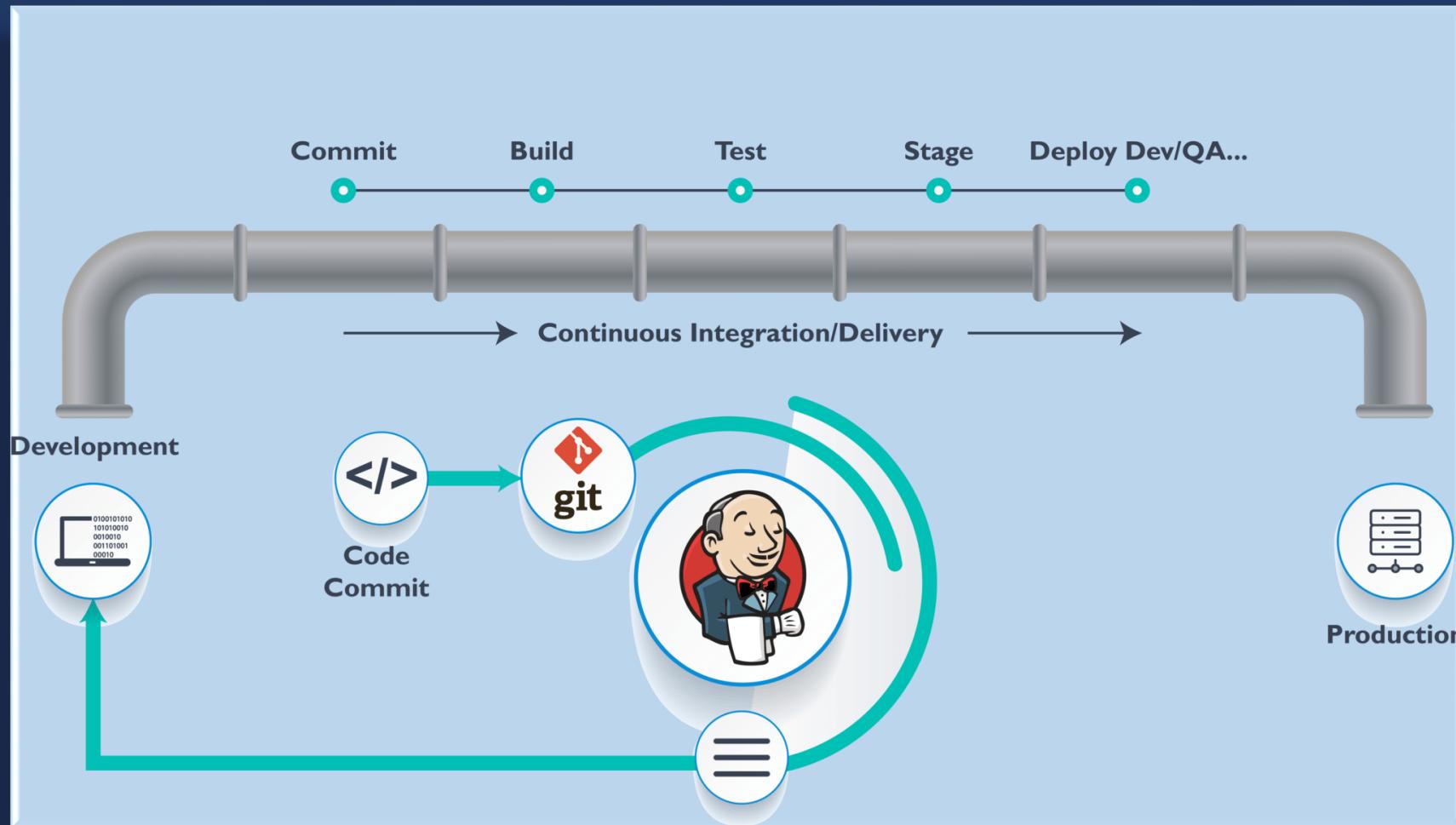


Software Production Engineering

SDLC, Jenkins and CI/CD Pipeline

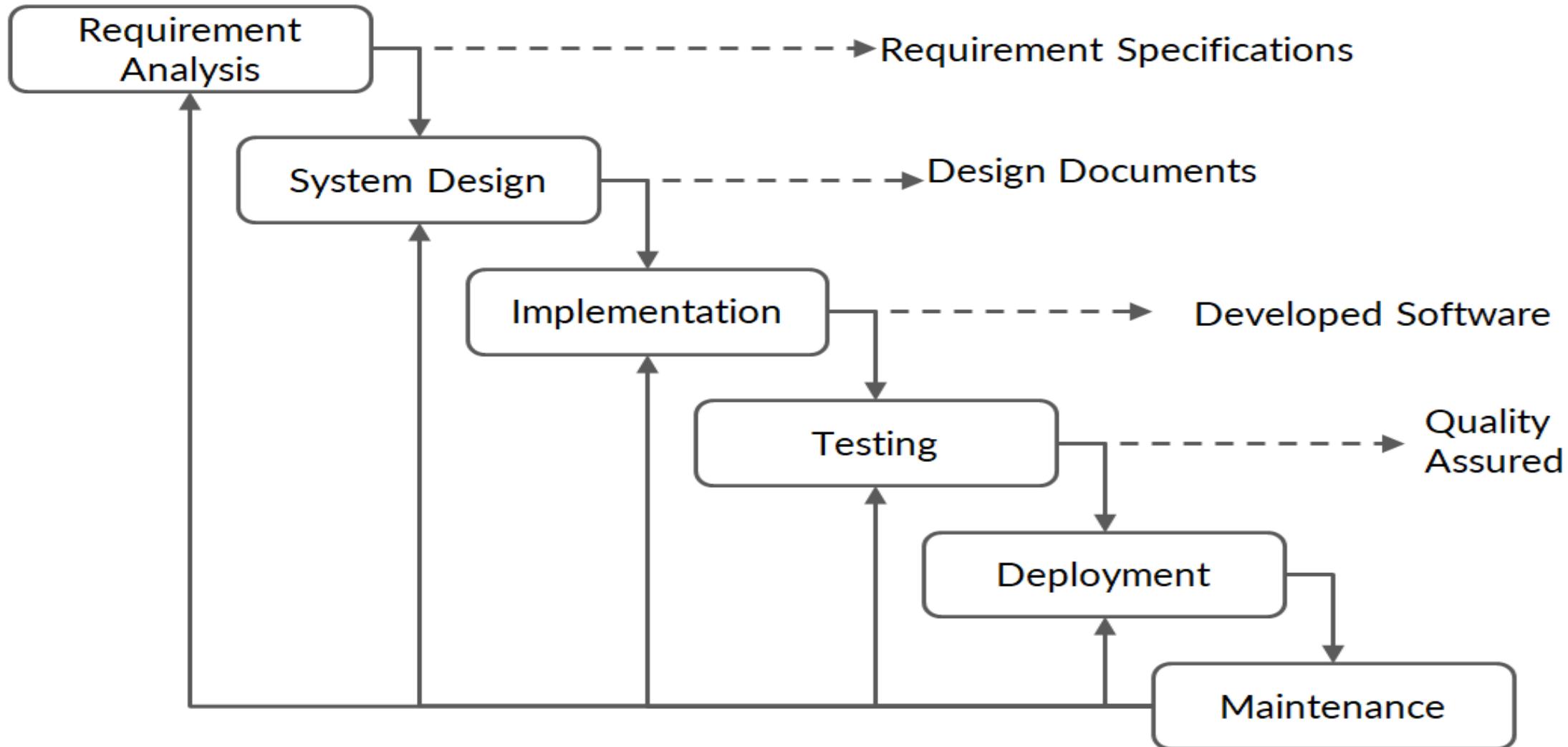


Agenda

- Understanding the working of traditional SDLC approach
- Introduction to Jenkins as a CI tool
- Creating Jobs in Jenkins and execute internally as well as remotely
- Sending email notifications to update status
- Parameterised jobs and their need
- User Management - Configuring role-based strategy plugin
- Integrating Jenkins with Git via Poll SCM and web hooks
- CI/CD pipeline creation using Jenkins file
- Setting up master slave architecture in Jenkins

Introduction to CI/CD

Software Development Life Cycle



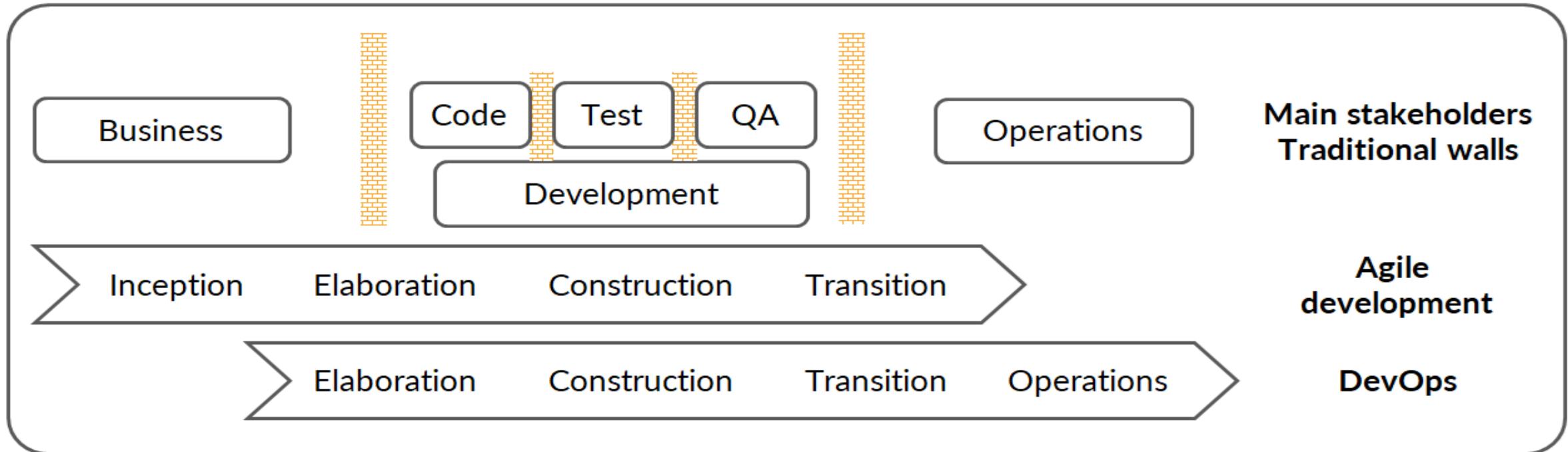
Introduction to CI/CD

- In software engineering, **continuous integration (CI)** is defined as a process of integrating all developers' code to a shared repository in a version-control system frequently.
- The term CI was first proposed by Grady Booch in 1991.
- CI is supposed to be used in conjunction with automated build, test, and QA.
- A build server compiles the code and reports the results to the developers on a regular basis, or even after each commit.
- Test: Unit test in the developer's local environment before committing to the mainline helps avoid one developer's work-in-progress while breaking another developer's copy.

Introduction to CI/CD

- Along with running the unit and integration checks, you can check the quality of your code and profile performance and format the documentation from the source code, thereby facilitating QA processes.
- This quality check application mainly aims to enhance software quality and reduce delivery time continuously.
- CI is connected closely with continuous delivery or continuous deployment, which is also called the CI/CD pipeline.
- ‘Continuous delivery’ ensures that the incremental feature of a software product checked in on the mainline is ready to deliver to the end user.
- ‘Continuous deployment’ makes the deployment process fully automated.

Drawbacks of Traditional SDLC



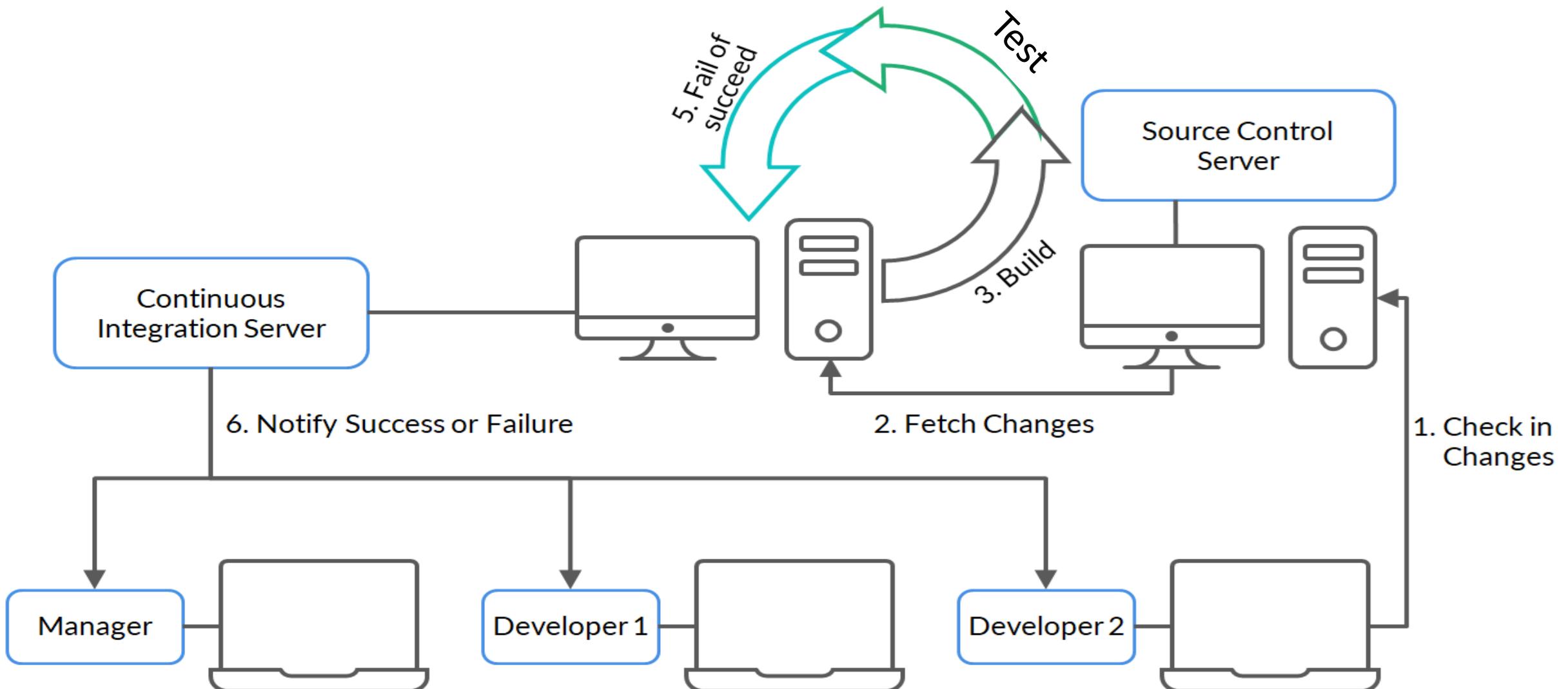
- Agile breaks the wall between Business and Development teams.
- DevOps breaks the wall between Development and Operations teams.
- DevOps centres on the concept of sharing (ideas, issues, processes, tools and goals).

Continuous Integration

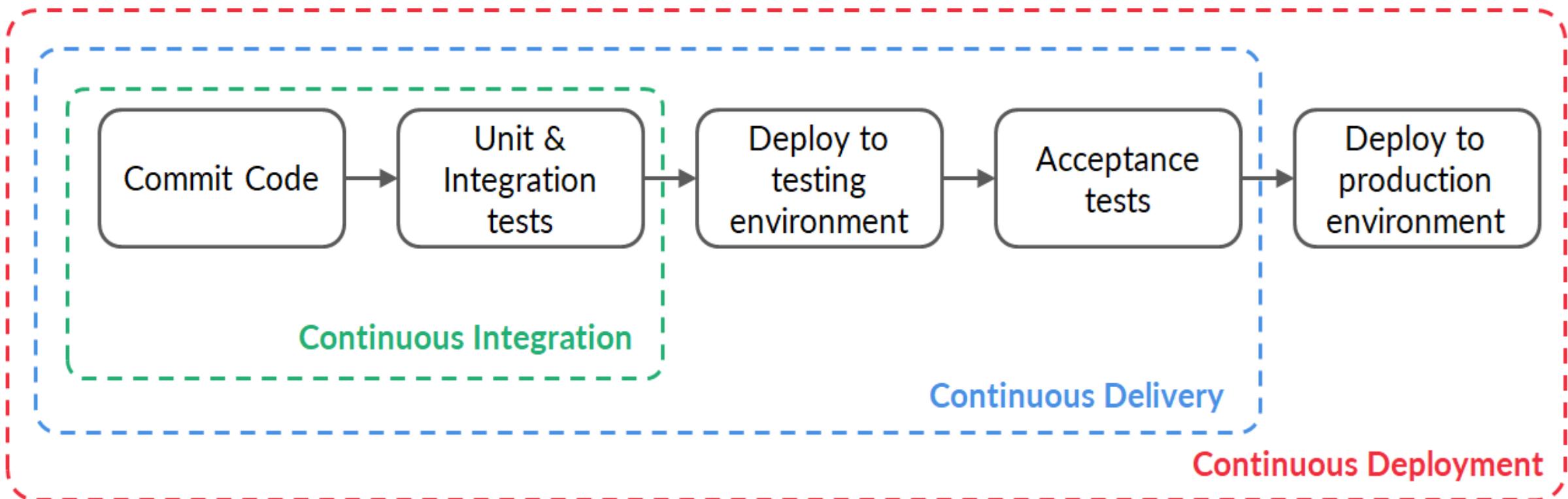
- CI, in its simplest form, involves a tool that continuously monitors version-control systems for any new changes.
- Whenever any new changes are pushed, this tool starts compiling and testing your application.
- If a bug is found or the code fails, the developers are immediately notified to fix the issue.
- If you integrate it with automated end-to-end acceptance checks, CI can serve as a feedback method, offering a straightforward image of the current state of development efforts.
- It will allow you to deploy the latest version of your application either automatically or as a one-click process.

