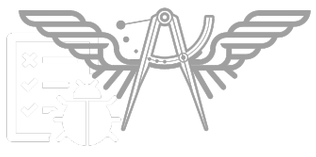
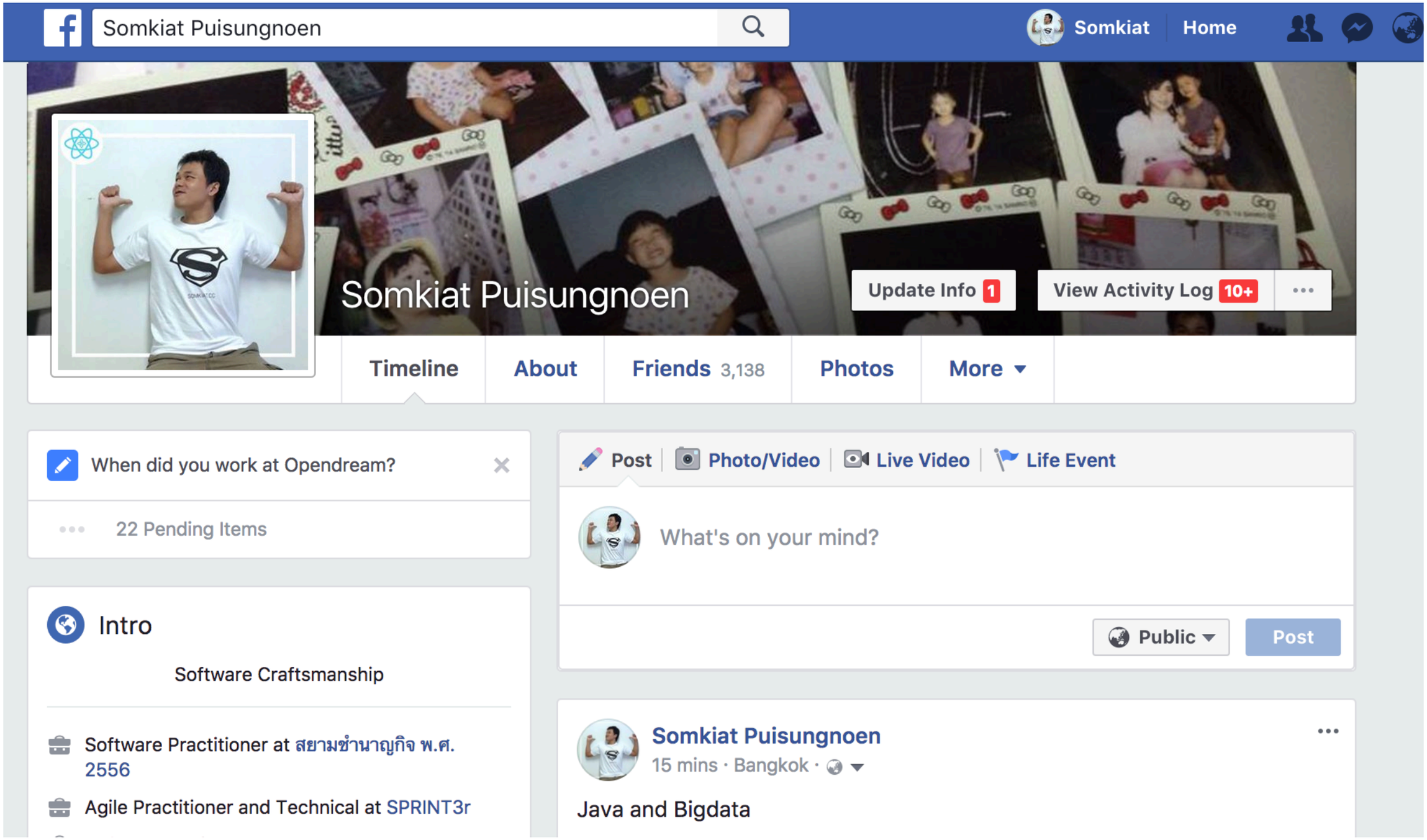
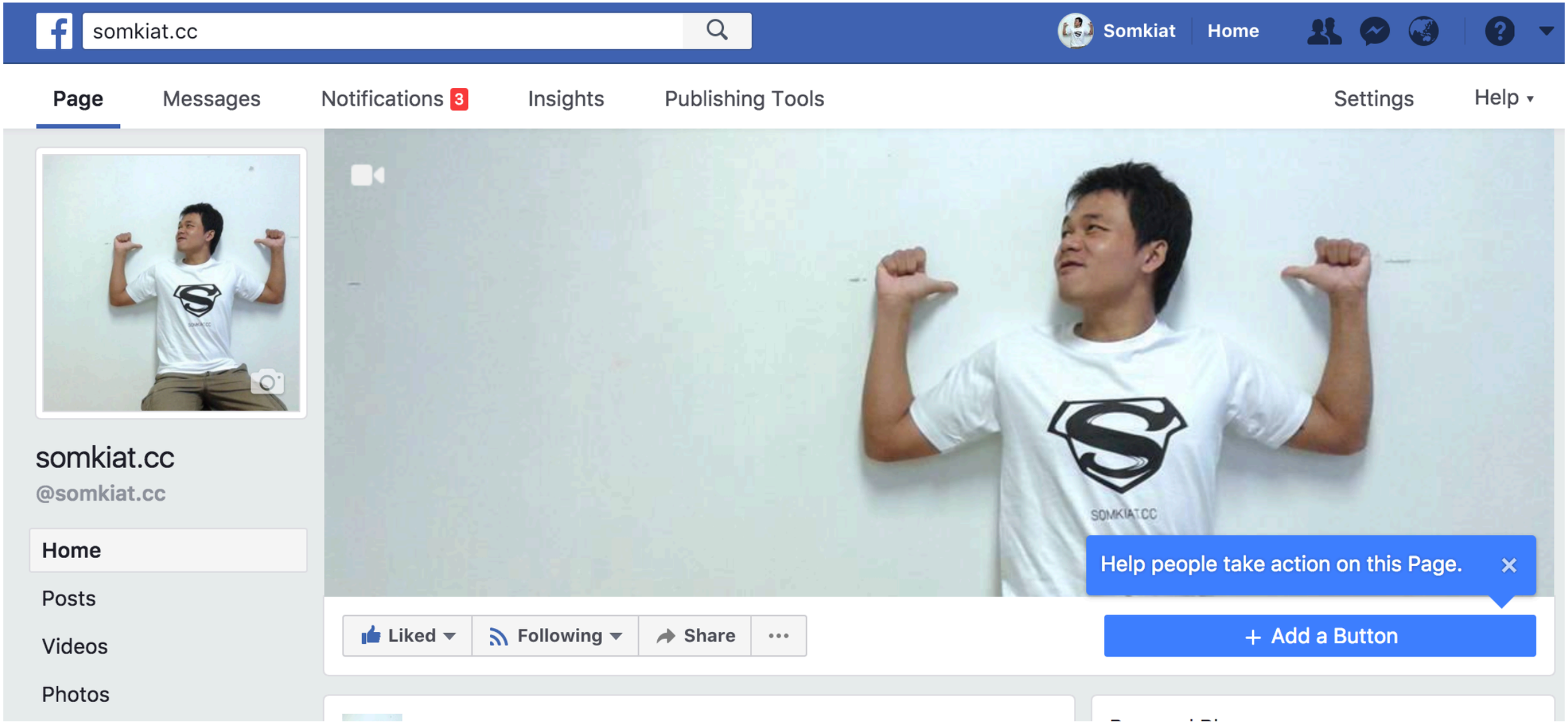
สยามชำนาญกิจ
SIAM CHAMNAN KIT

CI CD with Jenkins Workshop







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

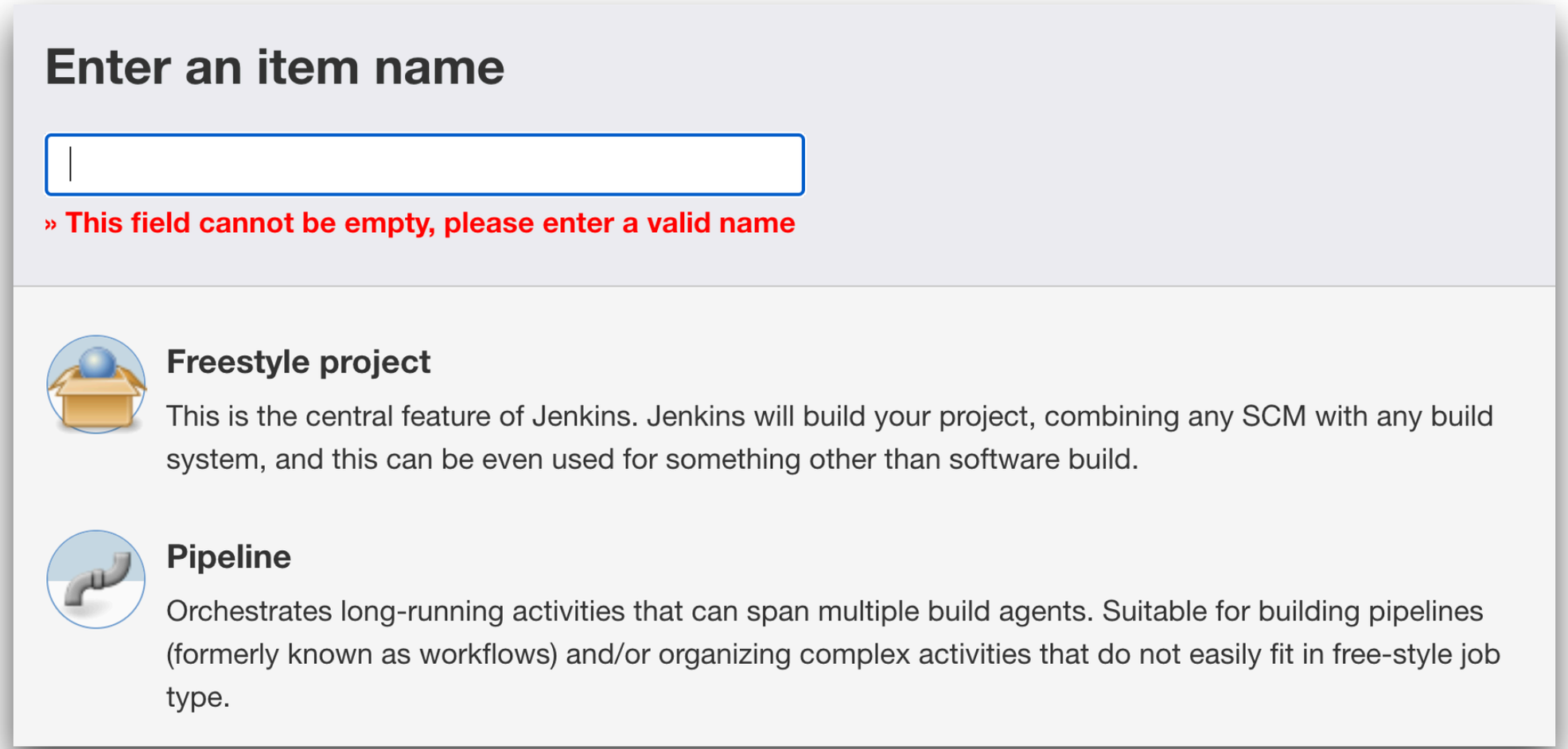
Working with Docker

Docker image
Blue Ocean plugins
Master and slave of Jenkins

Create first job

Create first job

Freestyle job
Pipeline as a Code



Freestyle Job

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

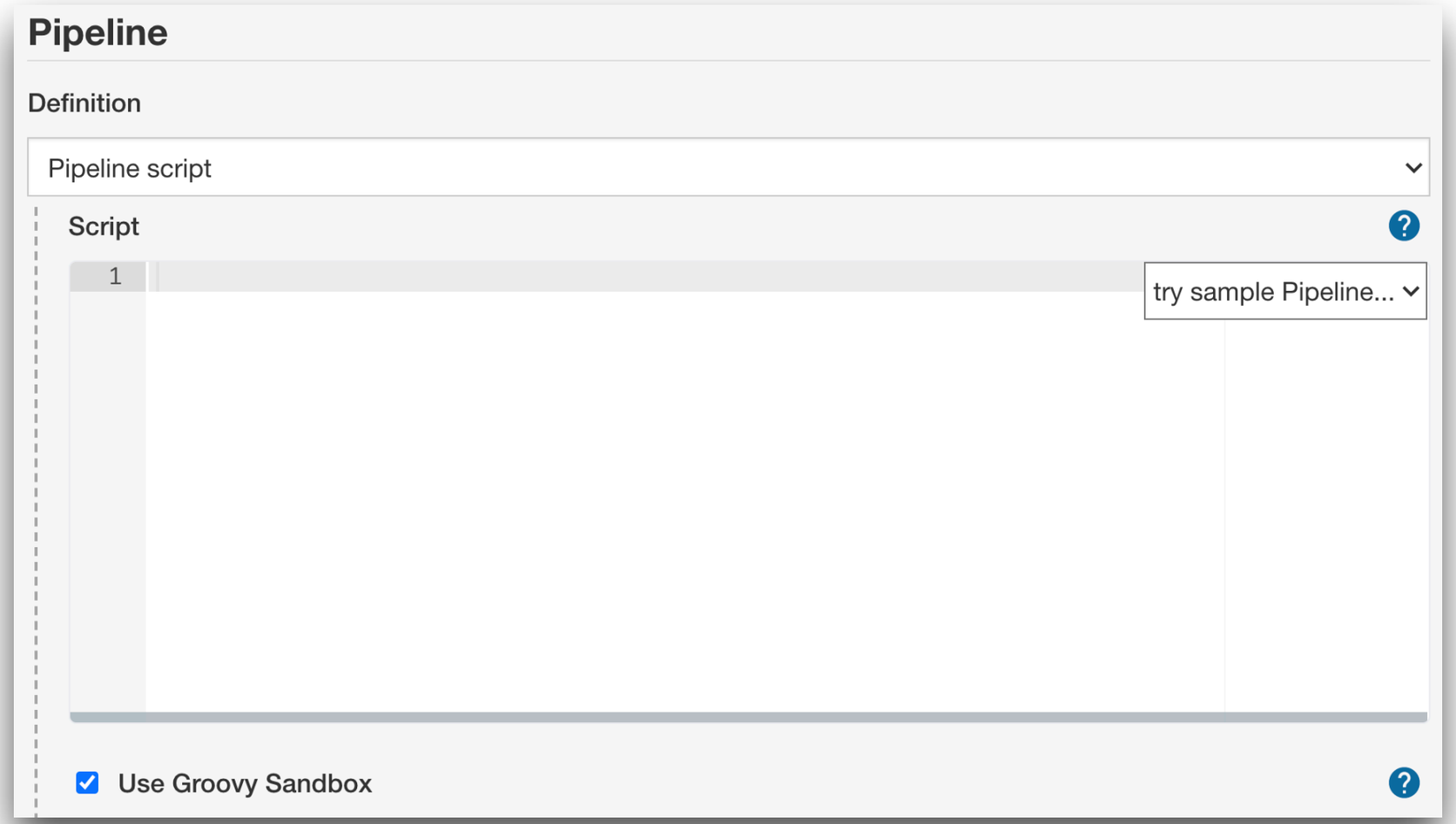
Pipeline as a Code

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

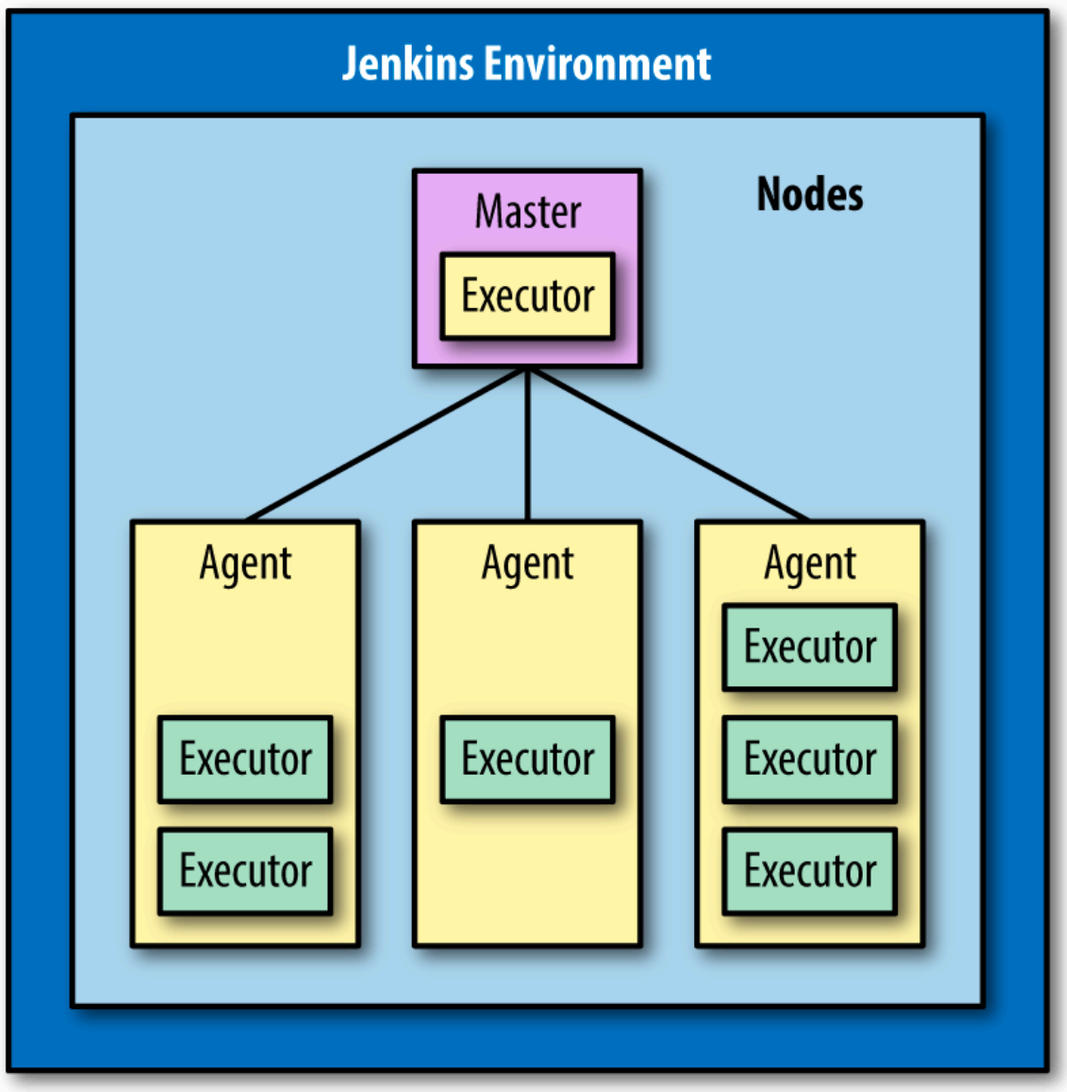
```
node('worker_node1') {
    stage('Source') {
        git ''
    }
    stage('Build') {
        sh ""
    }
}
```

