



Introducing Jenkins

DevOps Practitioner

transforming performance
through learning

Outline

- **CI/CD**
 - Principles
 - Practices of CI/CD
 - Tools
- **Introducing Jenkins**
 - What is Jenkins
 - Why is it popular
 - Distributed builds
- **Installing Jenkins on Centos**
 - The dashboard
 - Create a new project
 - Compile and Test

Objective

- **By the end of this session we should be able to**
 - Discuss the CI/CD best practices
 - Be able to explain what Jenkins is
 - Install Jenkins

CI/CD Principles

- **Continuous Integration**
 - Developers work on a new feature
 - It is added to the main code base automatically
 - TDD used to verify feature works and doesn't break anything
 - Automating the build process
- **Continuous Delivery**
 - Building code in a repeatable way
 - Creating artefacts for deployment
 - Verify the build and deploy if required
 - Code releases can be minute by minute, rather than months at a time

4

Continuous Integration – source control (specifically using git)

Developers work on a new feature – on the same time.

Code can be submitted when is done to repo. Repo monitors merging conflicts of a code from two people.

Testing – following TDD through out.

Automating the build process – when code is submitted, its auto tested and if it does not have conflicts – built.

Continuous Delivery – process to make sure, any time the commit happens to source control, it then builds a code.

Building code in a repeatable way –

Creating artefacts for deployment – always have deployment artefacts that is ready to go to deployment

Verify the build and deploy if required - when the code is build, it goes to deployment stage and goes through other specific tests: to make sure, that is safe to go into production

Code releases can be minute by minute, rather than months at a time

Best Practices – Continuous Integration

- **Maintain a Single Source Repository**
 - No more emailing code around!
 - Should contain everything required for the build including test and compilation scripts
- **Everyone Commits To the Mainline Every Day**
 - Commit each change when it works

5

Adapted From: <http://martinfowler.com/articles/continuousIntegration.html>

Maintain a Single Source Repository

One source repo for the project (not one repo for one person).

After you complete the work (and it work without errors), you push it to repo.

Best Practices – Building and Testing

- **Automate the build**
 - Don't allow human error to miss steps or get the wrong libraries!
- **Make the build self-testing**
 - Follow TDD principles, everything that can be tested should be
- **Test in a Clone of the Production Environment**
 - Test environment should mimic the production
- **Every Commit Should Build the Mainline on an Integration Machine**
 - This is where CI tools come in handy
 - Each commit should trigger a build – fail fast again
- **Keep the Build Fast**
 - Nothing is more dull than watching the build process
 - If the build is slow then people won't commit code until they have to

6

We want to automate every single step in the build process (locally with Maven, server based solution with Jenkins).

Make the build self-testing

Test environment should mimic the production – test it on the different machines

Every commit should build that branch in the server machine.

Every new code is submitted – should be build automatically

Build fast!!!

Best Practices - Culture

- **Fix Broken Builds Immediately**
 - Everyone should be involved with fixing problems
- **Make it Easy for Anyone to Get the Latest Executable**
 - Make it possible to allow people to demo the code
 - Everyone is using the same current copy
- **Everyone can see what's happening**
 - Communication!
 - Big Visible Charts!
 - This is the CI/CD version of agile best practices. Everyone should be informed at all times
- **Automate Deployment**
 - Set off deployment scripts
 - Repeatable, testable ideas

7

Fix Broken Builds Immediately – don't build a new code on a broken code

Don't hide the latest executable code from your team or boss (make sure that is in the single place and people knows how to get it)

Everyone needs to be informed what is going with the project on exact time (charts, board in the room etc).

Automate as much as humanly possible deployment process

CI/CD Tools

- **Many tools available. Needs to be able to**
 - Notice changes in source code
 - Get a copy of the source code
 - Build it (resolving dependencies)
 - Test it
 - Start deployment of code (often using docker, chef, puppet etc.)
- **Available tools:**
 - Jenkins / Hudson – Free! Open Source!
 - Bamboo – Atlassian's build engine
 - JFrog
 - Cruise Control
 - Others?

8

CI / CD tools are a collective group covering the source control, the package managers and the build tool itself. We have been using git and maven for the first two. We will be investigating Jenkins for the build process.

Introducing Jenkins

- **First released in 2005 as the Hudson project**
 - Created by Sun Microsystems
 - Oracle bought the company and so had control of the Hudson name
 - Name changed to Jenkins for the open source project
 - Both the Hudson and Jenkins project are still active but development has diverged
- **Created as a CI / CD tool**
 - Aim was to be easy to install and use
 - Plugins to connect to source control
 - Builds projects using the project setup – so can work with any language
 - Public / company facing dashboard lets everyone know the status of the current build



9

Open source version of Hudson was called Jenkins.