Continuous Integration

CONTINUOUS INTEGRATION



Continuous – Integration, Delivery and Deployment



Benefits of CI/CD

- Automation – ensures build, test, QA, delivery and deployment
- Improves consistencies and quality of code
- Delivers new features quickly to the end user
- Increases product visibility
- Helps avoid manual errors
- Helps fix any issues that may arise
- Reduces costs and labour

Introduction to Jenkins

Jenkins – An Introduction

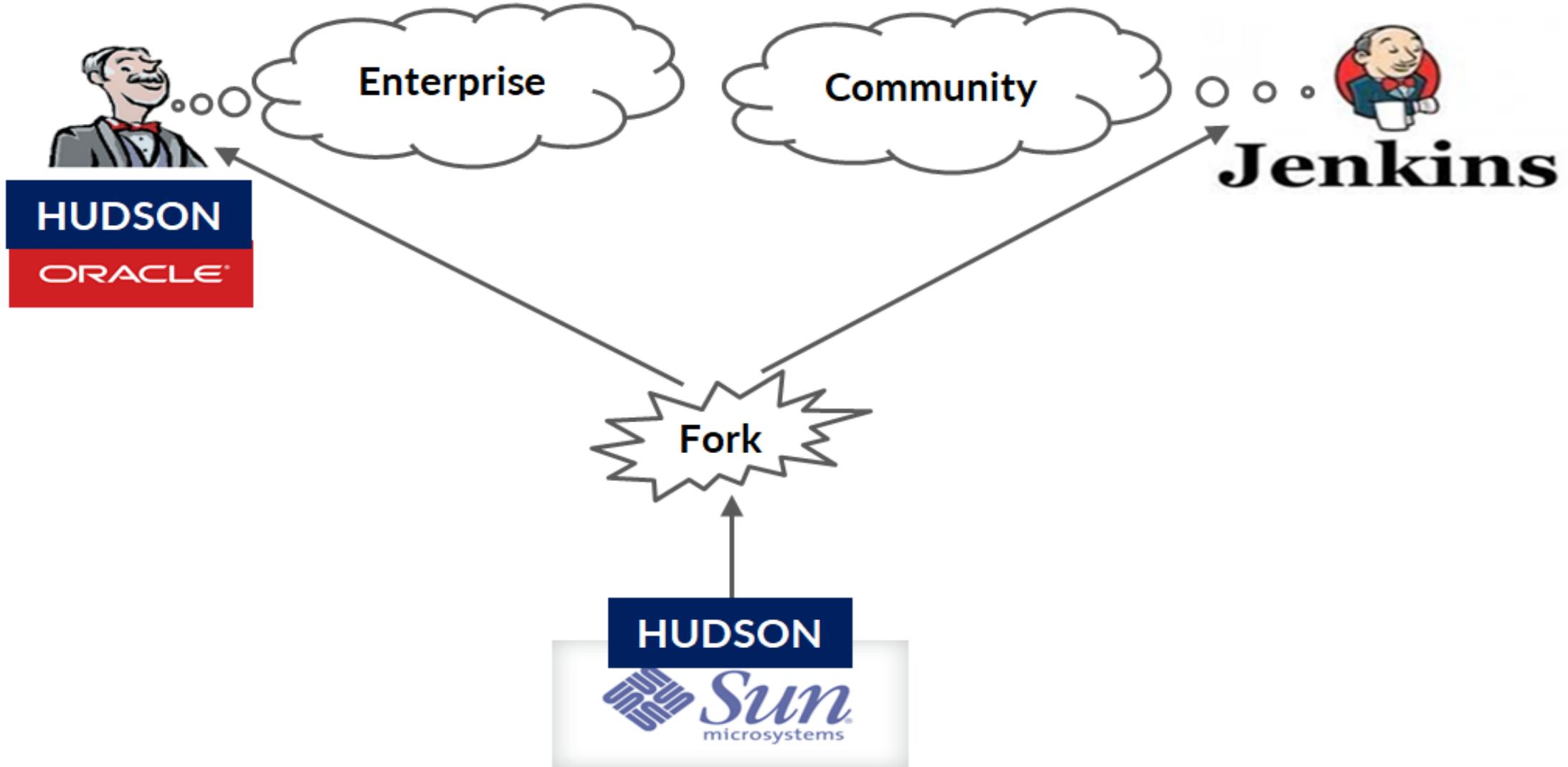


- Jenkins is basically a free and open-source automation server.
- Jenkins can be used to automate all types of software development tasks such as building, testing, and delivering or deploying software.
- Jenkins can be installed via native system packages, Docker, or it can be run standalone on any computer.

How Jenkins Evolved

- Jenkins is the outcome of an innovative developer, Kohsuke Kawaguchi, who began this project as a hobby under the name Hudson in late 2004 while working at Sun Microsystems.
- In 2009, Sun Microsystems was acquired by Oracle. At the end of 2010, tensions arose between the Hudson developer community and Oracle. Initially, the problem was with the Java.net infrastructure, which later worsened due to issues related to Oracle's claim to the Hudson trademark.
- In January 2011, the Hudson developer community decided to rename the project as Jenkins. Subsequently, they migrated the original Hudson code base to a new GitHub project and continued their work there.
- After this, the majority of users joined the Jenkins developer community, migrating to Jenkins.

History of Jenkins



Features of Jenkins

- Jenkins, originally called Hudson, is an open-source CI tool written in Java.
- Jenkins is a very user-friendly tool, with a simple, intuitive and visually appealing user interface.
- Jenkins has a very low learning curve.
- Jenkins is an extremely flexible tool and hundreds of open-source plugins are available, with more coming up every week.

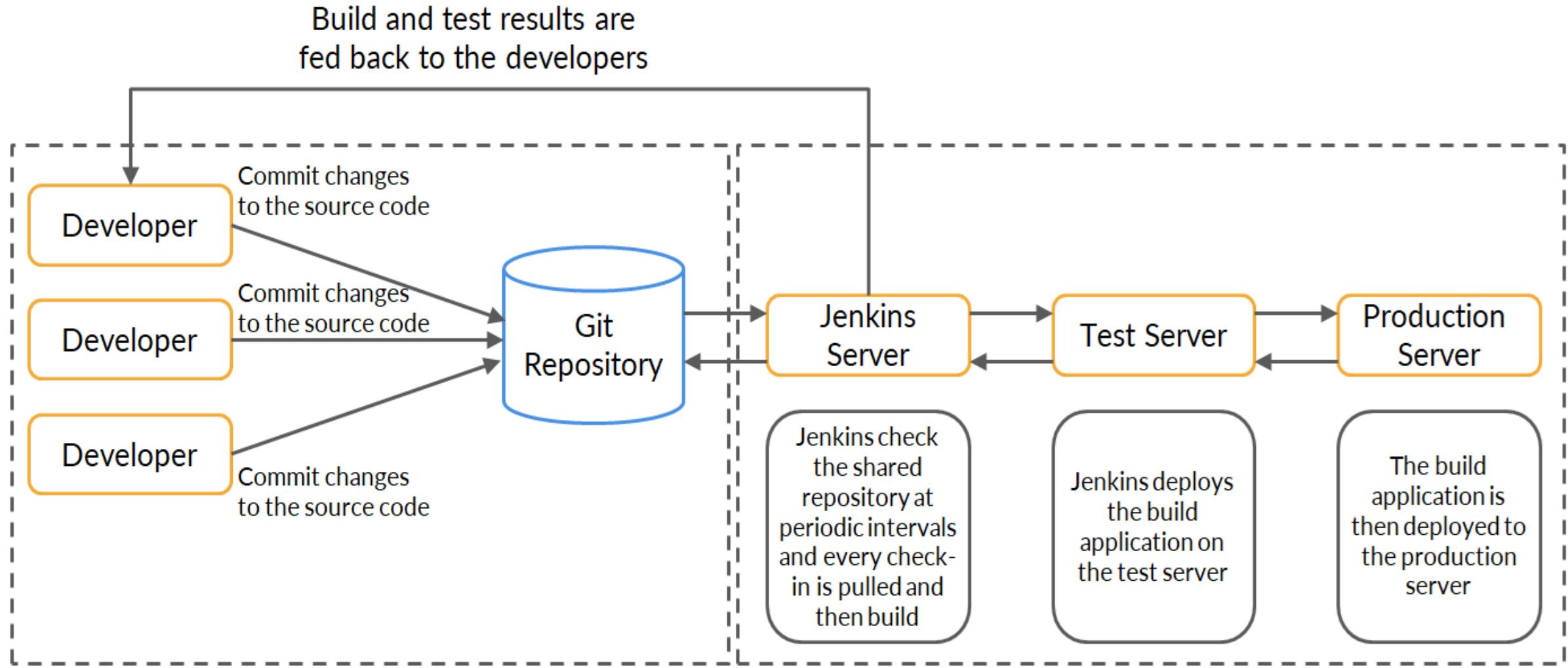
Features of Jenkins

- These plugins cover everything from version-control systems, build tools, code quality metrics and build notifiers to integration with external systems, UI customisations, and much more.
- The Jenkins community is a large, dynamic, reactive and welcoming bunch, with passionate champions, active mailing lists, IRC channels and a very vocal blog and Twitter account.
- The pace of development is quite fast, with the latest new features, bug fixes, and plugin updates being released on a weekly basis.
- Jenkins is a JAVA application, so it is platform independent.

Comparison of CI Tools

	Jenkins	TeamCity	Bamboo	Travis	Circle	Codeship
Pricing	Free	\$299-\$1999	\$10-\$800	\$69-\$489	\$50-\$3150	\$75-\$1500
Operating system	Windows, Linux, macOS, any Unix-like OS	Windows, Linux, macOS, Solaris, FreeBSD and more	Windows, Linux, macOS, Solaris	Linux, macOS	Linux, iOS, Android	Windows, macOS
Hosting	On premise/cloud	On premise	On premise/Bitbucket as cloud	On premise/cloud	Cloud	Cloud
Container support	✓	✓	✓	✓	✓	Yes for Pro version
Plugins	*****	***	**	****	***	****
Docs and support	Adequate	Good	Good	Poor	Good	Poor
Learning curve and usability	Easy	Medium	Medium	Easy	Easy	Easy
Use case	For big Projects	For enterprise needs	For Atlassian integrations	For small projects and startups	For fast development and high budget	For any project

Jenkins Architecture



Companies That Use Jenkins

Companies: 2836 companies reportedly use **Jenkins** in their tech stacks, including Facebook, Netflix, and Udemy

Developers: 33308 developers on StackShare have stated that they use **Jenkins**.



Facebook



Netflix



Udemy



Instacart



Robinhood



Twitch



Lyft



Delivery Hero



LinkedIn

Slack, Datadog, BrowserStack, Azure DevOps, and SonarQube are some of the popular **144 tools that integrate with Jenkins**.



Slack



Datadog



BrowserStack



Azure DevOps



SonarQube



Rancher



AWS CodePipeline



Mattermost



AWS CodeCommit

Configuring and Setting Up Jenkins

Jenkins Installation

Install Java

```
sudo apt-get update
```

```
sudo apt install -y openjdk-11-jdk
```

To check: java --version

```
openjdk version "11.0.11" 2021-04-20
```

```
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.18.04)
```

```
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.18.04, mixed mode, sharing)
```

Install Jenkins

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

```
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

Install ca-certificates (man 8 update-ca-certificates)

```
sudo apt install ca-certificates
```

Install Jenkins

```
sudo apt-get update
```

```
sudo apt-get install jenkins
```

To check Jenkins version: vim /var/lib/jenkins/config.xml

To copy admin password: sudo cat /var/lib/jenkins/secrets/initialAdminPassword



Execute https://<ipaddress>:8080

The screenshot shows a web browser window with the following details:

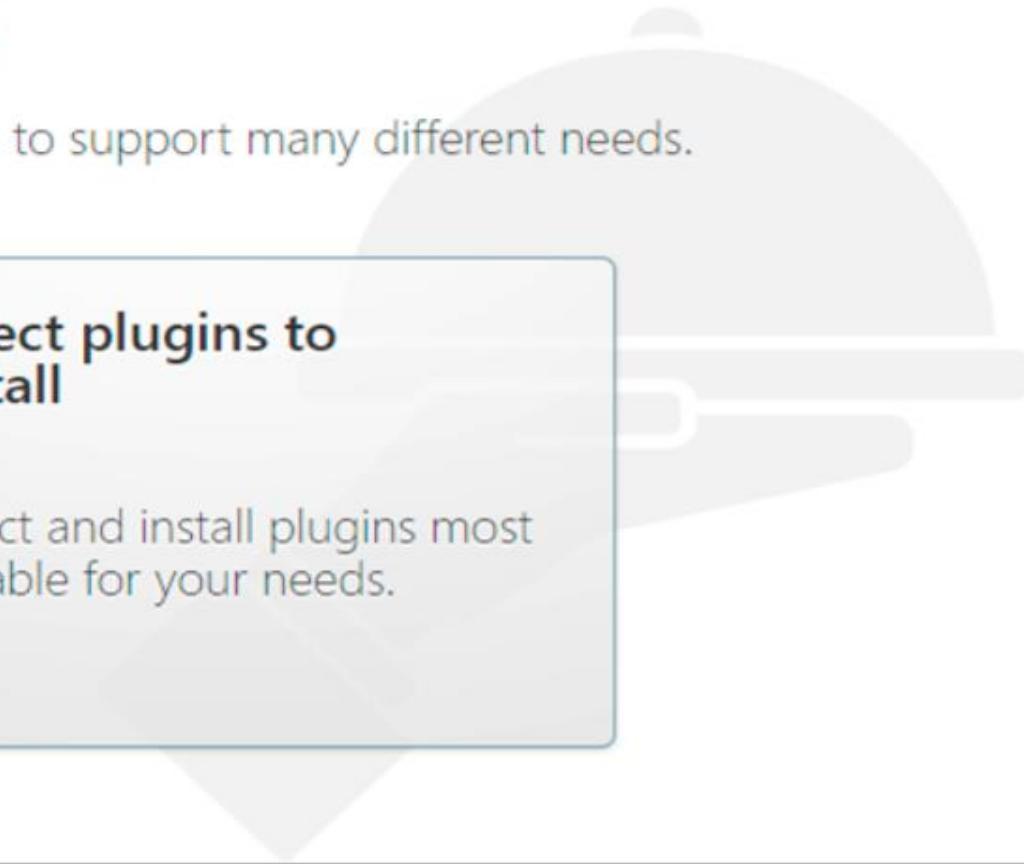
- Address Bar:** Shows the URL `3.95.168.69:8080/login?from=%2F` with a "Not secure" warning icon.
- Page Title:** "Getting Started"
- Section Header:** "Unlock Jenkins"
- Text Content:** "To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:"
- Text Example:** `/var/lib/jenkins/secrets/initialAdminPassword`
- Text Instruction:** "Please copy the password from either location and paste it below."
- Form Label:** "Administrator password" with a text input field.
- Text Output:** A yellow-highlighted area containing the command `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` followed by the password **5bd8dce3437a4a80886ba00389e2d16f**.
- Text Note:** "Copy and paste as Administrator password"
- Buttons:** A large "Continue" button at the bottom right.

Installation of Default Plugins

Getting Started X

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.

Install Suggested Plugins

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** SSH Credentials Credentials Binding ** SCM API ** Pipeline: API Timestamper ** Script Security ** Plugin Utilities API ** Font Awesome API ** Popper.js API ** JQuery3 API ** Bootstrap 4 API ** Snakeyaml API ** Jackson 2 API ** ECharts API ** Display URL API ** Pipeline: Supporting APIs ** Pipeline: Job ** Checks API ** JUnit ** Matrix Project ** - required dependency
✓ Timestamper	↻ Workspace Cleanup	↻ Ant	↻ Gradle	
↻ Pipeline	↻ GitHub Branch Source	↻ Pipeline: GitHub Groovy Libraries	↻ Pipeline: Stage View	
↻ Git	↻ SSH Build Agents	○ Matrix Authorization Strategy	↻ PAM Authentication	
↻ LDAP	↻ Email Extension	↻ Mailer		

Continue as Admin

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.263.4

[Skip and continue as admin](#)

[Save and Continue](#)

[Create User Accounts Later](#)

Save Jenkins Configuration

Instance Configuration

Jenkins URL:

`http://3.95.168.69:8080/`

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.

Start Jenkins

Getting Started

Jenkins is ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins Dashboard

The screenshot shows the Jenkins dashboard interface. At the top, there is a navigation bar with a search field, a help icon, and user account information for 'admin'.

The main content area features a large 'Welcome to Jenkins!' message. Below it, there are several calls-to-action:

- Start building your software project**: A button labeled 'Create a job' with an arrow icon.
- Set up a distributed build**: Buttons for 'Set up an agent' and 'Configure a cloud', each with an arrow icon.
- A link to 'Learn more about distributed builds' with a 'GO' button.

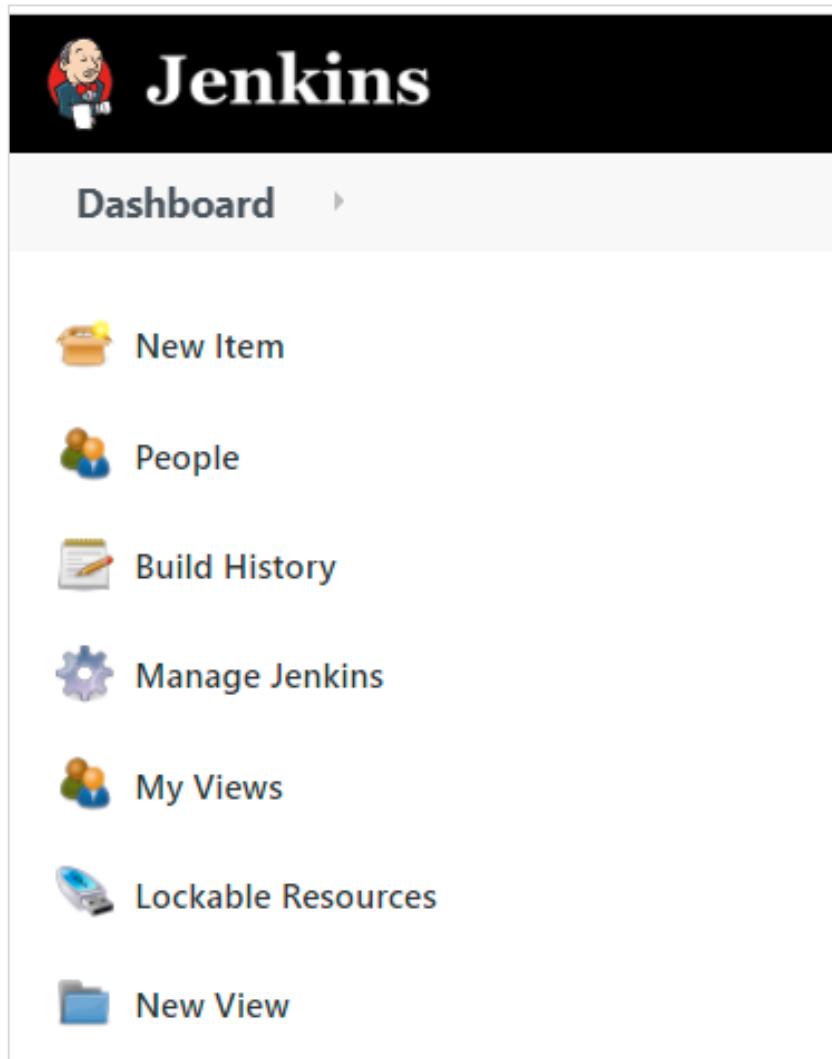
On the left side, there is a sidebar with the following links:

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Below the sidebar, there are two expandable sections:

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

Dashboard – Various Options



The image shows the Jenkins dashboard sidebar. At the top is a black header bar with the Jenkins logo (a cartoon character) and the word "Jenkins" in white. Below the header is a light gray sidebar containing the following options:

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Select New Item

Enter an item name

Project1
» 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.

OK

Manage Jenkins – Various Options

Jenkins

Dashboard >

- New Item
- People
- Build History
- Manage Jenkins**
- My Views
- Lockable Resources
- New View

Build Queue ^

No builds in the queue.

Build Executor Status ^

1 Idle

Manage Jenkins

System Configuration

-  **Configure System**
Configure global settings and paths.
-  **Global Tool Configuration**
Configure tools, their locations and automatic installers.
-  **Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
-  **Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

-  **Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
-  **Manage Credentials**
Configure credentials
-  **Configure Credential Providers**
Configure the credential providers and types
-  **Manage Users**
Create/delete/modify users that can log in to this Jenkins

Jenkins Plugin Manager – Advanced

The screenshot shows the Jenkins Plugin Manager - Advanced interface. At the top, there's a navigation bar with links for Dashboard, Plugin Manager, Back to Dashboard, Manage Jenkins, and Update Center. The main content area has tabs for Updates, Available, Installed, and Advanced, with Advanced selected. The first section, "HTTP Proxy Configuration", contains fields for Server, Port, User name, Password, and No Proxy Host, each with a help icon. Below this is a "Submit" button and an "Advanced..." link. The second section, "Upload Plugin", allows users to upload a .jpi file from outside the central plugin repository. It includes a "Choose File" button and an "Upload" button. The third section, "Update Site", lets users specify a URL for the update center, with the default value being <https://updates.jenkins.io/update-center.json>. A "Submit" button is also present here.

HTTP Proxy Configuration

Server: [Input field]

Port: [Input field]

User name: [Input field]

Password: [Input field]

No Proxy Host: [Input field]

Upload Plugin

You can upload an .jpi file to install a plugin from outside the central plugin repository.

File: Choose File No file chosen

Update Site

URL:

Understand How to Create Jobs in Jenkins

Start Jenkins