Declarative pipeline

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

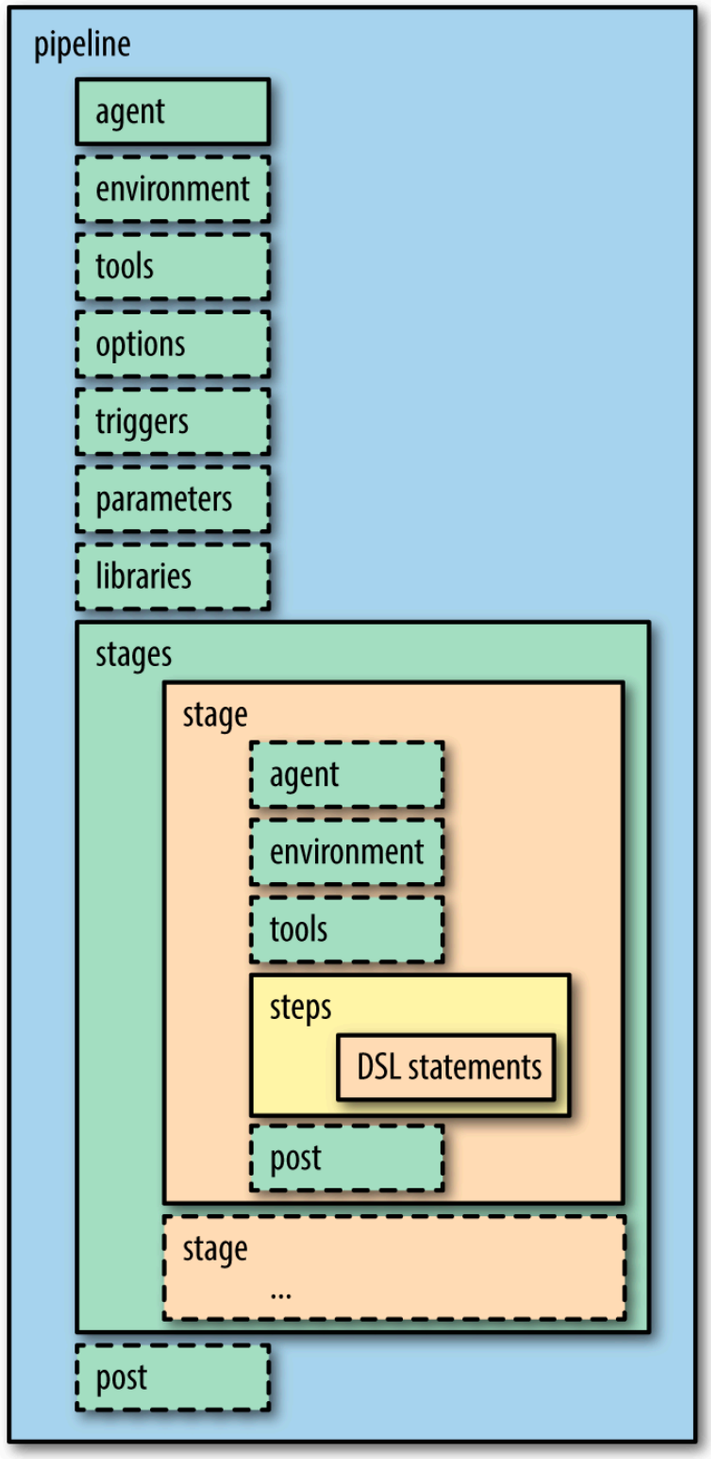
Basic of Jenkins

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

Target of Node/Agent of Jenkins

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

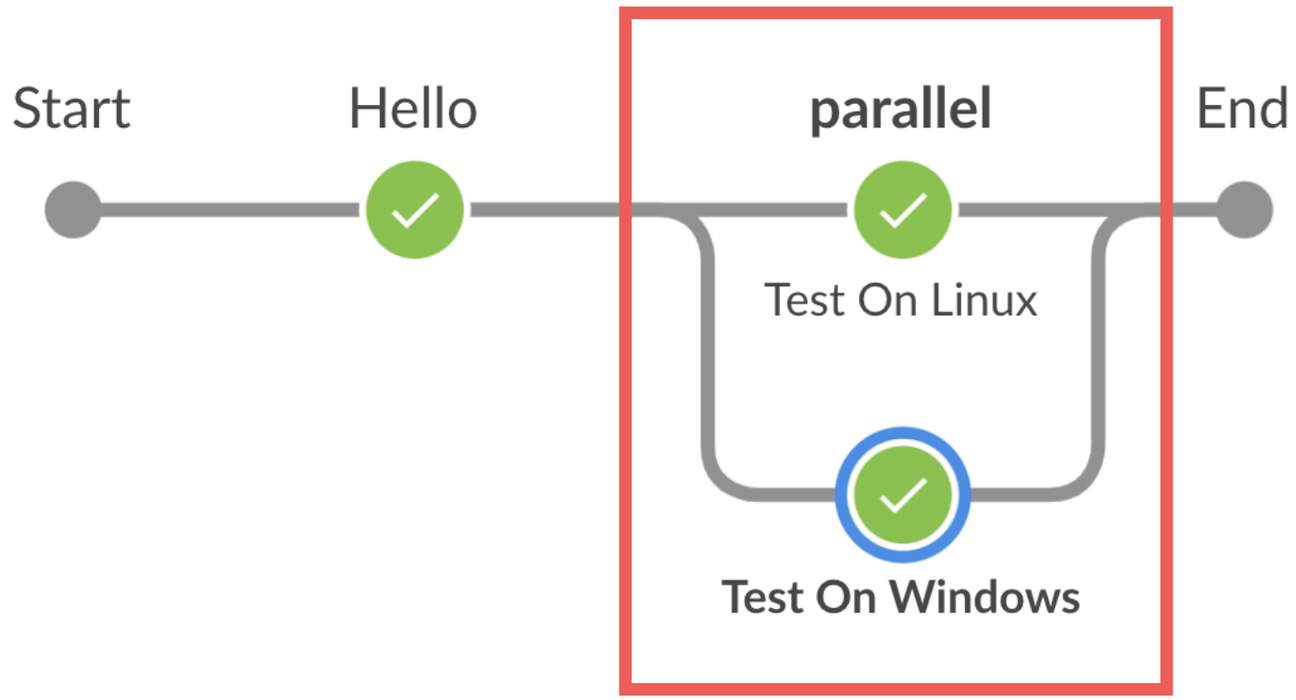
Post-conditions

Condition name	Description
always	Always execute the steps in the block
changed	If the current build's status is different from the previous build's status
success	If the current build was successful
failure	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

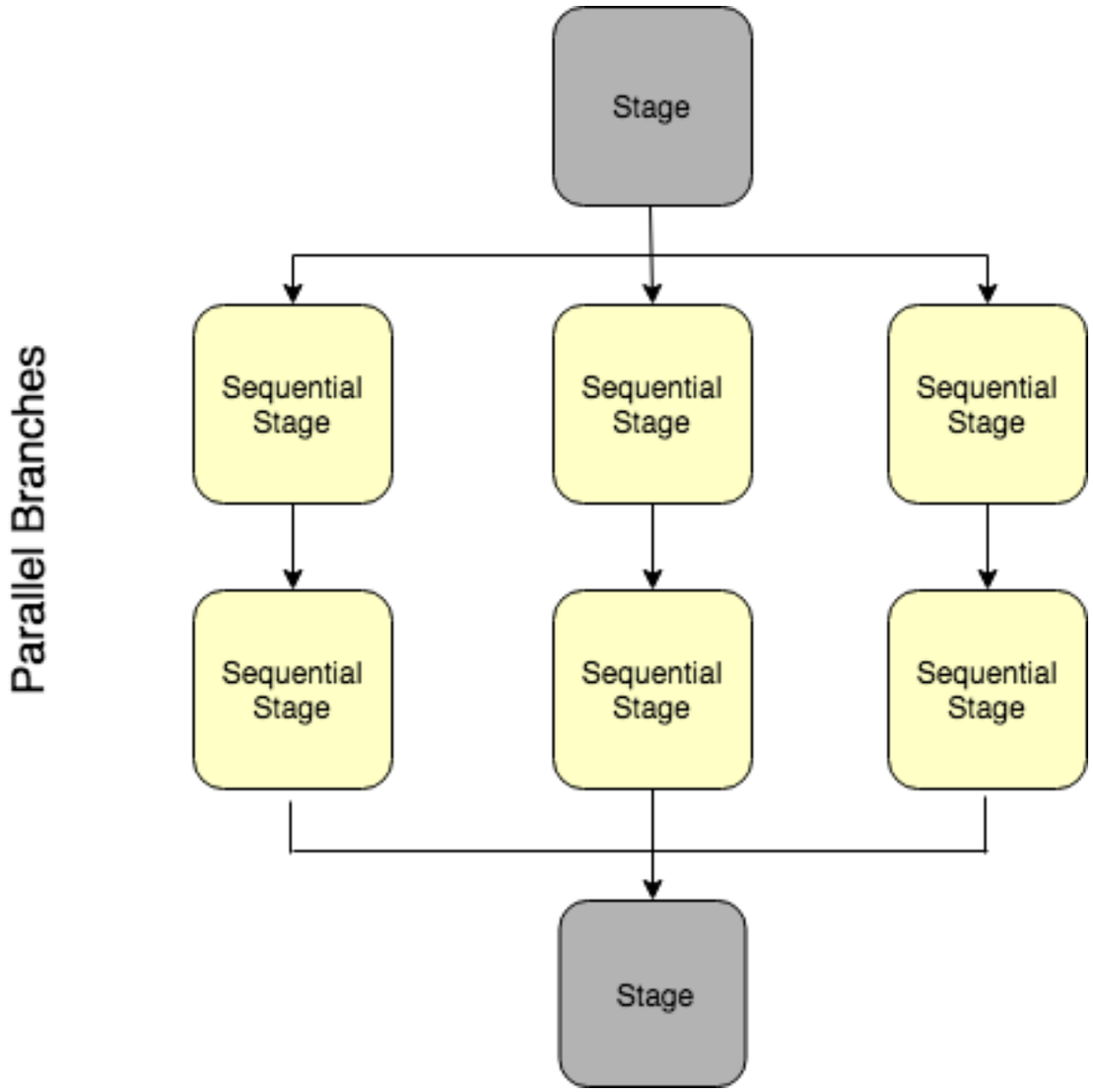
Parallel stages



Parallel stages

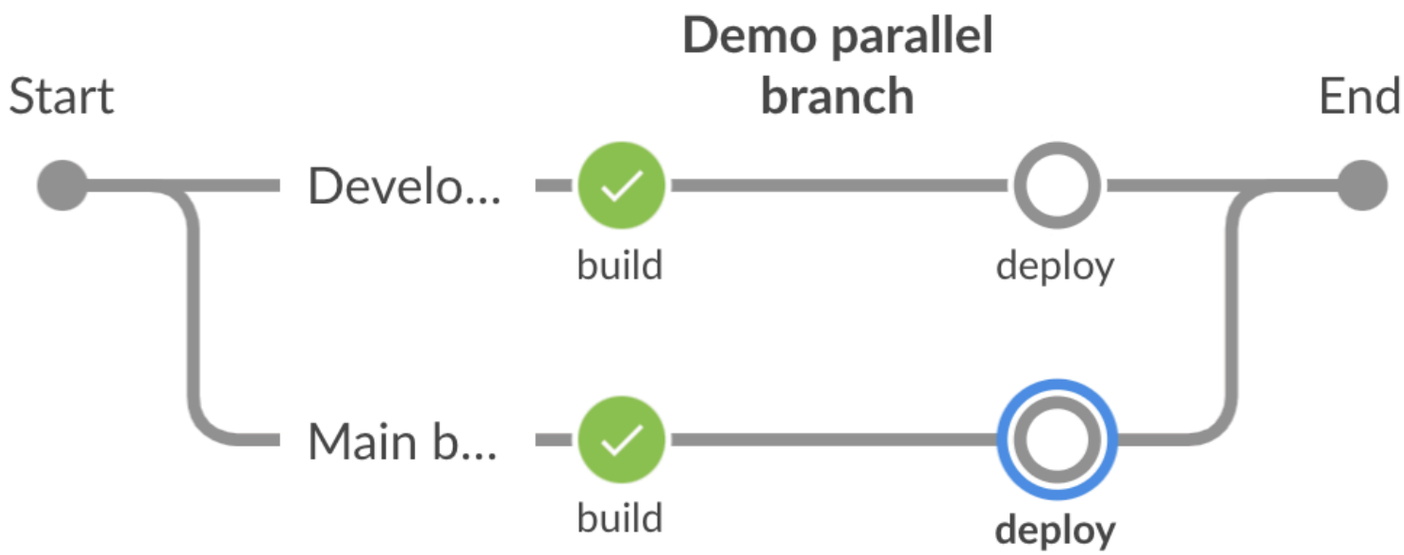
```
stage('parallel') {
    parallel {
        stage('Test On Windows') {
            steps {
                echo "Test On Windows"
            }
        }
        stage('Test On Linux') {
            steps {
                echo "Test On Linux"
            }
        }
    }
}
```

