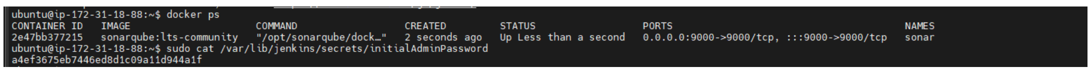
## 1. Installation of Jenkins on an EC2 Instance

- **Step 1:** Launch an EC2 instance.
  - Use Amazon Linux, Ubuntu, or any supported Linux distribution.
  - Ensure security groups allow ports 8080 (default Jenkins web interface), port 9000(SonarQube) and 22 (SSH).
- **Step 2:** Install Jenkins and Sonar-Server in Docker
  - For **Ubuntu**:

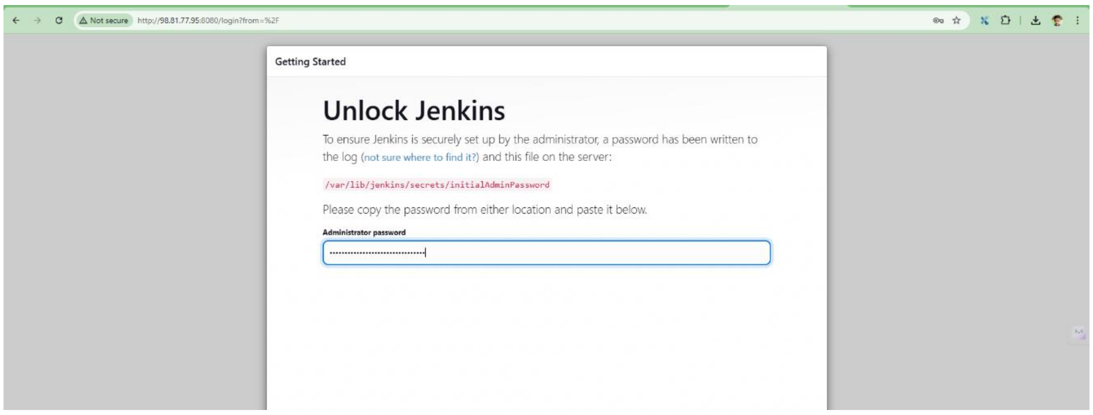
```
sudo apt update

sudo apt install fontconfig openjdk-17-jre -y

java -version

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl enable jenkins

# Install Docker

sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"

sudo apt update -y

sudo apt install docker-ce -y

sudo usermod -aG docker ${USER}

newgrp docker

sudo chmod 777 /var/run/docker.sock


docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

## 2. Setup of Jenkins Server and Configurations

- **Step 1:** Open Jenkins in the browser (http://<IP>:8080), and follow the instructions to unlock Jenkins using the initial password located at /var/lib/jenkins/secrets/initialAdminPassword.



