



CI/CD series

# CI/CD with Jenkins







Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชั่นนาคุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

Help people take action on this Page. ×

+ Add a Button

Home Posts Videos Photos

Liked Following Share ...



# 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 ?

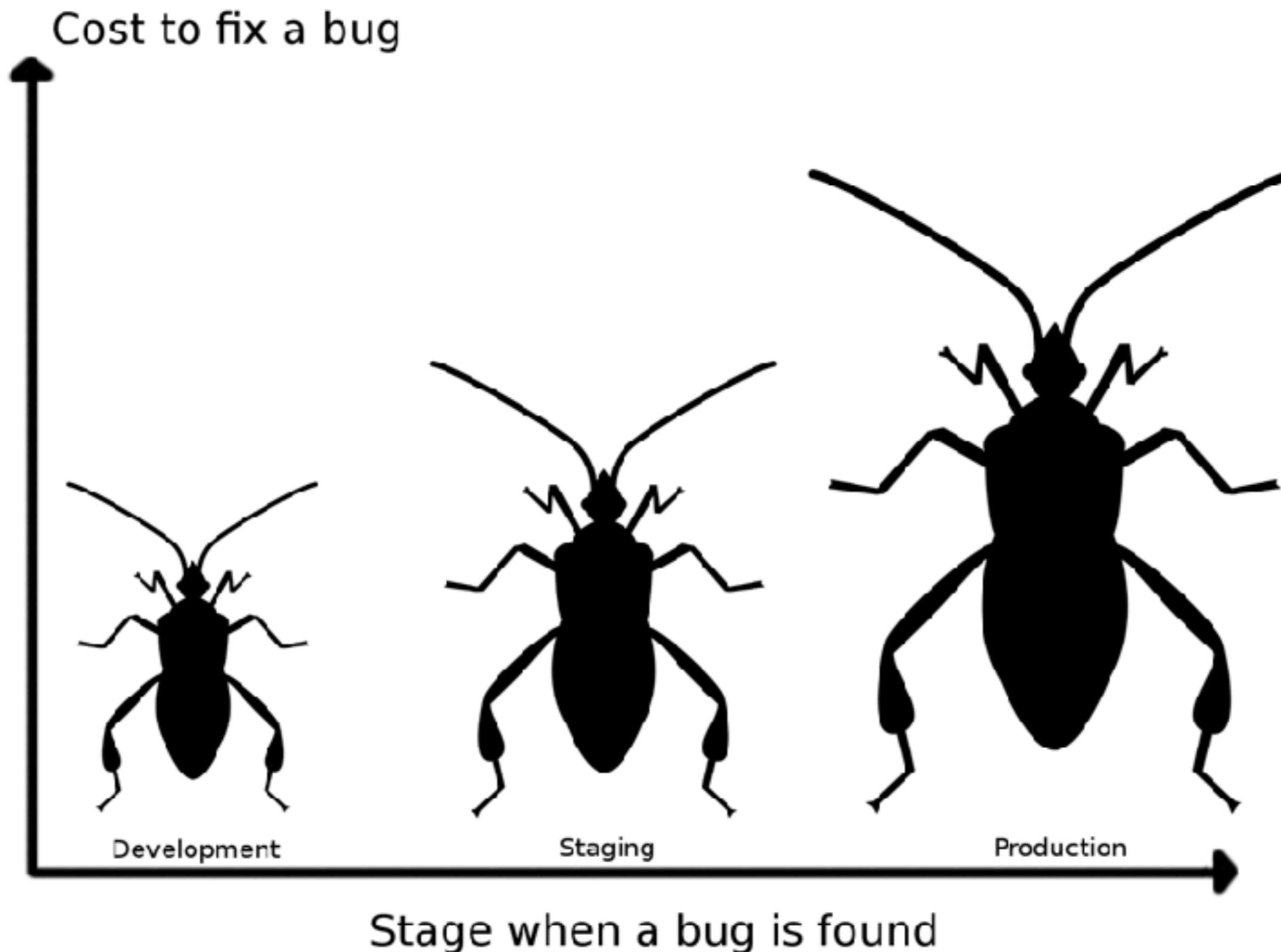


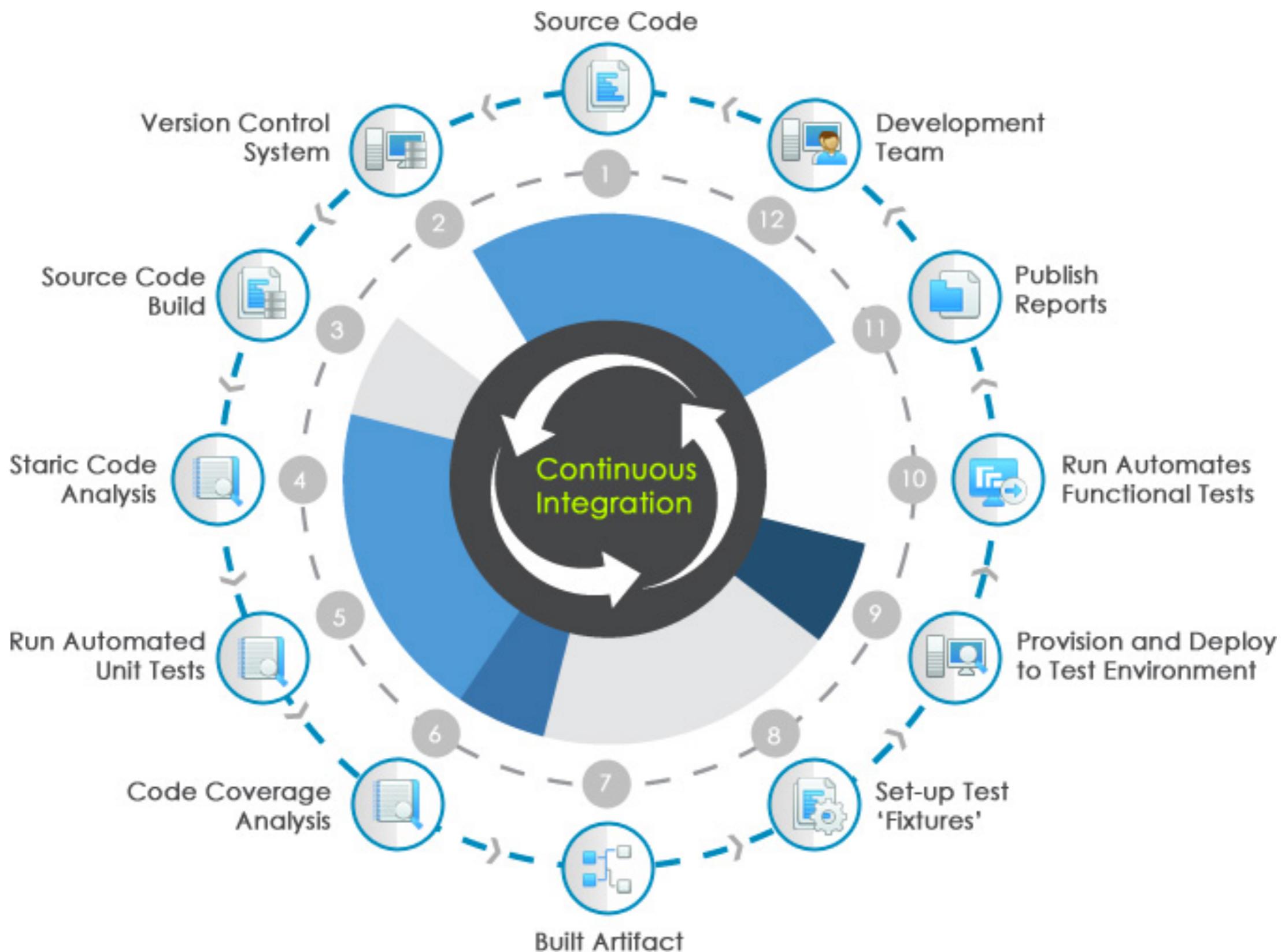
# The cost of integration

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

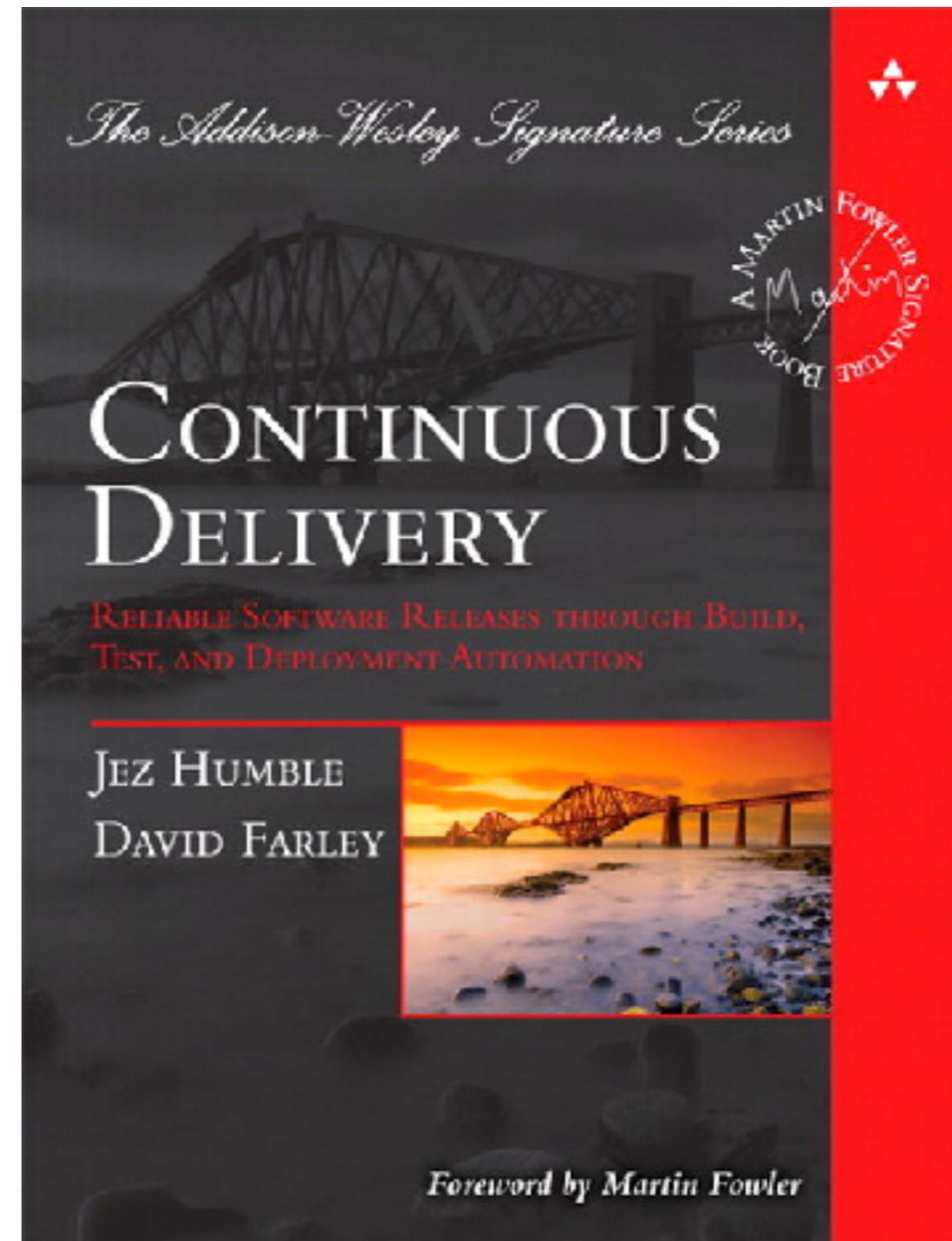
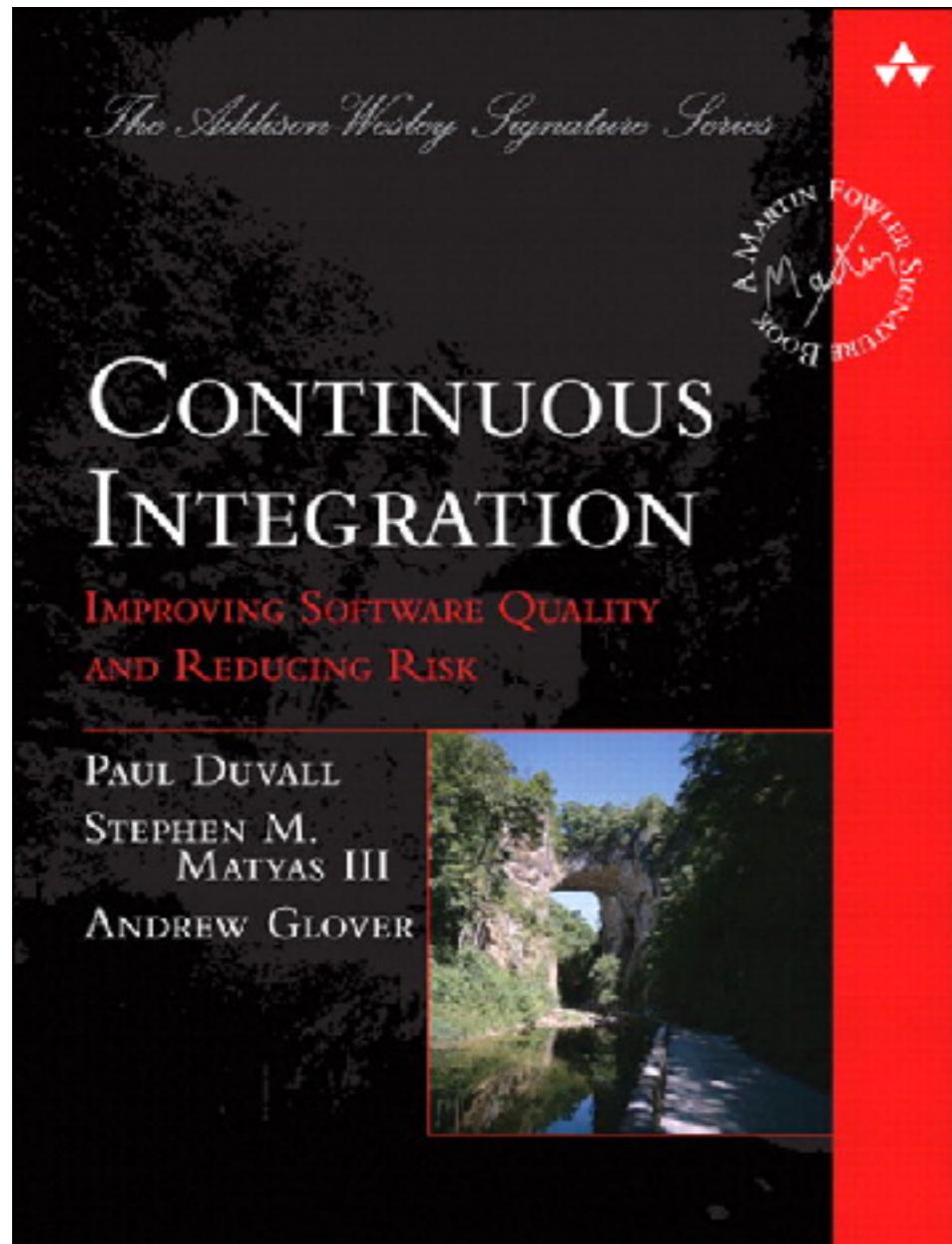
Strive for **fast feedback**



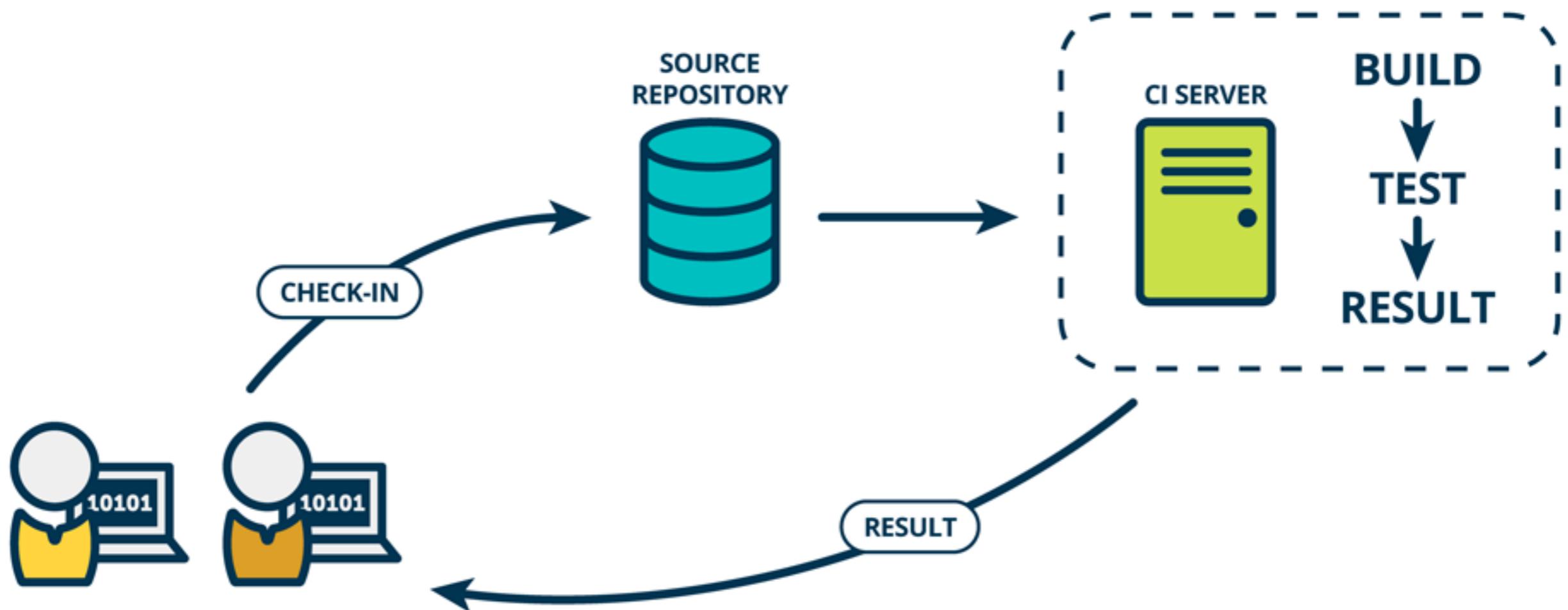
# **Practices of Continuous Integration**



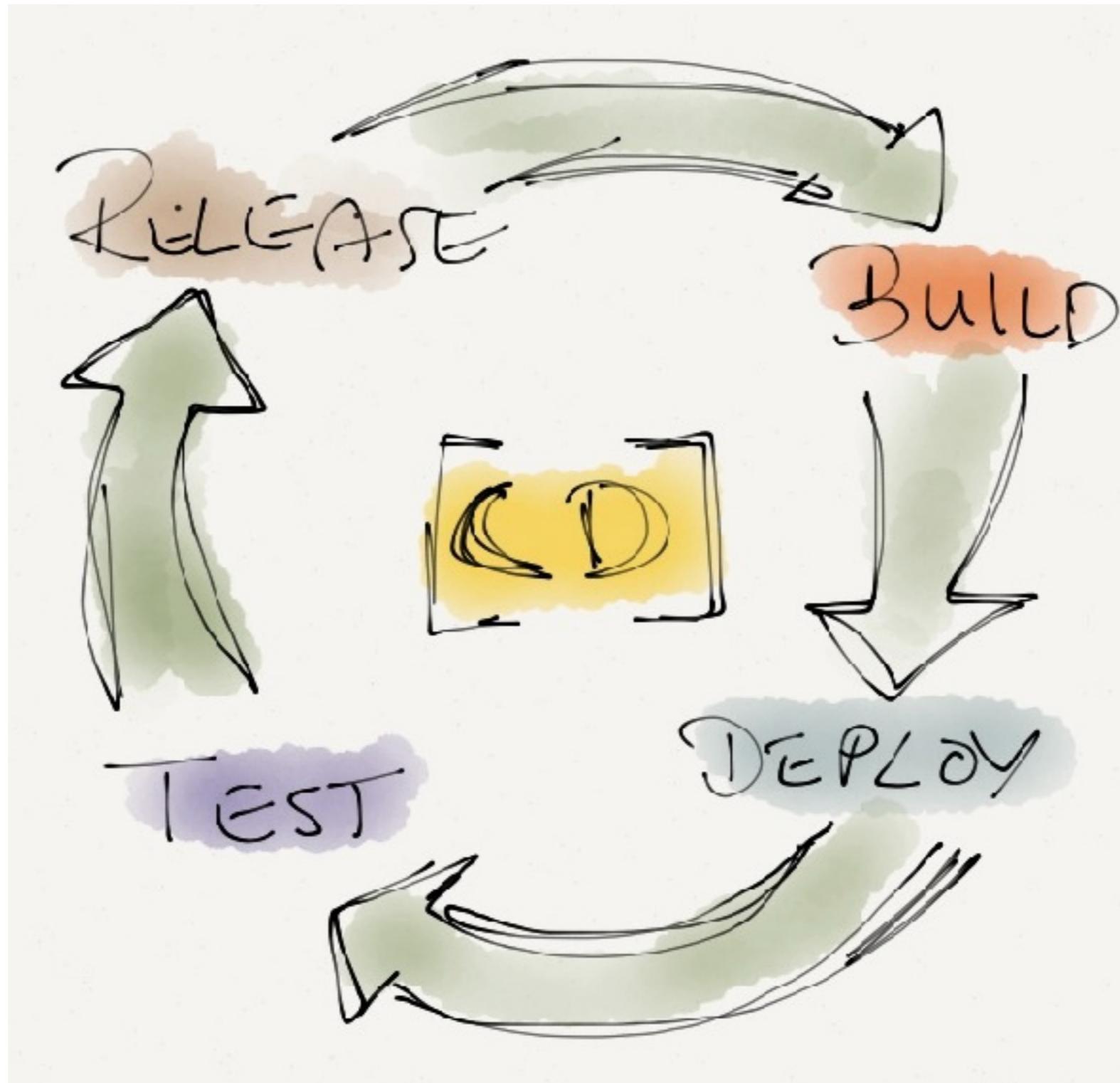
# Improve quality and reduce risk



# Continuous Integration



# CD ?



# CD ?

## CONTINUOUS DELIVERY



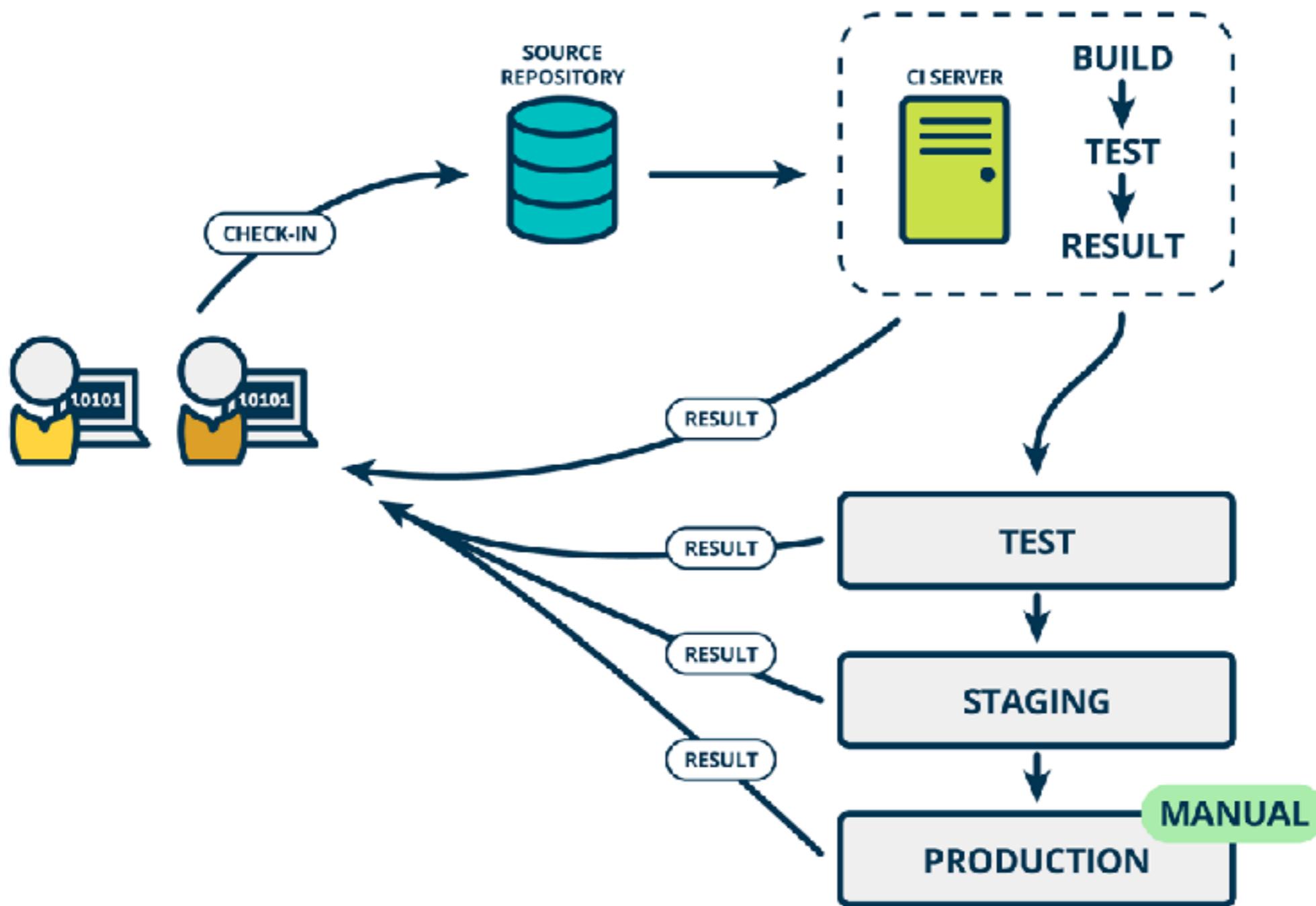
## CONTINUOUS DEPLOYMENT



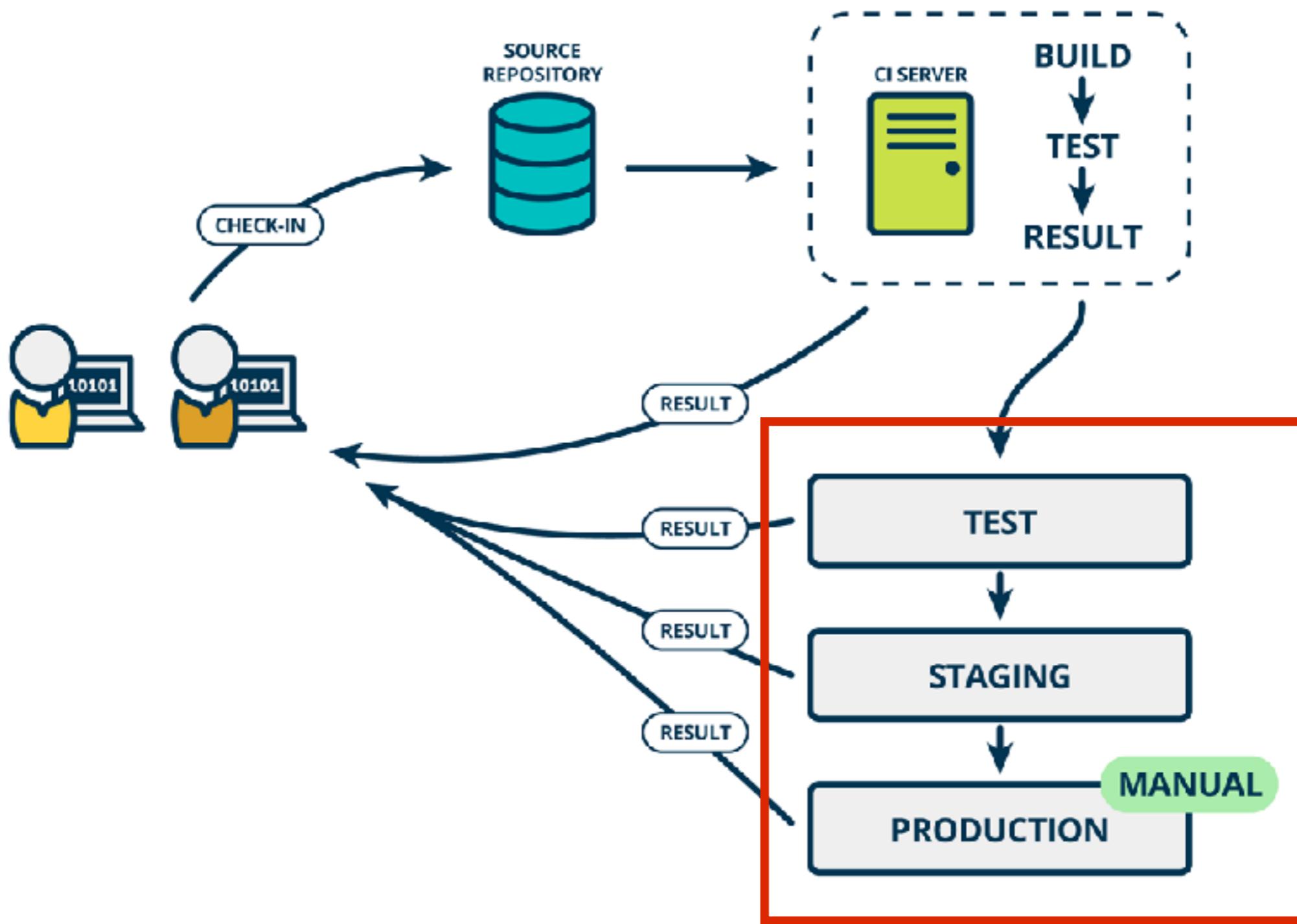
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



# Continuous Delivery



# Rise of DevOps



# **Continuous Integration**

**is a Software development practices**



# Practice 1

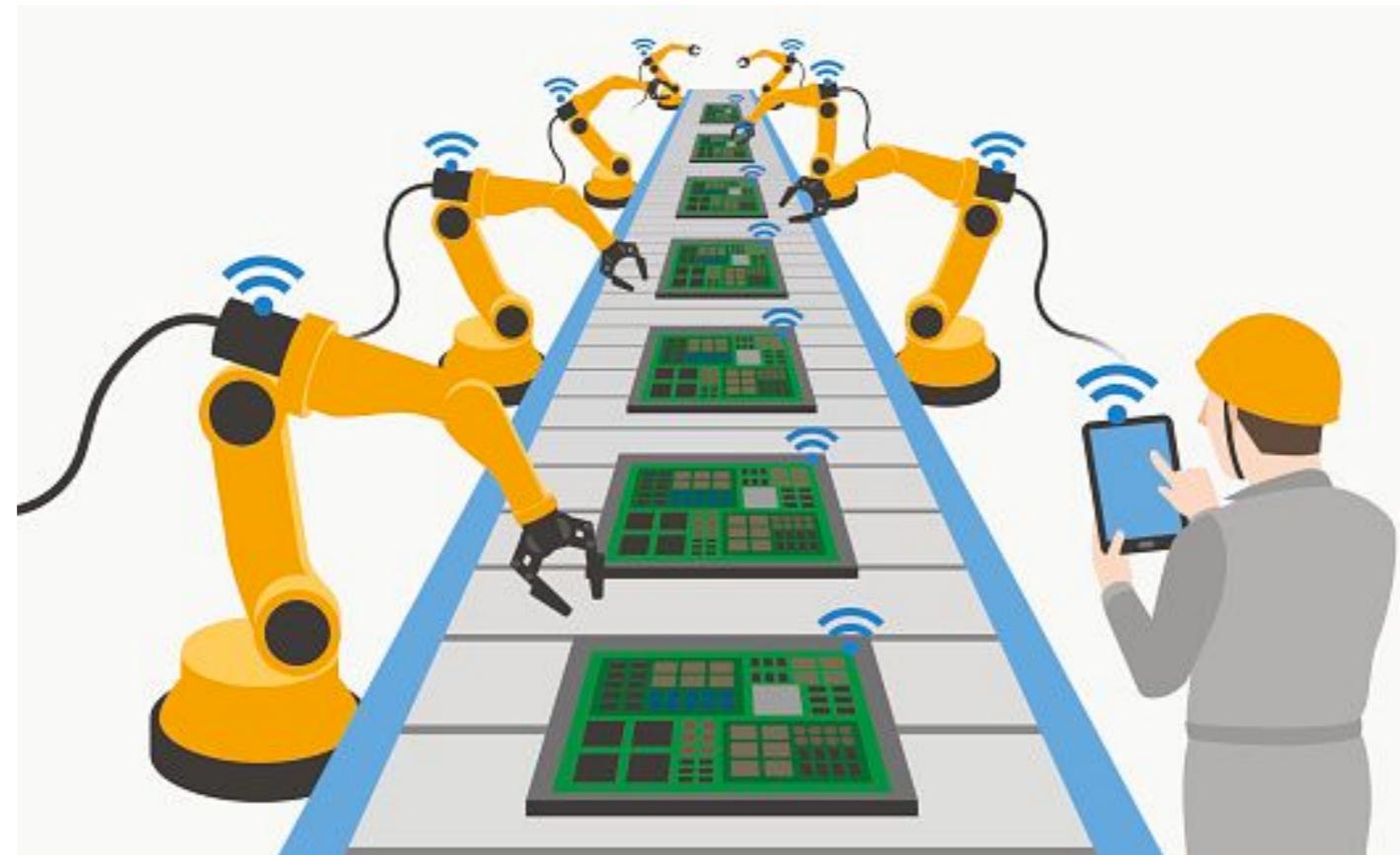
Maintain a single source repository

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



# Practice 2

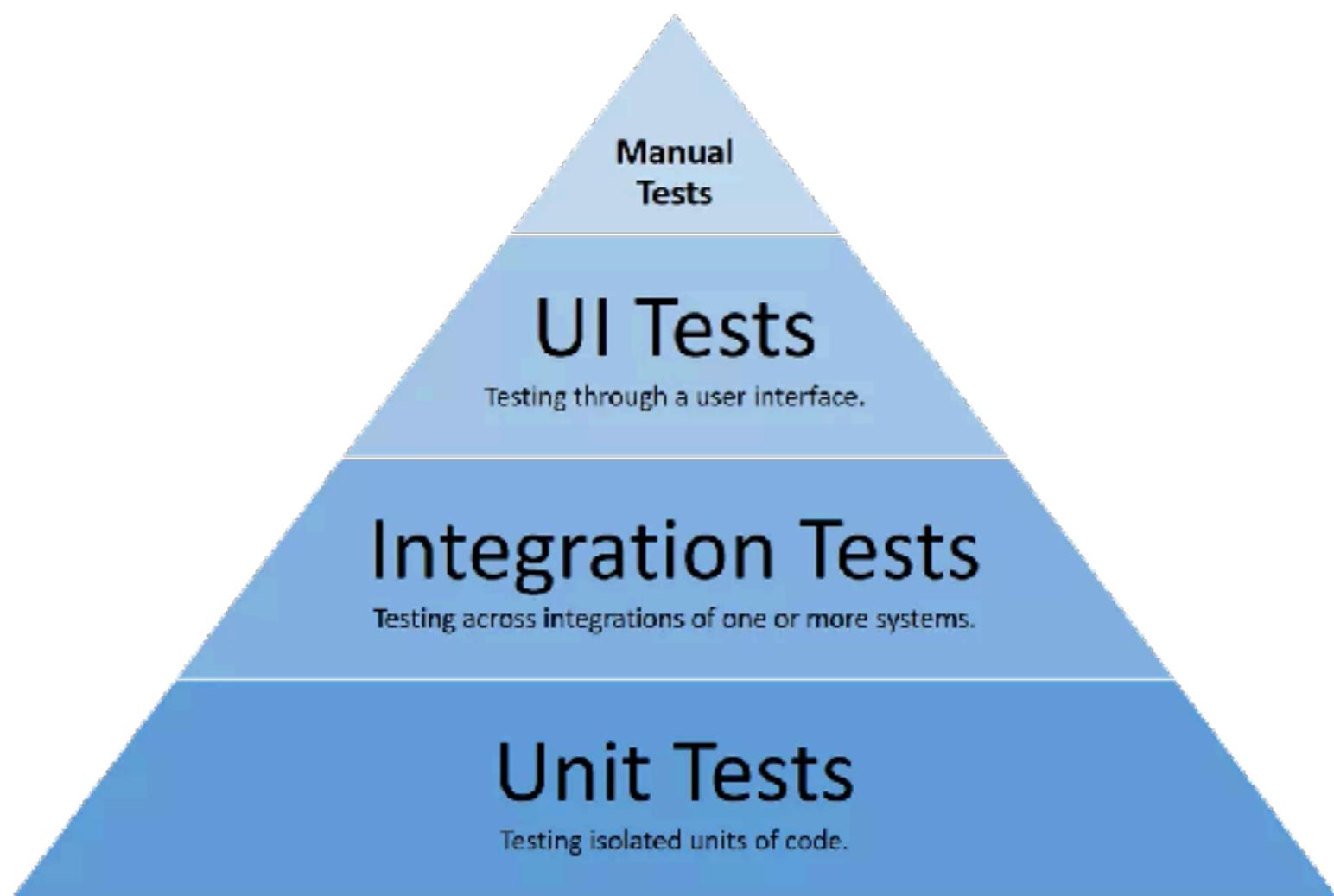
Automated the build  
Automated environment for builds