Parallel branches



https://www.jenkins.io/blog/2018/07/02/whats-new-declarative-piepline-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 {
        }
    }
```

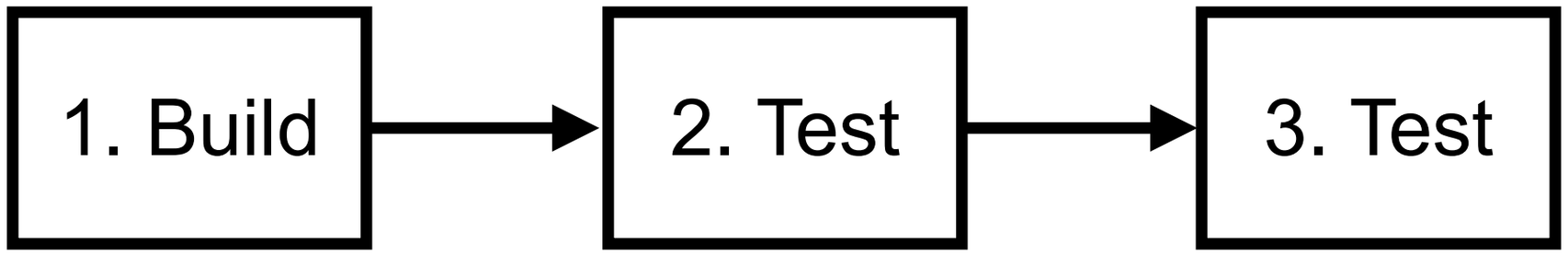
Global env for all

Use environment variables

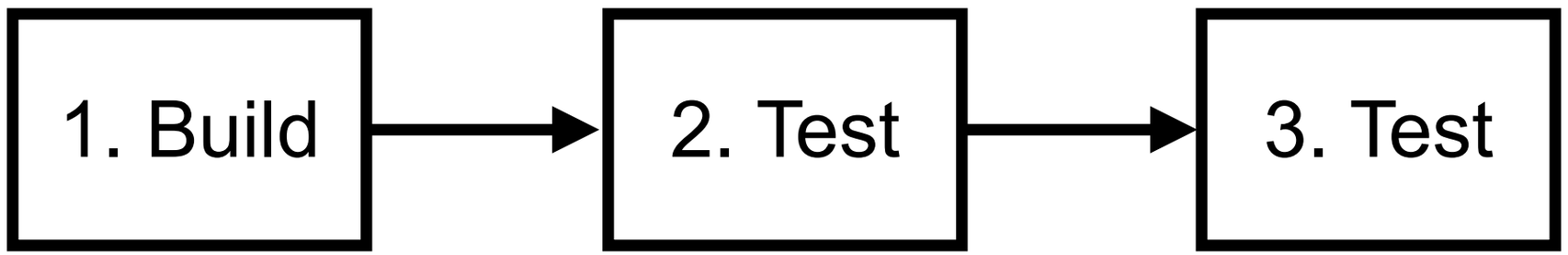
```
stages {
    stage ('Preparation') {

        environment {
            JENKINS_PATH = sh(script: 'pwd', , returnStdout: true).trim()
        }

        steps {
            echo "Hello world"
            echo "PATH=${JENKINS_PATH}"
            sh 'echo “JP=$JENKINS_PATH"'
        }
    }
}
```

Pipeline parameters

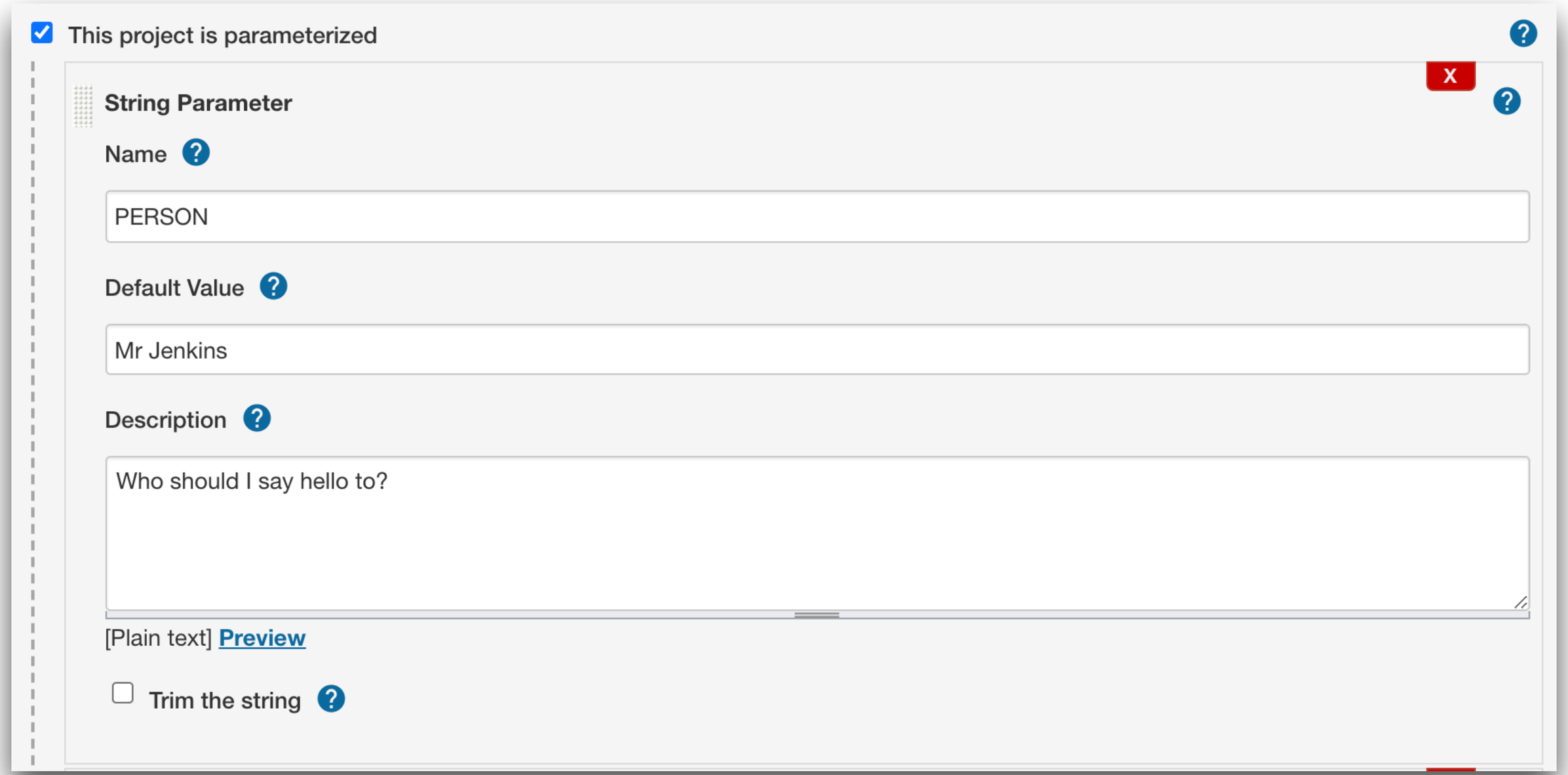
Parameters of pipeline



Branch ?
Tag ?
Version ?



Configure job/item in UI



Parameters directive

List of parameters that provide when trigger the pipeline

String, text (multiple line), boolean

Choice, password

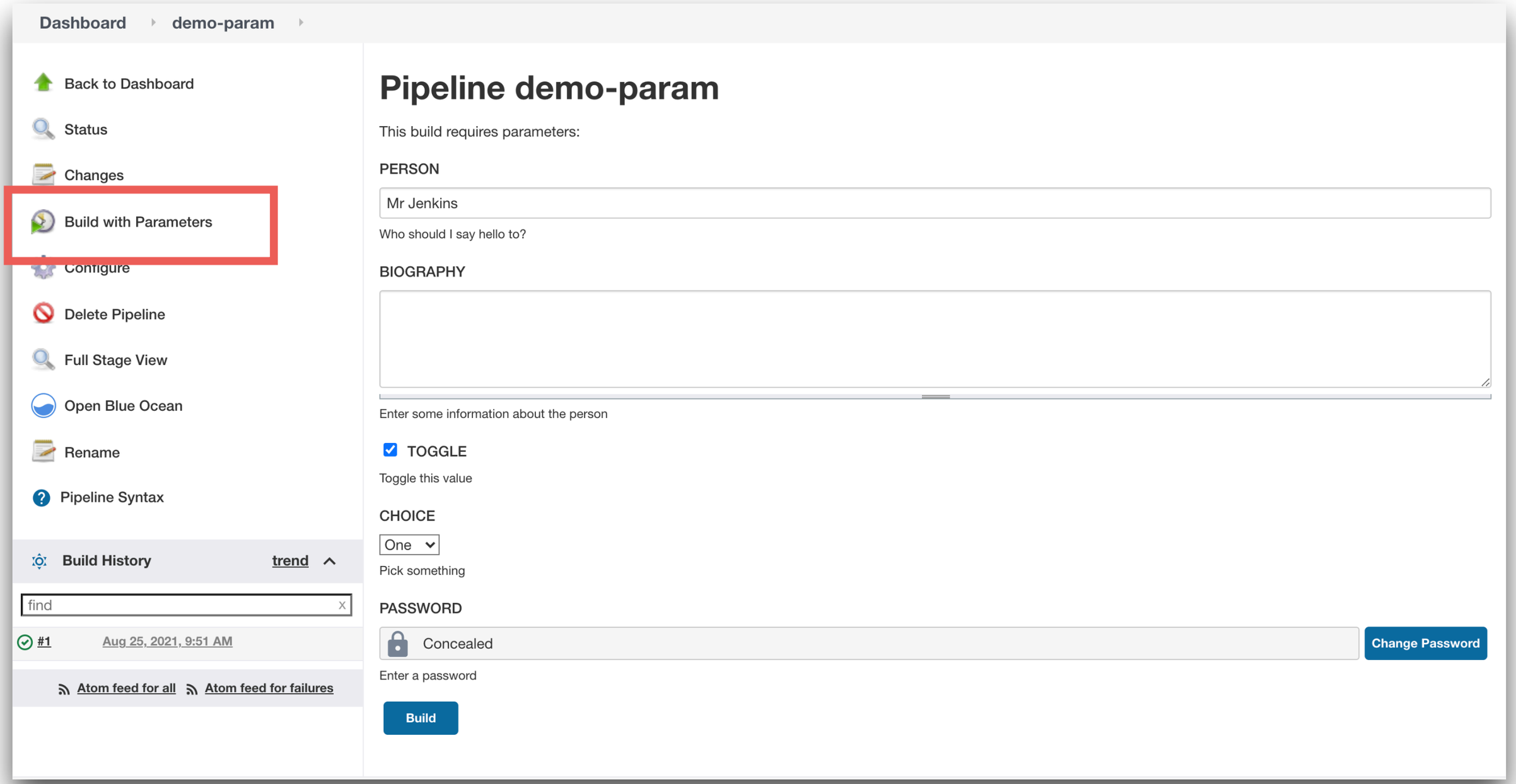
...

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

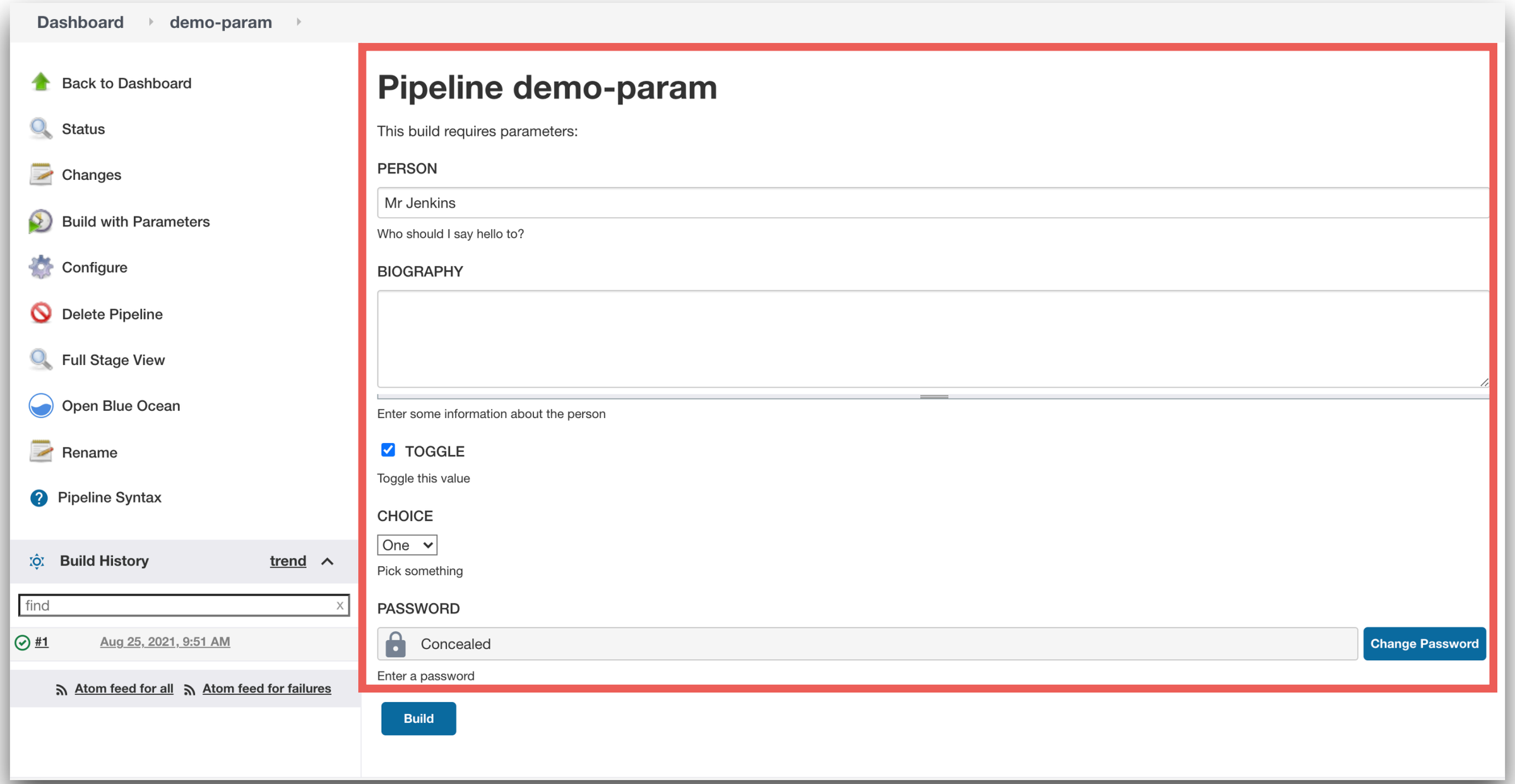
Use parameters directive

```
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



Example



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
...

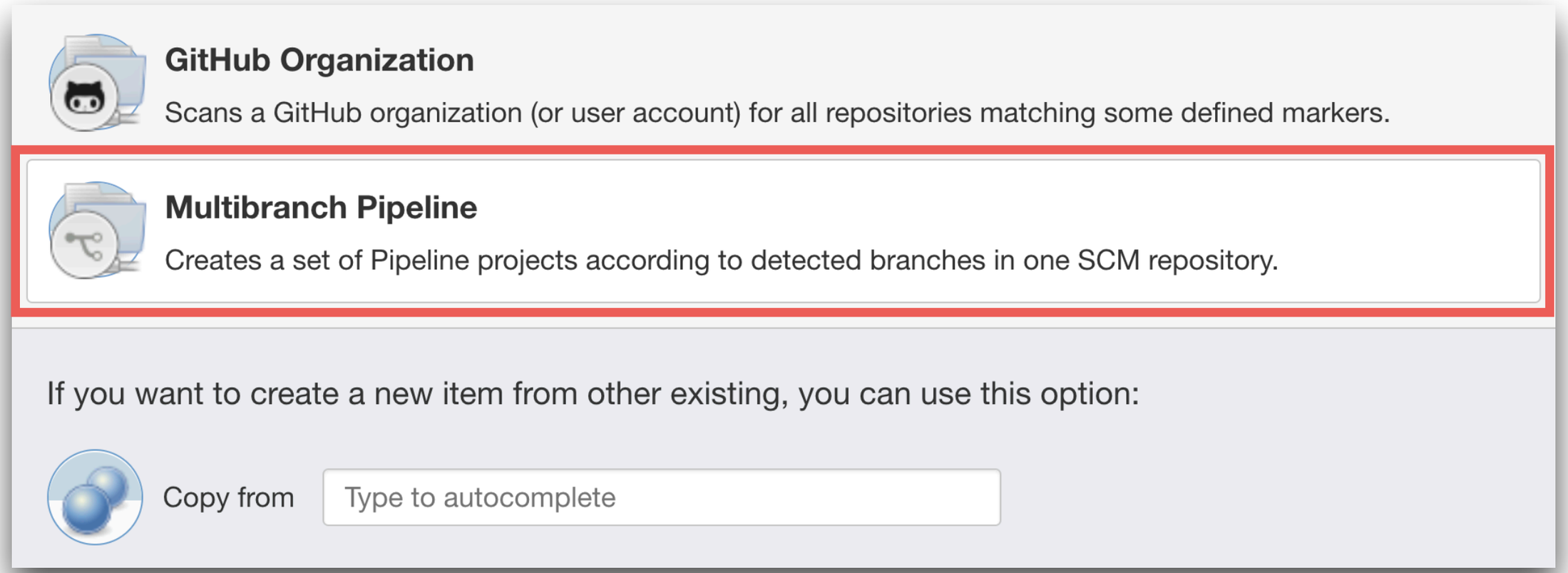
https://www.jenkins.io/doc/book/pipeline/syntax/#when

Multi-branch pipeline

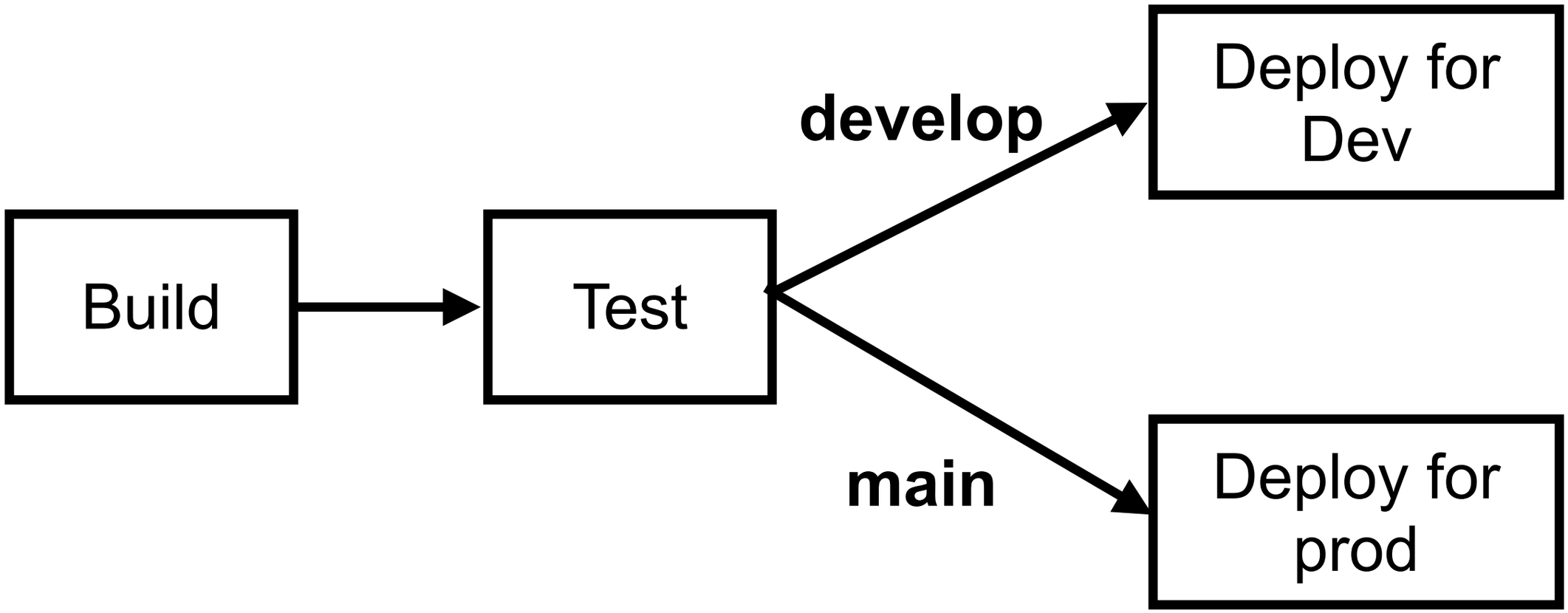
https://www.jenkins.io/doc/book/pipeline/multibranch/

Multi-branches pipeline

Create new item/job



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 ?

