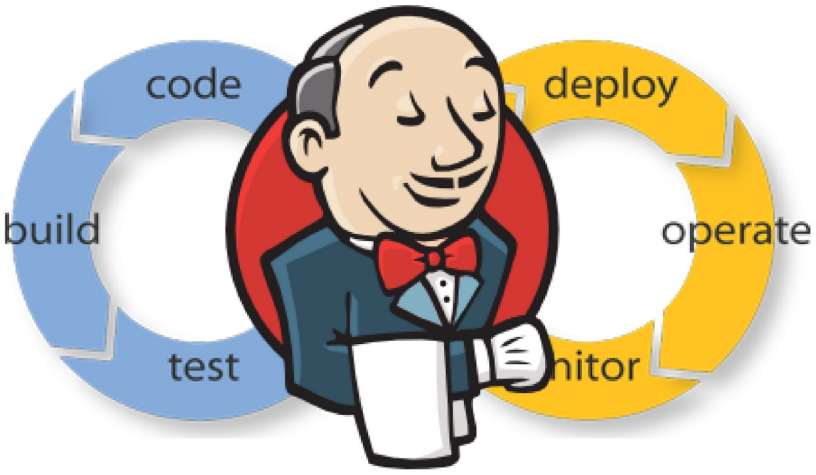


Jenkins Boost Camp

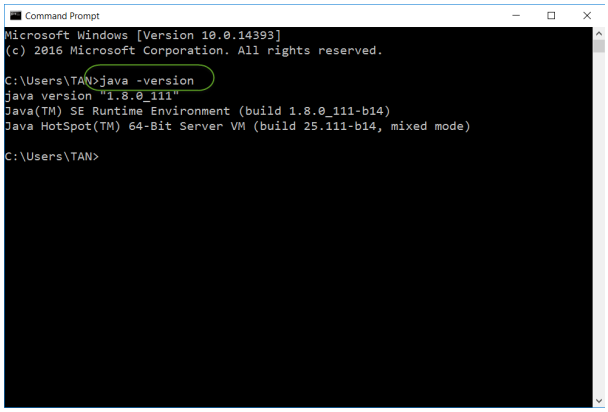
Jenkins Installation

In this chapter, we will jump into how to install and start up Jenkins in 3 environment : Window, Ubuntu and Mac.

In all environment, Jenkins will be installed as service, it mean Jenkins will automatically start up every time you machine power on or restart machine.

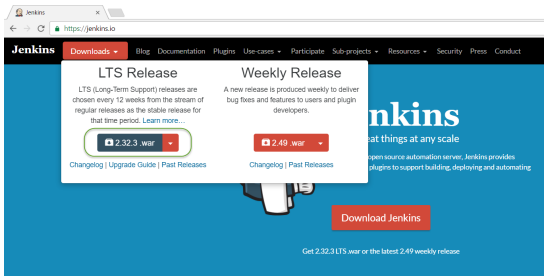
Jenkins on Windows

Check if java is installed : Java should be installed in Windows machine, to check if Java is installed, start a command prompt and typing in ***java -version*** Java version show up as below. Incase java not yet install, please install java before process to installation.

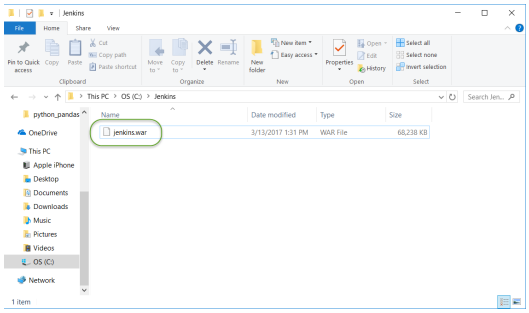


Install process

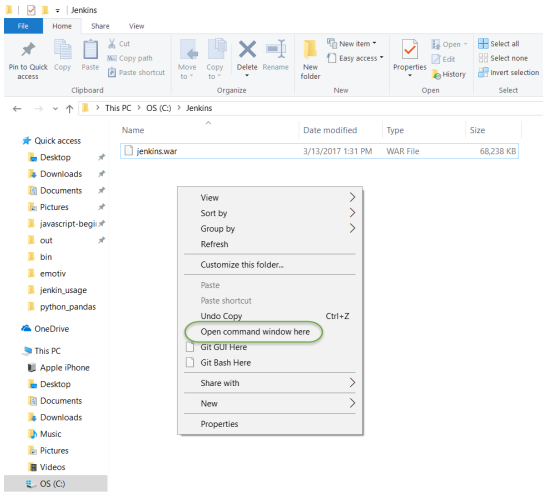
Step 1 : Go to ***https://jenkins.io/*** and download for stable version of Jenkins



Step 2 : Create a new folder for Jenkins (for example ***C:\Jenkins***) and put file ***jenkins.war*** inside this folder

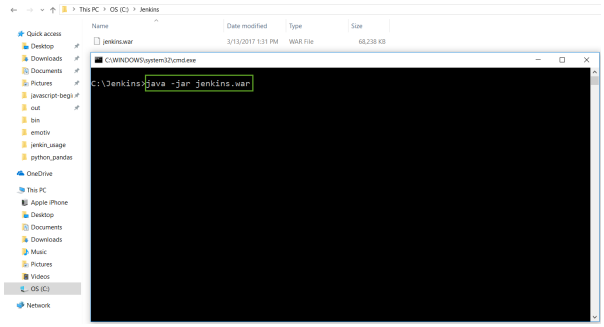


Step 3 : Press to shift and right click at same time to open context menu and select ***Open command window here***

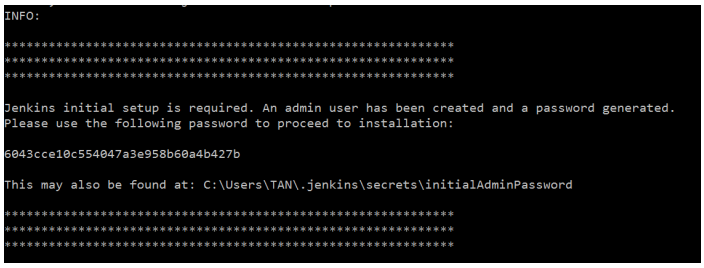


Step 4 : Typing in command prompt following to start installation ***java -jar jenkins.war***

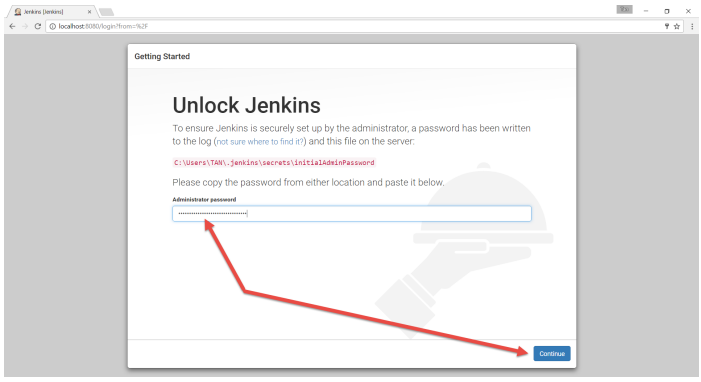
Then wait a while for complete running.



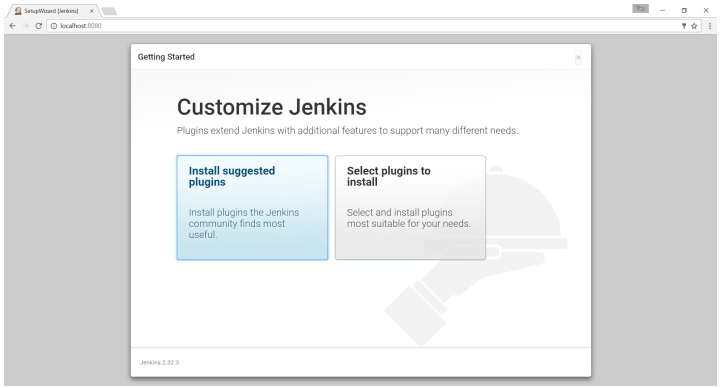
In the output log, you will see default password for first login of admin user, copy this password to notepad and save it for use on next step.



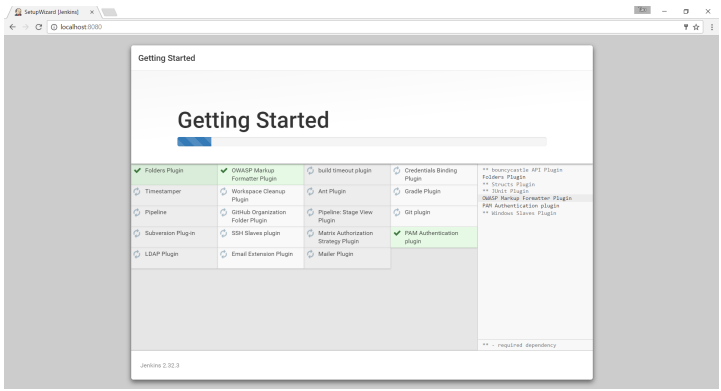
Step 5 : Now open browser and access to ***http://localhost:8080*** and then paste admin password to unlock Jenkins



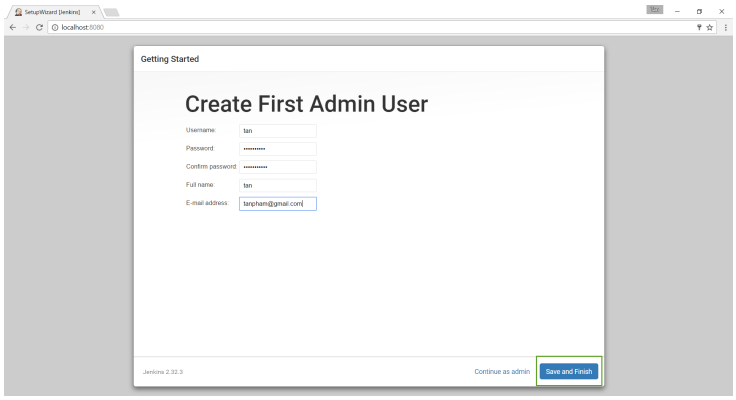
Wait a while and following screen show up, select default option ***Install suggested plugins***



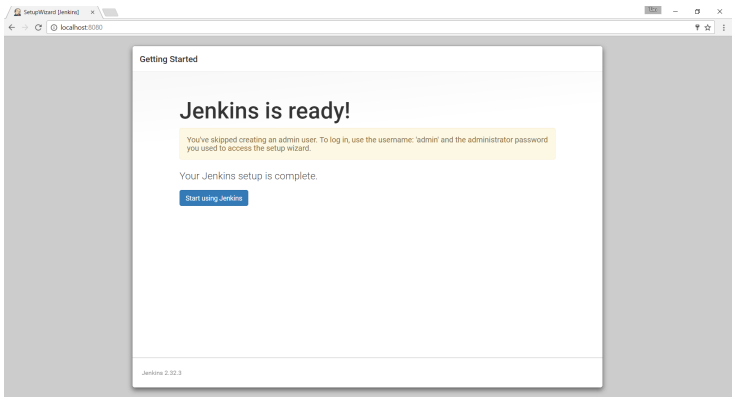
At this step default plugins will be installed.



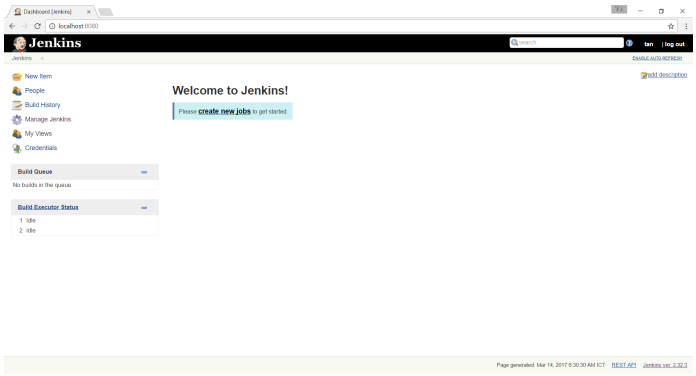
Step 6 : Enter your user name, password for a new admin, then click to ***Save and Finish***



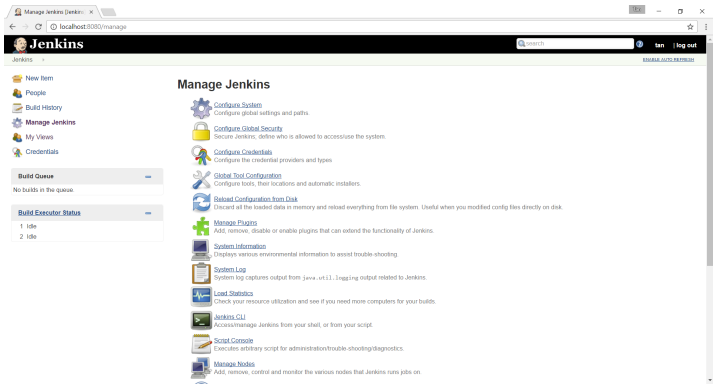
Step 7 : Click to ***Start using Jenkins***, Congregation at this step you already complete for Jenkins installation



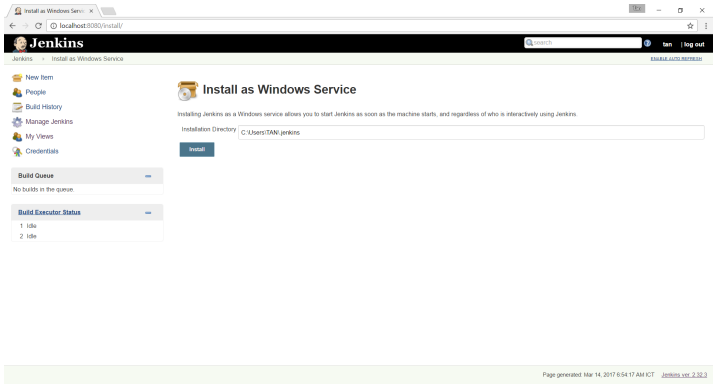
Jenkins auto login by new admin user already created



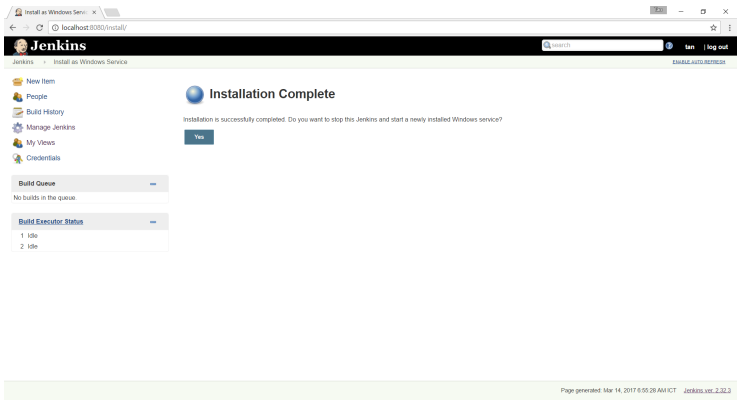
Step 8 : Click to ***Manage Jenkins*** from main screen, scroll down and click to ***Install as Windows Service***



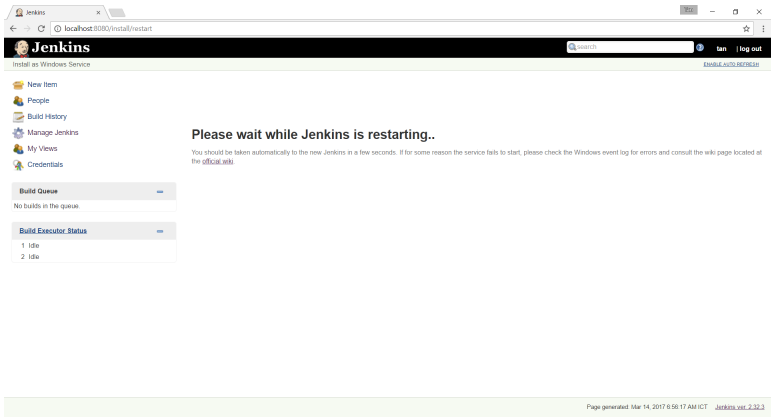
Click to ***Install***

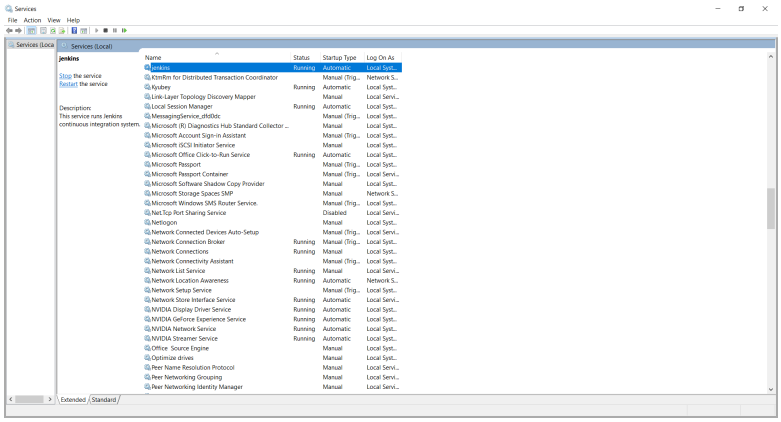


Click to ***Yes***



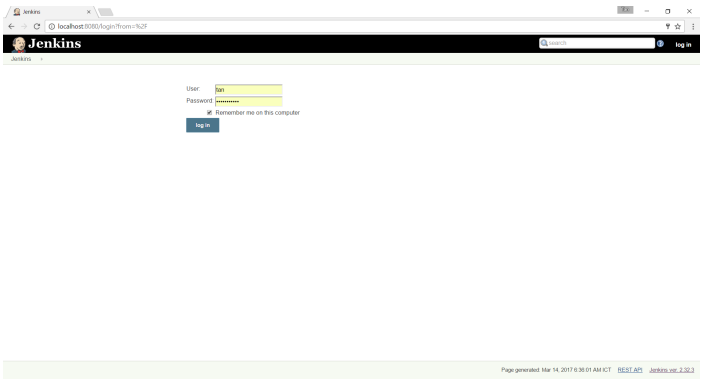
Wait some time and you will see jenkins service is running from windows service manager





That it, You complete install Jenkins on Windows as service. From now on every time Windows power up or restart, Jenkins will ready to access at ***http://localhost:8080***

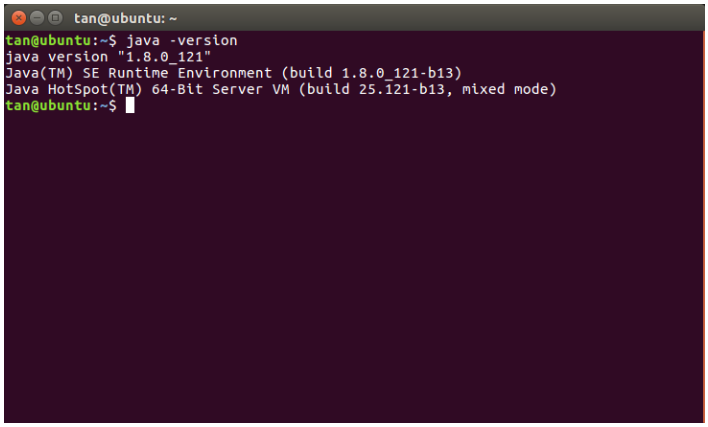
You can try to login Jenkins with above user created.



Jenkins on Ubuntu

Check if java is installed

Java should be installed in Ubuntu machine, to check if Java is installed, start a terminal and typing in ***java -version ,*** If Java already installed, you will see Java version show up.