```
ubuntu@ip-172-31-18-88:~$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED         STATUS                  PORTS                                        NAMES
2e47bb377215   sonarqube:lts-community  "/opt/sonarqube/dock…"   2 seconds ago   Up Less than a second   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
ubuntu@ip-172-31-18-88:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
a4ef3675eb7446ed8d1c09a11d944a1f
```



http://98.81.77.95:8080/login?from=%2F

Getting Started

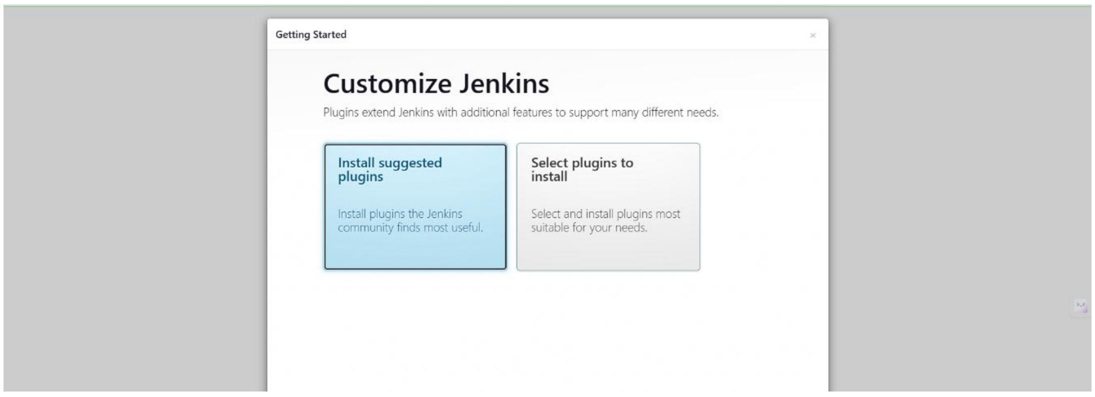
Unlock Jenkins

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:

/var/lib/jenkins/secrets/initialAdminPassword

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

Administrator password



Getting Started

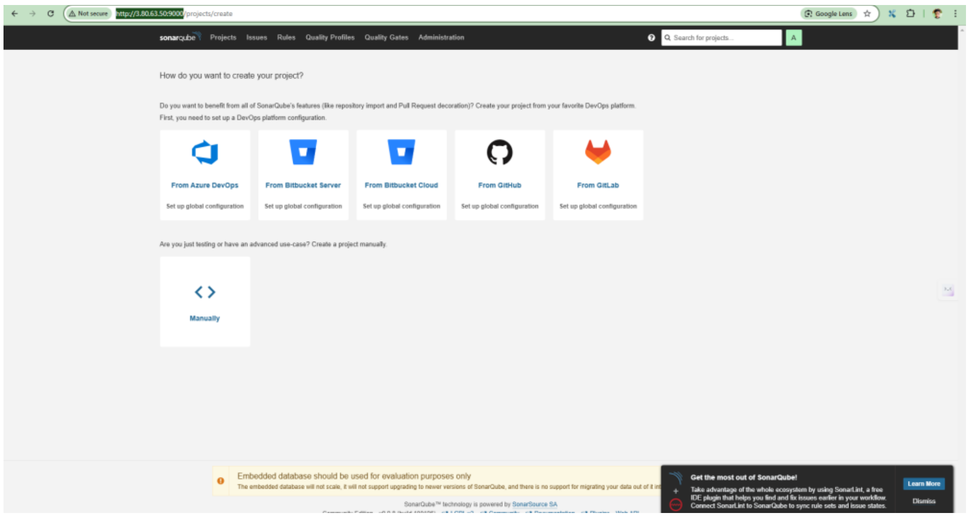
Customize Jenkins

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.

- **Step 4:** Open SonarServer in the browser (http://<IP>:9000), and Sonar-Server default user:password - admin:admin
    - Install the suggested plugins and reset the password for Jenkins and SonarQube



http://3.80.63.50:9000/projects/create

How do you want to create your project?

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.
First, you need to set up a DevOps platform configuration.

From Azure DevOps
Set up global configuration
From Bitbucket Server
Set up global configuration
From Bitbucket Cloud
Set up global configuration
From GitHub
Set up global configuration
From GitLab
Set up global configuration

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

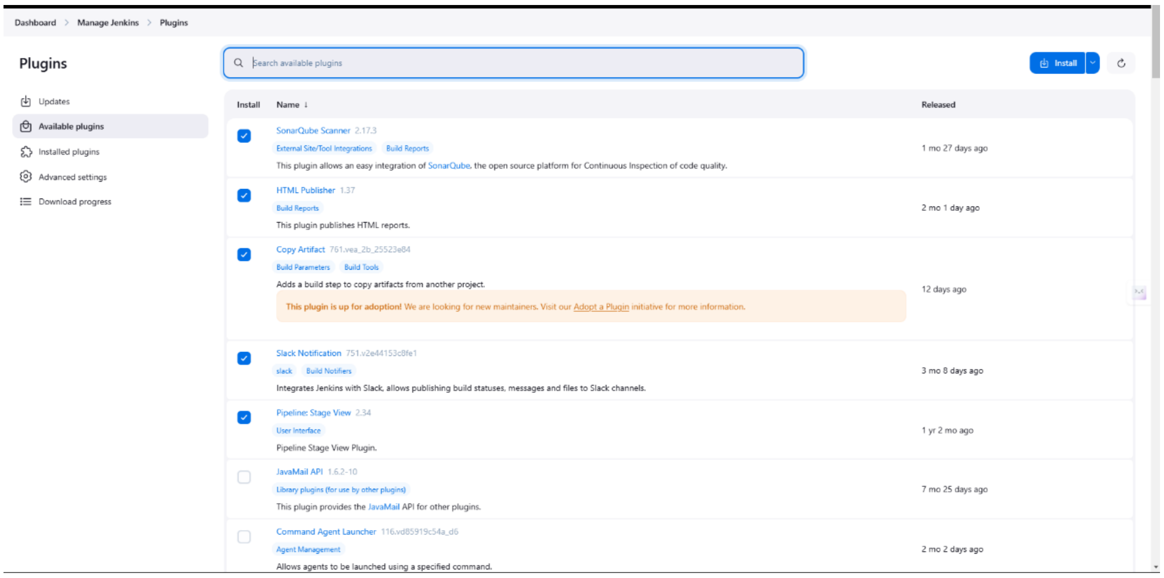
Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it in

SonarQube™ technology is powered by SonarSource SA

Get the most out of SonarQube!
Take advantage of the whole ecosystem by using SonarLint, a free IDE plugin that helps you find and fix issues earlier in your workflow. Connect SonarLint to SonarQube to sync rule sets and issue states.

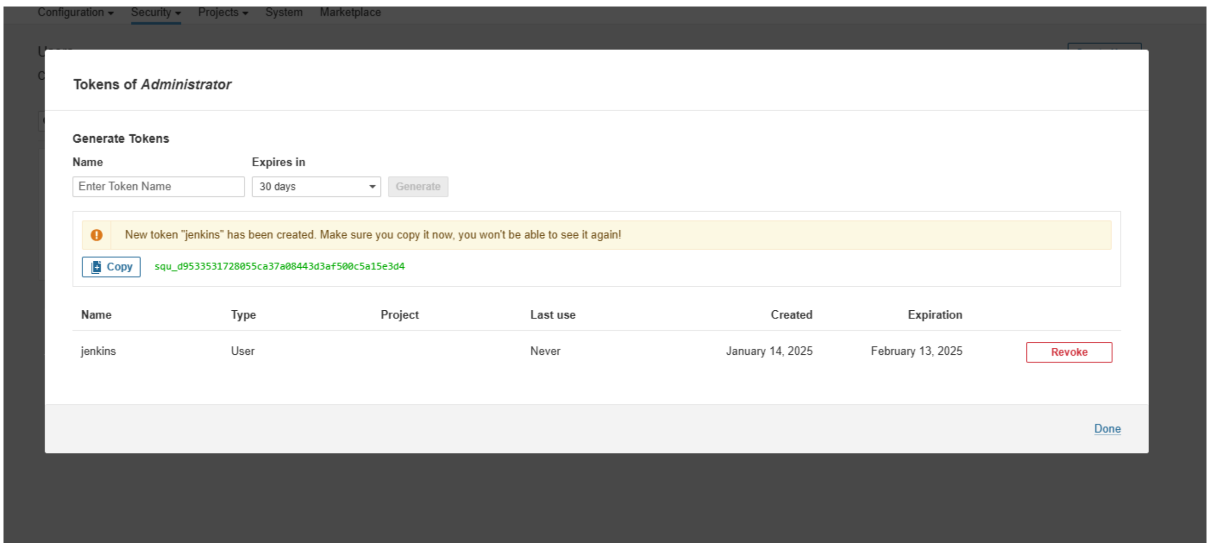
## 3. Installing Plugins and Tools, System Conf. in Jenkins

- **Step 1:** Install the **plugin** for Jenkins.
    - Navigate to **Manage Jenkins** > **Manage Plugins** > **Available** tab.
    - Search for
        - SonarQube Scanner
        - HTML Publisher
        - Copy Artifact
        - Slack Notification and click **Install without restart**.



| Install | Name ↓ | Released |
|---|---|---|
| ✓ | SonarQube Scanner 2.17.3<br>External Site/Tool Integrations · Build Reports<br>This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality. | 1 mo 27 days ago |
| ✓ | HTML Publisher 1.37<br>Build Reports<br>This plugin publishes HTML reports. | 2 mo 1 day ago |
| ✓ | Copy Artifact 761.vea_2b_25523e84<br>Build Parameters · Build Tools<br>Adds a build step to copy artifacts from another project.<br>This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information. | 12 days ago |
| ✓ | Slack Notification 751.v2e44153c8fe1<br>slack · Build Notifiers<br>Integrates Jenkins with Slack, allows publishing build statuses, messages and files to Slack channels. | 3 mo 8 days ago |
| ✓ | Pipeline: Stage View 2.34<br>User Interface<br>Pipeline Stage View Plugin. | 1 yr 2 mo ago |
|  | JavaMail API 1.6.2-10<br>Library plugins (for use by other plugins)<br>This plugin provides the JavaMail API for other plugins. | 7 mo 25 days ago |
|  | Command Agent Launcher 116.vd85919c54a_d6<br>Agent Management<br>Allows agents to be launched using a specified command. | 2 mo 2 days ago |

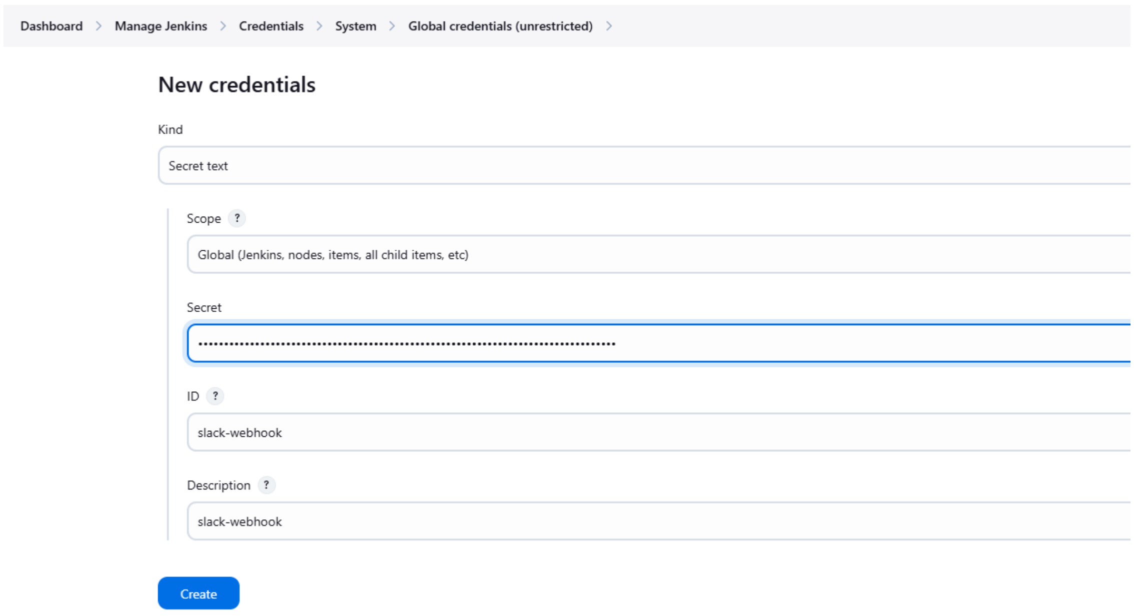
- **Step 2:** Add Credentials for different Plugins:
    - Add credentials for
        - sonar-token
        - slack-webhook
        - git



New token "jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