Deploy for production - 27s

Restart Test

echo "Deploy to production" — Shell Script <1s

Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input 26s

Finished using the web site? (Click "Proceed" to continue)

Proceed Abort

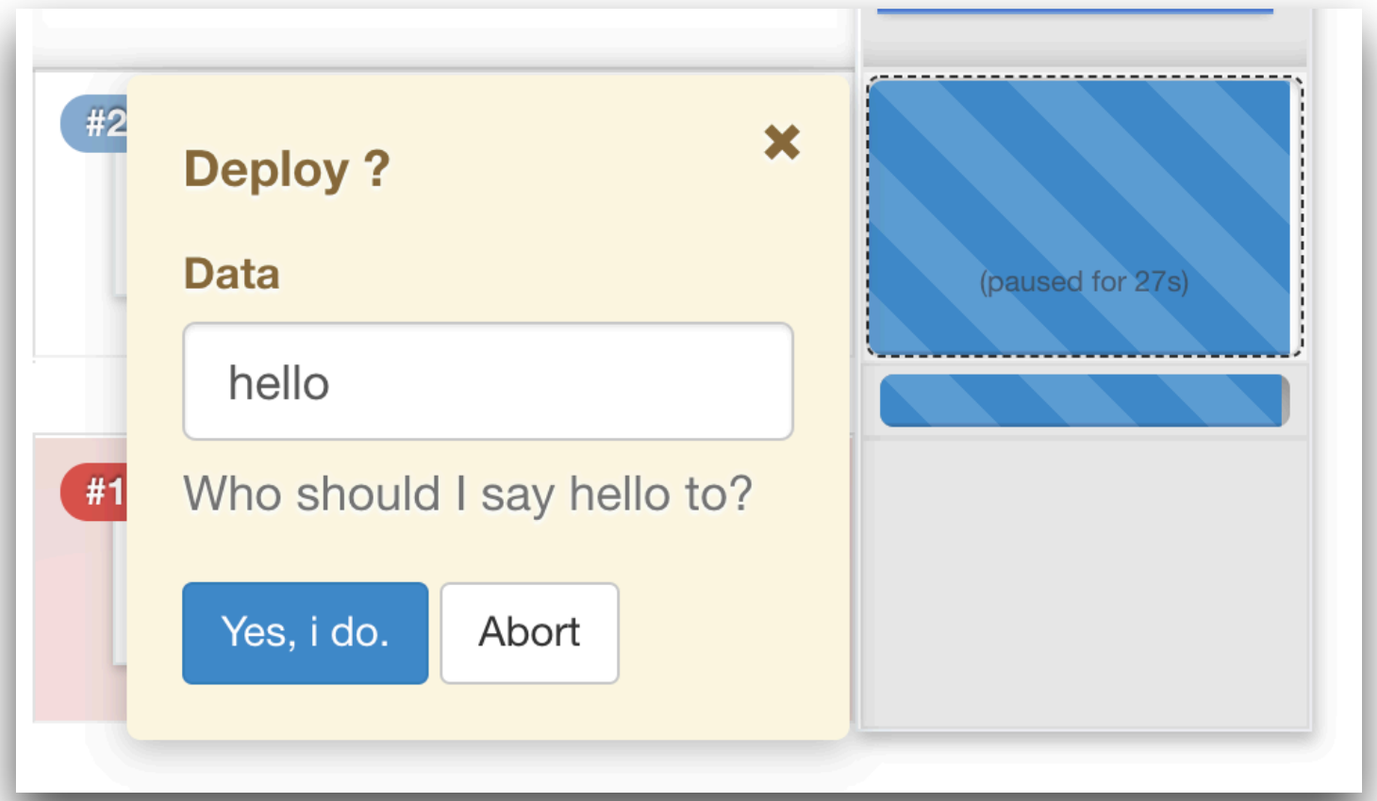
Approve or not ?

```
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

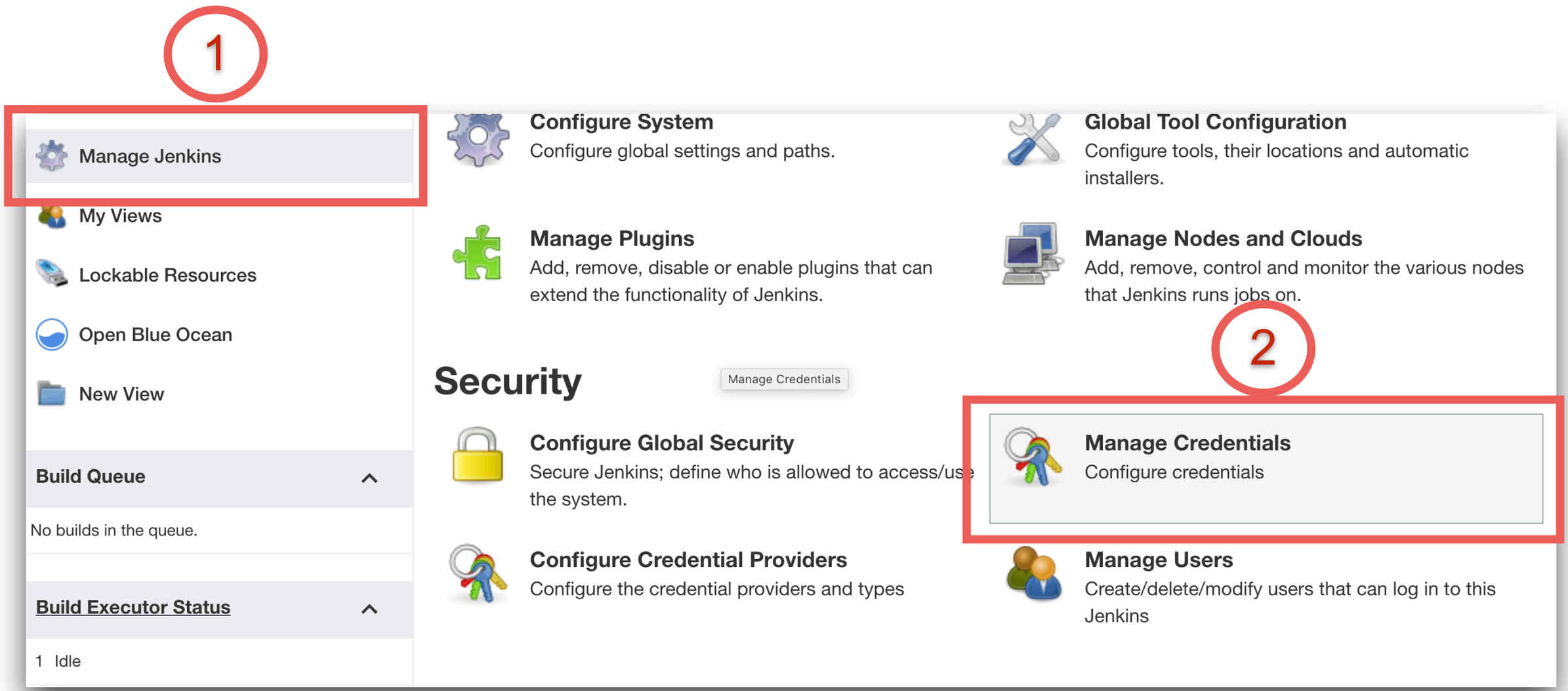
Manage credential

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

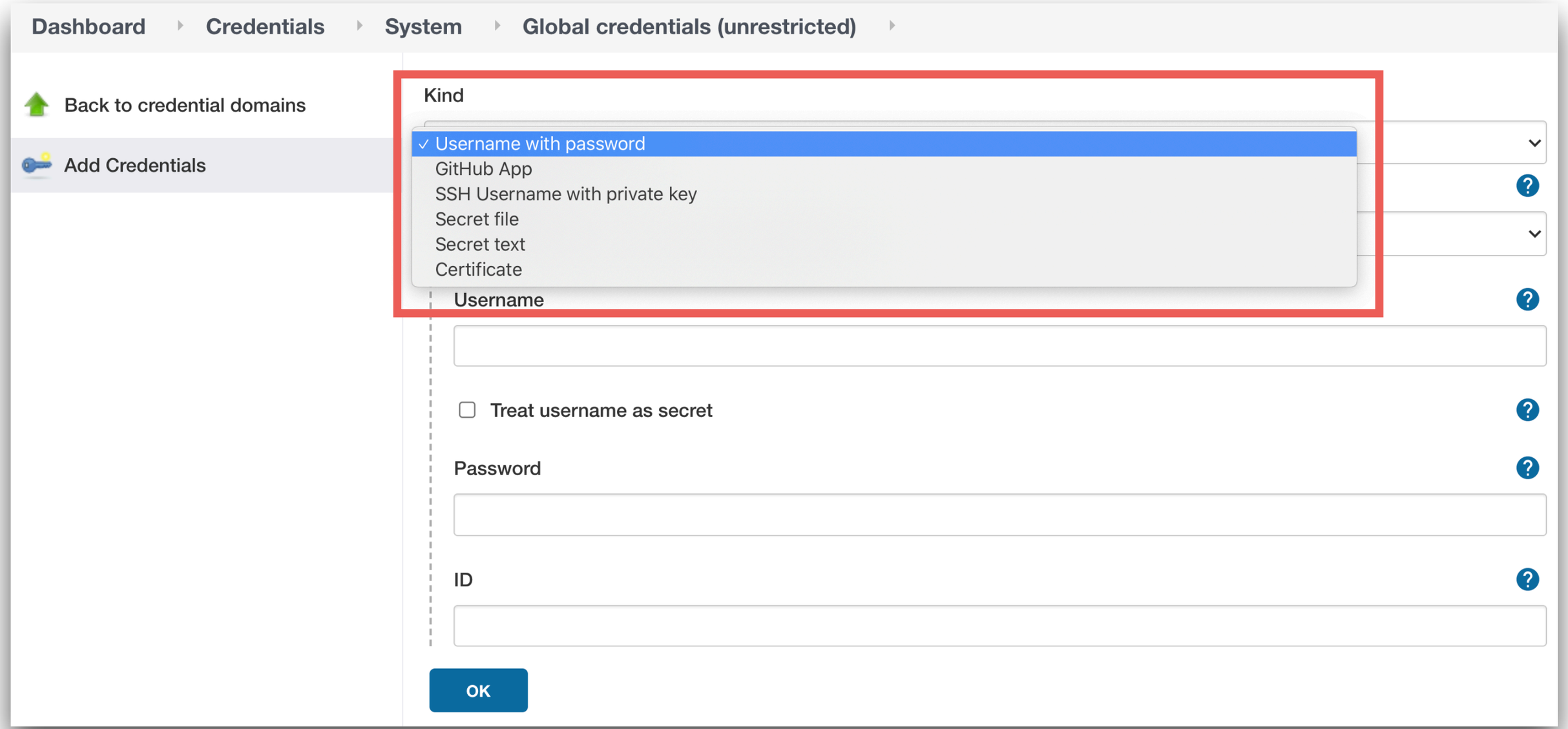


Create credentials

Global credential
By pipeline/job/item
By Jenkins’s user

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

Types of credential



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'

            }
        }
    }
}
```