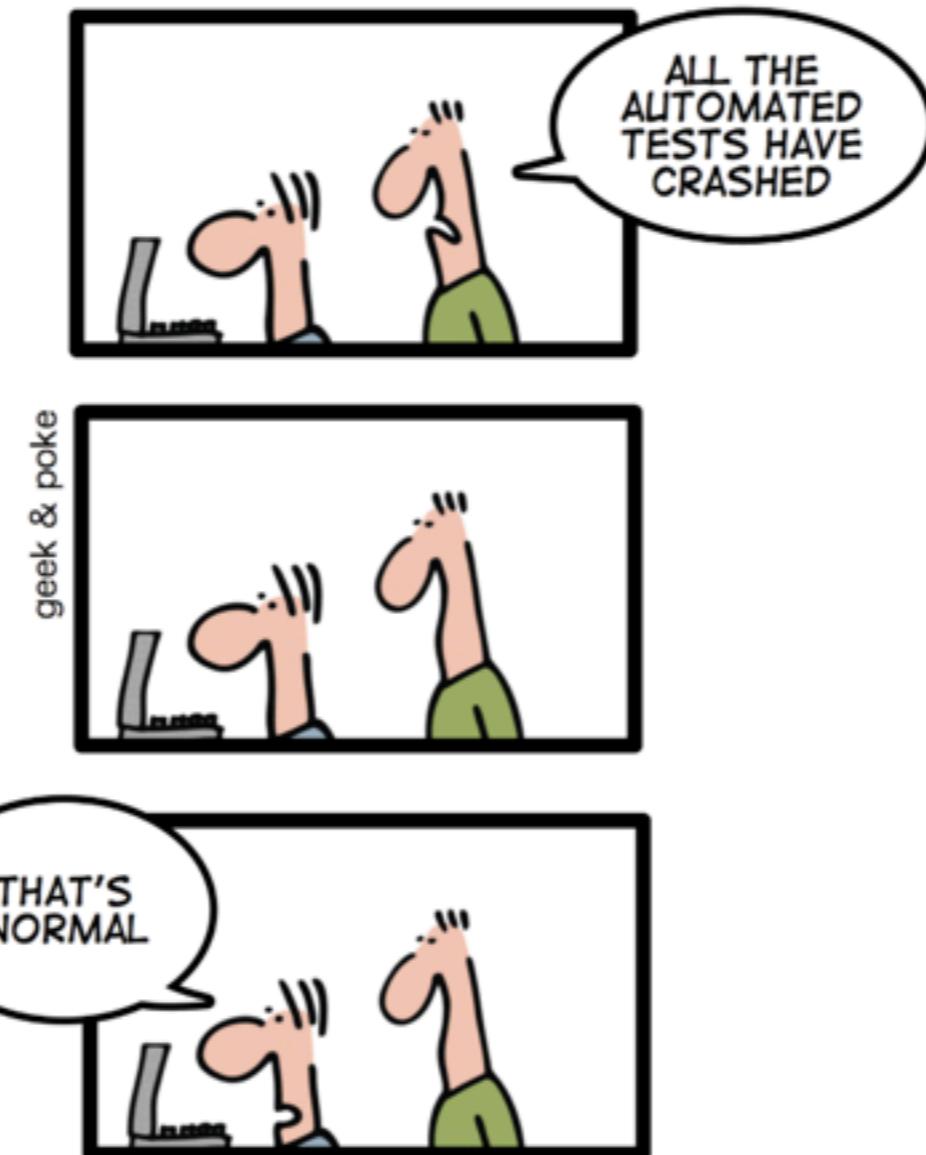
# Practice 3

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/>

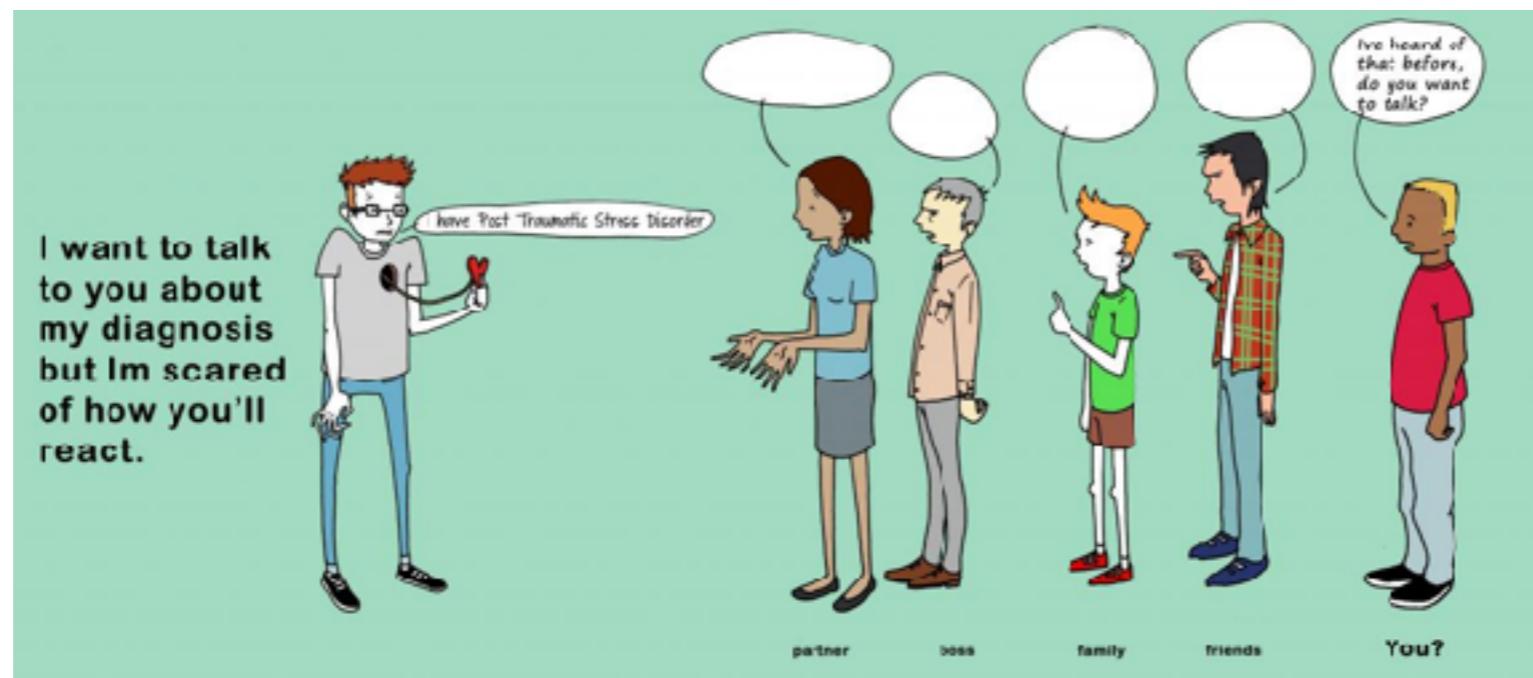


# Practice 4

Everyone **commits** to the mainline everyday

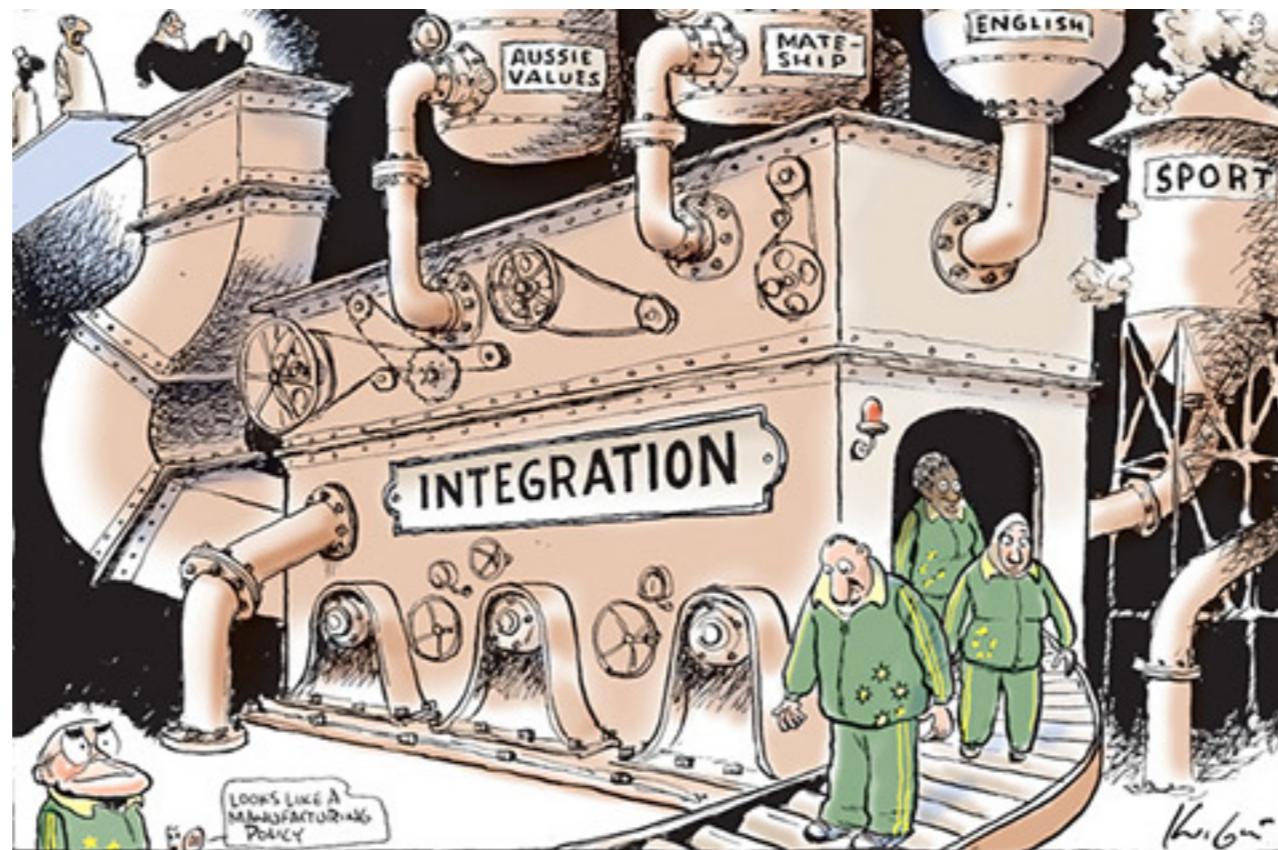
Integration is about **communication**

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

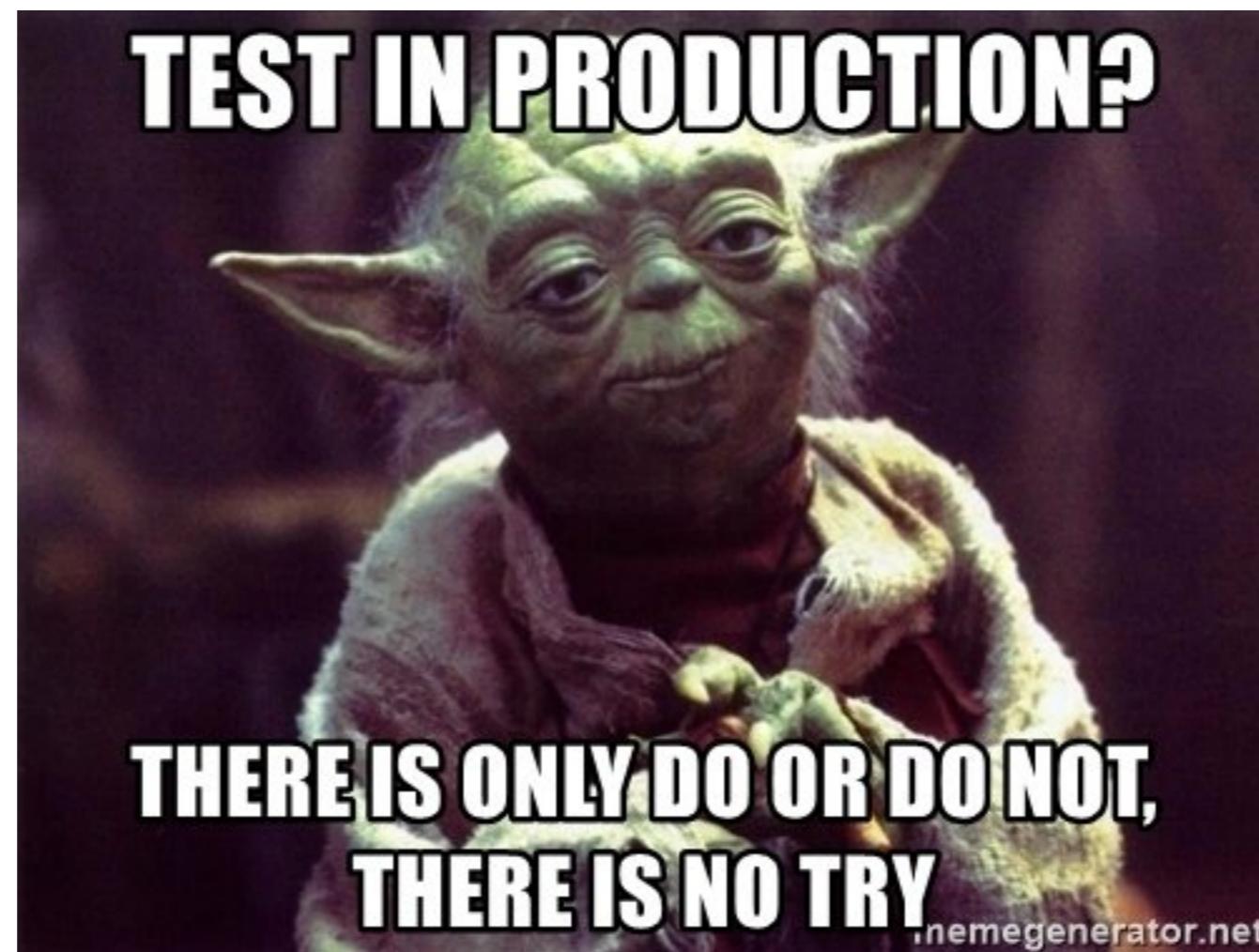
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

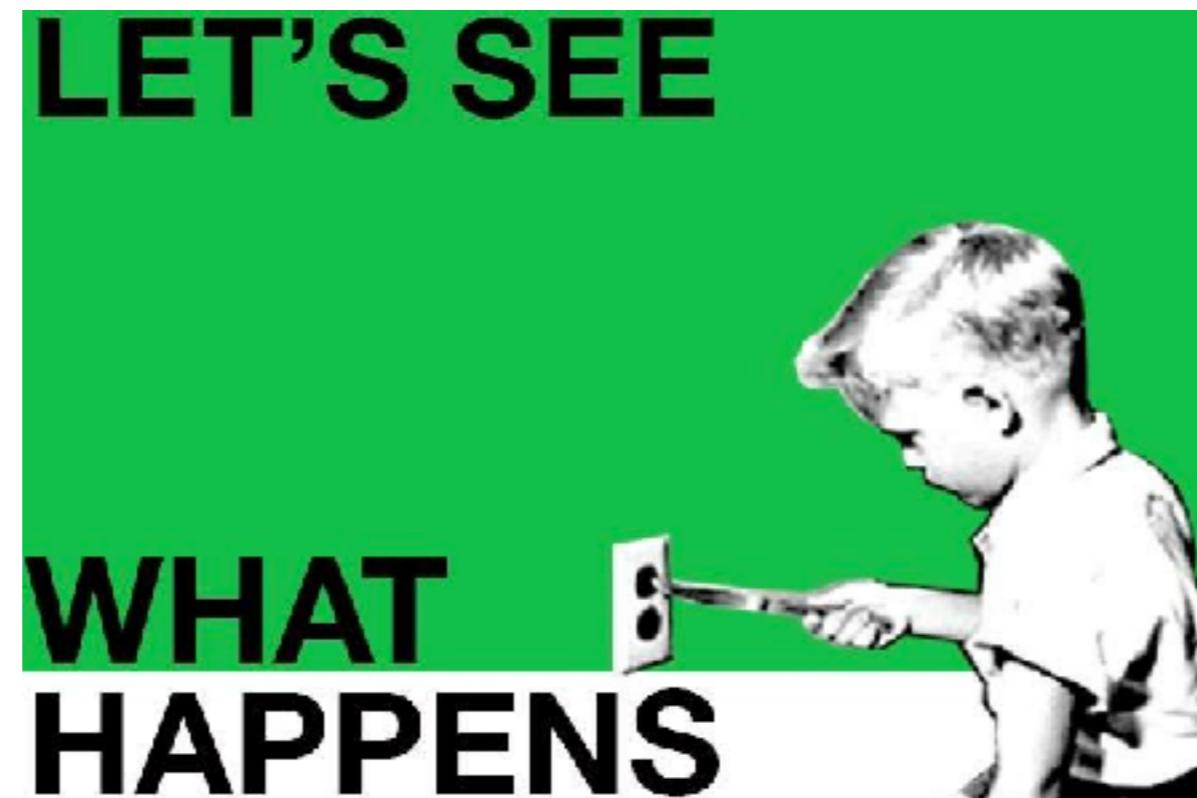
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



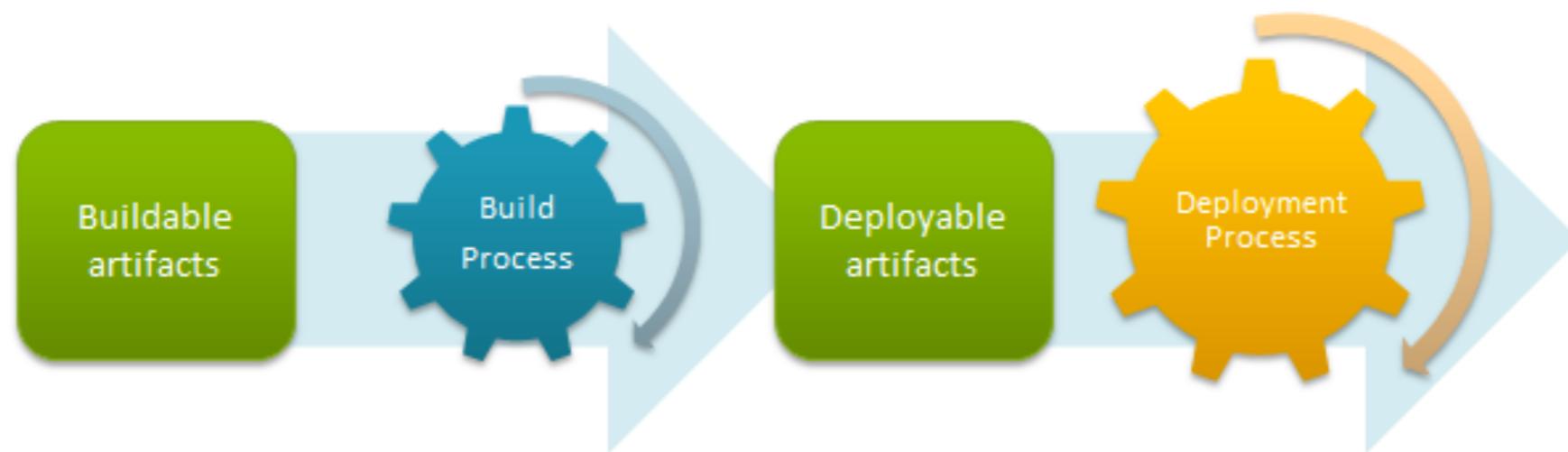
# Practice 10

**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, smelly pile of excrement with three wavy lines above it indicating odor.

JUST SAY NO!



# 2. Choose Branch strategy

Main only

Development isolation

Feature isolation

Release isolation

Service and Release isolation

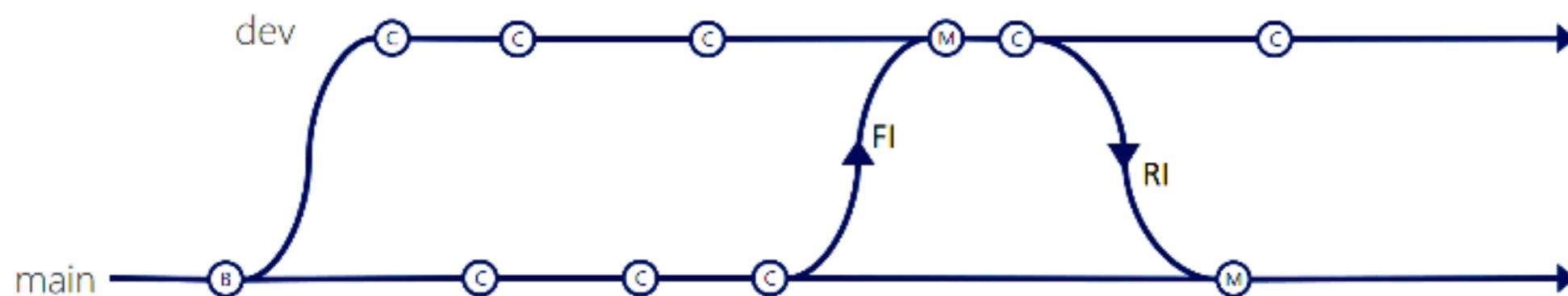
Service, Hotfix and Release isolation



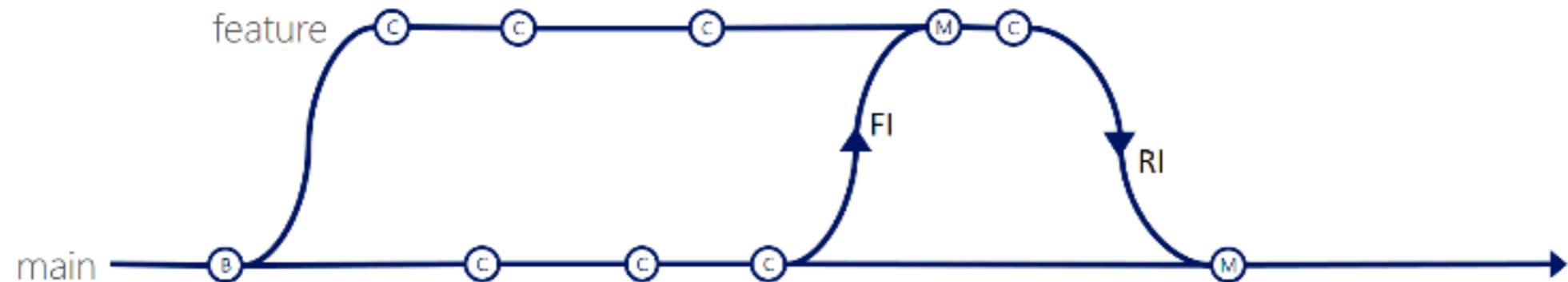
# Main only



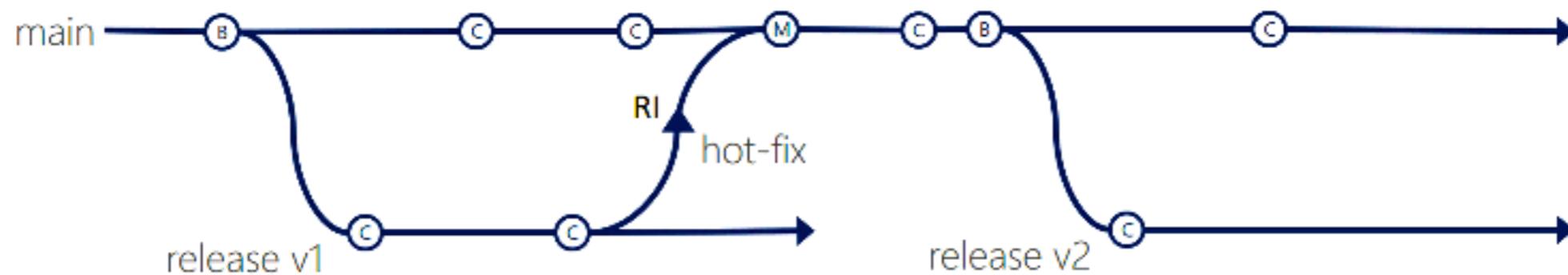
# Development isolation



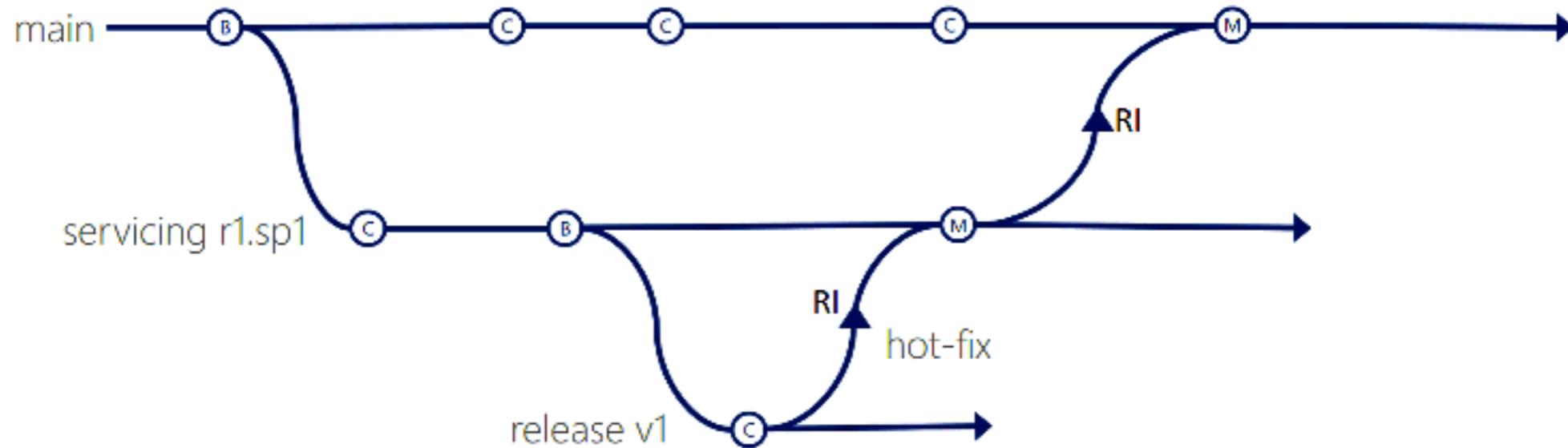
# Feature isolation



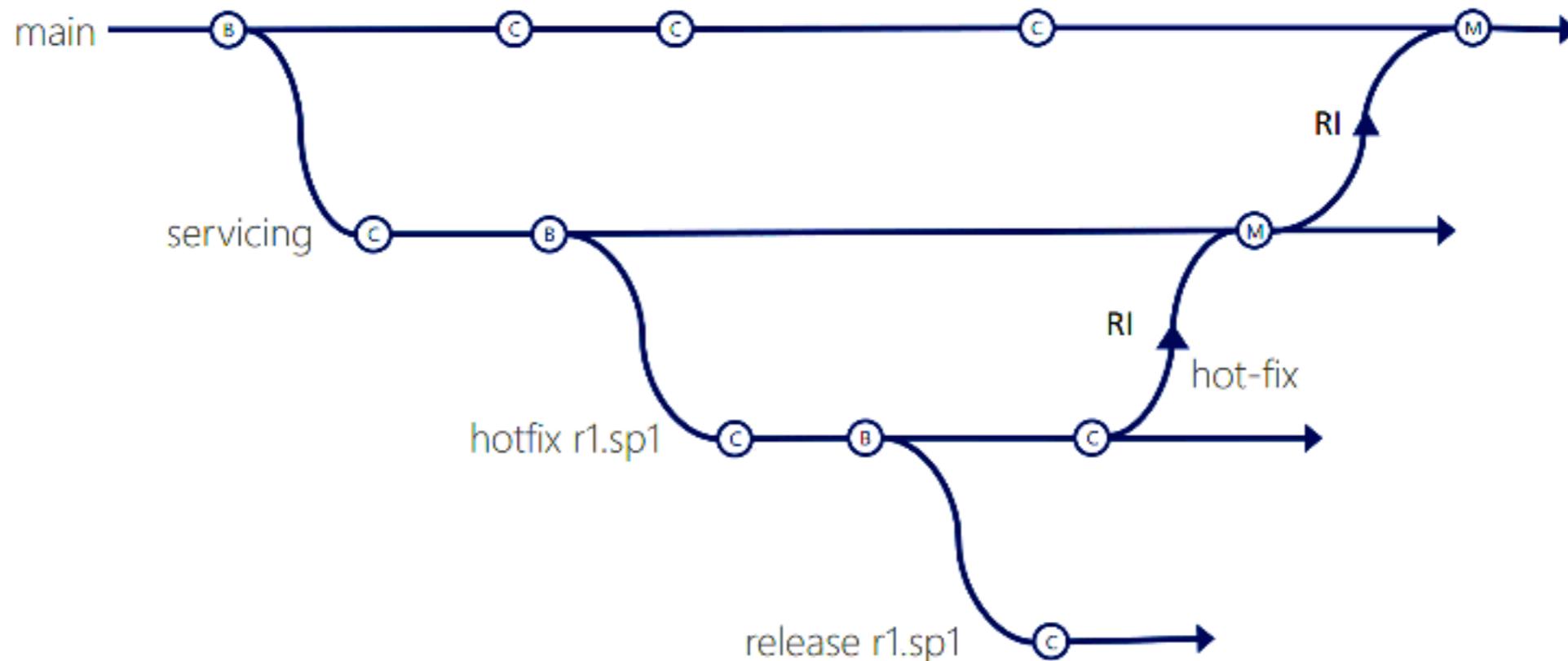
# Release isolation



# Service and Release isolation



# Service, Hotfix, Release isolation



# Validate, Validate and Validate

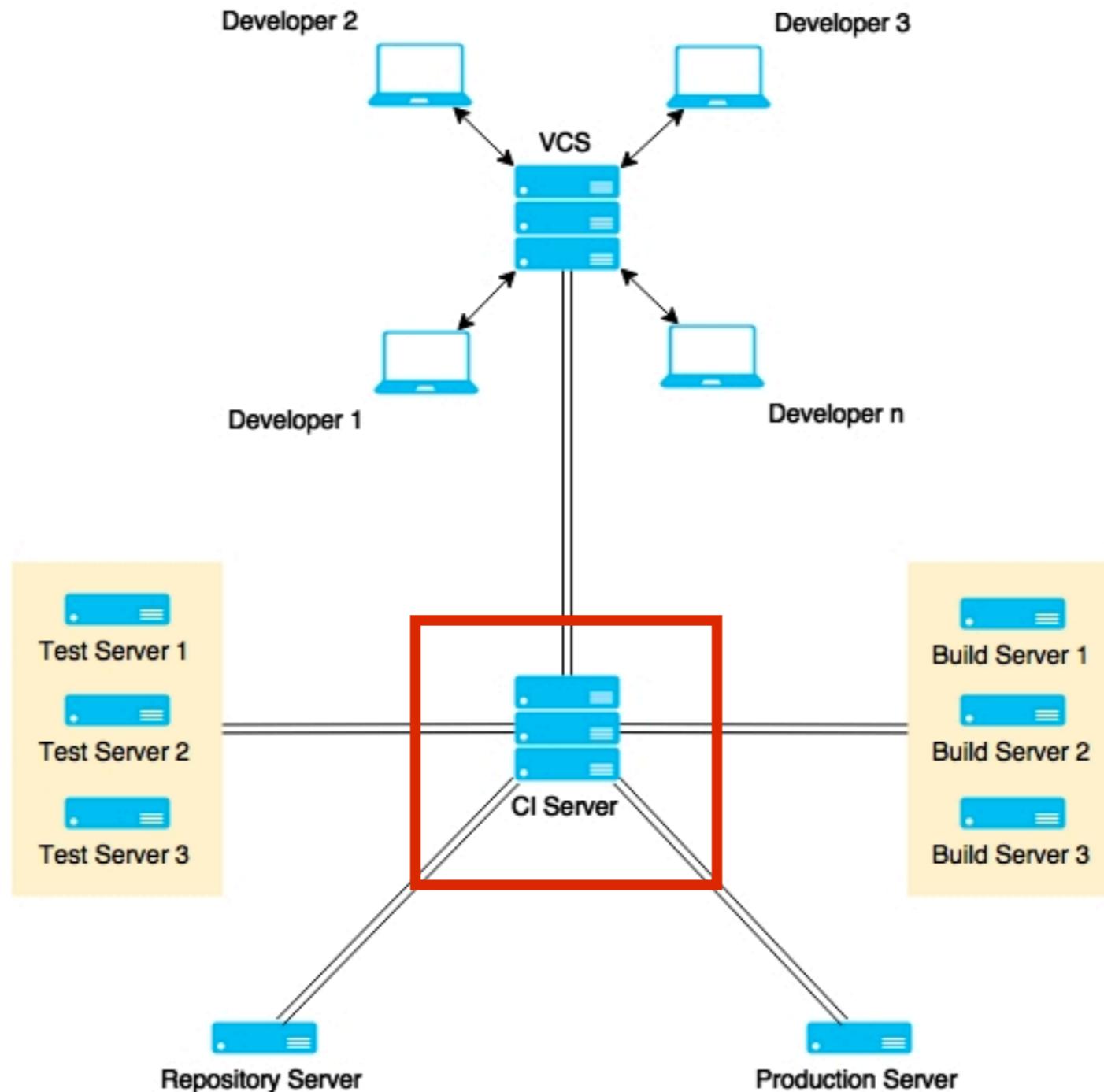


# Suggestion

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



# More ...

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



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



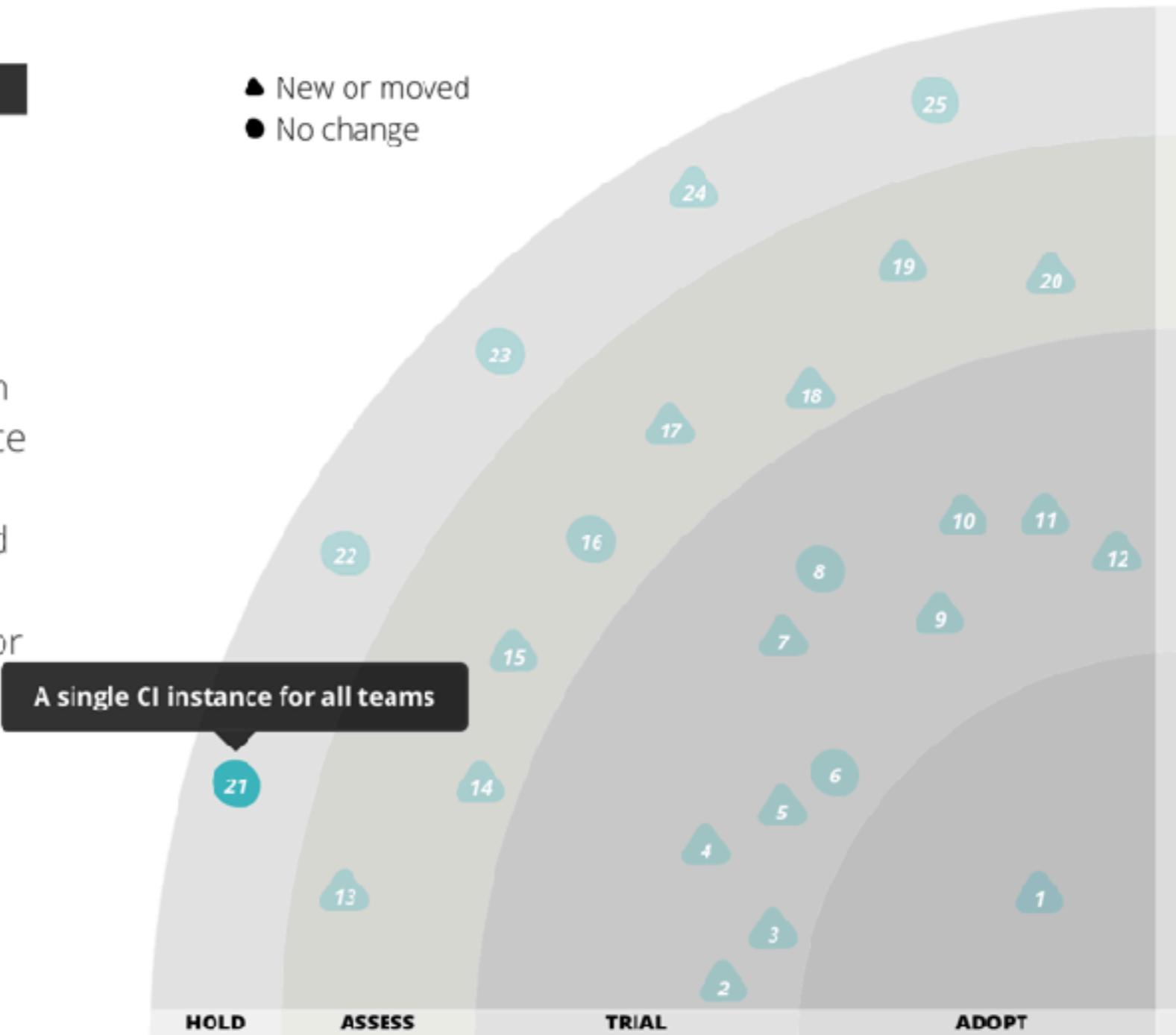
# Anti-pattern for CI Server

● HOLD ?

## 21. A single CI instance for all teams

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

- ▲ New or moved
- No change



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



# Let's start with CI/CD



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



# Jenkins

<https://jenkins.io/>



# Centralize Continuous Integration Server



# Jenkins

<https://jenkins.io/>



# Why Jenkins ?

Easy !!  
Extensible  
Scalable  
Opensource  
Large community  
**Lot of plugins**



# Who use Jenkins ?

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



Microsoft



redhat.

