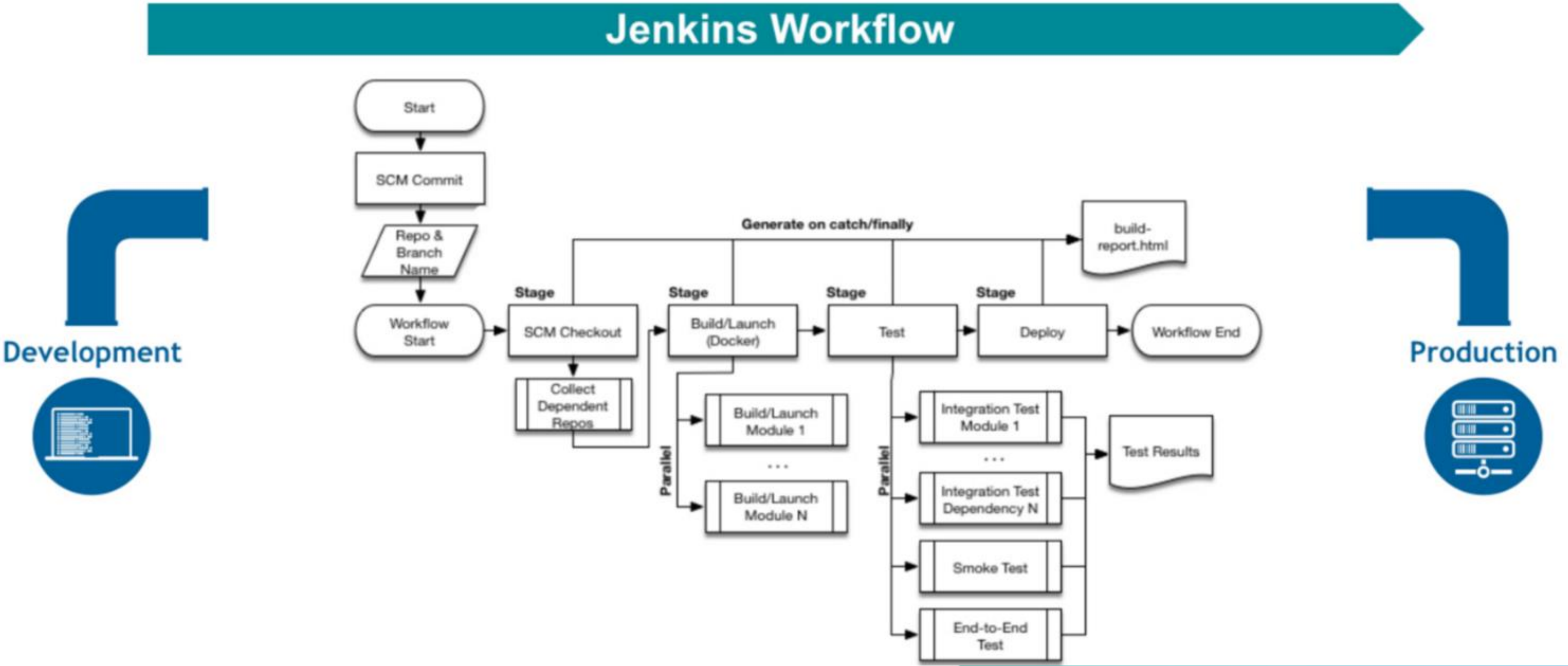
# Jenkins – Continuous Integration

# Jenkins

- Jenkins is a open source tool that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on
- It is a free source that can handle any kind of build or continuous integration
- You can integrate Jenkins with a number of testing and deployment technologies

# Why Jenkins

- Jenkins is a software that allows **continuous integration**
- Jenkins will be installed on a server where the central build will take place



Jenkins Workflow

Development

Production

Start

SCM Commit

Repo & Branch Name

Workflow Start

Generate on catch/finally

build-report.html

Stage

SCM Checkout

Collect Dependent Repos

Stage

Build/Launch (Docker)

Parallel

Build/Launch Module 1

...

Build/Launch Module N

Stage

Test

Parallel

Integration Test Module 1

...

Integration Test Dependency N

Smoke Test

End-to-End Test

Test Results

Stage

Deploy

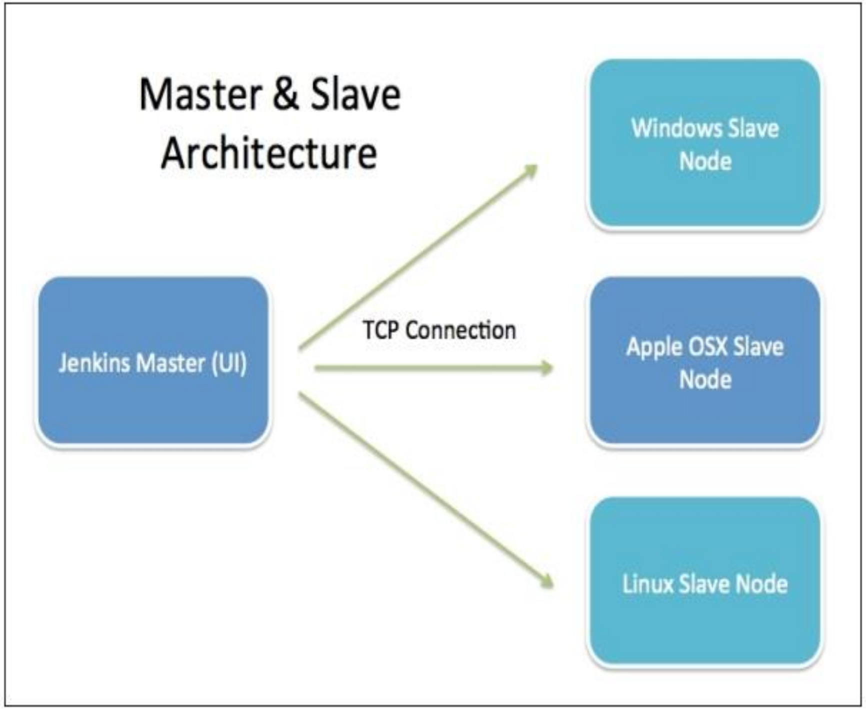
Workflow End

# What is Continuous Integration

- Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals
- This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle
- Continuous integration requires the developers to have frequent builds
- The common practice is that whenever a code commit occurs, a build should be triggered
- This can be done on any technology or platform

# Jenkins Architecture

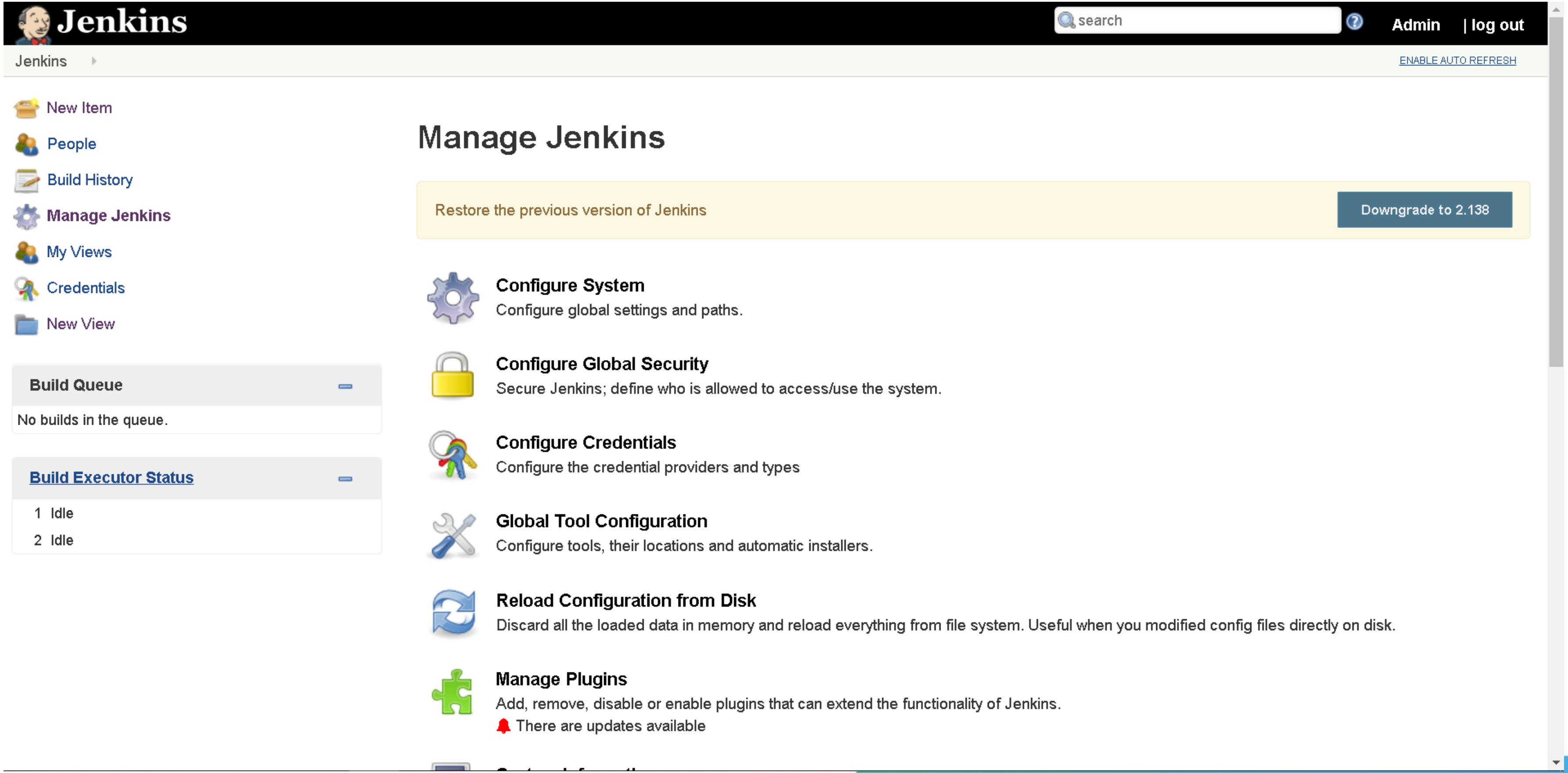
- Jenkins Architecture is based on the distributed. This has 2 components.
  - Jenkins Server
  - Jenkins Node/Slave/Build Server
- Your main Jenkins server is the master
- The master's job is to handle scheduling build jobs, dispatching builds to the slaves for the actual execution, monitor the slaves and recording and presenting the build results
- Even in a distributed architecture, a master instance of Jenkins can also execute build jobs directly
- The job of the slaves is to do as they are configured in the Jenkins Server, which involves executing build jobs dispatched by the master



# Installing Jenkins

- Follow lab document to install and configure Jenkins

# Jenkins Configuration – Manage Jenkins



**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

## Manage Jenkins

Restore the previous version of Jenkins

**Configure System**

Configure global settings and paths.

**Configure Global Security**

Secure Jenkins; define who is allowed to access/use the system.

**Configure Credentials**

Configure the credential providers and types

**Global Tool Configuration**

Configure tools, their locations and automatic installers.

**Reload Configuration from Disk**

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

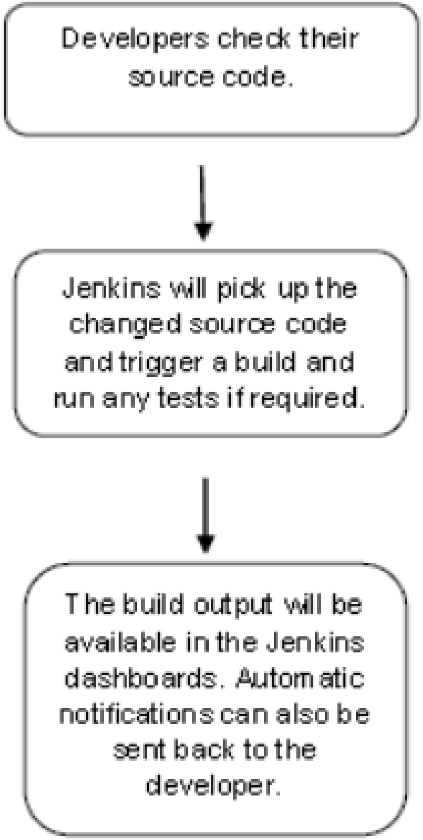
**Manage Plugins**

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

There are updates available

# Why Jenkins?

- Jenkins is a software that allows **continuous integration**
- Jenkins will be installed on a server where the central build will take place
- The following flowchart demonstrates a very simple workflow of how Jenkins works:



Developers check their source code.