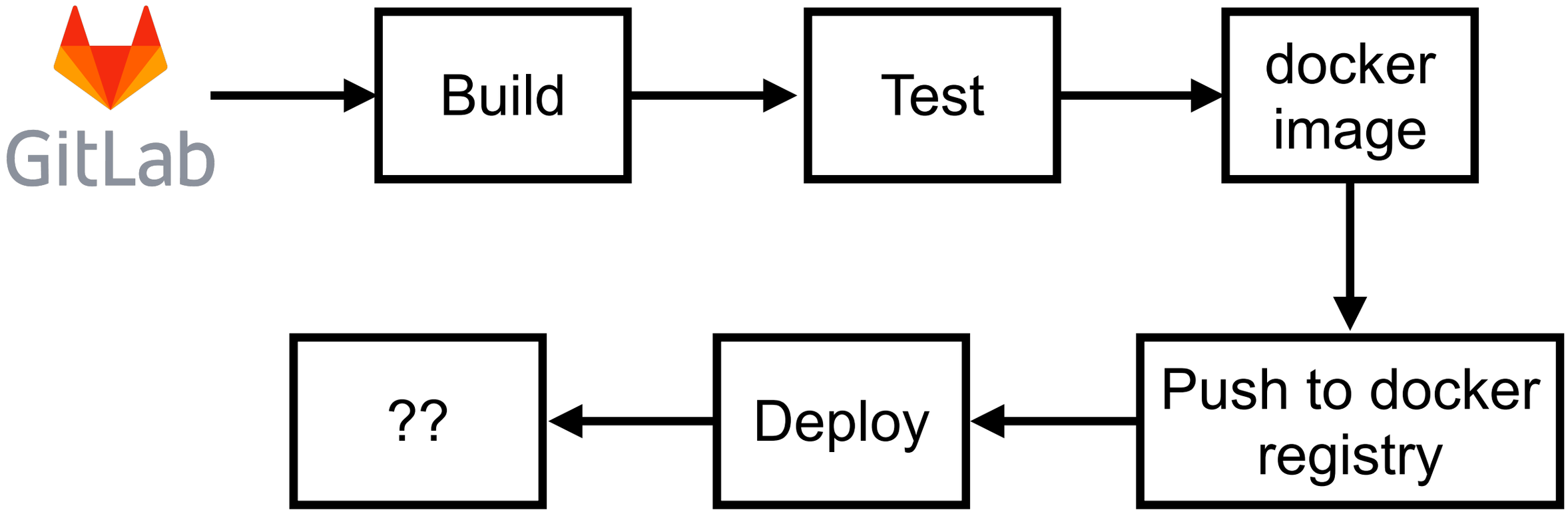
Example of Git login

$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



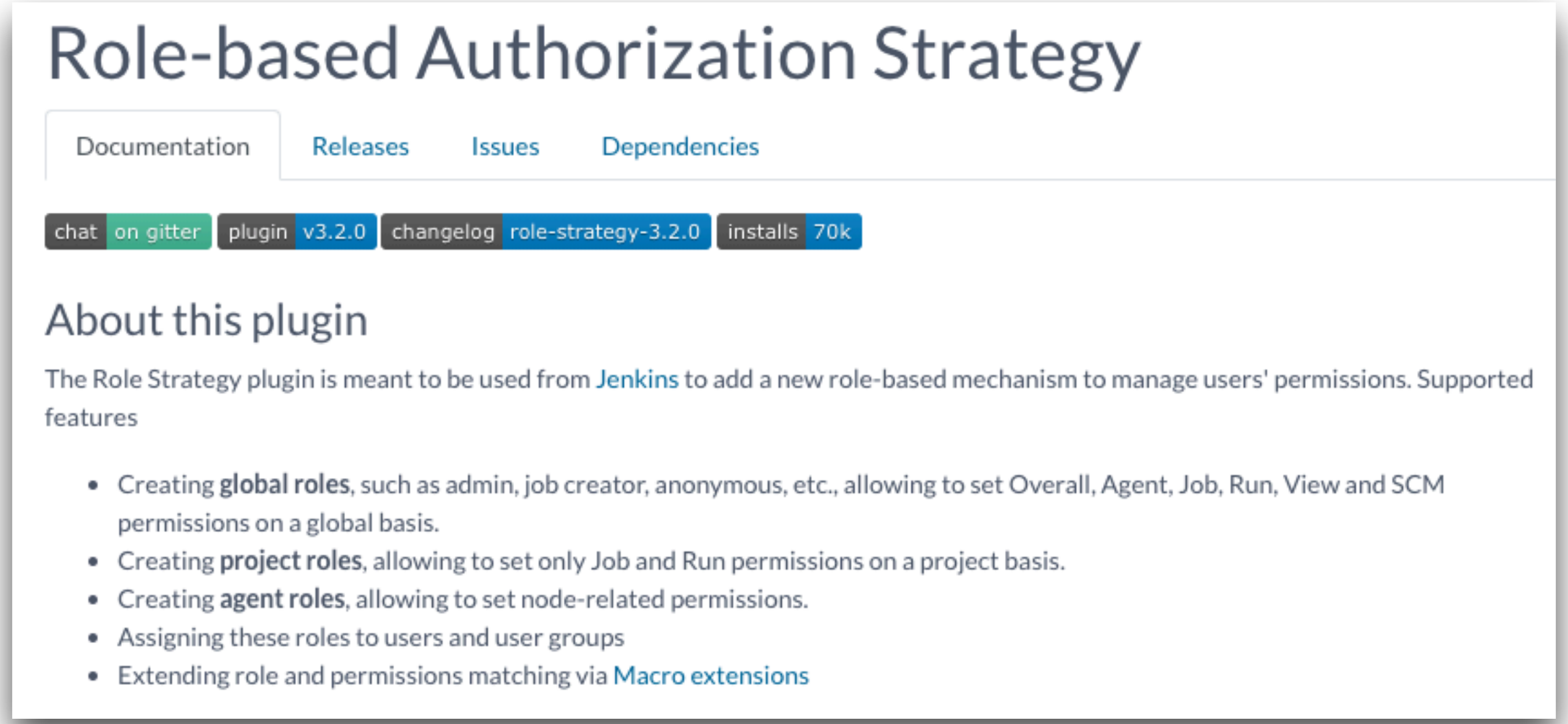
Security in Jenkins

Security in Jenkins

Username/password
Access controls
Role-bases authorization
Other plugins

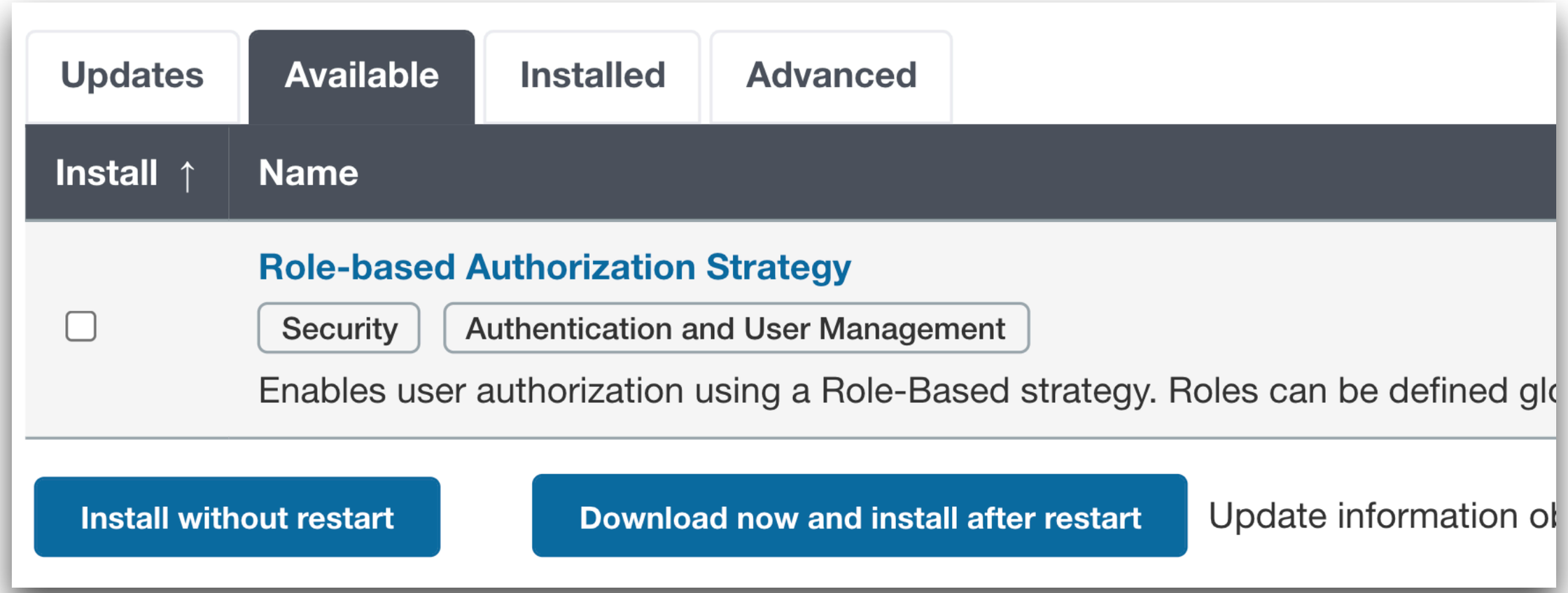
https://www.jenkins.io/doc/book/security/

Role-based Authorization Strategy



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

Role-based Authorization Strategy



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/

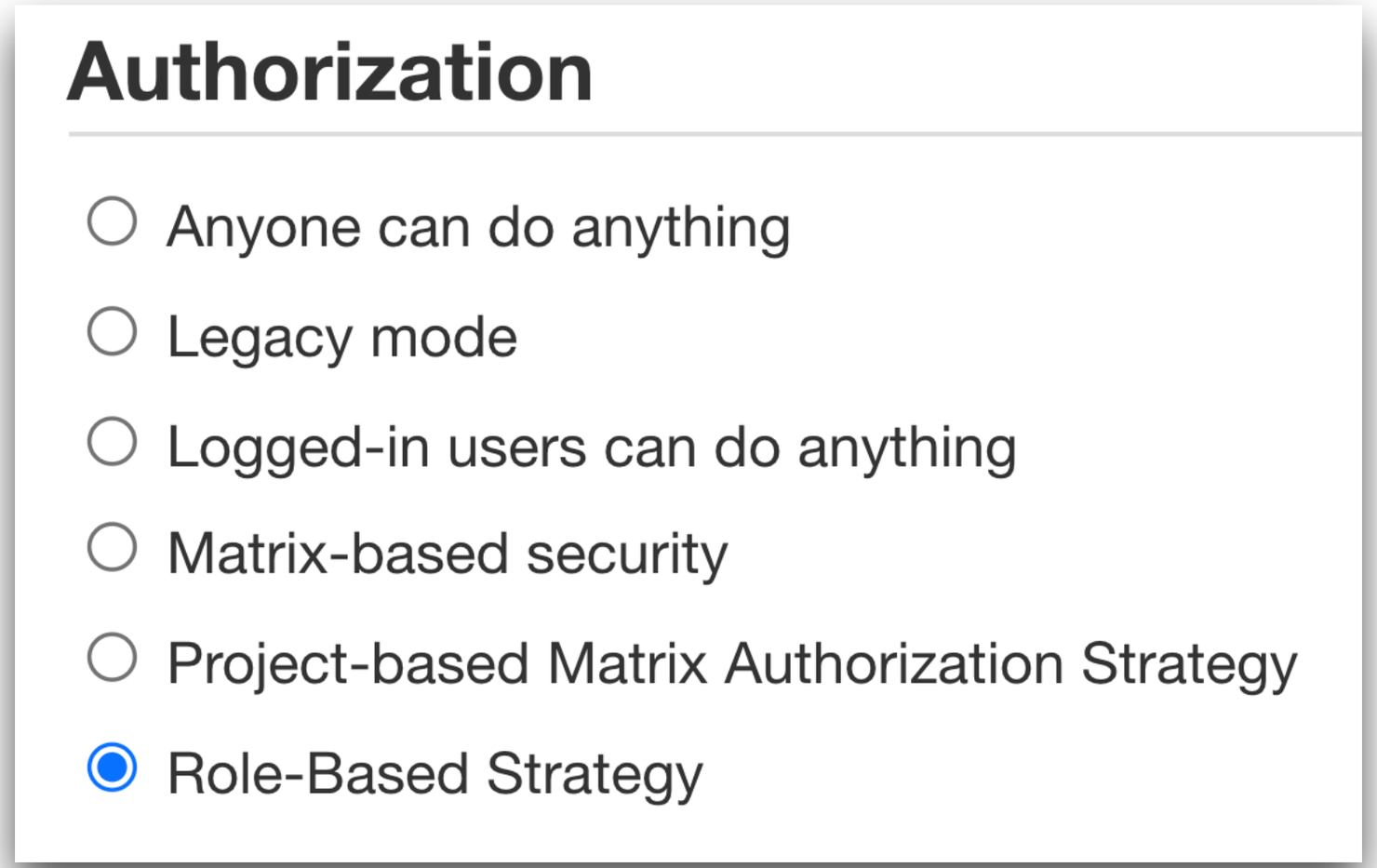
Role-based Authorization Strategy

Global roles
Project roles
Agent roles
Assign roles to users and groups

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

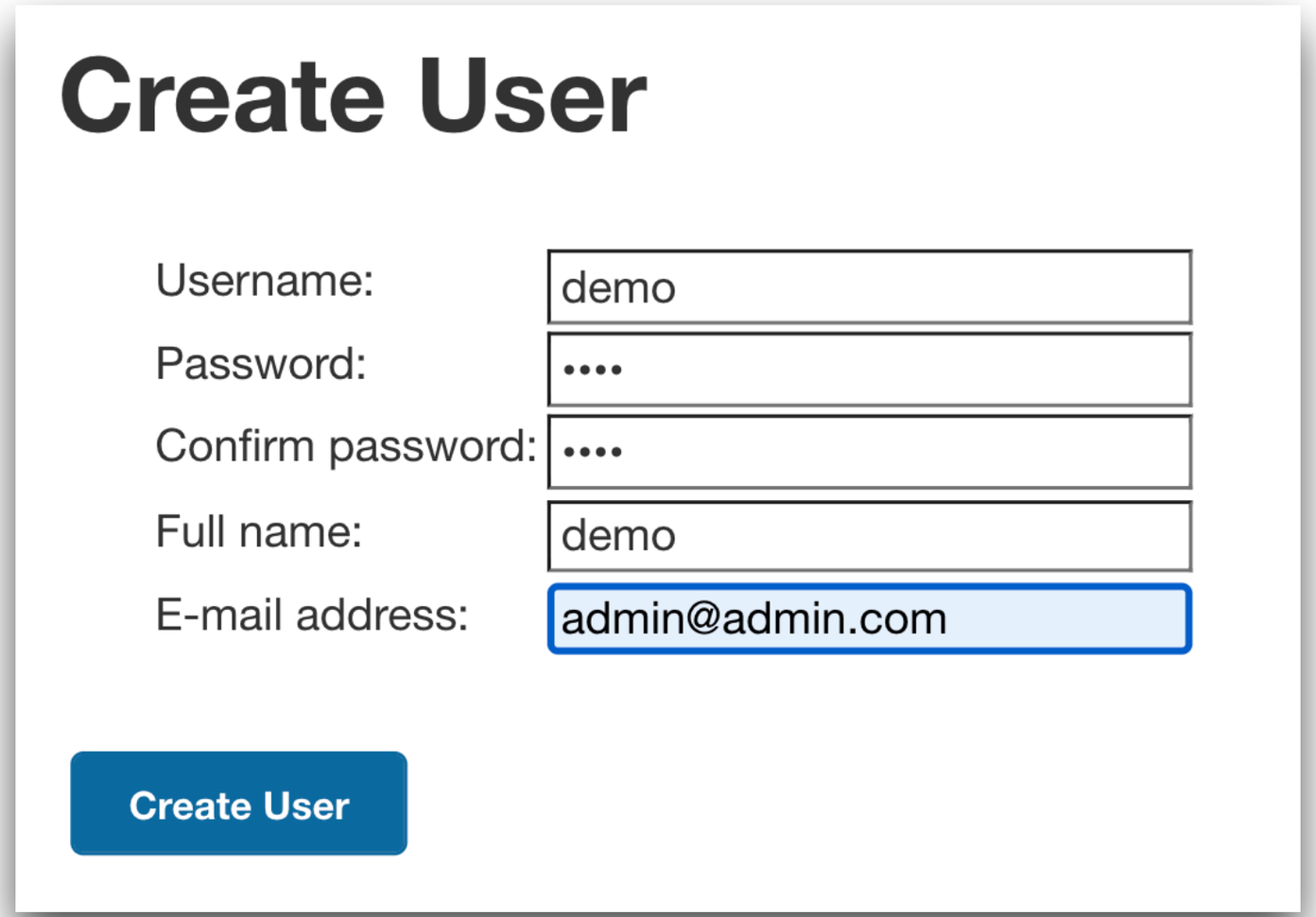
Working with Role-based

Manage Jenkins -> Configure global security



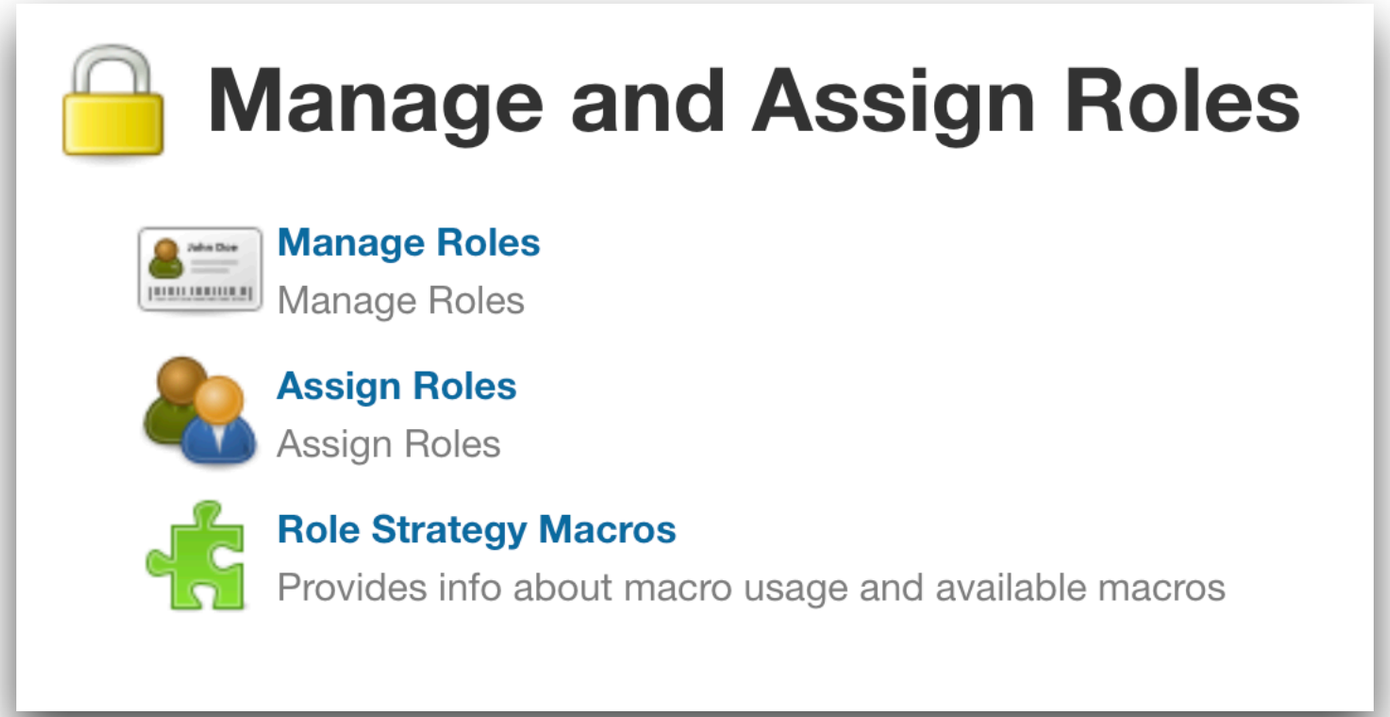
Create a new user

Manage Jenkins -> Manage Users



Create a new role

Manage Jenkins -> Manage and assign roles



Create a new role

Add new role = **general**

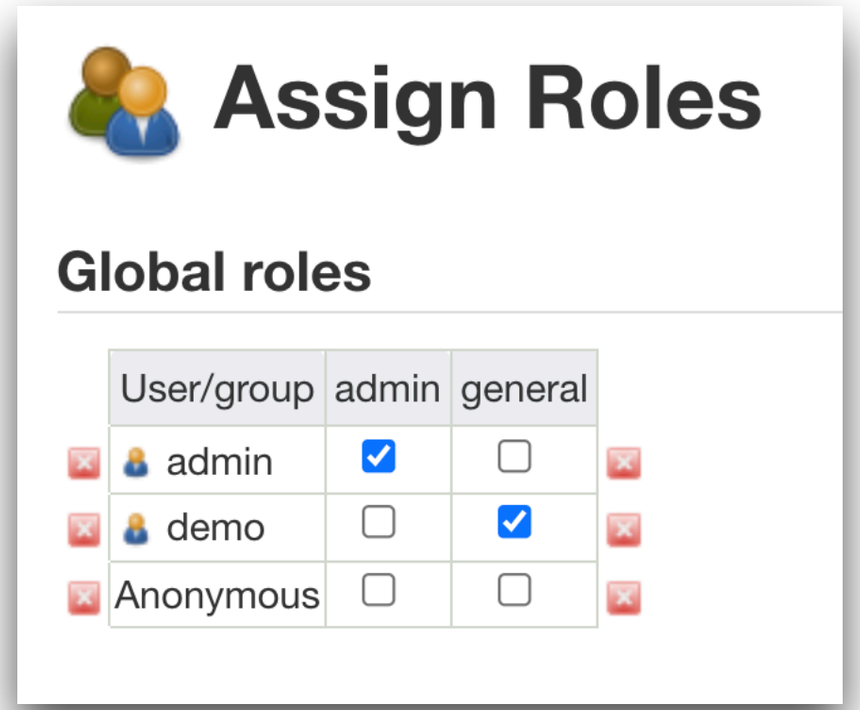
Manage Roles

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	D
admin	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	
general	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	

Assign roles to users

Add to global roles



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.