```
ubuntu@ip-172-31-81-117:~$ sudo service jenkins start
ubuntu@ip-172-31-81-117:~$ sudo service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
  Active: active (exited) since Tue 2021-03-09 01:13:37 UTC; 2min 52s ago
    Docs: man:systemd-sysv-generator(8)
   Tasks: 0 (limit: 1140)
  CGroup: /system.slice/jenkins.service

Mar 09 01:13:34 ip-172-31-81-117 systemd[1]: Starting LSB: Start Jenkins at boot time...
Mar 09 01:13:35 ip-172-31-81-117 jenkins[842]: Correct java version found
Mar 09 01:13:35 ip-172-31-81-117 jenkins[842]: * Starting Jenkins Automation Server jenkins
Mar 09 01:13:35 ip-172-31-81-117 su[943]: Successful su for jenkins by root
Mar 09 01:13:35 ip-172-31-81-117 su[943]: + ??? root:jenkins
Mar 09 01:13:36 ip-172-31-81-117 su[943]: pam_unix(su:session): session opened for user jenkins by (uid=0)
Mar 09 01:13:36 ip-172-31-81-117 su[943]: pam_unix(su:session): session closed for user jenkins
Mar 09 01:13:37 ip-172-31-81-117 jenkins[842]:     ...done.
Mar 09 01:13:37 ip-172-31-81-117 systemd[1]: Started LSB: Start Jenkins at boot time.
ubuntu@ip-172-31-81-117:~$ █
```

1. Know the status of **Jenkins**: `sudo service jenkins status`.
2. To **start Jenkins**: `sudo service jenkins start`.
3. To **stop Jenkins**: `sudo service jenkins stop`.
4. To **restart Jenkins**: `sudo service jenkins restart`. (stop and start)

Jenkins Workspace

Jenkins home directory:

```
ubuntu@ip-172-31-81-117:~$ sudo su - jenkins  
jenkins@ip-172-31-81-117:~$ pwd  
/var/lib/jenkins  
jenkins@ip-172-31-81-117:~$
```

Jenkins home directory: /var/lib/jenkins

It contains the details of the Jenkins server configuration, which is used to configure in the Manage Jenkins. The core configuration files are stored in config.xml.

It contains two important subdirectories: **jobs and workspace**.

Jobs directory contains configuration details of the build job in config.xml file.

Jenkins builds the given project in the workspace directory. Each project has its own directory in the workspace.

Click to Create a Job

The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with the following items:

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources

The main content area has a title "Welcome to Jenkins!" and a subtitle "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this, there is a button labeled "Start building your software project". A red arrow points from the text "Click to Create a Job" in the top banner to this button. At the bottom right of the button is a small "→" icon.

Freestyle Project

Enter an item name

HelloWorld Python Program

» 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.

OK

Example: HelloWorld Python Program

The screenshot shows the Jenkins dashboard with the following details:

- Dashboard:** The main navigation bar includes links for "New Item", "People", "Build History", "Manage Jenkins", and "My Views".
- User Information:** Shows "admin" logged in.
- Search Bar:** A search field with placeholder "search" and a help icon.
- View Details:** The "HelloWorld Python Program" view is displayed. It includes:
 - A status summary: S (Stable), W (Warning).
 - The view name: HelloWorld Python Program.
 - Build status: N/A (Last Success, Last Failure, Last Duration).
 - Icon selection: Icon: S M L (Small, Medium, Large).
 - Feed options: Atom feed for all, Atom feed for failures, Atom feed for just latest builds.

```
#!/usr/bin/python3
# This Python program will print Hello World...
print("Hello World ...\\n")
```

Build the Project

Jenkins

Dashboard > HelloWorld Python Program >

[Back to Dashboard](#)

Status

Changes

Workspace

Build Now

Configure

Project HelloWorld Python Program

HelloWorld Python Script

Workspace

Recent Changes



Console Output

Jenkins

Dashboard ▶ HelloWorld Python Program ▶ #15

Back to Project
 Status
 Changes
 Console Output
 View as plain text
 Edit Build Information

Console Output

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld Python Program
[HelloWorld Python Program] \$ /bin/sh -xe /tmp/jenkins9071003953682465907.sh
+ ./HelloWorld.py

Hello World...

Finished: SUCCESS

Check Log Files

```
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds$ ls  
1 10 2 3 4 5 6 7 8 9 legacyIds permalinks  
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds$ cd 4  
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds/4$ ls  
build.xml changelog.xml log  
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds/4$ cat log  
Started by user ha:///401Xwh1EUGVcmahxd0uDY/VpeIbwK+Jgej0qfQuwtGn1AAAAAx+LCAAAAAAAP9b85aBtbIQTGjNKU4P08vOT+v  
1x6OyILUoJzMv2y+/JJUBAhiZGBgqihh0NSjKDwzXb3Rd1LBUSYGJk8GtpzUvPSSDB8G5tKinBIGIZ+sxLJE/ZzEvHT94JKizLx0a6BxUmjGOU  
gZu/dLi1CL9xJTczDwAj6GcLcAAAAA=admin  
Running as SYSTEM  
Building in workspace /var/lib/jenkins/workspace/HelloWorld Python Program  
[HelloWorld Python Program] $ /bin/sh -xe /tmp/jenkins10839759464246091234.sh  
+ echo Hello World  
Hello World  
Finished: SUCCESS  
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds/4$ █
```

Custom Workspace

Dashboard > HelloWorld Python Program >

General	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions
<input type="checkbox"/> Execute concurrent builds if necessary					?
<input type="checkbox"/> Quiet period					?
<input type="checkbox"/> Retry Count					?
<input type="checkbox"/> Block build when upstream project is building					?
<input type="checkbox"/> Block build when downstream project is building					?
<input checked="" type="checkbox"/> Use custom workspace					?
Directory	/home/ubuntu				
Display Name					

```
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds$ ls
1 10 2 3 4 5 6 7 8 9 legacyIds permalinks
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds$ cd 9
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds/9$ cat log
Started by user ha:///4O1Xwh1EUGVcmahxd0uDY/VpeIbwK+Jgej0qfQuwtGnlAAAAAx+LCAAAAAAP9b85aBtbiiQTGj
1x6OyILUoJzMv2y+/JJUBAhizGBgqihhk0NSjKDwzXb3Rd1LBUSYGJk8GtpzUvPSSDB8G5tKinBIGIZ+sxLJE/ZzEvHT94JKiziL
gZu/dLi1CL9xJTczDwAj6GcLcAAAAA=admin
Running as SYSTEM
Building in workspace /home/ubuntu
[ubuntu] $ /bin/sh -xe /tmp/jenkins2121288571684403099.sh
+ ./HelloWorld.py

Hello World...

Finished: SUCCESS
ubuntu@ip-172-31-81-117:/var/lib/jenkins/jobs/HelloWorld Python Program/builds/9$
```

Custom Workspace: Console Output

 Back to Project

 Status

 Changes

 Console Output

 View as plain text

 Edit Build Information

 Delete build '#13'

 Previous Build



Console Output

Started by user admin

Running as SYSTEM

Building in workspace /home/ubuntu

```
[ubuntu] $ /bin/sh -xe /tmp/jenkins14930569387565051667.sh  
+ ./HelloWorld.py
```

Hellow World...

Finished: SUCCESS

Build Jenkins Job Remotely

Step 1: Choose the project.

Step 2: Configure – Build – enable Trigger builds remotely.

Step 3: Modify: JENKINS_URL/job/test1/build?token=TOKEN_NAME

<http://3.95.7.132:8080/job/test1/build?token=86470>.

Step 4: Open a browser and enter: <http://3.95.7.132:8080/job/test1/build?token=86470>

Step 5: Verify the build number.

Build Triggers

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

Authentication Token

Use the following URL to trigger build remotely: JENKINS_URL/job/test1/build?token=TOKEN_NAME or
/buildWithParameters?token=TOKEN_NAME

Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

Build after other projects are built ?

Build periodically ?

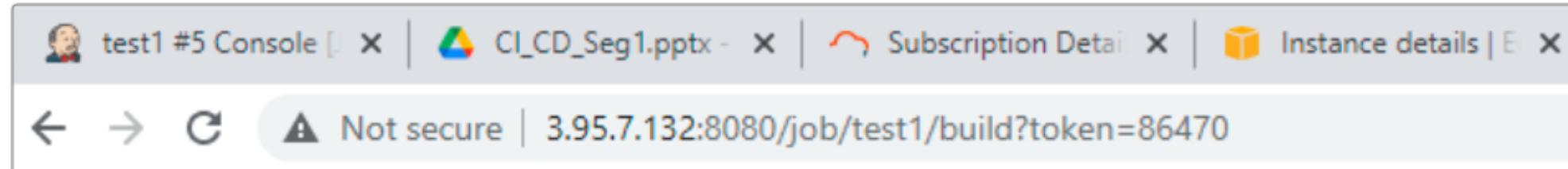
GitHub hook trigger for GITScm polling ?

Poll SCM ?

Save **Apply**

Trigger Build Job by URL

Open a browser, type the URL and enter



Check Build History in the Jenkins Server

A screenshot of the Jenkins 'Build History' page for the job 'test1'. The page has a header with a sun icon and the text 'Build History' and 'trend'. Below the header is a search bar containing the word 'find'. Underneath the search bar is a table with one row. The row contains a grey circular icon, the build number '#5', and a red 'X' button. Below the table, a note says '(pending—In the quiet period. Expires in 1.9 sec)'. The entire table row is highlighted with a red border.

Build by Remote Host – Check Console Output

Manual Build



Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test1
[test1] $ /bin/sh -xe /tmp/jenkins17988033407760365594.sh
+ cd ../../..
+ echo The total number of directories in the /var/lib/jenkins
The total number of directories in the /var/lib/jenkins
+ wc -l
+ grep ^d
+ ls -Rl
676
Finished: SUCCESS
```

Trigger Build Remotely



Console Output

```
Started by remote host 122.181.192.30
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test1
[test1] $ /bin/sh -xe /tmp/jenkins1040342982316183531.sh
+ cd ../../..
+ echo The total number of directories in the /var/lib/jenkins
The total number of directories in the /var/lib/jenkins
+ wc -l
+ grep ^d
+ ls -Rl
678
Finished: SUCCESS
```

Job Chaining in Jenkins

- Create 3 Jobs:
 - Job1: CPU INFORMATION
 - Job2: RAM INFORMATION
 - Job3: DISK INFORMATION
- Job2 will start after Job1 build is successful
- Job3 will start after job2 build is successful

Job Chaining in Jenkins – Job1 Configuration

The screenshot shows the Jenkins job configuration for 'CPU INFORMATION'. The 'Source Code Management' tab is selected, showing options for 'None' (selected) and 'Git'. The 'Build Triggers' tab is also visible. A list of triggers is present, each with a question mark icon for help:

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

The screenshot shows the Jenkins job configuration for 'CPU INFORMATION'. The 'Build' tab is selected. Under the 'Execute shell' section, the command is defined as:

```
echo "CPU INFORMATION"  
lscpu
```

Job Chaining in Jenkins – Job2 Configuration

The screenshot shows the configuration page for a Jenkins job named "RAM INFORMATION". The top navigation bar includes "Dashboard", "RAM INFORMATION", and tabs for "General", "Source Code Management", "Build Triggers" (which is selected), "Build Environment", "Build", and "Post-build Actions".

Build Triggers:

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?

Projects to watch: CPU INFORMATION.

Trigger options:

- Trigger only if build is stable
- Trigger even if the build is unstable
- Trigger even if the build fails

Build:

Execute shell

Command:

```
echo "Physical Memory Information"  
free -m
```

Job Chaining in Jenkins – Job3 Configuration

Dashboard > DISK INFORMATION >

General Source Code Management Build Triggers

Git

Build Triggers

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

Build after other projects are built

Projects to watch RAM INFORMATION,

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Build periodically

GitHub hook trigger for GITScm polling

Poll SCM

Dashboard > DISK INFORMATION >

General Source Code Management Build Triggers

Build

Execute shell

Command

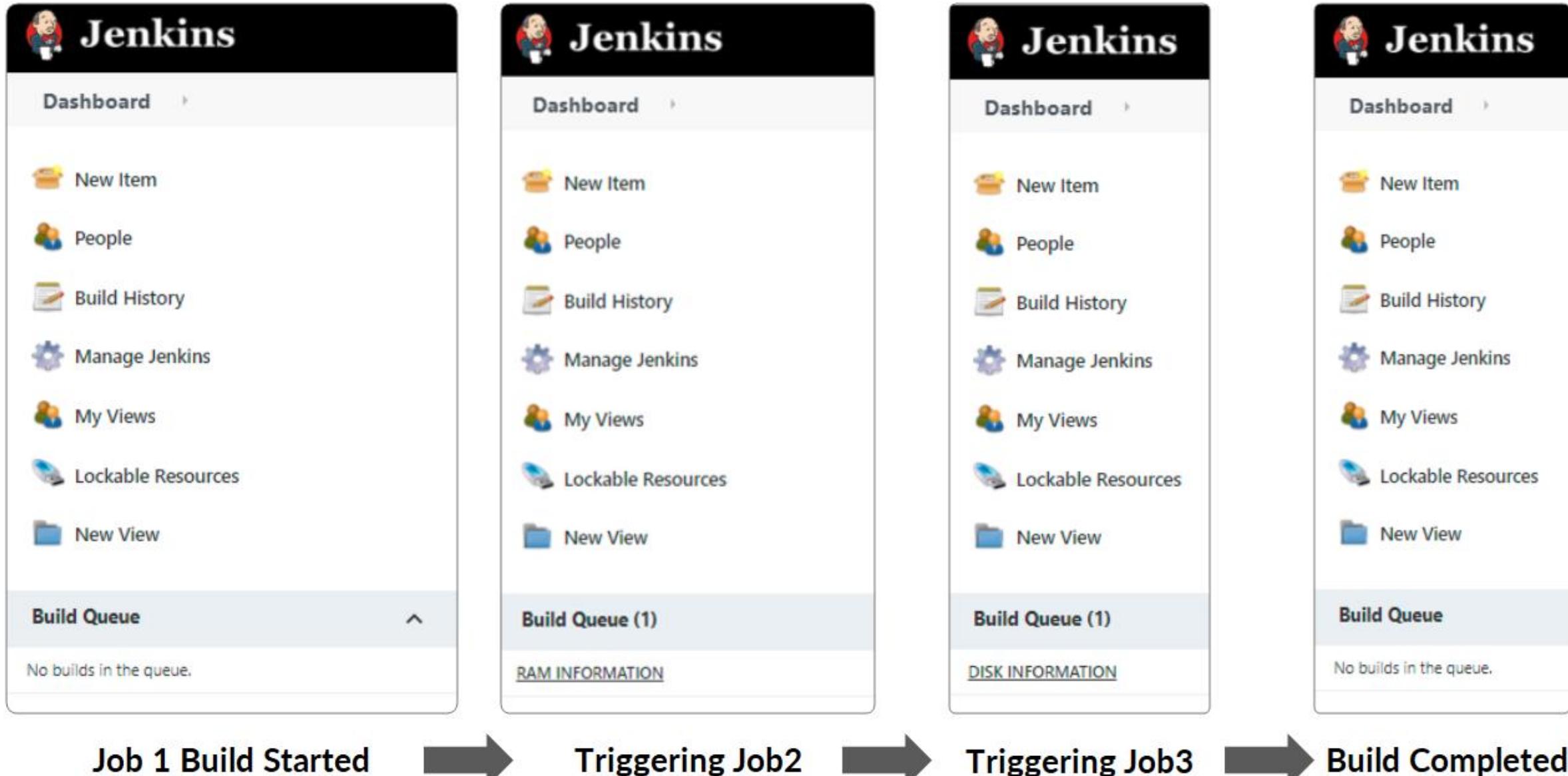
```
echo "DISK INFORMATION"  
df -h
```

Build Job1 – CPU INFORMATION

S	W	Name ↓
		CPU INFORMATION
		DISK INFORMATION
		HelloWorld Python Program
		RAM INFORMATION
		test1

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		CPU INFORMATION	20 min - #1	N/A	18 ms	
			20 min - #1	N/A	31 ms	
		ram	2 hr 6 min - #15	4 hr 4 min - #8	43 ms	
			20 min - #1	N/A	26 ms	

Monitor Job Chaining Builds



CPU INFORMATION – Console Output

Dashboard > CPU INFORMATION > #1

[Back to Project](#)

[Status](#)

[Changes](#)

Console Output

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#1'](#)

Console Output

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/CPU INFORMATION
[CPU INFORMATION] \$ /bin/sh -xe /tmp/jenkins4789611751145793460.sh
+ echo CPU INFORMATION
CPU INFORMATION
+ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 1
On-line CPU(s) list: 0
Thread(s) per core: 1

invpcid xsaveopt
Triggering a new build of RAM INFORMATION
Finished: SUCCESS

RAM INFORMATION – Console Output

Dashboard > RAM INFORMATION > #1

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#1'



Console Output

Started by upstream project "CPU INFORMATION" build number 1
originally caused by:

Started by user admin

Running as SYSTEM

Building in workspace /var/lib/jenkins/workspace/RAM INFORMATION
[RAM INFORMATION] \$ /bin/sh -xe /tmp/jenkins16378467277877437643.sh
+ echo Physical Memory Information

Physical Memory Information

+ free -m

	total	used	free	shared	buff/cache	available
Mem:	978	565	84	0	328	273
Swap:	0	0	0			

Triggering a new build of DISK INFORMATION

Finished: SUCCESS

DISK INFORMATION – Console Output

Dashboard > DISK INFORMATION > #1

 Back to Project
 Status
 Changes
 Console Output
View as plain text
 Edit Build Information
 Delete build '#1'

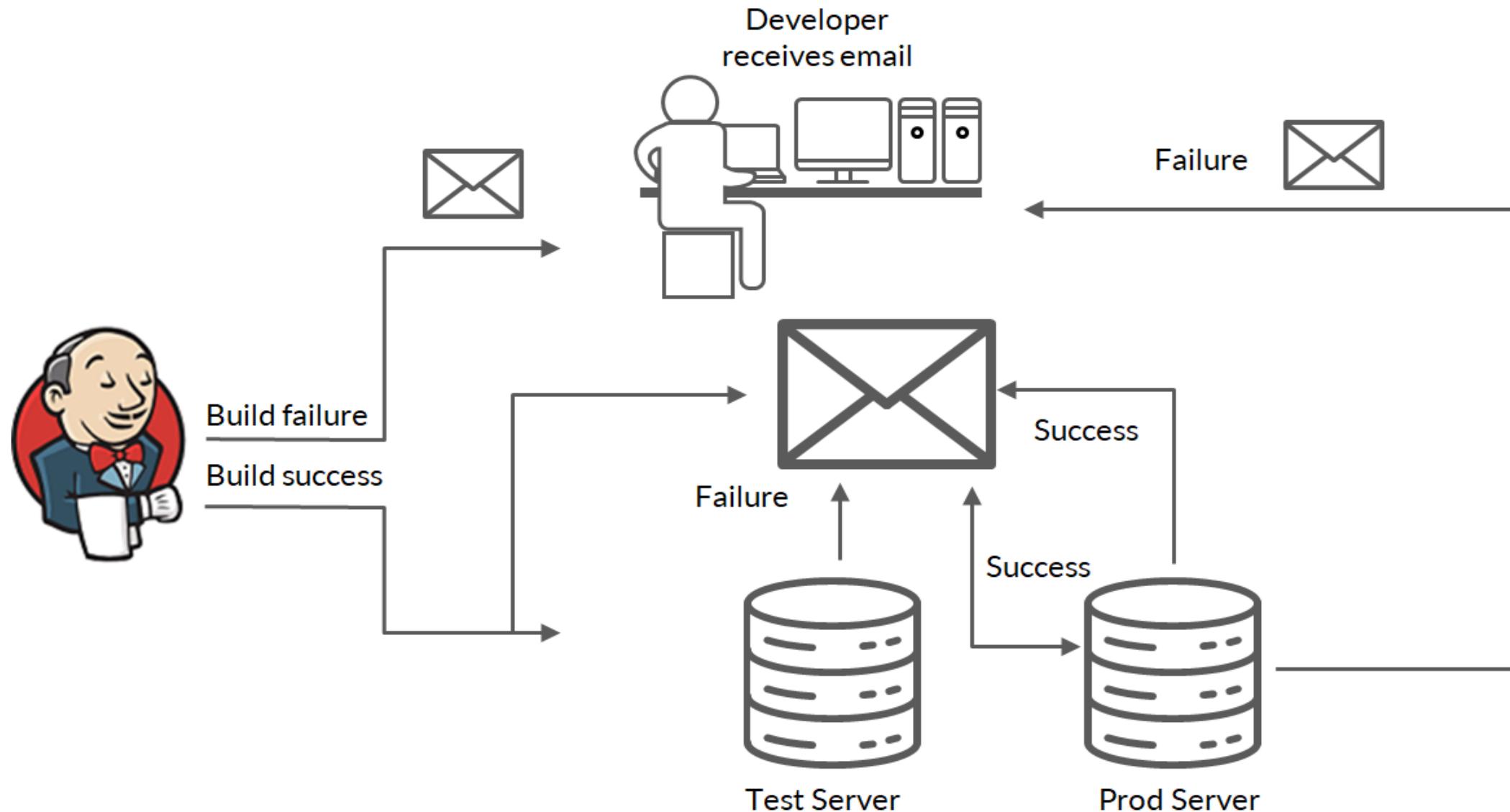
Console Output

Started by upstream project "RAM INFORMATION" build number 1
originally caused by:
Started by upstream project "CPU INFORMATION" build number 1
originally caused by:
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/DISK INFORMATION
[DISK INFORMATION] \$ /bin/sh -xe /tmp/jenkins11158508944207481281.sh
+ echo DISK INFORMATION
DISK INFORMATION
+ df -h