Jenkins will pick up the changed source code and trigger a build and run any tests if required.

The build output will be available in the Jenkins dashboards. Automatic notifications can also be sent back to the developer.

# Installing Git

- Download git and install on your VM (already done in Git lesson)

# Installing Maven

- Maven is an open source build tool for many platforms
- Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information
- Using maven we can build and manage any Java based project
- The primary goal of Maven is to provide developer with the following –
  - A comprehensive model for projects, which is reusable, maintainable, and easier to comprehend
  - Plugins or tools that interact with this declarative model
- To install maven run the following commands:

```
apt-get update
apt-get install maven
```

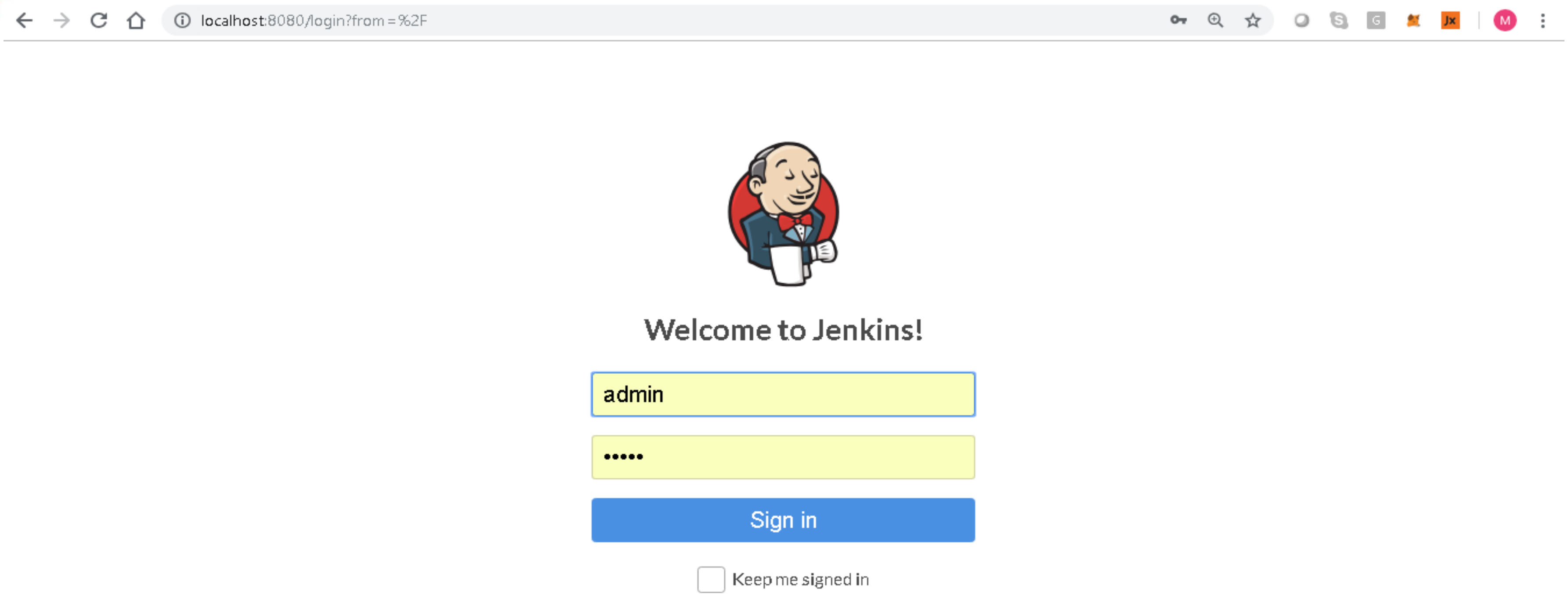
- To verify Maven installation run:

```
mvn --version
```

# Install Tomcat

- Go to http://tomcat.apache.org/
- Download the Ubuntu distribution and install it on your VM

# Jenkins Login



localhost:8080/login?from=%2F

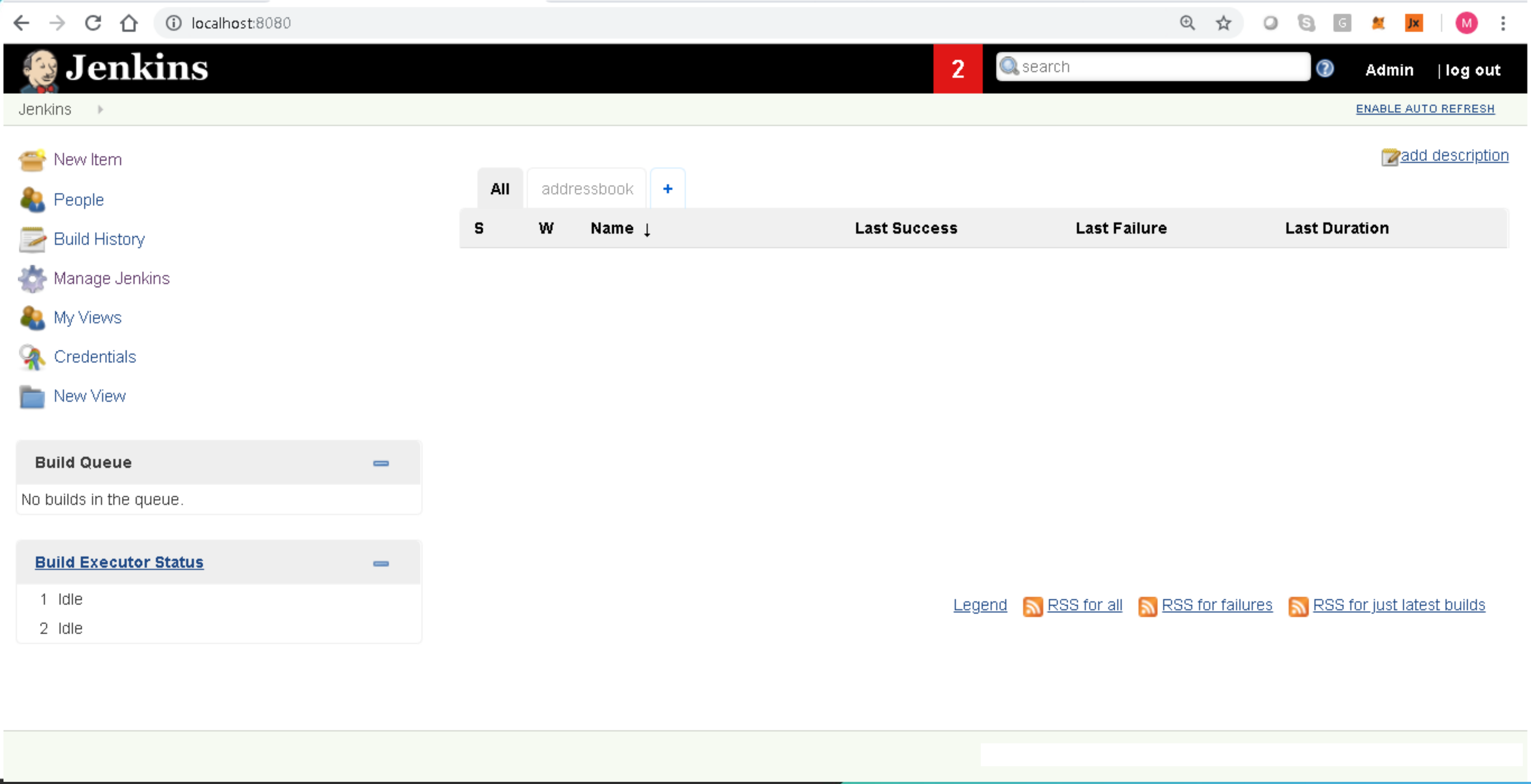
Welcome to Jenkins!

admin

•••••

□ Keep me signed in

# Jenkins Dashboard



localhost:8080

Jenkins

2

search

Admin | log out

Jenkins

ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Credentials
- New View

add description

All | addressbook | +

| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|---|---|---|---|---|

**Build Queue**

No builds in the queue.

