



Jenkins 101



Cheryl Fong

[View on Google slides](#)

Objective

Topics covered:

1. Installation ~ slide 3
2. Setup ~ slide 15
3. Connecting Jenkins to GitHub ~ slide 27
4. SSH via Jenkins e.g. SSH to AWS EC2 Instance ~ slide 50
5. Putting Everything Together - Deployment Example ~ slide 60

Installation Prerequisites

Minimum hardware requirements:

- 256 MB of RAM
- 1 GB of drive space (10 GB is a recommended minimum if running Jenkins as a Docker container)

Recommended hardware configuration for a *small team*:

- 1 GB+ of RAM
- 50 GB+ of drive space

Software requirements:

- Java: see the [Java Requirements](#) page
- Web browser: see the [Web Browser Compatibility](#) page

[Jenkins Official Prerequisites Guide](#)

Installation

Installation and setup on your machine:

- See the official Jenkins guides for:
 - i. [MacOS](#)
 - ii. [Linux](#) distros - Fedora and Debian/Ubuntu
 - iii. [Windows](#)

Note: It is recommended to install Jenkins on a provisioned server, not on a personal computer. So that you could run Jenkins 24/7 and would be able to access the Jenkins dashboard via a public IP address.

Installation and Setup via Docker are covered in the following slides.

Installation via Docker



This section assumes you have Docker up and running.

Note: your system must meet the “Minimum hardware requirements” but you may ignore the “Software requirements” on slide 3.

Otherwise Install Docker CE Edition on [MacOS](#), [Linux](#) or [Windows](#) before proceeding

If you've installed Jenkins natively (not through Docker), skip to slide: 15

Installation via Docker

Run the following command on your terminal to pull [the latest official jenkins container image](#):

```
docker pull jenkins/jenkins
```

Installation via Docker

```
[cherylfong@localhost ~]$ docker pull jenkins/jenkins
Using default tag: latest
latest: Pulling from jenkins/jenkins
c5e155d5a1d1: Pull complete
221d80d00ae9: Pull complete
4250b3117dca: Pull complete
3b7ca19181b2: Pull complete
350c4ab1d0b1: Pull complete
1cb16e1cfeec: Pull complete
9cf1a68a908d: Pull complete
113eac674a17: Pull complete
478cec5640f8: Pull complete
75e0025f6c58: Pull complete
11875edd3d91: Pull complete
d0284aa64861: Pull complete
4f459628b10b: Pull complete
bf372d0b3edb: Pull complete
8678c47b29ae: Pull complete
387a08834fce: Pull complete
9c333eb740f8: Pull complete
6a9c7cd4c144: Pull complete
Digest: sha256:70a356eeb3ff30307376aa077aec4fdec7340b0f347eb54e414d9bcae15ea92
Status: Downloaded newer image for jenkins/jenkins:latest
```

Installation via Docker

This is the command required to get Jenkins running, see the next slide before running this command:

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 jenkins/jenkins
```


Setup via Docker

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 jenkins/jenkins
```

The command explained:

1. Runs the Jenkins image container in detached mode *-d*
2. Connects the container to a volume named *jenkins_home* and attach it to the container's */var/jenkins_home* directory. (You can find the *jenkins_home* volume by running *Docker volume ls*)
3. *-p* specifies connecting the host 8080 port to the container's 8080 port
4. Read the [Build Executors](#) section on the usage of *-p 50000:50000*. It is likely that you'll not need this. Hence the following is sufficient:

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 jenkins/jenkins
```

Setup via Docker (Advanced)

Use case:

You have a website running on some IP address on port 8080 e.g. 13.52.51.45:8080 but you want to bind Jenkins to the same IP address.

1. Find an unoccupied port on the host/server e.g. 8082 (or 80 if nothing is bounded to it).
2. Bind the Jenkins container to the host 8082 port to its 8080 port, with the following command:

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8082:8080 jenkins/jenkins
```

This allows you to visit the Jenkins Dashboard at `http://13.52.51.45:8082`

Installation via Docker (Advanced++)

Use case:

You have a website running on some IP address on port 8080 e.g. 13.52.51.45:8080 that you've launched via **Docker Compose** but you want to bind Jenkins to the same IP address.

1. Find an unoccupied port on the host/server e.g. 8082 or (80 if nothing is bounded to it).
2. Find the network namespace created by Docker Compose.
 - a. Run: `docker network ls`
3. Bind the Jenkins container to the host 8082 port to its 8080 port, with the command on the next slide.

Setup via Docker (Advanced++)

Use case:

You have a website running on some IP address on port 8080 e.g. 13.52.51.45:8080 that you've launched via **Docker Compose** but you want to bind Jenkins to the same IP address.

```
ubuntu@ip-172-31-13-31:~$ docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
63c9b7cd942f	bridge	bridge	local
101c1c776bce	csc648-sp19-team08_default	bridge	local
85568599def2	host	host	local
a70e2a4d8f15	none	null	local

```
docker run -d -v jenkins_home:/var/jenkins_home --net csc648-sp19-team08_default -p 8082:8080 jenkins/jenkins
```

This allows you to visit the Jenkins Dashboard at `http://13.52.51.45:8082`

Setup via Docker (Advanced++)

Use case:

You have a website running on some IP address on port 8080 e.g. 13.52.51.45:8080 that you've launched via **Docker Compose** but you want to bind Jenkins to the same IP address.

```
docker run -d -v jenkins_home:/var/jenkins_home --net csc648-sp19-team08_default -p 8082:8080 jenkins/jenkins
```

```
[cherylfong@localhost csc648-sp19-team08]$ docker run --name jenkins -d -v jenkins_home:/var/jenkins_home --net csc648-sp19-team08_default -p 8082:8080 jenkins/jenkins  
debbd0a97f6aa10669f4f8a22f1de858bcc20c3f52406bf75023561aba2fbc1  
[cherylfong@localhost csc648-sp19-team08]$
```

The expected output above shows a hash value, i.e. hash 'debb . . . fbc1'. It varies from execution of the run command in detached mode. Your hash value will not be the same as the screenshot.

Setup via Docker (Advanced++)

Use case:

You have a website running on some IP address on port 8080 e.g. 13.52.51.45:8080 that you've launched via **Docker Compose** but you want to bind Jenkins to the same IP address.

You can verify that your jenkins container is running by executing, *docker ps -a* :

```
[cherylfong@localhost csc648-sp19-team08]$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
debbed0a97f6	jenkins/jenkins	"/sbin/tini -- /usr/..."	11 minutes ago	Up 11 minutes	50000/tcp, 0.0.0.0:8082->8080/tcp	jenkins

The *--name* option lets you provide a custom container name for the running Jenkins image container.

```
[cherylfong@localhost csc648-sp19-team08]$ docker run --name jenkins -d -v jenkins_home:/var/jenkins_home --net csc648-sp19-team08_default -p 8082:8080 jenkins/jenkins
debbed0a97f6aa10669f4f8a22f1de858bcc20c3f52406bf75023561aba2fbc1
[cherylfong@localhost csc648-sp19-team08]$
```

In this case, I've named the Jenkins image container as 'jenkins'.

Setup

This “Setup” section will guide you through setting up Jenkins via the recommended/default configuration.

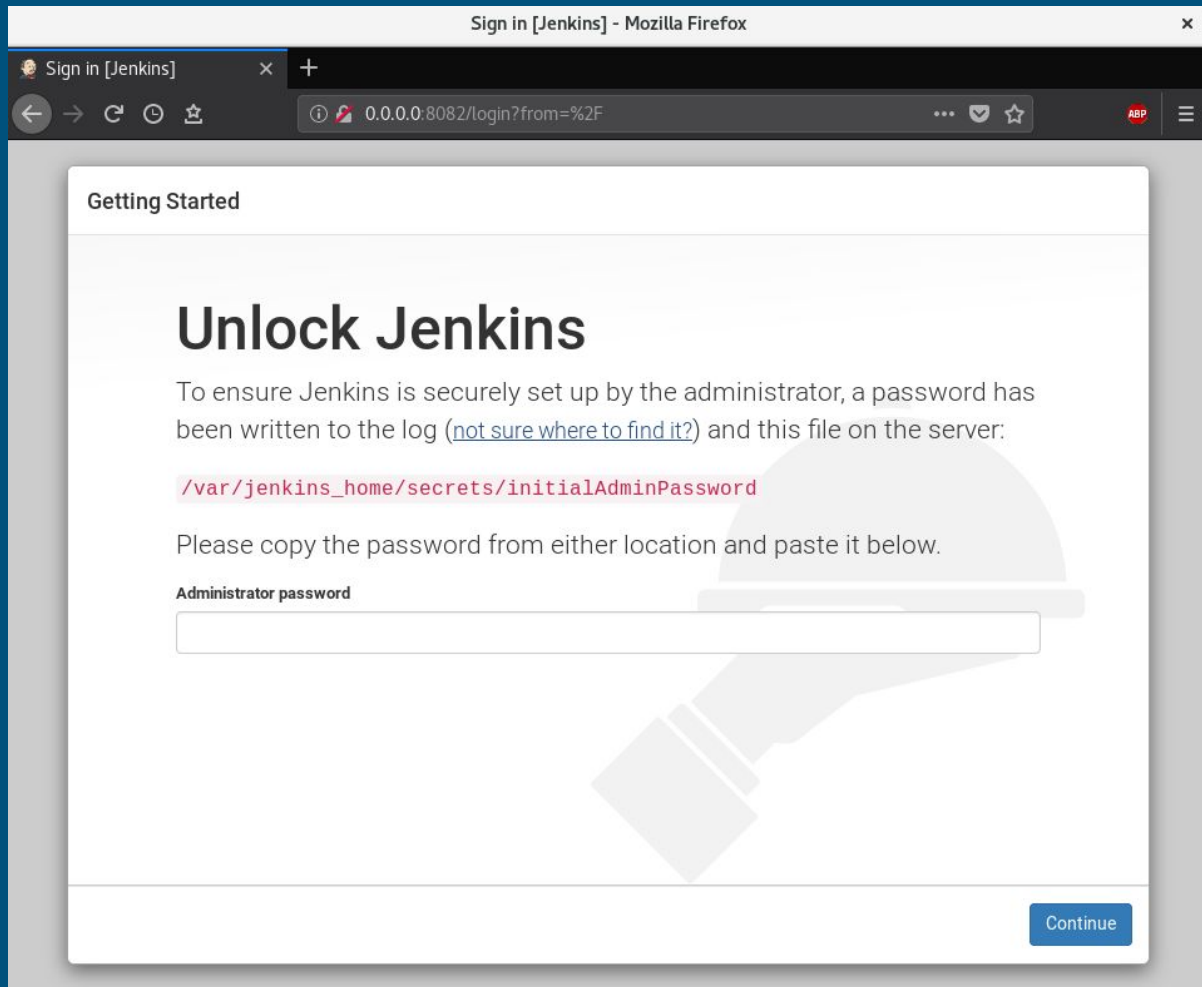


At this point you should have Jenkins installed on your machine natively or via Docker.

Access the Jenkins Dashboard by the IP address of the machine you’ve installed it on:

IP Examples	Docker or Natively Installed	Advanced Installation via Docker (on slide 10)
On a server with an exposed public accessible IP	http://13.52.51.45:80 or http://13.52.51.45	http://13.52.51.45:8082
On your personal computer (not recommended see slide 4)	http://0.0.0.0:80 or http://0.0.0.0 http://127.0.0.1:80 or http://127.0.0.1	http://0.0.0.0:8082/ or http://127.0.0.1:8082/

Setup



Setup

As instructed by Jenkins, copy and paste the initial administrator password into the field.

To obtain the password, SSH into your server and locate the `/var/jenkins_home/secrets/initialAdminPassword` file.

See the slide 18 to obtain the password from the Jenkins Image container.

Setup

My Jenkins Image container that is running is named 'jenkins'.

```
[cherylfong@localhost csc648-sp19-team08]$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
debbed0a97f6	jenkins/jenkins	"/sbin/tini -- /usr/..."	11 minutes ago	Up 11 minutes	50000/tcp, 0.0.0.0:8082->8080/tcp	jenkins

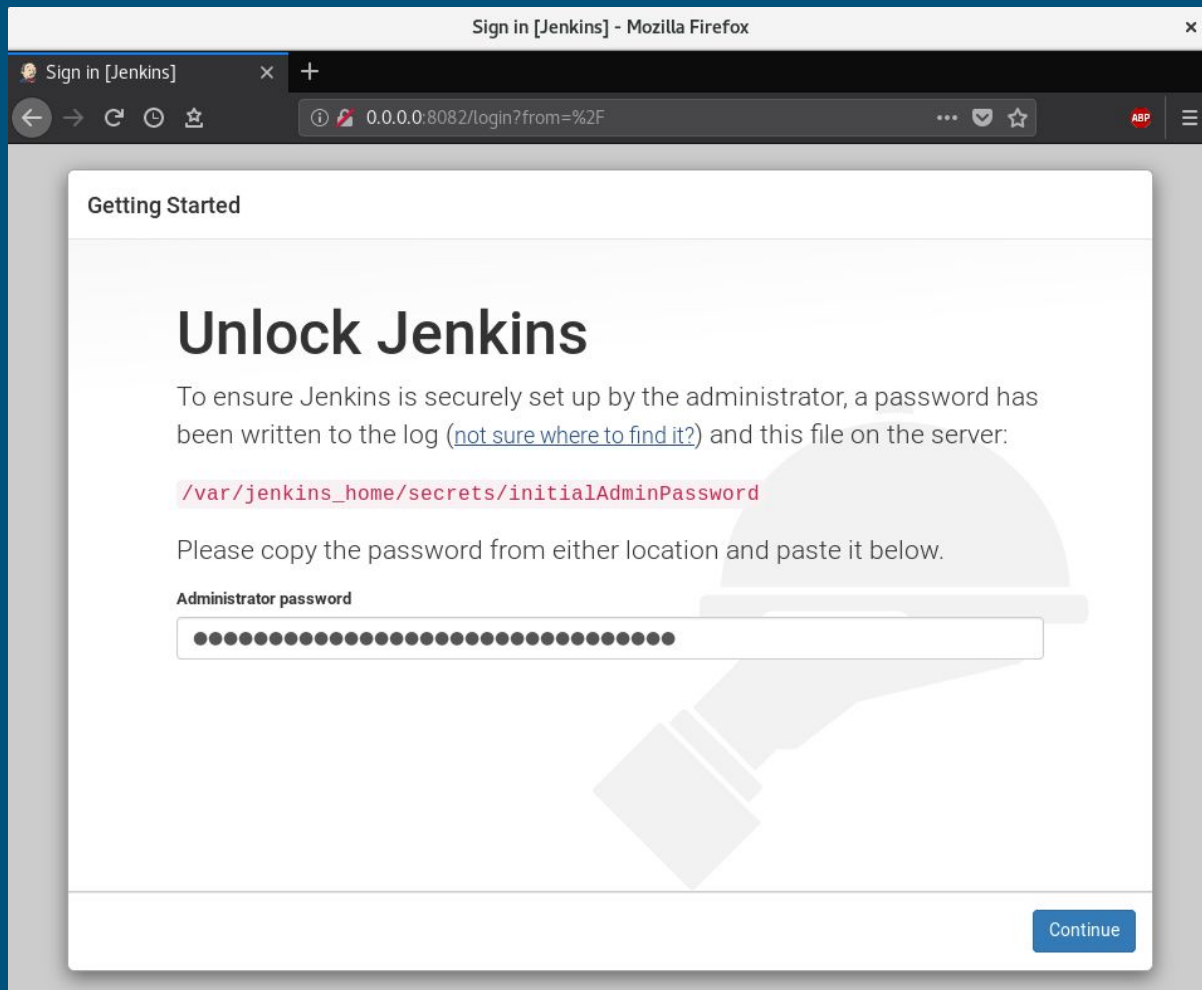
The following command will execute an interactive bash terminal into my 'jenkins' container:

```
[cherylfong@localhost csc648-sp19-team08]$ docker exec -it jenkins bash
jenkins@debbed0a97f6:/$
```

Copy and paste the output from the command into the password field on the browser:

```
jenkins@debbed0a97f6:/$ cat /var/jenkins_home/secrets/initialAdminPassword
570e6d85619241dca2c784139053ab26
```

Setup



Setup

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.

Setup

After selecting
“Install
suggested
plugins”:

Getting Started

Getting Started

<input checked="" type="checkbox"/> Folders	<input type="checkbox"/> OWASP Markup Formatter	<input type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding
<input type="checkbox"/> Timestampers	<input type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input type="checkbox"/> Gradle
<input type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Pipeline: Stage View
<input type="checkbox"/> Git	<input type="checkbox"/> Subversion	<input type="checkbox"/> SSH Slaves	<input type="radio"/> Matrix Authorization Strategy
<input type="radio"/> PAM Authentication	<input type="checkbox"/> LDAP	<input type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer

Folders

** JDK Tool

** Script Security

** Command Agent Launcher

** Struts

** Pipeline: Step API

** SCM API

** Pipeline: API

** JUnit

** bouncycastle API

** - required dependency

Jenkins 2.179

Setup

After “Install suggested plugins” have completed, you will be brought to this page automatically :

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.179

Continue as admin

Save and Continue

Setup

Create an admin username and password of your choice.

Jenkins will use the provided email address to send notifications (if configured in settings - can be changed later)

Getting Started

Create First Admin User

Username:

admin

Password:

•••••

Confirm password:

•••••

Full name:

admin

E-mail address:

admin@mail.com

Jenkins 2.179

Continue as admin

Save and Continue

Setup

This is running on a personal computer (it is not recommended see slide 4)

The “Jenkins URL” field should be automatically populated with the IP of where Jenkins is running.

Getting Started

Instance Configuration

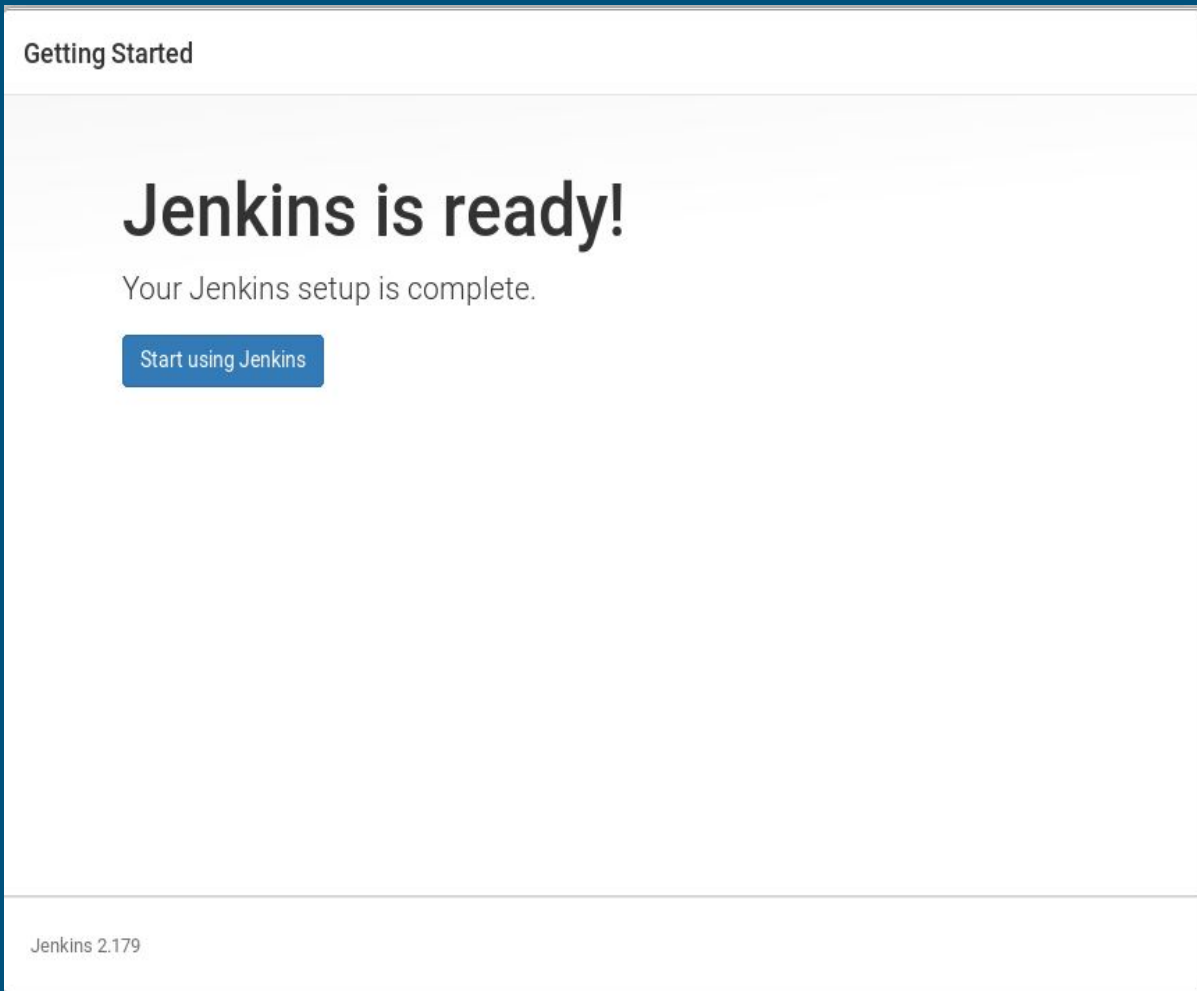
Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

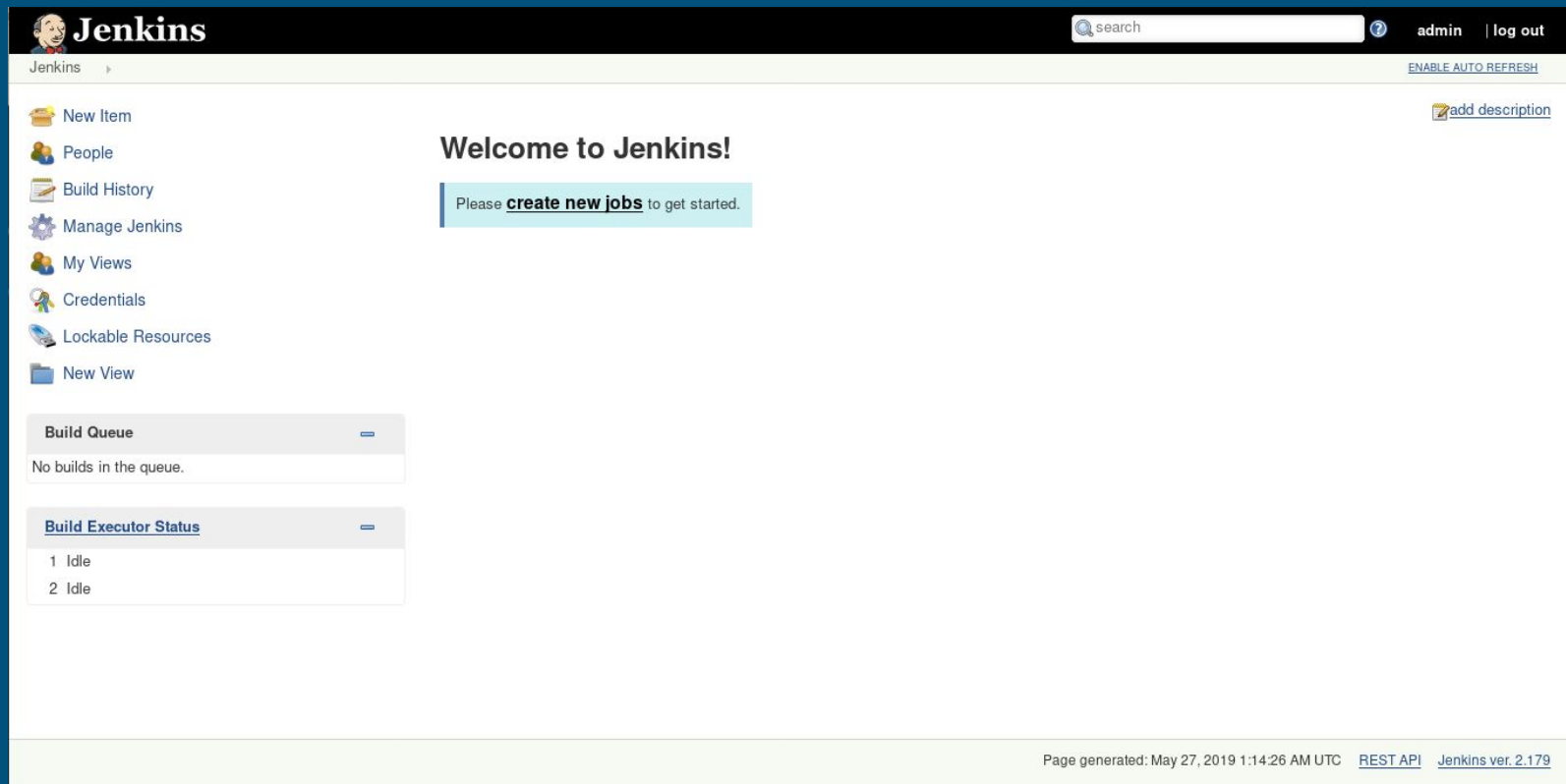
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Setup

After clicking “Save and Finish”, click “Start using Jenkins”.



Setup Complete



The screenshot shows the Jenkins web interface after a successful setup. The top navigation bar is black with the Jenkins logo, a search bar, and links for 'admin' and 'log out'. Below this, a light green sidebar on the left contains a list of navigation items: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', 'Lockable Resources', and 'New View'. The main content area has a white background with a 'Welcome to Jenkins!' heading. Below the heading is a light blue box with the text 'Please **create new jobs** to get started.' To the right of the welcome message is a link to 'add description'. At the bottom of the main area, there are two expandable panels. The first, 'Build Queue', shows 'No builds in the queue.' The second, 'Build Executor Status', shows two executors, both in an 'Idle' state. The footer of the page is light green and contains the text 'Page generated: May 27, 2019 1:14:26 AM UTC', a link to the 'REST API', and the version 'Jenkins ver. 2.179'.

Jenkins

search

admin | log out

Jenkins

ENABLE AUTO REFRESH

[add description](#)

Welcome to Jenkins!

Please **create new jobs** to get started.

Build Queue

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

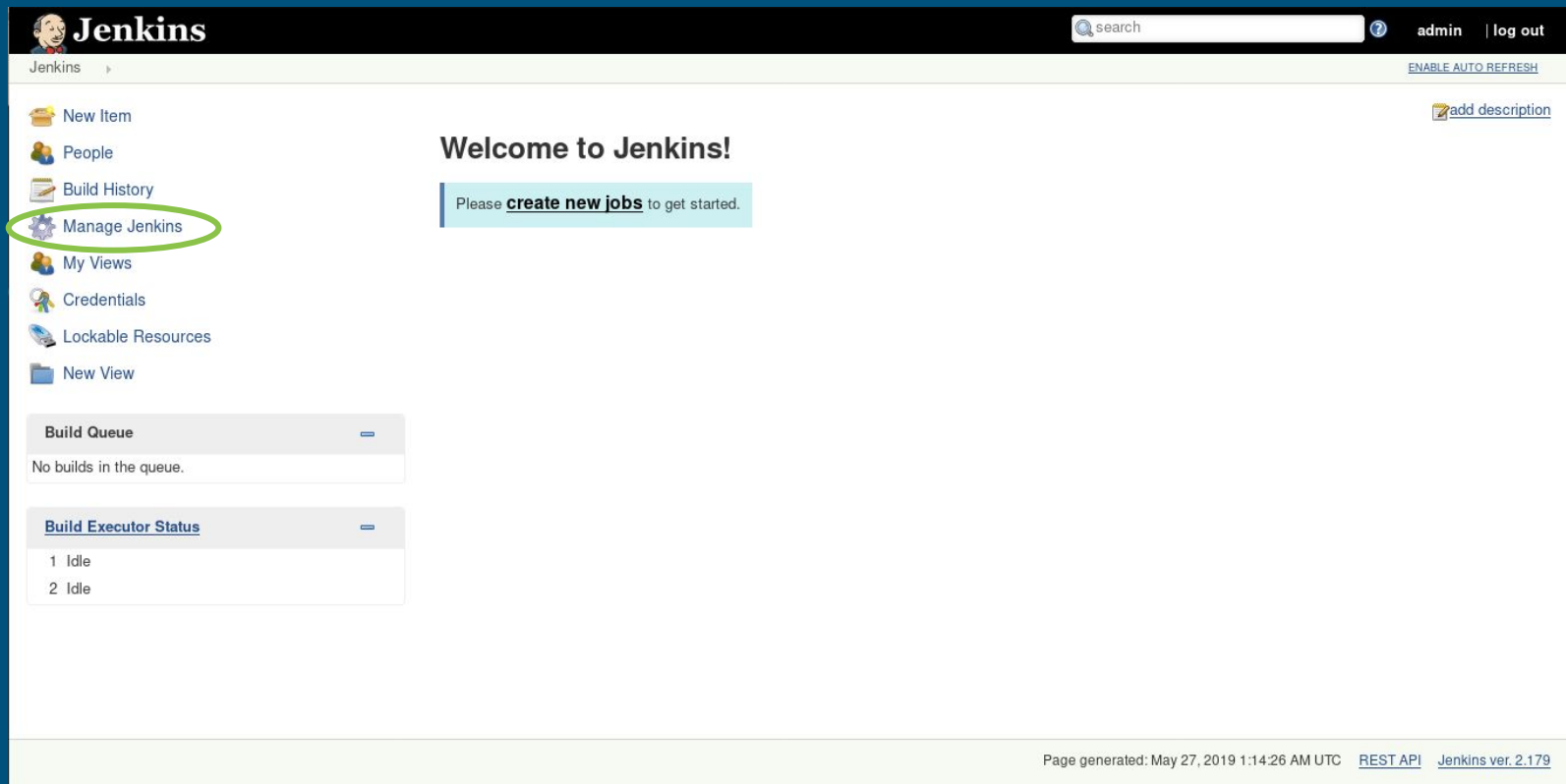
Page generated: May 27, 2019 1:14:26 AM UTC [REST API](#) [Jenkins ver. 2.179](#)

Connect Jenkins to GitHub

This “Connect Jenkins to GitHub” section will guide you through setting up Jenkins to pull code from a GitHub repository via the the Jenkins GitHub Plugin.

Click “Manage Jenkins” on the left side panel of the dashboard to get started.

Connect Jenkins to GitHub



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and links for 'admin' and 'log out'. The left sidebar contains a list of navigation items: 'New Item', 'People', 'Build History', 'Manage Jenkins' (circled in green), 'My Views', 'Credentials', 'Lockable Resources', and 'New View'. The main content area displays a 'Welcome to Jenkins!' message with a prompt to 'create new jobs'. Below this, there are two sections: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing two idle executors.

Jenkins

search

admin | log out

Jenkins

ENABLE AUTO REFRESH

add description

New Item

People

Build History

Manage Jenkins

My Views

Credentials

Lockable Resources

New View

Build Queue

No builds in the queue.

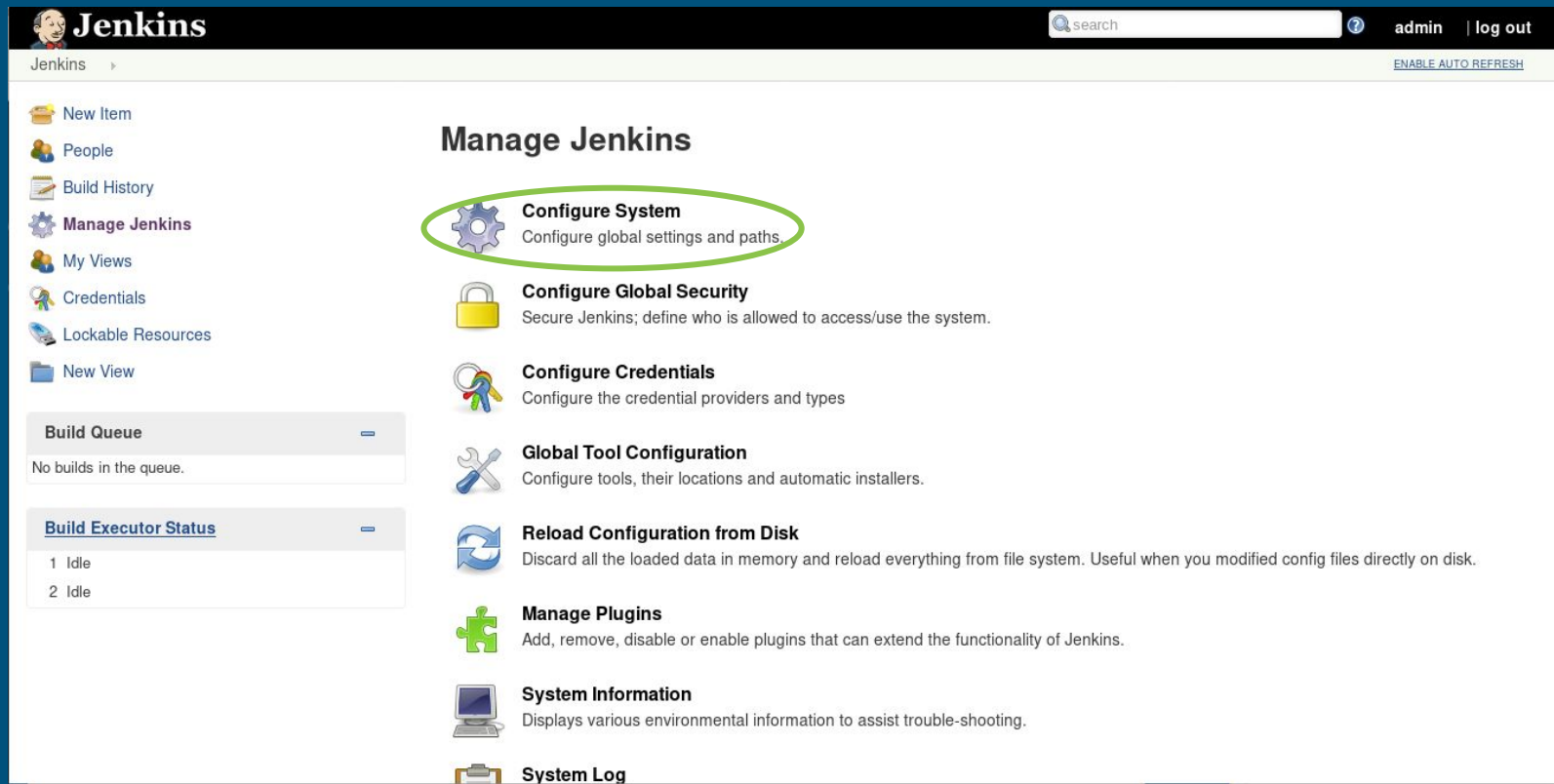
Build Executor Status

1 Idle

2 Idle

Page generated: May 27, 2019 1:14:26 AM UTC [REST API](#) [Jenkins ver. 2.179](#)

Connect Jenkins to GitHub



The screenshot displays the Jenkins web interface. At the top, the Jenkins logo and name are on the left, and a search bar, user name 'admin', and 'log out' link are on the right. Below the header, a left sidebar contains navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (highlighted in purple), 'My Views', 'Credentials', 'Lockable Resources', and 'New View'. The main content area is titled 'Manage Jenkins' and lists several configuration options, each with an icon and a description:

- Configure System** (gear icon): Configure global settings and paths. This option is circled in green.
- Configure Global Security** (lock icon): Secure Jenkins; define who is allowed to access/use the system.
- Configure Credentials** (key icon): Configure the credential providers and types.
- Global Tool Configuration** (wrench icon): Configure tools, their locations and automatic installers.
- Reload Configuration from Disk** (refresh icon): Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins** (puzzle piece icon): Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- System Information** (monitor icon): Displays various environmental information to assist trouble-shooting.
- System Log** (notepad icon):

On the left side of the main content area, there are two panels:

- Build Queue**: Shows 'No builds in the queue.'
- Build Executor Status**: Shows a list of executors: '1 Idle' and '2 Idle'.

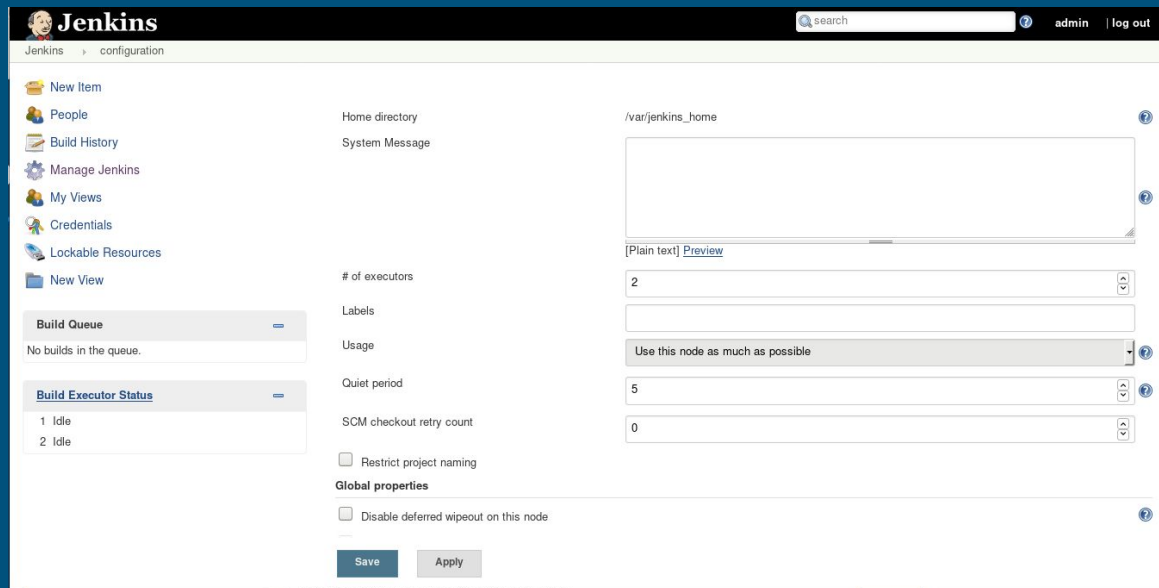
Connect Jenkins to GitHub

The screenshot shows the Jenkins web interface. At the top, there's a header with the Jenkins logo, a search bar, and links for 'admin' and 'log out'. Below the header, the breadcrumb 'Jenkins > configuration' is visible. On the left sidebar, there are links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', 'Lockable Resources', and 'New View'. The main content area is titled 'configuration' and contains several settings:

- Home directory:** Set to `/var/jenkins_home`.
- System Message:** A large text area for a system message, currently empty.
- # of executors:** Set to `2`.
- Labels:** An empty text field.
- Usage:** Set to 'Use this node as much as possible'.
- Quiet period:** Set to `5`.
- SCM checkout retry count:** Set to `0`.
- Restrict project naming:** An unchecked checkbox.
- Global properties:** A section with an unchecked checkbox for 'Disable deferred wipeout on this node'.

At the bottom, there are 'Save' and 'Apply' buttons. On the left, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle').

Connect Jenkins to GitHub



The screenshot shows the Jenkins configuration page. The left sidebar contains navigation links: New Item, People, Build History, Manage Jenkins, My Views, Credentials, Lockable Resources, and New View. The main content area is titled 'configuration' and contains several settings. The 'Global properties' section is highlighted, showing a checkbox for 'Disable deferred wipeout on this node'. The 'Build Queue' section shows 'No builds in the queue'. The 'Build Executor Status' section shows two executors: '1 Idle' and '2 Idle'. The 'Home directory' is set to '/var/jenkins_home'. The 'System Message' field is empty. The '# of executors' is set to '2'. The 'Labels' field is empty. The 'Usage' dropdown is set to 'Use this node as much as possible'. The 'Quiet period' is set to '5'. The 'SCM checkout retry count' is set to '0'. The 'Restrict project naming' checkbox is unchecked. The 'Save' and 'Apply' buttons are at the bottom.

Jenkins

configuration

New Item

People

Build History

Manage Jenkins

My Views

Credentials

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Home directory

System Message

/var/jenkins_home

of executors

2

Labels

Usage

Use this node as much as possible

Quiet period

5

SCM checkout retry count

0

☐ Restrict project naming

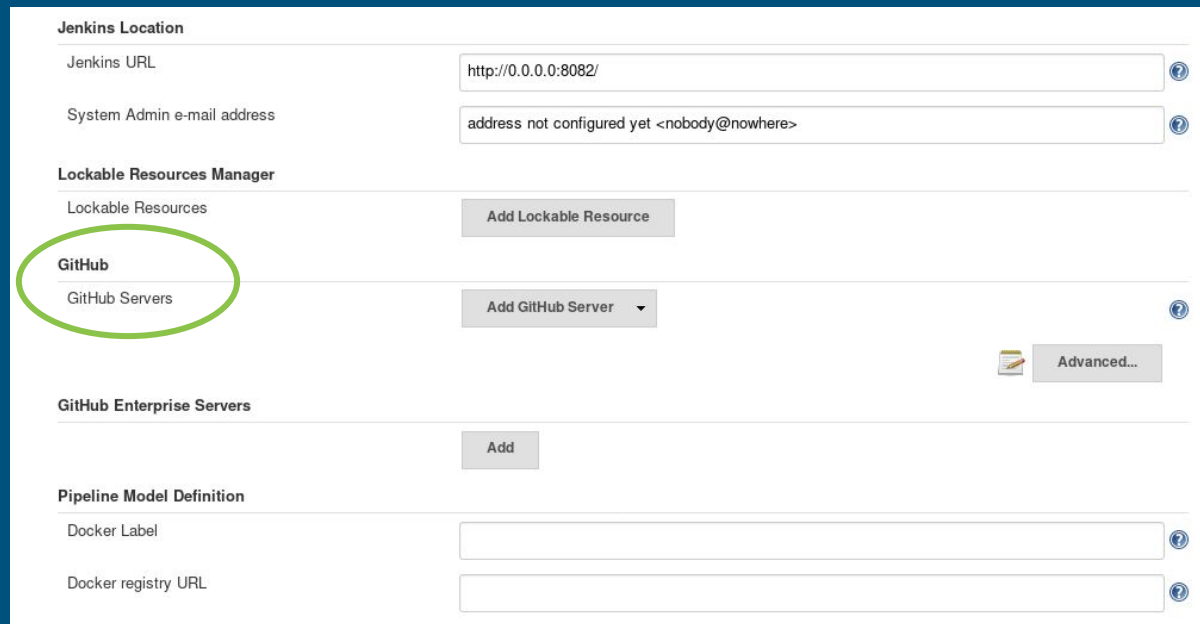
Global properties

☐ Disable deferred wipeout on this node

Save Apply

Scroll down to locate the GitHub section.

Connect Jenkins to GitHub



The screenshot displays the Jenkins configuration interface. The 'Jenkins Location' section includes fields for 'Jenkins URL' (http://0.0.0.0:8082/) and 'System Admin e-mail address' (address not configured yet <nobody@nowhere>). The 'Lockable Resources Manager' section has a table with 'Lockable Resources' and a button 'Add Lockable Resource'. Below this, the 'GitHub' section is highlighted with a green circle; it contains a 'GitHub Servers' field and a button 'Add GitHub Server'. To the right of this section is an 'Advanced...' button. The 'GitHub Enterprise Servers' section has an 'Add' button. The 'Pipeline Model Definition' section includes fields for 'Docker Label' and 'Docker registry URL'. Help icons are present next to several fields.

