



TDD series

Workshop :: TDD with Swift





Somkiat Puisungnoen

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 สยามชั่นนาคุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata

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 สยามชั่นนาคุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

Help people take action on this Page. ×

+ Add a Button

Home Posts Videos Photos

Liked Following Share ...



Day 1



Topics (1)

1. Pyramid testing for Mobile application
2. Design system with A-DAPT blueprint
3. Overview of Xcode tool
4. Good test structure and lifecycle



Topics (1)

5. Create Unit tests with XCTest
6. Create UI tests with XCTest
7. Good unit test with FIRST
8. Install fastland tool => scan, snapshot
9. Code coverage



Code coverage with Slather



SLATHER
Apply tests liberally

<https://github.com/SlatherOrg/slather>



Code coverage with Slather

```
$fastlane scan -s UI
```

```
$slather coverage -s --scheme UI  
IMC_Calculator.xcodeproj
```



Day 2



Topics

1. Overview application
2. Start to write tests
3. Code coverage
4. Review
5. Build your CI/CD system with Jenkins



**[https://github.com/up1/
workshop-starter-tdd-swift](https://github.com/up1/workshop-starter-tdd-swift)**



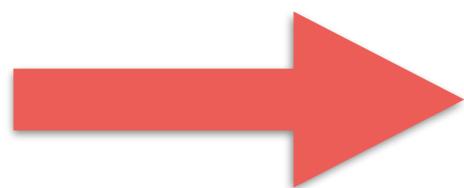
Overview application



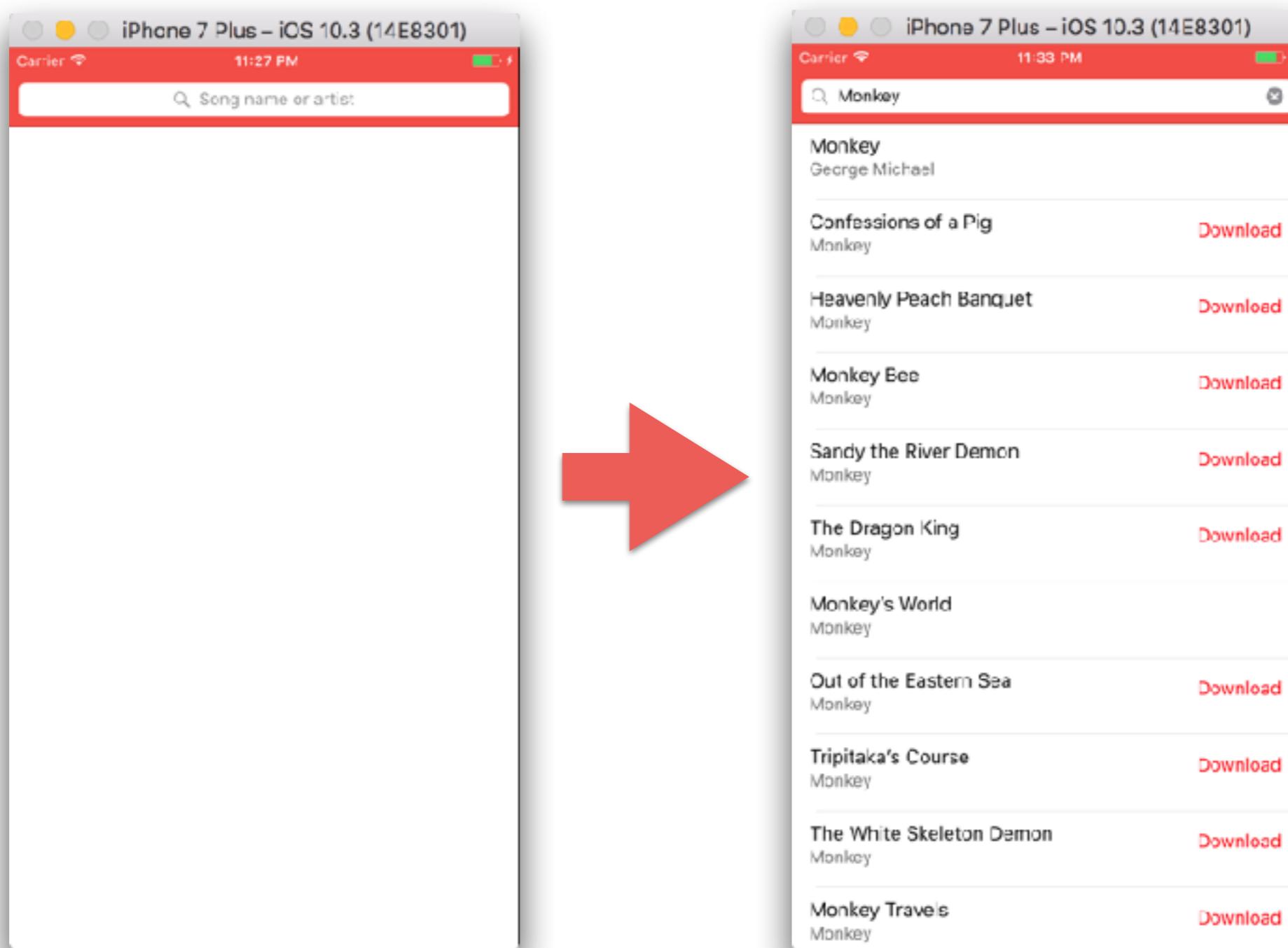
Architecture



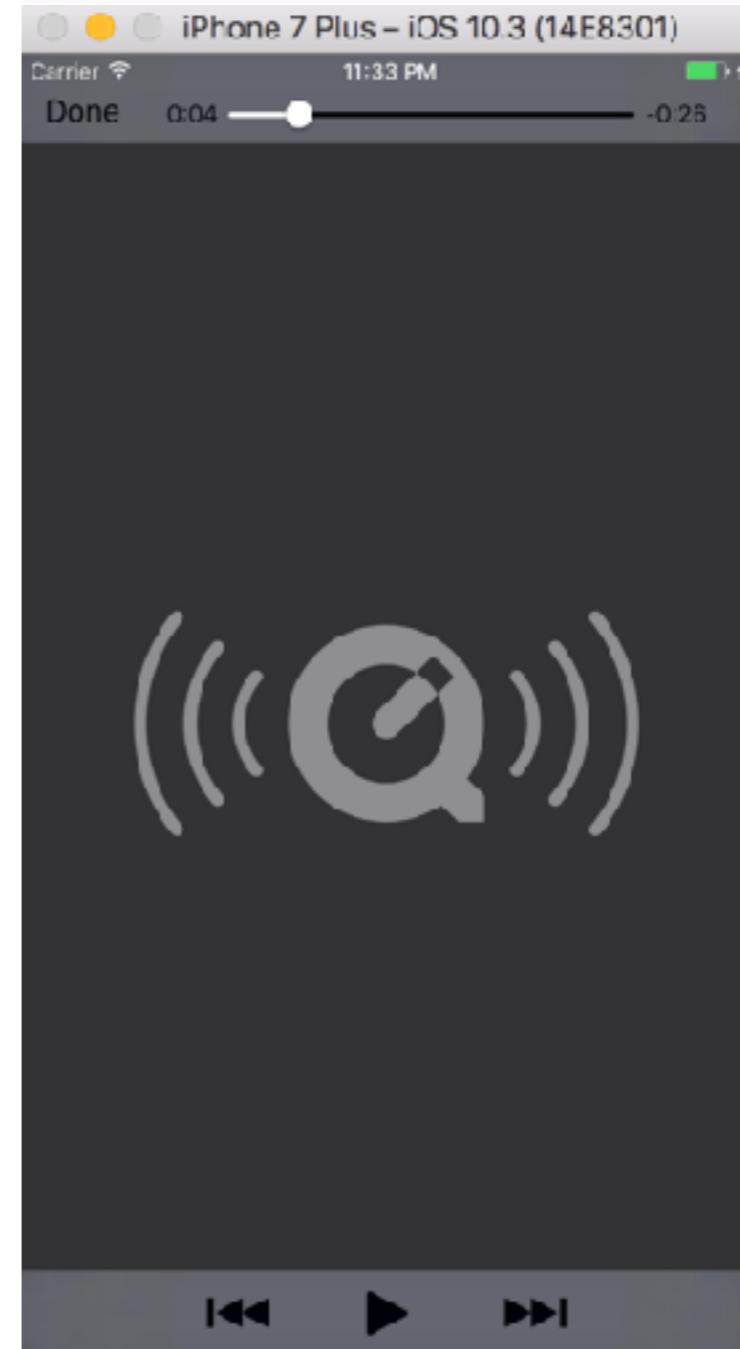
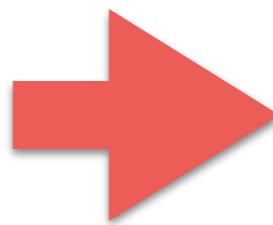
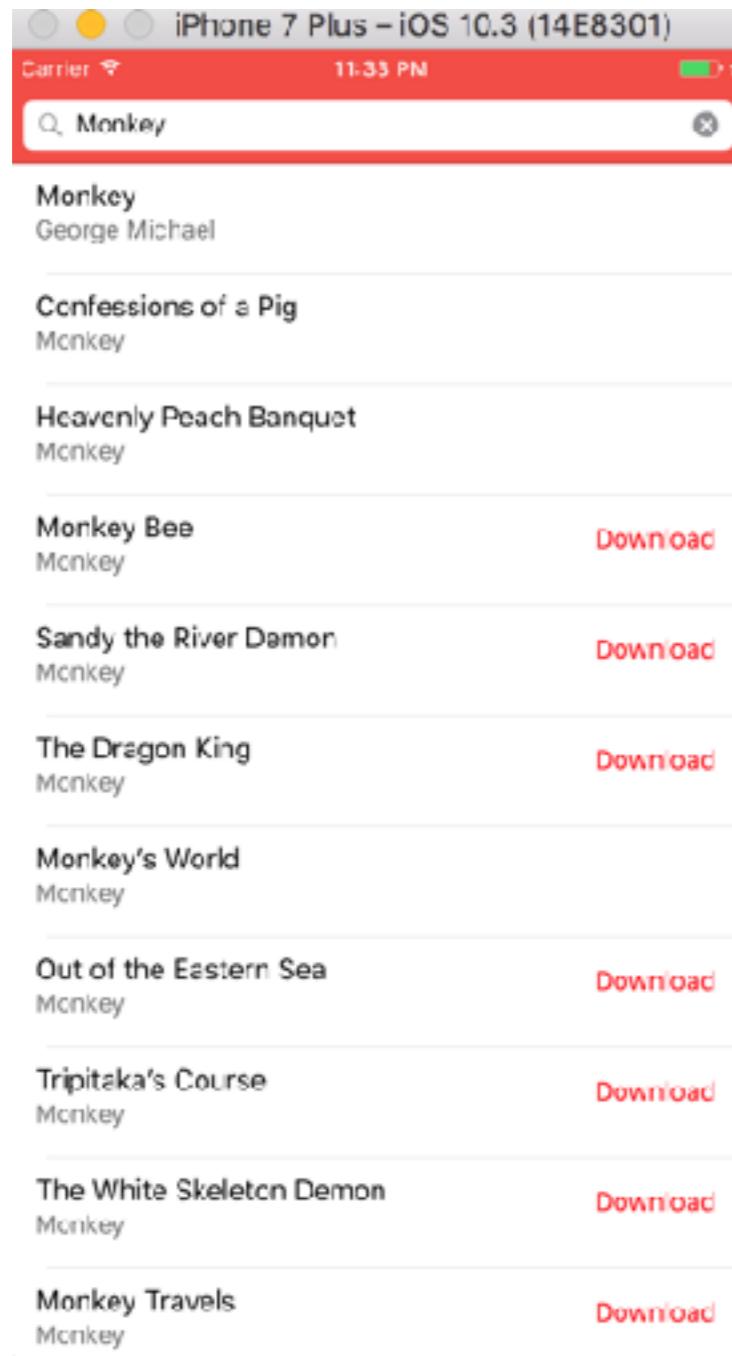
HTTP



Search songs from iTune (1)

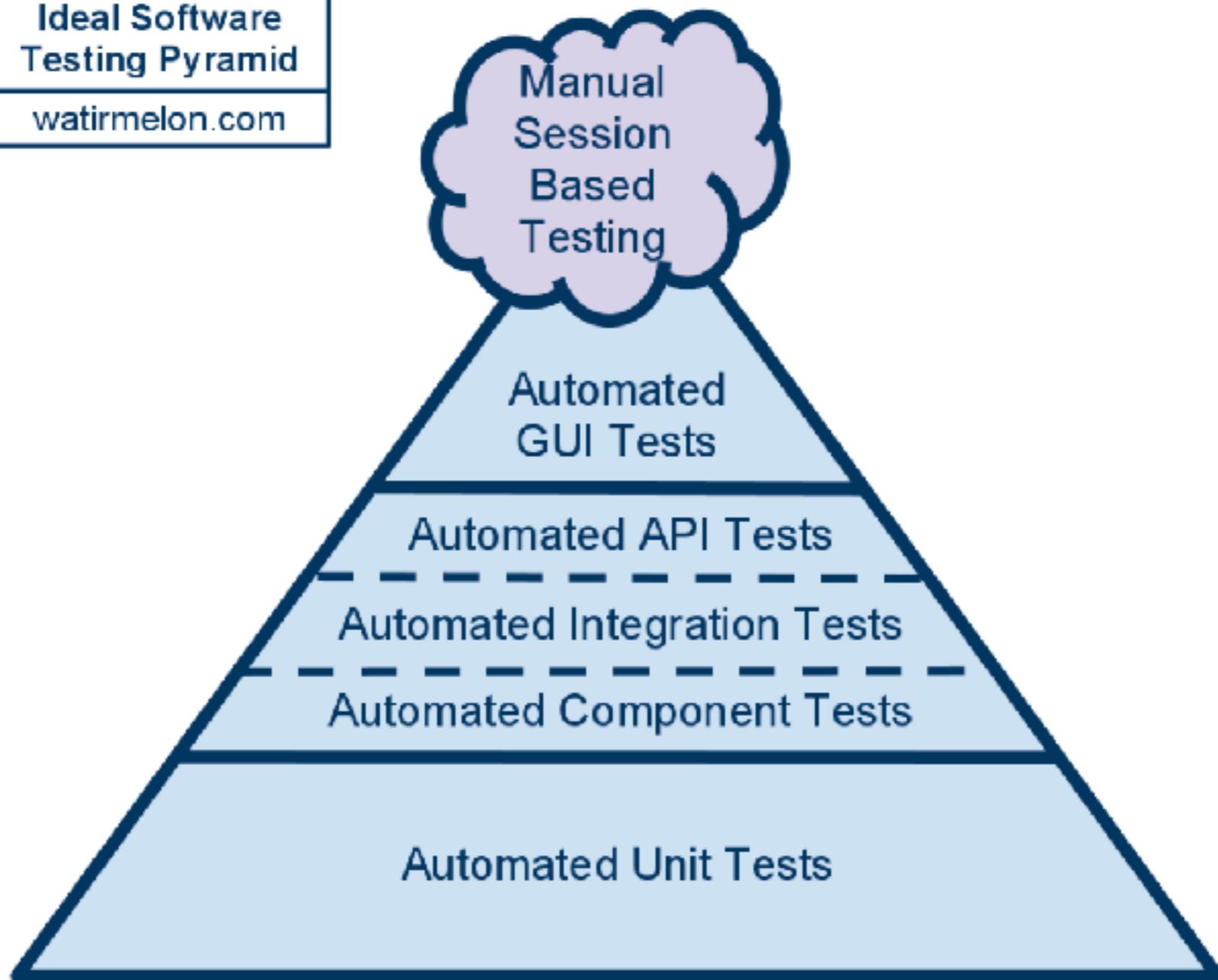


Search songs from iTune (1)