Install Process

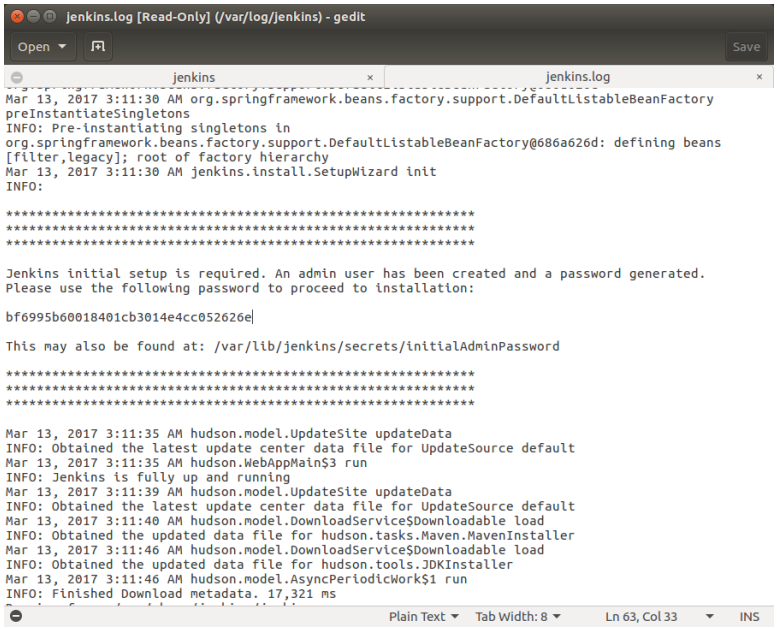
Step 1 : Open terminal and typing in following command