Jenkins project is more active than Hudson. There are a lot of references from Jenkins to Hudson

It gives a public dashboard – everyone can see what is going on with the project

What does Jenkins do

- **Jenkins is a one stop shop for testing and building code**
 - It can even deploy code onto servers if they already exist
- **Usual workflow:**
 - Developer checks some new code into git
 - Git informs all watchers that it has updated
 - The Jenkins git plugin starts the build process
 - Clones the project
 - Compiles, runs tests
 - Updates the dashboard with any feedback
 - Start the post-build steps or actions to deploy

Why is it popular

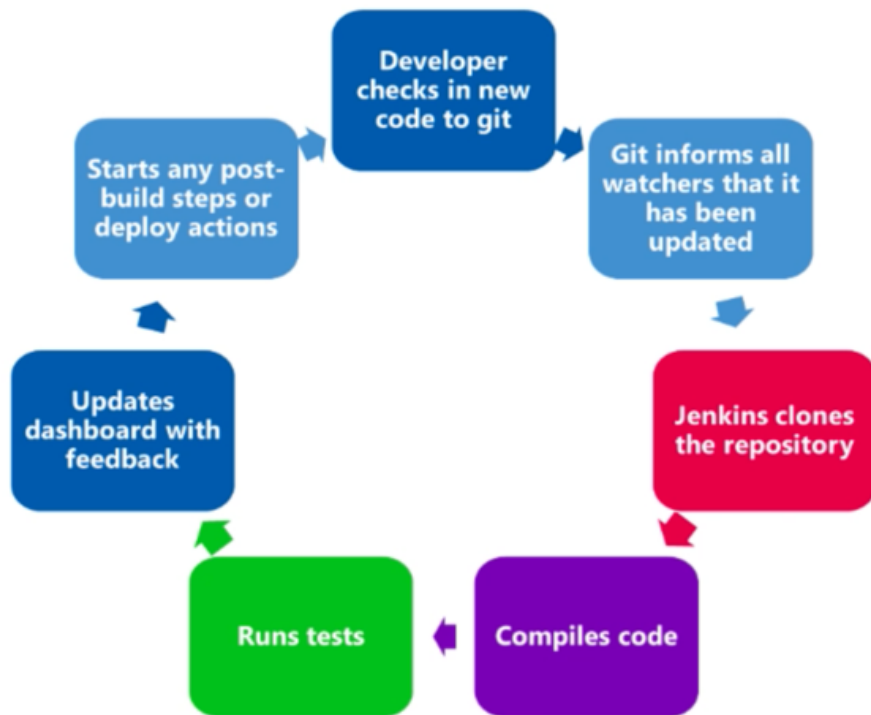
- **Allows companies to follow some of the best practices for CI/CD**
 - Automate the build
 - Make the build self testing
 - Keep the build fast
 - Make sure everyone can see the results of the build
 - Automate deployment
- **Easy installation and config**
 - Can be done all through the GUI – no need for more XML files
- **Many plugins available for any combination of source control, compiling, testing and deployment**
 - Still a very active project!
 - IDE Integration, orchestration tools, style checkers

11

Want to avoid integration hell: <http://c2.com/cgi/wiki?IntegrationHell>

Publically visible dashboards: https://www.google.co.uk/search?ie=UTF-8&q=%22Dashboard+%5BJenkins%5D%22&gws_rd=cr,ssl&ei=XLB2VZ6KEfCU7QbB_4PwBQ

Jenkins Flow



Distributed Jenkins Servers

- **Jenkins uses agents to scale systems**
 - By default there is one agent which will build using one core
 - May need to build for different architectures
 - Windows / Mac / Linux
 - Android / iOS / Windows Phone
 - Master / Slave servers
 - Can exist anywhere
- **Single report can be generated over all the builds**

Installing Jenkins on Centos 6 / Amazon Linux EC2

- **To install Jenkins we need to add a new repository**
 - Then we can install it using yum

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
    http://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import \
    http://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum install jenkins
```

- **To start the service:**

```
sudo service jenkins start
```

- **Jenkins requires Java 1.8 or above to run**

You may need to install wget:

```
sudo yum install -y wget
```

Installing Jenkins on other systems

- **Ubuntu**

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

- **Using Docker:**

```
$ docker run -d -p 8080:8080 jenkins
```

- **Windows:**

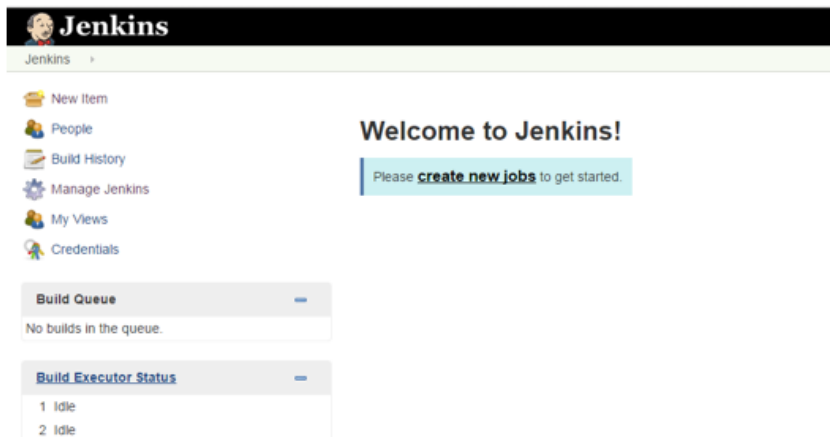
- Uses the installer

