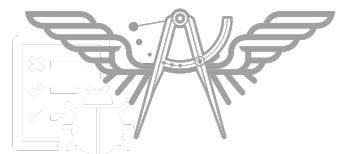




CI/CD series

# CI/CD with Jenkins





Somkiat Puisungnoen

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

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาณกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

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

Liked 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. Introduction to DevOps
8. Workshop



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



# Continuous Integration



# Why CI/CD ?

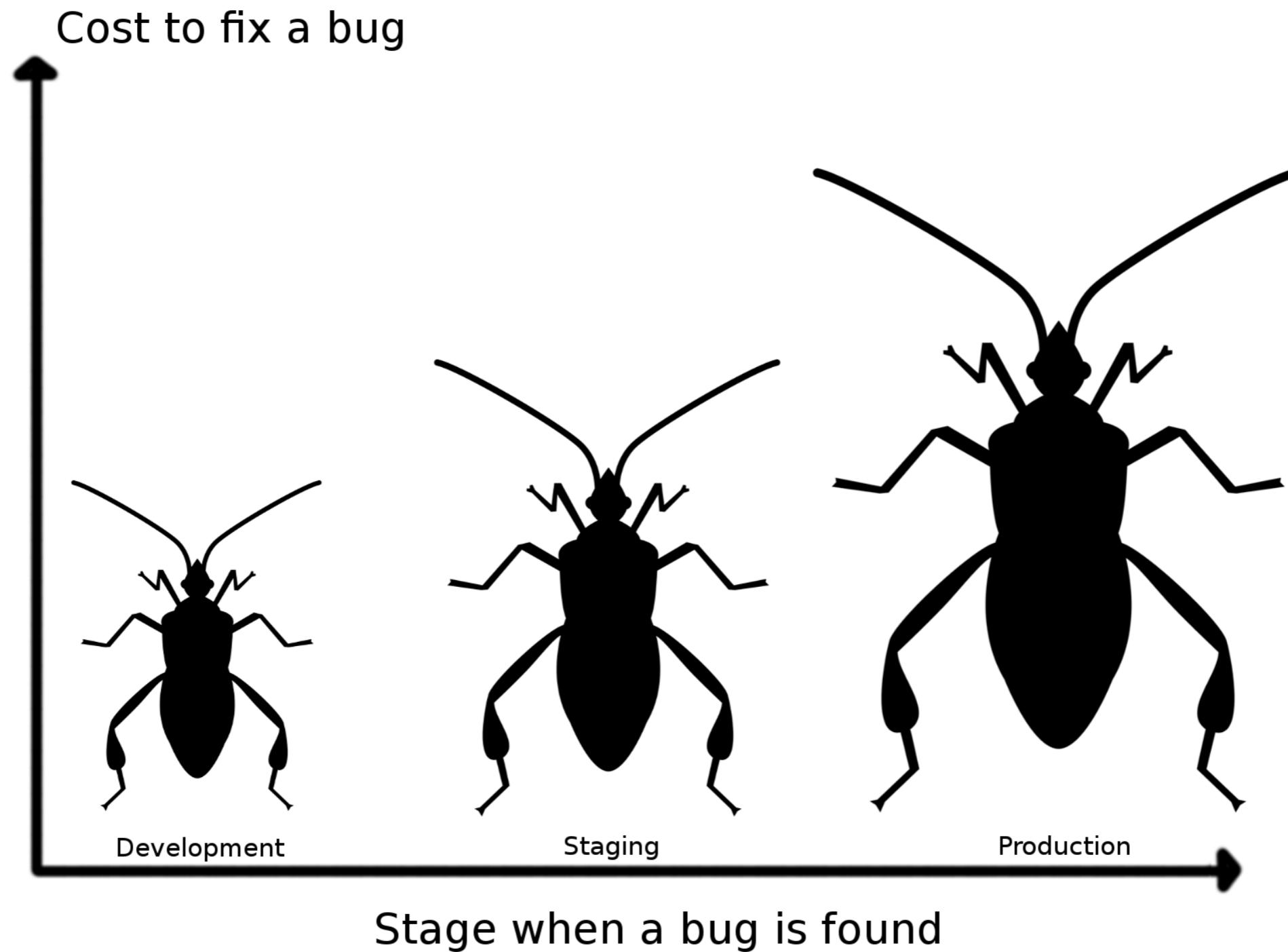


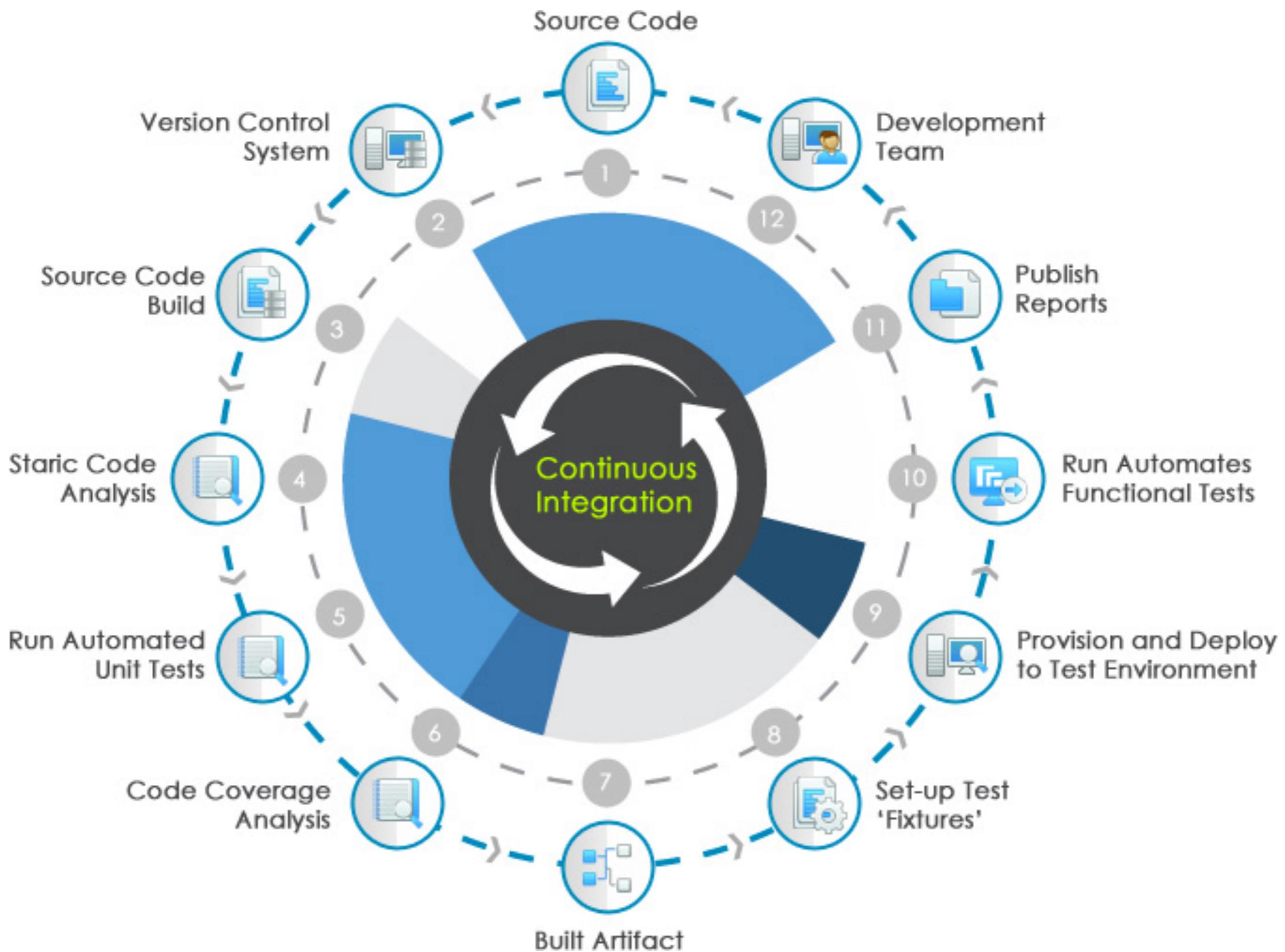
# The cost of integration

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

> go<sup>TM</sup>



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

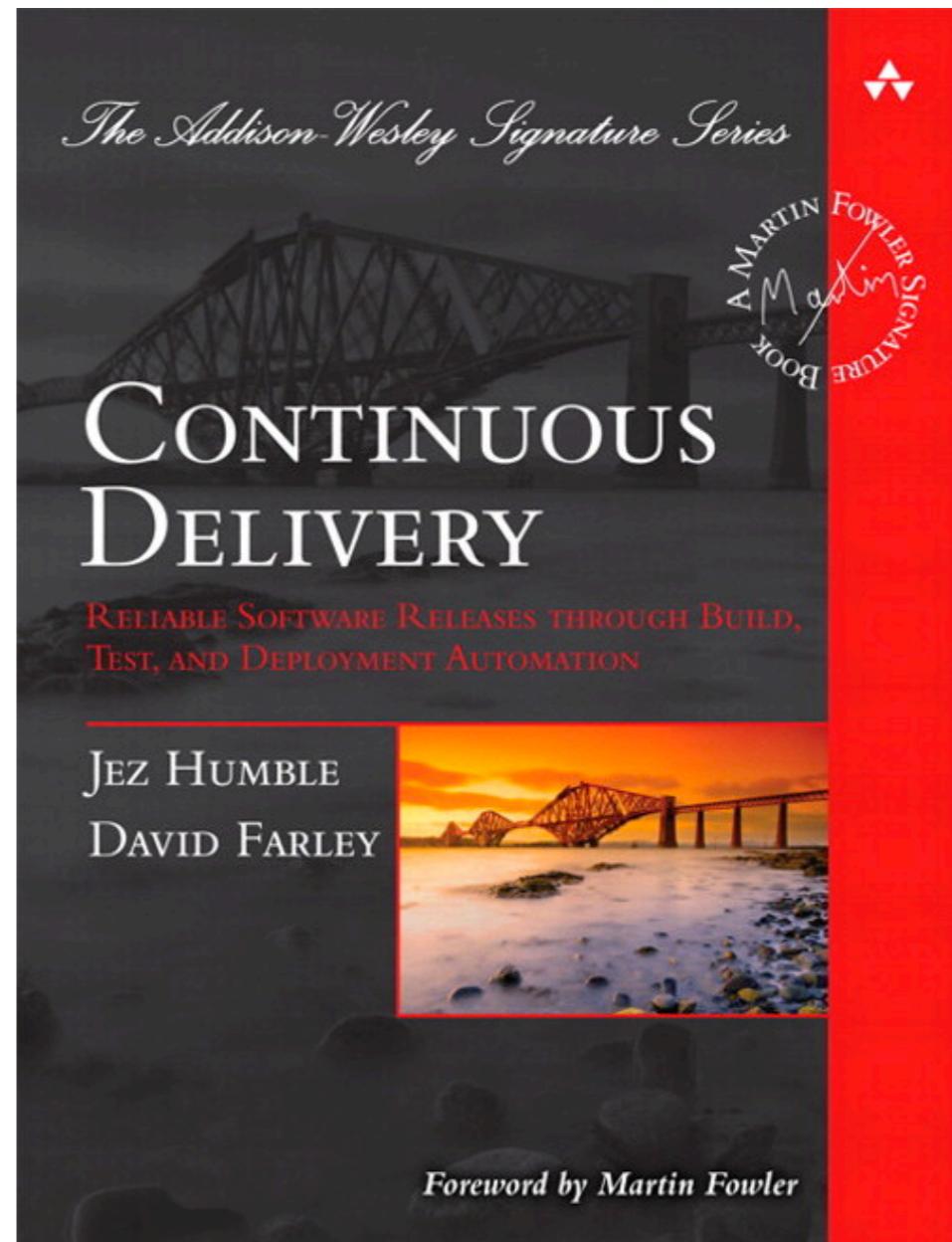
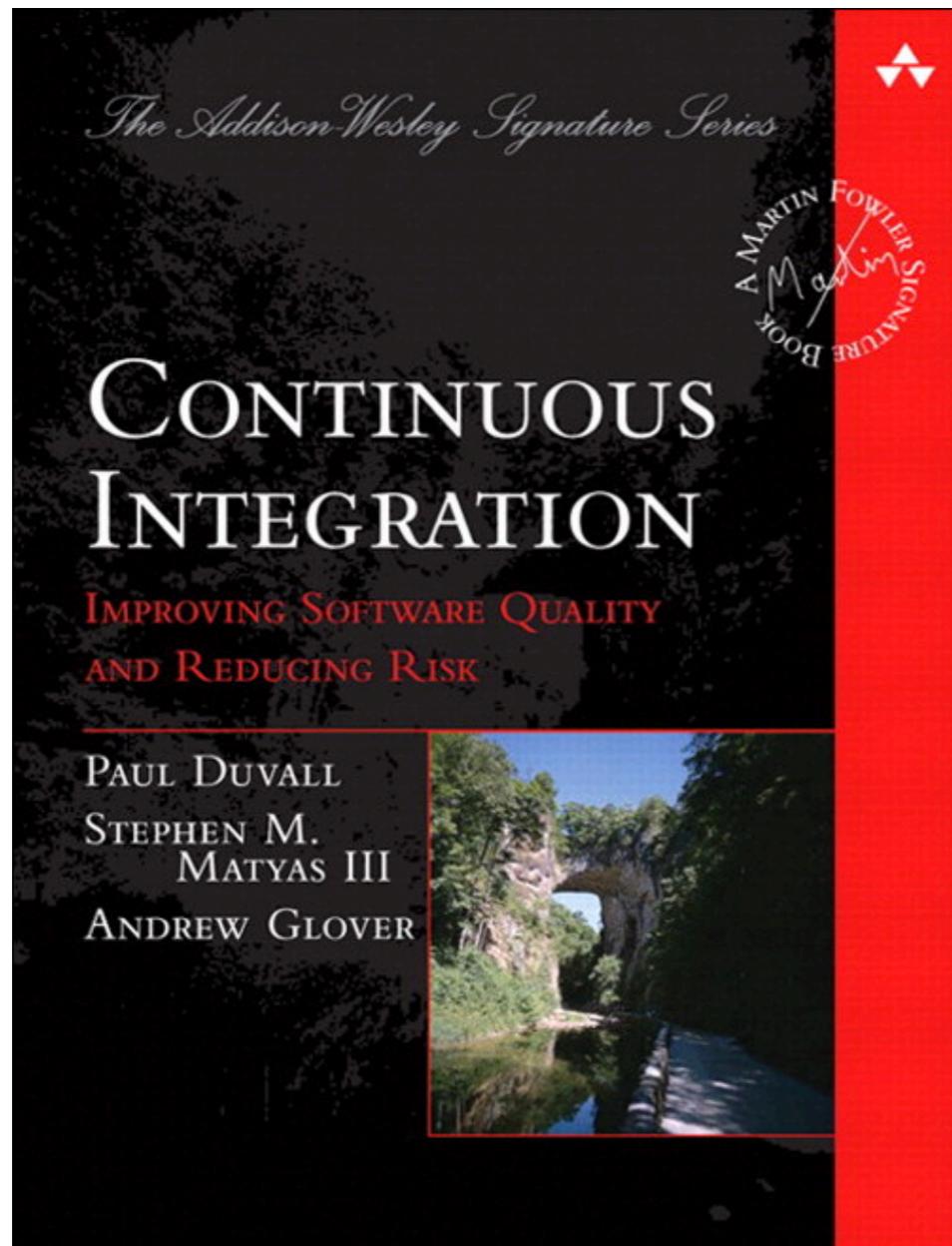
Strive for **fast feedback**



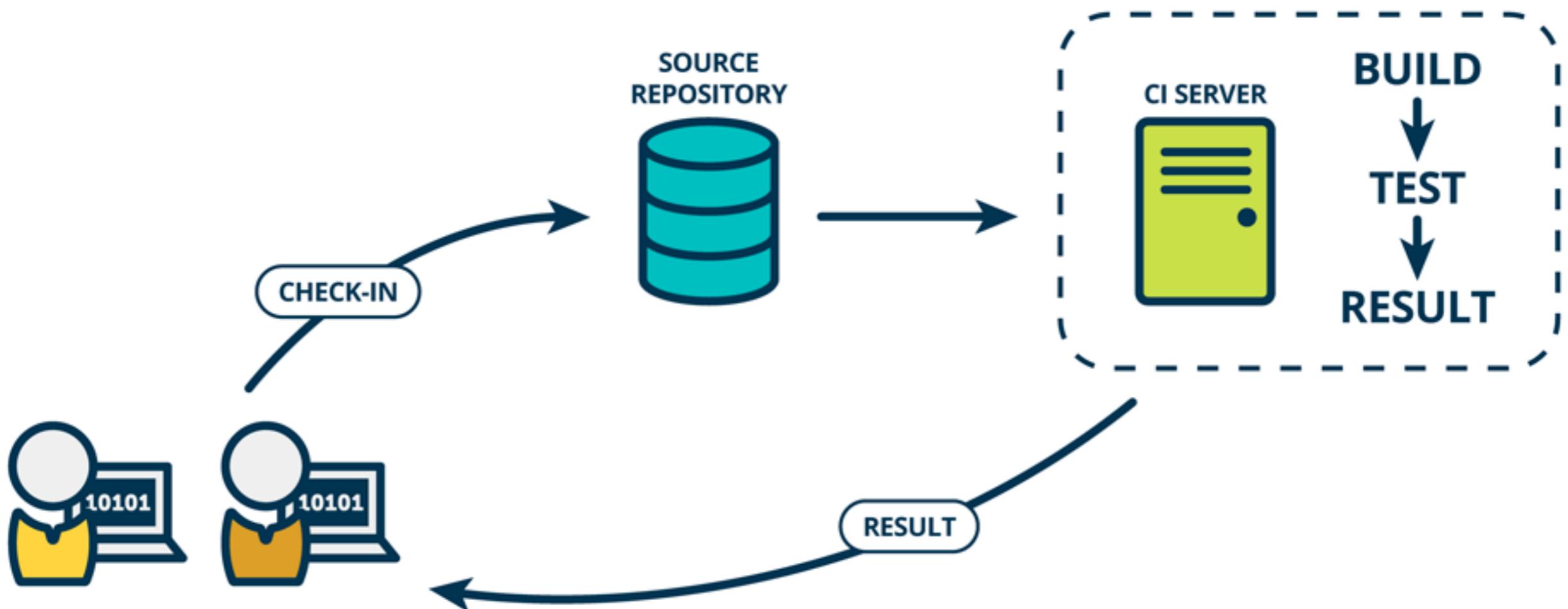
# **Practices of Continuous Integration**



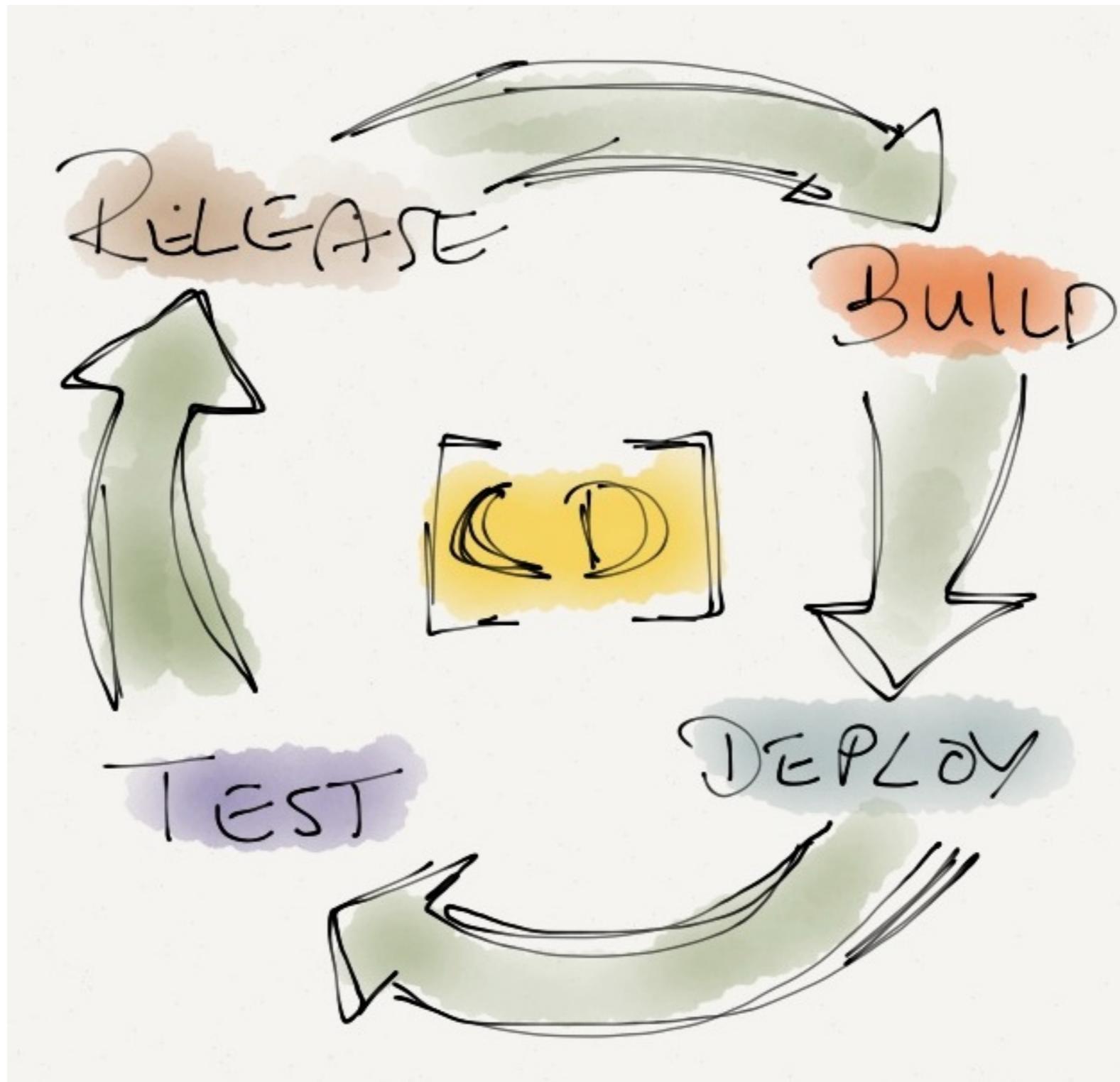
# Improve quality and reduce risk



# Continuous Integration



# CD ?



# CD ?

## CONTINUOUS DELIVERY



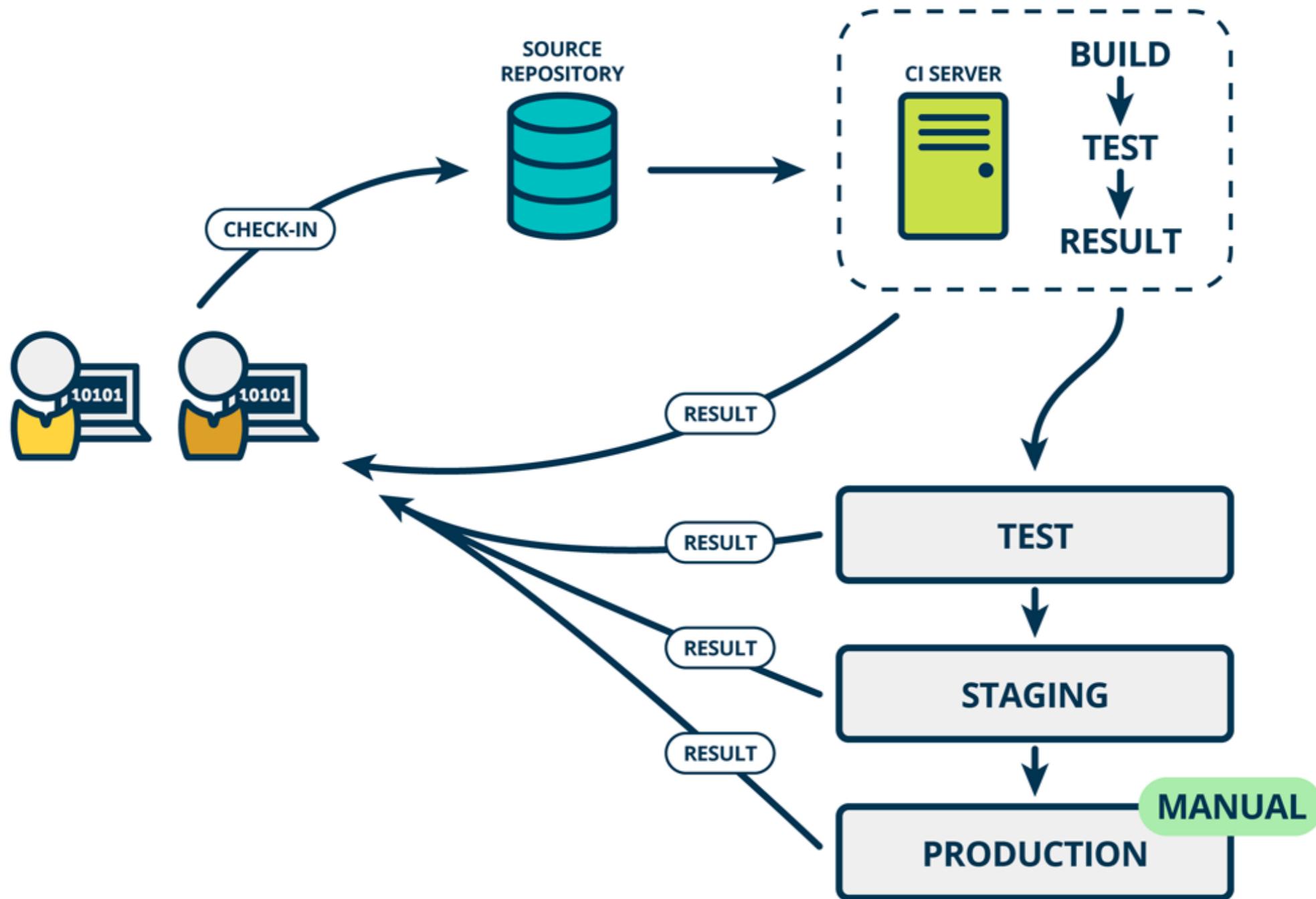
## CONTINUOUS DEPLOYMENT



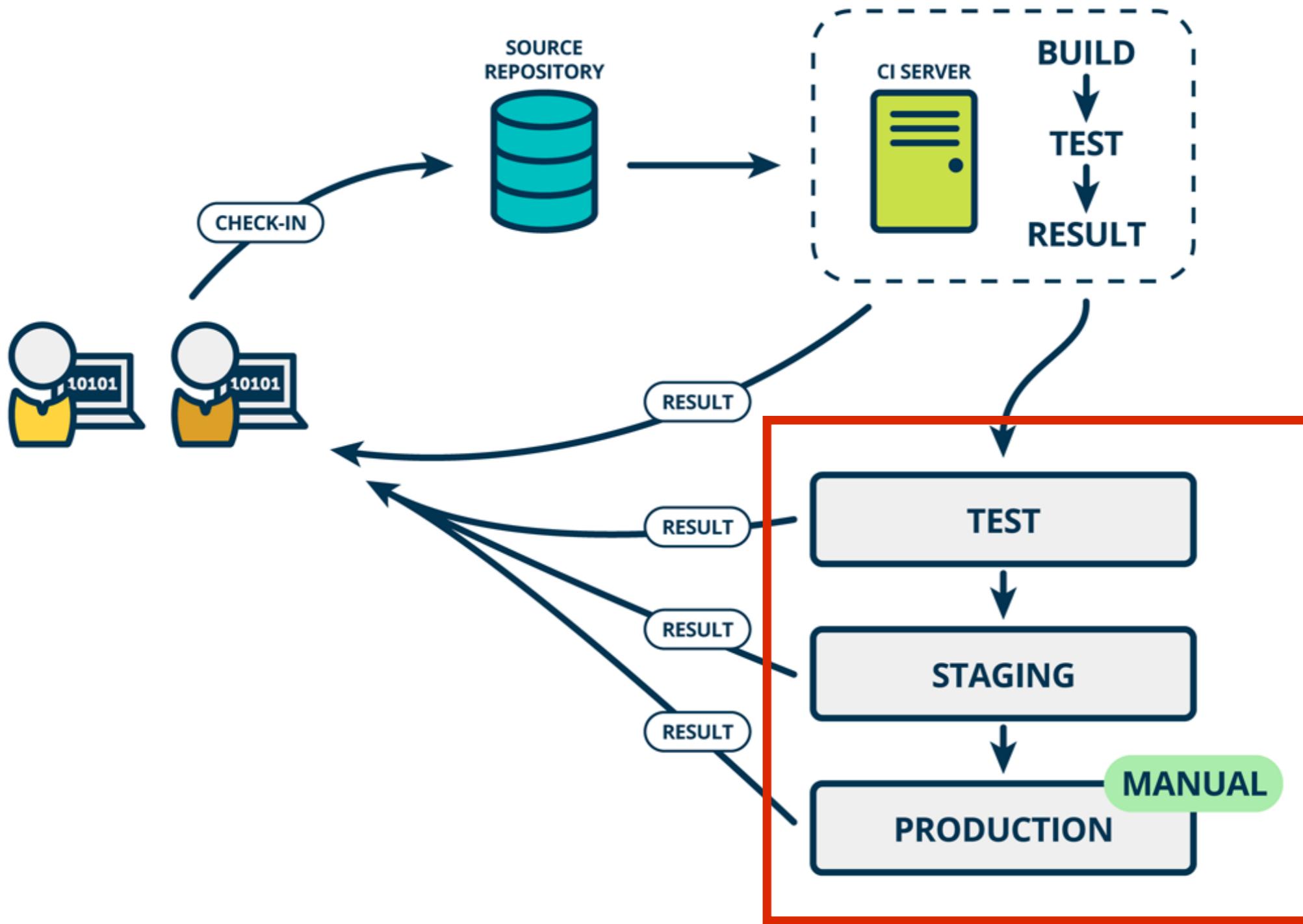
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



# Continuous Delivery



# Rise of DevOps



# **Continuous Integration**

## **is a Software development practices**



# Practice 1

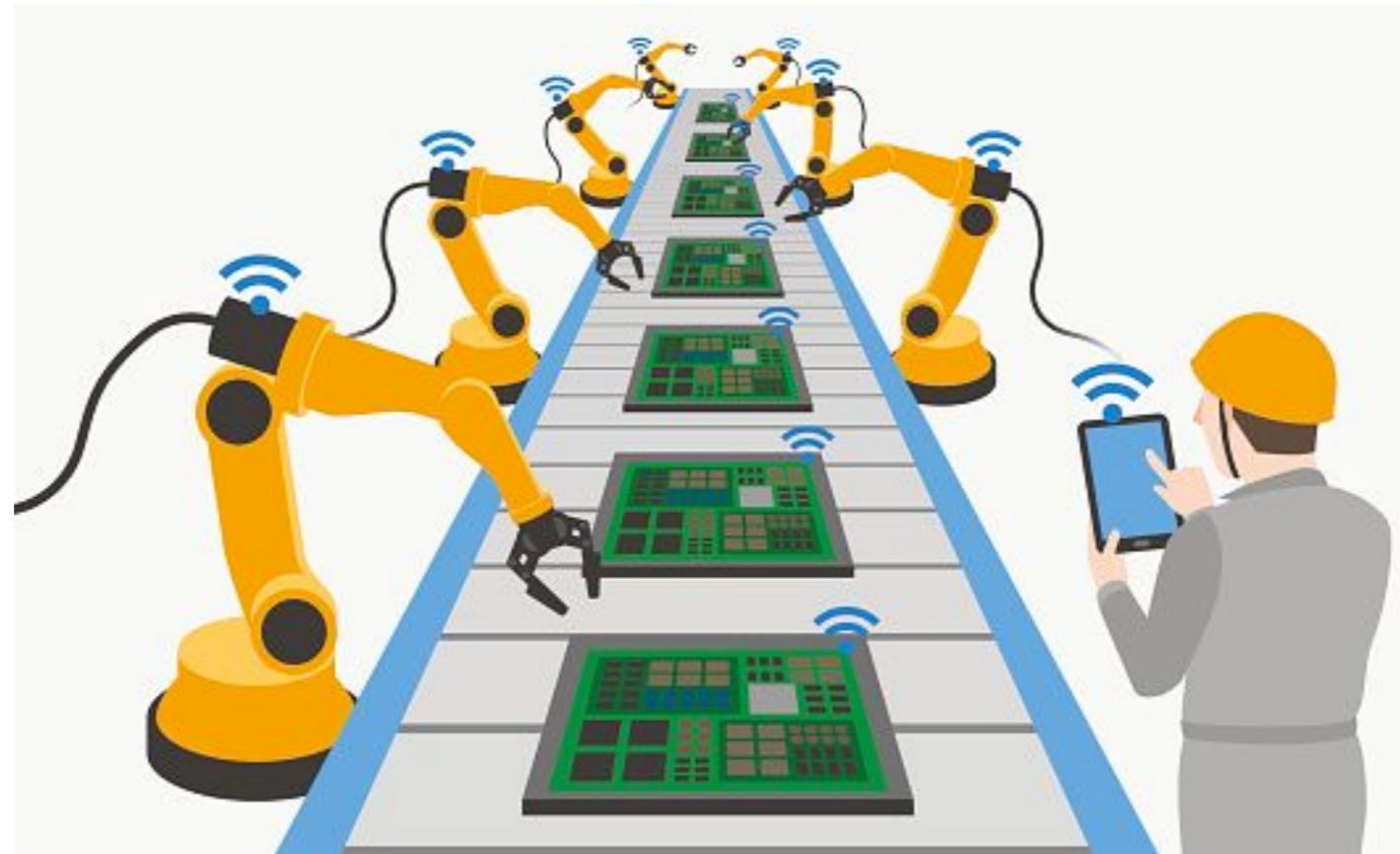
Maintain a single source repository

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



# Practice 2

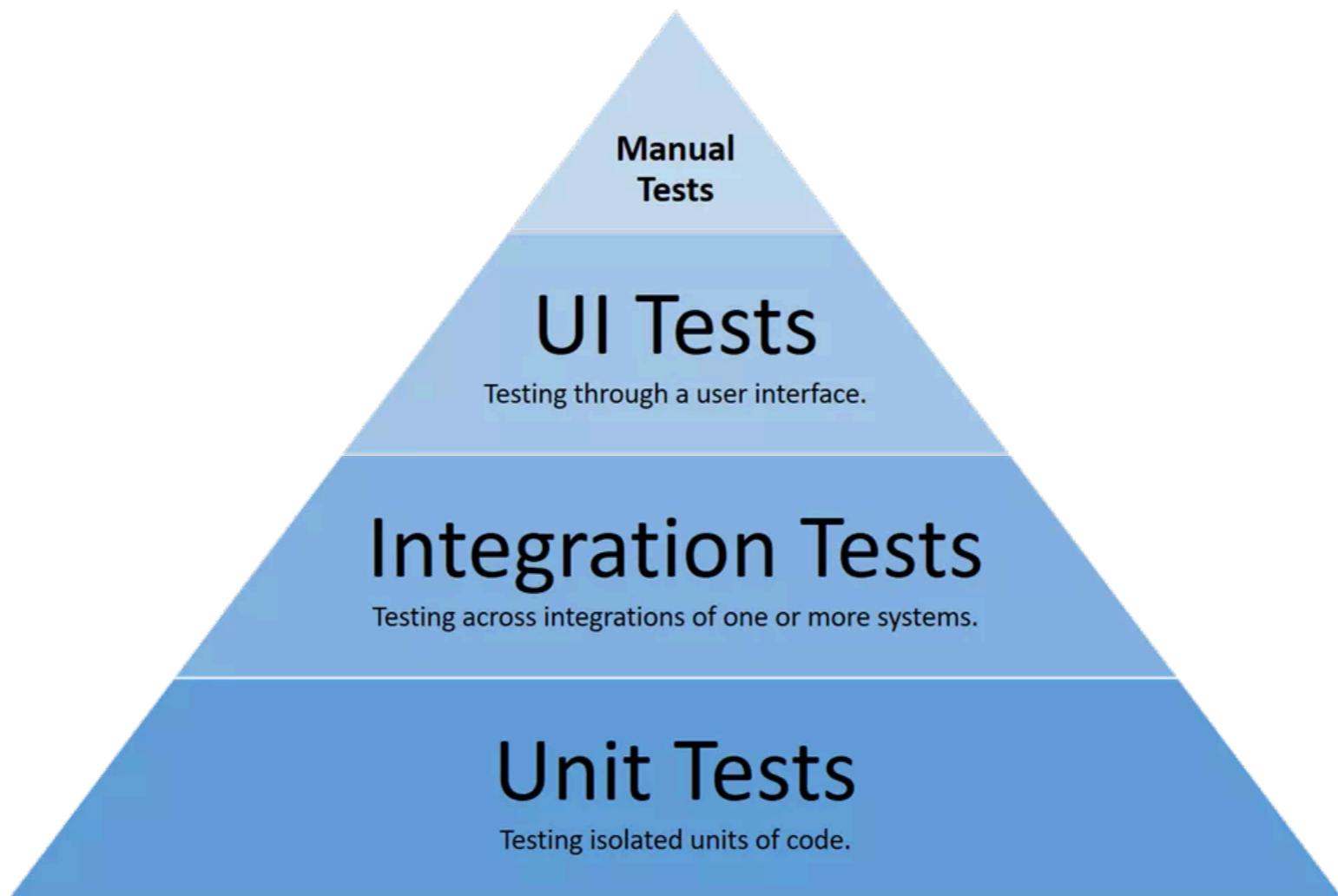
Automated the build  
Automated environment for builds