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

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

sudo apt-get update

sudo apt-get install jenkins
```

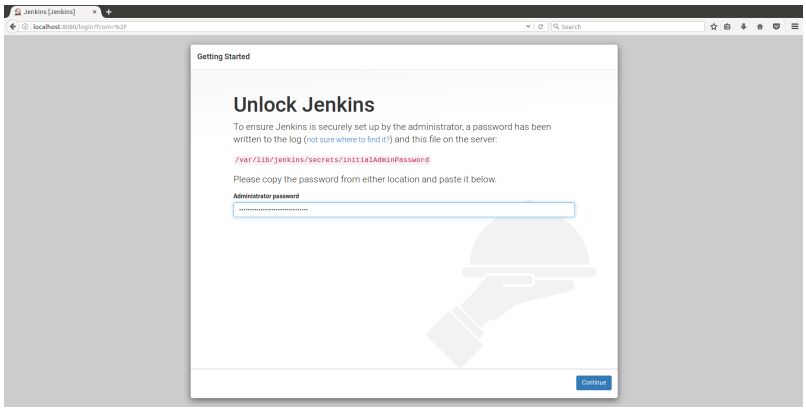
The above code do following job :

- Jenkins will be launched as a daemon up on start. See /etc/init.d/jenkins for more details.
- The 'jenkins' user is created to run this service.
- Log file will be placed in /var/log/jenkins/jenkins.log. Check this file if you are troubleshooting Jenkins.
- /etc/default/jenkins will capture configuration parameters for the launch like e.g JENKINS_HOME
- By default, Jenkins listen on port 8080. Access this port with your browser to start configuration.

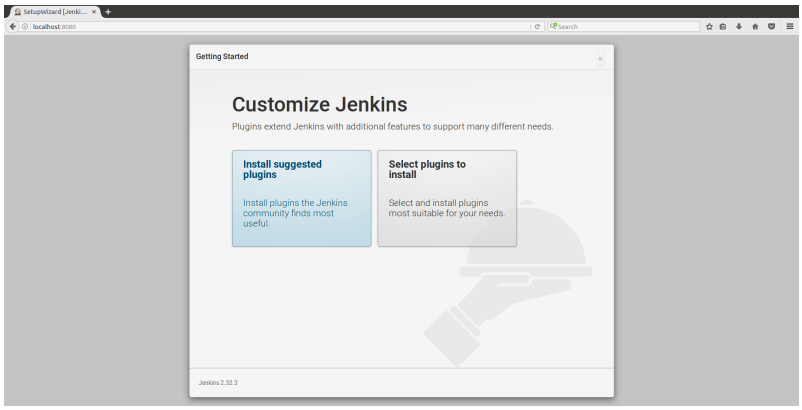
Step 2 : Open the file ***/var/log/jenkins/jenkins.log*** and copy the default password which is automate generated during installation

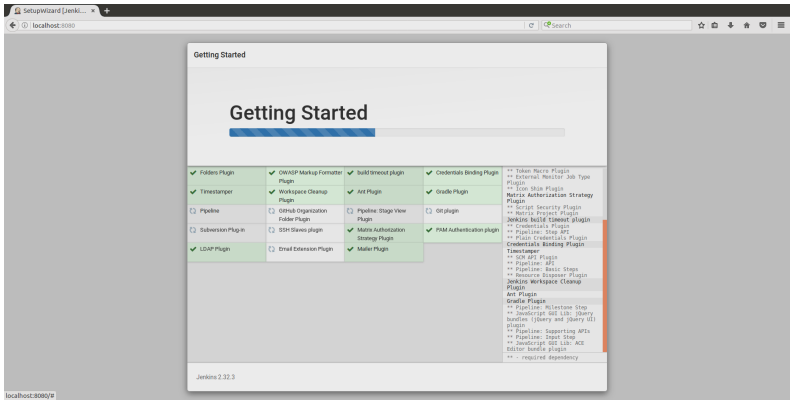


Step 3 : Open browser and access to Jenkins at **http://localhost:8080/** then paste the password above to unlock jenkins. Click to ***Continue*** button

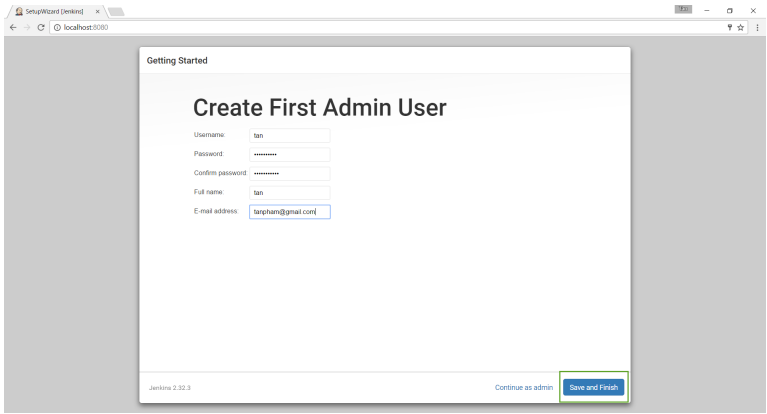


Step 4 : Click on ***Install suggested plugins***

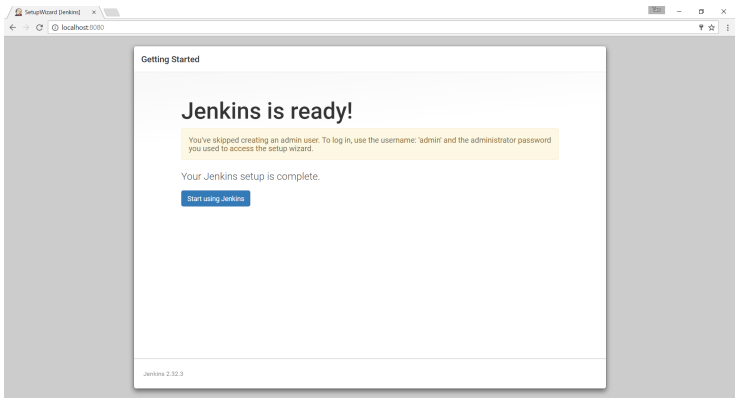




Step 5 : Enter your user name, password for a new admin.



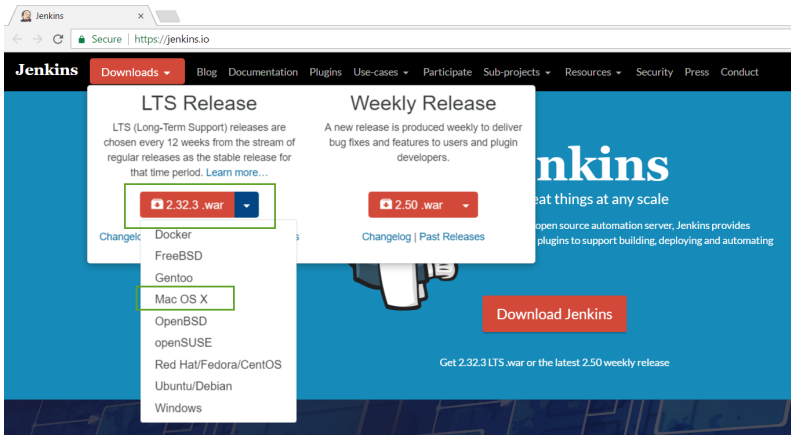
Step 6 : That it, now jenkins ready to use, click to ***Start using Jenkins***



From now Jenkins already installed in Ubuntu as daemon service, so every time you power on machine, Jenkins will start and ready to access at ***http://localhost:8080/***

Jenkins on Mac

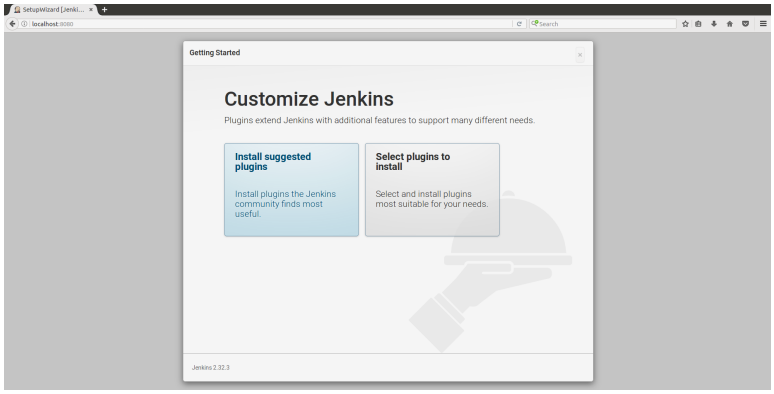
Step 1 : Download installer for Mac from https://jenkins.io/ .Then run the installer file.

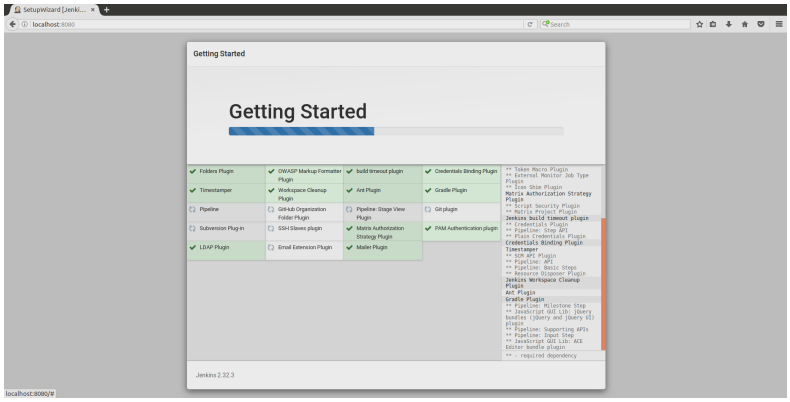


Step 2 : After complete install, you access Jenkins at ***http://localhost:8080/***

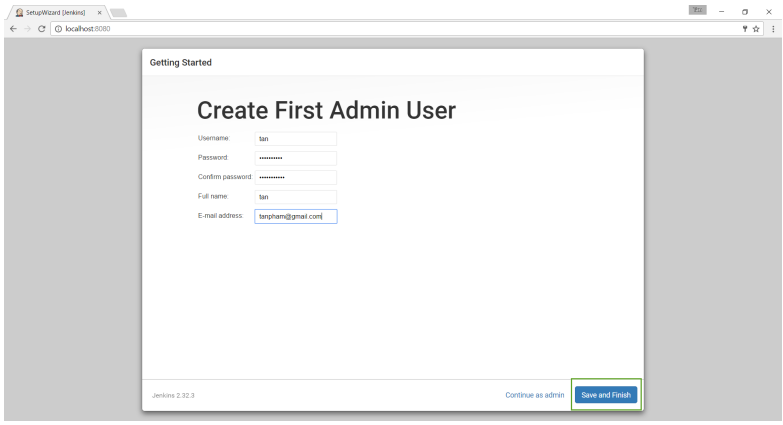
Step 3 : Open file ***/Users/Shared/Jenkins/Home/secrets/initialAdminPassword*** and copy password use for unlock Jenkins

Step 4 : Click on ***Install suggested plugins***

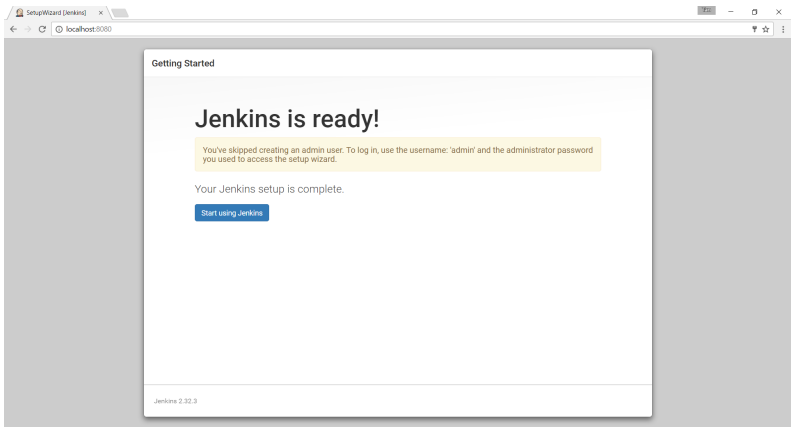




Step 5 : Enter your user name, password for a new admin.



Step 6 : That it, now jenkins ready to use, click to ***Start using Jenkins***



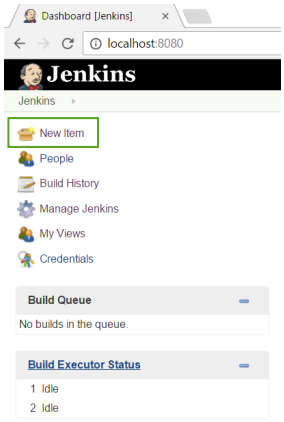
From now Jenkins already installed in Mac as a service, so every time you power on machine, Jenkins will start and ready to access at ***http://localhost:8080/***

Jenkins the Basic

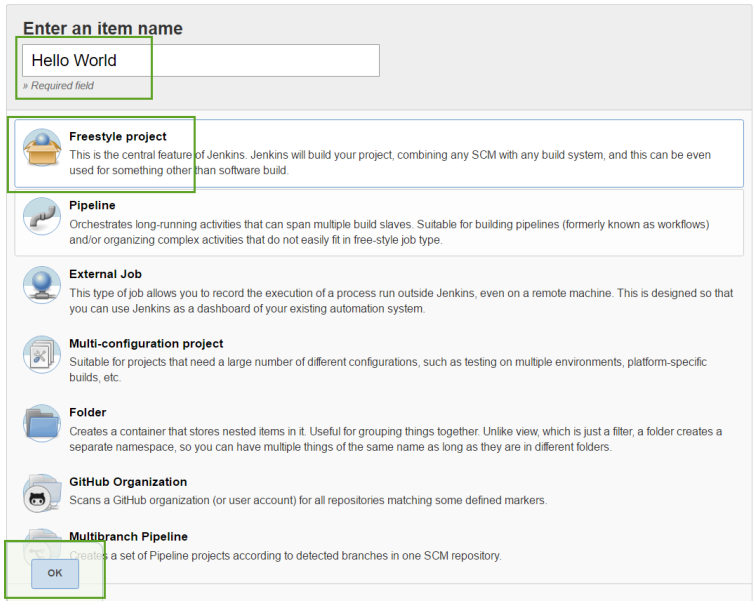
In this section I will explain Jenkins role in the world of continuous integration, continuous delivery and devops. And then we will go through steps in order to create “Hello World” Jenkins job.

Create “Hello world” Job

From Jenkins home page, click to **New Item**

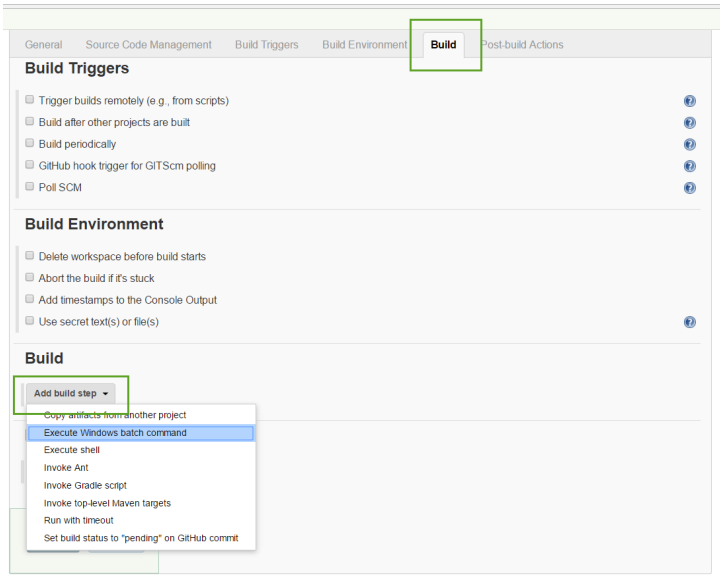


In create project page, typing in “Hello World” to project name, select **Freestyle project** and then click to save button.

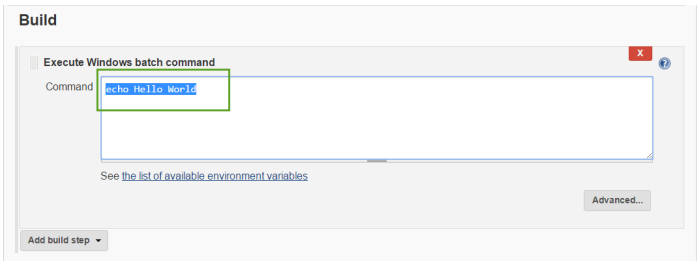


Add Build Step

In configure job page, select **Build** tab, click to **Add build step** and select **Execute Windows batch command** (in case you are using Linux please select **Execute shell**)



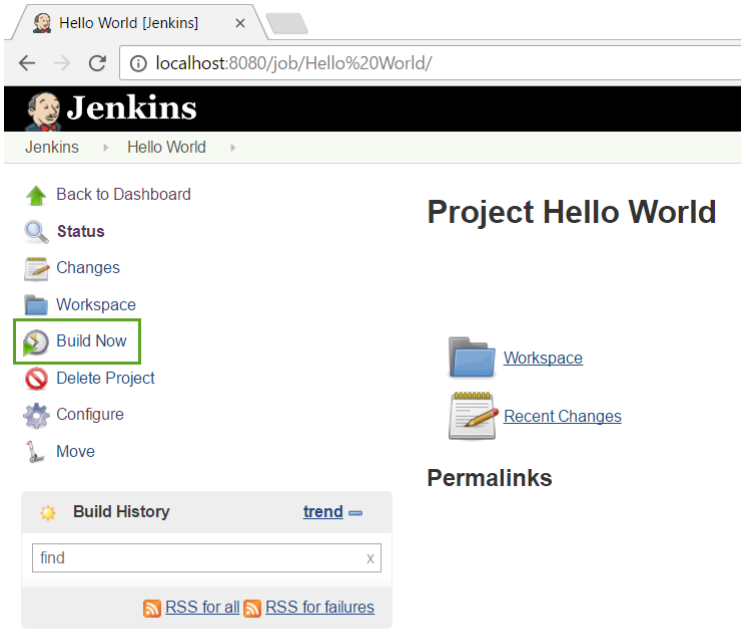
Typing in command “echo Hello World”. The purpose of this job is just echo to log message “Hello World” text.

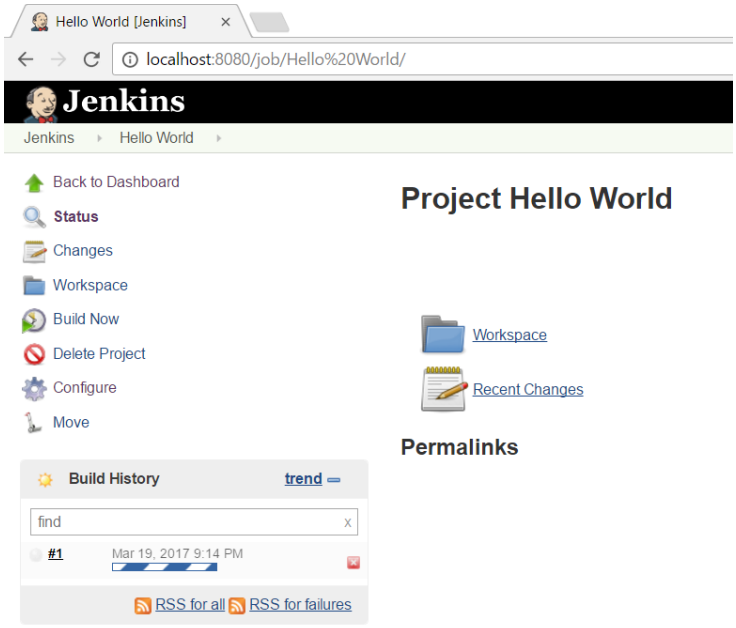


Click **Save** button and we complete the first job!

Start the First Build

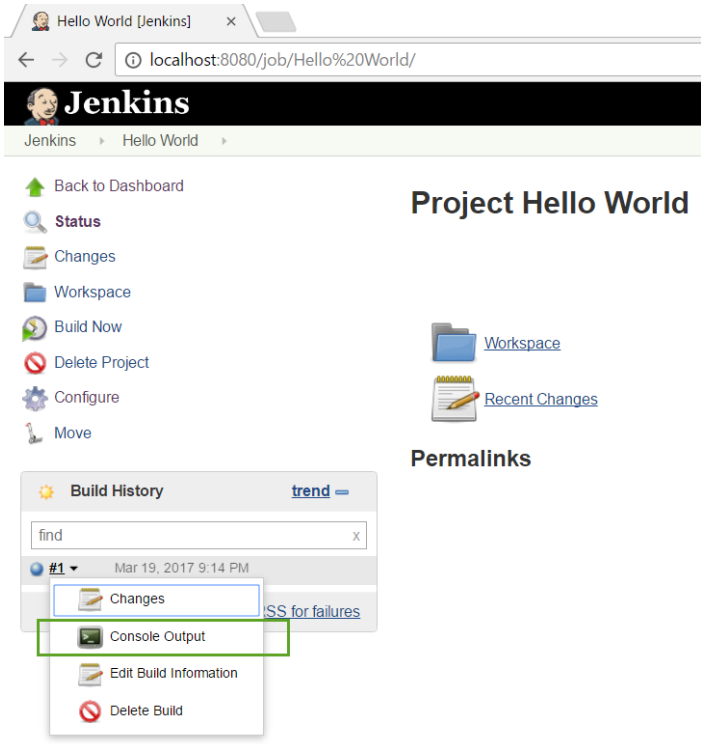
From job dashboard, click **Build Now** and you will see job run. This job just do one thing, invoke the command “echo hello world”





Console Output

Now job run completely, click to build number and select **Console Output.** You will see message “Hello World” show up inside build log. That it you just complete the first Jenkins job.

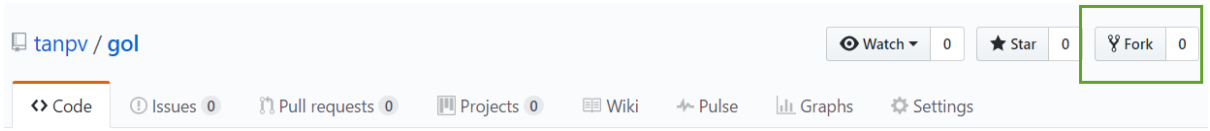


Intro GOL Project (or World without Jenkins)

In this book, project called GOL (game of life) is used to express the features related to Jenkins. So it will be helpful to review some information related to this project.

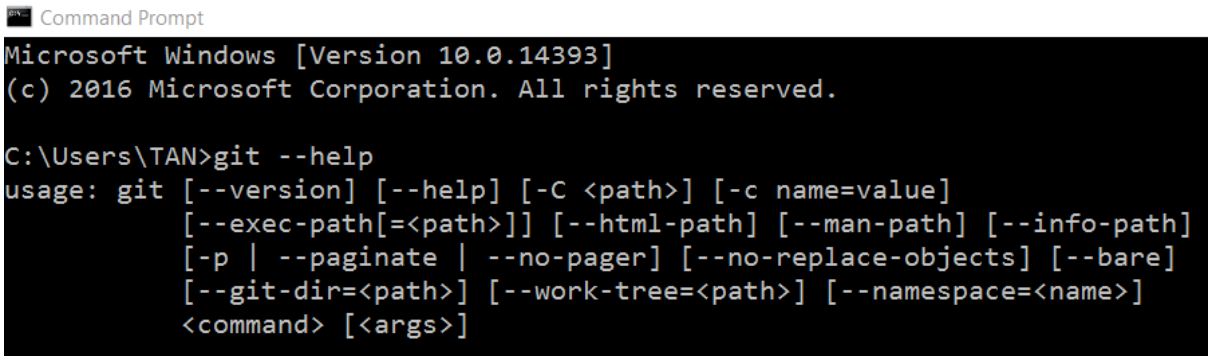
Git Installation and Clone GOL Repository

GOL project is hosted on Github at https://github.com/tanpv/gol. Please **fork** this repo to your Github account so you can try Jenkins features by yourself later.

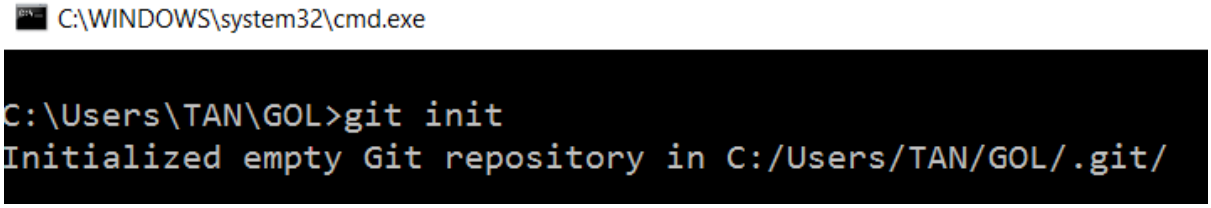


To install Git, download software at https://git-scm.com/ , and install with default option.

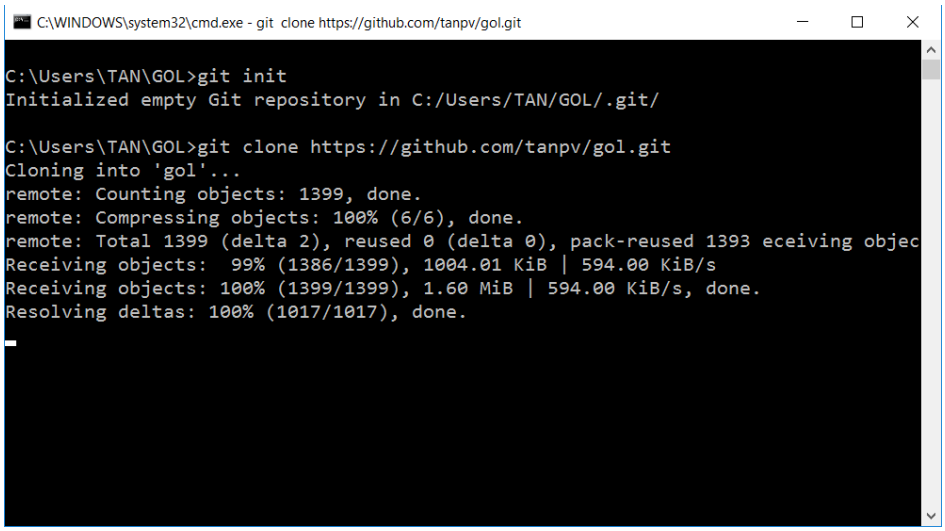
After install, open a cmd prompt and typing in **git --help** to check if install successfully



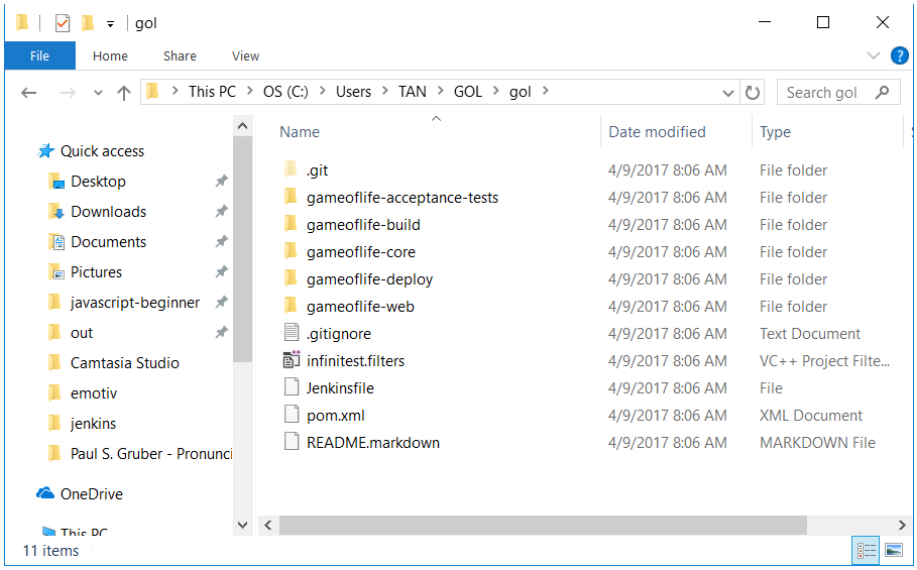
Now create a folder name GOL and init it with command **git init**



And finally, clone the repo from Github to local with command **git clone https://github.com/tanpv/gol.git**