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

Atlassian

Datadog

JFrog

Mac Cloud

PagerDuty

XMission

<https://jenkins.io/>



# Hardware requirements

For Jenkins server

RAM +2GB

More CPU

More Disk



# Installation



# Jenkins in containers

Apache Tomcat

Jetty

JBoss

Websphere

WebLogic

Glassfish



# Download Jenkins



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

<https://jenkins.io/>



# Use Long Term Support (LTS)

## Getting started with Jenkins

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

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

### Long-term Support (LTS)

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

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

[Deploy Jenkins 2.46.3](#)

 [Deploy to Azure](#)

[Download Jenkins 2.46.3 for:](#)

Docker

FreeBSD

### Weekly

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

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

[Download Jenkins 2.65 for:](#)

Arch Linux

Docker

FreeBSD

Gentoo



# Start Jenkins

\$java -jar jenkins.war

Default port of server is **8080**

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

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



# Change port of Jenkins

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



# Open in browser

<http://localhost:8080>

Getting Started

## Unlock Jenkins

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

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

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

Administrator password

Continue



# Copy password from console

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

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

Please use the following password to proceed to installation:

a4b3a5231b8048419192d0c5afd3fce8

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

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



# Customize plugins

Getting Started



## Customize Jenkins

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

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.46.3



# Waiting

## Getting Started

## Getting Started

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

\*\* - required dependency

Jenkins 2.46.3



# Success

Getting Started

## Installation Failures

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

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

Jenkins 2.46.3

[Continue](#)

[Retry](#)



# Create a new user

Getting Started

## Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.46.3

Continue as admin

Save and Finish



# Ready to use

Getting Started

## Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

---

Jenkins 2.46.3



# Welcome to Jenkins

The screenshot shows the Jenkins dashboard. At the top right, there is a search bar, a user icon labeled 'somkiat', and a 'log out' link. Below the header, there is a navigation menu with links: 'Jenkins' (selected), 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. A large central message says 'Welcome to Jenkins!' and 'Please [create new jobs](#) to get started.' On the left side, there are two collapsed sections: 'Build Queue' (which says 'No builds in the queue.') and 'Build Executor Status' (which lists '1 Idle' and '2 Idle'). At the bottom right of the dashboard, there is a footer with the text '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', 'My Views', 'Credentials', and 'New View'. Below that are two sections: 'Build Queue' (empty) and 'Build Executor Status' (two idle executors). The main area is titled 'Manage Jenkins' and contains several configuration options: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon, highlighted with a red box), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench icon), 'Reload Configuration from Disk' (refresh icon), 'Manage Plugins' (puzzle piece icon), 'System Information' (monitor icon), 'System Log' (document icon), and 'Load Statistics' (heartbeat icon). The 'Configure Global Security' option is specifically highlighted.

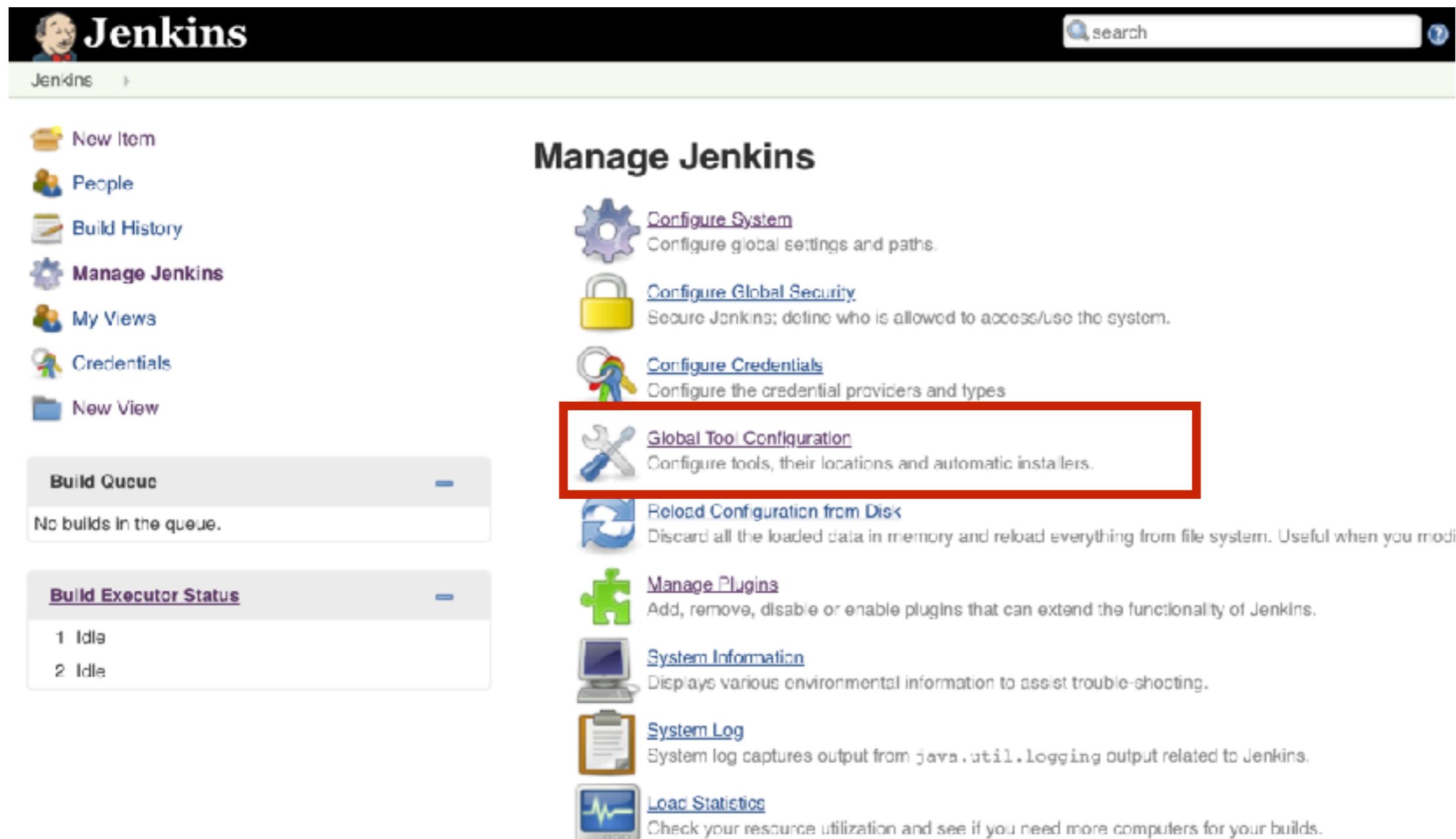
**Manage Jenkins**

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



# Global Tool Configuration

## Config tools, location and automatic installers



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

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



# Global Tool Configuration

Apache Maven  
JDK (Java Development Kit)  
Git  
Gradle  
Docker



# Manage Plugins

Add, remove, enable/disable plugins

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

**Manage Jenkins**

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



# Manage Plugins

Add, remove, enable/disable plugins

Filter:

Install	Name ↓	Version	Installed
		No updates	

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

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

This page lists updates to the plugins you currently use.



# Manage Plugins

## Filter plugins

Filter:

**Updates** Available Installed Advanced

Install	Name ↓	Version	Installed
No updates			

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

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

This page lists updates to the plugins you currently use.



# Finds Jenkins's plugin

Jenkins

Blog Documentation Plugins Use-cases Participate Sub-projects Resources

# Plugins Index

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

Browse Find plugins...

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



# Try to install a first plugin

Choose Available tab and select a plugin

The screenshot shows a Jenkins plugin management interface. At the top, there are tabs: 'Updates' (disabled), 'Available' (selected and highlighted with a red border), 'Installed' (disabled), and 'Advanced' (disabled). Below the tabs is a search bar labeled 'Filter:' with a magnifying glass icon. The main area displays a table of available plugins under the heading '.NET Development'. The columns are 'Name' and 'Version'. Each plugin entry includes a checkbox, a brief description, and the version number. At the bottom of the list are two buttons: 'Install without restart' and 'Download now and install after restart' (also highlighted with a red border). A status message 'Update information obtained: 9 hr 37 min ago' and a 'Check now' button are also present.

Name	Version
<a href="#">CCM Plug-in</a> This plug-in generates the trend report for CCM, an open source static code analysis program.	3.1
<a href="#">FxCop Runner plugin</a>	1.1
<a href="#">MSBuild Plugin</a>	1.27
<a href="#">MSTest plugin</a> Generates test reports for MSTest.	0.19
<a href="#">MSTestRunner plugin</a>	1.3.0
<a href="#">NAnt Plugin</a>	1.4.3
<a href="#">NCover plugin</a>	0.3
<a href="#">PowerShell plugin</a>	1.3
<a href="#">Violation Comments to Bitbucket Server Plugin</a> 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>	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

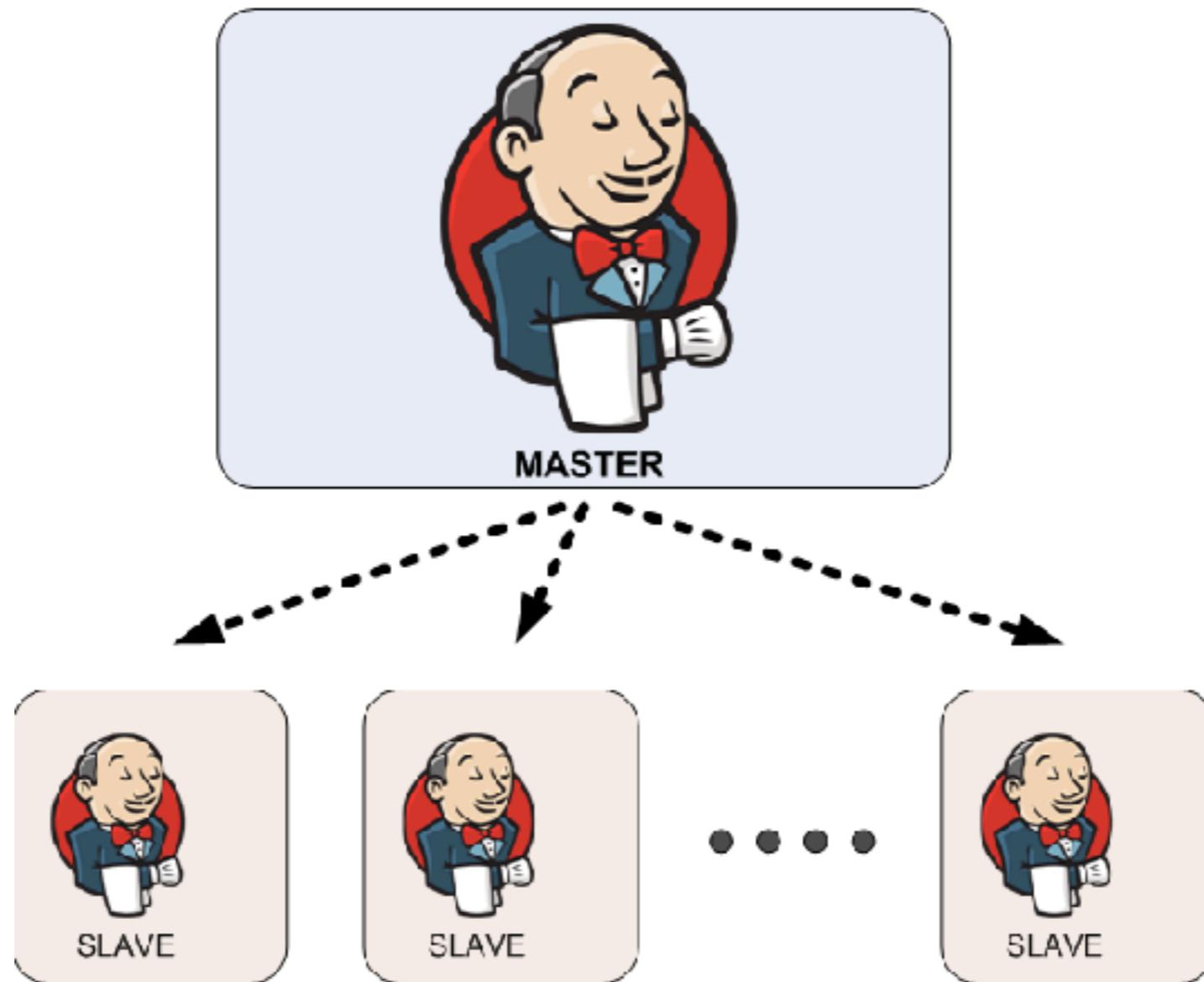
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Mac OS X (x86_64)	In sync	5.73 GB	463.00 MB	5.73 GB	0ms
	Data obtained	36 min	36 min	36 min	36 min	36 min	36 min

Refresh status



# Manage Nodes

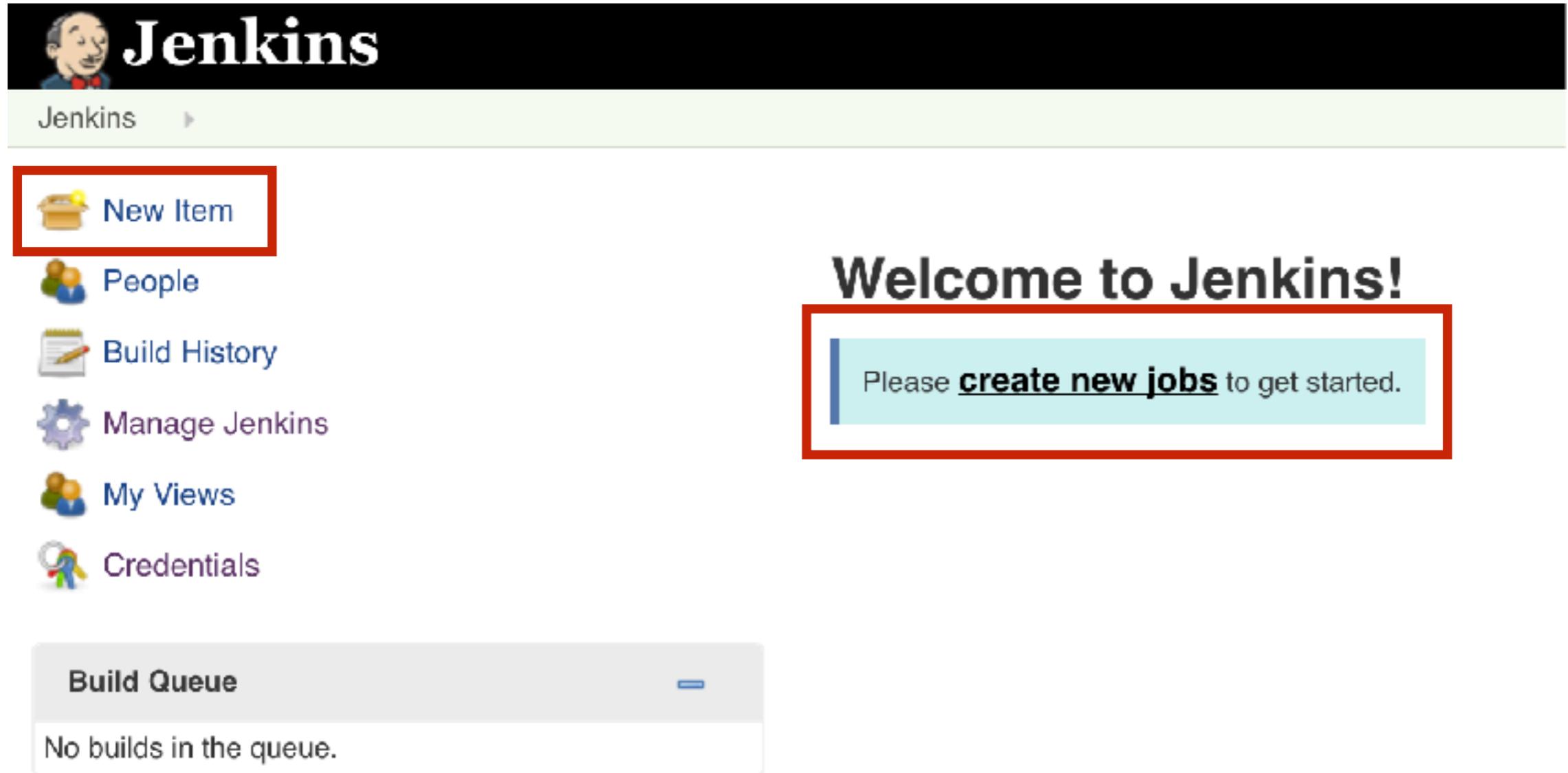
Master-slave concept to scale Jenkins



# Create a first Job



# 1. Create a new job



The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo. Below it is a navigation bar with the word "Jenkins" and a dropdown arrow. The main content area has a 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.



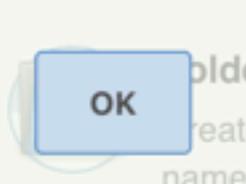
## External Job

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.



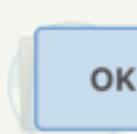
## External Job

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

Save [Apply](#)

None



# 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

Save    Apply

None



# 4.1 Advanced options

Screenshot of the Jenkins General configuration page. The 'General' tab is selected. The page contains several configuration options:

- 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 in dark grey. Below it, the 'Source Code Management' section displays three radio button options: 'None' (selected), 'Git', and 'Subversion'. The 'Build Triggers' section below contains five checkbox options, none of which are checked: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'.

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

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



# 6. Build trigger

When to run this job

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

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



# 6.1 Periodically

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

## Build Triggers

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

Schedule

**H 23 \* \* \***

⚠ No schedules so will never run

GitHub hook trigger for GITScm polling ?  
 Poll SCM ?



# 6.2 Poll SCM

Poll SCM ?

Schedule ?

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

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

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

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

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



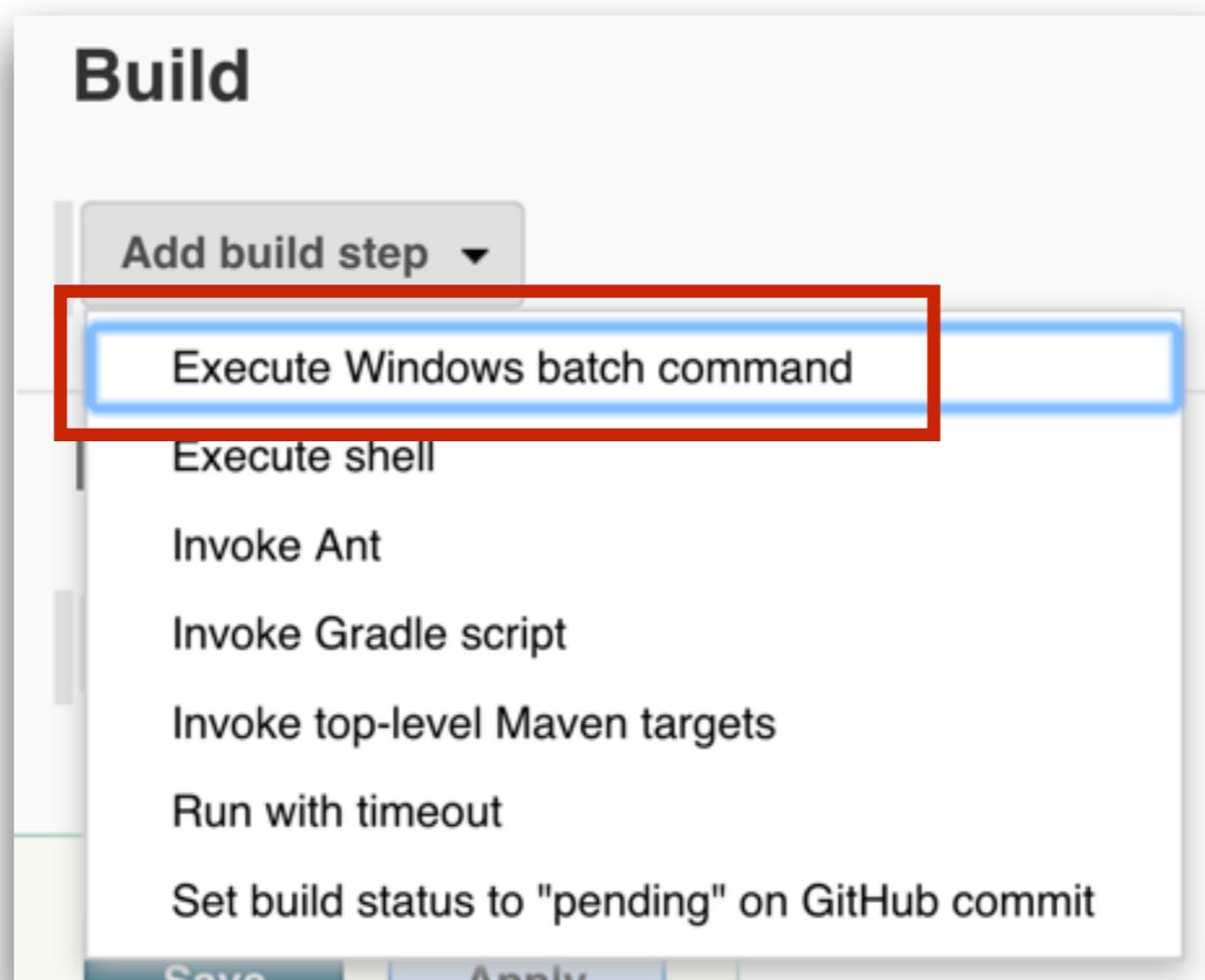
# 7. Add build step

What to run this job

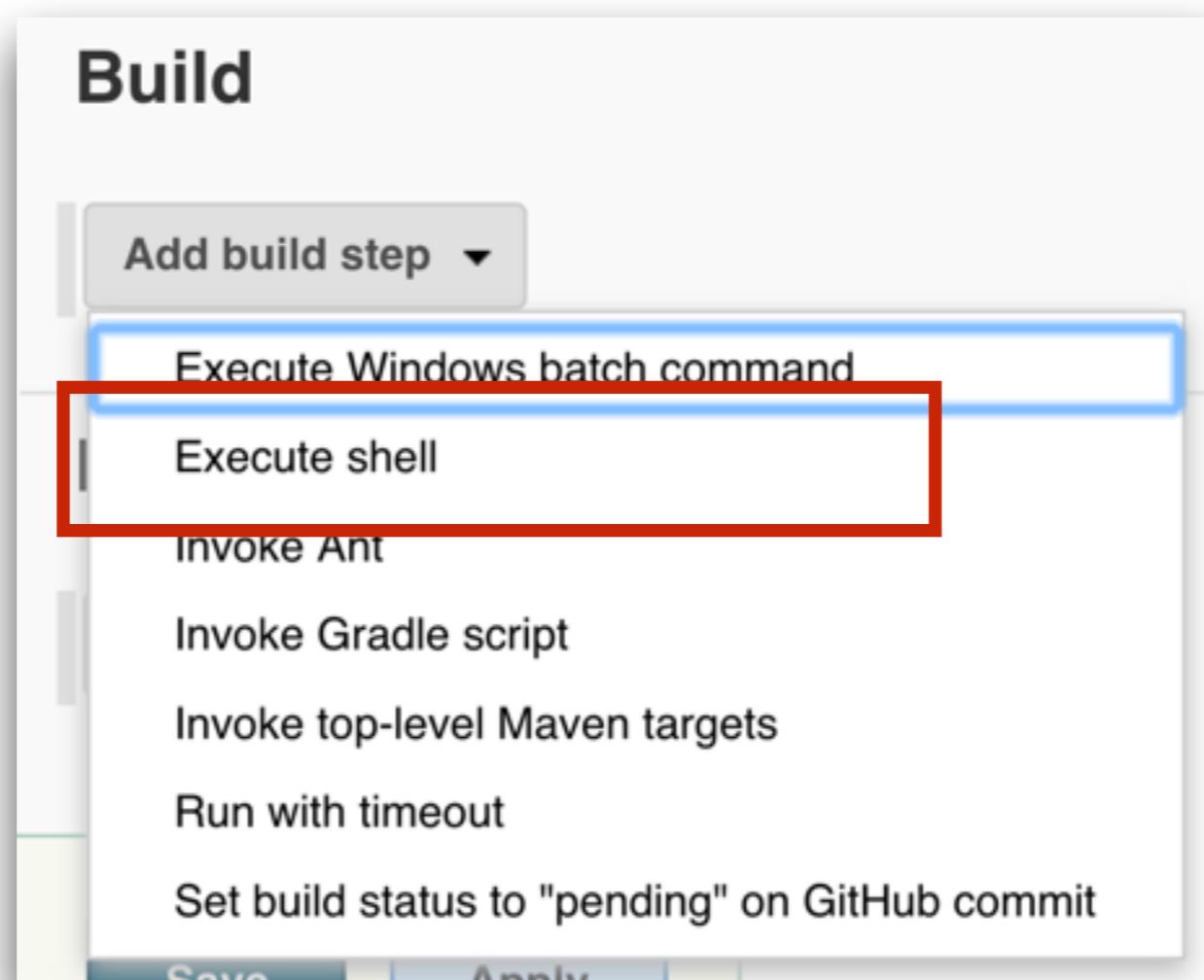
The screenshot shows the Jenkins job configuration interface. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, **Build**, and Post-build Actions. The **Build** tab is active. Below the tabs, there's a section titled **BUILD ENVIRONMENT** containing four checkboxes: Delete workspace before build starts, Abort the build if it's stuck, Add timestamps to the Console Output, and Use secret text(s) or file(s). Under the **Build** tab, there's a sub-section with a dropdown menu labeled "Add build step". The menu lists 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 "Add build step" panel are two buttons: "Save" and "Apply".



# 7.1 For Windows



# 7.2 For Linux/Mac



# 8. Post build actions

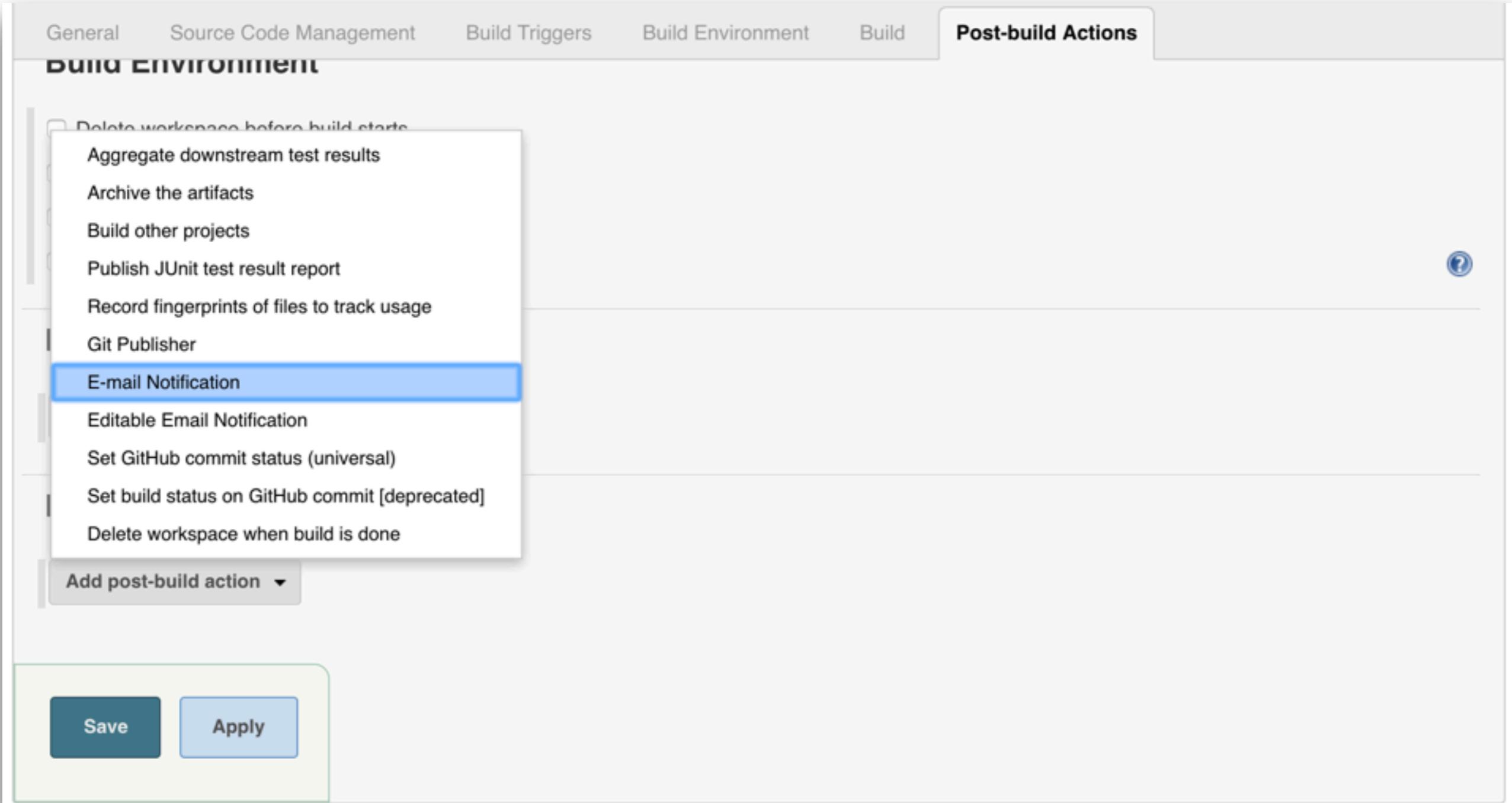
Generate reports

Send email

Run other jobs/projects



# 8. Post build actions



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

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

**BUILD ENVIRONMENT**

Do late 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**
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

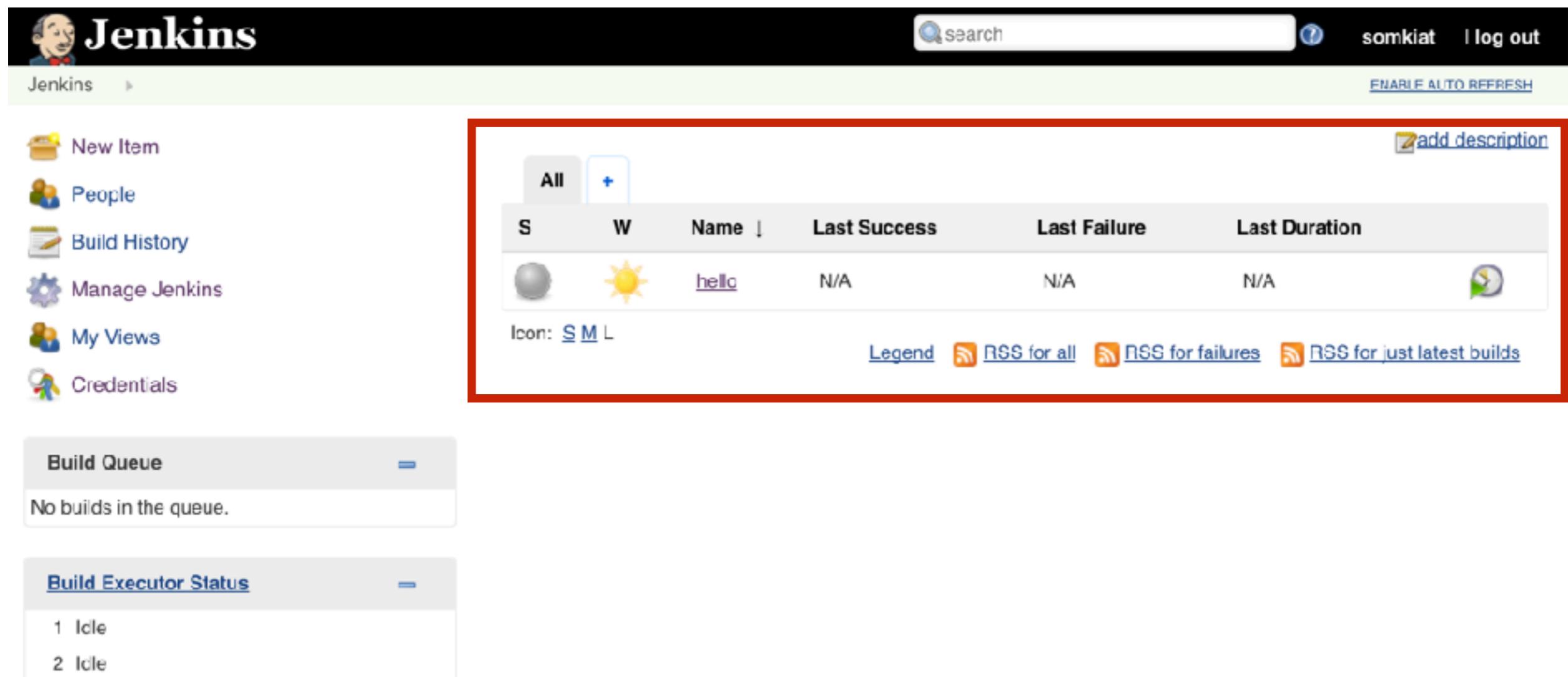
Add post-build action ▾

Save    Apply



# 9. Run your job

## Manual and Scheduler run



The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, My Views, and Credentials. The main area displays a table for a job named "hello". The job has a yellow sun icon, indicating it's successful. The table columns are: S (Status), W (Last Success), Name (hello), Last Success (N/A), Last Failure (N/A), and Last Duration. Below the table, there are icons for S (Small), M (Medium), and L (Large). At the bottom of the table area, there are links for Legend, RSS for all, RSS for failures, and RSS for just latest builds. A red box highlights the "hello" job entry in the main table.

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](#)



# 9. Run your job

Start build your job

Jenkins

search somkiat log out

New Item People Build History Manage Jenkins My Views Credentials

All +

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

add description

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

Build Queue

No builds in the queue.

Build Executor Status

1 Idle  
2 Idle



# 9. Run your job

Start build your job

The screenshot shows the Jenkins interface for a project named "hello". The top navigation bar shows "Jenkins" and "hello". The main content area is titled "Project hello". On the left, there is a sidebar with the following options:

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now** (this option is highlighted with a red box)
- Delete Project
- Configure

On the right side, there are two links:

- Workspace
- Recent Changes



# 9. Run your job

See your job's output

Jenkins > hello > #1

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

**Console Output**

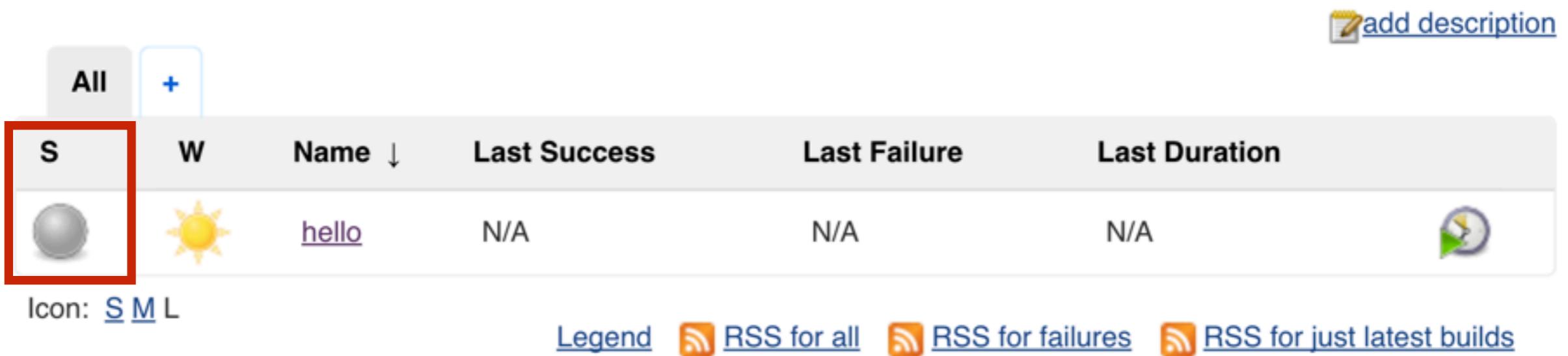
Started by user [Somkiat Puisungnoen](#)  
Building in workspace /Users/somkiat/Downloads/set/workspace/hello  
Finished: SUCCESS

Edit Build Information  
 Delete Build  
 Next Build



# 10. See job's status

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



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

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

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

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

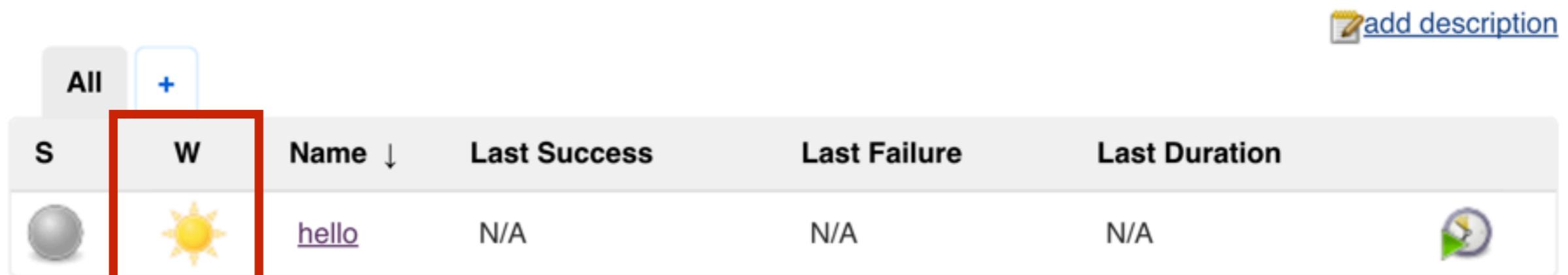
**Blue = build success**

**Red = build failure**

**Gray = disabled/never executed**



# 11. See job's health



A screenshot of a Jenkins job health status page. At the top, there are buttons for 'All' and '+'. To the right is a 'add description' button with a pencil icon. Below this is a table with columns: 'S' (Status), 'W' (Health), 'Name ↓', 'Last Success', 'Last Failure', and 'Last Duration'. The first row shows a grey circle icon for 'S' and a yellow sun icon for 'W'. The 'Name' is 'hello', 'Last Success' is 'N/A', 'Last Failure' is 'N/A', and 'Last Duration' is 'N/A'. To the right of the table is a small green and yellow icon. Below the table, it says 'Icon: S M L'. At the bottom, there are links for 'Legend', 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
	W	hello	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

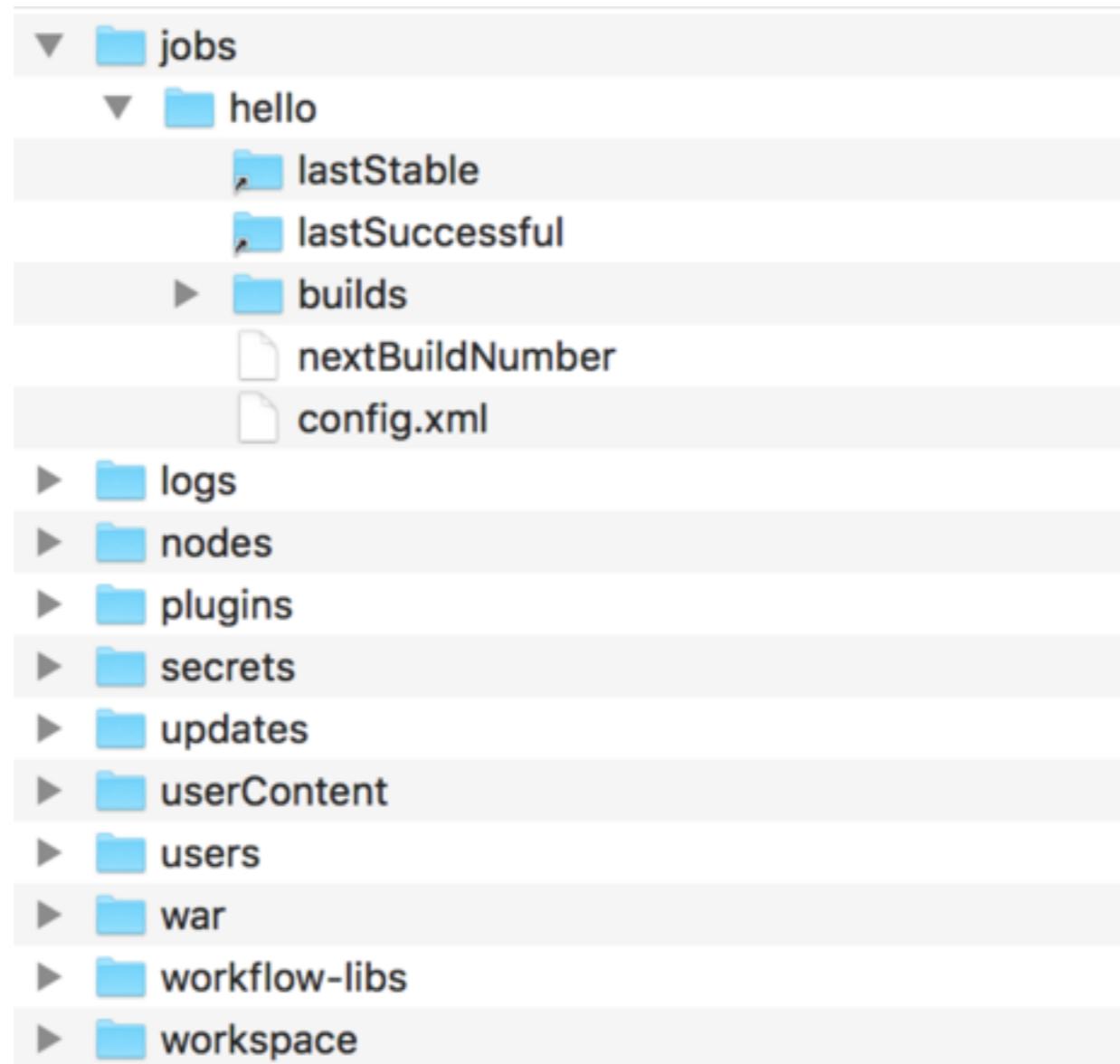


# Backup your Jenkins



# 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**

---

[Home](#)  
[7z Format](#)  
[LZMA SDK](#)  
[Download](#)  
[FAQ](#)  
[Support](#)  
[Links](#)

---

[English](#)  
[Chinese Simpl.](#)  
[Chinese Trad.](#)  
[Esperanto](#)  
[French](#)

**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

## 7-Zip

## License

**7-Zip** is **free software** with **open source**. The most of the code is under the [MIT 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. You can't use 7-Zip to compress files in a way that violates the law.