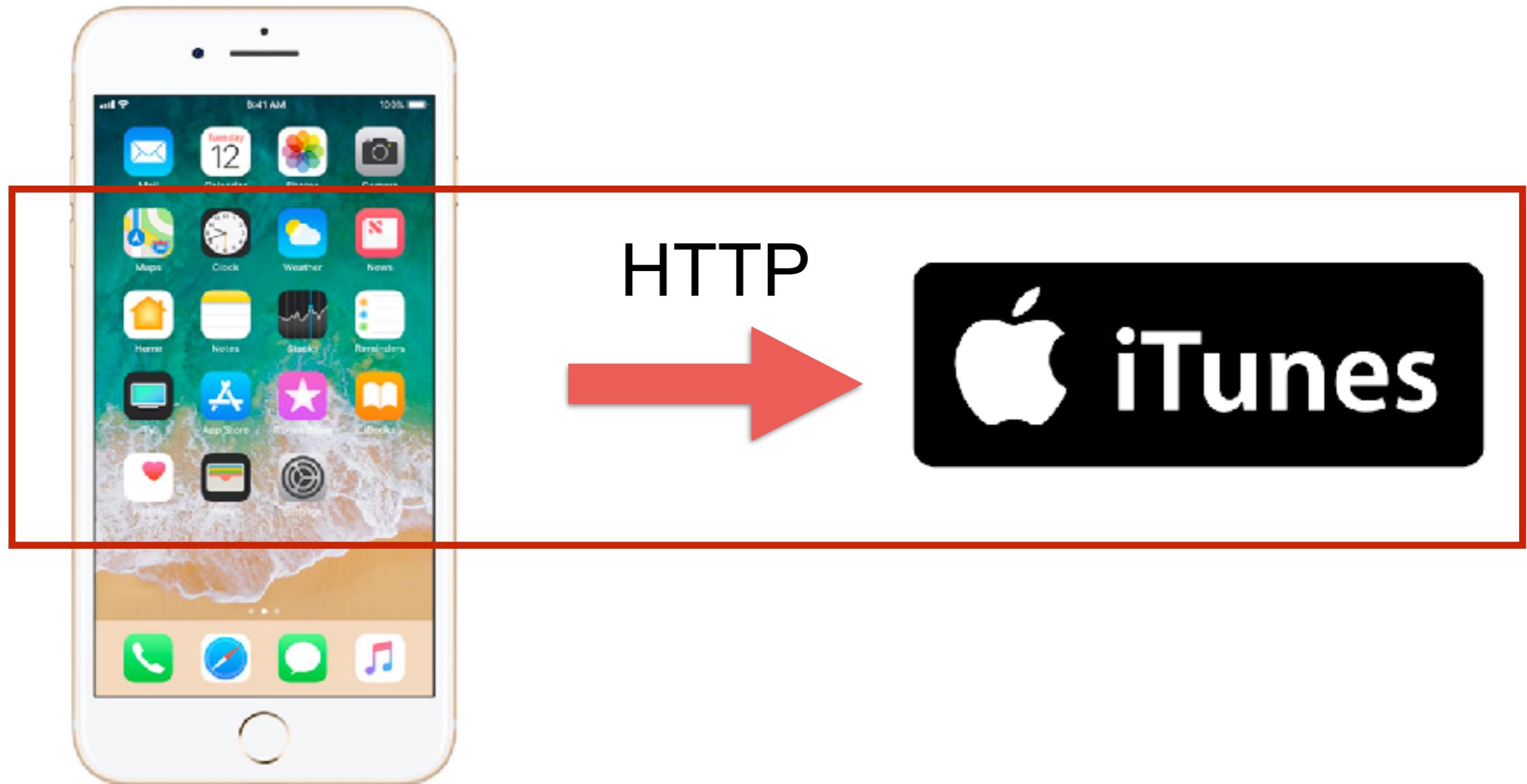


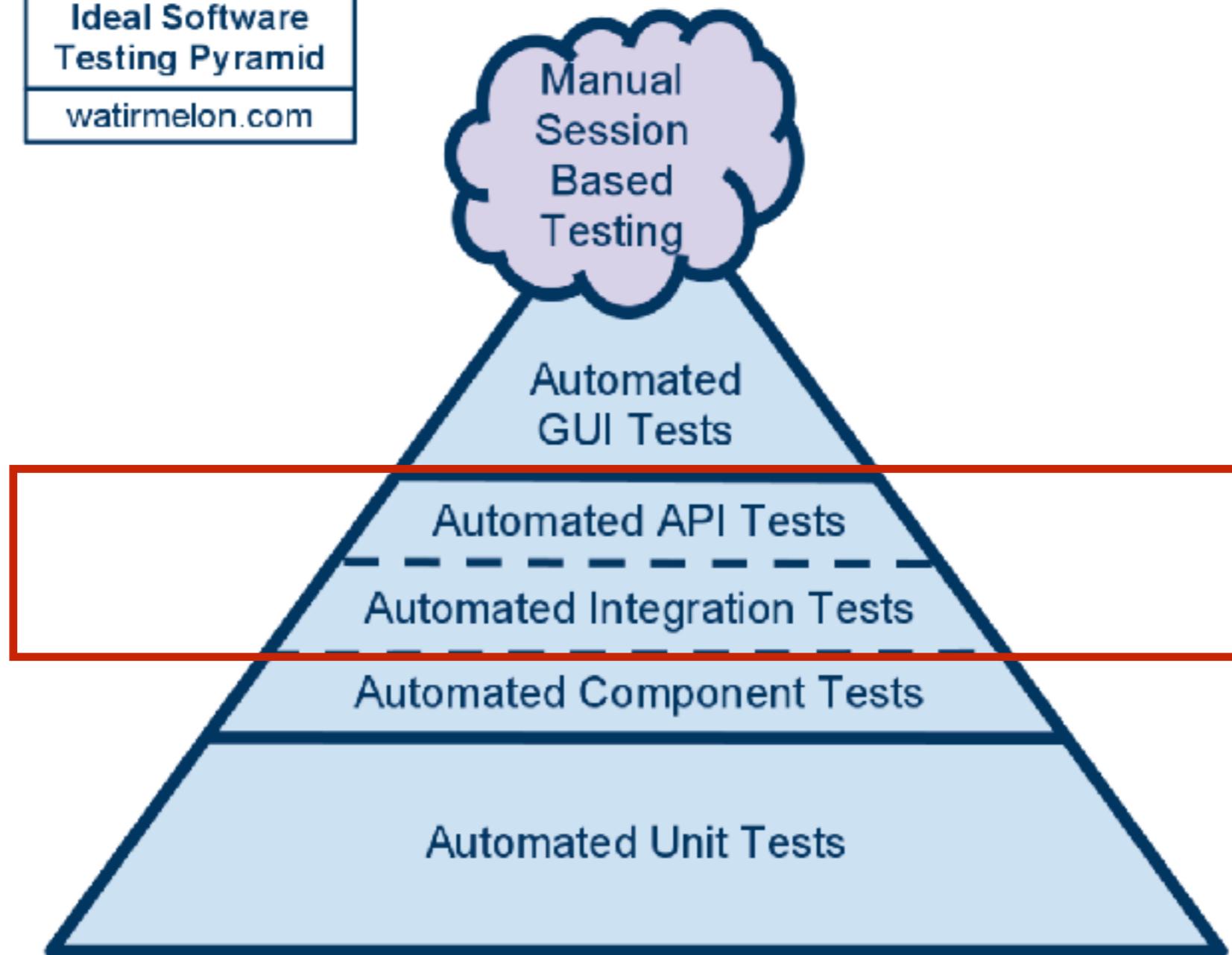
What to test ?





Call iTune API (Integration test)



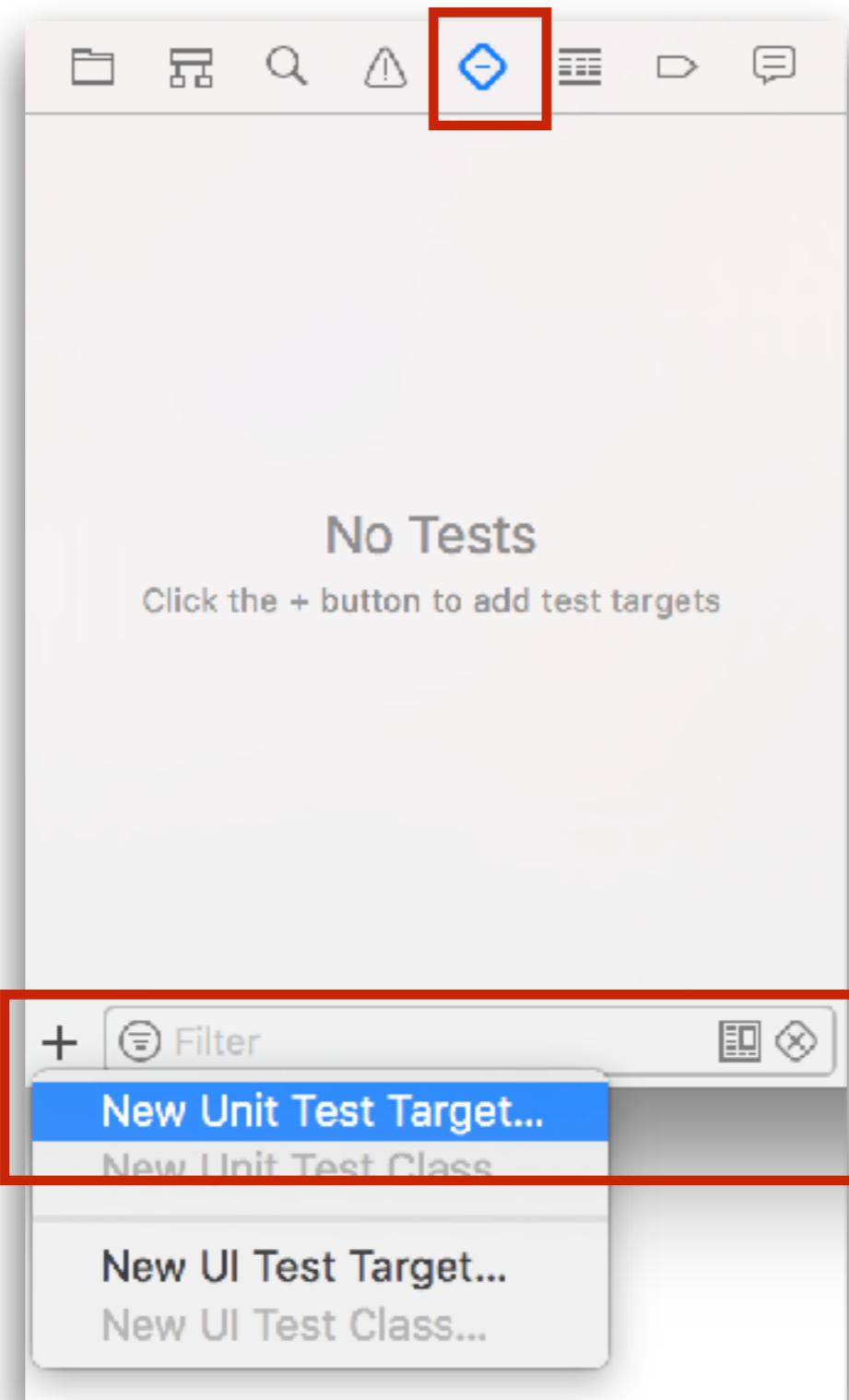


How to test ?



Add Unit tests target to project





First test :: Call real API



1. Create new test class

```
import XCTest  
  
@testable import HalTunes  
  
class HalTunesTests: XCTestCase {  
  
    override func setUp() {  
        super.setUp()  
    }  
  
    override func tearDown() {  
        super.tearDown()  
    }  
  
    func testExample() {  
    }  
  
}
```



2. Session to send request

```
class HalTunesTests: XCTestCase {
```

```
    var sessionUnderTest: URLSession!
```

```
    override func setUp() {
        super.setUp()
        sessionUnderTest = URLSession(configuration:
            URLSessionConfiguration.default)
    }
```

```
    override func tearDown() {
        sessionUnderTest = nil
        super.tearDown()
    }
```

```
}
```



3. Start to write a test (1)

Arrange to set URL and expectation result

```
func testSuccess_Call_To_iTune_API_Should_Retuen_Status_code_200() {  
  
    // Arrange  
    let url = URL(string: "https://itunes.apple.com/search?  
media=music&entity=song&term=monkey")  
  
    // What i expect to happen  
    let promise = expectation(description: "Status code: 200")  
  
}
```



3. Start to write a test (2)

```
// Act
let dataTask = sessionUnderTest.dataTask(with: url!) { data, response, error in
    // Assert
    if let error = error {
        XCTFail("Error: \(error.localizedDescription)")
        return
    } else if let statusCode = (response as? HTTPURLResponse)?.statusCode {
        if statusCode == 200 {
            // Success with condition
            promise.fulfill()
        } else {
            XCTFail("Status code: \(statusCode)")
        }
    }
}
dataTask.resume()

// Test running until all expectations are fulfilled
waitForExpectations(timeout: 5, handler: nil)
```



Second test :: Call real API



Fail Faster ?



Fail Faster

```
let promise = expectation(description: "Completion handler invoked")
var statusCode: Int?
var responseError: Error?

// Act
let dataTask = sessionUnderTest.dataTask(with: url!) {
    data, response, error in

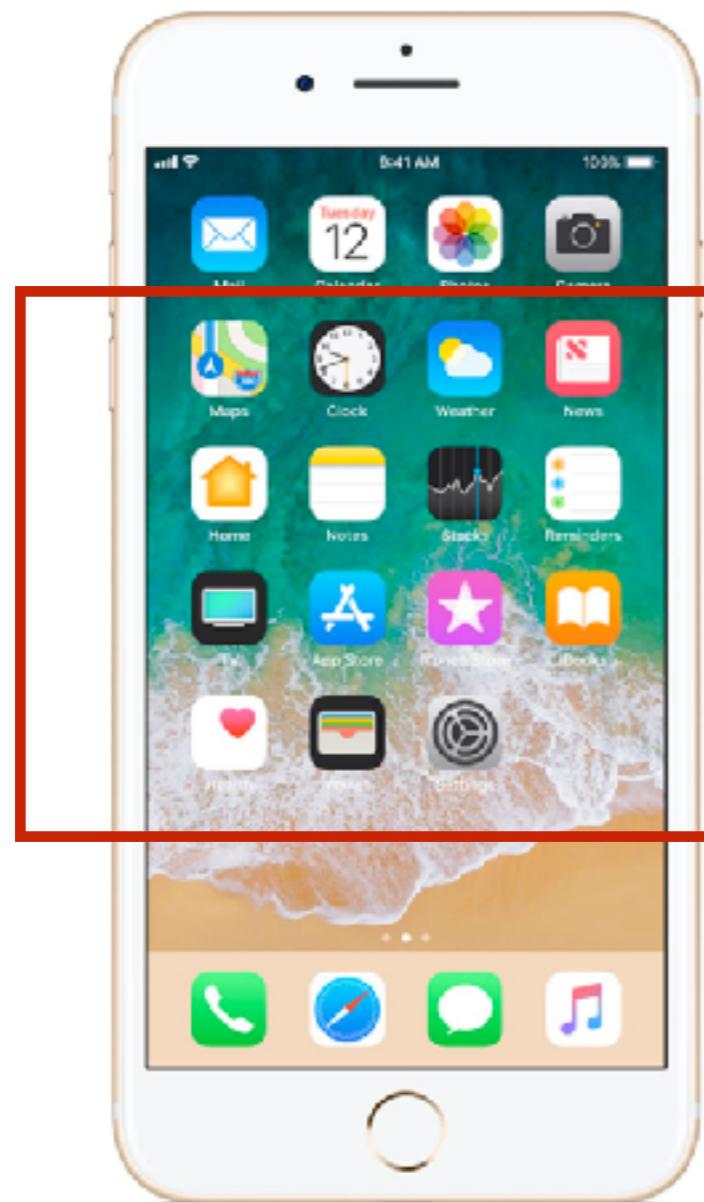
    statusCode = (response as? HTTPURLResponse)?.statusCode
    responseError = error
    promise.fulfill()
}
dataTask.resume()

waitForExpectations(timeout: 5, handler: nil)

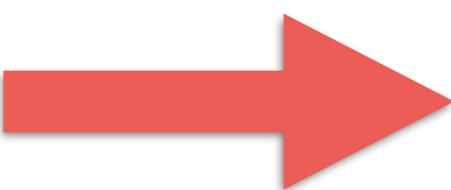
// Assert
XCTAssertNil(responseError)
XCTAssertEqual(statusCode, 200)
```



Call iTune API (Integration test)

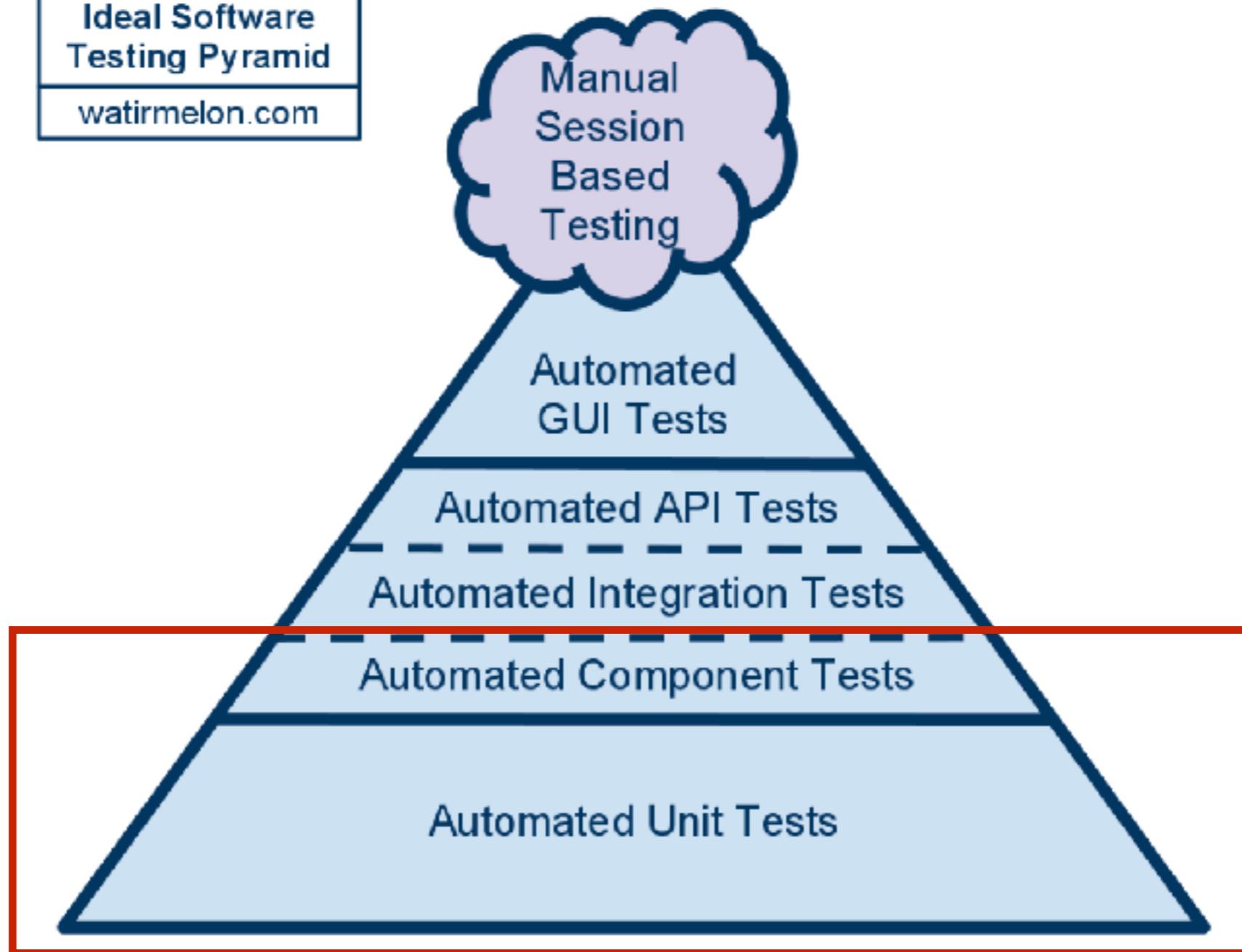


HTTP



What next to test ?





Unit test ?



Continuous Integration



Why CI/CD ?

