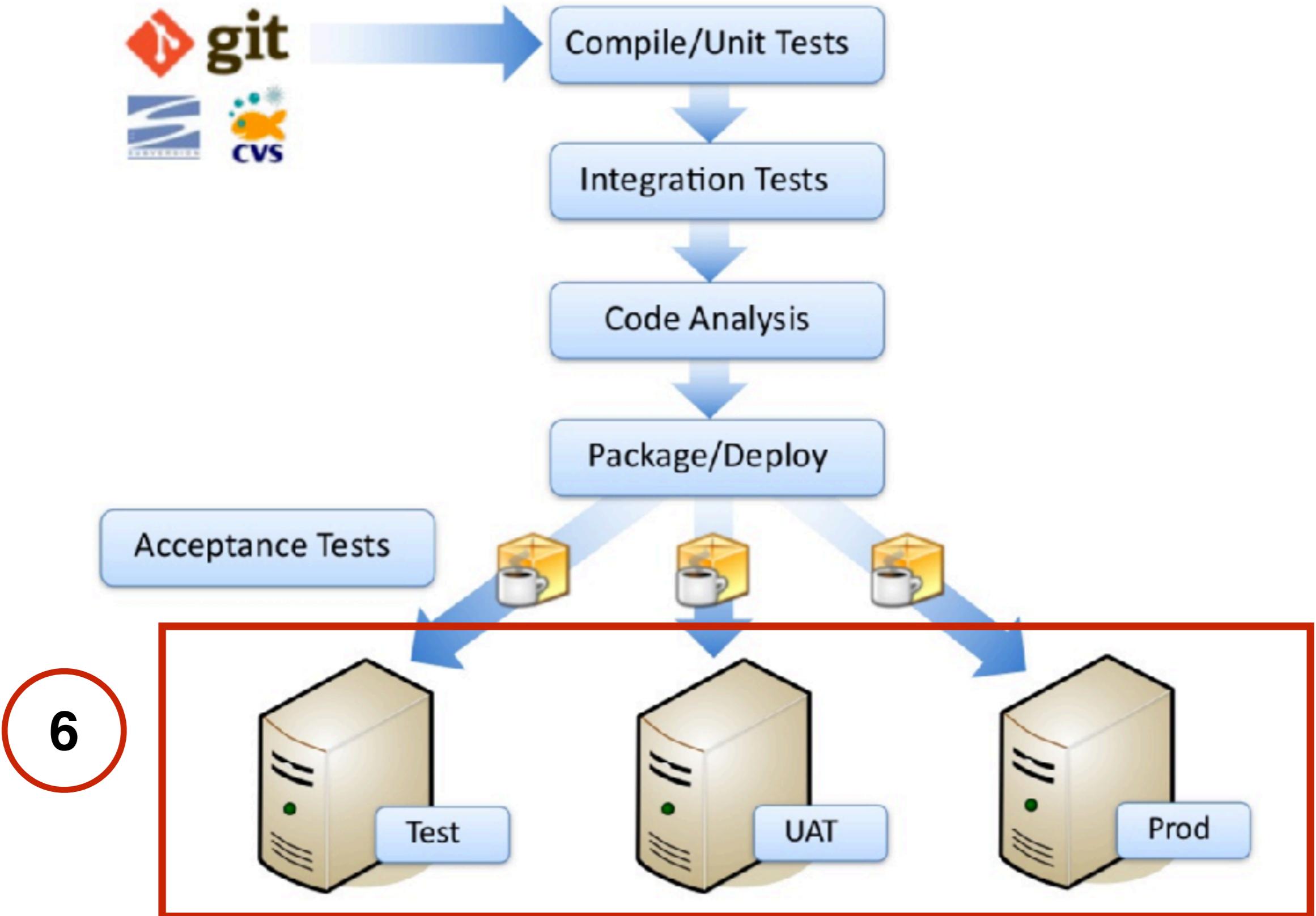
# Suggestion

We need to separate to more jobs





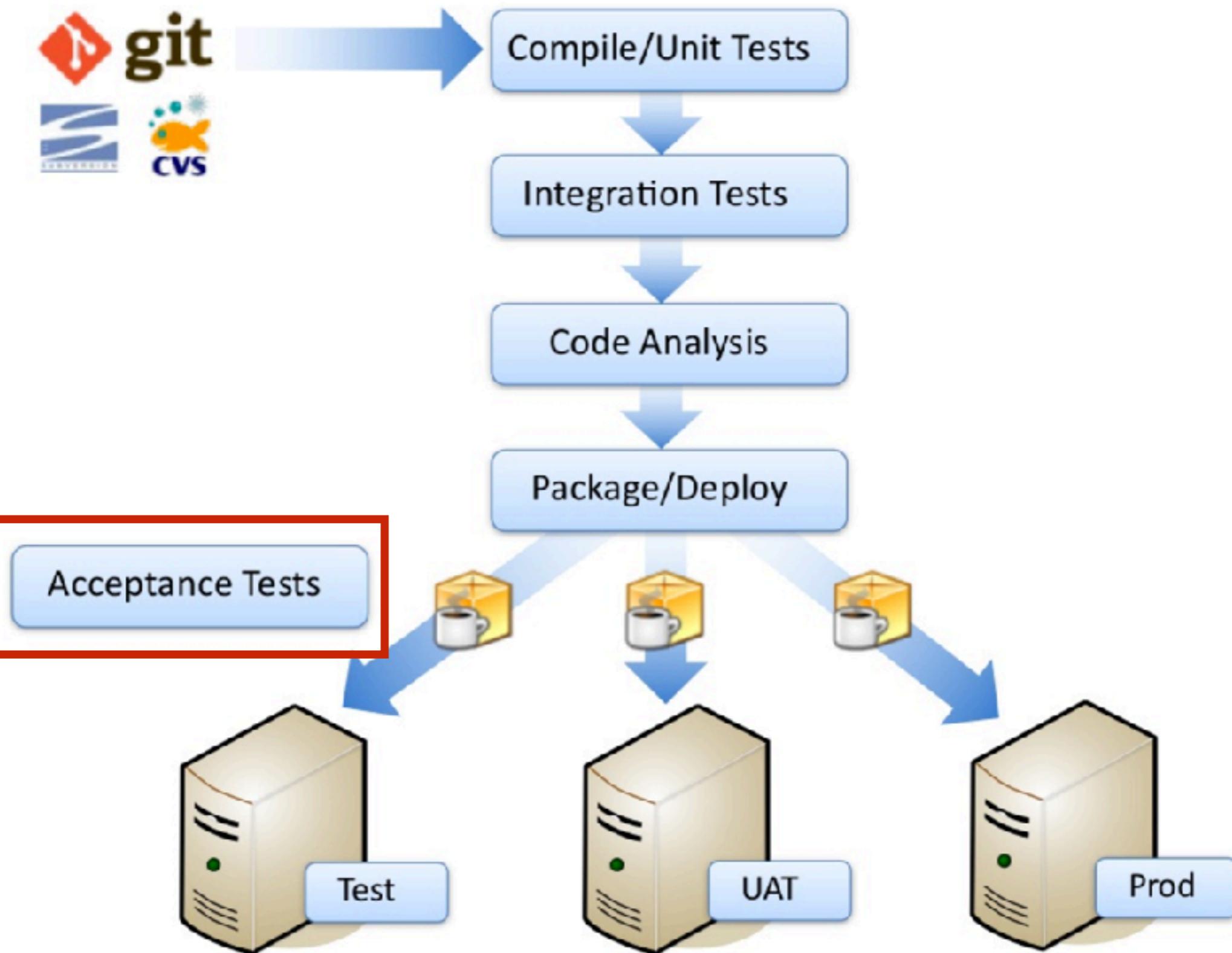






7

Acceptance Tests



# **Working with Pipeline as a Code**



# Pipeline as a Code

Declarative Pipeline

Scripted Pipeline

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



# Create a new job with pipeline

Enter an item name  
project01 » Required field

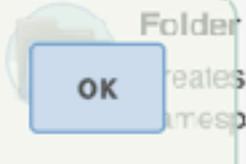
**Freestyle project**  
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**  
 Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**  
 Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**  
 This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

**Multi-configuration project**  
 Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**  
 Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



# Write your pipeline

Pipeline script or from .Jenkinsfile

**Pipeline**

Definition

- ✓ Pipeline script
- Pipeline script from SCM

Script 1 try sample Pipeline... ?