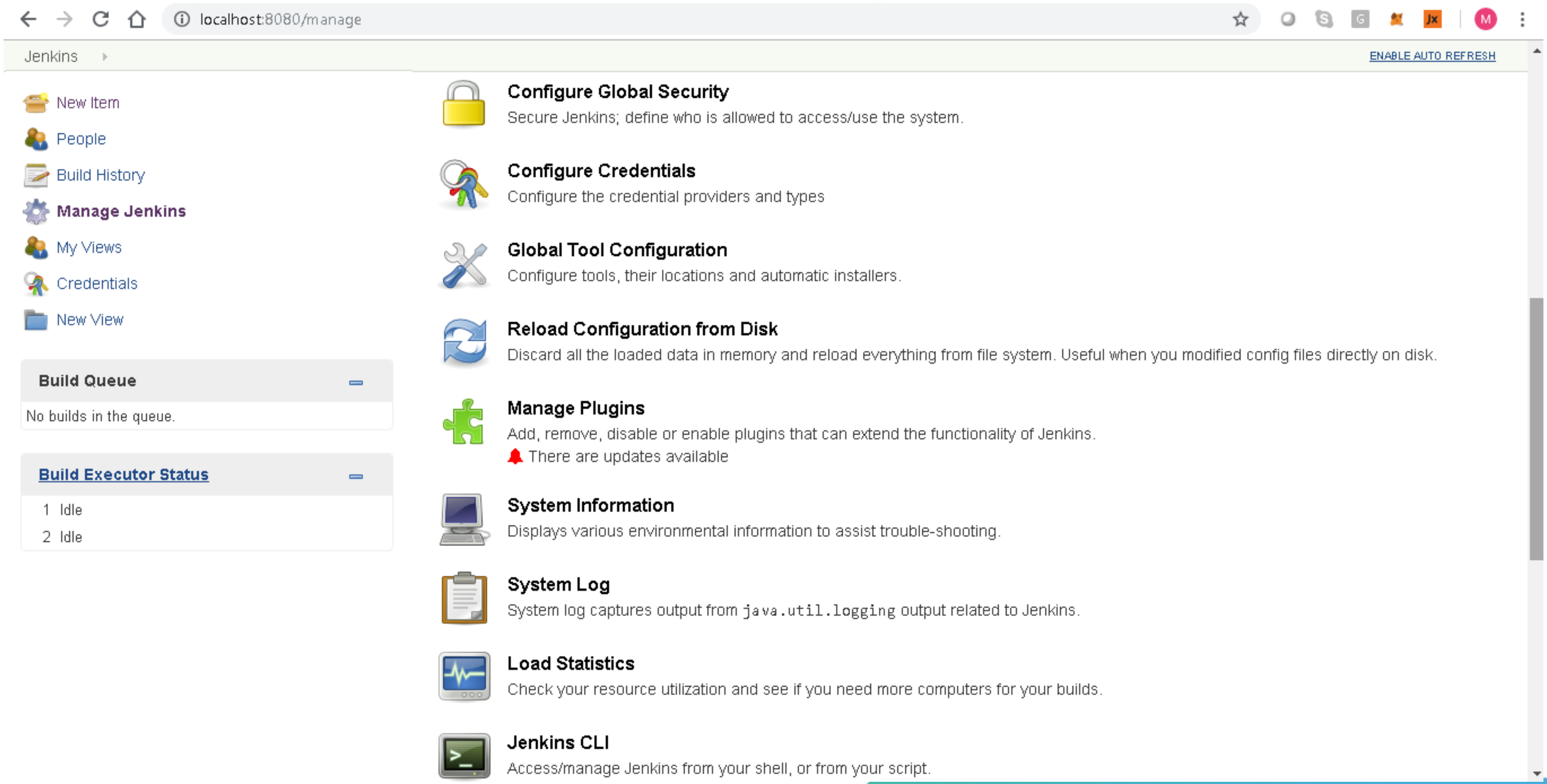
**Build Executor Status**

1 Idle

2 Idle

Legend RSS for all RSS for failures RSS for just latest builds

# Manage Jenkins



**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle
2 Idle

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

**Configure Credentials**
Configure the credential providers and types

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

**Reload Configuration from Disk**
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

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

**System Information**
Displays various environmental information to assist trouble-shooting.

**System Log**
System log captures output from `java.util.logging` output related to Jenkins.

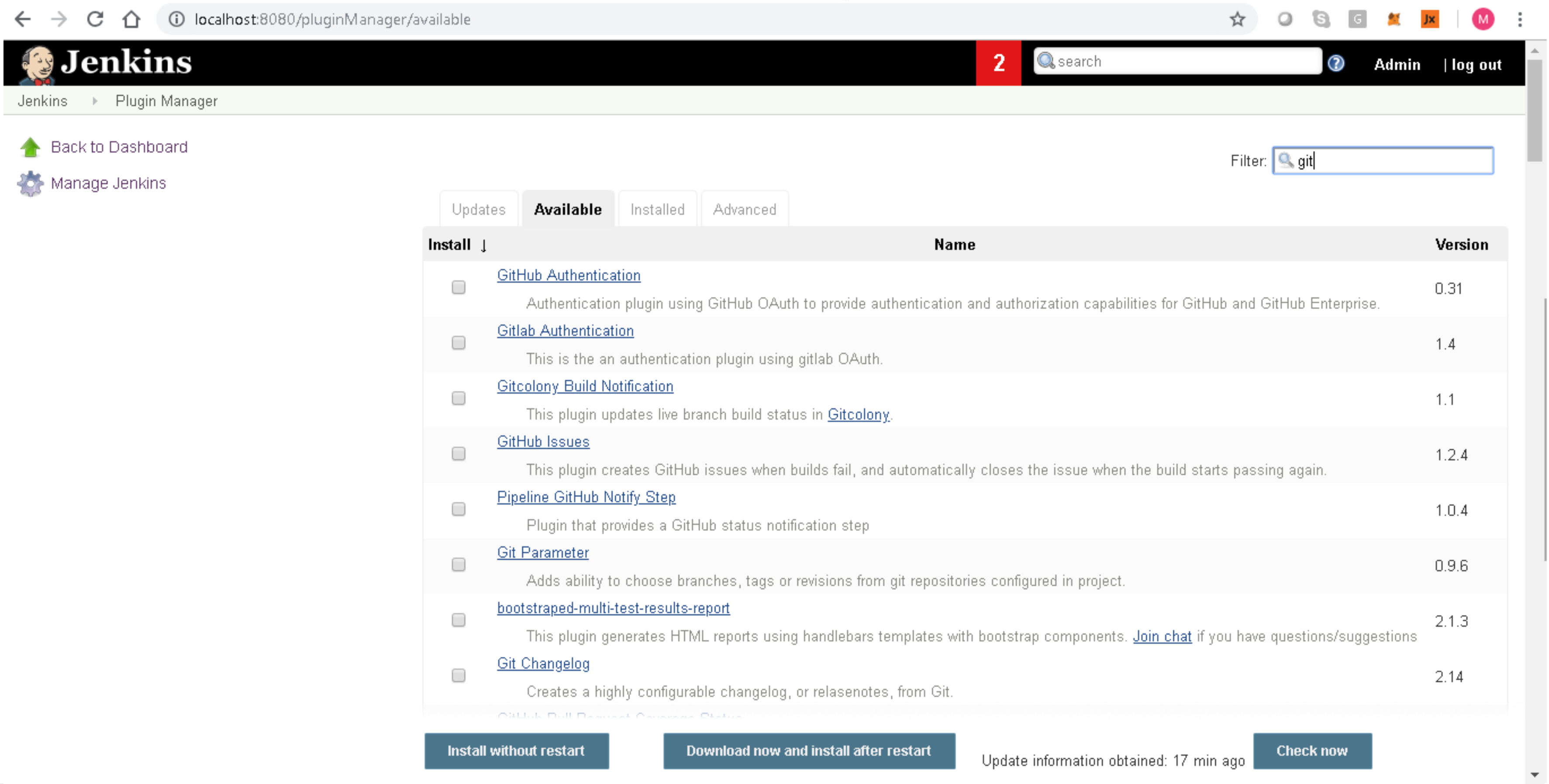
**Load Statistics**
Check your resource utilization and see if you need more computers for your builds.

**Jenkins CLI**
Access/manage Jenkins from your shell, or from your script.

# Manage Plugins

- Click the ‘Manage Plugins’ option
- Click the Available tab
- This tab will give a list of plugins which are available for downloading
- In the ‘Filter’ tab type ‘Git plugin’
- The list will then be filtered
- Check the Git Plugin option and click on the button ‘Install without restart’
- The installation will then begin and the screen will be refreshed to show the status of the download

# Install Plugins

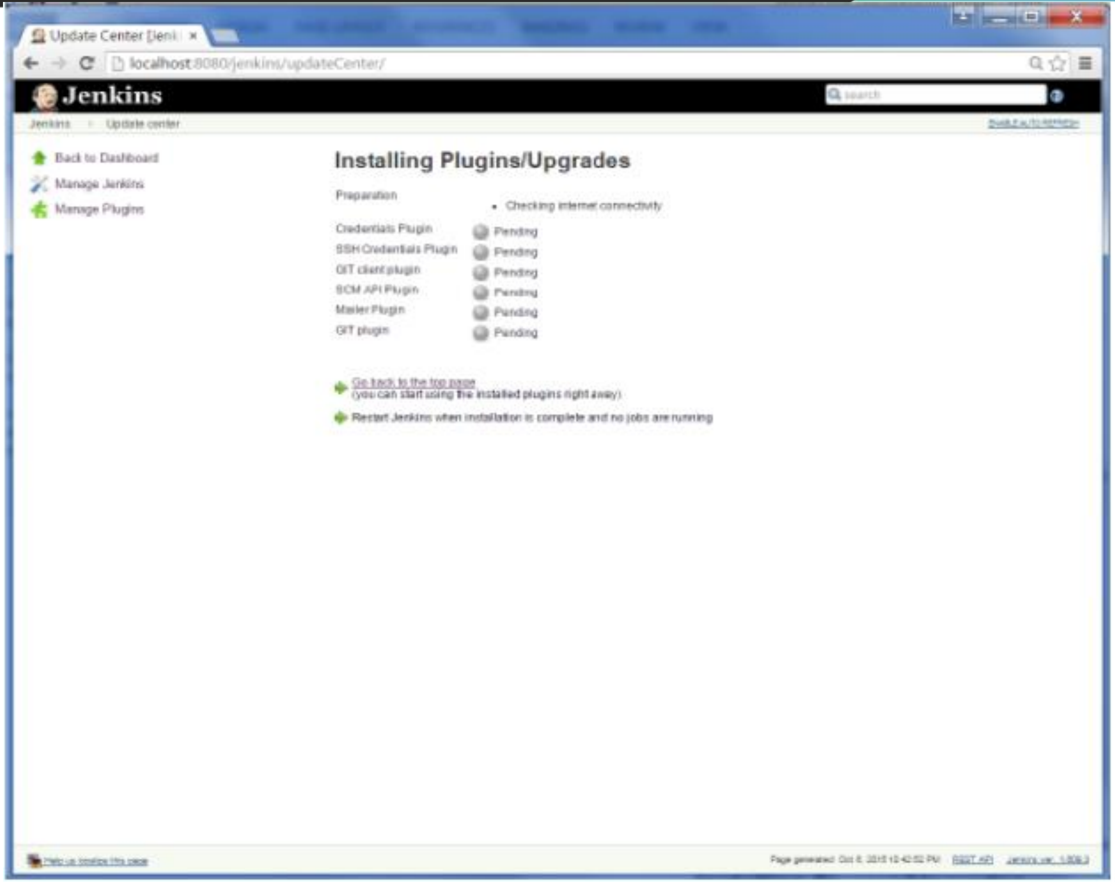


Filter: git

| Install ↓ | Name | Version |
|---|---|---|
| □ | GitHub Authentication<br>Authentication plugin using GitHub OAuth to provide authentication and authorization capabilities for GitHub and GitHub Enterprise. | 0.31 |
| □ | Gitlab Authentication<br>This is the an authentication plugin using gitlab OAuth. | 1.4 |
| □ | Gitcolony Build Notification<br>This plugin updates live branch build status in Gitcolony. | 1.1 |
| □ | GitHub Issues<br>This plugin creates GitHub issues when builds fail, and automatically closes the issue when the build starts passing again. | 1.2.4 |
| □ | Pipeline GitHub Notify Step<br>Plugin that provides a GitHub status notification step | 1.0.4 |
| □ | Git Parameter<br>Adds ability to choose branches, tags or revisions from git repositories configured in project. | 0.9.6 |
| □ | bootstraped-multi-test-results-report<br>This plugin generates HTML reports using handlebars templates with bootstrap components. Join chat if you have questions/suggestions | 2.1.3 |
| □ | Git Changelog<br>Creates a highly configurable changelog, or relasenotes, from Git. | 2.14 |

Update information obtained: 17 min ago

# Install Plugins



## Installing Plugins/Upgrades

Preparation

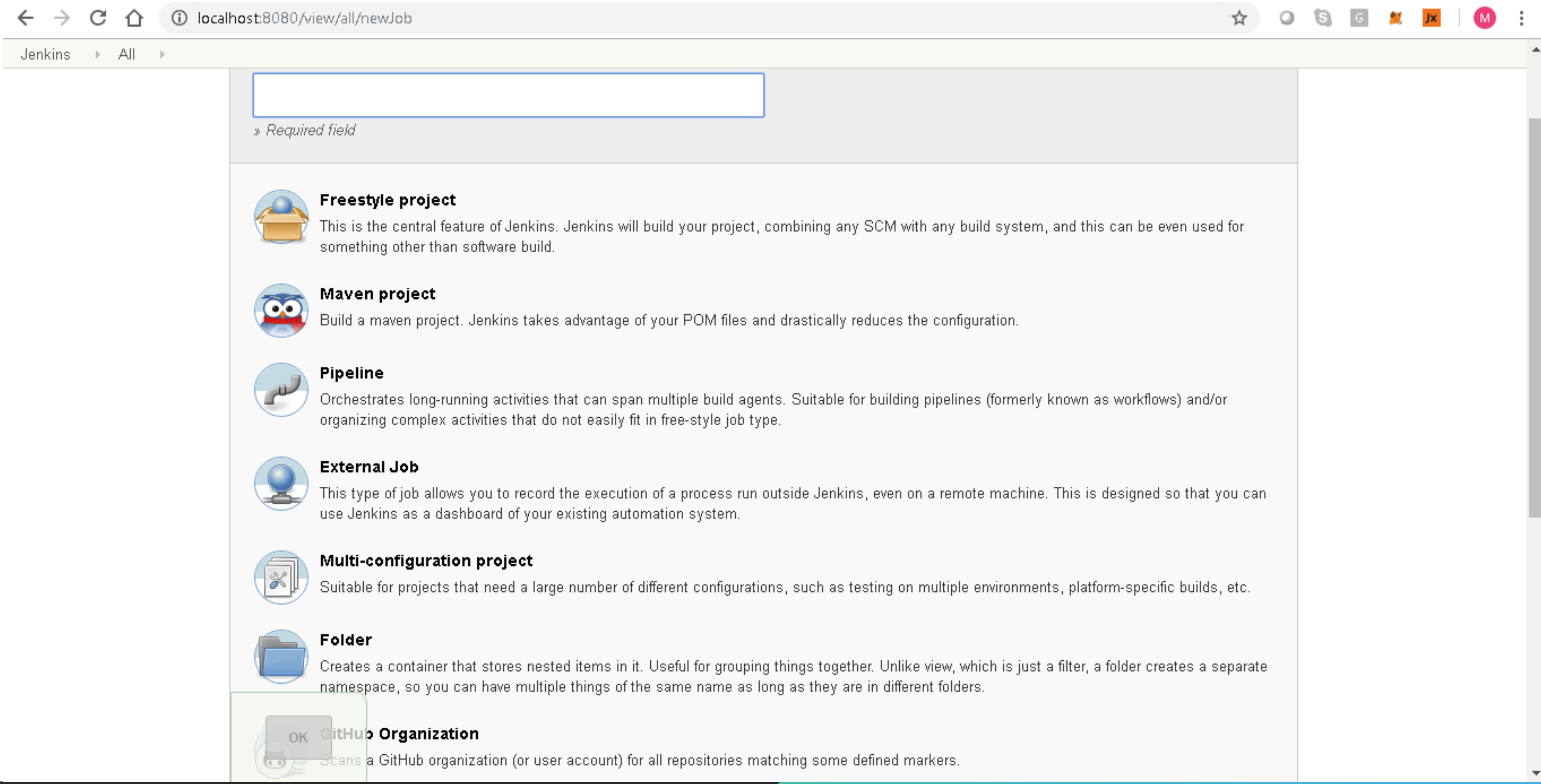
- Checking internet connectivity

| | |
|---|---|
| Credentials Plugin | Pending |
| SSH Credentials Plugin | Pending |
| GIT client plugin | Pending |
| SCM API Plugin | Pending |
| Mailer Plugin | Pending |
| GIT plugin | Pending |