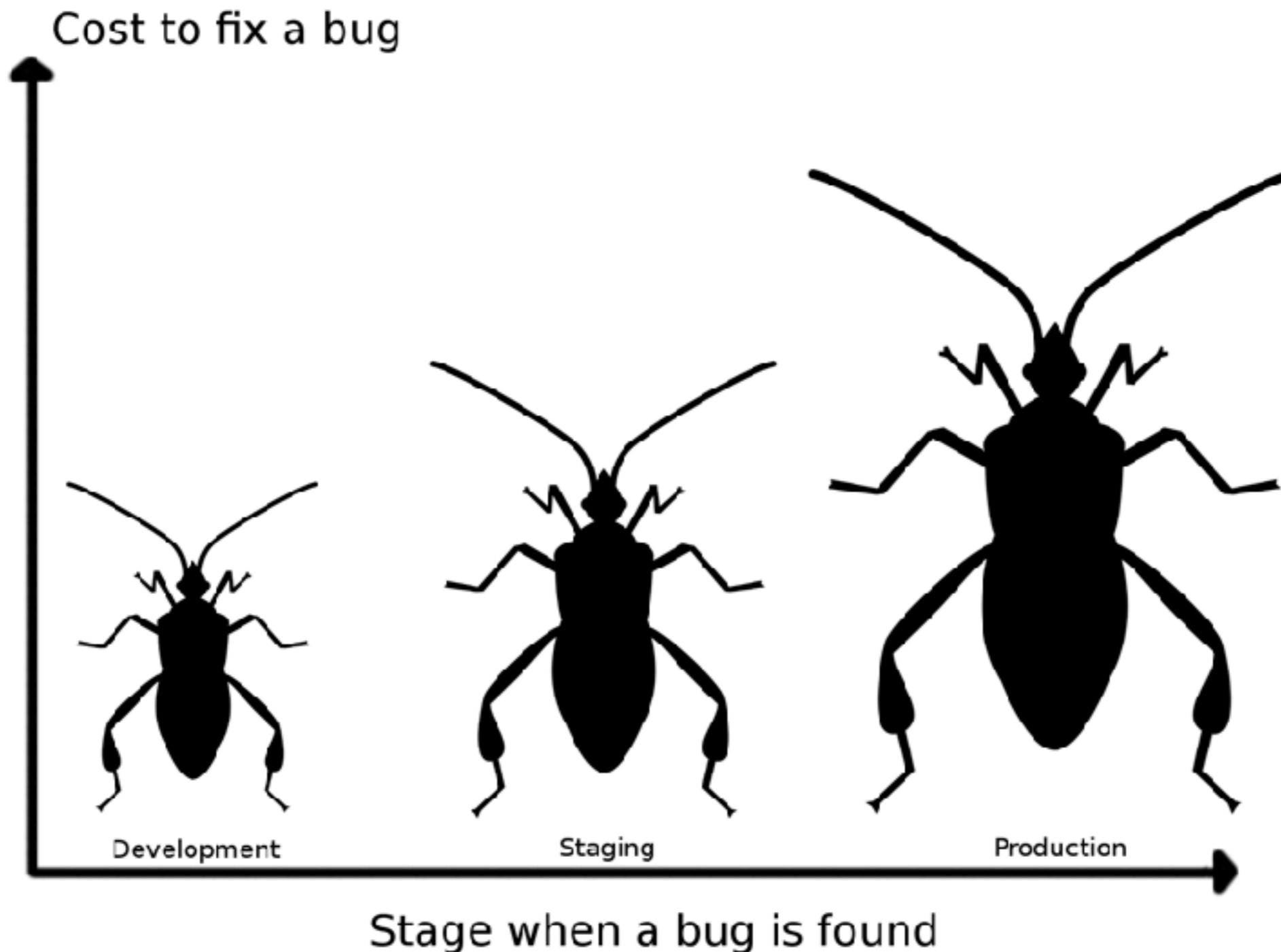


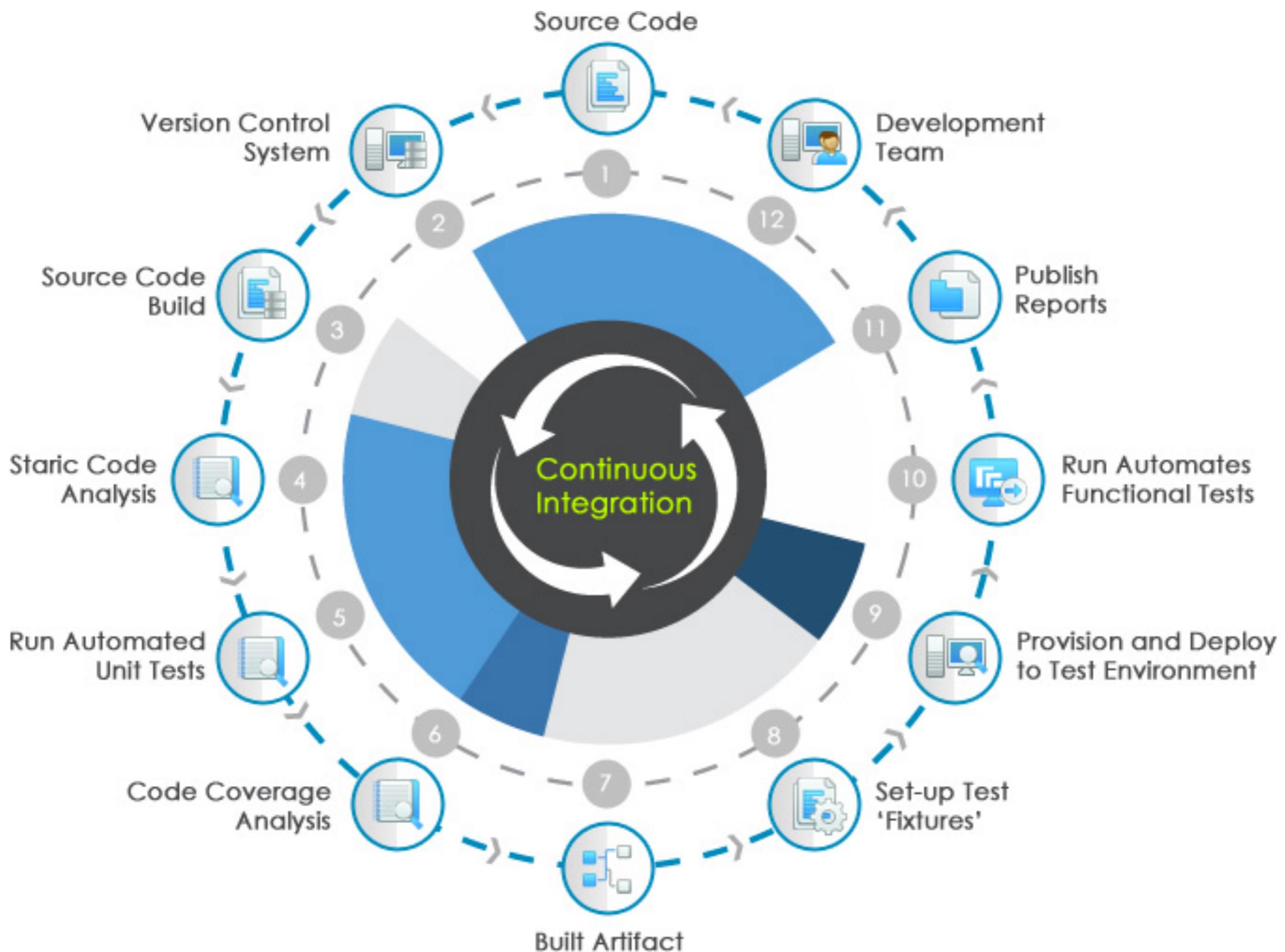
The cost of integration

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

> goTM



Hudson

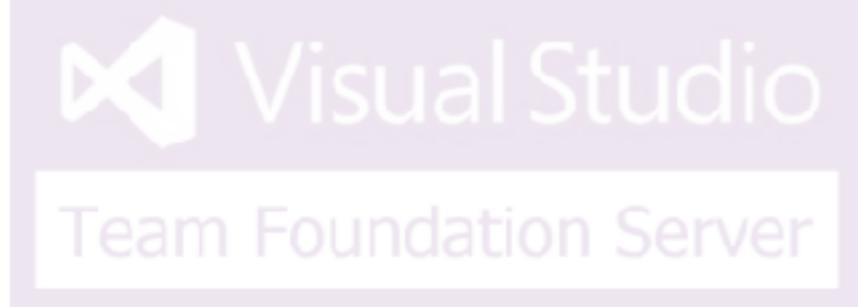




Jenkins

Bamboo

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



Hudson



Continuous Integration

Discipline to integrate frequently



Continuous Integration

Strive to make **small change**



Continuous Integration

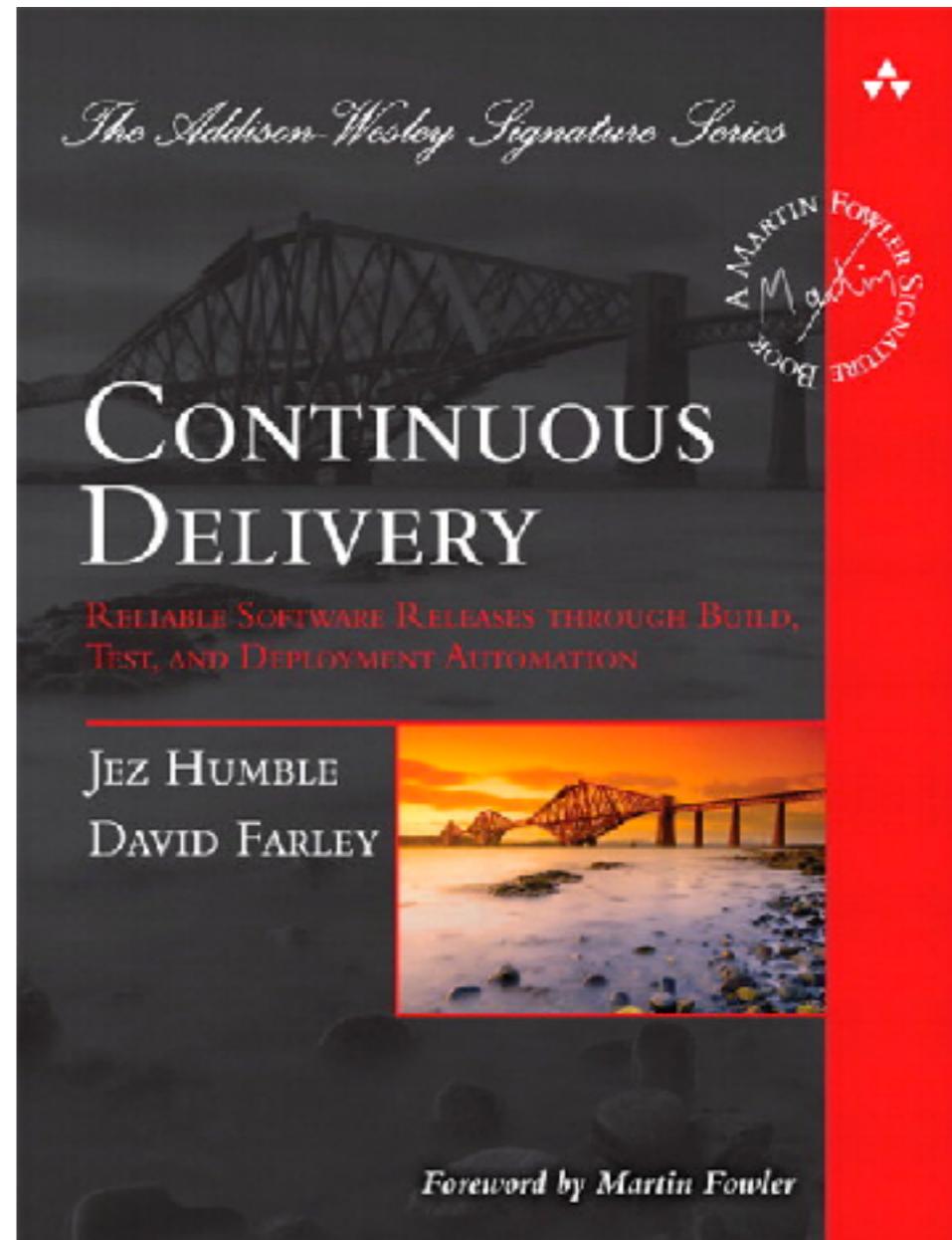
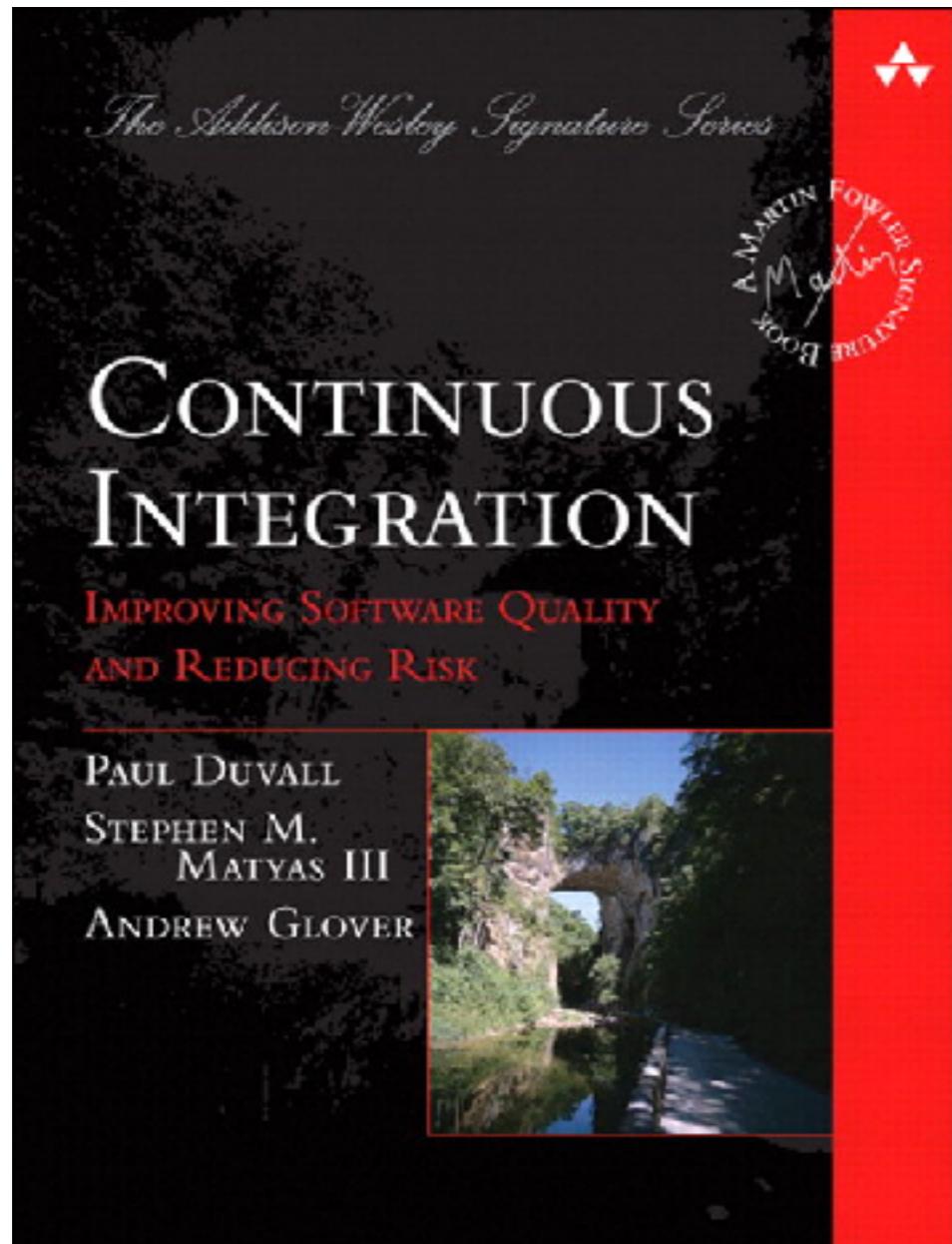
Strive for **fast feedback**