Use Groovy Sandbox ?

[Pipeline Syntax](#)



# 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

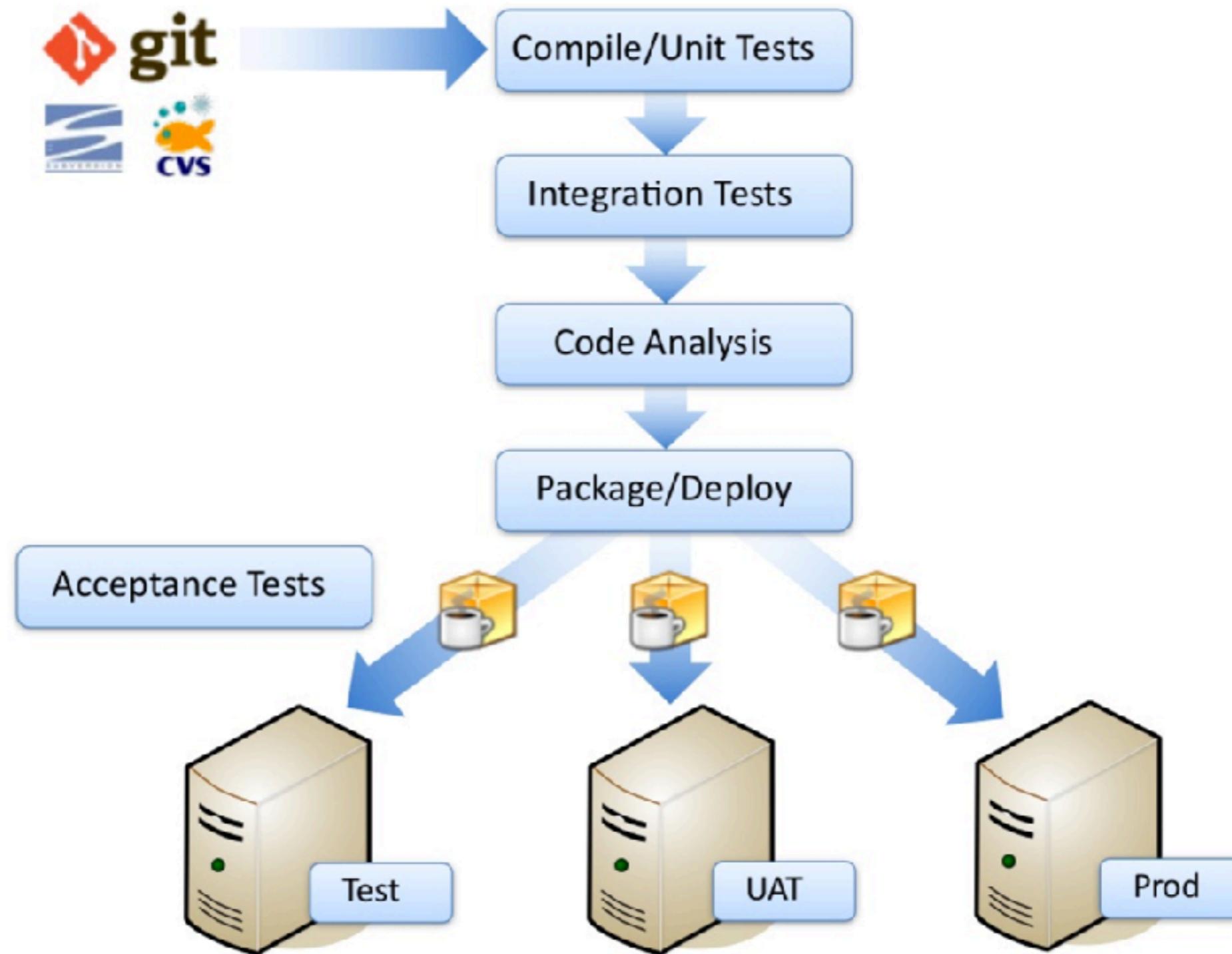
Files to archive

Advanced...

Generate Pipeline Script



# Build pipeline

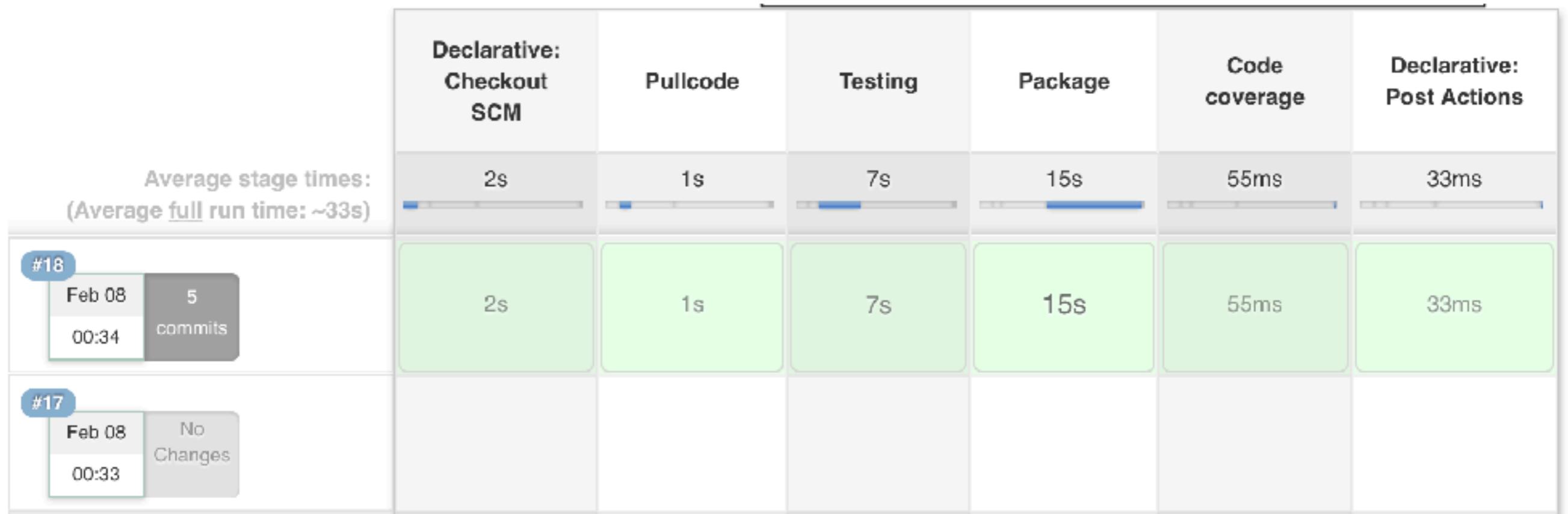


# Create pipeline as code

```
node {  
    stage('Pullcode') {  
        git 'https://github.com/up1/workshop-java-web-tdd.git'  
    }  
    stage('Testing') {  
        sh "mvn clean test"  
        junit 'target/surefire-reports/*.xml'  
    }  
    stage('Package') {  
        sh "mvn package"  
    }  
    stage('Code coverage') {  
        cobertura autoUpdateHealth: false, autoUpdateStability: false  
    }  
}
```



# Result



# Jenkinsfile

```
pipeline {  
    agent any  
    stages {  
        stage('Pullcode') {  
            steps {  
                git 'https://github.com/up1/workshop-java-web-tdd.git'  
            }  
        }  
        stage('Testing') {  
            steps {  
                sh "mvn clean test"  
                junit 'target/surefire-reports/*.xml'  
            }  
        }  
    }  
}
```



# Jenkinsfile

```
stage('Package') {
    steps {
        sh "mvn package"
    }
}
stage('Code coverage') {
    steps {
        cobertura autoUpdateHealth: false, autoUpdateStability: fa
    }
}
post {
    always {
        junit 'target/surefire-reports/*.xml'
    }
}
```