```
squ_d9533531728055ca37a08443d3af500c5a15e3d4
```

| Name | Type | Project | Last use | Created | Expiration |  |
|---|---|---|---|---|---|---|
| jenkins | User |  | Never | January 14, 2025 | February 13, 2025 | Revoke |



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Secret text

Scope ?

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

Secret

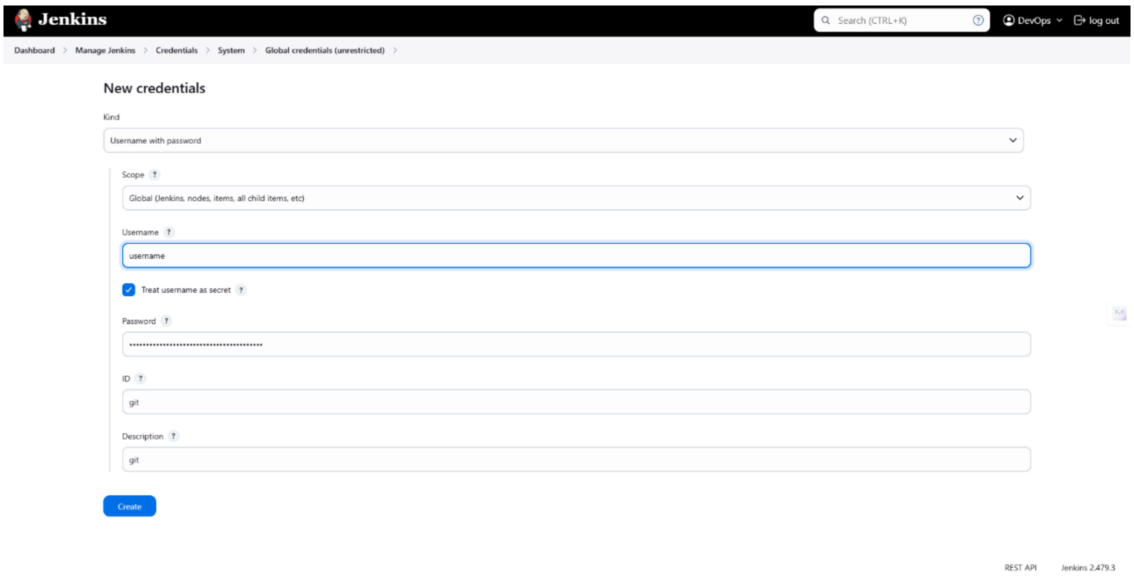
••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

ID ?

slack-webhook

Description ?

slack-webhook



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Username with password

Scope ?

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

Username ?

username

✓ Treat username as secret ?

Password ?

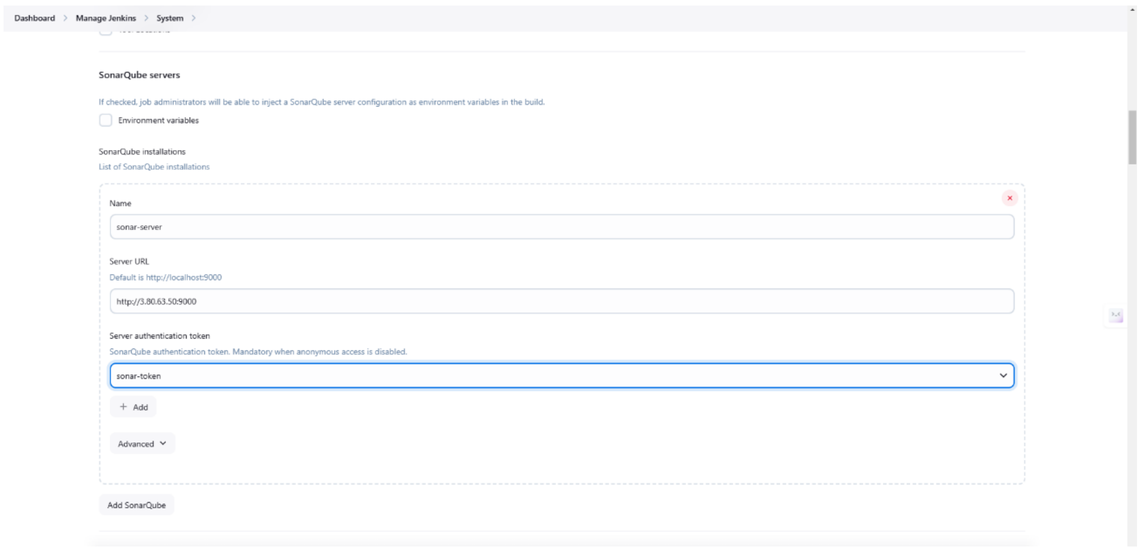
••••••••••••••••••••••••••••••••••••••••

ID ?

git

Description ?

git



Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables

SonarQube installations

List of SonarQube installations

Name

sonar-server

Server URL

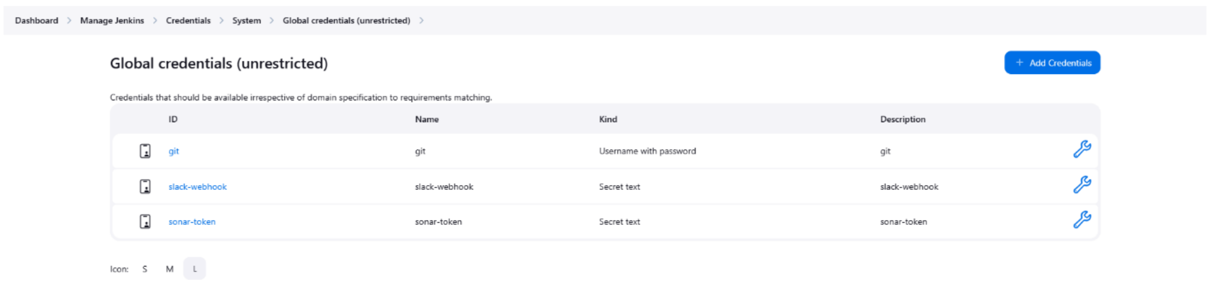
Default is http://localhost:9000

http://3.80.63.50:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

sonar-token

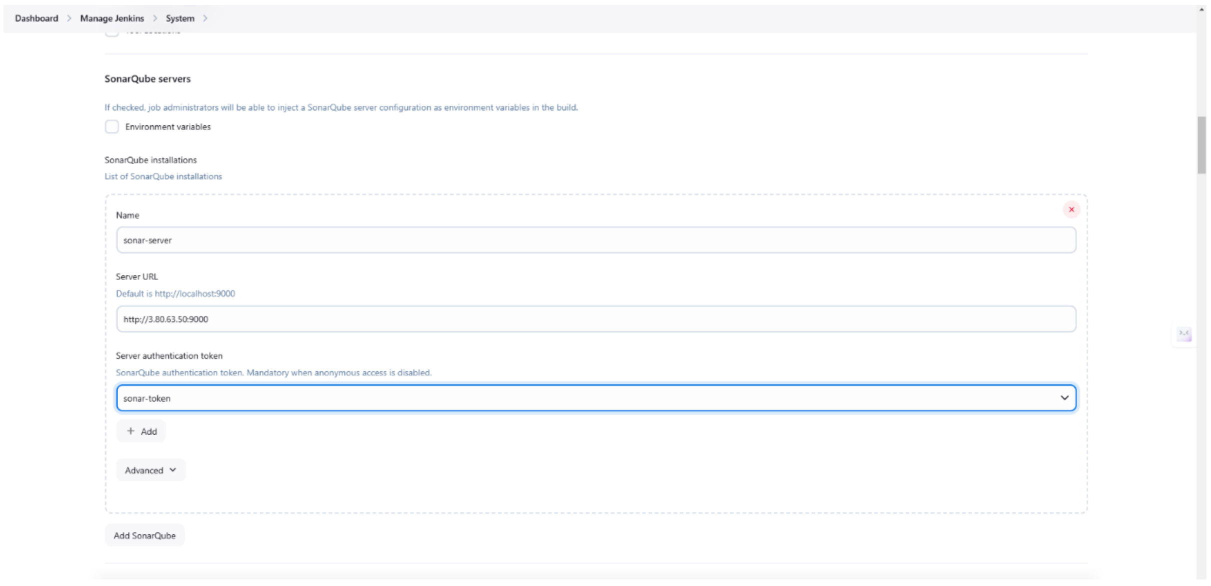

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

| ID | Name | Kind | Description |
|---|---|---|---|
| git | git | Username with password | git |
| slack-webhook | slack-webhook | Secret text | slack-webhook |
| sonar-token | sonar-token | Secret text | sonar-token |

- **Step 4:** Setting Up System Configuration in Jenkins:
    - Navigate to **Manage Jenkins** > **Configure System**.


Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables

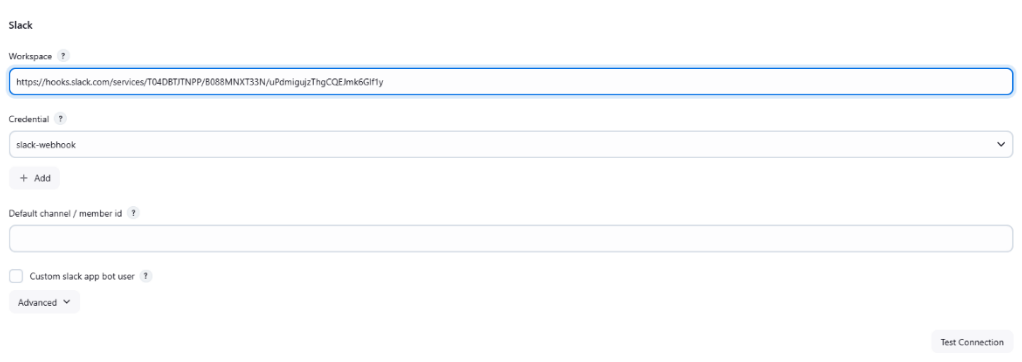
SonarQube installations

List of SonarQube installations

Name: sonar-server

Server URL (Default is http://localhost:9000): http://3.80.63.50:9000

Server authentication token (SonarQube authentication token. Mandatory when anonymous access is disabled.): sonar-token


Slack

Workspace: https://hooks.slack.com/services/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

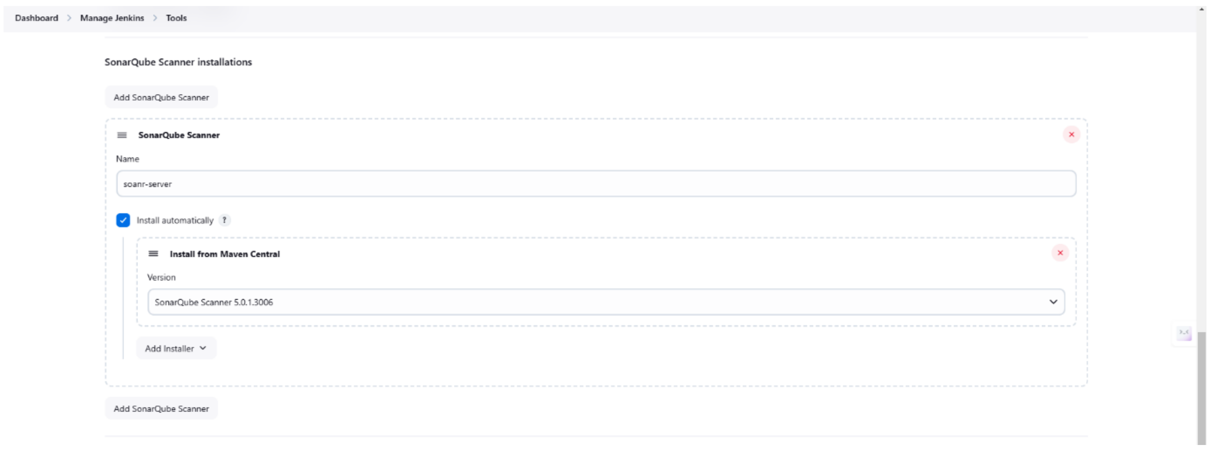
Credential: slack-webhook

Default channel / member id:

☐ Custom slack app bot user

- Add the System setting from the above Images

- **Step 4:** Setting Up Tools Configuration in Jenkins:
    - Navigate to **Manage Jenkins** > **Configure Tools**.



SonarQube Scanner installations

Add SonarQube Scanner

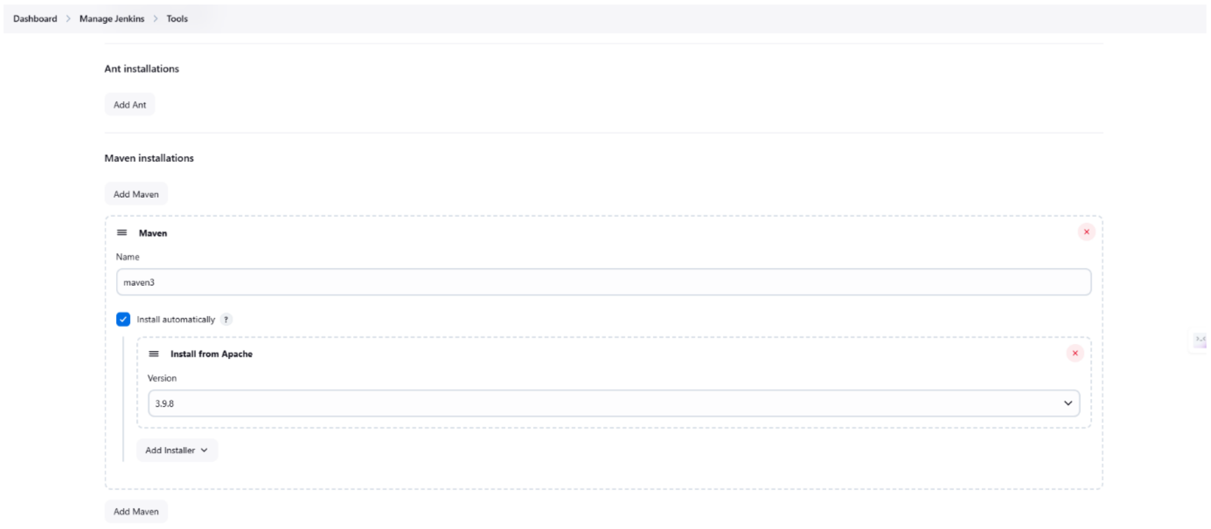
SonarQube Scanner

Name: soanr-server

✓ Install automatically

Install from Maven Central

Version: SonarQube Scanner 5.0.1.3006



Ant installations

Maven installations

Maven

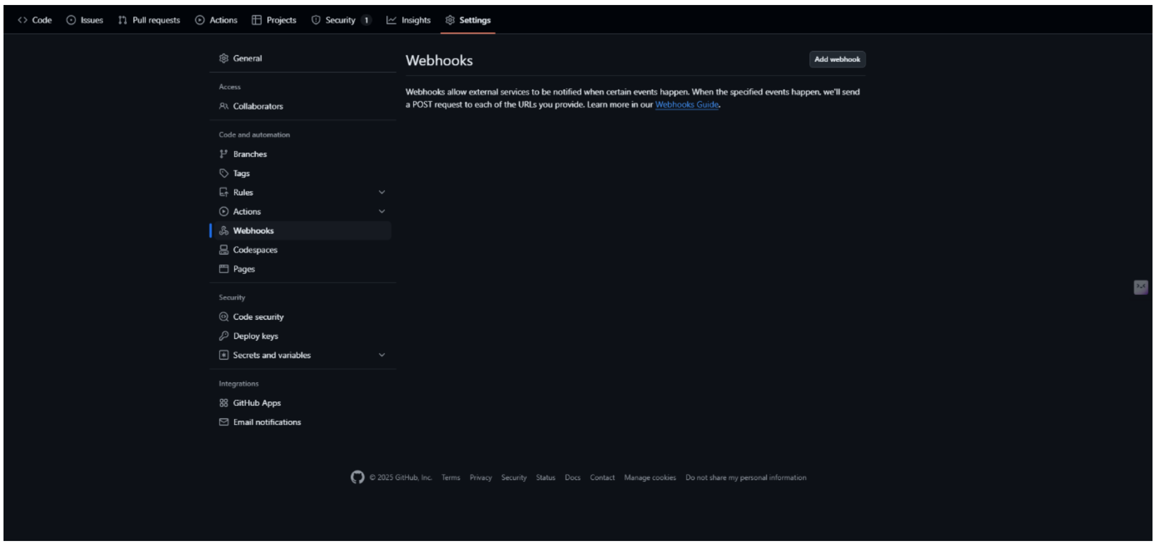
Name: maven3

✓ Install automatically

Install from Apache

Version: 3.9.8

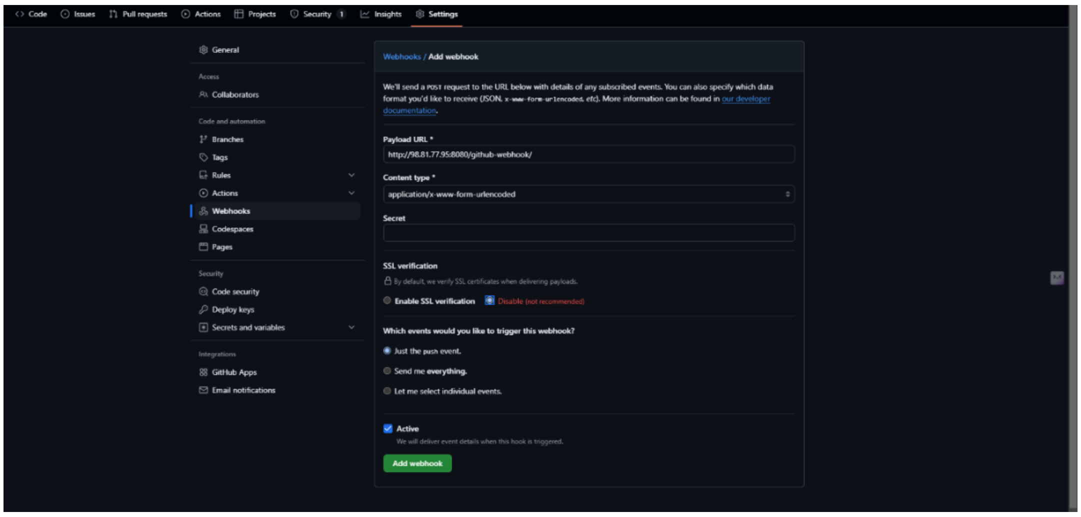
- **Step 5:** Enable **Webhooks** on GitHub.
    - Go to your GitHub repository > **Settings** > **Webhooks** > **Add webhook**.
    - Set the Payload URL to http://<jenkins-server>/github-webhook/.