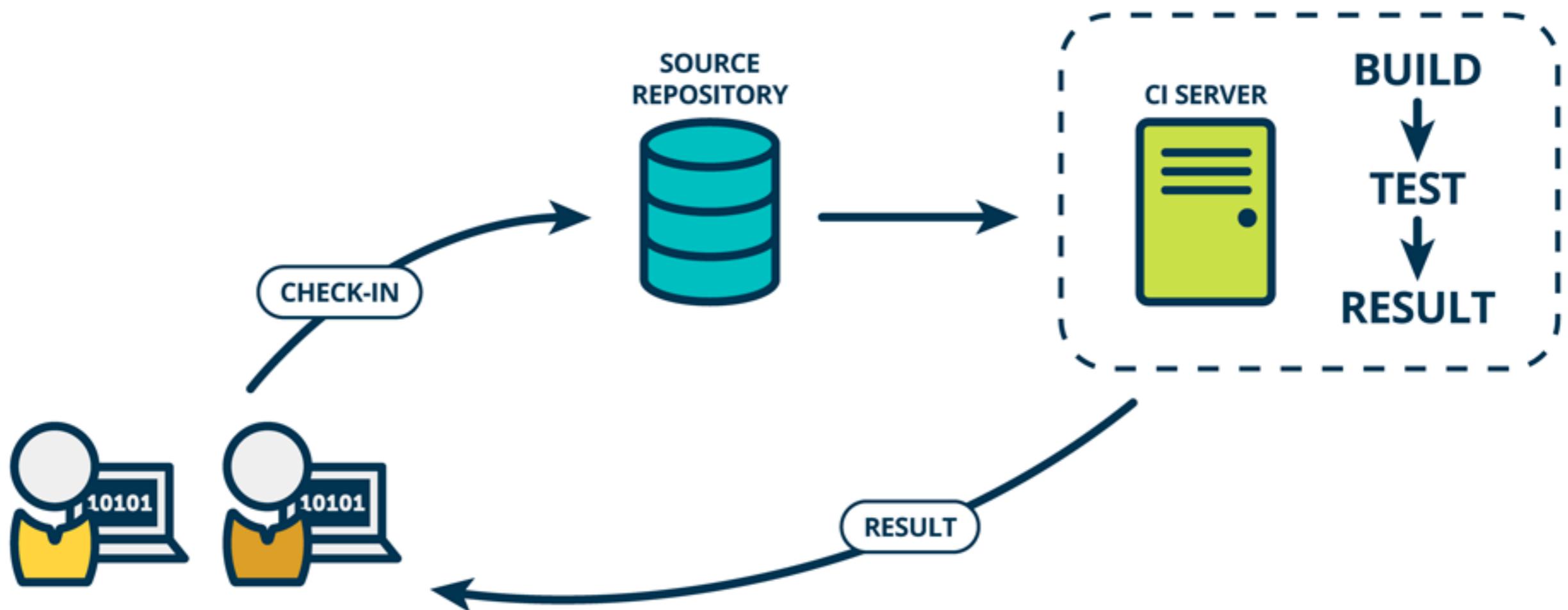
Practices of Continuous Integration



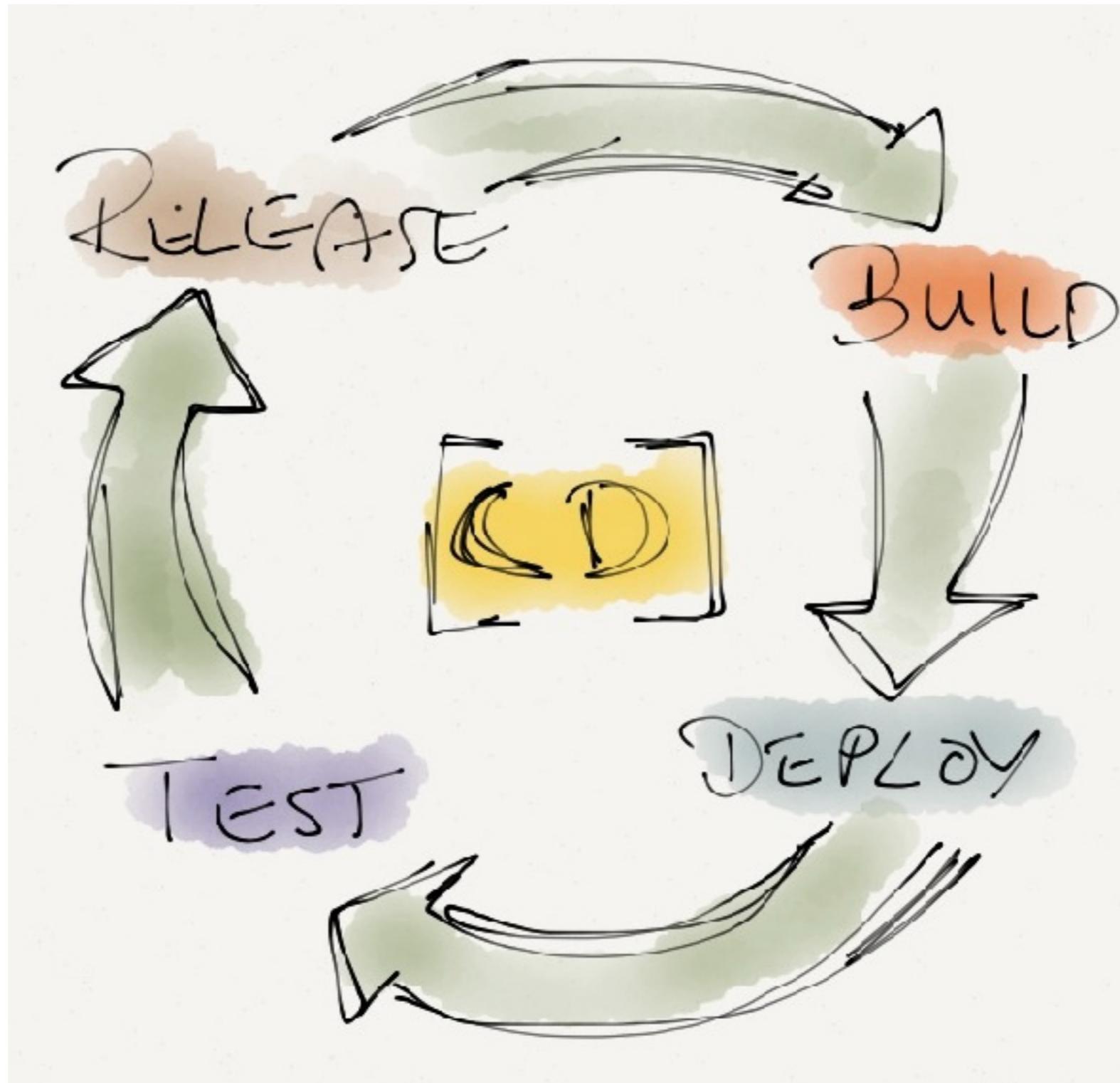
Improve quality and reduce risk



Continuous Integration



CD ?



CD ?

CONTINUOUS DELIVERY



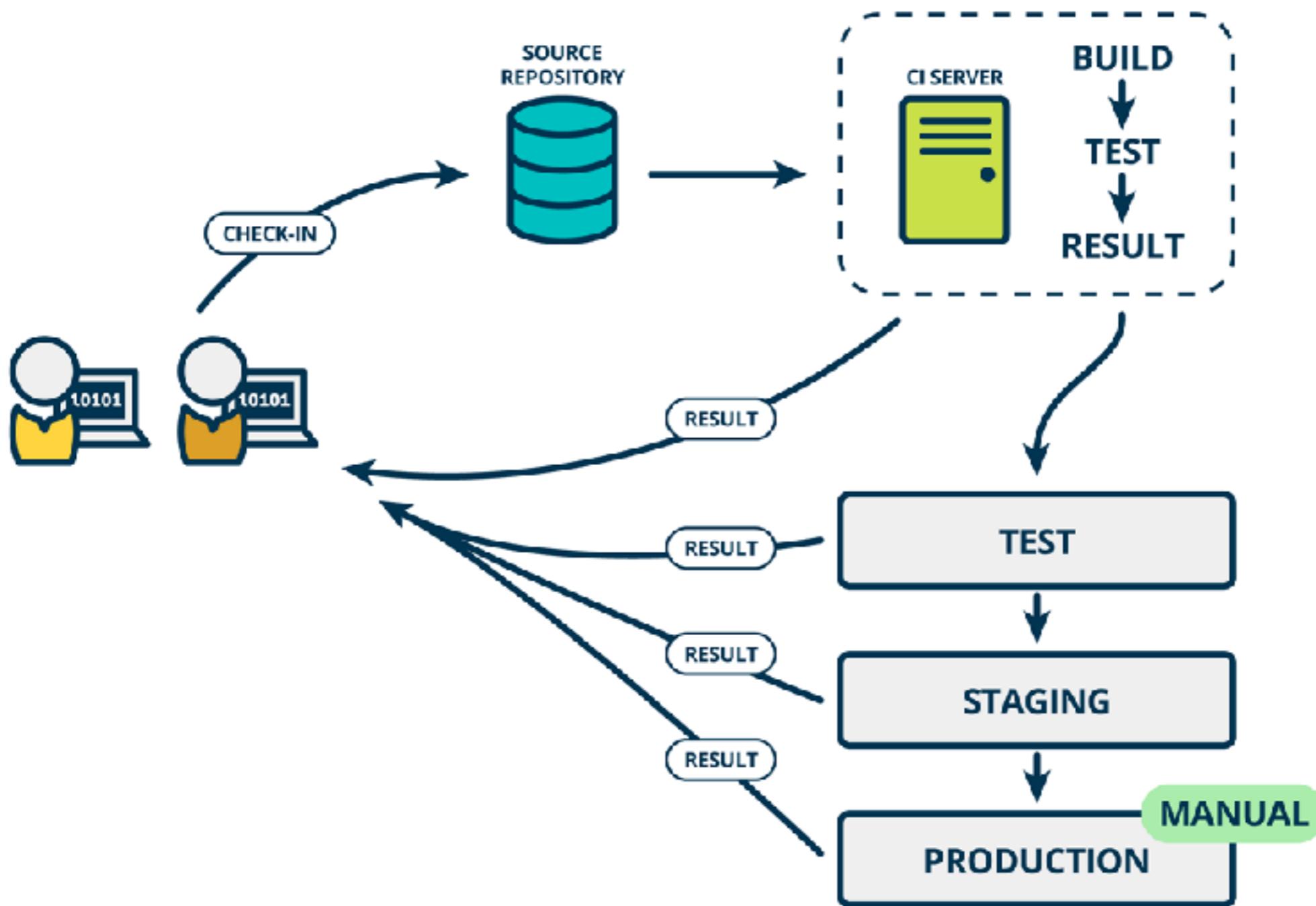
CONTINUOUS DEPLOYMENT



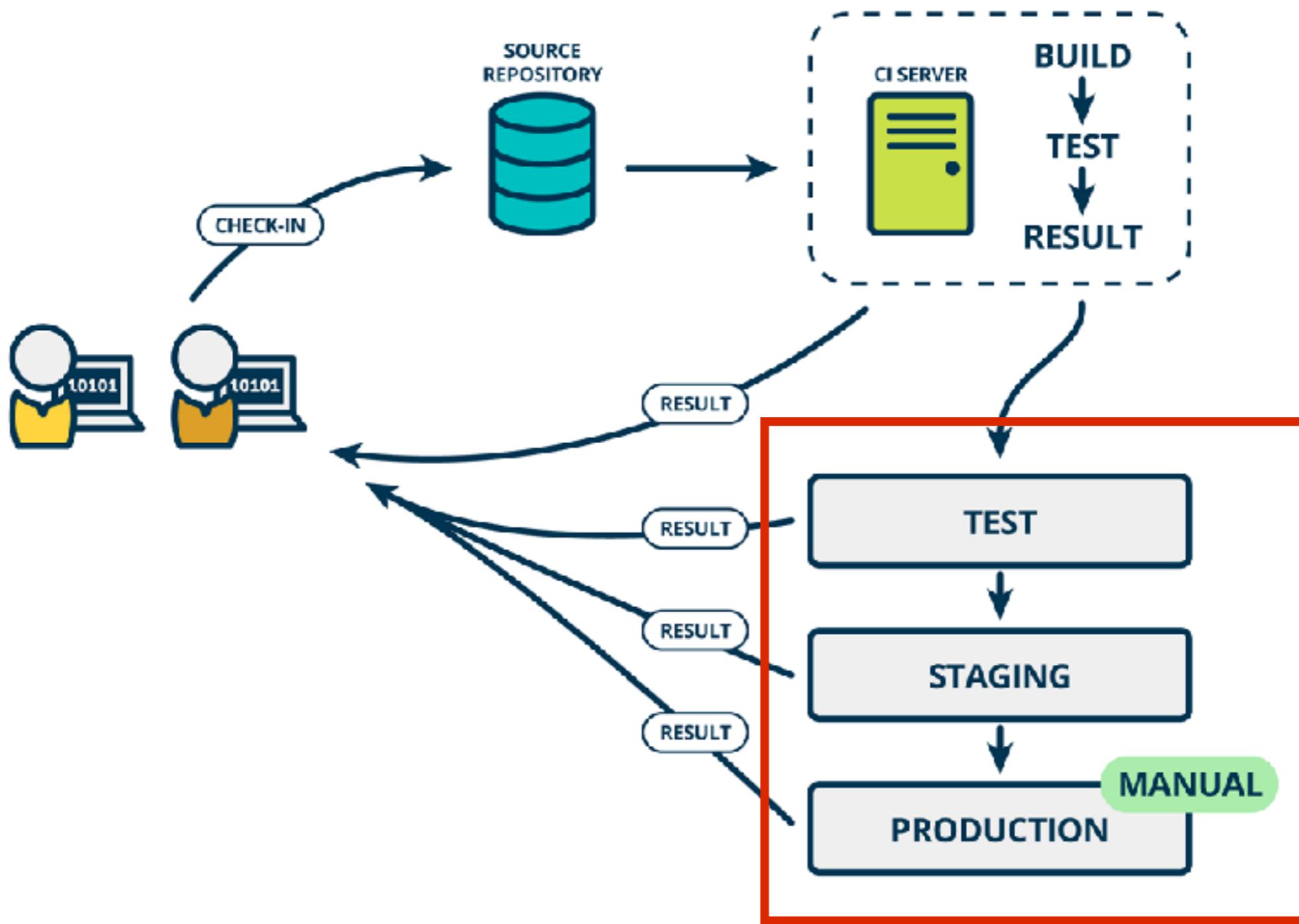
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



Continuous Delivery



Rise of DevOps



Continuous Integration

is a Software development practices



Practice 1

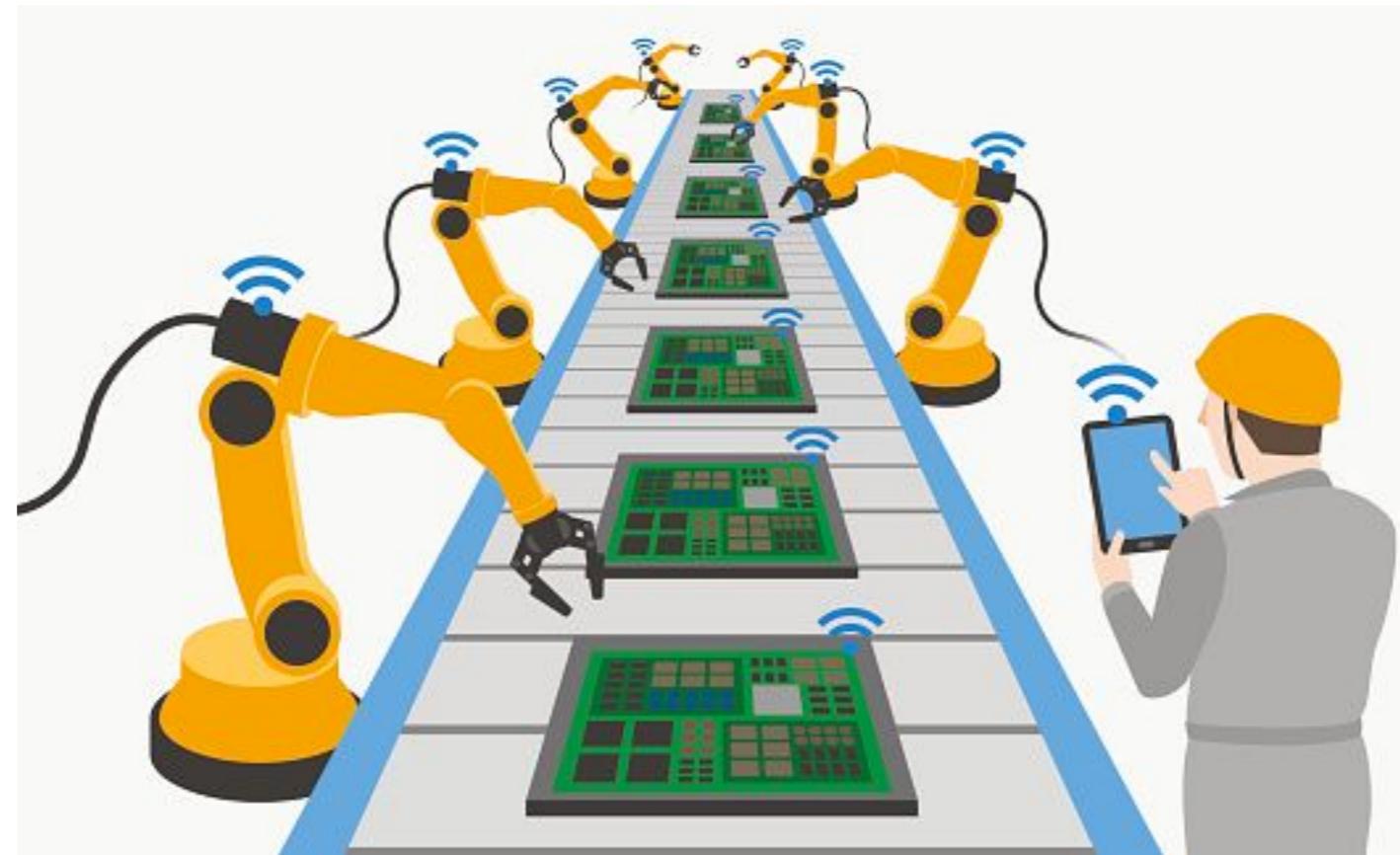
Maintain a single source repository

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



Practice 2

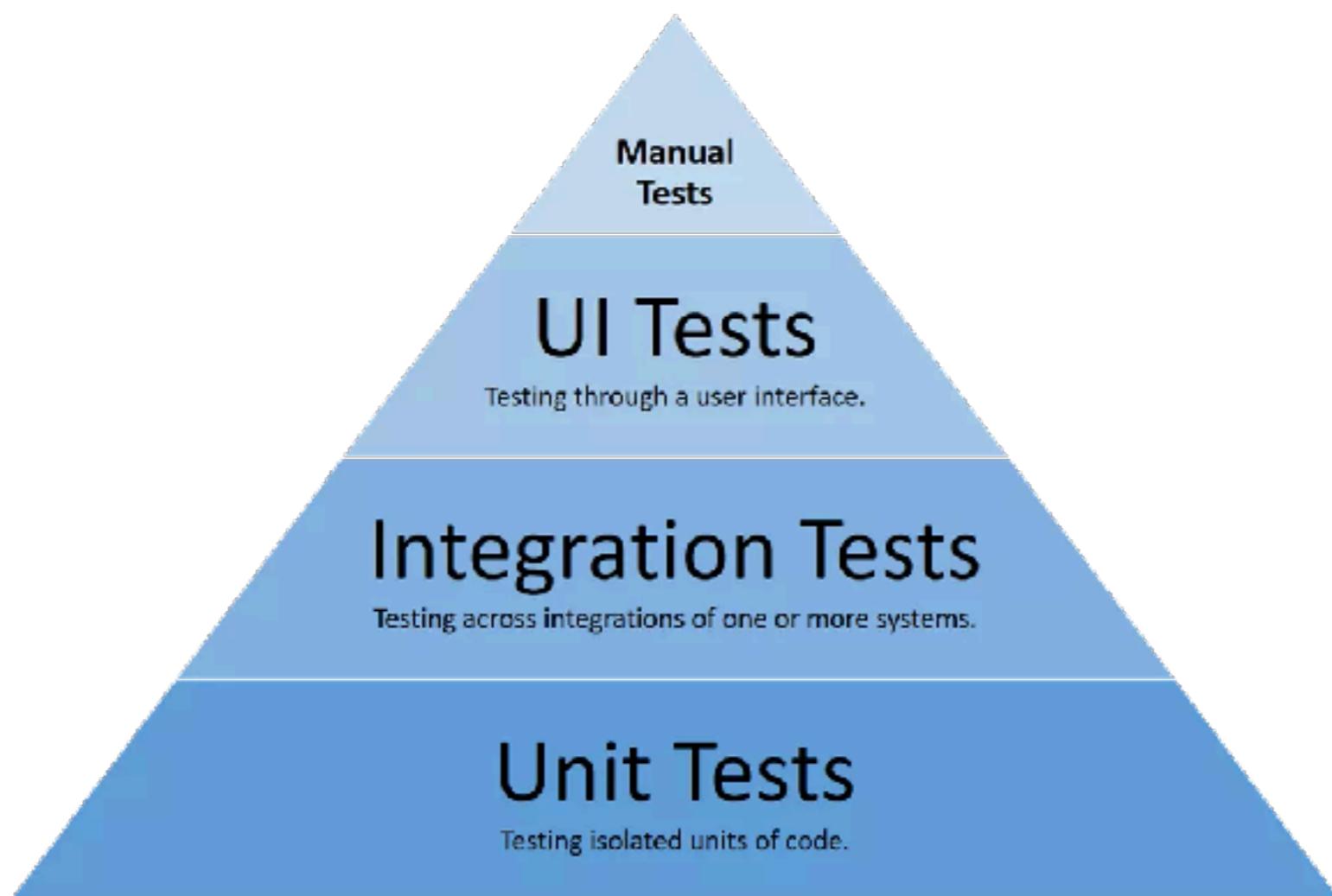
Automated the build
Automated environment for builds