Go back to the top page
(you can start using the installed plugins right away)

Restart Jenkins when installation is complete and no jobs are running

# Creating Jobs



» Required field

**Freestyle project**

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

**Maven project**

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

**Pipeline**

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

**External Job**

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

**Multi-configuration project**

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

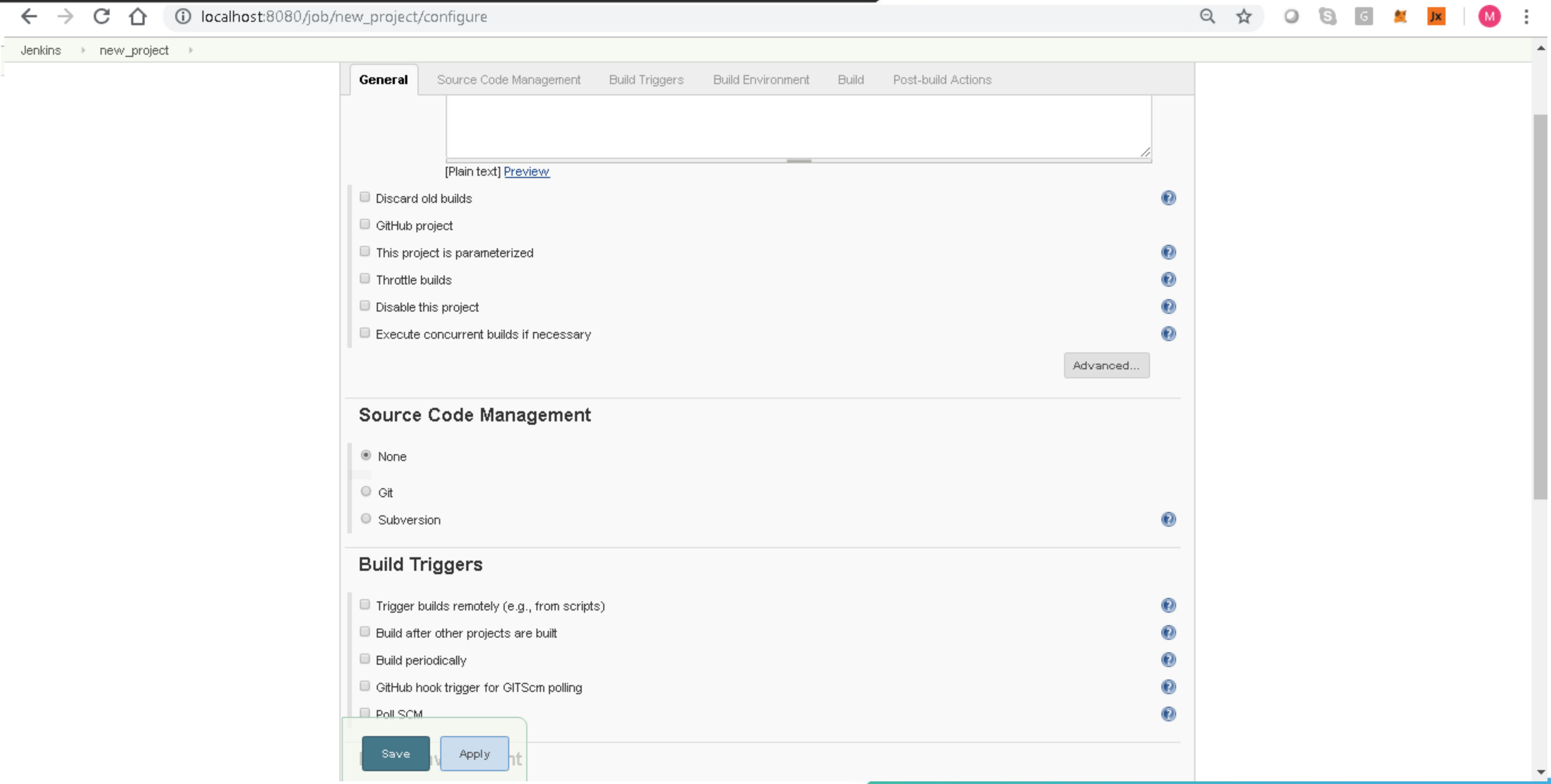
**Folder**

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

**GitHub Organization**

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

# Creating Jobs



localhost:8080/job/new_project/configure

Jenkins › new_project ›

[Plain text] Preview

□ Discard old builds

□ GitHub project

□ This project is parameterized

□ Throttle builds

□ Disable this project

□ Execute concurrent builds if necessary

Source Code Management

● None

○ Git

○ Subversion

Build Triggers

□ Trigger builds remotely (e.g., from scripts)

□ Build after other projects are built

□ Build periodically

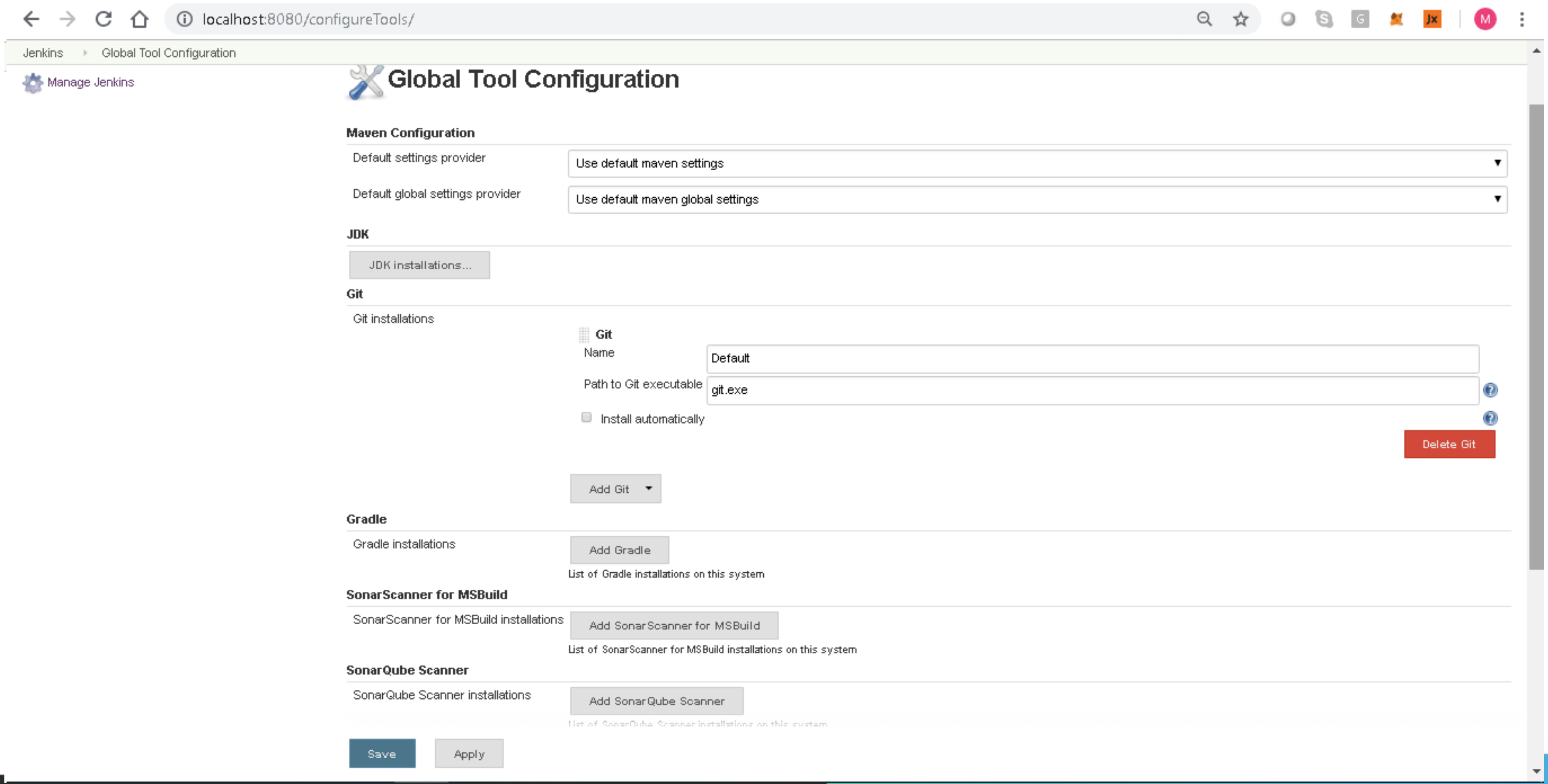
□ GitHub hook trigger for GITScm polling

□ Poll SCM

# Setting Up Maven in Jenkins

- Click Manage Jenkins
- Click Global Tool Configuration
- Click add Maven and enter Maven home path from your VM

# Global Tool Configuration



localhost:8080/configureTools/

Jenkins › Global Tool Configuration

Manage Jenkins

## Global Tool Configuration

### Maven Configuration

Default settings provider: Use default maven settings

Default global settings provider: Use default maven global settings

### JDK

### Git

Git installations

Git

Name: Default

Path to Git executable: git.exe

□ Install automatically

### Gradle

Gradle installations

List of Gradle installations on this system

### SonarScanner for MSBuild

SonarScanner for MSBuild installations

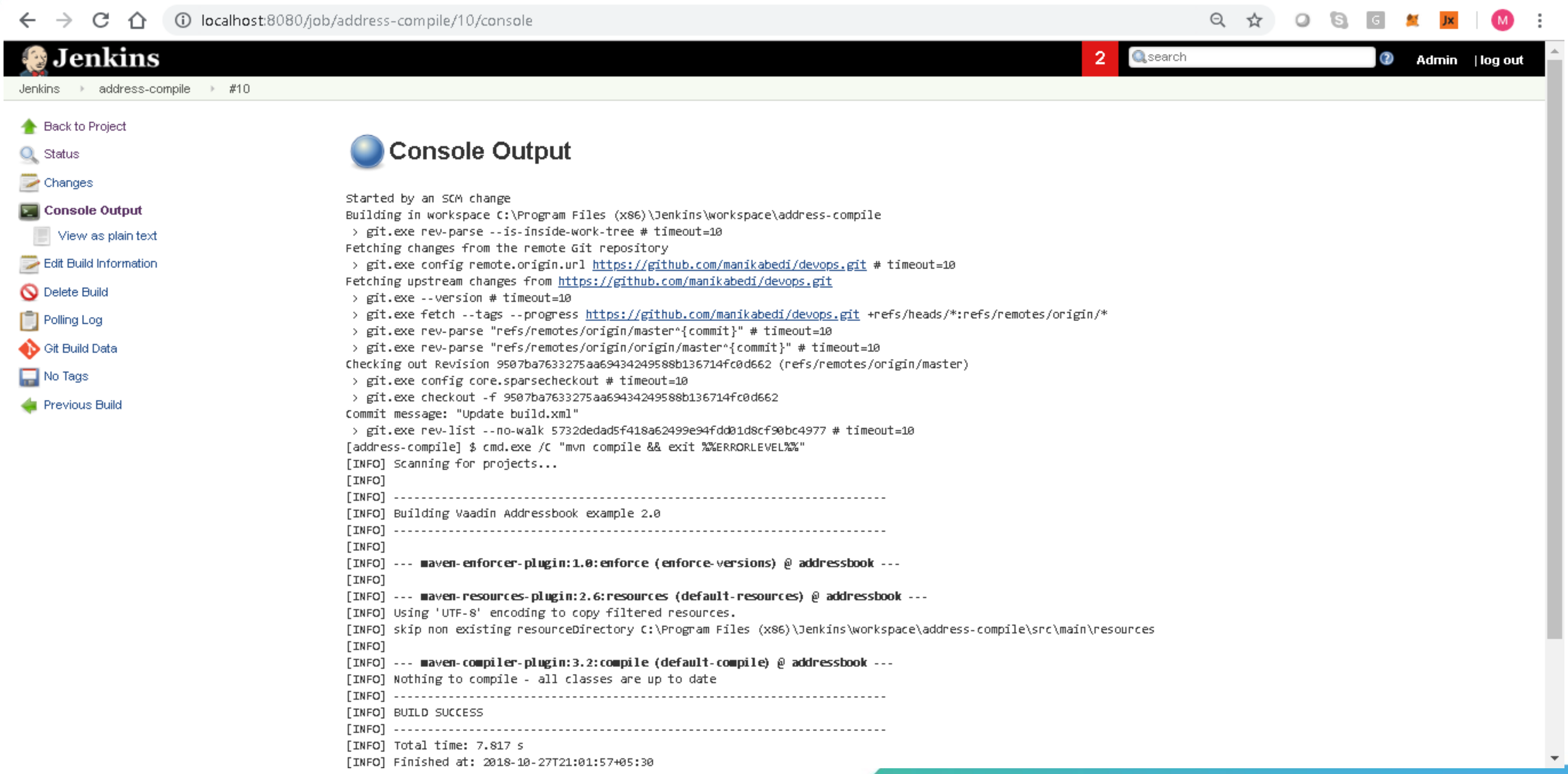
List of SonarScanner for MSBuild installations on this system