# Try it by yourself



# Useful plug-ins



# Useful plug-ins

Build Pipeline

Build Monitor View

Cobertura Report

Jacoco Report

Robotframework Report

Sound

Green Ball

SonarQube Scanner

Blue Ocean



# Jenkins Best Practices

<https://wiki.jenkins.io/display/JENKINS/Jenkins+Best+Practices>



# Jenkins Best Practices

**Always secure Jenkins**

In large system, don't build on the master

**Backup JENKINS\_HOME regularly**

Limit project name to the sane character set

<https://wiki.jenkins.io/display/JENKINS/Administering+Jenkins>



# Jenkins Best Practices

Always config your job to generate report  
Archive unused jobs before removing them  
Setting difference job for each branch  
Prevent resource collisions in job (parallel)  
Fail fast



# Recap Day 1



# Recap day 1

CI and CD concepts

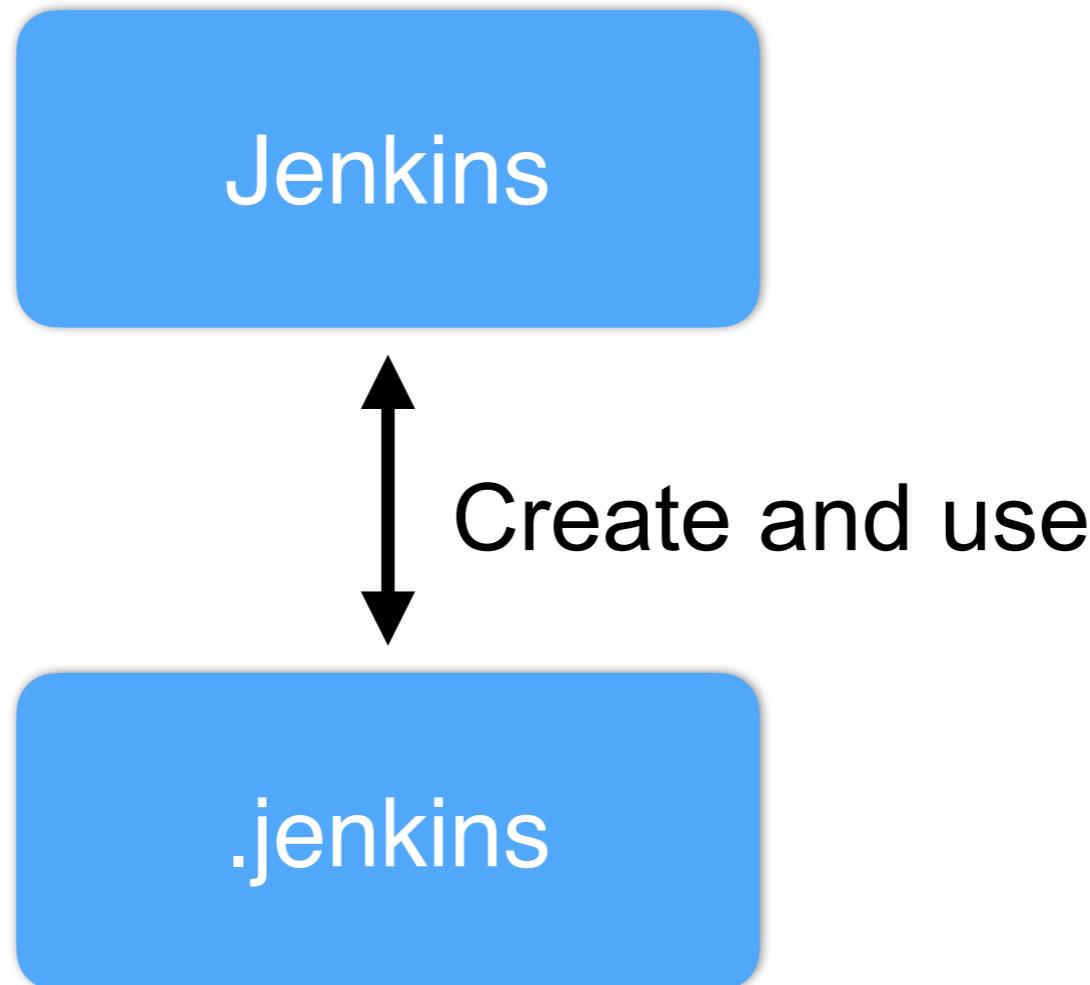
CI principles and practices

Install and config Jenkins