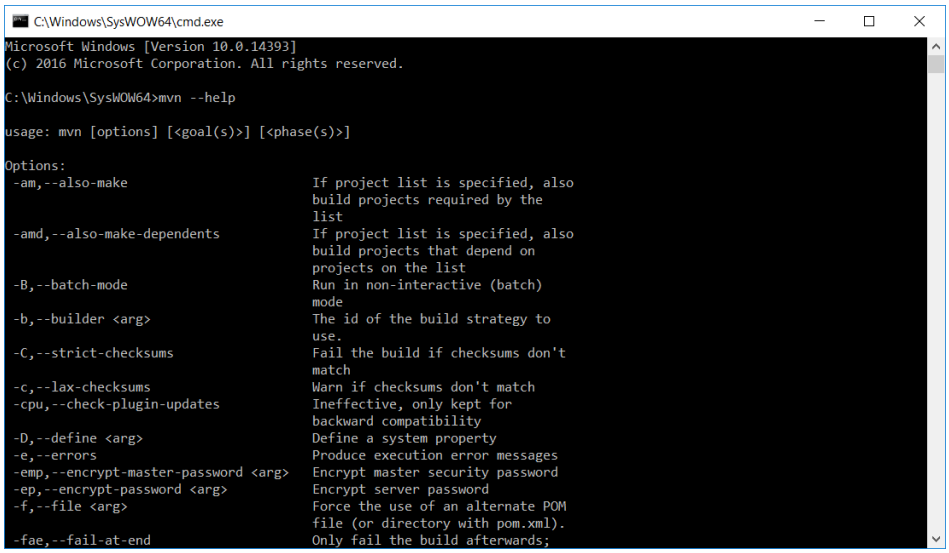


And finally, you will see GOL project is completely clone to local



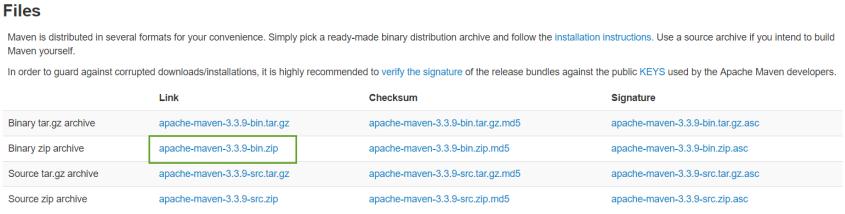
Maven Installation

First step, maven should be installed in machine which running Jenkins job. To check this, just open command prompt and typing in ***mvn --help ,*** you will some all available maven command.



In case maven not yet install follow these steps to do maven installation (for Window)

- Download maven from link http://maven.apache.org/download.cgi



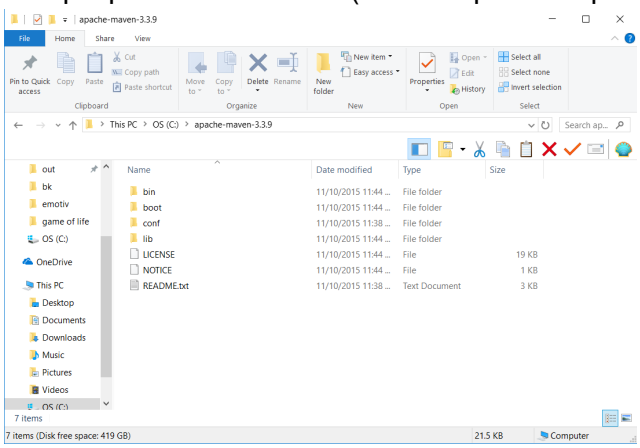
Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

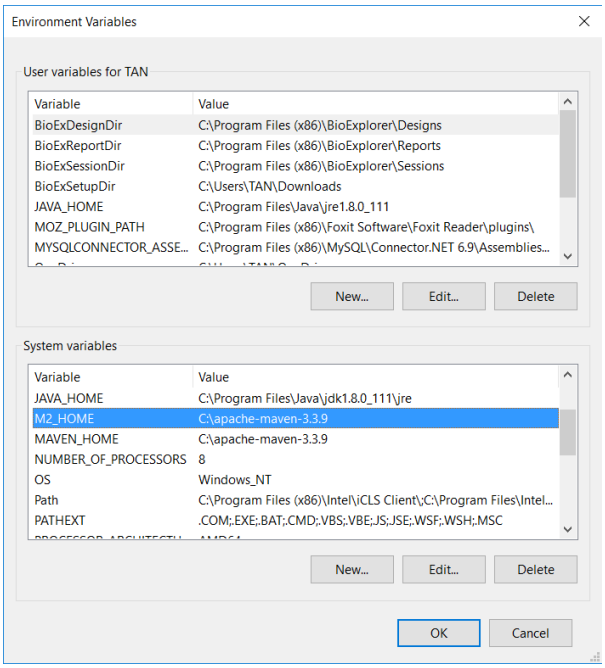
In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.9-bin.tar.gz	apache-maven-3.3.9-bin.tar.gz.md5	apache-maven-3.3.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.9-bin.zip	apache-maven-3.3.9-bin.zip.md5	apache-maven-3.3.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.9-src.tar.gz	apache-maven-3.3.9-src.tar.gz.md5	apache-maven-3.3.9-src.tar.gz.asc
Source zip archive	apache-maven-3.3.9-src.zip	apache-maven-3.3.9-src.zip.md5	apache-maven-3.3.9-src.zip.asc

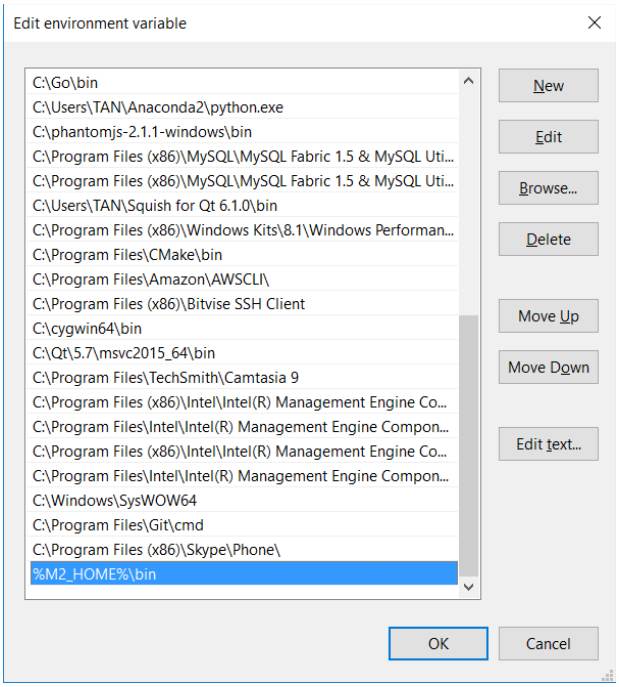
- Unzip zip file to one folder (for example I unzip to C:\apache-maven-3.3.9)



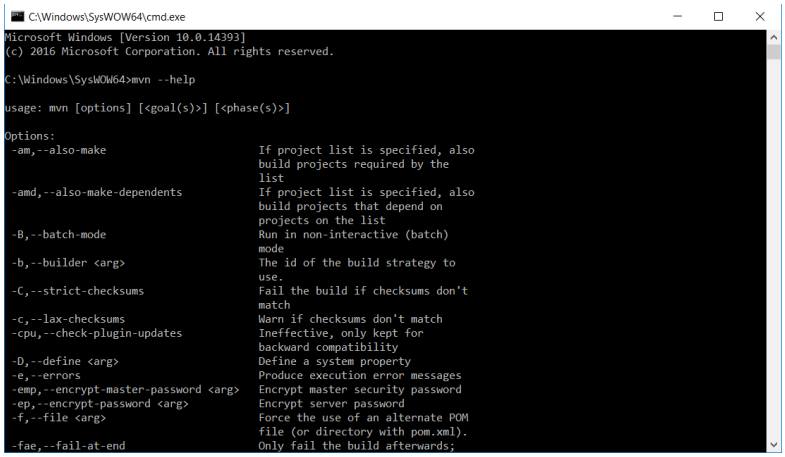
- Add M2_HOME and MAVEN_HOME variable point to above foder



- Add mavin bin to Path variable so you can call maven from any place



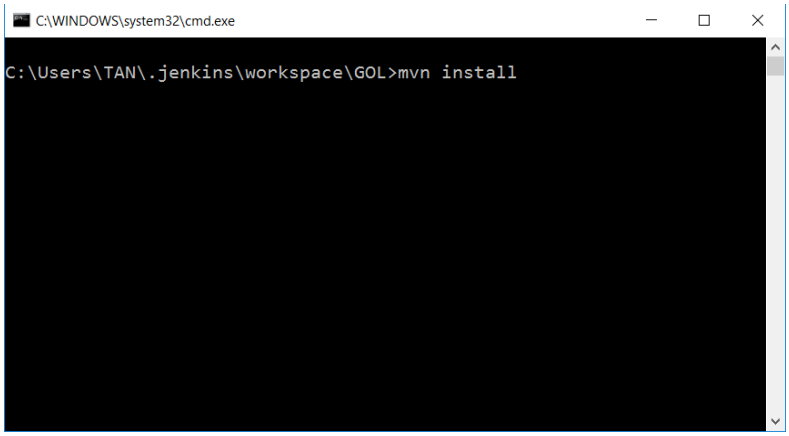
That it, to check maven installed successfully, open cmd and typing in ***mvn --help***
Help content for mvn should show up correctly



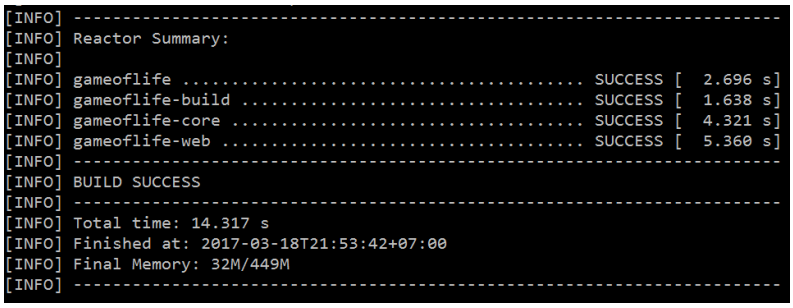
Run Maven Build Manually

Now maven is ready, we can try to run the build for source code which we just clone from Github repository at **source code management** step.

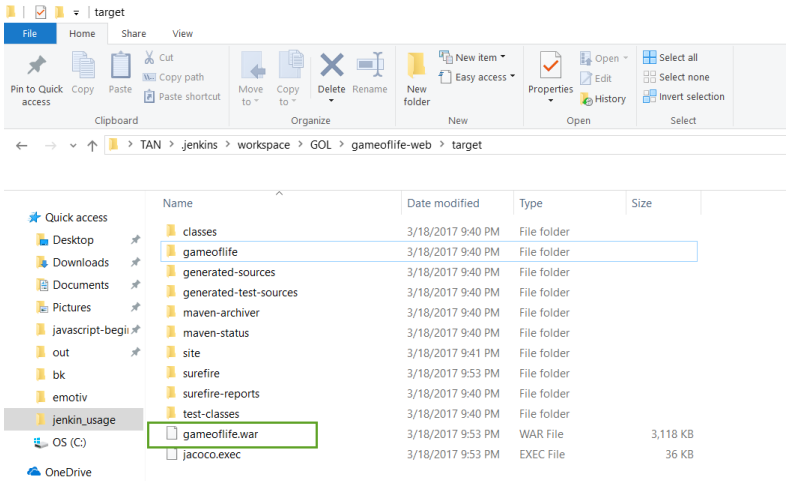
Open **cmd** and change directory to repository GOL folder. And then typing in **mvn install**
This command will run java source unit test and then build this web application to a .war file



You will build result from command prompt



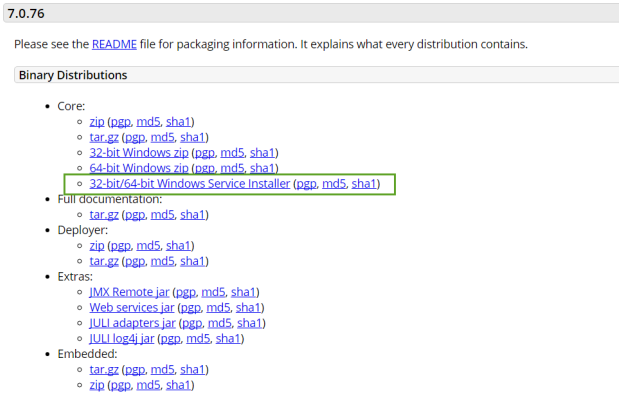
And can see actual .war file inside target folder of **gameoflife-web.** This folder could be use to deploy with web application server.



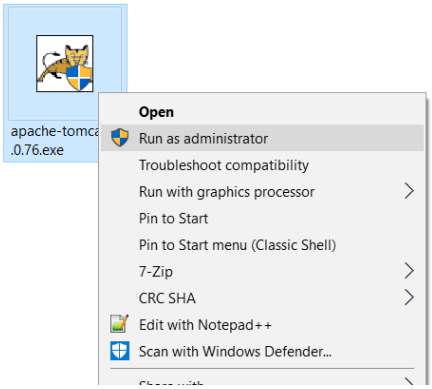
Install Tomcat Server as Service

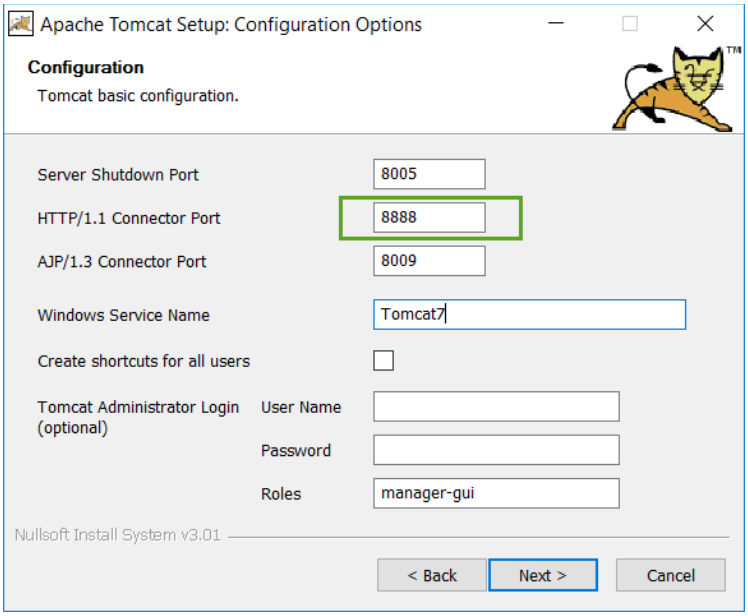
To install Tomcat 7 server, you just following these steps:

- Download Tomcat 7 from this page http://tomcat.apache.org/download-70.cgi , note that we will download version Windows Service Installer so Tomcat will be installed as window service

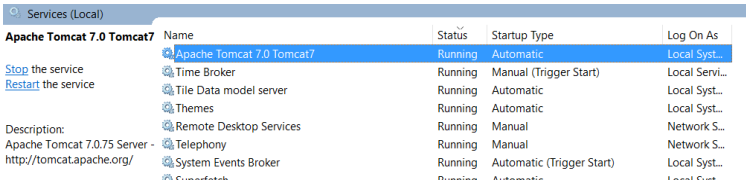


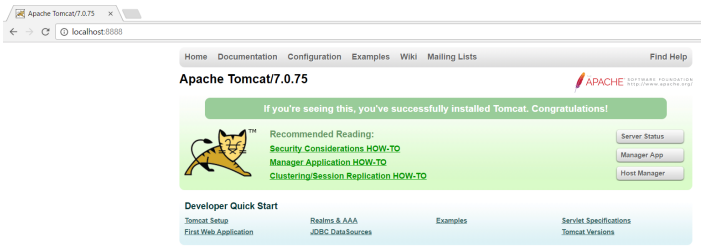
- Run installer and select port 8888 for running Tomcat (We already use 8080 for Jenkins installation)



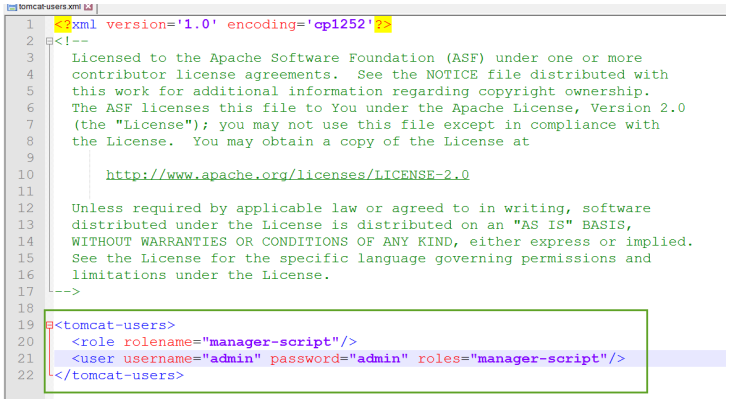


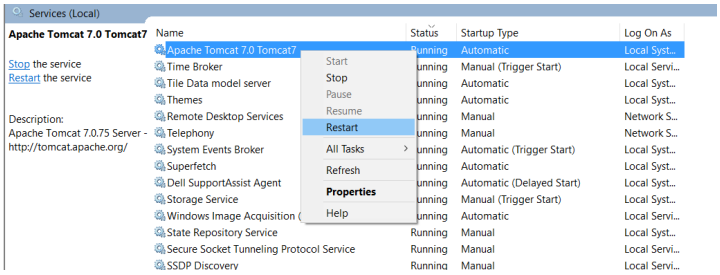
- After install successfully, a Tomcat service will running and you can access Tomcat server from localhost



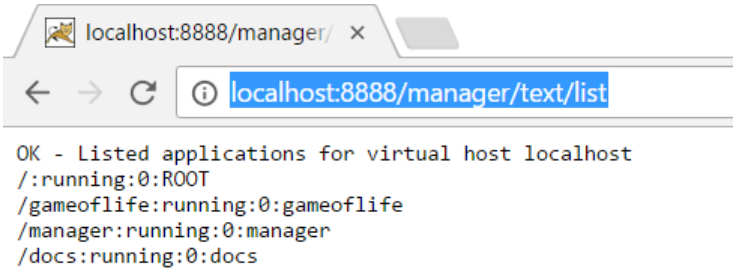


- To control Tomcat from Jenkins, one user with role “manager-script” should be added to file **conf\tomcat-users.xml .** Then restart Tomcat server by service.



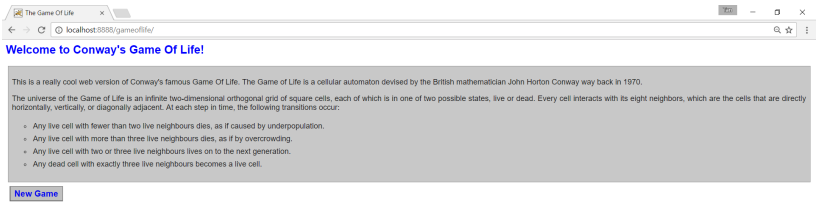


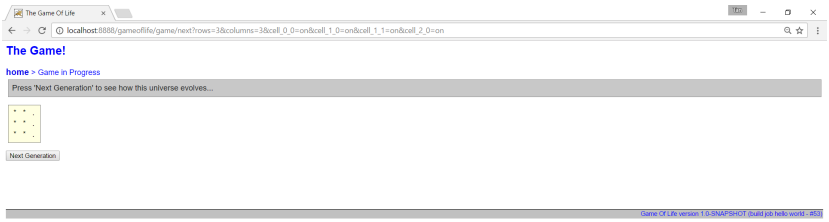
To check if admin user working, access to http://localhost:8888/manager/text/list and then put on user / password you just specify



Deploy GOL to Tomcat Server

Deploy this application to Tomcat server, and you will see some the app from browser and actually start a new game !.





Continuous Integration with Jenkins

This chapter will show you how to automate software build with Jenkins.

Jenkins in the Big Picture of CI, CD and DevOps

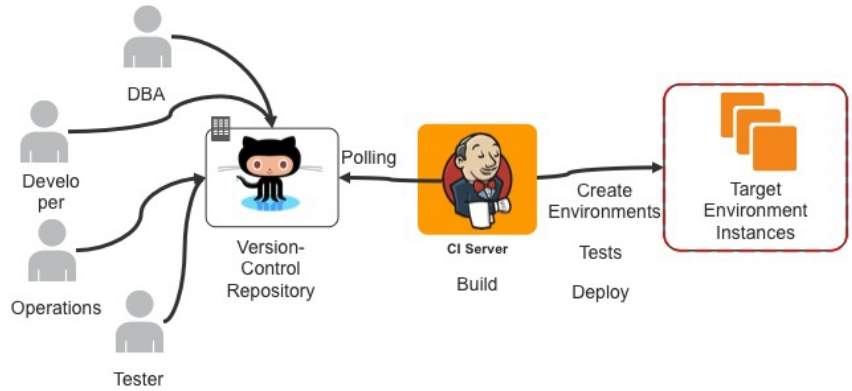


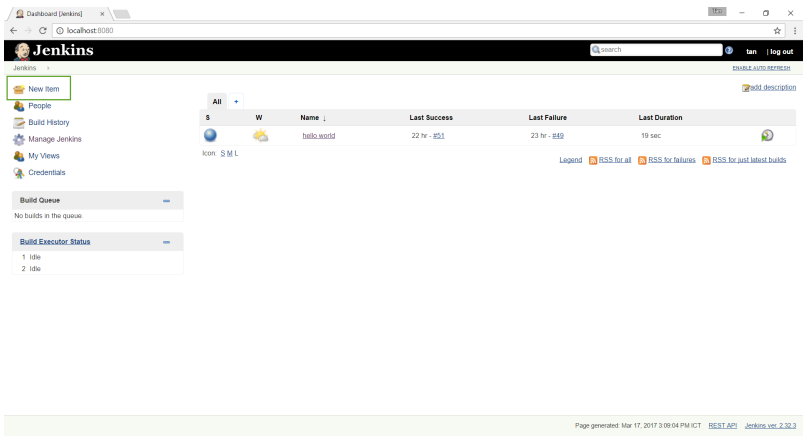
Image above show quite clear about Jenkins role in software development process.

- Developer do coding and commit their code to a version control repository like Github
- Jenkins actively check Github to see if have any change, in case has code change, Jenkins will pull the last code from Github to build machine (machine which run Jenkins job)

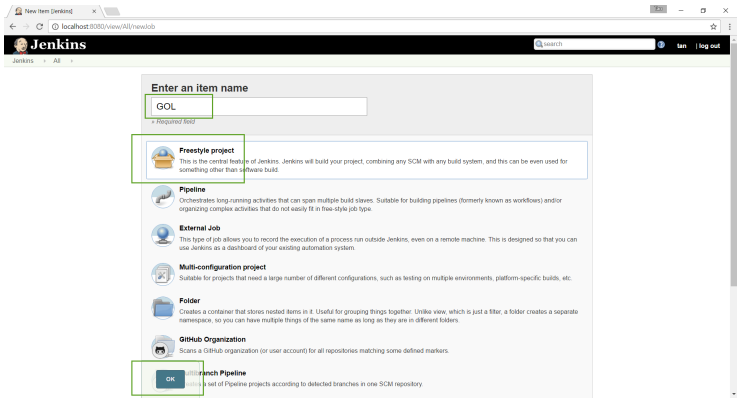
- Jenkins build software artifact from pulled source code (for example war file from .java source file)
- Jenkins run and show up unit test report
- Jenkins deploy artifact to target environment (for example a web server like : Tomcat, JBoss, WebSphere...)

Create GOL Job

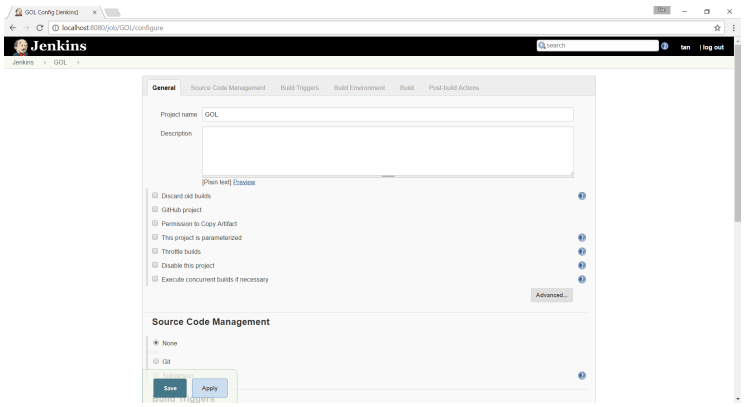
From home page, click to ***New Item*** link



Enter ***GOL*** for project name and then chose ***Freestyle project*** and finally click to ***OK*** button to create a new job with name is ***GOL***



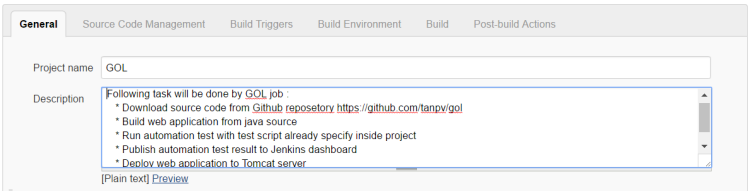