Jenkins Location	
Jenkins URL	http://0.0.0.0:8082/
System Admin e-mail address	address not configured yet <nobody@nowhere>

Lockable Resources Manager	
Lockable Resources	Add Lockable Resource
GitHub	
GitHub Servers	Add GitHub Server

Advanced...

GitHub Enterprise Servers	
	Add

Pipeline Model Definition	
Docker Label	
Docker registry URL	

This GitHub section may appear in different parts of “Configure System” depending on the Jenkins version. Version shown on slidedeck 2.179

Connect Jenkins to GitHub

If you cannot find the GitHub section on “Configure system”. Select “Manage Plugins” from “Manage Jenkins” (found on the left side panel of the dashboard).



Manage Plugins

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

Select “Installed” and scroll down to look for the GitHub Plugin.

The screenshot shows the Jenkins Plugin Manager interface. The top navigation bar includes the Jenkins logo, a search bar, and user information (admin | log out). The left sidebar contains links: Back to Dashboard, Manage Jenkins, and Update Center. The main content area displays the Plugin Manager with tabs for Updates, Available, Installed, and Advanced. The 'Installed' tab is selected and circled in green. Below the tabs is a table of installed plugins.

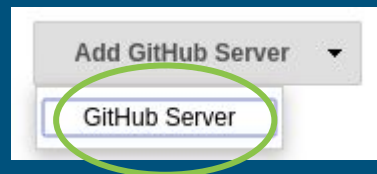
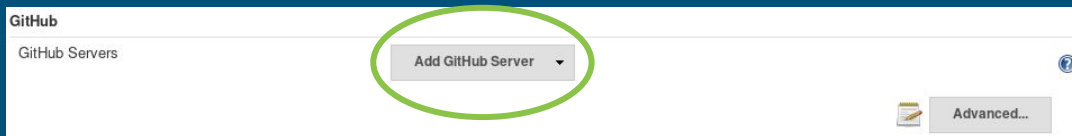
Enabled	Name ↓	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	Ant Plugin Adds Apache Ant support to Jenkins	1.9		Uninstall
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins.	4.5.5-3.0		Uninstall

Connect Jenkins to GitHub

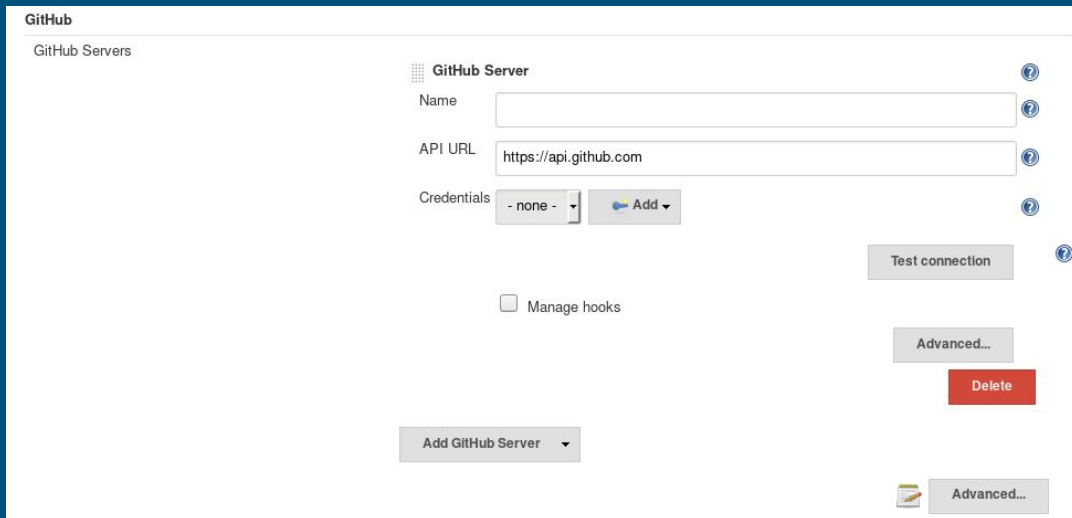
If you cannot find the GitHub section on “Configure system”. Make sure you have the following plugins installed. Especially the “GitHub plugin”.

<input checked="" type="checkbox"/>	Git client plugin Utility plugin for Git support in Jenkins	2.7.7	Uninstall
<input checked="" type="checkbox"/>	Git plugin This plugin integrates Git with Jenkins.	3.10.0	Uninstall
<input checked="" type="checkbox"/>	GIT server Plugin Allows Jenkins to act as a Git server.	1.7	Uninstall
<input checked="" type="checkbox"/>	GitHub API Plugin This plugin provides GitHub API for other plugins.	1.95	Uninstall
<input checked="" type="checkbox"/>	GitHub Branch Source Plugin Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	2.5.3	Uninstall
<input checked="" type="checkbox"/>	GitHub plugin This plugin integrates GitHub to Jenkins.	1.29.4	Uninstall

Connect Jenkins to GitHub



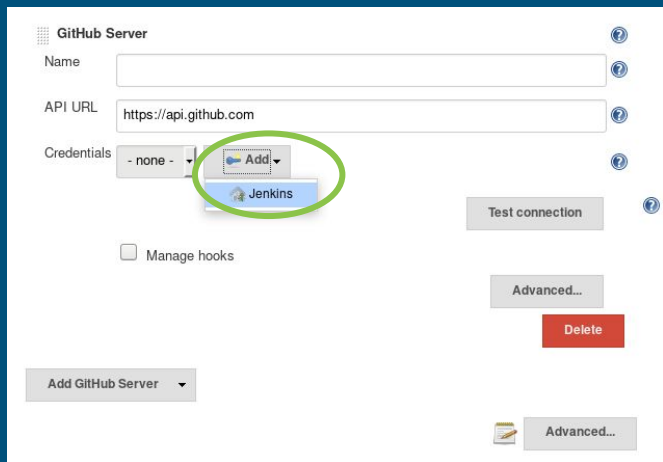
Select "Add GitHub Server"



Leave the "API URL" as is.

Provide any name of your choice to the "Name" field.

Connect Jenkins to GitHub



GitHub Server

Name

API URL

Credentials - none - ⚙ Add Jenkins

☐ Manage hooks

Test connection

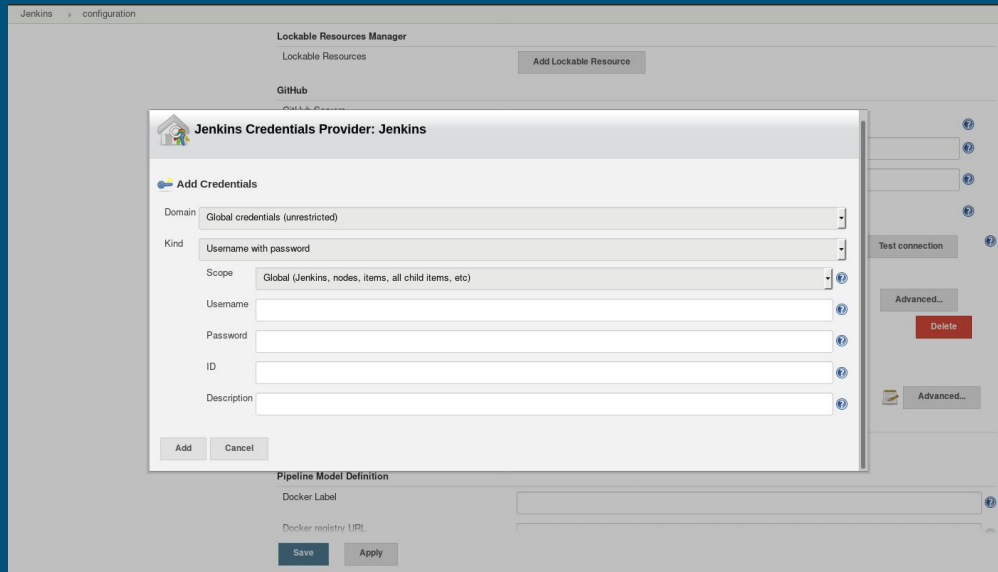
Advanced...

Delete

Add GitHub Server

Advanced...

Add a new Jenkins Credential.



Jenkins configuration

Lockable Resources Manager

Lockable Resources Add Lockable Resource

GitHub

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain Global credentials (unrestricted)

Kind Username with password

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

Username

Password

ID

Description

Add Cancel

Pipeline Model Definition

Docker Label

Docker registry URL

Save Apply

Test connection

Advanced...

Delete

Advanced...

Connect Jenkins to GitHub

Have the following fields filled as shown in the screenshot below:

The “Description” can be anything of your choice to describe the Secret Text.

The “ID” field can be left blank.



The screenshot shows the Jenkins 'Add Credentials' form. The title is 'Jenkins Credentials Provider: Jenkins'. Below the title is a section 'Add Credentials' with a key icon. The form has the following fields:

- Domain:** A dropdown menu with 'Global credentials (unrestricted)' selected.
- Kind:** A dropdown menu with 'Secret text' selected.
- Scope:** A dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected. There is a help icon (?) to the right.
- Secret:** A text input field filled with 20 black dots.
- ID:** An empty text input field. There is a help icon (?) to the right.
- Description:** A text input field with 'GitHub Token' entered. There is a help icon (?) to the right.

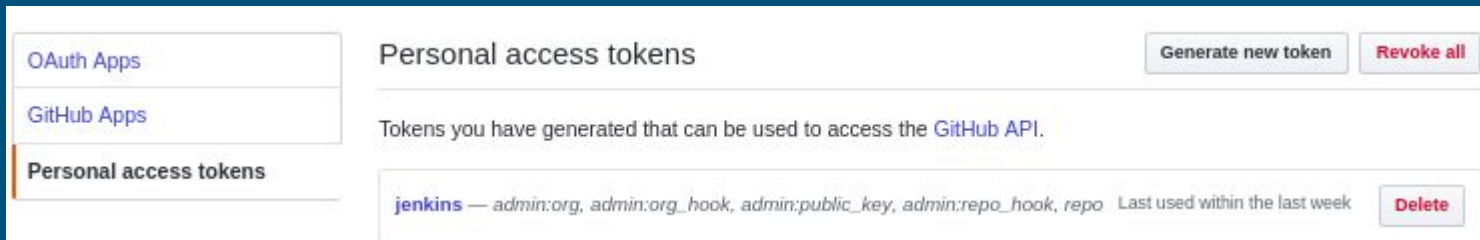
At the bottom of the form are two buttons: 'Add' and 'Cancel'.

To generate the “Secret” i.e. a token generated by GitHub see the next slide.

Connect Jenkins to GitHub

Log into GitHub and go to <https://github.com/settings/tokens> . Use the GitHub account that has access to the repository that you want Jenkins to pull code from.

Select “Personal access tokens” and click on “Generate new token”:



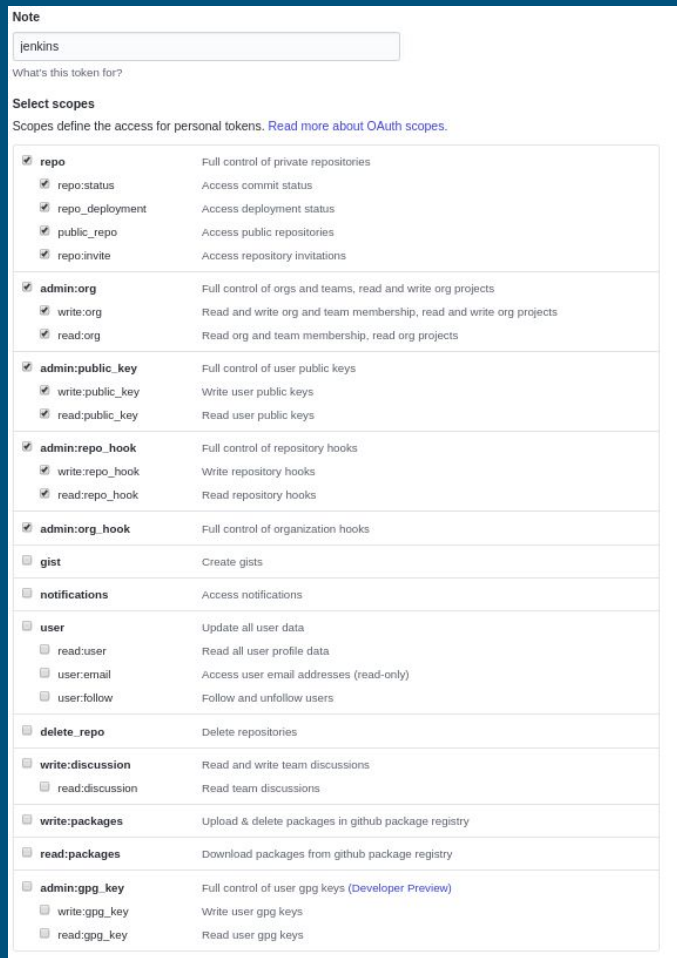
Connect Jenkins to GitHub

Select the follow scopes as shown in the screenshot.

If your repository is part of a organization, be sure to select “admin:org” and “admin:org_hook”.

Provide a descriptive note for this token in the “Note” field.

When done select,  .

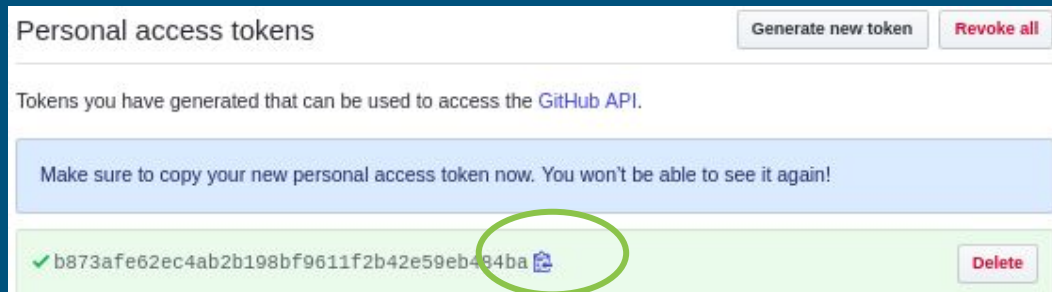


The screenshot shows the GitHub OAuth token creation page. At the top, there is a 'Note' field containing the text 'jenkins'. Below this is a section titled 'Select scopes' with a link to 'Read more about OAuth scopes'. The scopes are listed in a table with checkboxes for selection. The selected scopes are: repo, admin:org, admin:public_key, admin:repo_hook, admin:org_hook, delete_repo, write:discussion, write:packages, read:packages, and admin:gpg_key. The 'repo' scope is expanded, showing sub-scopes: repo:status, repo_deployment, public_repo, and repo:invite. The 'admin:org' scope is expanded, showing sub-scopes: write:org and read:org. The 'admin:public_key' scope is expanded, showing sub-scopes: write:public_key and read:public_key. The 'admin:repo_hook' scope is expanded, showing sub-scopes: write:repo_hook and read:repo_hook. The 'admin:org_hook' scope is expanded, showing sub-scopes: write:org_hook and read:org_hook. The 'delete_repo' scope is expanded, showing sub-scopes: delete_repo. The 'write:discussion' scope is expanded, showing sub-scopes: write:discussion. The 'write:packages' scope is expanded, showing sub-scopes: write:packages. The 'read:packages' scope is expanded, showing sub-scopes: read:packages. The 'admin:gpg_key' scope is expanded, showing sub-scopes: admin:gpg_key, write:gpg_key, and read:gpg_key.

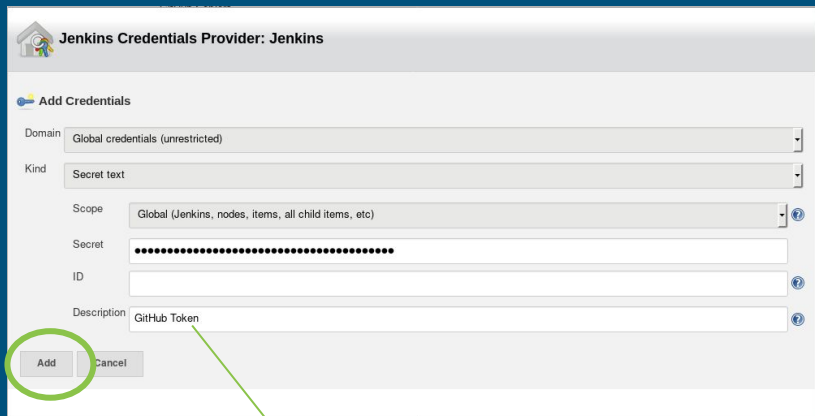
Scope	Description
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input checked="" type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input checked="" type="checkbox"/> read:org	Read org and team membership, read org projects
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input checked="" type="checkbox"/> write:public_key	Write user public keys
<input checked="" type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update all user data
<input type="checkbox"/> read:user	Read all user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> write:packages	Upload & delete packages in github package registry
<input type="checkbox"/> read:packages	Download packages from github package registry
<input type="checkbox"/> admin:gpg_key	Full control of user gpg keys (Developer Preview)
<input type="checkbox"/> write:gpg_key	Write user gpg keys
<input type="checkbox"/> read:gpg_key	Read user gpg keys

Connect Jenkins to GitHub

Copy the hash as shown in the screenshot, you can click on this icon to easily copy it.



Connect Jenkins to GitHub



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Secret text

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

Secret: [Redacted]

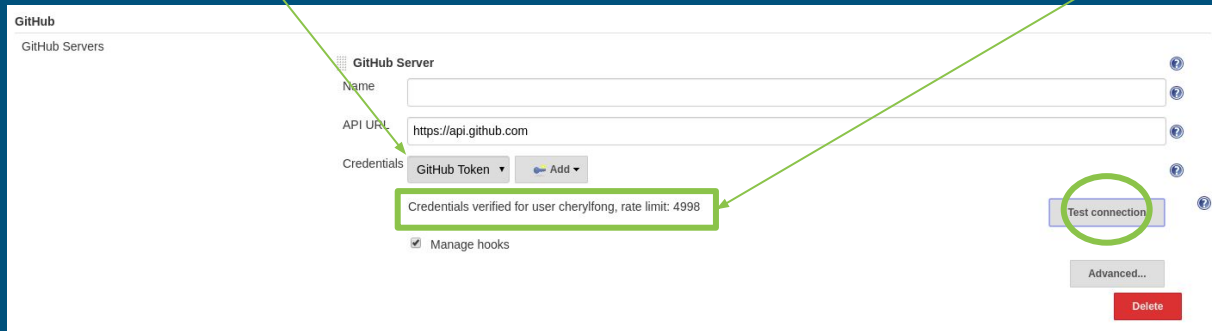
ID: [Redacted]

Description: GitHub Token

Add Cancel

After clicking
“Add”, test the
connection.

If successful, it
should state your
GitHub username
and the rate limit.



GitHub

GitHub Servers

GitHub Server

Name: [Redacted]

API URL: https://api.github.com

Credentials: GitHub Token Add

Credentials verified for user cheryllong, rate limit: 4998

☒ Manage hooks

Test connection

Advanced...

Delete

Connect Jenkins to GitHub

Up to this point, Jenkins will be able to access the source code on a repository via the GitHub Token. You need to make sure that the Token was created through an account that has push/pull access to the repository.

The settings on the next slides will allow Jenkins to push build status updates on GitHub and be triggered for builds when commits are pushed onto the “master” branch.

Connect Jenkins to GitHub

Set up a “Deploy key” on your repository, by going to:

<https://github.com/<name-of-organization>/<name-of-repository>/settings/keys>

Options

Collaborators & teams

Branches

Webhooks


Notifications

Integrations & services

Deploy keys

Deploy keys

Add deploy key

 **Jenkins**

Fingerprint: c5:a2:ad:13:c6:ce:e8:1b:25:a8:cb:62:53:64:88:25

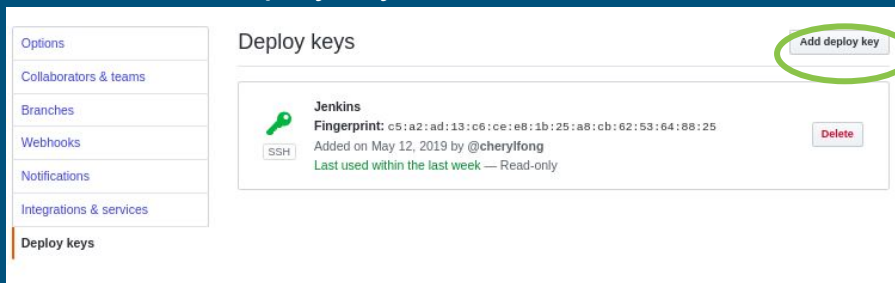
Added on May 12, 2019 by @cherylfong

Last used within the last week — Read-only

Delete

Connect Jenkins to GitHub

Select “Add deploy key”



Deploy keys / Add new

Title
Jenkins

Key
Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

☐ Allow write access
Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

See this GitHub guide on [Generating a new SSH Key and adding it to the SSH Agent](#) (Official GitHub Guide).

Connect Jenkins to GitHub

To Generate an SSH key on the Jenkins Image container, follow the commands in this screenshot:

```
[cherylfong@localhost csc648-sp19-team08]$ docker exec -it jenkins bash
jenkins@debbed0a97f6:/$ ssh-keygen -t rsa -b 4096 -C "Jenkins Deploy Key"
Generating public/private rsa key pair.
Enter file in which to save the key (/var/jenkins_home/.ssh/id_rsa):
Created directory '/var/jenkins_home/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/jenkins_home/.ssh/id_rsa.
Your public key has been saved in /var/jenkins_home/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:67tT/JQyFceN2cMiV//dBr+cG/9T7QswSzZJeX8kb+Y Jenkins Deploy Key
The key's randomart image is:
+---[RSA 4096]---+
|                 .0*  |
|                .00=0+|
|               00++ +|
|              . + .*+|
|             S. 0 . .%|
|            .B B .=*|
|           .. * . *E|
|          .. . .*=|
|         ++    0*|
+-----[SHA256]-----+
```

The passphrase was left blank deliberately.

Note down the passphrase if you choose to have it for this SSH key.

Connect Jenkins to GitHub

Finally add the SSH key to the SSH Agent.

```
jenkins@debbbed0a97f6:/$ eval "$(ssh-agent -s)"
Agent pid 752
jenkins@debbbed0a97f6:/$ ssh-add ~/.ssh/id_rsa
Identity added: /var/jenkins_home/.ssh/id_rsa (/var/jenkins_home/.ssh/id_rsa)
jenkins@debbbed0a97f6:/$
```

Slide 45 and 46 utilized content from [Generating a new SSH Key and adding it to the SSH Agent](#) (Official GitHub Guide).

Connect Jenkins to GitHub

After generating the SSH key pair, copy and paste the contents of the public key and paste it into the Key filed.