<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

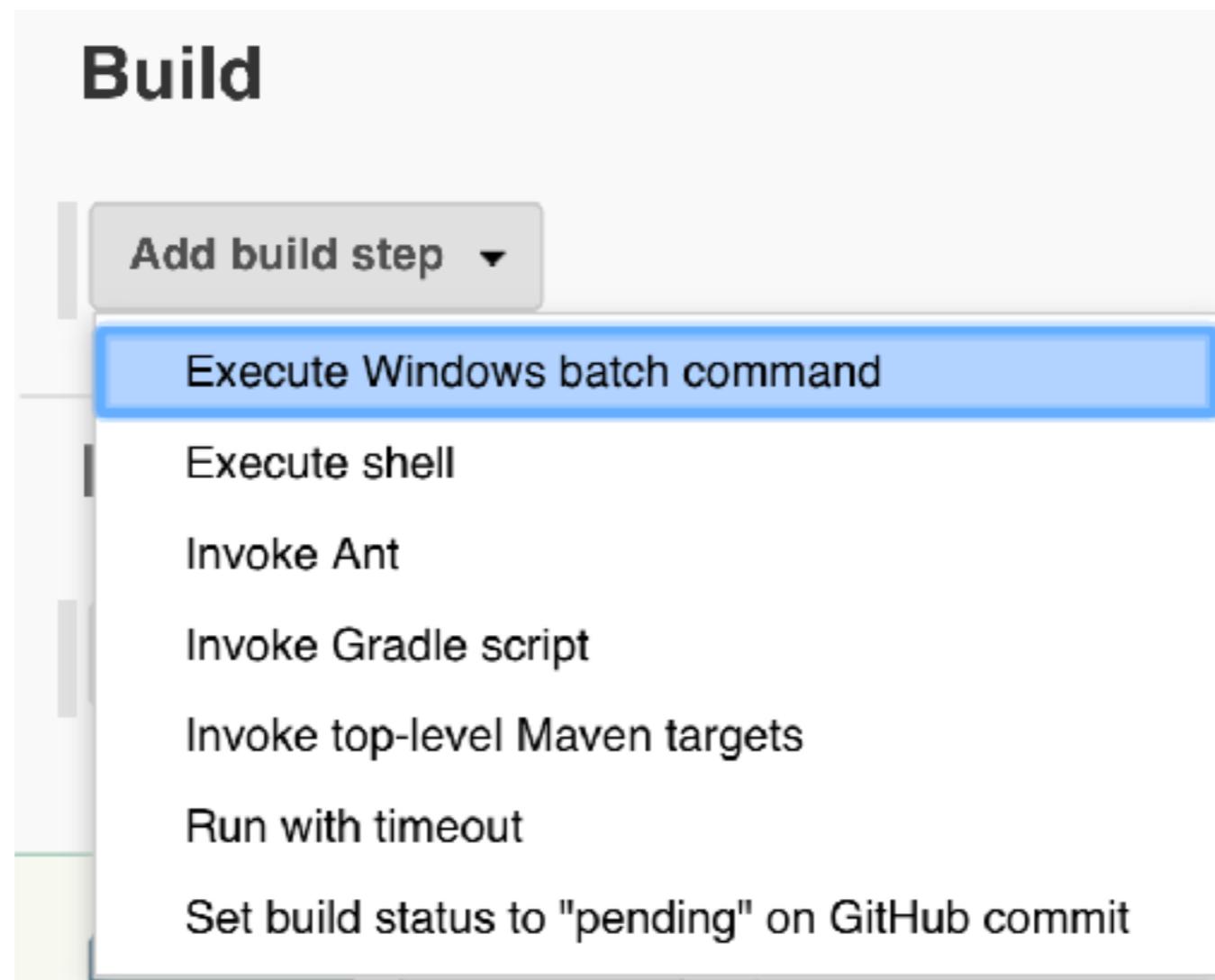
Using Build periodically => H 23 \* \* 7

The screenshot shows the Jenkins 'Build Triggers' configuration page. The top navigation bar includes tabs for General, Source Code Management, Build Triggers (which is selected and highlighted in black), Build Environment, Build, and Post-build Actions. The 'Build Triggers' section contains three options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', and 'Build periodically'. The third option is checked, and its configuration panel is expanded, showing the schedule 'H 23 \* \* 7'. Below the schedule, a note 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 two additional unselected triggers: 'GitHub hook trigger for GITScm polling' and 'Poll SCM'.



# 3. Build

Choose way to run a script



# 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 left sidebar contains links: "Back to Dashboard", "Status", "Changes", "Workspace", "Build Now" (which is highlighted with a red box), "Delete Project", and "Configure". Below the sidebar is a "Build History" section with a search bar and RSS feed links for "RSS for all" and "RSS for failures". The main content area is titled "Project backup\_jenkins" and includes links to "Workspace" and "Recent Changes". A "Permalinks" section is also present.

Jenkins

search

?

Jenkins > backup\_jenkins >

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Build History

trend

find

RSS for all RSS for failures

Project backup\_jenkins

Workspace

Recent Changes

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/t5/8kg23e_97z9dw44tfso/d6dqw000Cgn/T/hudson6420145093817228191.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/ccnfig.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.emailtext.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/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.lastsxecversion /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/var /Users/somkiat/data/slide/ci-cd/swpark/software/keep/workspace
/Users/somkiat/data/slide/ci-cd/swpark/software/keep/workspace

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2015-05-21
p7zip Version 15.02 (locale=utf8,Utf16=on,HugeFiles=on,64 bits,4 CPUs x64)

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

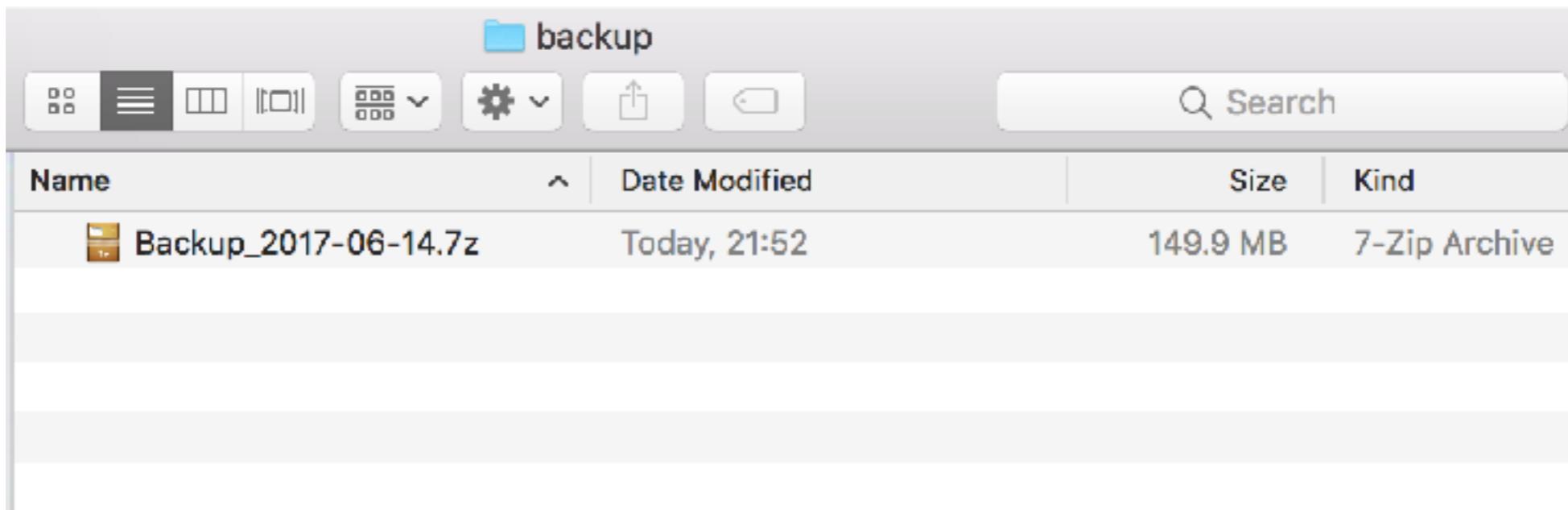
creating archive: /Users/somkiat/data/slide/ci-cd/swpark/software/backup/backup_2017-06-14.7z

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 manager interface. At the top, there is a red circle with the number '1'. Below it is a search bar with the placeholder 'Filter: periodic backup' and a magnifying glass icon. Below the search bar, there are tabs: 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A table lists available plugins. The first row shows the 'Periodic Backup' plugin, which is currently installed (version 1.3). The table has columns for 'Name' and 'Version'. At the bottom of the table, there are three buttons: 'Install without restart' (disabled), 'Download now and install after restart', and 'Check now'. A status message says 'Update information obtained: 9 hr 52 min ago'.

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

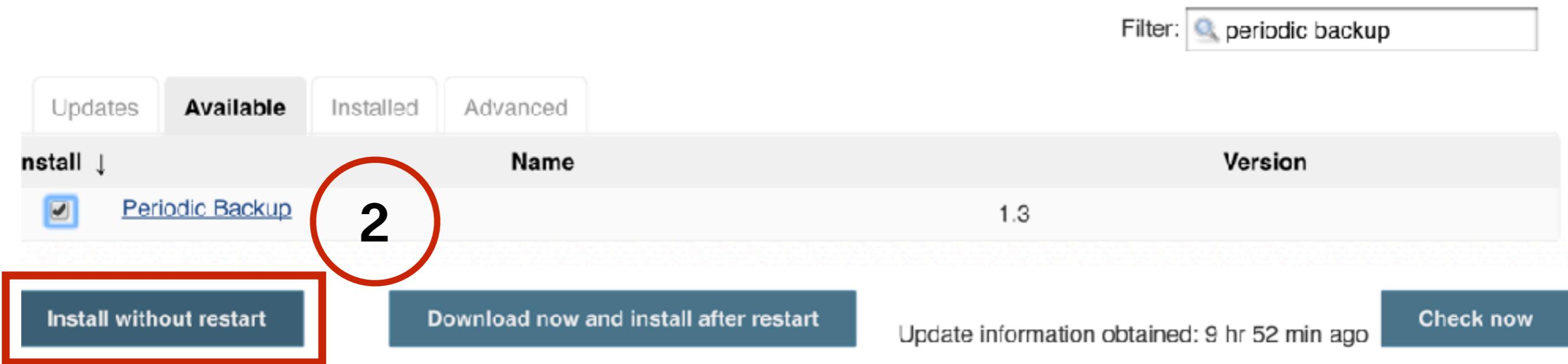
Filter:

Updates Available Installed Advanced

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

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

2



# 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

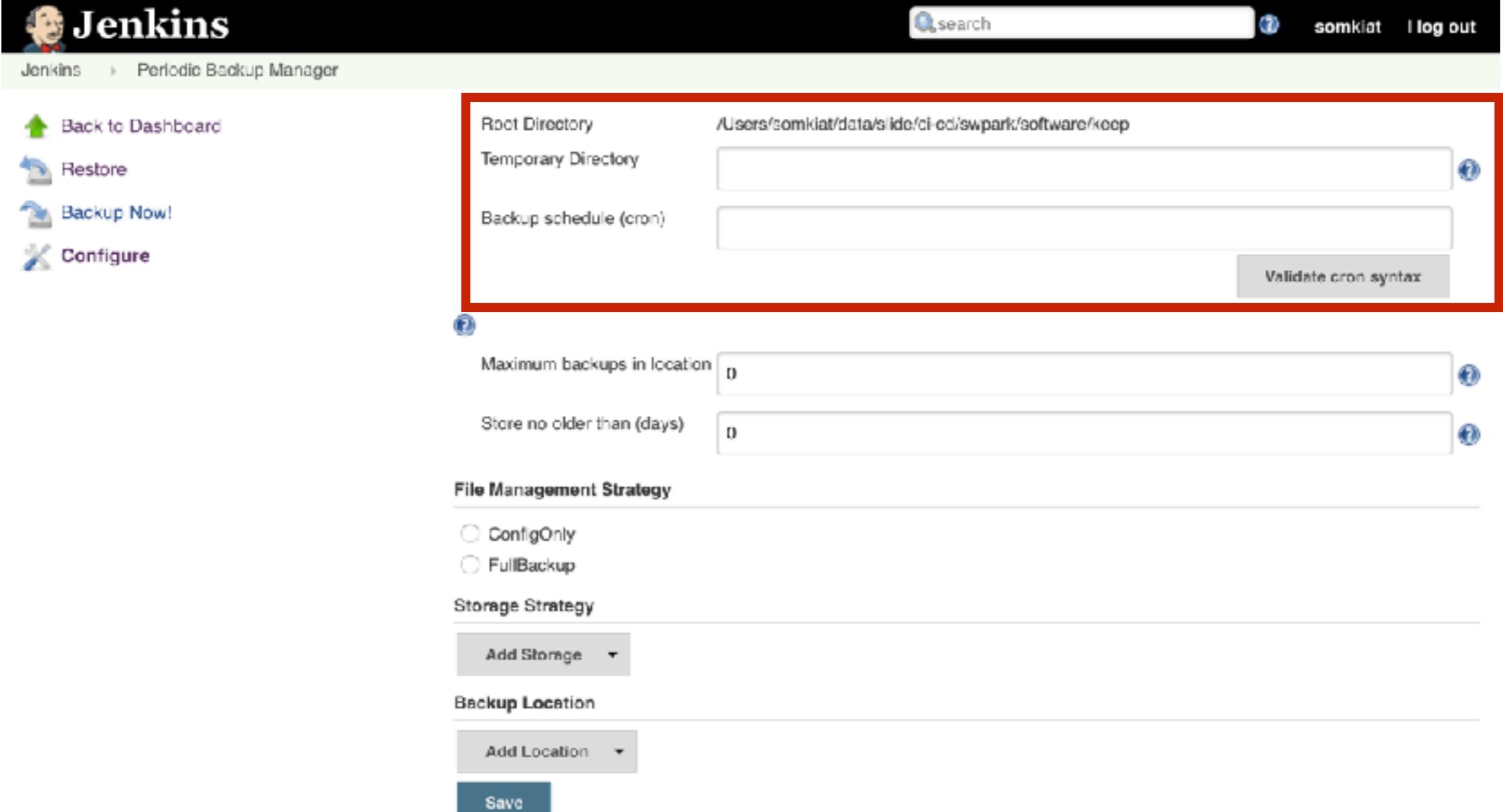
The screenshot shows the Jenkins Periodic Backup Manager configuration page. At the top, there is a navigation bar with a Jenkins logo, a search bar, and user information (somklat). Below the navigation bar, the page title is "Periodic Backup Manager". On the left, there is a sidebar with links: "Back to Dashboard", "Restore", "Backup Now!", and "Configure". The main content area contains several configuration fields:

- "Root Directory": /Users/somklat/data/slave/ci-cd/swpark/software/keep
- "Temporary Directory": (empty input field)
- "Backup schedule (cron)": (empty input field)
- "Validate cron syntax": A button to validate the cron schedule.
- "Maximum backups in location": 0
- "Store no older than (days)": 0
- "File Management Strategy": Radio buttons for "ConfigOnly" and "FullBackup".
- "Storage Strategy": A dropdown menu currently set to "Add Storage".
- "Backup Location": A dropdown menu currently set to "Add Location".
- "Save": A blue "Save" button at the bottom.



# 3. Using plugin

Temporary directory of achieve files while restore and backup process



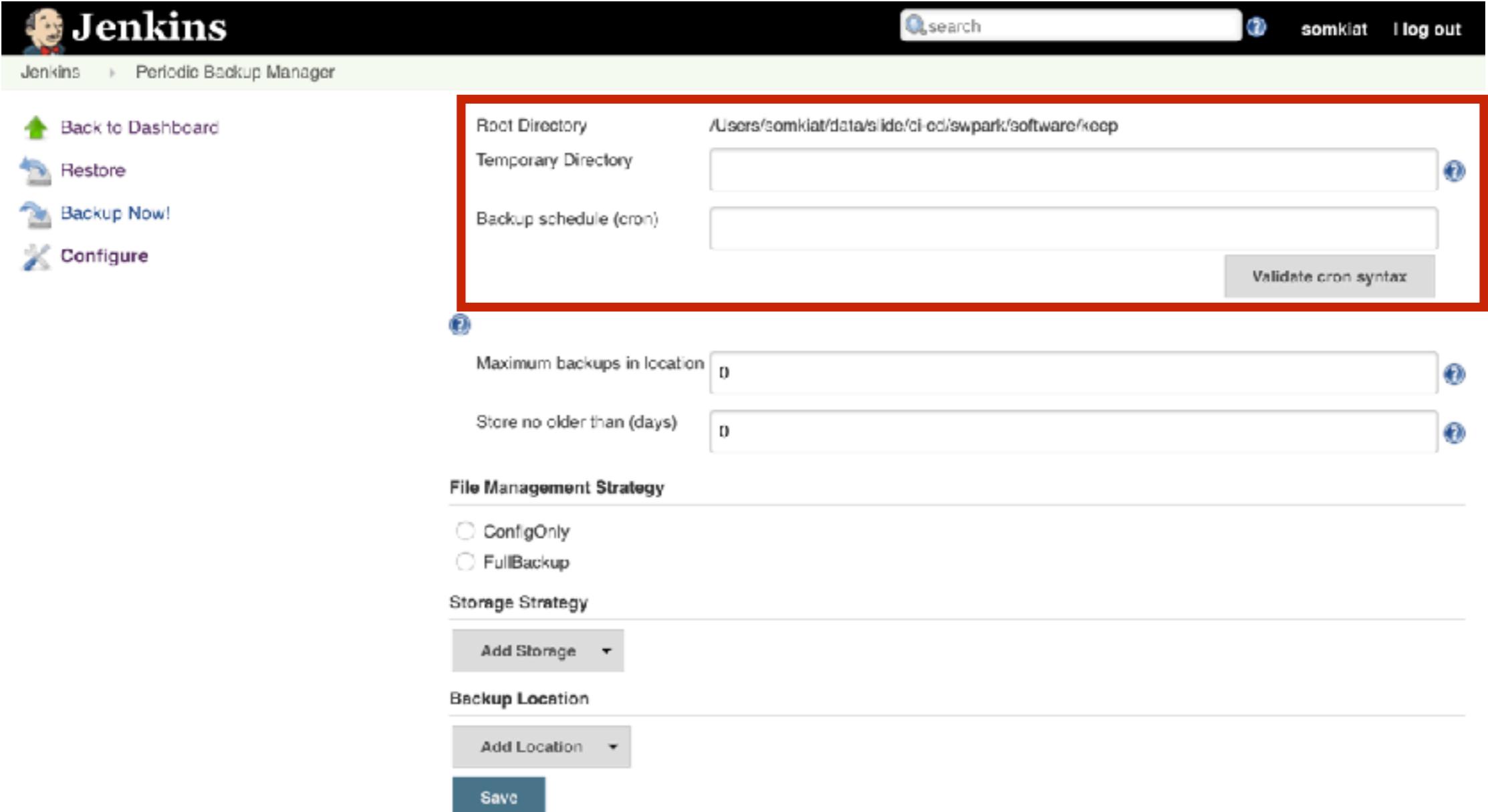
The screenshot shows the Jenkins Periodic Backup Manager configuration page. On the left, there's a sidebar with links: Back to Dashboard, Restore, Backup Now!, and Configure. The main area has several configuration sections:

- Root Directory:** /Users/somkiat/data/slide/ci-cd/swpark/software/keep
- Temporary Directory:** (highlighted with a red box)
- Backup schedule (cron):** (highlighted with a red box)
- Validate cron syntax:** (button next to the cron field)
- Maximum backups in location:** 0
- Store no older than (days):** 0
- File Management Strategy:** (radio buttons for ConfigOnly and FullBackup, both unselected)
- Storage Strategy:** (dropdown menu showing "Add Storage")
- Backup Location:** (dropdown menu showing "Add Location")
- Save:** (blue button at the bottom)



# 3. Using plugin

Backup schedule is a crontab format



The screenshot shows the Jenkins Periodic Backup Manager configuration page. On the left, there's a sidebar with links: Back to Dashboard, Restore, Backup Now!, and Configure. The main area has several input fields:

- Root Directory: /Users/somklat/data/slide/ci-cd/swpark/software/keep
- Temporary Directory: (empty)
- Backup schedule (cron): (empty) Validate cron syntax
- Maximum backups in location: 0
- Store no older than (days): 0

Below these are sections for File Management Strategy (ConfigOnly, FullBackup selected), Storage Strategy (Add Storage dropdown), and Backup Location (Add Location dropdown). At the bottom is a Save button.



# 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/somklat/data/slide/ci-cd/swpark/software/keep

Temporary Directory:

Backup schedule (cron):

Validate cron syntax

File Management Strategy

Maximum backups in location: 0

Store no older than (days): 0

Storage Strategy

ConfigOnly

FullBackup

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/somklat/data/slide/ci-cd/swpark/software/keep

Temporary Directory:

Backup schedule (cron):

Validate cron syntax

File Management Strategy

Maximum backups in location: 0

Store no older than (days): 0

Storage Strategy

ConfigOnly

FullBackup

Backup Location

Add Location

Save



# 4. Example of configuration

Root Directory	/Users/somkiat/data/slide/ci-cd/swpark/software/keep	
Temporary Directory	Users/somkiat/data/slide/ci-cd/swpark/software	(?)
Backup schedule (cron)	H 23 * * *	
	This cron is OK	<button>Validate cron syntax</button>
(?)		
Maximum backups in location	5	(?)
Store no older than (days)	15	(?)



# 5. File management strategy

**ConfigOnly** is backup only configuration

## File Management Strategy

---

- ConfigOnly
- FullBackup



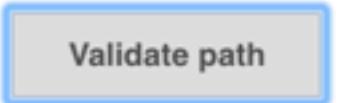
# 5. Backup locations

## Backup Location

 LocalDirectory

Backup directory path  

Enable this location 

directory "/Users/somkiat/backup" OK  

[Add Location ▾](#)

[Save](#)



# Try to run your job



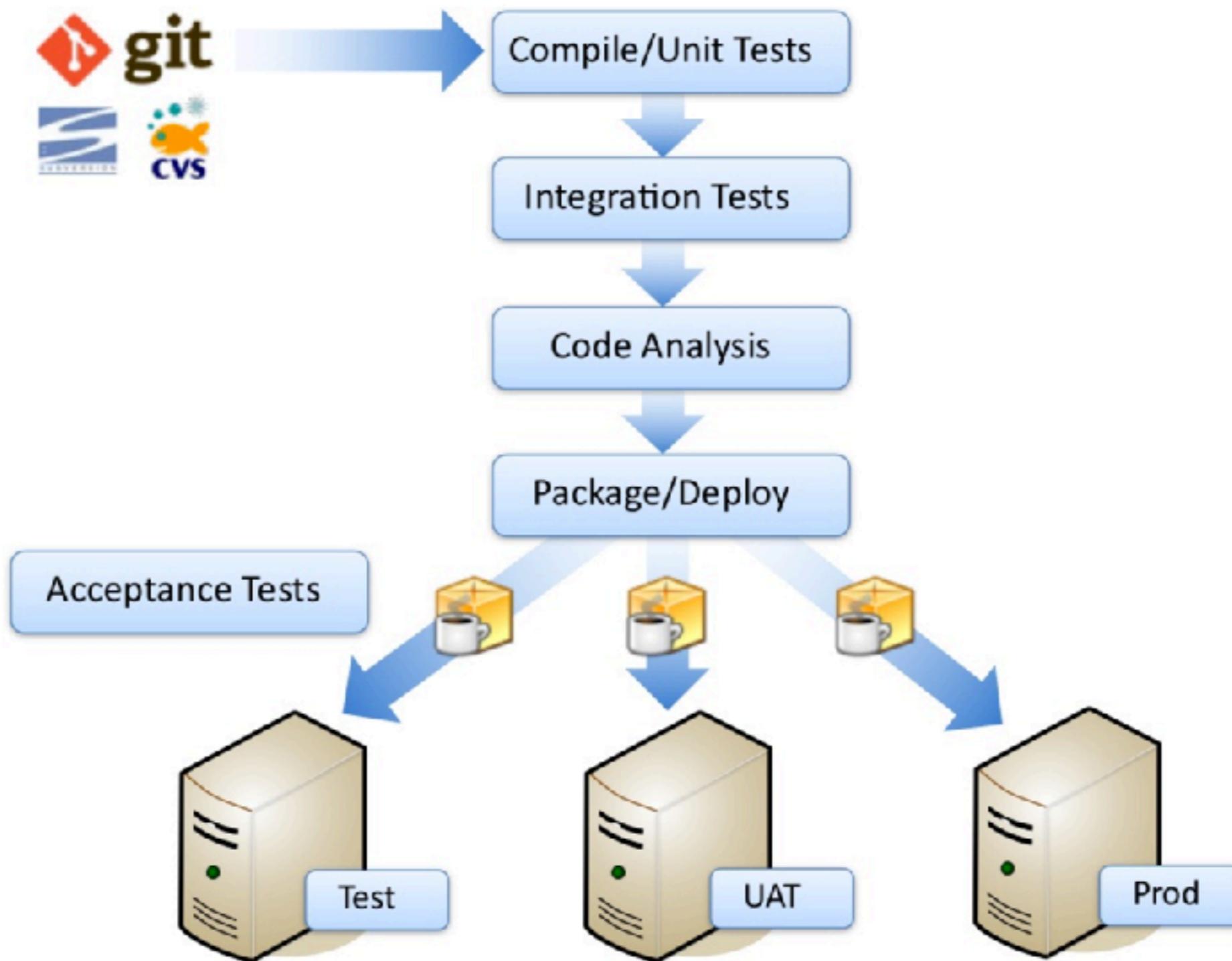
# Jenkins driven by Plugins !!



# Workshop with Java Web Application

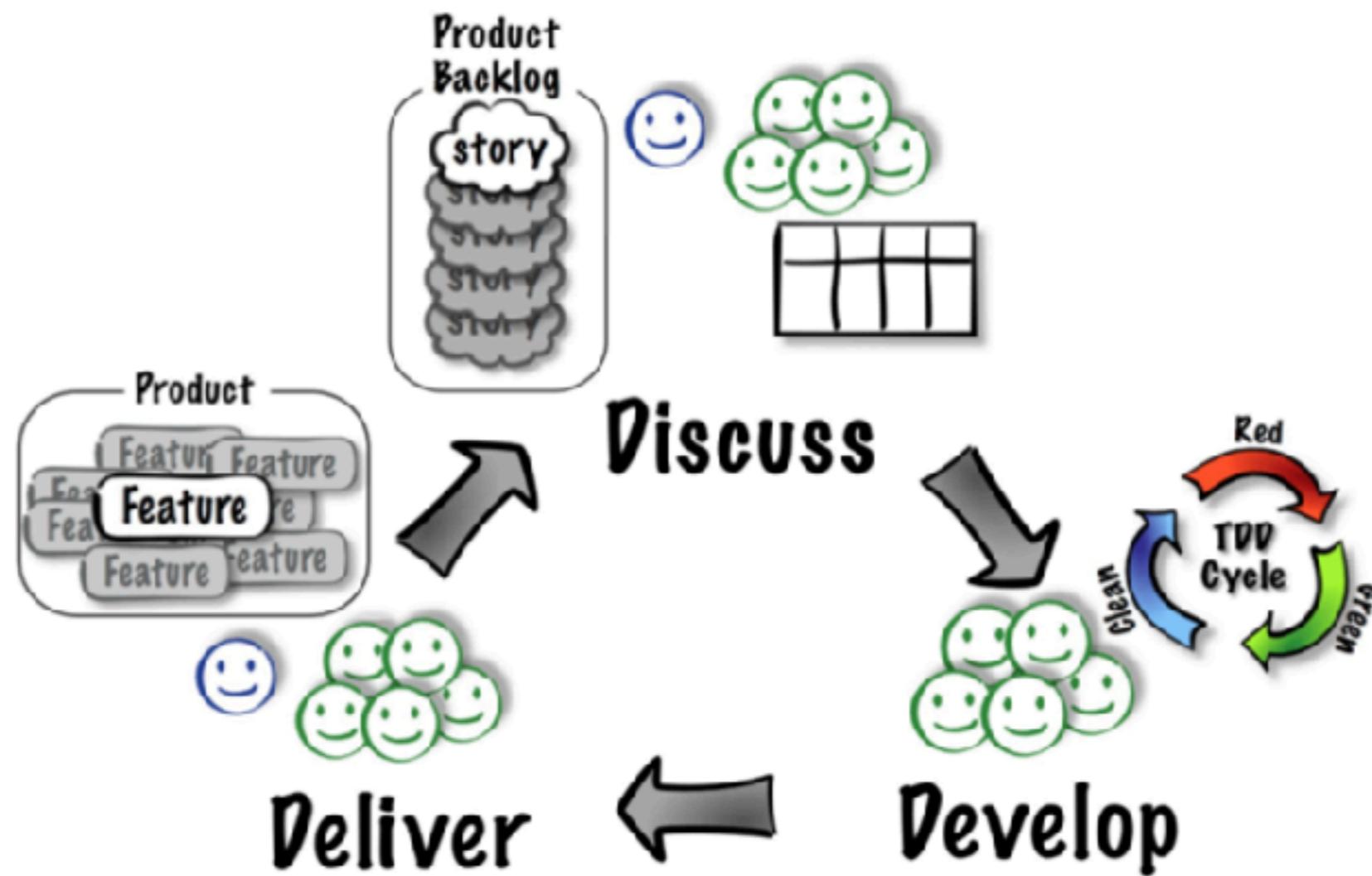


# Pipeline of this project



# 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 Git website at <https://git-scm.com/>. At the top left is the Git logo with the tagline "distributed even if your workflow isn't". A search bar is at the top right. Below the header, there's a brief introduction: "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 is a diagram illustrating the distributed nature of Git, showing multiple repositories connected by bidirectional arrows. Below this, a section titled "Learn Git in your browser for free with Try Git." includes a "Try It" button. The main content area has several sections: "About" (with a gear icon), "Documentation" (with a book icon), "Downloads" (with a download icon), and "Community" (with a speech bubble icon). On the right side, a large monitor displays the "Latest source Release 2.13.1" and a "Downloads for Mac" button.

git --distributed-even-if-your-workflow-isnt

Search entire site...

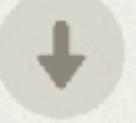
Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

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**.

 Learn Git in your browser for free with [Try Git](#).

 **About**  
The advantages of Git compared to other source control systems.

 **Documentation**  
Command reference pages, Pro Git book content, videos and other material.

 **Downloads**  
GUI clients and binary releases for all major platforms.

 **Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.13.1**  
Release Notes (2017-06-05)  
Downloads for Mac

<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 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 EE 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 displays two download options: "Java Platform (JDK) 8u131" with a Java logo and "NetBeans with JDK 8" with a 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, stating: "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." To the right is a sidebar titled "Java SDKs and Tools" with links to 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, and 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 a large "Maven" logo with a stylized feather icon. Below the header, the breadcrumb navigation shows "Apache / Maven / Welcome to Apache Maven". On the right, it says "Last Published: 2017-06-03". The main content area has a title "Welcome to Apache Maven". It describes Maven as a software project management and comprehension tool based on the POM model. It includes sections for "Use" (Download, Install, Run Maven; Configure, Use Maven and Maven Plugins), "Extend" (Write Maven Plugins; Improve the Maven Repository), and "Contribute" (Help Maven; Develop Maven). A sidebar on the left lists links such as Welcome, License, Download, Install, Configure, Run, IDE Integration, About Maven, What is Maven?, Features, FAQ, Support and Training, Documentation, Maven Plugins, Index (category), and 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

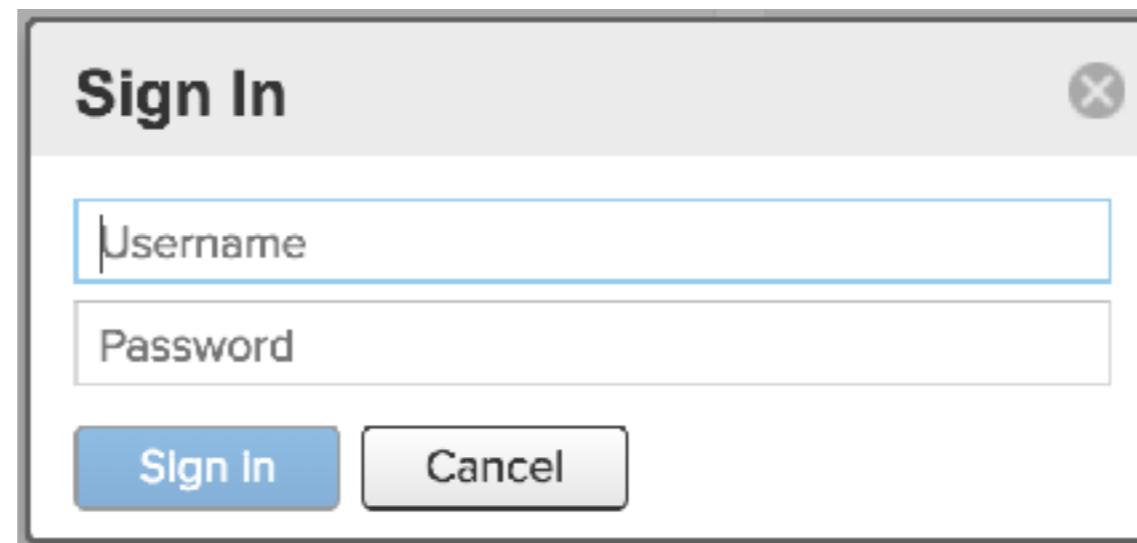
The screenshot shows the Nexus Repository Manager OSS 3.8.0-02 interface. The URL in the browser bar is <http://localhost:8081>. The top navigation bar includes a logo, the text "Nexus Repository Manager OSS 3.8.0-02", a search bar labeled "Search components", and a close button. On the left, a sidebar titled "Browse" has a "Welcome" tab selected, along with links for "Search" and "Browse". The main content area is titled "Welcome" and "Welcome to Nexus Repository Manager!". It features a "Get Started" section with four items: "New in 3.8.0" (with a star icon), "Configuration" (with a wrench icon), "Repository Formats" (with a folder icon), and "Documentation" (with a book icon). Each item has a link to its respective documentation page.