Then a new screen will show up allow us to configure everything for **GOL** job.



Description is place to describe what mission of this job and steps inside the job. Put follow text into ***Description*** session.

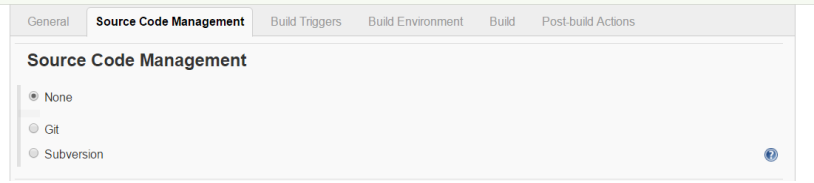
“ Following task will be done by GOL job :

- Check change on Github repo and automatically download source code from Github repository https://github.com/tanpv/gol
- Build web application from java source
- Run automation test with test script already specify inside project
- Publish automation test result to Jenkins dashboard
- Deploy web application to Tomcat server ”



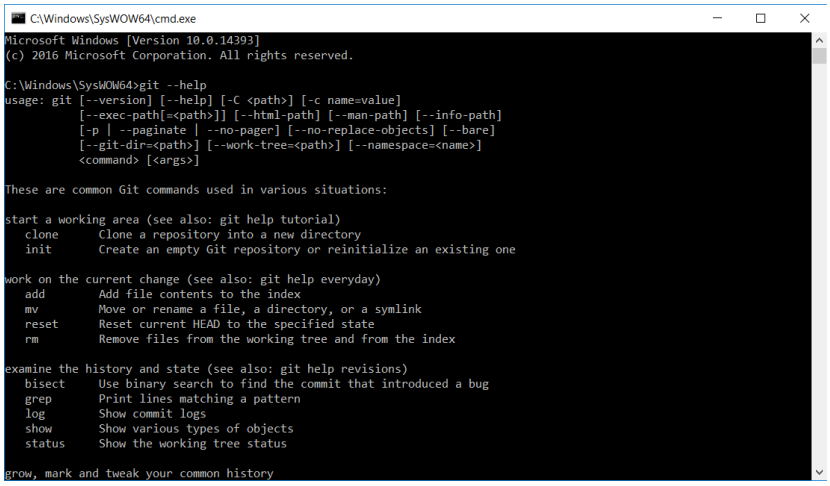
Source Code Management

Purpose of this part is answer for question ***Where to get source code ?*** for our software build. From job configure, click into ***Source Code Management*** tab



Check if git is installed in local machine

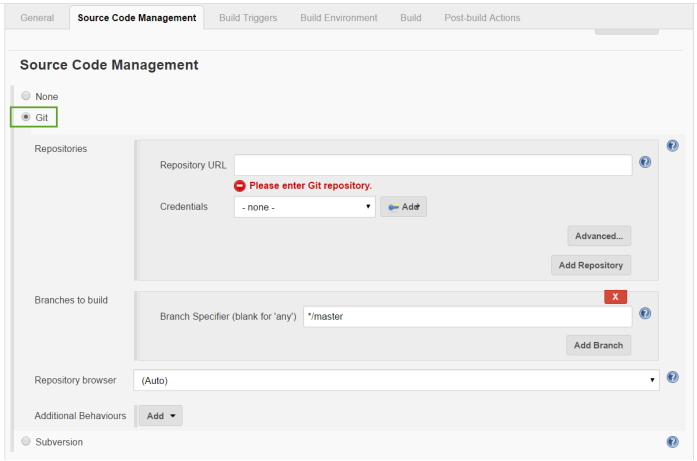
In order to clone source code from Github, local machine should install Git.
To check if local machine already install **git** or not, open command prompt and typing **git --help**



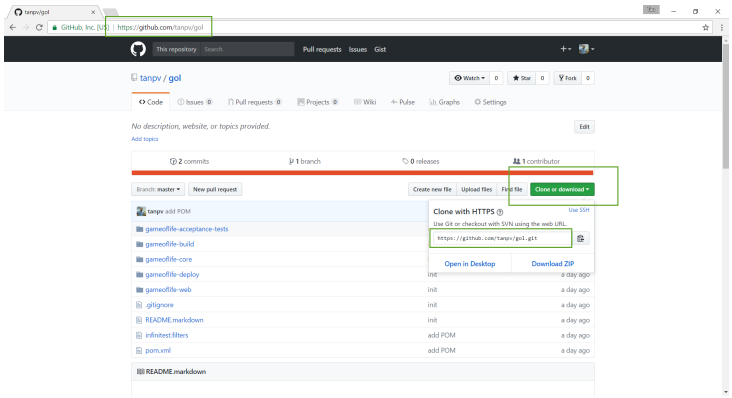
To download and install git, please refer to this link https://git-scm.com/downloads

Specify source code repository

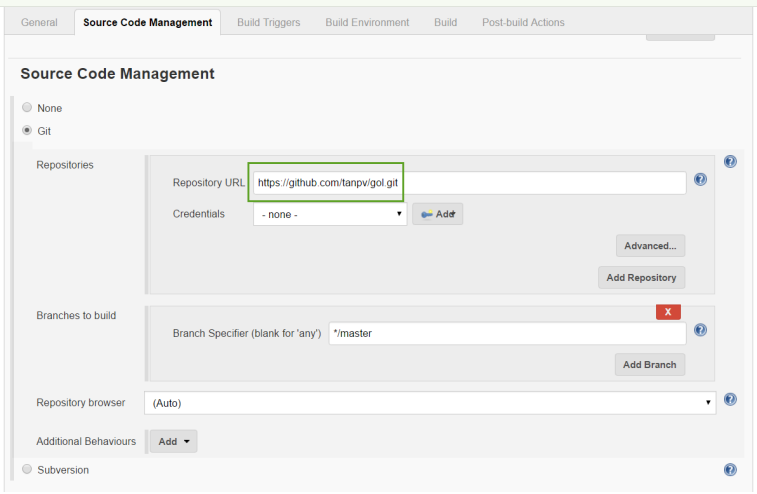
Because we host project source code on Github, so you will choose ***Git*** as below.



To get repo link, go to Github https://github.com/tanpv/gol then click to ***Clone or download*** button, you will see the repo link. Copy this link.



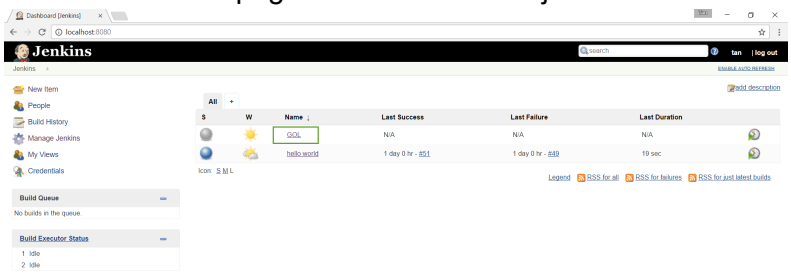
Put the repo link ***https://github.com/tanpv/gol.git*** in to ***Repository URL*** section. The branch will be default as ****/master***



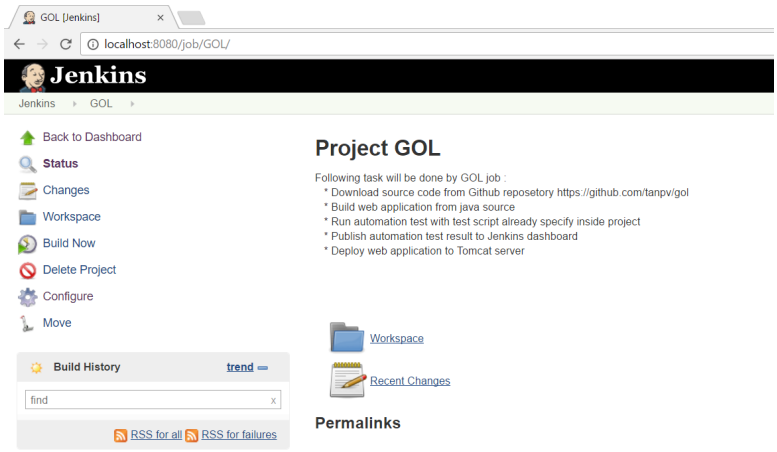
Click to ***Save*** button and that it, we already tell to Jenkins where to get source code.

Check if setting working fine

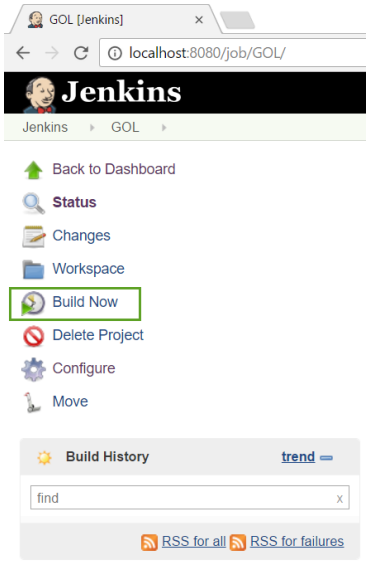
This is the time to try for first running and see if source code from Github is clone or not. Go back to home page then click to GOL job



The dashboard for GOL will show up, you will see describe which we just added before

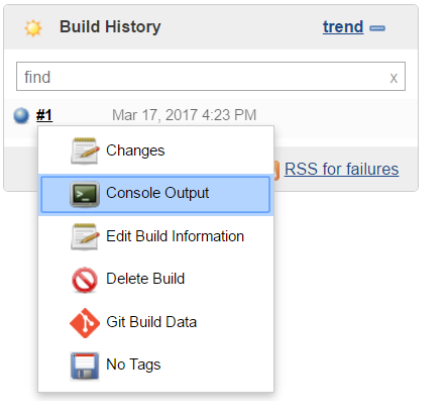


Click in to ***Build Now*** to activate job run immediately.

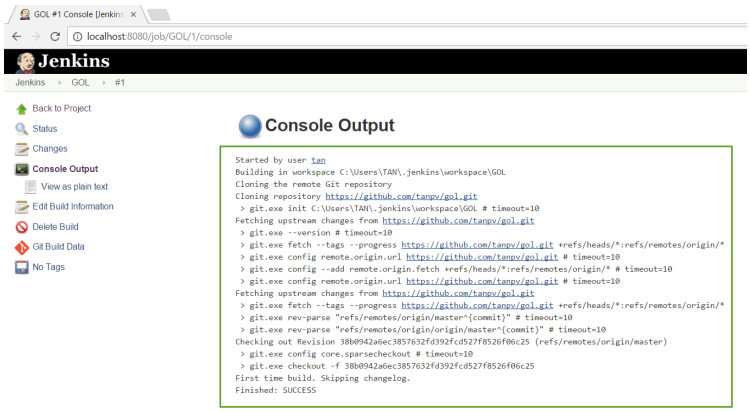


You will see GOL start running as expected

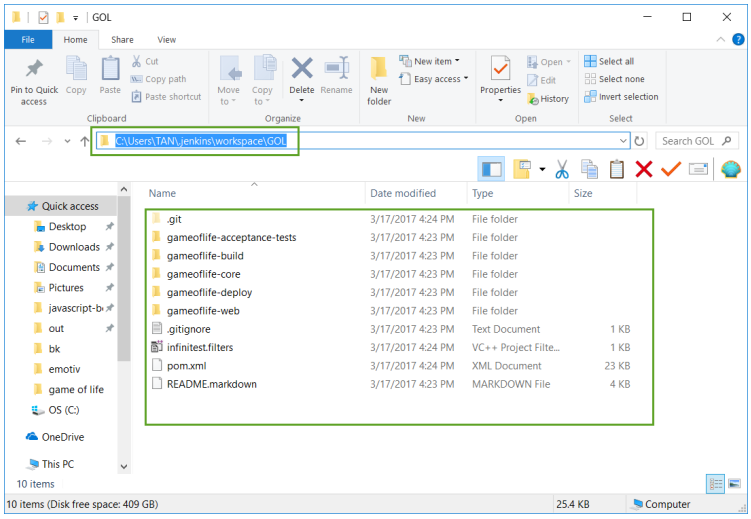
To see how job running, select ***Console Output*** from context menu while you click to build number



Then the job log will show up every thing happen in order to clone source code from Github repository. You can see that the build already success running.

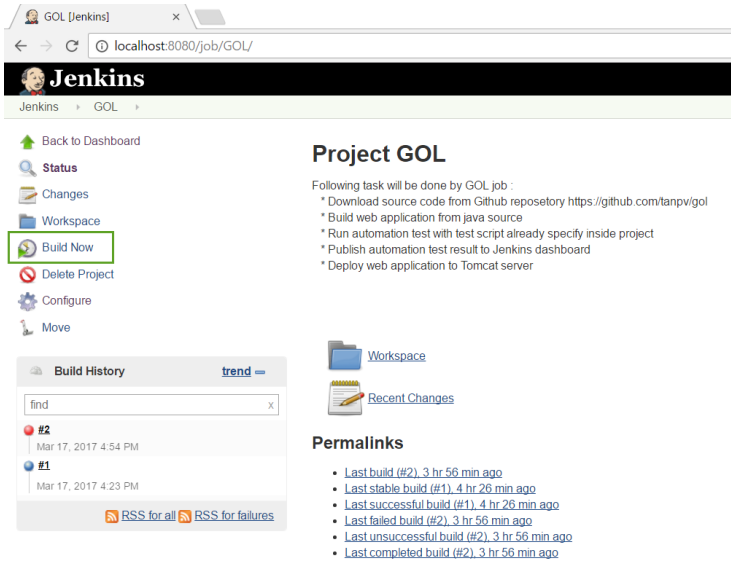


Now we could go to local repository to see the source code already downloaded. Each job will have it's own folder inside **workspace** folder

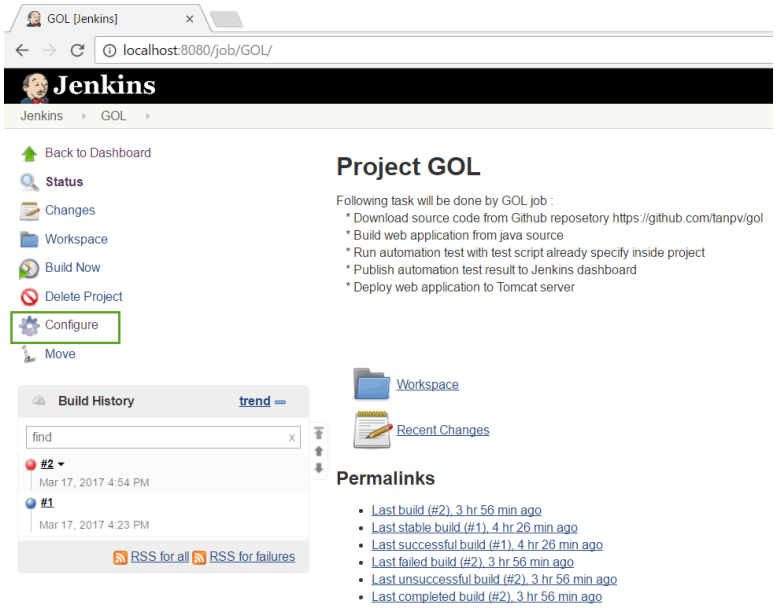


That it, we complete the first step of this job.

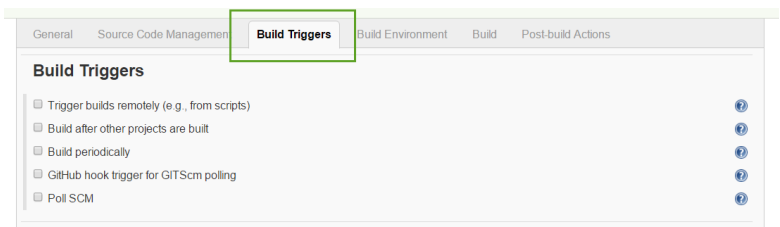
Build Triggers



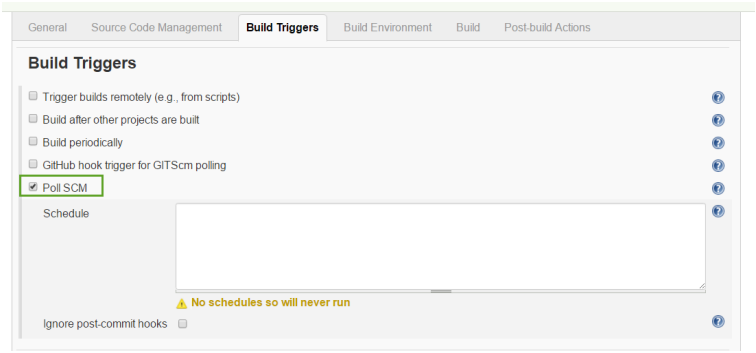
The first way and most simple way to start LOG job is just click in to ***Build Now*** button. But this way is manually and not so cool. We want to trigger Jenkins job running automatically, so to do this, click to ***Configure*** from LOG dashboard.



Then select to ***Build Triggers*** tab you will some option to configure so job could trigger automatically



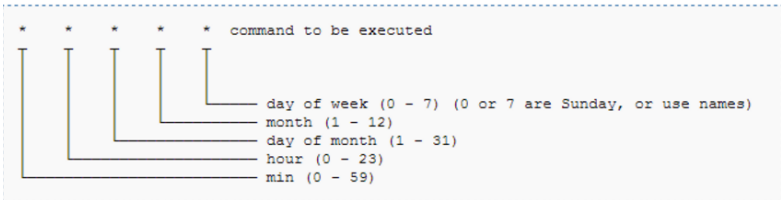
Have 5 ways to automate trigger Jenkins job as show above. In this book I will focus on **Poll SCM,** the good way in configure for a continuous system. Now you click on **Poll SCM,** you will see a text box show up for setting schedule



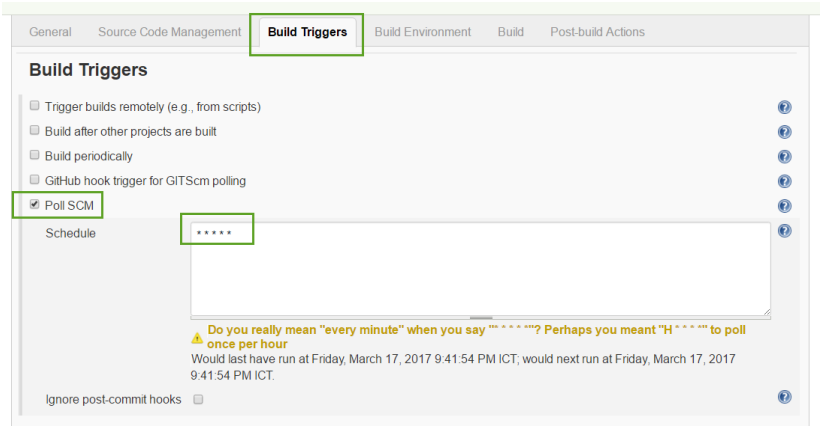
So following are steps show up how ***Poll SCM*** work :

- Base on schedule setting, Jenkins will actively check Github repository to see if have any change from repository
- If have any change on Github repo Jenkins job will trigger

Entry	Description	Equivalent To
`@yearly (or @annually)`	Run once a year at midnight in the morning of January 1	`0 0 1 1 *`
`@monthly`	Run once a month at midnight in the morning of the first of the month	`0 0 1 * *`
`@weekly`	Run once a week at midnight in the morning of Sunday	`0 0 * * 0`
`@daily`	Run once a day at midnight	`0 0 * * *`
`@hourly`	Run once an hour at the beginning of the hour	`0 * * * *`
`@reboot`	Run at startup	`@reboot`



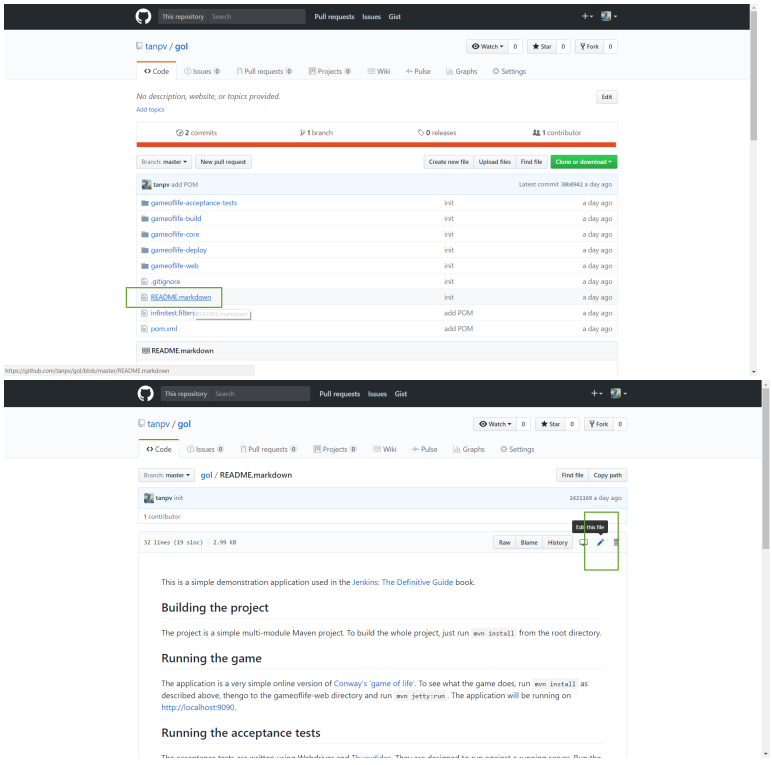
Schedule with ***Poll SCM*** work follow cron schedule rule, above image show some cron schedule example. Basically, we want to catch the change on Github repository as soon as possible, so follow text will be add to ***Schedule*** * * * * * . This mean we want to check change on Github every minute !

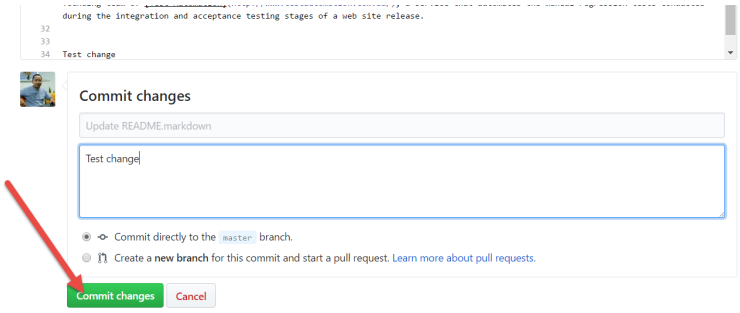