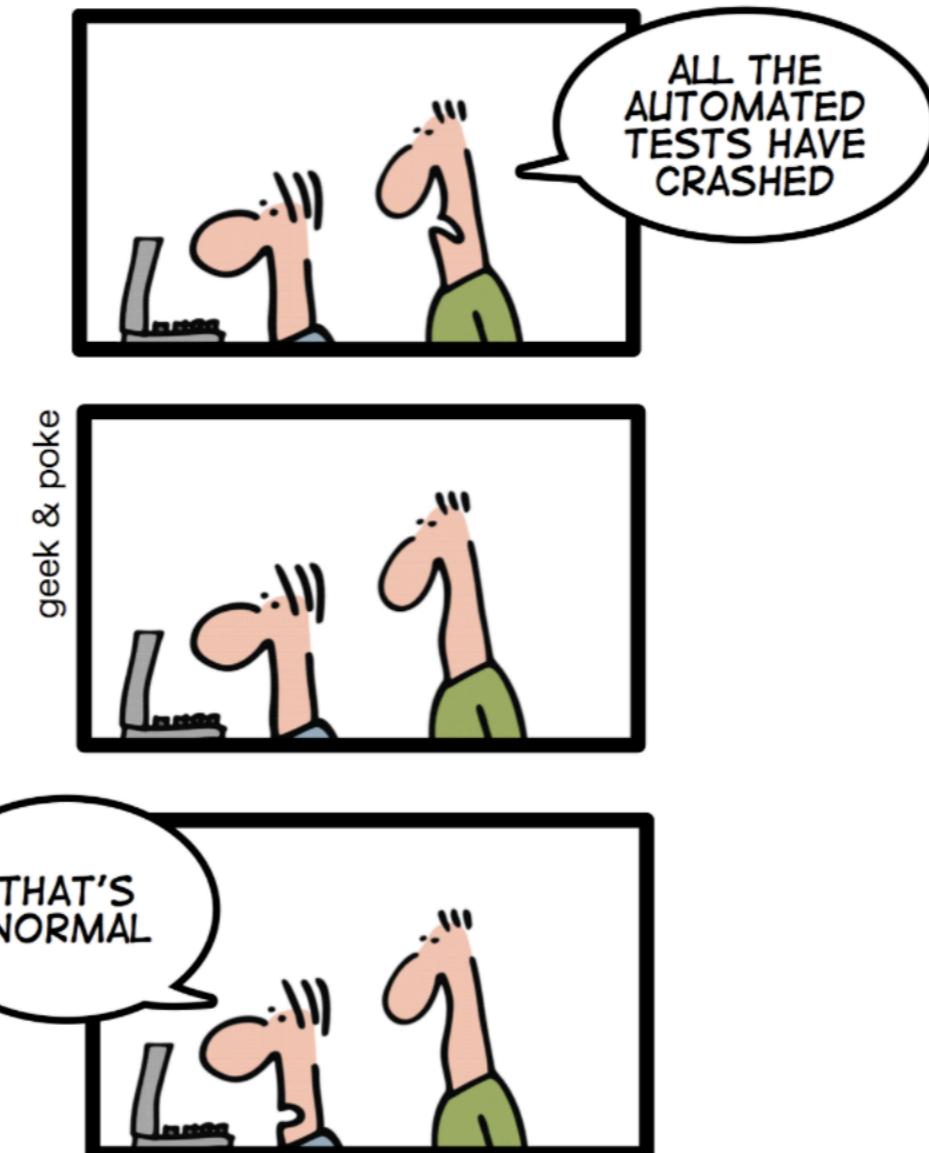
# Practice 3

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/>

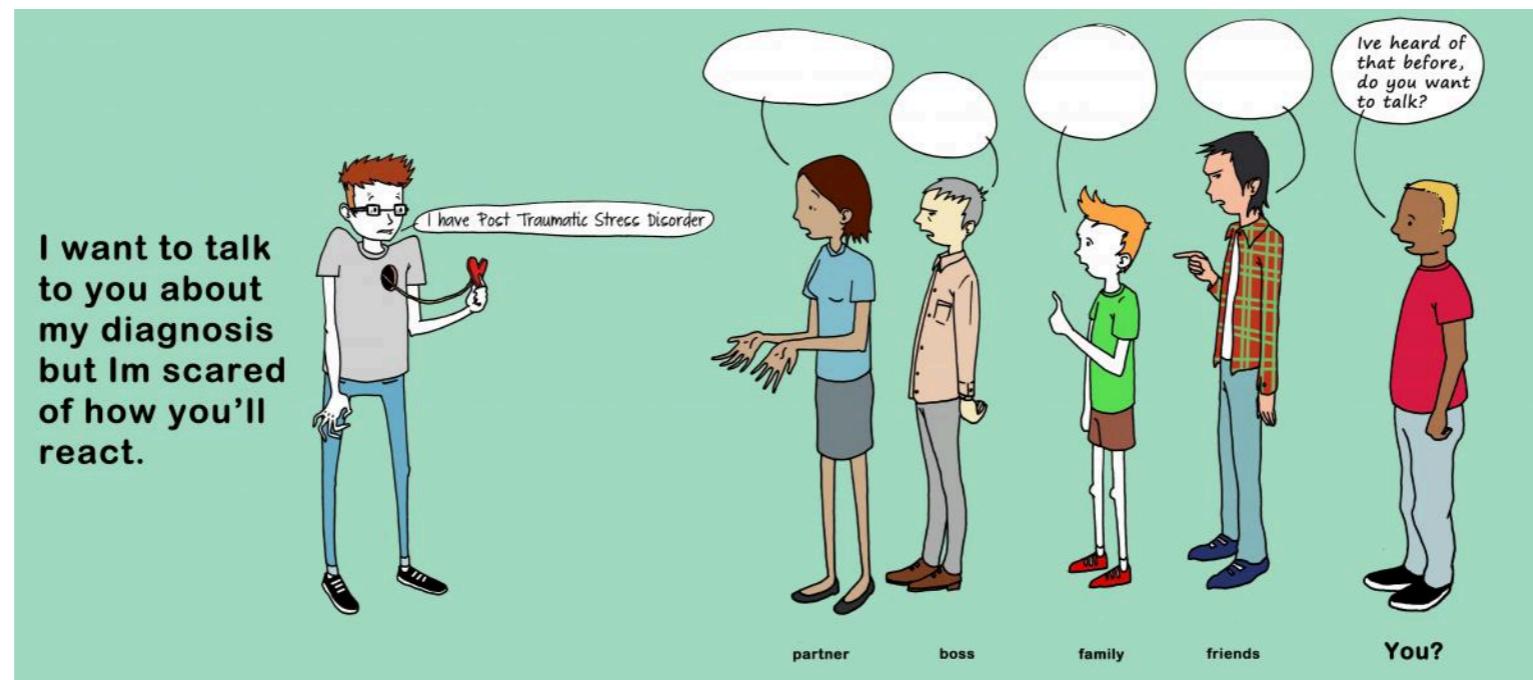


# Practice 4

**Everyone commits to the mainline everyday**

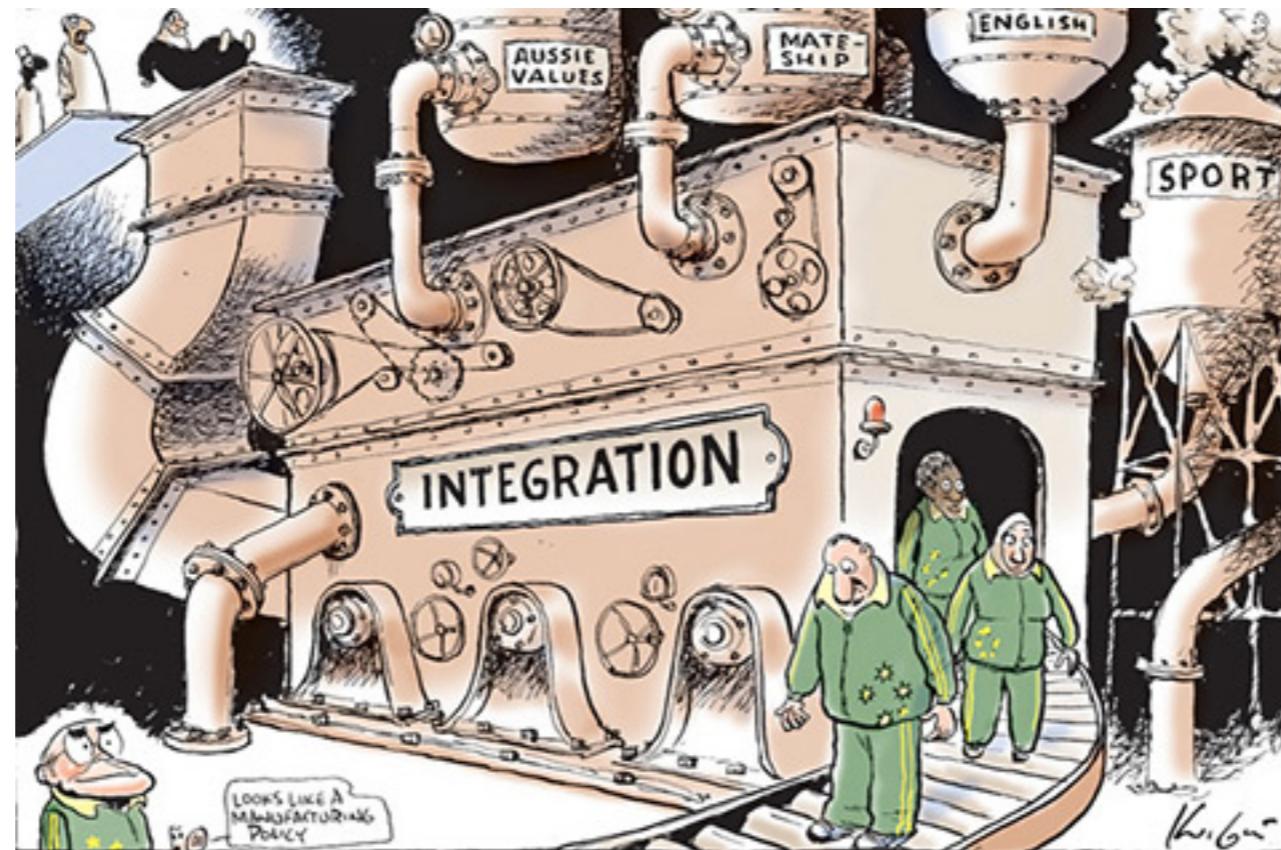
**Integration is about communication**

**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

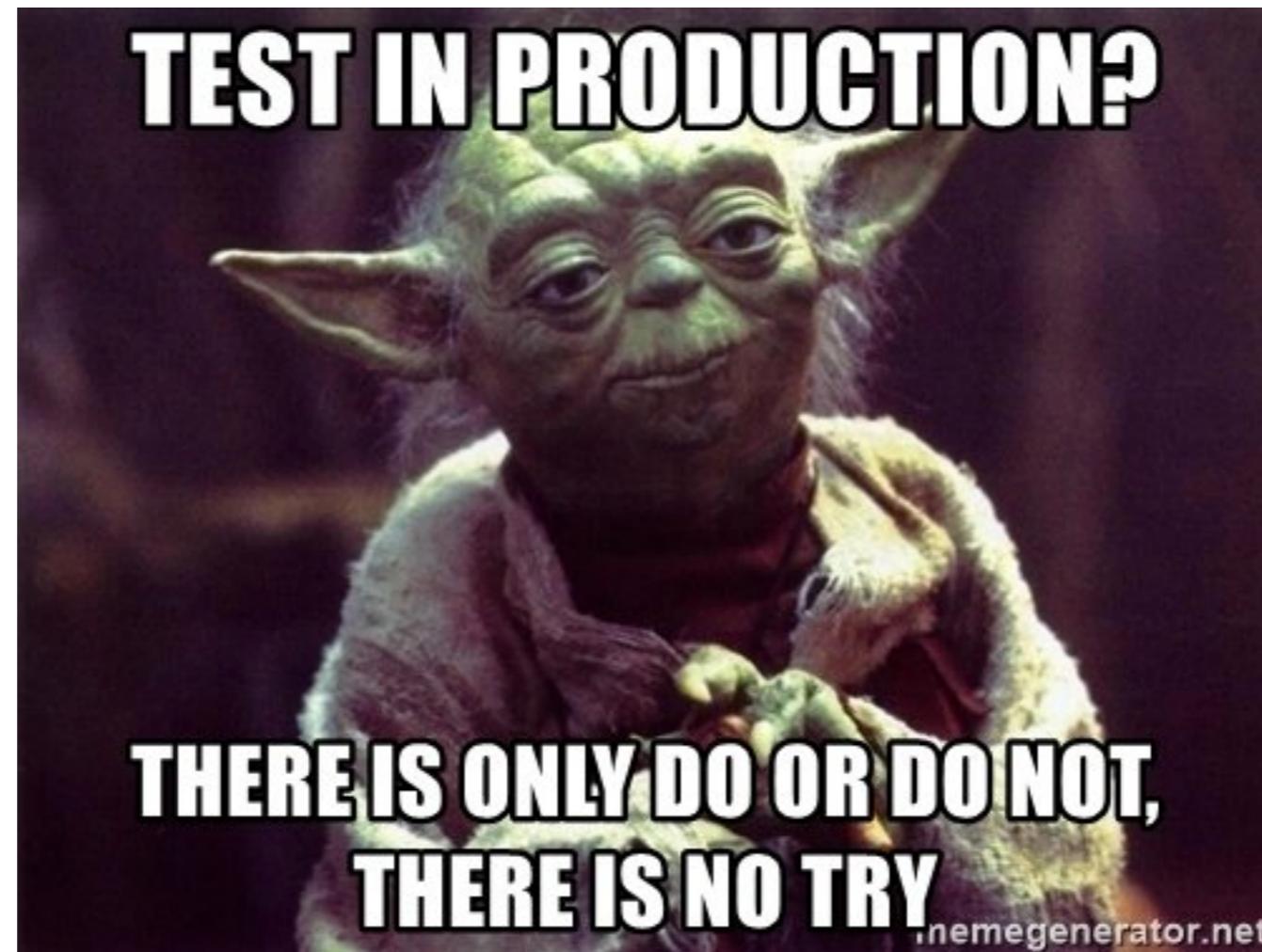
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

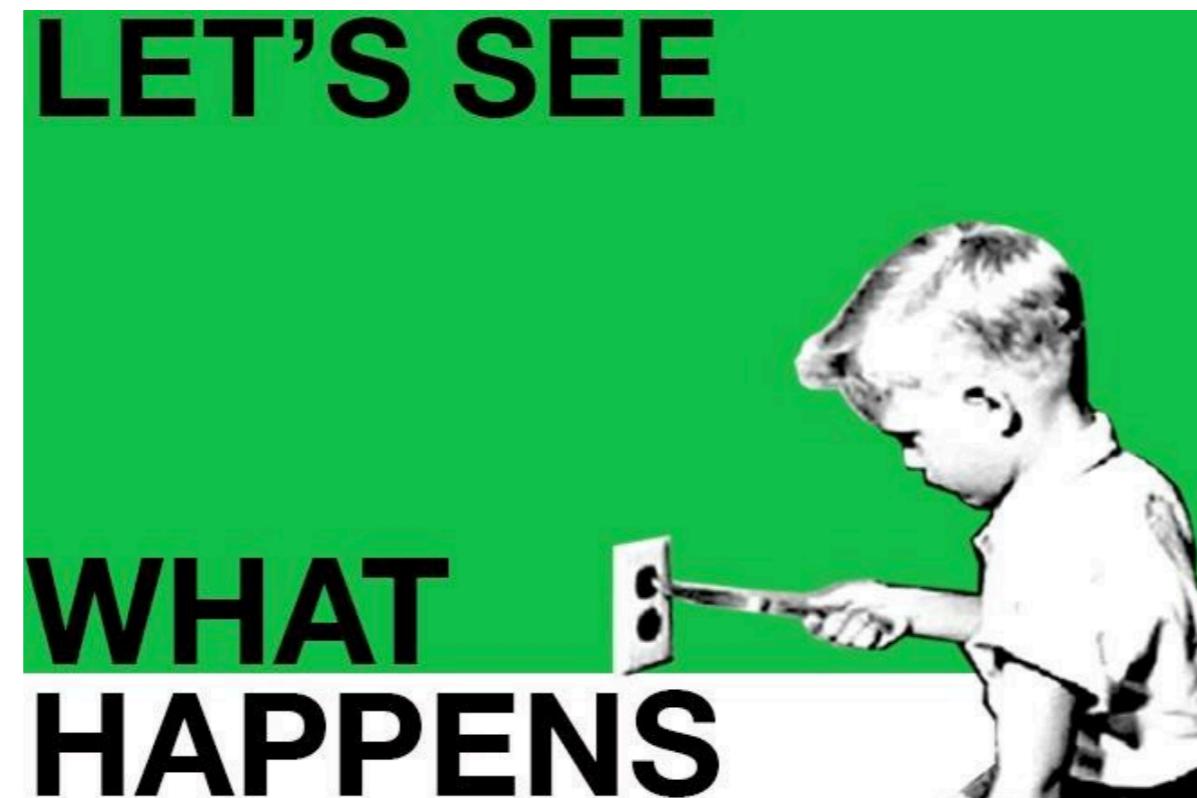
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



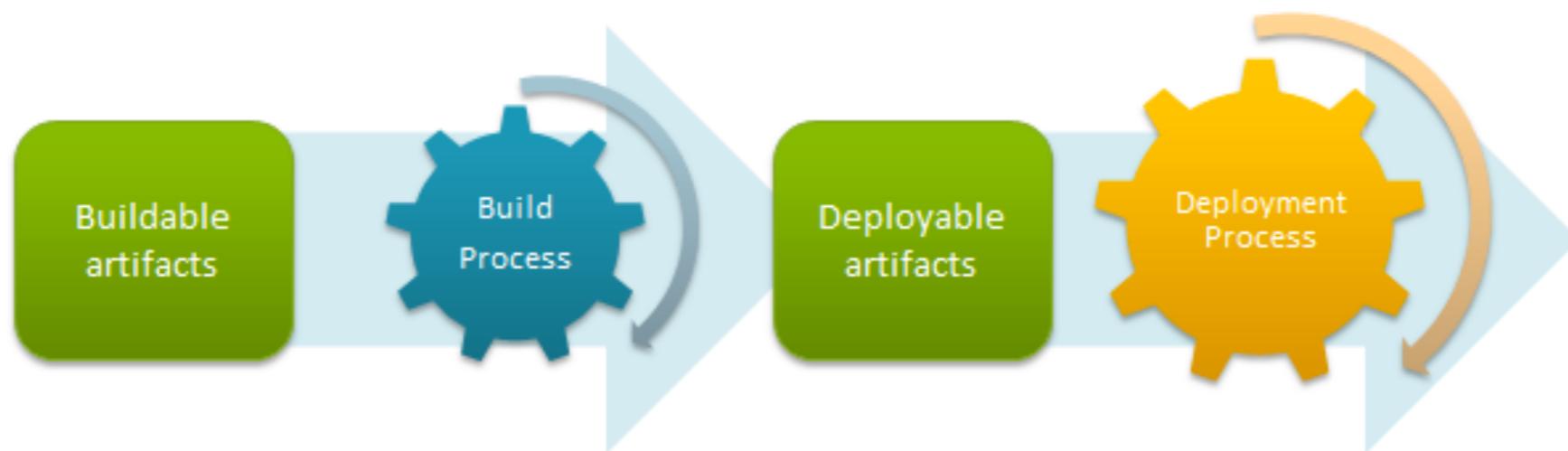
# Practice 10

**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臭屎 (stinky shit).

JUST SAY NO!



## 2. Choose Branch strategy

Main only

Development isolation

Feature isolation

Release isolation

Service and Release isolation

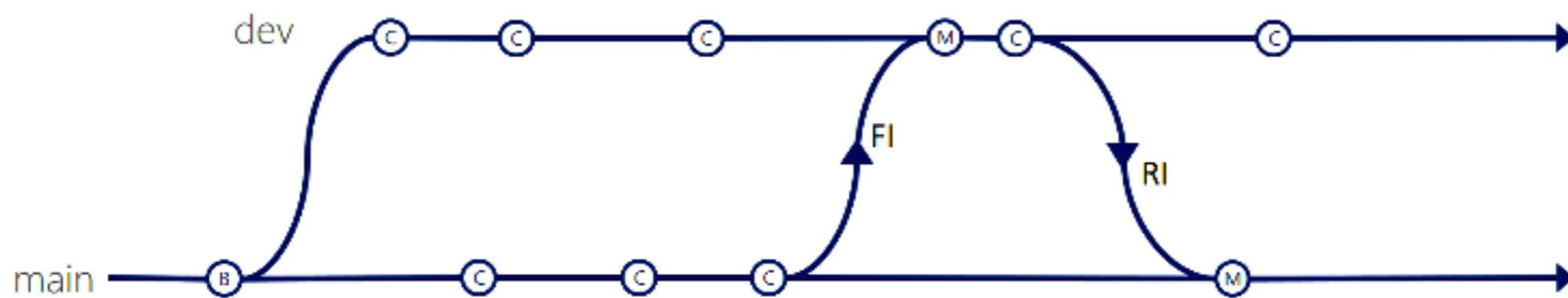
Service, Hotfix and Release isolation



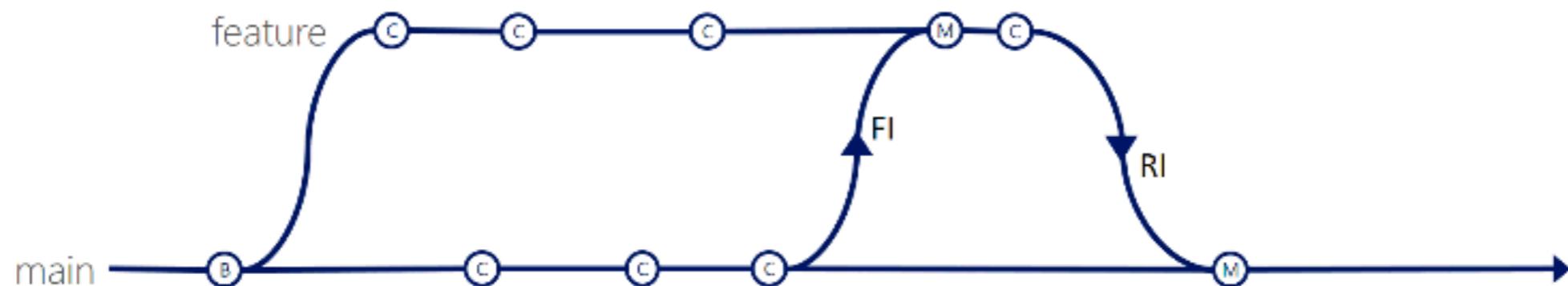
# Main only



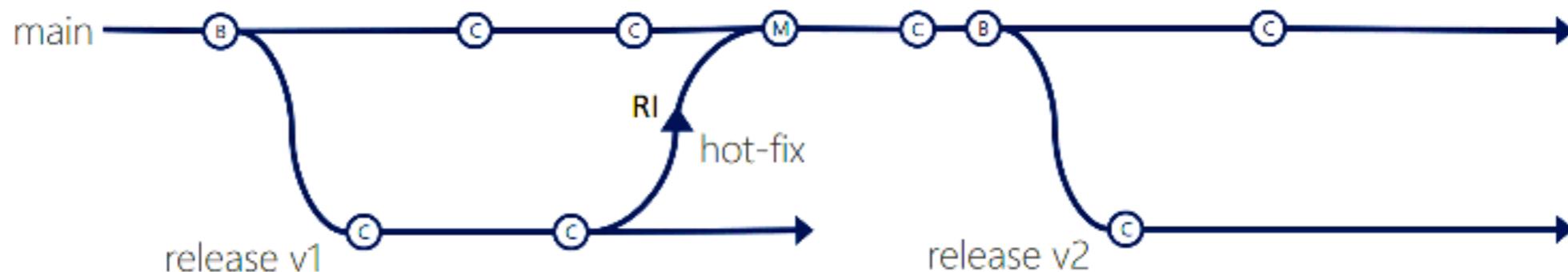
# Development isolation



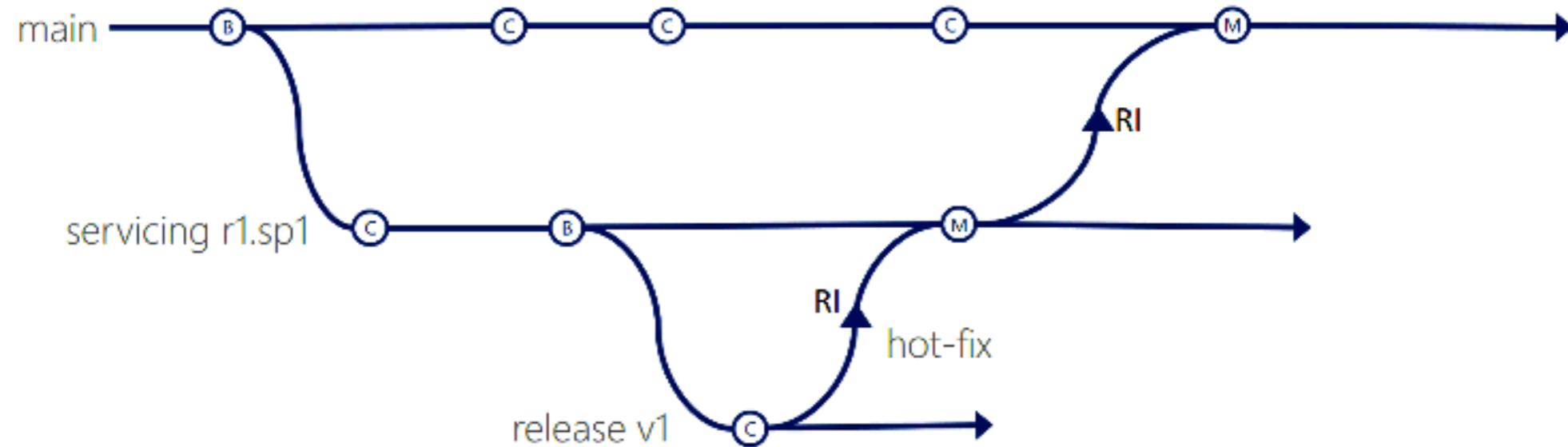
# Feature isolation



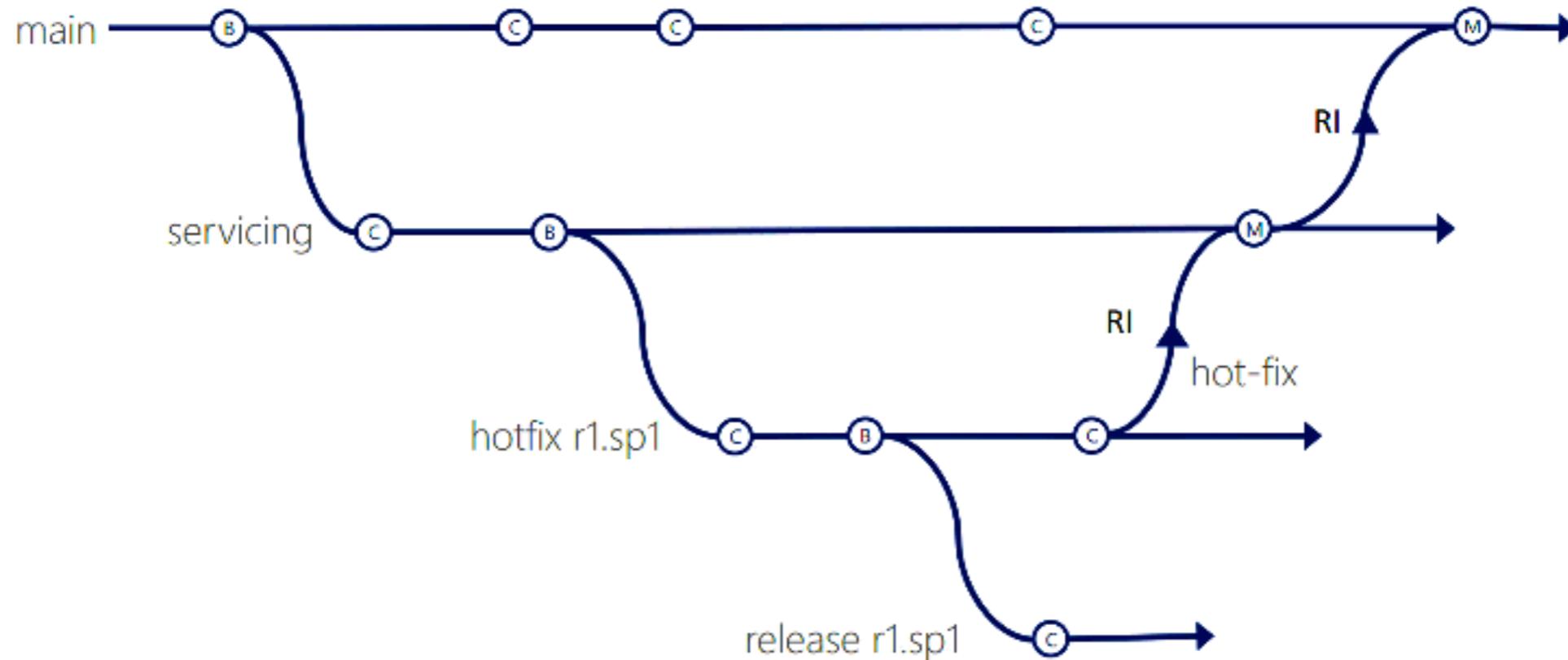
# Release isolation



# Service and Release isolation



# Service, Hotfix, Release isolation



# Validate, Validate and Validate

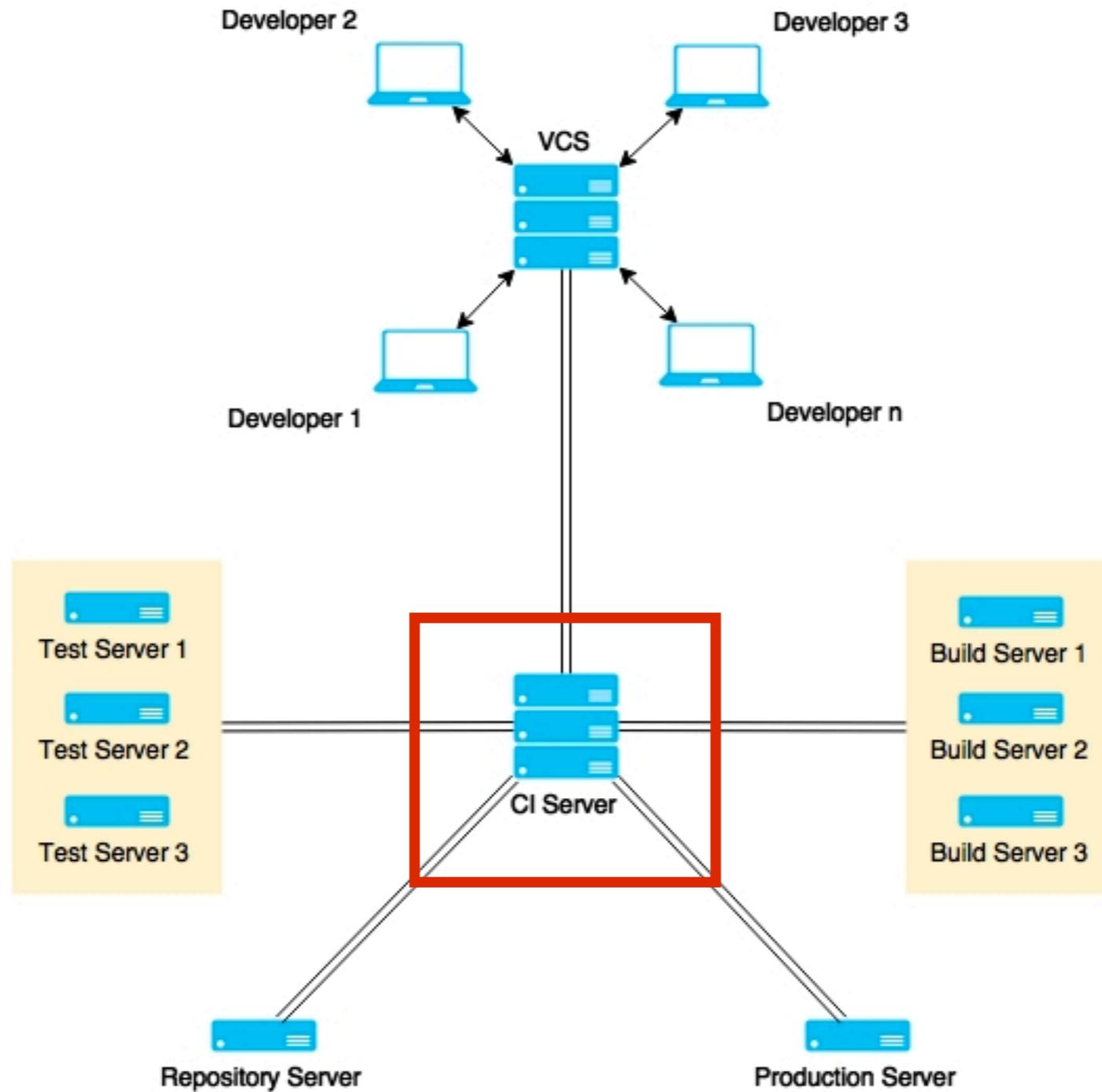


# Suggestion

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”**



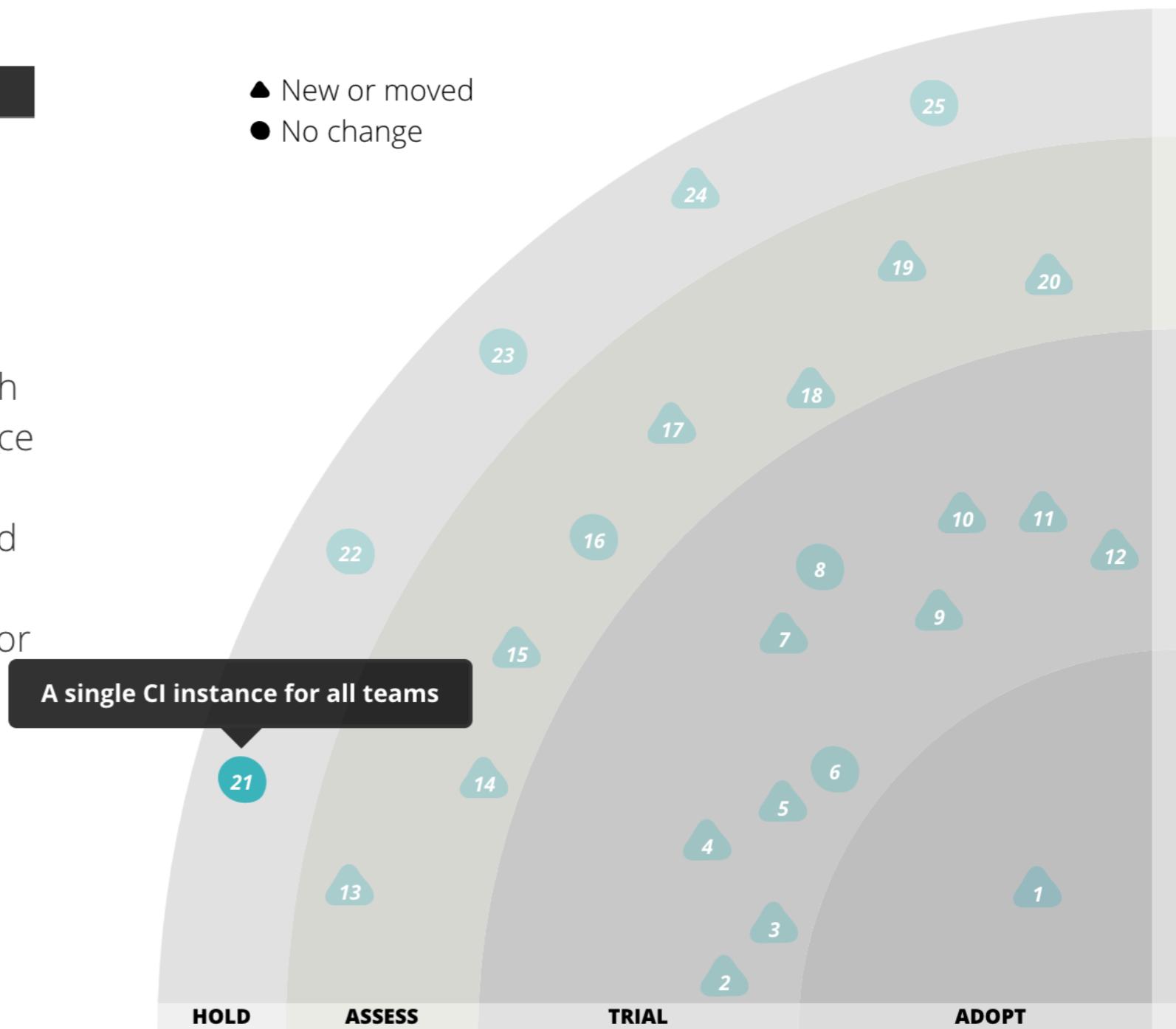
# Anti-pattern for CI Server

● 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

cloudbees



OSL  
OPEN SOURCE LAB

rackspace



redhat.

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

Atlassian

Datadog

JFrog

Mac Cloud

PagerDuty

XMission

<https://jenkins.io/>



# Hardware requirements

For Jenkins server

RAM +2GB

More CPU

More Disk



# Installation



# Jenkins in containers

Apache Tomcat

Jetty

JBoss

Websphere

WebLogic

Glassfish



# Download Jenkins

Blog Documentation Plugins Use-cases ▾ Participate Sub-projects ▾ Resources ▾



**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  
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/swpi/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

search [?](#) somkiat | log out

[ENABLE AUTO REFRESH](#)

New Item [add description](#)

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.

**Build Executor Status**  
1 Idle  
2 Idle

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 ...



# **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` 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 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` 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

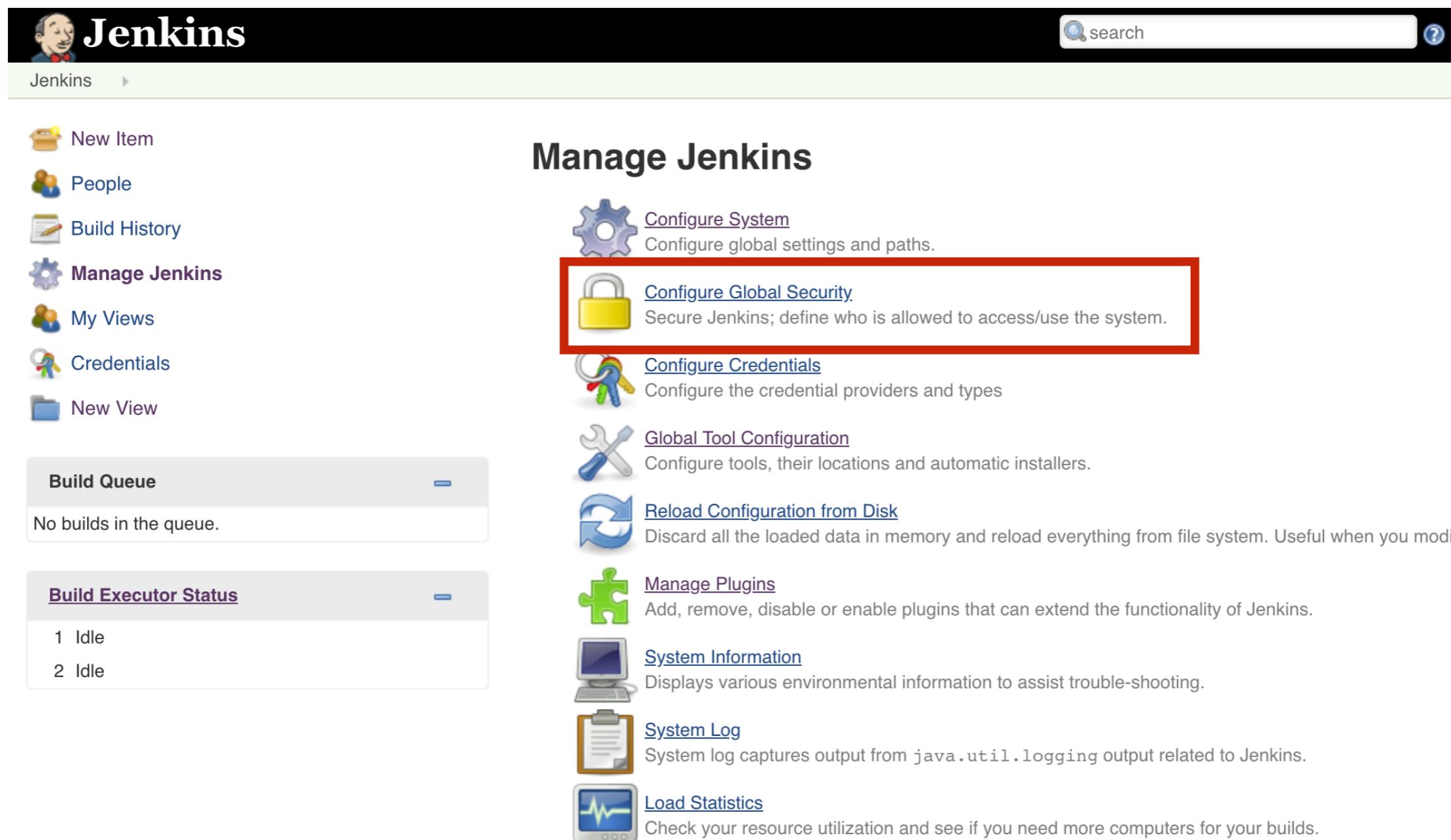
Environment variables

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' (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 content area is titled 'Manage Jenkins' and lists 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' (clipboard icon), and 'Load Statistics' (ECG icon). The 'Configure Global Security' item is described as 'Secure Jenkins; define who is allowed to access/use the system.'

