Jenkins:

Continuous integration with Jenkins on Amazon EC2

What is continuous integration

Example: Joe and Jane working on a Project. They each individually implement few classes and code them up and unit tested and made sure everything working as expected. Thinking that program is going to be robust.

Once they are done unit testing integrate them. Everything breaks. (code fails to compile or bugs). This situation is going to be known as integration hell

Integration hell is extremely risky for a project.

Difficult to determine how long will take to resolve the integration problems. May vastly exceed budget and schedule.

To avoid these situations Continuous Integration program came up (Originated from eXtreme Programming(XP). To avoid integration hell, Integrate continuously throughout the entire project.

Mitigates risks associated with integration software.

Rather than developed and waiting to Integrate weeks and months integrate early and integrate often (i.e. on every change).

Every time developer pushes the new change to the Repository, project code checked out and build and compiled and all of the testes should be run.

Continuous Integration server

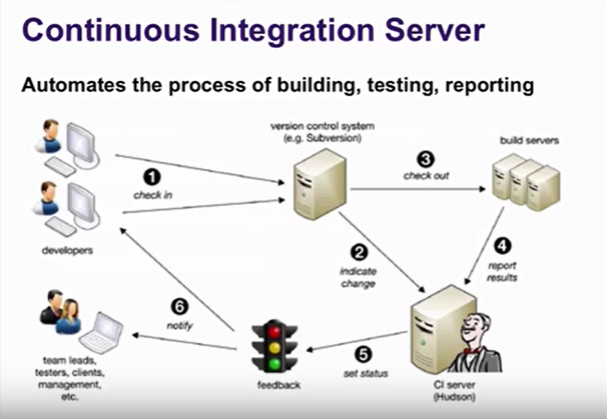
Automates the process of building testing and reporting.

1. Developer pushes the changes into Repository (Ex: GitHub)

2. CI server will be notified (either CI server pull the repository or Repository call the CI server to notify the changes).

3. When the CI server sees the changes has been pushed to the repository it will clone the repository on the server or may actually instruct the build servers to check out the code and build the project. If the build fails send a notification to the Project Team. If the build is successful it will run all the project tests and if any test fails again send notifications to the team

4. Finally CI server generates various reports. (ex: Line coverage of Project tests)



Benefits of the CI server:

Developers might forget to run the tests. (Don't break the build)

It might take too long to run the tests

We might need to test the code in various environments.

Different architectures (32-bit, 64-bit, ARM PowerPC)

Different platforms (Windows, Linux, Mac, Solaris)

CI server can use them (other platforms) as build slaves to run the changes on different platforms. When the CI server detects the change in Repository CI server will instruct the build slaves to check out the latest version of the code to build and test it.

CI server can provide some reports useful insights to the team.

Can track the metrics like line coverage (percentage of lines executed by program tests.)

Can run all sorts of utilities on our code. (Check style and find bugs)

Can deploy automatically (deploy a web project to the test or staging server automatically)

Popular CI servers: Jenkins, Hudson, CruiseControl, TeamCity etc

Tasks:

* Setup an Amazon EC2 instance
* Install Jenkins
* Configure GitHub to notify Jenkins of Changes
* Configure Jenkins
* Clone Repository when notification received
* Build the Project
* Run the Project tests
* Generate Line Coverage Reports.

Step 1: Create EC2 instance (Ubuntu Server 13.10)

Switch to Terminal window

$ ssh -i ~/Downloads/Wordpress-Linux.pem Ubuntu@ Hostname

$ chmod 600 ~/Downloads/Wordpress-Linux.pem

$ exit

make it command ssh -i ~/Downloads/Wordpress-Linux.pem Ubuntu@ Hostname shorter

$ vim ~/.ssh/config

Inside the editor

Host ec2

Hostname ec2-50-17-10-64.compute-1.amazonaws.com

User Ubuntu

IdentityFile ~/.ssh/ Wordpress-Linux.pem

$ mv ~/Downloads/Wordpress-Linux.pem ~/.ssh

$ ssh ec2 (Shorter form of the ssh -i ~/Downloads/Wordpress-Linux.pem Ubuntu@ Hostname)

Fixing Locales in Ubuntu 13.04 on Amazon EC2

$ sudo apt-get install language-pack-en

### Installing Jenkins

### Add the Repository for the Jenkins package. In order to do that we need to install the public key for the Jenkins Repository to tell Ubuntu that this is a trusted Repository.