Click to **Save** and Jenkins will move you back to job dashboard

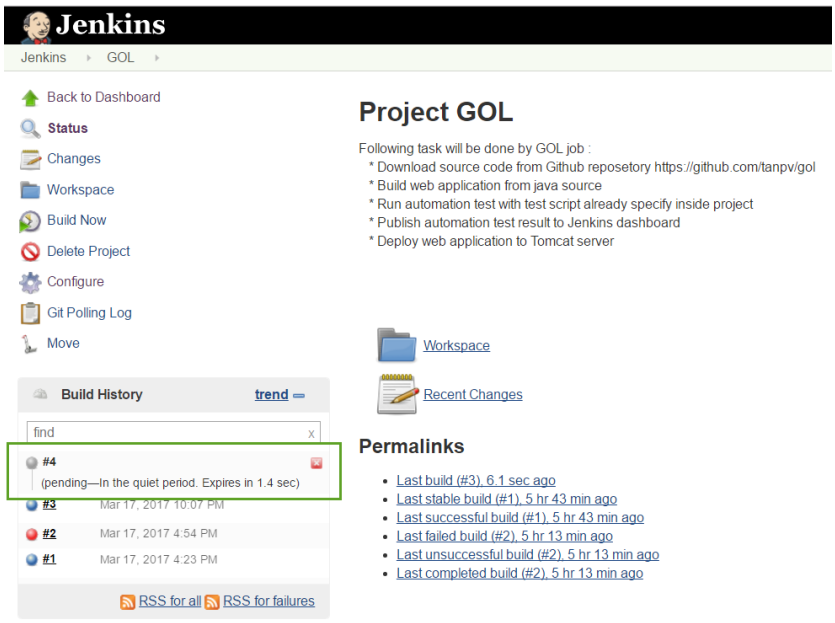
Now is the time to check if this setting really work ?

I will go to my GOL project repo on Github and change content of file ***README.markdown*** right from web browser (Github support commit change right from web browser)

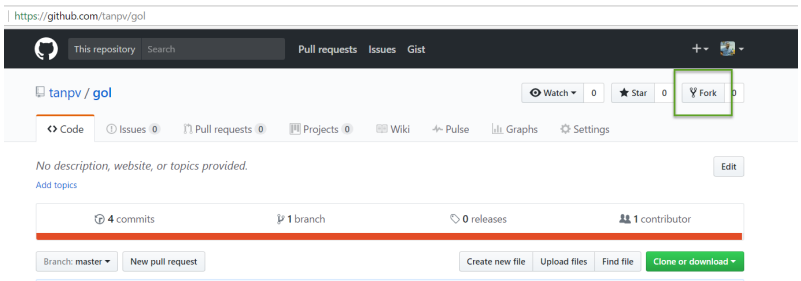


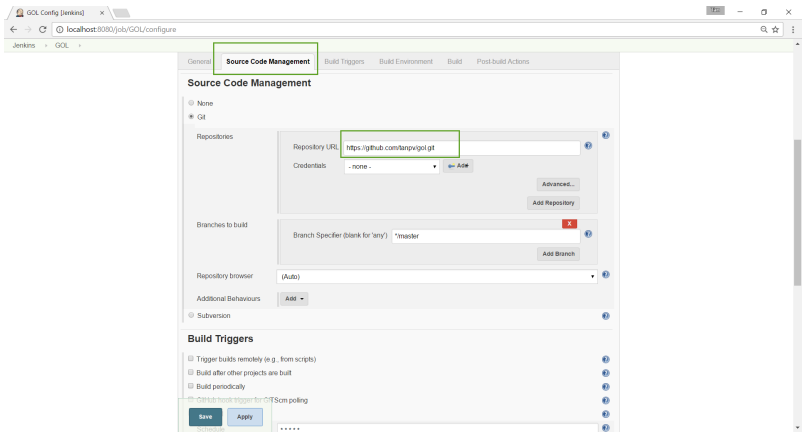


Now come back to job dash board, wait for about 1 minute, you will see new job is planning and run.



That it, it work !!!, to practice by yourself, just clone GOL project to your account by **Fork** button and then change the GOL configuration point to your github account as show below.



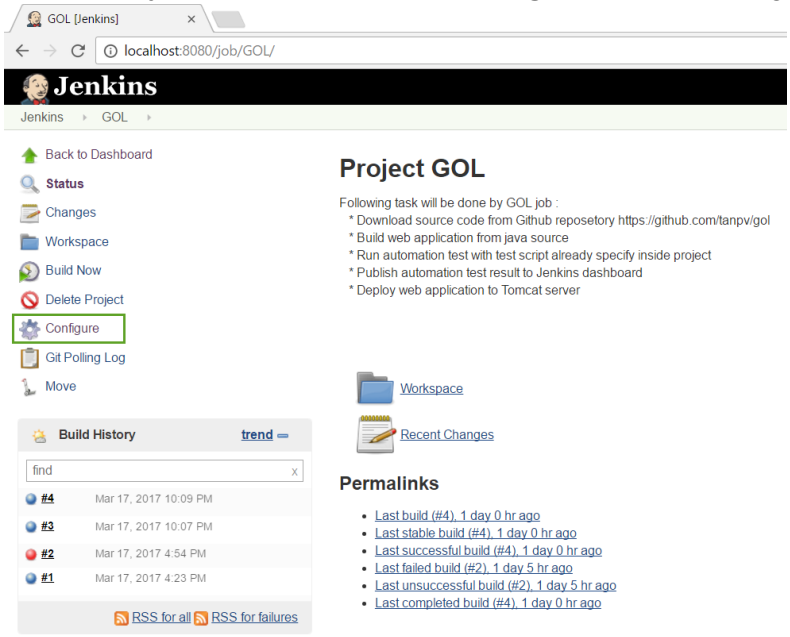


Build

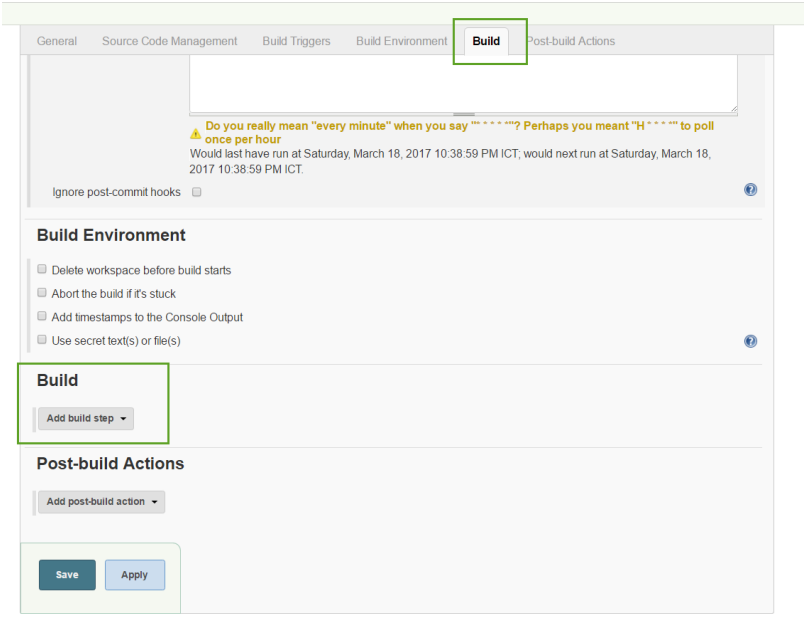
In this section, We will setting to build java web app with maven.

Configure Build Step

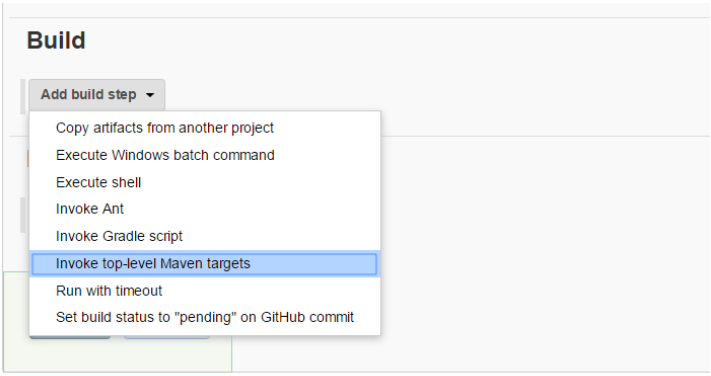
From GOL job dashboard, click to **Configure** link, the configure page will show up



From configure page, select **Build** tab, page will scroll down to **Add build step**

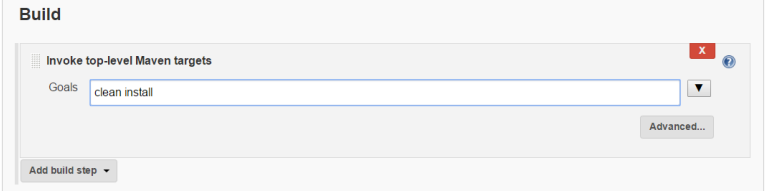


Click to **Add build step** and select **Invoke top-level Maven targets**



Section to adding maven command will show up as below

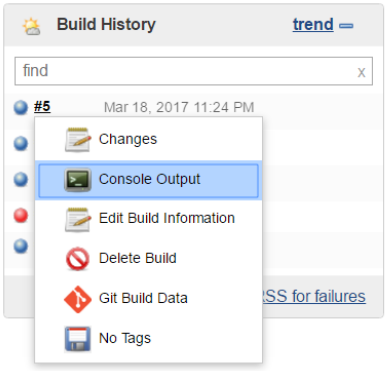
Put **Goals** maven command **clean install**



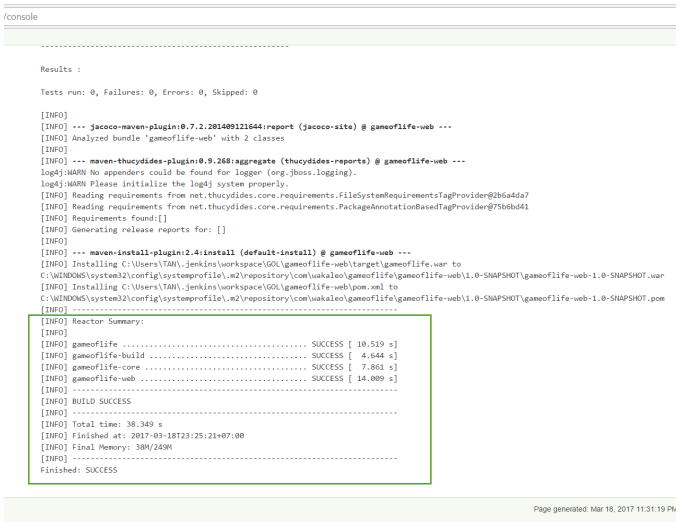
Then finally click in to **Save** button. That it we already complete configure for build maven step.

Check Log and Workspace

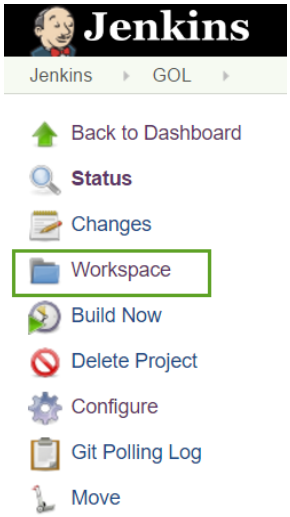
Now from dashboard of GOL, just click to **Build Now** to see how Jenkins job work by open console log



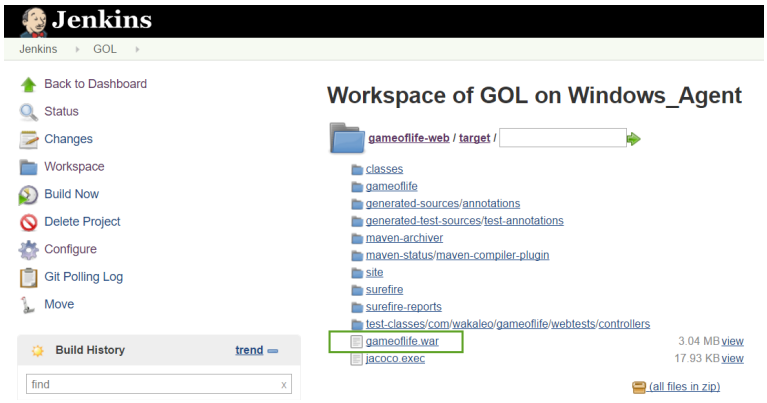
Scroll this log to bottom you will see build job run successfully



From GOL home page, click to **Workspace**



You will access directory where build happen and actually could see the war file inside **Gameoflife-web/target/.** You could actually download war file from here.

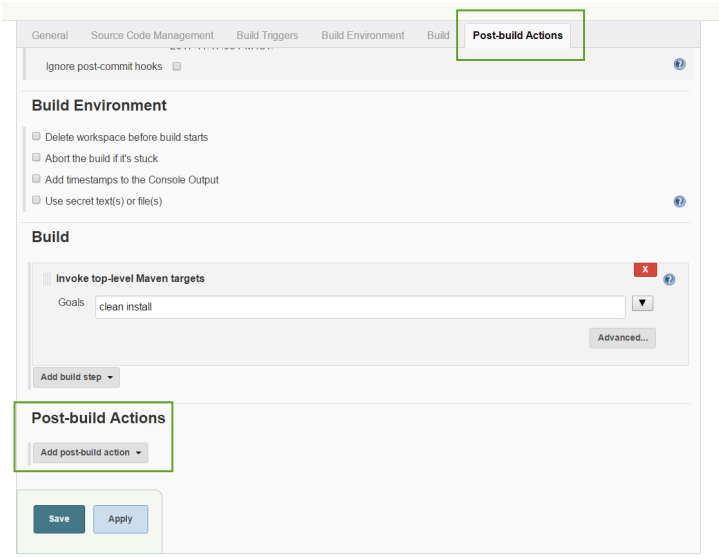


That it, We already finish setting up build step.

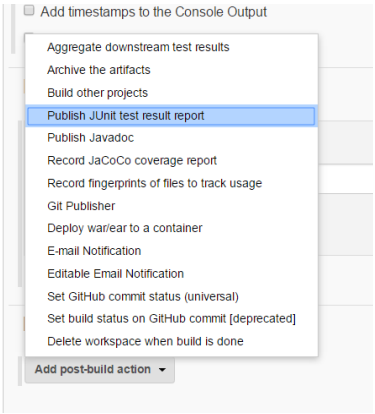
Continuous Inspection with Jenkins

Job Configure for Test Report

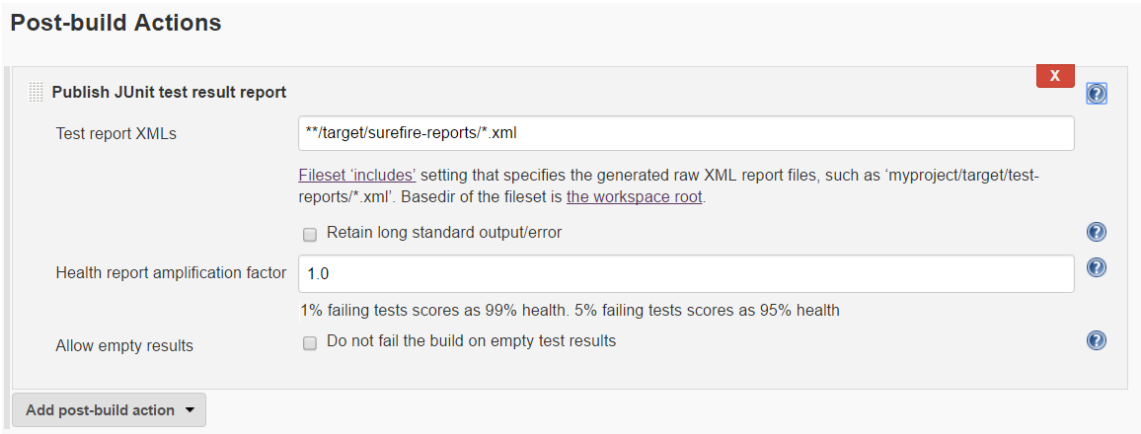
From job dashboard, click to **Configure.** Because showing unit test result on dashboard is done after job build successfully, so we click to tab **Post-build Actions.**



Click to **Add post-build action,** from dropout menu, select **Publish JUnit test result report**



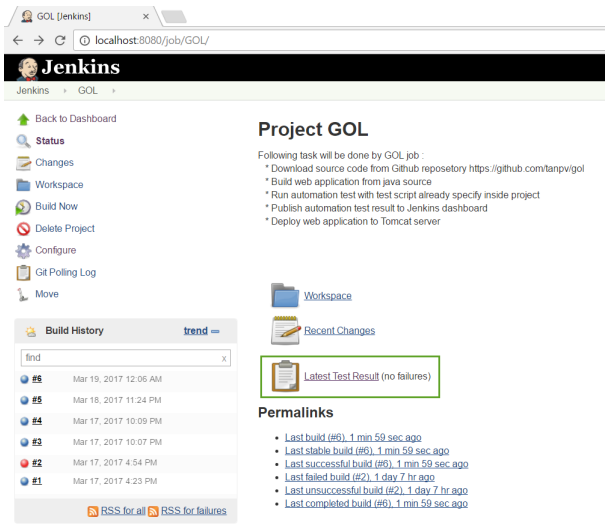
Input path to maven test result ****/target/surefire-reports/*.xml**



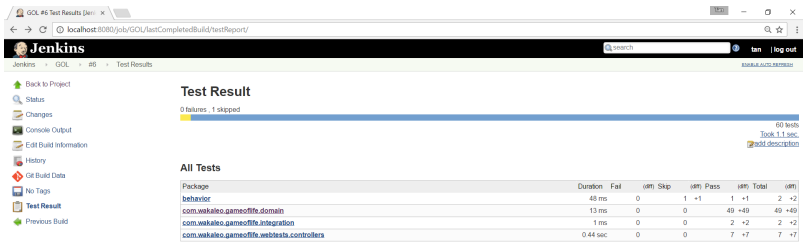
Click **Save** button to finish configure

Test Report on Job Dashboard

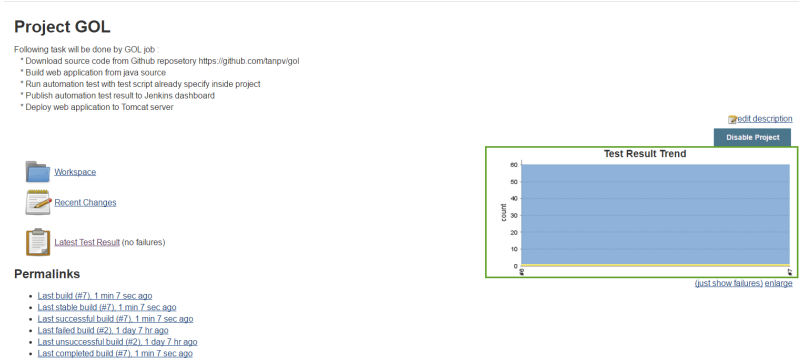
From job dashboard, click to **Build Now.** After job run successfully, you will see a link lead to test result. Click to **Latest Test Result**



The test report show up, from here you can see many statistic related to test.



Continue run second time, you will see a chart on job dash board which show up how trending in test result or changing from build to build.

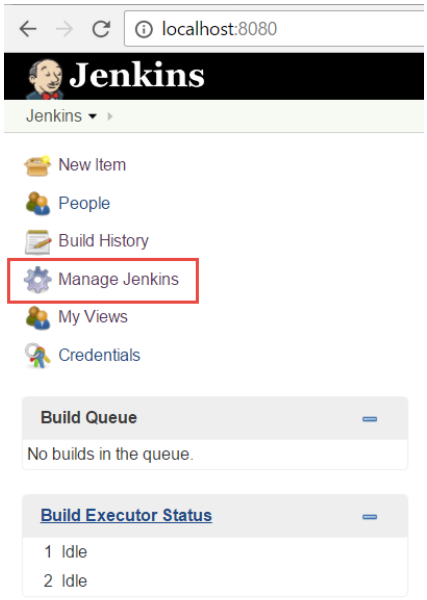


Continuous Delivery with Jenkins

This session will show you how to deploy web application (.war file) to a Tomcat web server.

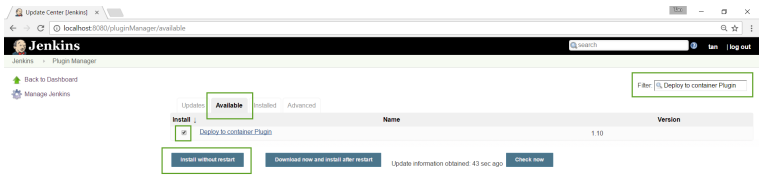
Install “Deploy to Container Plugin”

In order to deploy war file in Tomcat server, we need to install “Deploy to Container Plugin”. From Jenkins home page, click to **Manage Jenkins**

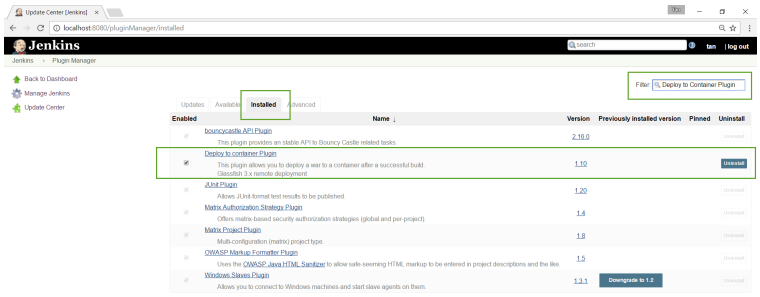


Then click to **Manage Plugin**

Click to tab **Available,** typing “Deploy to Container Plugin”, then click to **Install without restart** button

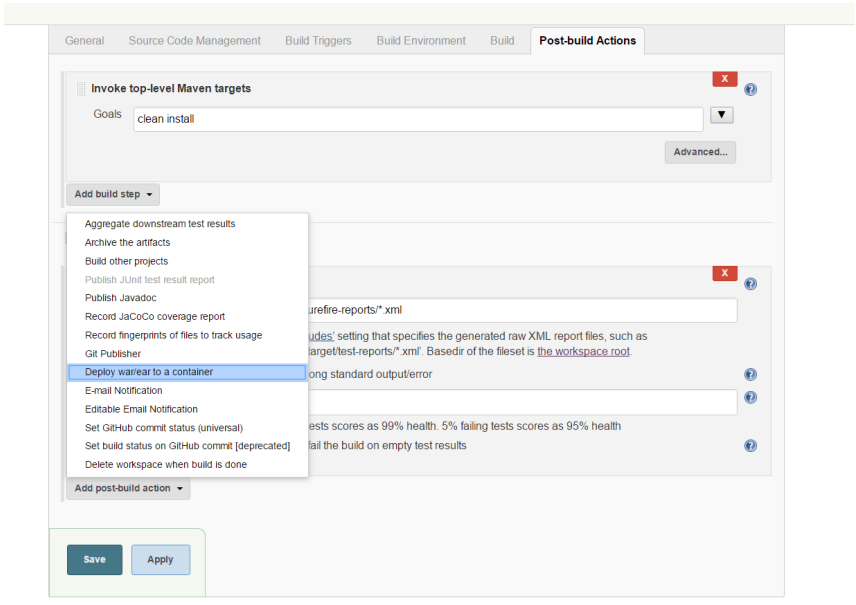


After install you will see it show up inside **Installed** tab

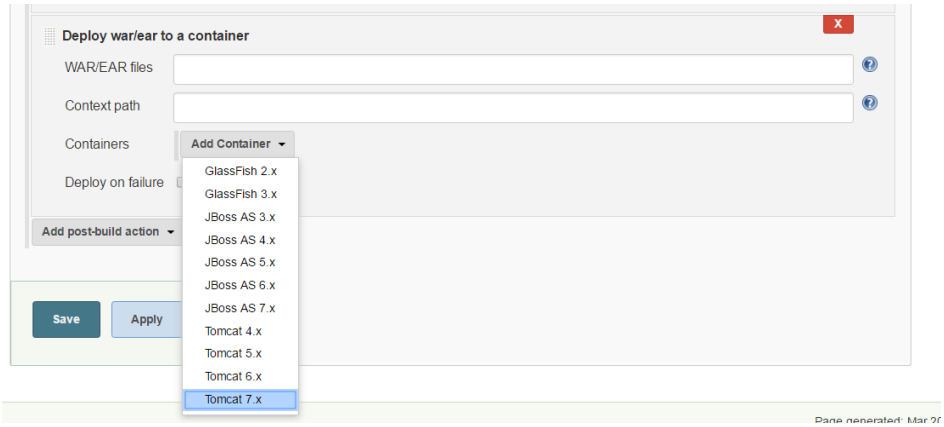


Add Deploy Step

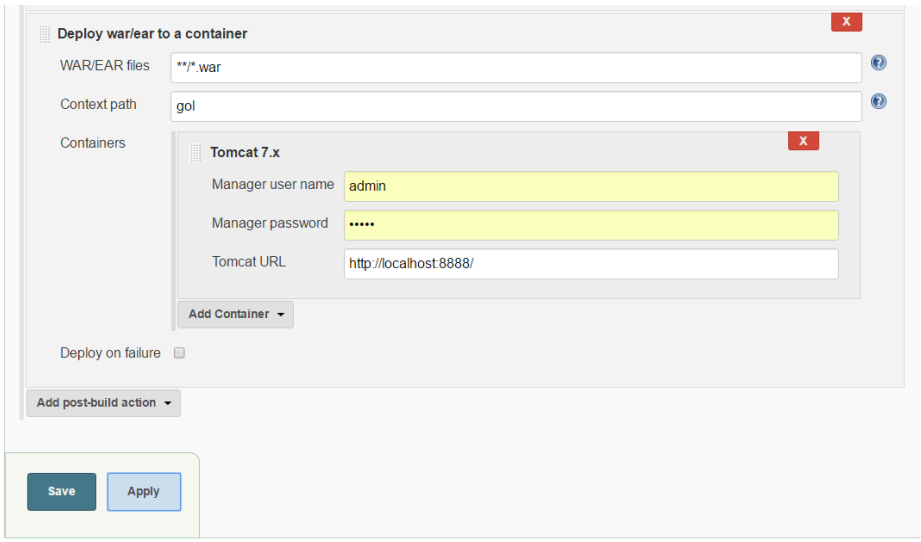
Open GOL job configuration, click to “**Post-build Actions**” tab, click to “**Add post-build action**”, and select item “**Deploy war/ear to a container**” from context menu



Select “Tomcat 7.x” from **Containers**

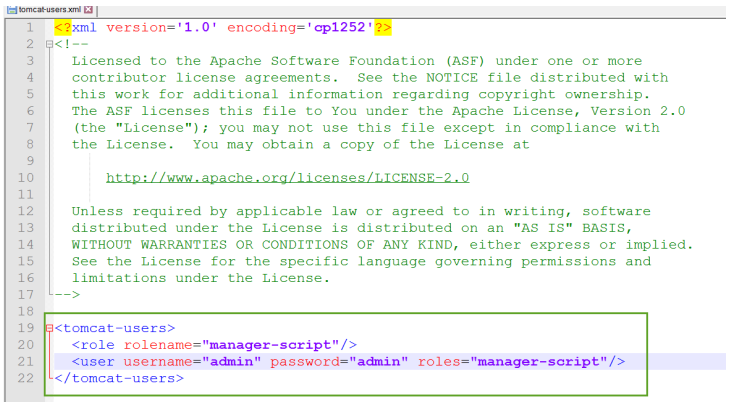


Input path to **war file**, input **context path**, and finally input **admin user/password** which already created at step install Tomcat server.



Finally click to **Save** button.

Note that to control Tomcat from Jenkins, one user with role “manager-script” should be added to file **conf\tomcat-users.xml .** Then restart Tomcat server by service.

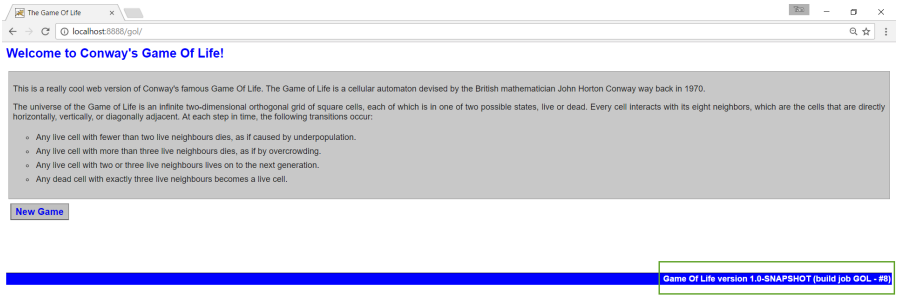


Check if Deploy Step Run Successfully

From GOL job dashboard, click into **Build Now,** wait some time until job complete and you will see consolog for step deploy GOL to Tomcat server