# Login

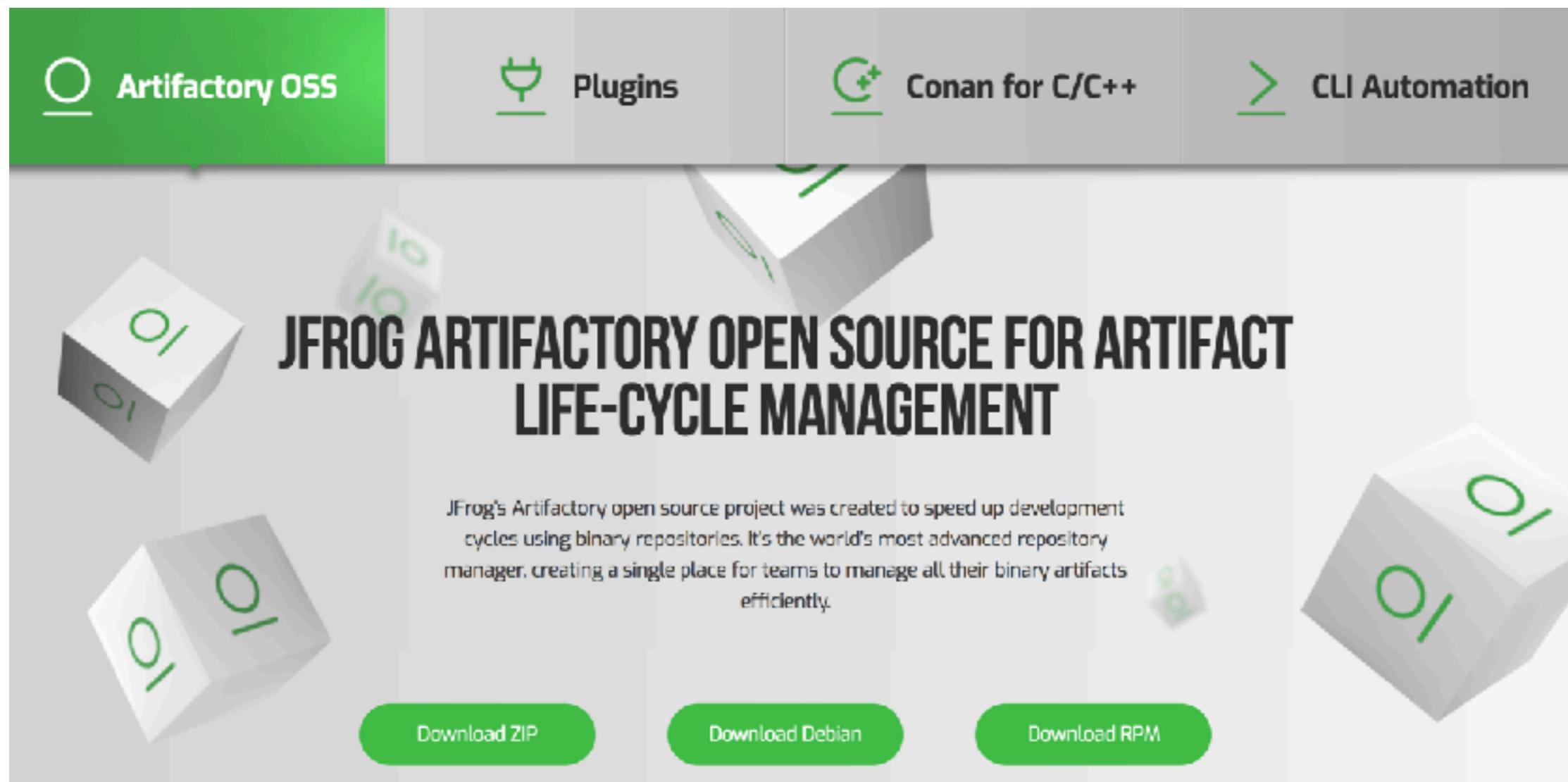
user = admin

password = admin123



# 4.2 JFrog Artifactory

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



<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, followed by a blue ribbon icon. Below the navigation, a large blue banner features the text "The leading product for CONTINUOUS CODE QUALITY". It includes three icons: "Code Smells" with a radiation symbol, "Bugs" with a bug icon, and "Vulnerabilities" with a lock 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 stylized 'S' is followed by the text "Used by more than 80,000 organizations". A footer navigation bar at the bottom lists "On 20+ Code Analyzers", "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: 'Bugs' (0), 'Vulnerabilities' (0), and 'Code Smells' (0). 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

Bugs Vulnerabilities Code Smells

Log in Read documentation

Multi-Language

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

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



# 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**

The Apache Tomcat<sup>®</sup> 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	<input type="checkbox"/>
Name	Default
Path to Git executable	git 
<input type="checkbox"/> Install automatically 	





# Apache Maven

Manage Jenkins -> Global tool configuration

## Maven

### Maven installations

#### Maven

Name

 Required

Install automatically 

#### Install from Apache

Version

**Delete Installer**

**Add Installer** ▾

**Delete Maven**

**Add Maven**

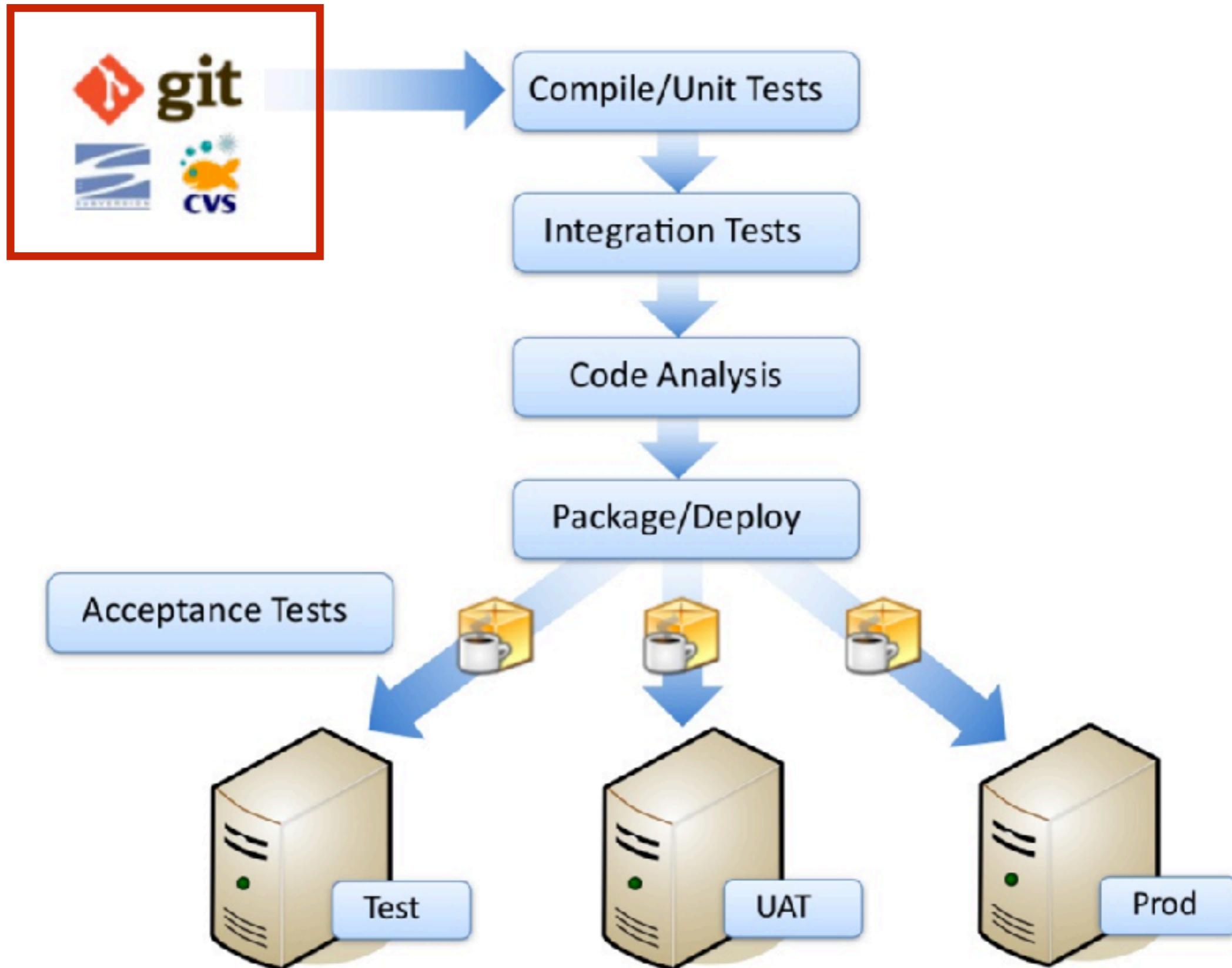
List of Maven installations on this system



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



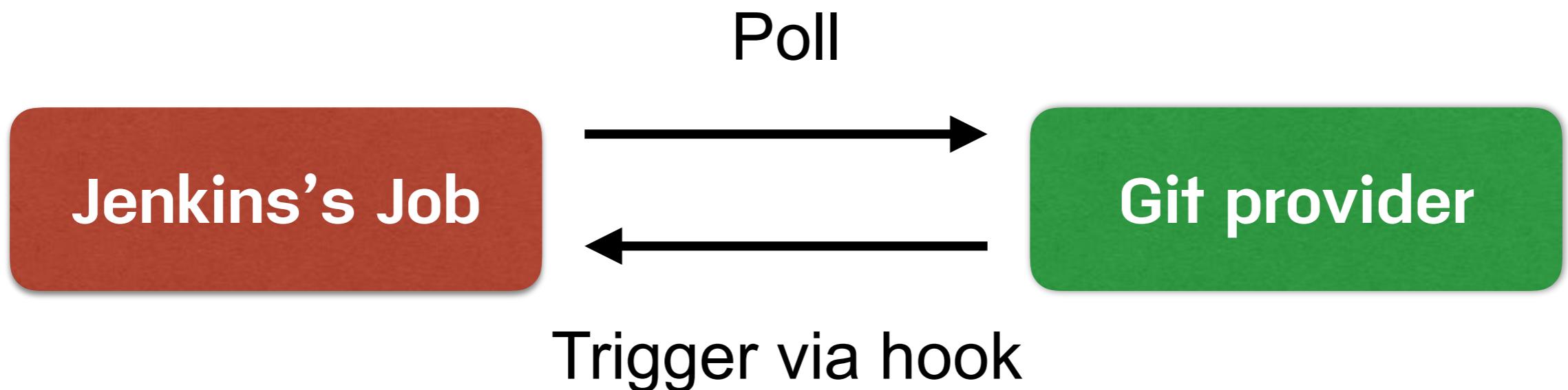
1



# 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

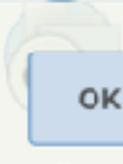
» Required field

 **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, there is a single branch entry with the specifier `*/*master` and an "Add Branch" button. The "Repository browser" is set to "(Auto)". At the bottom, there is an "Additional Behaviours" section with an "Add" button, and "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

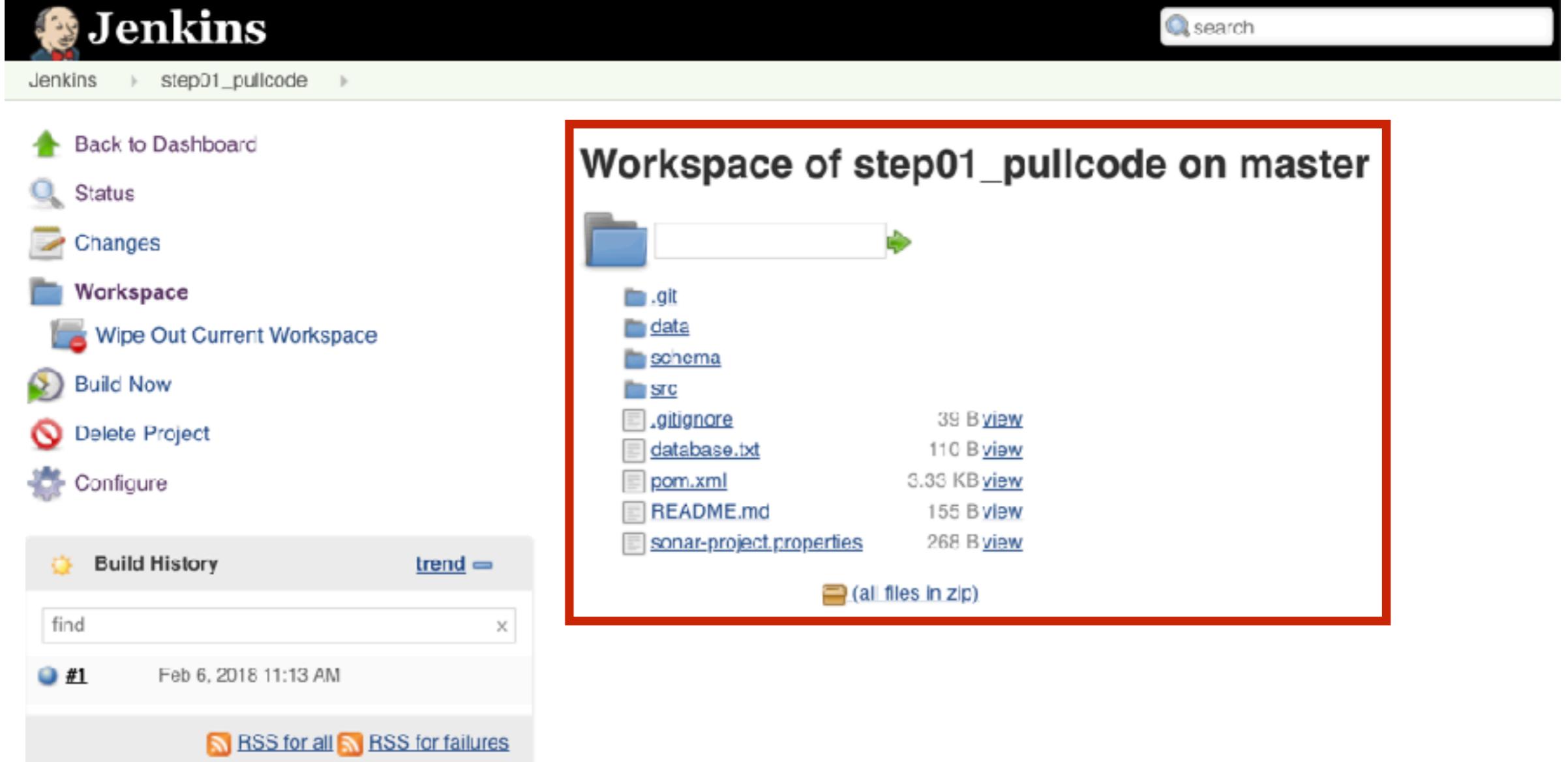
RSS for all RSS for failures

Permalinks



# Step 4 See result

See all source code in workspace



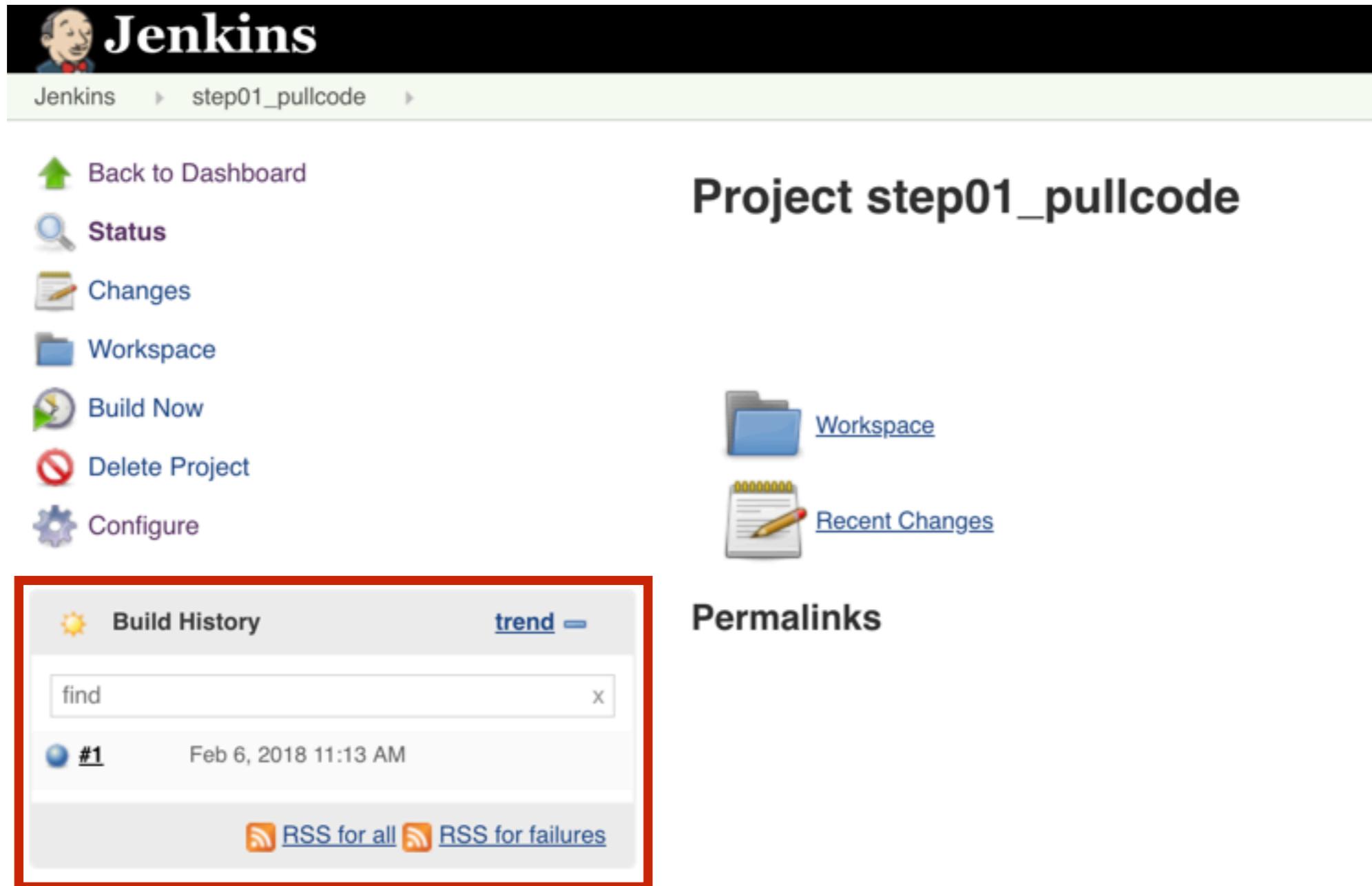
The screenshot shows the Jenkins interface for the project "step01\_pullcode". The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace (which is selected and highlighted in blue), Wipe Out Current Workspace, Build Now, Delete Project, and Configure. The main content area displays the "Workspace of step01\_pullcode on master" with a red border around it. It shows a file tree with .git, data, schema, and src folders, along with files like .gitignore, database.txt, pom.xml, README.md, and sonar-project.properties. A "View All" link is also present. At the bottom right of the workspace area is a "zip" icon labeled "(all files in zip)".

File/Folder	Size	Action
.git		
data		
schema		
src		
.gitignore	39 B	<a href="#">view</a>
database.txt	110 B	<a href="#">view</a>
pom.xml	3.33 KB	<a href="#">view</a>
README.md	155 B	<a href="#">view</a>
sonar-project.properties	268 B	<a href="#">view</a>



# Step 4 See result

Build History keep in logs file !!

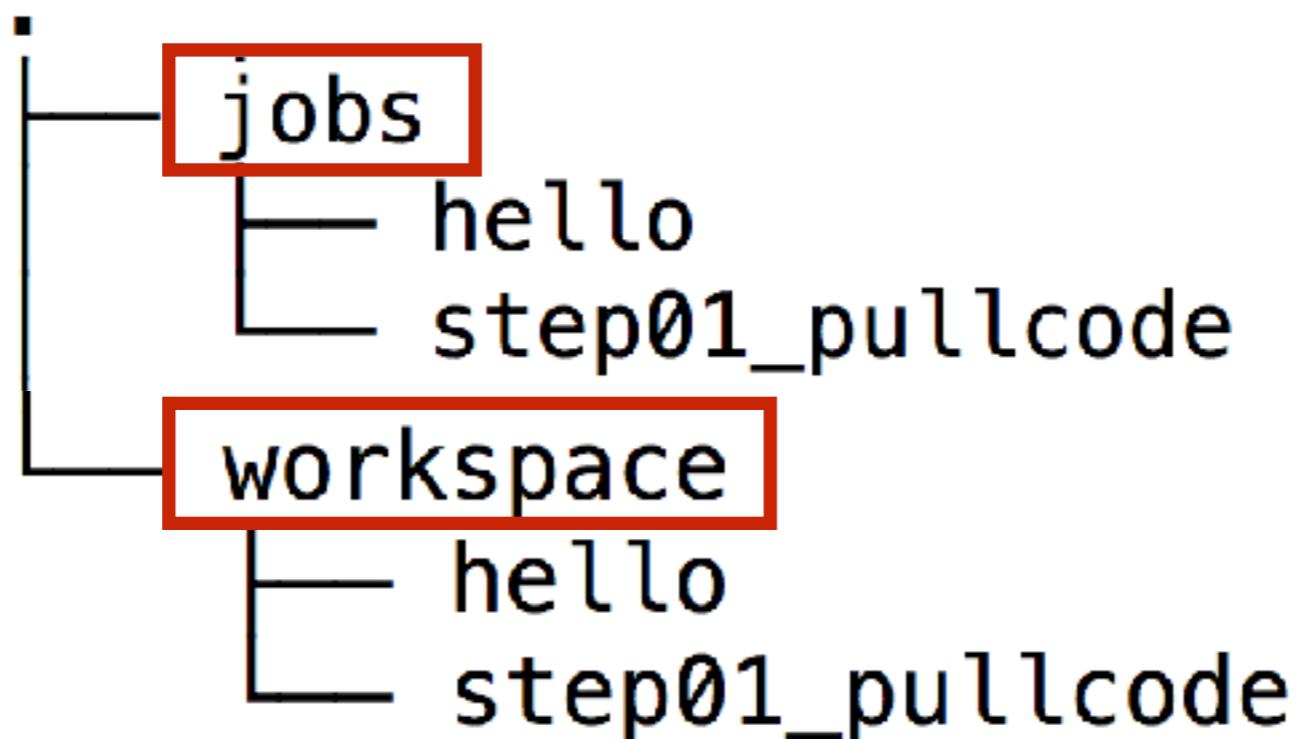


The screenshot shows the Jenkins interface for the project "step01\_pullcode". The top navigation bar includes the Jenkins logo and the current path: Jenkins > step01\_pullcode. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. 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 includes 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". A red box highlights the "Discard old builds" section, which includes a checkbox, a strategy dropdown set to "Log Rotation", and fields for "Days to keep builds" and "Max # of builds to keep".

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

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 search bar with the text "Filter: discard" highlighted by a red box. Below the search bar, there are tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". The main table lists available plugins:

Install ↓	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". A status message "Update information obtained: 1 day 11 hr ago" is also present.



# 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

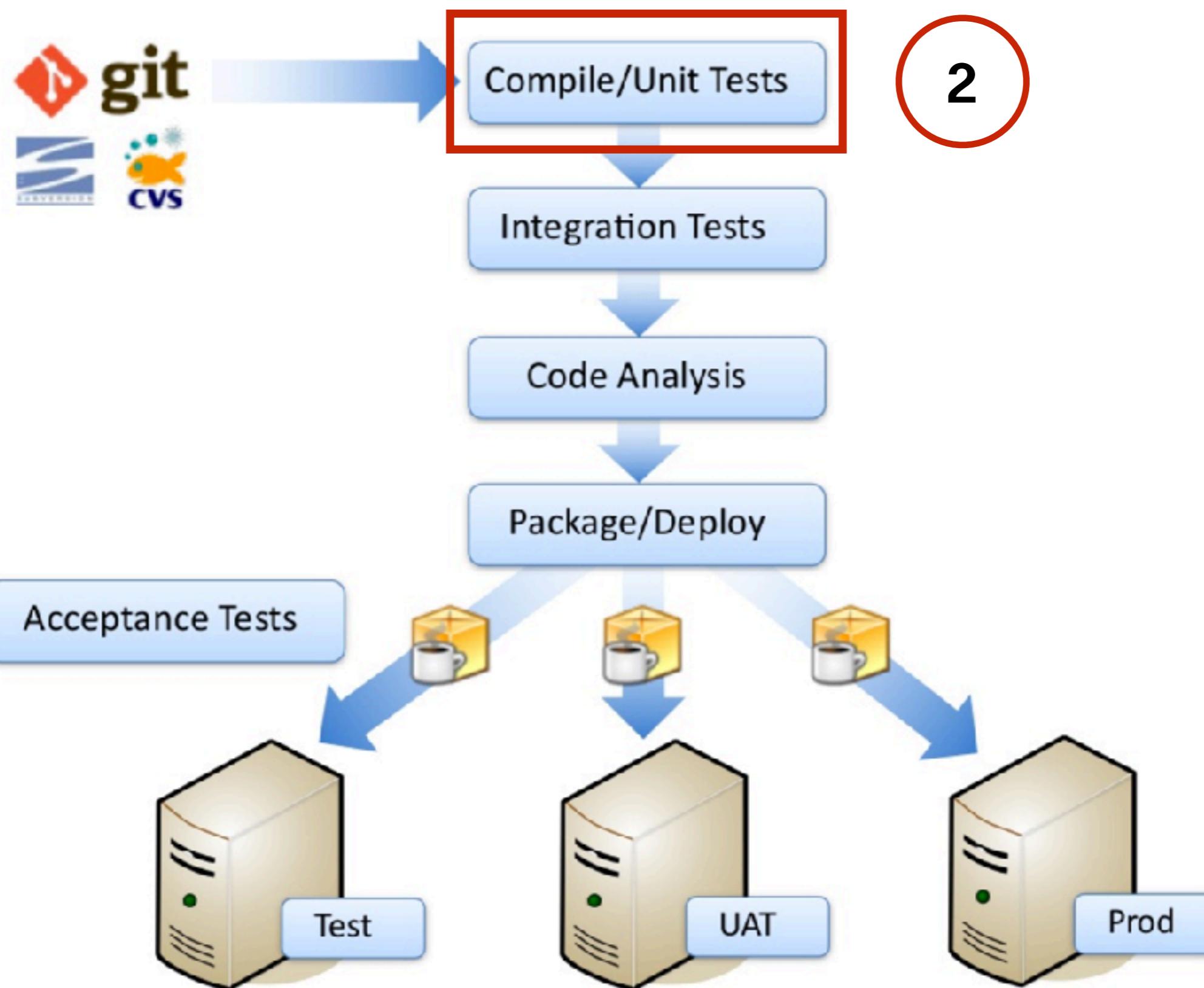
Use a crontab format

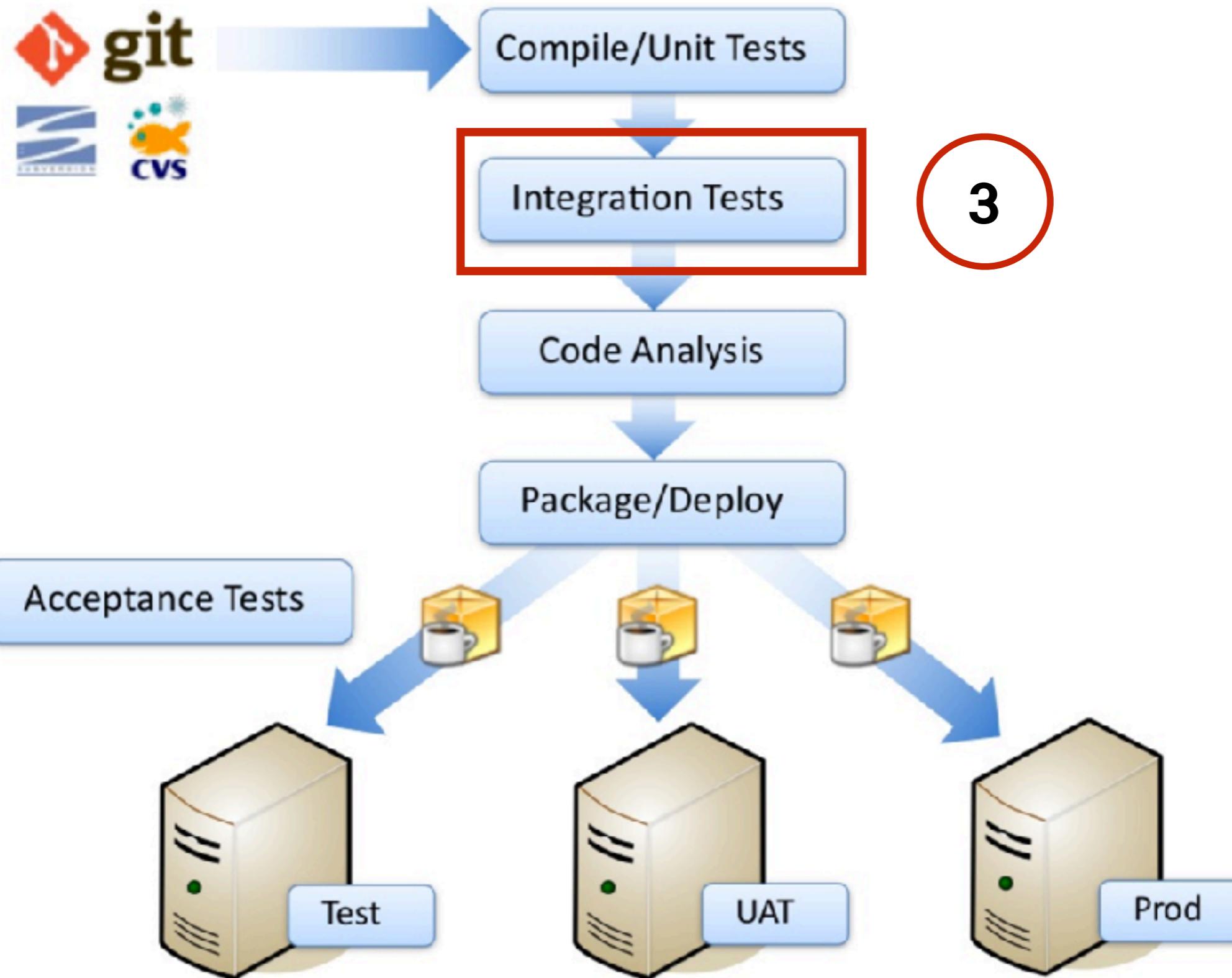
Name	Description
Minute	Minutes within the hour (0–59)
Hour	The hour of the day (0–23)
DOM	The day of the month (1–31)
Month	The month (1–12)
DOW	The day of the week (0–7) where 0 and 7 are Sunday.



# 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

The screenshot shows the Jenkins 'Build' configuration screen. A red circle labeled '1' highlights the 'Execute shell' option in the 'Add build step' dropdown menu. The 'Execute shell' step is currently selected, as indicated by the blue background. A red circle labeled '2' highlights the 'mvn clean package' command entered in the 'Command' field of the 'Execute shell' step configuration panel.

**Build**

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"

**Build**

Execute shell

Command `mvn clean package`

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

Advanced...

Add build step ▾



# Try to run and see output

Scroll down to button and see Success status

Jenkins > step01\_pulicode > #2

[Back to Project](#) [Status](#) [Changes](#) [Console Output](#) [View as plain text](#) [Edit Build Information](#) [Git Build Data](#) [No Tags](#) [Previous Build](#)

**Console Output** Progress:  %

```
Started by user Somkiat Puisungnien
Building on master in workspace /Users/somkiat/downloads/set/workspace/step01_pulicode
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/upl/workshop-java-web-tdd.git # timeout=10
Fetching upstream changes from https://github.com/upl/workshop-java-web-tdd.git
> git --version # timeout=10
> git fetch --tags --progress https://github.com/upl/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 7807e5a4628bf77301ad9268c97631b4f48b10d8 (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: 2018-02-06T11:40:25+07:00
[INFO] Final Memory: 29M/173M
[INFO] -----
Finished: SUCCESS
```



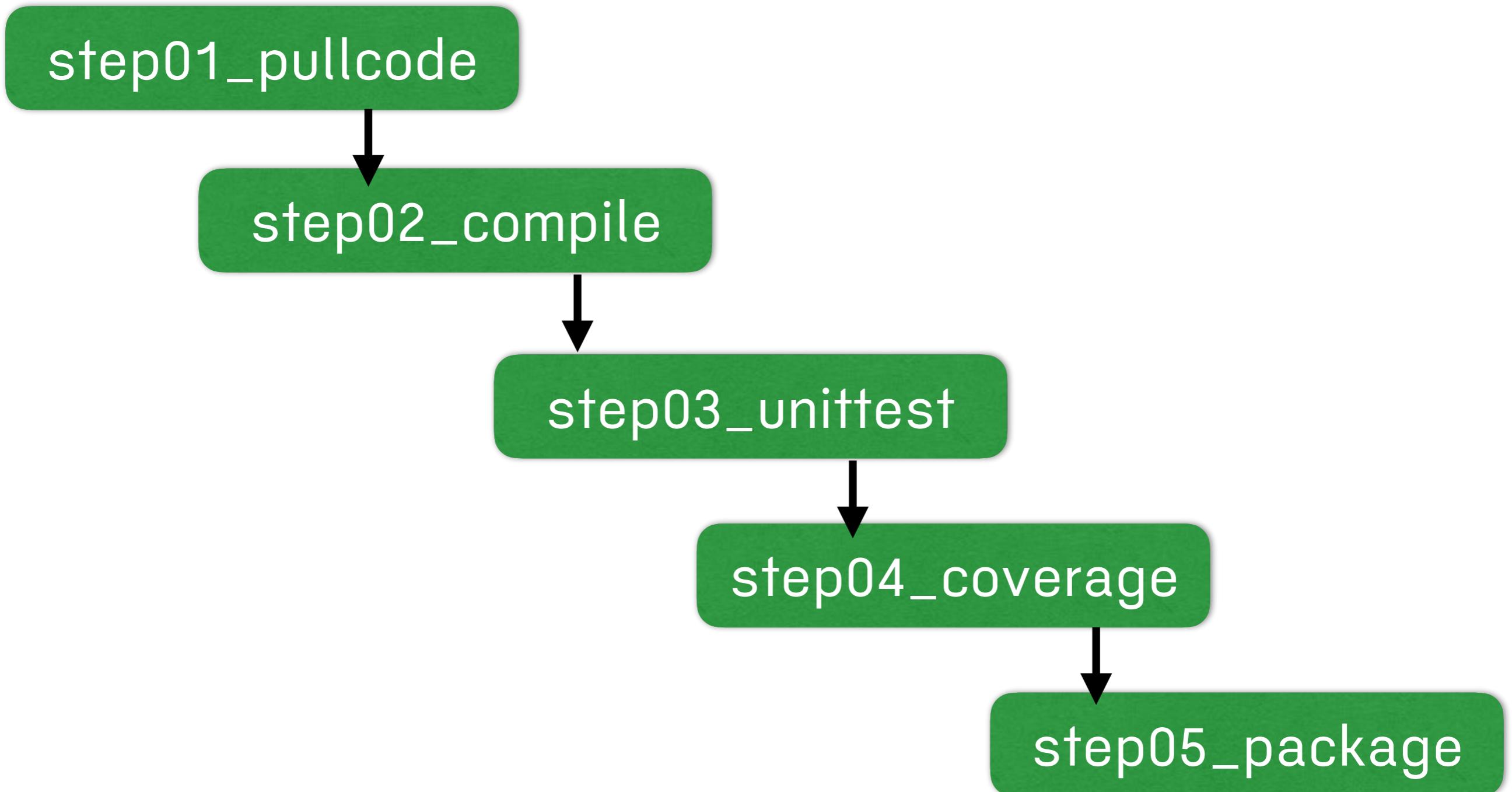
# Consideration for output

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  
step02\_build » 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.