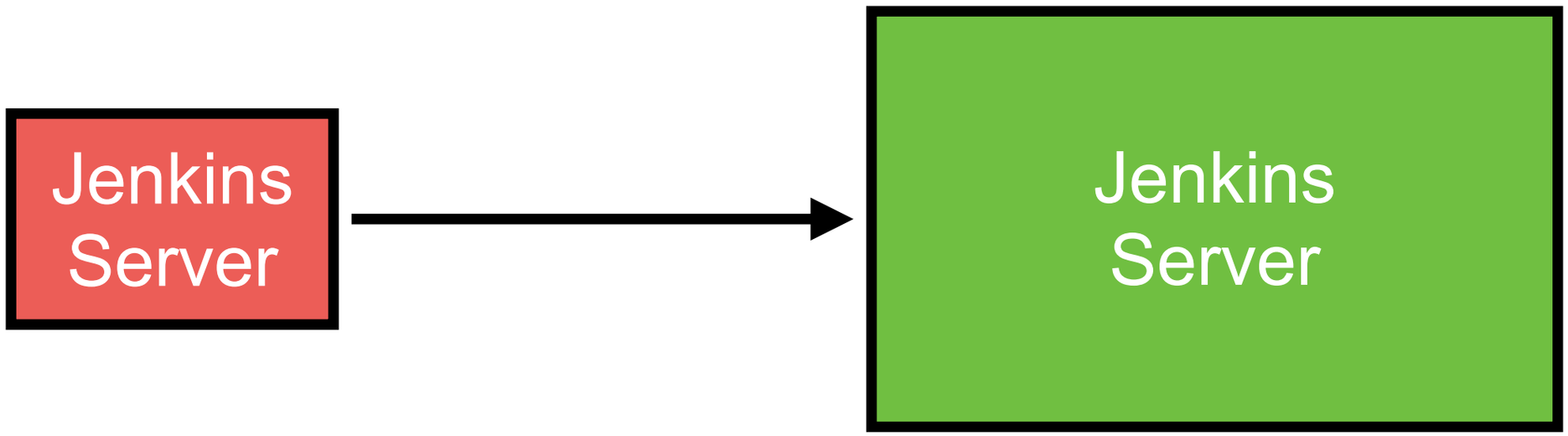
The Client ID and the Client Secret will be used to configure the Jenkins Security Realm. Keep the page open to the application registration so this information can be copied to your Jenkins configuration.

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

Scale Jenkins

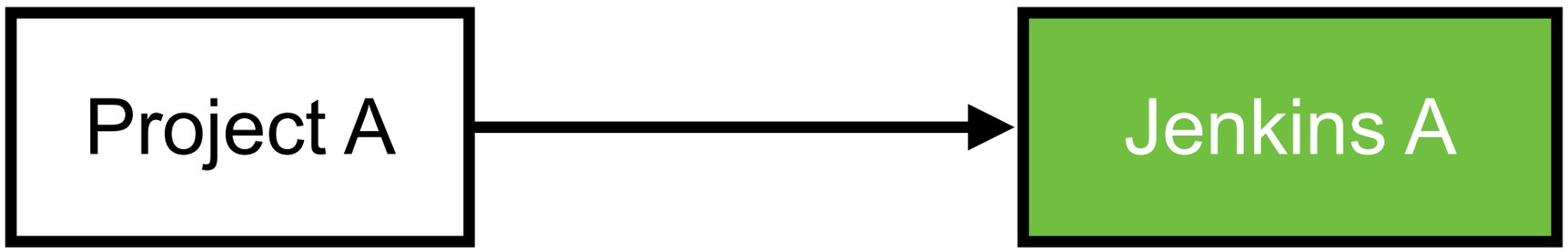
Scale Jenkins

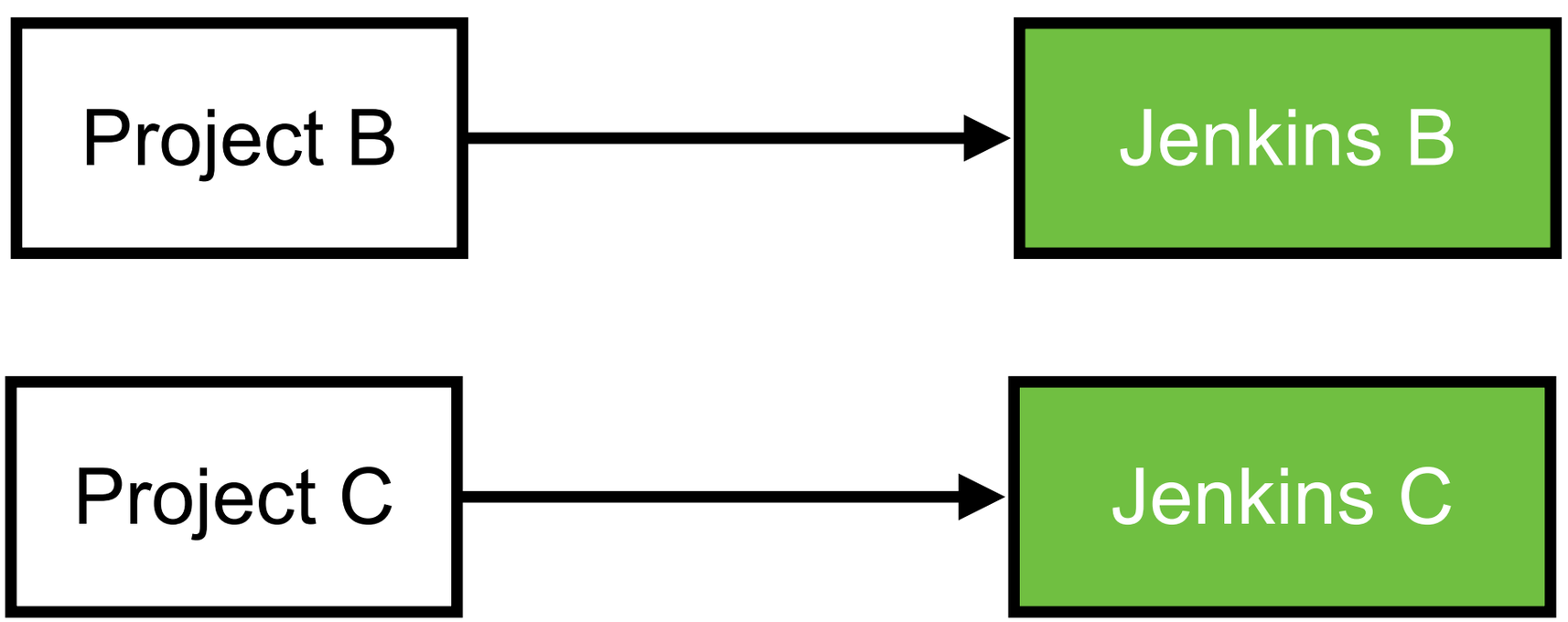
Add more resources for all projects



Scale Jenkins

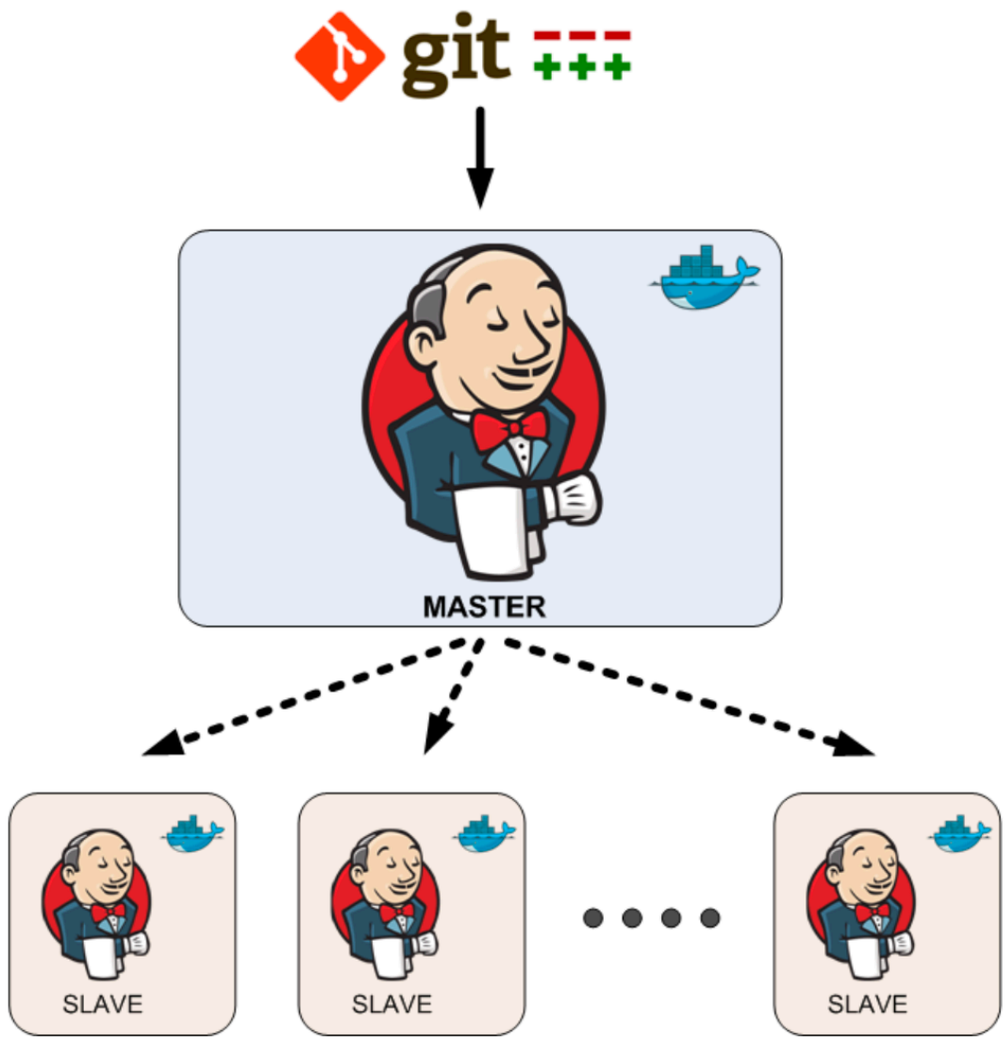
Jenkins server per project





Scale Jenkins

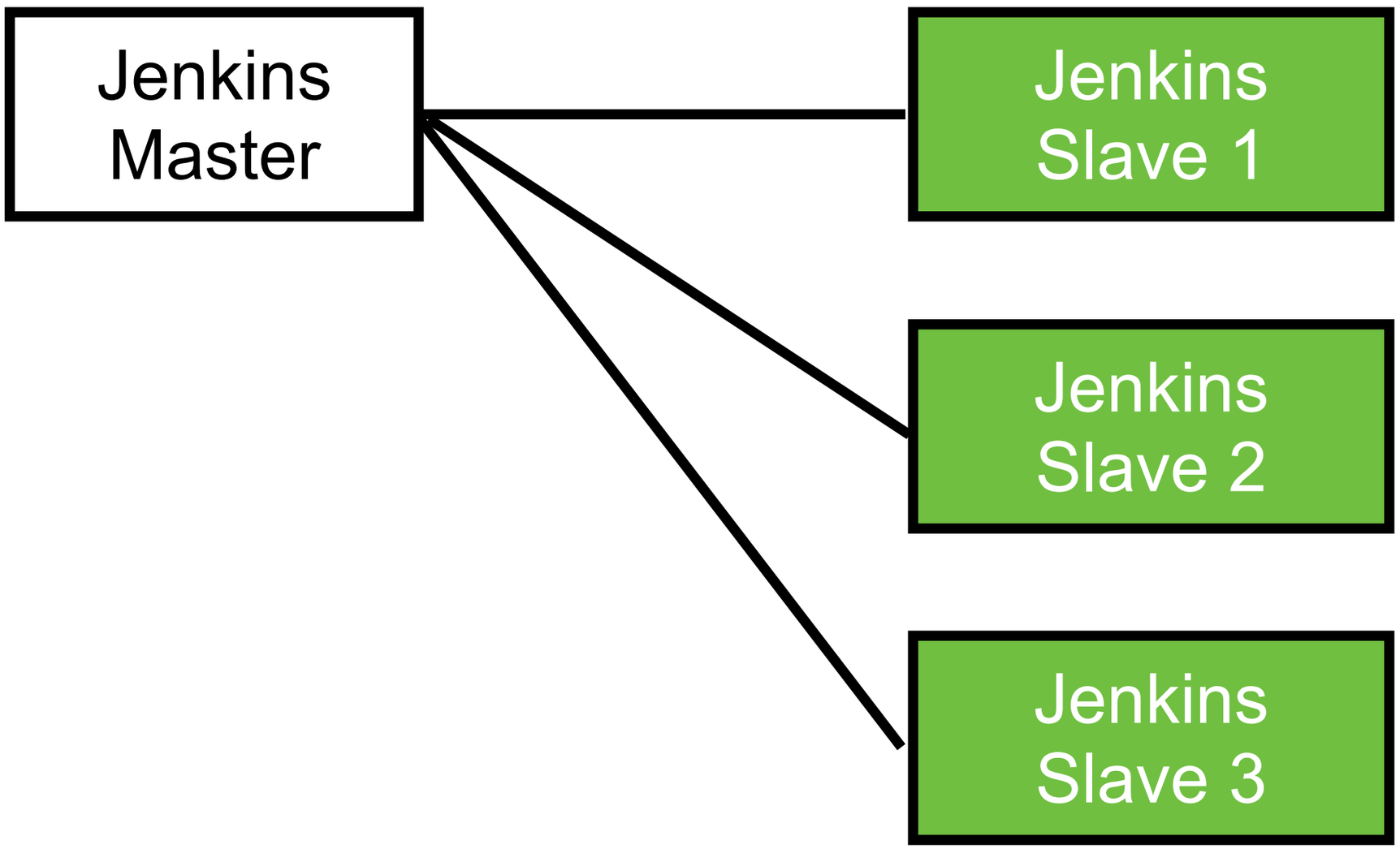
Master and Slaves (Add more nodes)