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	476M	0	476M	0%	/dev
tmpfs	98M	784K	98M	1%	/run
/dev/xvda1	7.7G	2.3G	5.5G	30%	/
tmpfs	490M	0	490M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	490M	0	490M	0%	/sys/fs/cgroup
/dev/loop0	32M	32M	0	100%	/snap/snapd/11036
/dev/loop1	34M	34M	0	100%	/snap/amazon-ssm-agent/3552
/dev/loop2	56M	56M	0	100%	/snap/core18/1988
tmpfs	98M	0	98M	0%	/run/user/111
/dev/loop3	33M	33M	0	100%	/snap/snapd/11107
tmpfs	98M	0	98M	0%	/run/user/1000

Finished: SUCCESS

Email Notification



Jenkins Email Notification Configuration

- If build failed or the build status changed from failed to success, then Jenkins has a provision to send an email notification to the given recipients.
- Install **Email Extension Plugin** and **Mailer Plugin**



Email Configuration Setup

Jenkins

search

admin log out

Dashboard > Plugin Manager

Back to Dashboard

Manage Jenkins

Update Center

email

Updates Available Installed Advanced

Enabled	Name	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	Email Extension Plugin	2.82		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Mailer Plugin	1.32.1		<button>Uninstall</button>

This image shows the Jenkins Plugin Manager interface. A search bar at the top contains the text "email". Below it, tabs for "Updates", "Available", "Installed", and "Advanced" are visible, with "Installed" being the active tab. A table lists two installed plugins: "Email Extension Plugin" and "Mailer Plugin". Both plugins have a checked checkbox column, indicating they are enabled. The "Email Extension Plugin" is version 2.82, and the "Mailer Plugin" is version 1.32.1. Each plugin row includes a "Uninstall" button.

email configuration setup



Manage Jenkins



Configure System
Configure global settings and paths.

Email Configuration Setup

Jenkins Location

Jenkins URL

http://3.95.7.132:8080/

System Admin e-mail address

Jenkins-Master <drbalat.raju@gmail.com>

Extended E-mail Notification

SMTP server

smtp.gmail.com

SMTP Port

465

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix



Test configuration by sending test e-mail

Save

Apply

Email Configuration Setup

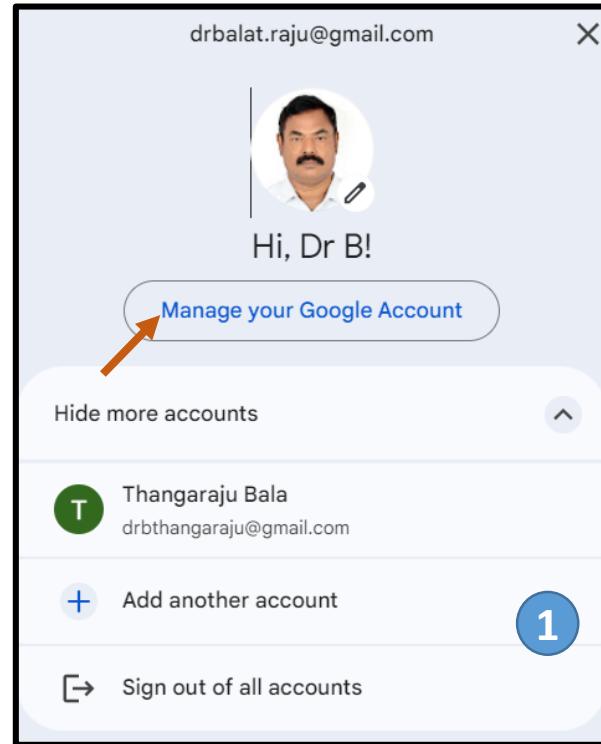
E-mail Notification

SMTP server	<input type="text" value="smtp.gmail.com"/>	?
Default user e-mail suffix	<input type="text"/>	?
<input checked="" type="checkbox"/> Use SMTP Authentication		?
User Name	<input type="text" value="drbalat.raju@gmail.com"/>	
Password	<input type="password" value="Concealed"/> Change Password	
Use SSL	<input checked="" type="checkbox"/>	?
Use TLS	<input type="checkbox"/>	?
SMTP Port	<input type="text" value="465"/>	?
Reply-To Address	<input type="text"/>	
Charset	<input type="text" value="UTF-8"/>	
<input checked="" type="checkbox"/> Test configuration by sending test e-mail		
Test e-mail recipient	<input type="text" value="drbthangaraju@gmail.com"/>	Test configuration
Save	Apply	

Annotations:

- An orange arrow points to the "Use SSL" checkbox, which is checked.
- A text message "Password is copied from google app password." is displayed below the password field, with an orange arrow pointing to the checked "Use SSL" checkbox.
- An orange arrow points to the "Test configuration" button at the bottom right.

Configure Jenkins email notification with gmail



This screenshot shows the "Security" settings page. It features a "Security tips" section with a green shield icon containing a checkmark. A blue circle with the number "3" is placed next to the shield icon. The main menu on the left is identical to the previous screen: Home, Personal info, Data & privacy, Security (selected), and People & sharing. The right side has a heading "Security" and a sub-section titled "You have security tips".

This screenshot shows the "2-Step Verification" section. It includes a "Devices you trust" section with a "REVOKE ALL" button and a "Devices that don't need a second step" section. A blue circle with the number "4" is placed near the "REVOKE ALL" button. The right side also contains sections for "App passwords" and "App passwords" (with a count of "1 password"). An orange arrow points from the bottom right towards the "REVOKE ALL" button.

This screenshot shows the main "Google Account" screen. It features a large profile picture and the greeting "Welcome, Dr B Thangaraju". The main menu on the left is: Home (selected and highlighted in blue), Personal info, Data & privacy, Security (highlighted with a red arrow), and People & sharing. A blue circle with the number "2" is placed near the "Welcome" message. The right side has a "Manage your info, privacy, and security" section with a "Learn more" link.

Configure Jenkins email notification with gmail

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Name	Created	Last used
Mail on my Windows Computer	1:09 PM	—

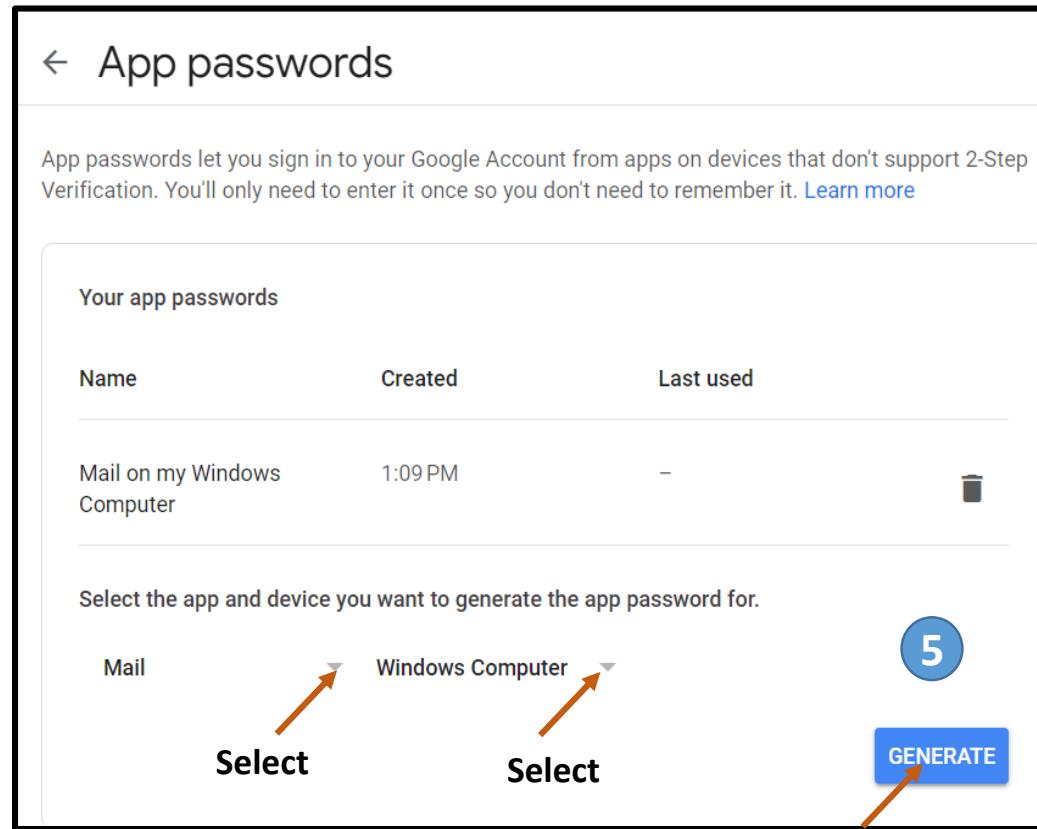
Select the app and device you want to generate the app password for.

Mail Windows Computer

Select **Select**

5

GENERATE



Generated app password

Your app password for Windows Computer

pnmr alwy wtyc gcxa

How to use it

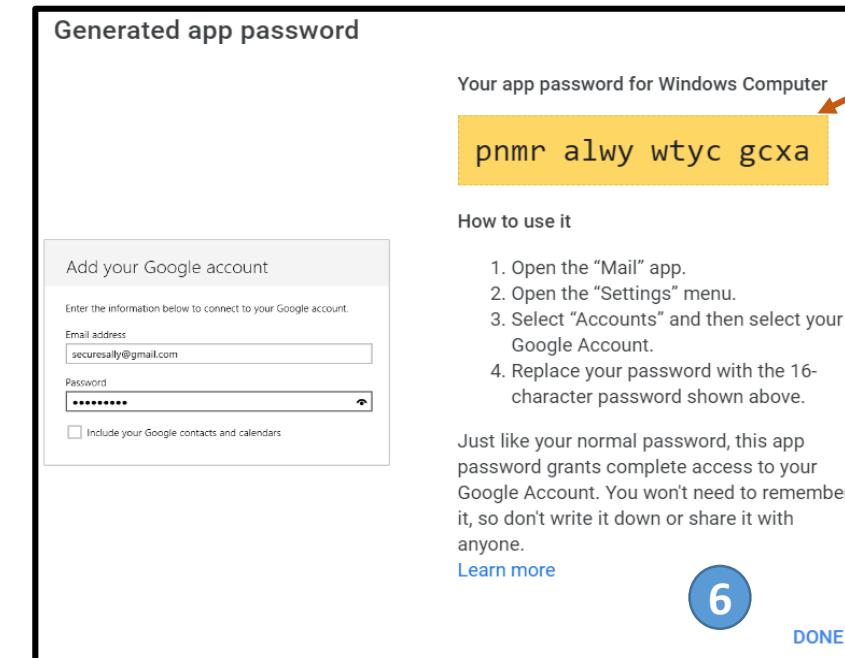
1. Open the "Mail" app.
2. Open the "Settings" menu.
3. Select "Accounts" and then select your Google Account.
4. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

[Learn more](#)

6

DONE



Copy and past it in Jenkins email configuration setup



Post-Build Actions for a Project

Go to Dashboard
Select Project -TestEmail
Click Configure

Post-build Actions

E-mail Notification

Recipients drbthangaraju@gmail.com 

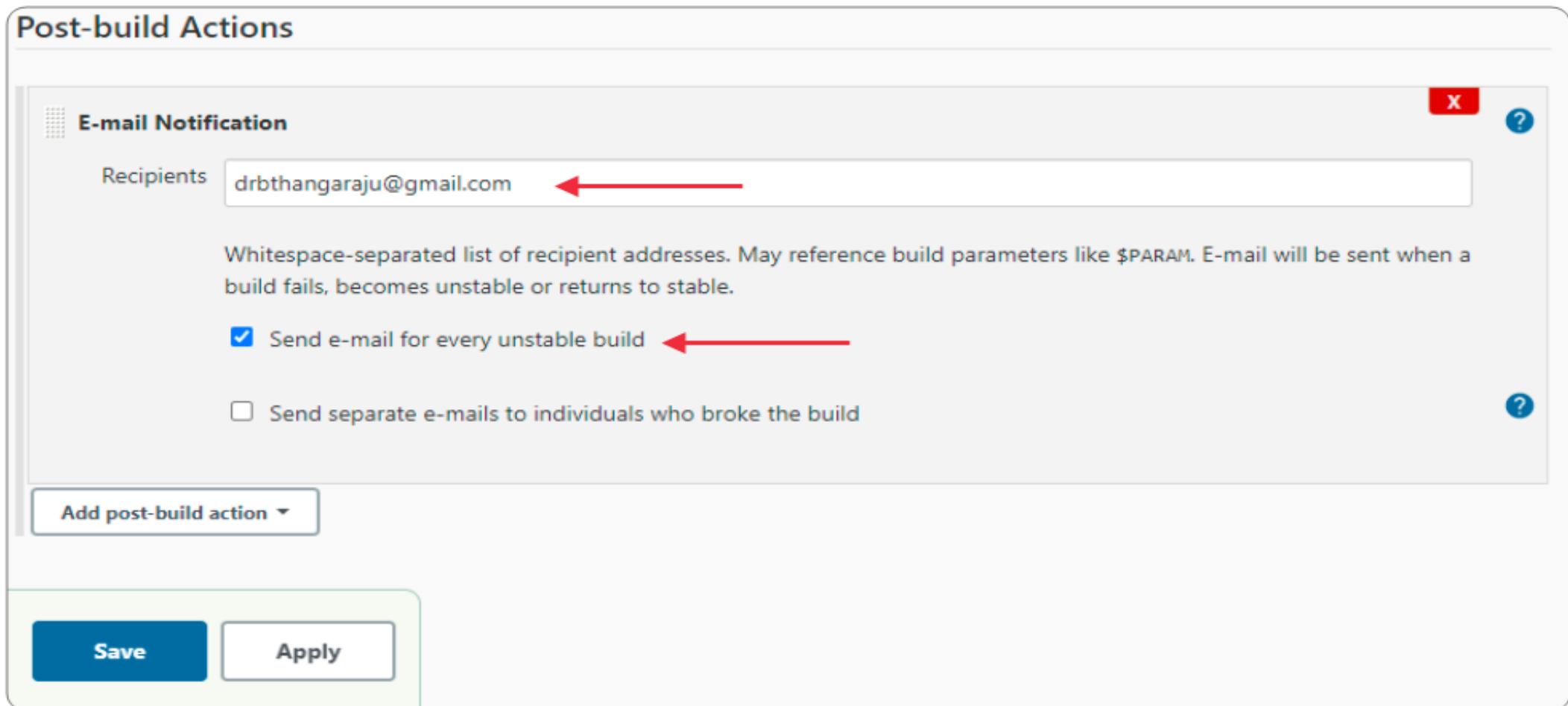
Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

Send e-mail for every unstable build 

Send separate e-mails to individuals who broke the build 

Add post-build action ▾

Save **Apply**



Email Notification

The screenshot shows a Gmail inbox with the following details:

- Compose** button
- Inbox** tab (highlighted, 26 messages)
- Primary** tab
- Social** tab
- Promotions** tab (4 new messages: Medical Benefits, MOTILAL OS...)
- Starred** tab
- Compose** button
- C**, **⋮** buttons
- Search mail** bar
- 1-32 of 32** message count with navigation arrows

The inbox contains the following messages:

- Jenkins-Master** (3:42 PM): Build failed in Jenkins: TestEmail #8 - See <<http://3.95.7.132:8080/job/TestEmail/8/display/redirect>>...
Jenkins build is back to normal : TestEmail #5 - See <<http://3.95.7.132:8080/job/TestEmail/5/display/redirect>>...
Build failed in Jenkins: TestEmail #4 - See <<http://3.95.7.132:8080/job/TestEmail/4/display/redirect>>...
- Jenkins-Master** (3:48 PM): Build failed in Jenkins: TestEmail #4 - See <<http://3.95.7.132:8080/job/TestEmail/4/display/redirect>>...