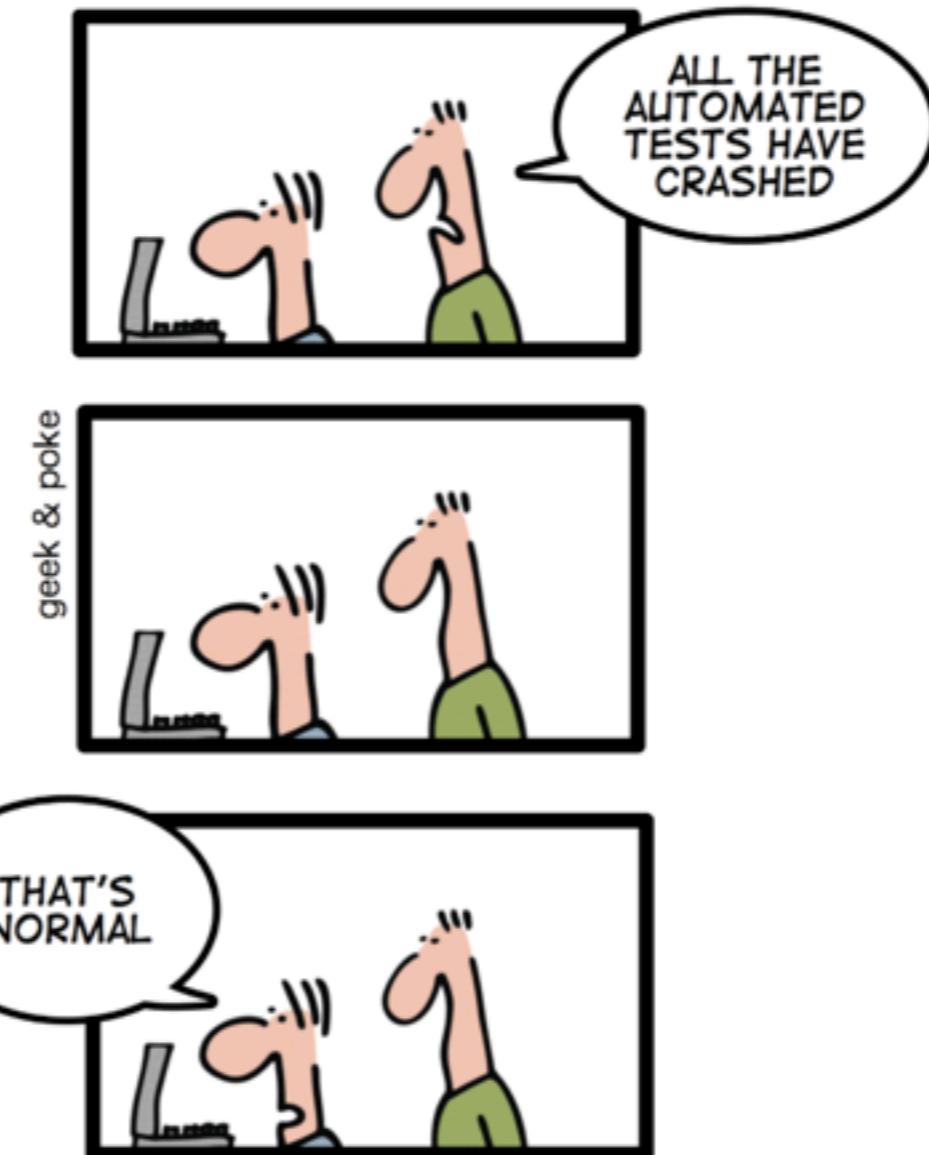
Practice 3

Make your build **self-testing**

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



*TODAY: CONTINUOUS INTEGRATION
GIVES YOU THE COMFORTING
FEELING TO KNOW THAT
EVERYTHING IS NORMAL*



<http://geekandpoke.typepad.com/>

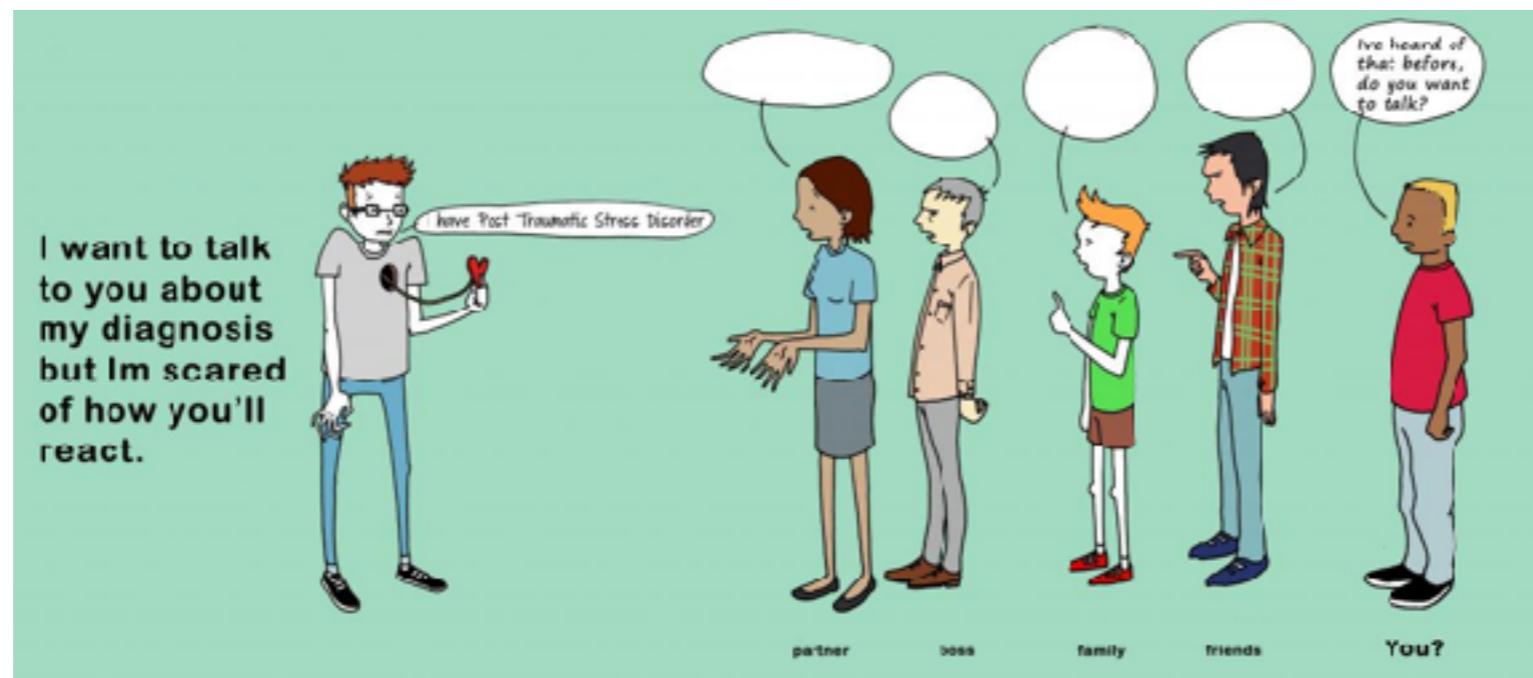


Practice 4

Everyone **commits** to the mainline everyday

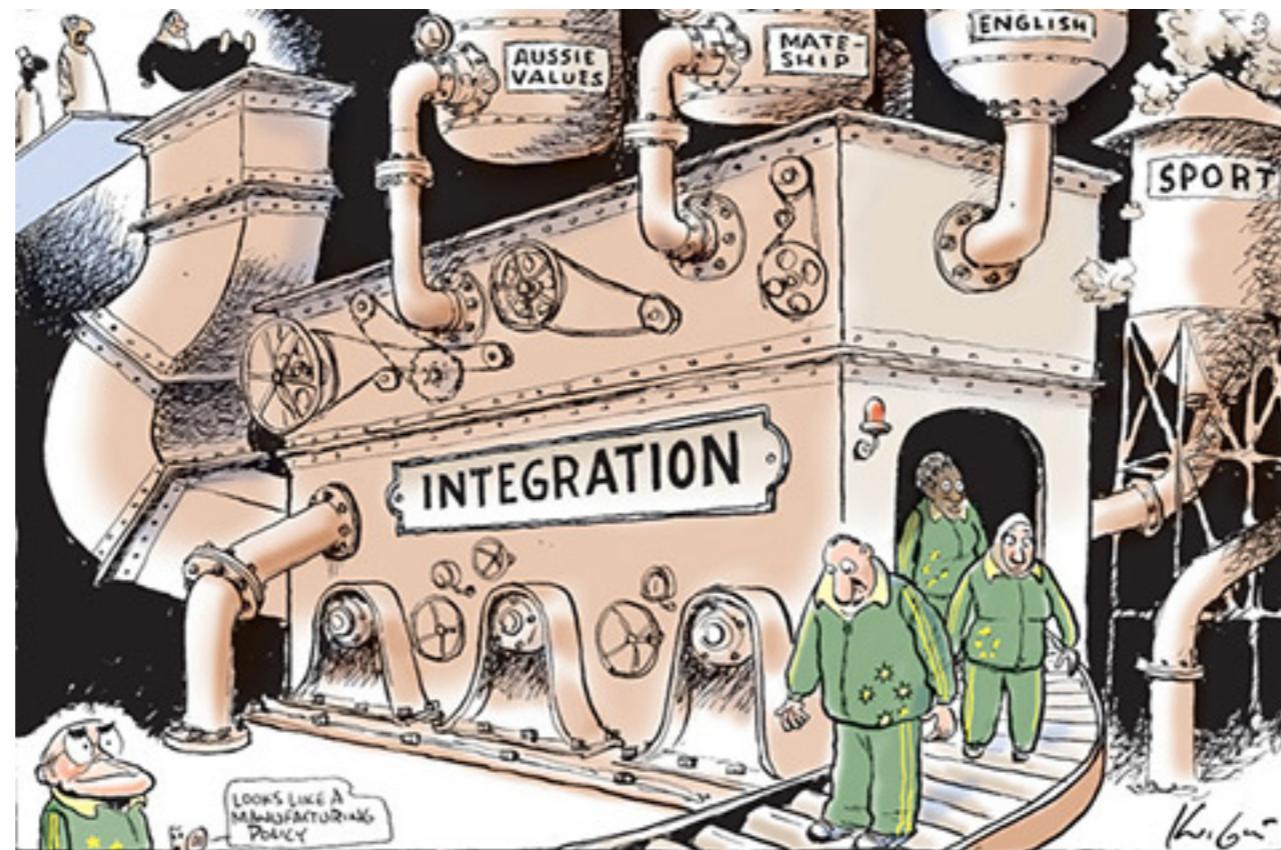
Integration is about **communication**

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

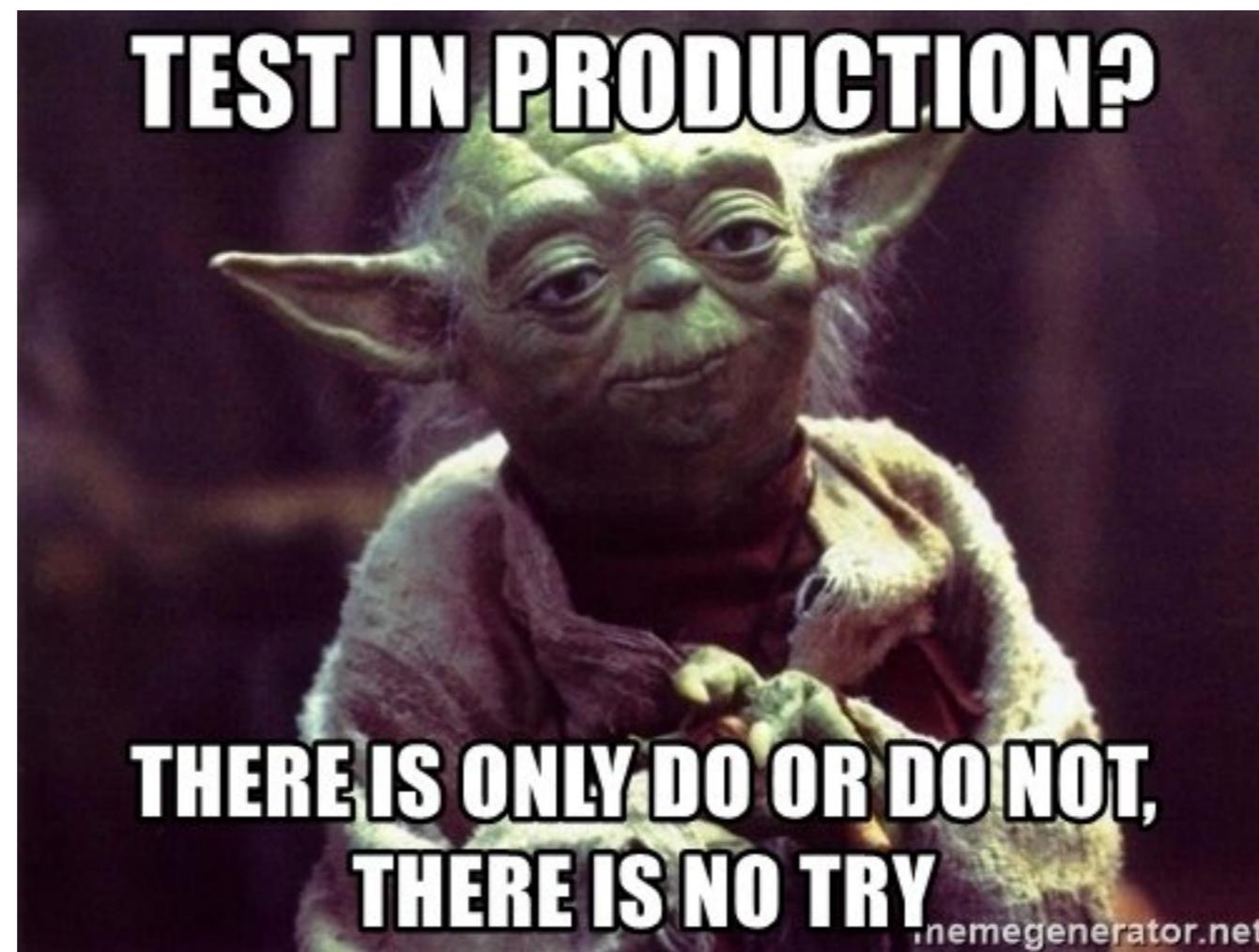
Keep the build **fast**

Continuous Integration is to provide rapid feedback



Practice 8

Test in clone of the **Production** environment



Practice 9

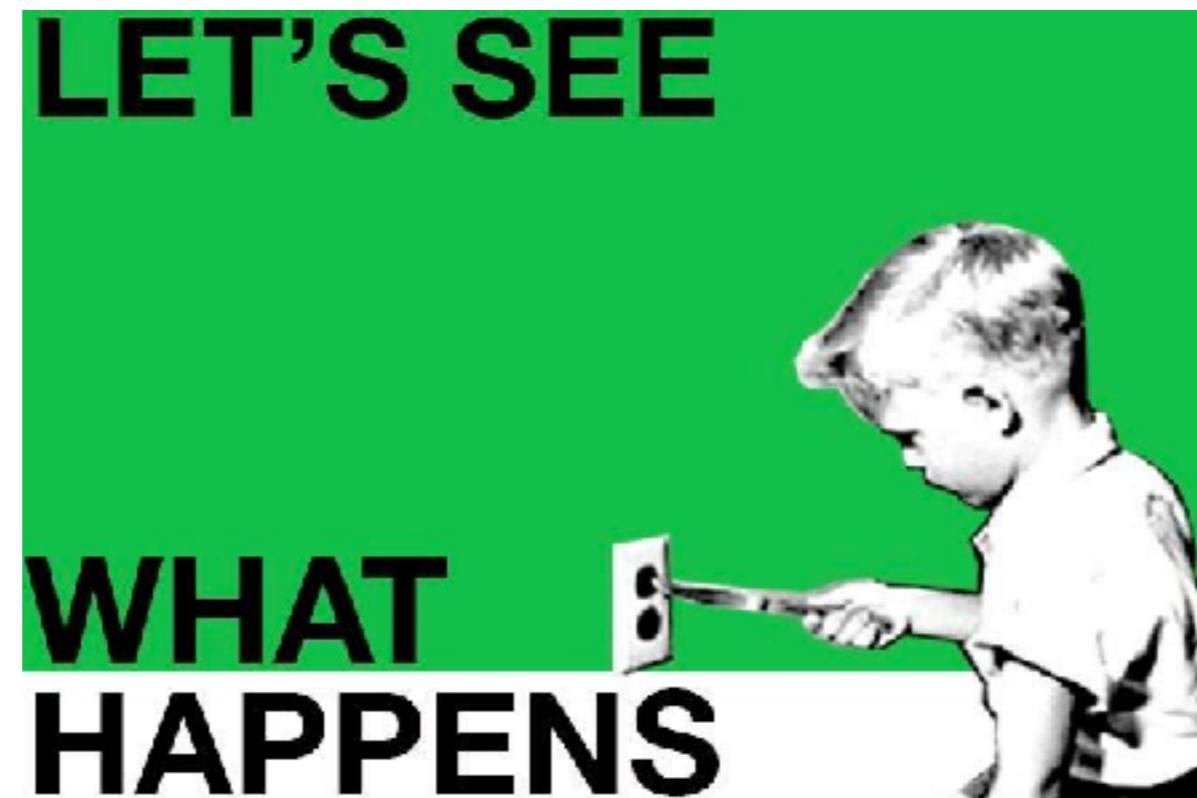
Make it easy for anyone to get
the latest executable

Make sure well known place where people can find



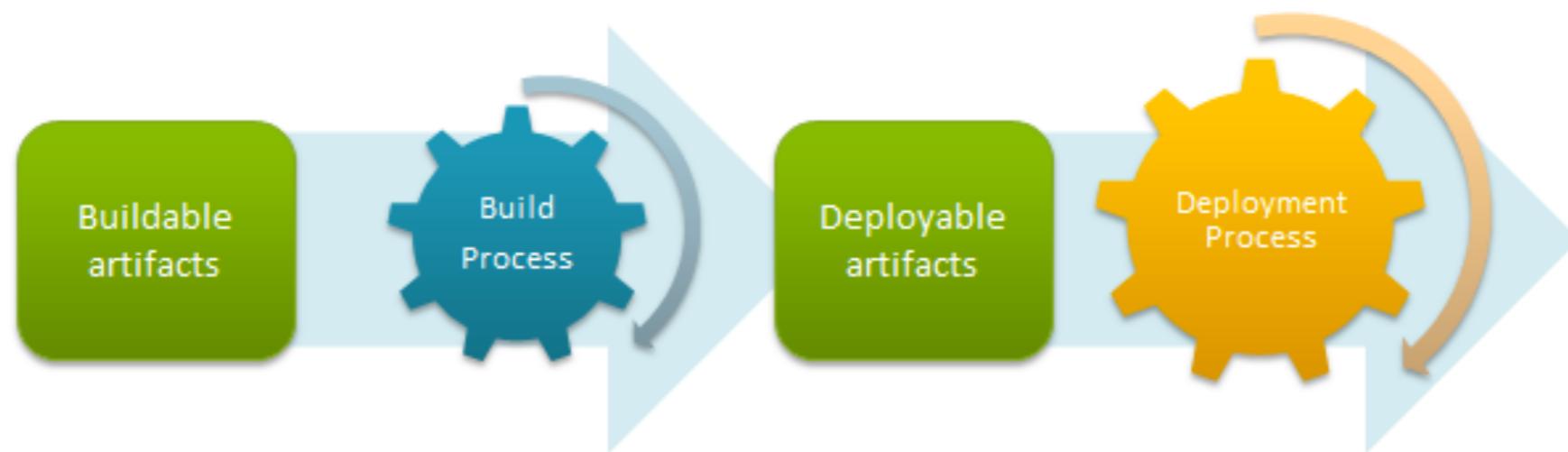
Practice 10

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



Practice 11

Automated deployment



How to achieve the CI ?