Add new nodes/agents

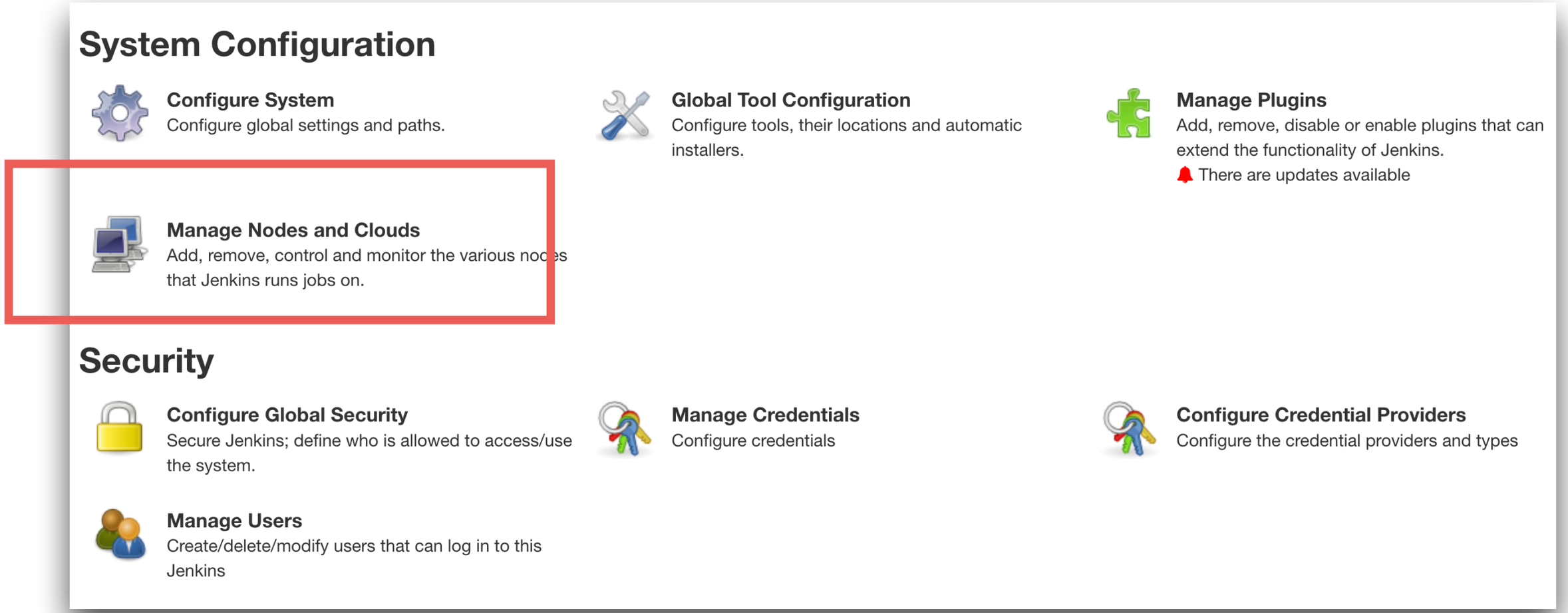
Scale Jenkins

Jenkins server per project



Create new node

Manage Jenkins -> Manage nodes and clouds



Create new node

Detail of new node

Name

worker01

worker01

Description

Number of executors

1

Remote root directory

/var/jenkins_home

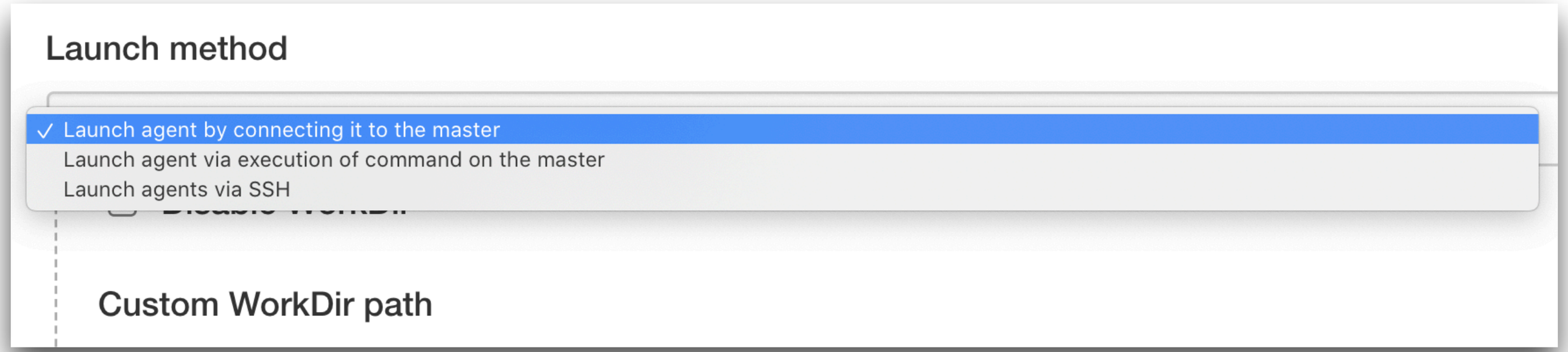
/var/jenkins_home

Labels

worker01

Launch methods

By agent from JNLP/SSH



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 @se re
```

Projects tied to worker01

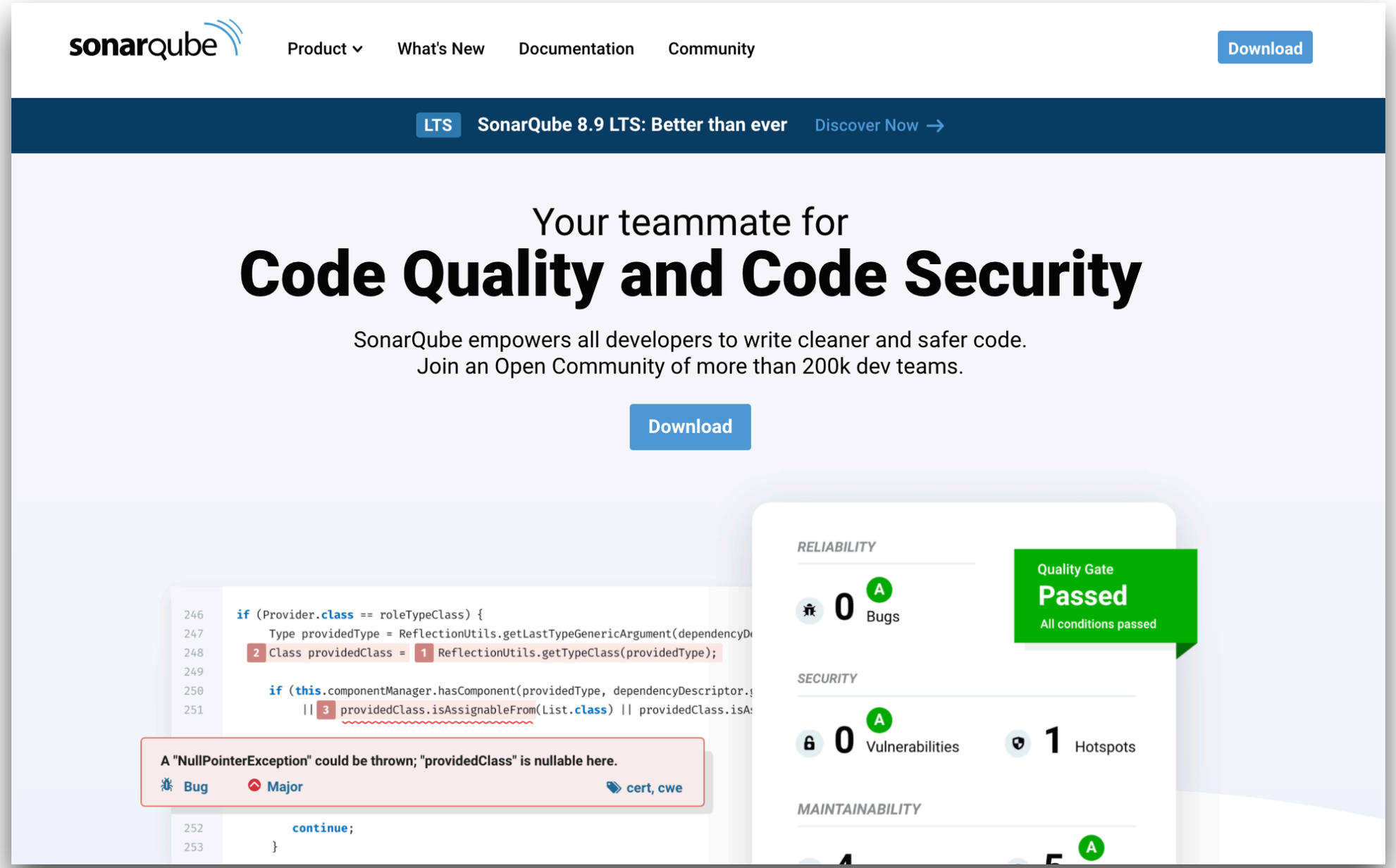
None

Success with new node

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

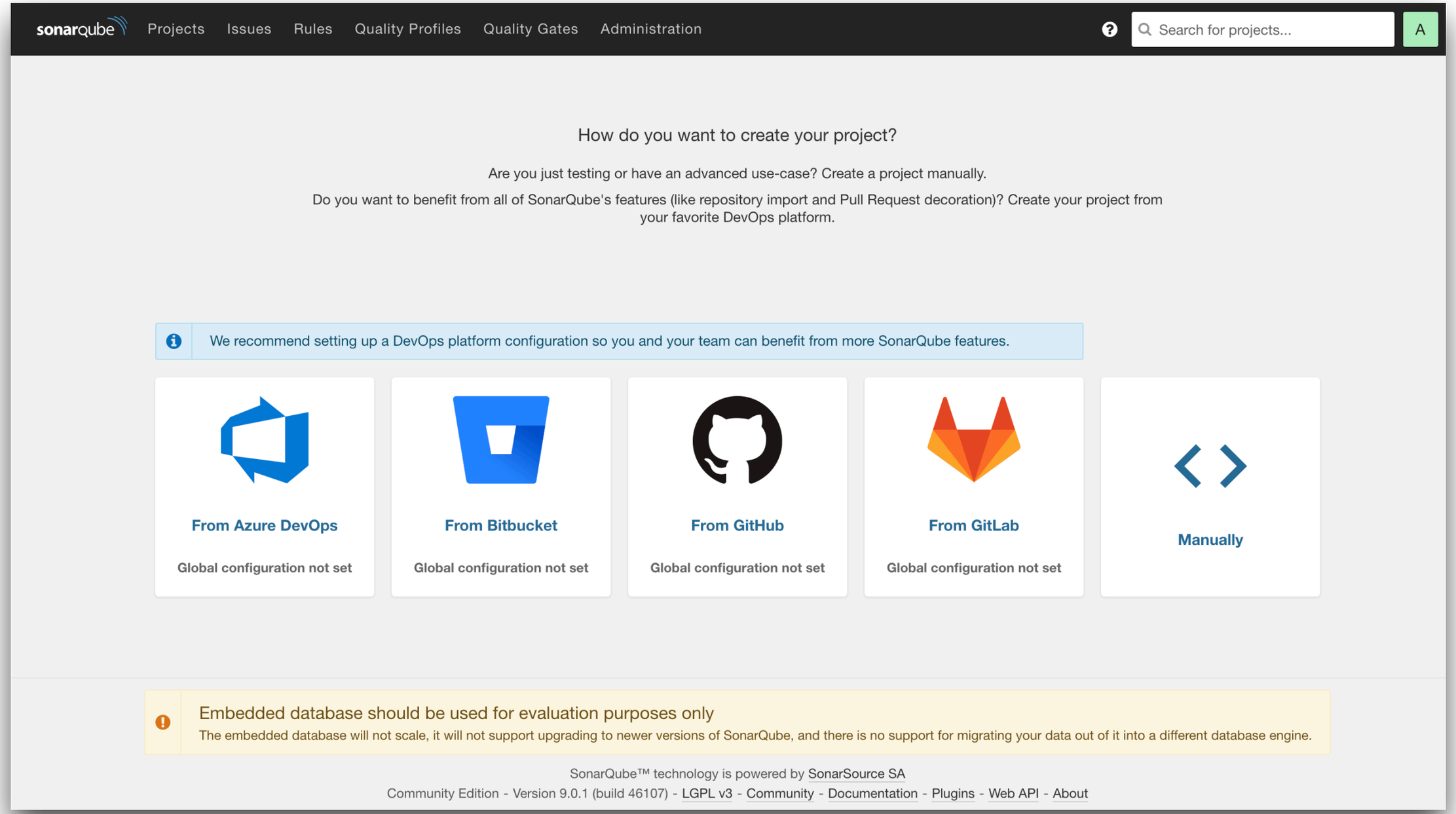
Working with SonarQube

SonarQube



https://www.sonarqube.org/

SonarQube Community version

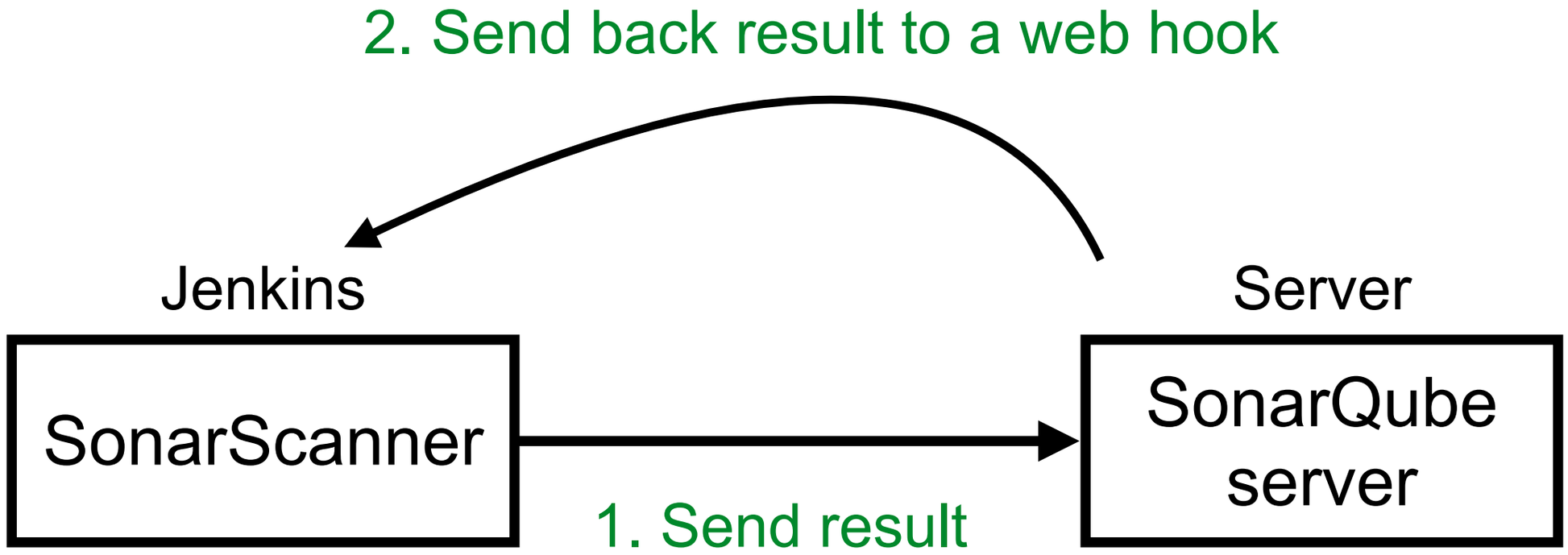


SonarQube and Scanner

File configuration

sonar-project.properties

Jenkins + SonarQube



File configuration

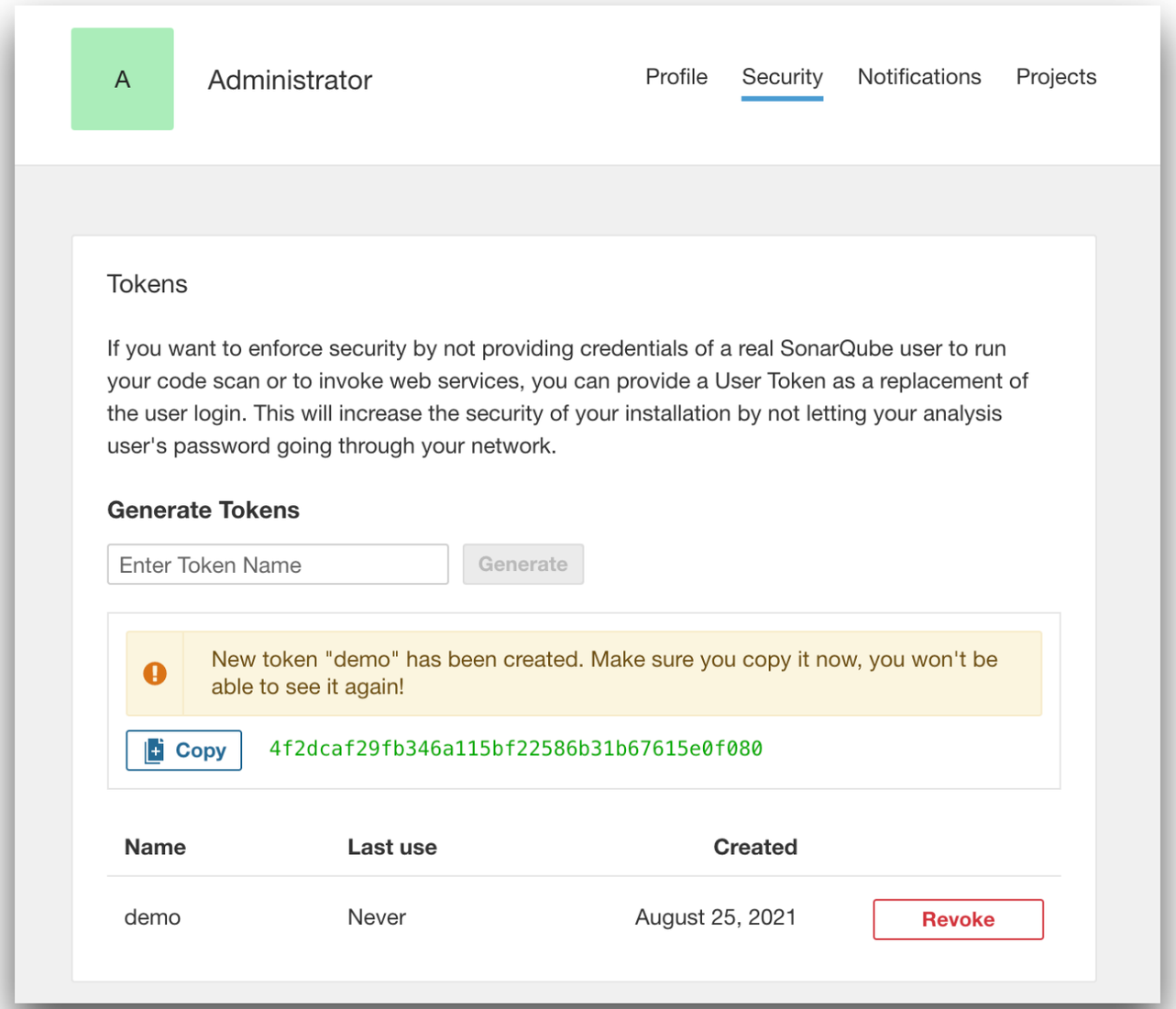
sonar-project.properties

SonarScanner

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