Workshop with Java Web application



# Recap day 1 :: Jenkins Home



`${JENKINS_HOME} = [Home directory]/.jenkins`



# Recap day 1 :: Jenkins Home

set JENKINS\_HOME=<>

Jenkins

Home 1

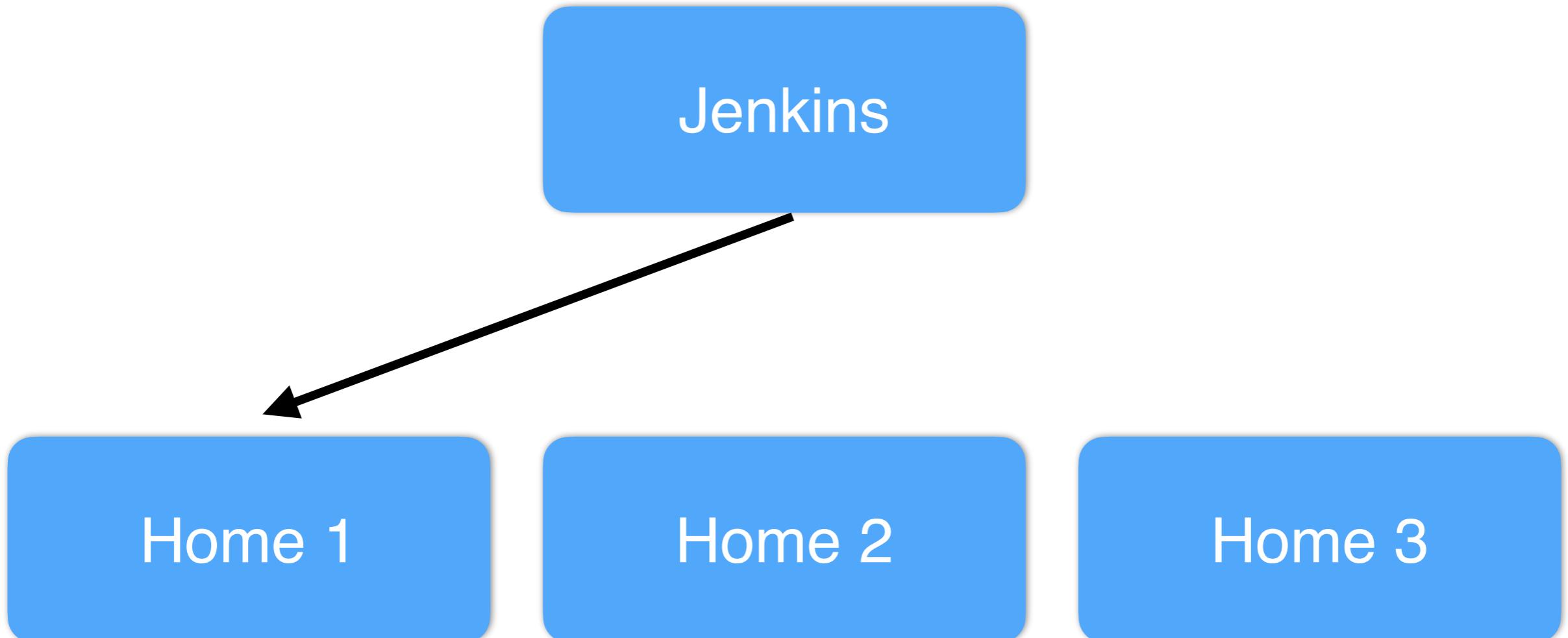
Home 2

Home 3



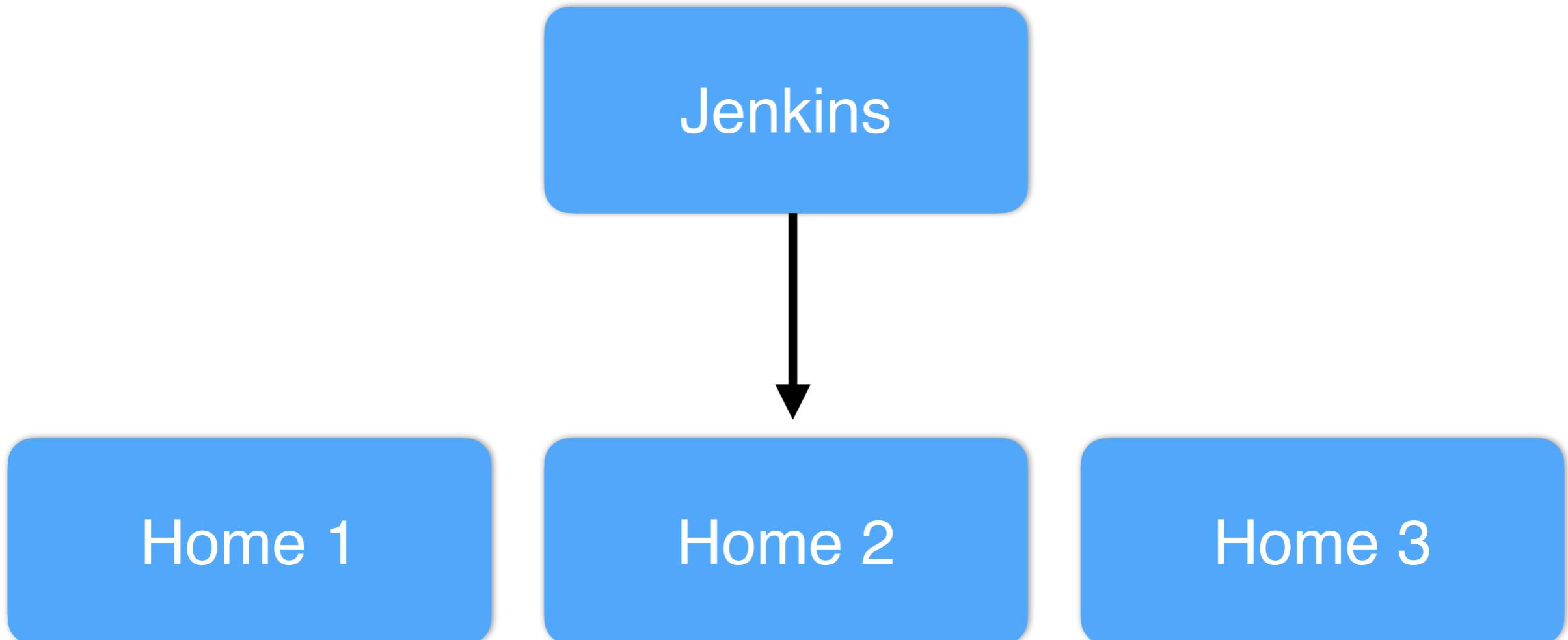
# Recap day 1 :: Jenkins Home

set JENKINS\_HOME=Home 1



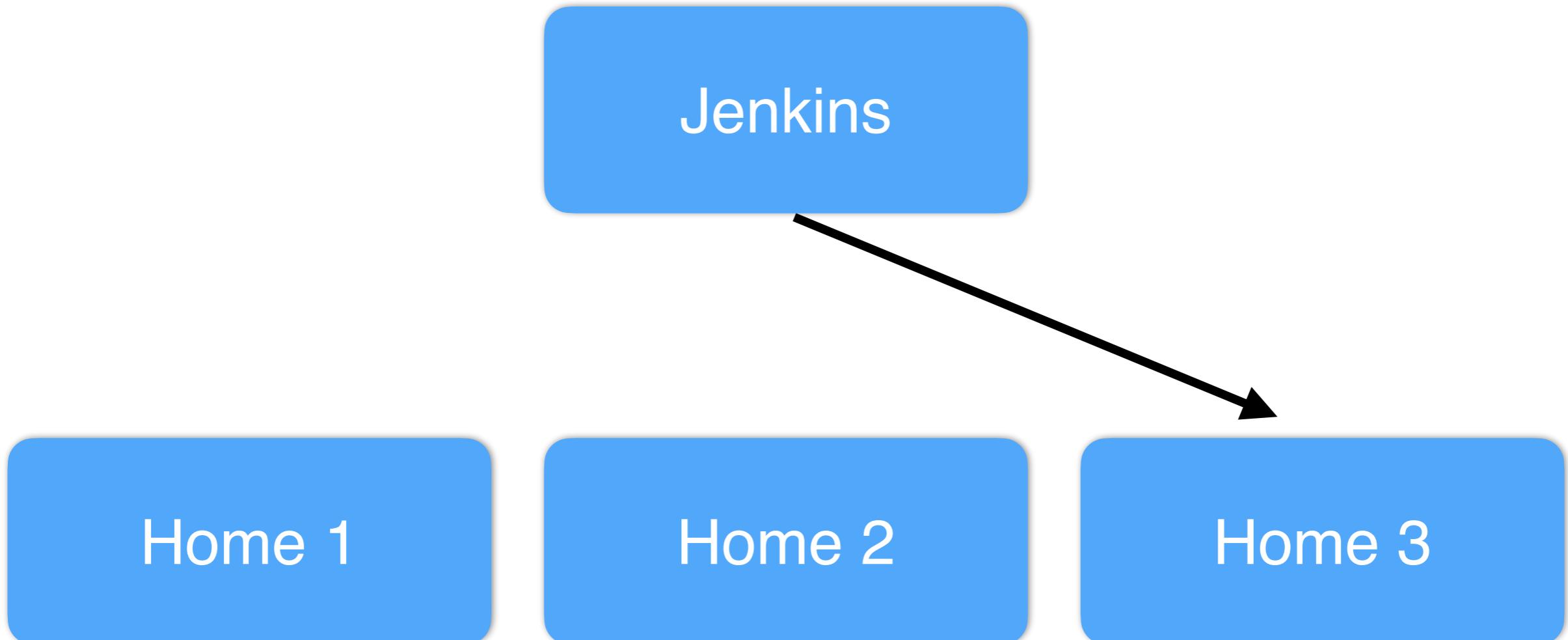
# Recap day 1 :: Jenkins Home

set JENKINS\_HOME=Home 2