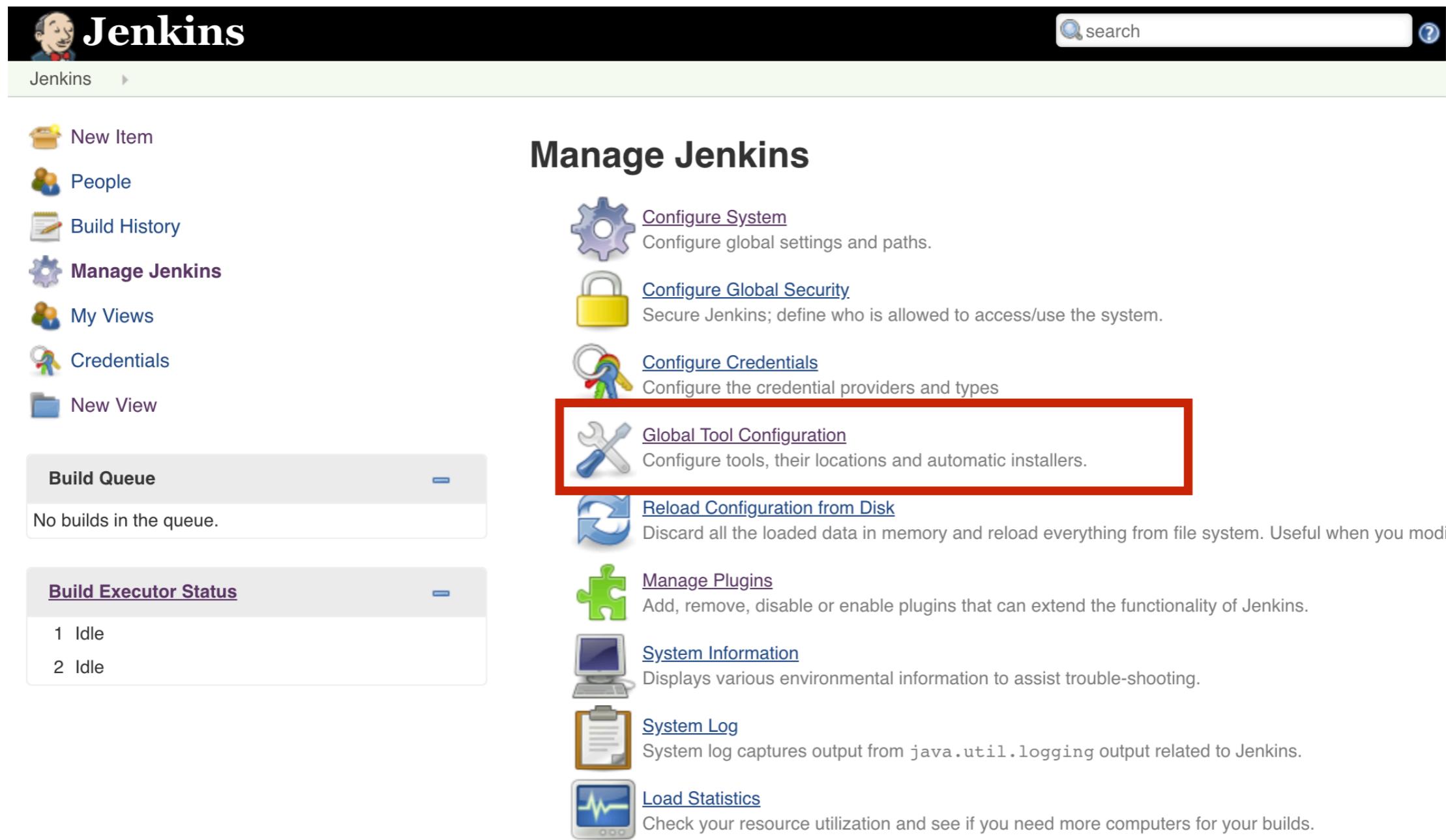
**Manage Jenkins**

-  [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` 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 and highlighted in purple), My Views, Credentials, and New View. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 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 (which is highlighted with a red box), Reload Configuration from Disk, Manage Plugins, 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 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` 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 purple), My Views, Credentials, and New View. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The main content area is titled "Manage Jenkins" and contains several configuration links: 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:

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

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

Filter:

Updates Available Installed Advanced

Install ↓

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

[Install without restart](#) [Download now and install after restart](#)

Update information obtained: 9 hr 37 min ago [Check now](#)



# 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

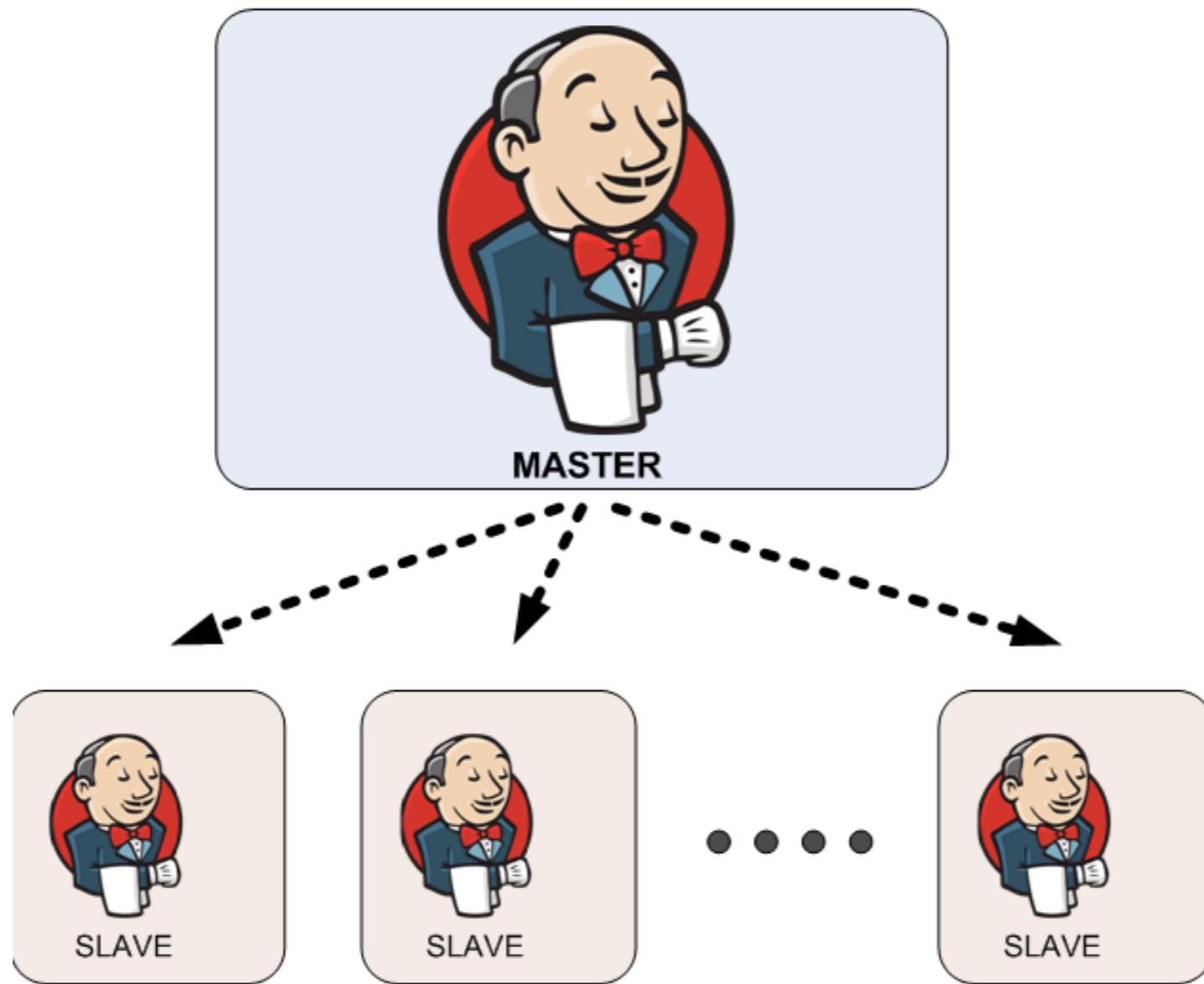
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	<a href="#">master</a>	Mac OS X (x86_64)	In sync	5.73 GB	463.00 MB	5.73 GB	0ms
	Data obtained	36 min	36 min	36 min	36 min	36 min	36 min

Refresh status



# Manage Nodes

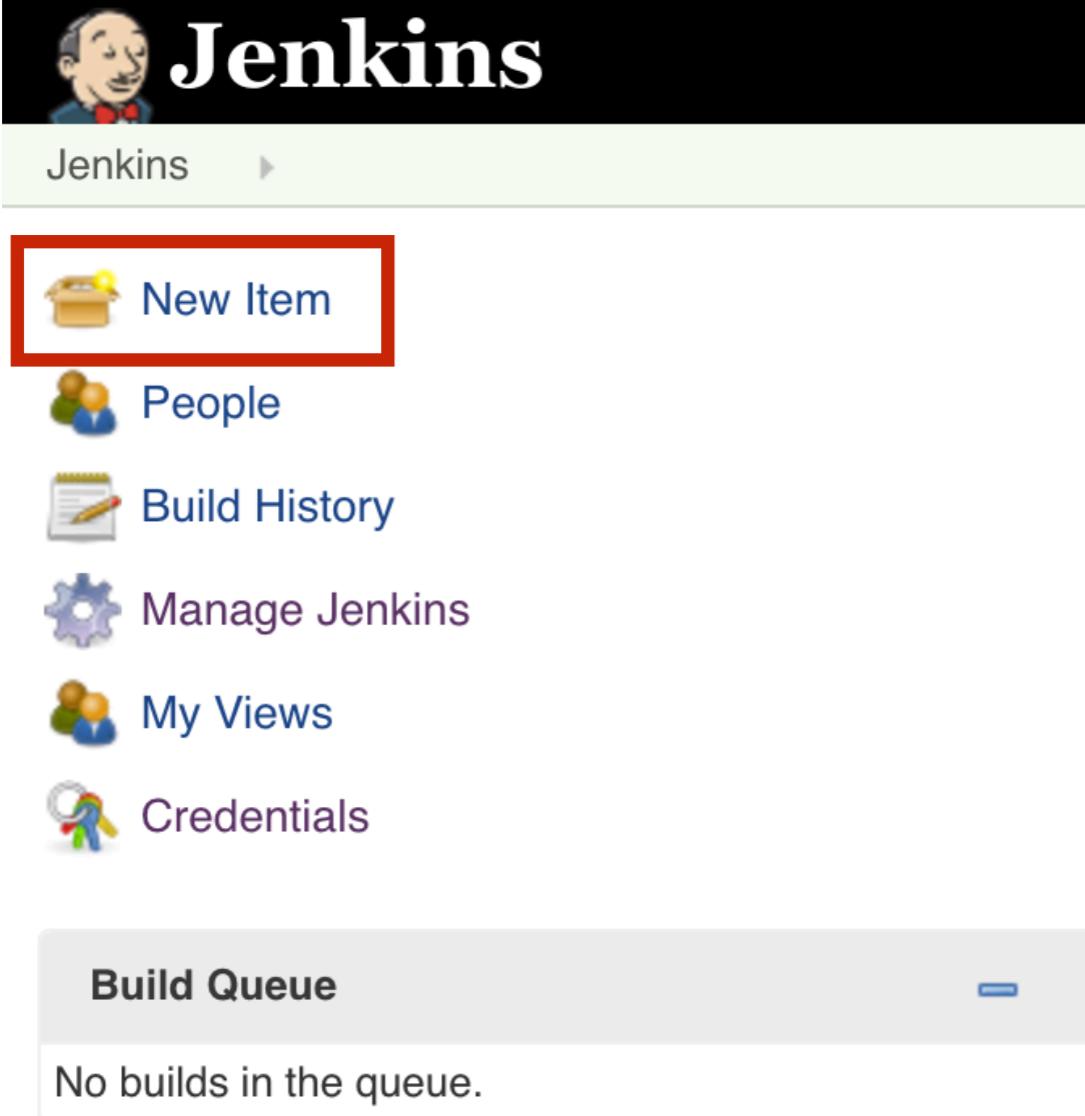
Master-slave concept to scale Jenkins



# 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 right-pointing arrow. The main content area has a large "Welcome to Jenkins!" message. To the left is a sidebar with links: "New Item" (highlighted with a red border), "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.



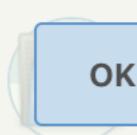
### External Job

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.



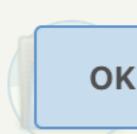
## External Job

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

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

General      Source Code Management      Build Triggers      Build Environment

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

Display Name

- 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 by a red border. Below it, the 'Source Code Management' section displays three options: 'None' (selected), 'Git', and 'Subversion'. Under the 'Build Triggers' section, several triggers are listed, each with 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. Build trigger

When to run this job

The screenshot shows a Jenkins configuration page with a tab navigation bar at the top. The tabs are 'General', 'Source Code Management', 'Build Triggers' (which is selected and highlighted in bold), and 'Build Environment'. Below the tabs, the page title is 'Build Triggers'. A vertical scroll bar is visible on the left side of the main content area. The content area lists five triggers, 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



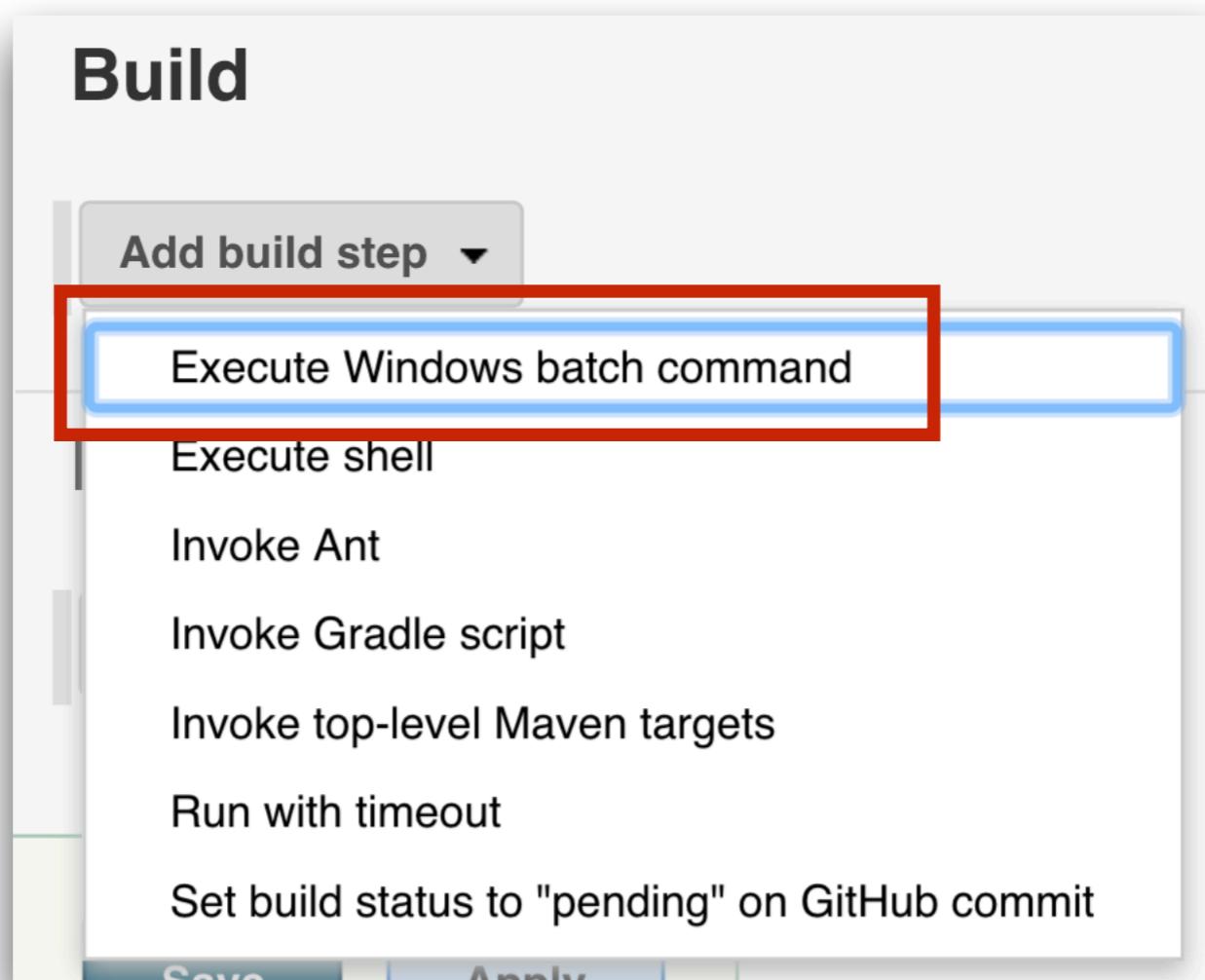
# 7. Add build step

What to run this job

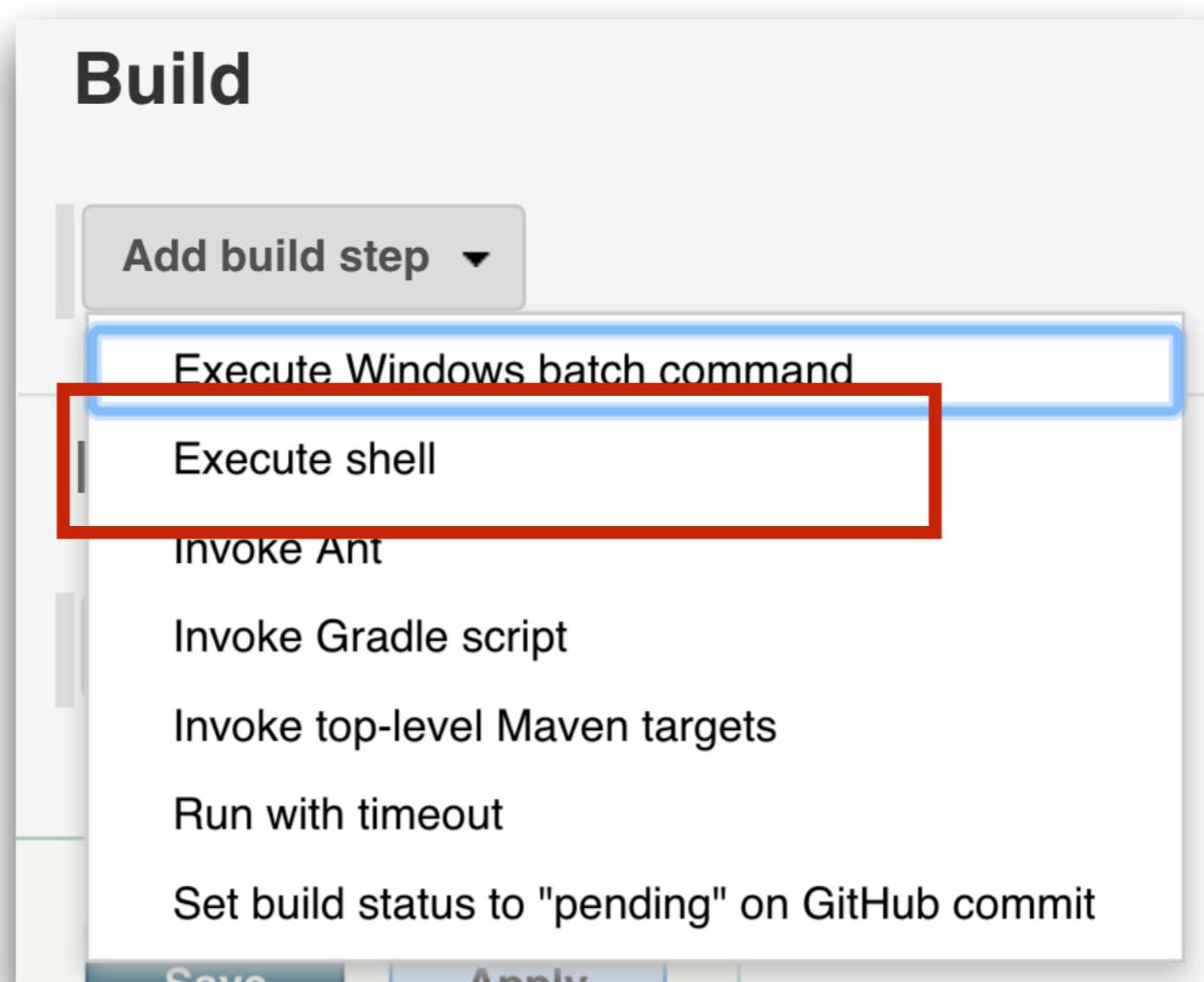
The screenshot shows the Jenkins job configuration interface. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, **Build**, and Post-build Actions. The **Build** tab is selected. Below the tabs, there are two sections: **Build Environment** and **Build**. The **Build** section contains a button labeled "Add build step ▾". A dropdown menu is open, listing several options: Execute Windows batch command (which is highlighted with a blue border), 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

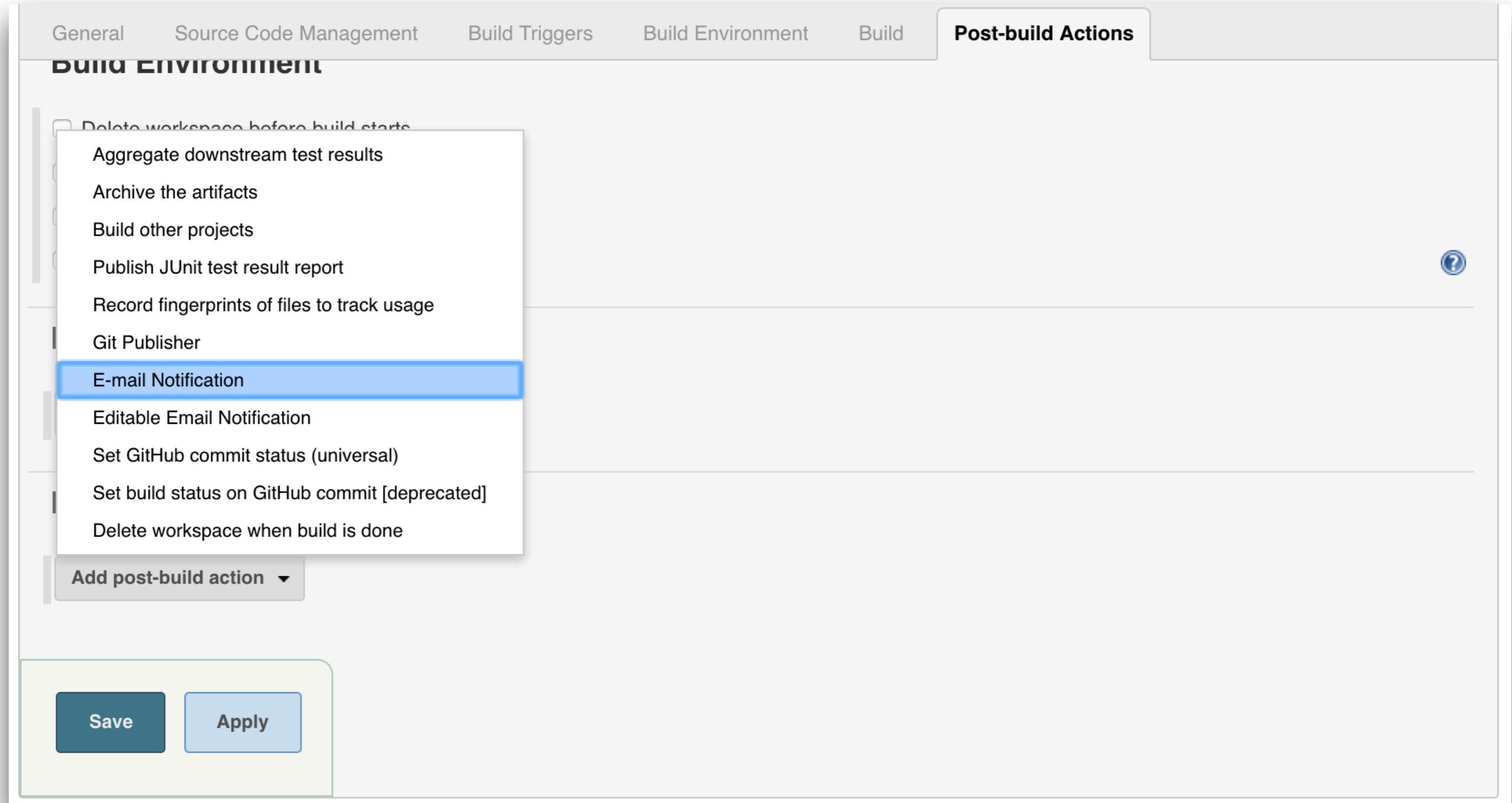
Generate reports

Send email

Run other jobs/projects



# 8. Post build actions

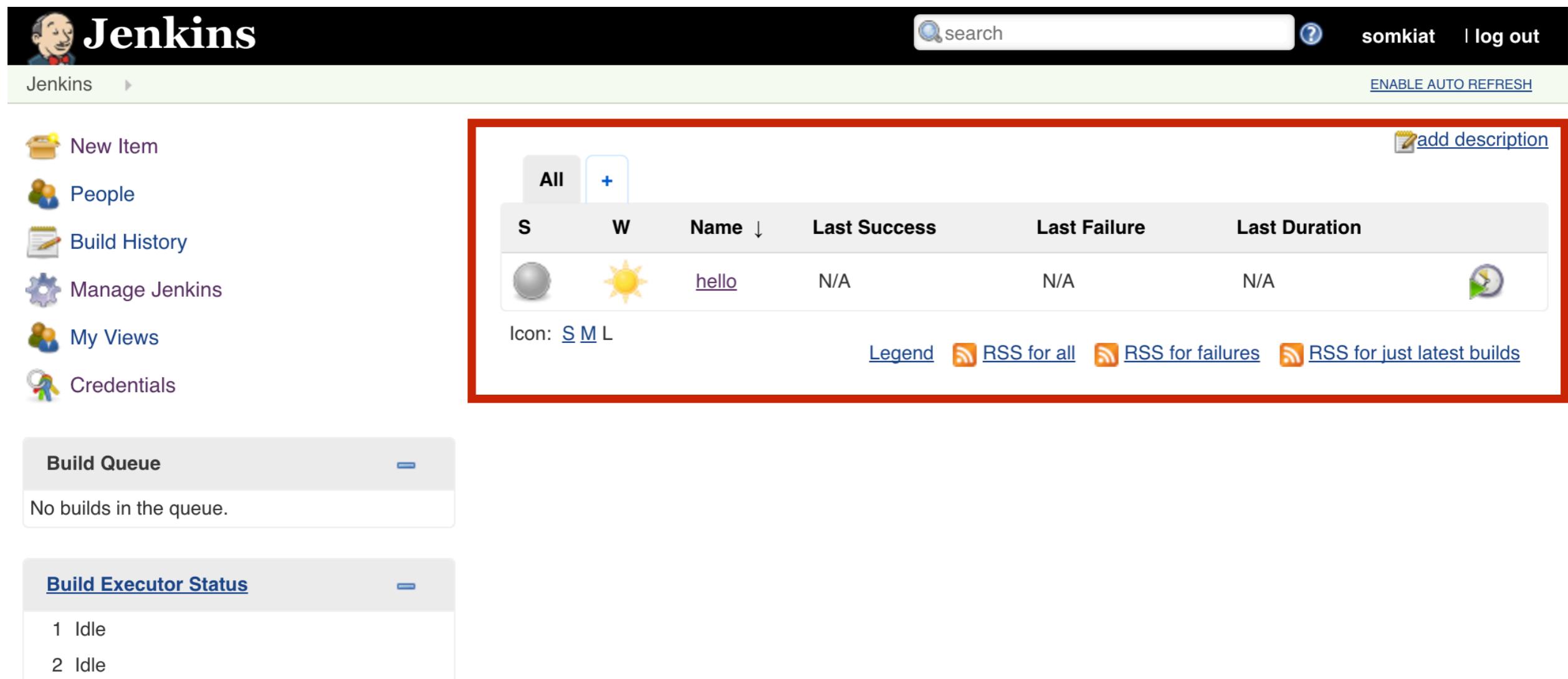


The screenshot shows the Jenkins configuration interface for a job. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. The Post-build Actions tab is selected, highlighted in dark blue. Below the tabs, the title "BUILD ENVIRONMENT" is displayed in bold black capital letters. A sidebar on the left lists several actions: Delete workspace before build starts, 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 (which is currently selected and highlighted in blue), Editable Email Notification, Set GitHub commit status (universal), Set build status on GitHub commit [deprecated], and Delete workspace when build is done. At the bottom of the sidebar is a button labeled "Add post-build action ▾". In the bottom right corner of the sidebar, there are two buttons: "Save" and "Apply".



# 9. Run your job

## Manual and Scheduler run



The screenshot shows the Jenkins dashboard with a red box highlighting the main 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.

**Job List:**

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">hello</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

Jenkins search somkiat | log out

Jenkins ▶ ENABLE AUTO REFRESH

New Item add description

People

Build History

Manage Jenkins

My Views

Credentials

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
Grey	Sun	<a href="#">hello</a>	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 the 'hello' project. At the top, there's a navigation bar with 'Jenkins' and 'hello'. Below it is the main content area for 'Project hello'. On the left, there's a sidebar with several options: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now' (which is highlighted with a red box), 'Delete Project', and 'Configure'. To the right of the sidebar, there are two links: 'Workspace' and 'Recent Changes'. The 'Build Now' button is the focal point of the image.



# 9. Run your job

See your job's output

Jenkins ➤ hello ➤ #1

Back to Project  
 Status  
 Changes  
**Console Output** View as plain text

**Console Output**

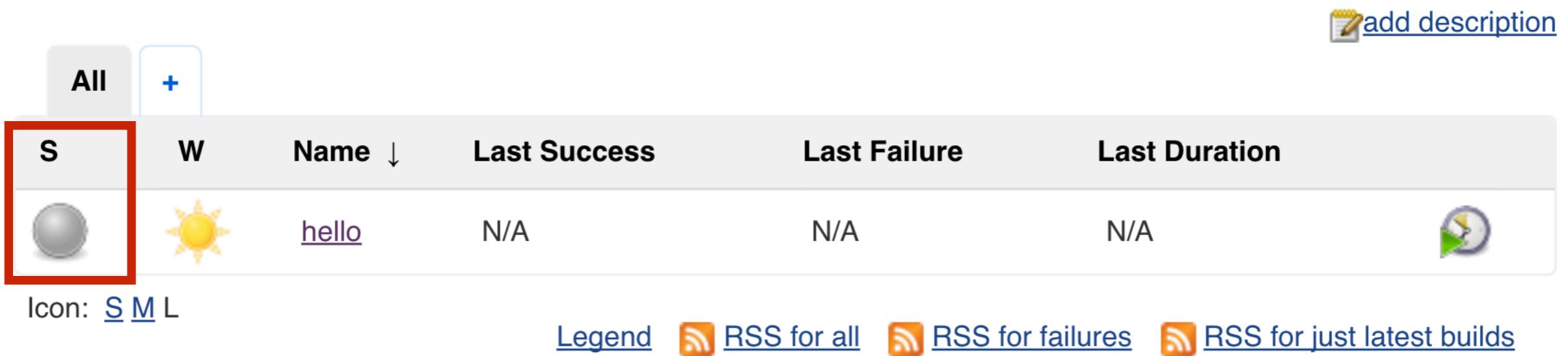
Started by user [Somkiat Puisungnoen](#)  
Building in workspace /Users/somkiat/Downloads/set/workspace/hello  
Finished: SUCCESS

Edit Build Information  
 Delete Build  
 Next Build



# 10. See job's status

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



The screenshot shows a Jenkins job status table. At the top, there are buttons for 'All' and '+'. On the right, there is a 'add description' button with a pencil icon. The table has columns: 'Name' (sorted by name), 'Last Success', 'Last Failure', and 'Last Duration'. A single row is shown for a job named 'hello'. The 'Name' column contains a yellow sun icon, 'Last Success' says 'N/A', 'Last Failure' says 'N/A', and 'Last Duration' has a small green and yellow icon. Below the table, it says 'Icon: S M L' with links to 'Legend', 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'. The 'S' icon in the legend is highlighted with a red border.

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](#)

**Blue = build success**

**Red = build failure**

**Gray = disabled/never executed**



# 11. See job's health

[add description](#)

All[+](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">hello</a>	N/A	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

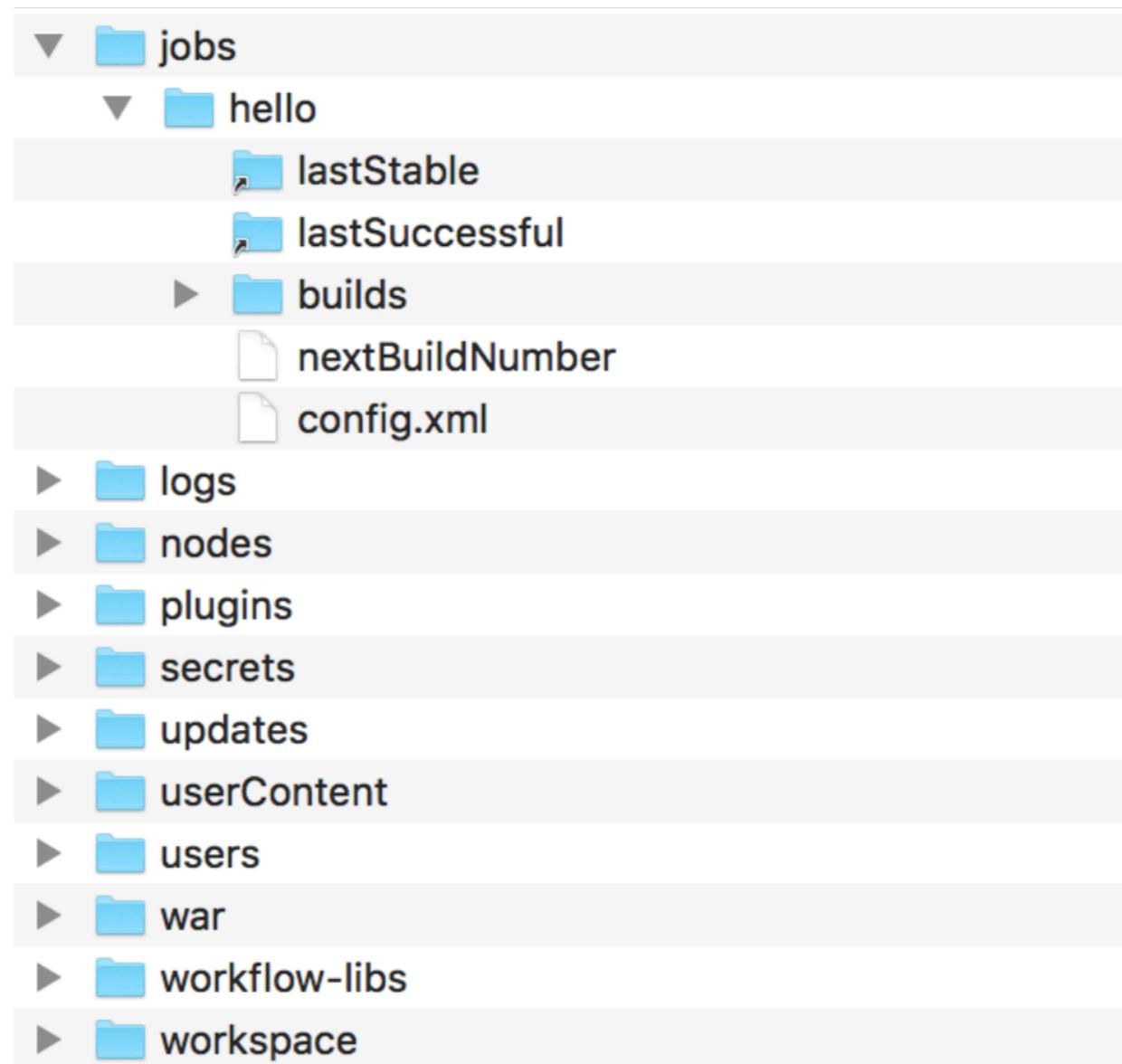


# Backup your Jenkins



# Backup your Jenkins

All data that must be backup



# Backup your Jenkins

What to backup ?

When to backup ?

How to backup ?



# Backup your Jenkins

Try to create new Jenkins's job to backup  
How to backup ?



# We use 7zip to compress files



## 7-Zip

**7-Zip** is a file archiver with a high compression ratio.

**Download 7-Zip 18.01 (2018-01-28) for Windows:**

Link	Type	Windows	Size
<a href="#">Download</a>	.exe	32-bit x86	1 MB
<a href="#">Download</a>	.exe	64-bit x64	1 MB

## License

**7-Zip** is **free software** with **open source**. The most of the code is under the [LGPLv2.1](#) license, under the BSD 3-clause License. Also there is unRAR license restriction for some parts of the code.

Read [7-Zip License](#) information.

You can use 7-Zip on any computer, including a computer in a commercial organization, without paying any fees or royalties to the authors or copyright holders of 7-Zip.

<http://www.7-zip.org/>



# Write your backup script



# For Windows

## Windows

REM Store the current date inside a variable named "DATE"

```
for /f %%i in ('date /t') do set DATE=%%i
```

REM 7-Zip command to create an archive

```
"C:\Program Files\7-Zip\7z.exe" a -t7z  
C:\Jenkins_Backup\Backup_%DATE%.7z C:\Jenkins\*
```



# For Linux/Mac

Mac OS

```
for /f %%i in ('date /t') do set DATE=%%i  
7z a -t7z $BACKUP\Backup_$DATE.7z $JENKINS_HOME\*
```



# Ready to start with Jenkins



# 1. Create a new job

Job name = backup\_jenkins

**Enter an item name**

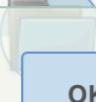
» Required field

 **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.

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

**OK**



# 2. Add build trigger

Using Build periodically => H 23 \* \* 7

The screenshot shows the Jenkins configuration interface for a job. The top navigation bar includes tabs for General, Source Code Management, Build Triggers (which is selected), Build Environment, Build, and Post-build Actions. The Build Triggers section is titled "Build Triggers" and contains three options: "Trigger builds remotely (e.g., from scripts)", "Build after other projects are built", and "Build periodically". The "Build periodically" option is checked, and its schedule is set to "H 23 \* \* 7". A note below the schedule states: "Would last have run at Sunday, June 11, 2017 11:40:39 PM ICT; would next run at Sunday, June 18, 2017 11:40:39 PM ICT." There are also sections for GitHub hook trigger for GITScm polling and Poll SCM, neither of which is selected.



# 3. Build

Choose way to run a script

The screenshot shows a dropdown menu titled "Add build step" with a downward arrow. The menu lists several options: "Execute Windows batch command" (which is highlighted with a blue background), "Execute shell", "Invoke Ant", "Invoke Gradle script", "Invoke top-level Maven targets", "Run with timeout", and "Set build status to "pending" on GitHub commit".

- Add build step ▾
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit



# 3. Build

Copy and paste your script to Jenkins

Build

**Execute shell**

Command

```
BACKUP=/Users/somkiat/data/slide/ci-cd/swpark/software/backup
DATE=`date +%Y-%m-%d`
7z a -t7z $BACKUP/Backup_${DATE}.7z ${JENKINS_HOME}/*
```

See [the list of available environment variables](#)

**Advanced...**

Add build step ▾



# 4. Save and run

The screenshot shows the Jenkins interface for a project named "backup\_jenkins". The top navigation bar includes the Jenkins logo, a search bar, and a help icon. The left sidebar contains links: "Back to Dashboard", "Status", "Changes", "Workspace", "Build Now" (which is highlighted with a red border), "Delete Project", and "Configure". Below the sidebar is a summary card with "Build History" (yellow sun icon), a search bar containing "find", and RSS feed links for "RSS for all" and "RSS for failures". The main content area is titled "Project backup\_jenkins". It features a "Workspace" link with a folder icon and a "Recent Changes" link with a notepad icon. At the bottom, there is a section titled "Permalinks".



# 5. See console output

Jenkins Jenkins backup\_jenkins #2

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

## Console Output

```
Started by user somkiat
Building in workspace /Users/somkiat/data/slide/ci-cd/swpark/software/keep/workspace/backup_jenkins
[backup_jenkins] $ /bin/sh -xe /var/folders/t5/8kg23s_97z9dw44tfcl6dqw000gn/T/hudson8420145093817228191.sh
+ BACKUP=/Users/somkiat/data/slide/ci-cd/swpark/software/backup
++ date +%Y-%m-%d
+ DATE=2017-06-14
+ 7z a -t7z /Users/somkiat/data/slide/ci-cd/swpark/software/backup/Backup_2017-06-14.7z /Users/somkiat/data/slide/ci-cd/swpark/software/keep/config.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/hudson.model.UpdateCenter.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/hudson.plugins.emailext.ExtendedEmailPublisher.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/hudson.plugins.git.GitTool.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/hudson.tasks.Mailer.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/identity.key.enc /Users/somkiat/data/slide/ci-cd/swpark/software/keep/jenkins.CLI.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/jenkins.install.InstallUtil.installingPlugins /Users/somkiat/data/slide/ci-cd/swpark/software/keep/jenkins.install.InstallUtil.lastExecVersion /Users/somkiat/data/slide/ci-cd/swpark/software/keep/jenkins.install.UpgradeWizard.state /Users/somkiat/data/slide/ci-cd/swpark/software/keep/jobs /Users/somkiat/data/slide/ci-cd/swpark/software/keep/logs /Users/somkiat/data/slide/ci-cd/swpark/software/keep/nodeMonitors.xml /Users/somkiat/data/slide/ci-cd/swpark/software/keep/nodes /Users/somkiat/data/slide/ci-cd/swpark/software/keep/plugins /Users/somkiat/data/slide/ci-cd/swpark/software/keep/secret.key /Users/somkiat/data/slide/ci-cd/swpark/software/keep/secret.key.not-so-secret /Users/somkiat/data/slide/ci-cd/swpark/software/keep/secrets /Users/somkiat/data/slide/ci-cd/swpark/software/keep/updates /Users/somkiat/data/slide/ci-cd/swpark/software/keep/userContent /Users/somkiat/data/slide/ci-cd/swpark/software/keep/users /Users/somkiat/data/slide/ci-cd/swpark/software/keep/war /Users/somkiat/data/slide/ci-cd/swpark/software/keep/workflow-libs /Users/somkiat/data/slide/ci-cd/swpark/software/keep/workspace

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=utf8,Utf16=on,HugeFiles=on,64 bits,4 CPUs x64)

Scanning the drive:
703 folders, 2801 files, 203646776 bytes (195 MiB)

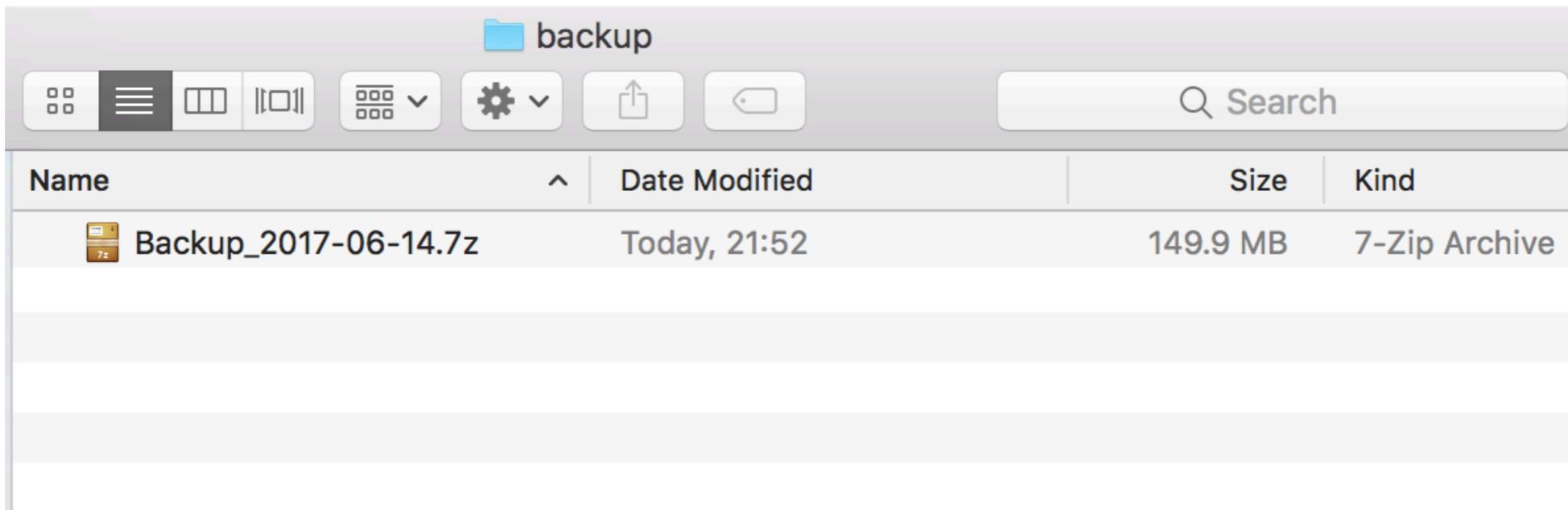
Creating archive: /Users/somkiat/data/slide/ci-cd/swpark/software/backup/Backup_2017-06-14.7z

Items to compress: 3504

Files read from disk: 2726
Archive size: 149922651 bytes (143 MiB)
Everything is Ok
Finished: SUCCESS
```



# 6. See your backup file



- 7. Upload file to somewhere ?**
- 8. Restore from backup file ?**



# Backup with plugin



# 1. Backup with plugin

Install plugin = Periodic Backup Plugin

The screenshot shows the Jenkins plugin store interface. A red circle labeled '1' is positioned above a search bar containing the text 'periodic backup'. Below the search bar, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A table lists available plugins, with one row highlighted for the 'Periodic Backup' plugin. The table has columns for 'Name' (Periodic Backup) and 'Version' (1.3). Below the table are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'. A status message indicates 'Update information obtained: 9 hr 52 min ago'.

Install ↓	Name	Version
<input type="checkbox"/> <a href="#">Periodic Backup</a>		1.3

Install without restart    Download now and install after restart    Check now

Update information obtained: 9 hr 52 min ago

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



# 2. Install plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/> Periodic Backup		1.3

**2**

**Install without restart**      Download now and install after restart      Update information obtained: 9 hr 52 min ago      Check now



# 3. Using plugin

Manage Jenkins -> Periodic Backup Manager



## Manage Users

Create/delete/modify users that can log in to this Jenkins



## Periodic Backup Manager

Periodically backup your Hudson data and save the day.