### SonarQube Scanner

SonarQube Scanner installations

List of SonarQube Scanner installations on this system

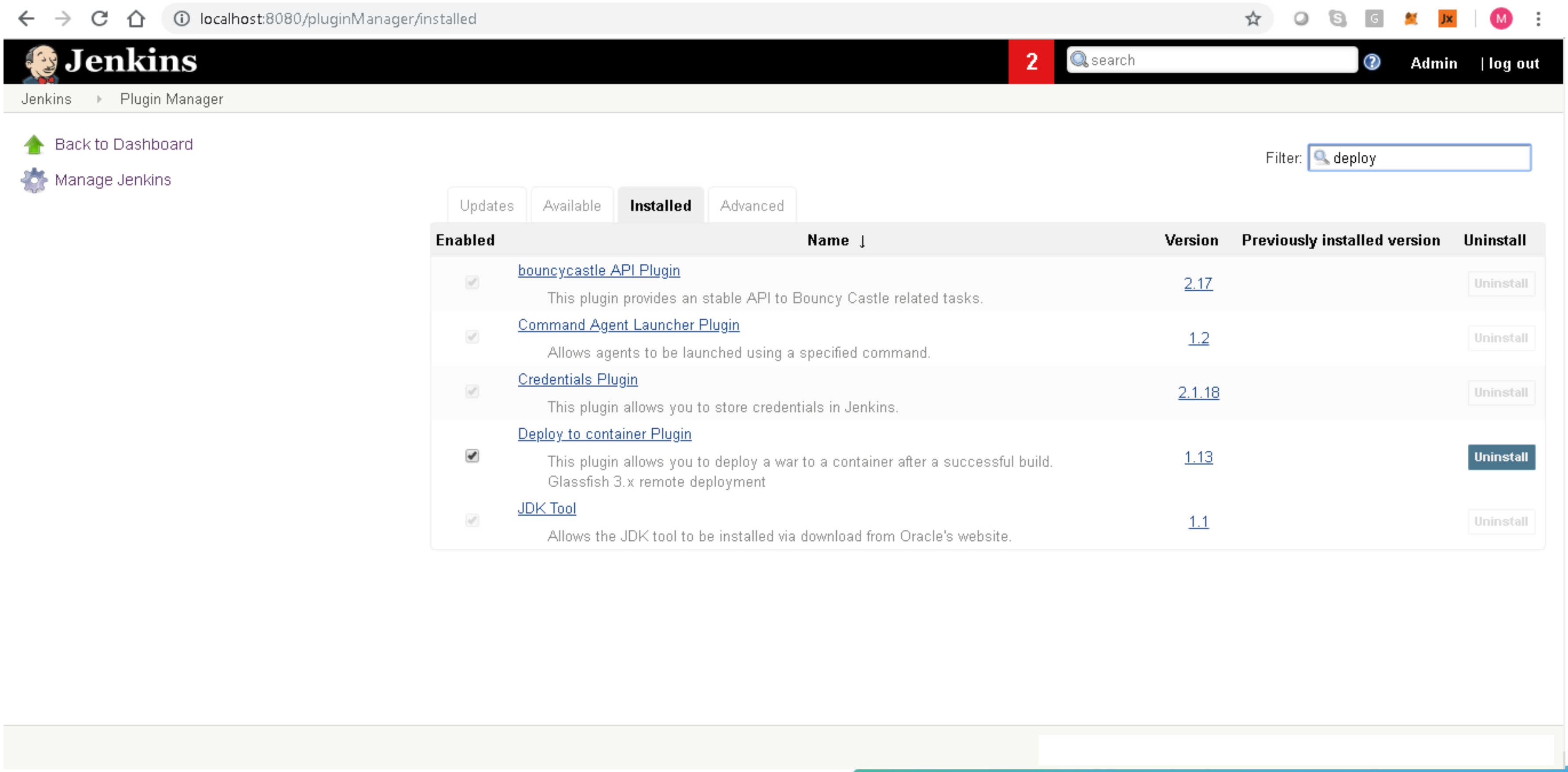
# Running Job



## Console Output

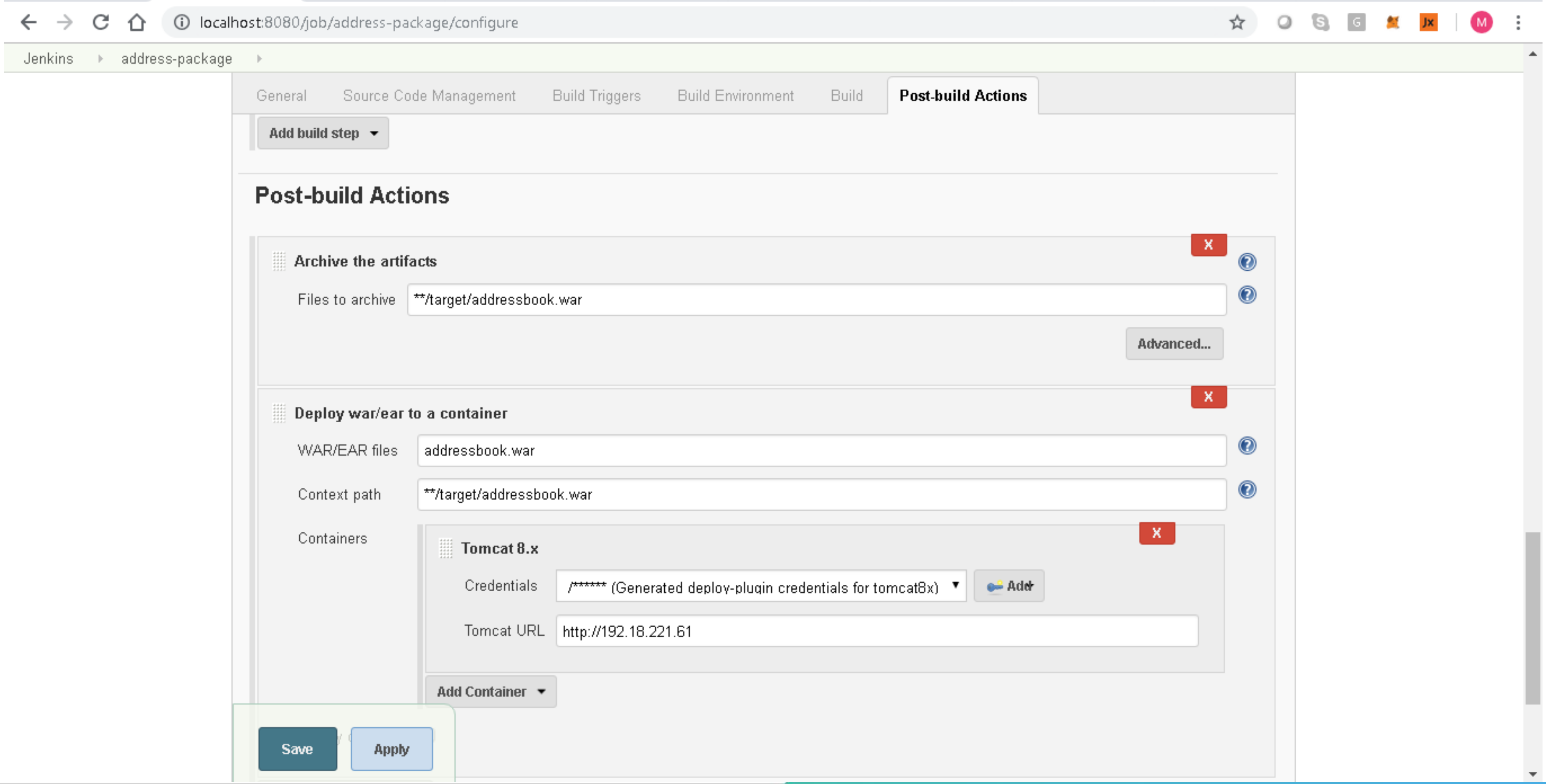
```
Started by an SCM change
Building in workspace C:\Program Files (x86)\Jenkins\workspace\address-compile
 > git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/manikabedi/devops.git # timeout=10
Fetching upstream changes from https://github.com/manikabedi/devops.git
 > git.exe --version # timeout=10
 > git.exe fetch --tags --progress https://github.com/manikabedi/devops.git +refs/heads/*:refs/remotes/origin/*
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
 > git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 9507ba7633275aa69434249588b136714fc0d662 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 9507ba7633275aa69434249588b136714fc0d662
Commit message: "Update build.xml"
 > git.exe rev-list --no-walk 5732dedad5f418a62499e94fdd01d8cf90bc4977 # timeout=10
[address-compile] $ cmd.exe /C "mvn compile && exit %%ERRORLEVEL%%"
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Vaadin Addressbook example 2.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-enforcer-plugin:1.0:enforce (enforce-versions) @ addressbook ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ addressbook ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Program Files (x86)\Jenkins\workspace\address-compile\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.2:compile (default-compile) @ addressbook ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 7.817 s
[INFO] Finished at: 2018-10-27T21:01:57+05:30
```

# Jenkins Deployment Plugin



| Enabled | Name ↓ | Version | Previously installed version | Uninstall |
|---|---|---|---|---|
| ☑ | bouncycastle API Plugin<br>This plugin provides an stable API to Bouncy Castle related tasks. | 2.17 | | Uninstall |
| ☑ | Command Agent Launcher Plugin<br>Allows agents to be launched using a specified command. | 1.2 | | Uninstall |
| ☑ | Credentials Plugin<br>This plugin allows you to store credentials in Jenkins. | 2.1.18 | | Uninstall |
| ☑ | Deploy to container Plugin<br>This plugin allows you to deploy a war to a container after a successful build.<br>Glassfish 3.x remote deployment | 1.13 | | Uninstall |
| ☑ | JDK Tool<br>Allows the JDK tool to be installed via download from Oracle's website. | 1.1 | | Uninstall |

# Jenkins Deployment



Post-build Actions

Archive the artifacts

Files to archive: **/target/addressbook.war

Deploy war/ear to a container

WAR/EAR files: addressbook.war

Context path: **/target/addressbook.war

Containers

Tomcat 8.x

Credentials: /****** (Generated deploy-plugin credentials for tomcat8x)