```
jenkins@debbbed0a97f6: ~/.ssh$ pwd
/var/jenkins_home/.ssh
jenkins@debbbed0a97f6: ~/.ssh$ ls
id_rsa id_rsa.pub
jenkins@debbbed0a97f6: ~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDB4qXc3Lbfjgl2oj0sZrqe15bN5t8lPn5BGQhIueZ
4X9dm7TkDM0Y307AaKKvbQU5K/zqUfMXAkwIyD6/PSIKsMPZjsFsx8bmSbV0SoK/u2ErI37mSvFmh
rYKG5Xt3rGBxwtWiSqGlzRCLWZySG+RSG/xKZDH005izsLZK6KxGKDNiv8MC/eF9vjP3H6WJy7FkURW
UdtfHZZx5kitvd0Y8EGoHx7Rwg33cXdfaEj/C1t88B/YncfsgUpe5e/rb0hkVEJWVSFKOUnifSaHtCm
ToYmIiqextzbdeC1JEqdJeRfVijK83gVFA50Z8wvLhdqizH2t3CL4gPxcYqp3+W7Cf+1fQr1NxLLaXS
YnBywQotag8UKuj4uxIE9lgx4BMLrz2WIArm3J4jxnVqogWpfpT4Q+M6HVPsvJKh2EP8H8WwezMsusA
HiqaRhRmnm2XZoJUP3zEwbnCVKcJpuQcqsyz09W3F3ZX1EI62MSOl1VZg5D/I54Iwr6kg5gCoT0tUP
Z51bf5prs3aPthWgRtd8c/A2MNSLiUV183G6ARjYyCqQH1ND+l+9r5dv5WAhEo8Da7KA4uxtVkkSuWg
WuPJtM36Uv3rSZRGxSzknH1icsefRgNvFJqVfDzLCLw/0+wxSxGgP8BngjuXMQCLBQcFM/KJA0V9zq
oQMRvAauX5Q== Jenkins Deploy Key
```

Deploy keys / Add new

Title

Jenkins

Key

ssh-rsa

```
AAAAB3NzaC1yc2EAAAADAQABAAQDB4qXc3Lbfjgl2oj0sZrqe15bN5t8lPn5BGQhIueZ4X9dm7TkDM0Y
307AaKKvbQU5K/zqUfMXAkwIyD6/PSIKsMPZjsFsx8bmSbV0SoK/u2ErI37mSvFmhrYKG5Xt3rGBxwtWiSq
GlzRCLWZySG+RSG/xKZDH005izsLZK6KxGKDNiv8MC/eF9vjP3H6WJy7FkURWUdtfHZZx5kitvd0Y8EGoHx7
Rwg33cXdfaEj/C1t88B/YncfsgUpe5e/rb0hkVEJWVSFKOUnifSaHtCmToYmIiqextzbdeC1JEqdJeRfVijK83gV
A5OZ8wvLhdqizH2t3CL4gPxcYqp3+W7Cf+1fQr1NxLLaXS YnBywQotag8UKuj4uxIE9lgx4BMLrz2WIArm3J4jxn
VqogWpfpT4Q+M6HVPsvJKh2EP8H8WwezMsusAHiqaRhRmnm2XZoJUP3zEwbnCVKcJpuQcqsyz09W3F3
ZX1EI62MSOl1VZg5D/I54Iwr6kg5gCoT0tUPZ51bf5prs3aPthWgRtd8c/A2MNSLiUV183G6ARjYyCqQH1ND+l+9
r5dv5WAhEo8Da7KA4uxtVkkSuWgWuPJtM36Uv3rSZRGxSzknH1icsefRgNvFJqVfDzLCLw/0+wxSxGgP8Bngj
uXMQCLBQcFM/KJA0V9zqoQMRvAauX5Q== Jenkins Deploy Key
```

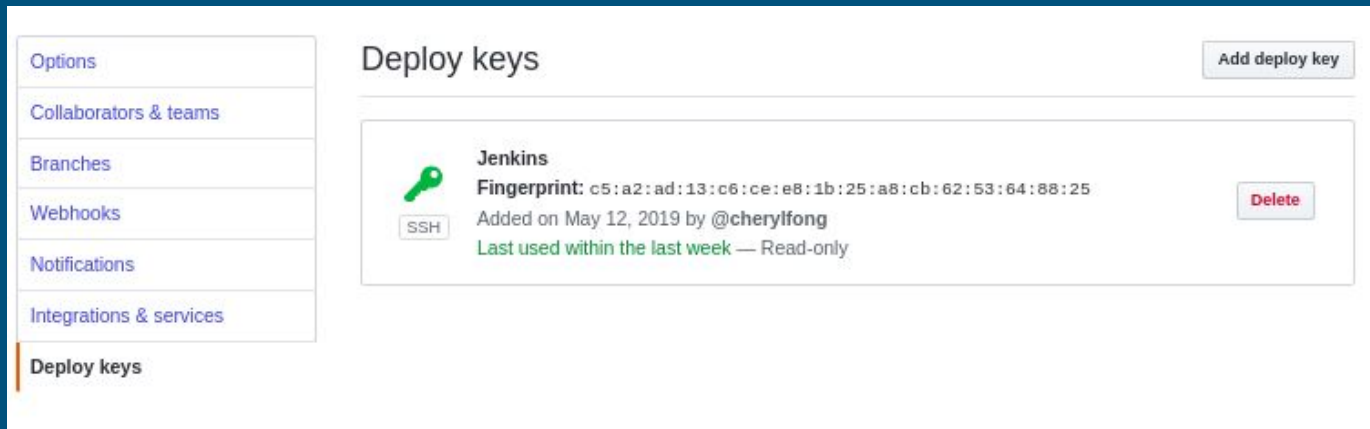
☐ Allow write access

Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

Leave the “Allow write access” unchecked and click “Add key”.

Connect Jenkins to GitHub



The screenshot displays the Jenkins web interface. On the left is a sidebar with navigation links: Options, Collaborators & teams, Branches, Webhooks, Notifications, Integrations & services, and Deploy keys (which is highlighted with an orange bar). The main content area is titled 'Deploy keys' and includes an 'Add deploy key' button in the top right. Below this, a table lists the configured keys. One key is shown for the user 'Jenkins' with a green key icon and an 'SSH' label. The key's fingerprint is 'c5:a2:ad:13:c6:ce:e8:1b:25:a8:cb:62:53:64:88:25'. It was added on May 12, 2019, by '@cherylfong' and is currently in 'Read-only' mode, having been last used within the last week. A 'Delete' button is located to the right of the key entry.

User	Key Type	Fingerprint	Added By	Added On	Last Used	Permissions	Actions
Jenkins	SSH	c5:a2:ad:13:c6:ce:e8:1b:25:a8:cb:62:53:64:88:25	@cherylfong	May 12, 2019	within the last week	Read-only	Delete

At this point you should have the above. See slide 64 on the Deployment Example section to see how this will be used.

SSH via Jenkins

This section “SSH” via Jenkins, will provide Jenkins permissions to SSH into a remote server e.g. an AWS EC2 server.

SSH via Jenkins

Go to “Manage Plugins” under “Manage Jenkins” on the left side of the Jenkins Dashboard

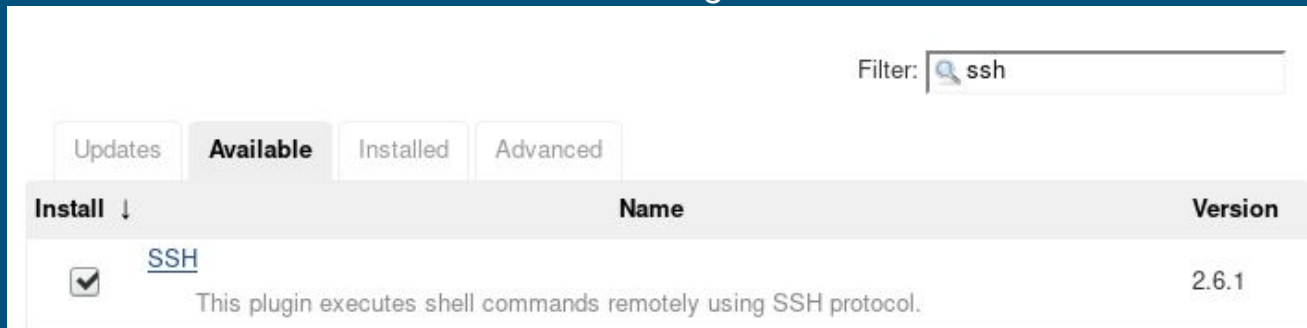


Manage Plugins

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

SSH via Jenkins

Select “Available” and search for “SSH Plugin”:



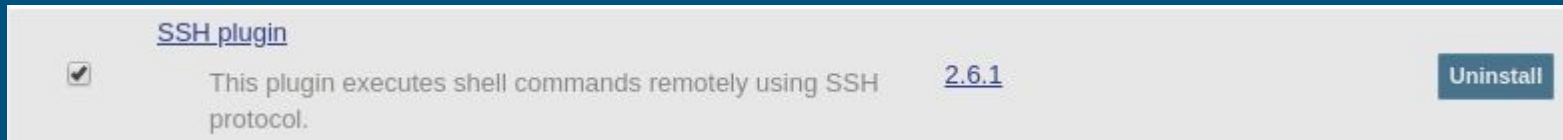
The screenshot shows the Jenkins 'Available' tab for plugins. A search filter 'ssh' is applied. The 'SSH' plugin is selected with a checkbox. The table below lists the available plugins.

Install ↓	Name	Version
<input checked="" type="checkbox"/>	SSH This plugin executes shell commands remotely using SSH protocol.	2.6.1

Select “Install without restart”.

Install without restart

You should be able to see this plugin under the “Installed” tab:

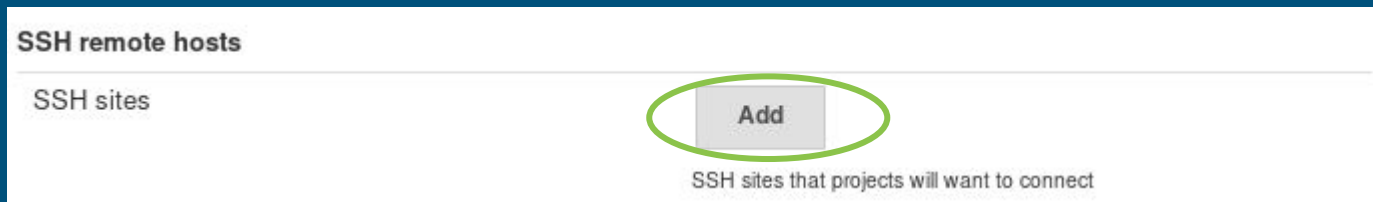


The screenshot shows the 'Installed' tab for plugins. The 'SSH plugin' is listed with a checkbox, description, version, and an 'Uninstall' button.

<input checked="" type="checkbox"/>	SSH plugin This plugin executes shell commands remotely using SSH protocol.	2.6.1	Uninstall
-------------------------------------	--	-----------------------	-----------

SSH via Jenkins

Select “Configure System” under “Manage Jenkins” and scroll down to find:



Click on “Add”.

SSH via Jenkins

SSH remote hosts

SSH sites

Hostname	<input type="text" value="13.52.51.45"/>	
Port	<input type="text" value="22"/>	
Credentials	<div>ubuntu (SSH Key 868) ▼ Add ▼</div>	
Pty	<input type="checkbox"/>	
serverAliveInterval	<input type="text" value="0"/>	
timeout	<input type="text" value="0"/>	

Fill the “Hostname” with the Public IP of your EC2 instance or remote server.

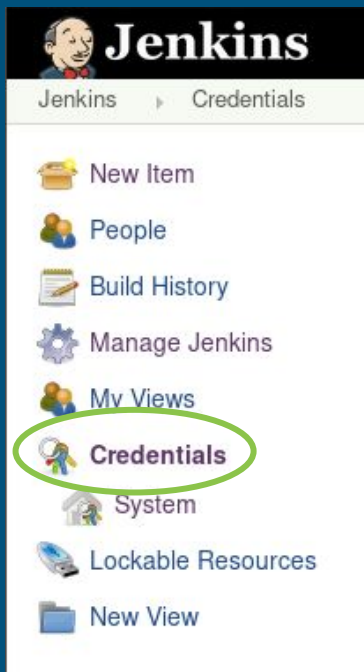
Have “Port” as 22. Port 22 is default for SSH.

Fill in the rest of the fields as shown in the screenshot.

See the next slide for filling in the “Credentials”.

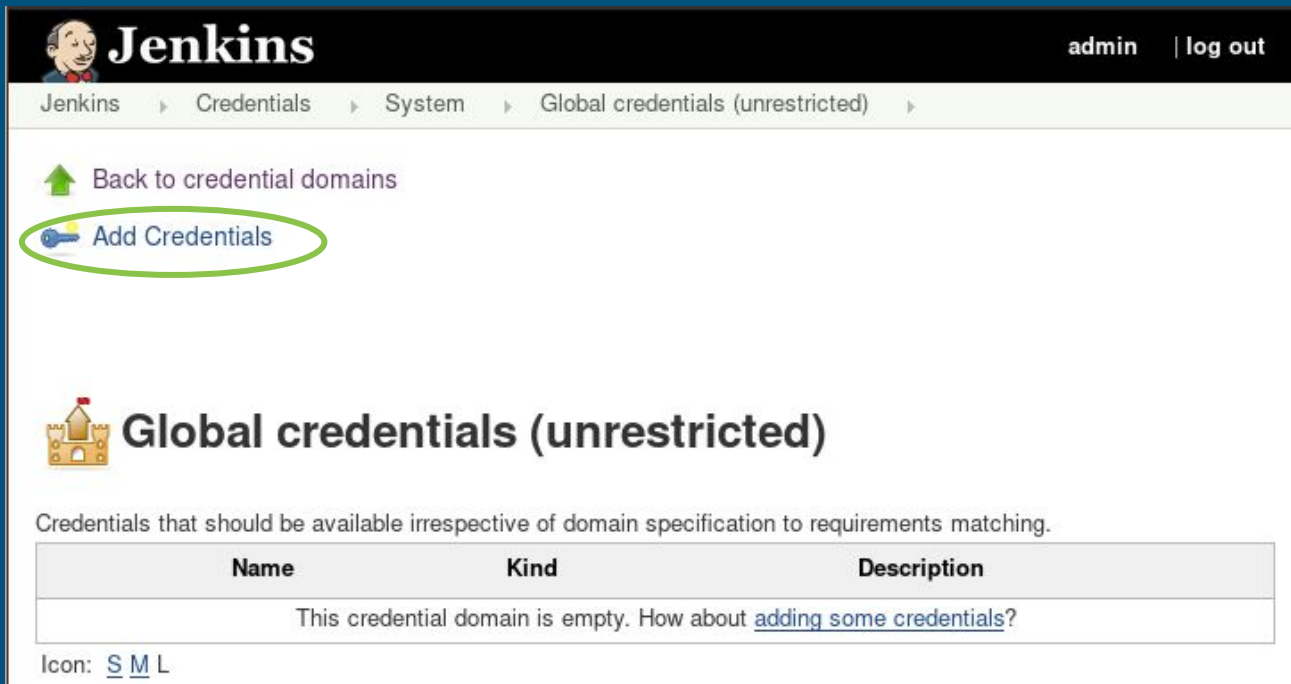
SSH via Jenkins

On the left side of the dashboard, under the same list as “Manage Jenkins”, select “Credentials”



Click on “(global)” as shown in the screenshot above.

SSH via Jenkins



The screenshot shows the Jenkins web interface. At the top, the Jenkins logo and name are on the left, and 'admin | log out' is on the right. Below the header is a breadcrumb trail: 'Jenkins > Credentials > System > Global credentials (unrestricted)'. On the left side, there are two links: 'Back to credential domains' with an upward arrow icon, and 'Add Credentials' with a key icon. The 'Add Credentials' link is circled in green. Below this, there is a section titled 'Global credentials (unrestricted)' with a castle icon. Underneath the title, it says 'Credentials that should be available irrespective of domain specification to requirements matching.' Below this text is a table with three columns: 'Name', 'Kind', and 'Description'. The table is currently empty, and a message at the bottom of the table says 'This credential domain is empty. How about [adding some credentials?](#)'.

Jenkins admin | log out

Jenkins > Credentials > System > Global credentials (unrestricted)

Back to credential domains

Add Credentials

Global credentials (unrestricted)

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

Name	Kind	Description
This credential domain is empty. How about adding some credentials?		

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

Click on “Add Credentials”.

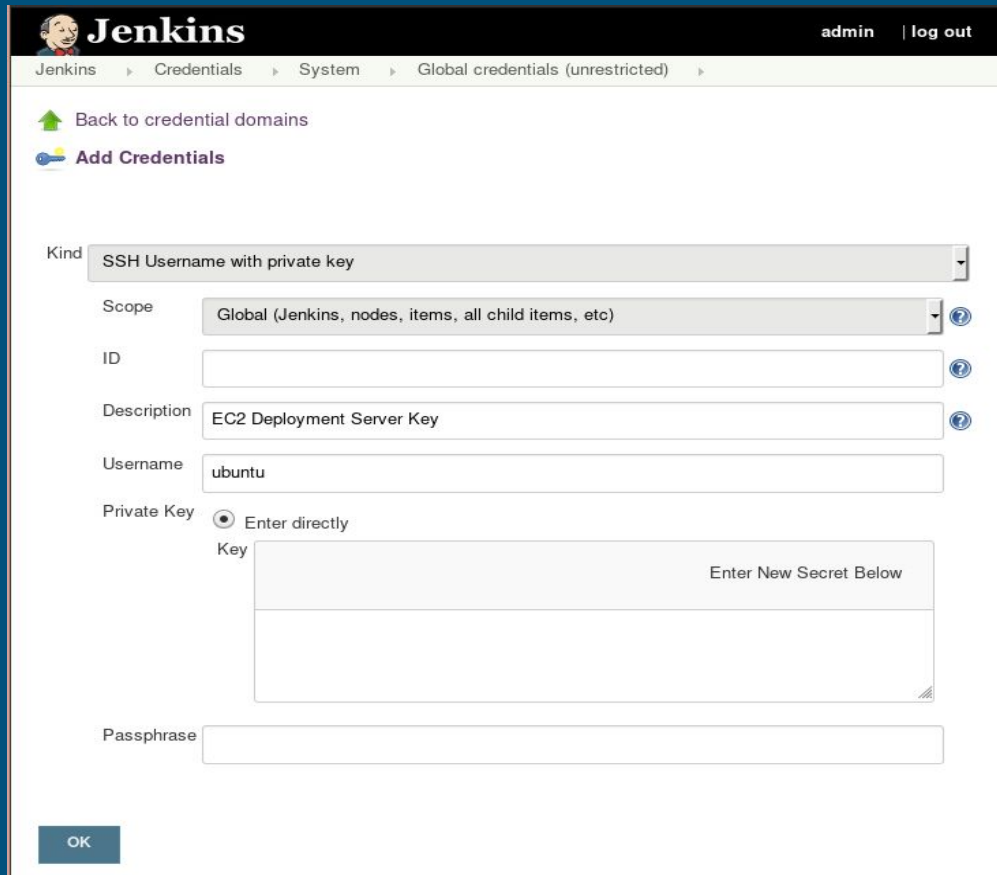
SSH via Jenkins

Select “Kind” as “SSH Username with private key”.

“Username” as the username used to SSH into the remote server or EC2 instance.

The “Private Key” is the contents of the “.pem” file as provided by AWS when provisioning an EC2 instance.

Otherwise, use a private key as generated by the SSH Keygen. Do not use the same key pair as the Deploy key in slide 48.



The screenshot shows the Jenkins web interface for adding a new credential. The breadcrumb trail is: Jenkins > Credentials > System > Global credentials (unrestricted). The page title is 'Jenkins' with a user 'admin' and a 'log out' link. A green arrow icon with the text 'Back to credential domains' is at the top left. Below it is a blue key icon with the text 'Add Credentials'. The 'Kind' dropdown is set to 'SSH Username with private key'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field is empty. The 'Description' field contains 'EC2 Deployment Server Key'. The 'Username' field contains 'ubuntu'. The 'Private Key' section has a radio button selected for 'Enter directly'. Below this is a large text area for the 'Key' with a placeholder 'Enter New Secret Below'. At the bottom of this area is a small 'OK' button. The 'Passphrase' field is empty.

Jenkins admin | log out

Jenkins > Credentials > System > Global credentials (unrestricted)

Back to credential domains

Add Credentials

Kind: SSH Username with private key

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

ID:

Description: EC2 Deployment Server Key

Username: ubuntu

Private Key: ☒ Enter directly

Key: Enter New Secret Below

Passphrase:

OK

SSH via Jenkins

Select “Kind” as “SSH Username with private key”.

“Username” as the username used to SSH into the remote server or EC2 instance.

The “Private Key” is the contents of the “.pem” file as provided by AWS when provisioning an EC2 instance.

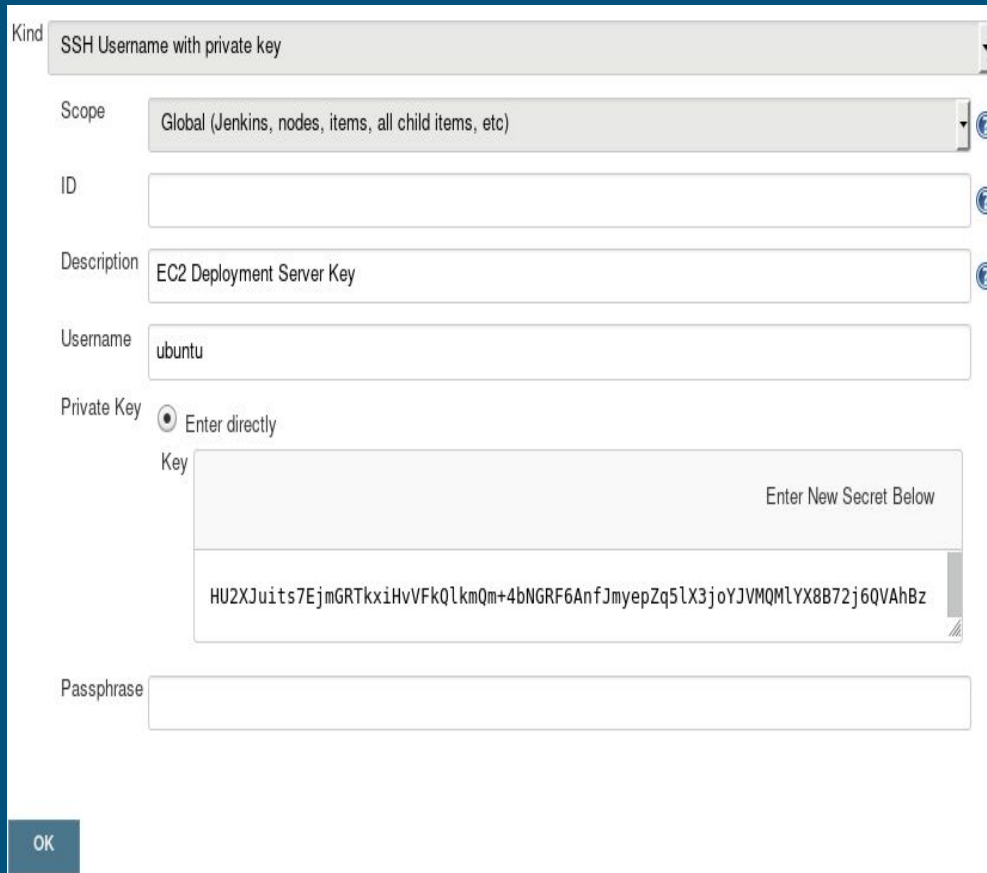
Otherwise, use a private key as generated by the SSH Keygen. Do not use the same key pair as the Deploy key in slide 48.

