

# CI CD with Jenkins Workshop





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata



Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

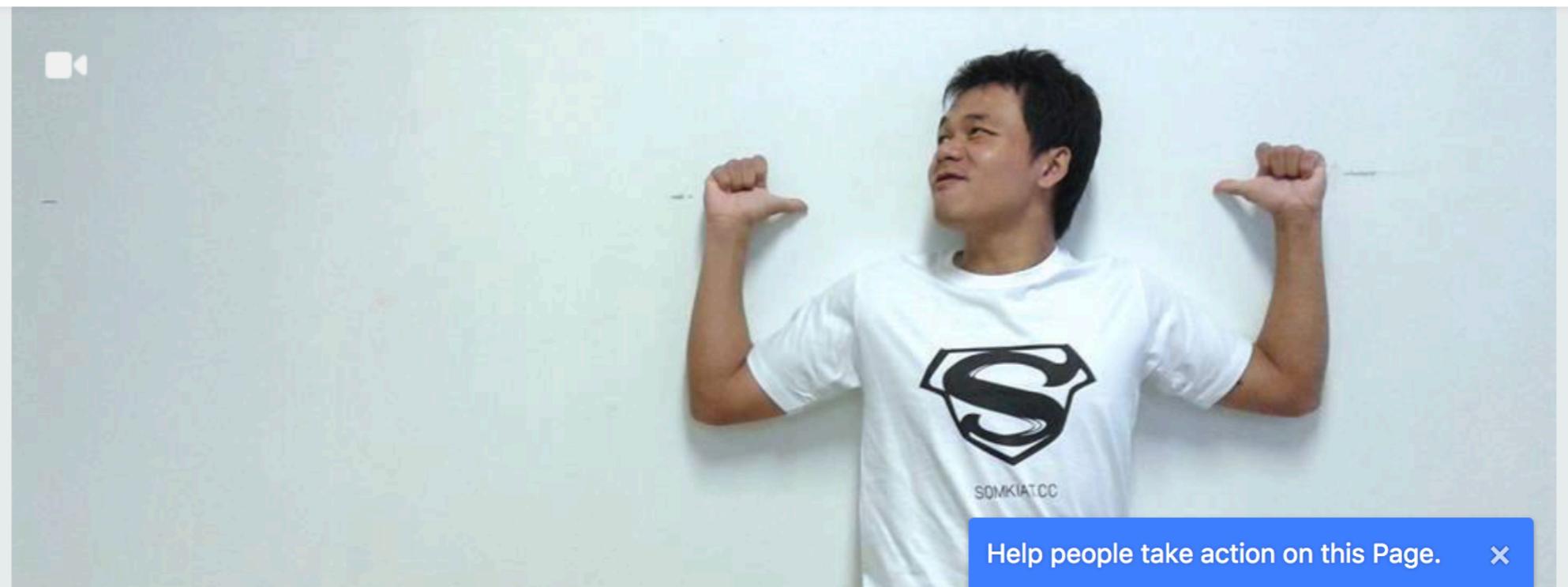
@somkiat.cc

Home

Posts

Videos

Photos



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



# Pipeline with Jenkins 101



# Topics

Pipeline as a Code in Jenkins  
Integration with GitLab  
Scripted vs Declarative pipeline  
Write and Run pipeline





# Setup Jenkins



# Working with Docker

Docker image

Blue Ocean plugins

Master and slave of Jenkins



# Create first job



# Create first job

## Freestyle job Pipeline as a Code

**Enter an item name**

» This field cannot be empty, please enter a valid name

---

 **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 agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



# Freestyle Job

Configure with web UI  
Changes are not tracked  
Separated jobs for each step in pipeline  
Manual process



# Pipeline as a Code

Configure with code

Changes are tracked by SCM

Easy to import/export

Parameterized and reuse

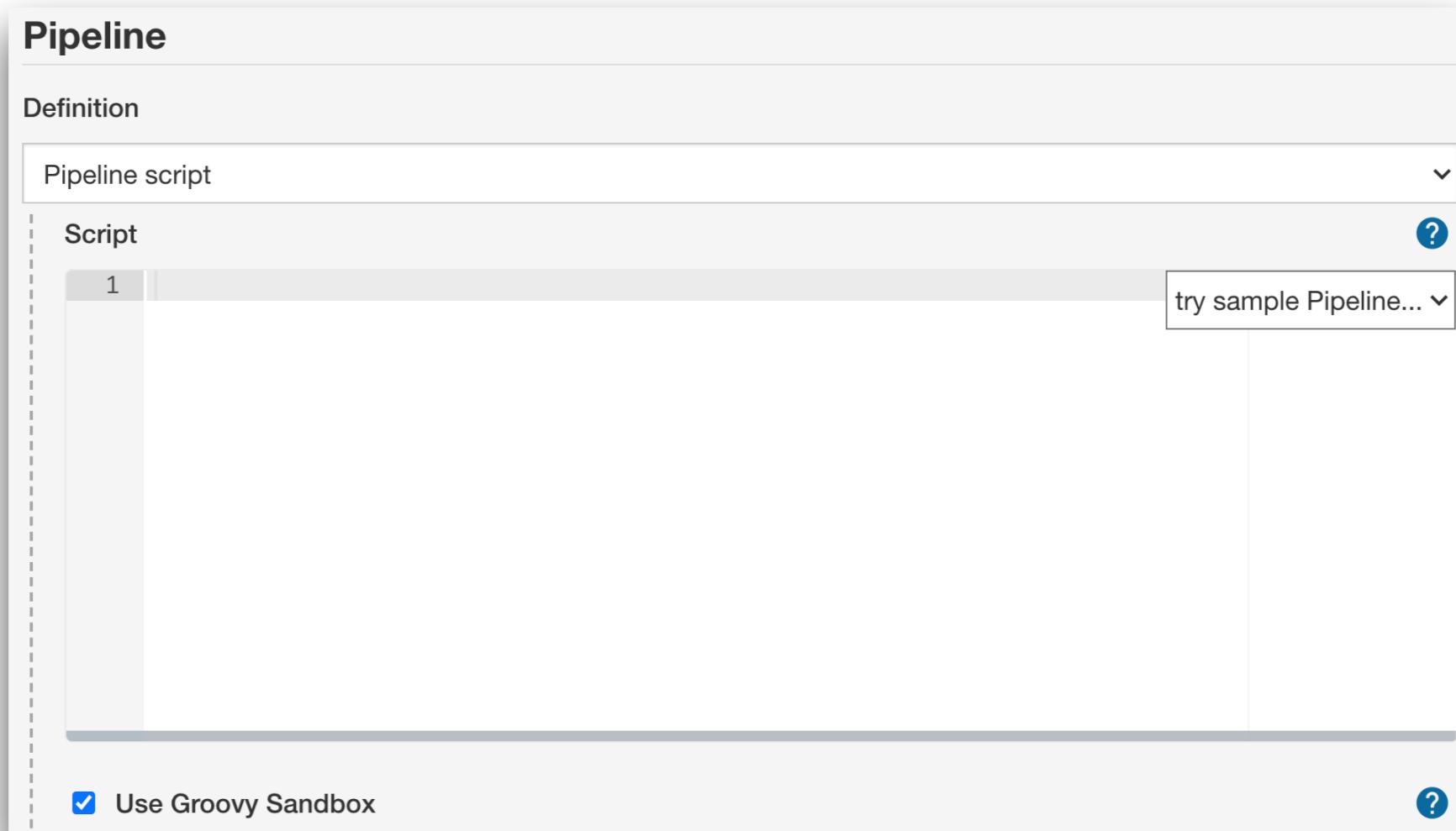
Auto generate

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



# Types of pipeline as a code

Scripted pipeline (Groovy)  
Declarative pipeline (New)



## Scripted pipeline

```
node('worker_node1') {  
    stage('Source') {  
        git "  
    }  
    stage('Build') {  
        sh ""  
    }  
}
```

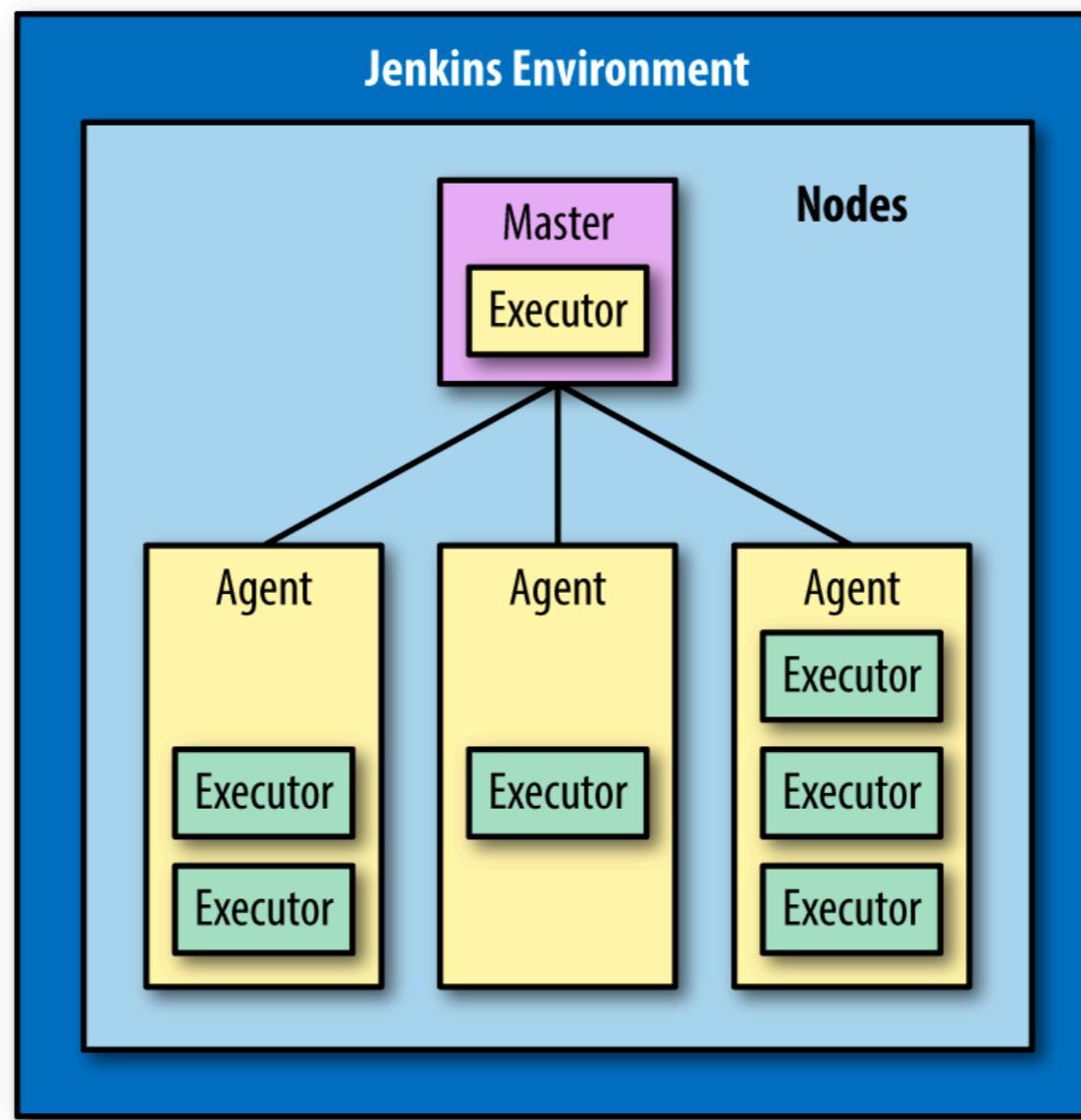
## Declarative pipeline

```
pipeline {  
    agent {label 'worker_node1'}  
    stages {  
        stage('Source') {  
            steps {  
                git "  
            }  
        }  
        stage('Build') {  
            steps {  
                sh ""  
            }  
        }  
    }  
}
```