    - Set Content type to **application/json** and enable events for **push**.



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *: http://98.81.77.95:8080/github-webhook/

Content type *: application/x-www-form-urlencoded

Secret:

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification [✓ Disable (not recommended)]

Which events would you like to trigger this webhook?

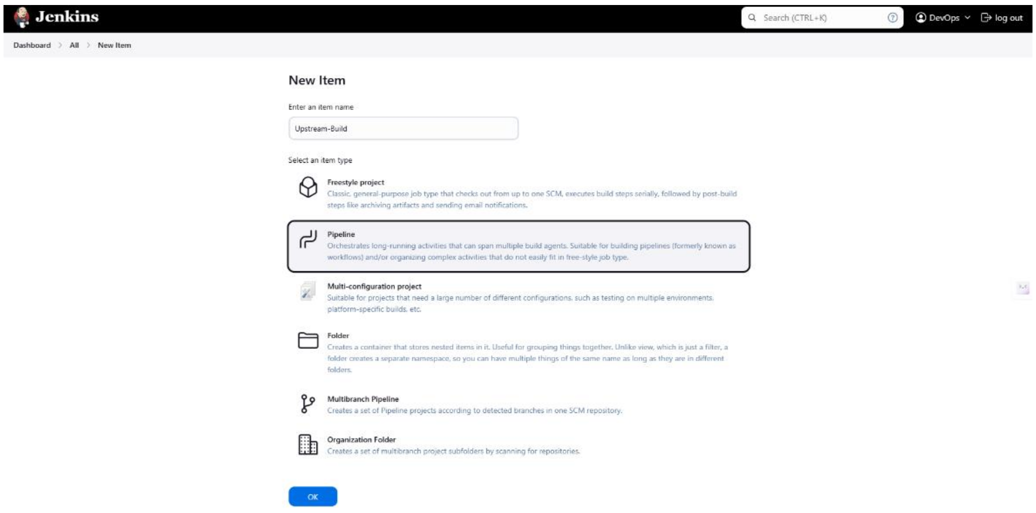
[✓ Just the push event.]
Send me everything.
Let me select individual events.

[✓ Active]
We will deliver event details when this hook is triggered.

## 4. Creating Jobs in Jenkins

**Step 1:** Upstream Job

- Go to your Jenkins dashboard **New Items** > **Pipeline**
- Follow the below Steps for Upstream



New Item

Enter an item name: Upstream-Build

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

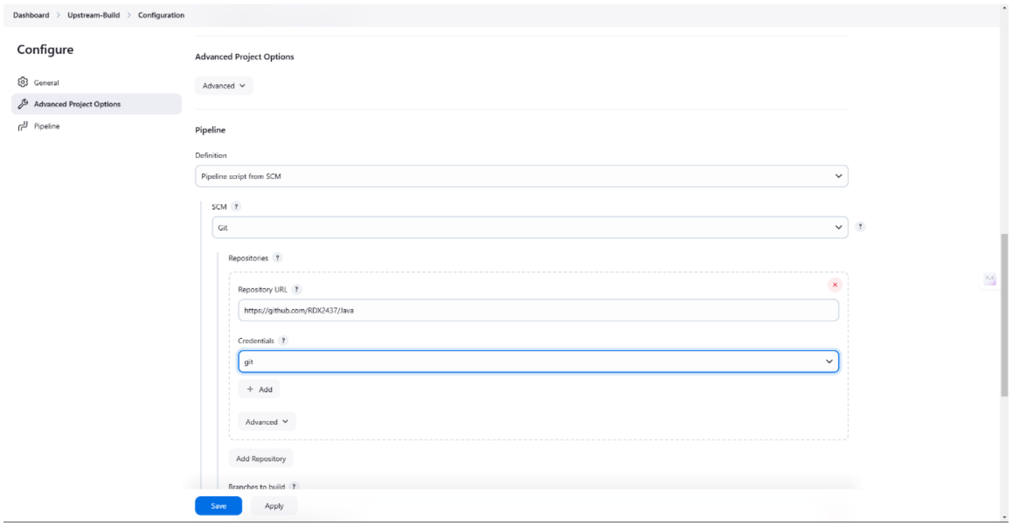
[✓ Pipeline]
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.



Advanced Project Options

Pipeline

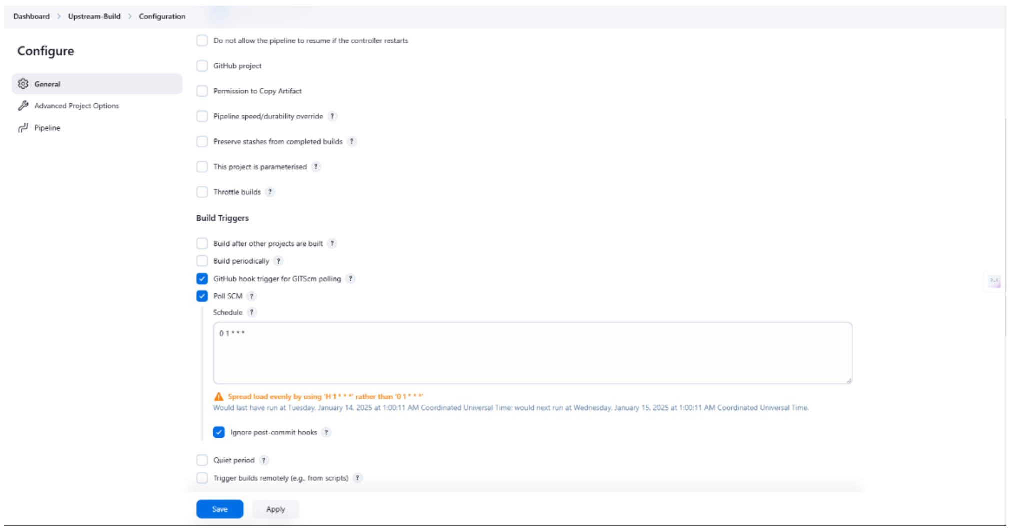
Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/RDX2437/Java

Credentials: git


Dashboard > Upstream-Build > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project
- [ ] Permission to Copy Artifact
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] This project is parameterised
- [ ] Throttle builds

Build Triggers

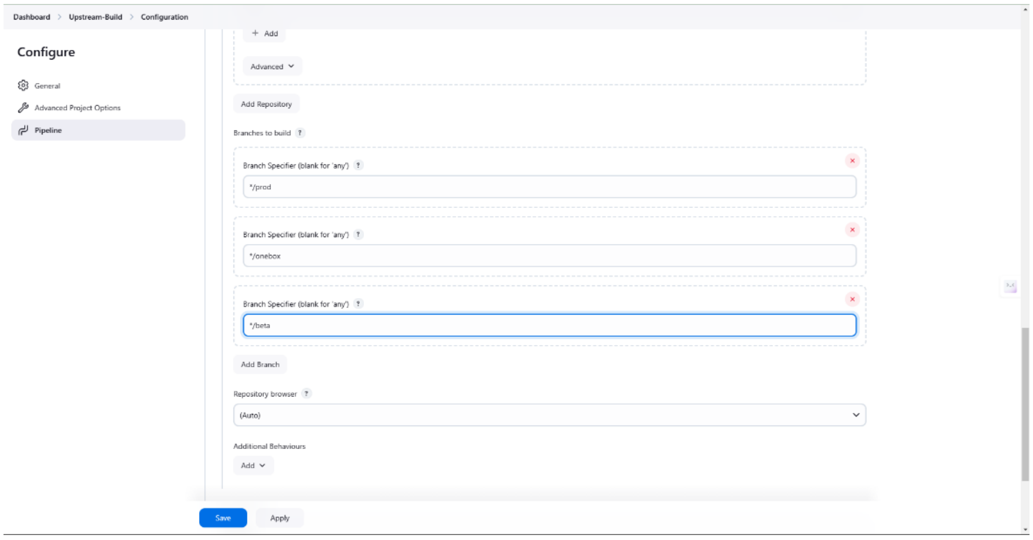
- [ ] Build after other projects are built
- [ ] Build periodically
- [✓] GitHub hook trigger for GITScm polling
- [✓] Poll SCM

Schedule