```
Recording test results
Deploying C:\Users\TAN\.jenkins\workspace\GOL\gameoflife-web\target\gameoflife.war to container Tomcat 7.x Remote
  [C:\Users\TAN\.jenkins\workspace\GOL\gameoflife-web\target\gameoflife.war] is not deployed. Doing a fresh deployment.
  Deploying [C:\Users\TAN\.jenkins\workspace\GOL\gameoflife-web\target\gameoflife.war]
Finished: SUCCESS
```

And now you try to access GOL web app at http://localhost:8888/gol/

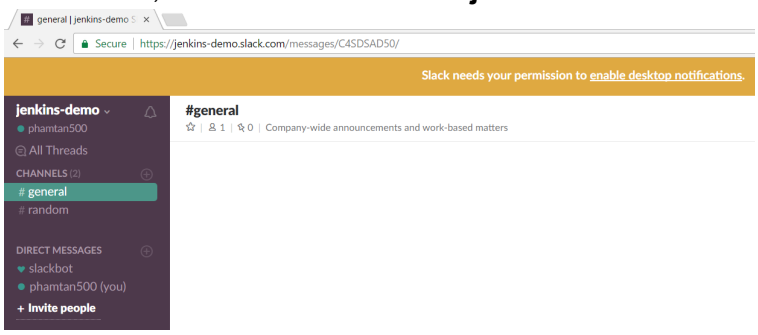


That it, deployment step working fine now. And the great thing is you know you are testing on what build on Jenkins.

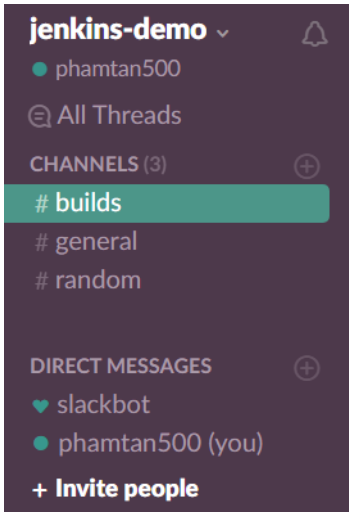
Continuous Monitoring with Jenkins

Create Slack Team and Install Jenkins CI App

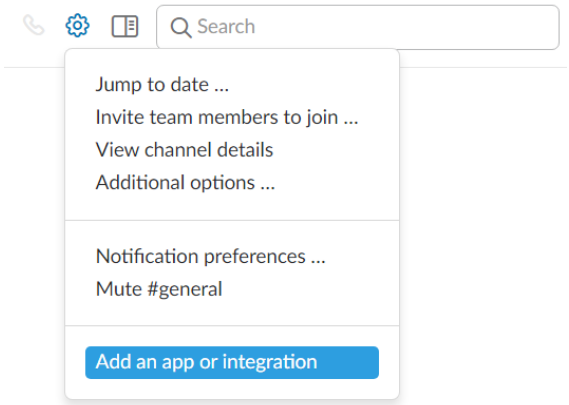
From Slack, create a new team call **jenkins-demo**



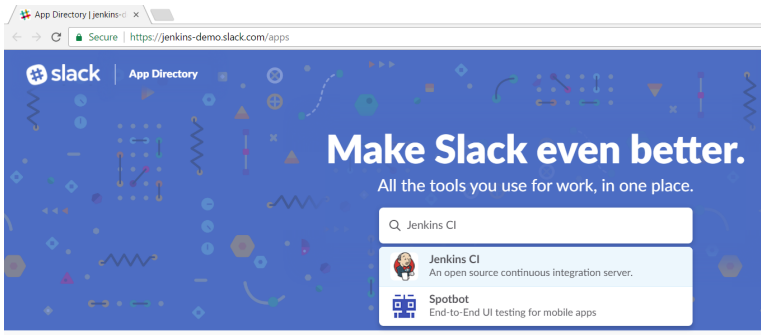
Create a new channel call **#build,** this channel will be used to notify message from Jenkins



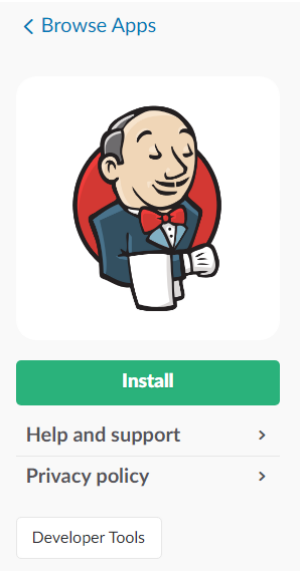
From **jenkins-demo** home page, click to **Add an app or integration**



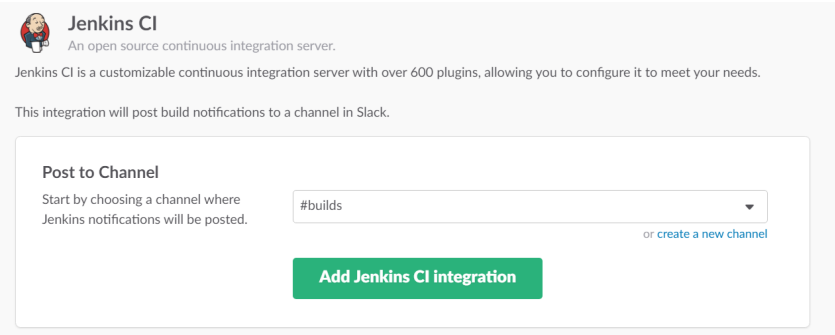
Search for **Jenkins CI** application



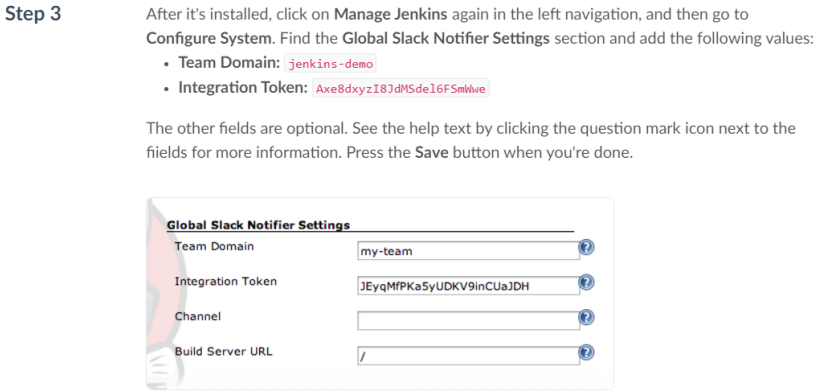
Click to **Install** button



Select **#build** and click **Add Jenkins CI integration**



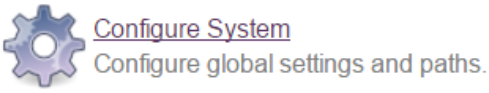
Copy information related to team domain and token, this information will be use to configure Jenkins job later



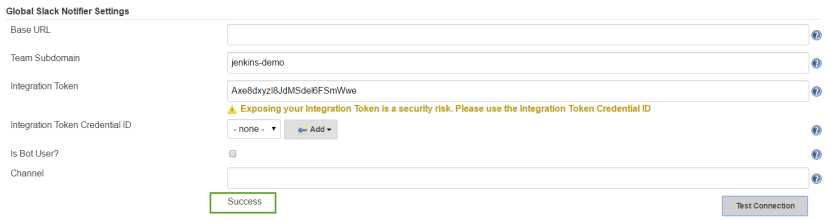
Install Slack Notification Plugin to Jenkins

Go to **Plugin Manager** and install **Slack Notification** plugin

Go to **Manage Jenkins,** click to **Configure System**



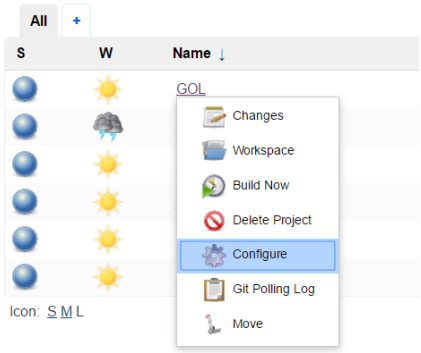
Search for section **Global Slack Notifier Settings,** and put in information for **Team Subdomain** and **Integration Token** (This infor get from previous step), then click to **Test Connection,** you will see connection successfully. Finally click to **Save** button.



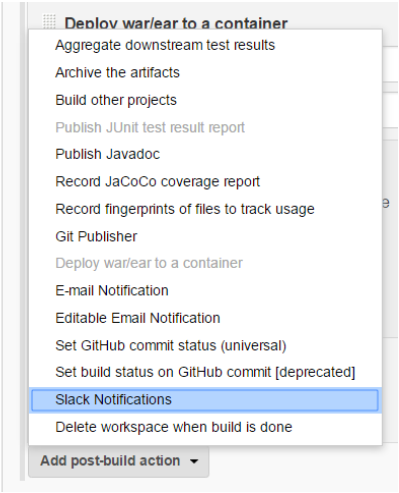
That it, now Slack notification is ready used inside Jenkins job.

Configure GOL Using Slack Notification

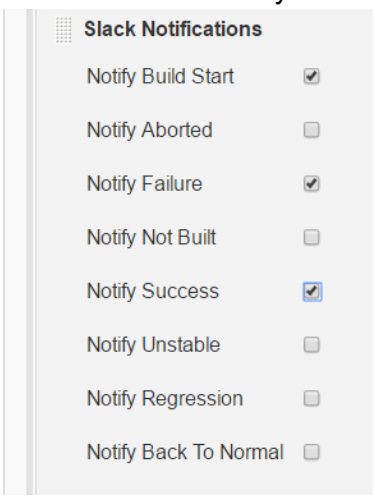
From Jenkins home page, click to **Configure** of GOL job



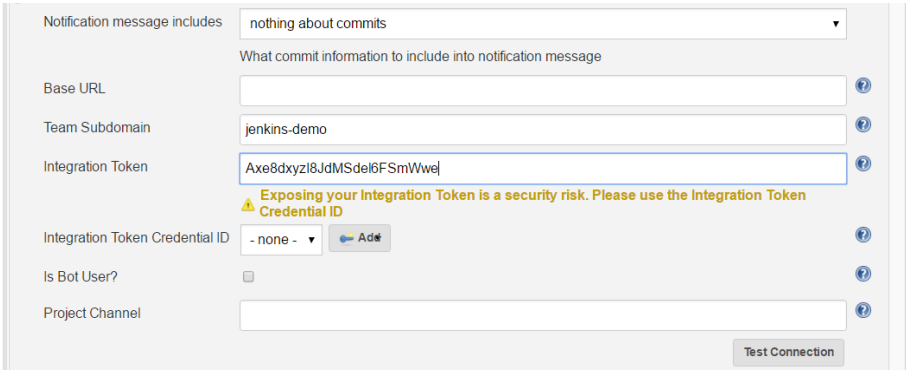
From **Configure** page, click to tab **Post-build Actions,** then add a **Slack Notifications** step.



Select when the notify will be send



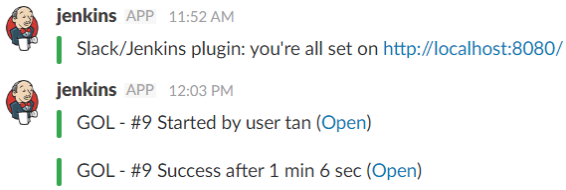
Click to **Advanced** and put information for **Team Subdomain** and **Integration Token.** Finally click to **Save** button.



Check if Notification Work

From GOL home page click to **Build Now**

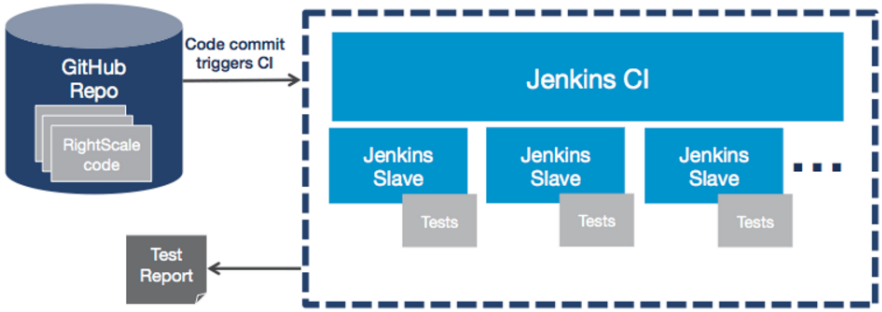
And notification message will be send to **#build** channel as expected



That it, from now on, GOL build and deploy job will be automatically watched by Slack Notification.

Distributed Build System with Jenkins

Jenkins Architecture

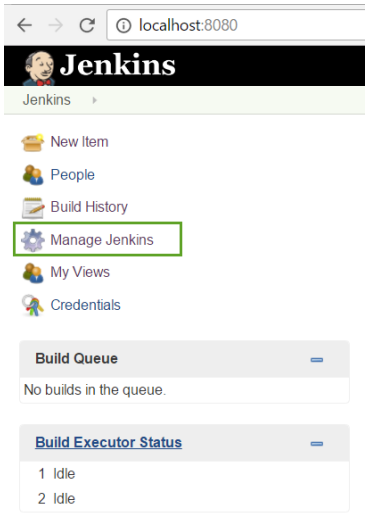


Normally, job will do not run on machine which install Jenkins but running on slave machines and these slaves machine are controlled by master machine where Jenkins is installed.

Windows Slave Agent

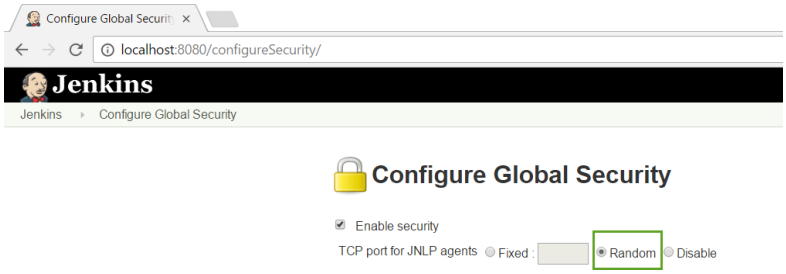
Enable Launch Slave Agents via Java Web Start

From Jenkins home page, click to ***Manage Jenkins***



Then click to ***Configure Global Security***

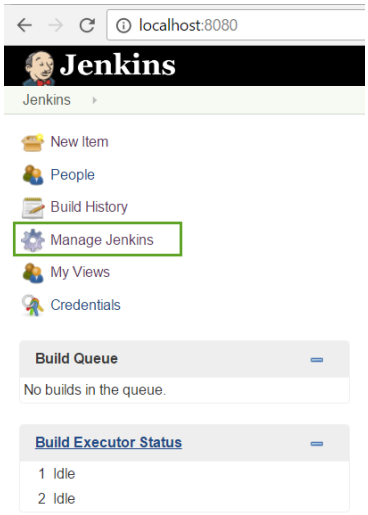
Select ***Random*** for option ***TCP port for JNLP agents***



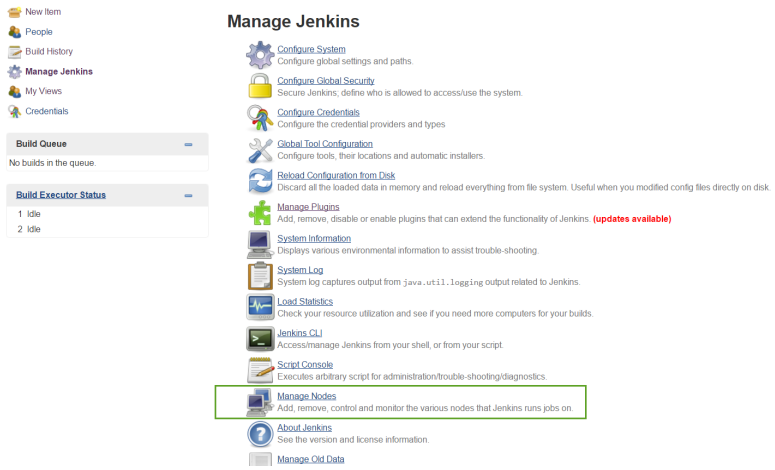
Then finally click ***Save*** button

Add New Windows Slave

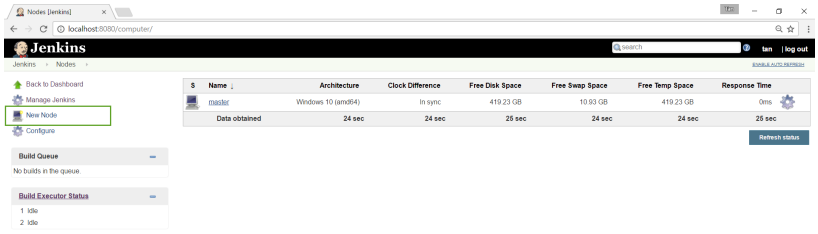
From Jenkins home page, click to ***Manage Jenkins***



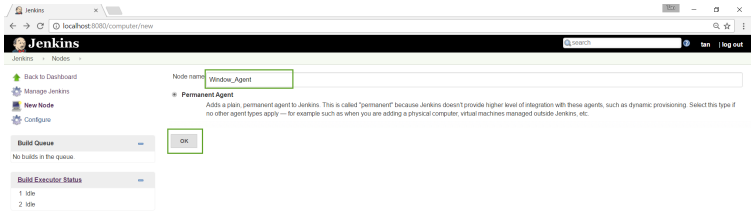
On ***Manage Jenkins*** page, scroll down and click to ***Manage Nodes***



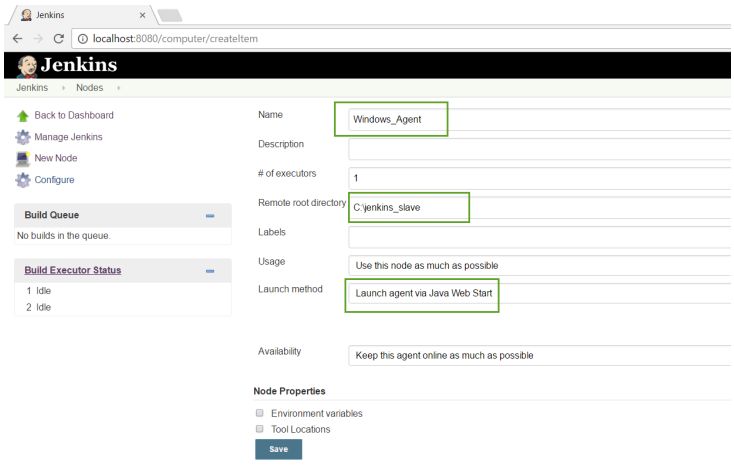
From there you can see all computer currently exist in system. We just have one master machine. To create new node, click to ***New Node***



Input agent name and click to **OK** button

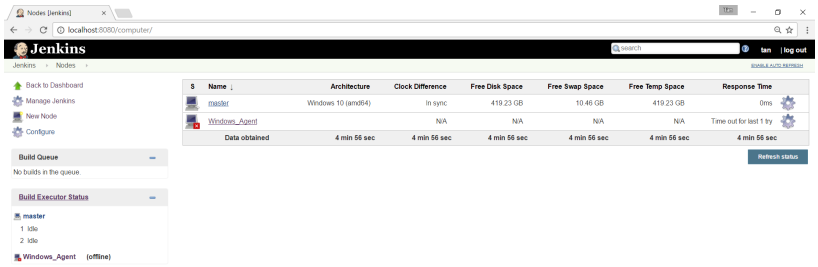


Input ***Remote root directory***, select ***Launch agent via Java Web Start,*** Finally click to ***Save*** button

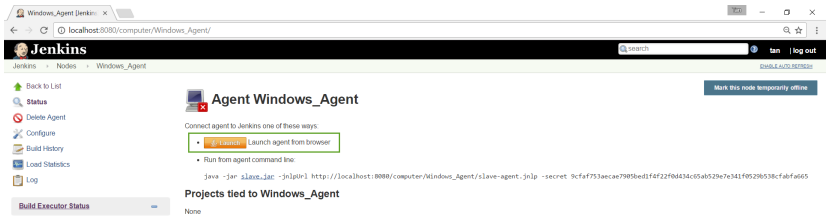


Launch Slave Agent

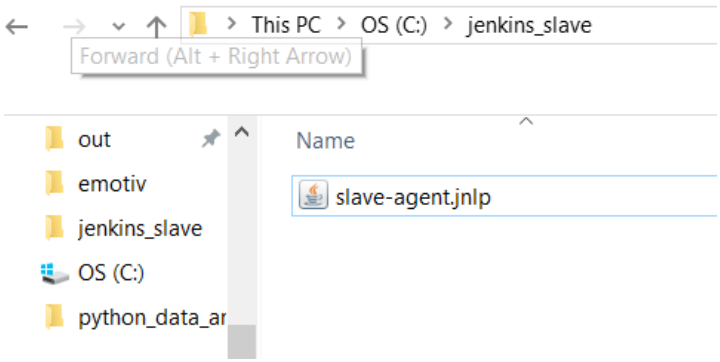
From computer which we want to install as slave (for demo, I use same machine for both master and slave) access to link http://localhost:8080/computer/
You could see that currently Window_Agent is disconnected with Jenkins.



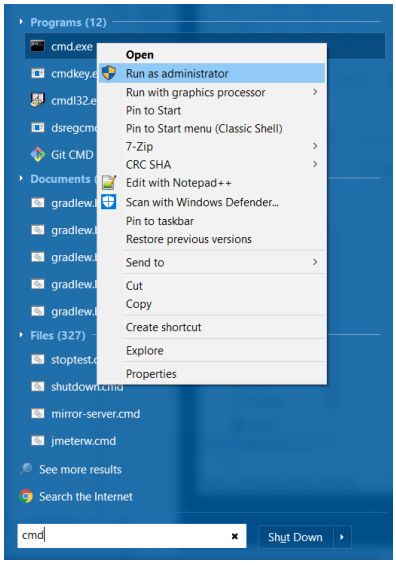
Click to ***Windows_Agent*** and then click to ***Launch*** button



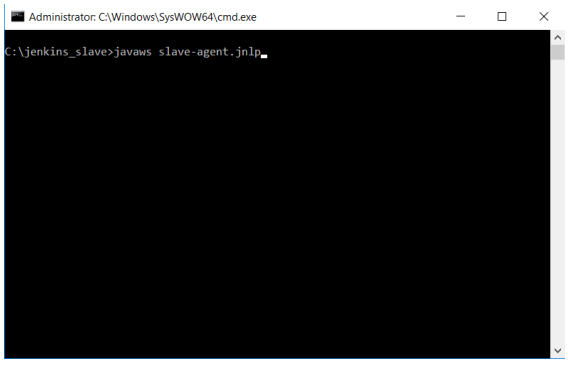
File ***slave-agent.jnlp*** will be downloaded to local, copy this file to ***C:\jenkins_slave***



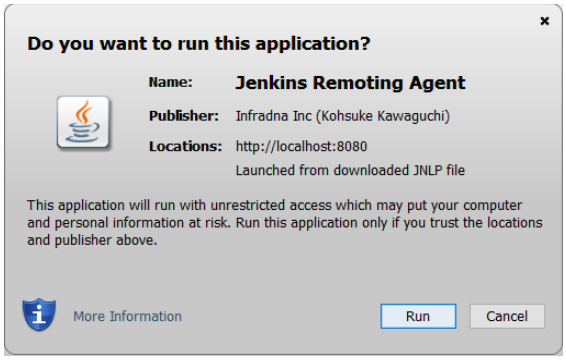
Start a command prompt as administrator.



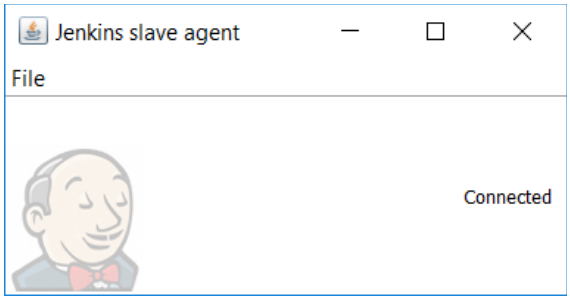
Change directory to ***C:\jenkins_slave*** then execute command ***javaws slave-agent.jnlp***



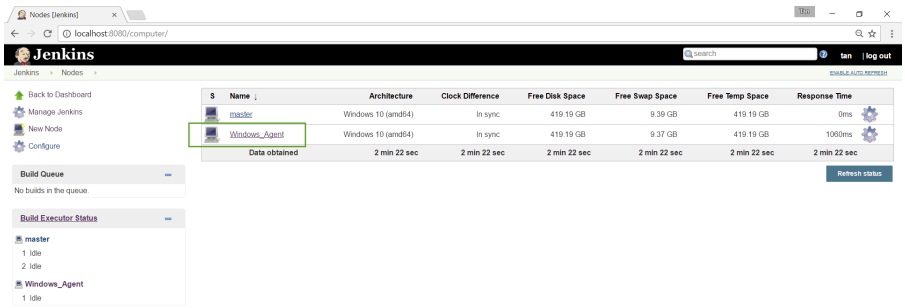
A dialog will show up, click to ***Run*** button



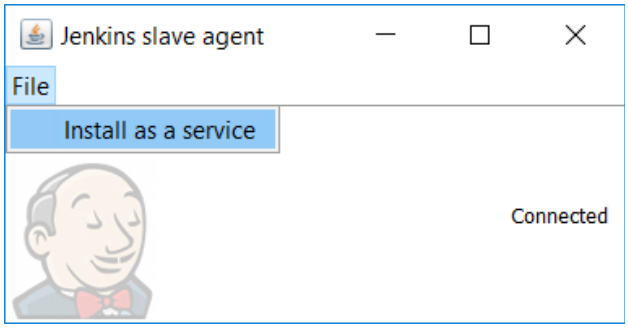
A dialog will show up and message that connected successfully



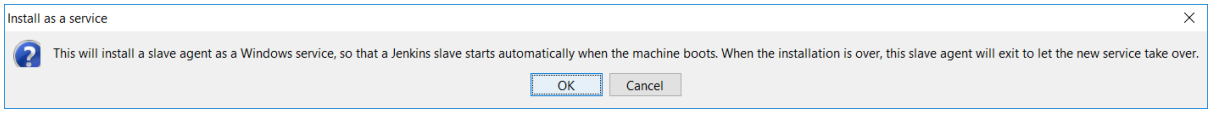
Go back to link http://localhost:8080/computer/ and refresh, you will see Window_Agent up and running successfully.



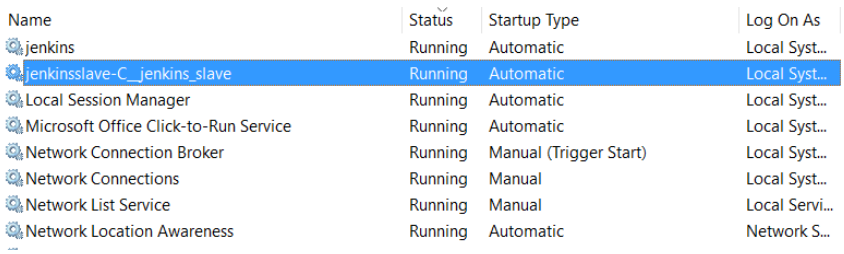
Now Window_Agent ready for use. But we will do one more step to setting up slave agent as a Windows service. From ***Jenkins slave agent*** dialog, click to ***File*** and then select ***Install as a service***



Then click ***OK*** for question



Current dialog will close up and new service for slave agent is installed successfully.

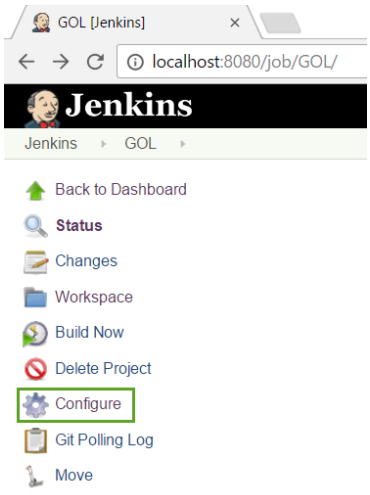


Name	Status	Startup Type	Log On As
jenkins	Running	Automatic	Local Syst...
jenkinsslave-C__jenkins_slave	Running	Automatic	Local Syst...
Local Session Manager	Running	Automatic	Local Syst...
Microsoft Office Click-to-Run Service	Running	Automatic	Local Syst...
Network Connection Broker	Running	Manual (Trigger Start)	Local Syst...
Network Connections	Running	Manual	Local Syst...
Network List Service	Running	Manual	Local Servi...
Network Location Awareness	Running	Automatic	Network S...

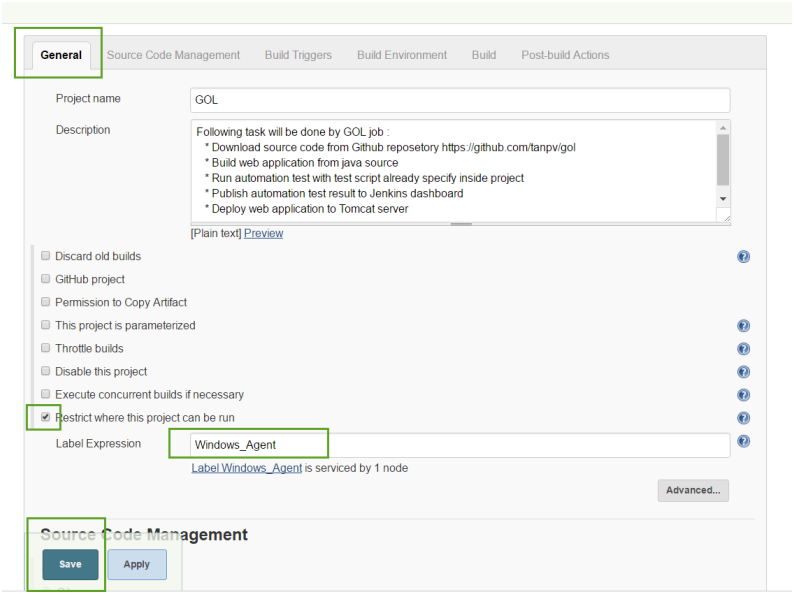
From now on, everytime machine power up, this machine will automatically connect to jenkins master and ready for running job.

Running Job with Windows Slave Agent

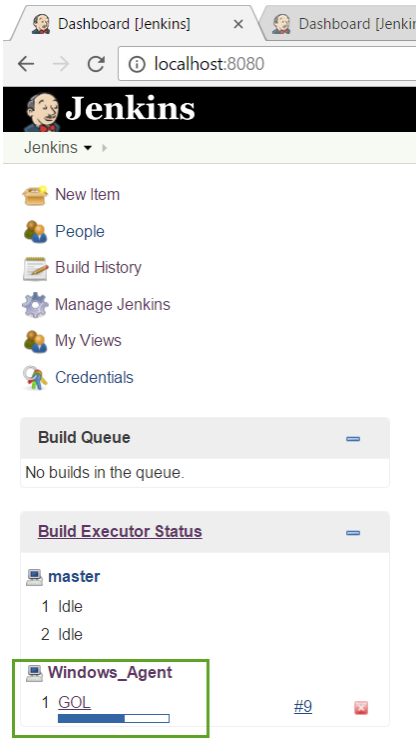
From GOL job dashboard, click to ***Configure***



From ***General*** tab, click to ***Restrict where this project can be run,*** then enter “Windows_Agent” in ***Label Expression***. This mean we only want GOL job running on machine represented by “Windows_Agent” . Then finally click to ***Save*** button.



Now try to run GOL from it's dashboard by click on **Build Now**
From Jenkins home page, you could see GOL running with **Windows_Agent** machine

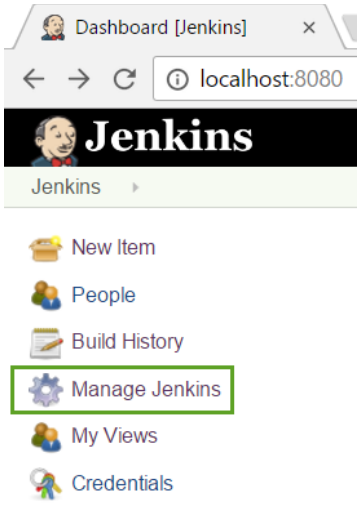


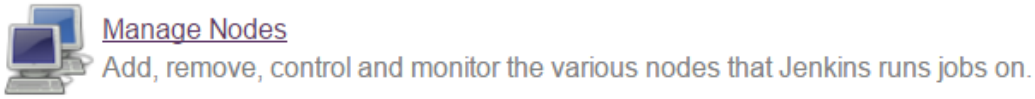
Ubuntu Slave Agent

Enable Launch Slave Agents via Java Web Start

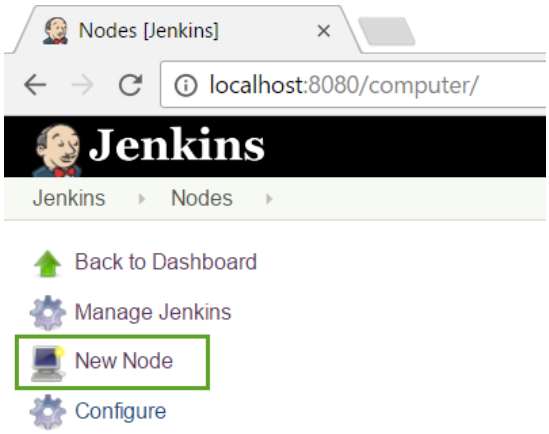
Add New Ubuntu Slave Agent

From jenkins home page, click to **Manage Jenkins,** then click to **Manage Nodes**

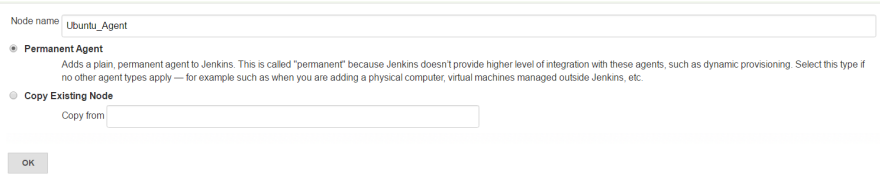




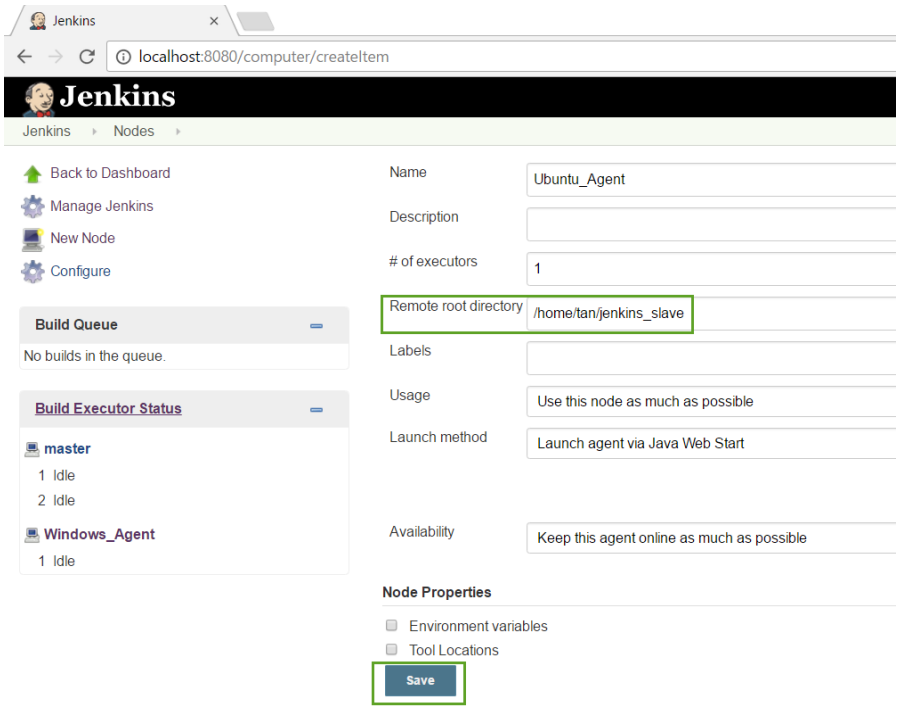
From **Nodes** screen, click to **New Node**



Put the name “Ubuntu_Agent”, click to **Save** button

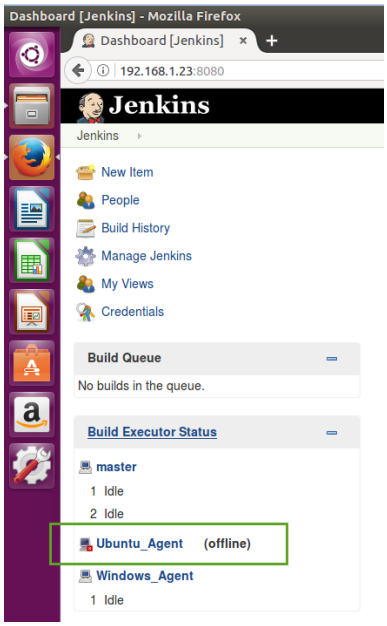


Specify **Remote root directory,** then click **Save** button



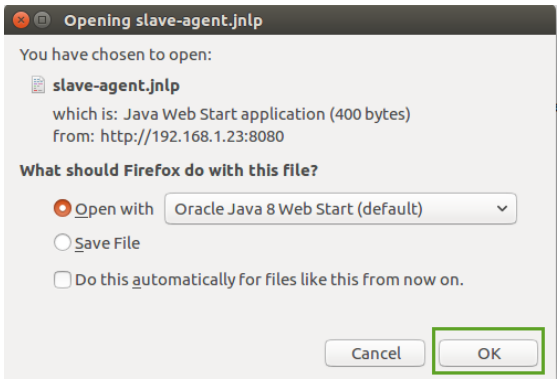
Launch Slave Agent

Go to Ubuntu machine which you want it become a jenkins agent and click to **Ubuntu_Agent**



Click to **Launch** button

Click to **OK** button



One dialog will show up that agent and master already connect

Now you will see **Ubuntu_Agent** up and running from Jenkins home page. And you could create job which need to running on Ubuntu machine.

Jenkins Pipeline

What is Jenkins Pipeline ?

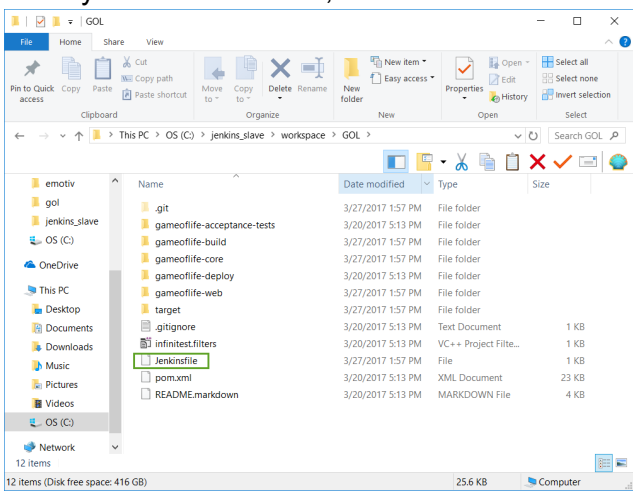
Jenkins Pipeline is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline DSL.

Following are common steps to use pipeline:

- Create a **Jenkinsfile** which describe what kind of command will run while pipeline running.
- Add this **Jenkinsfile** to the root of project on Github repository. **Jenkinsfile** use Groovy grammar.
- Create a new job with type **Pileline**
- Run pipeline job

Create a Jenkinsfile

From your local machine, create a new file with name **Jenkinsfile**

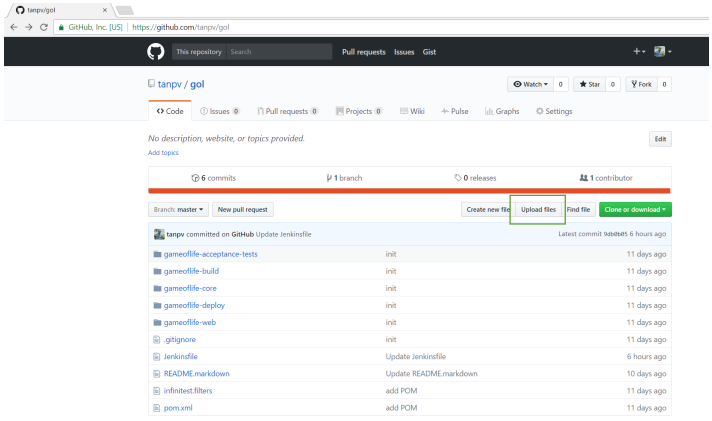


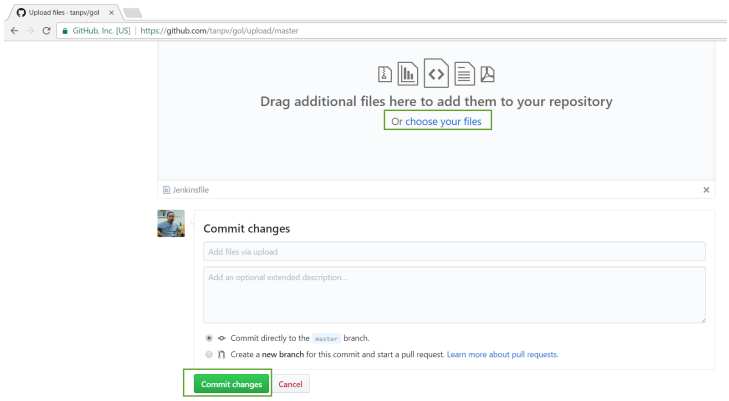
Put following code into **Jenkinsfile,** this code basically will run maven command to build web application.