# Basic of Jenkins

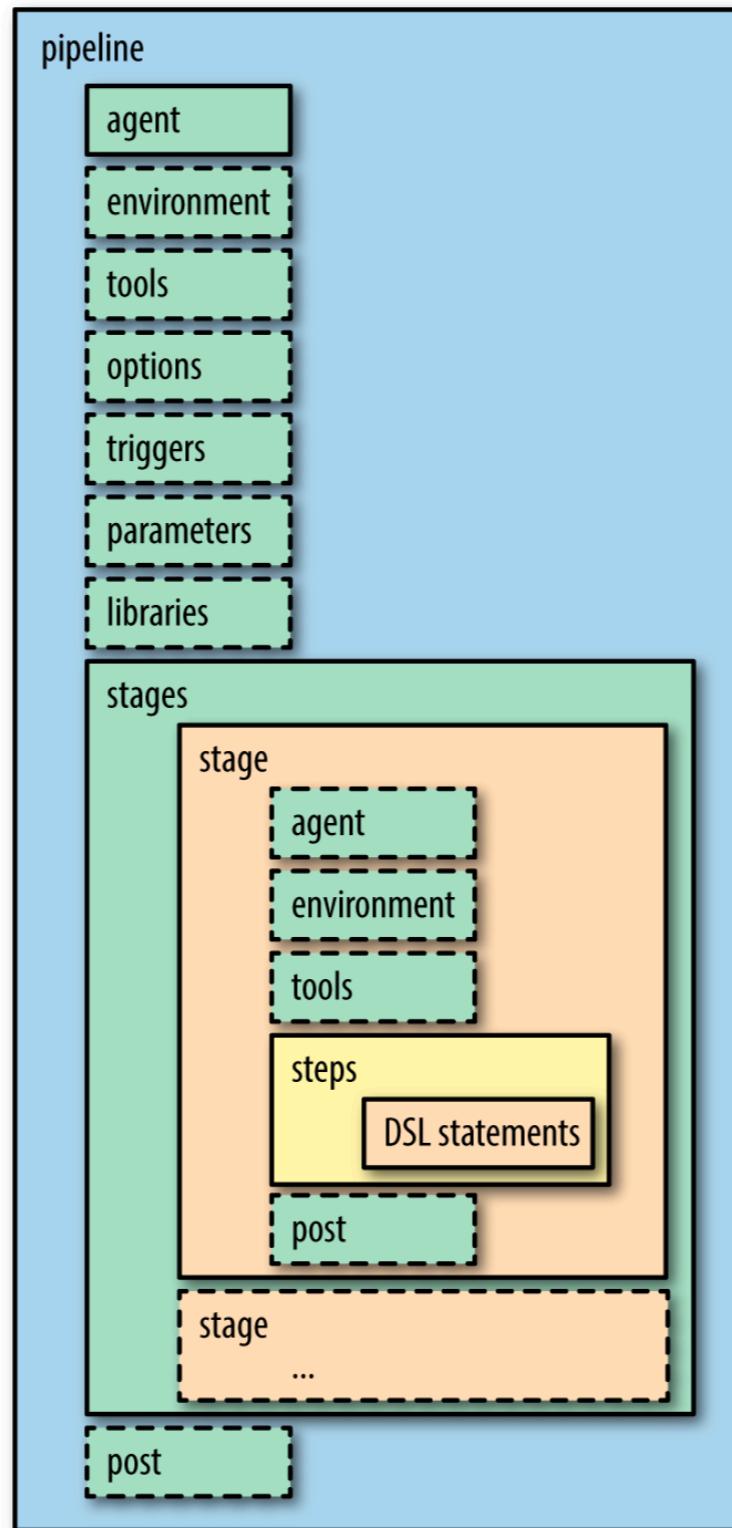
## Master, Node, Agent and Executor



# Create pipeline with Declarative pipeline



# Declarative pipeline structure



# Create Pipeline

Basic flow

Parallel pipeline

Alert and notification

Approve before deploy



# Hello pipeline

```
pipeline {  
    agent any  
  
    stages {  
        stage('Hello') {  
            steps {  
                echo 'Hello World'  
            }  
        }  
    }  
}
```



# Agent

```
pipeline {  
    agent any  
    Target of Node/Agent of Jenkins  
    stages {  
        stage('Hello') {  
            steps {  
                echo 'Hello World'  
            }  
        }  
    }  
}
```



# Stages -> stage

```
pipeline {  
    agent any  
  
    stages {  
        stage('Hello') {  
            steps {  
                echo 'Hello World'  
            }  
        }  
    }  
}
```

**Group of Works/Jobs**



# Steps

```
pipeline {  
    agent any  
  
    stages {  
        stage('Hello') {  
            steps {  
                echo 'Hello World'  
            }  
        }  
    }  
}
```

**Lowest level of function in Jenkins DSL**  
**Can be used with Groovy commands**



# Stages -> Steps -> Post

```
stages {  
    stage('name1') {  
        steps {  
            ...  
        }  
        post {  
            ...  
        }  
    }  
    post {  
        ...  
    }  
}
```



# Stages -> Steps -> Post

```
stages {  
    stage('name1') {  
        steps {  
            ...  
        }  
        post {  
            ...  
        }  
    }  
}  


post {  
    ...  
}


```



# Post-conditions

Always  
Success  
Changed  
Aborted  
Failure  
Unstable

...

<https://www.jenkins.io/doc/book/pipeline/syntax/#post>



# Post-conditions

Condition name	Description
<b>always</b>	Always execute the steps in the block
changed	If the current build's status is different from the previous build's status
<b>success</b>	If the current build was successful
<b>failure</b>	If the current build failed
unstable	If the current build's status was unstable (test failure)

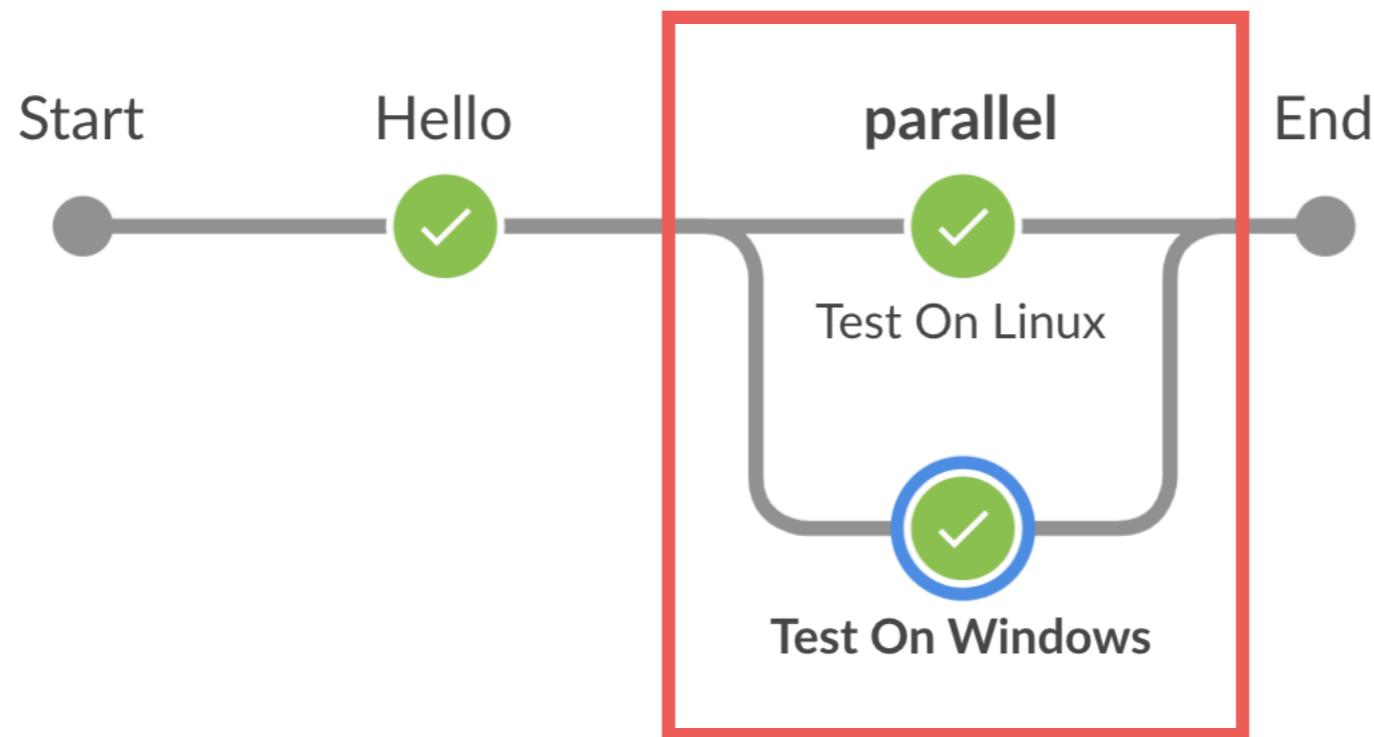
<https://www.jenkins.io/doc/book/pipeline/syntax/#post>



# Parallel stages



# Parallel stages

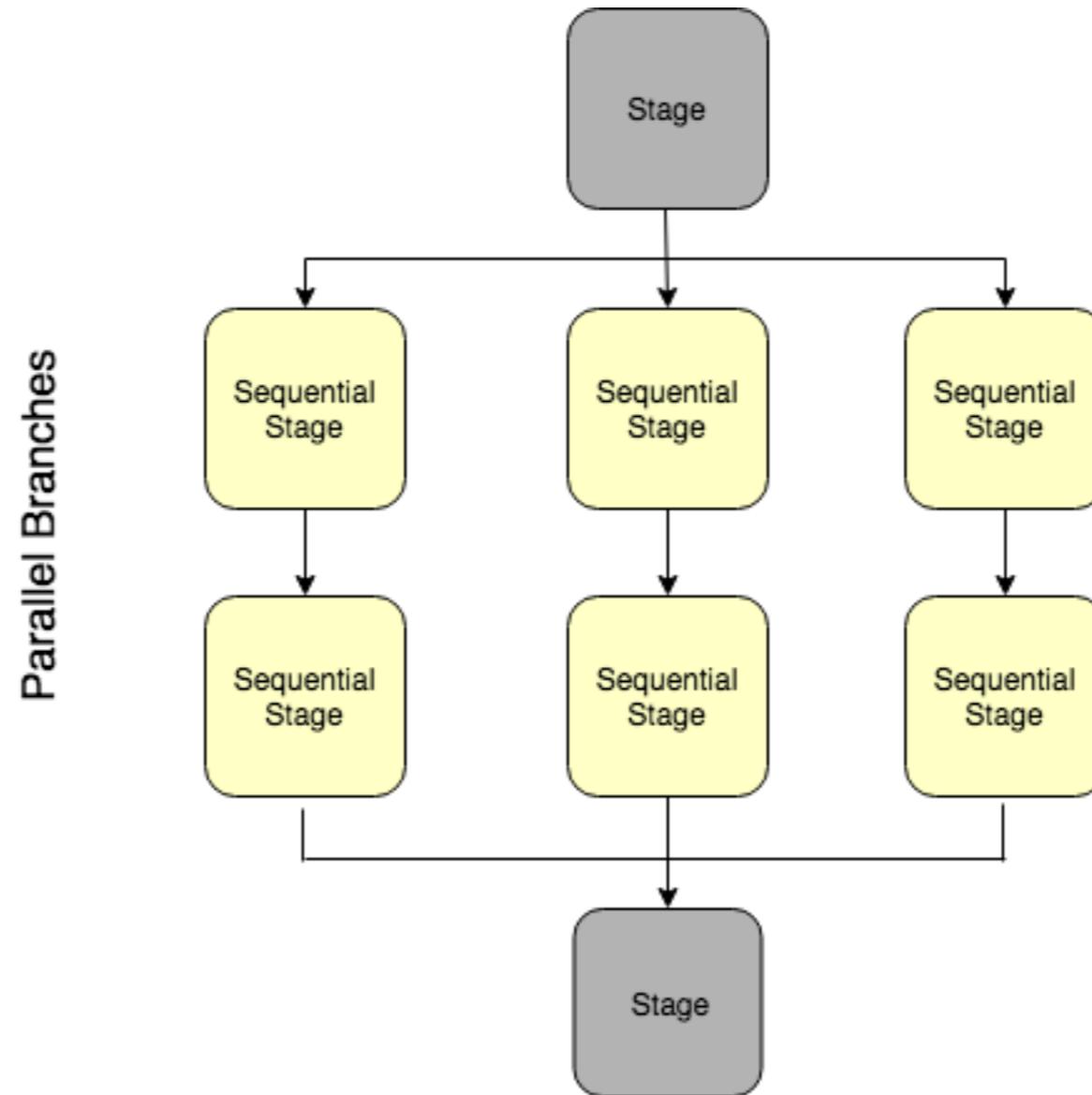


# Parallel stages

```
stage('parallel') {  
    parallel {  
        stage('Test On Windows') {  
            steps {  
                echo "Test On Windows"  
            }  
        }  
        stage('Test On Linux') {  
            steps {  
                echo "Test On Linux"  
            }  
        }  
    }  
}
```



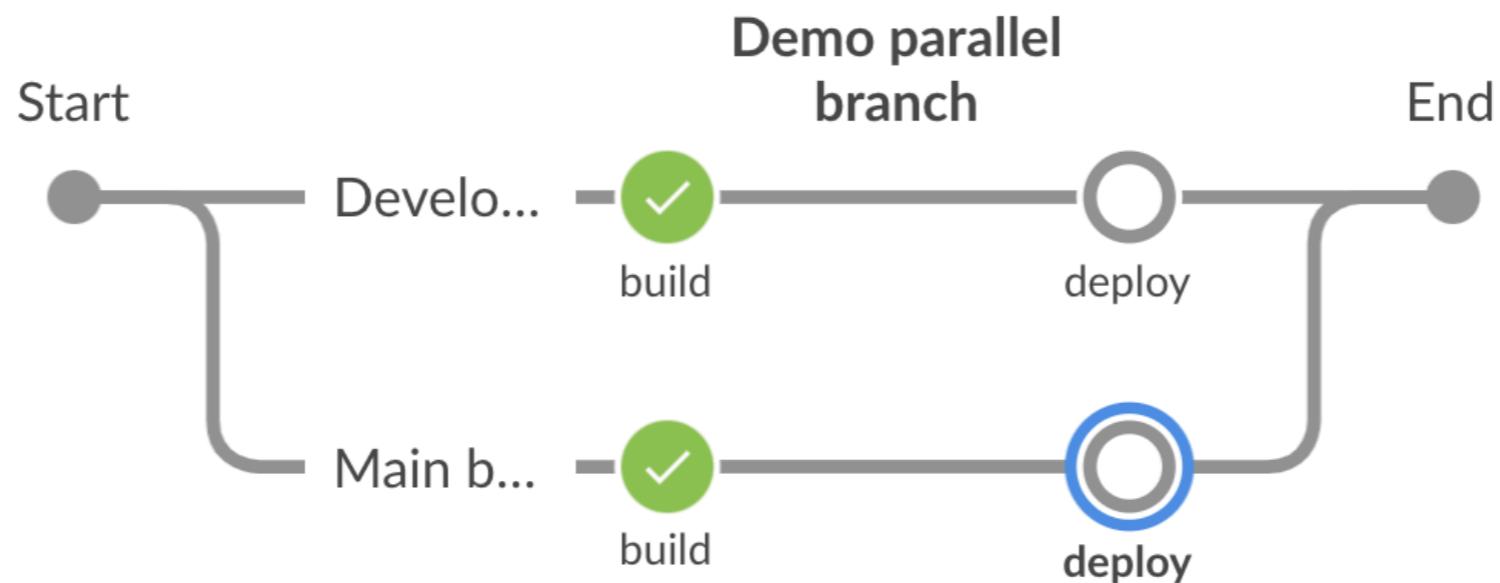
# Parallel branches



<https://www.jenkins.io/blog/2018/07/02/whats-new-declarative-pipeline-13x-sequential-stages/>