```
0 1 * * *
```

⚠ Spread load evenly by using 'H 1 * * *' rather than '0 1 * * *'
Would last have run at Tuesday, January 14, 2025 at 1:00:11 AM Coordinated Universal Time; would next run at Wednesday, January 15, 2025 at 1:00:11 AM Coordinated Universal Time.

- [✓] Ignore post-commit hooks
- [ ] Quiet period
- [ ] Trigger builds remotely (e.g., from scripts)


Dashboard > Upstream-Build > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Branches to build

Branch Specifier (blank for 'any'): */prod

Branch Specifier (blank for 'any'): */onebox

Branch Specifier (blank for 'any'): */beta

Repository browser: (Auto)

Additional Behaviours

- Create a File named as Jenkinsfile in the Repo and add the below pipeline, stages

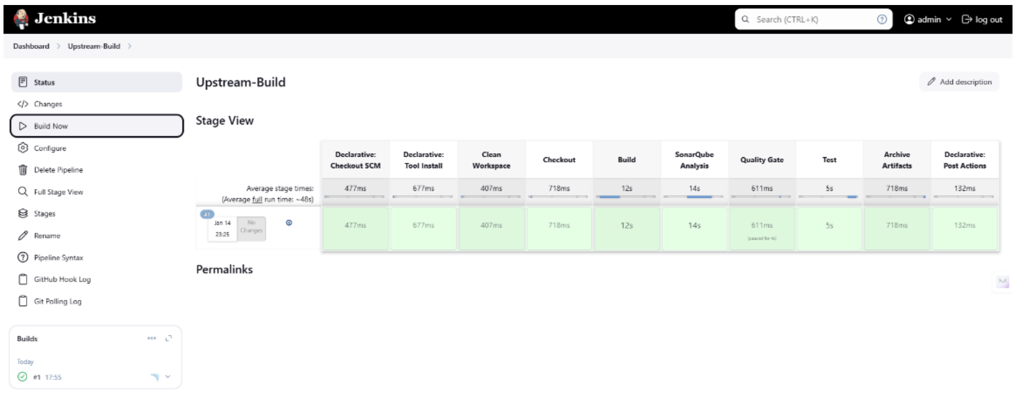
```
pipeline {
agent any