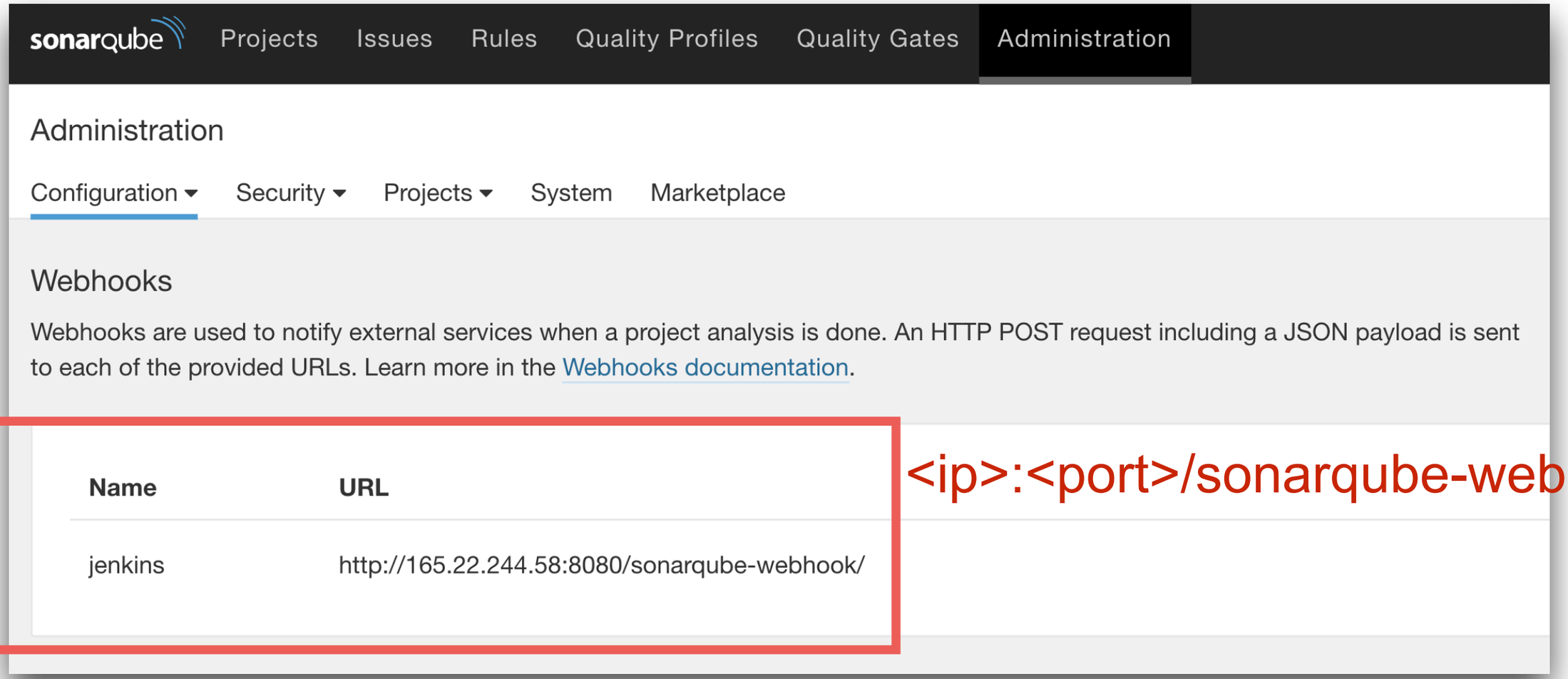
Generate secret token

User -> My Account -> Security



Create SonarQube Webhook

Administration -> Configuration -> Webhooks



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

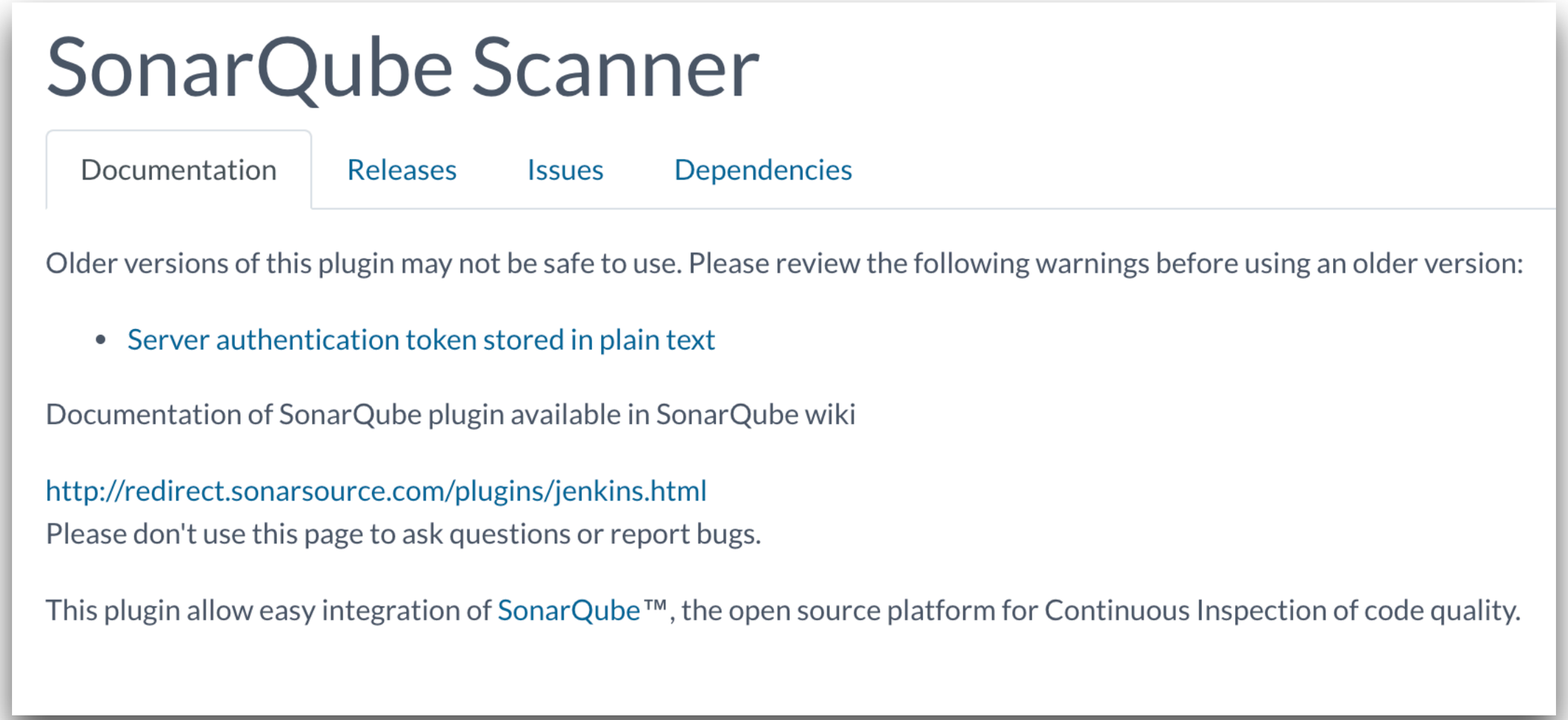
This plugin lets you centralize the configuration of SonarQube server connection details in Jenkins global configuration.

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

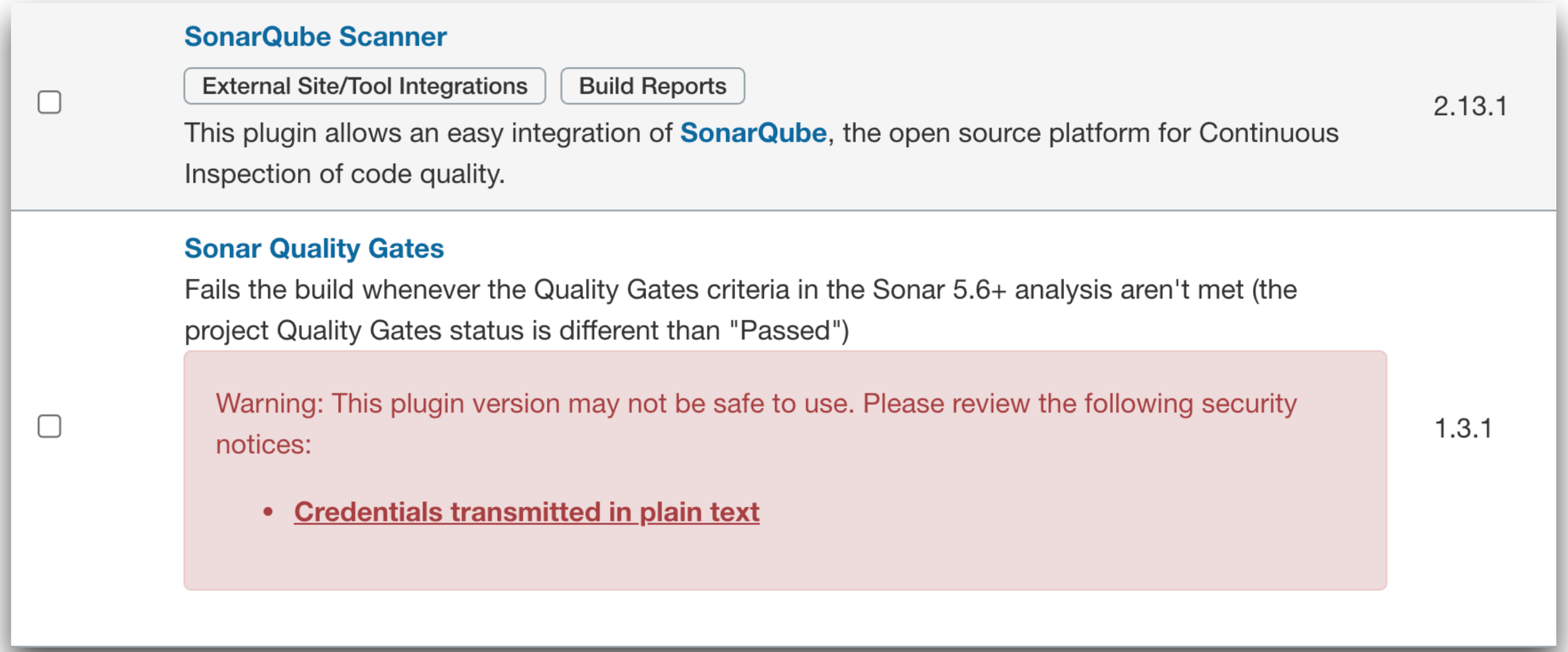
https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/

SonarScanner for Jenkins



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

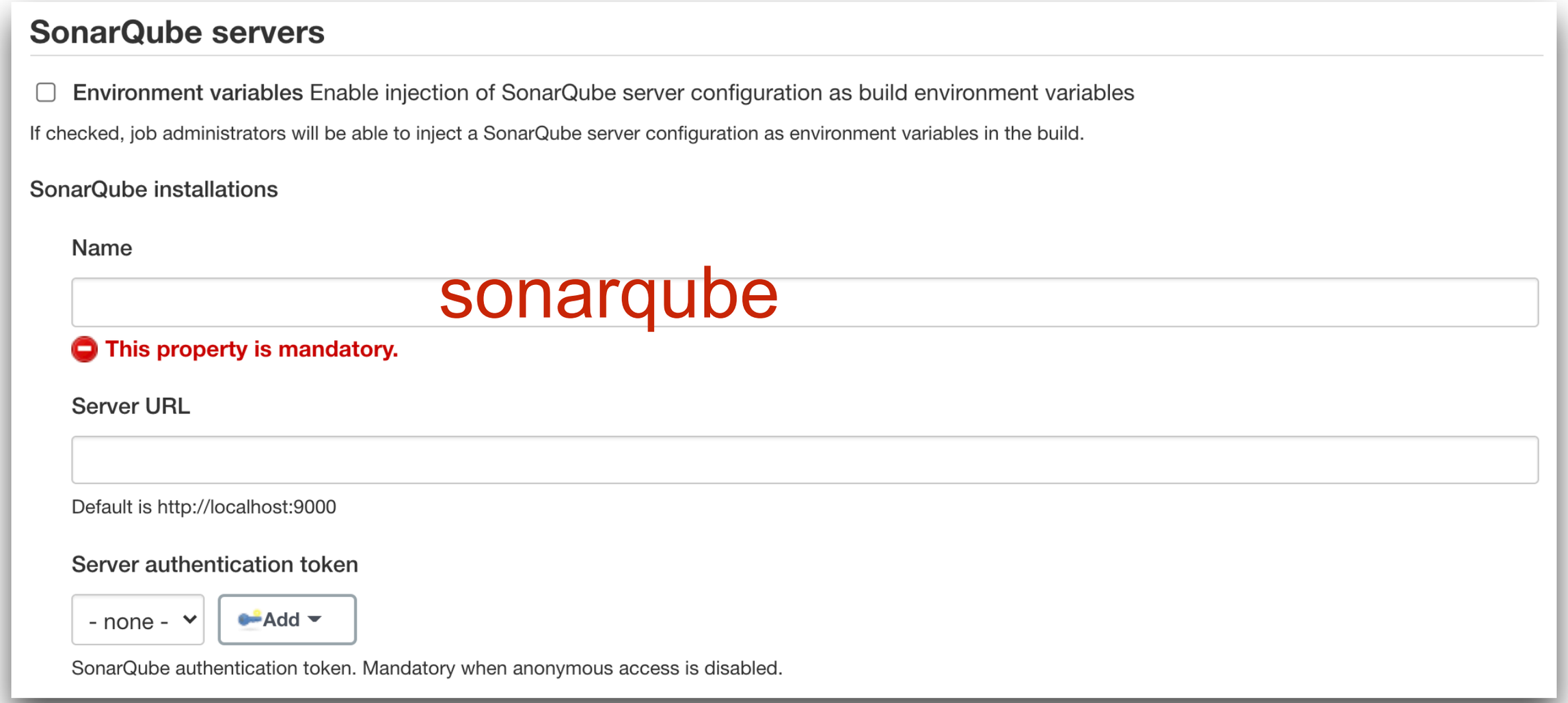
SonarScanner for Jenkins



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

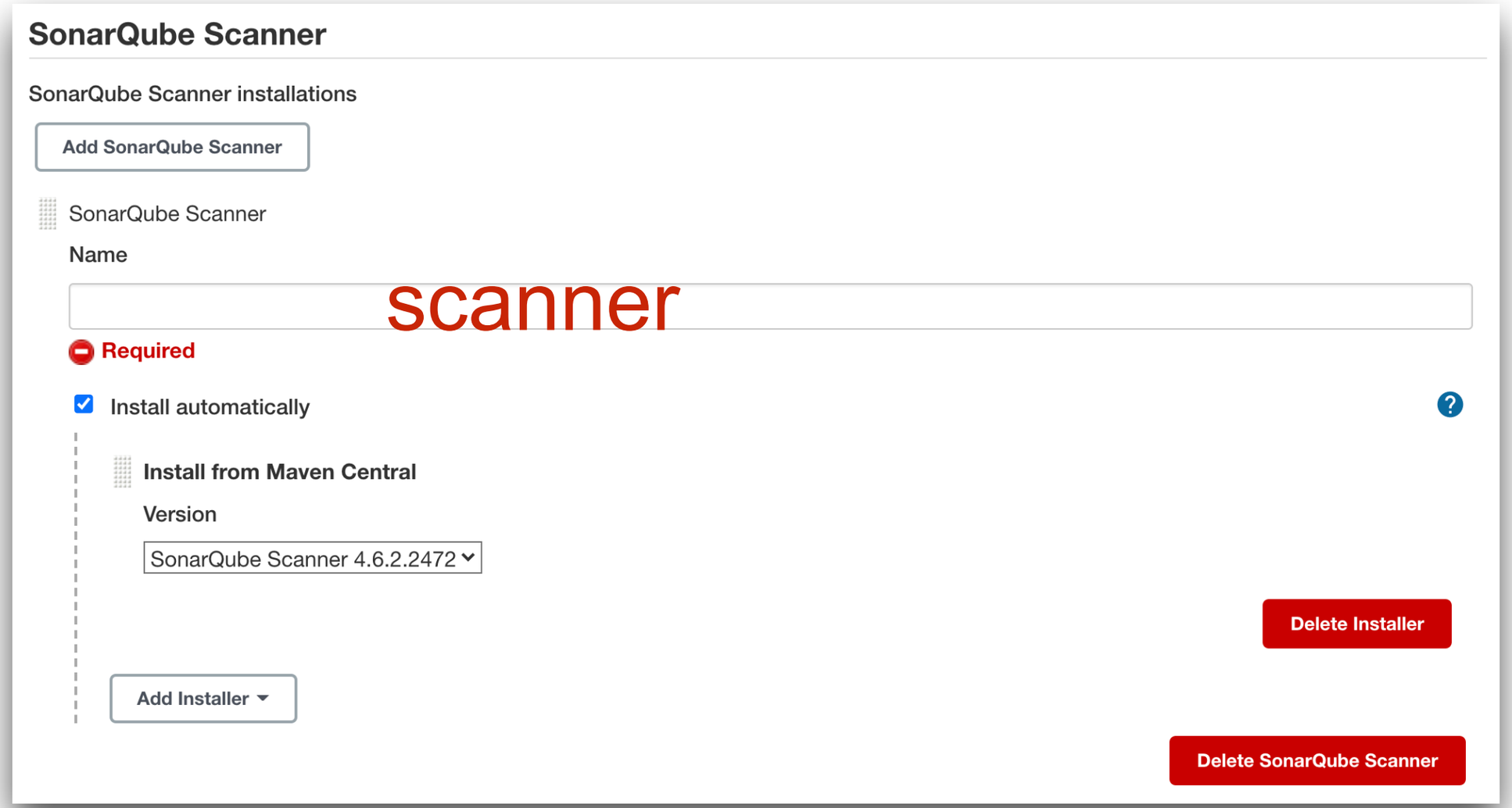
Config SonarQube Server

Manage Jenkins -> Configure System



Config SonarQube Scanner

Manage Jenkins -> Global tool configuration



Pipeline in Jenkins

Reference to SonarQube Scanner

```
stage('Sonarqube') {
    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