# Parallel branches



# Parallel branches

```
pipeline {  
    agent none  
  
    stages {  
        stage("Demo parallel branch") {  
            parallel {  
                stage("Main branch") {  
                    ....  
                }  
                ....  
            }  
            stage("Develop branch") {  
                ....  
            }  
        }  
    }  
}
```

**Pipeline for main branch**



# Parallel branches

```
pipeline {  
    agent none  
  
    stages {  
        stage("Demo parallel branch") {  
            parallel {  
                stage("Main branch") {  
                    ....  
                }  
                stage("Develop branch") {  
                    ....  
                }  
            }  
        }  
    }  
}
```

**Pipeline for develop branch**



# Conditions with when

Branch  
Environment  
Equals  
Expression  
Tag

...

<https://www.jenkins.io/doc/book/pipeline/syntax/#when>



# Conditionals :: main branch

```
stages {  
    stage("build") {  
        steps {  
            echo "build in main"  
        }  
    }  
    stage("deploy") {  
        when {  
            branch "main"  
        }  
        steps {  
            echo "deploy for master"  
        }  
    }  
}
```

## Working with conditions



# Working with Environment variables



# Use environment variables

Define for all stages

Define for specific stages/steps

Support plain text, user/pass and credential

<https://www.jenkins.io/doc/book/pipeline/syntax/#environment>



# Use environment variables

```
pipeline {  
    agent any  
  
    environment {  
        field = 'some'  
    }  
    stages {  
  
    }  
}
```

**Global env for all**



# Use environment variables

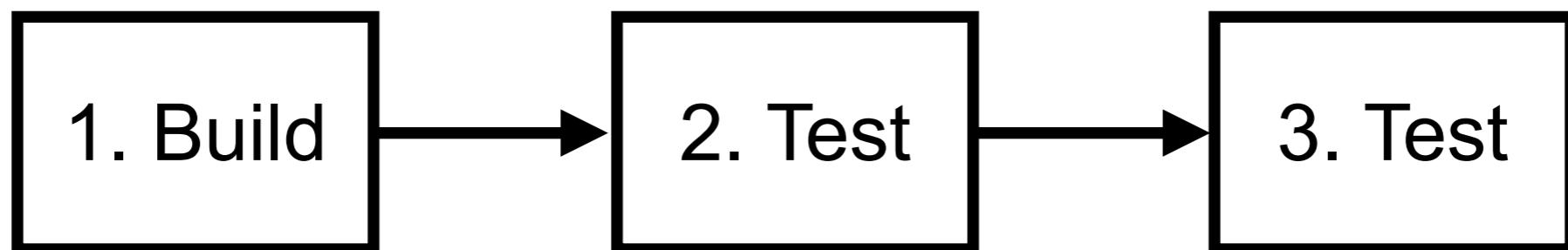
```
stages {  
    stage ('Preparation') {  
  
        environment {  
            JENKINS_PATH = sh(script: 'pwd', , returnStdout: true).trim()  
        }  
  
        steps {  
            echo "Hello world"  
            echo "PATH=${JENKINS_PATH}"  
            sh 'echo "JP=$JENKINS_PATH"'  
        }  
    }  
}
```



# Pipeline parameters



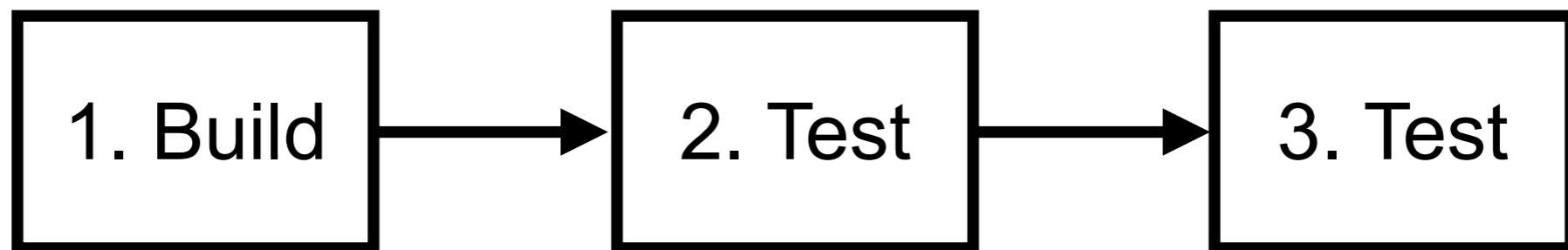
# Parameters of pipeline



Branch ?

Tag ?

Version ?



# Configure job/item in UI

This project is parameterized ?

**String Parameter** X ?

Name ?  
PERSON

Default Value ?  
Mr Jenkins

Description ?  
Who should I say hello to?

[Plain text] [Preview](#)

Trim the string ?



# Parameters directive

List of parameters that provide when trigger the pipeline

String, text (multiple line), boolean  
Choice, password

...

<https://www.jenkins.io/doc/book/pipeline/syntax/#parameters>



# Use parameters directive

```
pipeline {  
    agent any  
  
    parameters {  
        string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who should I say hello to?')  
  
        booleanParam(name: 'TOGGLE', defaultValue: true, description: 'Toggle this value')  
  
        choice(name: 'CHOICE', choices: ['One', 'Two', 'Three'], description: 'Pick something')  
  
        password(name: 'PASSWORD', defaultValue: 'SECRET', description: 'Enter a password')  
    }  
}
```



# Example

Dashboard > demo-param >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Build with Parameters](#)

[Configure](#)

[Delete Pipeline](#)

[Full Stage View](#)

[Open Blue Ocean](#)

[Rename](#)

[Pipeline Syntax](#)

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

find

#1 Aug 25, 2021, 9:51 AM

[Atom feed for all](#) [Atom feed for failures](#)

## Pipeline demo-param

This build requires parameters:

**PERSON**

Mr Jenkins

Who should I say hello to?

**BIOGRAPHY**

Enter some information about the person

**TOGGLE**  
Toggle this value

**CHOICE**

One

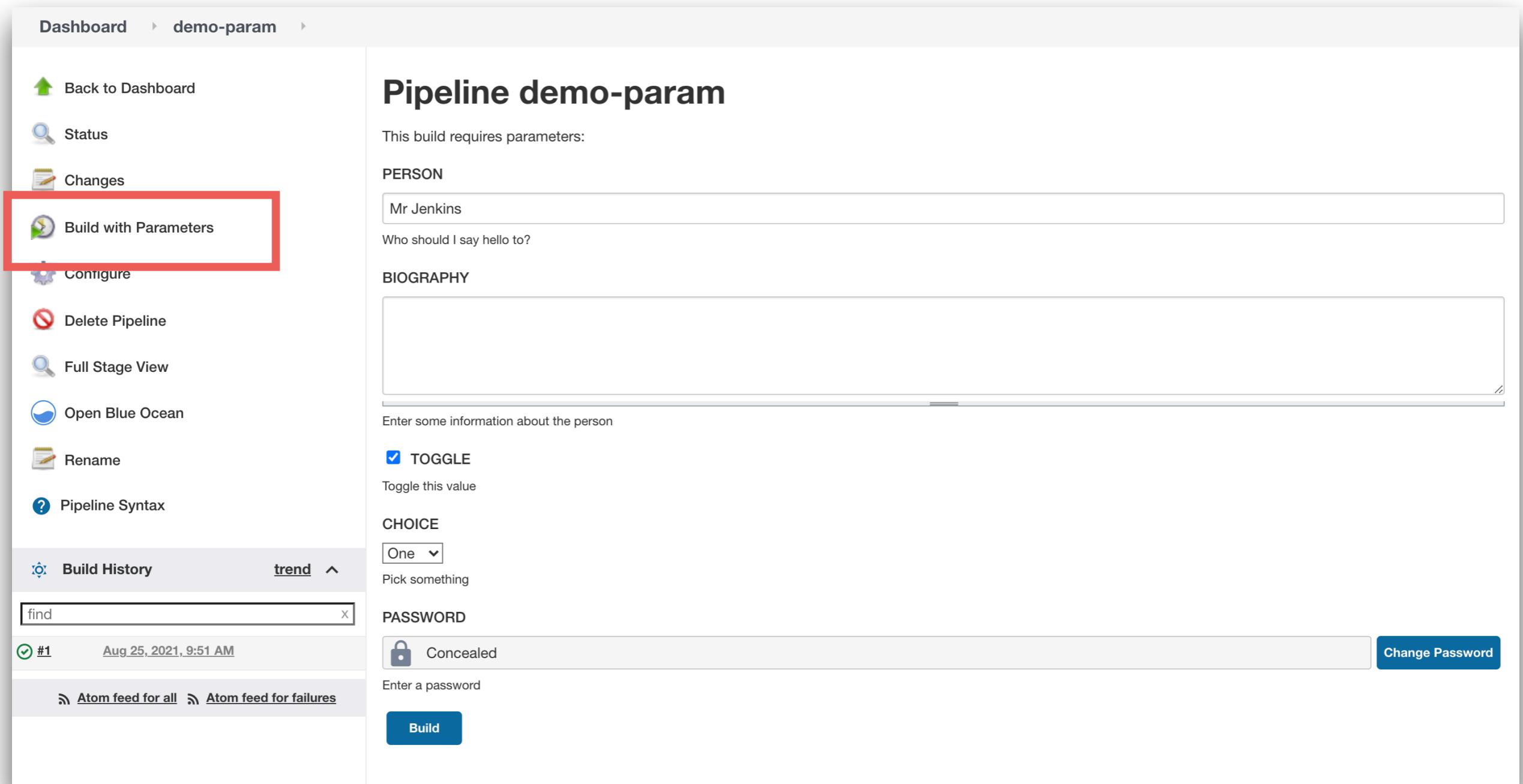
Pick something

**PASSWORD**

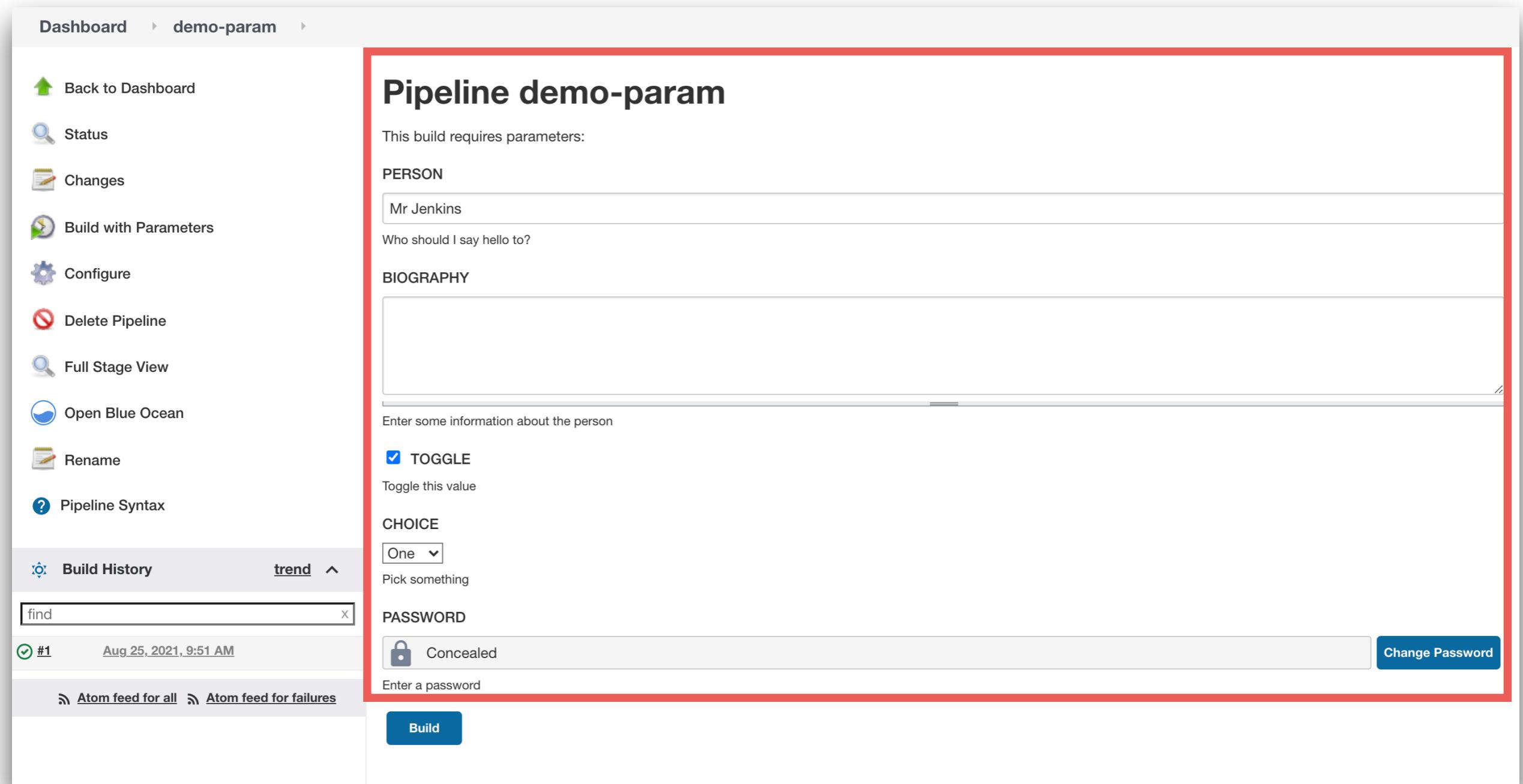
Concealed [Change Password](#)

Enter a password

**Build**



# Example



The screenshot shows the Jenkins Pipeline configuration page for a pipeline named "demo-param". The left sidebar contains standard Jenkins navigation links: Back to Dashboard, Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax, Build History (with a trend dropdown), and a search bar. Below the search bar, there are links for Atom feed for all and Atom feed for failures.