tools {
maven 'maven3'
}
environment {
SCANNER_HOME = tool 'sonar-scanner'
SONARQUBE_SERVER = 'sonar-server' // The name of your SonarQube server
SLACK_CHANNEL = 'jenkins-alert' // Slack channel for notifications
ARTIFACTS_DIR = "target"  // Directory for generated artifacts
MAVEN_OPTS = '--add-opens java.base/java.lang=ALL-UNNAMED'
```

```
}

stages {
// Clean the workspace before starting the build
stage('Clean Workspace') {
steps {
cleanWs()  // Clean the workspace before starting the build
}
}
// Checkout the code from the GitHub repository
stage('Checkout') {
steps {
checkout scm
}
}

// Build the project using Maven and generate artifacts
stage('Build') {
steps {
script {
echo 'Building the project...'
sh 'mvn clean install'  // Build the project using Maven
}
}
}

// Stage 3: Run SonarQube analysis to check code quality
stage('SonarQube Analysis') {
steps {
withSonarQubeEnv(SONARQUBE_SERVER) {
sh '''
```

```
$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Java \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=Java
'''
}
}
}

//  Wait for the Quality Gate to pass or fail
stage('Quality Gate') {
steps {
script {
waitForQualityGate abortPipeline: true, credentialsId: 'sonar-token'
}
}
}

//  Run unit tests and generate test reports in HTML format
stage('Test') {
steps {
script {
echo 'Running tests...'
sh 'mvn test'  // Run unit tests
junit '**/target/surefire-reports/*.xml'  // Publish JUnit test results
publishHTML(target: [
reportName: 'Test Results',
reportDir: 'target/surefire-reports',  // Ensure this is the correct directory for the HTML report
reportFiles: 'surefire-report.html',  // Ensure this is the correct file for the HTML report
keepAll: true
])
```

```
}
}
}
// Archive generated artifacts for downstream job
stage('Archive Artifacts') {
steps {
script {
sh 'ls -al target'  // List the contents of the target directory
archiveArtifacts artifacts: 'target/*.war', allowEmptyArchive: false
}
}
}

}
post {
success {
slackSend (channel: SLACK_CHANNEL, message: "Pipeline completed successfully :tada:",
color: 'good')
}
failure {
slackSend (channel: SLACK_CHANNEL, message: "Pipeline failed :x:", color: 'danger')
}
}
}
```

- After adding the above pipeline run the build Manually


Dashboard > Upstream-Build >

Upstream-Build

Stage View

|  | Declarative: Checkout SCM | Declarative: Tool Install | Clean Workspace | Checkout | Build | SonarQube Analysis | Quality Gate | Test | Archive Artifacts | Declarative: Post Actions |
|---|---|---|---|---|---|---|---|---|---|---|
| Average stage times: (Average full run time: ~48s) | 477ms | 677ms | 407ms | 718ms | 12s | 14s | 611ms | 5s | 718ms | 132ms |
| #1 Jan 14 23:25 No Changes | 477ms | 677ms | 407ms | 718ms | 12s | 14s | 611ms (paused for 4s) | 5s | 718ms | 132ms |

Permalinks

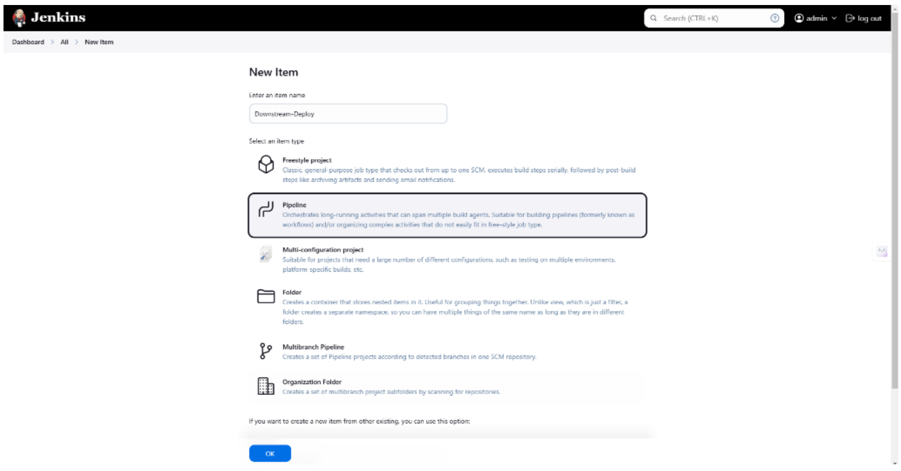
Builds

Today

#1 17:55

**Step 2:** Downstream Job

- Go to your Jenkins dashboard **New Items** > **Pipeline**
- Follow the below Steps for Downstream


Dashboard > All > New Item

New Item

Enter an item name

Downstream-Deploy

Select an item type

**Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**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.

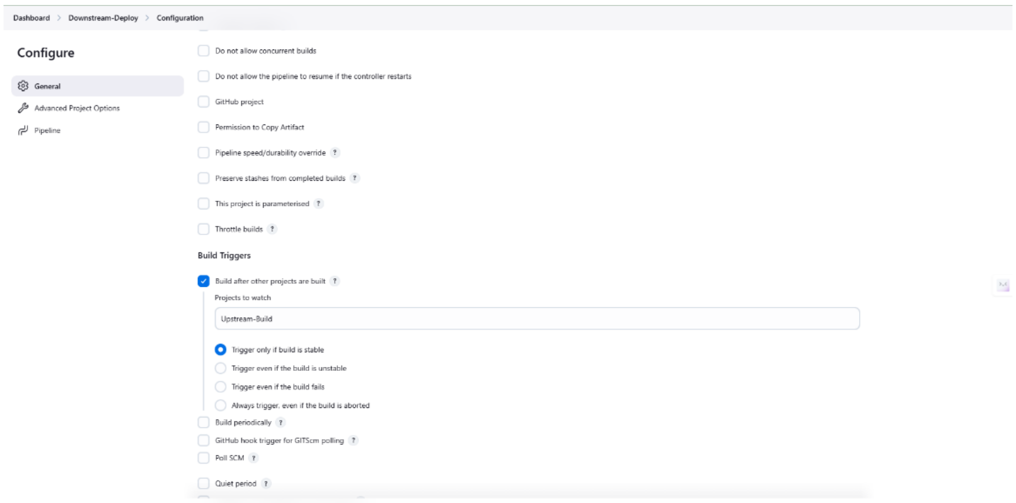
**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**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.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

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


Dashboard > Downstream-Deploy > Configuration

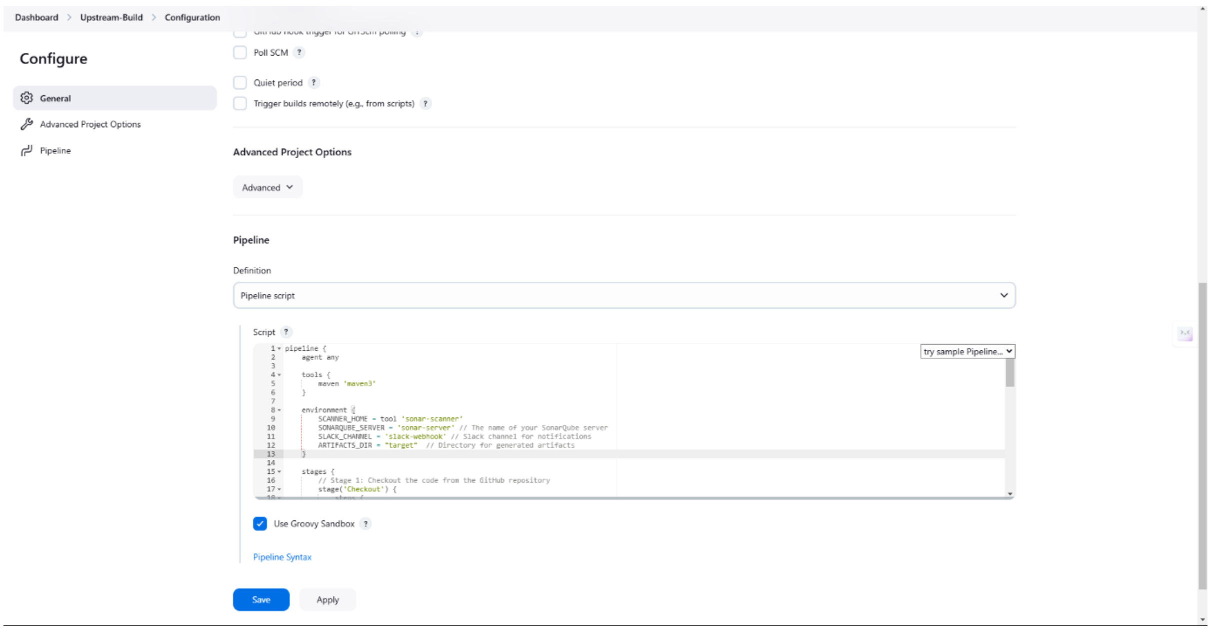
Configure

- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project
- [ ] Permission to Copy Artifact
- [ ] Pipeline speed/durability override ?
- [ ] Preserve stashes from completed builds ?
- [ ] This project is parameterised ?
- [ ] Throttle builds ?

Build Triggers

- [✓] Build after other projects are built ?
  - Projects to watch: Upstream-Build
  - [✓ Trigger only if build is stable]
  - Trigger even if the build is unstable
  - Trigger even if the build fails
  - Always trigger, even if the build is aborted
- [ ] Build periodically ?
- [ ] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?

- Add a pipeline script at the last and use the below pipeline


Dashboard > Upstream-Build > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

- [ ] Poll SCM
- [ ] Quiet period
- [ ] Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Advanced

Pipeline

Definition: Pipeline script

Script