## In-process Script Approval

Allows a Jenkins administrator to review proposed scripts process and so could bypass security restrictions.



# 3. Using plugin

Try to configure backup plugin

The screenshot shows the Jenkins Periodic Backup Manager configuration page. At the top, there is a navigation bar with a Jenkins logo, a search bar, and user information (somkiat | log out). Below the navigation bar, the page title is "Periodic Backup Manager". On the left side, there is a sidebar with four buttons: "Back to Dashboard" (green arrow), "Restore" (blue arrow), "Backup Now!" (blue arrow), and "Configure" (gear icon). The main content area contains several configuration fields:

- Root Directory:** /Users/somkiat/data/slide/ci-cd/swpark/software/keep
- Temporary Directory:** (empty input field)
- Backup schedule (cron):** (empty input field)
- Validate cron syntax:** (button)
- Maximum backups in location:** 0
- Store no older than (days):** 0
- File Management Strategy:** (radio buttons for ConfigOnly and FullBackup, both unselected)
- Storage Strategy:** (button "Add Storage")
- Backup Location:** (button "Add Location")
- Save:** (button)



# 3. Using plugin

Temporary directory of achieve files while restore and backup process

Jenkins

Periodic Backup Manager

Back to Dashboard

Restore

Backup Now!

Configure

Root Directory /Users/somkiat/data/slide/ci-cd/swpark/software/keep

Temporary Directory

Backup schedule (cron)

Validate cron syntax

Maximum backups in location 0

Store no older than (days) 0

File Management Strategy

ConfigOnly

FullBackup

Storage Strategy

Add Storage

Backup Location

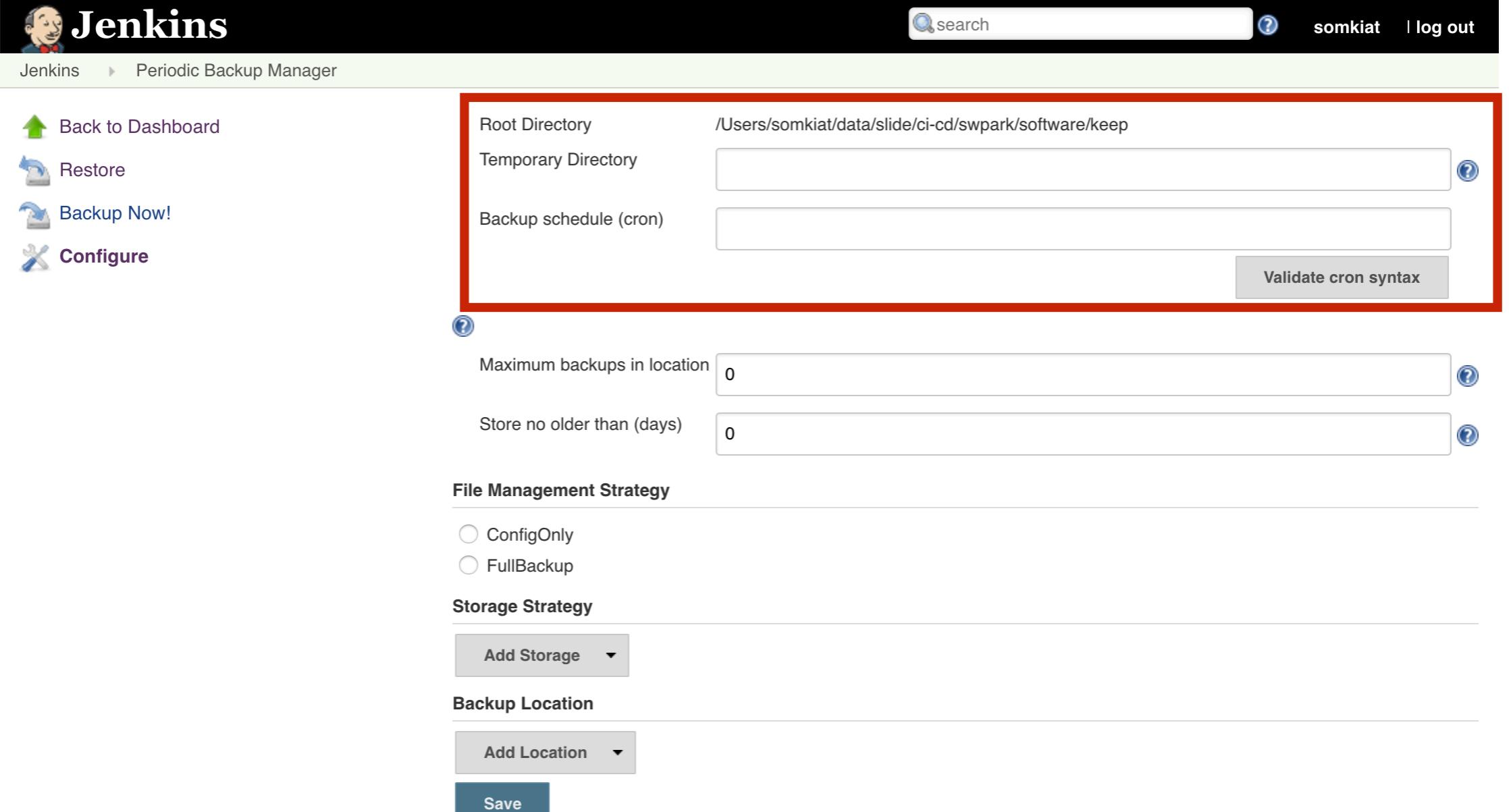
Add Location

Save



# 3. Using plugin

Backup schedule is a crontab format



Jenkins

Periodic Backup Manager

Back to Dashboard

Restore

Backup Now!

Configure

Root Directory: /Users/somkiat/data/slide/ci-cd/swpark/software/keep

Temporary Directory:

Backup schedule (cron):

Validate cron syntax

Maximum backups in location: 0

Store no older than (days): 0

File Management Strategy

ConfigOnly

FullBackup

Storage Strategy

Add Storage

Backup Location

Add Location

Save



# 3. Using plugin

Maximum backups in location is total # of backups

Jenkins

Periodic Backup Manager

Back to Dashboard | Restore | Backup Now! | Configure

Root Directory: /Users/somkiat/data/slide/ci-cd/swpark/software/keep

Temporary Directory:

Backup schedule (cron):

Validate cron syntax

Maximum backups in location: 0

Store no older than (days): 0

File Management Strategy

Storage Strategy

Add Storage

Backup Location

Add Location

Save



# 3. Using plugin

Store no older than (days), delete backups < days

Jenkins

Periodic Backup Manager

Back to Dashboard | Restore | Backup Now! | Configure

Root Directory: /Users/somkiat/data/slide/ci-cd/swpark/software/keep

Temporary Directory:

Backup schedule (cron):

Validate cron syntax

Maximum backups in location: 0

Store no older than (days): 0

File Management Strategy

Storage Strategy

Add Storage

Backup Location

Add Location

Save



# 4. Example of configuration

Root Directory /Users/somkiat/data/slide/ci-cd/swpark/software/keep

Temporary Directory  ?

Backup schedule (cron)

This cron is OK

Validate cron syntax



Maximum backups in location  ?

Store no older than (days)  ?



# 5. File management strategy

**ConfigOnly** is backup only configuration

## File Management Strategy

---

- ConfigOnly
- FullBackup



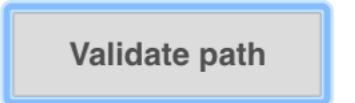
# 5. Backup locations

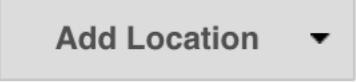
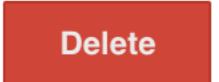
## Backup Location

 LocalDirectory

Backup directory path  

Enable this location 

directory "/Users/somkiat/backup" OK 



# Try to run your job



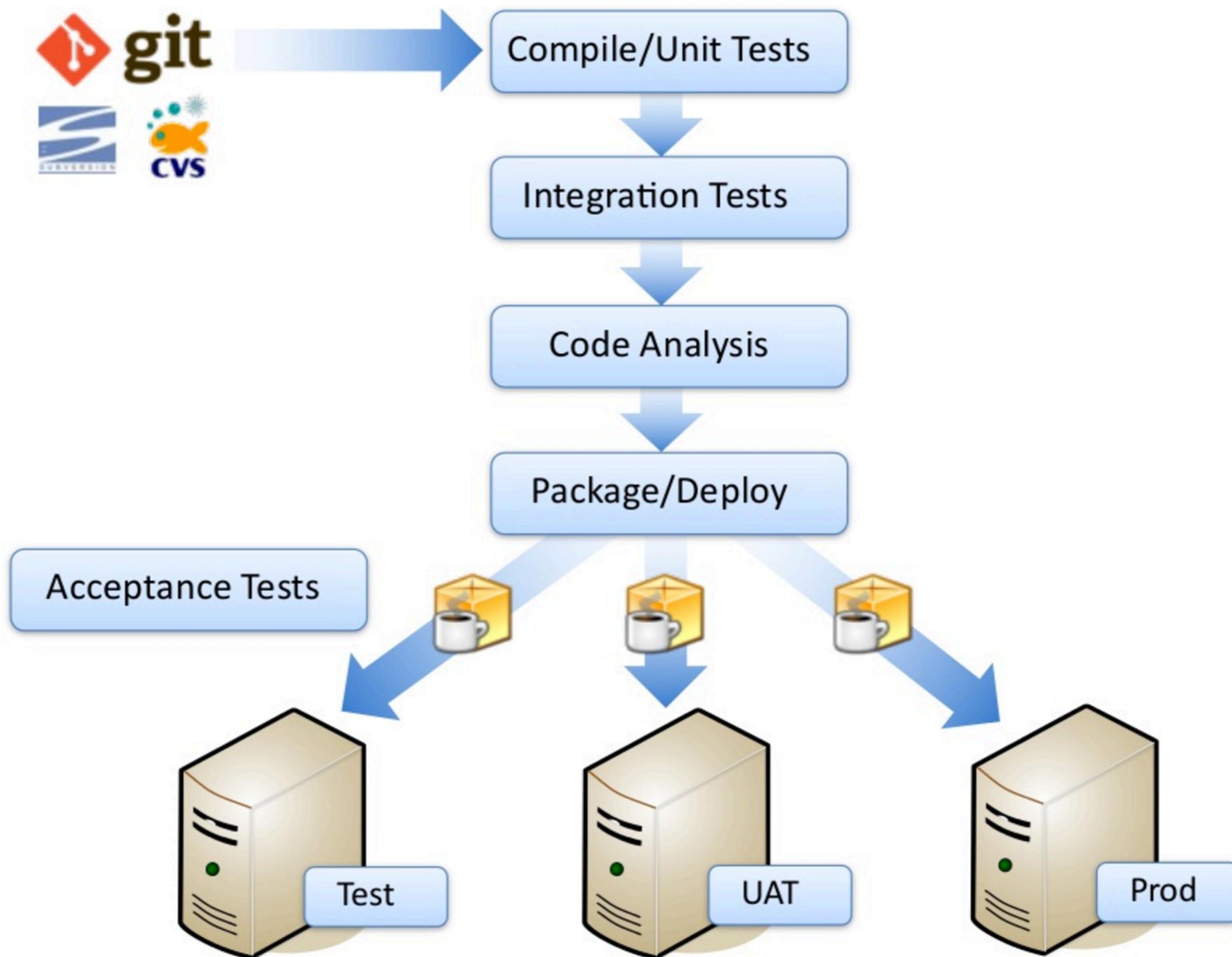
# Jenkins driven by Plugins !!



# Workshop with Java Web Application

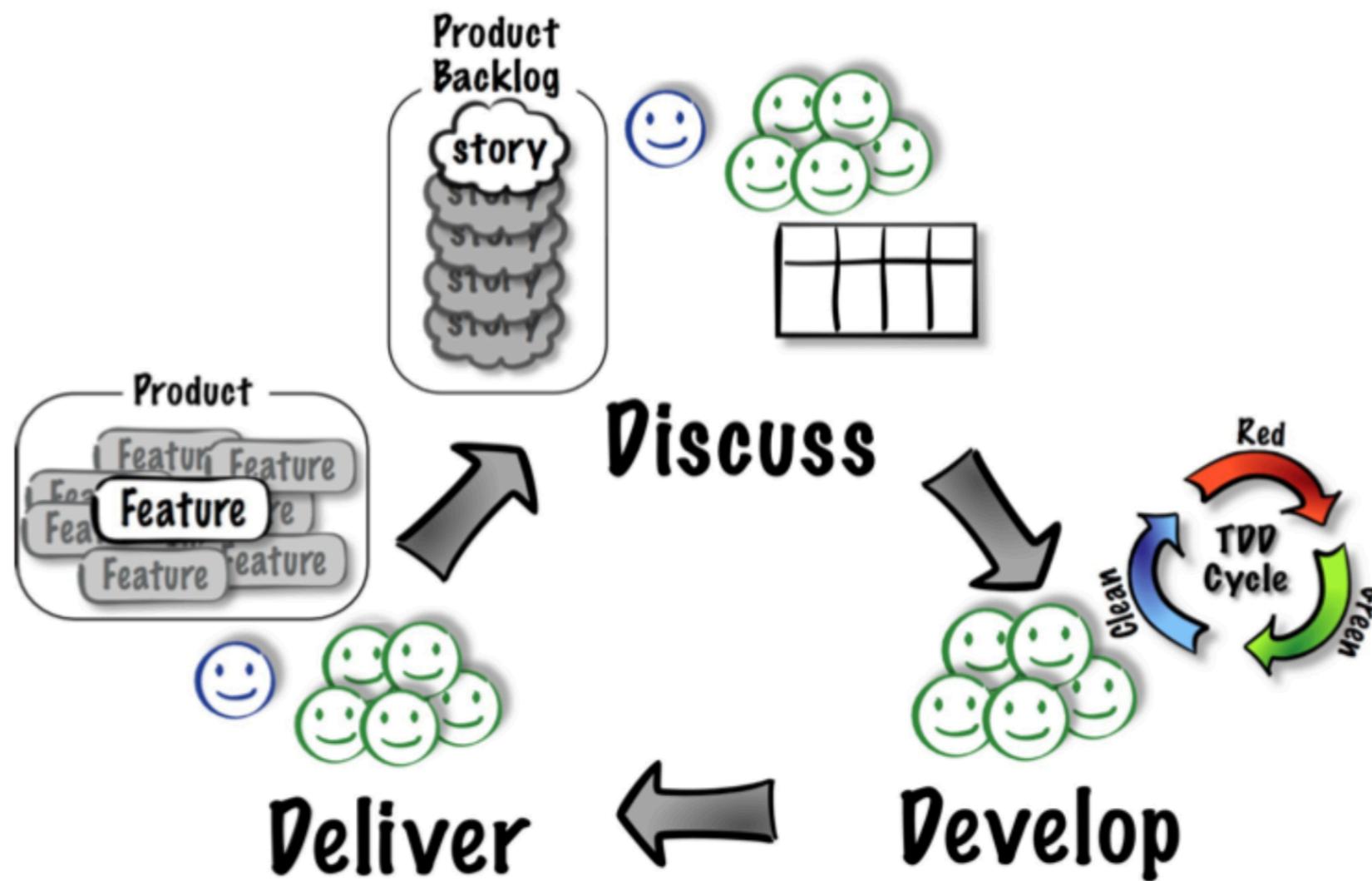


# Pipeline of this project



# 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

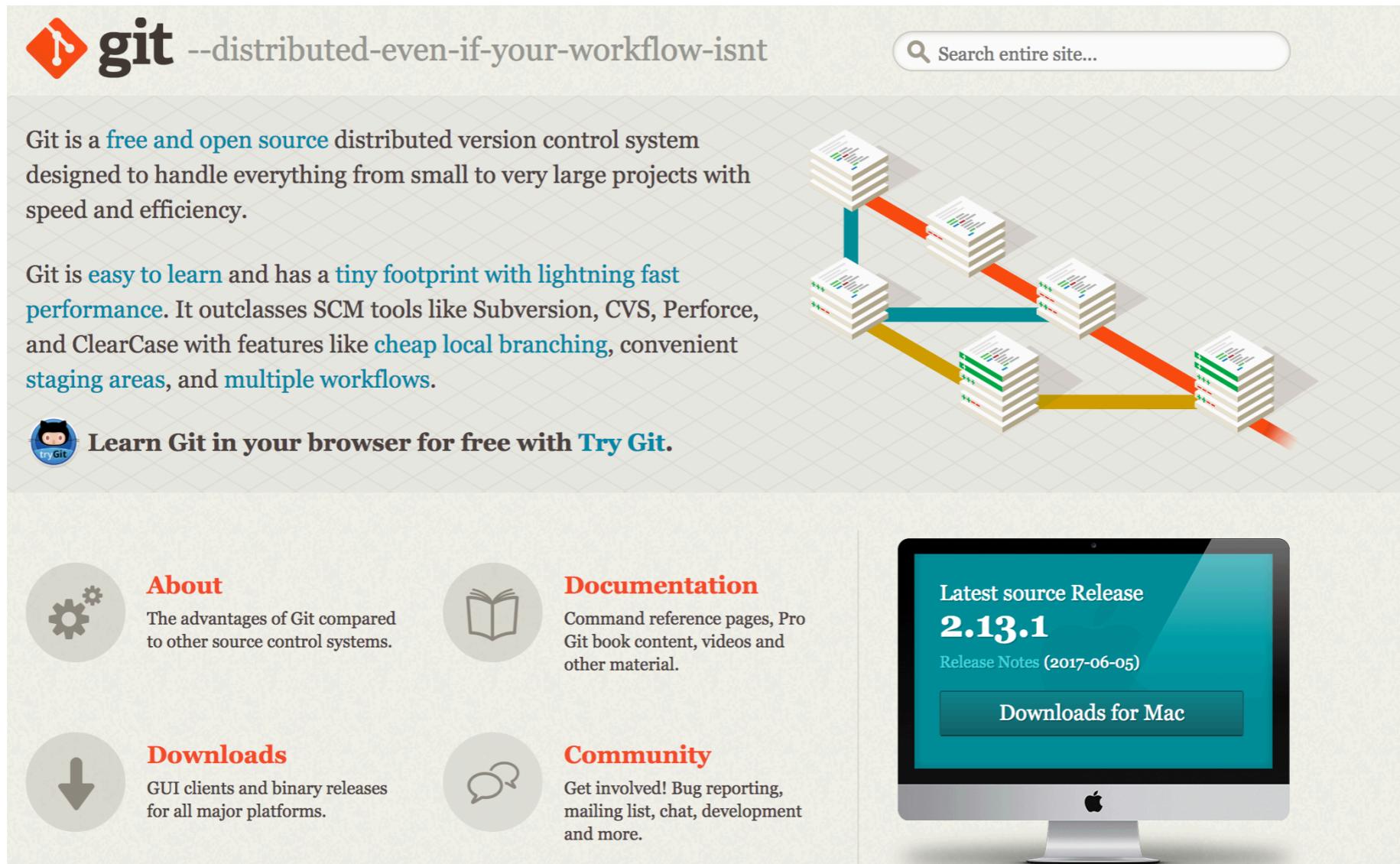


# Try to install toolsets !!



# 1. Git

## Distributed version control system



The screenshot shows the official website for Git (<https://git-scm.com/>). At the top left is the Git logo (a red diamond with a white 'g'). Next to it is the slogan "git --distributed-even-if-your-workflow-isn't". A search bar is located at the top right. Below the header, there is a brief introduction to Git: "Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency." To the right of this text is a diagram illustrating the distributed nature of Git, showing multiple repositories connected by bidirectional arrows. Below the introduction, another paragraph highlights Git's features: "Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows." A "Try Git" button with a GitHub cat icon is present. The main content area is divided into several sections: "About" (advantages over other systems), "Documentation" (command reference, book, videos), "Downloads" (GUI clients, binary releases), and "Community" (bug reporting, mailing list, chat). On the right side, a computer monitor displays the latest source release information: "Latest source Release 2.13.1" with a link to "Release Notes (2017-06-05)" and a "Downloads for Mac" button.

<https://git-scm.com/>



# 2. JDK 1.8

## Java Development Kit 1.8 from Oracle

The screenshot shows the Oracle Java SE Downloads page. At the top, there's a navigation bar with the Oracle logo, a menu icon, a search bar, and user account and call links. Below the navigation is a breadcrumb trail: Oracle Technology Network > Java > Java SE > Downloads. The main content area has tabs for Overview, Downloads (which is selected), Documentation, Community, Technologies, and Training. On the left is a sidebar with links for Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine. The central part of the page features two download cards: one for "Java Platform (JDK) 8u131" with the Java logo and another for "NetBeans with JDK 8" with the NetBeans logo. Below these are sections for "Java Platform, Standard Edition" and "Java SE 8u131". A yellow callout box highlights an important note about MD5-signed JARs. At the bottom, there are links for Installation Instructions and Release Notes, along with a "JDK DOWNLOAD" button.

Java SE Downloads

Java Platform (JDK) 8u131

NetBeans with JDK 8

**Java Platform, Standard Edition**

**Java SE 8u131**

Java SE 8u131 includes important security fixes and bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.

[Learn more](#)

**Important planned change for MD5-signed JARs**

Starting with the April Critical Patch Update releases, planned for April 18 2017, all JRE versions will treat JARs signed with MD5 as unsigned. [Learn more and view testing instructions](#).

For more information on cryptographic algorithm support, please check the [JRE and JDK Crypto Roadmap](#).

▪ Installation Instructions

▪ Release Notes

JDK DOWNLOAD

**Java SDKs and Tools**

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

**Java Resources**

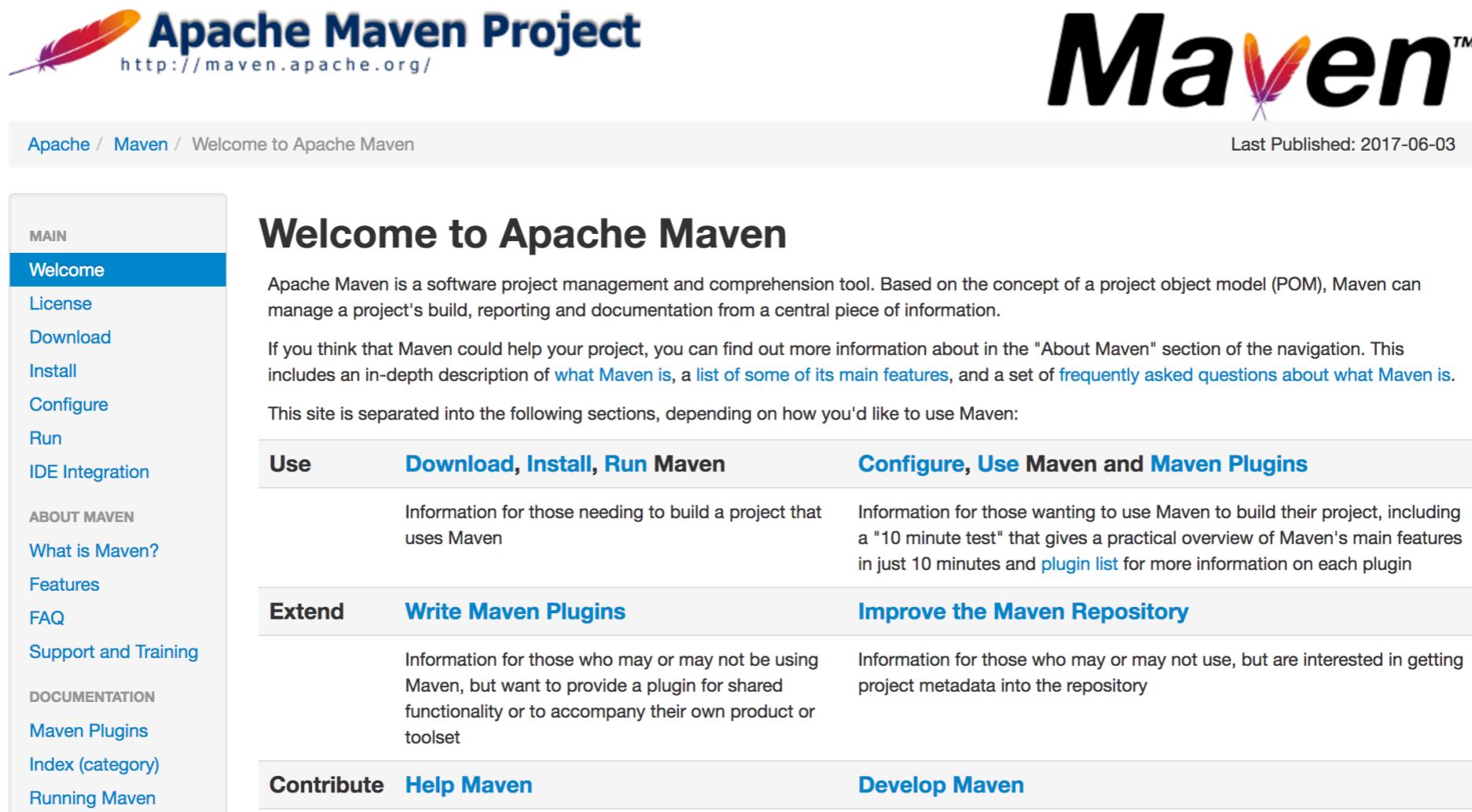
- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Java.net](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



# 3. Apache Maven

Dependency management tool for Java project



The screenshot shows the Apache Maven Project homepage. At the top left is the Apache logo and the text "Apache Maven Project" with the URL "http://maven.apache.org/". To the right is the large "Maven™" logo. Below the header, a breadcrumb navigation shows "Apache / Maven / Welcome to Apache Maven". On the right, a timestamp reads "Last Published: 2017-06-03". The main content area features a section titled "Welcome to Apache Maven" with a brief introduction. It then branches into four categories: "Use", "Extend", "Contribute", and "Develop". Each category has a sub-section: "Download, Install, Run Maven" (under Use), "Write Maven Plugins" (under Extend), "Help Maven" (under Contribute), and "Configure, Use Maven and Maven Plugins" (under Develop). The sidebar on the left contains links for "MAIN" (Welcome, License, Download, Install, Configure, Run, IDE Integration) and "ABOUT MAVEN" (What is Maven?, Features, FAQ, Support and Training). It also includes "DOCUMENTATION" (Maven Plugins, Index (category), Running Maven).

<http://maven.apache.org/>



# 4. Artifactory Server

Sonatype Nexus Repository  
Frog Artifactory



# 4.1 Sonatype Nexus Repository

To keep the artifact such as JAR/WAR/EAR file



Nexus Repository

<https://my.sonatype.com/>



# Start nexus server

\$nexus start

The screenshot shows the Nexus Repository Manager OSS 3.8.0-02 web interface. The URL in the browser is <http://localhost:8081>. The page title is "Nexus Repository Manager OSS 3.8.0-02". The left sidebar has a "Browse" menu with "Welcome" selected, and links for "Search" and "Browse". The main content area features a "Welcome" message "Welcome to Nexus Repository Manager!" and a "Get Started" section with the following items:

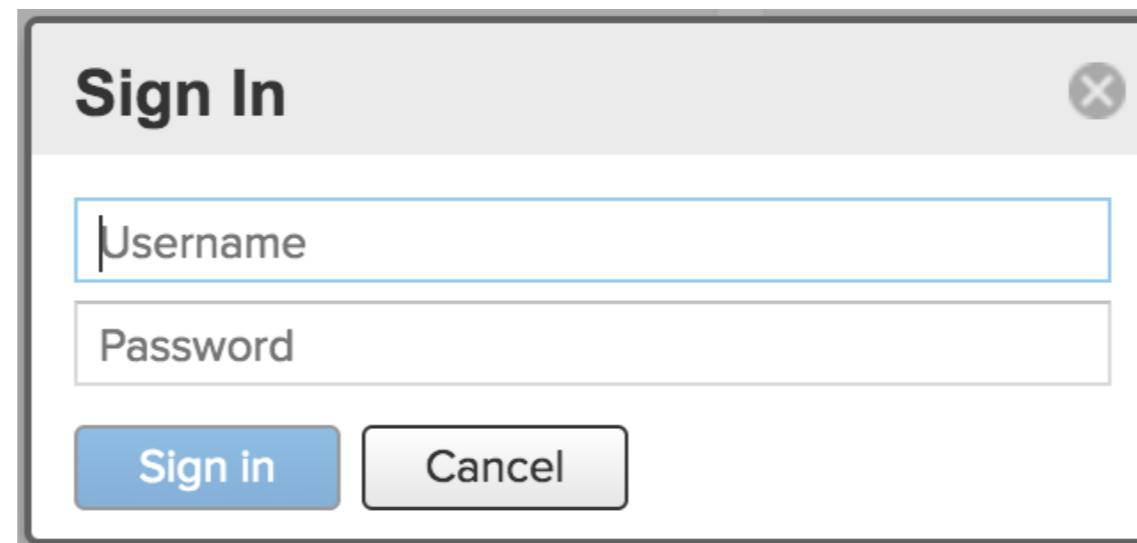
- New in 3.8.0** (with a star icon) - See the release notes [↗](#)
- Configuration** (with a wrench icon) - Set things up properly [↗](#)
- Repository Formats** (with a folder icon)
  - Bower [↗](#)
  - Docker [↗](#)
  - Git LFS [↗](#)
  - Maven [↗](#)
  - .NET/NuGet [↗](#)
  - Node/npm [↗](#)
  - PyPI [↗](#)
  - Raw [↗](#)
  - RubyGems [↗](#)
  - Yum [↗](#)
- Documentation** (with a book icon) - Visit our new help site [↗](#)



# Login

user = admin

password = admin123



The image shows a standard 'Sign In' dialog box. At the top, it says 'Sign In' and has a close button (X) in the top right. Below that are two input fields: one for 'Username' and one for 'Password'. At the bottom are two buttons: 'Sign in' (highlighted in blue) and 'Cancel'.



# 4.2 JFrog Artifactory

To keep the artifact such as JAR/WAR/EAR file

The screenshot shows the homepage of the JFrog Artifactory OSS project. At the top, there is a navigation bar with four items: "Artifactory OSS" (selected), "Plugins", "Conan for C/C++", and "CLI Automation". Below the navigation bar, the main title "JFROG ARTIFACTORY OPEN SOURCE FOR ARTIFACT LIFE-CYCLE MANAGEMENT" is displayed. To the left of the title, there are several 3D cube icons with the letters "O", "I", and "A" on them. To the right of the title, there is another set of similar cube icons. Below the title, a paragraph of text explains the purpose of the project: "JFrog's Artifactory open source project was created to speed up development cycles using binary repositories. It's the world's most advanced repository manager, creating a single place for teams to manage all their binary artifacts efficiently." At the bottom of the page, there are three green download buttons labeled "Download ZIP", "Download Debian", and "Download RPM".

<https://jfrog.com/open-source/>



# 5. SonarQube

To analyze the source code of project

The screenshot shows the official SonarQube website. At the top, there's a navigation bar with links for FEATURES, DOWNLOADS, ROADMAP, COMMUNITY, and BLOG. The main heading "sonarQube" is on the left with a blue icon. Below the navigation, a large blue banner features the text "The leading product for CONTINUOUS CODE QUALITY". It includes three icons: "Code Smells" (radioactive symbol), "Bugs" (bug icon), and "Vulnerabilities" (padlock icon). At the bottom of the banner are buttons for "DOWNLOAD" and "USE ONLINE". To the right of the banner, a green circular icon with a white "S" is followed by the text "Used by more than 80,000 organizations". Below the banner, a footer navigation bar lists "On 20+ Code Analyzers" with a right-pointing arrow, and links for Java, JavaScript, C#, C/C++, COBOL, and "AND MORE".

<https://www.sonarqube.org>



# Start SonarQube Server

**http://localhost:9000**

The screenshot shows the SonarQube interface. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, and Quality Gates. A search bar and a 'Log in' button are also present. Below the header, the main content area features a 'Continuous Code Quality' section with a large '0' indicating no projects analyzed. To the right, there are three metrics: 0 Bugs, 0 Vulnerabilities, and 0 Code Smells. Further down, a 'Multi-Language' section lists over 20 supported programming languages, each with a corresponding link.

sonarqube Projects Issues Rules Quality Profiles Quality Gates

?

Search for projects, sub-projects and files

Log in

Continuous Code Quality

0 Projects Analyzed

0 Bugs

0 Vulnerabilities

0 Code Smells

Log in Read documentation

Multi-Language

20+ programming languages are supported by SonarQube thanks to our in-house code analyzers, including:

<a href="#">Java</a>	<a href="#">C/C++</a>	<a href="#">C#</a>	<a href="#">COBOL</a>	<a href="#">ABAP</a>	<a href="#">HTML</a>	<a href="#">RPG</a>	<a href="#">JavaScript</a>	<a href="#">TypeScript</a>	<a href="#">Objective C</a>	<a href="#">XML</a>
<a href="#">VB.NET</a>	<a href="#">PL/SQL</a>	<a href="#">T-SQL</a>	<a href="#">Flex</a>	<a href="#">Python</a>	<a href="#">Groovy</a>	<a href="#">PHP</a>	<a href="#">Swift</a>	<a href="#">Visual Basic</a>	<a href="#">PL/I</a>	



# Login

user = admin

password = admin

Log In to SonarQube

[Log in](#) [Cancel](#)



# 6. Apache Tomcat

## Java Web Server to run Java application



### Apache Tomcat®

Search...

**Apache Tomcat**

- Home
- Taglibs
- Maven Plugin

**Download**

- Which version?
- Tomcat 9
- Tomcat 8
- Tomcat 7
- Tomcat Connectors
- Tomcat Native
- Taglibs
- Archives

**Documentation**

- Tomcat 9.0
- Tomcat 8.5
- Tomcat 8.0
- Tomcat 7.0
- Tomcat Connectors
- Tomcat Native
- Wiki
- Migration Guide

**Apache Tomcat**

The Apache Tomcat® software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. The Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket specifications are developed under the [Java Community Process](#).

The Apache Tomcat software is developed in an open and participatory environment and released under the [Apache License version 2](#). The Apache Tomcat project is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, [click here](#).

Apache Tomcat software powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Some of these users and their stories are listed on the [PoweredBy](#) wiki page.

Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo are trademarks of the Apache Software Foundation.

**Tomcat 7.0.84 Released** 2018-01-24

The Apache Tomcat Project is proud to announce the release of version 7.0.84 of Apache Tomcat. This release contains a number of bug fixes and improvements compared to version 7.0.82. The notable changes compared to 7.0.82 include:

- Java 9 is fully supported
- Updated the packaged version of the Tomcat Native Library to 1.2.16 to pick up the latest Windows binaries built with APR 1.6.3 and OpenSSL 1.0.2m
- Add a new system property (`org.apache.jasper.runtime.BodyContentImpl.BUFFER_SIZE`) to control the size of the buffer used by Jasper when buffering tag bodies.

Full details of these changes, and all the other changes, are available in the [Tomcat 7 changelog](#).

<http://tomcat.apache.org/>



# Configure software in Jenkins



# JDK

Manage Jenkins -> Global tool configuration

## Global Tool Configuration

### Maven Configuration

Default settings provider

Use default maven settings



Default global settings provider

Use default maven global settings



### JDK

JDK installations



Name

Required

Install automatically



#### Install from java.sun.com

Version Java SE Development Kit 9.0.4



I agree to the Java SE Development Kit License Agreement

Installing JDK requires Oracle account. [Please enter your username/password](#)

Delete Installer

Add Installer ▾



# Git

Manage Jenkins -> Global tool configuration

## Git

### Git installations

 **Git**

Name	<input type="text" value="Default"/>
Path to Git executable	<input type="text" value="git"/> 
<input type="checkbox"/> Install automatically	
<b>Delete Git</b> 	

**Add Git** ▾



# Apache Maven

Manage Jenkins -> Global tool configuration

## Maven

Maven installations



Maven

Name

Required

Install automatically



Install from Apache

Version

Delete Installer

Add Installer



Add Maven

List of Maven installations on this system

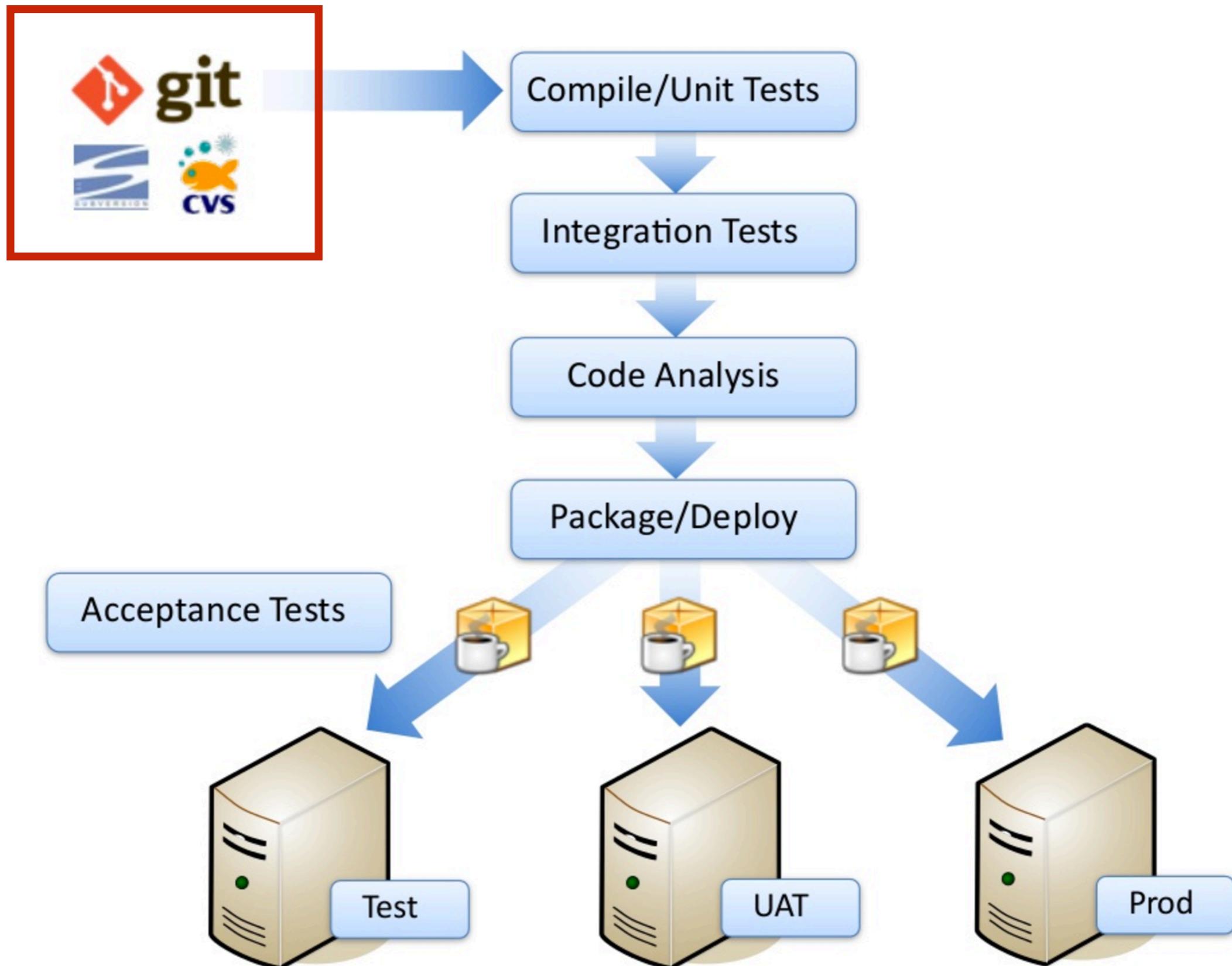
Delete Maven



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



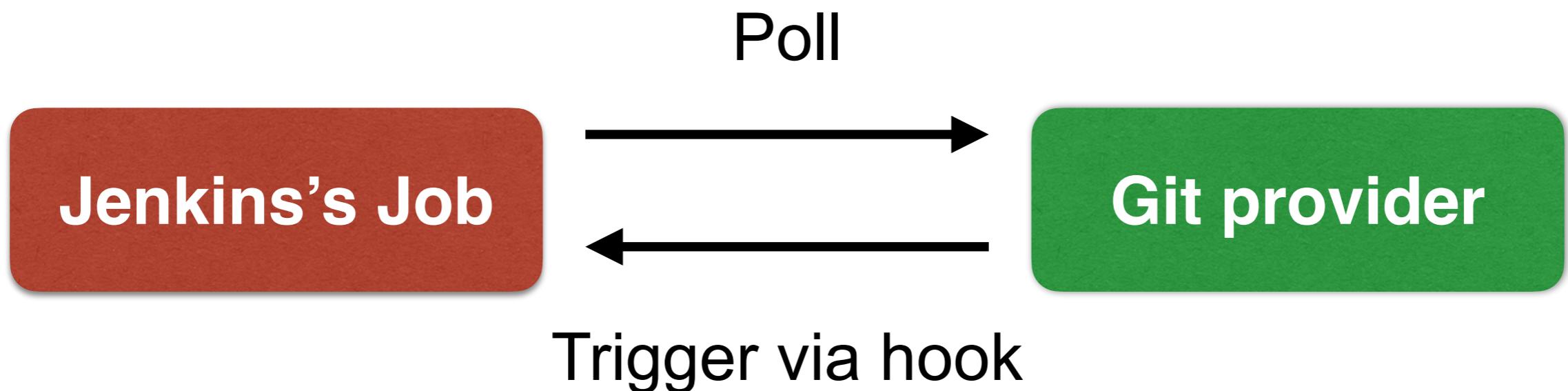
1



# 1. Working with Git

Pull source code from Git provider such as Github

Keep source code in Job's workspace



# Example code

<https://github.com/up1/workshop-java-web-tdd>



# Step 1 Create new job

Job name = step01\_pullcode

Enter an item name

step01\_pullcode

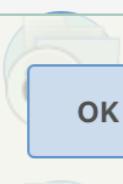
» Required field

 **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.

 **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.

 **GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.  
**OK**



# Step 2 Source code management

## Add url of git provider repository

The screenshot shows the Jenkins configuration interface for a job's "Source Code Management" section. The "Source Code Management" tab is selected. Under the "Repositories" section, the "Git" option is chosen. The "Repository URL" field contains the value `https://github.com/up1/workshop-java-web-tdd.git`. A red error message below the field says "Please enter Git repository." The "Credentials" dropdown is set to "- none -". There are "Advanced..." and "Add Repository" buttons. In the "Branches to build" section, the "Branch Specifier" field contains `*/master`. There is an "Add Branch" button. The "Repository browser" dropdown is set to "(Auto)". In the "Additional Behaviours" section, there is an "Add" button with a dropdown menu. At the bottom are "Save" and "Apply" buttons.