# 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

The screenshot shows the Jenkins General configuration page. The 'General' tab is selected. Under the 'Build' section, there is a checkbox labeled 'Use custom workspace' which is checked. Below it, the 'Directory' field contains the value `\${JENKINS\_HOME}/workspace/step01\_pullcode`. A red error message 'Custom workspace is empty.' is displayed below the directory field. There are also other configuration options like 'Quiet period', 'Retry Count', 'Block build when upstream project is building', 'Block build when downstream project is building', and '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

Workspace of step02_build on master	
<a href="#">.git</a>	
<a href="#">data</a>	
<a href="#">schema</a>	
<a href="#">src</a>	
<a href="#">target</a>	
<a href="#">.gitignore</a>	39 B <a href="#">view</a>
<a href="#">database.txt</a>	110 B <a href="#">view</a>
<a href="#">pom.xml</a>	3.33 KB <a href="#">view</a>
<a href="#">README.md</a>	155 B <a href="#">view</a>
<a href="#">sonar-project.properties</a>	268 B <a href="#">view</a>
	<a href="#">(all files in zip)</a>



# 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: 10

Recent Changes

Latest Test Result (no failures)

Build History

trend =

Build #	Timestamp
#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](#)

(just show failures) [enlarge](#)



# 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

- 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 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](#) [add description](#) [Disable Project](#)

## Project step03\_coverage

[!\[\]\(d3374425e5e9d42e4c9a094ac97d443f\_img.jpg\) Coverage Report](#) [!\[\]\(a08a02f52bc3dacf612ba70b08c17fb5\_img.jpg\) Workspace](#) [!\[\]\(c094e49f3f4e92fe5d56c68ddf924436\_img.jpg\) Recent Changes](#)

**Code Coverage**

Packages	50%
Files	43%
Classes	38%
Methods	29%
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](#)

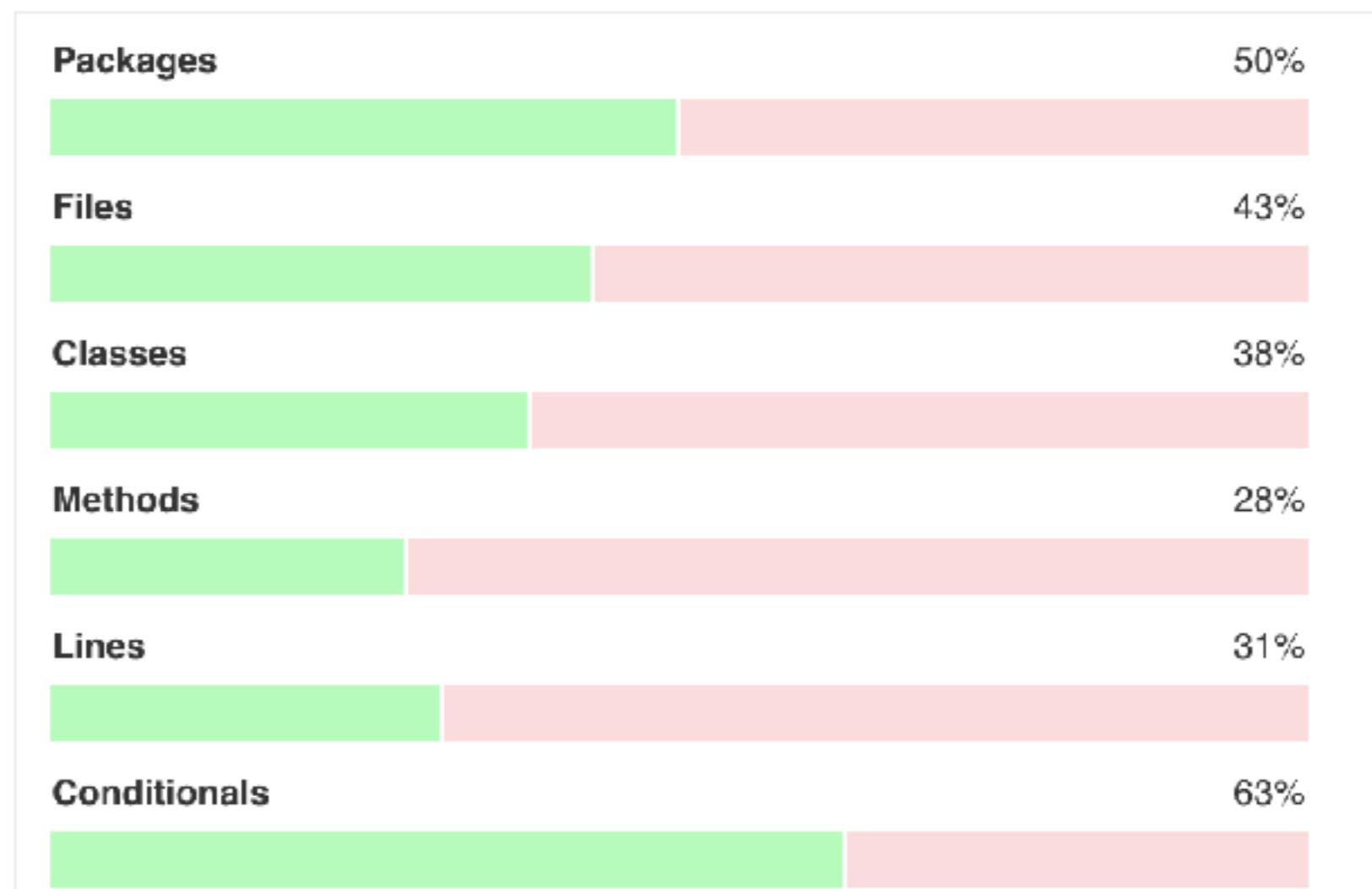
Build History trend →  
find  
#2 Feb 6, 2018 4:36 PM  
#1 Feb 6, 2018 4:35 PM  
[RSS for all](#) [RSS for failures](#)



# 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



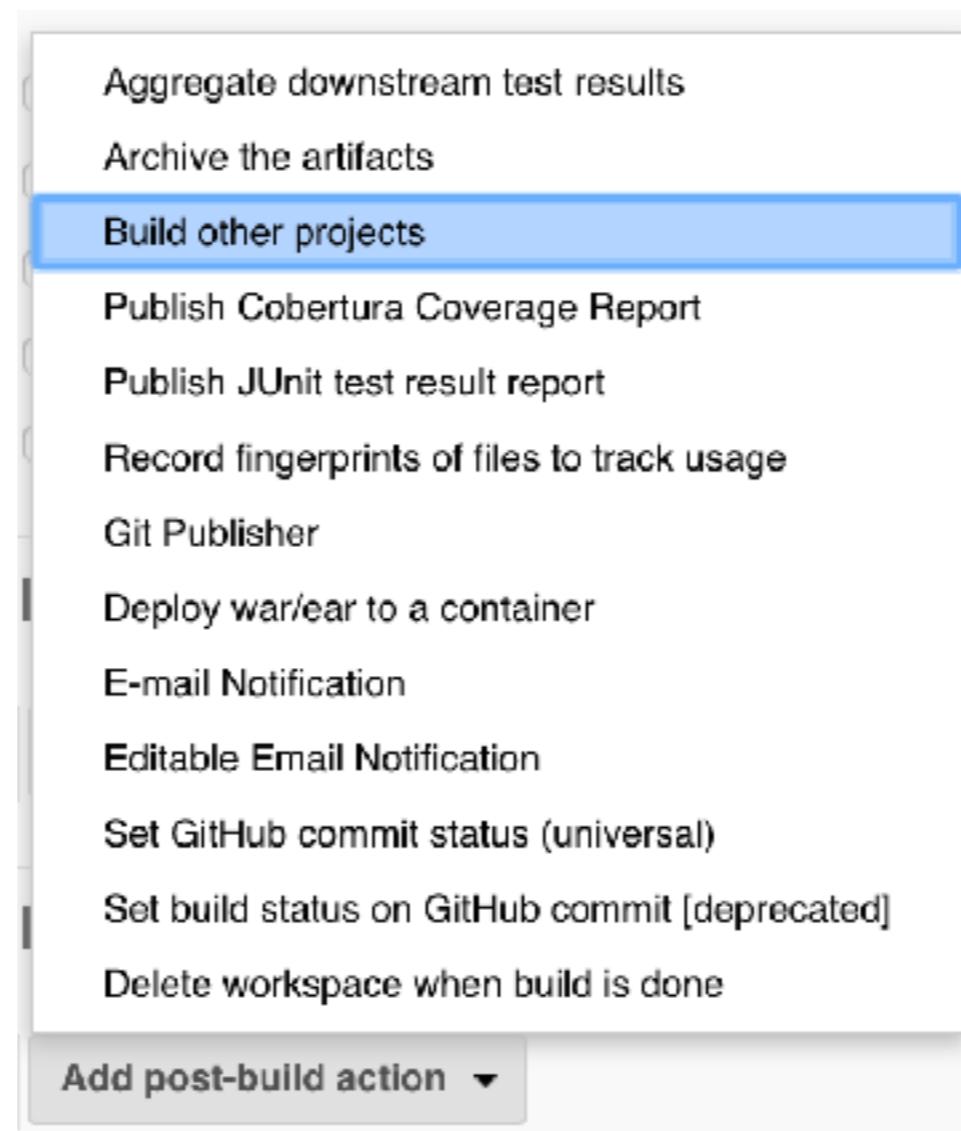
# One more thing !!



# 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**

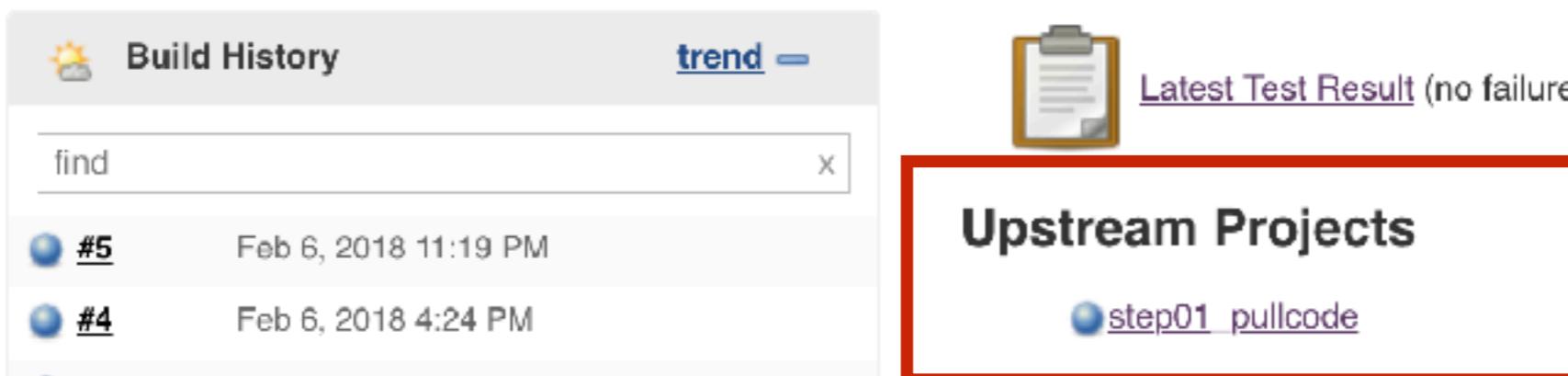
[Workspace](#)

[Recent Changes](#)

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

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



# 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   
Use the default build

---

Trigger Options

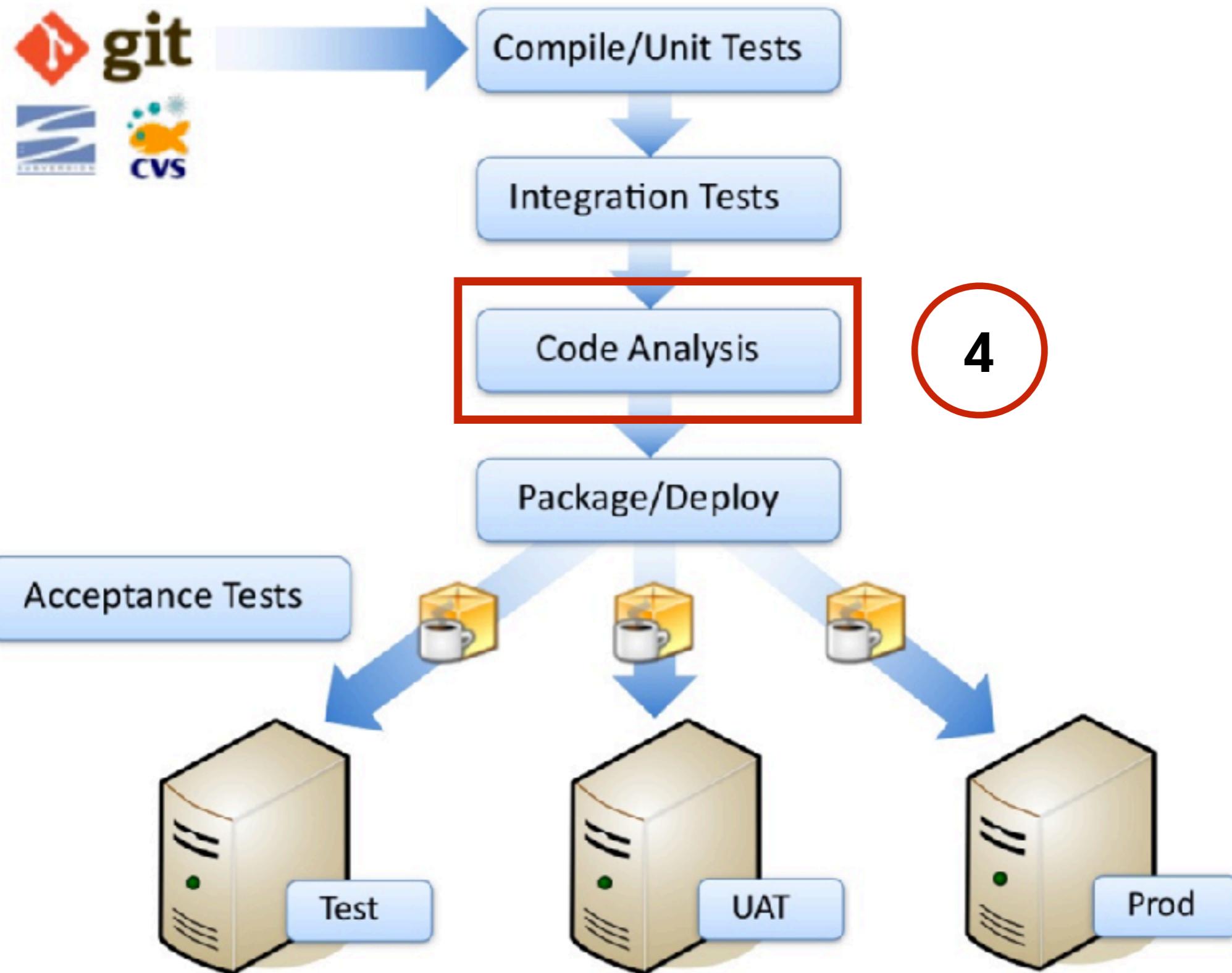
Build Cards



# Build pipeline in view

The screenshot shows the Jenkins Build Pipeline interface. At the top, there is a navigation bar with the Jenkins logo, a search bar, and user information for 'Somkiat Pulsungnoen'. Below the navigation bar, the title 'Build Pipeline' is centered. Underneath the title are six icons with labels: Run, History, Configure, Add Step, Delete, and Manage. The main area displays two green rectangular boxes representing build steps. The first box is labeled '#3 step01\_pullcode' and contains the following details: 'Feb 6, 2018 11:19:18 PM', '1.6 sec', and 'somkiat'. The second box is labeled '#5 step02\_build' and contains the following details: 'Feb 6, 2018 11:19:27 PM', '10 sec', and 'somkiat'. A green arrow points from the first box to the second box, indicating a sequential flow between the steps.





# 4. Code analysis

Working with SonarQube

Start SonarQube server



# Step 1 Install SonarQube plugin

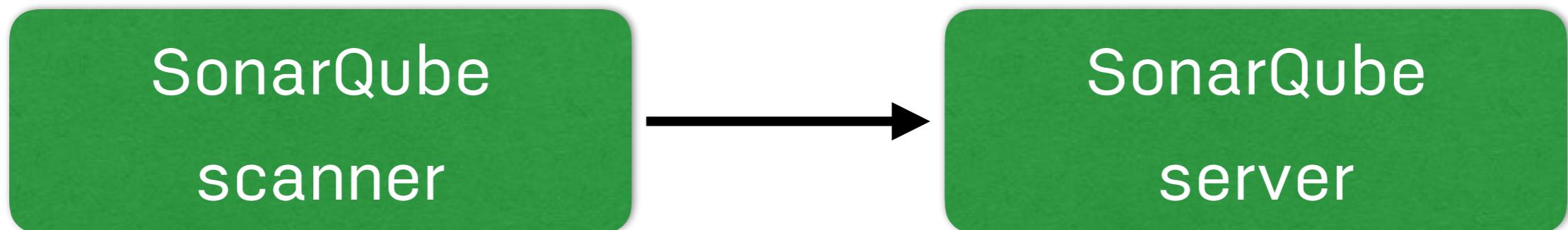
The screenshot shows the Jenkins Manage Plugins interface. A red box highlights the search bar at the top right containing the text "sonarqube". Below the search bar, there are tabs: Updates, Available (which is selected), Installed, and Advanced. Under the Available tab, a table lists available plugins. The first plugin listed is "SonarQube Scanner for Jenkins" (version 2.6.1), which has a checked checkbox next to it. The second plugin listed is "Mashup Portlets" (version 1.0.8), which has an unchecked checkbox next to it. A detailed description for the Mashup Portlets plugin follows its name, explaining its features: "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 of the page are three buttons: "Install without restart", "Download now and install after restart" (which is highlighted in blue), and "Update information obtain".

Install ↓	Name	Version
<input checked="" type="checkbox"/>	<a href="#">SonarQube Scanner for Jenkins</a>	2.6.1
<input type="checkbox"/>	<a href="#">Mashup Portlets</a> 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).	1.0.8



# Step 2 Install SonarQube Scanner

Send analyzed result to SonarQube server



Run on Jenkins's Job



# Step 3 Install Sonar Scanner

The screenshot shows a documentation page for the SonarQube Scanner. At the top, there's a navigation bar with 'Scanners / Analyzing Source Code / Analyzing with SonarQube Scanner' and a '...' button. Below the title 'Analyzing with SonarQube Scanner' is a note that it was created by OLD - Evgeny Mandrikov and last modified by Julien Henry on May 12, 2017. A note below states the package is compatible with SonarQube 5.6+ (LTS) and includes links for Linux 64 bit, Windows 64 bit, Mac OS X 64 bit, and Any\*. A note at the bottom says the package expects a JVM installed. On the left, there's a 'Table of Contents' sidebar with links to Features, Installation, Use, Troubleshooting, and Going Further.

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)

[Linux 64 bit](#)   [Windows 64 bit](#)   [Mac OS X 64 bit](#)   [Any\\*](#)

\*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: sonarqube Server URL: <input type="text"/> Default is http://localhost:9000 Server version: 5.3 or higher Configuration fields depend on the SonarQube server version. Server authentication token: <input type="text"/> SonarQube authentication token. Mandatory when anonymous access is disabled. SonarCube account login: <input type="text"/> SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3. SonarCube 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

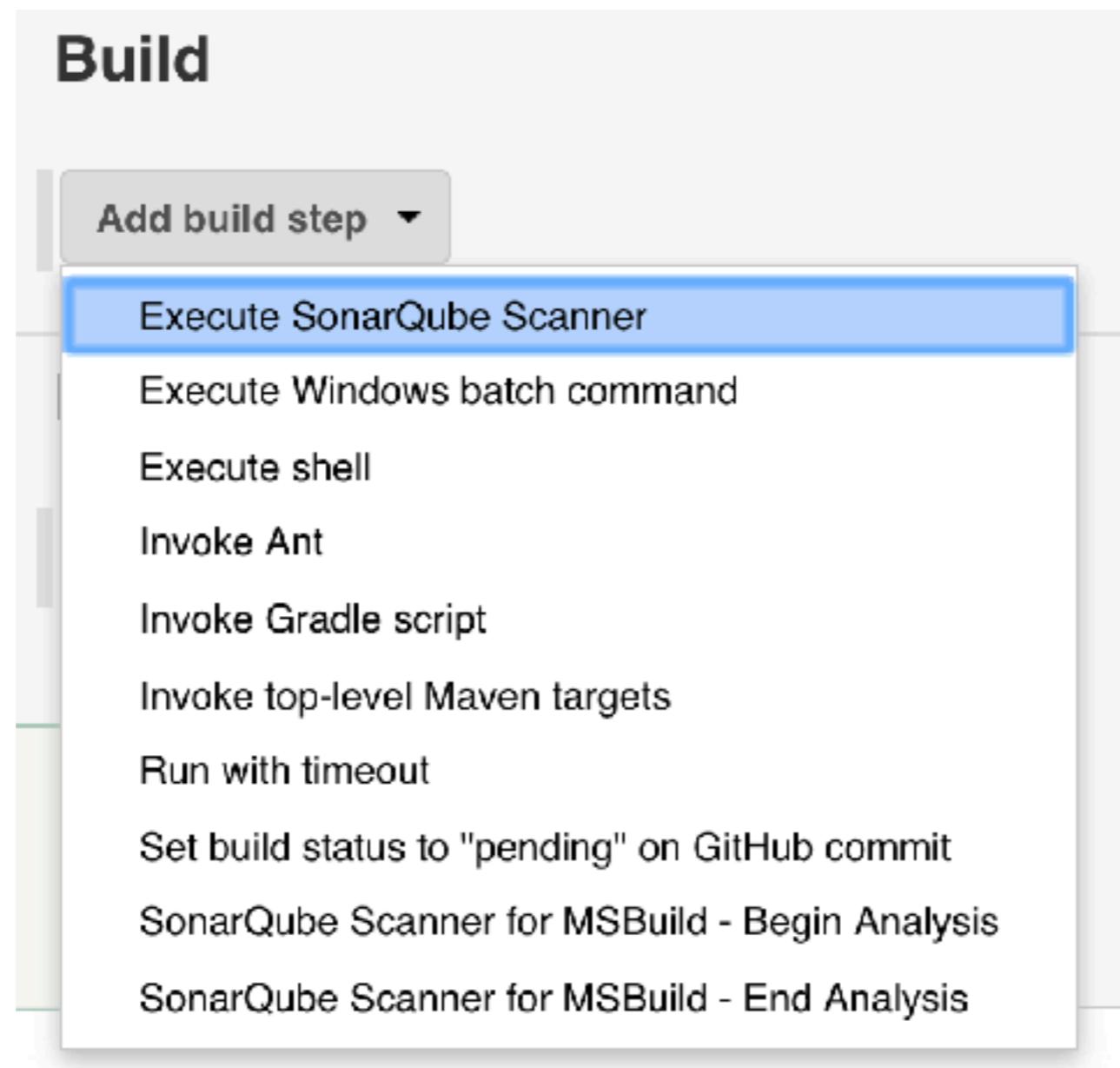
step02\_build

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  
[SonarQube](#)

Workspace  
[Workspace](#)

Recent Changes  
[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

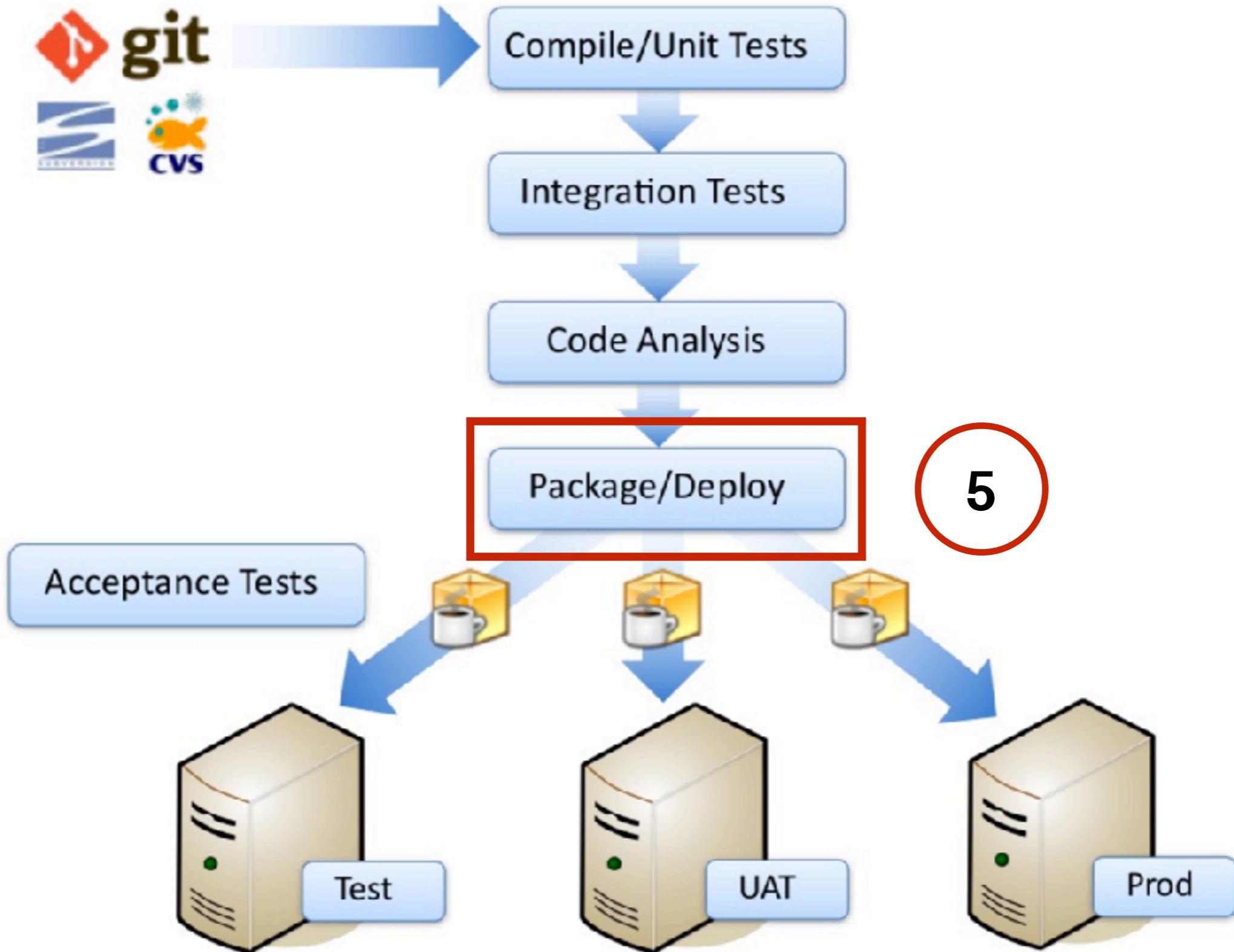
Result in Sonarqube server

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



# Try it by yourself





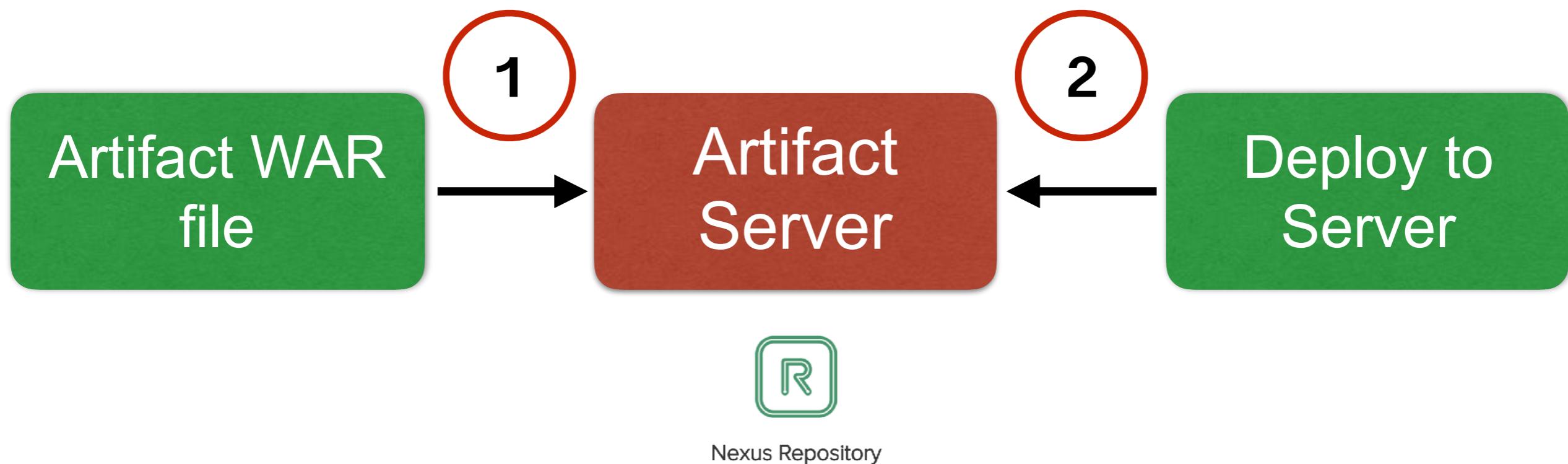
# 5. Package and Deploy



# 5.1 Package and Deploy

Try to build WAR file and send to Nexus Repos

Try to deploy WAR file to WebServer



# Setup Nexus Repository



# Step 0 Start server

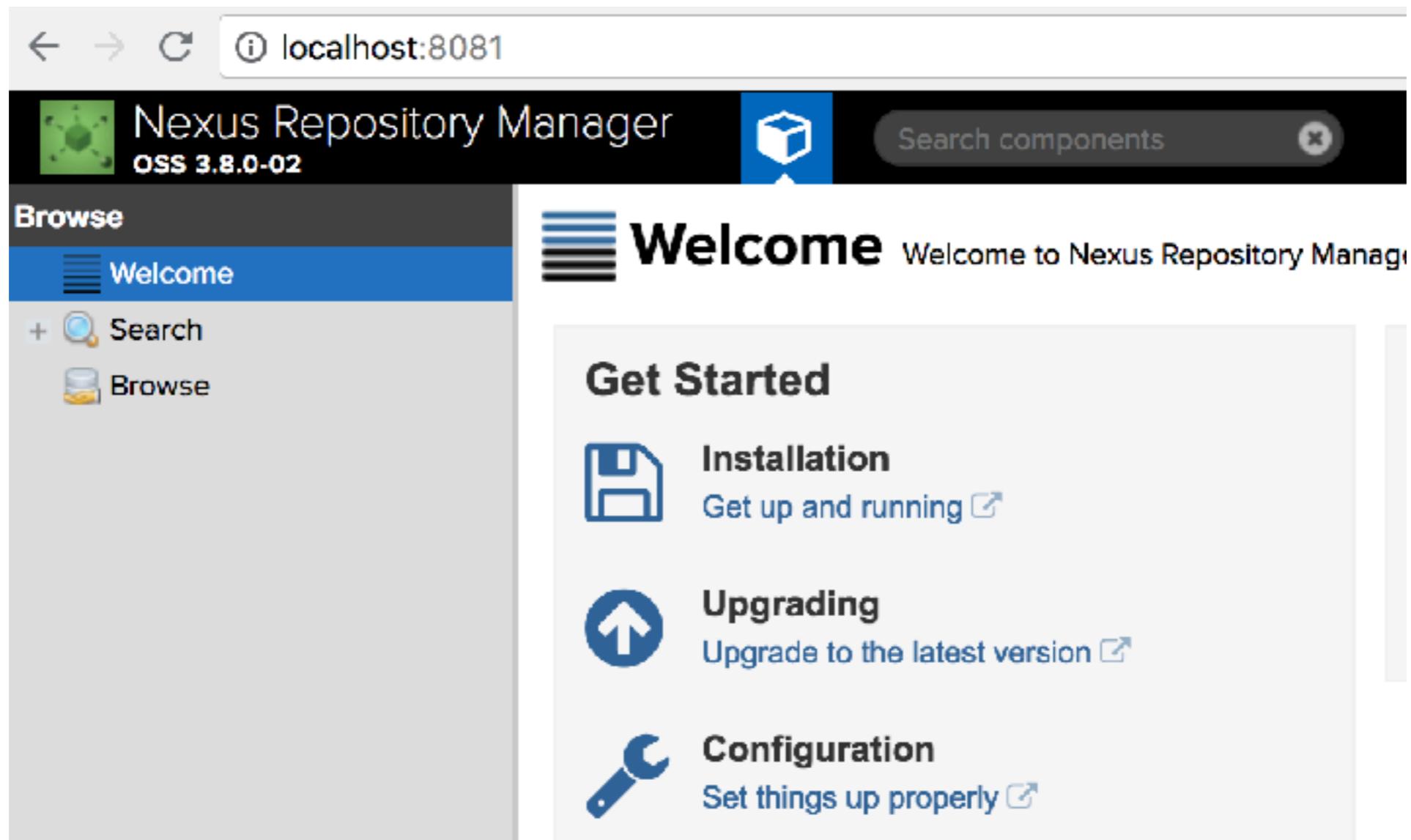
```
$nexus /run #For Windows
```

```
$nexus run #For Linux/Mac
```

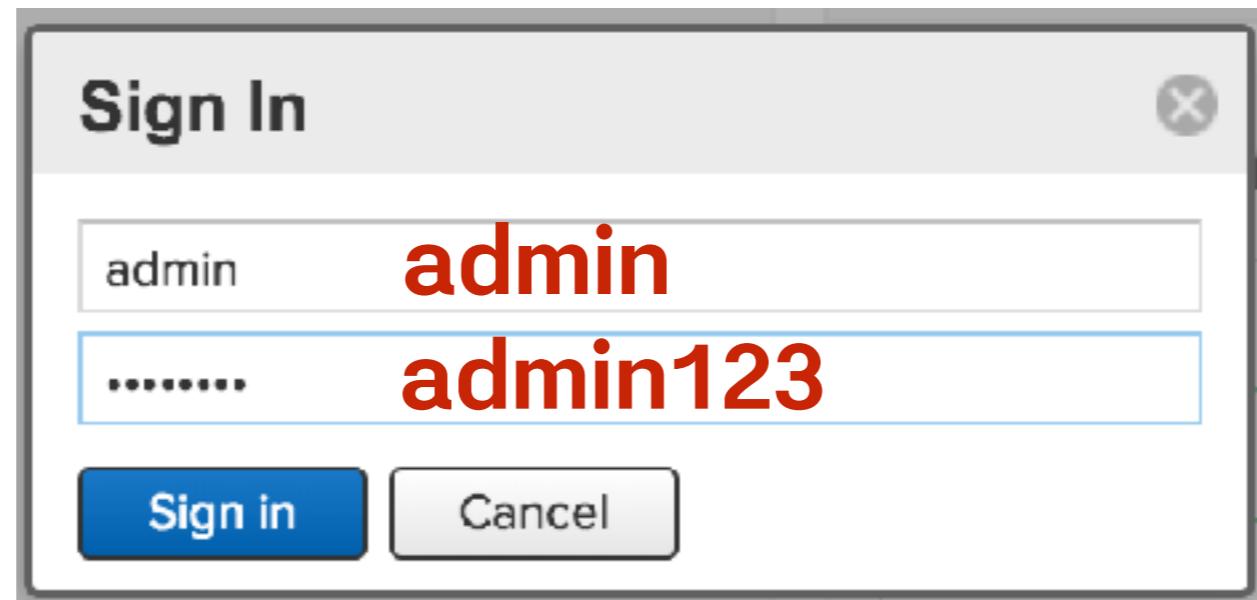
<https://help.sonatype.com/repomanager3/installation/installation-methods>



# Go to <server>:8081



# 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 is titled "Administration" and 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 database icon), and "Content Selectors" (represented by a circular icon).



# Step 3 Select repositories menu

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", and a search bar labeled "Search components". The left sidebar, titled "Administration", contains the following menu 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" and "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", and a search bar labeled "Search components". 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 plus sign icon, which is highlighted with a red box. Below this, there is 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 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

The screenshot shows the JBoss Seam Admin interface under the 'Administration' section. The left sidebar lists various administrative categories: 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, and System API. The 'Repositories' item is highlighted with a blue background.

The main content area is titled 'Repositories' and shows a sub-path: 'Select Recipe' / 'Create Repository: pypi (hosted)'. A red box highlights the 'Name' field, which contains the value 'demo\_web'. Below it is the 'Online' checkbox, which is checked and accompanied by the text: 'If checked, the repository accepts incoming requests'.

Below this section is a 'Storage' configuration block. It includes a 'Blob store:' dropdown set to 'default' and a 'Strict Content Type Validation:' checkbox, which is checked.

Further down is a 'Hosted' configuration block. It includes a 'Deployment policy:' dropdown set to 'Disable redeploy'.