```
[cherylfong@localhost .ssh]$ cat 868-server.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAKtCG8Ag95rZRfmmHpL+Y4LjF0fullMl21Ziy1paTM1e+uqTh3cYlVukvturN
0hyQdoUvKcflW0kL1C8wRdxrfGQtG24o0d8ts2ZtTY0577cJSLaqczjj+HxN2vQcPBGGt11bckJg
S5WYJn6+CREQe2BE8cS0Jgu/+3z8B4PGQNLsY7L0ytozxm3gQ5LT8yjmRgq/Ej/zuEd0yYt9SpU
IEij3Crbe0dLhiudbUtGkNX9EkyMkM9HWzeM1eNp9p6EdtXxE5XbwCAKD9ELyWxrGU0t4FyT914W
0REVLdr+Ilt14gmAhImVIZ6fKsVJyzMYz4VDQlqtqGxkSc7YkqjSFwIDAQABAoIBAD7QiWtZm2KC
zwtPC4Ipl7d0VxUongeyyu1Ij/vQINrux+QDd7/zbzZoPsn8XGNxZT3QbgVqPk+oSS9XoUfCrxph
fLRpSJ9Nrm/pRRDHouFrAiLmJLEz3sUbE5Q3wbfoS6C+tPzhY3jTa5I1b6nSAGUC5RkYi5R/imAt
7BuhuI66IWA9cbjfWxetP0eBrM3sMgfj0MyAjbd/LHQQ3gECgYAuc0ghz5ryb0gB83+eJTcYF9C+
1T6pl0BdjK0zZEFvfbEkEFQRnkvsJ3yu2+ett7gecfZSCGESmImisflWbjQxLDPlILpVxK0Xirwr
mFNpdWb9dR/LMCABocp9gqD0Wg/4E7TRGGsZkk+dH1YawqSu99RBmNB8wabfDuG99Ux/bw==
-----END RSA PRIVATE KEY-----[cherylfong@localhost .ssh]$
```

SSH via Jenkins

SSH Keys provided by AWS (i.e. .pem file) usually do not have a passphrase associated.

If your generated SSH key has a passphrase, add the passphrase to the "Passphrase" field.



The image shows a screenshot of the Jenkins 'SSH Key' configuration page. The form is titled 'SSH Username with private key' under the 'Kind' dropdown. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field is empty. The 'Description' field contains 'EC2 Deployment Server Key'. The 'Username' field contains 'ubuntu'. The 'Private Key' section has the 'Enter directly' radio button selected. Below it, the 'Key' field contains a long alphanumeric string: 'HU2XJuits7EjmGRTkxiHvVFkQlkmQm+4bNGRF6AnfJmyepZq5LX3joYJVMQMLYX8B72j6QVAhBz'. The 'Passphrase' field is empty. An 'OK' button is at the bottom left.

Kind: SSH Username with private key

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

ID:

Description: EC2 Deployment Server Key

Username: ubuntu

Private Key: ☒ Enter directly

Key: HU2XJuits7EjmGRTkxiHvVFkQlkmQm+4bNGRF6AnfJmyepZq5LX3joYJVMQMLYX8B72j6QVAhBz

Passphrase:

OK

SSH via Jenkins

Add the “Credentials” that were just created in slide 59.

Then check the connection, it should say “Successful connection”.

SSH remote hosts

SSH sites

Hostname	<input type="text" value="13.52.51.45"/>	
Port	<input type="text" value="22"/>	
Credentials	<div>ubuntu (SSH Key 868) ▼ Add ▼</div>	
Pty	<input type="checkbox"/>	
serverAliveInterval	<input type="text" value="0"/>	
timeout	<input type="text" value="0"/>	
<div>Successfull connection</div>		<div>Check connection</div>
		<div>Delete</div>

Add

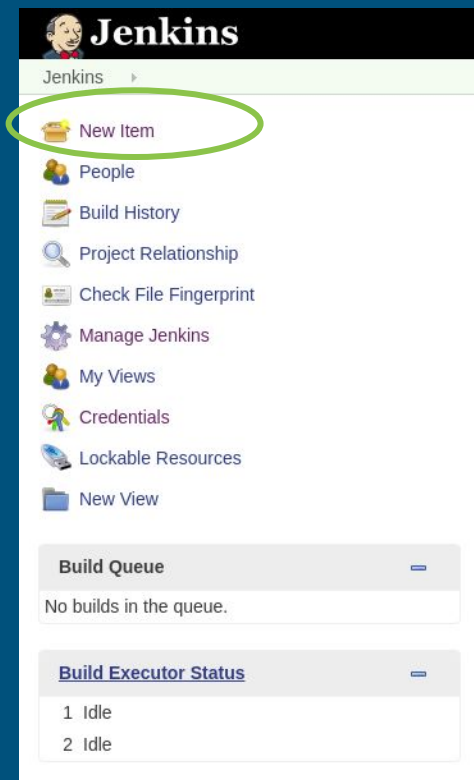
SSH sites that projects will want to connect

Deployment Example

This example, will utilize:

1. The **GitHub** section in “Configure System” (slide 32 onwards)
2. The **SSH remote hosts** section in “Configure System” (slide 50 onwards)
3. The Deploy key (slide 43 onwards)

Deployment Example



The screenshot shows the Jenkins dashboard. The 'New Item' button, represented by a folder icon, is circled in green. A green arrow points from this button towards the right-hand screenshot.

Jenkins

Jenkins >

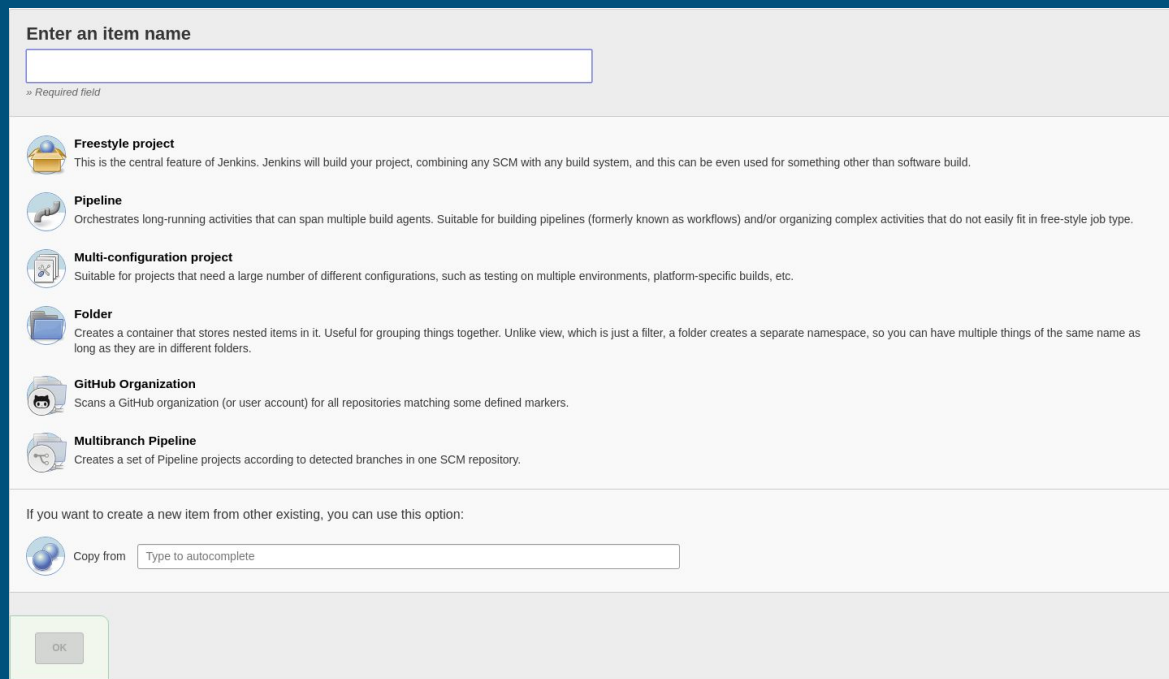
- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views
- Credentials
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle



The screenshot shows the 'Enter an item name' dialog in Jenkins. It features a text input field for the item name, a list of project types with their descriptions, and a 'Copy from' section for creating items from existing ones. An 'OK' button is at the bottom.

Enter an item name

* Required field

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**
Orchestrates long-running activities that can span multiple build 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.
- 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.

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

Copy from

OK

Deployment Example


Provide a name for your deployment pipeline and select “Freestyle project”.


Click “OK” when done.


Enter an item name


Test


* Required field


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

**Pipeline**
Orchestrates long-running activities that can span multiple build 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.

**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.

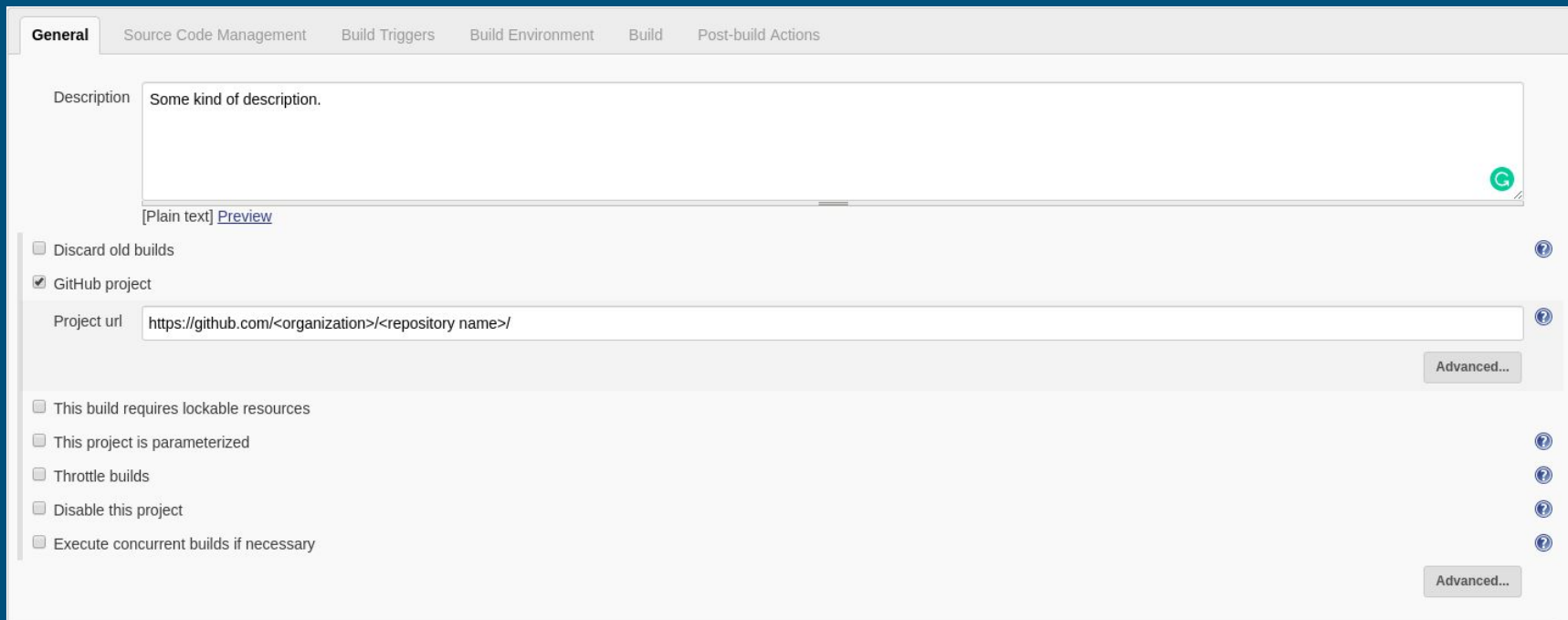
If you want to create a new item from other existing, you can use this option:

Copy from

OK

Deployment Example

Fill in the fields as shown in the screenshot below:

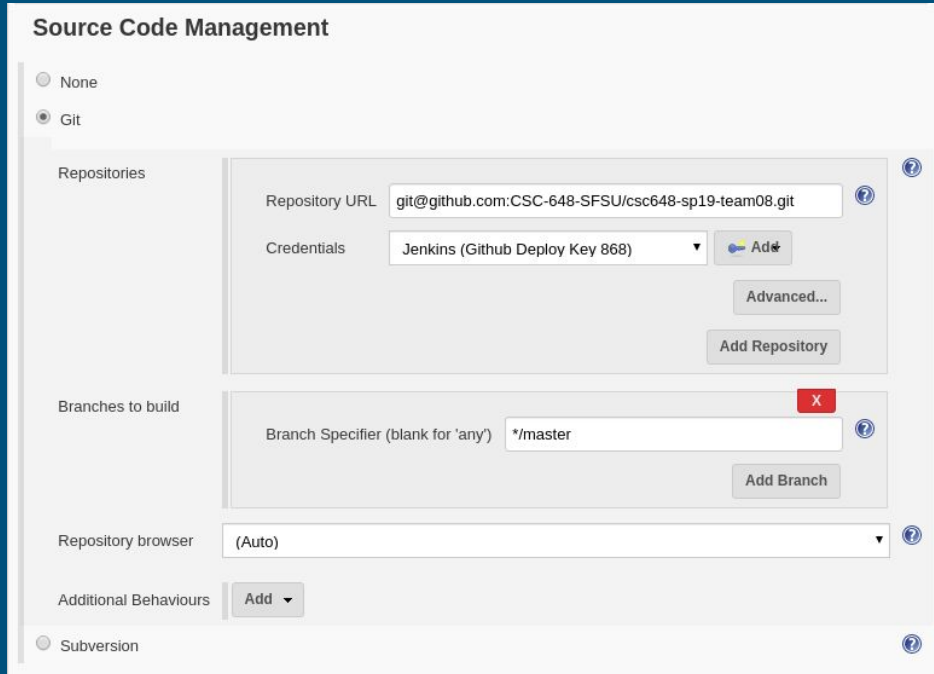


The screenshot shows the Jenkins configuration page for a new project, with the 'General' tab selected. The page has a top navigation bar with tabs: 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab contains the following fields and options:

- Description:** A text area containing 'Some kind of description.' with a green 'G' icon in the bottom right corner.
- [Plain text] Preview:** A link to preview the description.
- Discard old builds:** An unchecked checkbox.
- GitHub project:** A checked checkbox.
- Project url:** A text field containing 'https://github.com/<organization>/<repository name>/'. A question mark icon is to the right.
- Advanced...:** A button to expand advanced options.
- Other options (all unchecked):**
 - This build requires lockable resources
 - This project is parameterized
 - Throttle builds
 - Disable this project
 - Execute concurrent builds if necessary
- Advanced...:** A button to expand advanced options at the bottom.

Deployment Example

Fill in the fields as shown in the screenshot below:



The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' radio button is selected under the 'Source Code Management' section. The 'Repositories' section contains a 'Repository URL' field with the value 'git@github.com:CSC-648-SFSU/csc648-sp19-team08.git' and a 'Credentials' dropdown menu set to 'Jenkins (Github Deploy Key 868)'. There are buttons for 'Advanced...', 'Add Repository', and 'Add'. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value '*/master' and an 'Add Branch' button. The 'Repository browser' dropdown is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. The 'Subversion' radio button is unselected.

Under “Credentials”, select the Deploy key as created on slide 43 onwards.

Under Repository URL:

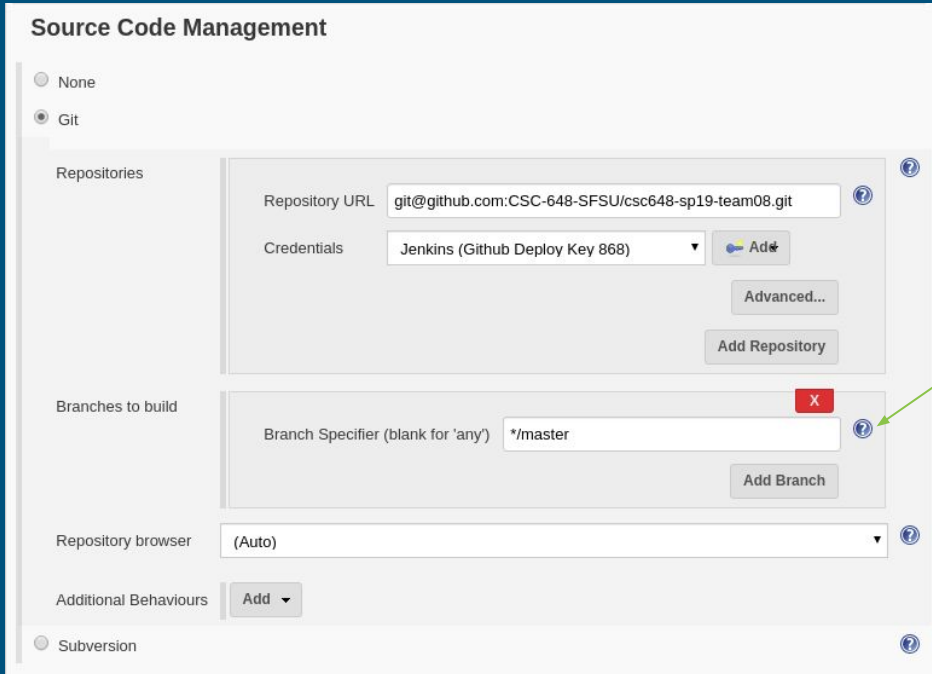
Provide the following that can be found on the GitHub repository homepage.



The screenshot shows the GitHub repository clone/download options. The 'Clone or download' button is highlighted. Below it, the 'Clone with SSH' option is selected, showing the repository URL 'git@github.com:CSC-648-SFSU/csc648-sp19-team08.git' and a button to 'Download ZIP'.

Deployment Example

Fill in the fields as shown in the screenshot below:



The screenshot shows the 'Source Code Management' configuration page in Jenkins. It features a sidebar with radio buttons for 'None', 'Git' (selected), and 'Subversion'. The main area is divided into sections: 'Repositories' with fields for 'Repository URL' (git@github.com:CSC-648-SFSU/csc648-sp19-team08.git) and 'Credentials' (Jenkins (Github Deploy Key 868)), and 'Branches to build' with a 'Branch Specifier (blank for 'any')' field containing '*/master'. There are also buttons for 'Add Repository', 'Add Branch', and 'Advanced...'. At the bottom, there is a 'Repository browser' dropdown set to '(Auto)' and an 'Additional Behaviours' section with an 'Add' button. A green arrow points from the text 'Select the branch that you want Jenkins to build, under "Branches to build":' to the question mark icon next to the 'Branch Specifier' field.

Select the branch that you want Jenkins to build, under “Branches to build”:

In the screenshot, this Jenkins Pipeline will build the “master” branch. Select the question mark for more information.

Deployment Example

Fill in the fields as shown in the screenshot below:

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

?

?

?

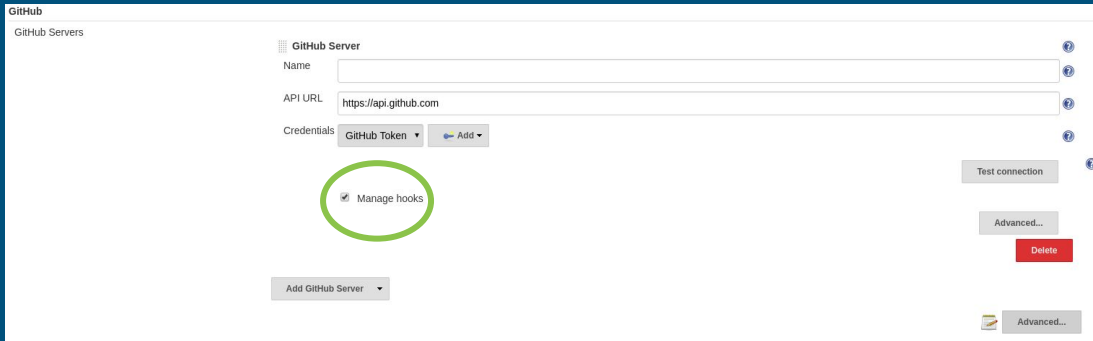
?

?

Deployment Example

To verify that the Build Triggers works:

Manage hooks must be selected, when completing slide 4.



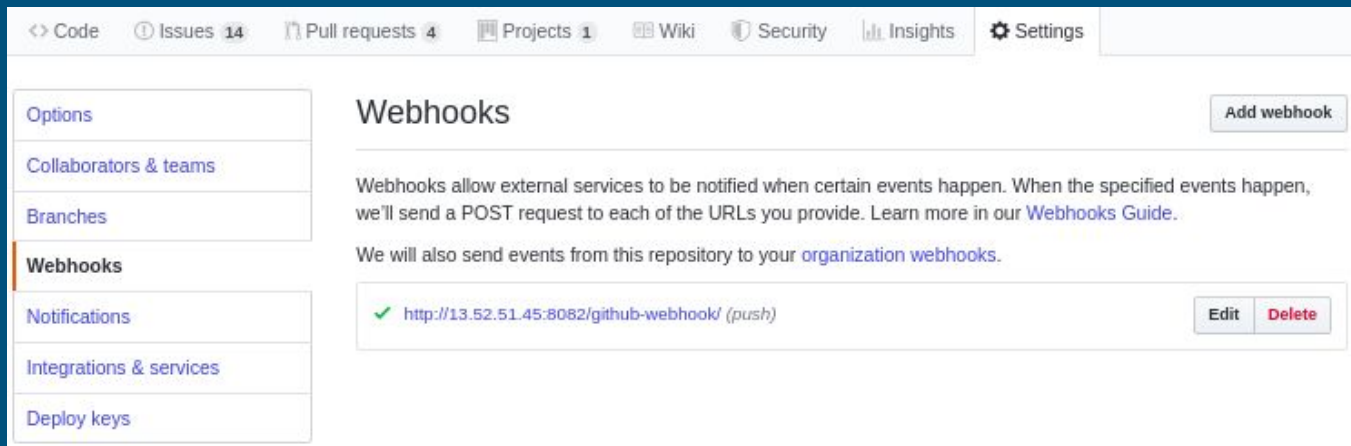
The screenshot shows the 'Add GitHub Server' configuration page in the GitHub interface. The form includes fields for 'Name', 'API URL' (pre-filled with 'https://api.github.com'), and 'Credentials' (set to 'GitHub Token'). A green circle highlights the 'Manage hooks' checkbox, which is checked. Other visible elements include a 'Test connection' button, an 'Advanced...' link, a 'Delete' button, and an 'Add GitHub Server' button at the bottom left.

Next go to <https://github.com/<organization>/<repository-name>/settings/hooks>

Deployment Example

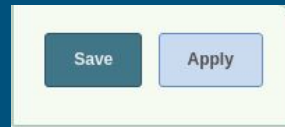
To verify that the Build Triggers works:

This Webhooks setting should be automatically populated for you when you “Save” or “Apply” the Jenkins Freestyle project.



The screenshot shows the GitHub repository settings page for a repository. The top navigation bar includes links for Code, Issues (14), Pull requests (4), Projects (1), Wiki, Security, Insights, and Settings. The left sidebar contains a list of settings categories: Options, Collaborators & teams, Branches, Webhooks (selected), Notifications, Integrations & services, and Deploy keys. The main content area is titled 'Webhooks' and includes an 'Add webhook' button. Below the title, there is explanatory text: '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](#).' and 'We will also send events from this repository to your [organization webhooks](#).' A table lists the configured webhooks, showing a single entry with a green checkmark, the URL 'http://13.52.51.45:8082/github-webhook/' (push), and buttons for 'Edit' and 'Delete'.

Webhook	Events	Actions
✓ http://13.52.51.45:8082/github-webhook/ (push)		Edit Delete



Deployment Example

To verify that the Build Triggers works:

The Webhooks settings for your Jenkins Freestyle Project should look something like this.

Add a new webhook if you do not see the webhook on slide 68.

Webhooks

Add webhook

Note the payload URL:

<https://public-ip:port-number/github-webhook/>

The rest of the fields in the screenshot are default settings.

Webhooks / Manage 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 *

Content type

Secret

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.

Deployment Example

To verify that the Build Triggers works:

This will confirm your settings from the previous slides are working.

Updating your webhook by clicking “Update webhook”, will deliver a payload to Jenkins. You can send another by pressing the “...” button and “Redeliver”.

A successful delivery will have a check mark shown on the right →

Options

Collaborators & teams

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Webhooks / Manage 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 *

Content type

Secret

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.

Recent Deliveries

<input checked="" type="checkbox"/>	<input type="checkbox"/>	63ec3ca2-7cf1-11e9-90eb-c743656b8221	2019-05-22 17:05:54	<input type="button" value="Redeliver"/>
-------------------------------------	--------------------------	--------------------------------------	---------------------	--

Request Response **296** Completed in 0.17 seconds.

Headers

Request URL: http://13.52.51.45:8082/github-webhook/
Request method: POST
content-type: application/json
Expect:

Deployment Example

SSH into the EC2 server or remote host:

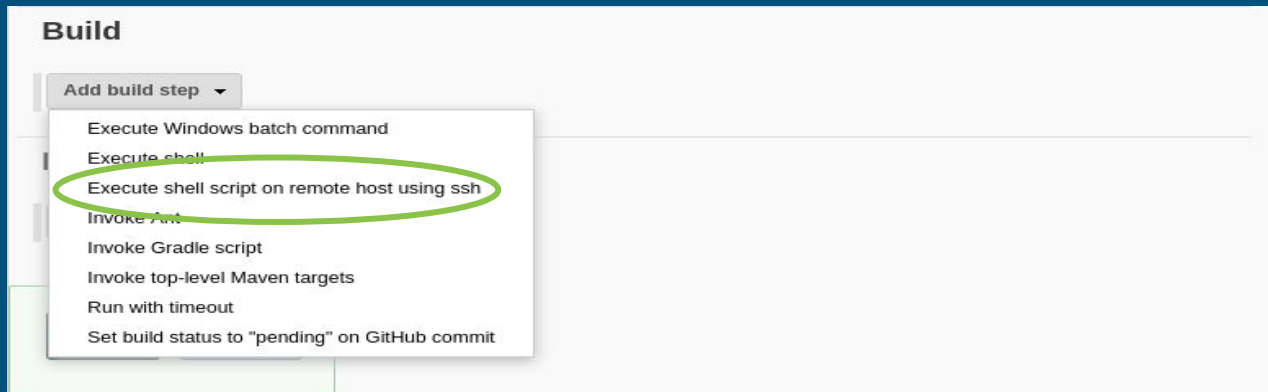
Scroll down of the Freestyle project settings to look for “Build”.

Build

Add build step ▼

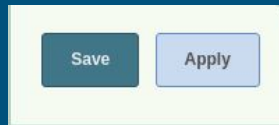
Deployment Example

Select “Execute shell script on remote host using ssh”:



Deployment Example

The “SSH site not specified” warning will disappear once, the “Apply” button for the Freestyle project has been clicked.



Build

Execute shell script on remote host using ssh ✕

SSH site

Command

SSH Site not specified



Execute each line ☐

Hide command from console output ☐

Add build step ▾

Deployment Example

The following git commands will update git references, checkout to the master branch and then pull the latest updates on master.

 Execute shell script on remote host using ssh 

SSH site

ubuntu@13.52.51.45:22

Command


