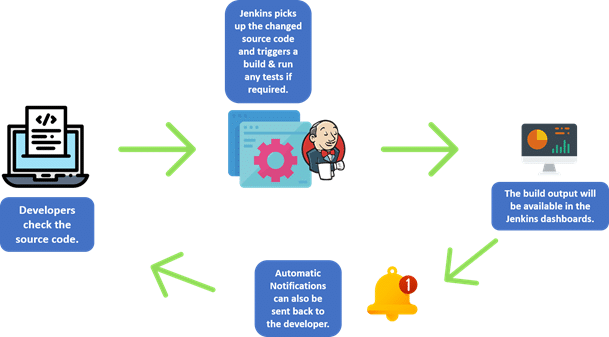
**What is Jenkins?**

Jenkins is a software that allows users to do continuous integration on their application/software life cycle. It gets installed on the server where the central build takes place.

Now, let’s understand its workflow.

## Workflow Management

The below image depicts the workflow of Jenkins.

  
To manage Jenkins, we have to click on the **Manage**tab on the left-hand side.

### Configuring the System

* Jenkins can be used to manage paths to various tools to use in builds.
* It dynamically adds the config fields after the plugins are installed.

### Managing Plugins

* Plugins can be removed, updated, and installed through the Manage plugin screen.

***Practice from learning from these***[***Jenkins Interview Questions***](https://intellipaat.com/blog/interview-question/jenkins-interview-questions-answers/)***.***

### System Information

* List of all the current java properties and system environment variables
  + System log: To view Jenkins log in real-time
  + Script console: To run Groovy scripts on the server
  + Manage nodes: To configure the number of builds required
  + Shutdown: To prepare to shut down and to prevent any new build from being started. After all current builds are finished, Jenkins will shut down cleanly

### Unit Testing

Jenkins provides a host of plugins for unit testing other technologies, such as MSTest for .NET unit testing.

**Testing a Unit in Jenkins**:

**Step 1**: Open the dashboard, choose an existing project, and click on Configure

**Step 2**: Browse toward Add a build step and invoke Ant

**Step 3**: Click on Advanced

**Step 4**: Enter the location of build.xml in the build file section

**Step 5**: Click on the Add post-build option and then on the Publish JUnit test result report

**Step 6**: Ensure that the report is in the folder of the project workspace. Basically, **\*.xml** tells Jenkins to pick up the resulting .xml files that are produced by running the JUnit test cases. Later, click on Save

**Step 7:** Now, click on Build and check logs to see if it is successful

### Automated Testing

It helps automate a part of the development process for continuous integration, with a focus on continuous delivery.

Without automated testing, we would have to carry forth some artifacts that are not viable. Hence, using software like [***Selenium***](https://intellipaat.com/blog/tutorial/selenium-tutorial/introduction/) helps us a lot by running automated tests.

Let’s check an example:

**Step 1:** Go to plugins and choose Selenium plugins, and click it to install

**Step 2:** Go to Configure system and select Selenium jar, and then click on Save

**Step 3:** Go to the dashboard and select the config option for the project at hand

**Step 4:** Click on Add build step and choose SeleniumHQ htmlSuite Run

**Step 5:** Add the required details and click on Save, execute and build

This way, the test is executed and a report is built.

**Learn more crucial DevOps concepts from this**[**DevOps Training in Sydney**](https://intellipaat.com/devops-certification-training-sydney/)**!**

### Notifications

Jenkins comes with a feature to add email notifications to the build project.

* Go to Manage Jenkins → Configure system
* In the email notification space, enter the required STMP server and use email suffixes
* Configure recipients so that they would receive notifications about broken or unstable builds
* Notification plugins, such as Tikal Knowledge, allow job status notification for JSON and XML formats
* **Options:**
  + **Format:** Either JSON or XML
  + **Protocol:** TCP, UDP, or HTTP
  + **Event:** A job event that triggers a notification
  + **URL:** The destination to send notifications to
  + **Timeout:** A default timeout at 30 sec

### Code Analysis

Jenkins provides utilities for static code analysis. Some tools are as follows: Checkstyle, FindBugs, PMD, etc.