1. Use good version control

Local
Centralize
Distributed



VSS = A brown粪便 emoji with three wavy 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

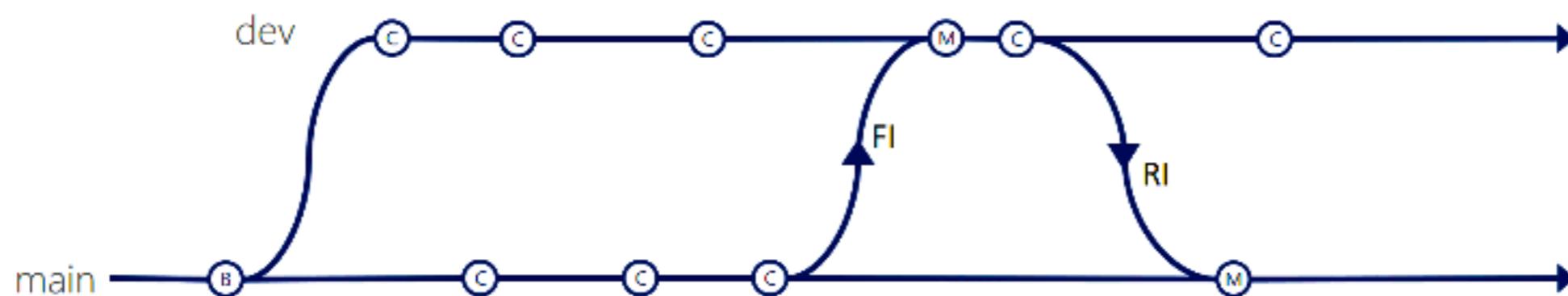
Service, Hotfix and Release isolation



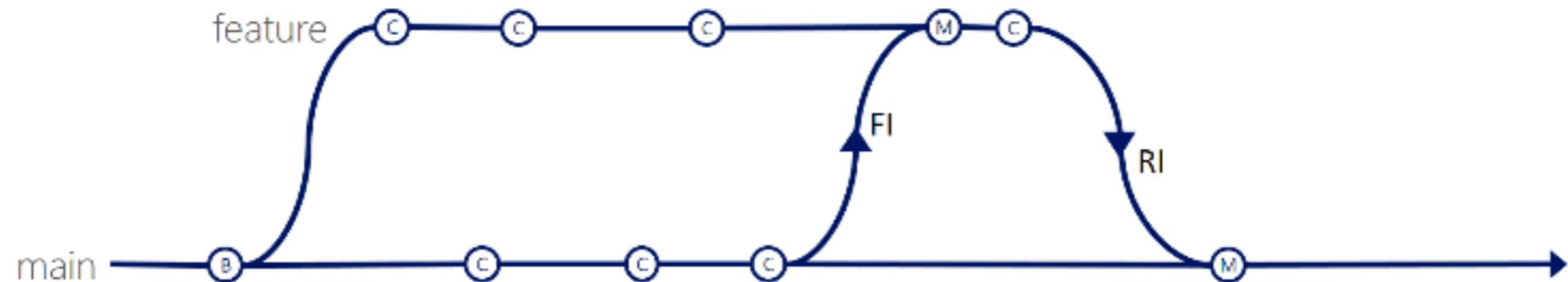
Main only



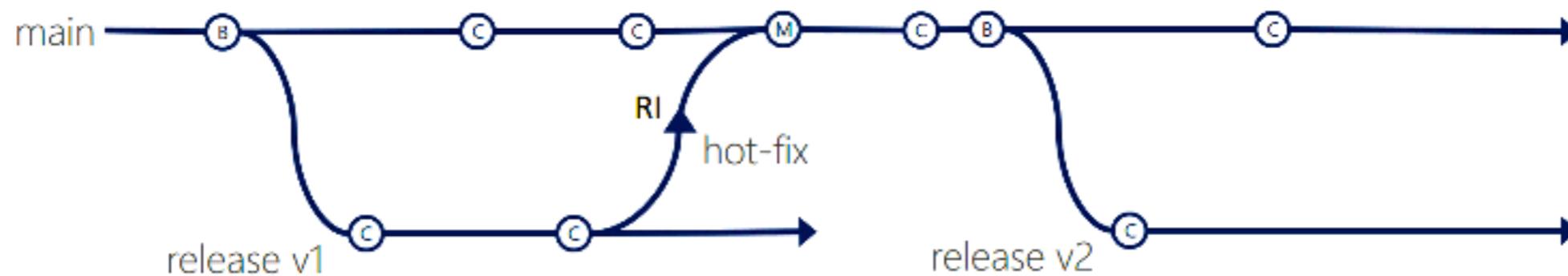
Development isolation



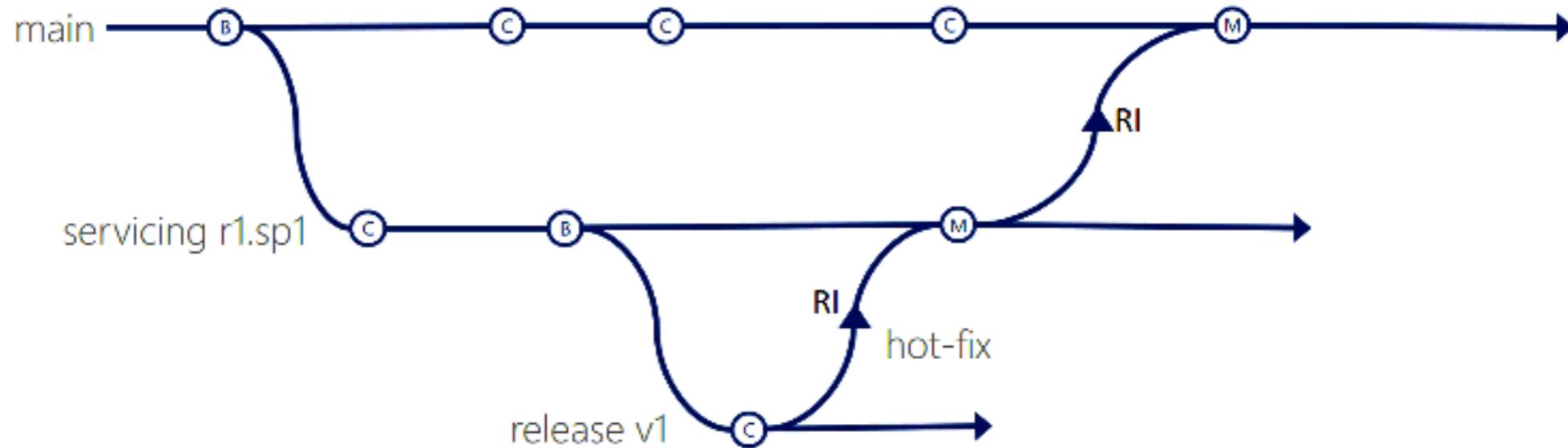
Feature isolation



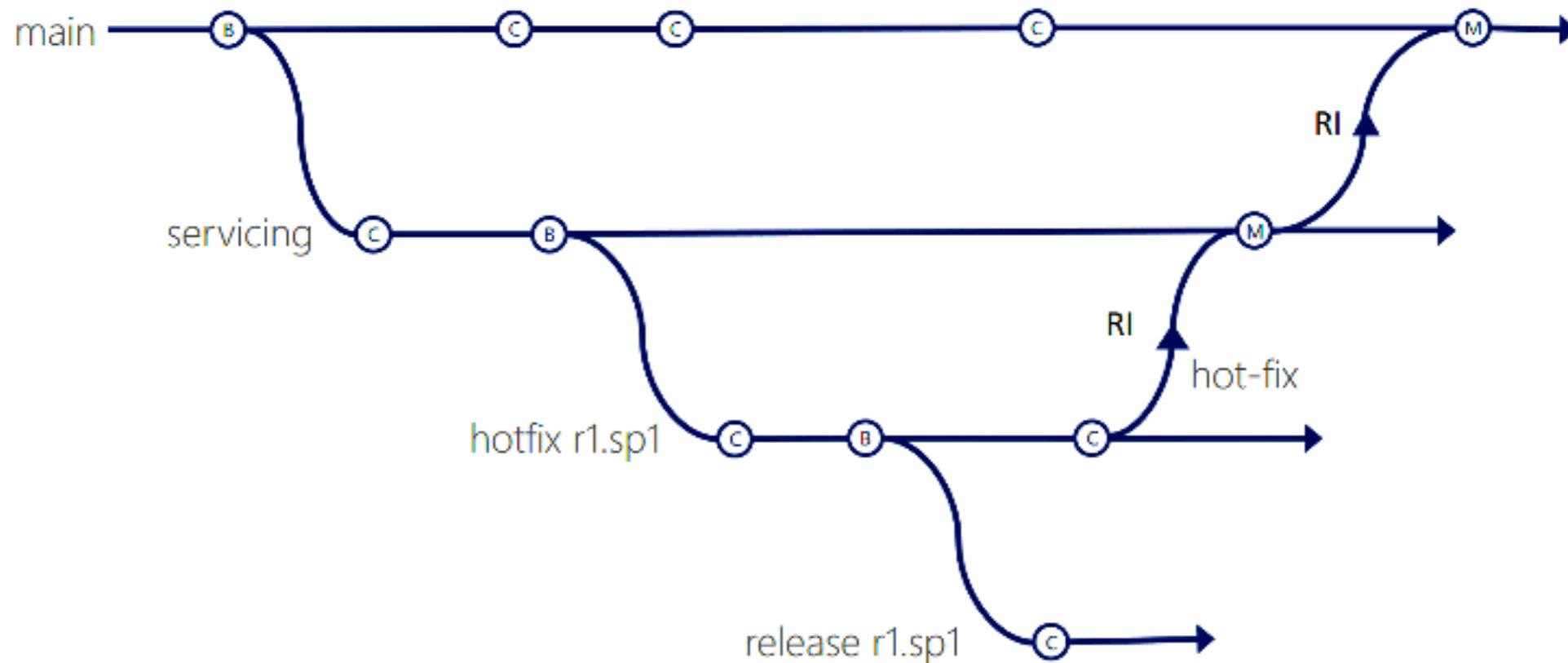
Release isolation



Service and Release isolation



Service, Hotfix, Release isolation



Validate, Validate and Validate

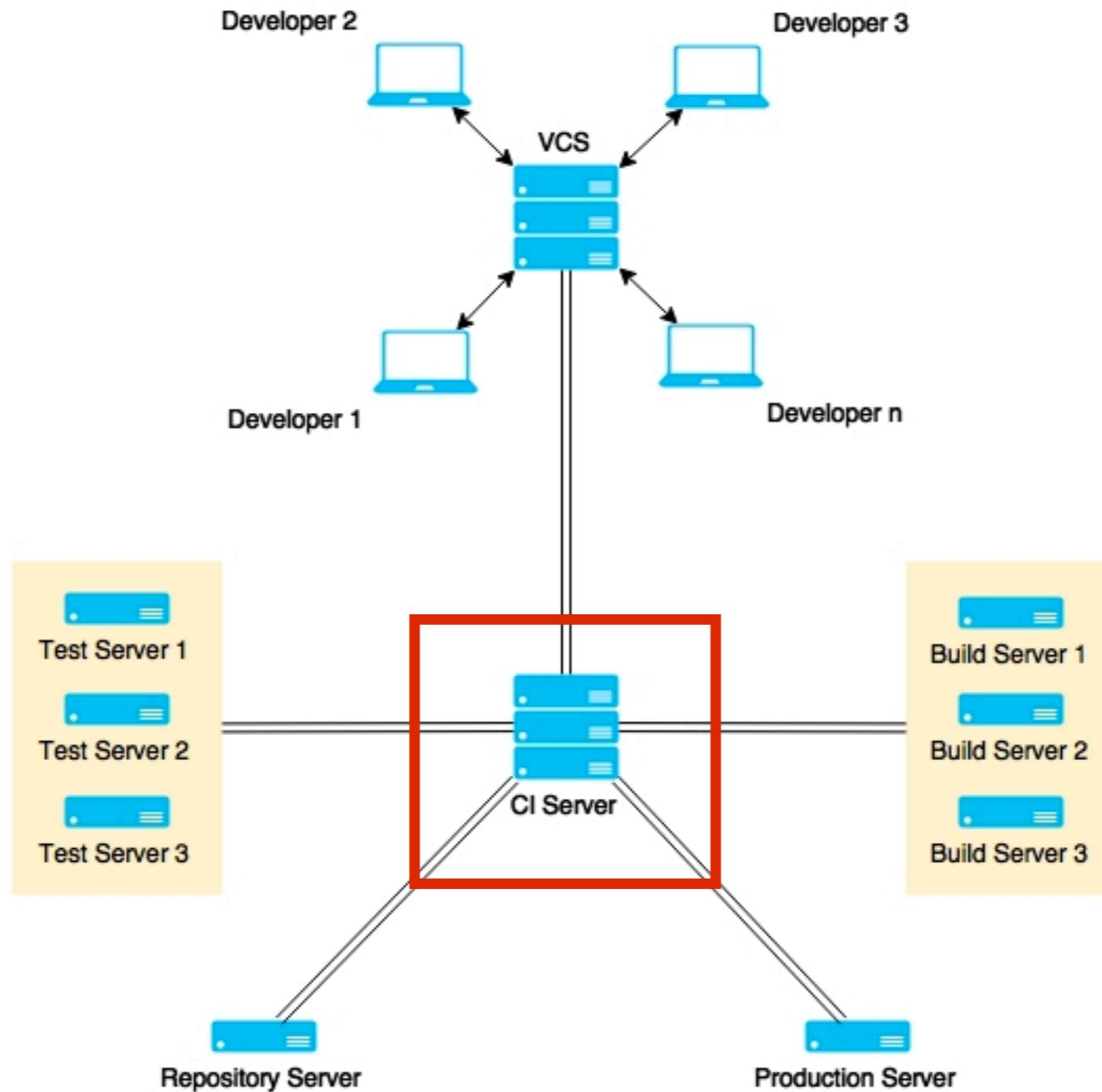


Suggestion

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



3. Use good CI tool





Jenkins



TeamCity



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”**



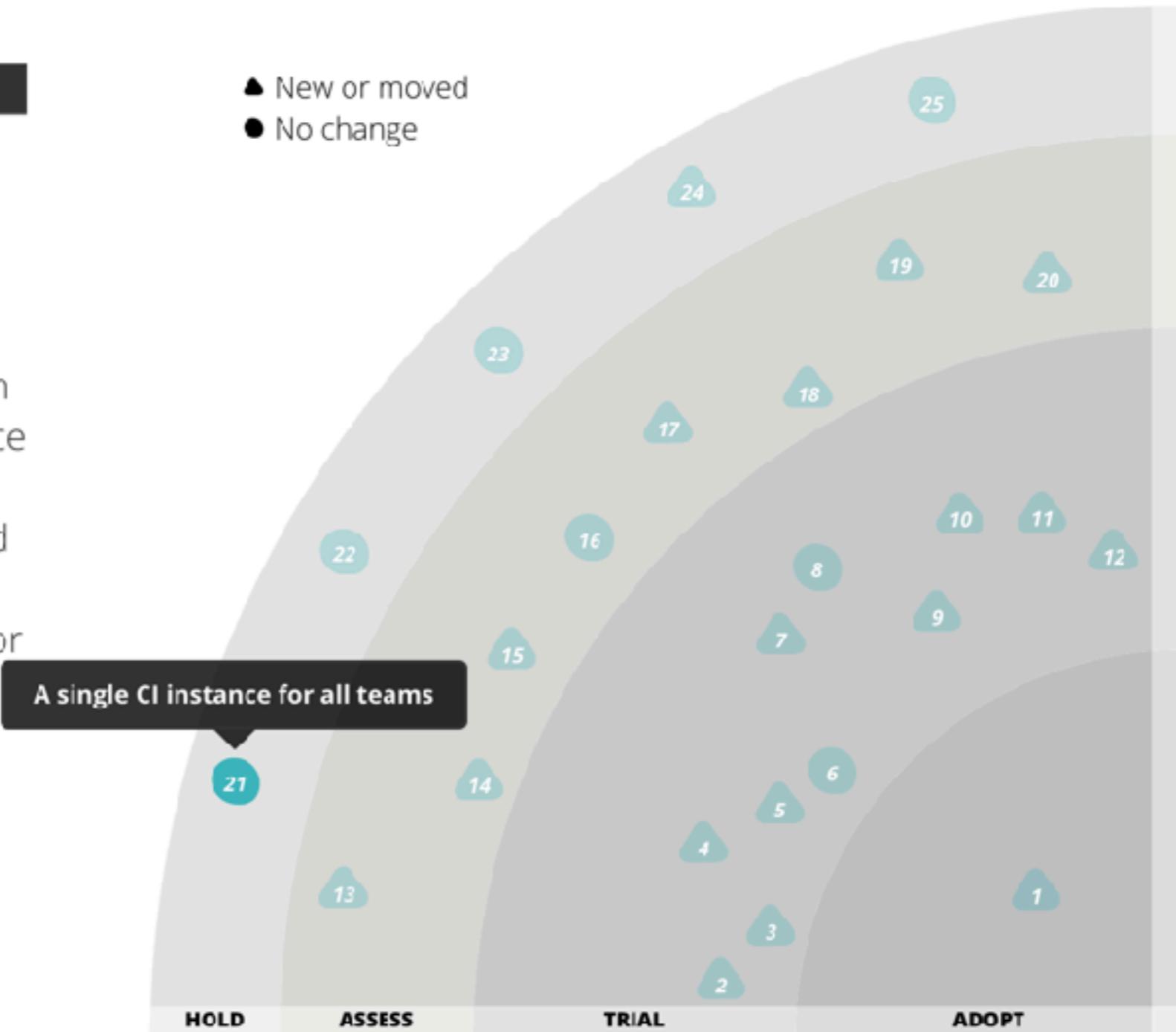
Anti-pattern for CI Server

● HOLD ?