```
pipeline {
    agent any

    tools {
        maven 'maven3'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
        SONARQUBE_SERVER = 'sonar-server' // The name of your SonarQube server
        SLACK_CHANNEL = 'slack-webhook' // Slack channel for notifications
        ARTIFACTS_DIR = "target"  // Directory for generated artifacts
    }

    stages {
        // Stage 1: Checkout the code from the GitHub repository
        stage('Checkout') {
```

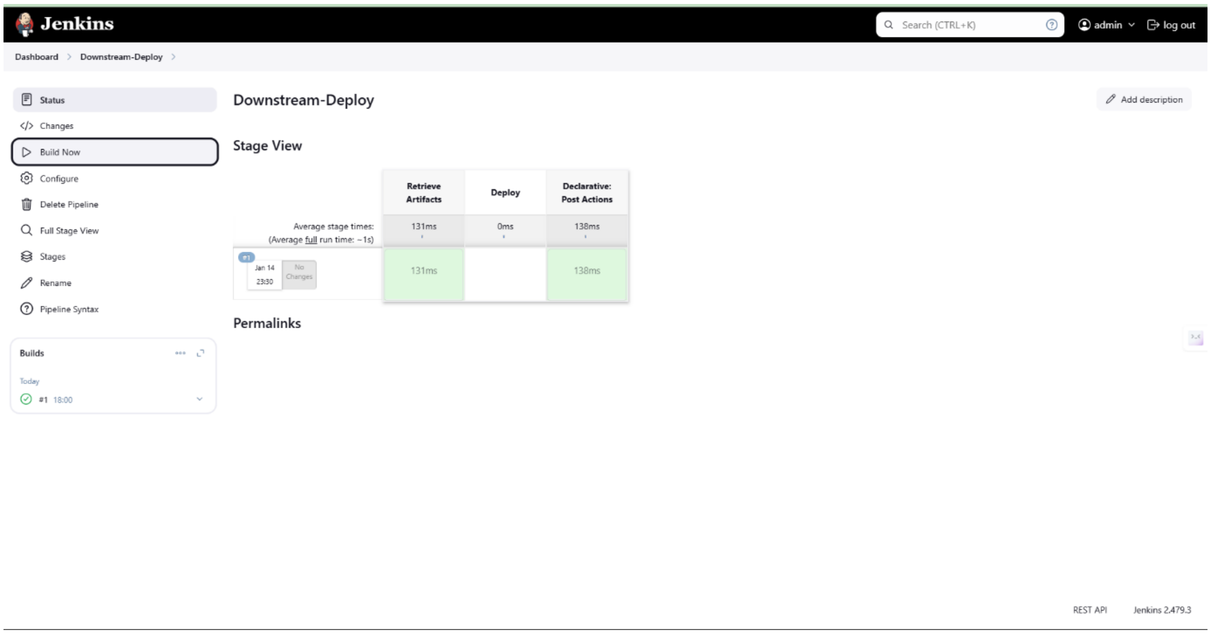
- [✓] Use Groovy Sandbox

Pipeline Syntax

Save Apply

```
pipeline {
  agent any

  environment {
    SLACK_CHANNEL = '#jenkins-alert'  // Slack channel for notifications
  }

  stages {
    // Retrieve artifacts from upstream job
    stage('Retrieve Artifacts') {
      steps {
        copyArtifacts(
          projectName: 'Upstream-Build',  // Replace with your upstream pipeline
name
          filter: 'target/*.war',  // Specify the artifact(s) you want to copy
          target: 'target'  // Directory to copy artifacts to
        )
      }
    }
    // Deploy the application based on the branch
```

```
        stage('Deploy') {
            when {
                anyOf {
                    branch 'prod'
                    branch 'onebox'
                    branch 'beta/*'
                }
            }
            steps {
                script {
                    if (env.BRANCH_NAME == 'prod') {
                        echo "Deploying to production (prod) environment..."
                        sh './deploy-prod.sh'  // Example production deploy script
                    } else if (env.BRANCH_NAME == 'onebox') {
                        echo "Deploying to staging (onebox) environment..."
                        sh './deploy-staging.sh'  // Example staging deploy script
                    } else if (env.BRANCH_NAME.startsWith('beta/')) {
                        echo "Deploying to beta environment for ${env.BRANCH_NAME}..."
                        sh './deploy-beta.sh'  // Example beta-specific deploy script
                    }
                }
            }
        }

    }

    post {
        success {
            slackSend(channel: SLACK_CHANNEL, message: "Downstream Build
${env.BUILD_ID} completed successfully :tada:", color: 'good')
        }
        failure {
            slackSend(channel: SLACK_CHANNEL, message: "Downstream Build
${env.BUILD_ID} failed. Please check the logs! :x:", color: 'danger')
        }
    }
}
```



|  | Retrieve Artifacts | Deploy | Declarative: Post Actions |
|---|---|---|---|
| Average stage times: (Average full run time: ~1s) | 131ms | 0ms | 138ms |
| #1 Jan 14 23:30 No Changes | 131ms |  | 138ms |

Now whenever you build the Upstream get success Downstream will get triggered automatically

## Region-Specific Deployments

- **Step 1:** Define region-based deployment logic within your build scripts (e.g., using environment variables for AWS region).
- **Step 2:** Use Jenkins environment variables or parameters to pass region information to the deployment scripts.

---

This breakdown will help you set up a fully functioning Jenkins CI/CD pipeline with integrated GitHub, quality scanning, notifications, build automation, and deployments across different environments.