# Step 3 Manual build job

Add url of git provider repository



The screenshot shows the Jenkins interface for the project "step01\_pullcode". The top navigation bar includes the Jenkins logo and the path "Jenkins > step01\_pullcode". On the left, there is a sidebar with links: "Back to Dashboard", "Status", "Changes", "Workspace", "Build Now" (which is highlighted with a red box), "Delete Project", and "Configure". The main content area is titled "Project step01\_pullcode" and contains sections for "Workspace" and "Recent Changes". At the bottom, there is a "Build History" section with a search bar and RSS feed links for "RSS for all" and "RSS for failures".



# Step 4 See result

See all source code in workspace

The screenshot shows the Jenkins interface for the project "step01\_pullcode". The left sidebar has a red box around the "Workspace" link. The main content area shows a blue folder icon with the text "Workspace" and a red box around it. Below it is a "Recent Changes" link.

Jenkins

Jenkins > step01\_pullcode >

- Back to Dashboard
- Status
- Changes
- Workspace**
- Build Now
- Delete Project
- Configure

**Project step01\_pullcode**

Workspace

Recent Changes

Build History

trend —

find

#1 Feb 6, 2018 11:13 AM

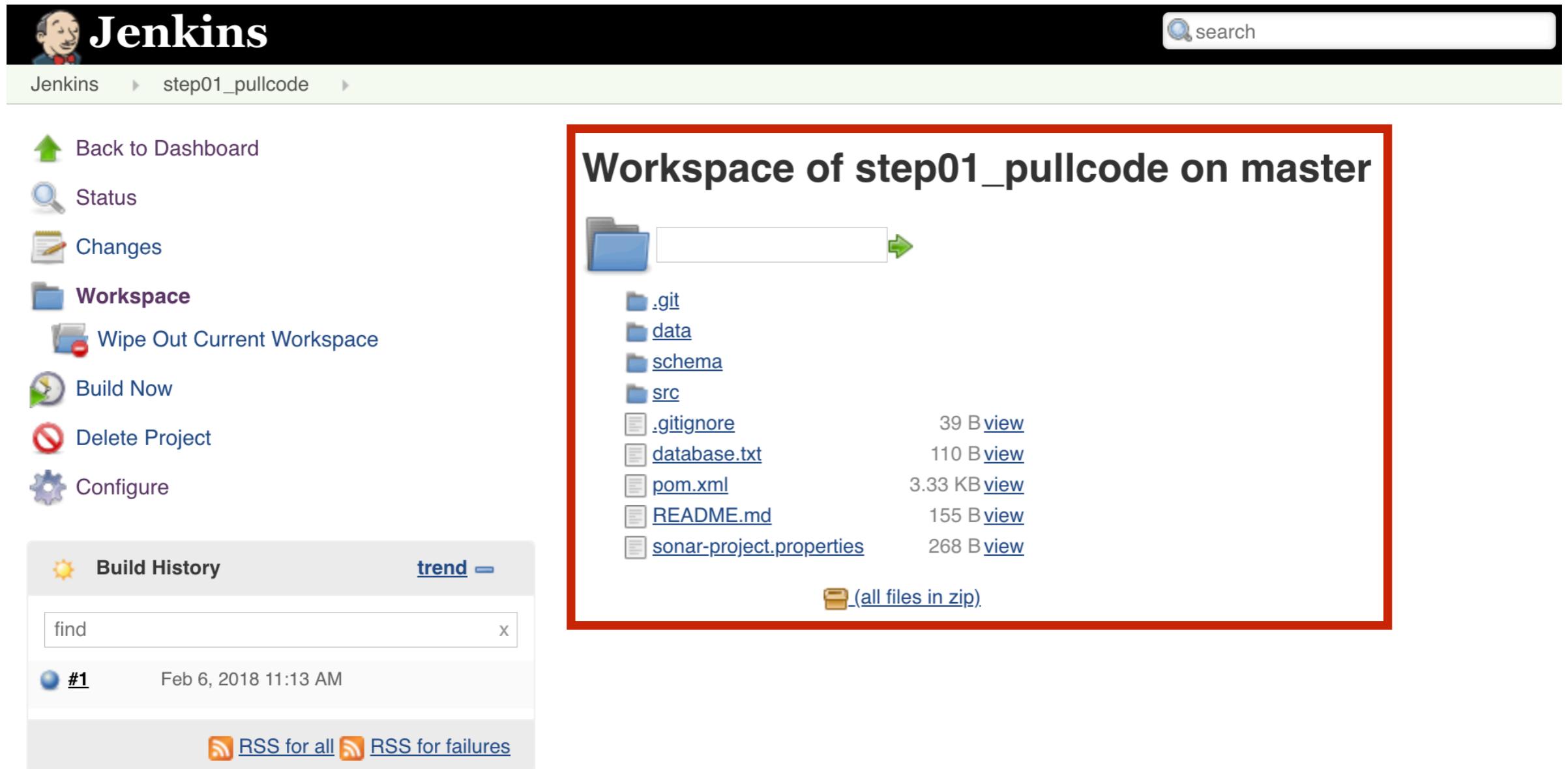
RSS for all RSS for failures

Permalinks



# Step 4 See result

See all source code in workspace



The screenshot shows the Jenkins workspace page for the project "step01\_pullcode". The top navigation bar includes the Jenkins logo, a search bar, and the current path: Jenkins > step01\_pullcode >. On the left, a sidebar provides links to Back to Dashboard, Status, Changes, Workspace (which is highlighted in blue), Wipe Out Current Workspace, Build Now, Delete Project, and Configure. Below this is the Build History section, which shows one build (#1) from Feb 6, 2018, at 11:13 AM. At the bottom are RSS feed links. The main content area is titled "Workspace of step01\_pullcode on master" and displays a file tree with the following contents:

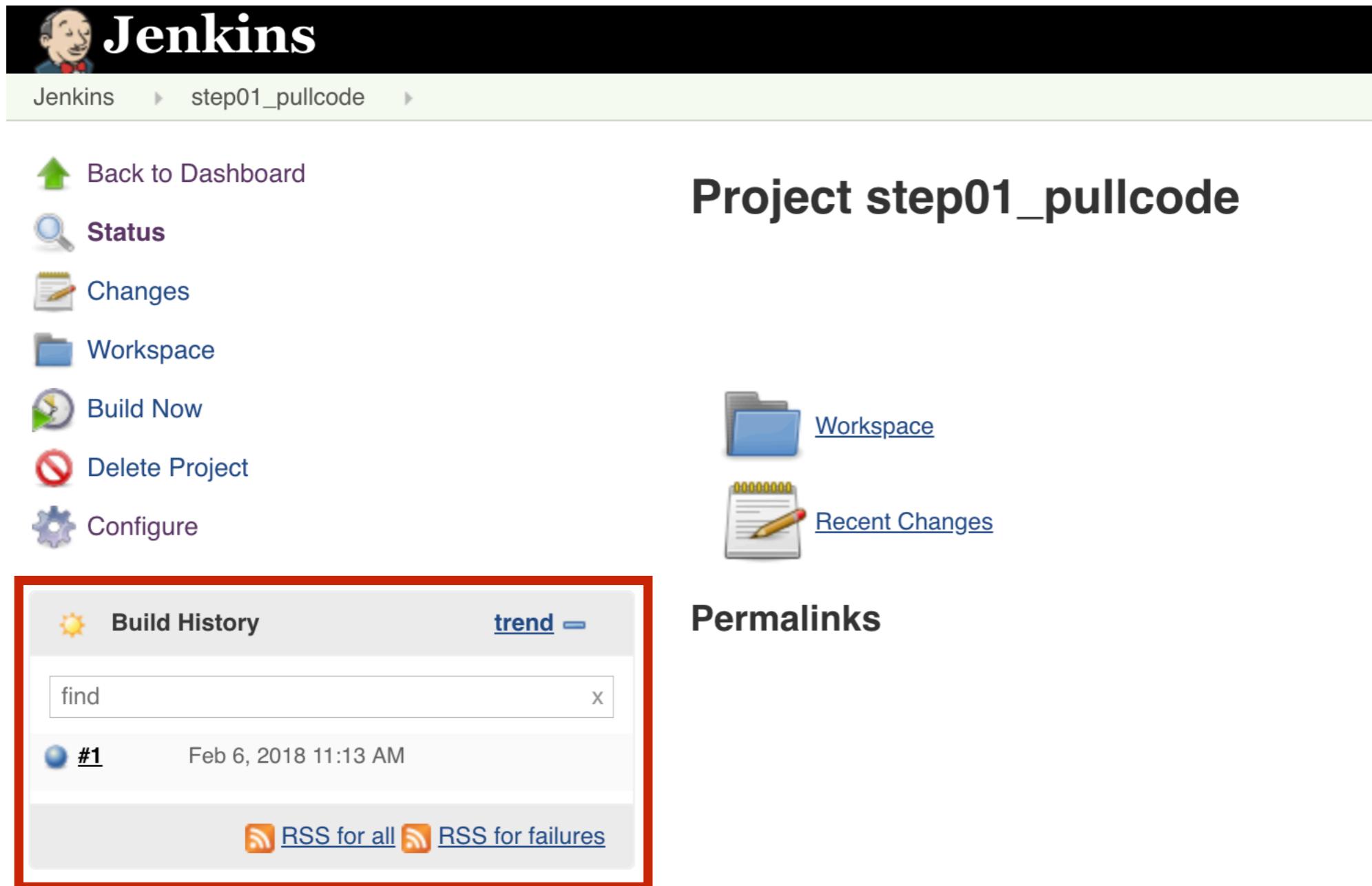
- .git
- data
- schema
- src
- .gitignore 39 B [view](#)
- database.txt 110 B [view](#)
- pom.xml 3.33 KB [view](#)
- README.md 155 B [view](#)
- sonar-project.properties 268 B [view](#)

At the bottom right of the workspace view is a link "(all files in zip)".



# Step 4 See result

Build History keep in logs file !!

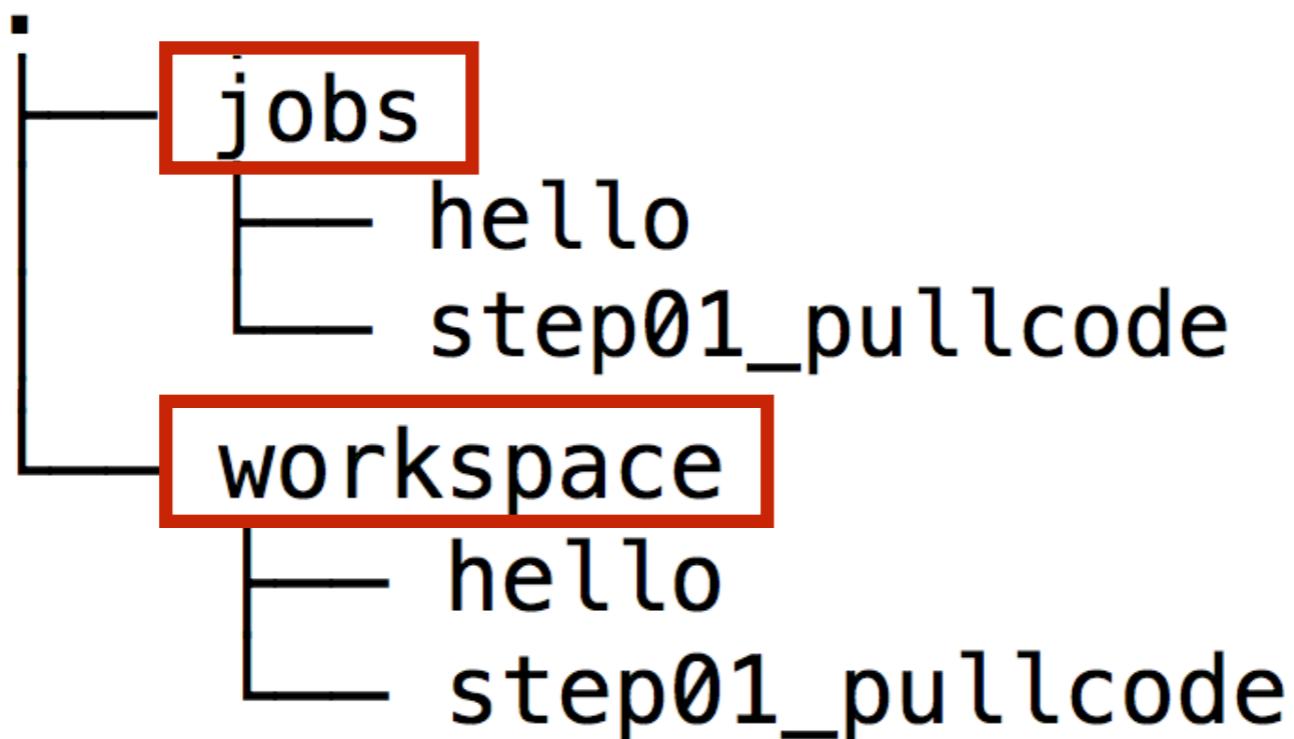


The screenshot shows the Jenkins interface for the project "step01\_pullcode". The top navigation bar includes the Jenkins logo and the current path: Jenkins > step01\_pullcode. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main content area is titled "Project step01\_pullcode". It features a "Workspace" link with a folder icon and a "Recent Changes" link with a document icon. The central part of the page is the "Build History" section, which is highlighted with a red box. This section has a "trend" dropdown, a search bar with placeholder "find", and a table with one entry: "#1" from "Feb 6, 2018 11:13 AM". At the bottom of this section are two RSS feed links: "RSS for all" and "RSS for failures".



# Where is your data ?

In JENKINS's HOME



# Where is your data ?

In JENKINS's HOME

```
step01 pullcode
  └── builds
    ├── 1
    │   ├── lastFailedBuild -> -1
    │   ├── lastStableBuild -> 1
    │   ├── lastSuccessfulBuild -> 1
    │   ├── lastUnstableBuild -> -1
    │   └── lastUnsuccessfulBuild -> -1
    └── legacyIds
        └── config.xml
        └── lastStable -> builds/lastStableBuild
        └── lastSuccessful -> builds/lastSuccessfulBuild
    └── nextBuildNumber
```



# Be careful !!

## Harddisk Full

step01 pullcode

```
└── builds
    └── 1
        ├── lastFailedBuild -> -1
        ├── lastStableBuild -> 1
        ├── lastSuccessfulBuild -> 1
        ├── lastUnstableBuild -> -1
        └── lastUnsuccessfulBuild -> -1
    └── legacyIds
    └── config.xml
    └── lastStable -> builds/lastStableBuild
    └── lastSuccessful -> builds/lastSuccessfulBuild
└── nextBuildNumber
```



# Need to manage the build history

Choose Discard old builds in General tab

The screenshot shows the Jenkins General configuration page for a project named "step01\_pullcode". The "General" tab is selected. A red box highlights the "Discard old builds" section. This section contains a checked checkbox labeled "Discard old builds", a dropdown menu set to "Log Rotation", and two input fields: "Days to keep builds" and "Max # of builds to keep". Below these fields are their respective descriptions: "if not empty, build records are only kept up to this number of days" and "if not empty, only up to this number of build records are kept". At the bottom left are "Save" and "Apply" buttons, and at the bottom right is an "Advanced..." button.

Project name: step01\_pullcode

Description:

[Plain text] [Preview](#)

Discard old builds [?](#)

Strategy: Log Rotation

Days to keep builds:

if not empty, build records are only kept up to this number of days

Max # of builds to keep:

if not empty, only up to this number of build records are kept

[Advanced...](#)

[Save](#) [Apply](#)



# Discard old builds

Default Strategy is LogRotation that have 2 options

1. Days to keep build
2. Maximum # of build to keep



# Discard old builds

If you need more, see in plugins section

The screenshot shows the Jenkins plugin marketplace interface. At the top, there is a navigation bar with tabs: 'Updates', 'Available' (which is selected and highlighted in dark grey), 'Installed', and 'Advanced'. To the right of the tabs is a search bar with the placeholder text 'Filter:' and a magnifying glass icon. A red rectangular box highlights this search bar. Below the tabs, there is a heading 'Install' with a downward arrow, followed by two plugin entries:

	Name	Version
<input type="checkbox"/>	<a href="#">enhanced-old-build-discarder</a>	1.0
<input type="checkbox"/>	<a href="#">Discard Old Build plugin</a> Add post-build step to discard old builds with detail configuration	1.05

At the bottom of the page, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'. To the right of the 'Check now' button, the text 'Update information obtained: 1 day 11 hr ago' is displayed.



# Step 5 Add Build Triggers

Use Poll SCM to check any changes from repos

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 ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Schedule

\*\*\*\*\*  
\* \* \* \* \*

⚠️ Do you really mean "every minute" when you say "\* \* \* \* \*"? Perhaps you meant "H \* \* \* \*" to poll once per hour

Would last have run at Tuesday, February 6, 2018 11:33:56 AM ICT; would next run at Tuesday, February 6, 2018 11:33:56 AM ICT.

Ignore post-commit hooks  ?



# Schedule setting

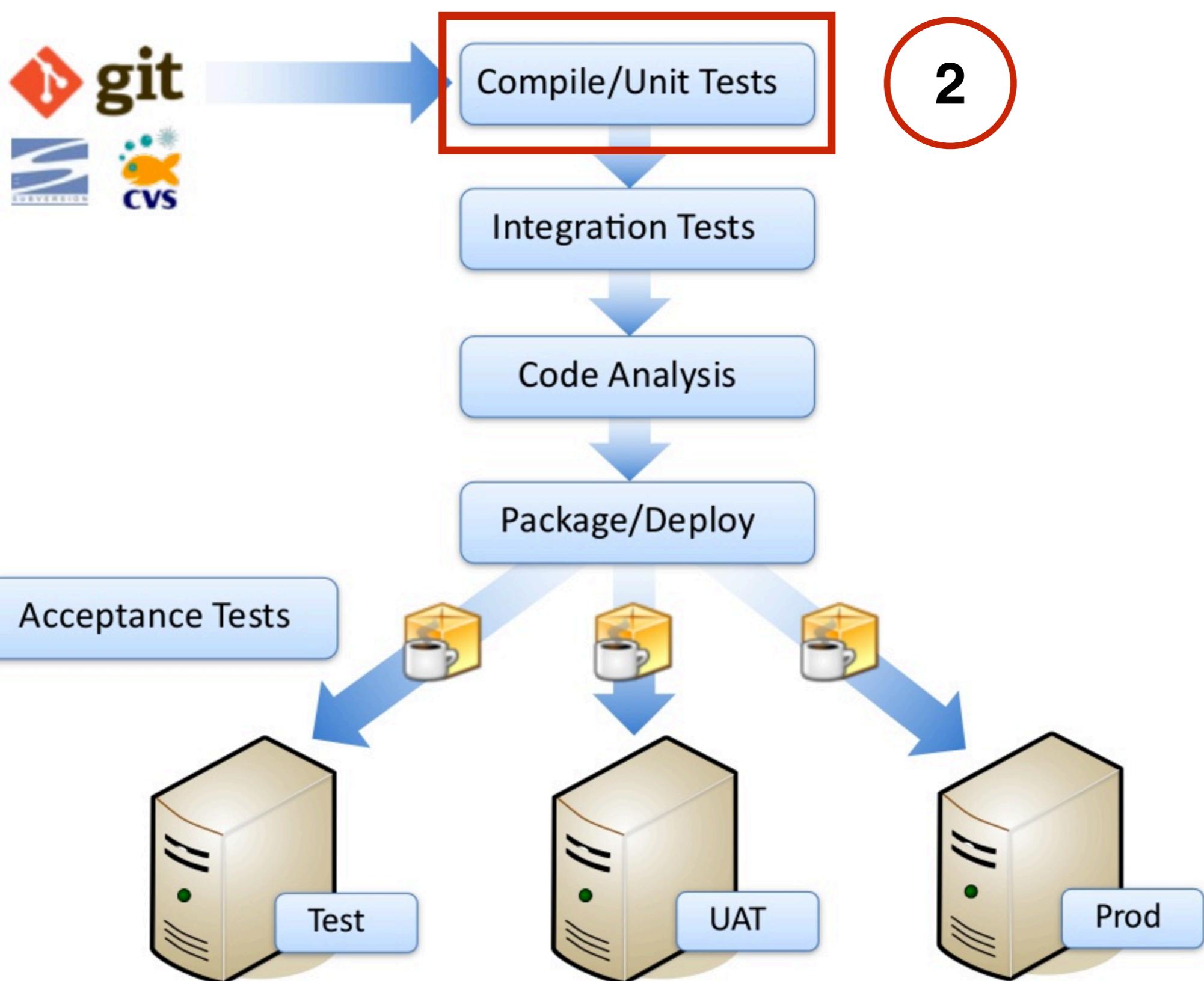
Use a crontab format

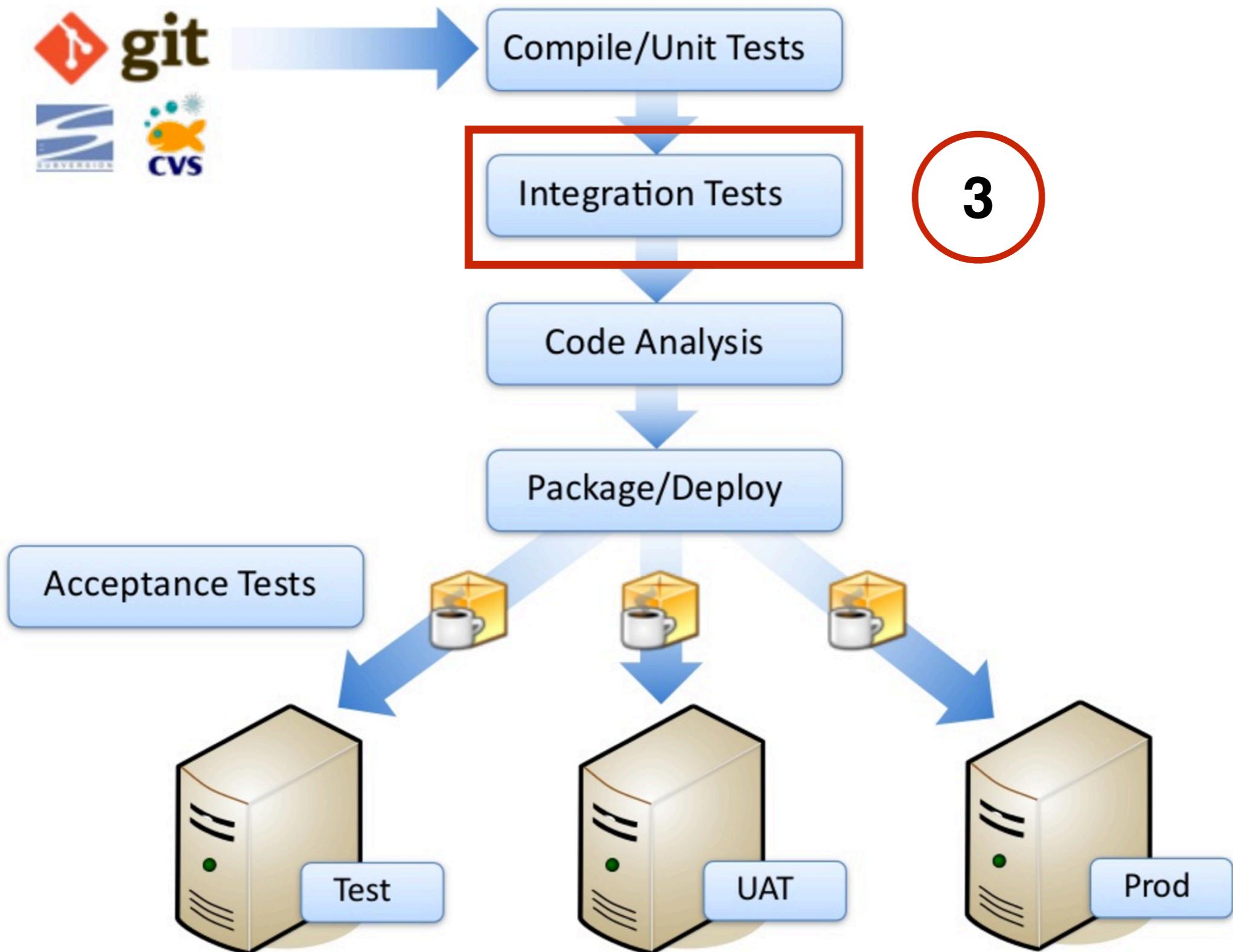
Name	Description
Minute	Minutes within the hour (0–59)
Hour	The hour of the day (0–23)
DOM	The day of the month (1–31)
Month	The month (1–12)
DOW	The day of the week (0–7) where 0 and 7 are Sunday.



# Try it by yourself







# 2. Compile and run unit tests

JDK

Apache Maven  
JUnit



# How to compile and test ?

We using the Apache Maven to manage

\$mvn clean package



# Add build steps

**Build**

Add build step ▾

- Execute Windows batch command
- 1 Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending"

Save   Advanced

**Build**

Execute shell

Command mvn clean package

2

**mvn clean package**

See [the list of available environment variables](#)

Advanced...

Add build step ▾

A red circle labeled '1' highlights the 'Execute shell' option in the 'Add build step' dropdown menu. A red circle labeled '2' highlights the 'mvn clean package' command entered in the 'Command' field of the 'Execute shell' step configuration.



# Try to run and see output

Scroll down to button and see Success status

Jenkins > step01\_pullcode > #2

Back to Project Status Changes **Console Output** View as plain text Edit Build Information Git Build Data No Tags Previous Build

Progress:  X

## Console Output

```
Started by user Somkiat Puisungnoen
Building on master in workspace /Users/somkiat/Downloads/set/workspace/step01_pullcode
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/up1/workshop-java-web-tdd.git # timeout=10
Fetching upstream changes from https://github.com/up1/workshop-java-web-tdd.git
> git --version # timeout=10
> git fetch --tags --progress https://github.com/up1/workshop-java-web-tdd.git
+refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 7807e5a4628bf77301ad9268c9763184f48b10d8 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10

[INFO] Cobertura Report generation was successful.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 27.644 s
[INFO] Finished at: 2018-02-06T11:48:25+07:00
[INFO] Final Memory: 29M/173M
[INFO] -----
Finished: SUCCESS
```



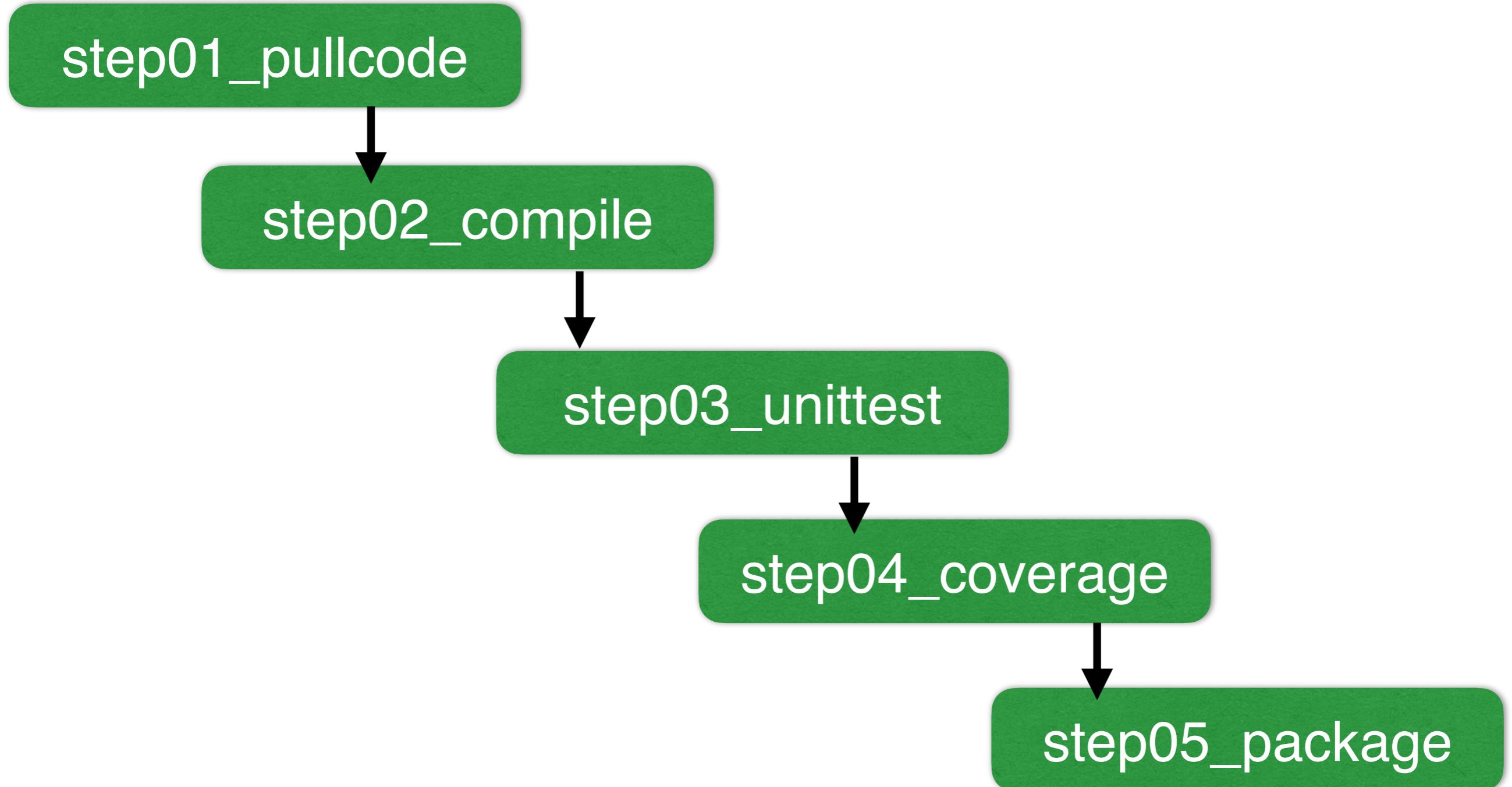
# Consideration for output

1. Compile was success
2. Testing was success
3. Create artifact file or WAR file was success
4. Test/code coverage was success



# Suggestion

We need to separate to more jobs



# Step to run each job

Using Apache maven

Step	Command
Compile	mvn clean test
Unit test	mvn clean test
Coverage	mvn clean package
Package/Artifact	mvn clean package



# Create new job to compile&test

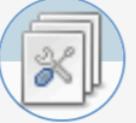
Job name = step02\_build

Enter an item name

» 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.

 **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, folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**OK**



# Add build steps

**Build**

**Execute shell**

Command `mvn clean test`

**mvn clean test**

[See the list of available environment variables](#)

[Advanced...](#)

[Add build step ▾](#)



# **Problem !!**

# **Where is a source code ?**



# Source code from step01



`$JENKIN_HOME/workspaces`



# Source code from step01

General => Advances => Use custom workspace

Screenshot of Jenkins project configuration showing the 'General' tab selected. The 'Use custom workspace' checkbox is checked, and the 'Directory' field contains the value \${JENKINS\_HOME}/workspace/step01\_pullcode. A red error message below the field states 'Custom workspace is empty.'

General		Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions
<input type="checkbox"/> Quiet period						?
<input type="checkbox"/> Retry Count						?
<input type="checkbox"/> Block build when upstream project is building						?
<input type="checkbox"/> Block build when downstream project is building						?
<input checked="" type="checkbox"/> Use custom workspace						?
Directory	<input type="text" value="\${JENKINS_HOME}/workspace/step01_pullcode"/>					
<b>Custom workspace is empty.</b>						
Display Name						?
<input type="checkbox"/> Keep the build logs of dependencies						?

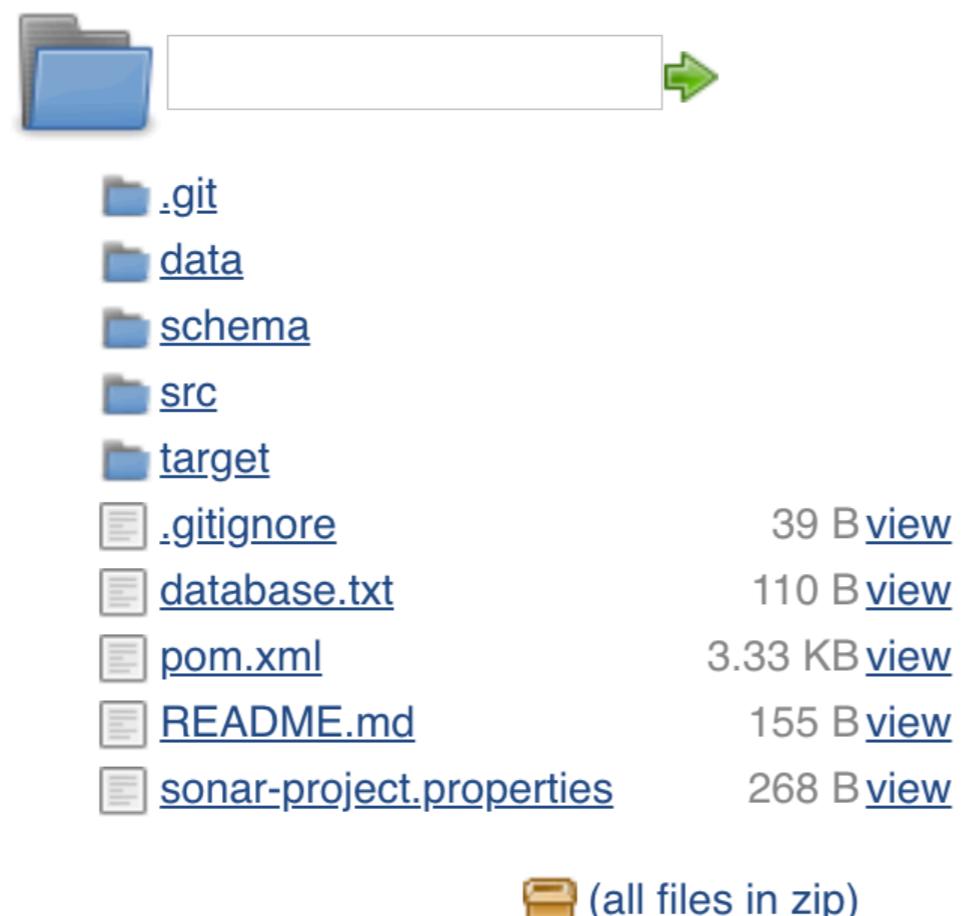
**`\${JENKINS\_HOME}/workspace/step01\_pullcode`**



# Try to manual build

All source code that you need to compile and test

## Workspace of step02\_build on master



# Try it by yourself



# Generate test result report



# Generate test result

Post-build Actions => Publish JUnit test result report

- 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 ▾**



# Generate test result

Set path of test reports (XML files)

## Post-build Actions

**Publish JUnit test result report**

Test report XMLs **target/surefire-reports/\*.xml**

Fileset ‘includes’ setting that specifies the generated raw XML report files, such as ‘myproject/target/test-reports/\*.xml’. Basedir of the fileset is the workspace root.

Retain long standard output/error

Health report amplification factor **1.0**

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Do not fail the build on empty test results

Add post-build action ▾



# Generate test result

See result of testing, trending of testing

Jenkins > step02\_build >

ENABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

**Project step02\_build**

[add description](#)

[Disable Project](#)

**Test Result Trend**

count

#3 #4

(just show failures) [enlarge](#)

[Workspace](#)

[Recent Changes](#)

[Latest Test Result \(no failures\)](#)

**Build History** [trend](#)

find

#4 Feb 6, 2018 4:24 PM

#3 Feb 6, 2018 4:24 PM

#2 Feb 6, 2018 4:21 PM

#1 Feb 6, 2018 12:13 PM

[RSS for all](#) [RSS for failures](#)

**Permalinks**

- [Last build \(#4\), 39 sec ago](#)
- [Last stable build \(#4\), 39 sec ago](#)
- [Last successful build \(#4\), 39 sec ago](#)
- [Last failed build \(#2\), 3 min 22 sec ago](#)
- [Last unsuccessful build \(#2\), 3 min 22 sec ago](#)
- [Last completed build \(#4\), 39 sec ago](#)



# Generate test result

Detail of result in each package

## Test Result

0 failures ( $\pm 0$ )

11 tests ( $\pm 0$ )

Took 0.39 sec.

 [add description](#)

## All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
<a href="#">dao</a>	0.39 sec	0	0	2	2
<a href="#">service</a>	7 ms	0	0	9	9



# Generate code coverage report



# Code coverage tools

Cobertura

Jacoco

Clover



# Install Cobertura plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/> Cobertura Plugin		1.12
<input type="checkbox"/> GitHub Pull Request Coverage Status	Addon for <a href="https://wiki.jenkins-ci.org/display/JENKINS/GitHub+pull+request+builder+plugin">https://wiki.jenkins-ci.org/display/JENKINS/GitHub+pull+request+builder+plugin</a> give you possibility to publish code coverage status (Cobertura and Jenkins) to Pull Request Commits. So you will see coverage of your PR and comparision with master.	1.9.1

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 1 day 16 hr ago [Check now](#)



# Install Jacoco plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/> <a href="#">JaCoCo plugin</a>		2.2.1

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 1 day 16 hr ago [Check now](#)



# Install Clover plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/>	<a href="#">Clover plugin</a>	4.8.0
<input checked="" type="checkbox"/>	<a href="#">Clover PHP plugin</a>	0.5

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 1 day 16 hr ago [Check now](#)



# Using Cobertura in job

Post-build Actions => Publish cobertura coverage report

The screenshot shows a list of Jenkins post-build actions. The 'Publish Cobertura Coverage Report' option is highlighted with a blue background, indicating it is selected or the current topic of discussion. Other options listed include: 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], and Delete workspace when build is done. At the bottom of the list is a button labeled 'Add post-build action ▾'.

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish Cobertura Coverage Report**
- 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 ▾



# Using Cobertura in job

Post-build Actions => Publish cobertura coverage report

## Post-build Actions

X

**Publish Cobertura Coverage Report**

Cobertura xml report pattern `**/target/site/cobertura/coverage.xml`

This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use `**/target/site/cobertura/coverage.xml`). The path is relative to the module root unless you have configured your SCM with multiple modules, in which case it is relative to the workspace root. Note that the module root is SCM-specific, and may not be the same as the workspace root.

Cobertura must be configured to generate XML reports for this plugin to function.

NOTE: If concurrent builds are enabled for this job, and a later build finishes before an earlier build, the later build will reduce or skip trend analysis/charting.

[Advanced...](#)

[Add post-build action ▾](#)



# Using Cobertura in job

Advances => Coverage Metric Targets

Coverage Metric Targets

Methods	80.0	0.0	0.0	X
Lines	80.0	0.0	0.0	X
Conditionals	70.0	0.0	0.0	X

Add

Configure health reporting thresholds.  
For the ☀ row, leave blank to use the default value (i.e. 80).  
For the ⚡ and 🌽 rows, leave blank to use the default values (i.e. 0).

Target must come from Deliver team !!