It provides details as given below:

* Total warning in a job
* Shows a new or fixed warning of a build
* Provides trend reports showing warnings per build
* Warnings per module, package, or category
* Detailed reports of found warnings

### Automated Deployment

There are plugins available to transfer the build files to the server, e.g., ‘Deploy to container’ plugin. To use this plugin, we have to head to:

Manage Jenkins → Manage Plugins → Available → Deploy to container Plugin → install the plugin → restart Jenkins server

This ‘Deploy to container’ plugin takes the war/ear file and carries that to the running remote application build.

In the second step, we have to go to our build and click on the Configure option. After that, we will choose ‘Deploy war/ear to a container.’

In the third and final step, we will enter the required details of the server in ‘Deploy war/ear to a container’ and press Save.

**Prepare yourself for the best job by going through these**[***Top DevOps Interview Questions and Answers***](https://intellipaat.com/blog/interview-question/devops-interview-questions/)**!**

### Server Maintenance

Now, let’s check out a few commands that are used for server maintenance.

**Commands in Jenkins (URLs):**

* To shutdown Jenkins:

http://localhost:8080/jenkins/exit

* To restart Jenkins:

http://localhost:8080/jenkins/restart

* To reload the configuration:

http://localhost:8080/jenkins/reload

### Backing up Jenkins Home

* Go to ‘Configure system’ in ‘Manage Jenkins’
* Select a partition that has the most free space as Jenkins might fail if the drive becomes full
* Perform automated clean-ups to avoid this

***If you are planning to get certified in DevOps, then enroll in Intellipaat’s***[***DevOps Course in Bangalore***](https://intellipaat.com/devops-certification-training-bangalore/)***now!***

### Building a Pipeline

Let’s check out the steps for building a pipeline:

**Step 1**: Go to Manage plugin and install Build pipeline plugin

**Step 2**: To see a build pipeline, click on the plus (+) icon on the dashboard

**Step 3**: Enter any name for the view and click on the view name, and then choose the Build pipeline view

**Step 4**: Accept the default settings and be sure to add the name of the project at hand

Now, a view of the entire pipeline with status will be visible.

### Managing Plugins

* To uninstall plugins, go to Manage plugins and click on the Installed tab, and then on Uninstall for the plugin that we want to uninstall. Ensure to restart Jenkins for changes to take effect
* In the case of requiring to install an older version of the plugin, download the same from the official site and click on the Upload option to do it manually

### Setting up Security Options

In Jenkins, we have secure config options for different users in place. To enable them:

* Click on Manage Jenkins and then on Configure global security
* Set parameters in the Enable security section
* Add users and go to the Manage users option to provide permissions
* To set authorizations, go to Configure global security and click on Matrix-based security
* Add and give the required permissions to the users

### Backing up Plugins

The Backup plugin adds a new Backup manager item in the Manage Jenkins page. This setting allows:

* To tweak backup settings via Setup
* To back up Jenkins config
* To restore config from a previous backup

Alternatively, we can use SCM (Sync Configuration plugin) or ThinBackup for global and job configurations.

### Remote Testing

Selenium tests can be run on remote slave machines via master–slave and Selenium suite plugin installation.

**Step 1:** Go to the Master Jenkins server and manage nodes

**Step 2:** Click on Configure for the slave machine

**Step 3:** Set the launch method as ‘Launch slave agents via Java Web Start’

**Step 4:** Open a browser instance of Master Jenkins on the slave machine, manage nodes, and then select DXBMEM30

**Step 5:** Scroll down, select the launch option, and hit Run

**Step 6:** Configure tests to run on the slave

**Step 7:** Make sure that the Selenium part of the job is configured. Make sure that the Sample.html file and the selenium-server.jar