$ wget -q -O - <http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key> | sudo apt-key add -

Below command to tell Ubuntu where the package Repository is

$ echo "deb <http://pkg.jenkins-ci.org/debian> binary/" | sudo tee -a /etc/apt/sources.list.d/jenkins.list

$ sudo apt-get update

$ sudo apt-get install jenkins

Once Jenkins is installed we can confirm that Jenkins is running by typing the below command

$ ps -ef | grep jenkins

$ sudo apt-get install 'apache2'

apache2 modules:

$ sudo a2enmod proxy

$ sudo a2enmod proxy\_http

Configure the apache2 actually proxy the requests from port 80 to 8080

$ sudo vim /etc/apache2/sites-available/jenkins.conf

<VirtualHost \*:80>

ServerName ec2-50-17-10-64.compute1.amazonaws.com

ProxyRequests Off

<Proxy \*>

Order deny,allow

Allow from all

</Proxy>

ProxyPreserveHost on

ProxyPass / <http://localhost:8080/>

</VirtualHost>

$ sudo a2ensite jenkins (Enabling site jenkins)

$ sudo service apache2 reload (service apache2 reload)

Installing Jenkins on Ubuntu server link:

https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu

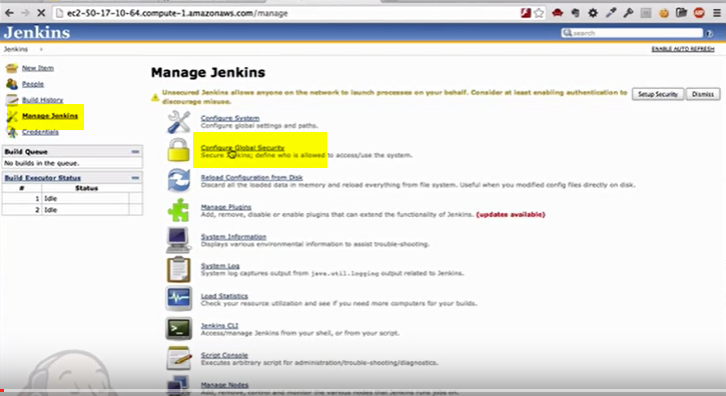
Configure Jenkins:

By default, Jenkins doesn't security enabled.

Launch the site Jenkins.

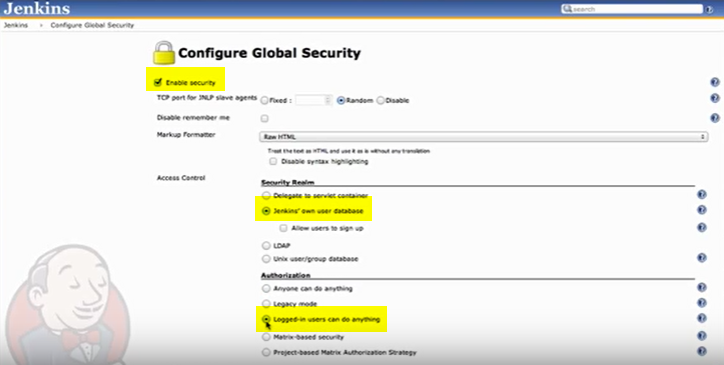
1. Click on Manage Jenkins

2. Configure Global Security



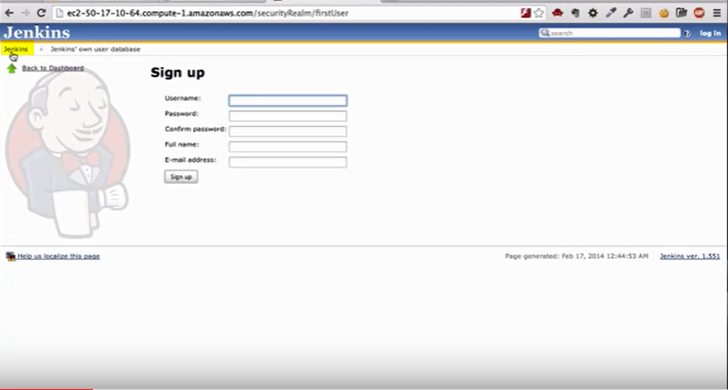
3. Enable Security (Please look at the below screenshot for the details)

and click on save button.



We did disable to ability to sign up. Fortunately, Jenkins allows us to sign in first user.

Click on the Jenkins link on the Top left most corner and then signup.