Email Notification

Build failed in Jenkins: TestEmail #8 ➔ [Inbox](#) ×

 Jenkins-Master <drbalat.raju@gmail.com>
to me ▾

See <<http://3.95.7.132:8080/job/TestEmail/8/display/redirect>>

Changes:

Started by user admin
Running as SYSTEM
Building in workspace <<http://3.95.7.132:8080/job/TestEmail/ws/>>
[TestEmail] \$ /bin/sh -xe /tmp/jenkins8426018425159410393.sh
+ fre -m
/tmp/jenkins8426018425159410393.sh: 2: /tmp/jenkins8426018425159410393.sh: fre: not found
Build step 'Execute shell' marked build as failure

Email Notification

Jenkins build is back to normal : TestEmail #5 ➔ Inbox x

Jenkins-Master <drbalat.raju@gmail.com>
to me ▾

See <<http://3.95.7.132:8080/job/TestEmail/5/display/redirect>>



Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/TestEmail
[TestEmail] $ /bin/sh -xe /tmp/jenkins11996699832483977733.sh
+ free -m
      total        used        free      shared  buff/cache   available
Mem:       978         565          70          0        342        268
Swap:          0           0           0
Sending e-mails to: drbthangaraju@gmail.com
Finished: SUCCESS
```

Parameterized Jenkins Job

Parameterized Project

```
# !/bin/bash
# A simple calculator shell program

a=$1
b=$2

echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
ch=$3

case $ch in
 1)res=`expr $a + $b` ;;
 2)res=`expr $a - $b` ;;
 3)res=`expr $a \* $b` ;;
 4)res=`expr $a / $b` ;;
esac
echo "Result : $res"
```

Step1. Create a freestyle project: **calculator parameterized**

Step 2. Develop a simple calculator shell program in the default workspace

Step 3. During execution, you can choose the a, b and choice through Jenkins ‘This project is parameterized’ option

Enter the First and Second Parameters

Dashboard > calculator parameterized

General Source Code Management Build Triggers Build Environment Build Post-build Actions

GitLab Connection

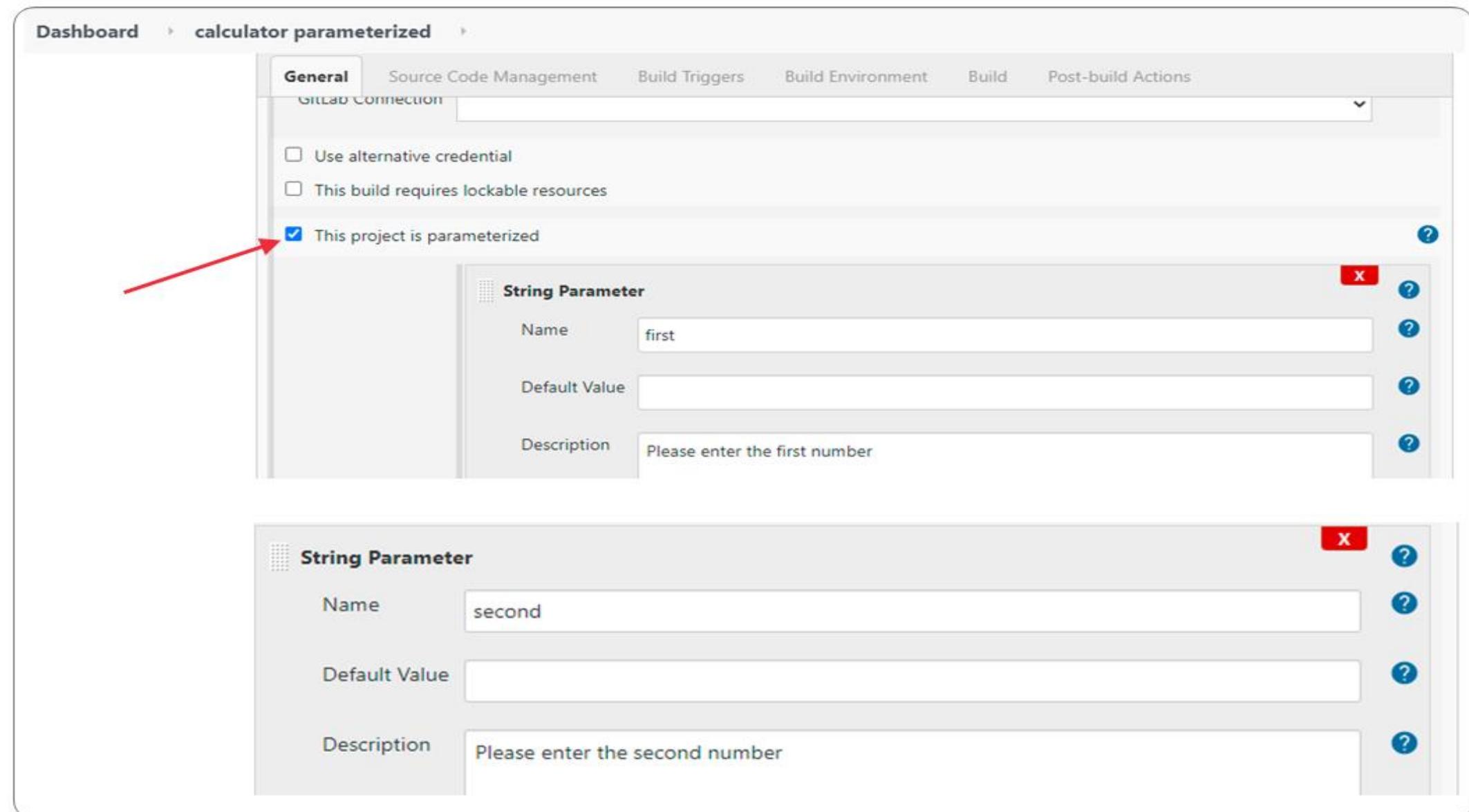
Use alternative credential
 This build requires lockable resources
 This project is parameterized

String Parameter

Name	first
Default Value	
Description	Please enter the first number

String Parameter

Name	second
Default Value	
Description	Please enter the second number



Choice Parameter

Choice Parameter

Name ch ?

Choices

- 1
- 2
- 3
- 4

Description

- 1 - add
- 2 - subtract
- 3 - multiply
- 4 - divide

Build Step

Build

Execute shell X ?

Command

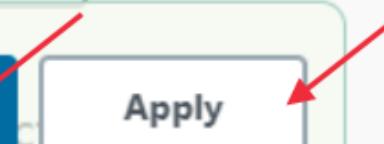
```
chmod u+x c.sh  
./c.sh ${first} ${second} ${ch}
```

See [the list of available environment variables](#)

[Advanced...](#)

Add build step ▾

P Save Apply



Build Job With Parameter

The screenshot shows the Jenkins interface for the project "calculator parameterized". The top navigation bar includes the Jenkins logo, a search bar, and a breadcrumb trail: Dashboard > calculator parameterized. On the left, a sidebar lists project actions: Back to Dashboard, Status, Changes, Workspace, and Build with Parameters. A red arrow points to the "Build with Parameters" link. The main content area is titled "Project calculator parameterized" and displays two links: "Workspace" and "Recent Changes".

Dashboard > calculator parameterized

Back to Dashboard

Status

Changes

Workspace

Build with Parameters

Recent Changes

Workspace

Project calculator parameterized

Variables Values and Select Choice

Project calculator parameterized

This build requires parameters:

first

Please enter the first number

second

Please enter the second number

ch

1 - add

2 - subtract

3 - multiply

4 - divide

Build

This build requires parameters:

first

Please enter the first number

second

Please enter the second number

ch

1 - add

2 - subtract

3 - multiply

4 - divide

Build



Console Output



Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/calculator parameterized
[calculator parameterized] $ /bin/sh -xe /tmp/jenkins5141576588902714368.sh
+ chmod u+x c.sh
+ ./c.sh 255 5 4
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
Result : 51
Finished: SUCCESS
```

Understand How to Manage Users in Jenkins

Importance of User Management

- In a typical project environment, many employees working on a project can access the Jenkins server to run their build or test jobs. This can create security and authorization issues.
- So, you need to give every Jenkins user appropriate permission to enable the Jenkins server's safety and security.
- In Jenkins, different configuration options are available to enable, edit or disable various security features.
- By default, anonymous users have no permissions and logged in users have complete control.

Importance of User Management

- Jenkins admin manages these users based on their roles. Jenkins provides capabilities to add users, edit users and provide different roles to each user. For this, Jenkins provides a role-based authentication plugin.
- The ‘Configure Global Security’ option helps a Jenkins administrator to enable, configure or disable key security features to the Jenkins environment.

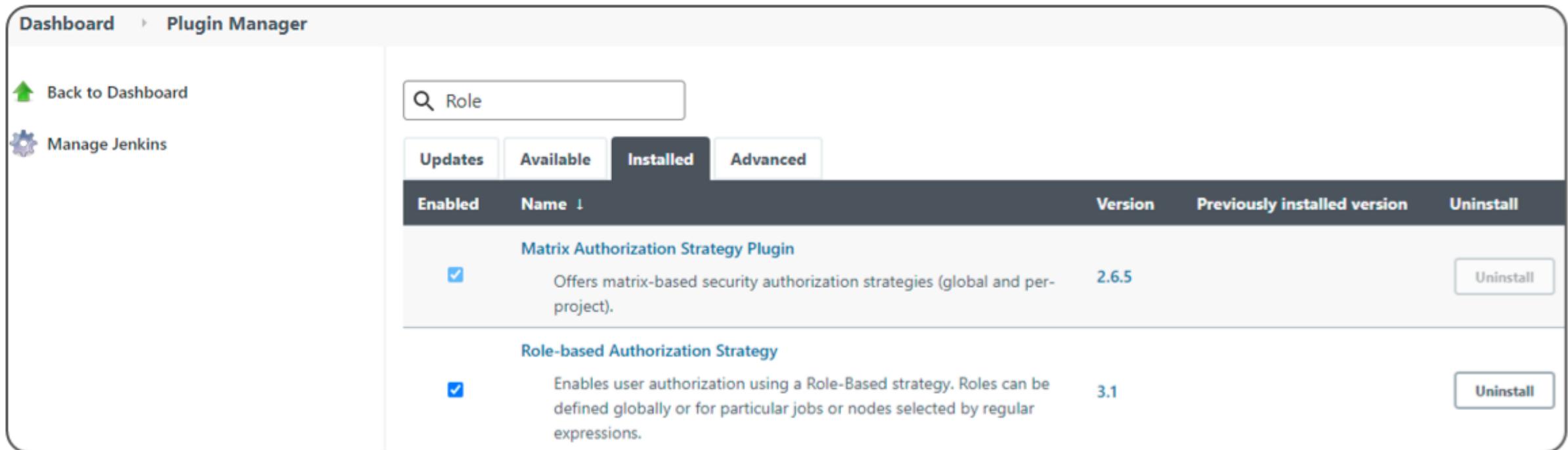
Security Realm

- To configure authentication and authorization schemes in Jenkins, you need to use Security Realm and Authorization configurations.
- Security Realm informs the Jenkins environment how and from where to pull user information.
- Authorization configuration informs the Jenkins environment about which users can access which aspects of Jenkins and to what extent.
- The Security Realm/authentication specifies who can access the Jenkins environment, whereas Authorization specifies what they can access.
- Matrix-based security allows the administrator a granular control over assigning users:
 - Provides the most security and flexibility
 - Recommended for production environments

Role-Based Authentication Strategy

- Used to add a new role-based mechanism to manage users' permission
- Roles can be assigned to users and user groups
- Global Roles: Admin, Developer, Tester, QA and Anonymous
- Allow to set permission: Agent, Job, Run, View and SCM
- Project/Item roles: Allow additional access control for each project separately in the Project configuration screen and give access control to specific user to access only the specified projects
- Agent roles: Set node-related permissions

Required Plugins



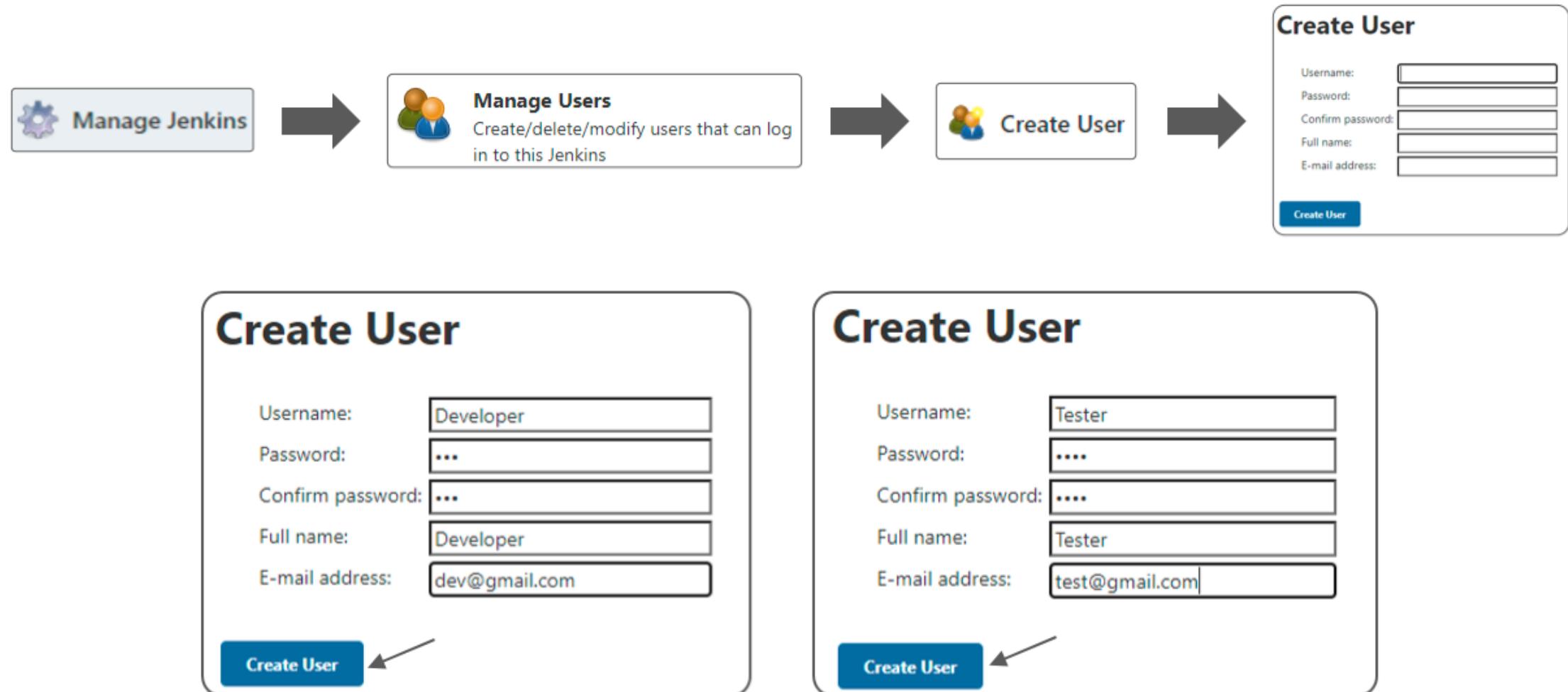
The screenshot shows the Jenkins Plugin Manager interface. The left sidebar has links for 'Back to Dashboard' and 'Manage Jenkins'. The main area has a search bar with 'Role' typed in. Below it are tabs: 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. A table lists installed plugins:

Enabled	Name	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	Matrix Authorization Strategy Plugin	2.6.5		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Role-based Authorization Strategy	3.1		<button>Uninstall</button>

Demo

- Create two users: Developer and Tester
- Configure Global Security: Enable Role-based Authentication strategy
- Manage and Assign roles:
 - Manage roles: Create a Global Roles - add ProjectMember and enable only required access
 - Item roles: Create two roles - Developer and Tester. Enable all the options.
 - Manage and Assign roles: Add Developer and Tester as ProjectMember
 - Item roles: Developer and Tester users should be assigned to Developer and Tester roles, respectively. Set pattern as Prog.* for Developer and Test.* for Tester
- Create two projects: Program1 and TestProject1
- Login as Developer or Tester and view/build/create/delete the Projects

User Account Creation



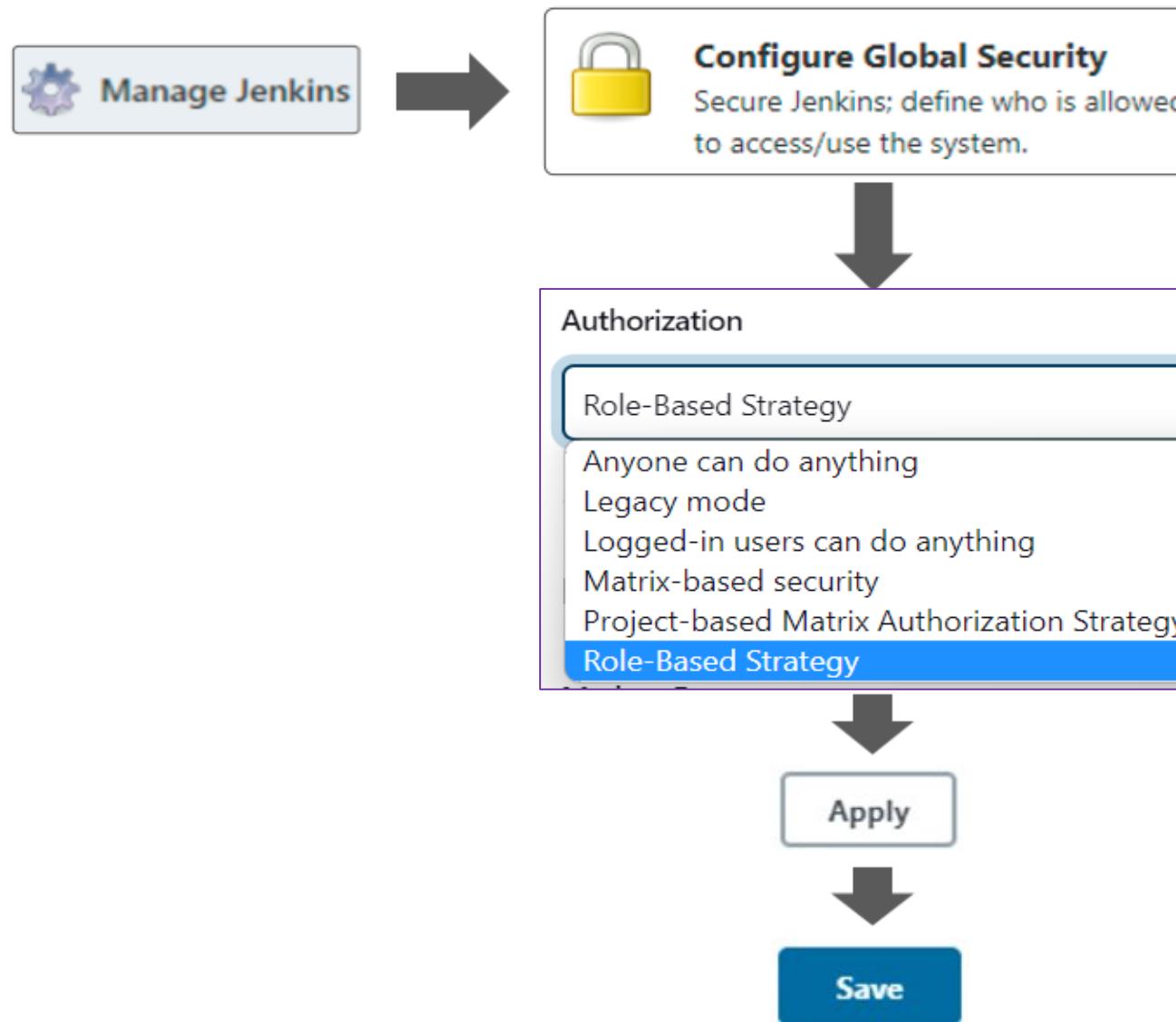
List and Delete User

Users

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID	Name	
 admin	admin	
 Developer	Developer	 
 Tester <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;"> Builds</div>  Configure  My Views  Delete	Tester	 

Enable Role-Based Authentication Strategy



Manage and Assign Roles

Handle permissions by creating roles and assigning them to users/groups

Manage and Assign Roles



[Manage Roles](#)
Manage Roles



Centralized user management with well-defined roles and privileges



[Assign Roles](#)
Assign Roles



A role can be assigned to a user to indicate the set of privileges assigned to the user.



[Role Strategy Macros](#)
Provides info about macro usage and available macros ros

Manage Roles

Global roles

Role	Pattern	Job										View			SCM	
		Tag	Read	Delete	Create	Configure	Update	Run	Replay	Delete	Workspace	Read	Move	Discover	Configure	Create
ProjectMember	○ "Prog.*"	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>												
admin	○ "Test.*"	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Add ProjectMember role in Global roles and enable required permissions

Item roles

Role	Pattern	Job										View			SCM	
		Tag	Read	Delete	Create	Configure	Update	Run	Replay	Delete	Workspace	Read	Move	Discover	Configure	Create
Developer	○ "Prog.*"	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>									
Tester	○ "Test.*"	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>									

Add Developer and Tester roles and set pattern as Prog.* for Developer and Test.* for Tester.