```
https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+as+a+Windows+service
```

Exercise

- **Install and configure a Jenkins server on a virtual machine**
 - Clone / Create a webserver project

The Dashboard



The screenshot shows the Jenkins dashboard interface. At the top is the Jenkins logo and name. Below it is a navigation menu with links: New Item, People, Build History, Manage Jenkins, My Views, and Credentials. The main area features a 'Welcome to Jenkins!' message with a prompt to 'create new jobs'. Below this are two expandable sections: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing two idle executors.

Jenkins

Jenkins >

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

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue

No builds in the queue.

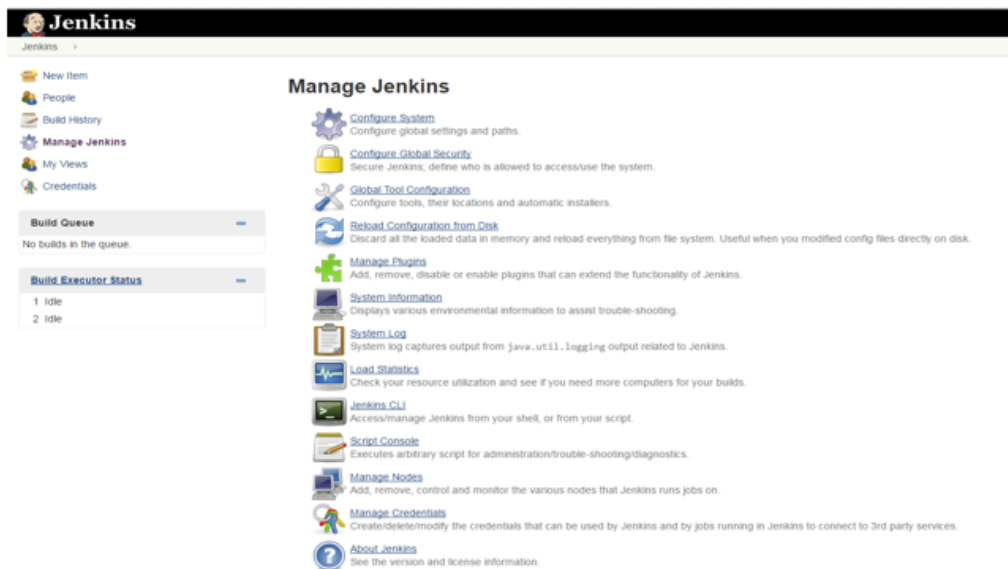
Build Executor Status

1	Idle
2	Idle

- **Main control centre for Jenkins**
- **Configure projects from here**

Managing Jenkins

- The Manage Jenkins link on the home page contains all the admin side of Jenkins



The screenshot displays the Jenkins web interface. At the top, the Jenkins logo and name are visible. Below the header, a sidebar on the left contains navigation links: New Item, People, Build History, Manage Jenkins (highlighted), My Views, and Credentials. The main content area is titled 'Manage Jenkins' and lists various administrative tasks, each with an icon and a brief description:

- Configure System**: Configure global settings and paths.
- Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
- Global Tool Configuration**: Configure tools, their locations and automatic installers.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from java.util.Logging output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**: Access/manage Jenkins from your shell, or from your script.
- Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.
- About Jenkins**: See the version and license information.


On the left sidebar, the 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows two executors, both in an 'Idle' state.


Create a new project


- Click the new item link
 - Different project styles are available
 - We will be using Maven projects


Enter an item name


Required field


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


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

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

 **External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See [the documentation for more details](#).

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









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

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

OK

Compilation and Testing

- We can manually trigger builds
- ... Or create triggers based on source control commits
- ... Or based on time

-  Back to Dashboard
-  Status
-  Changes
-  Workspace
-  Build Now
-  Delete Maven project
-  Configure
-  Modules

Source Code Management

- ☒ None
- ☐ CVS
- ☐ CVS Projectset
- ☐ Subversion

Build Triggers

- ☒ Build whenever a SNAPSHOT dependency is built
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Poll SCM

To read more about Jenkins

Books:

- Learning Continuous Integration with Jenkins - Second Edition (2017. link: <https://www.packtpub.com/virtualization-and-cloud/learning-continuous-integration-jenkins-second-edition>)
- Jenkins 2.x Continuous Integration Cookbook (2017. link: <https://www.safaribooksonline.com/library/view/jenkins-2x-continuous/9781788297943/>)
- Continuous Integration, Delivery, and Deployment (2017. link: <https://www.safaribooksonline.com/library/view/continuous-integration-delivery/9781787286610/>)

Tutorials:

- <https://www.tutorialspoint.com/jenkins/>
- <https://uk.ctl.io/knowledge-base/cloud-application-manager/tutorials/jenkins-ci-cd-tutorial/>
- <http://www.vogella.com/tutorials/Jenkins/article.html>

Summary

- **CI/CD**
 - Principles
 - Practices of CI/CD
 - Tools
- **Introducing Jenkins**
 - What is Jenkins
 - Why is it popular
 - Distributed builds
- **Installing Jenkins on Centos**
 - The dashboard
 - Create a new project
 - Compile and Test