# See result of report

Jenkins ▶ step03\_coverage ▶ [ENABLE AUTO REFRESH](#)

[Back to Dashboard](#) [Status](#) [Changes](#) [Workspace](#) [Build Now](#) [Delete Project](#) [Configure](#) [Coverage Report](#)

**Project step03\_coverage**

[Coverage Report](#) [Workspace](#) [Recent Changes](#)

**Code Coverage**

Packages	50%
Files	43%
Classes	38%
Methods	28%
Lines	31%
Conditionals	63%

**Permalinks**

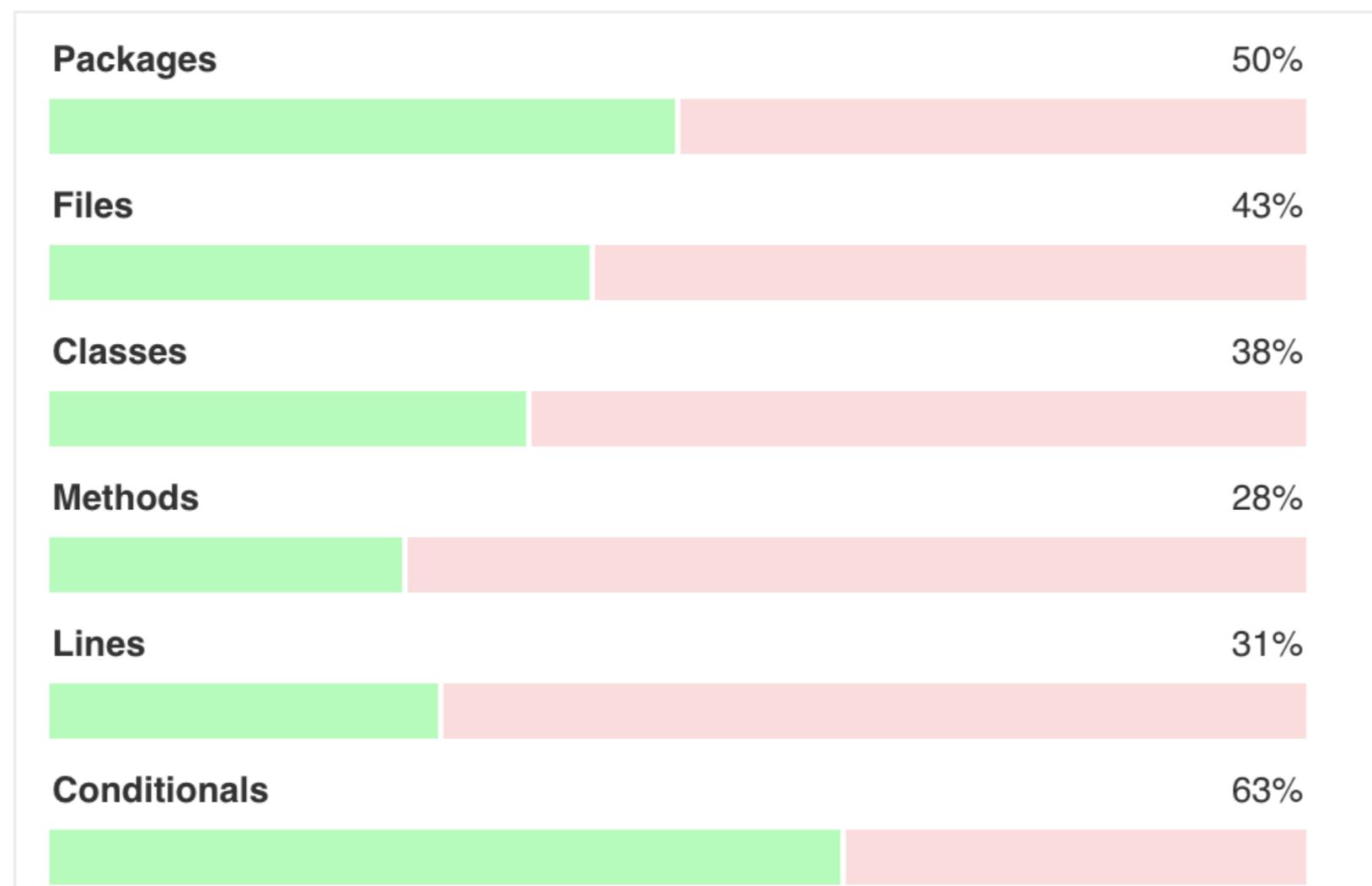
- [Last build \(#2\), 3 min 44 sec ago](#)
- [Last stable build \(#2\), 3 min 44 sec ago](#)
- [Last successful build \(#2\), 3 min 44 sec ago](#)
- [Last failed build \(#1\), 4 min 39 sec ago](#)
- [Last unsuccessful build \(#1\), 4 min 39 sec ago](#)
- [Last completed build \(#2\), 3 min 44 sec ago](#)



# See result of report

## Cobertura Coverage Report

### Trend



# See result of report

## Project Coverage summary

Name	Packages	Files	Classes	Methods	Lines	Conditionals
Cobertura Coverage Report	50% <div style="width: 50%; background-color: #28a745; display: inline-block;"></div> 2/4	43% <div style="width: 43%; background-color: #28a745; display: inline-block;"></div> 3/7	38% <div style="width: 38%; background-color: #28a745; display: inline-block;"></div> 3/8	28% <div style="width: 28%; background-color: #28a745; display: inline-block;"></div> 7/25	31% <div style="width: 31%; background-color: #28a745; display: inline-block;"></div> 22/70	63% <div style="width: 63%; background-color: #28a745; display: inline-block;"></div> 10/16

## Coverage Breakdown by Package

Name	Files	Classes	Methods	Lines	Conditionals
<a href="#">demo.dao</a>	50% <div style="width: 50%; background-color: #28a745; display: inline-block;"></div> 1/2	50% <div style="width: 50%; background-color: #28a745; display: inline-block;"></div> 1/2	50% <div style="width: 50%; background-color: #28a745; display: inline-block;"></div> 3/6	60% <div style="width: 60%; background-color: #28a745; display: inline-block;"></div> 12/20	33% <div style="width: 33%; background-color: #28a745; display: inline-block;"></div> 2/6
<a href="#">demo.network</a>	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/1	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/1	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/5	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/7	N/A
<a href="#">demo.service</a>	67% <div style="width: 67%; background-color: #28a745; display: inline-block;"></div> 2/3	50% <div style="width: 50%; background-color: #28a745; display: inline-block;"></div> 2/4	40% <div style="width: 40%; background-color: #28a745; display: inline-block;"></div> 4/10	42% <div style="width: 42%; background-color: #28a745; display: inline-block;"></div> 10/24	100% <div style="width: 100%; background-color: #28a745; display: inline-block;"></div> 8/8
<a href="#">demo.web</a>	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/1	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/1	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/4	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/19	0% <div style="width: 0%; background-color: #dc3545; display: inline-block;"></div> 0/2



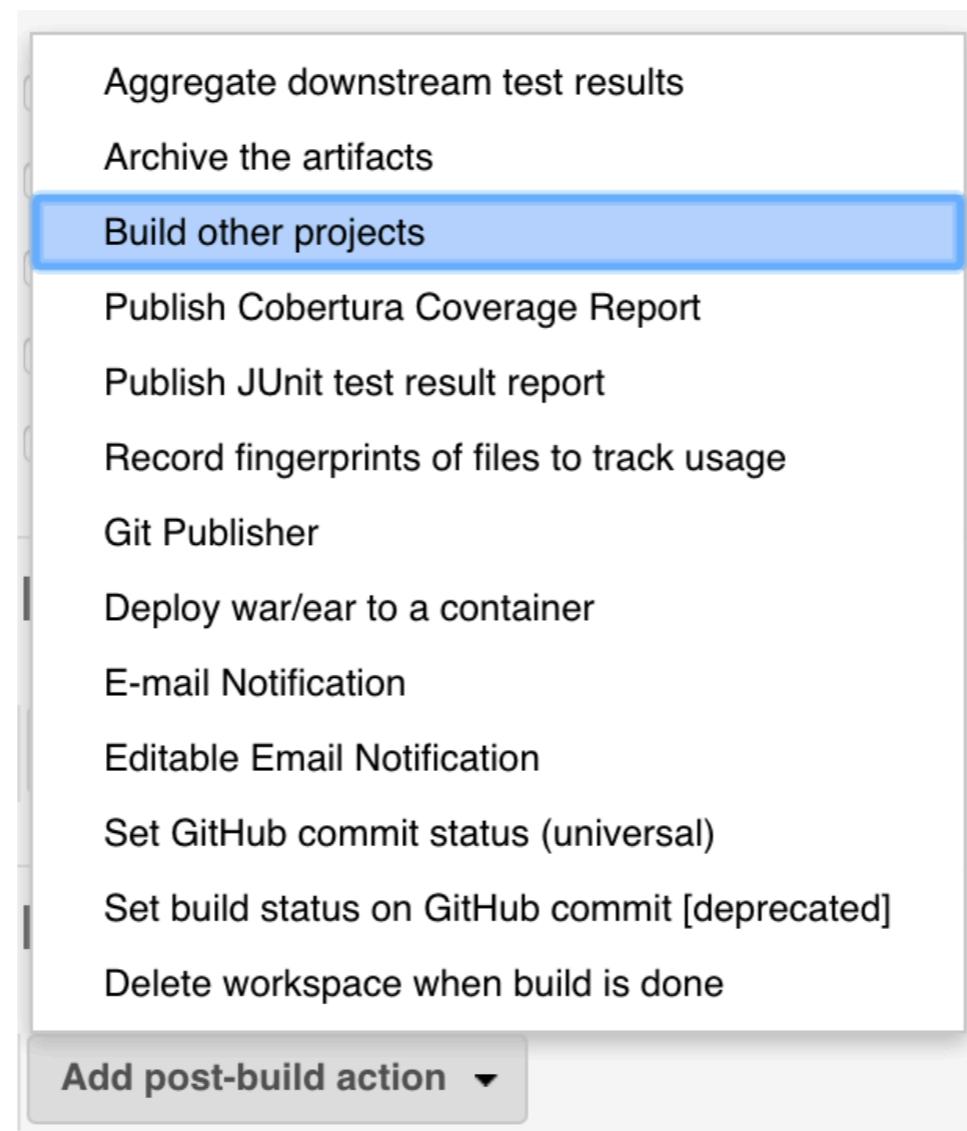
# One more thing !!



# Build other projects

Configure in job's step01\_pullcode

Post-build actions => Build other projects



# Build other projects

Specified project name = step02\_build

**Post-build Actions**

**Build other projects**

Projects to build step02\_build,

Trigger only if build is stable  
 Trigger even if the build is unstable  
 Trigger even if the build fails

Add post-build action ▾



# Try to build job = step01\_pullcode

Downstream projects ?

Jenkins ➤ step01\_pullcode ➤

-  Back to Dashboard
-  Status
-  Changes
-  Workspace
-  Build Now
-  Delete Project
-  Configure
-  Git Polling Log

**Project step01\_pullcode**

 [Workspace](#)

 [Recent Changes](#)

**Downstream Projects**

-  [step02\\_build](#)

 [Build History](#) [trend](#) ➤



# See step02\_build

Upstream projects ?

Jenkins > step02\_build >

Back to Dashboard  
 Status  
 Changes  
 Workspace  
 Build Now  
 Delete Project  
 Configure

**Project step02\_build**

[Workspace](#)  
 [Recent Changes](#)  
 [Latest Test Result \(no failures\)](#)

**Build History** [trend](#)

find

#	Date
#5	Feb 6, 2018 11:19 PM
#4	Feb 6, 2018 4:24 PM
#3	Feb 6, 2018 4:24 PM

**Upstream Projects**

[step01\\_pullcode](#)



# Need to see result ?



# Install Build pipeline plugin

Filter:

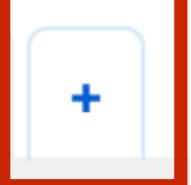
Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/>	<a href="#">Build Pipeline Plugin</a> This plugin provides build pipeline functionality to Hudson and Jenkins. This allows a chain of jobs to be visualised in a new view. Manual jobs in the pipeline can be triggered by a user with the appropriate permissions manually confirming.	1.5.8
<input type="checkbox"/>	<a href="#">CodeScene Plugin</a> CodeScene detects potential maintenance problems and early warnings in your codebase. The earlier you can react to those findings, the better. That's why CodeScene offers integration points that let you incorporate the analysis results into your build pipeline. This plugin lets you use CodeScene's Delta Analysis to catch potential problems before they are delivered to your main branch.	1.1.1

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 1 day 23 hr ago [Check now](#)



# Create Build pipeline in view

All		S	W	Name ↓	Last Success
				<a href="#">hello</a>	1 day 9 hr - #2
				<a href="#">step01_pullcode</a>	5 min 56 sec - #3
				<a href="#">step02_build</a>	5 min 47 sec - #5
				<a href="#">step03_coverage</a>	6 hr 49 min - #2



# Create Build pipeline in view

Fill in view name and choose build pipeline view

View name

**Build Pipeline View**  
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

**List View**  
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

**My View**  
This view automatically displays all the jobs that the current user has an access to.

**OK**



# Create Build pipeline in view

Choose a initial job (Start job)

Name

Description

[Plain text] [Preview](#) [?](#)

Filter build queue  [?](#)

Filter build executors  [?](#)

Build Pipeline View Title

---

**Pipeline Flow**

Layout  [▼](#)

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.

Upstream / downstream config

Select Initial Job ✓ hello

- step01\_pullcode
- step02\_build
- step03\_coverage
- step06\_analyze
- step07\_upload
- step08\_deploy

Standard build config

Use the default build config

Trigger Options

Build Cards

OK [Apply](#)



# Build pipeline in view

The screenshot shows the Jenkins interface for a build pipeline. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information (Somkiat Puisungnoen, log out). Below the navigation bar, the main title is "Build Pipeline". Underneath the title are several icons: Run, History, Configure, Add Step, Delete, and Manage. The main content area displays two pipeline steps. The first step, "step01\_pullcode", is highlighted with a green background and contains the following details:

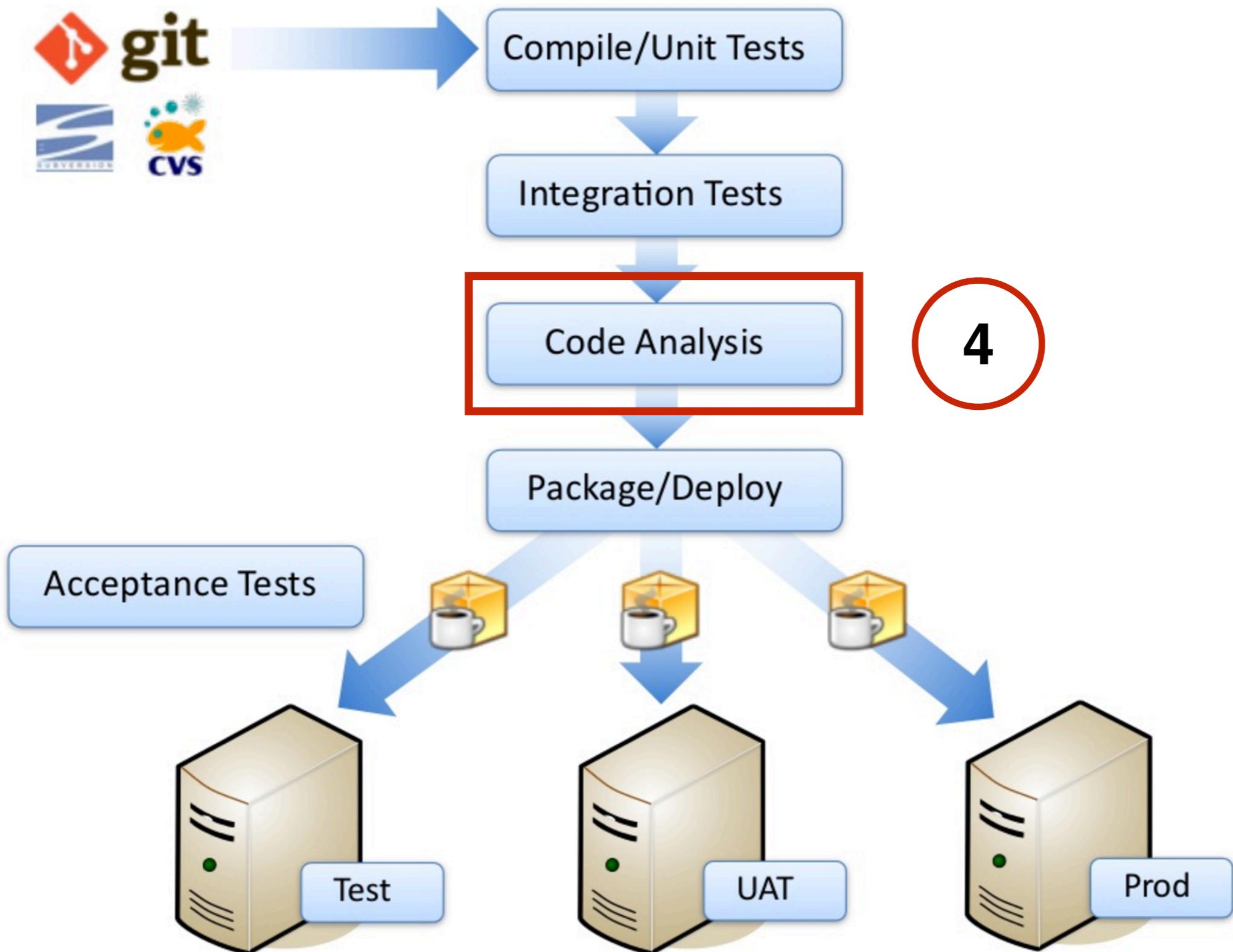
- Pipeline #3
- #3 step01\_pullcode
- Feb 6, 2018 11:19:18 PM
- 1.6 sec
- somkiat

A green arrow points from this step to the second step, "step02\_build". The second step also has a green background and contains:

- #5 step02\_build
- Feb 6, 2018 11:19:27 PM
- 10 sec

At the bottom right of each step, there are small icons for opening the step's configuration or history.





# 4. Code analysis

Working with SonarQube

Start SonarQube server



# Step 1 Install SonarQube plugin

The screenshot shows the Jenkins Manage Plugins interface. A red box highlights the search bar at the top right containing the text "sonarqube". Below the search bar, there are tabs: Updates, Available (which is selected), Installed, and Advanced. A table lists available plugins. The first plugin listed is "SonarQube Scanner for Jenkins" with version 2.6.1, which has a checked checkbox under "Install". The second plugin listed is "Mashup Portlets" with version 1.0.8, which has an unchecked checkbox under "Install". A detailed description for the Mashup Portlets plugin follows:

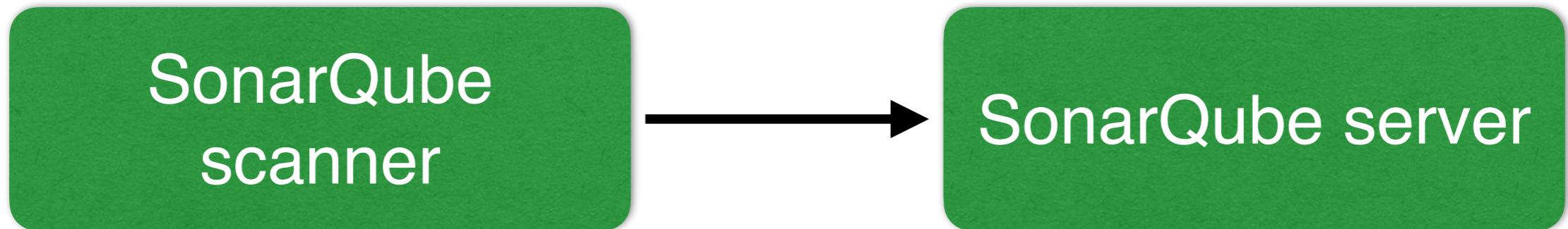
**Mashup Portlets**  
Additional Dashboard Portlets: Generic JS Portlet (lets you pull in arbitrary content via JS), Recent Changes Portlet (shows the SCM changes for a given job), SonarQube Portlets (show SonarQube statistics directly in Jenkins) and Test Results Portlet (shows the test results for a given job).

At the bottom, there are three buttons: "Install without restart", "Download now and install after restart" (which is highlighted in blue), and "Update information obtain".



# Step 2 Install SonarQube Scanner

Send analyzed result to SonarQube server



Run on Jenkins's Job



# Step 3 Install Sonar Scanner

 Scanners / Analyzing Source Code / Analyzing with SonarQube Scanner ...

## Analyzing with SonarQube Scanner

Created by OLD - Evgeny Mandrikov, last modified by Julien Henry on May 12, 2017

By [SonarSource](#) – GNU LGPL 3 – [Issue Tracker](#) – [Sources](#)

**Download SonarQube Scanner 3.0.3**

Compatible with SonarQube 5.6+ (LTS)

<a href="#">Linux 64 bit</a>	<a href="#">Windows 64 bit</a>	<a href="#">Mac OS X 64 bit</a>	<a href="#">Any*</a>
------------------------------	--------------------------------	---------------------------------	----------------------

\*This package expects that a JVM is already installed on the system - with same Java requirements as the SonarQube server.

**Table of Contents**

- [Features](#)
- [Installation](#)
- [Use](#)
- [Troubleshooting](#)
- [Going Further](#)

<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>



# Step 4.1 Configure SonarQube scanner

Manage Jenkins => Global Tool Configuration

## SonarQube Scanner for MSBuild

SonarQube Scanner for MSBuild installations

Add SonarQube Scanner for MSBuild

List of SonarQube Scanner for MSBuild installations on this system

## SonarQube Scanner

SonarQube Scanner installations

SonarQube Scanner

Name

Required

SONAR\_RUNNER\_HOME

Install automatically



Delete SonarQube Scanner

Add SonarQube Scanner

List of SonarQube Scanner installations on this system



# Step 4.2 Configure SonarQube server

## Manage Jenkins => Configuration System

**SonarQube servers**

---

Environment variables	<input type="checkbox"/> Enable injection of SonarQube server configuration as build environment variables If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.
SonarQube installations	Name <input type="text" value="sonarqube"/> Server URL <input type="text"/> Default is http://localhost:9000 Server version <input type="text" value="5.3 or higher"/> <span style="float: right;">▼</span> Configuration fields depend on the SonarQube server version. Server authentication token <input type="text"/> SonarQube authentication token. Mandatory when anonymous access is disabled. SonarQube account login <input type="text"/> SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3. SonarQube account password <input type="text"/> SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

[Advanced...](#)

[Delete SonarQube](#)



# Step 5 Create a new job

Job name = step06\_analyze

**Enter an item name**

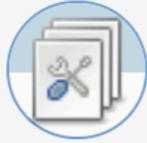
step06\_analyze

» *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.

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



# Step 5 Create a new job

## Tips copy from other project

if you want to create a new item from other existing, you can use this option:



Copy from

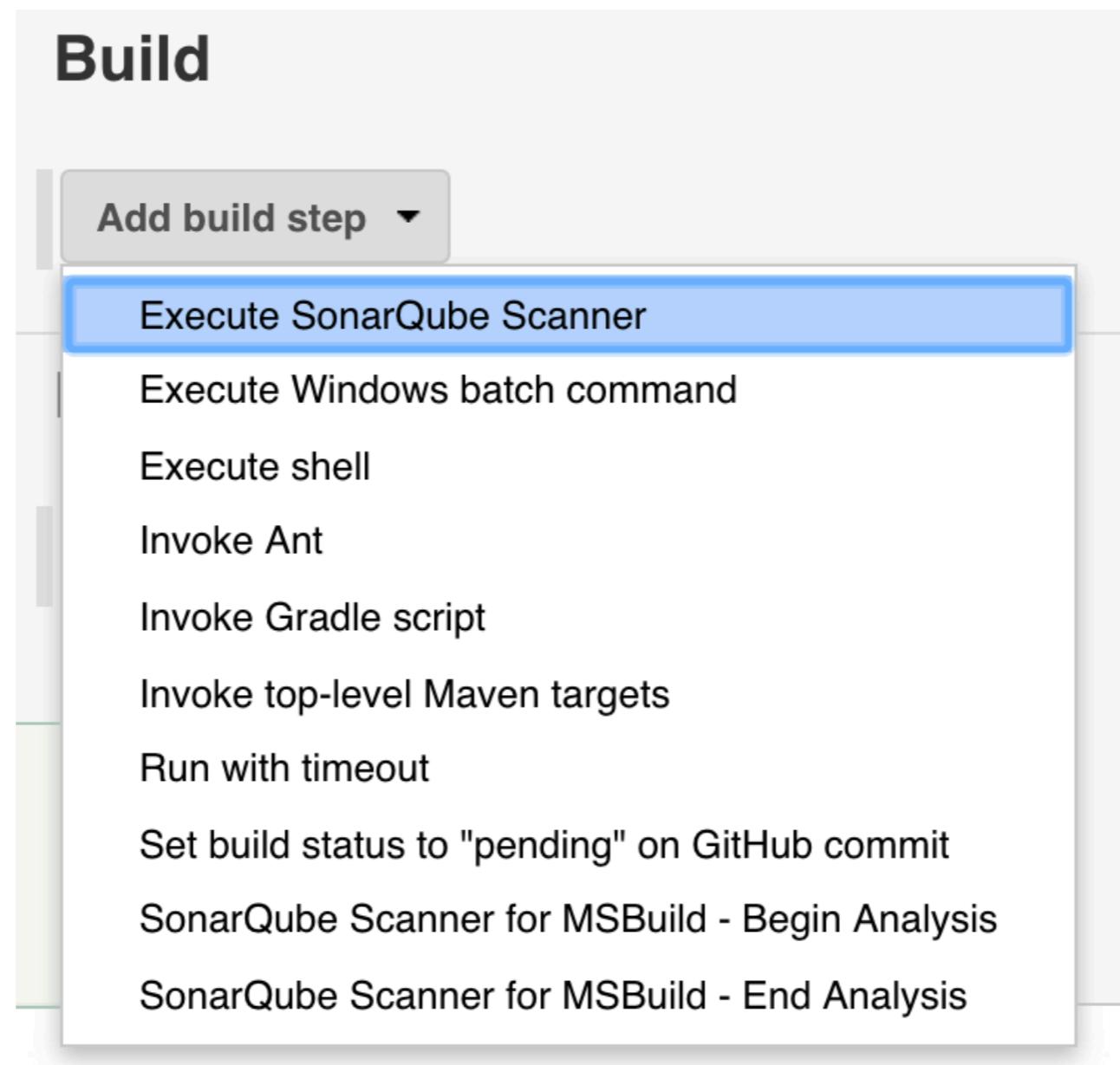
step02\_build

OK



# Step 6 Add SonarQube to Build step

Build => Execute SonarQube Scanner



# Step 6 Add SonarQube to Build step

## Configure your build

### Build

**Execute SonarQube Scanner**

**Task to run**  ?

**JDK**  ?

JDK to be used for this SonarQube analysis

**Path to project properties**  ?

**Analysis properties**  ?

**Additional arguments**  ▼ ?

**JVM Options**  ▼ ?

**Add build step ▾**



# Step 7 Add sonar-project.properties

Configuration file of project for SonarQube

## sonar-project.properties

```
# must be unique in a given SonarQube instance
sonar.projectKey=my:project
# this is the name and version displayed in the SonarQube UI. Was
sonar.projectName=My project
sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Replace '
# This property is optional if sonar.modules is set.
sonar.sources=.

# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8
```



# Example sonar-project.properties

sonar.projectKey=java-example

sonar.projectName=java-example

sonar.projectVersion=1.0

sonar.sources=src

sonar.java.source=src/main/java

sonar.java.binaries=target/classes

sonar.language=java

sonar.sourceEncoding=UTF-8

sonar.junit.reportPaths=target/surefire-reports



# Step 8 Run and See result

Jenkins

Jenkins > 4\_sonarqube >

- [Back to Dashboard](#)
- [Status](#)
- [Changes](#)
- [Workspace](#)
- [Build Now](#)
- [Delete Project](#)
- [Configure](#)
- [SonarQube](#)

**Project 4\_sonarqube**

SonarQube

Workspace

Recent Changes

**SonarQube Quality Gate**

my project **OK**

server-side processing: **Success**

**Upstream Projects**

2 compile unittest

Build History	trend
find	x
#4 Jun 15, 2017 10:33 PM	
#3 Jun 15, 2017 10:18 PM	
#2 Jun 15, 2017 10:17 PM	
#1 Jun 15, 2017 4:14 PM	



# Step 8 Run and See result

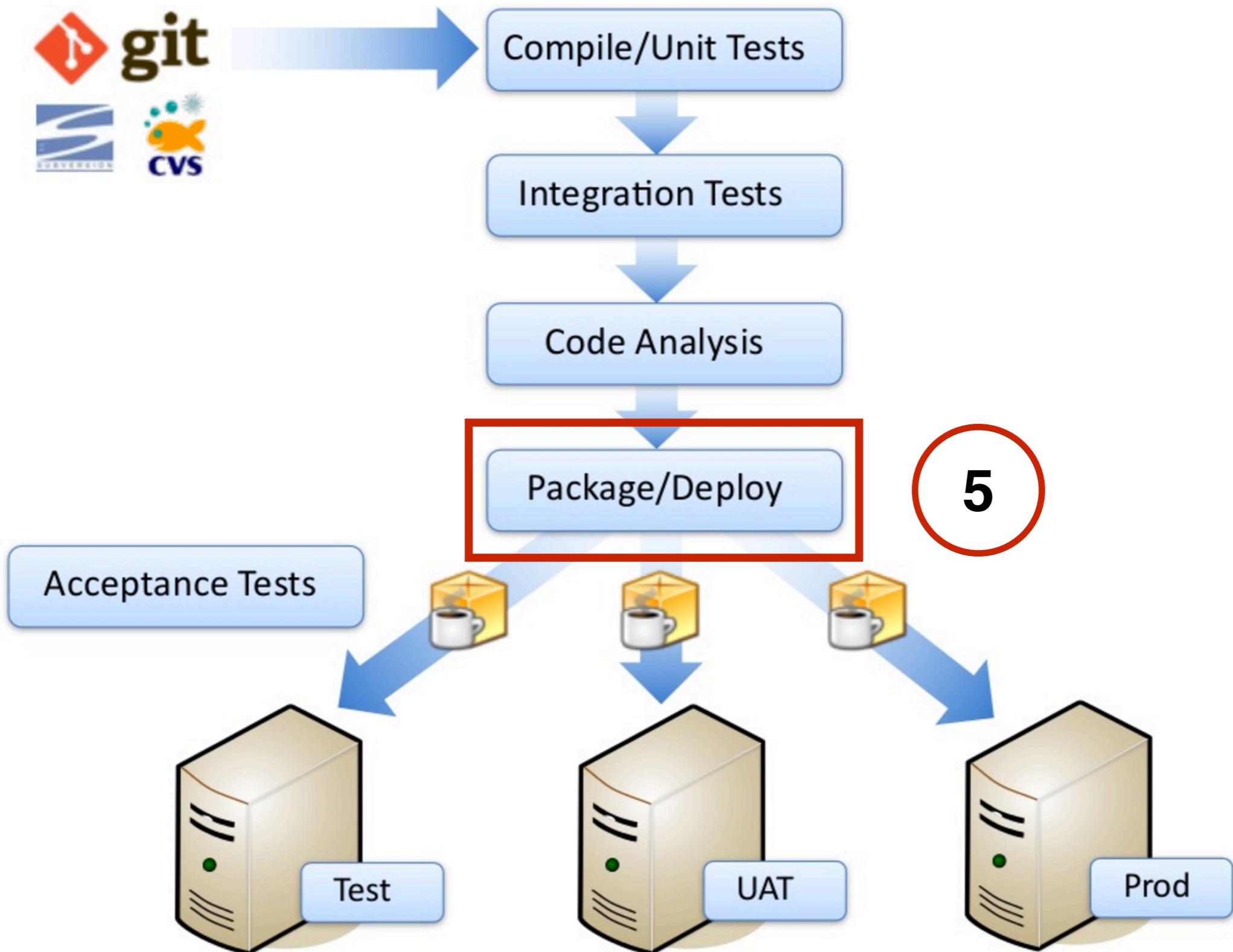
Result in Sonarqube server

PROJECTS				
QG	NAME ▲	VERSION	LOC	BUGS
★	✓  my project	1.0	172	0
1 results				



# Try it by yourself





# 5. Package and Deploy



# 5.1 Package and Deploy

Try to build WAR file and send to Nexus Repos

Try to deploy WAR file to WebServer



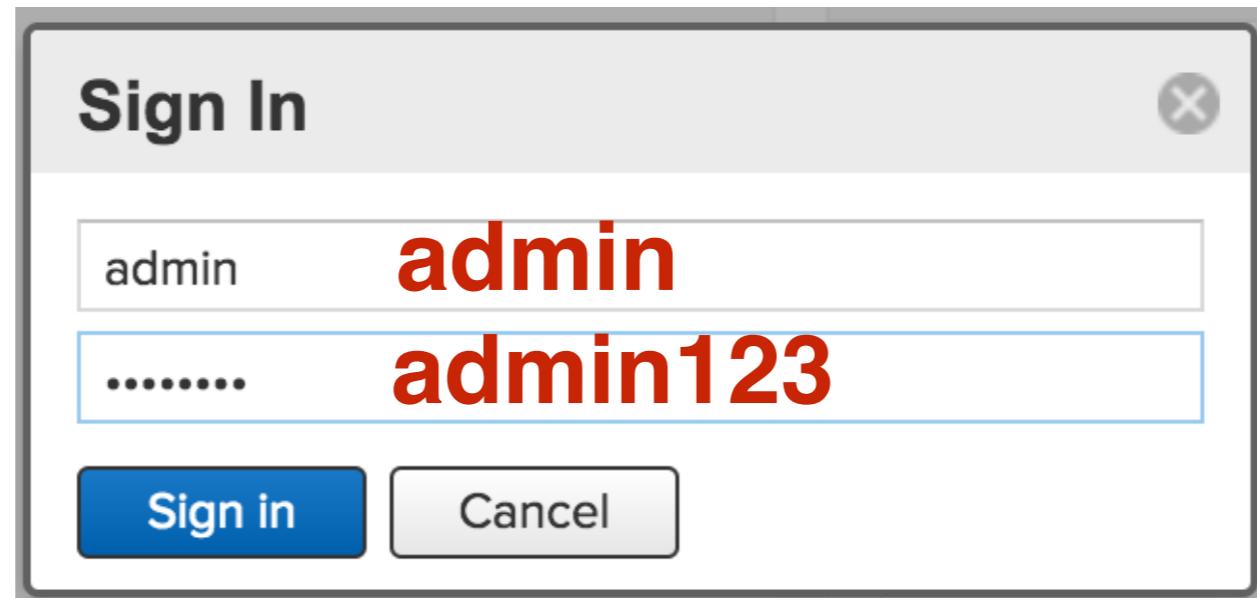
Nexus Repository



# Setup Nexus Repository



# Step 1 Try login to JFrog Artifactory



# Step 2 Go to administrator section

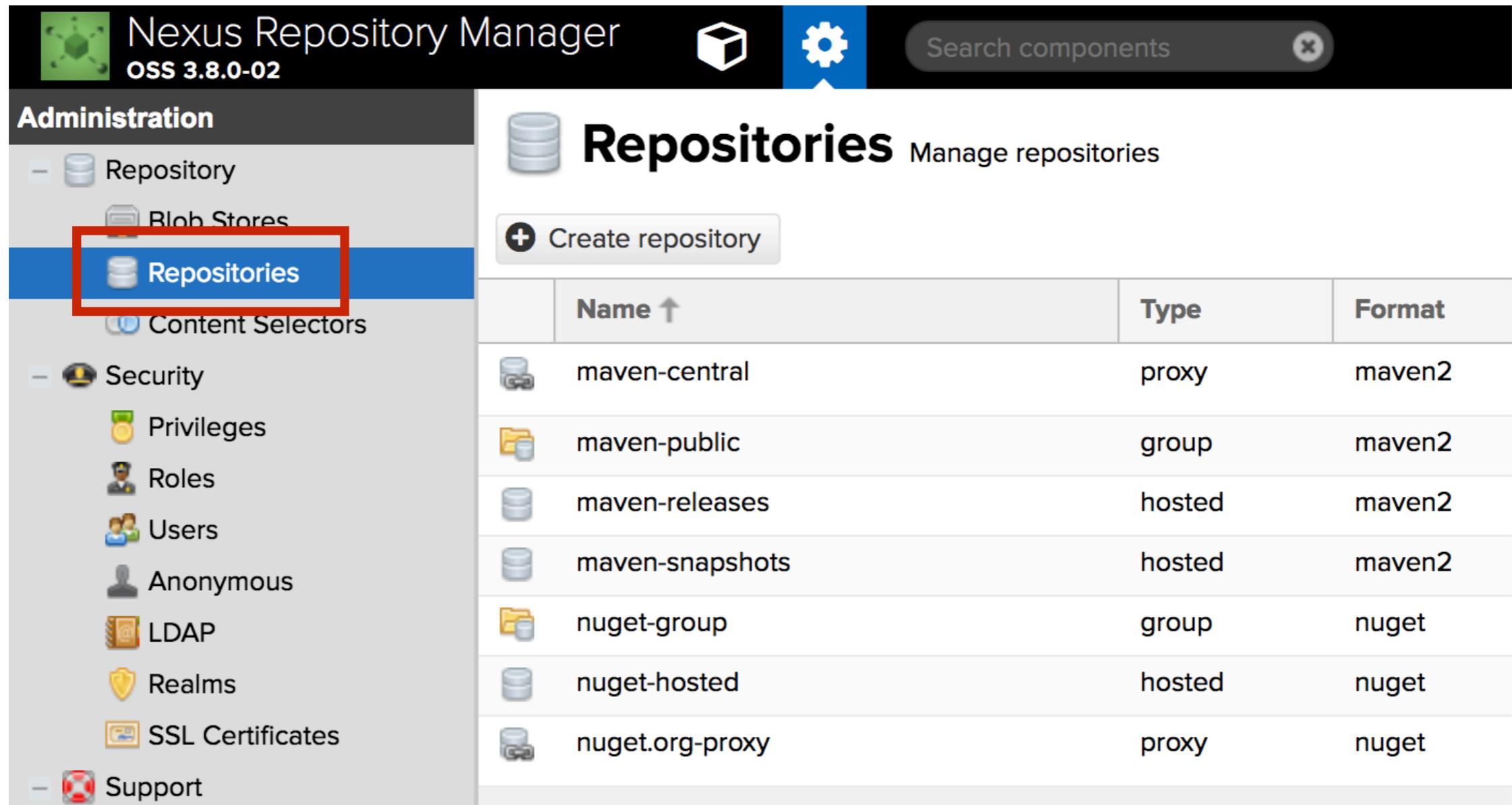
The screenshot shows the Nexus Repository Manager OSS 3.8.0-02 interface. The top navigation bar includes the logo, the text "Nexus Repository Manager OSS 3.8.0-02", a search bar labeled "Search components", and a gear icon which is highlighted with a red box. The left sidebar, titled "Administration", contains the following menu items:

- Repository** (selected, indicated by a blue background)
- Blob Stores
- Repositories
- Content Selectors
- Security**
  - Privileges
  - Roles
  - Users
  - Anonymous
  - LDAP
  - Realms
  - SSL Certificates
- Support**
  - Analytics

The main content area is titled "Repository" and "Repository administration". It features three large icons: "Blob Stores" (represented by a server icon), "Repositories" (represented by a stack of cylinders icon), and "Content Selectors" (represented by two overlapping circles icon).



# Step 3 Select repositories menu



The screenshot shows the Nexus Repository Manager interface. The top bar displays the logo and version "Nexus Repository Manager oss 3.8.0-02". Below the header is a navigation sidebar titled "Administration" with the following items:

- Repository
- Blob Stores
- Repositories** (highlighted with a red box)
- Content Selectors
- Security
  - Privileges
  - Roles
  - Users
  - Anonymous
  - LDAP
  - Realms
  - SSL Certificates
- Support

The main content area is titled "Repositories Manage repositories". It features a "Create repository" button and a table listing existing repositories:

Name ↑	Type	Format
maven-central	proxy	maven2
maven-public	group	maven2
maven-releases	hosted	maven2
maven-snapshots	hosted	maven2
nuget-group	group	nuget
nuget-hosted	hosted	nuget
nuget.org-proxy	proxy	nuget



# Step 4 Create repository

The screenshot shows the Nexus Repository Manager interface. The top navigation bar includes the logo, the text "Nexus Repository Manager oss 3.8.0-02", a search bar labeled "Search components", and two icons. The left sidebar, titled "Administration", contains several sections: "Repository", "Blob Stores", "Repositories" (which is selected and highlighted in blue), "Content Selectors", "Security", "Privileges", "Roles", "Users", "Anonymous", "LDAP", "Realms", "SSL Certificates", and "Support". The main content area is titled "Repositories" and "Manage repositories". It features a "Create repository" button with a red border. Below it is a table with columns "Name", "Type", and "Format". The table lists the following repositories:

Name	Type	Format
maven-central	proxy	maven2
maven-public	group	maven2
maven-releases	hosted	maven2
maven-snapshots	hosted	maven2
nuget-group	group	nuget
nuget-hosted	hosted	nuget
nuget.org-proxy	proxy	nuget



# Step 5 Choose raw (hosted)

Administration

- Repository
- Blob Stores
- Repositories**
- Content Selectors

- Security

- Privileges
- Roles
- Users
- Anonymous
- LDAP
- Realms
- SSL Certificates

- Support

- Analytics
- Logging
  - Log Viewer
- Metrics
- Support ZIP
- System Information

- System

- API
- Bundles
- Capabilities

**Repositories** / Select Recipe

Recipe ↑
maven2 (hosted)
maven2 (proxy)
npm (group)
npm (hosted)
npm (proxy)
nuget (group)
nuget (hosted)
nuget (proxy)
pypi (group)
pypi (hosted)
pypi (proxy)
raw (group)
<b>raw (hosted)</b>
raw (proxy)
rubygems (group)
rubygems (hosted)
rubygems (proxy)
yum (hosted)
yum (proxy)



# Step 6 Fill in name of repos

Administration

- Repository
- Blob Stores
- Repositories**
- Content Selectors

- Security
  - Privileges
  - Roles
  - Users
  - Anonymous
  - LDAP
  - Realms
  - SSL Certificates
- Support
  - Analytics
  - Logging
    - Log Viewer
  - Metrics
  - Support ZIP
  - System Information
- System
  - API

**Repositories** / **Select Recipe** / **Create Repository: pypi (hosted)**

**Name:** A unique identifier for this repository  
**demo\_web** **demo\_web**

**Online:**  If checked, the repository accepts incoming requests

**Storage**

**Blob store:**  
Blob store used to store asset contents  
**default**

**Strict Content Type Validation:**  
 Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

**Hosted**

**Deployment policy:**  
Controls if deployments of and updates to artifacts are allowed  
**Disable redeploy**

**Create repository** **Cancel**



# Step 7 Create new job

job name = step07\_upload

**Enter an item name**

step07\_upload

» A job already exists with the name 'step07\_upload'

---

-  **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.



# Step 8 Add build step

Choose Execute shell/windows batch command

Build

Execute shell

X ?

Command

```
curl -v --user 'admin:admin123' --upload-file ./demo.war \
http://localhost:8081/repository/demo_web/$BUILD_NUMBER/demo.war
```

See [the list of available environment variables](#)

Advanced...