The main content area is titled "Pipeline demo-param" and displays the following parameters:

- PERSON**: A text input field containing "Mr Jenkins" with the placeholder "Who should I say hello to?"
- BIOGRAPHY**: A large text area with the placeholder "Enter some information about the person".
- TOGGLE**: A checked checkbox labeled "Toggle this value".
- CHOICE**: A dropdown menu set to "One" with the placeholder "Pick something".
- PASSWORD**: A password input field with a lock icon, labeled "Concealed", and the placeholder "Enter a password". To the right is a "Change Password" button.

At the bottom of the configuration area is a blue "Build" button.



# Git parameter plugin

## Git Parameter

Documentation

Releases

Issues

Dependencies

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- [Stored XSS vulnerability](#)
- [Multiple stored XSS vulnerabilities](#)

Adds ability to choose branches, tags or revisions from git repository configured in project.

## Plugin Info

This plugin allows you to assign git branch, tag, pull request or revision number as parameter in your builds.

<https://plugins.jenkins.io/git-parameter/>



# Notification in pipeline



# Notification

Both success and failure

Email  
LINE  
Slack  
Sound

...

<https://www.jenkins.io/doc/book/pipeline/syntax/#when>



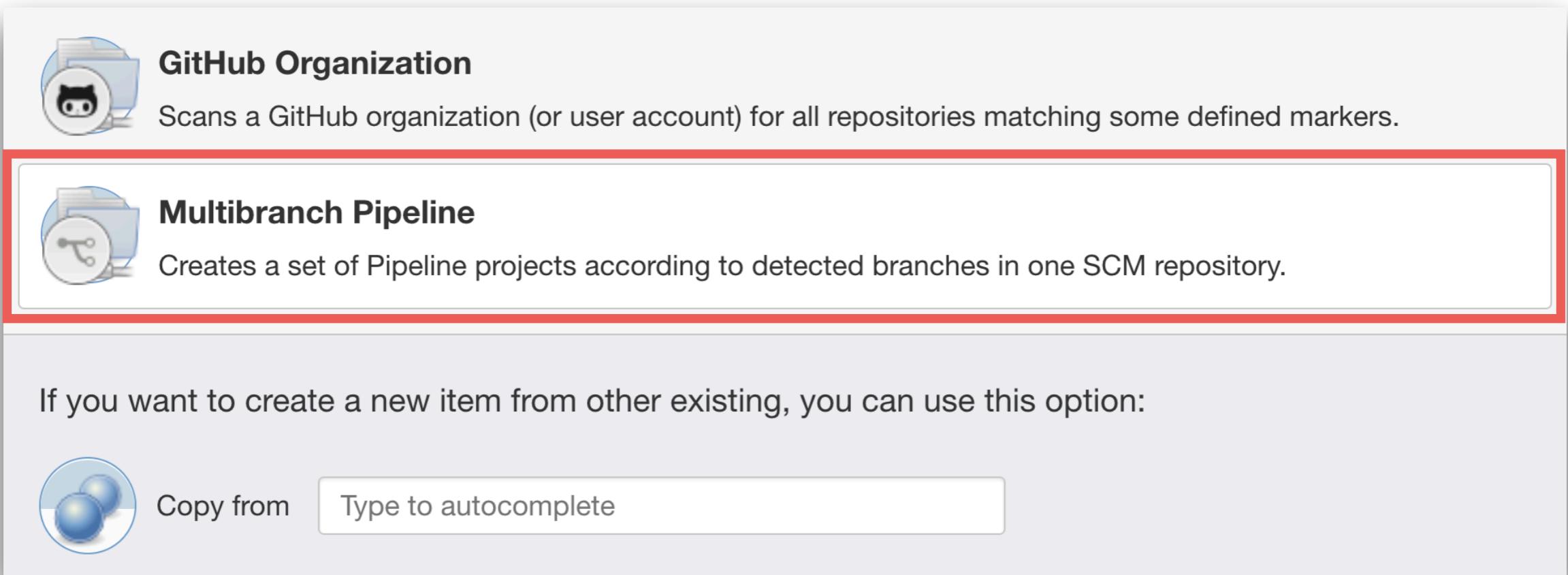
# Multi-branch pipeline

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



# Multi-branches pipeline

## Create new item/job



The screenshot shows the Jenkins 'New Item' creation interface. It lists two items:

- GitHub Organization**: Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.

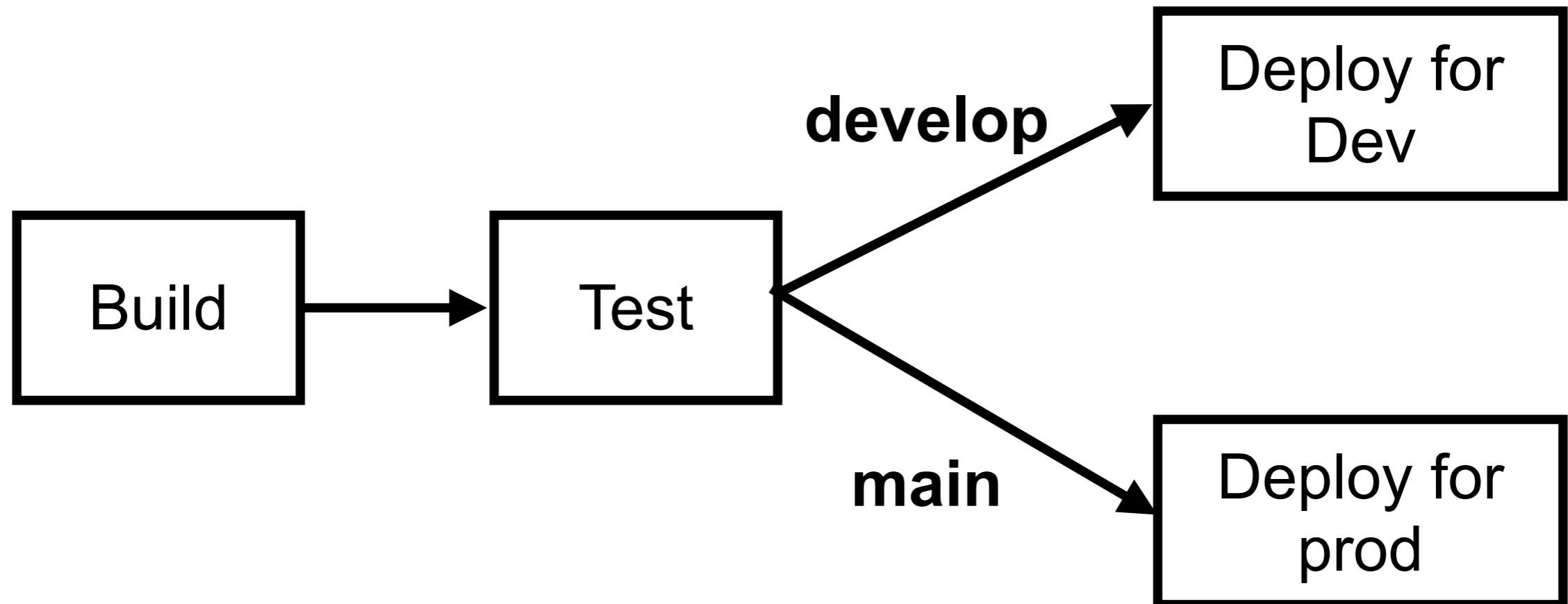
The 'Multibranch Pipeline' option is highlighted with a red border. Below the list, there is a section for copying from existing items:

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

Copy from



# Write pipeline with multi-branch



<https://www.jenkins.io/doc/tutorials/build-a-multibranch-pipeline-project/>



# Write pipeline with multi-branch

```
stages {  
    stage("build") {  
        steps {  
            echo "build in main"  
        }  
    }  
}  
  
stage("deploy") {  
    when {  
        branch "main"  
    }  
    steps {  
        echo "deploy for master"  
    }  
}
```