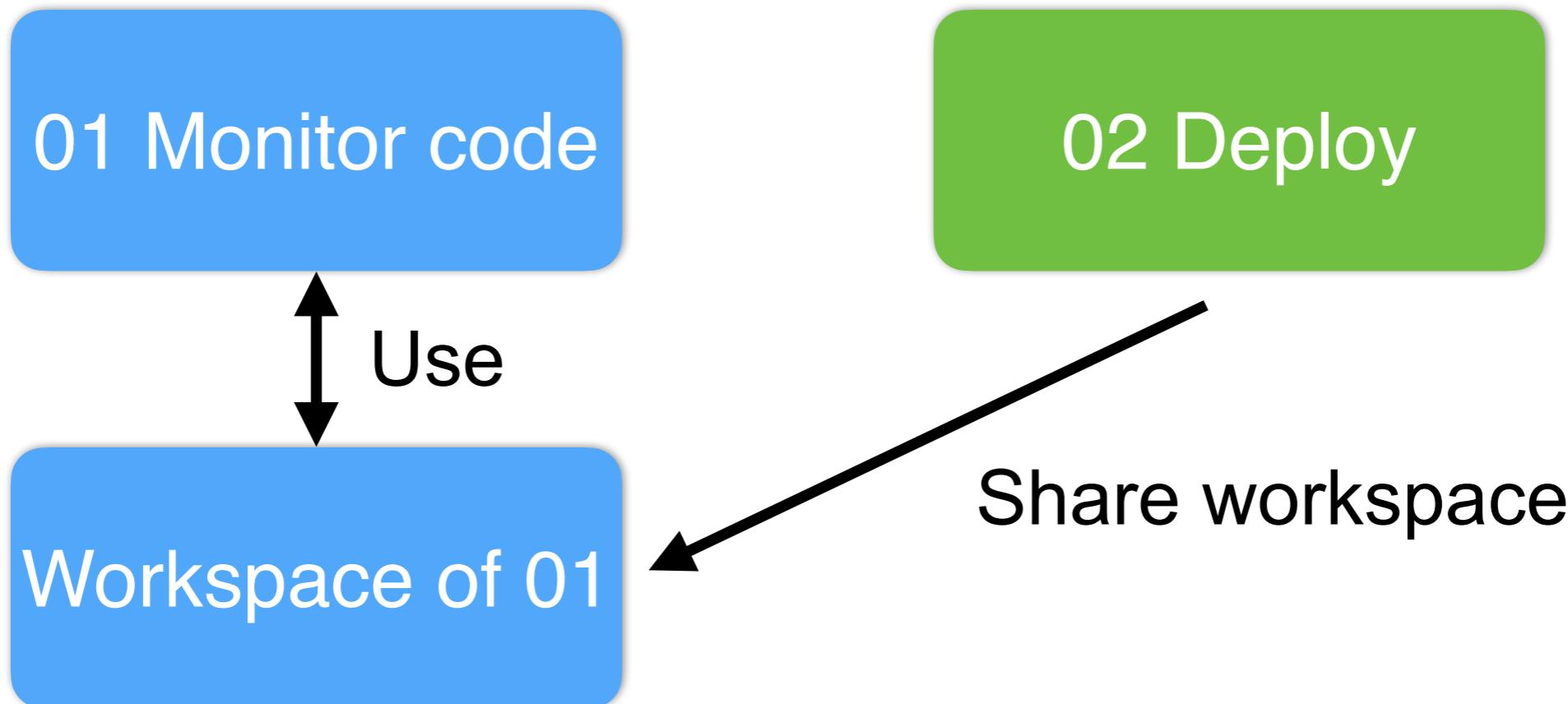


# Recap day 1 :: Jenkins Home

set JENKINS\_HOME=Home 3



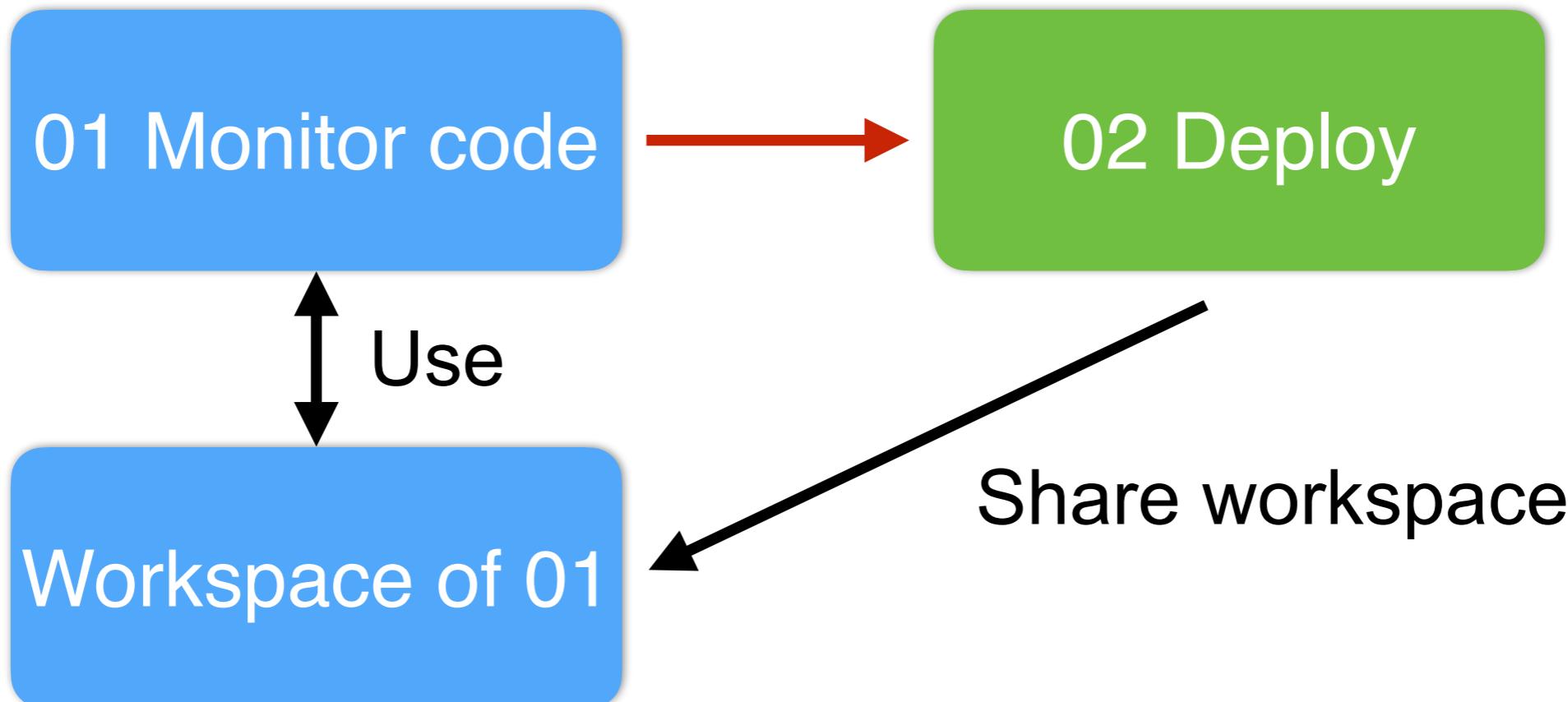
# Recap day 1 :: pipeline



`${JENKINS_HOME}/workspace`



# Recap day 1 :: pipeline



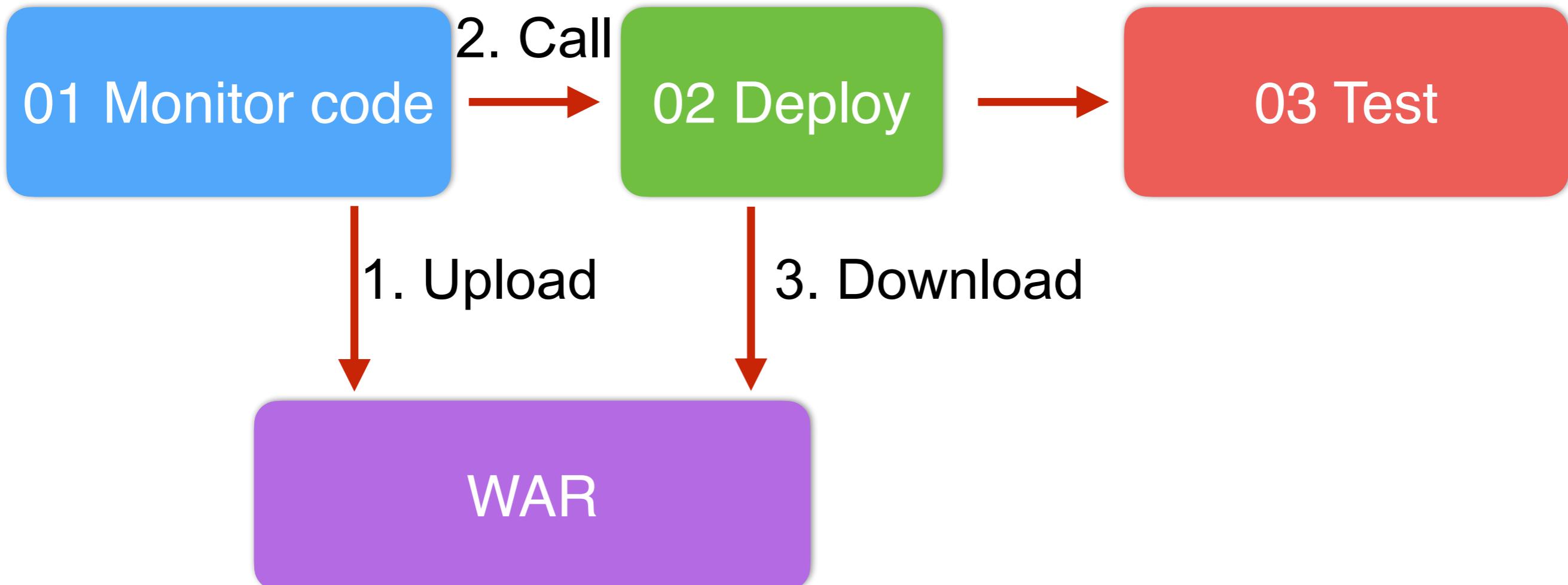
`${JENKINS_HOME}/workspace`



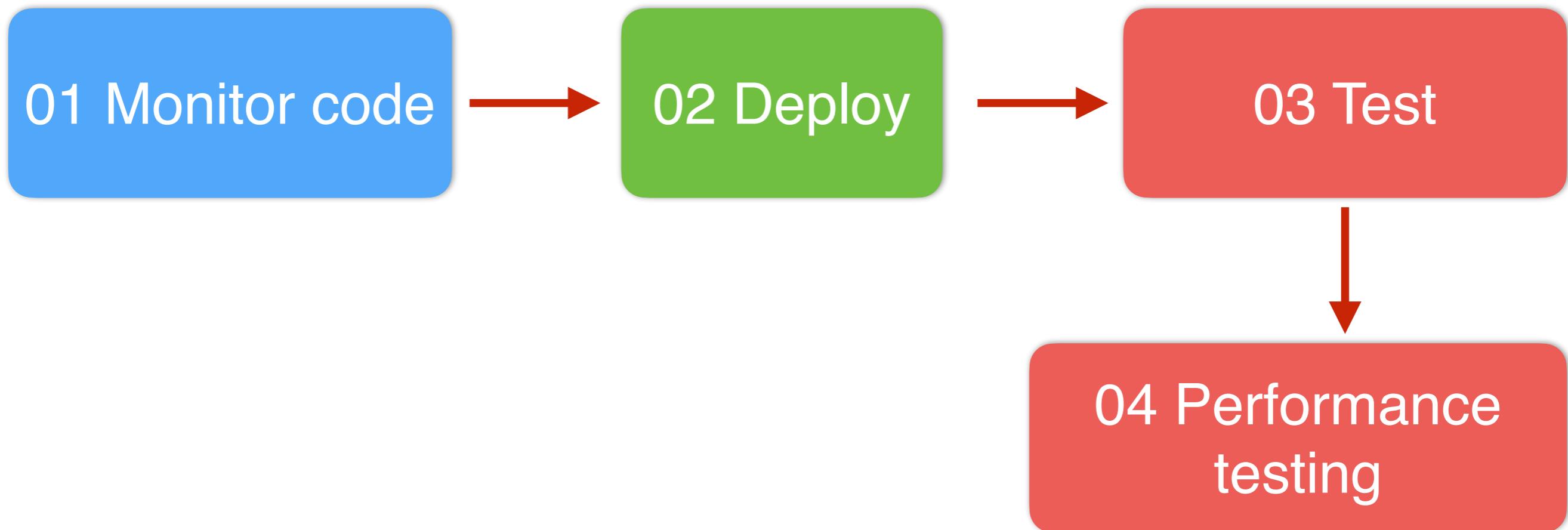
# UI testing



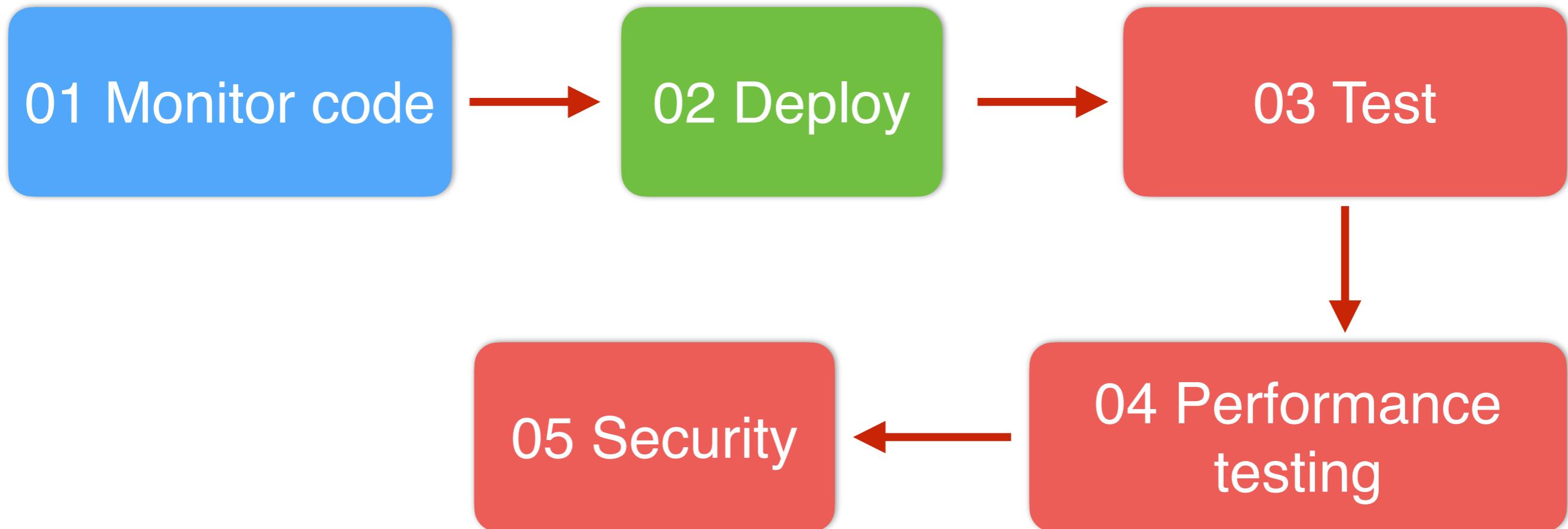
# Share Software Package (Artifact)