Assign Roles: Global and Item Roles

Dashboard > Manage Jenkins > Manage and Assign Roles

+ New Item

People

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

Assign Roles

Global roles

User/group	ProjectMember	admin	
Developer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Tester	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Prof. B. Thangaraju	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Item roles

User/group	Developer	Tester	
Developer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Tester	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

List of Projects Created

Jenkins

search

admin 2 log out

Dashboard >

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
Grey	Sunny	Program1	N/A	N/A	N/A
Blue	Sunny	TestProject	15 min - #3	N/A	7 ms

Icon: S M L

Legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Login as Developer and View the Dashboard

The screenshot shows the Jenkins dashboard. On the left sidebar, there are links for 'New Item', 'People', 'Build History', and 'My Views'. The main area displays a table with one row for 'Program1'. The columns are labeled 'All', 'S', 'W', 'Name ↓', 'Last Success', 'Last Failure', and 'Last Duration'. Below the table, there are icons for 'Icon: S M L', 'Legend', and three 'Atom feed' options: 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. A small '90' is visible in the bottom right corner of the dashboard area.

All	S	W	Name ↓	Last Success	Last Failure	Last Duration
			Program1	N/A	N/A	N/A

Icon: S M L Legend Atom feed for all Atom feed for failures Atom feed for just latest builds

90

- Developer can view only the jobs started with Prog.*
- Developer can build the available Job.
- Developer can create a new development job, but the job name should starts with Prog.
- Developer cannot see: Tester projects, configure the system and manage plugin.

Login as Tester and View the Dashboard

The screenshot shows the Jenkins dashboard for a user named 'Tester'. On the left sidebar, there are links for 'New Item', 'People', 'Build History', and 'My Views'. The main area displays a table with one job entry:

All	S	W	Name	Last Success	Last Failure	Last Duration
			TestProject	1 hr 26 min - #1	N/A	8 ms

Below the table, there is a legend: 'Icon: SML'. At the bottom, there are three feed links: 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

- Tester can view only the jobs started with Test.*
- Tester can build the available Job
- Tester can create a new test job, but the job name should starts with Test.
- Tester cannot see: Developer projects, configure the system and manage plugin

Tester: Build and Verify

Jenkins

Dashboard > TestProject > #3

Back to Project Status Changes Console Output View as plain text Edit Build Information

Console Output

Started by user Tester
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/TestProject
Finished: SUCCESS

Integrating Jenkins With Git

Introduction

- So far, We have seen how to Trigger build remotely and build after other projects are built (chains of Jenkins job).
- Build periodically – You can trigger the jobs periodically with crontab time format.
- In this section, let's see how to GitHub hook trigger for GitScm polling and Poll SCM.
- First we need to integrate with GitHub repository.
- Next, Install Git Plugin.

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

Install Git Plugin

<input checked="" type="checkbox"/>	Git client plugin Utility plugin for Git support in Jenkins	3.6.0	Uninstall
 <input checked="" type="checkbox"/>	Git plugin This plugin integrates Git with Jenkins.	4.6.0	Uninstall
<input checked="" type="checkbox"/>	GIT server Plugin Allows Jenkins to act as a Git server.	1.9	Uninstall
<input checked="" type="checkbox"/>	GitHub API Plugin This plugin provides GitHub API for other plugins.	1.123	Uninstall
<input checked="" type="checkbox"/>	GitHub Branch Source Plugin Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	2.9.7	Uninstall
 <input checked="" type="checkbox"/>	GitHub plugin This plugin integrates GitHub to Jenkins.	1.33.1	Uninstall

Poll SCM

Poll Source Code Management (SCM) vs Build Periodically

- **Build Periodically:** Jenkins builds periodically even if there are no changes in the project.
- **Poll SCM:** Jenkins builds periodically only if any new changes are made in the project.

Poll SCM

- * * * * * - for every minute, Jenkins polls periodically the GitHub to check whether any new commits were made.
- If there are any changes pushed since the last build, then Jenkins automatically builds the project.

Create a New Freestyle Project

Dashboard > All

Enter an item name

HelloWorld Python Program

» A job already exists with the name 'HelloWorld Python Program'

 **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.

OK

Copy Your Project GitHub URL

The screenshot shows a GitHub repository page for the user 'BThangaraju' named 'Jenkins'. The page includes navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these are statistics: master branch, 1 branch, 0 tags, and a Go to file button. On the right, there's a 'Code' dropdown menu with options for Clone (HTTPS, SSH, GitHub CLI), a copy link button, and instructions to use Git or SVN. A message encourages adding a README. At the bottom, there are links for GitHub Desktop and ZIP download.

BThangaraju / Jenkins

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master ▾ 1 branch 0 tags Go to file Add file ▾ Code ▾

BThangaraju New commit

HelloWorld.py New commit

Help people interested in this repository understand your project by adding a README

Clone

HTTPS SSH GitHub CLI

<https://github.com/BThangaraju/Jenkins>

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

Enable Git

- Select Git in the Source Code Management and enter GitHub URL in the given Repository URL option.

Source Code Management

None

Git

Repositories

Repository URL	<input type="text" value="https://github.com/BThangaraju/Jenkins.git"/>	?
Credentials	<input type="button" value="- none -"/> <input type="button" value="Add"/>	?
		<input type="button" value="Advanced..."/>
		<input type="button" value="Add Repository"/>

Branches to build

Branch Specifier (blank for 'any')	<input type="text" value="*/master"/>	X	?
------------------------------------	---------------------------------------	---	---

Poll SCM Fields and Format

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

field	allowed values
-----	-----
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or use names)

Examples:

Every fifteen minutes (perhaps at :07, :22, :37, :52):

H/15 * * * *

Every ten minutes in the first half of every hour (three times, perhaps at :04, :14, :24):

H(0-29)/10 * * * *

Once every two hours at 45 minutes past the hour starting at 9:45 AM and finishing at 3:45 PM every weekday:

45 9-16/2 * * 1-5

Once in every two hour slot between 8 AM and 4 PM every weekday (perhaps at 9:38 AM, 11:38 AM, 1:38 PM, 3:38 PM):

H H(8-15)/2 * * 1-5

Once a day on the 1st and 15th of every month except December:

H H 1,15 1-11 *

Select Poll SCM and Enter the Value

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 ?

Schedule

⚠ Do you really mean "every minute" when you say "***"? Perhaps you meant "H *****" to poll once per hour**

Would last have run at Friday, March 12, 2021 at 3:27:40 AM Coordinated Universal Time; would next run at Friday, March 12, 2021 at 3:27:40 AM Coordinated Universal Time.

Ignore post-commit hooks ?

Save

Apply

Enter Command to Execute the Project

Build

Execute shell

Command `./HelloWorld.py`

X ?

See the list of available environment variables

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply

The screenshot shows the Jenkins 'Build' configuration page. At the top left, it says 'Build'. Below that is a section titled 'Execute shell' with a 'Command' field containing the text './HelloWorld.py'. There are 'X' and '?' buttons in the top right corner of this section. Below the command field is a link 'See the list of available environment variables'. To the right of this link is a 'Advanced...' button. At the bottom left of this section is a 'Add build step ▾' button. Below this section is another titled 'Post-build Actions' with a 'Add post-build action ▾' button. At the very bottom are two buttons: a blue 'Save' button and a white 'Apply' button.

Jenkins Dashboard

The screenshot shows the Jenkins dashboard interface. On the left sidebar, there are several navigation links: New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. The main area displays a view for the 'HelloWorld Python Program' job. The view title is 'HelloWorld Python Program'. It features two icons: a blue sphere and a yellow sun. Below the title, there is a link labeled 'Icon: S M L'. At the bottom of the view, there is a terminal-like window showing the source code of the 'HelloWorld.py' file. The code is as follows:

```
ubuntu@ip-172-31-81-117: ~/git/Jenkins
#!/usr/bin/python3
# This Python program will print Hello World...
print("Hello World . . . \n")
```

Console Output

Dashboard > HelloWorld Python Program > #1

Back to Project Status Changes Console Output View as plain text Edit Build Information Delete build '#1' Git Build Data

Console Output After Building Manually

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld Python Program
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/BThangaraju/Jenkins.git
> git init /var/lib/jenkins/workspace/HelloWorld Python Program # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.17.1'
> git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision ee595dd77f6cb3b018d86fa0824e755b020a5elb (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f ee595dd77f6cb3b018d86fa0824e755b020a5elb # timeout=10
Commit message: "New commit"
First time build. Skipping changelog.
[HelloWorld Python Program] $ /bin/sh -xe /tmp/jenkins7351324438992846629.sh
+ ./HelloWorld.py

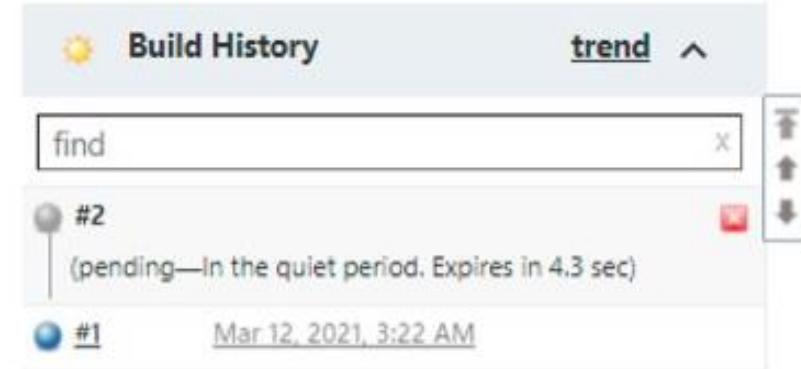
Hello World...

Finished: SUCCESS
```

Commit Changes and Build Automatically

```
ubuntu@ip-172-31-81-117:~/git/Jenkins
#!/usr/bin/python3
# This Python program will print Hello World
print("\n Hello World...\n")
print("\n Hello World...\n")

ubuntu@ip-172-31-81-117:~/git/Jenkins$ git add .
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git commit -m "First commit"
[master a8dc0e6] First commit
 1 file changed, 1 insertion(+)
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git push origin master
Username for 'https://github.com': b.thangaraju@iiitb.ac.in
Password for 'https://b.thangaraju@iiitb.ac.in@github.com':
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/BThangaraju/Jenkins.git
 ee595dd..a8dc0e6 master -> master
ubuntu@ip-172-31-81-117:~/git/Jenkins$ vim HelloWorld.py
```



The screenshot shows the Jenkins Build History interface. At the top, there is a search bar labeled 'find' and a 'trend' button. Below the search bar, build #2 is listed as 'pending—In the quiet period. Expires in 4.3 sec'. Below it, build #1 is listed with the timestamp 'Mar 12, 2021, 3:22 AM'. To the right of the build list are three small icons: a magnifying glass, an upward arrow, and a downward arrow.



This screenshot shows the Jenkins Build History interface again. The search bar now contains 'find'. Below the search bar, build #2 is listed with the timestamp 'Mar 12, 2021, 3:33 AM'. Below it, build #1 is listed with the timestamp 'Mar 12, 2021, 3:22 AM'. At the bottom of the screen, there are two links: 'Atom feed for all' and 'Atom feed for failures'.

- After Committing changes in the local Git repo, we need to push the changes into our GitHub repository.
- Then Jenkins check the GitHub repository periodically and automatically build the Job.

GitHub Hook Trigger for GITScm Polling

- For GitHub hook trigger for GITScm polling, Jenkins will not build the job periodically; instead, when you commit new changes in the GitHub repository, only then will Jenkins trigger the job to build.

Working With Webhooks

The screenshot shows the GitHub repository settings for 'BThangaraju / Jenkins'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Settings' tab is highlighted with a red underline and a blue arrow pointing to it from the left sidebar. The left sidebar contains options like Options, Manage access, Security & analysis, Branches, Webhooks (with a blue arrow pointing to it), and Notifications. The main 'Settings' section displays the repository name 'Jenkins' with a 'Rename' button, a 'Template repository' checkbox (unchecked), and a 'Social preview' section for customizing social media previews.

BThangaraju / Jenkins

Unwatch 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Settings

Repository name

Jenkins Rename

Template repository

Template repositories let users generate new repositories with the same directory structure and files. [Learn more](#).

Social preview

Upload an image to customize your repository's social media preview.

Working With Webhooks

The screenshot shows a GitHub repository page for 'BThangaraju / Jenkins'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Settings tab is currently selected, indicated by a red underline. On the left, there's a sidebar with options: Options (selected), Manage access, and Security & analysis. The main content area is titled 'Webhooks' and contains the following 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](#)'. A blue arrow points from the text 'Learn more in our [Webhooks Guide](#)' towards the 'Add webhook' button, which is located in a grey box on the right side of the page.

BThangaraju / Jenkins

Unwatch 1 | Star 0 | Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Options

Manage access

Security & analysis

Webhooks

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

Add webhook

Working With Webhooks

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed (JSON, x-www-form-urlencoded, etc). More information can be found in our documentation.

Payload URL *

http://52.87.100.176:8080/github-webhook/

Content type

application/json

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.

Add webhook

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

Build History	
<input type="text"/> find	X
 #3	Mar 12, 2021, 3:51 AM
 #2	Mar 12, 2021, 3:33 AM
 #1	Mar 12, 2021, 3:22 AM

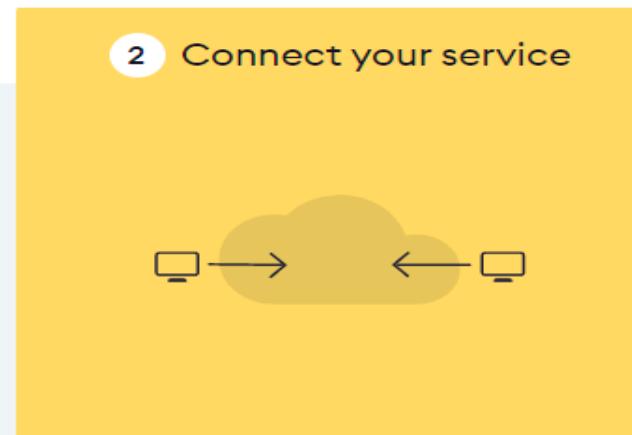
ngrok - secure introspectable tunnels to localhost

1. Ngrok exposes local servers behind NATs (Network Address Translation) and firewalls to the public internet over secure tunnels.
2. ngrok provides a real-time web UI where you can introspect all HTTP traffic running over your tunnels.
3. ngrok allows you to expose a web server running on your local machine to the internet. Just tell ngrok what port your web server is listening on.
4. Example: Expose a web server on port 80 of your local machine to the internet. `$ngrok http 8080`
5. If different port number for ex: 5001 then: `$ngrok http https://localhost:5001`

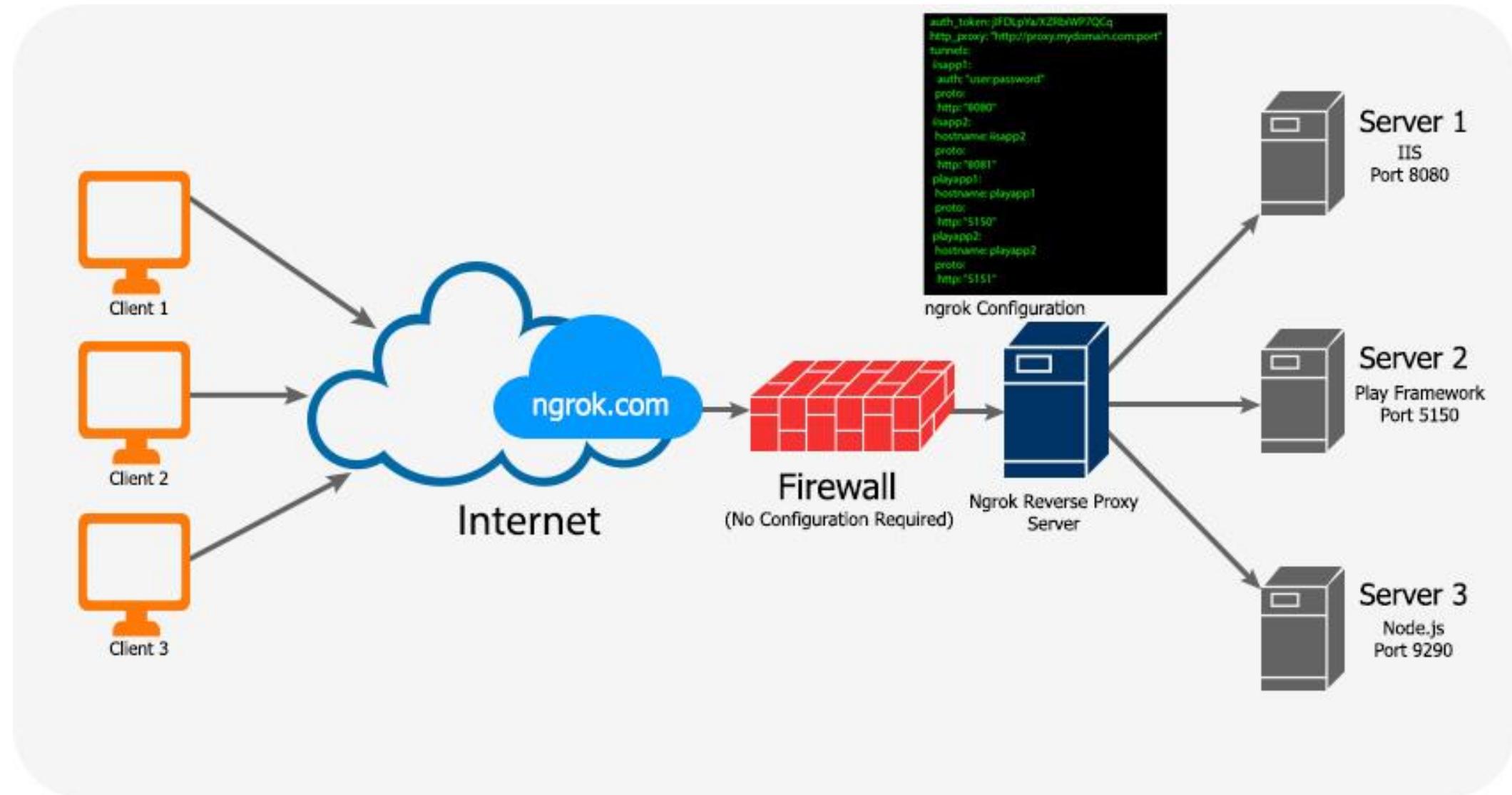
ngrok - secure introspectable tunnels to localhost

1. Ngrok exposes local servers behind NATs (Network Address Translation) and firewalls to the public internet over secure tunnels.
2. **ngrok** provides a real-time web UI where you can introspect all HTTP traffic running over your tunnels.

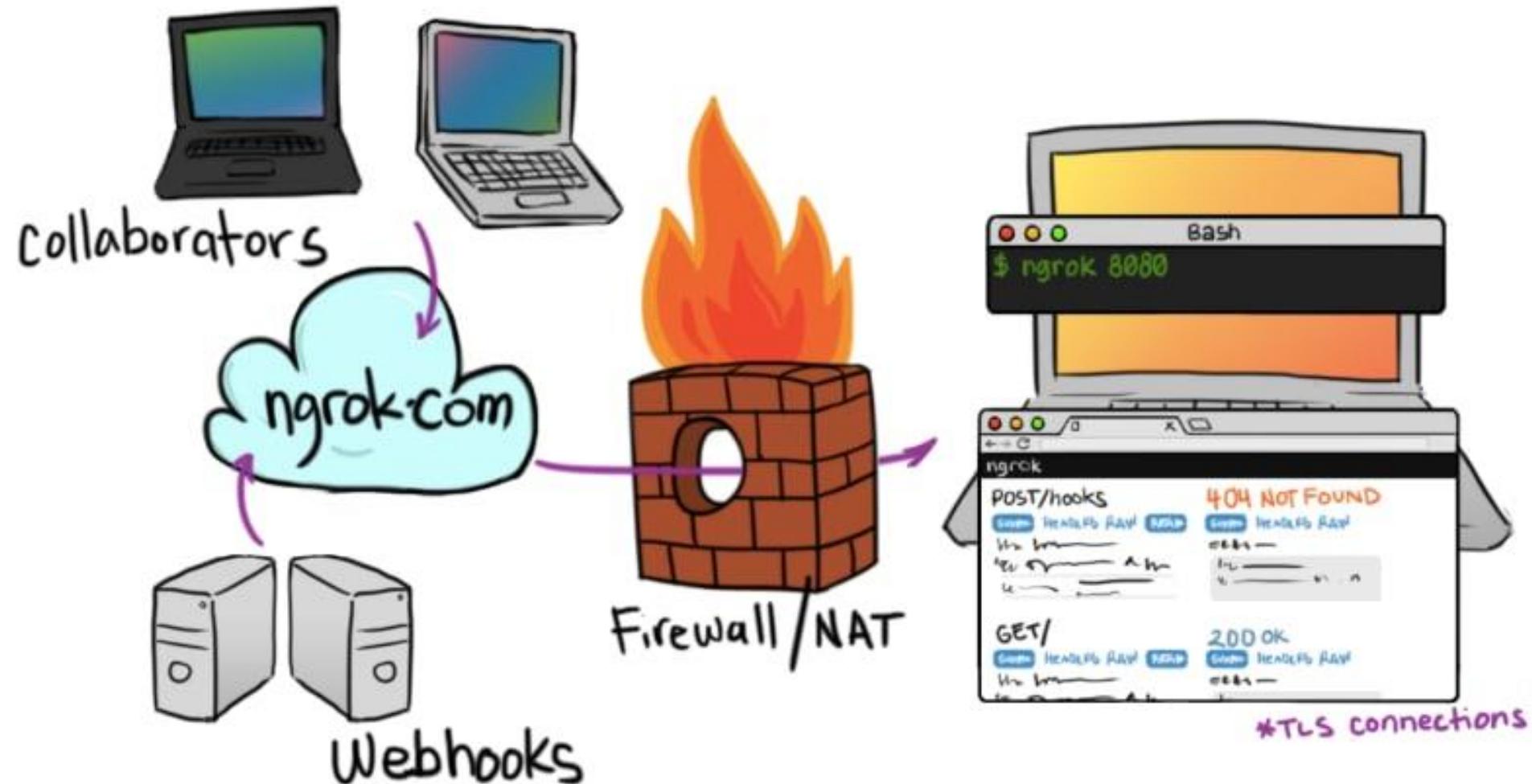
HOW IT WORKS



Ngrok Architecture



Ngrok Connection



Ngrok Installation

1. Sign up in <https://ngrok.com/>
2. Download ngrok from: <https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz>
3. Then extract ngrok from the terminal: \$sudo tar xvzf ~/Downloads/ngrok-stable-linux-amd64.tgz -C /usr/local/bin
4. Copy Authtoken from: <https://dashboard.ngrok.com/get-started/your-authtoken>
5. Add Authtoken: \$ngrok authtoken <token>
6. Execute \$ngrok http 8080; copy the public ip address for your local host.