Tomcat URL: http://192.18.221.61

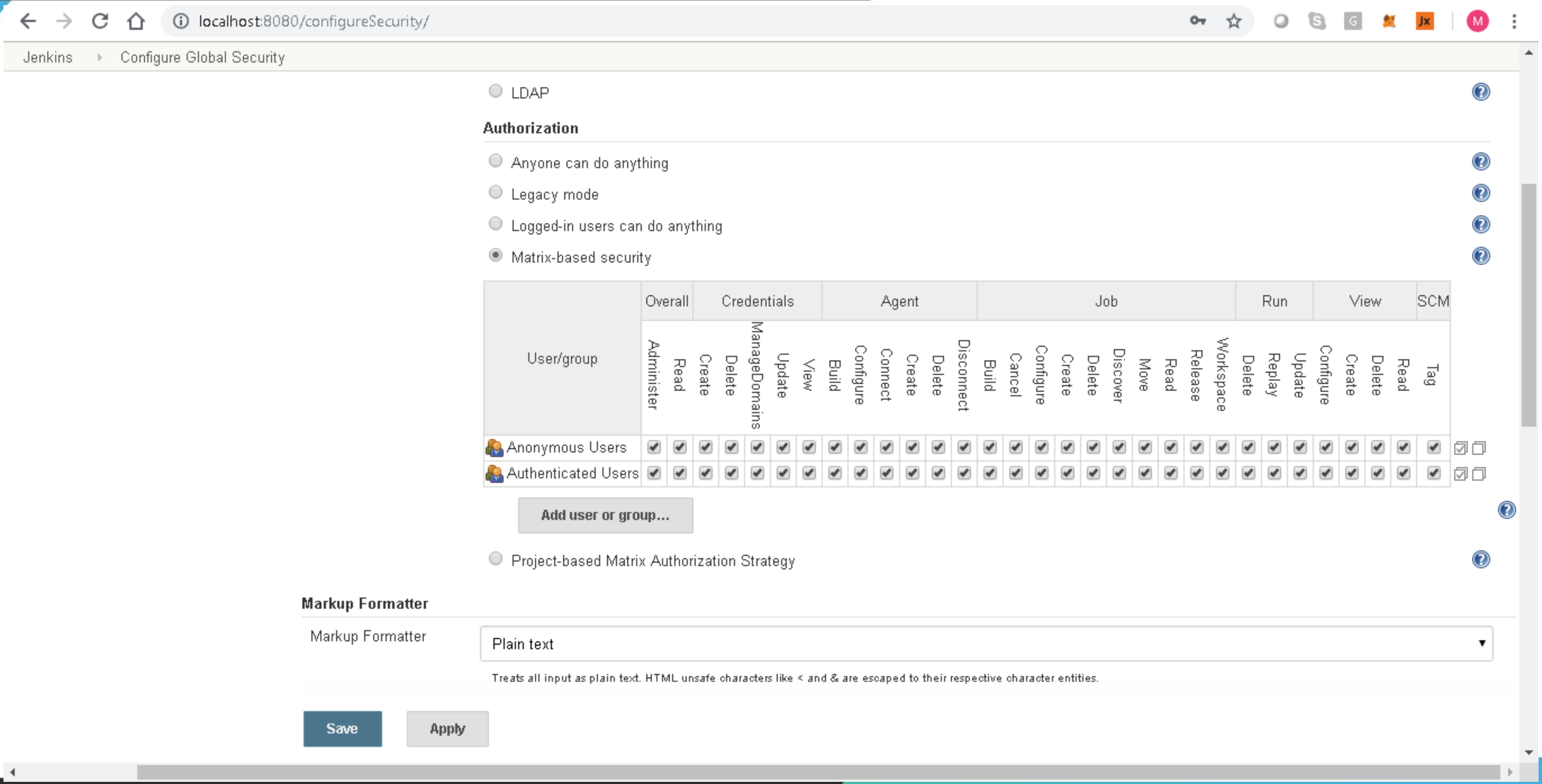
# Integrate Jenkins with Ant

- Install Ant plugin in Jenkins (if not installed as part of initial Jenkins Setup)
- Install Ant on your VM using commands:

```
apt-get update
apt-get install ant
ant --version
```

- Go to Global Tool Configuration in Jenkins
- Click on Add Ant and enter your ant home folder

# Authentication



localhost:8080/configureSecurity/

Jenkins ▸ Configure Global Security

○ LDAP

**Authorization**

○ Anyone can do anything

○ Legacy mode

○ Logged-in users can do anything

● Matrix-based security

| User/group | Overall | | Credentials | | | | | Agent | | | | | | Job | | | | | | | | | | Run | | | View | | | | SCM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Administer | Read | Create | Delete | ManageDomains | Update | View | Build | Configure | Connect | Create | Delete | Disconnect | Build | Cancel | Configure | Create | Delete | Discover | Move | Read | Release | Workspace | Delete | Replay | Update | Configure | Create | Delete | Read | Tag |
| Anonymous Users | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| Authenticated Users | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |

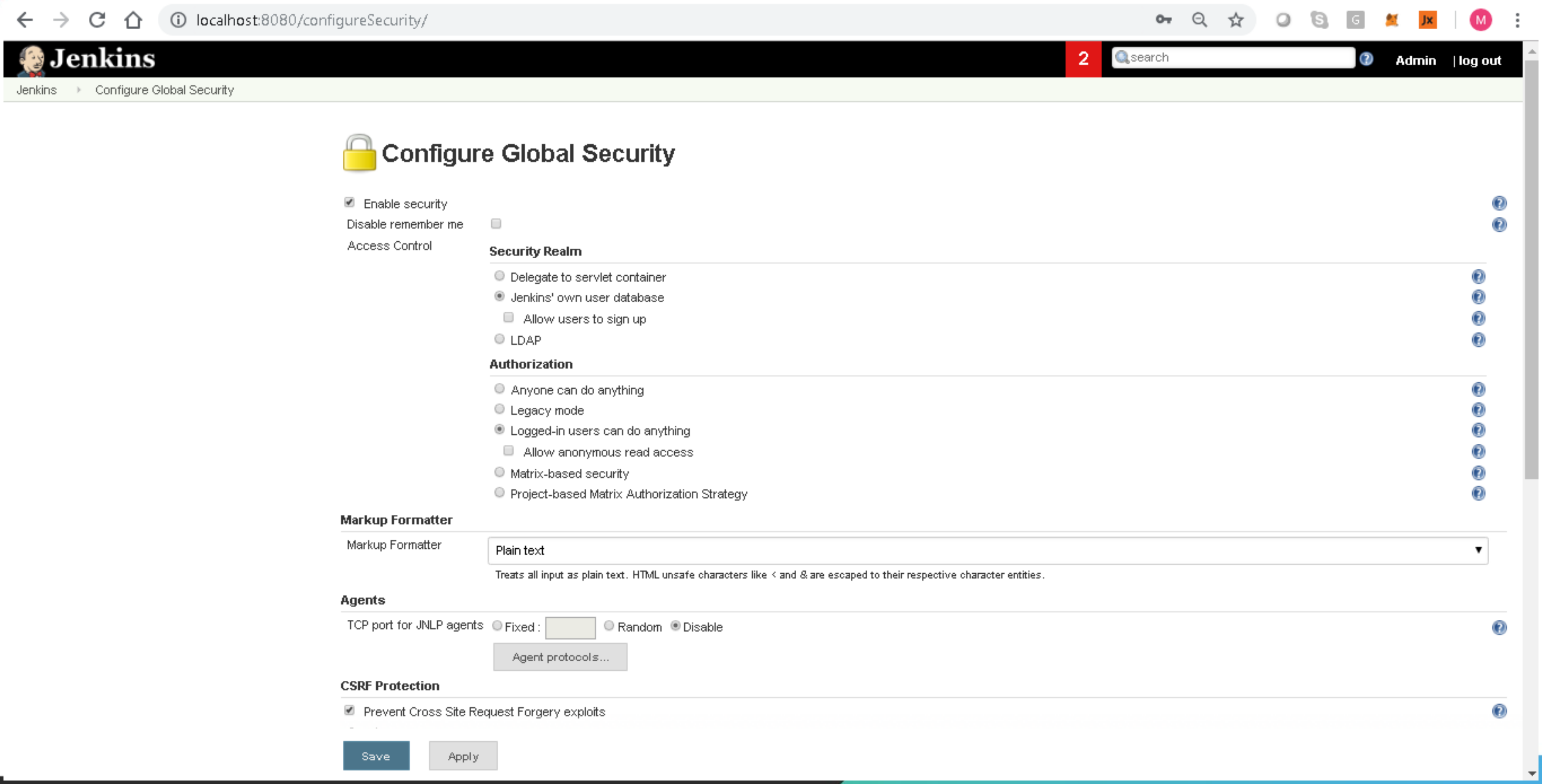
○ Project-based Matrix Authorization Strategy

**Markup Formatter**

Markup Formatter: Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

# Security



## Configure Global Security

☑ Enable security

Disable remember me □

Access Control

**Security Realm**

○ Delegate to servlet container
● Jenkins' own user database
□ Allow users to sign up
○ LDAP

**Authorization**

○ Anyone can do anything
○ Legacy mode
● Logged-in users can do anything
□ Allow anonymous read access
○ Matrix-based security
○ Project-based Matrix Authorization Strategy

**Markup Formatter**

Markup Formatter: Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

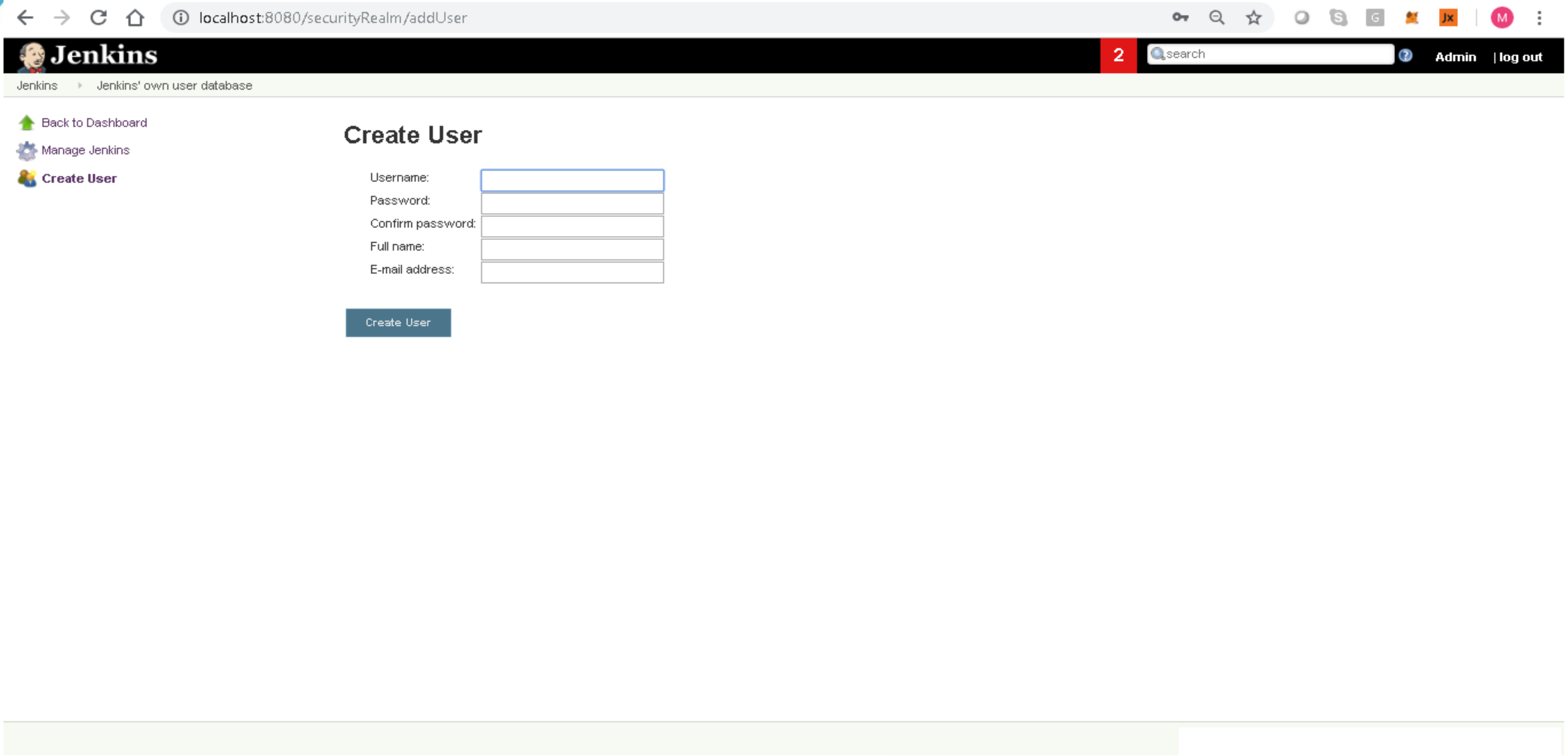
**Agents**

TCP port for JNLP agents ○ Fixed : ○ Random ● Disable

**CSRF Protection**

☑ Prevent Cross Site Request Forgery exploits

# Create a New User



## Create User

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Confirm password: |  |
| Full name: |  |
| E-mail address: |  |