At the bottom of the form are two buttons: 'Create repository' (in blue) and '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" with the placeholder "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

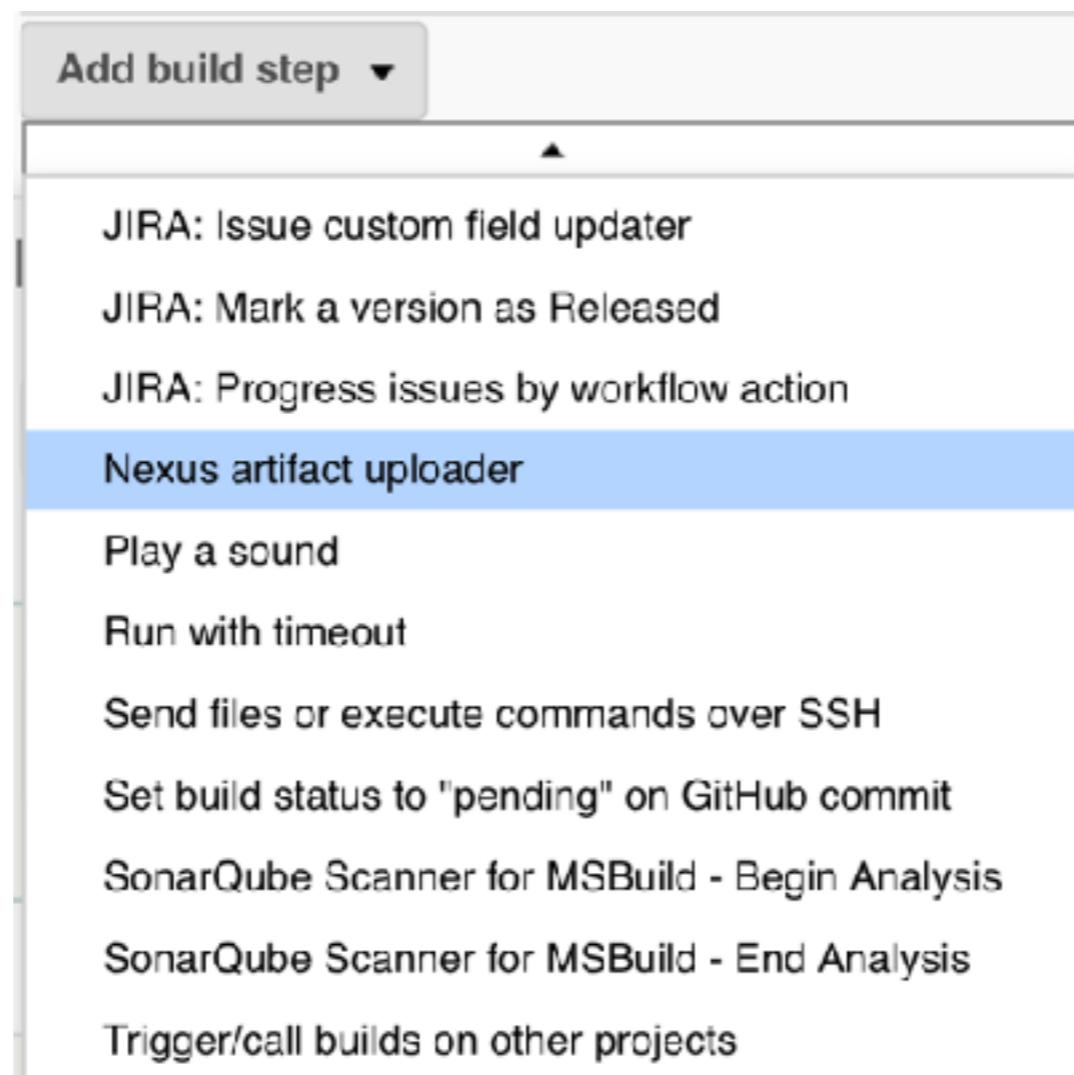
[Install without restart](#) [Download now and install after restart](#) Update information obtained: 1 ↗



# Config Nexus Artifact Uploader

Use to upload artifact file to Nexus Repository

Add build step => Nexus artifact uploader



# See result in Nexus Artifactory

The screenshot shows the Nexus Repository Manager interface. The top navigation bar includes the logo, "Nexus Repository Manager OSS 3.8.0-02", a search bar labeled "Search components", and a gear icon for settings. The left sidebar, titled "Browse", has links for Welcome, Search (which is expanded), Custom, Maven, NuGet, and Raw. The "Raw" link is currently selected and highlighted in blue. The main content area is titled "Raw" and "Search for components in Raw repositories". It features a search bar with "Name" and "Any" placeholder, and a "More criteria" button. Below is a table with columns "Name ↑" and "Group". The table lists various WAR files and their corresponding group paths:

	Name ↑	Group
1	war/demo/10/demo-10.war	/war/demo/10
2	war/demo/10/demo-10.war.md5	/war/demo/10
3	war/demo/10/demo-10.war.sha1	/war/demo/10
4	war/demo/11/demo-11.war	/war/demo/11
5	war/demo/11/demo-11.war.md5	/war/demo/11
6	war/demo/11/demo-11.war.sha1	/war/demo/11
7	war/demo/6/demo-6.war	/war/demo/6
8	war/demo/6/demo-6.war.md5	/war/demo/6
9	war/demo/6/demo-6.war.sha1	/war/demo/6
10	war/demo/7/demo-7.war	/war/demo/7
11	war/demo/7/demo-7.war.md5	/war/demo/7
12	war/demo/7/demo-7.war.sha1	/war/demo/7
13	war/demo/8/demo-8.war	/war/demo/8
14	war/demo/8/demo-8.war.md5	/war/demo/8



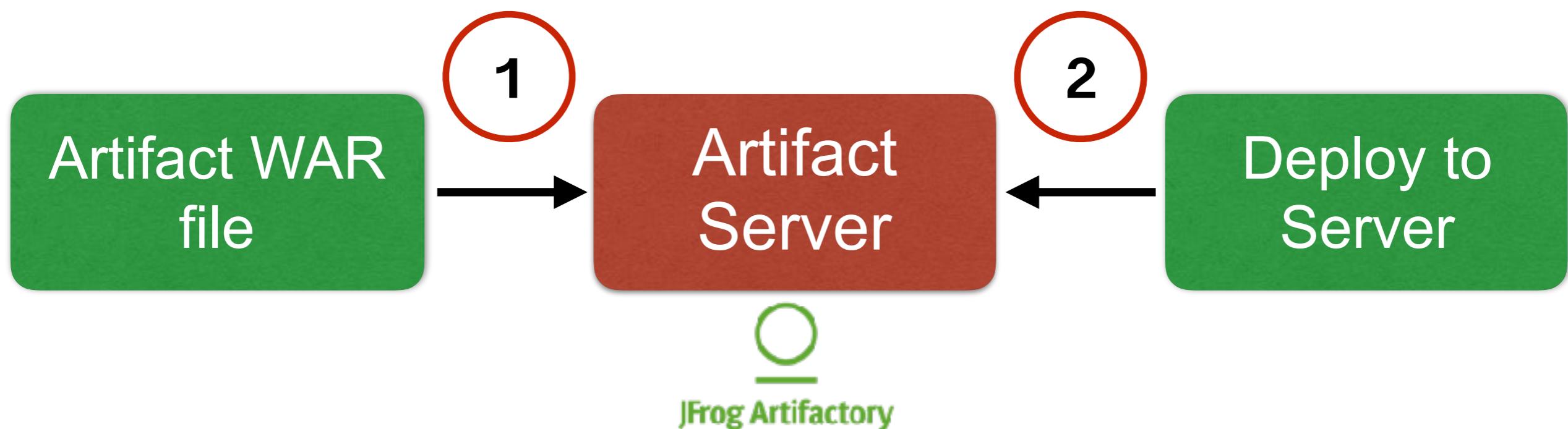
# Let's try by yourself



# 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! ×

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, Groups, Artifactory, and Help. The main area is titled "Local Repositories" and shows a table with one repository listed:

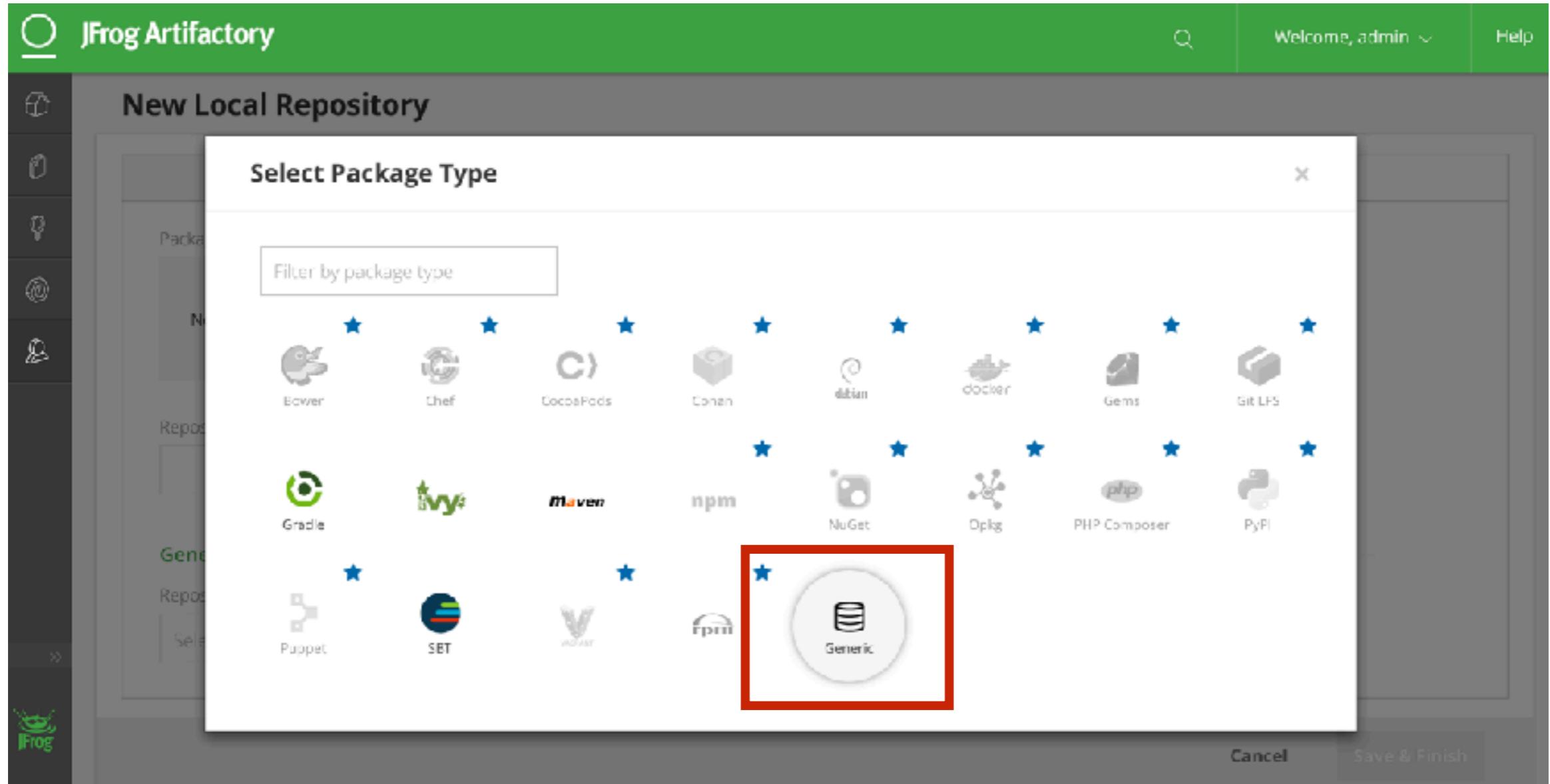
Repository Key	Type	Recycle Bin
example-repo-local	Generic	

A context menu is open on the right side of the screen, titled "Create Repositories". It includes the following options:

- Quick Setup
- Local Repository** (highlighted with a red box)
- Remote Repository
- Virtual Repository
- Distribution Repository
- Add User
- Add Group
- Add Permission
- Edit Profile
- Log Out



# Step 3 Choose Generic repos



# 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 'General' section, the 'Repository Key' field is filled with 'dev' and has a red border around it. At the bottom right, the 'Save & Finish' button is also highlighted with a red box.



# Step 4 Install Artifactory plugin

The screenshot shows the Jenkins plugin manager interface. At the top, there is a search bar with the placeholder text "Filter: artifactory". Below the search bar, there are four tabs: "Updates", "Available" (which is selected), "Installed", and "Advanced". The main table has three columns: "Install ↓", "Name", and "Version". A single row is visible in the table, representing the "Artifactory Plugin". The plugin name is "Artifactory Plugin", the version is "2.11.0", and its description is "Integrates Artifactory to Jenkins". To the left of the plugin name is a checkbox which is checked. At the bottom of the table, there are three buttons: "Install without restart", "Download now and install after restart" (which is highlighted in blue), and "Update information obtained".

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

The screenshot shows the Jenkins 'Configure System' page under the 'Artifactory' section. It displays a configuration for a single Artifactory server. The 'Enable Push to Bintray' checkbox is checked. The 'Use the Credentials Plugin' checkbox is unchecked. The 'Artifactory' server is defined with a 'Server ID' of 'artifactory' and a 'URL' of 'http://localhost:8081/artifactory'. Below this, there are fields for 'Default Deployer Credentials' with 'Username' and 'Password' inputs. A 'Test Connection' button is highlighted with a blue border. Other buttons visible include 'Advanced...', 'Delete', and 'Add Artifactory Server'. At the bottom, there is a note about proxying Docker images and buttons for 'Save' and 'Apply'.

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

Use Different Resolver Credentials

**Test Connection**

Delete

Add Artifactory Server

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

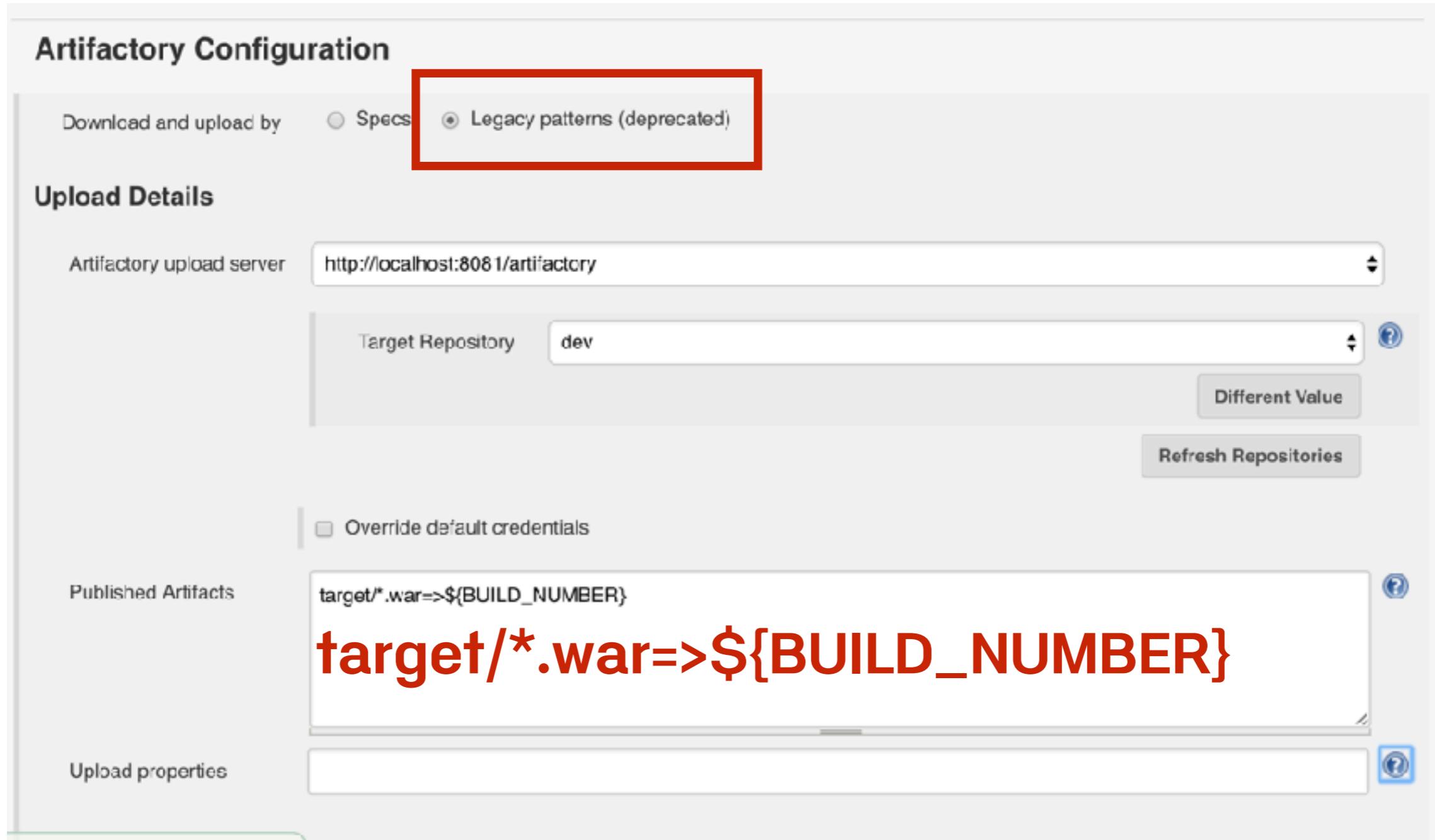
Target Repository `dev` ? Different Value Refresh Repositories

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, Artifacts, Pipelines, and Examples. The main area is titled "Artifact Repository Browser". It has a navigation bar with "Tree", "Simple", and "Search" tabs, and a checkbox icon. The left pane displays a tree view of artifacts under the "dev" repository. A red box highlights the "dev" node, which contains three sub-folders: "12", "13", and "14", each containing a "demo.war" file. Below these are collapsed nodes for "Example repo" and "local". The right pane shows detailed information for the "dev" repository. The "General" tab is selected, showing the following details:

Info	Value
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

Jenkins

Project 5\_create\_war\_file

Upstream Projects

2 compile unities:

Artifactory Build Info

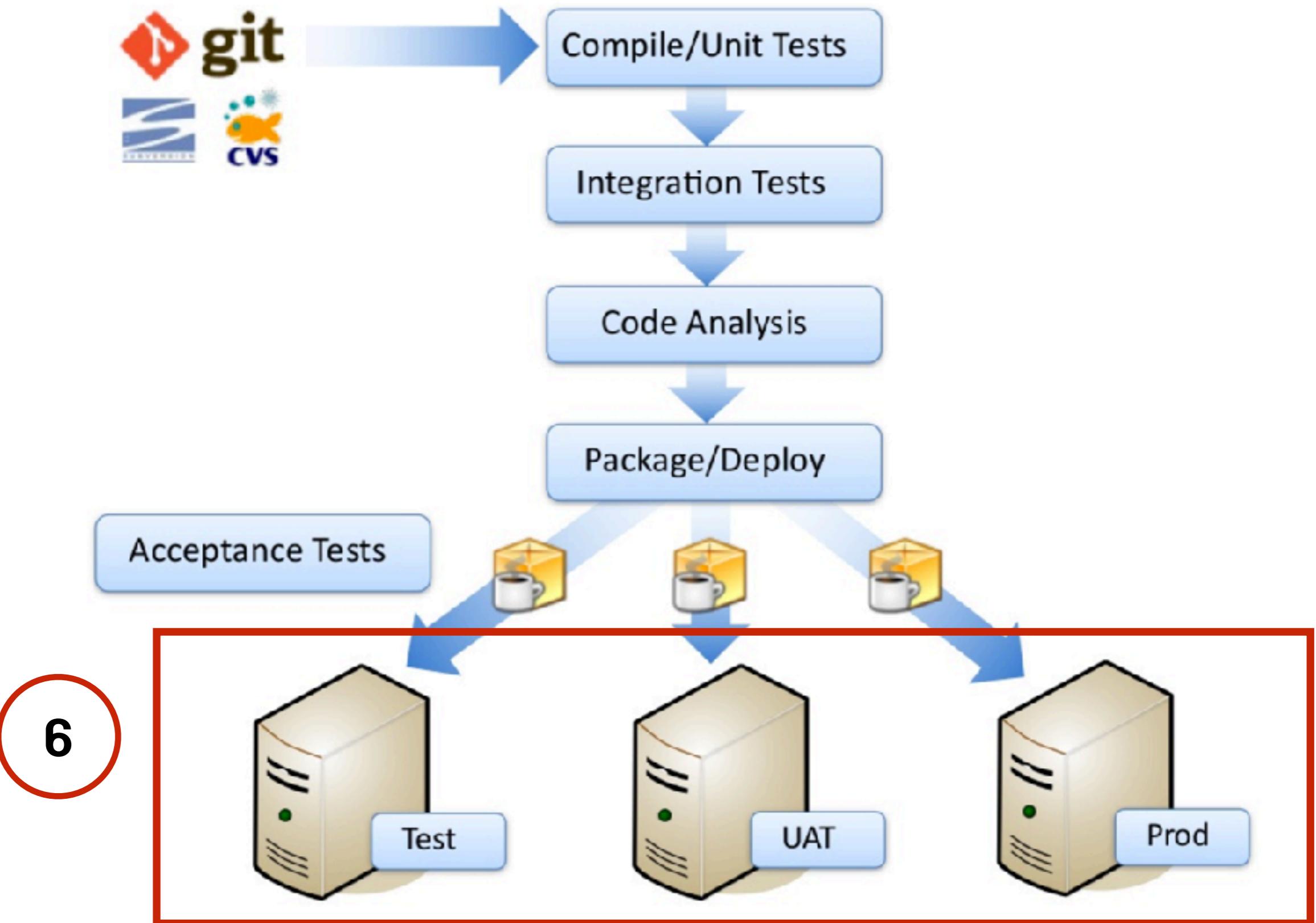
Workspace

Recent Changes

Build History

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





# 6. Deploy to other server

Try to deploy WAR file to WebServer

**Apache Tomcat**

**JBoss**

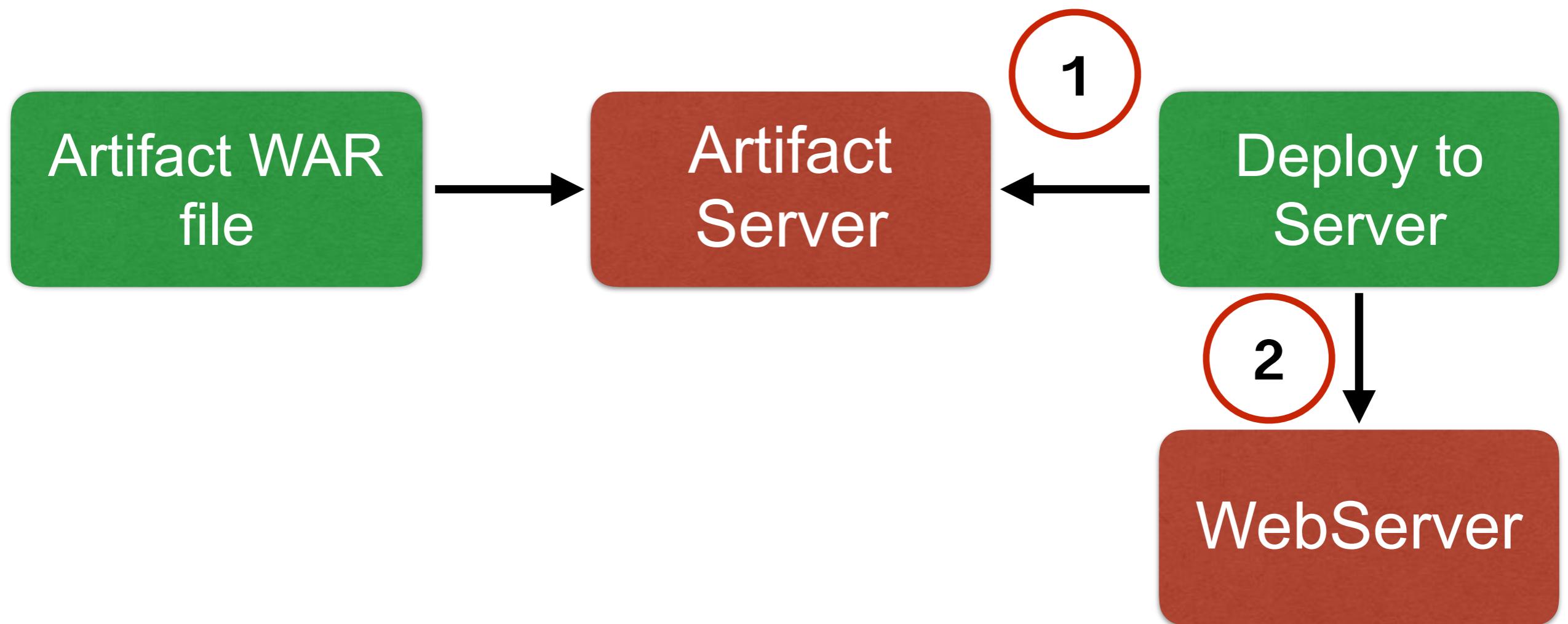
**IBM Websphere**

**Web Logic**



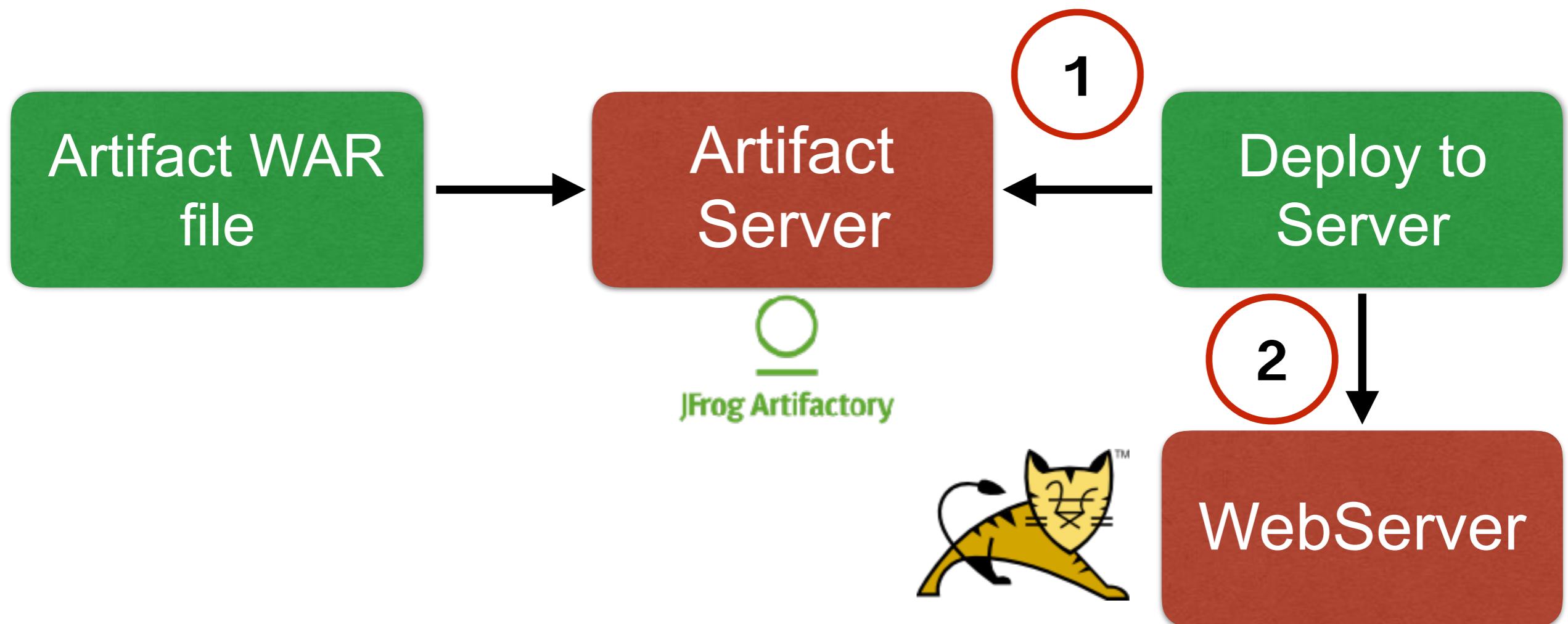
# 6. Deploy to other server

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="float: right;">?</span>
Context path	<input type="text"/> <span style="float: right;">?</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 Repository



# Download with command line

Add step in build step of job

Build

Execute shell

X ?

Command

```
wget --output demo.war -q <url of JFrog artifactory server>/demo.war  
curl -X GET -o demo.war -u admin:admin123  
<url of artifact server>/demo.war
```

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

Advanced...

Add build step ▾



# Try to deploy WAR



# 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

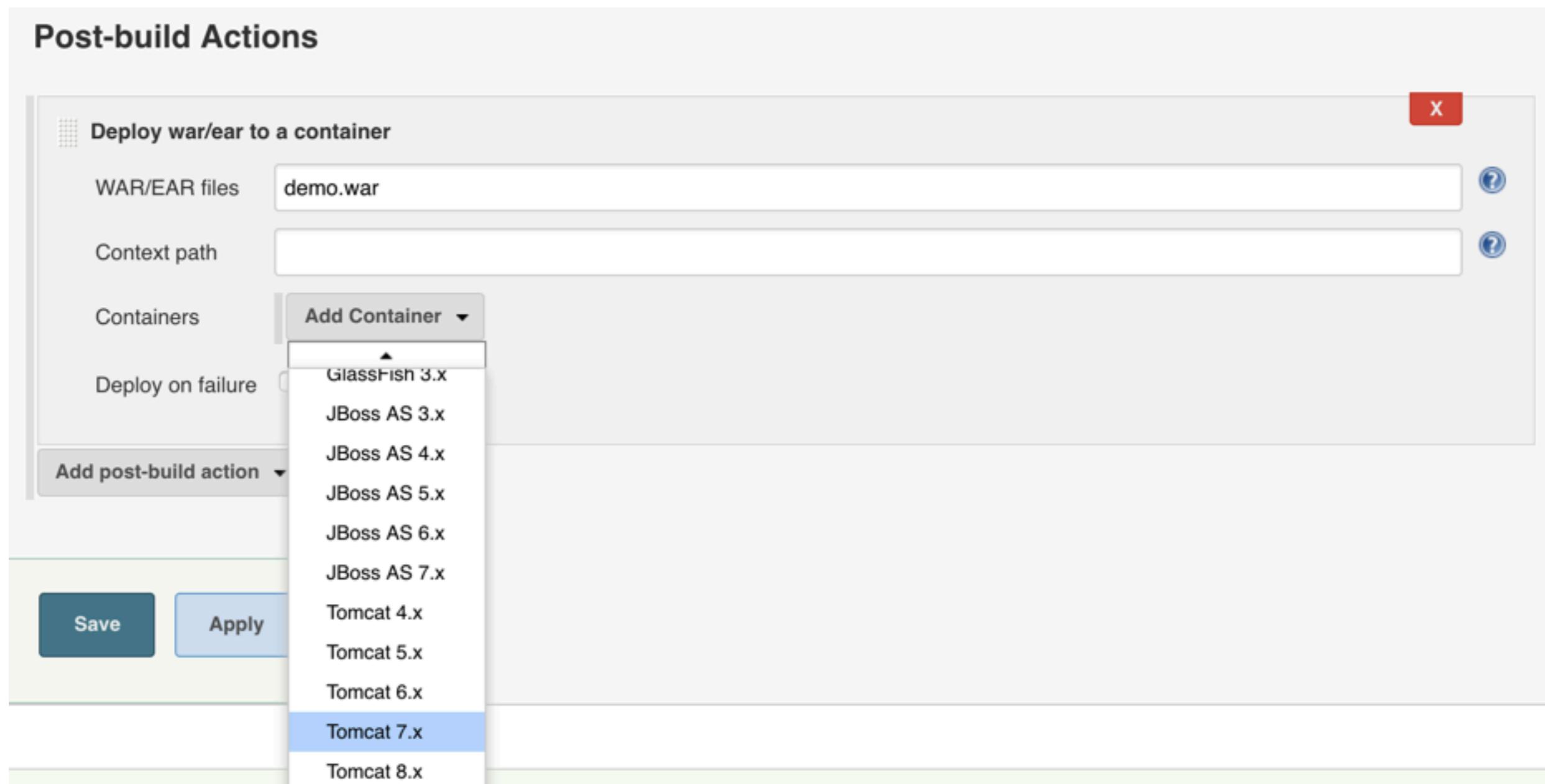
Context path:

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  ?

Context path  ?

Containers

**Tomcat 8.x**

Credentials

Tomcat URL

Deploy on failure



# Deploy fail

Need username and password ?



## Console Output

```
Started by user Somkiat Puisungcen
Building on master in workspace /Users/somkiat/Downloads/set/workspace/step08_deploy
[step08_deploy] $ /bin/sh -xe
/var/folders/t5/8kg23s_97z9dw44tfcl5dqw000gn/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

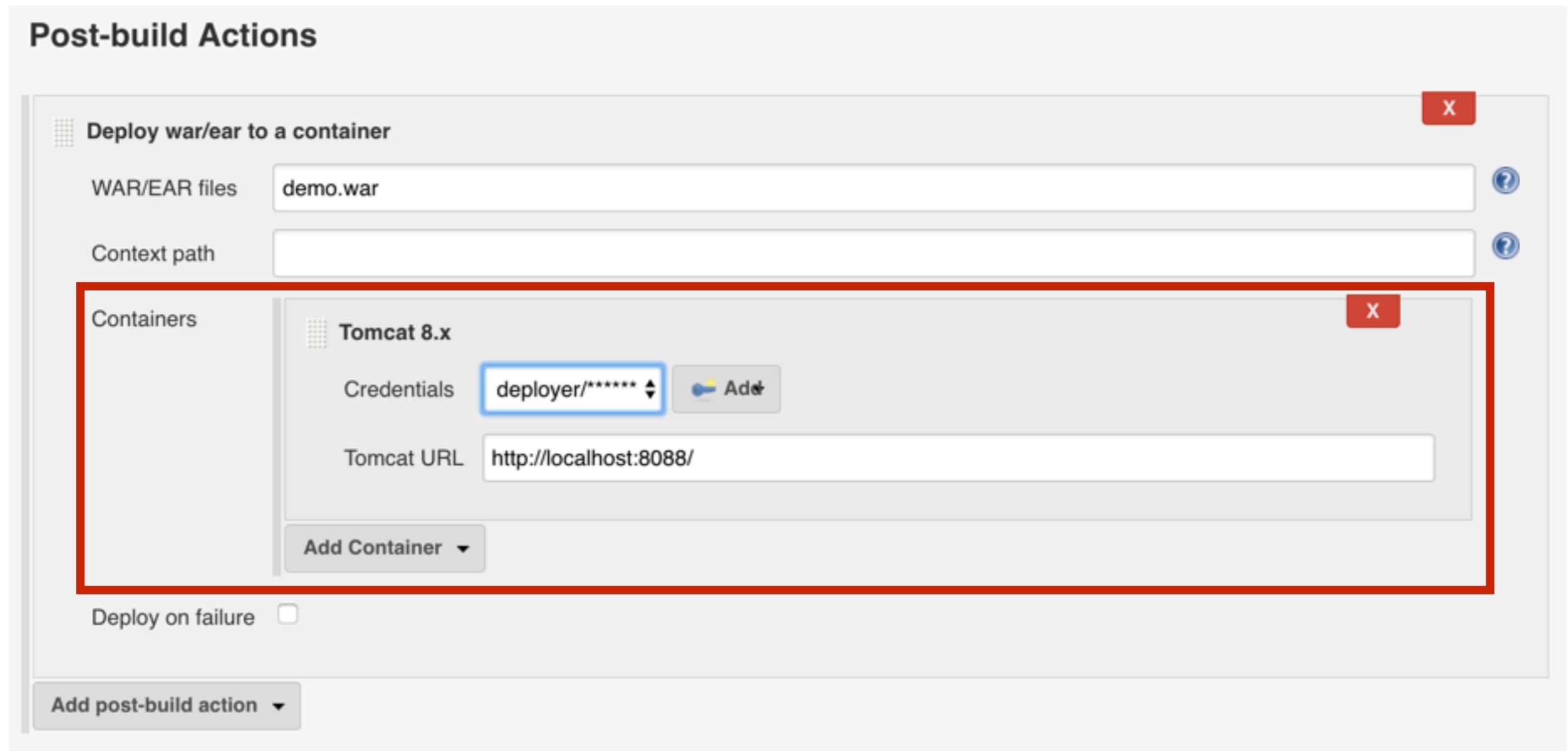
Credentials: **deployer/\*\*\*\*\*\*\*\*\***

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
```

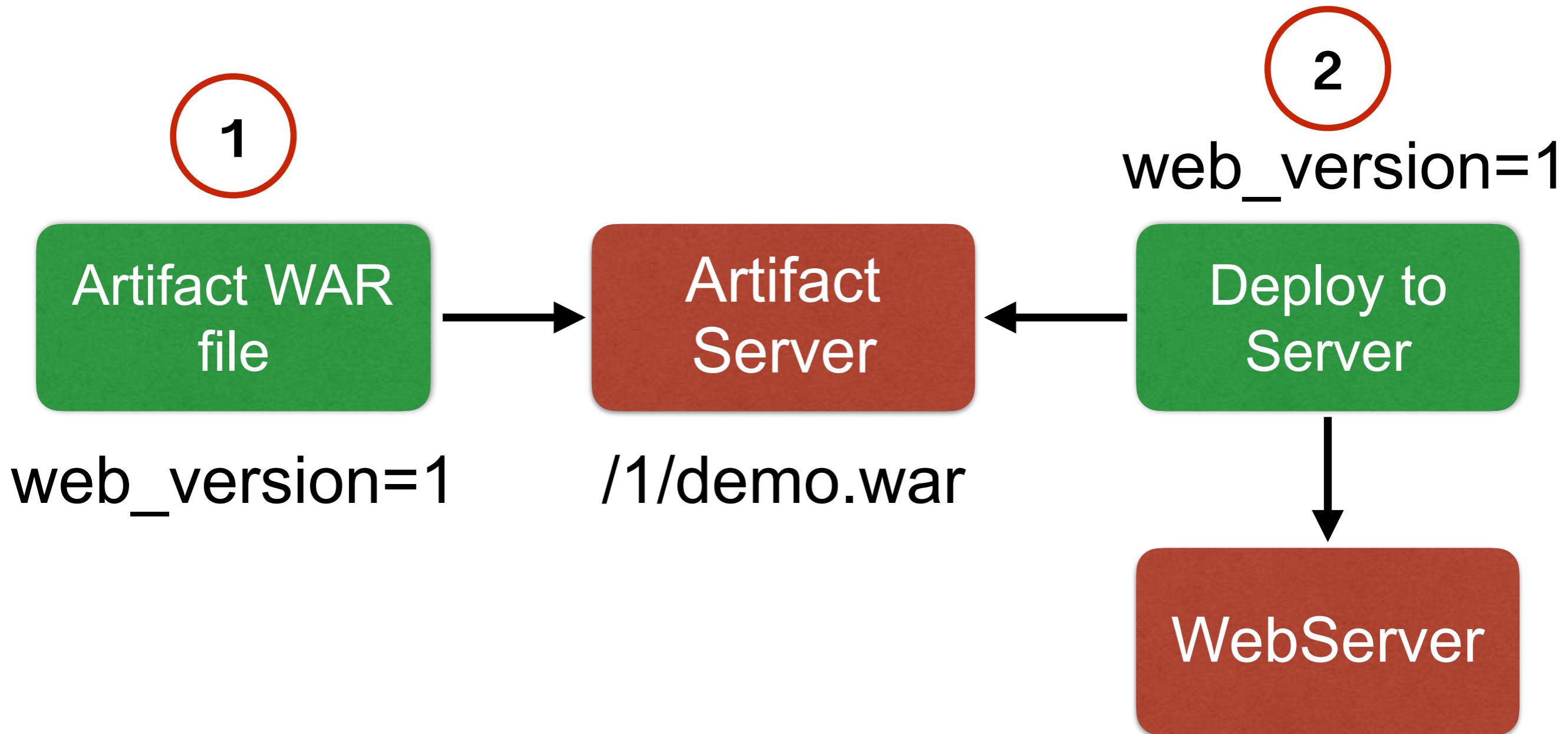


# **How to manage version of WAR file ?**



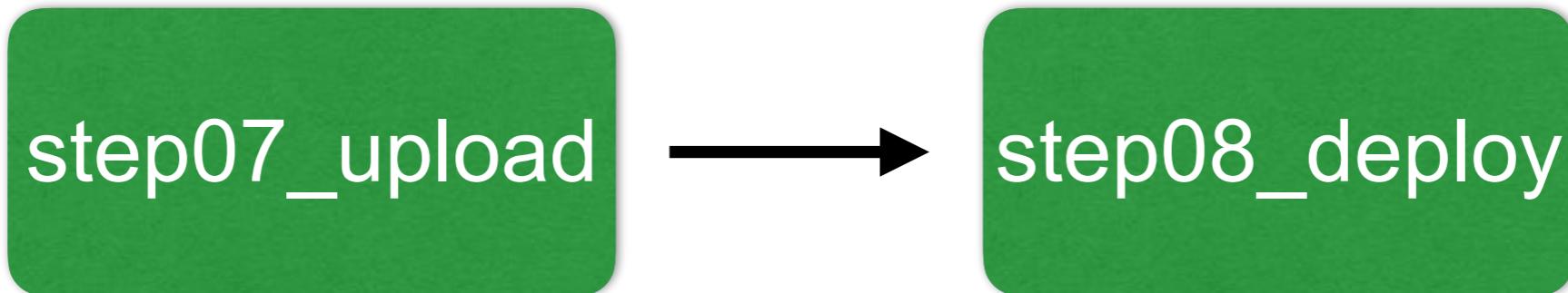
# Manage version number

Send data/value between jobs ?



# Manage version number

Send data/value between jobs ?