21. A single CI instance for all teams

We're compelled to caution, again, against creating **a single CI instance for all teams**. While it's a nice idea in theory to consolidate and centralize Continuous Integration (CI) infrastructure, in reality we do not see enough maturity in the tools and products in this space to achieve the desired outcome. Software delivery teams which must use the centralized CI offering regularly have long delays depending on a central team to perform minor configuration tasks, or to troubleshoot problems in the shared infrastructure and tooling. At this stage, we continue to recommend that organizations limit their centralized investment to establishing patterns, guidelines and support for delivery teams to operate their own CI infrastructure.

- ▲ New or moved
- No change



<https://www.thoughtworks.com/radar/techniques>



Let's start with CI/CD



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



Jenkins

<https://jenkins.io/>



Centralize Continuous Integration Server



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 explains: "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



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
ngine 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

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Includes the Jenkins logo, a search bar, a user icon for "somkiat", and a "log out" link.
- Left Sidebar:** A vertical menu with links: "New Item", "People", "Build History", "Manage Jenkins", "My Views", and "Credentials".
- Main Content:** A large "Welcome to Jenkins!" message with a sub-instruction: "Please [create new jobs](#) to get started."
- Build Queue:** A section indicating "No builds in the queue."
- Build Executor Status:** A section showing "1 Idle" and "2 Idle" executors.
- Page Footer:** Includes a timestamp "Page generated: Jun 14, 2017 2:08:57 PM ICT", a "REST API" link, and "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 ...