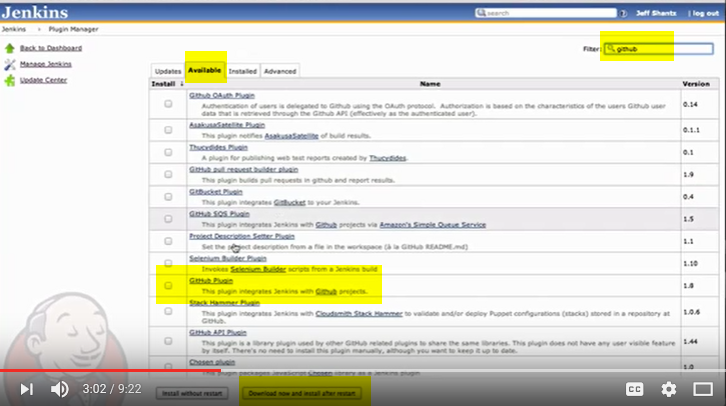


Want to setup global configurations for Jenkins.

Before we do that go to the manage Jenkins and click on mange plugins.

Click on all the checkboxes and click on Install without restart.

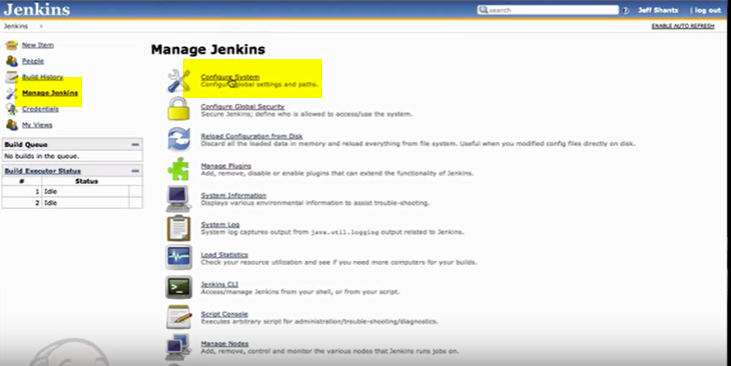
Then go back to manage Plugins. Follow the below screenshot



then restart the Jenkins.

Once it is restarted you can create the accounts for other users by clicking on Manage Jenkins and then manage users tab.

Then go back to manage Jenkins and click on Configure System.



Switch back Ec2 instance

Installing Java/Maven/Git

$ sudo add-apt-repository ppa:webupd8team/java

$ sudo apt-get update

$ sudo apt-get install oracle-java7-installer maven git-core

Verify and make sure softwares are installed

$ javac -version

find the java home by typing the below command

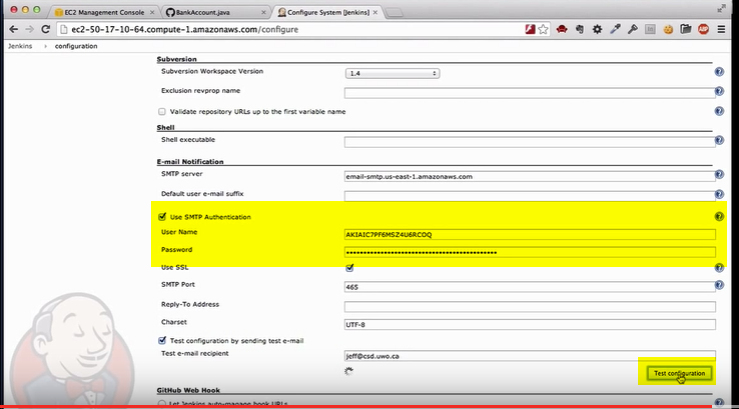
$ readlink -f /usr/bin/javac

/usr/lib/jvm/java-7-oracle/bin/javac (copy everyting before bin and paste in Jenkins site)

$ mvn -version

$ readlink -f /usr/bin/mvn

/usr/share/maven/bin/mvn (copy everyting before bin and paste in Jenkins site)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A **Project Object Model** or **POM** is the fundamental unit of work in **Maven**. It is an XML file that contains information about the project and configuration details used by **Maven** to build the project. It contains default values for most projects.

GIT

git config --global user.email "anil.devops@gmail.com"

git config --global user.name "anil"

git init

git add .

git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: BankAccount.java

new file: README.md

new file: TestBankAccount.java

new file: pom.xml

new file: pom2.xml

git commit -m "testing uploading java files"

git push -u origin master

git status

On branch master

nothing to commit, working directory clean

ubuntu@ip-172-31-16-54:~/anil/jeffcode$ git push -u origin master

Username for 'https://github.com': anil.devops@gmail.com

Password for 'https://anil.devops@gmail.com@github.com':

Counting objects: 7, done.

Compressing objects: 100% (7/7), done.

Writing objects: 100% (7/7), 1.59 KiB | 0 bytes/s, done.

Total 7 (delta 0), reused 0 (delta 0)

To https://github.com/anil-devops/test-jeff.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

ubuntu@ip-172-31-16-54:~/anil/jeffcode$