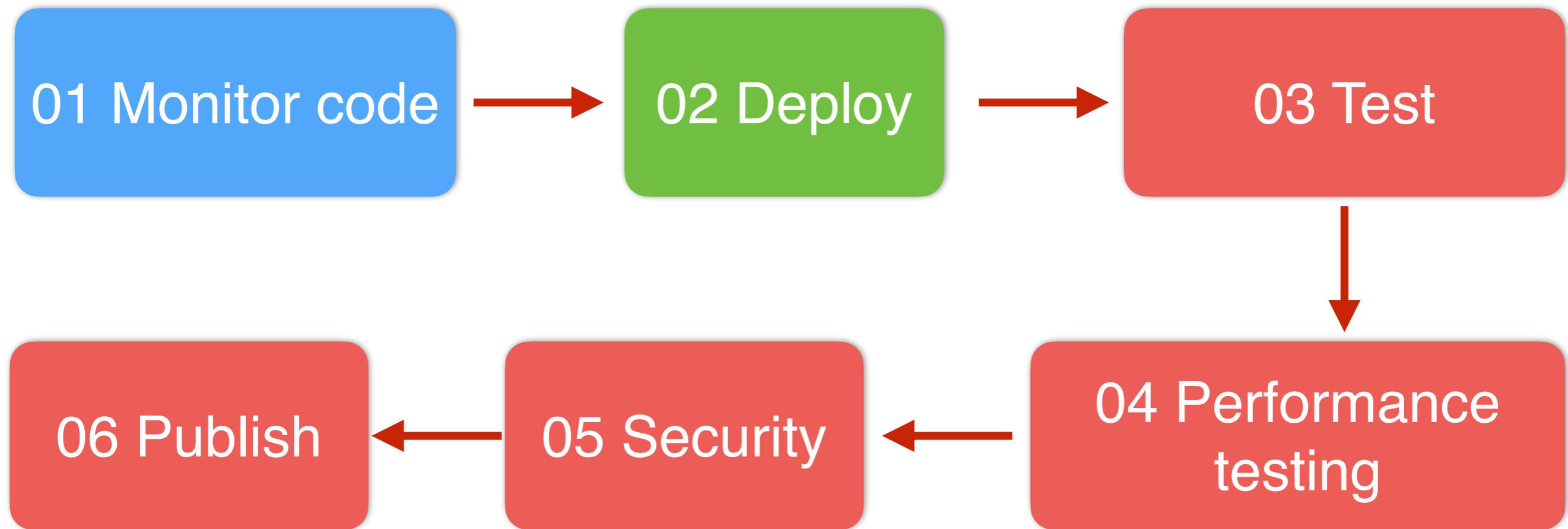
# Performance testing



# Security testing



# Publish Artifact



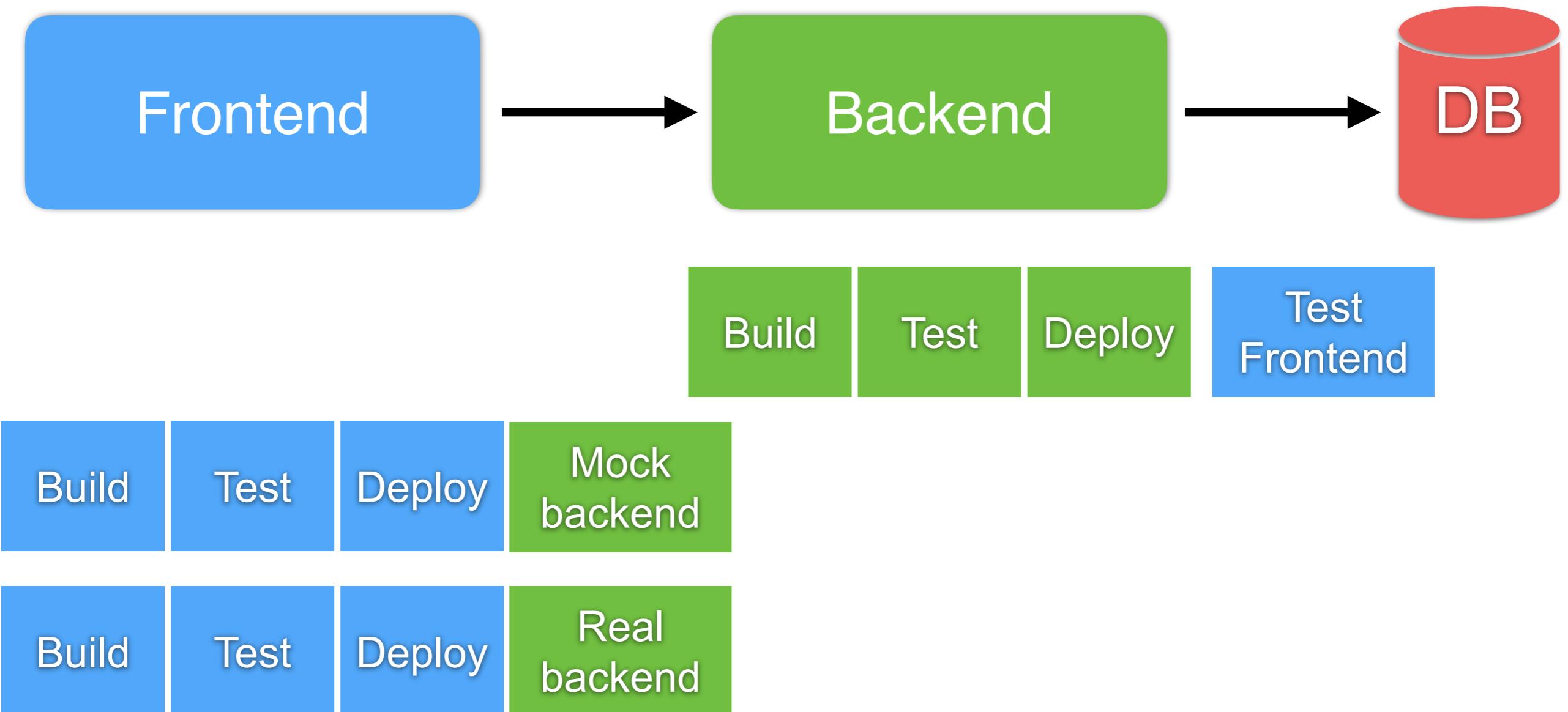
# Design your pipeline



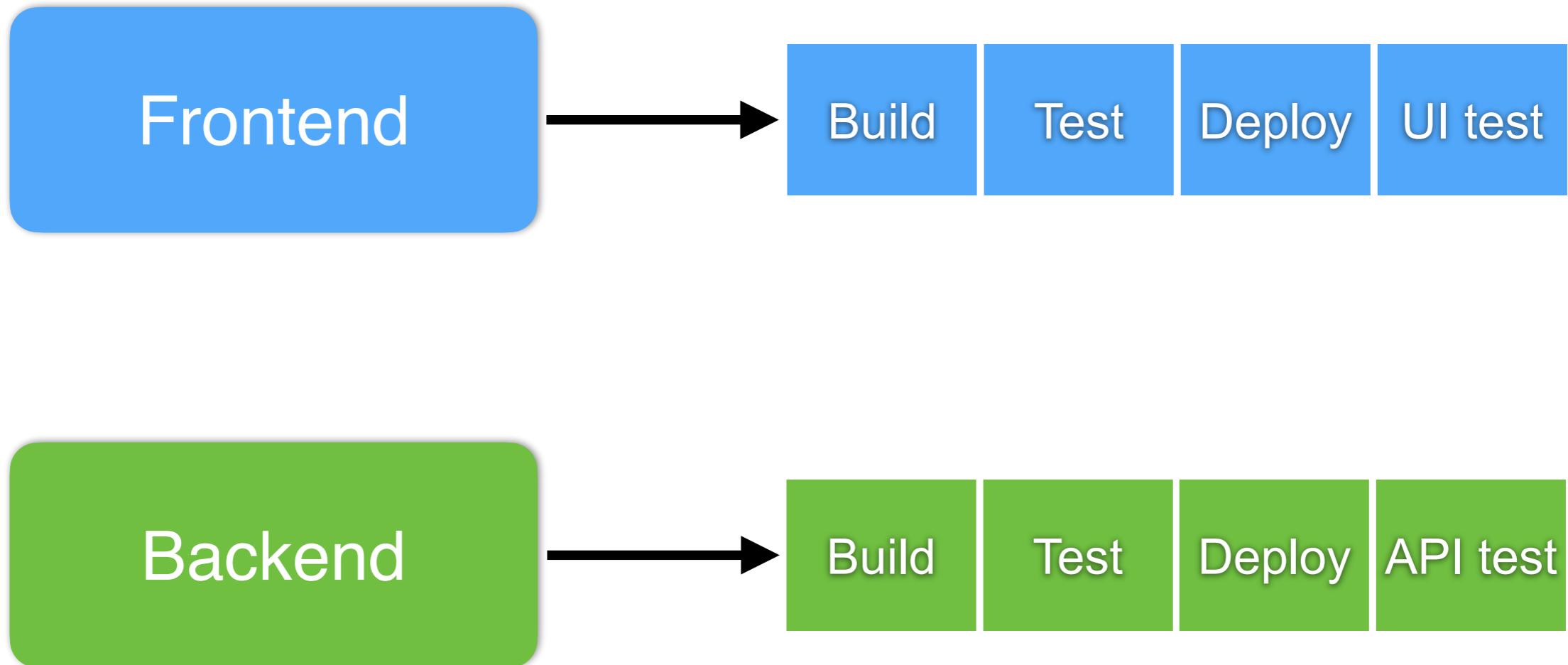
# Use cases



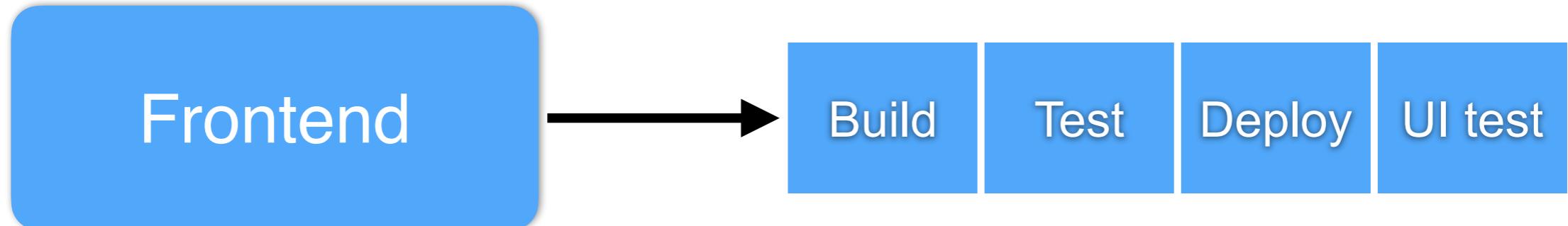