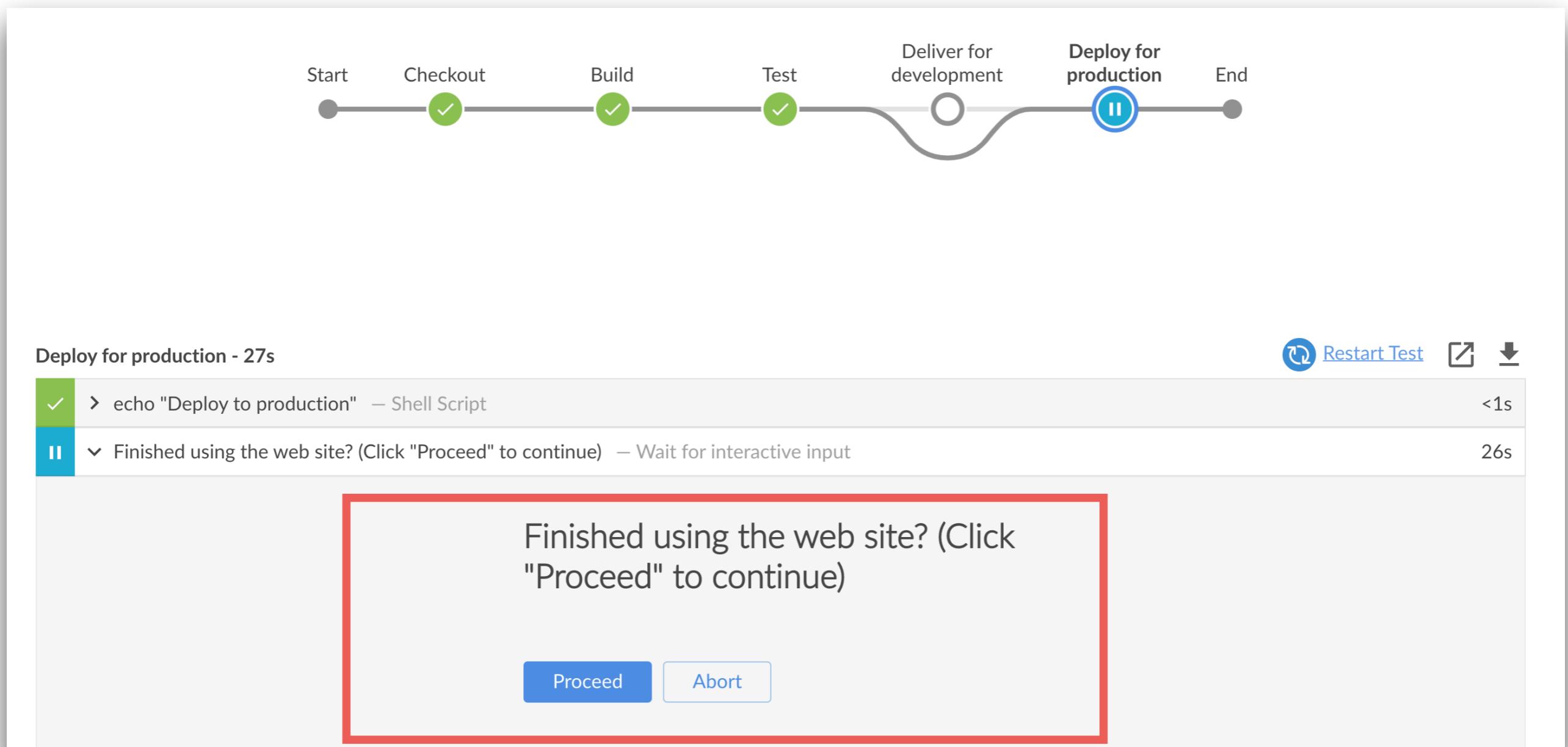
**Execute when in main branch only**



# Approve or not ?



# Approve or not ?



# Approve or not ?

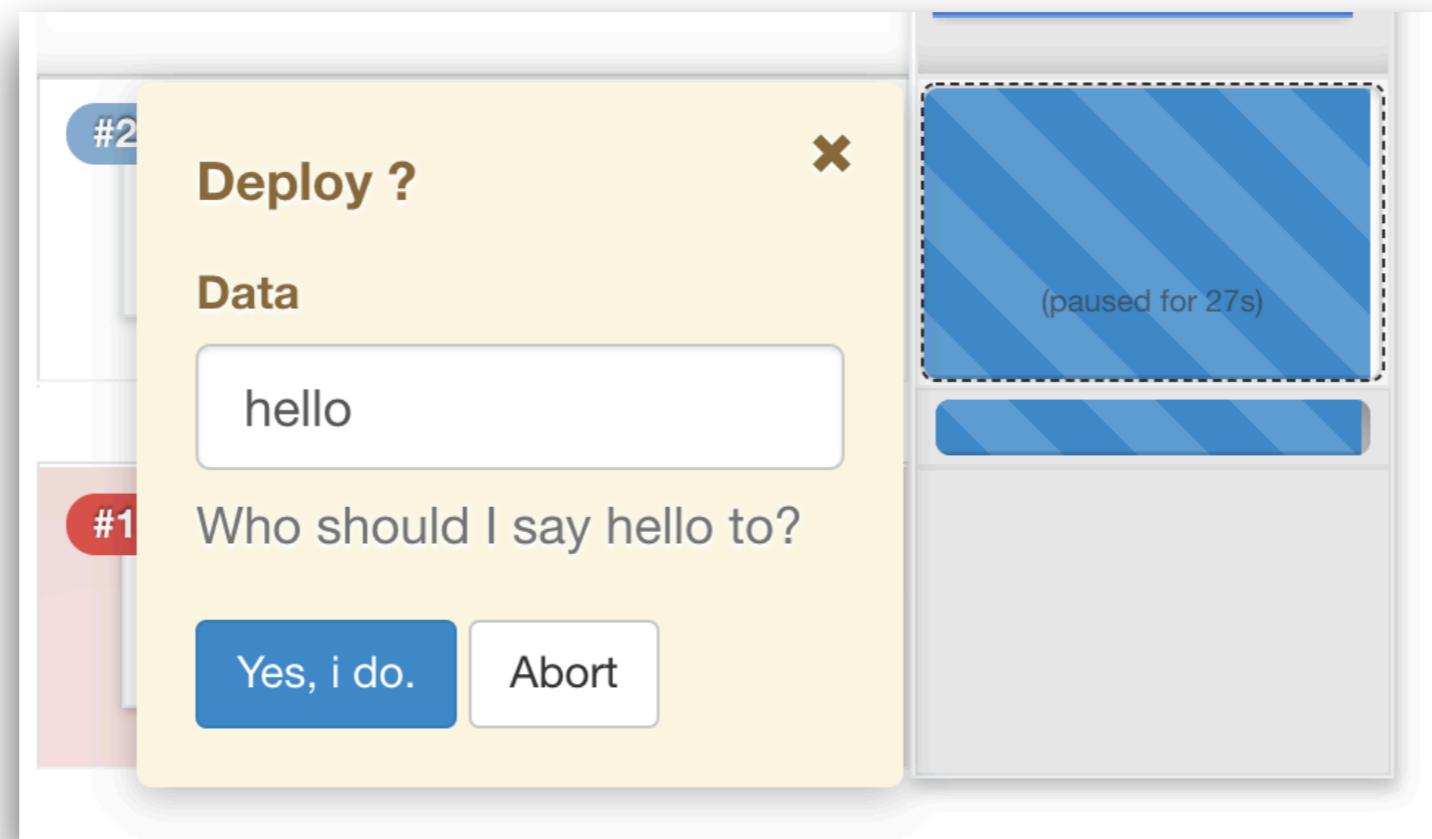
```
stage('Deliver for development') {  
    when {  
        branch 'develop'  
    }  
    steps {  
        sh 'echo "Deploy to development"  
  
        input message: 'Deploy ? (Click "Proceed" to continue')  
        sh 'echo "Deploy to development passed"'  
    }  
}
```

<https://www.jenkins.io/doc/book/pipeline/syntax/#input>



# Working with input directive

## Show prompt for input



<https://www.jenkins.io/doc/book/pipeline/syntax/#input>



# Working with input directive

## Show prompt for input

```
stage("build") {  
    input {  
        message "Deploy ?"  
        ok "Yes, i do."  
        submitter "Somkiat.p"  
        parameters {  
            string( name: 'Data', defaultValue: 'hello',  
                   description: 'Who should I say hello to?')  
        }  
    }  
    steps {  
        echo "Hello, ${Data}."  
    }  
}
```

<https://www.jenkins.io/doc/book/pipeline/syntax/#input>



# Manage credential in Jenkins



# Manage credential

Username and Password

Secret text

Secret file

SSH username with private key

Certificate

Docker host certificate authentication

<https://www.jenkins.io/doc/book/using/using-credentials/>



# Manage credential

Manage Jenkins -> Manage Credentials

The screenshot shows the Jenkins 'Manage Jenkins' page. A red circle labeled '1' highlights the 'Manage Jenkins' menu item in the sidebar. A red box highlights the 'Manage Credentials' link under the 'Security' section. A red circle labeled '2' highlights the 'Manage Credentials' link in the main content area.

**Manage Jenkins**

- My Views
- Lockable Resources
- Open Blue Ocean
- New View

**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle

**Security**

[Manage Credentials](#)

**Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.

**Configure Credential Providers**  
Configure the credential providers and types

**Global Tool Configuration**  
Configure tools, their locations and automatic installers.

**Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

**Manage Credentials**  
Configure credentials

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



# Create credentials

Global credential

By pipeline/job/item

By Jenkins's user

<https://www.jenkins.io/doc/book/using/using-credentials/>



# Types of credential

The screenshot shows the Jenkins Global credentials configuration interface. The navigation path is: Dashboard > Credentials > System > Global credentials (unrestricted). A red box highlights the 'Kind' dropdown menu, which contains the following options:

- ✓ Username with password (selected)
- GitHub App
- SSH Username with private key
- Secret file
- Secret text
- Certificate

Below the 'Kind' dropdown, there are fields for 'Username' (empty), 'Password' (empty), and 'ID' (empty). There is also a checkbox labeled 'Treat username as secret' (unchecked). At the bottom is an 'OK' button.



# Example of Docker login

```
$docker login -u <user> -p <password>
```

```
stages {  
    stage('Push to docker hub') {  
        steps {  
            withCredentials(  
                [usernamePassword(credentialsId: 'docker_login',  
                    passwordVariable: 'PASSWORD',  
                    usernameVariable: 'USERNAME')]) {  
  
                    sh 'echo "$PASSWORD" | docker login -u "$USERNAME" --password-stdin'  
                }  
            }  
        }  
    }  
}
```



# Example of Docker login

```
$docker login -u <user> -p <password>
```

```
stages {  
    stage('Push to docker hub') {  
        steps {  
            withCredentials(  
                [usernamePassword(credentialsId: 'docker_login',  
                    passwordVariable: 'PASSWORD',  
                    usernameVariable: 'USERNAME')]) {  
  
                    sh 'echo "$PASSWORD" | docker login -u "$USERNAME" --password-stdin'  
                }  
            }  
        }  
    }  
}
```



# Example of Git login

\$git pull origin main

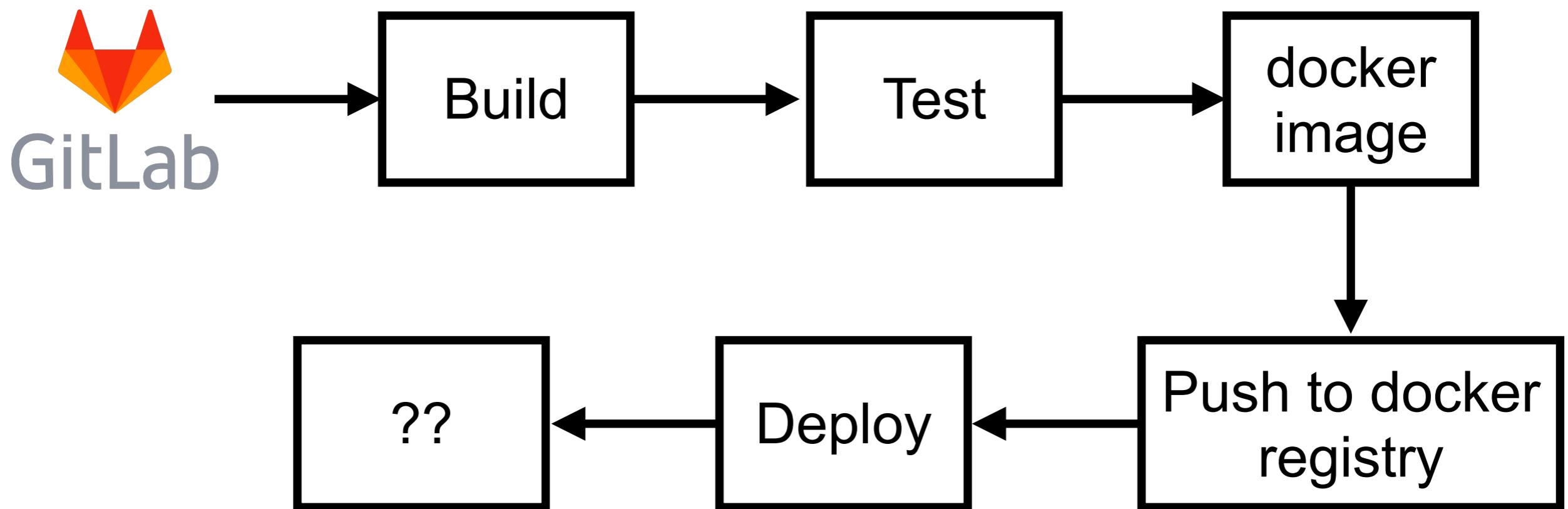
```
stages {  
    stage('Push to docker hub') {  
        steps {  
            withCredentials(  
                [usernamePassword(credentialsId: 'docker_login',  
                    passwordVariable: 'PASSWORD',  
                    usernameVariable: 'USERNAME')]) {  
  
                    sh 'echo "$PASSWORD" | docker login -u "$USERNAME" --password-stdin'  
                }  
            }  
        }  
    }  
}
```



# Pipeline workshop



# Pipeline workshop



# Security in Jenkins



# Security in Jenkins

Username/password

Access controls

**Role-bases authorization**

Other plugins

<https://www.jenkins.io/doc/book/security/>



# Role-based Authorization Strategy

## Role-based Authorization Strategy

Documentation    [Releases](#)    [Issues](#)    [Dependencies](#)

[chat on gitter](#)    [plugin v3.2.0](#)    [changelog role-strategy-3.2.0](#)    installs 70k

### About this plugin

The Role Strategy plugin is meant to be used from [Jenkins](#) to add a new role-based mechanism to manage users' permissions. Supported features

- Creating **global roles**, such as admin, job creator, anonymous, etc., allowing to set Overall, Agent, Job, Run, View and SCM permissions on a global basis.
- Creating **project roles**, allowing to set only Job and Run permissions on a project basis.
- Creating **agent roles**, allowing to set node-related permissions.
- Assigning these roles to users and user groups
- Extending role and permissions matching via [Macro extensions](#)

<https://plugins.jenkins.io/role-strategy/>



# Role-based Authorization Strategy

The screenshot shows the Jenkins plugin marketplace interface. At the top, there are four tabs: 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. Below the tabs, there are two buttons: 'Install ↑' and 'Name'. The 'Name' button is highlighted in white. The main content area displays the 'Role-based Authorization Strategy' plugin. It has a blue header with the plugin's name. Below the header are two buttons: 'Security' and 'Authentication and User Management'. A description text reads: 'Enables user authorization using a Role-Based strategy. Roles can be defined glo...'. At the bottom, there are three buttons: 'Install without restart' (blue), 'Download now and install after restart' (blue), and 'Update information' (gray).