`web_version=1`

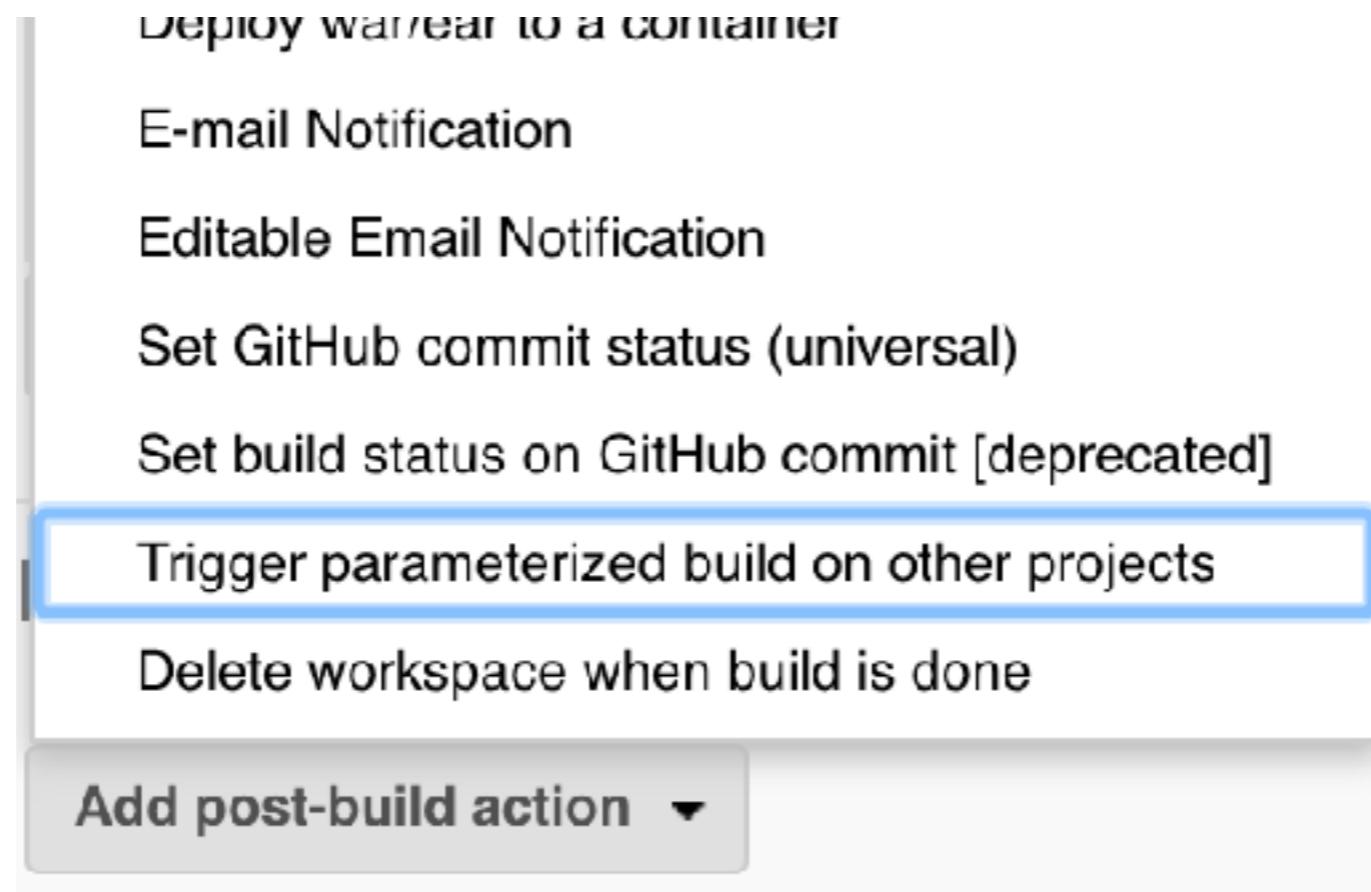


# **In job = step07\_upload**



# Add post-build actions

In step07\_upload



The screenshot shows a dropdown menu titled "Add post-build action" with a downward arrow. Inside the menu, there is a list of actions:

- 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** (highlighted with a blue border)
- Delete workspace when build is done



# 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**

The screenshot shows the Jenkins 'Post-build Actions' configuration page. A specific action named 'Trigger parameterized build on other projects' is selected. Under 'Build Triggers', the 'Projects to build' field contains 'step08\_deploy,' which is highlighted in red with the error message 'Blank project name in the list'. The 'Trigger when build is' dropdown is set to 'Stable'. There is also an unchecked checkbox for 'Trigger build without parameters'. Below this, a dropdown menu titled 'Add Parameters' is open, showing options like 'Boolean parameters', 'Build on the same node', etc., with 'Predefined parameters' currently selected. At the bottom left, a dropdown menu titled 'Add post-build action' is open, showing options like 'Predefined parameters', 'Restrict matrix execution to a subset', etc., with 'Predefined parameters' also highlighted in blue.



# 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

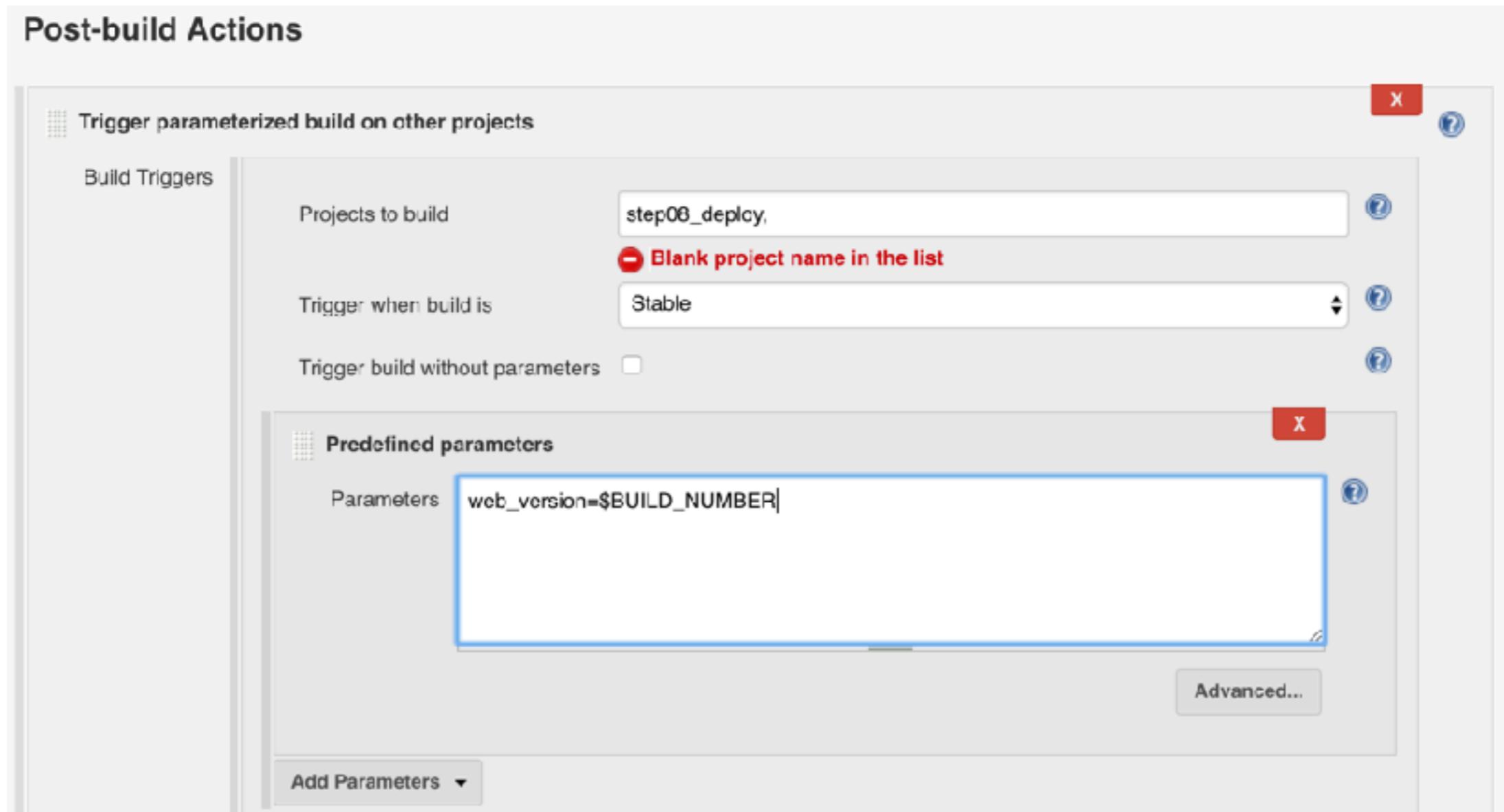
Trigger build without parameters:

**Predefined parameters**

Parameters: web\_version=\$BUILD\_NUMBER

**Advanced...**

**Add Parameters ▾**



# **In job = step08\_deploy**



# Receive parameters

Choose This project is parameterized

The screenshot shows the Jenkins General configuration page for a project. The 'General' tab is selected. Under the 'This project is parameterized' checkbox, a 'String Parameter' is defined with the name 'web\_version'. The 'Add Parameter' button is visible at the bottom left of the parameter list.

String Parameter	
Name	web_version
Default Value	
Description	

[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...

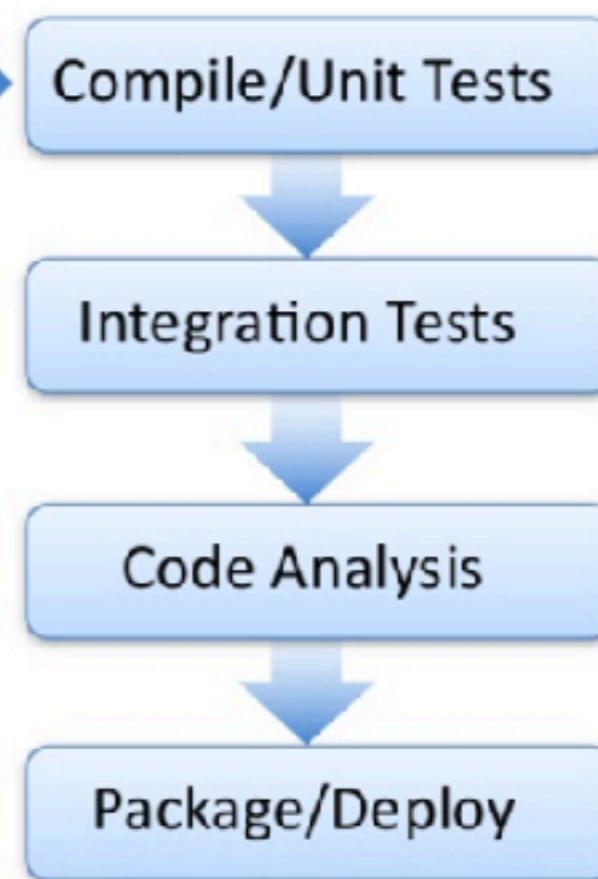
Add build step ▾





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`

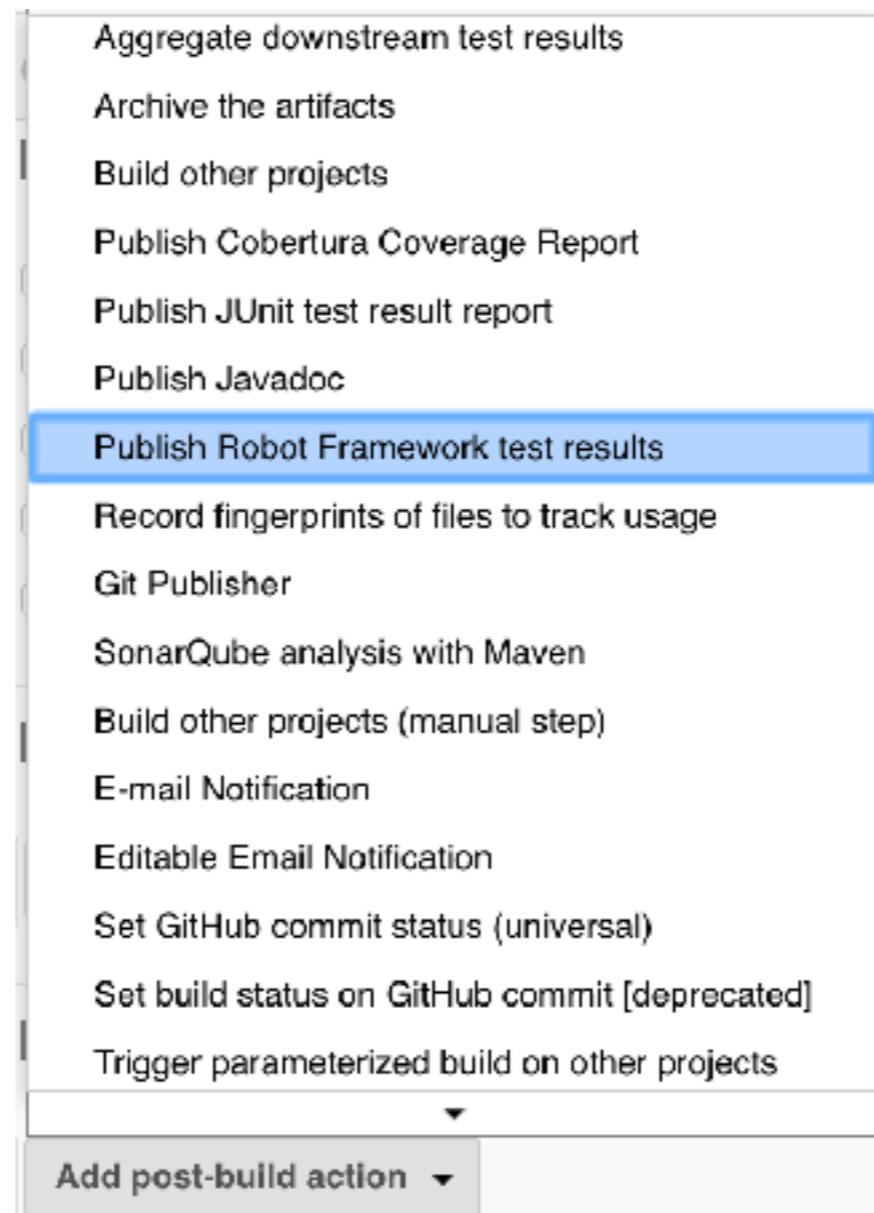
**pybot \*.robot**

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 %  **Entry must be percentage value between 0-100**

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

Use thresholds for critical tests only

Add post-build action ▾



# Report

 [add description](#)

[Disable Project](#)

Robot Framework Tests Trend (all tests)



[Workspace](#)



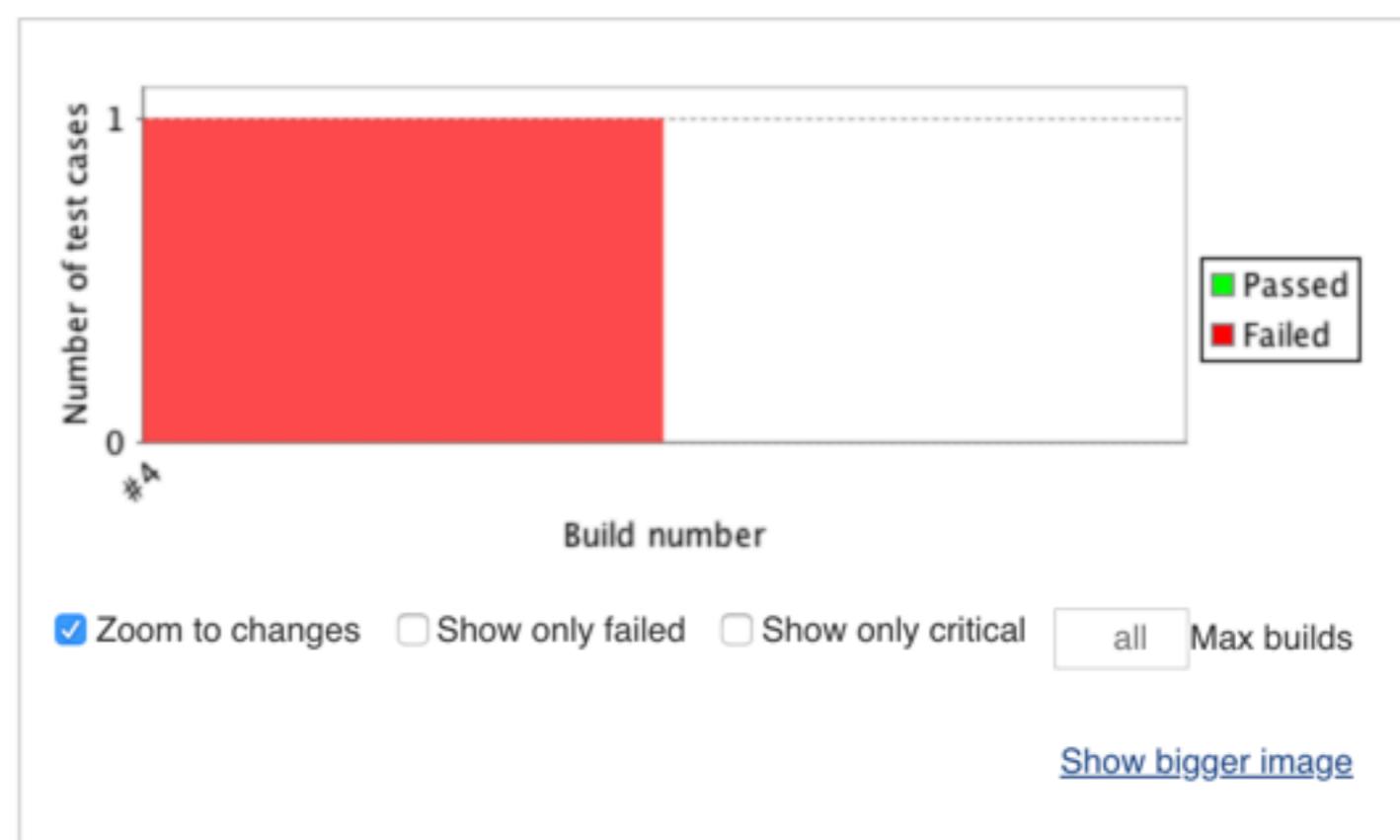
[Recent Changes](#)



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](#)



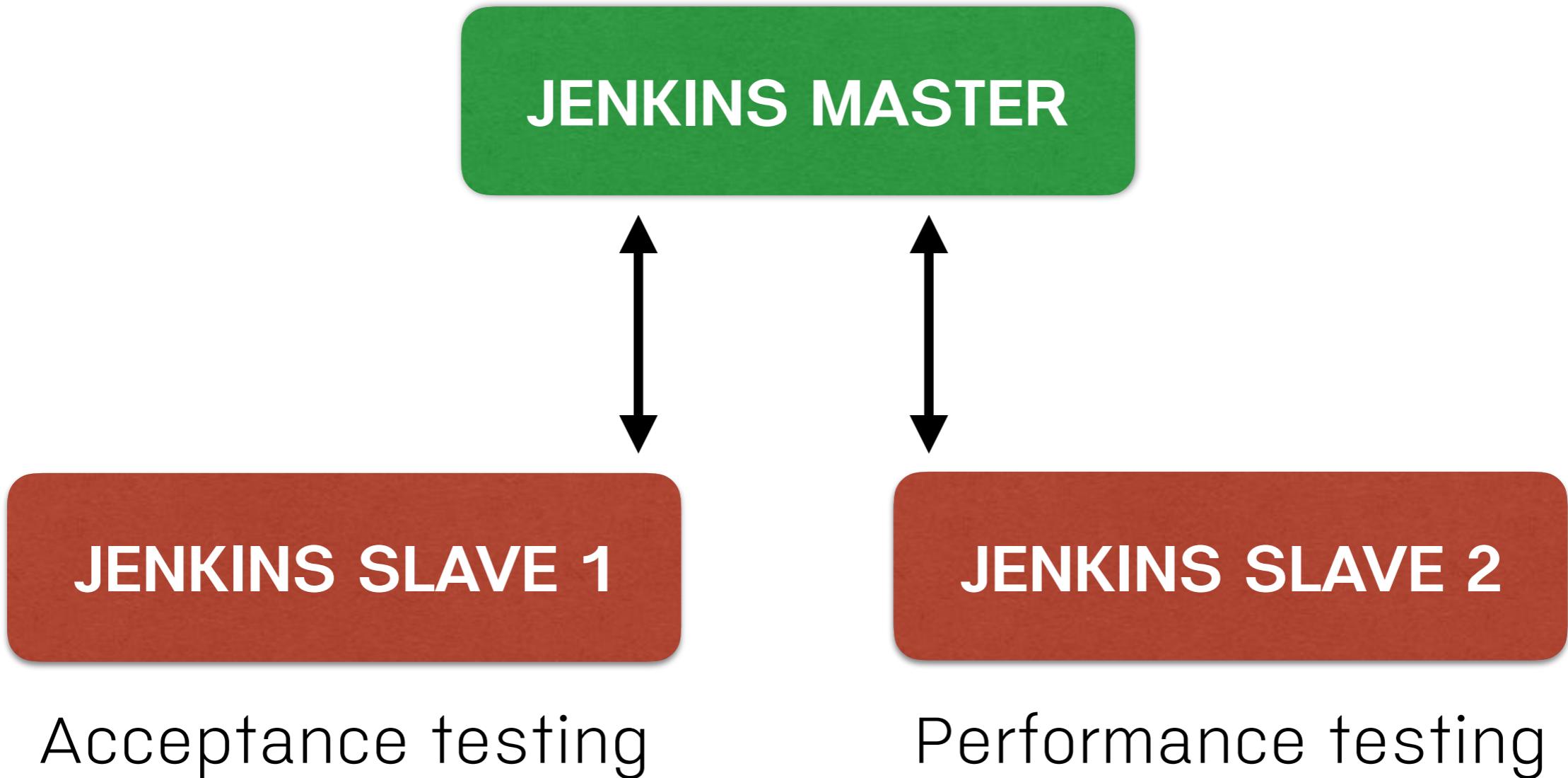
# Try it by yourself



# Jenkins Master-Slave

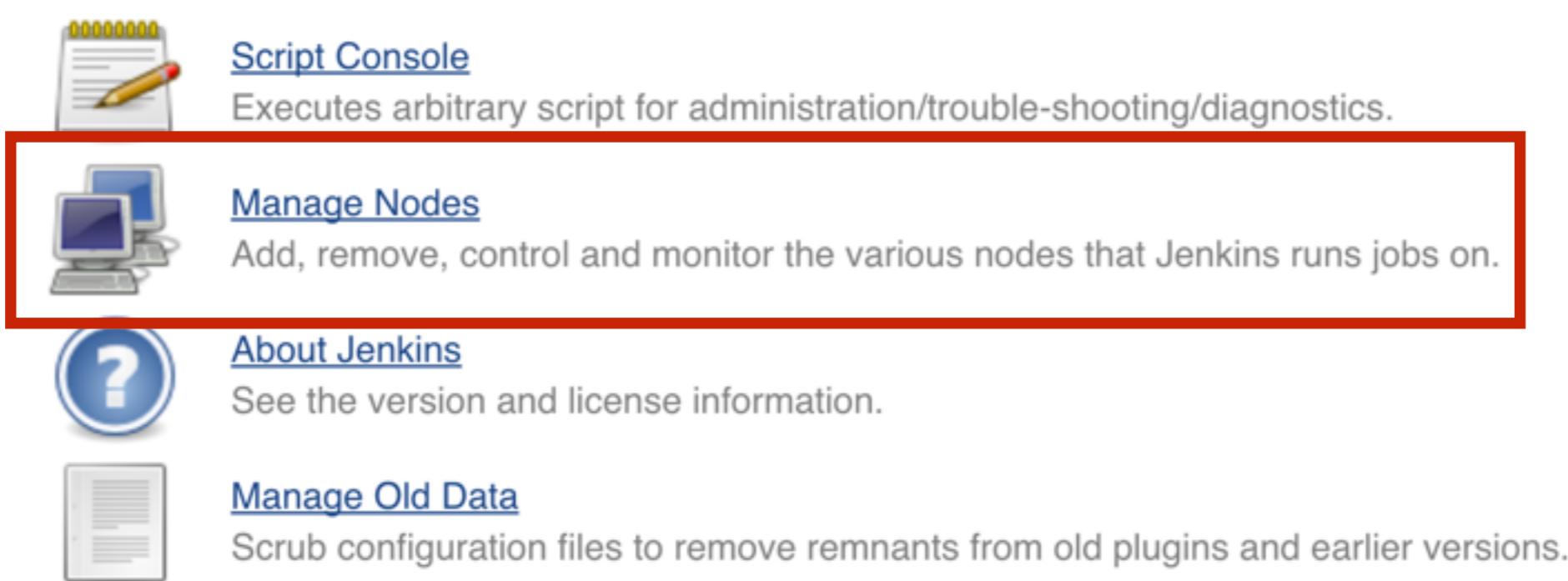


# Need more Jenkins's node



# Add new node

## 1. Manage Jenkins -> Manage Nodes



# Add new node

2. Choose new node and fill in name

The screenshot shows the Jenkins web interface. In the top left, there's a logo and the word "Jenkins". To its right is a search bar with a magnifying glass icon and the word "search". Further right are links for "Somkiat" and "I log out". Below the header, a navigation bar has "Jenkins" and "Nodes" selected. On the left, 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 shows a "Node name" field containing "testing". A radio button labeled "Permanent Agent" is selected, with a descriptive text explaining it adds a plain, permanent agent to Jenkins. At the bottom right of this section is an "OK" button. Below this, there are two collapsed sections: "Build Queue" (showing "No builds in the queue.") and "Build Executor Status" (showing "1 Idle" and "2 Idle").



# Add new node

## 3. Edit node and save

The screenshot shows the Jenkins web interface for managing nodes. The top navigation bar includes the Jenkins logo, a search bar, and user information for 'Somkiat'. The left sidebar contains links for Back to List, Status, Delete Agent, Configure, Build History, Load Statistics, Script Console, Log, System Information, and Disconnect. The main content area displays the configuration for the 'testing' node:

- Name: testing
- Description: (empty)
- # of executors: 1
- Remote root directory: /Users/somkiat/data/slides/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/slides/ci-cd/swpark/slave.jar
- Availability: Keep this agent online as much as possible

Below the main configuration, there is a section titled "Node Properties" with checkboxes for Environment variables and Tool Locations. At the bottom right is a "Save" button.



# 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 'Project hello' job. At the top left is the Jenkins logo and the word 'Jenkins'. Below it is a breadcrumb navigation: 'Jenkins > hello >'. On the left, there's 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. Below the sidebar is a button for 'Build History' and a 'trend' dropdown. The main content area is titled 'Project hello'. It features two sections: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). At the bottom right is a 'Permalinks' section.



# 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. Under the "Build" section, there is a checkbox labeled "Restrict where this project can be run" which is checked. Below it, the "Label Expression" field contains "te" and "testing". A red box highlights the "Restrict where this project can be run" checkbox and the "Label Expression" field.

**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

[PREVIOUS...](#) [NEXT...](#)

[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  
project01 » Required field

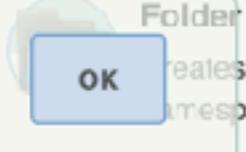
**Freestyle project**  
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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

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

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

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

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



# Write your pipeline

Pipeline script or from .Jenkinsfile

**Pipeline**

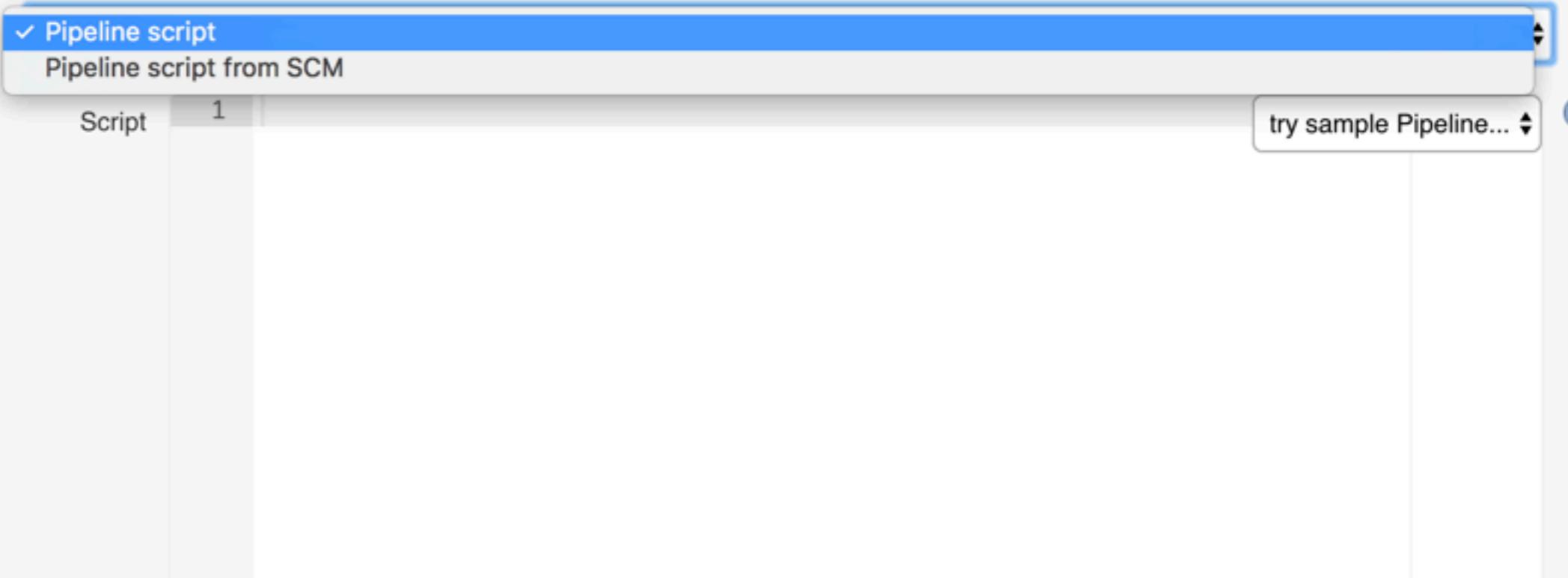
Definition

✓ Pipeline script  
Pipeline script from SCM

Script 1 try sample Pipeline... ?

Use Groovy Sandbox ?

[Pipeline Syntax](#)



# Pipeline syntax

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator page. At the top left is the Jenkins logo. To its right is a search bar with a magnifying glass icon and a help icon. Below the header, the breadcrumb navigation shows 'Jenkins > xxxx > Pipeline Syntax'. On the left, a sidebar lists links: Back, Snippet Generator (selected), Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, and IntelliJ IDEA GDSL. The main content area has a title 'Overview' followed by a detailed description of the Snippet Generator's purpose. Below this is a 'Steps' section with a sample step 'archiveArtifacts: Archive the artifacts' and a 'Files to archive' input field. An 'Advanced...' button is located at the bottom right of this section. At the bottom of the page is a large 'Generate Pipeline Script' button.

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

**Steps**

Sample Step archiveArtifacts: Archive the artifacts

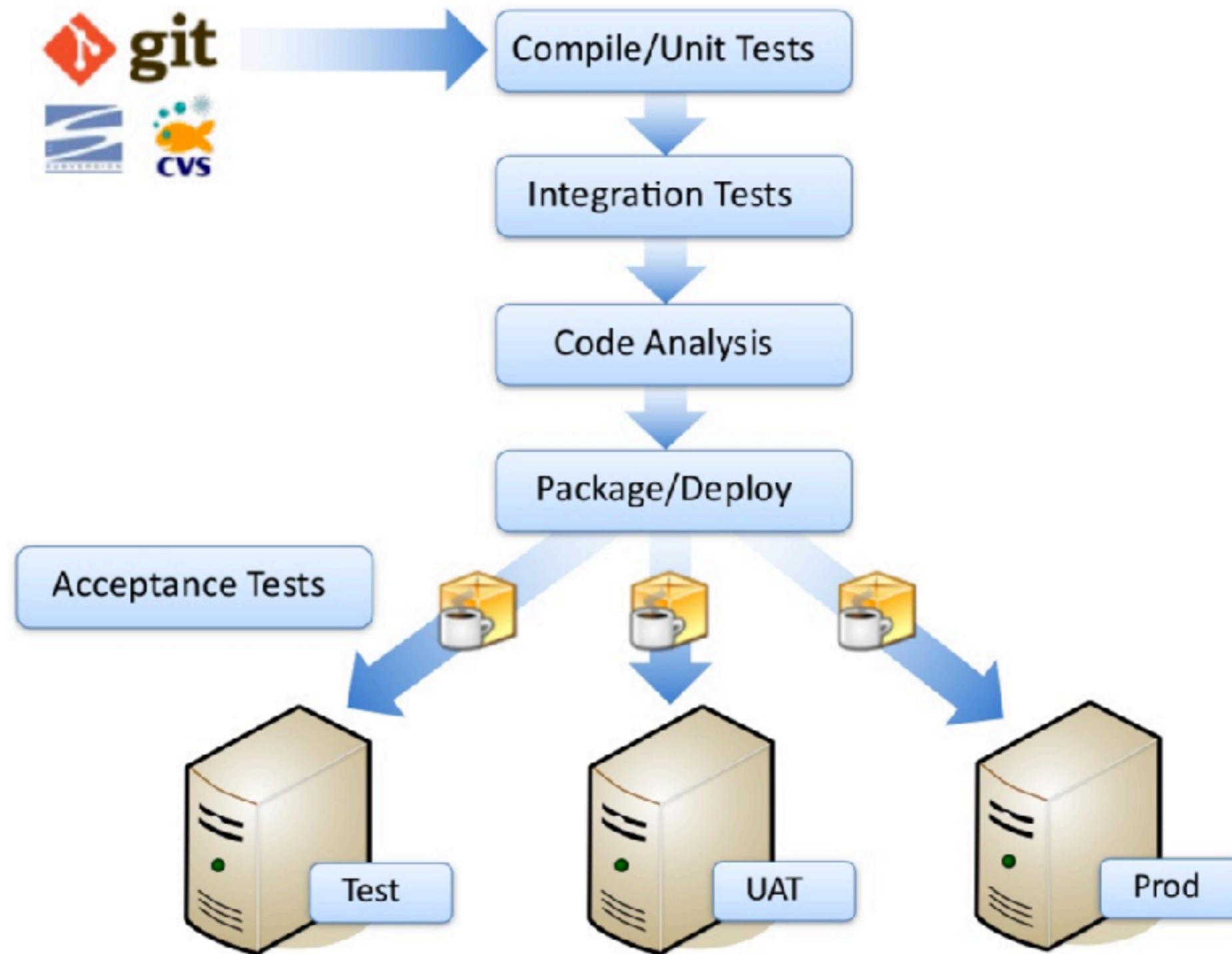
Files to archive

Advanced...

Generate Pipeline Script



# Build pipeline

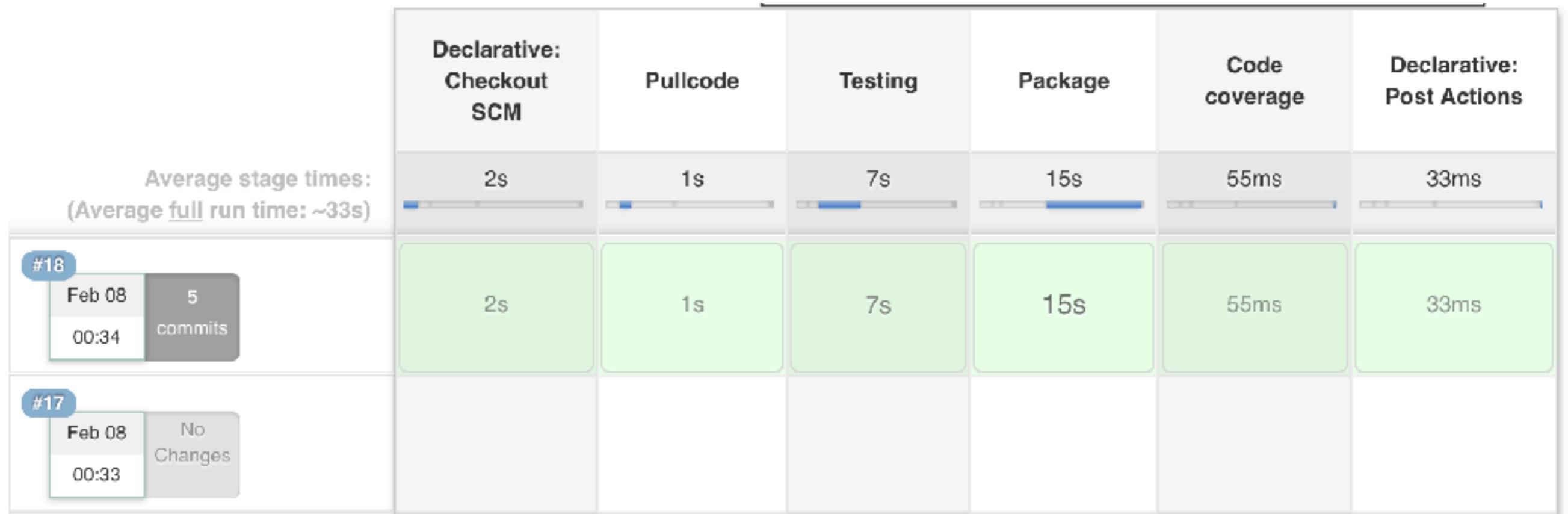


# Create pipeline as code

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



# Result



# Jenkinsfile

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



# Jenkinsfile

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



# Try it by yourself



# Useful plug-ins



# Useful plug-ins

Build Pipeline

Build Monitor View

Cobertura Report

Jacoco Report

Robotframework Report

Sound

Green Ball

SonarQube Scanner

Blue Ocean



# Jenkins Best Practices

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



# Jenkins Best Practices

Always secure Jenkins

In large system, don't build on the master

Backup `JENKINS_HOME` regularly

Limit project name to the sane character set

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



# Jenkins Best Practices

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