# Use cases



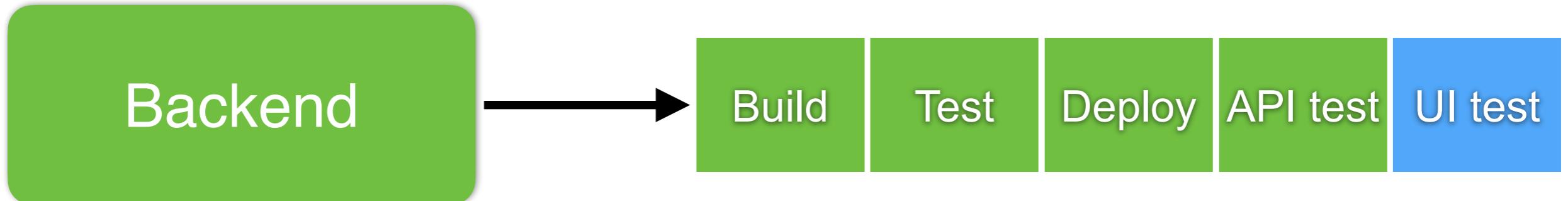
# Pipeline



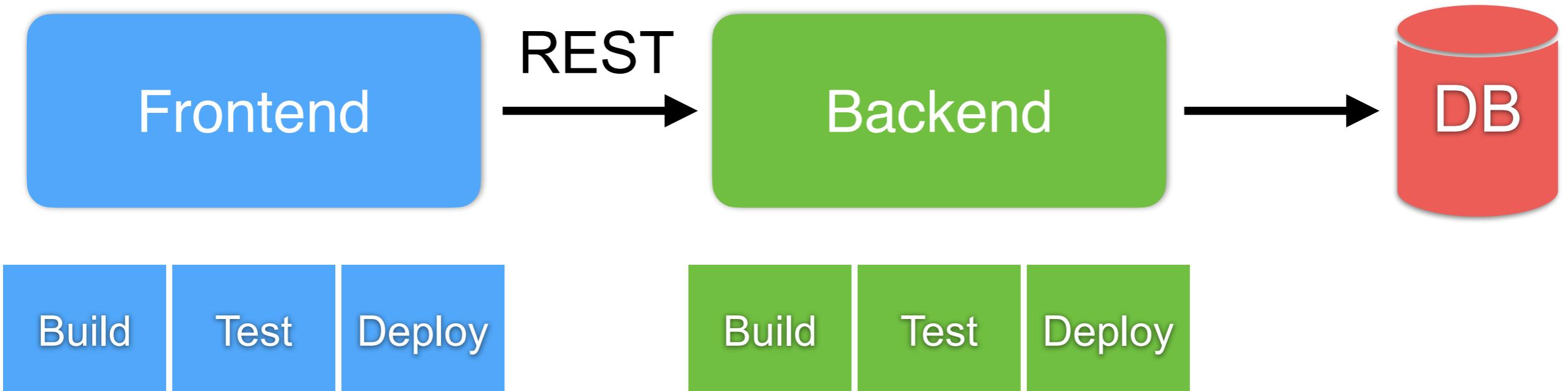
# When frontend is changed



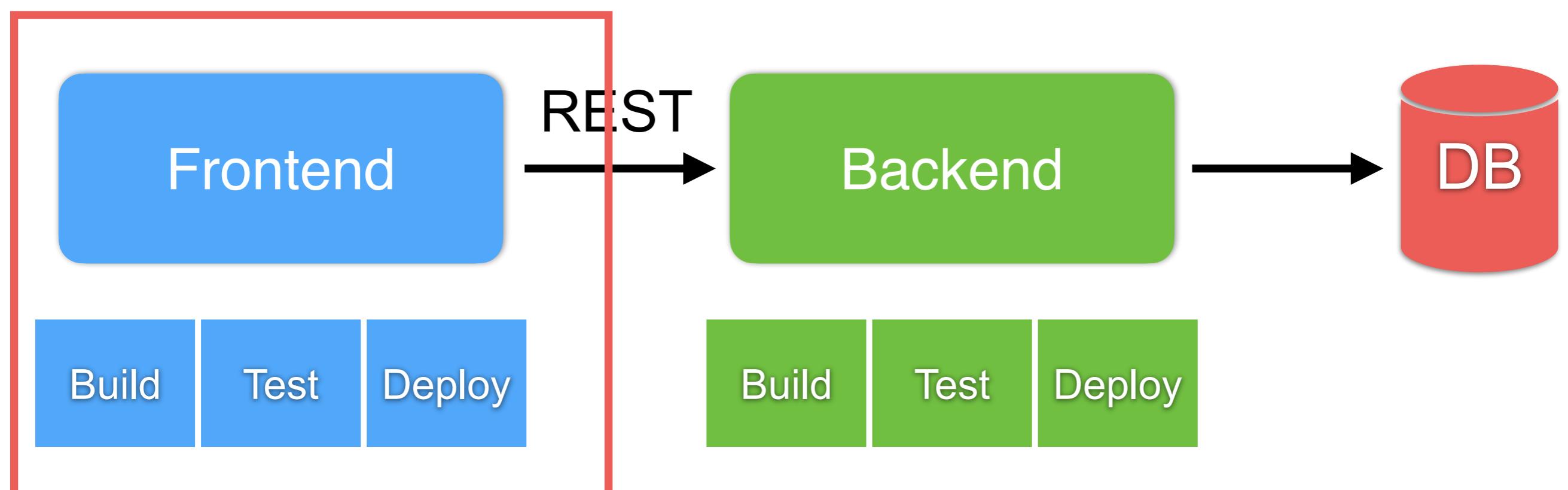
# When backend is changed



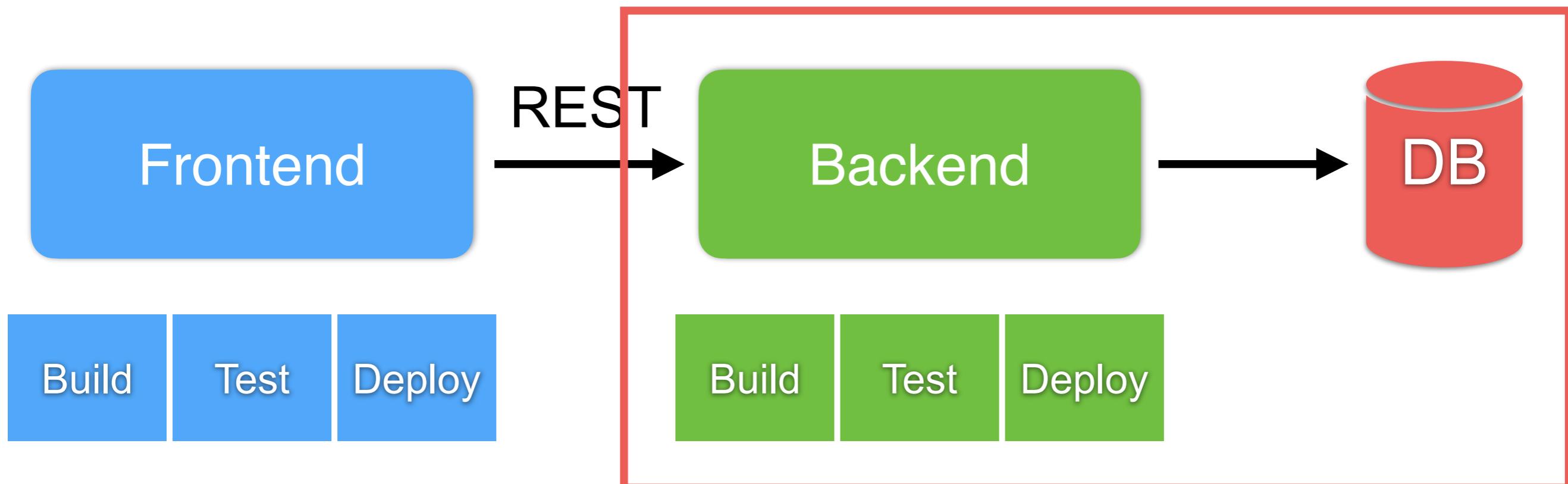
# Flow



# How to test frontend ?



# How to test backend ?



# Next ...