Add build step ▾



# Step 8 Add build step

Choose Execute shell/windows batch command

**curl -v --user 'admin:admin123'**

**--upload-file ./demo.war**

**http://localhost:8081/repository/demo\_web/**

**\$BUILD\_NUMBER/demo.war**



# Step 9 Run and see result in nexus

The screenshot shows the Nexus Repository Manager interface. The top navigation bar includes the logo, the text "Nexus Repository Manager OSS 3.8.0-02", a gear icon, and a search bar labeled "Search components". The left sidebar, titled "Browse", has a "Raw" item selected, which is highlighted in blue. The main content area is titled "Raw" and says "Search for components in Raw repositories". It features a search bar for "Name" set to "Any" and a "More criteria" button. Below is a table with two rows:

Name ↑	Group
8/demo.war	/8
9/demo.war	/9



# If you need the plugins !!

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input type="checkbox"/>	<a href="#">Nexus Artifact Uploader</a> Nexus Artifact Uploader is to upload artifact to Nexus Repo	2.10
<input type="checkbox"/>	<a href="#">Maven Artifact ChoiceListProvider (Nexus)</a> A ChoiceListProvider to read Maven artifact information from a Nexus repository or MavenCentral	1.2.4
<input type="checkbox"/>	<a href="#">Nexus Task Runner Plugin</a>	0.9.2
<input type="checkbox"/>	<a href="#">Repository Connector</a> Repository Connector adds a build step which allows to resolve artifacts from a maven repository like nexus.	1.2.3
<input type="checkbox"/>	<a href="#">Nexus Platform Plugin</a>	1.6.20180123-131927.f506018

**Install without restart**    **Download now and install after restart**

Update information obtained: 1 ↗



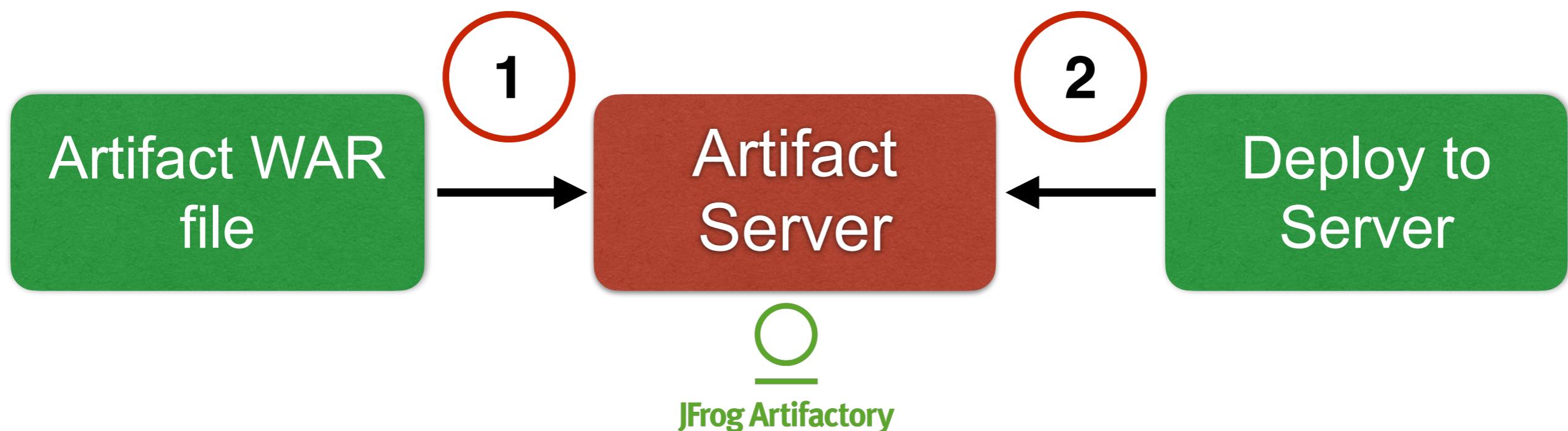
# Let's try by yourself



# 5.2 Package and Deploy

Try to build WAR file and send to Frog Artifactory

Try to deploy WAR file to WebServer



# Setup JFrog Artifactory



# Step 1 Try login to JFrog Artifactory

Welcome to JFrog Artifactory! X

Username \* **admin**

Password \* **password**

---

Remember me

**Log In**



# Step 2 Create local repository

The screenshot shows the JFrog Artifactory interface. On the left is a sidebar with icons for Home, Artifacts, Search, and User. The main area is titled "Local Repositories" and shows one repository named "example-repo-local" of type "Generic". A context menu is open on the right, titled "Create Repositories", with options: "Quick Setup", "Local Repository" (which is highlighted with a red box), "Remote Repository", "Virtual Repository", and "Distribution Repository". Below the menu are links for "Add User", "Add Group", "Add Permission", "Edit Profile", and "Log Out".

JFrog Artifactory

Welcome, admin

Help

Local Repositories

1 Repository

Filter by Repository Key

Repository Key	Type	Reca
example-repo-local	Generic	

Create Repositories

- Quick Setup
- Local Repository**
- Remote Repository
- Virtual Repository
- Distribution Repository

Add User

Add Group

Add Permission

Edit Profile

Log Out



# Step 3 Choose Generic repos

The screenshot shows the JFrog Artifactory interface with a modal dialog titled "Select Package Type". The dialog lists various package types, each represented by an icon and a star rating. A red box highlights the "Generic" option, which features a database icon. The modal has "Cancel" and "Save & Finish" buttons at the bottom.

Package Type	Rating
Bower	★
Chef	★
CocoaPods	★
Conan	★
debain	★
docker	★
Gems	★
Git LFS	★
Gradle	★
Ivy	★
Maven	★
npm	★
NuGet	★
Opkg	★
PHP Composer	★
PyPI	★
Puppet	★
SBT	★
VAGRANT	★
RPM	★
Generic	★



# Step 3 Choose Generic repos

Fill in name of repo = dev

The screenshot shows the 'New Local Repository' configuration interface in JFrog Artifactory. The 'Basic' tab is selected. In the 'Package Type \*' section, 'Generic' is chosen. The 'Repository Key \*' field contains 'dev' and is highlighted with a red box. At the bottom right, the 'Save & Finish' button is also highlighted with a red box.



# Step 4 Install Artifactory plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/> <a href="#">Artifactory Plugin</a>	Integrates Artifactory to Jenkins	2.11.0

[Install without restart](#) [Download now and install after restart](#) Update information obtained



# Step 5 Configure Artifactory plugin

Manage Jenkins => Configure System

**Artifactory**

Enable Push to Bintray [?](#)

Use the Credentials Plugin [?](#)

**Artifactory servers**

 **Artifactory**

Server ID  [?](#)

URL  [?](#)

**Default Deployer Credentials**

Username  [?](#)

Password  [?](#)

[Advanced...](#)

Found Artifactory 5.3.2 [Test Connection](#)

Use Different Resolver Credentials

[Delete](#)

[Add Artifactory Server](#)

List of Artifactory servers that projects will want to deploy artifacts and build info to

Enable Build-Info proxy for Docker images

[Save](#) [Apply](#)



# Step 6 Add to build steps

Build Environments => Generic-Artifactory Integration

**Build Environment**

- Delete workspace before build starts
- Provide Configuration files
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Ant/Ivy-Artifactory Integration
- Create Delivery Pipeline version
- Generic-Artifactory Integration
- Gradle-Artifactory Integration
- Maven3-Artifactory Integration
- Use secret text(s) or file(s)



# Step 6 Add to build steps

## Configure job to upload

**Artifactory Configuration**

Download and upload by  Specs  Legacy patterns (deprecated)

**Upload Details**

Artifactory upload server `http://localhost:8081/artifactory`

Target Repository `dev` ? Different Value Refresh Repositories

Override default credentials

Published Artifacts `target/*.war=>${BUILD_NUMBER}` ?

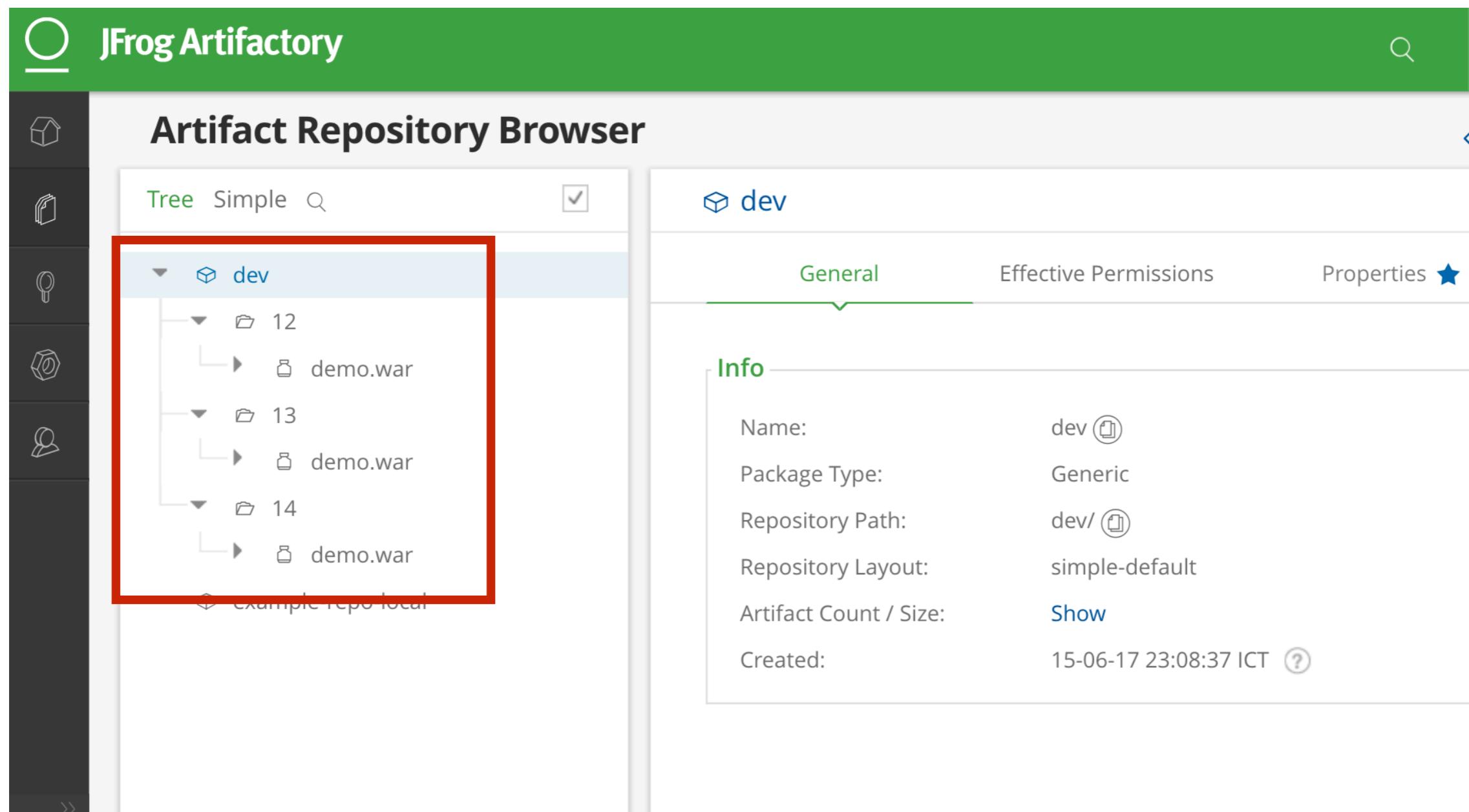
**target/\*.war=>\${BUILD\_NUMBER}**

Upload properties ?



# Step 7 Run and see output

See in Artifactory Server



The screenshot shows the JFrog Artifactory interface. On the left, there's a sidebar with icons for Home, Repositories, Search, and User. The main area is titled "Artifact Repository Browser". It has tabs for "Tree", "Simple", and "Search". The "Tree" tab is selected. A red box highlights the tree view which shows a repository named "dev". Inside "dev", there are three sub-folders: "12", "13", and "14", each containing a "demo.war" artifact. Below the tree, it says "Example repo local". To the right, there's a detailed view of the "dev" repository. The "General" tab is selected. The "Info" section contains the following details:

Name:	dev
Package Type:	Generic
Repository Path:	dev/
Repository Layout:	simple-default
Artifact Count / Size:	Show
Created:	15-06-17 23:08:37 ICT



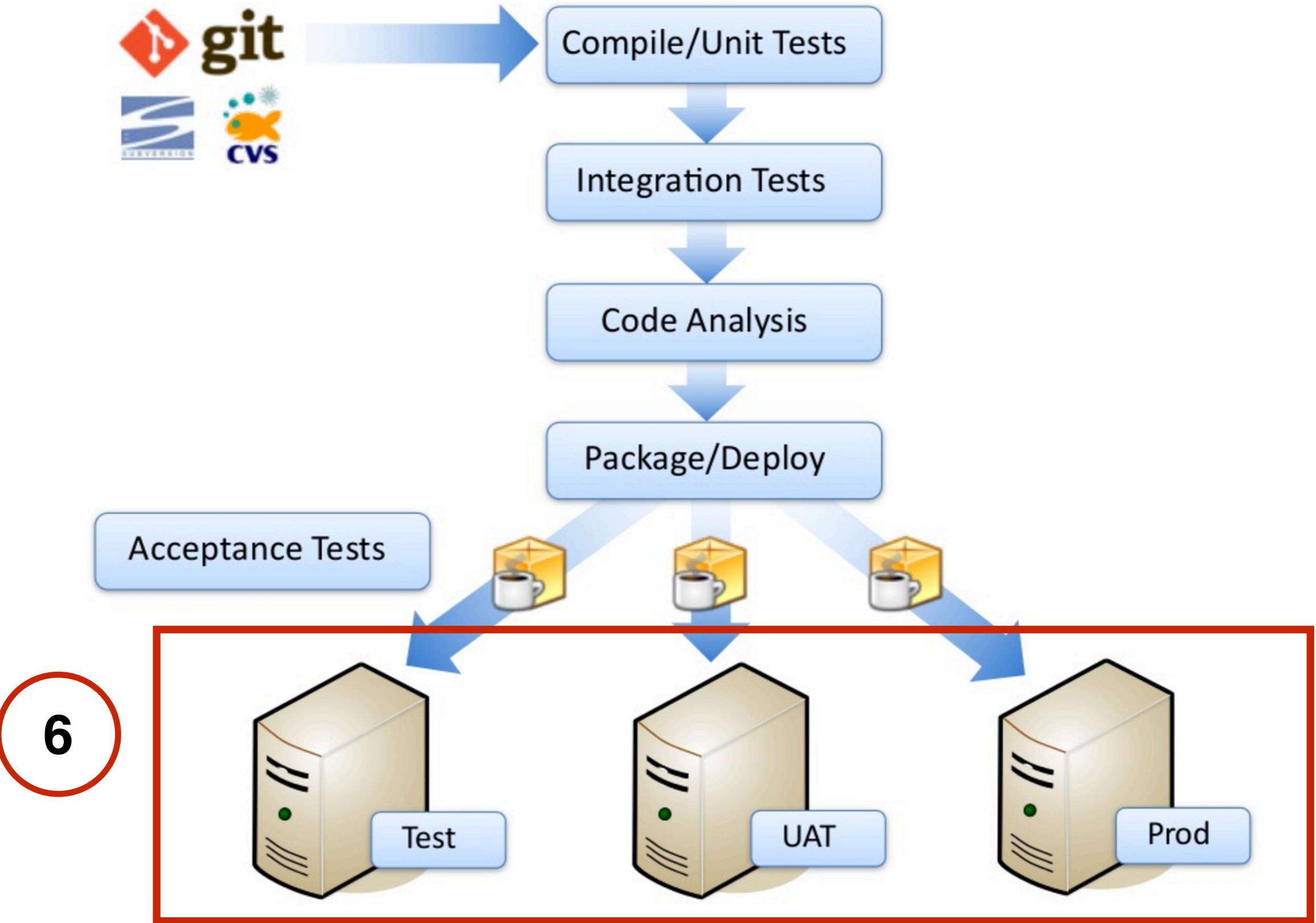
# Step 7 Run and see output

See in Jenkins's job

The screenshot shows the Jenkins interface for the project '5\_create\_war\_file'. The left sidebar contains links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Artifactory Build Info'. The main content area is titled 'Project 5\_create\_war\_file'. It features a red box highlighting the 'Artifactory Build Info' section, which includes links to 'Artifactory Build Info', 'Workspace', and 'Recent Changes'. Below this is the 'Upstream Projects' section, which lists '2\_compile\_unittest'. The 'Permalinks' section provides links to various build logs: Last build (#14), Last stable build (#14), Last successful build (#14), Last failed build (#8), Last unsuccessful build (#8), and Last completed build (#14). On the left, there is a 'Build History' table with rows for builds #14 through #9, each with a timestamp and a set of three circular icons.

#	Build Number	Date	Actions
14	#14	Jun 15, 2017 11:44 PM	○ Ô ↴
13	#13	Jun 15, 2017 11:44 PM	○ Ô ↴
12	#12	Jun 15, 2017 11:40 PM	○ Ô ↴
11	#11	Jun 15, 2017 11:38 PM	○ Ô ↴
10	#10	Jun 15, 2017 11:28 PM	○ Ô ↴
9	#9	Jun 15, 2017 11:27 PM	○ Ô ↴





# 6. Deploy to other server

Try to deploy WAR file to WebServer

**Apache Tomcat**

**JBoss**

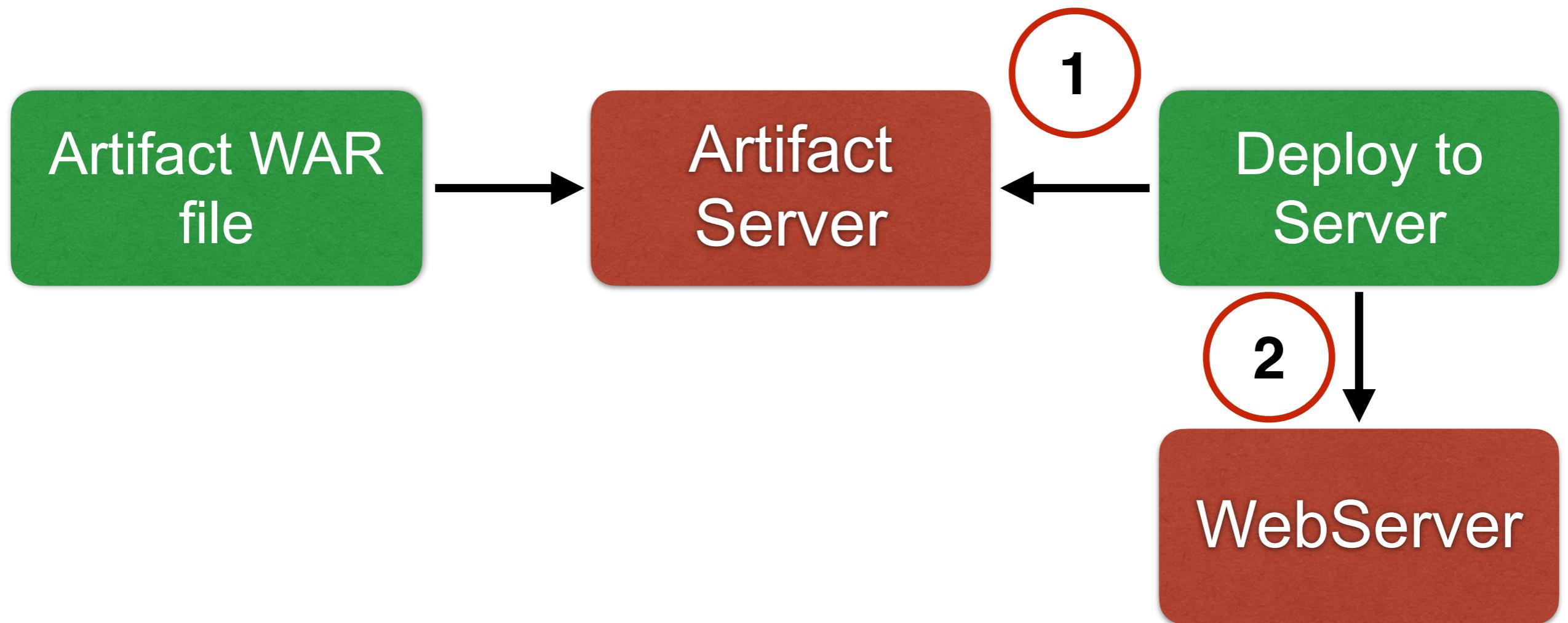
**IBM Websphere**

**Web Logic**



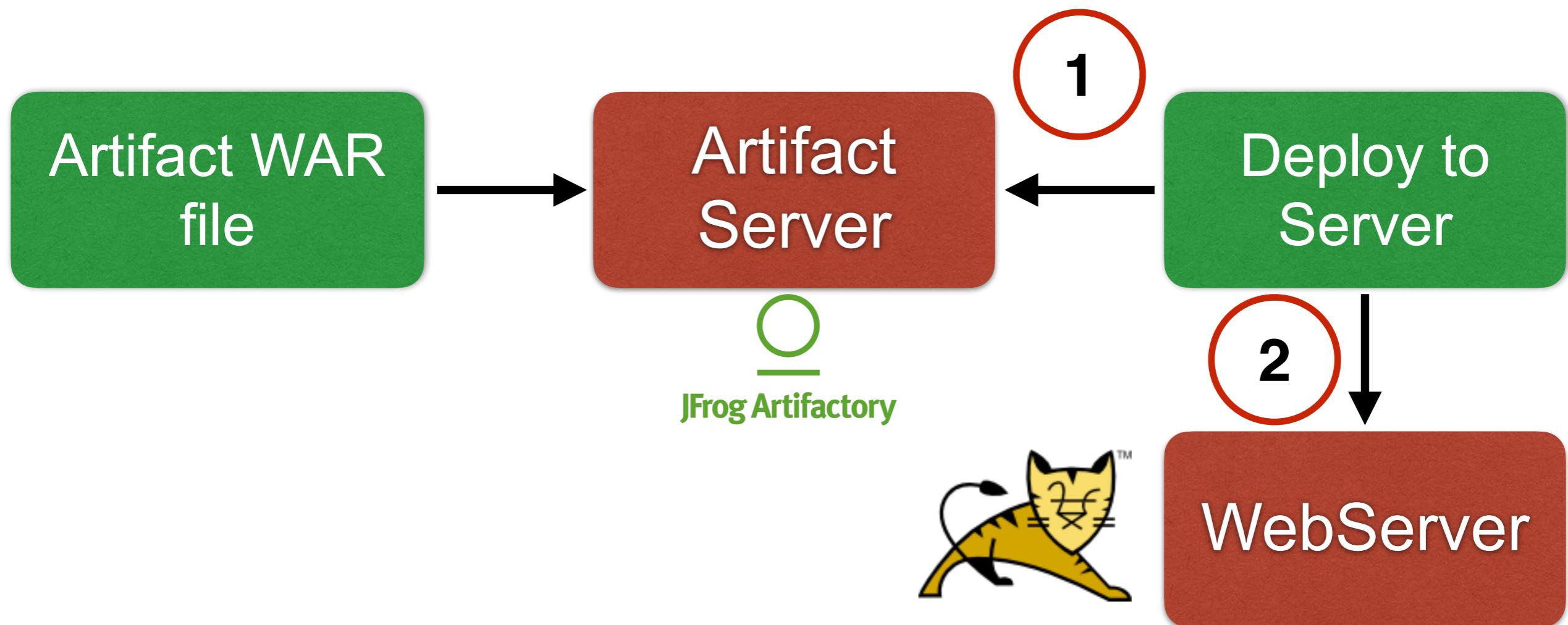
# 6. Deploy to other server

Deploy WAR file from Artifact Server to Web Server



# 6. Deploy to other server

Deploy WAR file from Artifact Server to Web Server



# Deploy to container plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input checked="" type="checkbox"/> <a href="#">Deploy to container Plugin</a>		1.10

[Install without restart](#) [Download now and install after restart](#)

Update information obtained: 2 hr 16 mi



# Add to Post-build actions

Post-build actions => Deploy war/ear to a container

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish Cobertura Coverage Report
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- Deploy war/ear to a container**
- 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 ▾**



# Deploy war/ear to a container

Specify WAR/EAR file to deploy

## Post-build Actions

Deploy war/ear to a container X

WAR/EAR files	<input type="text" value="demo.war"/> <span style="color: blue;">?</span>
Context path	<input type="text"/> <span style="color: blue;">?</span>
Containers	<input type="button" value="Add Container ▾"/>
Deploy on failure	<input type="checkbox"/>

[Add post-build action ▾](#)



# Where is my WAR file ?



# Download WAR file from Artifactory

## Add step in build step of job

### Build

Execute shell

X ?

Command

```
wget --output demo.war -q <url of JFrog artifactory server>/demo.war
```

**wget -O demo.war -q**

**<url of artifact server>/demo.war**

See [the list of available environment variables](#)

Advanced...

Add build step ▾



# Deploy war/ear to a container

Choose a container to deploy

**Post-build Actions**

**Deploy war/ear to a container**

WAR/EAR files: demo.war

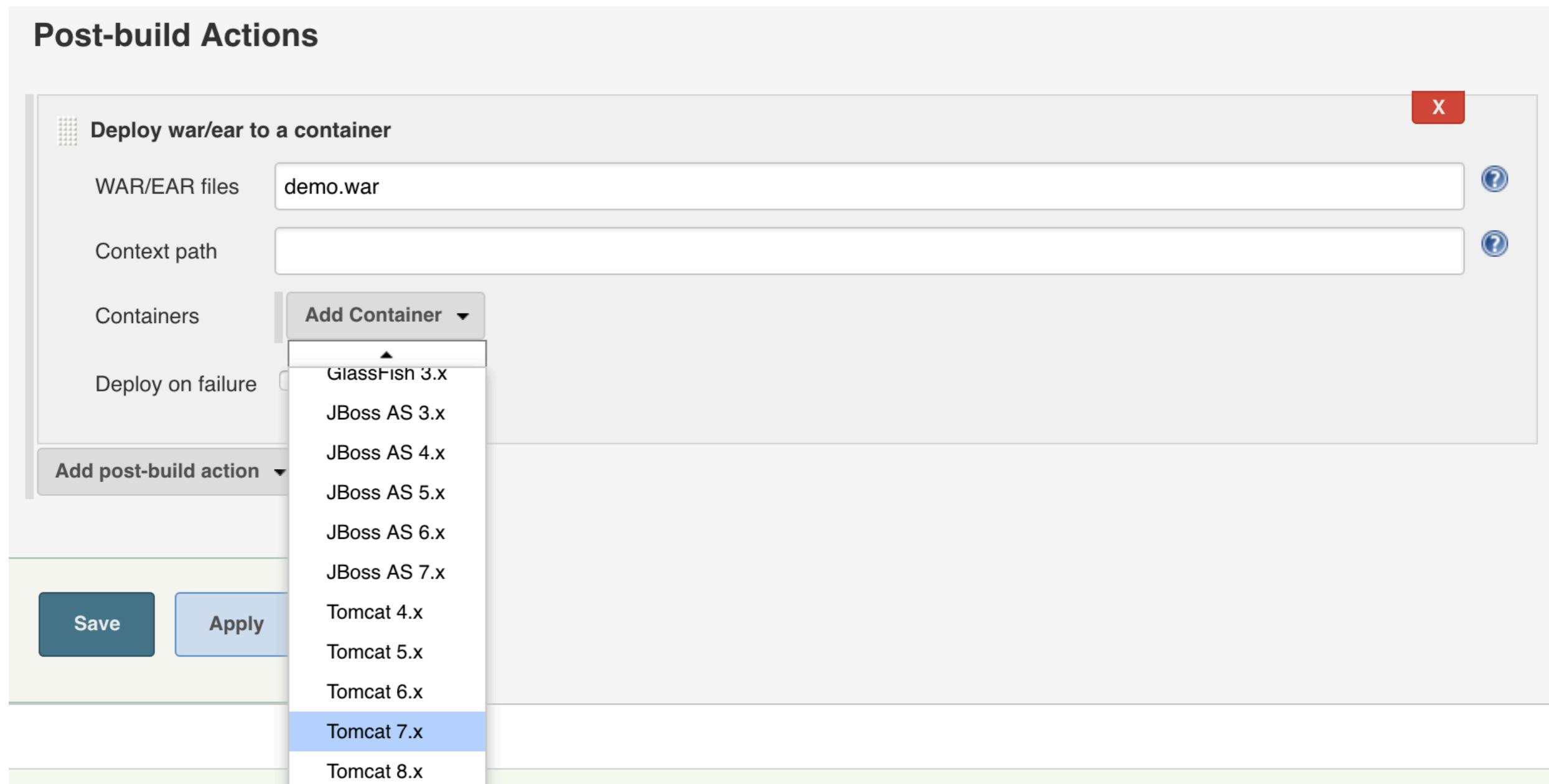
Context path:

Containers: Add Container ▾

- GlassFish 3.x
- JBoss AS 3.x
- JBoss AS 4.x
- JBoss AS 5.x
- JBoss AS 6.x
- JBoss AS 7.x
- Tomcat 4.x
- Tomcat 5.x
- Tomcat 6.x
- Tomcat 7.x**
- Tomcat 8.x

Add post-build action ▾

Save    Apply



# Deploy war/ear to a container

## Config credential and Tomcat URL

**Post-build Actions**

**Deploy war/ear to a container**

WAR/EAR files: demo.war

Context path:

Containers:

**Tomcat 8.x**

Credentials: - none -

Tomcat URL: http://localhost:8088/

Add Container

Deploy on failure

Add post-build action



# Deploy fail

Need username and password ?



## Console Output

```
Started by user Somkiat Puisungnoen
Building on master in workspace /Users/somkiat/Downloads/set/workspace/step08_deploy
[step08_deploy] $ /bin/sh -xe
/var/folders/t5/8kg23s_97z9dw44tfc1d6dqw000gn/T/jenkins3211027651275797278.sh
+ wget -O demo.war -q http://localhost:8081/repository/demo\_web/8/demo.war
Deploying /Users/somkiat/Downloads/set/workspace/step08_deploy/demo.war to container Tomcat 8.x Remote
with context
ERROR: Build step failed with exception
org.codehaus.cargo.container.ContainerException: The [cargo.remote.username] and [cargo.remote.password]
properties are mandatory and need to be defined in your configuration.
    at
org.codehaus.cargo.container.tomcat.internal.AbstractTomcatManagerDeployer.createManager(AbstractTomcatM
anagerDeployer.java:318)
    at
org.codehaus.cargo.container.tomcat.internal.AbstractTomcatManagerDeployer.getTomcatManager(AbstractTomc
atManagerDeployer.java:83)
    at
org.codehaus.cargo.container.tomcat.internal.AbstractTomcatManagerDeployer.redeploy(AbstractTomcatManage
rDeployer.java:173)
    at hudson.plugins.deploy.CargoContainerAdapter.deploy(CargoContainerAdapter.java:77)
    at
```



# Setup Apache Tomcat

Add user and roles in file tomcat-users.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>

    <role rolename="manager-script"/>
    <user username="deployer" password="password"
roles="manager-script"/>

</tomcat-users>
```



# Add credential for deploy

 Jenkins Credentials Provider: Jenkins

 Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: deployer

Password: .....

ID:

Description:

**Add** **Cancel**



# Add credential for deploy

**Post-build Actions**

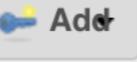
**Deploy war/ear to a container**

WAR/EAR files: demo.war

Context path:

Containers

**Tomcat 8.x**

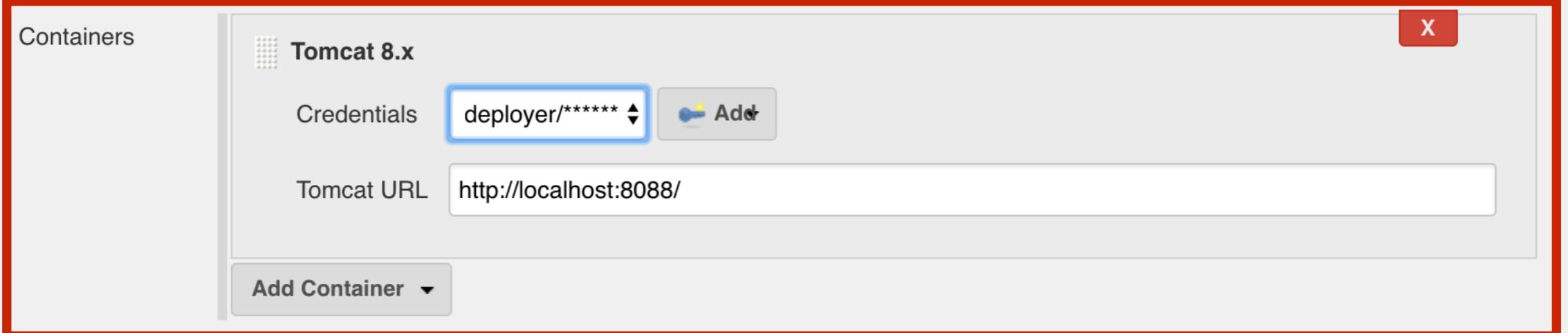
Credentials: **deployer/\*\*\*\*\*\*\*\*\*** 

Tomcat URL: http://localhost:8088/

Add Container ▾

Deploy on failure

Add post-build action ▾



# Try to build again and see output



## Console Output