```
ngrok by @inconshreveable                                     (Ctrl+C to quit)

Session Status                                              online
Account                                                       Thangaraju (Plan: Free)
Version                                                       2.3.40
Region                                                        United States (us)
Web Interface                                                 http://127.0.0.1:4040
Forwarding                                                    http://5336-122-167-77-156.ngrok.io -> http://localhost:8080
                                                               https://5336-122-167-77-156.ngrok.io -> http://localhost:8080

Connections                                                 ttl     opn     rt1     rt5      p50      p90
                                                               0       0     0.00    0.00    0.00    0.00
```

1 Click on **Settings** in the sidebar.

2 Click on **Developer settings** in the sidebar.

3 Click on **Personal access tokens**.

4 Click on **Generate new token**.

5 Enter a token name: `jenkins-webhook`.

6 Set expiration to `30 days`.

7 Select scopes: `admin:repo_hook`, `write:repo_hook`, and `read:repo_hook`.

8 Click **Generate token**.

9 Confirm access by entering the password `sudo` and clicking **Confirm**.

Personal access tokens (classic)

[Generate new token ▾](#)

[Revoke all](#)

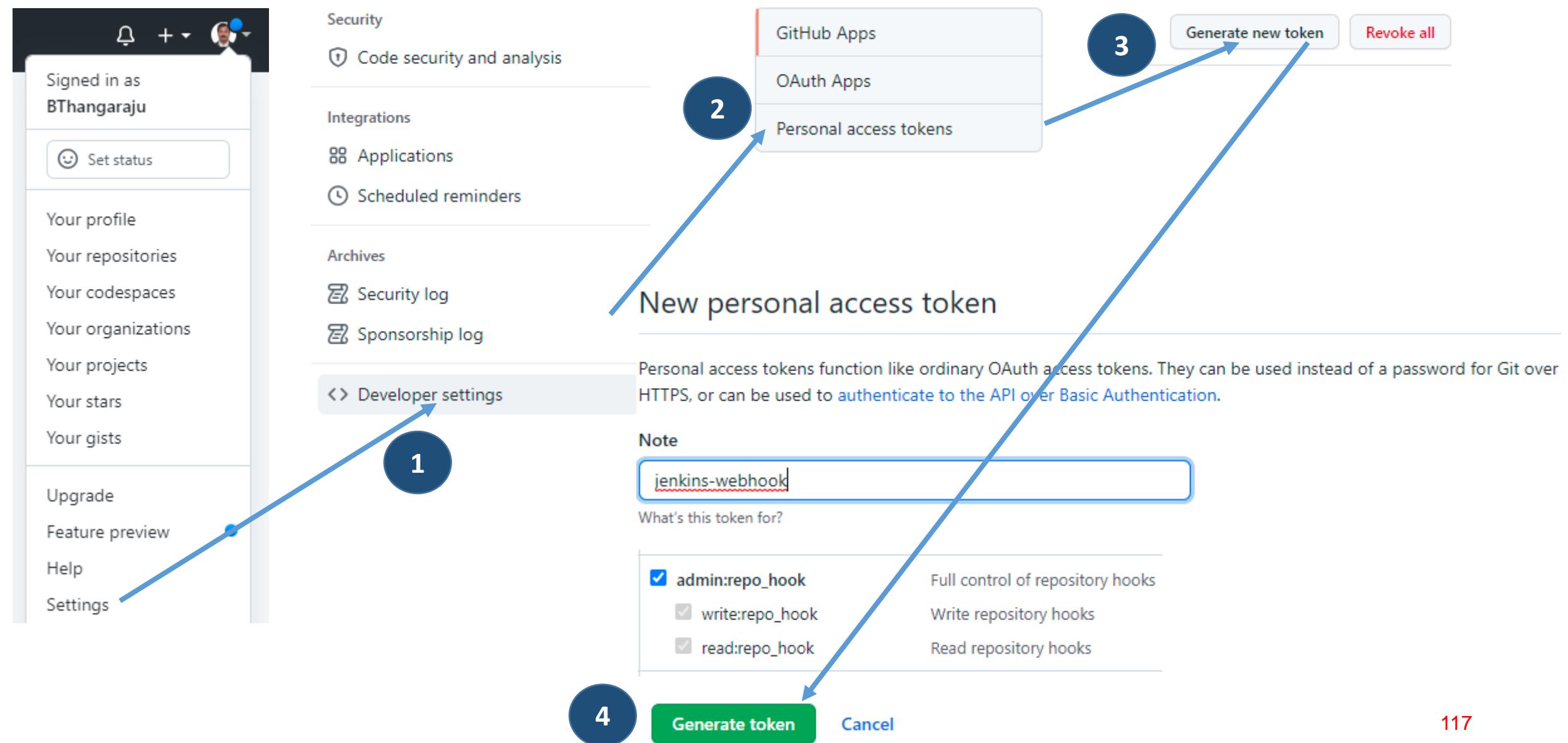
Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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 

[Delete](#)

Secret Text creation -GitHub



Webhook Configuration -GitHub

Personal access tokens

[Generate new token](#)[Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_mroSnRC21IuiU7FEZ0fNi3NS7nkYuC3YSmJT  **copy**

[Delete](#)**5****Secret Text**

Create Webhook from Github repository

BThangaraju / Jenkins Public

[Pin](#)[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[Insights](#)[Settings](#)[Code and automation](#)[Branches](#)[Actions](#)[Webhooks](#)**Webhooks****2****3**[Add webhook](#)

Webhook Configuration -GitHub

Payload URL *

http://0fcd-122-167-77-156.ngrok.io/github-webhook/

4

Copied from ngrok public IP address

Content type

application/x-www-form-urlencoded

Secret

If you've lost or forgotten this secret, you can change it, but be aware that changing it will break your webhook. You'll need to update it on all of the services that use it. — [Change Secret](#)

Copy from personal access tokens from webhook configuration - GitHub

5

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.

6

[Update webhook](#)

[Delete webhook](#)

GitHub Server Configuration –Jenkins Dashboard

The image shows the Jenkins Dashboard and the GitHub configuration interface. A red arrow numbered 1 points from the 'Manage Jenkins' link in the sidebar to the 'Configure System' gear icon. Another red arrow numbered 2 points from the 'Configure System' section to the 'Jenkins URL' input field containing the ngrok public IP address. A third red arrow numbered 3 points from the 'ngrok public IP address' to the 'Name' field in the GitHub configuration. A fourth red arrow numbered 4 points from the 'Secret text' dropdown in the Jenkins credentials provider to the 'Secret' input field. A fifth red arrow numbered 5 points from the 'Secret' input field back to the 'Secret text' dropdown. A sixth red arrow numbered 6 points from the 'Save' button to the 'Apply' button.

Dashboard →

- New Item
- People
- Build History
- Manage Jenkins **1**
- My Views
- Lockable Resources
- New View

Configure System
Configure global settings and paths.

Jenkins Location

Jenkins URL **2**
http://0fcd-122-167-77-156.ngrok.io/
ngrok public IP address

GitHub

GitHub Servers

GitHub Server

Name **3**: github

API URL **4**: https://api.github.com

Credentials **5**

Secret text **6**

Add

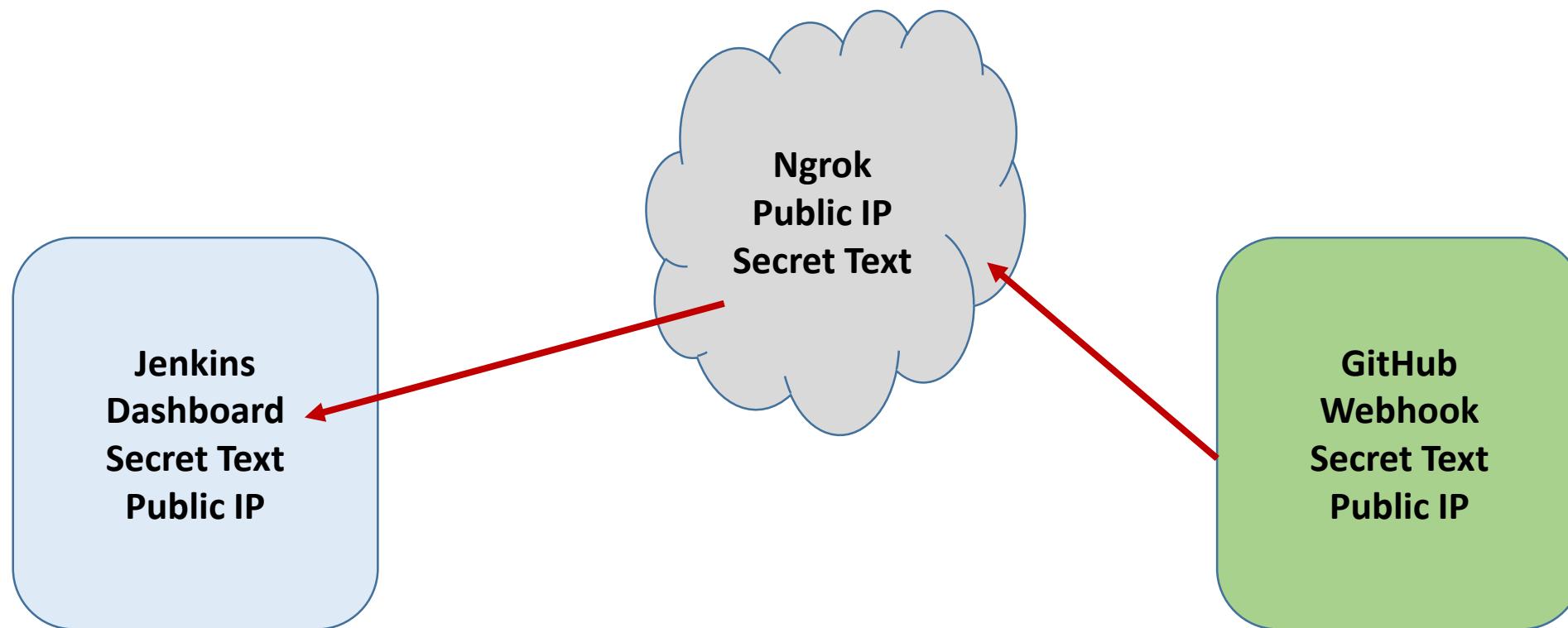
Cancel

Save

Apply

Copy from personal access tokens from webhook configuration - GitHub

GitSCM Polling



Introduction to Jenkins Pipeline

Jenkins Pipeline Introduction

Pipeline:

- It can be specified as code, so you can write pipeline script and maintain version control in the Git repository.
- It will provide continuous release of the application.
- Sequence of stages to perform the given tasks such as pulling code from the Git repository, static code analysis, building project, executing unit test, automated tests, performance tests and deploying application.

Types of Pipeline

- **Declarative**
 - New method
 - Easy to use for beginners
 - Groovy language skill is desirable
- **Scripted**
 - Traditional
 - Based on Groovy Domain Specific Language (DSL)
 - Multiple features - very expressive and flexible tool
 - Difficult to use for beginners
 - Should have working experience on Groovy language

Declarative vs Scripted Pipeline Script Template

```
pipeline {  
    agent any  
    stages {  
        stage('build code') {  
            steps {  
                /*write steps */  
            }  
        }  
        stage ('test') {  
            steps {  
                /*write steps */  
            }  
        }  
    }  
}
```

Declarative Pipeline Script

```
node {  
    stage ('build code' {  
        /*write steps */  
    }  
    stage ('test') {  
        /*write steps */  
    }  
}
```

Scripted Pipeline Script

Pipeline – contains all the script content
Agent and Node – defines the agent where the pipeline will run
Stages – contains all the stages
Steps – way to execute various jobs

Pipeline Directives

Directive allows you to define a list of parameters to be used in the script. Parameters should be provided once the pipeline is triggered.

- **Environment** – defined as environment variables
- **Input** – prompt for input
- **Options** – configure pipeline-specific options like retry, timeout, etc.
- **Parallel** – list of nested stages to be run in parallel
- **Parameters** – list of parameters to provide when triggering the Pipeline (e.g., string, password)
- **Post** – run at the end of a Pipeline's execution (e.g., add some notification or other end of Pipeline tasks)

Pipeline Directives

- **Tools** – defining tools or packages to auto-install and put on the PATH (e.g., maven, Jdk, gradle)
- **Triggers** – determines how pipelines should be triggered (e.g., cron, poll SCM)
- **When** – determine executing stage depending on the given condition

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

Jenkins Pipeline

Pipeline:

- Defined as a suite of plugins that helps you orchestrate simple or complex automation

Jenkins Pipeline:

- Provides tool for modeling delivery pipelines, “as Code” (Pipeline as Code)
- Implement and integrate CD pipeline.

CI/CD Pipeline:

- Integrates SDLC stages, steps to execute tasks in each stage, trigger jobs for a given order and show pipeline status with logs
- Automation from Continuous build to Continuous Monitoring (build, test, staging, deploy and monitor)

Demo – Jenkins Pipeline CI/CD

Declarative Pipeline Script

```
pipeline {  
    agent any  
    stages {  
        stage('Build') {  
            steps {  
                echo 'This is job building stage'  
            }  
        }  
        stage('Test') {  
            steps {  
                echo 'This is Testing stage'  
            }  
        }  
        stage('Staging') {  
            steps {  
                echo 'This is Staging environment'  
            }  
        }  
        stage('Deploy') {  
            steps {  
                echo 'This is Deploying stage'  
            }  
        }  
        stage('Monitor') {  
            steps {  
                echo 'This is Monitoring stage'  
            }  
        }  
    }  
}
```

Demo – Jenkins Pipeline CI/CD

Back to Dashboard

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend ^

find

#1 Mar 13, 2021, 6:22 AM

Pipeline PipelineDemo1

Recent Changes

Stage View

Average stage times:
(Average full run time: ~9s)

#1 Mar 13 11:52 No Changes

CI/CD Pipeline View

Build	Test	Staging	Deploy	Monitor
368ms	141ms	127ms	112ms	109ms
368ms	141ms	127ms	112ms	109ms

Permalinks

Demo Programs

- Git Repository

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'master' (with a dropdown arrow), '1 branch' (with a dropdown arrow), '0 tags' (with a dropdown arrow), 'Go to file', 'Add file', and a green 'Code' button with a dropdown arrow. Below this, a list of commits is displayed:

Author	Commit Message	Date	Commits
BThangaraju	Create Test.py	41c44c5 16 hours ago	22 commits
	HelloWorld.py	Update HelloWorld.py	16 hours ago
	Prog1.py	Create Prog1.py	16 hours ago
	Test.py	Create Test.py	16 hours ago

Below the commits, there is a list of three items:

- HelloWorld.py –program to git clone example.
- Prog1.py – for building the code.
- Test.py –for testing the Prog1.py source code.

Demo Programs

```
ubuntu@ip-172-31-57-56: ~
#!/usr/bin/python3
# This Python program will print Hello World...
print("Hello World ...\\n")
```

HelloWorld.py

```
ubuntu@ip-172-31-81-117: ~/git-demo
#!/usr/bin/python3
# Source code for summation of two numbers

def summation(data):
    return sum(data)
```

Prog1.py

```
ubuntu@ip-172-31-81-117: ~
#!/usr/bin/python3
# Test case for adding two numbers
import unittest

from Prog1 import summation

class TestSum(unittest.TestCase):
    def test_list_int(self):
        """
        Test case to add two numbers
        """
        data = [23, 32]
        result = summation(data)
        self.assertEqual(result, 55)

if __name__ == '__main__':
    unittest.main()
```

Test.py

Integration With Git

Source Code Management

None

Git

Repositories

Repository URL	<input type="text" value="https://github.com/BThangaraju/Jenkins.git"/>	?
Credentials	<input type="button" value="- none -"/>	<input type="button" value="Add"/>

Git Clone, Build and Test a Python Script