<https://plugins.jenkins.io/role-strategy/>



# Role-based Authorization Strategy

Global roles

Project roles

Agent roles

Assign roles to users and groups

<https://plugins.jenkins.io/role-strategy/>



# Role-based Authorization Strategy

Global roles

Project roles

Agent roles

Assign roles to users and groups

<https://plugins.jenkins.io/role-strategy/>



# Working with Role-based

Manage Jenkins -> Configure global security

## Authorization

- Anyone can do anything
- Legacy mode
- Logged-in users can do anything
- Matrix-based security
- Project-based Matrix Authorization Strategy
- Role-Based Strategy



# Create a new user

Manage Jenkins -> Manage Users

## Create User

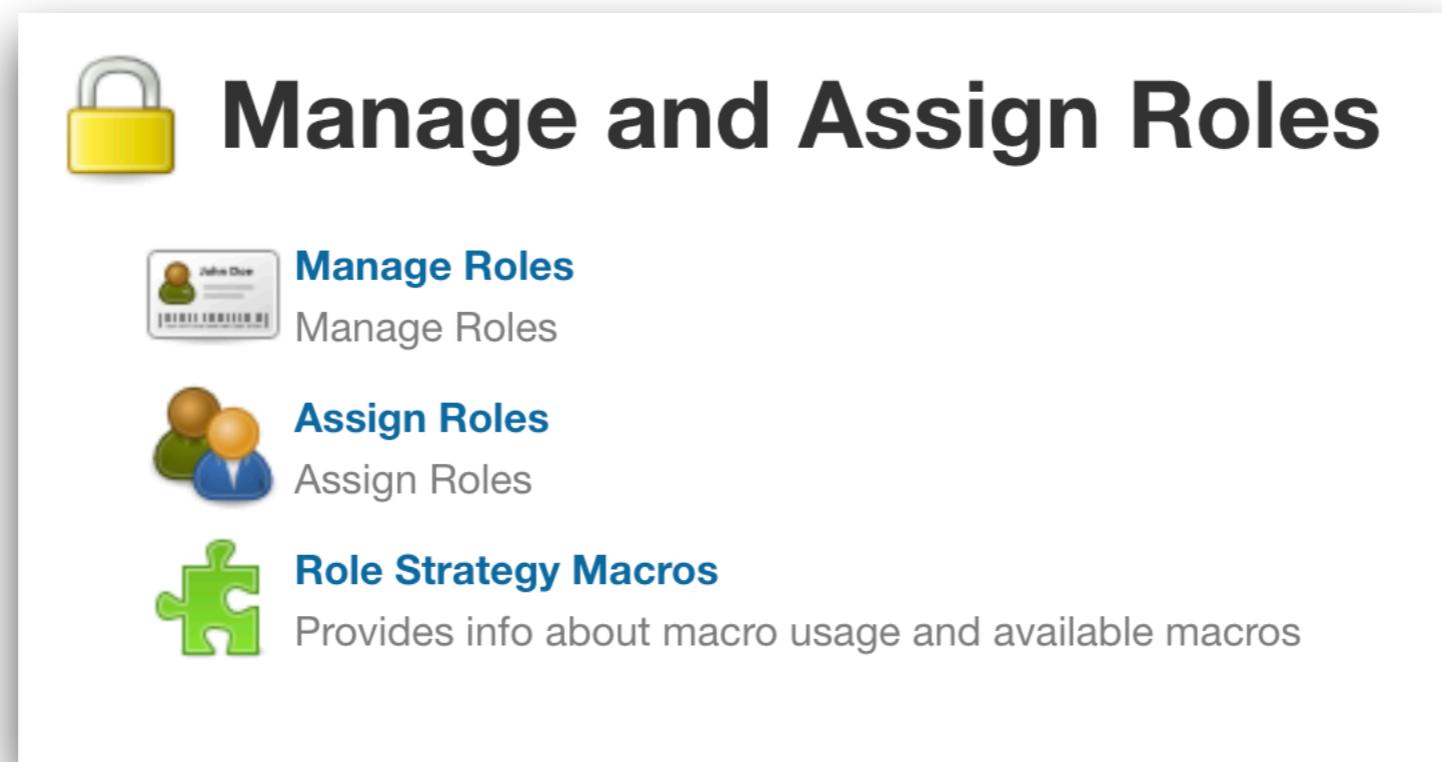
Username:	<input type="text" value="demo"/>
Password:	<input type="password" value="...."/>
Confirm password:	<input type="password" value="...."/>
Full name:	<input type="text" value="demo"/>
E-mail address:	<input type="text" value="admin@admin.com"/>

**Create User**



# Create a new role

Manage Jenkins -> Manage and assign roles



# Create a new role

Add new role = general



## Manage Roles

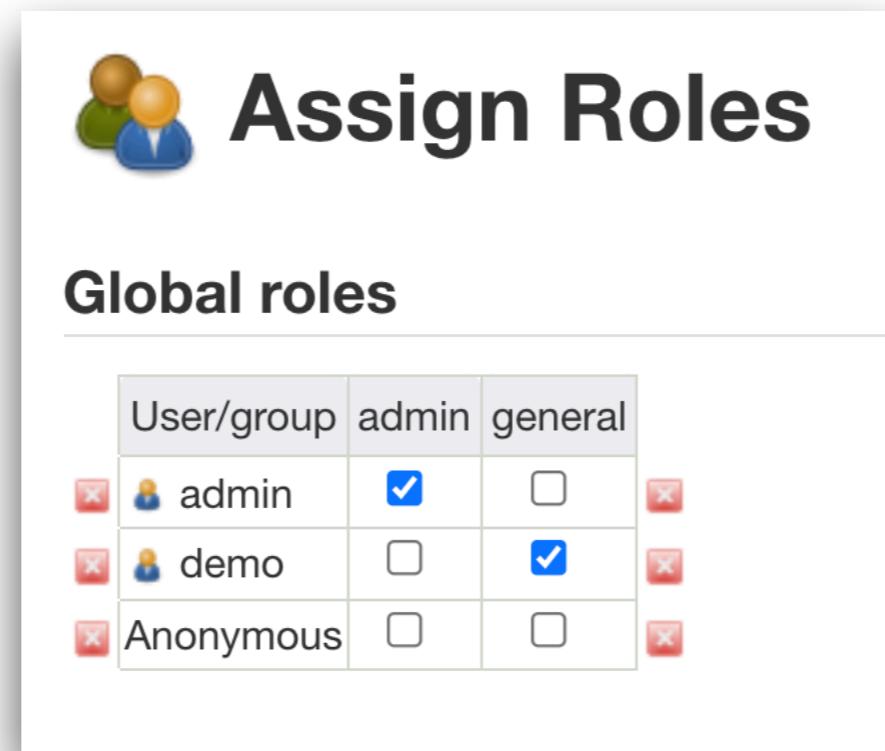
### Global roles

Role	Overall		Credentials				Agent								Job											
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Do		
admin	<input checked="" type="checkbox"/>																									
general	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	



# Assign roles to users

Add to global roles



The screenshot shows the Jenkins 'Assign Roles' configuration page. At the top, there is a logo of three stylized human figures and the title 'Assign Roles'. Below this, the section 'Global roles' is displayed. A table lists users and groups along with their assigned roles: 'admin' (admin, general), 'demo' (general), and 'Anonymous' (none). The 'admin' user has both 'admin' and 'general' roles checked. The 'demo' user has only the 'general' role checked. The 'Anonymous' user has no roles checked.

User/group	admin	general
admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>
demo	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>



**Log out  
and Login with new user**



# Gitlab Authentication plugin



# Gitlab Authentication plugin

## Gitlab Authentication

Documentation

Releases

Issues

Dependencies

The GitLab Authentication Plugin provides a means of using GitLab for authentication and authorization to secure Jenkins. GitLab Enterprise is also supported.

### Setup

Before configuring the plugin you must create a GitLab application registration. In the Scopes section mark **api**.

the authorization callback URL takes a specific value. It must be `http://myserver.example.com:8080/securityRealm/finishLogin` where myserver.example.com:8080 is the location of the Jenkins server.

The Client ID and the Client Secret will be used to configure the Jenkins Security Realm. Keep the page open to the application registration so this information can be copied to your Jenkins configuration.