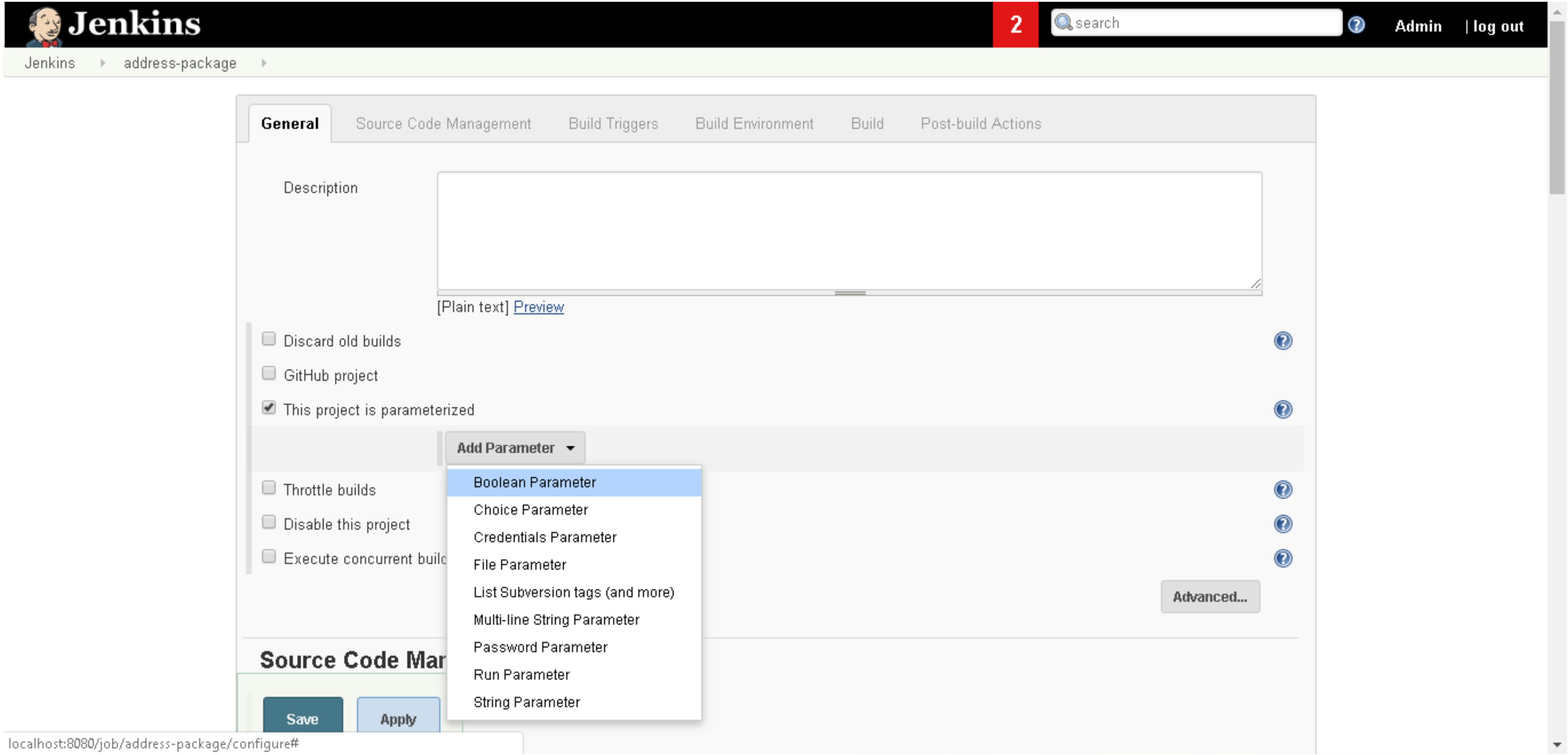
# Best Practices for Jenkins

- Always secure Jenkins
- In larger systems, don't build on the master
- Backup Jenkins Home regularly
- Limit project names to a sane (e.g. alphanumeric) character set
- Use "file fingerprinting" to manage dependencies
- The most reliable builds will be clean builds, which are built fully from Source Code Control
- Integrate tightly with your issue tracking system, like JIRA or bugzilla, to reduce the need for maintaining a Change Log
- Integrate tightly with a repository browsing tool like FishEye if you are using Subversion as source code management tool
- Always configure your job to generate trend reports and automated testing when running a Java build

# Best Practices for Jenkins

- Set up Jenkins on the partition that has the most free disk-space
- Archive unused jobs before removing them
- Setup a different job/project for each maintenance or development branch you create
- Prevent resource collisions in jobs that are running in parallel
- Avoid scheduling all jobs to start at the same time
- Set up email notifications mapping to ALL developers in the project, so that everyone on the team has his pulse on the project's current status
- Take steps to ensure failures are reported as soon as possible
- Write jobs for your maintenance tasks, such as clean up operations to avoid full disk problems
- Tag, label, or baseline the codebase after the successful build

# Jenkins Parameterized Builds



Description

[Plain text] Preview

□ Discard old builds

□ GitHub project

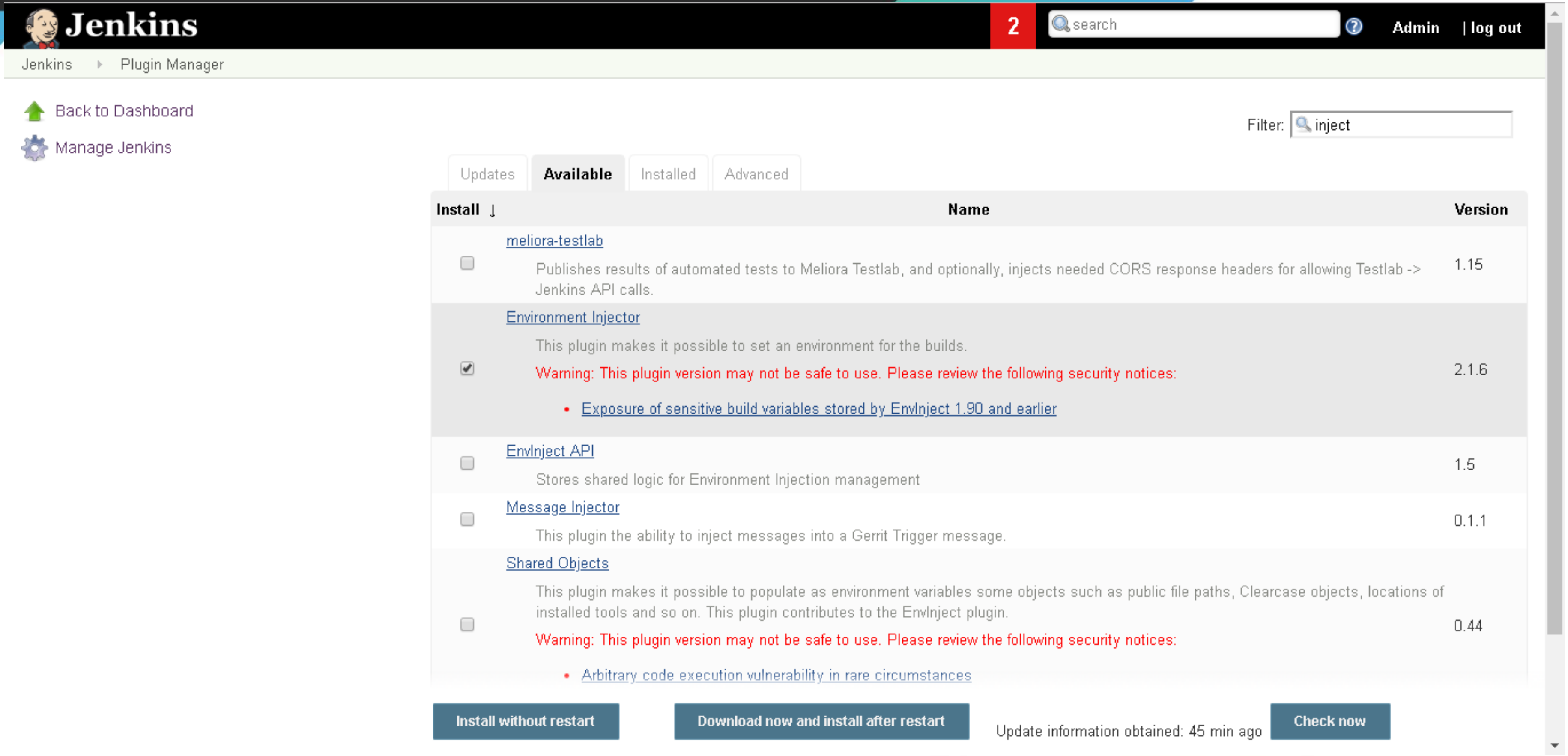
☑ This project is parameterized

□ Throttle builds

□ Disable this project

□ Execute concurrent build

# Environment Inject Plugin



Filter: inject

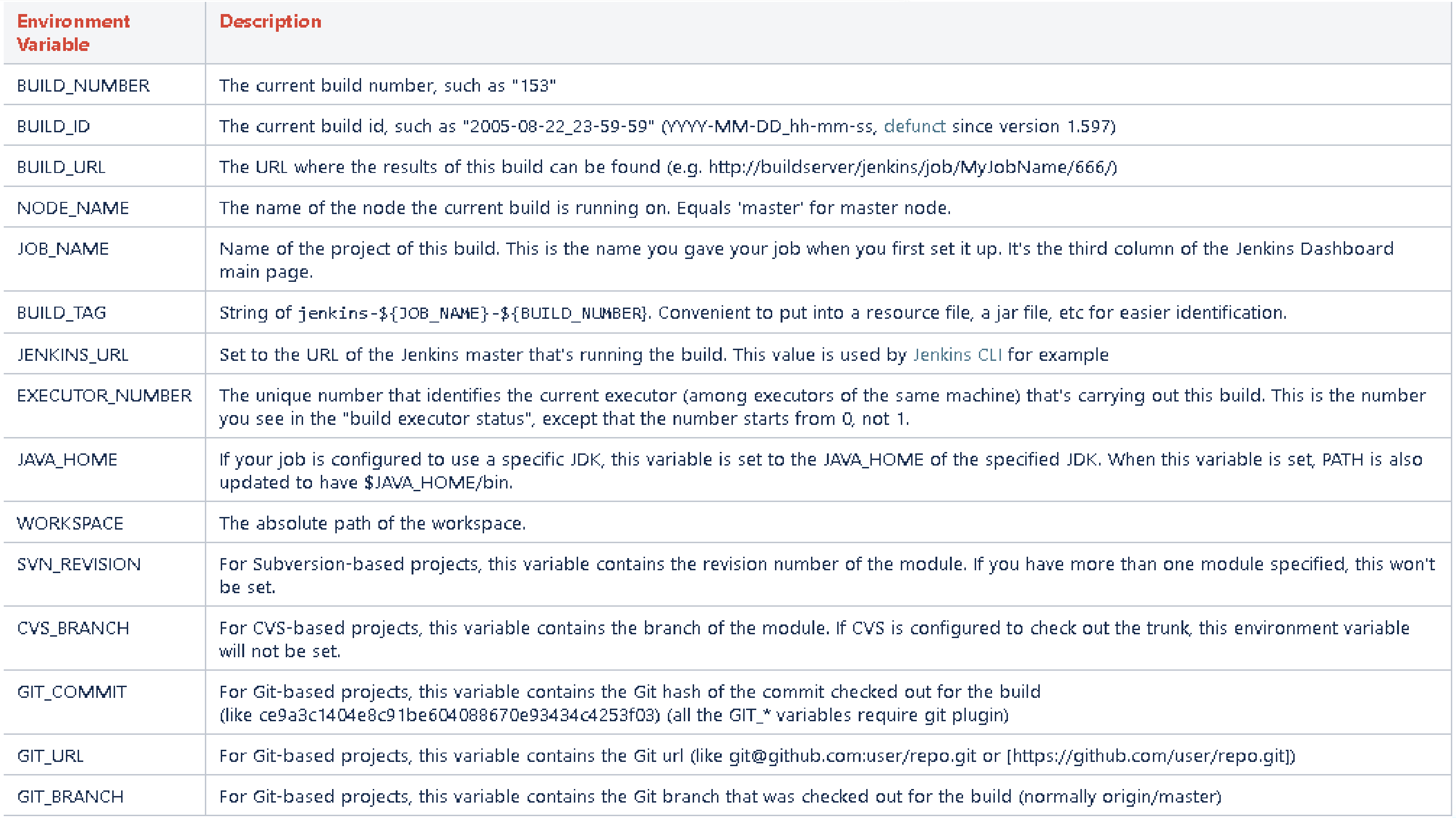
| Install ↓ | Name | Version |
|---|---|---|
| □ | meliora-testlab<br>Publishes results of automated tests to Meliora Testlab, and optionally, injects needed CORS response headers for allowing Testlab -> Jenkins API calls. | 1.15 |
| ☑ | Environment Injector<br>This plugin makes it possible to set an environment for the builds.<br>Warning: This plugin version may not be safe to use. Please review the following security notices:<br>• Exposure of sensitive build variables stored by EnvInject 1.90 and earlier | 2.1.6 |
| □ | EnvInject API<br>Stores shared logic for Environment Injection management | 1.5 |
| □ | Message Injector<br>This plugin the ability to inject messages into a Gerrit Trigger message. | 0.1.1 |
| □ | Shared Objects<br>This plugin makes it possible to populate as environment variables some objects such as public file paths, Clearcase objects, locations of installed tools and so on. This plugin contributes to the EnvInject plugin.<br>Warning: This plugin version may not be safe to use. Please review the following security notices:<br>• Arbitrary code execution vulnerability in rare circumstances | 0.44 |