Disable Jenkins's security



Set useSecurity=false

<JENKINS HOME>/config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<hudson>
    <disabledAdministrativeMonitors/>
    <version>2.89.3</version>
    <numExecutors>2</numExecutors>
    <mode>NORMAL</mode>
    <useSecurity>false</useSecurity>
    <authorizationStrategy class="hudson.security.LoggedInAuthorizationStrategy">
        <denyAnonymousReadAccess>true</denyAnonymousReadAccess>
    </authorizationStrategy>
```



Learn to use Jenkins in the right way



Manage Jenkins

For Administrator to config anything in Jenkins

Manage Jenkins

- [Configure System](#)
Configure global settings and paths.
- [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Credentials](#)
Configure the credential providers and types
- [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from java.util.logging output related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Configure System

Global setting and paths

Jenkins search ?

New Item People Build History Manage Jenkins My Views Credentials New View

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle

Manage Jenkins

-  [Configure System](#)
Configure global settings and paths.
-  [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
-  [Configure Credentials](#)
Configure the credential providers and types
-  [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
-  [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modi
-  [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
-  [System Information](#)
Displays various environmental information to assist trouble-shooting.
-  [System Log](#)
System log captures output from java.util.logging output related to Jenkins.
-  [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Configure System

JENKINS Home

of executors

Label name of node

Environment variables

Email notification



Configure Global Security

Setting for secure Jenkins

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', and 'New View'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main area is titled 'Manage Jenkins' and contains several configuration options: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon, highlighted with a red box), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench icon), 'Reload Configuration from Disk' (refresh icon), 'Manage Plugins' (puzzle piece icon), 'System Information' (monitor icon), 'System Log' (document icon), and 'Load Statistics' (heartbeat icon). A search bar and a help icon are at the top right.

Manage Jenkins

- [Configure System](#)
Configure global settings and paths.
- [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Credentials](#)
Configure the credential providers and types
- [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modi
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from `java.util.logging` related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Global Tool Configuration

Config tools, location and automatic installers

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected), My Views, Credentials, and New View. Below that are sections for Build Queue (empty) and Build Executor Status (2 Idle). The main area is titled "Manage Jenkins" and lists several configuration options: Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration (which is highlighted with a red box), Reload Configuration from Disk, Manage Plugins, System Information, System Log, and Load Statistics.

Link	Description
Configure System	Configure global settings and paths.
Configure Global Security	Secure Jenkins; define who is allowed to access/use the system.
Configure Credentials	Configure the credential providers and types
Global Tool Configuration	Configure tools, their locations and automatic installers.
Reload Configuration from Disk	Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
Manage Plugins	Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
System Information	Displays various environmental information to assist trouble-shooting.
System Log	System log captures output from <code>java.util.logging</code> related to Jenkins.
Load Statistics	Check your resource utilization and see if you need more computers for your builds.



Global Tool Configuration

Apache Maven
JDK (Java Development Kit)
Git
Gradle
Docker



Manage Plugins

Add, remove, enable/disable plugins

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected and highlighted in blue), My Views, Credentials, and New View. Below that are sections for Build Queue (empty) and Build Executor Status (2 Idle). The main content area is titled "Manage Jenkins" and lists several configuration options: Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration, Reload Configuration from Disk, Manage Plugins (which is highlighted with a red box), System Information, System Log, and Load Statistics.

Manage Jenkins

- [Configure System](#)
Configure global settings and paths.
- [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Credentials](#)
Configure the credential providers and types
- [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modi
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from `java.util.logging` related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Manage Plugins

Add, remove, enable/disable plugins

Filter:

Updates	Available	Installed	Advanced
Install	Name ↓	Version	Installed
		No updates	

Update information obtained: 14 hr ago [Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.



Manage Plugins

Filter plugins

Filter:

Updates Available Installed Advanced

Install	Name ↓	Version	Installed
No updates			

Update information obtained: 14 hr ago [Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.



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, Available (which is highlighted with a red border), Installed, and Advanced. Below the tabs is a search bar labeled 'Filter:' with a magnifying glass icon. The main area displays a table of available plugins under the heading '.NET Development'. The columns are 'Name' and 'Version'. Each plugin entry includes a checkbox, a brief description, and the version number. At the bottom, there are three buttons: 'Install without restart', 'Download now and install after restart' (which is also highlighted with a red border), and 'Check now'.

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

Buttons at the bottom:

- 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

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

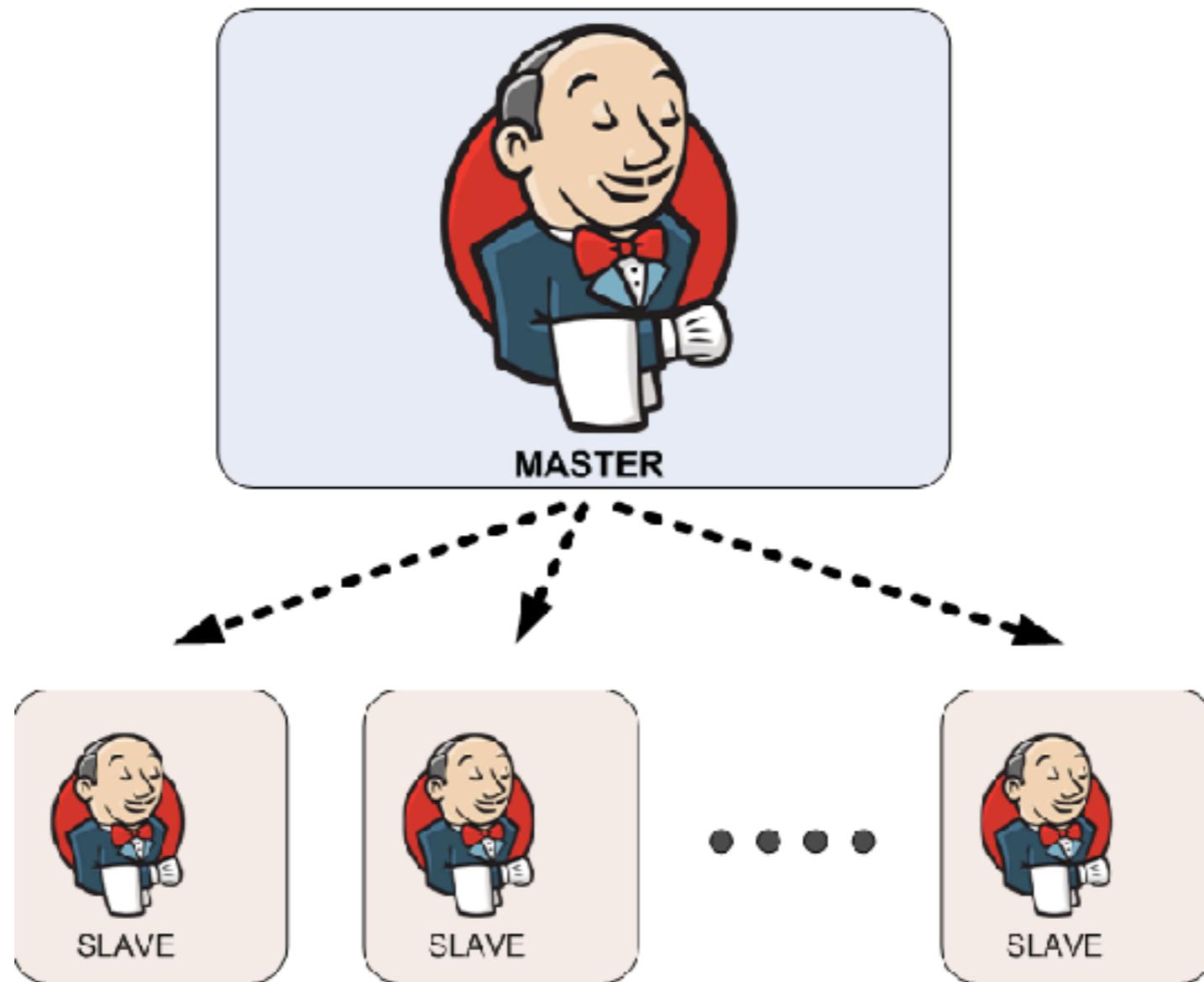
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Mac OS X (x86_64)	In sync	5.73 GB	463.00 MB	5.73 GB	0ms
	Data obtained	36 min	36 min	36 min	36 min	36 min	36 min

Refresh status



Manage Nodes

Master-slave concept to scale Jenkins



Create a first Job



1. Create a new job

The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo. Below it is a navigation bar with the word "Jenkins" and a dropdown arrow. The main content area has a "Welcome to Jenkins!" message in large bold letters. Below it is a teal box containing the text "Please create new jobs to get started." A red box highlights the "New Item" button in the left sidebar, which has a icon of a folder with a plus sign. The sidebar also lists "People", "Build History", "Manage Jenkins", "My Views", and "Credentials". At the bottom left is a "Build Queue" section with the message "No builds in the queue."

New Item

People

Build History

Manage Jenkins

My Views

Credentials

Welcome to Jenkins!

Please create new jobs to get started.

Build Queue

No builds in the queue.



2. Fill in a job name

Enter an item name

hello

» 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.



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.



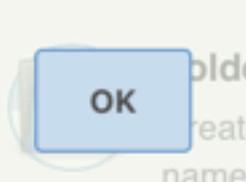
External Job

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.



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 space, so you can have multiple things of the same name as long as they are in different folders.



3. Choose type of job

Enter an item name

hello

» 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.



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.



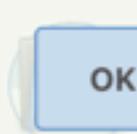
External Job

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.



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 space, so you can have multiple things of the same name as long as they are in different folders.



3. Choose type of job

Enter an item name

hello

» 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.



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.



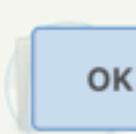
External job

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.



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 space, so you can have multiple things of the same name as long as they are in different folders.



4. General section

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Project name: hello

Description:

[Plain text] [Preview](#)

Discard old builds [?](#)

GitHub project [?](#)

This project is parameterized [?](#)

Throttle builds [?](#)

Disable this project [?](#)

Execute concurrent builds if necessary [?](#)

[Advanced...](#)

Source Code Management

None

Save **Apply**



4.1 Advanced options

The screenshot shows the Jenkins 'General' configuration page for a project named 'hello'. The 'General' tab is selected. The 'Project name' field contains 'hello'. The 'Description' field is empty. Below these fields is a large text area containing '[Plain text] Preview'. Underneath the preview area is a list of checkboxes:

- Discard old builds (with a help icon)
- GitHub project (with a help icon)
- This project is parameterized (with a help icon)
- Throttle builds (with a help icon)
- Disable this project (with a help icon)
- Execute concurrent builds if necessary (with a help icon)

At the bottom right of the page, there is a red-bordered button labeled 'Advanced...'.

Below the main configuration area, there is a 'Source Code Management' section with a 'None' status, a 'Save' button, and an 'Apply' button.



4.1 Advanced options

The screenshot shows the 'General' configuration page for a Jenkins job. The top navigation bar includes tabs for 'General', 'Source Code Management', 'Build Triggers', and 'Build Environment'. The 'General' tab is selected. Below the tabs, there is a list of checkboxes for advanced options:

- Quiet period
- Retry Count
- Block build when upstream project is building
- Block build when downstream project is building
- Use custom workspace

Below these options is a 'Display Name' field, which is currently empty. At the bottom of the configuration section is another checkbox:

- Keep the build logs of dependencies



5. Source code management

By default is Git and Subversion

The screenshot shows the Jenkins configuration interface for a job. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. The 'Source Code Management' tab is selected, highlighted in dark grey. Below it, the 'Source Code Management' section displays three radio button options: 'None' (selected), 'Git', and 'Subversion'. The 'Build Triggers' section, also under the 'Source Code Management' tab, contains five checkbox options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. The 'Poll SCM' option is selected, indicated by a blue checked mark.

Source Code Management
<input checked="" type="radio"/> None
<input type="radio"/> Git
<input type="radio"/> Subversion

Build Triggers
<input type="checkbox"/> Trigger builds remotely (e.g., from scripts)
<input type="checkbox"/> Build after other projects are built
<input type="checkbox"/> Build periodically
<input type="checkbox"/> GitHub hook trigger for GITScm polling
<input checked="" type="checkbox"/> Poll SCM



6. Build trigger

When to run this job

The screenshot shows a Jenkins configuration page with a navigation bar at the top containing tabs: General, Source Code Management, Build Triggers (which is selected and highlighted in bold), and Build Environment. The main content area is titled 'Build Triggers' and contains a list of five options, each preceded by an unchecked checkbox:

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM



6.1 Periodically

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Build Triggers

Trigger builds remotely (e.g., from scripts) ?
 Build after other projects are built ?
 Build periodically ?

Schedule

H 23 * * *

⚠ No schedules so will never run

GitHub hook trigger for GITScm polling ?
 Poll SCM ?



6.2 Poll SCM

Poll SCM ?

Schedule ?

No schedules so will only run due to SCM changes if triggered by a post-commit hook

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW
MINUTE Minutes within the hour (0–59)
HOUR The hour of the day (0–23)
DOM The day of the month (1–31)
MONTH The month (1–12)
DOW The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

- * specifies all valid values
- M–N specifies a range of values
- M–N/X or */X steps by intervals of X through the specified range or whole valid range
- A,B,...,Z enumerates multiple values



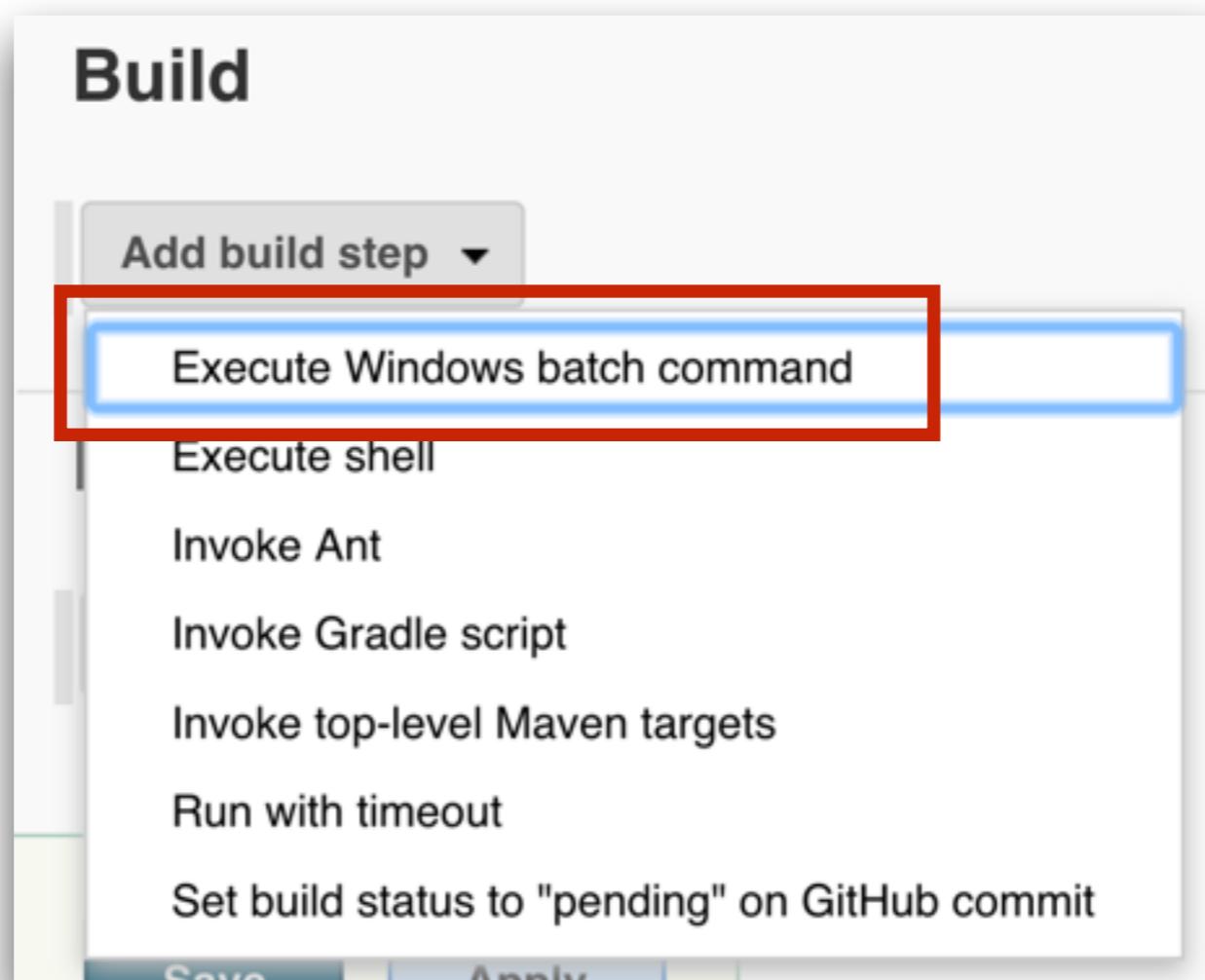
7. Add build step

What to run this job

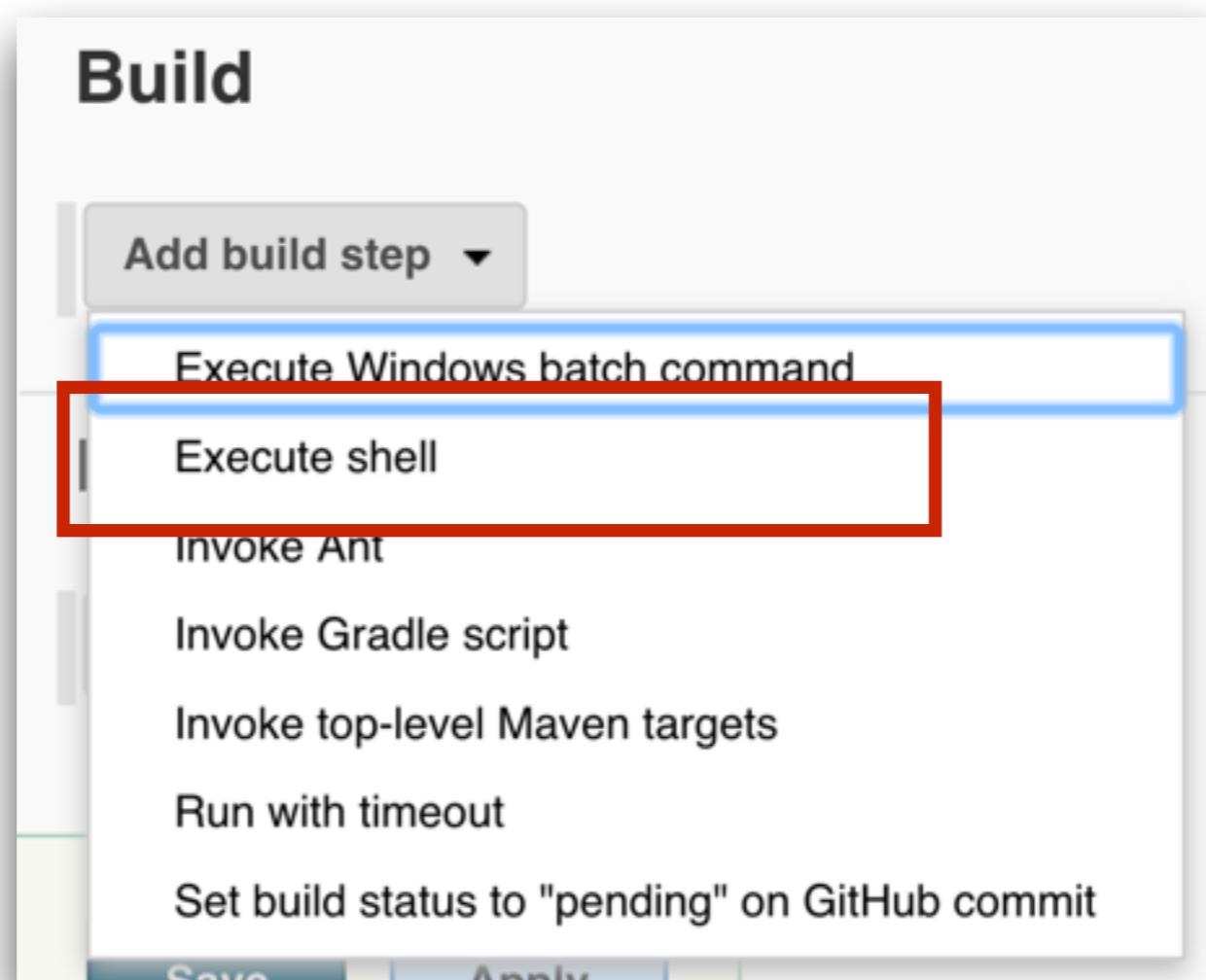
The screenshot shows the Jenkins job configuration interface. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, **Build**, and Post-build Actions. The **Build** tab is active. Below the tabs, there are two sections: **BUILD ENVIRONMENT** and **Build**. The **Build** section contains a dropdown menu labeled "Add build step ▾". A list of build steps is displayed, with "Execute Windows batch command" highlighted by a blue selection box. Other options include Execute shell, Invoke Ant, Invoke Gradle script, Invoke top-level Maven targets, Run with timeout, and Set build status to "pending" on GitHub commit. At the bottom of the dropdown are "Save" and "Apply" buttons.



7.1 For Windows



7.2 For Linux/Mac



8. Post build actions

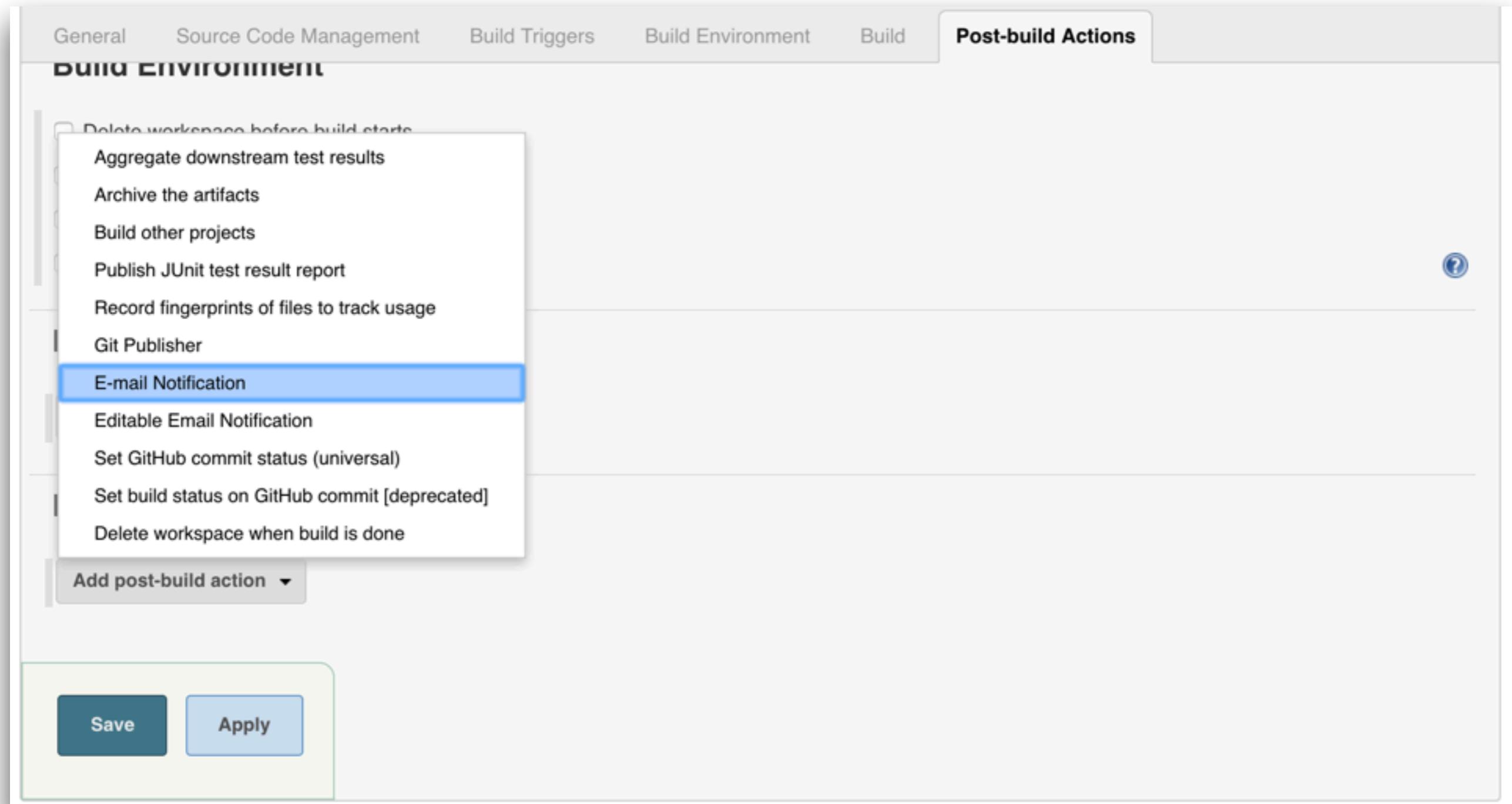
Generate reports

Send email

Run other jobs/projects



8. Post build actions



The screenshot shows the Jenkins configuration interface for a job. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. The Post-build Actions tab is selected. Below the tabs, a section titled "BUILD ENVIRONMENT" is visible. A dropdown menu is open under "Post-build Actions", listing various actions. The "E-mail Notification" option is highlighted with a blue selection bar. At the bottom of the configuration area, there are "Save" and "Apply" buttons.

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification**
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

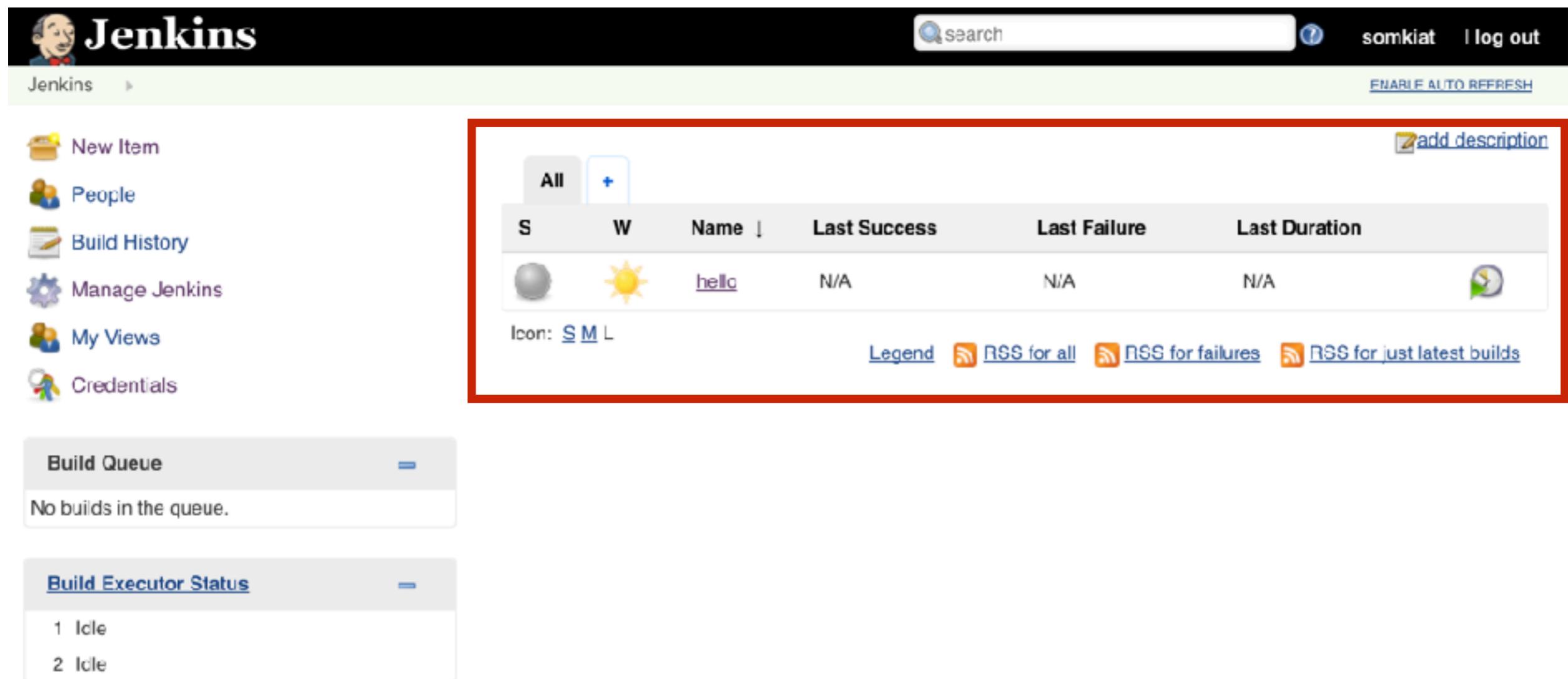
Add post-build action ▾

Save Apply



9. Run your job

Manual and Scheduler run



The screenshot shows the Jenkins dashboard with a red box highlighting the job list area. The job 'hello' is listed with a yellow sun icon, indicating it is successful. The dashboard also includes sections for Build Queue and Build Executor Status.

S	W	Name	Last Success	Last Failure	Last Duration
		hello	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

Legend: [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle



9. Run your job

Start build your job

Jenkins

search somkiat log out

New Item People Build History Manage Jenkins My Views Credentials

All +

S	W	Name	Last Success	Last Failure	Last Duration
		hello	N/A	N/A	N/A

Icon: S M L Legend RSS for all RSS for failures RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle



9. Run your job

Start build your job

The screenshot shows the Jenkins interface for a project named "hello". The top navigation bar shows "Jenkins" and "hello". The main content area has a title "Project hello". On the left, there is a sidebar with several options: "Back to Dashboard", "Status", "Changes", "Workspace", "Build Now", "Delete Project", and "Configure". The "Build Now" button is highlighted with a red box. To the right of the sidebar, there are two links: "Workspace" and "Recent Changes".



9. Run your job

See your job's output

Jenkins > hello > #1

Back to Project
 Status
 Changes
Console Output View as plain text
 Edit Build Information
 Delete Build
 Next Build

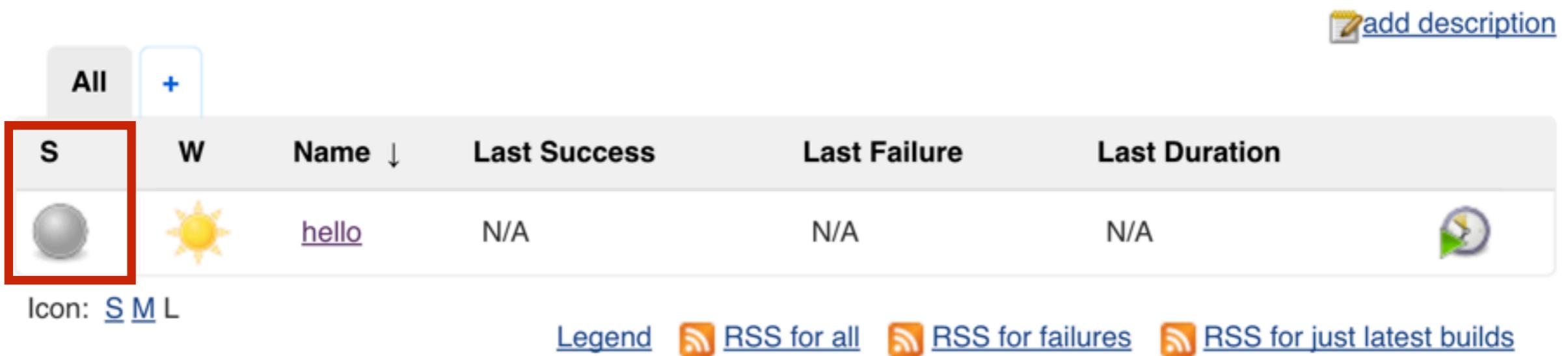
Console Output

Started by user [Somkiat Puisungnoen](#)
Building in workspace /Users/somkiat/Downloads/set/workspace/hello
Finished: SUCCESS



10. See job's status

By default is **Blue**, **Red** and **Gray**



The screenshot shows a Jenkins dashboard with a single job listed:

All	W	Name ↓	Last Success	Last Failure	Last Duration
S	W	hello	N/A	N/A	N/A

Icon: [S](#) [M](#) [L](#)

Legend: [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Blue = build success

Red = build failure

Gray = disabled/never executed



11. See job's health



A screenshot of a Jenkins job health status page. At the top right is a blue 'add description' button. Below it is a toolbar with 'All' selected and a '+' icon. The main table has columns: Status (S), Name (hello), Last Success (N/A), Last Failure (N/A), and Last Duration (N/A). The 'Status' column for the 'hello' job shows a yellow sun icon, which is highlighted with a red box. Below the table, there's a legend: 'Icon: S M L' where S is a grey circle, M is a yellow sun, and L is a green triangle. At the bottom are three RSS feed links: 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

S	Name ↓	Last Success	Last Failure	Last Duration
	hello	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

[Legend](#)  [RSS for all](#)  [RSS for failures](#)  [RSS for just latest builds](#)

Sunny = 100% success rate

Cloudy = 60% success rate

Raining = 40% success rate



Let's workshop