```
pipeline {  
    agent any  
    stages {  
        stage('Clone Git') { /*you can also specify git location */  
            steps {  
                git 'https://github.com/BThangaraju/Jenkins.git'  
            }  
        }  
        stage('Build Code') {  
            steps {  
                sh "chmod u+x Prog1.py"  
                sh "./Prog1.py"  
            }  
        }  
        stage('Test Code') {  
            steps {  
                sh "chmod u+x Test.py"  
                sh "./Test.py"  
            }  
        }  
    }  
}
```

You can execute any commands for example:
You can compile and execute a.c
sh “gcc a.c”
sh “./a.out”

Pipeline View

Dashboard > PipelineDemo >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#)

[Configure](#)

[Delete Pipeline](#)

[Full Stage View](#)

[Rename](#)

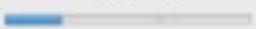
[Pipeline Syntax](#)

Pipeline PipelineDemo

 Recent Changes

Stage View

Average stage times:
(Average full run time: ~2s)

Clone Git	Build Code	Test Code
393ms	667ms	634ms
		
393ms	667ms	634ms

#14 Mar 13 12:14 No Changes

Jenkins Distributed Architecture

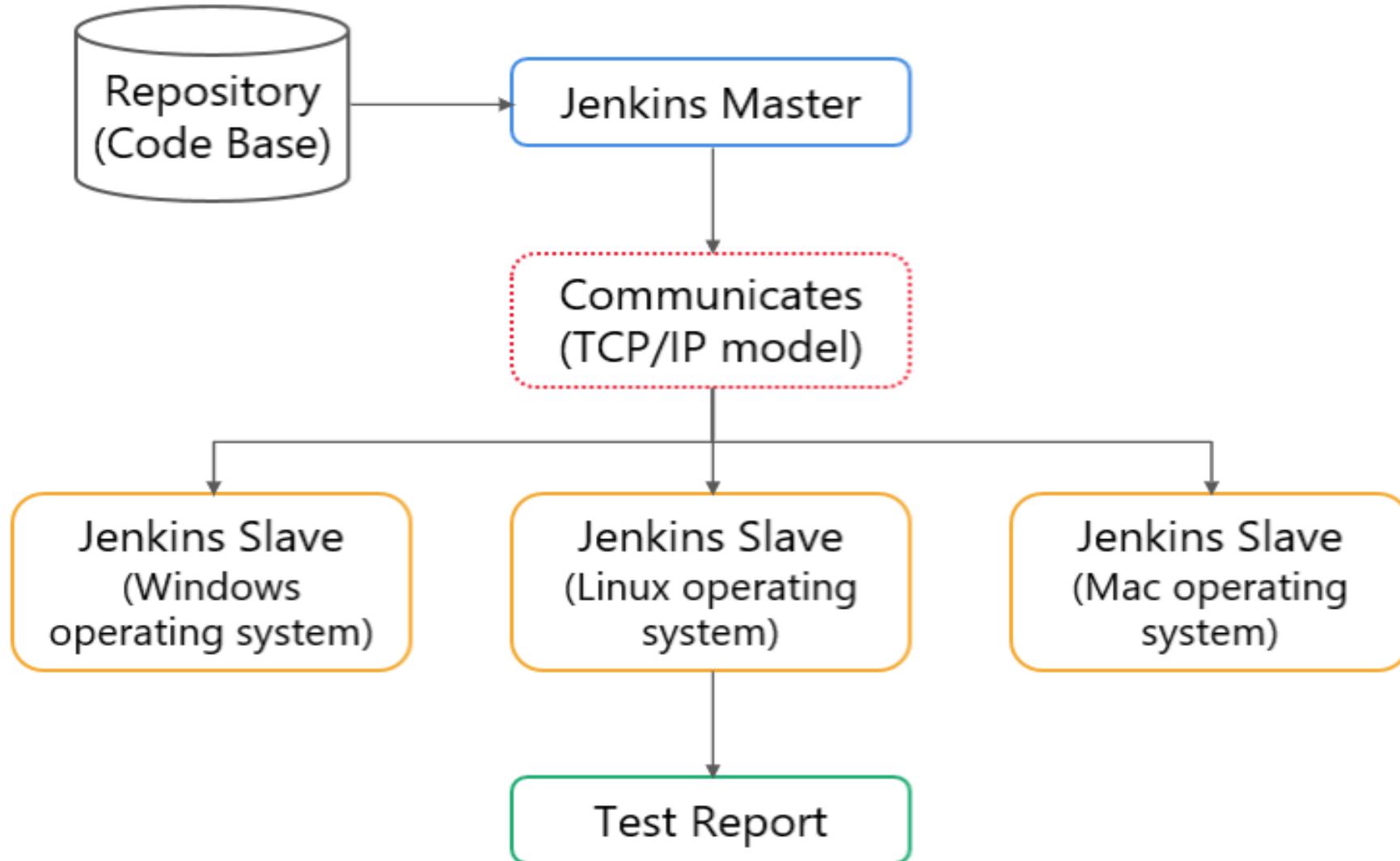
Scaling Jenkins

- Initially, when you start working with Jenkins, you have a single server to carry out all the tasks. The single Jenkins server is also called as Master node or Jenkins controller.
- Single Jenkins server is not enough to meet certain requirements like:
 - When you configure more jobs
 - When you orchestrate more frequent builds
 - When more developers depend on one controller
 - When you add incremental features in large and complex projects frequently
 - When you need different environments (diff OS) to test the build

Scaling Jenkins

- Instead of adding new team members or new projects to an existing single Jenkins controller, you can create additional Jenkins controllers to accommodate new teams or projects.
- The Jenkins distributed architecture enables us to use various environments for each build project, dividing the workload across multiple agents running jobs concurrently.
- Jenkins' distributed architecture is based on the idea of 'Master + Agent'. The master is responsible for coordination and providing the GUI and API endpoints, while the Agents perform the work.
- The Jenkins master manages the Jenkins agents and orchestrates their work by scheduling jobs on agents and monitoring them.
- Agents can link to the Jenkins controller via local or cloud computers.

Jenkins Distributed Architecture



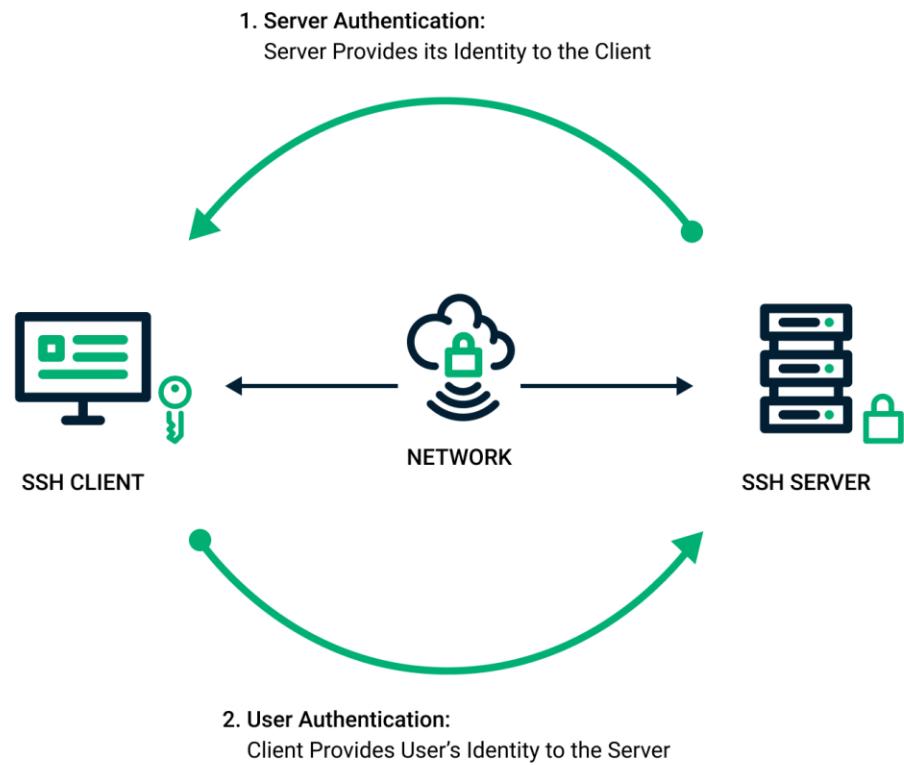
SSH Key

- SSH key is a secure access credential used in the Secure Shell (SSH) protocol.
- SSH keys use key pairs based on public key infrastructure (PKI) technology, the gold standard for digital identity authentication and encryption, to provide a secure and scalable method of authentication.
- As the SSH protocol is widely used for communication in cloud services, network environments, file transfer tools, configuration management tools, and other computer-dependent services, to authenticate identity and protect those services from unintended use or malicious attacks.
- SSH keys not only improve security, but also enable the automation of connected processes, single sign-on (SSO), and identity and access management at scale that today's businesses require.

SSH Key

- The SSH key pair : Private and Public key. The public key is used by both the user and the remote server to encrypt messages.
- On the remote server side, it is saved in a file that contains a list of all authorized public keys.
- On the user's side, it is stored in SSH key management software or in a file on their computer.
- The private key remains only on the system being used to access the remote server and is used to decrypt messages.
- In computer security, challenge-response authentication is a **set of protocols used to protect digital assets and services from unauthorized users, programs or activities.**

SSH Key



- When a user or process requests a connection to the remote server using the SSH client, a challenge-response sequence is initiated to complete authentication.
- The SSH server recognizes that a connection is being requested and sends an encrypted challenge request using the shared public key information.
- The SSH client then decrypts the challenge message and responds back to the server.
- This challenge-response sequence happens automatically between the SSH client and server without any manual action by the user.

Host Setup

From Host:

```
#sudo su - jenkins  
#ssh-keygen  
#ls .ssh/; id_rsa id_rsa.pub
```

Docker Setup:

```
#sudo apt-get install docker.io  
#sudo service docker status  
#sudo docker pull ubuntu  
#sudo docker run --it --name Jenkins_Agent ubuntu /bin/bash
```

In the browser: Manage Jenkins -> Manage Credentials -> click global -> click Add Credentials -> for Kind - select ssh username with private key and enter the details.

ID: Master_Jenkins_Private_key

Description: Jenkins Master Private Key to Add Multiple Agents

Username: jenkins

Private Key -> Enter Directly ->copy and paste id_rsa from the server (/var/lib/jenkins/.ssh/id_rsa)

Check the docker.service in Host

```
jenkins@ip-172-31-81-117:~/.ssh$ systemctl status docker.service
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; disabled; vendor preset: enabled)
  Active: active (running) since Sat 2021-03-20 02:16:40 UTC; 3h 49min ago
    Docs: https://docs.docker.com
   Main PID: 3527 (dockerd)
     Tasks: 10
    CGroup: /system.slice/docker.service
            └─3527 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
jenkins@ip-172-31-81-117:~/.ssh$
```

The screenshot shows the Jenkins web interface. At the top, there is a navigation bar with a Jenkins logo, a search bar, and user information for 'admin'. Below the navigation bar, the URL 'Dashboard > Credentials' is visible. On the left side, there is a sidebar with links for 'New Item', 'People', 'Build History', and 'Dashboard'. The main content area is titled 'Credentials' and displays a table with columns: T, P, Store, Domain, ID, and Name. One row is visible, showing 'Jenkins' under 'Domain', '(global)' under 'ID', and 'jenkins (Jenkins Master Private Key to add multiple agents)' under 'Name'. A red arrow points from the text 'Master_Jenkins_Private_key' to the '(global)' entry in the table.

Configure Jenkins Global Credentials

Global credentials (unrestricted) > jenkins (Jenkins Master Private Key to add multiple agents)

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

ID: Master_Jenkins_Private_key

Description: Jenkins Master Private Key to add multiple agents

Username: jenkins

Private Key:

Enter directly

Key: cat id_rsa; copy and paste here and save

Docker Configuration

Inside the container:

```
#adduser jenkins  
#usermod -aG sudo jenkins  
#apt-get update  
#apt-get install sudo  
#su - jenkins
```

ssh server configuration

```
#sudo apt-get install openssh-server  
#sudo service ssh restart  
#service ssh status  
#sudo apt install openjdk-11-jdk  
#java --version
```

From Host:

```
#ssh-copy-id jenkins@172.17.0.2
```

To Check Docker container:

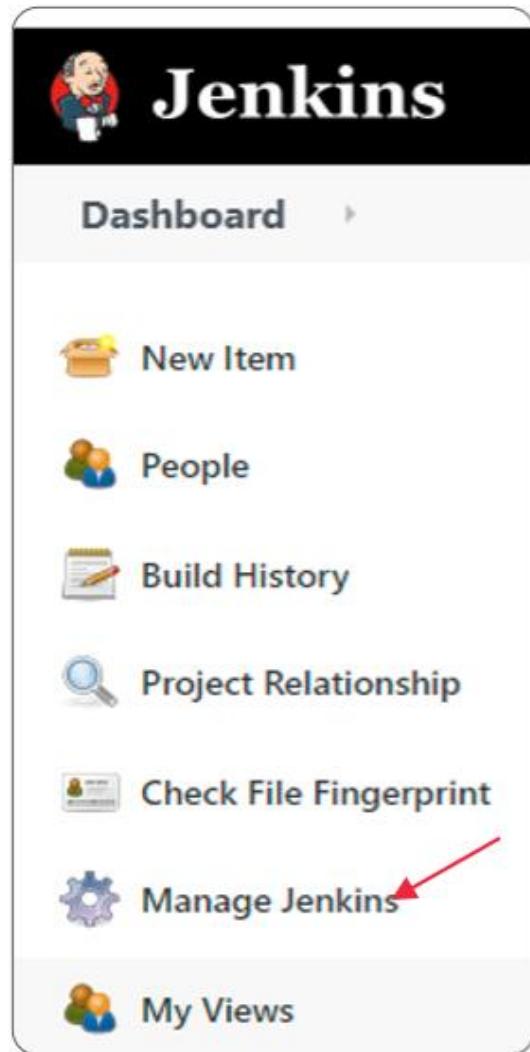
```
#cat /home/jenkins/.ssh/authorized_keys  
It should be same as: id_rsa.pub in host.
```

Copy Authorized Keys From Host to Container

- In the Host: ssh-copy-id [jenkins@172.17.0.2](#)
- Check the copied authorized_keys in the container

```
jenkins@9f9f046f179f:~$ pwd
/home/jenkins
jenkins@9f9f046f179f:~$ cd .ssh
jenkins@9f9f046f179f:~/ssh$ ls
authorized_keys
jenkins@9f9f046f179f:~/ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDj6rSfwR3N+4
0qU3Hrpkb1hiPhYQ9otB6kGmEnTGnB5OAT/t32qsIBLVIx4zWC0
ORxKqe6I6z+2CIfC+4FYSe0kFgznmOISv66FNWs1UHipsPV8P
2Vu/vHGaRDNuPN jenkins@ip-172-31-81-117
jenkins@9f9f046f179f:~/ssh$ █
```

Configure Agent in Jenkins



Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



The screenshot shows the Jenkins Nodes screen. On the left, there is a sidebar with the following items:

- Back to Dashboard
- Manage Jenkins
- New Node** (highlighted with a red arrow)
- Configure Clouds
- Node Monitoring

Configure Agent in Jenkins

The image shows two side-by-side configuration panels for a Jenkins agent.

Left Panel (Agent Configuration):

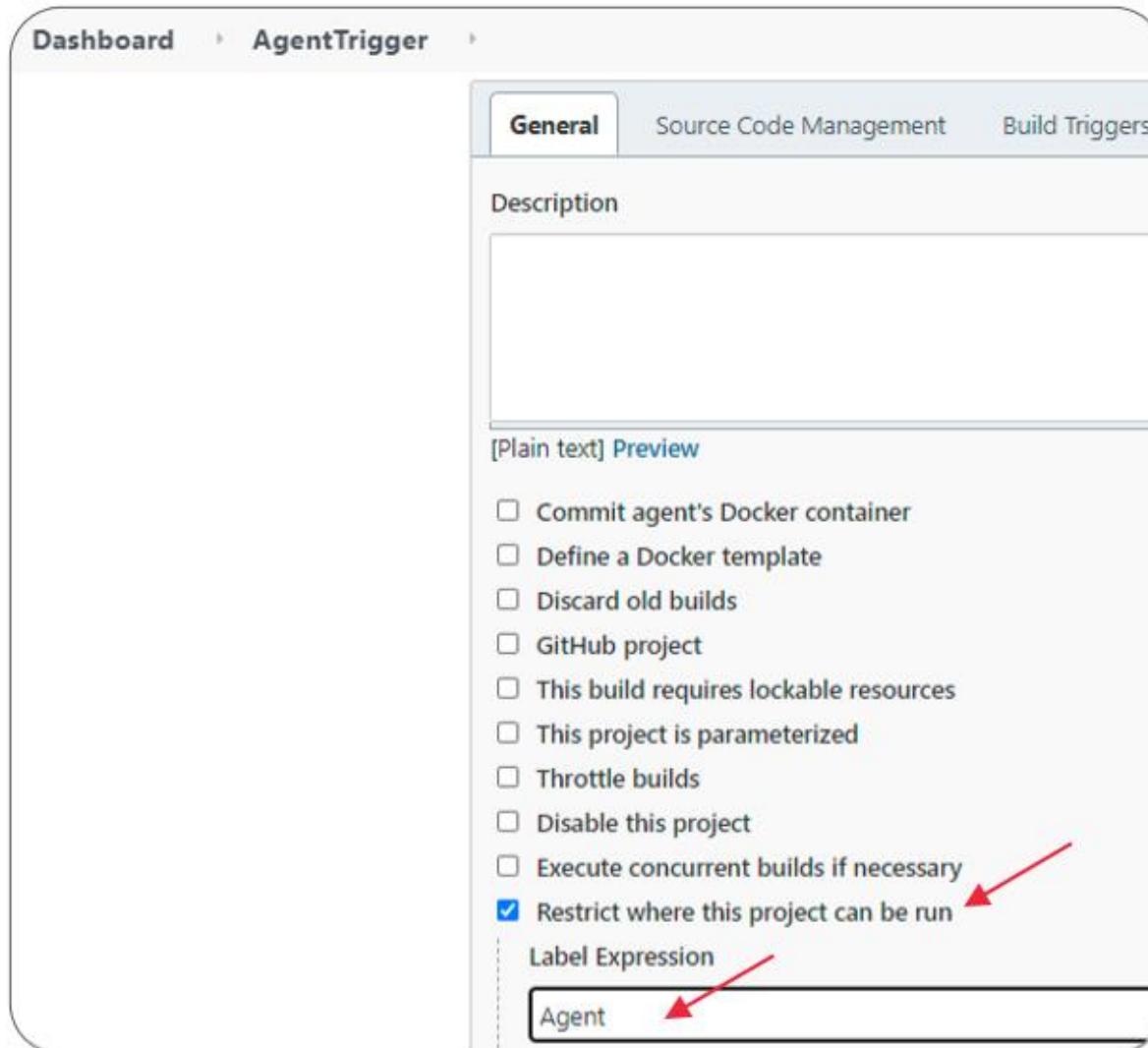
- Name:** Agent (highlighted by a red arrow)
- Description:** This is my Jenkins Agent (highlighted by a red arrow)
- # of executors:** 1 (highlighted by a red arrow)
- Remote root directory:** /home/jenkins (highlighted by a red arrow)
- Labels:** AGENT-label (highlighted by a red arrow)

Right Panel (Advanced Configuration):

- Usage:** Use this node as much as possible (highlighted by a red arrow)
- Launch method:** Launch agents via SSH (highlighted by a red arrow)
- Host:** 172.17.0.2 (highlighted by a red arrow)
- Credentials:** jenkins (Jenkins Master Private Key to add multiple agents) (highlighted by a red arrow)
- Add** button (highlighted by a red arrow)
- Host Key Verification Strategy:** Known hosts file Verification Strategy
- Availability:** Keep this agent online as much as possible

A large red arrow points from the "Save" button at the bottom right of the right panel towards the bottom center of the image.

Create a New Project as AgentTrigger



- Add build script as: df –ha
- Save and build manually

Console Output

Jenkins

Dashboard > AgentTrigger > #1

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#1'](#)

Console Output

Started by user admin
Running as SYSTEM
Building remotely on Agent (AGENT-label) in workspace /home/jenkins/workspace/AgentTrigger
[AgentTrigger] \$ /bin/sh -xe /tmp/jenkins18118038230484866561.sh

```
+ df -ha
Filesystem      Size  Used Avail Use% Mounted on
overlay         7.7G  3.9G  3.8G  51% /
proc              0     0     0   -  /proc
tmpfs            64M    0    64M  0%  /dev
devpts            0     0     0   -  /dev/pts
sysfs             0     0     0   -  /sys
tmpfs            490M    0   490M  0%  /sys/fs/cgroup
cgroup            0     0     0   -  /sys/fs/cgroup/systemd
cgroup            0     0     0   -  /sys/fs/cgroup/cpu,cpuacct
cgroup            0     0     0   -  /sys/fs/cgroup/pids
```

Benefits of Jenkins Distributed Architecture

- Higher Performance
- High Availability
- Failover Mechanism
- Enhanced Security
- Rollback Mechanism from Machine Failure

Thank You