```
docker ps -a
cd /home/ubuntu/csc648-sp19-team08/ && git fetch
cd /home/ubuntu/csc648-sp19-team08/ && git status
cd /home/ubuntu/csc648-sp19-team08/ && git checkout master
cd /home/ubuntu/csc648-sp19-team08/ && git pull
```

Execute each line

☒

Hide command from console output

☐



Deployment Example

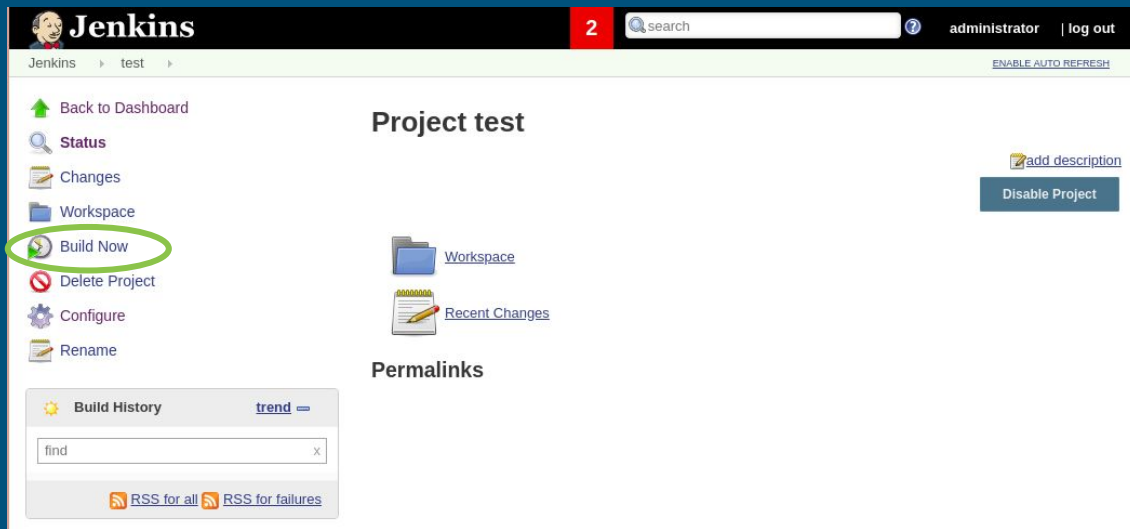
Save the Freestyle project configuration by clicking “Save”.



You will automatically be brought to the Project dashboard.

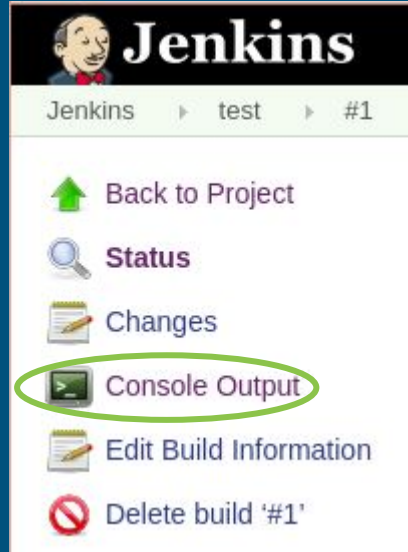
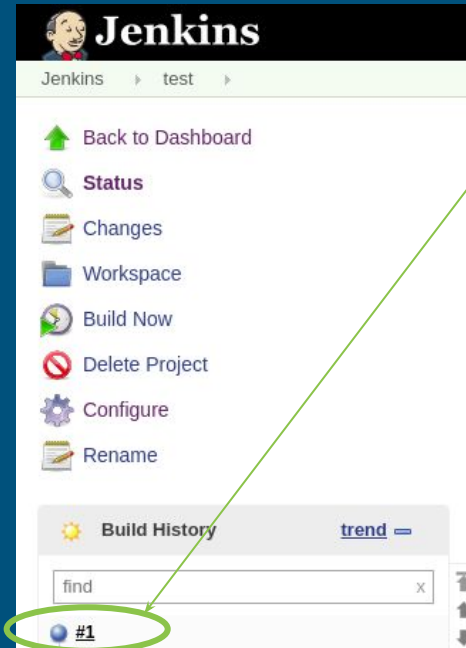


Click “Build Now”.



Deployment Example

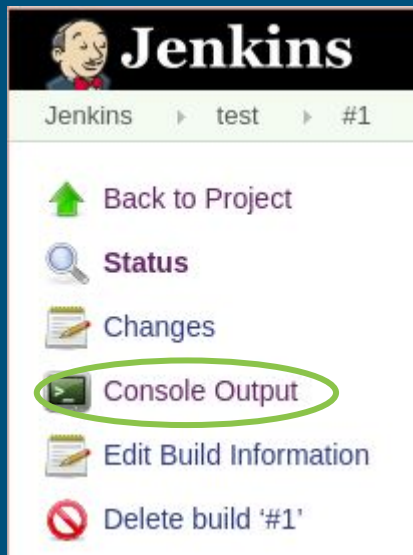
After clicking “Build Now”, you’ll see build #1:



Click on the “Console Output”.

Deployment Example

The “Console Output” should look something like this:



SSH INTO AWS EC2 868 INSTANCE

[SSH] commands:

```
docker ps -a
cd /home/ubuntu/csc648-sp19-team08/ && git fetch
cd /home/ubuntu/csc648-sp19-team08/ && git status
cd /home/ubuntu/csc648-sp19-team08/ && git checkout master
cd /home/ubuntu/csc648-sp19-team08/ && git pull
```

From <https://github.com/CSC-648-SFSU/csc648-sp19-team08>

29246ea..91dcfad master -> origin/master

On branch master

Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

nothing to commit, working tree clean

Already on 'master'

Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

Updating 29246ea..91dcfad

Fast-forward

```
...M5 docs and files go here, no code should be here | 0
.../M5/CSC648-848 Spring 2019 Milestone5 Team 8.pdf | Bin 0 -> 10150332 bytes
```

```
2 files changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 Milestones/M5/ALL M5 docs and files go here, no code should be
```

```
here
```

```
create mode 100644 Milestones/M5/CSC648-848 Spring 2019 Milestone5 Team 8.pdf
```

[SSH] completed

[SSH] exit-status: 0

Deployment Example

The “Console Output” log continued:

```
[Set GitHub commit status (universal)] SUCCESS on repos
```

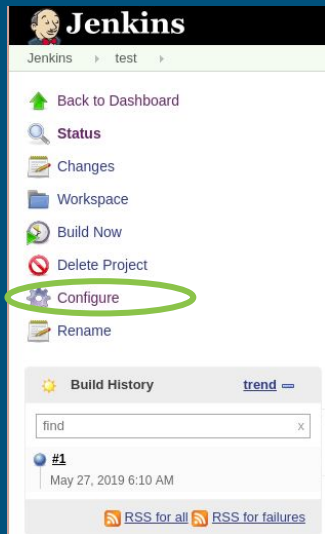
```
team08/commit/791de-fed3eb767f2e6cdb234cdac5c29ada19d82c  
Finished: SUCCESS
```

The “Finished: SUCCESS” indicates that this build is successful. All SSH Commands must return 0, for an entire build to be successful. As shown on slide 78:

```
[SSH] completed  
[SSH] exit-status: 0
```

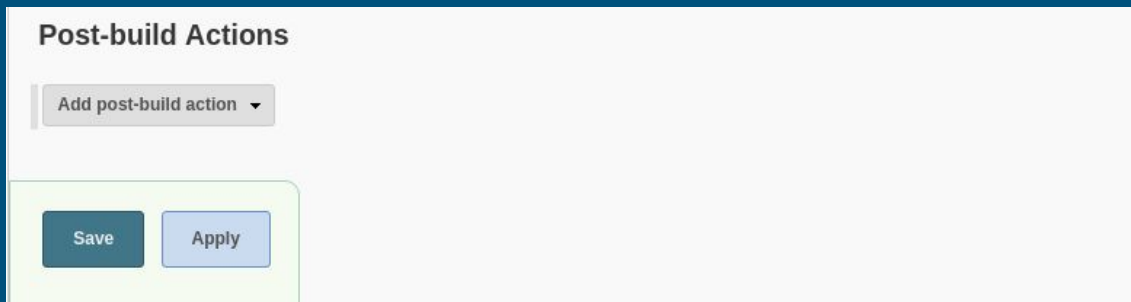
Deployment Example

Posting Build Statuses on Github:



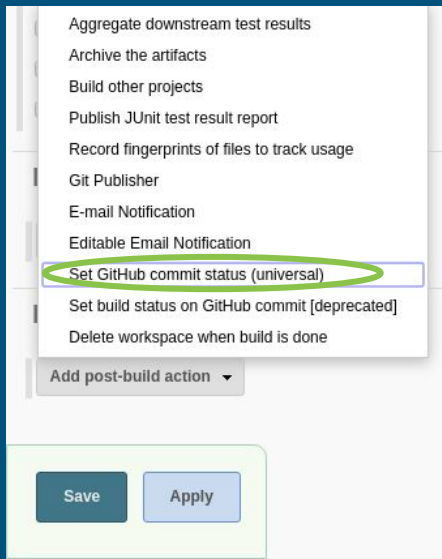
Go back to the Project's dashboard and select "Configure".

Scroll down to find "Post-build Actions":



Deployment Example

Posting Build Statuses on Github:

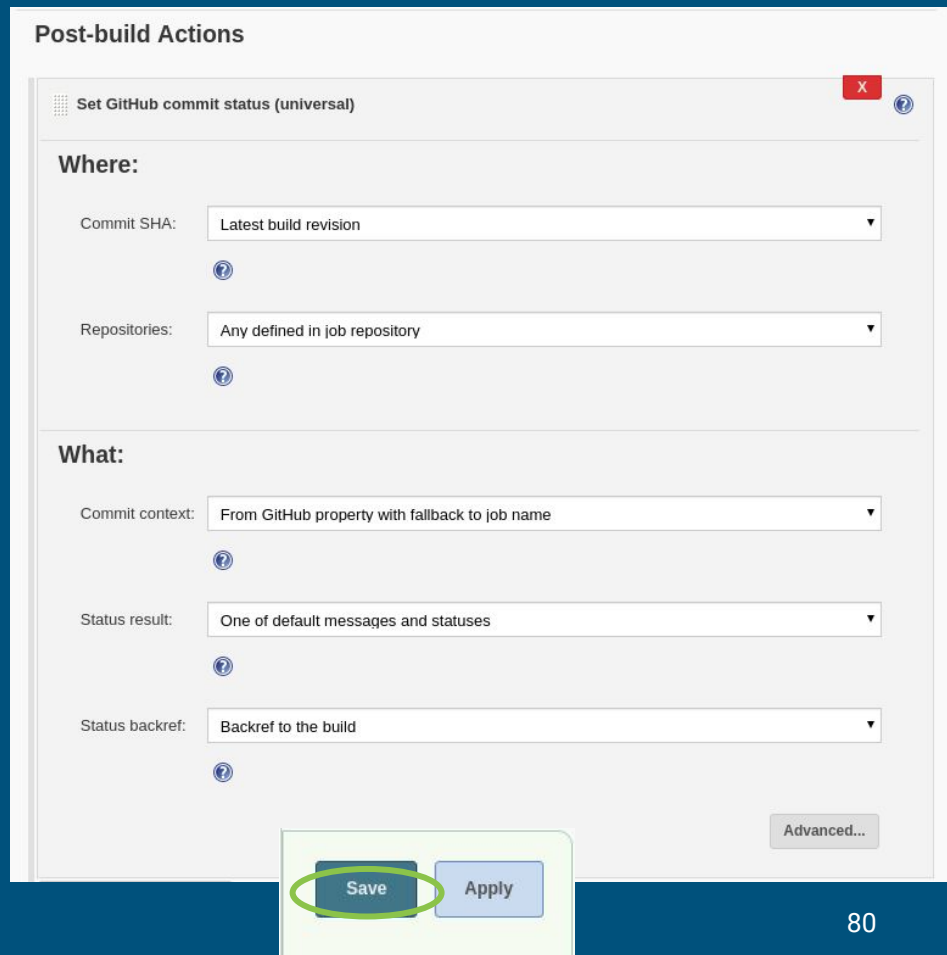


Select "Set Github commit status (universal)".



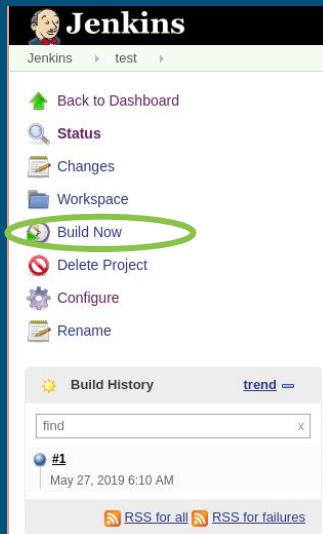
Fill in the fields as shown in the screenshot. These are default settings.

Click "Save" when complete.



Deployment Example

Posting Build Status on Github:



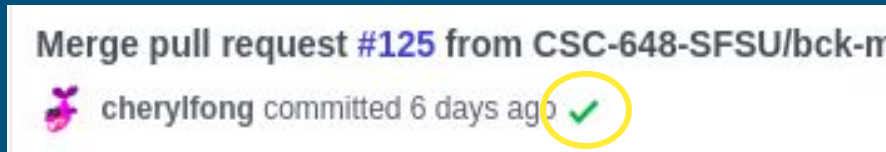
Click “Build Now” to initiate a build.

Upon completion of the build you should see a checkmark or a crossmark next to the latest commit on your GitHub repository.

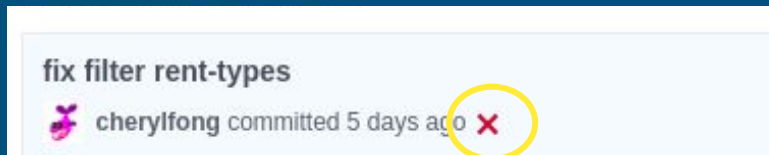
Go to the repository’s commit history at:

<https://github.com/<organization>/<repository-name>/commits/master>

Successful Build:



Unsuccessful Build:





FIN



Cheryl Fong

- View this slidedeck on [Google slides](#)
 - For questions or suggestions please email: cheryl.fong@qq.com
- 