<https://plugins.jenkins.io/gitlab-oauth/>

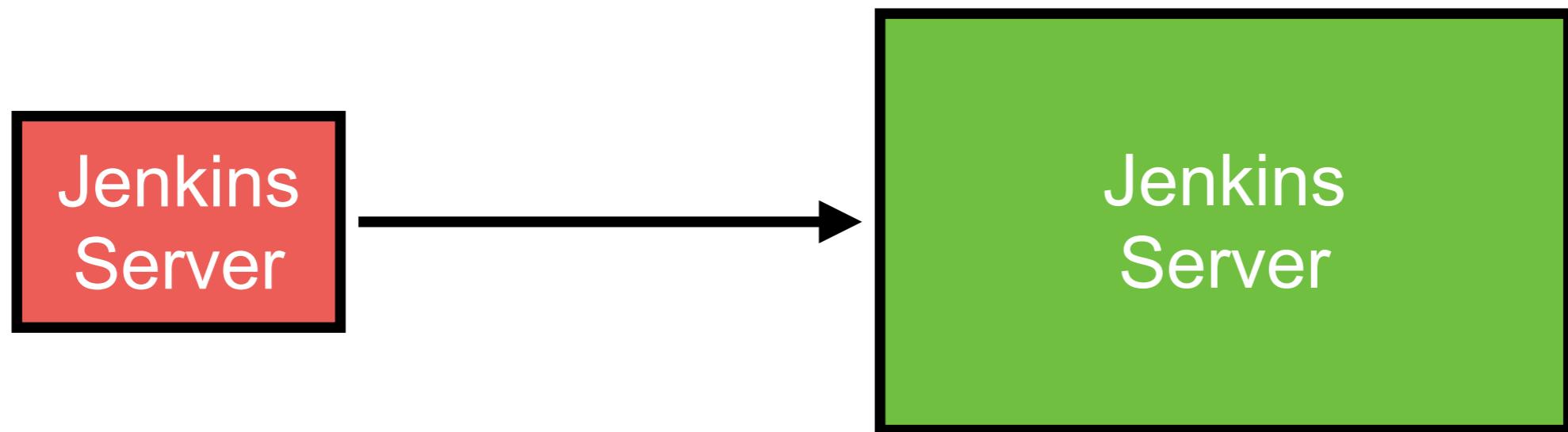


# Scale Jenkins



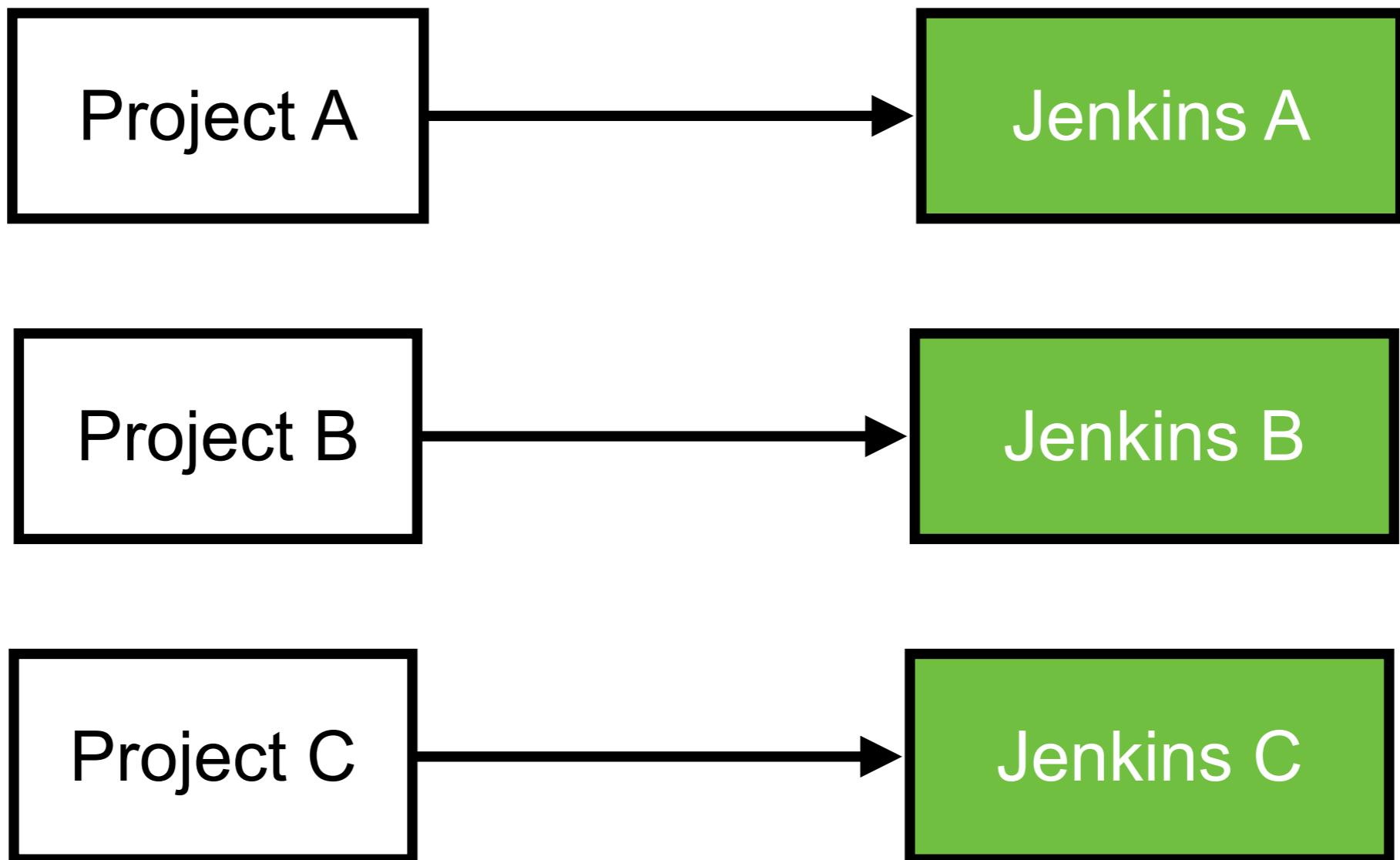
# Scale Jenkins

Add more resources for all projects



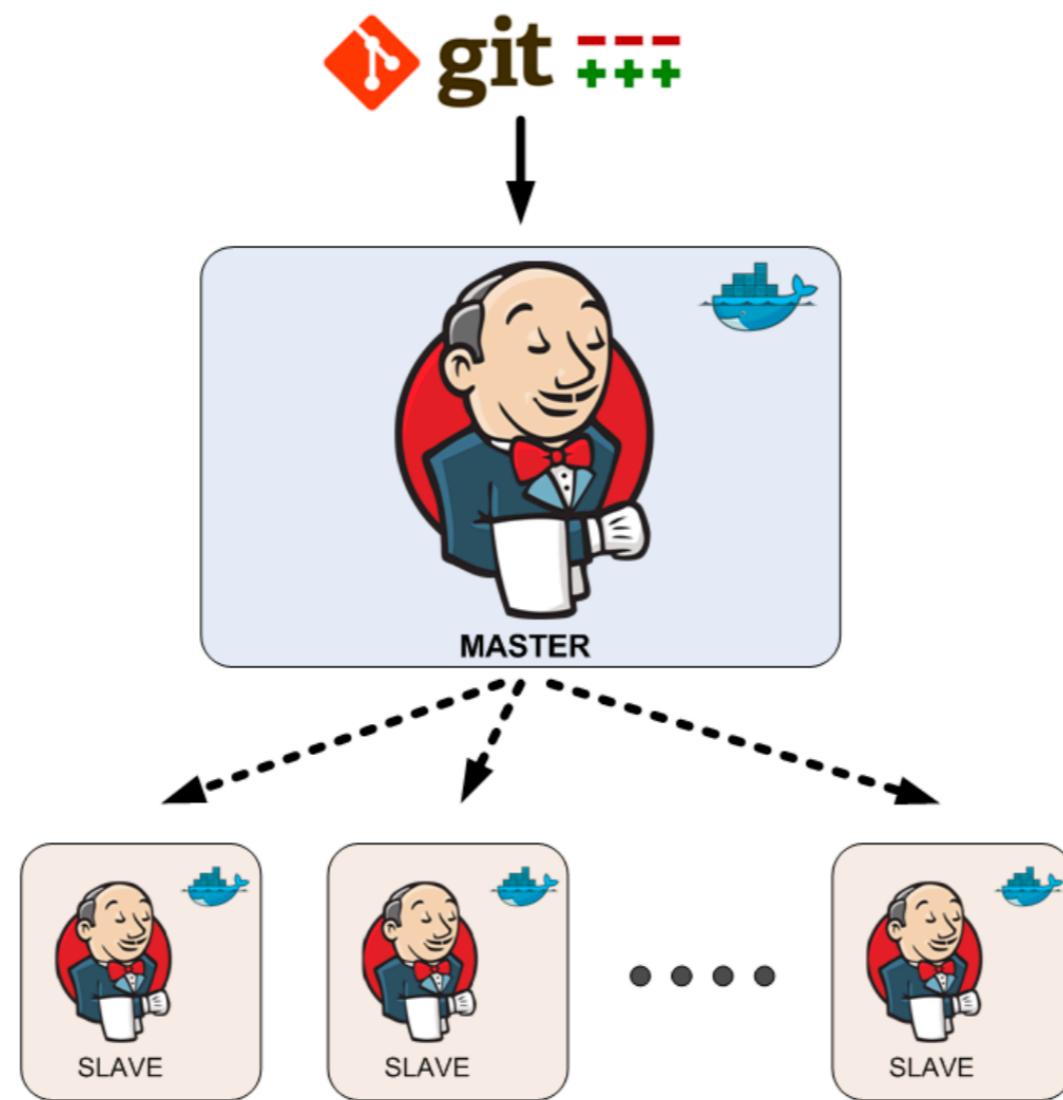
# Scale Jenkins

Jenkins server per project



# Scale Jenkins

Master and Slaves (Add more nodes)

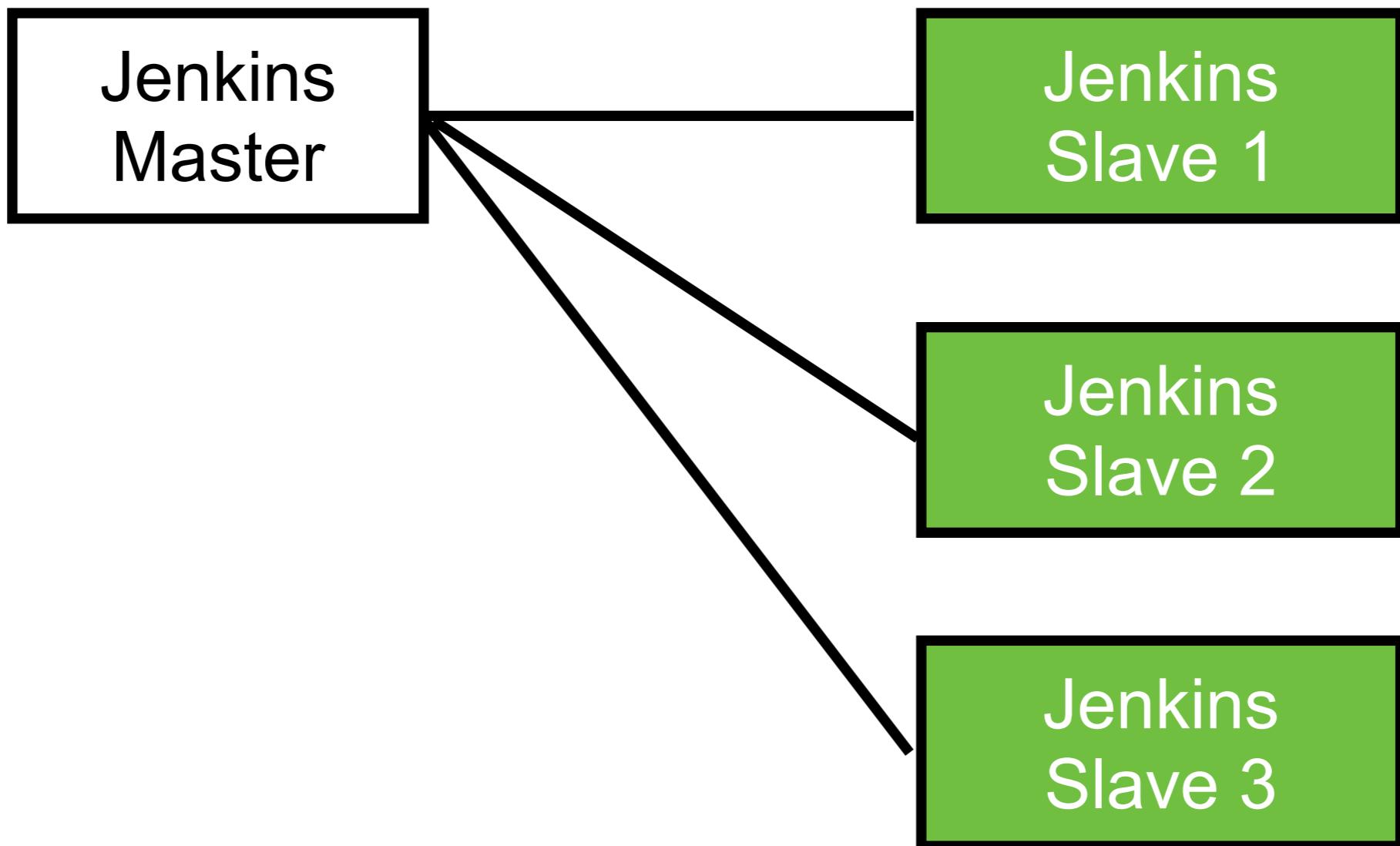


# Add new nodes/agents



# Scale Jenkins

Jenkins server per project



# Create new node

Manage Jenkins -> Manage nodes and clouds

## System Configuration

 **Configure System**  
Configure global settings and paths.

 **Global Tool Configuration**  
Configure tools, their locations and automatic installers.

 **Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.  
⚠ There are updates available

 **Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

## Security

 **Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.

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

 **Manage Credentials**  
Configure credentials

 **Configure Credential Providers**  
Configure the credential providers and types



# Create new node

## Detail of new node

Name	<input type="text" value="worker01"/> <b>worker01</b>
Description	<input type="text"/>
Number of executors	<input type="text" value="1"/>
Remote root directory	<input type="text" value="/var/jenkins_home"/> <b>/var/jenkins_home</b>
Labels	<input type="text" value="worker01"/>



# Launch methods

## By agent from JNLP/SSH

**Launch method**

- Launch agent by connecting it to the master
  - Launch agent via execution of command on the master
  - Launch agents via SSH

Custom WorkDir path



# Start node with JNLP

## How to start a new agent/node ?



### Agent worker01

Connect agent to Jenkins one of these ways:

- [Java Web Start is not available for the JVM version running Jenkins](#)
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://165.22.244.58:8080/computer/worker01/jenkins-agent.jnlp -secret  
276c622e46b2b0cd0efd93c97b0e51b84b47b2e1c51f11858aae67b01bd68681 -workDir "/var/jenkins_home"
```

Run from agent command line, with the secret stored in a file:

```
echo 276c622e46b2b0cd0efd93c97b0e51b84b47b2e1c51f11858aae67b01bd68681 > secret-file  
java -jar agent.jar -jnlpUrl http://165.22.244.58:8080/computer/worker01/jenkins-agent.jnlp -secret @secre
```

### Projects tied to worker01

None



# Success with new node

S	Name ↓	Architecture	Clock Difference	Free Disk Space
	<b>master</b>	Linux (amd64)	In sync	51.29 GB
	<b>worker01</b>	Linux (amd64)	In sync	51.29 GB
	<b>Data obtained</b>	<b>1 sec</b>	<b>0.98 sec</b>	<b>0.92 sec</b>



# Working with SonarQube



# SonarQube

The screenshot shows the SonarQube homepage. At the top, there's a navigation bar with links for Product, What's New, Documentation, Community, and a prominent blue 'Download' button. A banner at the top of the main content area says 'LTS SonarQube 8.9 LTS: Better than ever Discover Now →'. Below this, a large heading reads 'Your teammate for Code Quality and Code Security'. A subtext below it says 'SonarQube empowers all developers to write cleaner and safer code. Join an Open Community of more than 200k dev teams.' On the left, there's a code editor snippet with annotations. A red box highlights a warning: 'A "NullPointerException" could be thrown; "providedClass" is nullable here.' It also shows a 'Bug Major' issue and a 'cert, cwe' link. On the right, there's a summary card for the 'Quality Gate' which is 'Passed' with 'All conditions passed'. The card also displays metrics for Reliability (0 Bugs), Security (0 Vulnerabilities, 1 Hotspots), and Maintainability (4). The overall theme is light blue and white.

<https://www.sonarqube.org/>



# SonarQube Community version

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects... A

How do you want to create your project?

Are you just testing or have an advanced use-case? Create a project manually.

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.

**i** We recommend setting up a DevOps platform configuration so you and your team can benefit from more SonarQube features.



**From Azure DevOps**  
Global configuration not set



**From Bitbucket**  
Global configuration not set



**From GitHub**  
Global configuration not set



**From GitLab**  
Global configuration not set



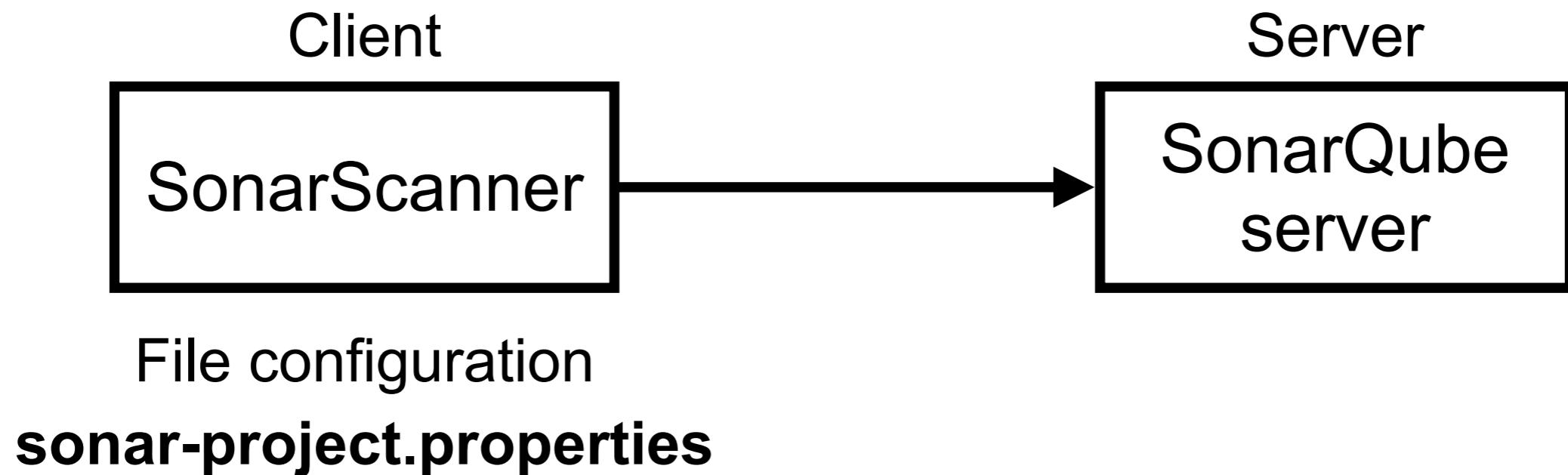
**Manually**

**!** Embedded database should be used for evaluation purposes only  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

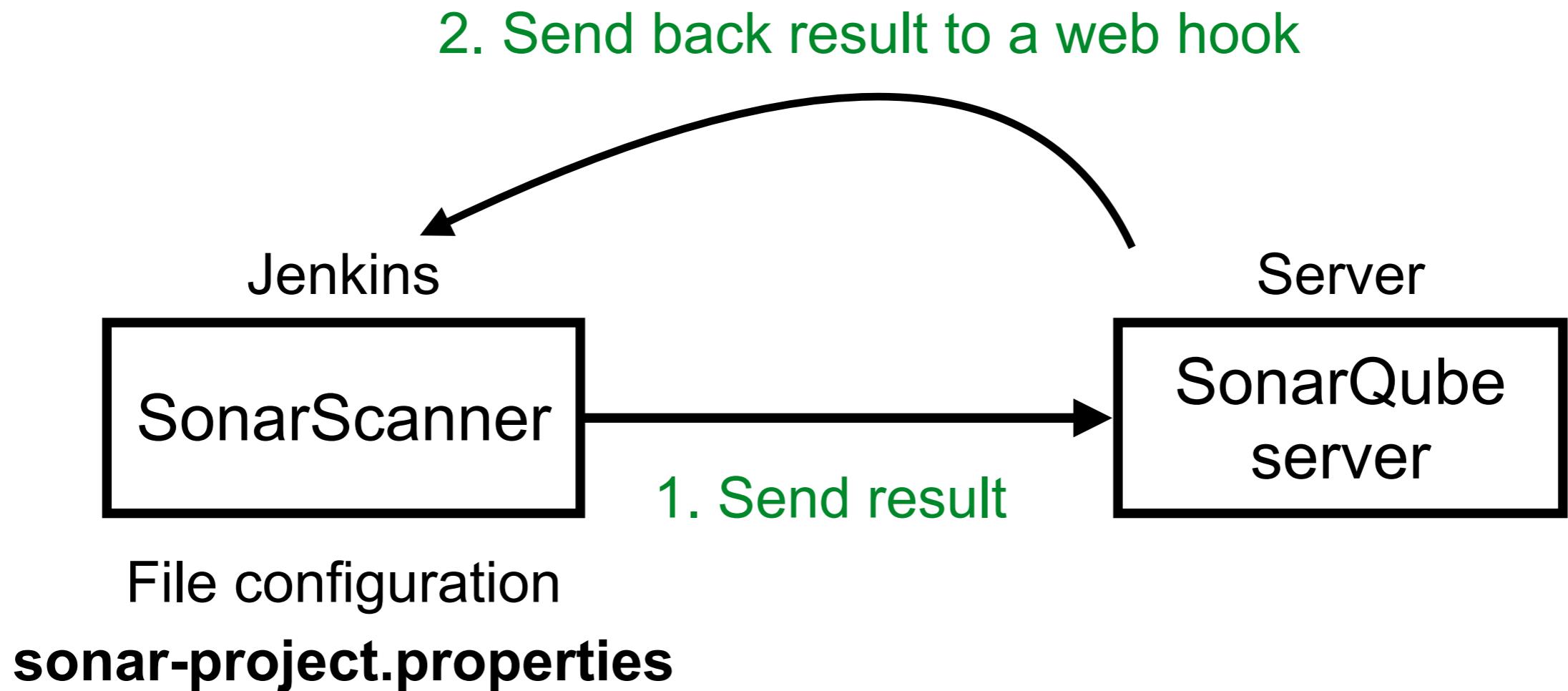
SonarQube™ technology is powered by SonarSource SA  
Community Edition - Version 9.0.1 (build 46107) - LGPL v3 - Community - Documentation - Plugins - Web API - About



# SonarQube and Scanner



# Jenkins + SonarQube



# SonarScanner

**SonarScanner**

---

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

---

**4.6.2** [Show more versions](#)

2021-05-07

Update dependencies, bug fix

[Linux 64-bit](#) [Windows 64-bit](#) [Mac OS X 64-bit](#) [Docker](#)

[Any \(Requires a pre-installed JVM\)](#) [Release notes](#)

The SonarScanner is the scanner to use when there is no specific scanner for your build system.

<https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/>



# Generate secret token

User -> My Account -> Security

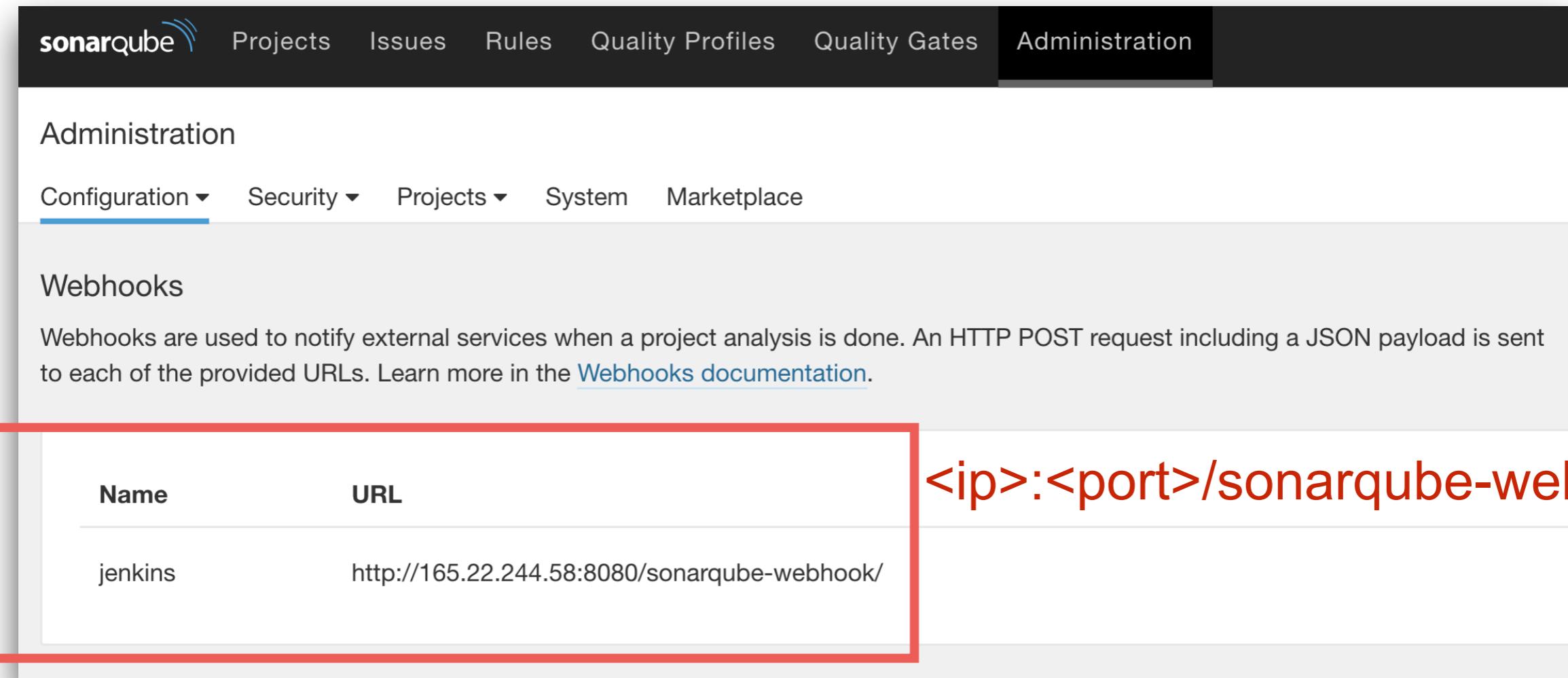
The screenshot shows the 'Tokens' section of the SonarQube 'Security' tab. A green button labeled 'A' is at the top left. The navigation bar includes 'Administrator', 'Profile', 'Security' (which is underlined), 'Notifications', and 'Projects'. The 'Tokens' section has a heading and a descriptive text about using tokens for security instead of user credentials. Below this is a 'Generate Tokens' form with an 'Enter Token Name' input field and a 'Generate' button. A success message box displays a warning icon and the text: 'New token "demo" has been created. Make sure you copy it now, you won't be able to see it again!'. A 'Copy' button next to the token ID '4f2dcaf29fb346a115bf22586b31b67615e0f080' is highlighted. A table lists the token details: Name (demo), Last use (Never), Created (August 25, 2021), and a 'Revoke' button.

Name	Last use	Created	
demo	Never	August 25, 2021	<a href="#">Revoke</a>



# Create SonarQube Webhook

Administration -> Configuration -> Webhooks



The screenshot shows the SonarQube administration interface. The top navigation bar includes links for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration link is currently selected. Below this, a secondary navigation bar shows Configuration (underlined), Security, Projects, System, and Marketplace. The main content area is titled "Administration" and "Webhooks". A descriptive text explains that webhooks notify external services when a project analysis is done via HTTP POST requests with JSON payloads. It links to the "Webhooks documentation". A table lists existing webhooks:

Name	URL
jenkins	http://165.22.244.58:8080/sonarqube-webhook/

A red box highlights the "jenkins" row in the table. To the right of the table, the URL "http://<ip>:<port>/sonarqube-webhook/" is displayed in red text.



# SonarScanner for Jenkins

## SonarScanner for Jenkins

By [SonarSource](#)

GNU LGPL 3

[Issue Tracker](#)

**2.13.1**

[Show more versions](#)

2021-04-30

Update dependencies

[Download](#) [Release notes](#)

This plugin lets you centralize the configuration of SonarQube server connection details in Jenkins global configuration.

Then you can trigger SonarQube analysis from Jenkins using standard Jenkins Build Steps or [Jenkins Pipeline DSL](#) to trigger analysis with:

- [SonarScanner](#)
- [SonarScanner for Maven](#)
- [SonarScanner for Gradle](#)
- [SonarScanner for .NET](#)

<https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/>



# SonarScanner for Jenkins

## SonarQube Scanner

Documentation

Releases

Issues

Dependencies

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Server authentication token stored in plain text

Documentation of SonarQube plugin available in SonarQube wiki

<http://redirect.sonarsource.com/plugins/jenkins.html>

Please don't use this page to ask questions or report bugs.

This plugin allow easy integration of **SonarQube™**, the open source platform for Continuous Inspection of code quality.

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



# SonarScanner for Jenkins

## SonarQube Scanner



External Site/Tool Integrations

Build Reports

2.13.1

This plugin allows an easy integration of **SonarQube**, the open source platform for Continuous Inspection of code quality.

## Sonar Quality Gates

Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")



Warning: This plugin version may not be safe to use. Please review the following security notices:

1.3.1

- Credentials transmitted in plain text

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



# Config SonarQube Server

Manage Jenkins -> Configure System

## SonarQube servers

**Environment variables** 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

 **This property is mandatory.**

Server URL

Default is http://localhost:9000

Server authentication token

- none -   Add 

SonarQube authentication token. Mandatory when anonymous access is disabled.



# Config SonarQube Scanner

Manage Jenkins -> Global tool configuration

SonarQube Scanner

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name **scanner**

Required

Install automatically [?](#)

Install from Maven Central

Version **SonarQube Scanner 4.6.2.2472**

[Delete Installer](#)

[Add Installer ▾](#)

[Delete SonarQube Scanner](#)



# Pipeline in Jenkins

```
stage('Sonarqube') {
```

## Reference to SonarQube Scanner

```
    environment {
        scannerHome = tool 'scanner'
    }
    steps {
        withSonarQubeEnv('sonarqube') {
            sh "${scannerHome}/bin/sonar-scanner"
        }
        timeout(time: 10, unit: 'MINUTES') {
            waitForQualityGate abortPipeline: true
        }
    }
}
```



# Pipeline in Jenkins

```
stage('Sonarqube') {  
  
    environment {  
        scannerHome = tool 'scanner'  
    }  
    steps {  
        withSonarQubeEnv('sonarqube') {  
            sh "${scannerHome}/bin/sonar-scanner"  
        }  
        timeout(time: 10, unit: 'MINUTES') {  
            waitForQualityGate abortPipeline: true  
        }  
    }  
}
```

**Use config of SonarQube Server**



# Pipeline in Jenkins

```
stage('Sonarqube') {  
  
    environment {  
        scannerHome = tool 'scanner'  
    }  
    steps {  
        withSonarQubeEnv('sonarqube') {  
            sh "${scannerHome}/bin/sonar-scanner"  
        }  
        timeout(time: 10, unit: 'MINUTES') {  
            waitForQualityGate abortPipeline: true  
        }  
    }  
}
```

**Waiting result from SonarQube Server  
via webhook**



# Result of SonarQube via webhook

## SonarQube Quality Gate

demo01 Passed

server-side processing: Success

## Permalinks

- [Last build \(#43\), 17 sec ago](#)
- [Last stable build \(#43\), 17 sec ago](#)
- [Last successful build \(#43\), 17 sec ago](#)
- [Last failed build \(#40\), 24 min ago](#)
- [Last unsuccessful build \(#42\), 6 min 27 sec ago](#)
- [Last completed build \(#43\), 17 sec ago](#)



# Using Docker

```
docker container run \
    --rm \
    -e SONAR_HOST_URL="${SONAR_URL}" \
    -e SONAR_LOGIN="${SONAR_SECRET}" \
    -v ${SOURCE_PATH}:/usr/src \
    sonarsource/sonar-scanner-cli
```

<https://hub.docker.com/r/sonarsource/sonar-scanner-cli>