```
Started by user Somkiat Puisungnoen
Building on master in workspace /Users/somkiat/Downloads/set/workspace/step08_deploy
[step08_deploy] $ /bin/sh -xe /var/folders/t5/8kg23s_97z9dw44tfc1d6dqw0000gn/T/jenkins8009642458152071862.sh
+ wget -O demo.war -q http://localhost:8081/repository/demo\_web/8/demo.war
Deploying /Users/somkiat/Downloads/set/workspace/step08_deploy/demo.war to container Tomcat 8.x Remote with
context
[/Users/somkiat/Downloads/set/workspace/step08_deploy/demo.war] is not deployed. Doing a fresh deployment.
Deploying [/Users/somkiat/Downloads/set/workspace/step08_deploy/demo.war]
Finished: SUCCESS
```

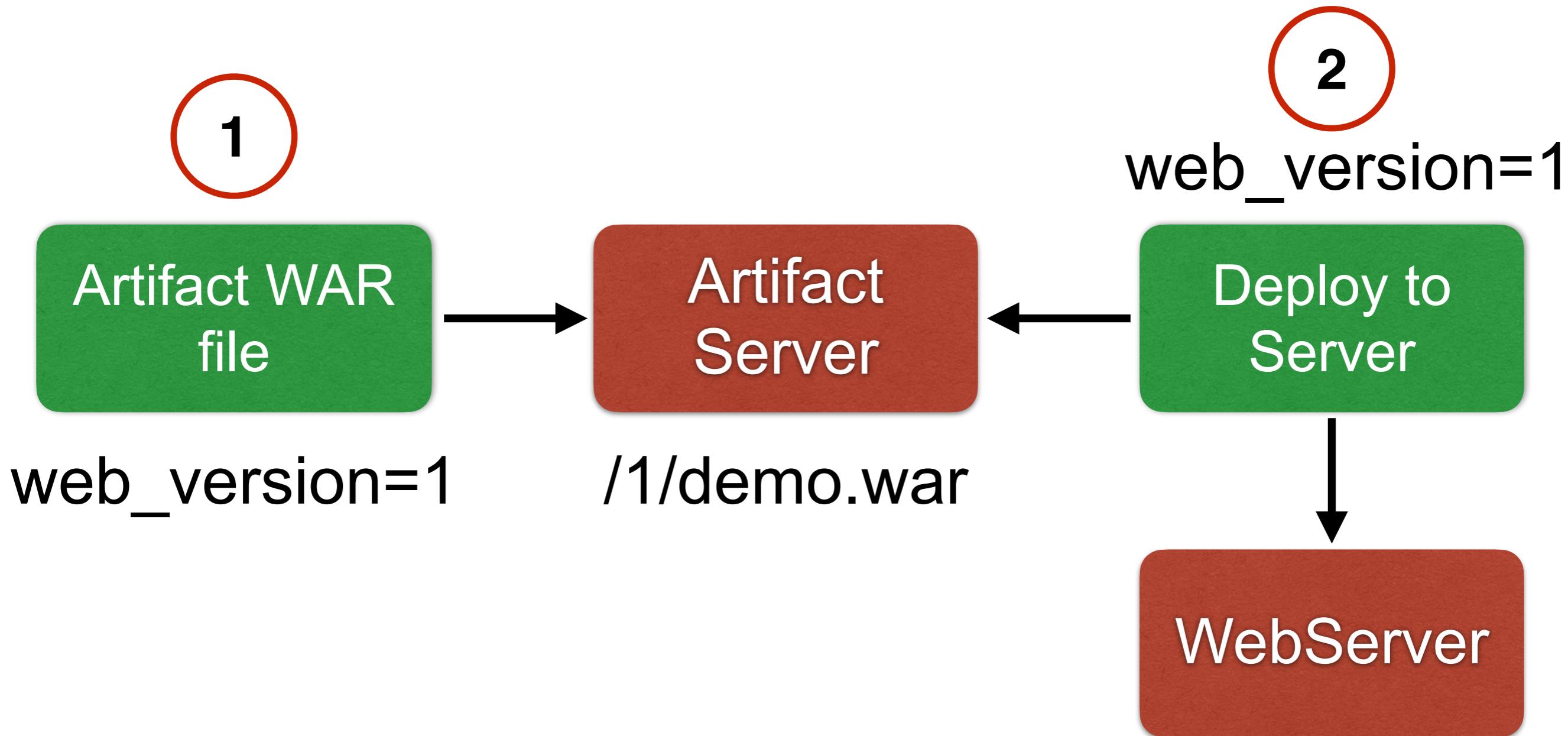


# **How to manage version of WAR file ?**



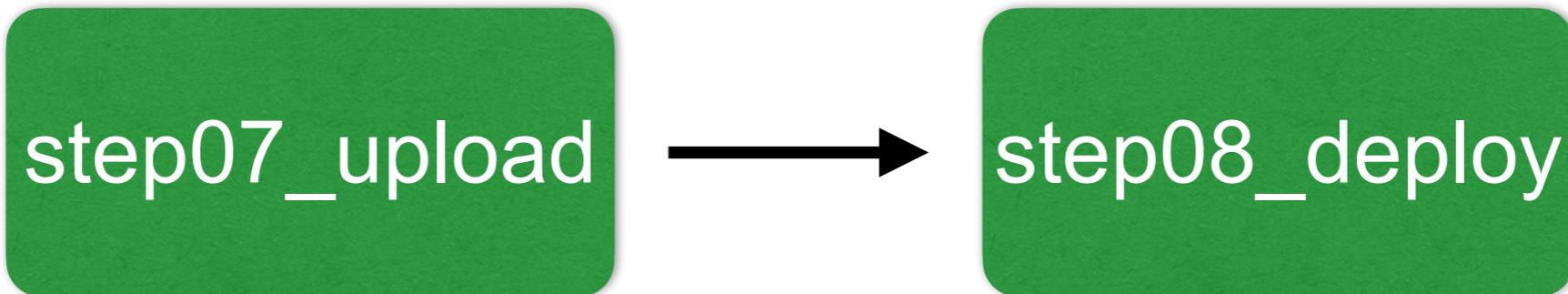
# Manage version number

Send data/value between jobs ?



# Manage version number

Send data/value between jobs ?



`web_version=1`

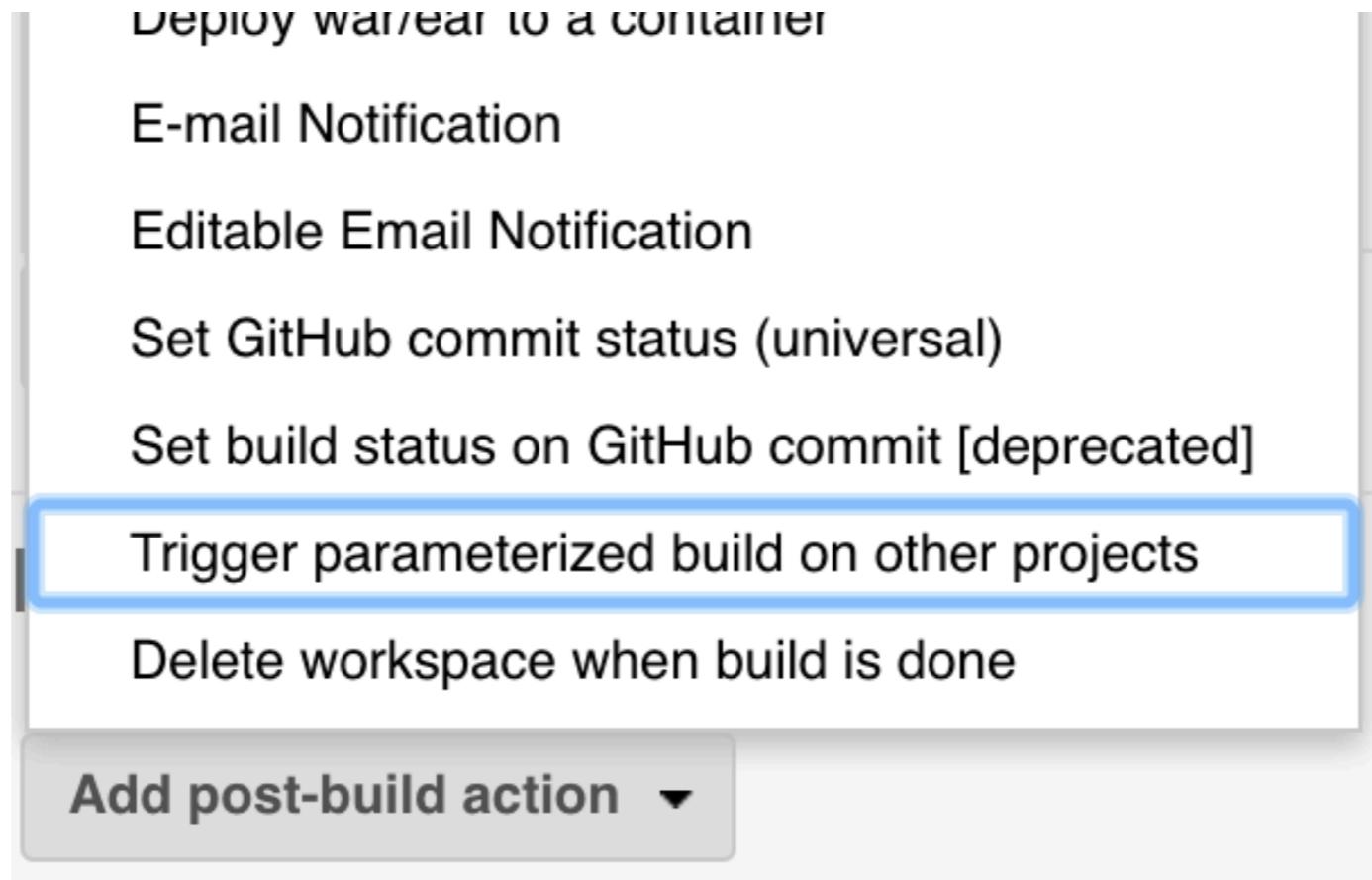


**In job = step07\_upload**



# Add post-build actions

In step07\_upload



A screenshot of a Jenkins interface showing a dropdown menu for 'Add post-build action'. The menu lists several options: 'Deploy war/ear to a container', 'E-mail Notification', 'Editable Email Notification', 'Set GitHub commit status (universal)', 'Set build status on GitHub commit [deprecated]', 'Trigger parameterized build on other projects' (which is highlighted with a blue border), and 'Delete workspace when build is done'. Below the menu is a button labeled 'Add post-build action ▾'.

- Deploy war/ear to a container
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Trigger parameterized build on other projects
- Delete workspace when build is done

Add post-build action ▾



# Add post-build actions

Add predefined parameters to send to other jobs

**Post-build Actions**

**Trigger parameterized build on other projects**

**Build Triggers**

Projects to build: step08\_deploy, **Blank project name in the list**

Trigger when build is: Stable

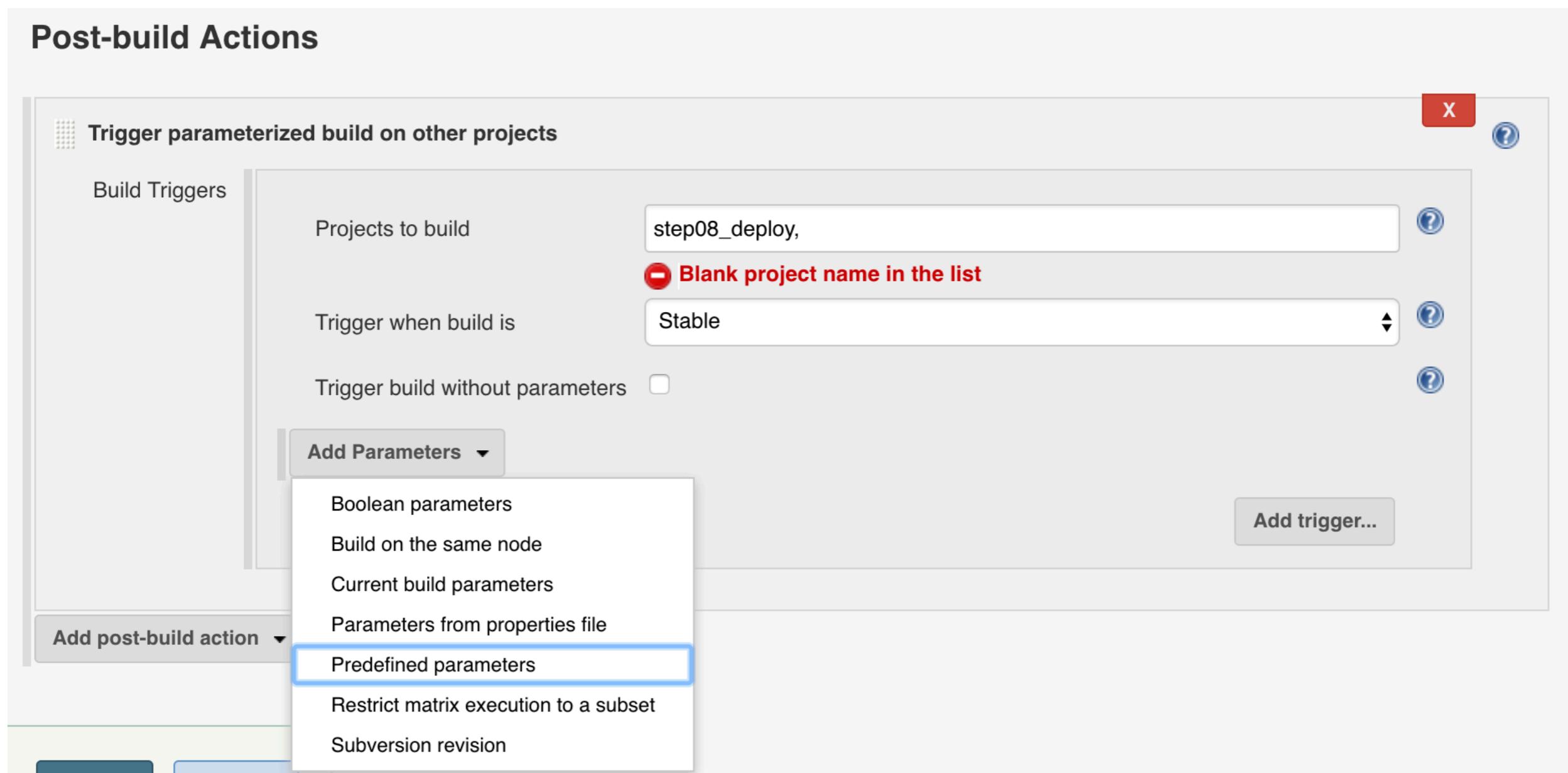
Trigger build without parameters:

**Add Parameters** ▾

- Boolean parameters
- Build on the same node
- Current build parameters
- Parameters from properties file
- Predefined parameters**
- Restrict matrix execution to a subset
- Subversion revision

**Add trigger...**

**Add post-build action** ▾



# Add post-build actions

We send **web\_version=\$BUILD\_NUMBER**

**Post-build Actions**

**Trigger parameterized build on other projects**

**Build Triggers**

Projects to build: step08\_deploy,  
**Blank project name in the list**

Trigger when build is: Stable

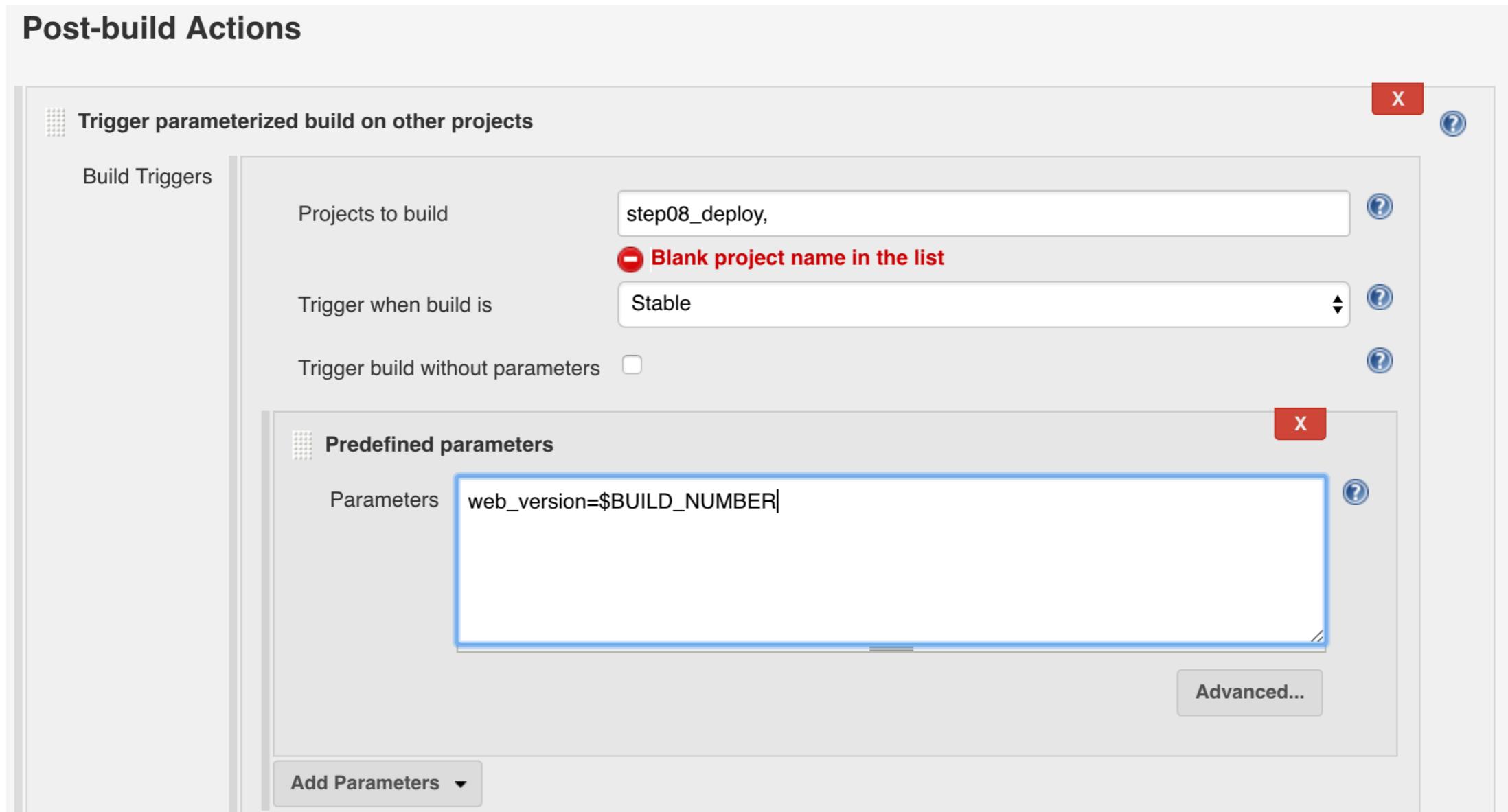
Trigger build without parameters:

**Predefined parameters**

Parameters: web\_version=\$BUILD\_NUMBER

**Add Parameters ▾**

**Advanced...**



# **In job = step08\_deploy**



# Receive parameters

# Choose This project is parameterized

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

Discard old builds (?)

GitHub project

This project is parameterized (?)

**String Parameter** X (?)

Name	web_version	<span>(?)</span>
Default Value		<span>(?)</span>
Description		<span>(?)</span>

[Plain text] [Preview](#)

[Add Parameter ▾](#)



# Try to use this parameter

\$web\_version in my build steps

## Build

Execute shell

X ?

Command `wget -O demo.war -q http://localhost:8081/repository/demo_web/$web_version/demo.war`

**wget --output demo.war -q**

**<url of artifact server>/\$web\_version/demo.war**

See [the list of available environment variables](#)

Advanced...

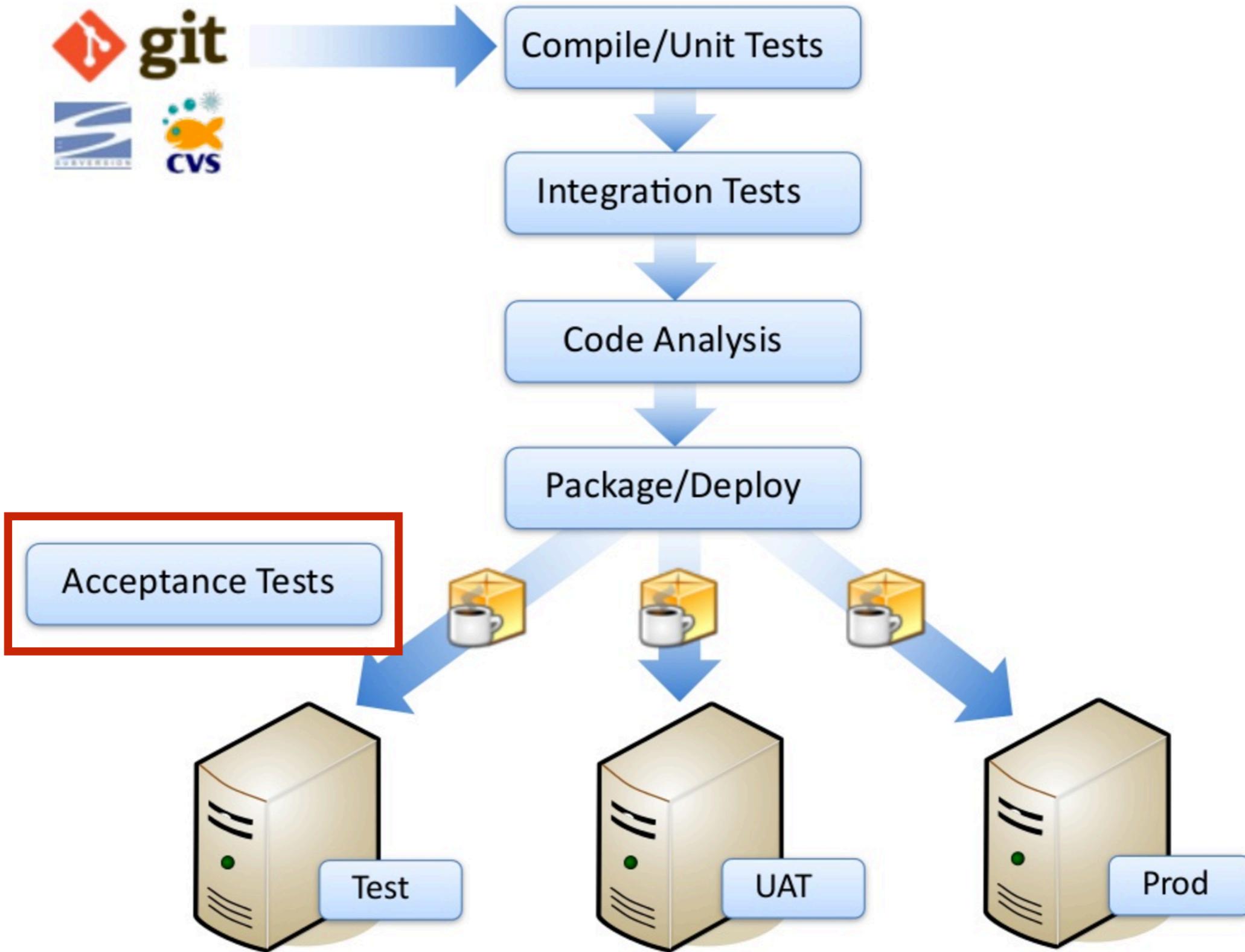
Add build step ▾





7

Acceptance Tests



# 7. Run acceptance tests

Try to test Web UI with Robotframework



# Install Robotframework

\$pip install robotframework

\$pip install robotframework-selenium2library



# Run Robotframework

\$pybot

```
[ ERROR ] Expected at least 1 argument, got 0.  
Try --help for usage information.
```



# Robotframework plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<input type="checkbox"/> <a href="#">Robot Framework</a>	Shows Robot Framework test results in project	1.6.4

[Install without restart](#) [Download now and install after restart](#) Update information obtained: ·



# Add build step to run with Robot

## Build

### Execute shell

Command `pybot *.robot`

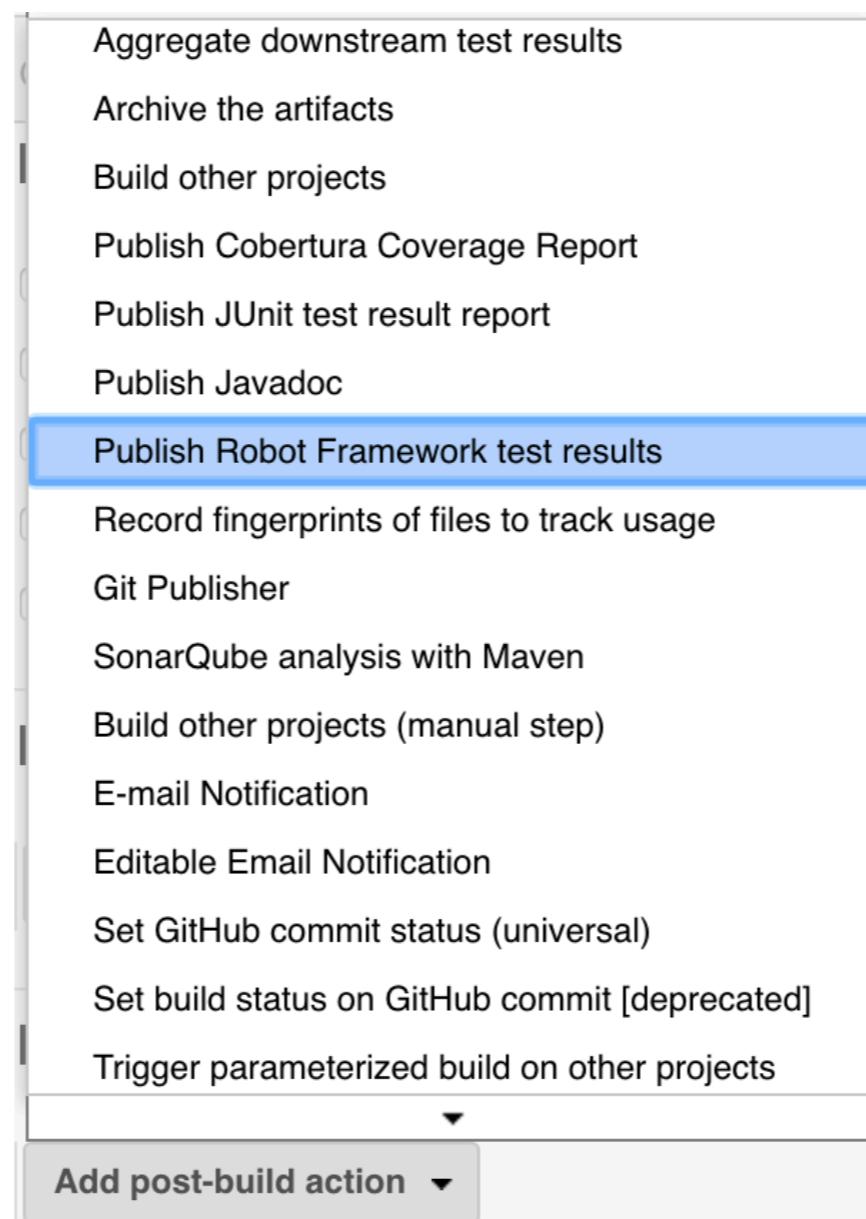
**pybot \*.robot**

See [the list of available environment variables](#)



# Add Robotframework report

Add post build action => Publish Robot Framework



# Add Robotframework report

## Configuration your report

**Post-build Actions**

**Publish Robot Framework test results**

Directory of Robot output

Path to directory containing robot xml and html files (relative to build workspace)  Advanced...

Thresholds for build result

Yellow icon: %  **Entry must be percentage value between 0-100**

Green icon: %  **Entry must be percentage value between 0-100**

Use thresholds for critical tests only

Add post-build action ▾



# Report

 [add description](#)

[Disable Project](#)



[Workspace](#)



[Recent Changes](#)

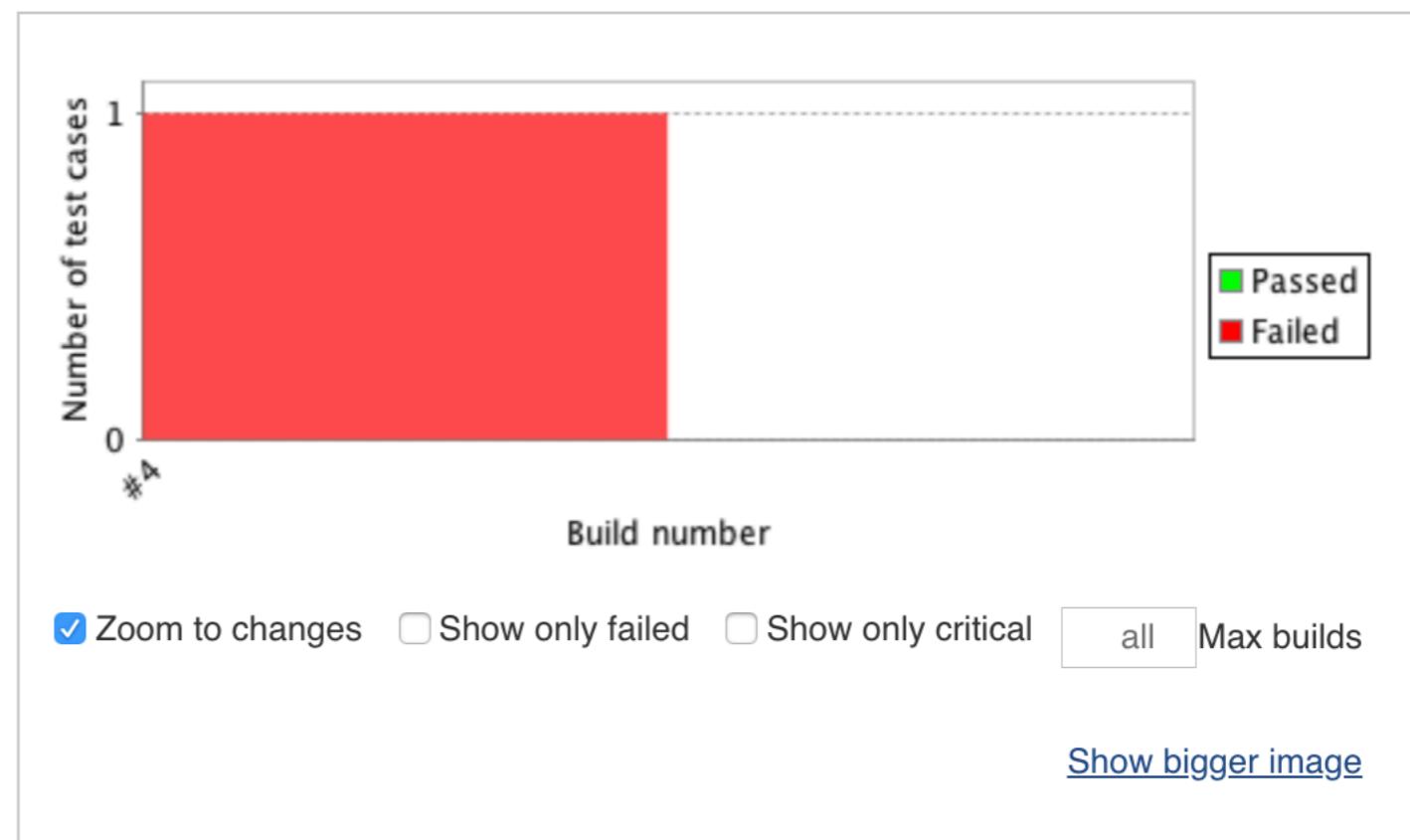


**Latest Robot Results:**

	Total	Failed	Passed	Pass %
Critical tests	1	1	0	0.0
All tests	1	1	0	0.0

- [Browse results](#)
- [Open report.html](#)
- [Open log.html](#)

**Robot Framework Tests Trend (all tests)**



# Try it by yourself

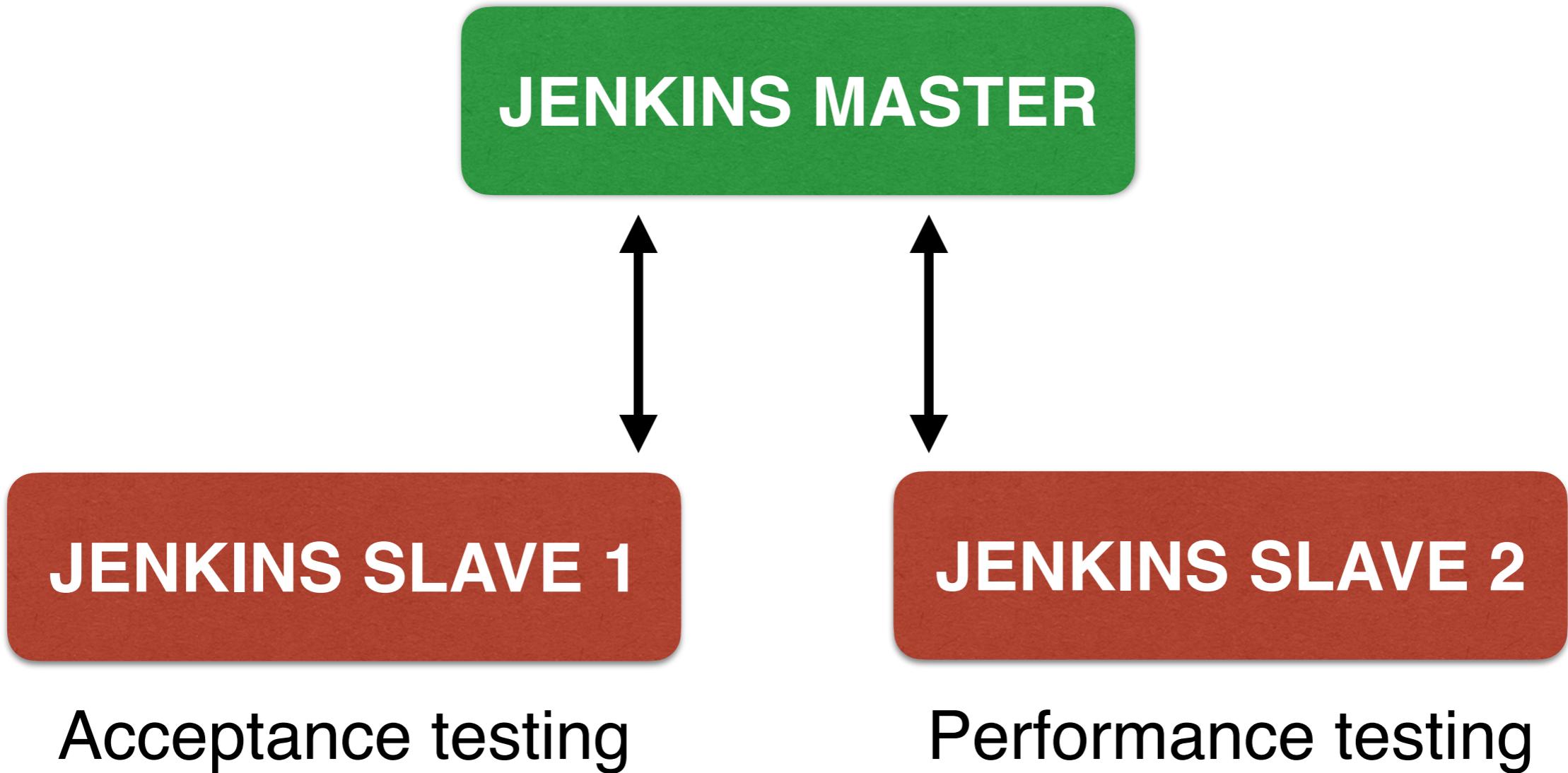


# Jenkins

# Master-Slave

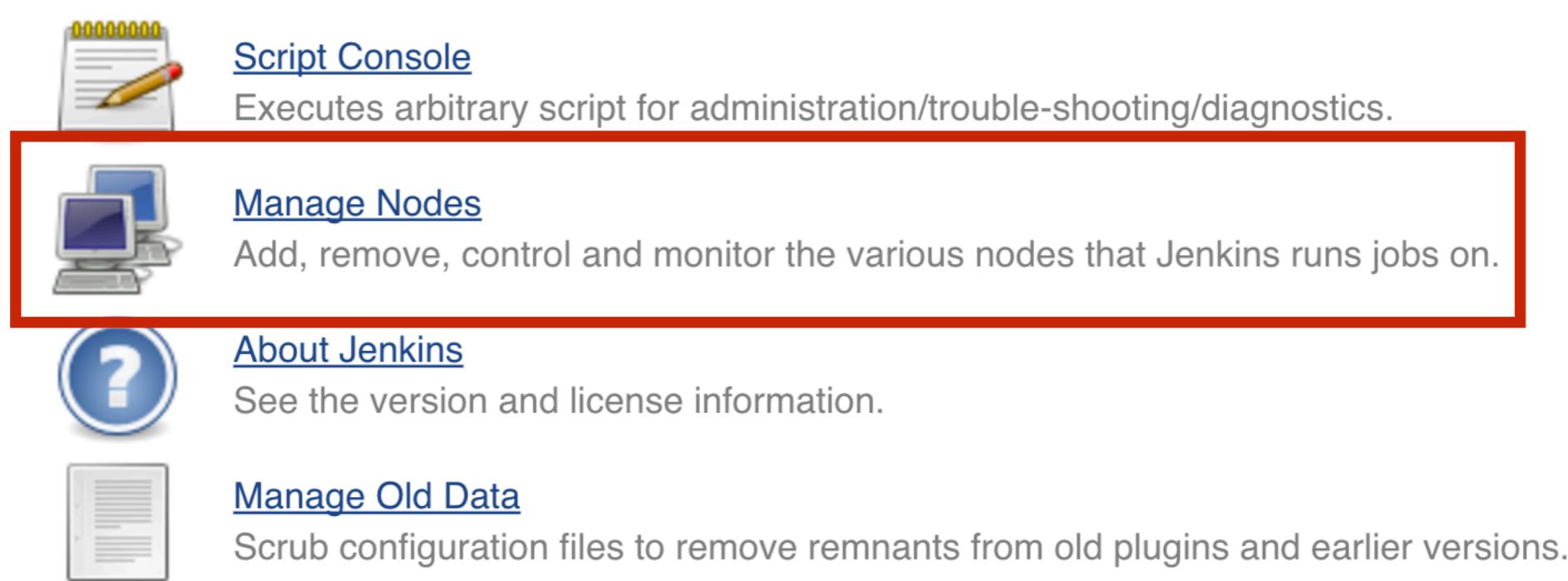


# Need more Jenkins's node



# Add new node

## 1. Manage Jenkins -> Manage Nodes



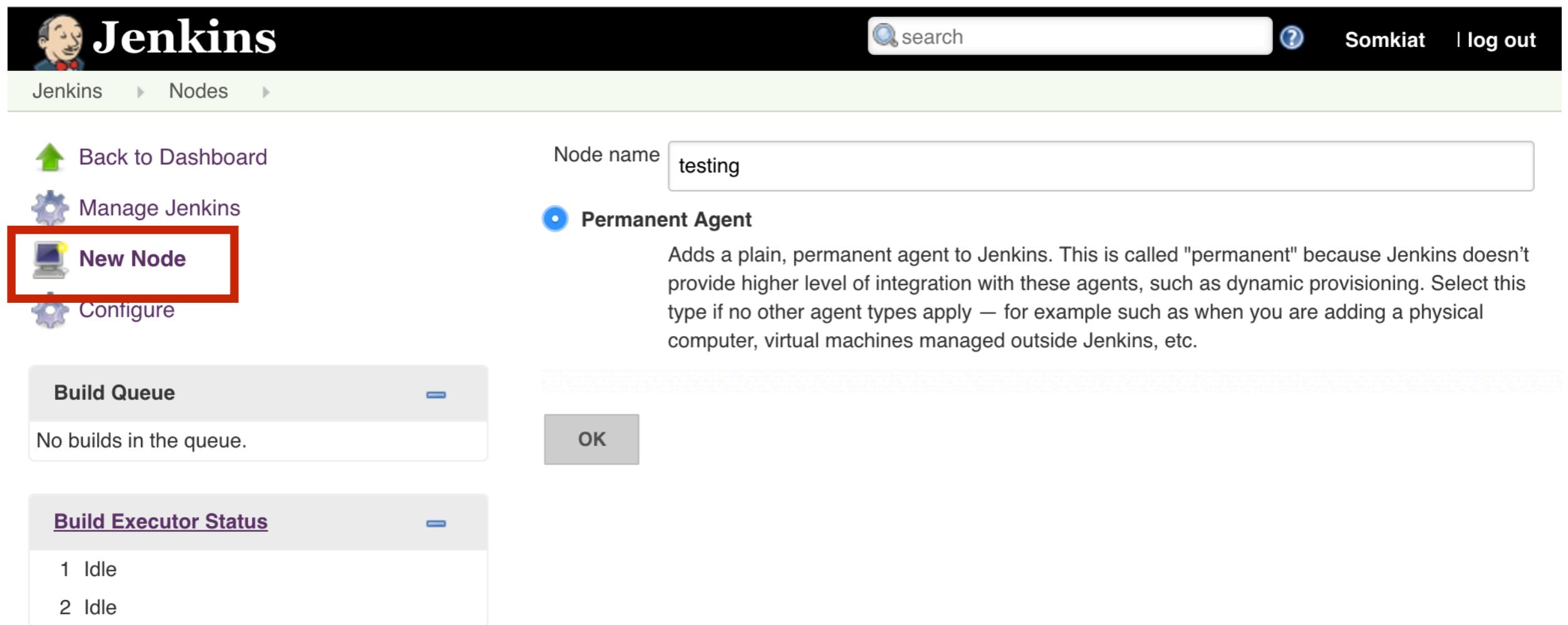
The screenshot shows the Jenkins 'Manage Jenkins' sidebar. The 'Manage Nodes' option is highlighted with a red box. The other options are:

- Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on. (This option is highlighted with a red box.)
- About Jenkins**: See the version and license information.
- Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.



# Add new node

2. Choose new node and fill in name



The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with a logo, the word "Jenkins", a search bar, and user information ("Somkiat | log out"). Below the header, the URL path "Jenkins > Nodes" is visible. On the left side, there's a sidebar with links: "Back to Dashboard", "Manage Jenkins", "New Node" (which is highlighted with a red box), and "Configure". The main content area is titled "Add New Node" and contains a form. The "Node name" field has "testing" typed into it. Below it, a radio button labeled "Permanent Agent" is selected, with a descriptive text explaining its function: "Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.". At the bottom right of the form is an "OK" button.



# Add new node

## 3. Edit node and save

Jenkins

Nodes > testing

search Somkiat | log out

Back to List Status Delete Agent Configure Build History Load Statistics Script Console Log System Information Disconnect

Name: testing

Description:

# of executors: 1

Remote root directory: /Users/somkiat/data/slide/ci-cd/swpark/node\_testing

Labels: testing

Usage: Only build jobs with label expressions matching this node

Launch method: Launch agent via execution of command on the master

Launch command: java -jar /Users/somkiat/data/slide/ci-cd/swpark/slave.jar

Availability: Keep this agent online as much as possible

Build Executor Status: 1 Idle

Node Properties:

Environment variables  
 Tool Locations

Save



# Add new node

## 4. List of all nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	<a href="#">master</a>	Mac OS X (x86_64)	In sync	21.28 GB	1.02 GB	21.28 GB	0ms	
	<a href="#">testing</a>	Mac OS X (x86_64)	In sync	21.28 GB	1.02 GB	21.28 GB	4025ms	
Data obtained	1 min 27 sec		1 min 27 sec	1 min 27 sec	1 min 26 sec	1 min 27 sec	1 min 27 sec	

[Refresh status](#)



# Run job with new node

Choose your job and click configure

The screenshot shows the Jenkins interface for the 'hello' project. At the top, there's a navigation bar with the Jenkins logo and the path 'Jenkins > hello'. Below this is a sidebar with links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. The 'Configure' link is highlighted with a red box. To the right, the main content area is titled 'Project hello'. It shows two sections: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). At the bottom, there are links for 'Build History' and 'trend'.



# Run job with new node

## Define label expression with selected node

The screenshot shows the Jenkins General configuration page for a project named "hello". The "General" tab is selected. In the "Restrict where this project can be run" section, the "Restrict where this project can be run" checkbox is checked. The "Label Expression" field contains "te" and has a dropdown menu open with "testing" listed. The entire "Restrict where this project can be run" section is highlighted with a red border.

**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 [?](#)

Restrict where this project can be run [?](#)

Label Expression te

testing

[ADVANCED...](#)

[Save](#) [Apply](#)



# 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**

*» 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.

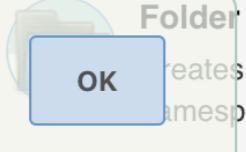
 **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **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

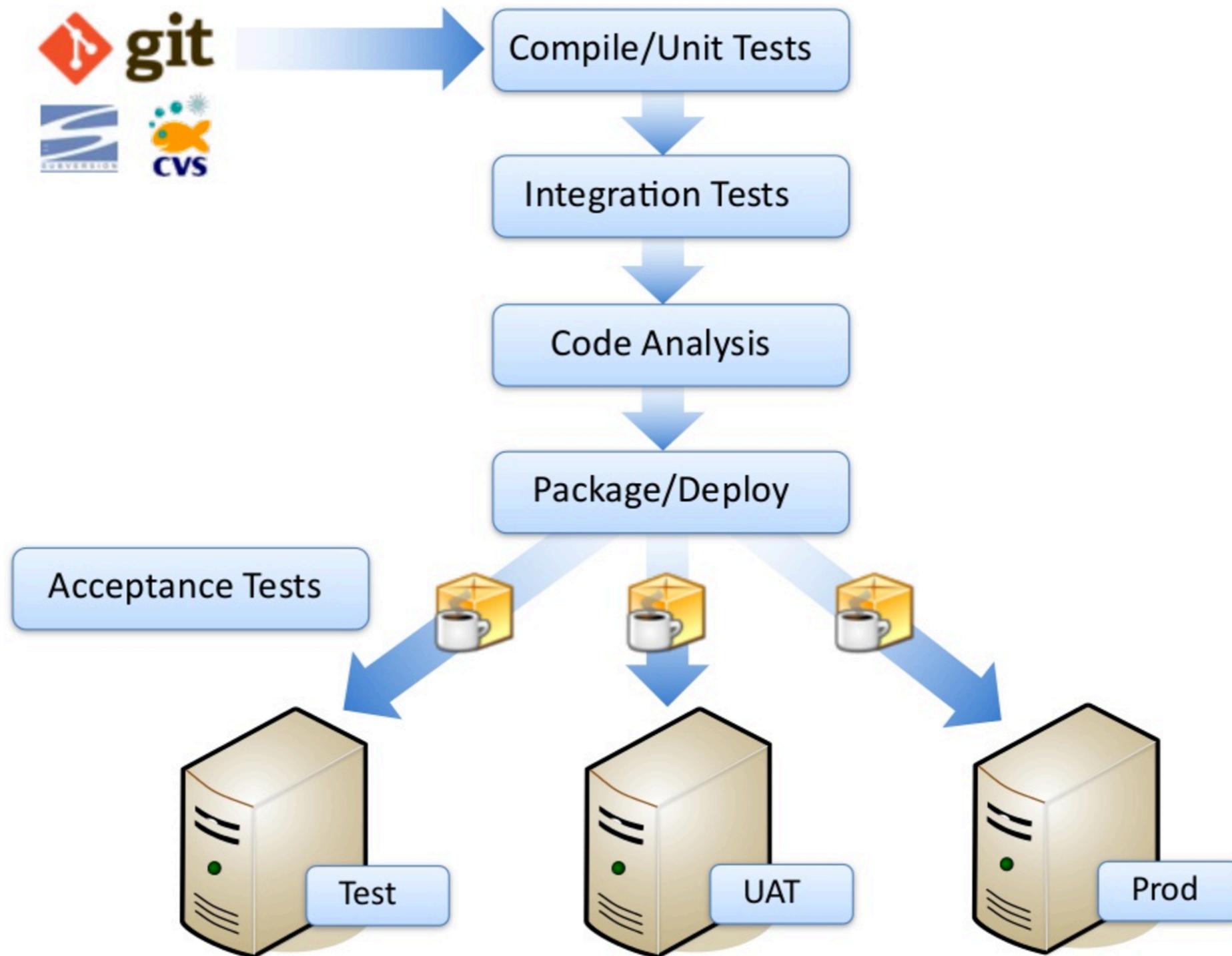
Script 1 try sample Pipeline... ?

Use Groovy Sandbox ?

[Pipeline Syntax](#)



# Build pipeline

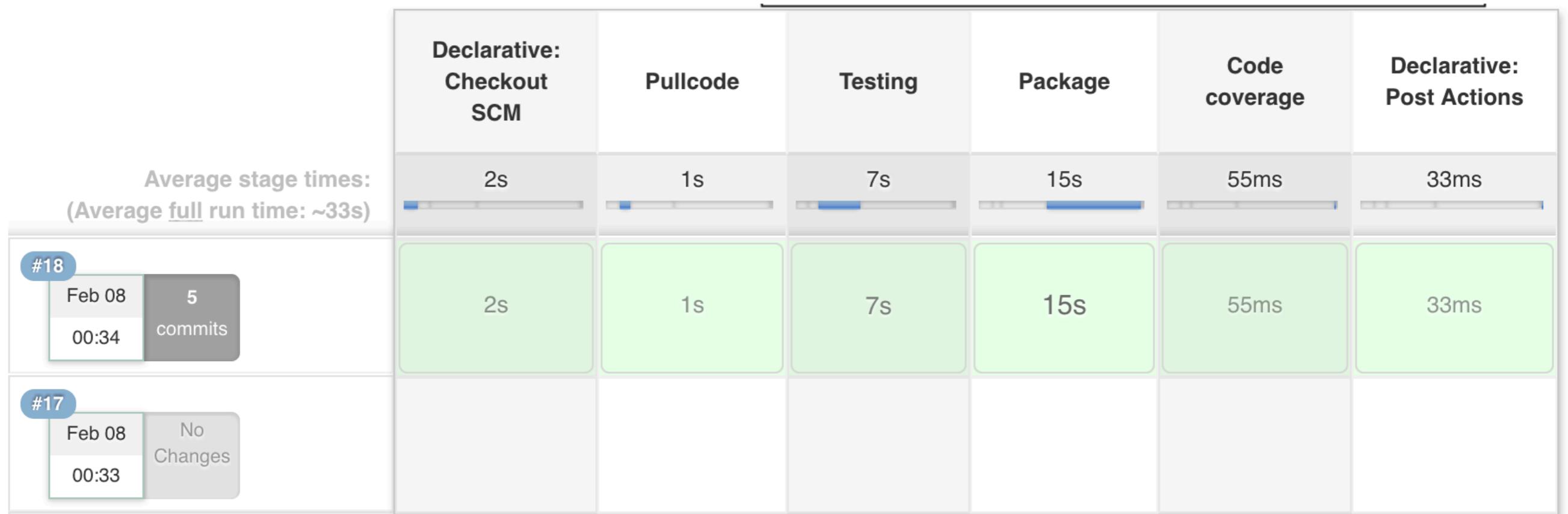


# Create pipeline as code

```
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: false
    }
}
post {
    always {
        junit 'target/surefire-reports/*.xml'
    }
}
```



# Try it by yourself