Update information obtained: 45 min ago

# Use of Jenkins Environment Variables

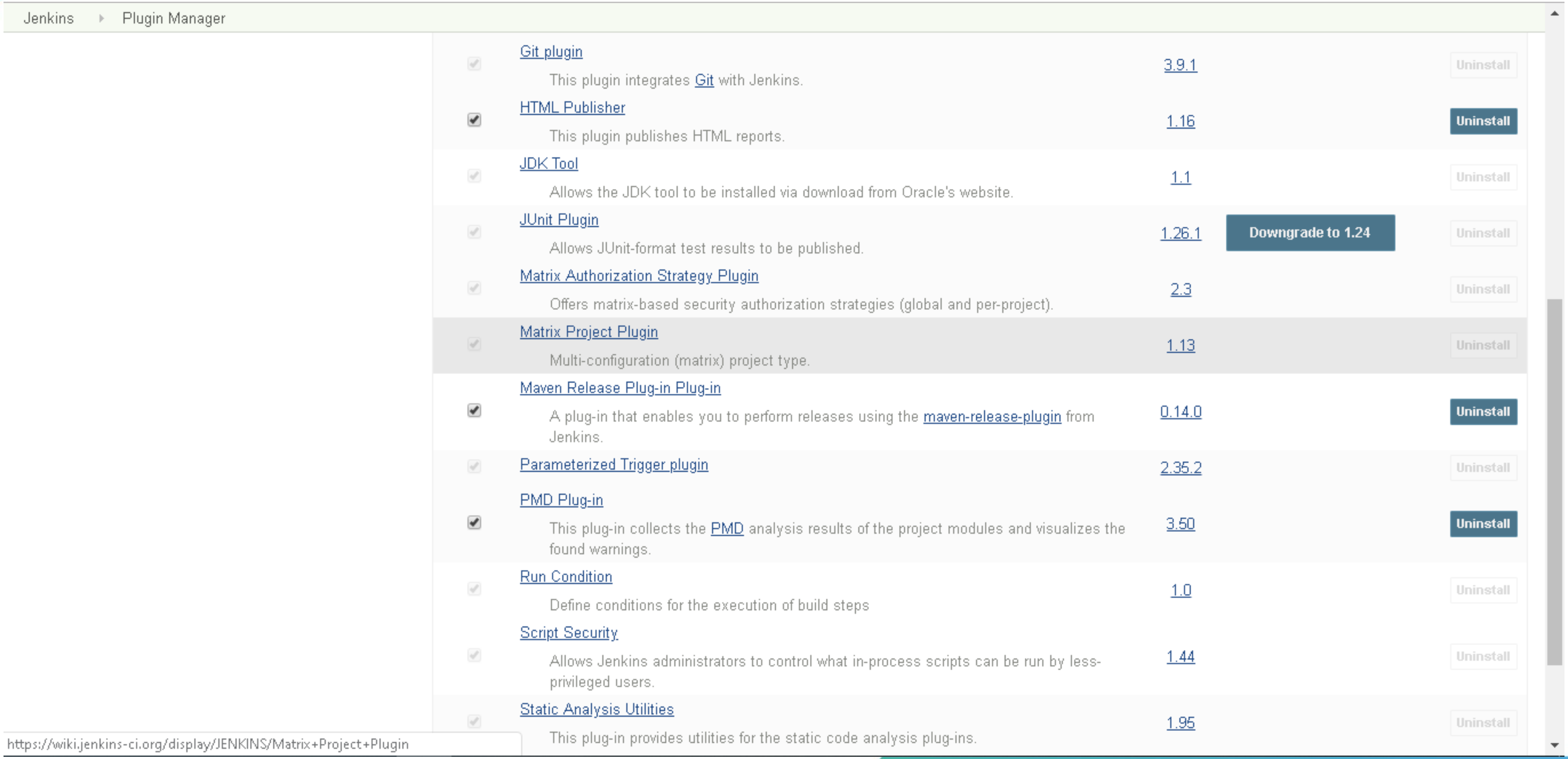
- When a Jenkins job executes, it sets some environment variables that you may use in your shell script, batch command, Ant script or Maven POM
- https://wiki.jenkins.io/display/JENKINS/Building+a+software+project#Buildingasoftwareproject-belowJenkinsSetEnvironmentVariables

# Jenkins Environment Variables



| Environment Variable | Description |
|---|---|
| BUILD_NUMBER | The current build number, such as "153" |
| BUILD_ID | The current build id, such as "2005-08-22_23-59-59" (YYYY-MM-DD_hh-mm-ss, defunct since version 1.597) |
| BUILD_URL | The URL where the results of this build can be found (e.g. http://buildserver/jenkins/job/MyJobName/666/) |
| NODE_NAME | The name of the node the current build is running on. Equals 'master' for master node. |
| JOB_NAME | Name of the project of this build. This is the name you gave your job when you first set it up. It's the third column of the Jenkins Dashboard main page. |
| BUILD_TAG | String of `jenkins-${JOB_NAME}-${BUILD_NUMBER}`. Convenient to put into a resource file, a jar file, etc for easier identification. |
| JENKINS_URL | Set to the URL of the Jenkins master that's running the build. This value is used by Jenkins CLI for example |
| EXECUTOR_NUMBER | The unique number that identifies the current executor (among executors of the same machine) that's carrying out this build. This is the number you see in the "build executor status", except that the number starts from 0, not 1. |
| JAVA_HOME | If your job is configured to use a specific JDK, this variable is set to the JAVA_HOME of the specified JDK. When this variable is set, PATH is also updated to have $JAVA_HOME/bin. |
| WORKSPACE | The absolute path of the workspace. |
| SVN_REVISION | For Subversion-based projects, this variable contains the revision number of the module. If you have more than one module specified, this won't be set. |
| CVS_BRANCH | For CVS-based projects, this variable contains the branch of the module. If CVS is configured to check out the trunk, this environment variable will not be set. |
| GIT_COMMIT | For Git-based projects, this variable contains the Git hash of the commit checked out for the build (like ce9a3c1404e8c91be604088670e93434c4253f03) (all the GIT_* variables require git plugin) |
| GIT_URL | For Git-based projects, this variable contains the Git url (like git@github.com:user/repo.git or [https://github.com/user/repo.git]) |
| GIT_BRANCH | For Git-based projects, this variable contains the Git branch that was checked out for the build (normally origin/master) |

# Project Based Matrix Plugin

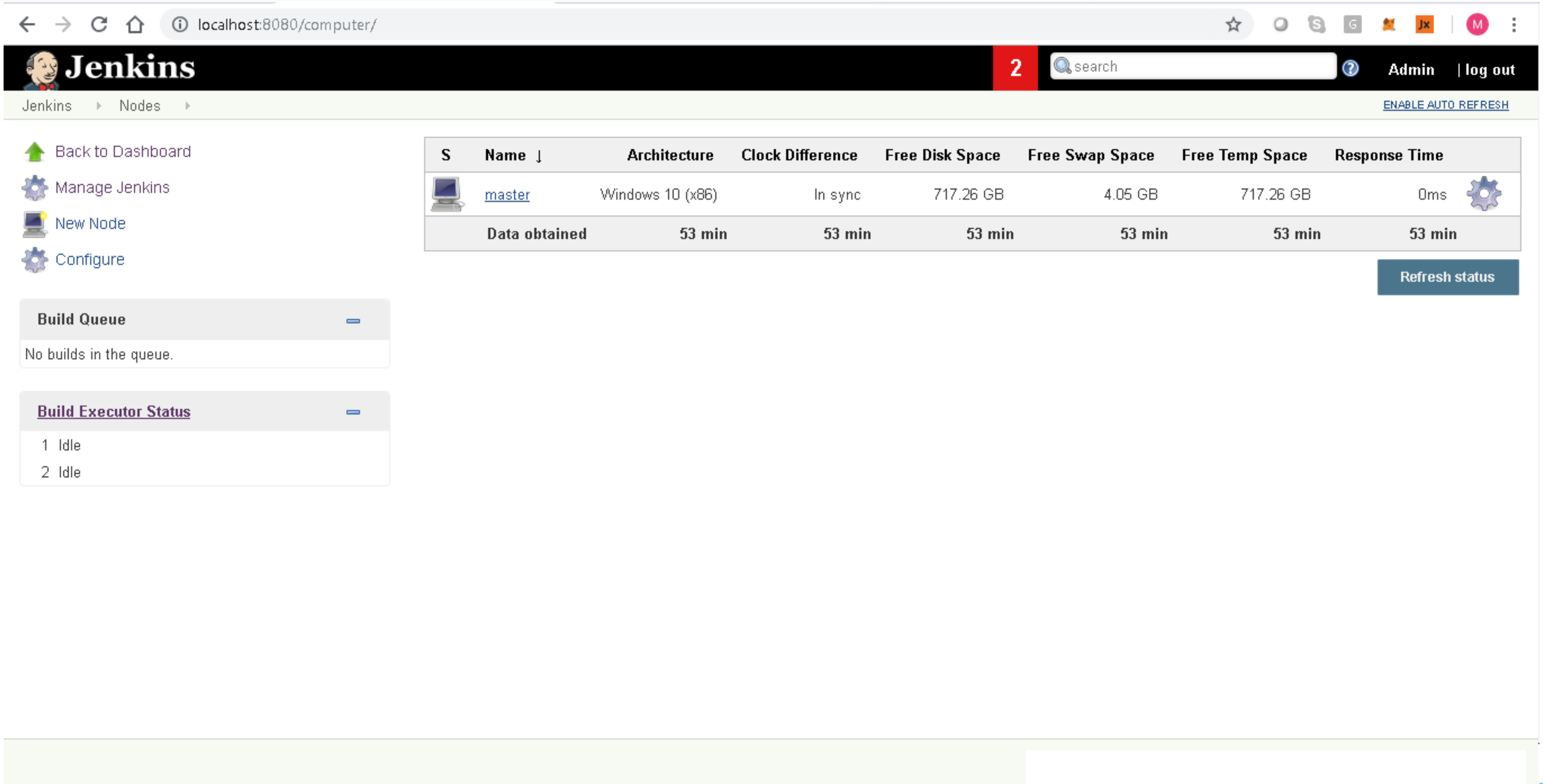


Jenkins ▸ Plugin Manager

| | | | | |
|---|---|---|---|---|
| ☑ | Git plugin<br>This plugin integrates Git with Jenkins. | 3.9.1 | | Uninstall |
| ☑ | HTML Publisher<br>This plugin publishes HTML reports. | 1.16 | | Uninstall |
| ☑ | JDK Tool<br>Allows the JDK tool to be installed via download from Oracle's website. | 1.1 | | Uninstall |
| ☑ | JUnit Plugin<br>Allows JUnit-format test results to be published. | 1.26.1 | Downgrade to 1.24 | Uninstall |
| ☑ | Matrix Authorization Strategy Plugin<br>Offers matrix-based security authorization strategies (global and per-project). | 2.3 | | Uninstall |
| ☑ | Matrix Project Plugin<br>Multi-configuration (matrix) project type. | 1.13 | | Uninstall |
| ☑ | Maven Release Plug-in Plug-in<br>A plug-in that enables you to perform releases using the maven-release-plugin from Jenkins. | 0.14.0 | | Uninstall |
| ☑ | Parameterized Trigger plugin | 2.35.2 | | Uninstall |
| ☑ | PMD Plug-in<br>This plug-in collects the PMD analysis results of the project modules and visualizes the found warnings. | 3.50 | | Uninstall |
| ☑ | Run Condition<br>Define conditions for the execution of build steps | 1.0 | | Uninstall |
| ☑ | Script Security<br>Allows Jenkins administrators to control what in-process scripts can be run by less-privileged users. | 1.44 | | Uninstall |
| ☑ | Static Analysis Utilities<br>This plug-in provides utilities for the static code analysis plug-ins. | 1.95 | | Uninstall |

https://wiki.jenkins-ci.org/display/JENKINS/Matrix+Project+Plugin

# Configuring Jenkins Hub and Node in the cloud (AWS)



| S | Name ↓ | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|---|---|---|---|---|---|---|
| | master | Windows 10 (x86) | In sync | 717.26 GB | 4.05 GB | 717.26 GB | 0ms |
| | Data obtained | 53 min | 53 min | 53 min | 53 min | 53 min | 53 min |

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

# Case Study

- Follow Lab Exercise 2 for the complete Case Study

# THANK YOU