```
// start of pipeline
pipeline {
  // where pipeline job will run
  agent any
  // start of stages : build, test, deploy ...
  stages {
    // start of stage : build
    stage('build') {
      // start of running steps inside one stage
      steps {
        // invoke command to build with maven
```

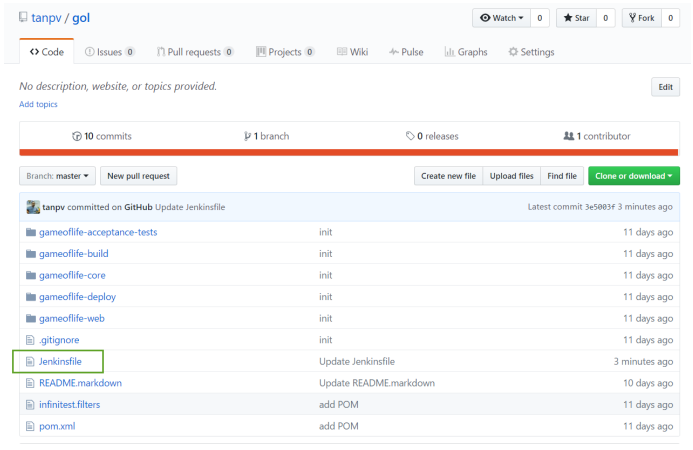
```
            bat 'mvn clean install'
        }
    }
  }
}
```

Access to GOL github repo from browser, click to **Upload files,** then drag drop **Jenkinsfile** which already created. Finally click to **Commit changes**



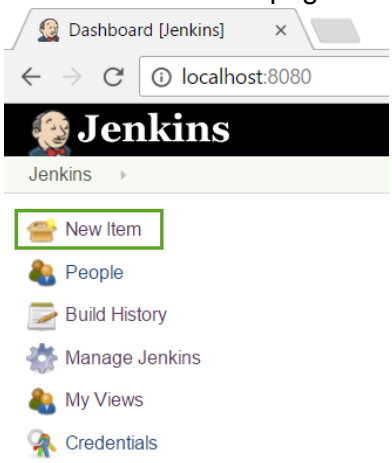


Wait some time and you will see the file show up inside GOL repository

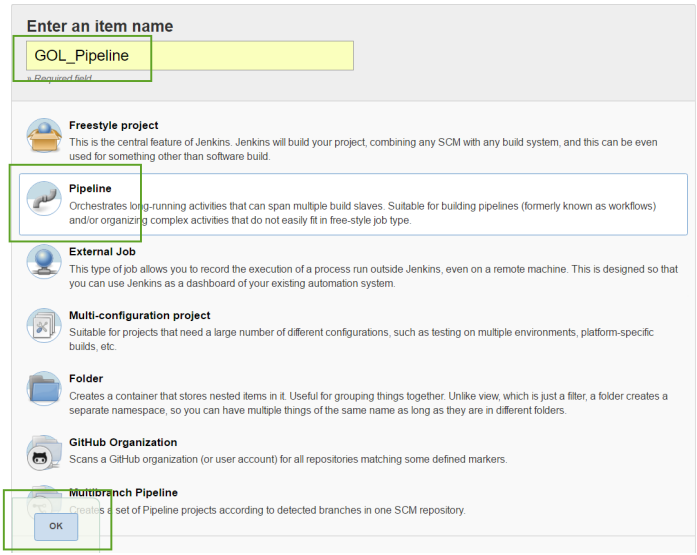


Create Pipeline Job

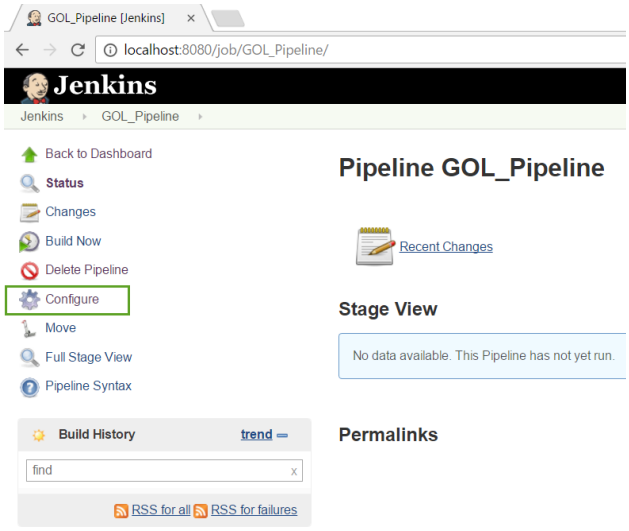
From Jenkins home page click to **New Item**



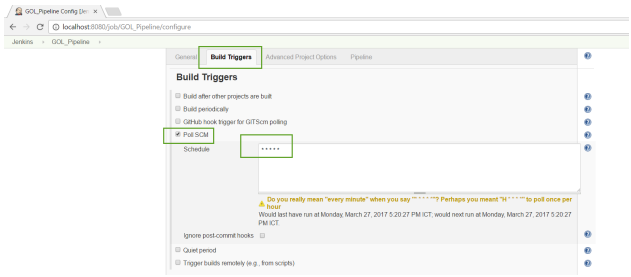
Typing job name GOL_Pileline, select **Pipeline** job style, and click to **Save** button



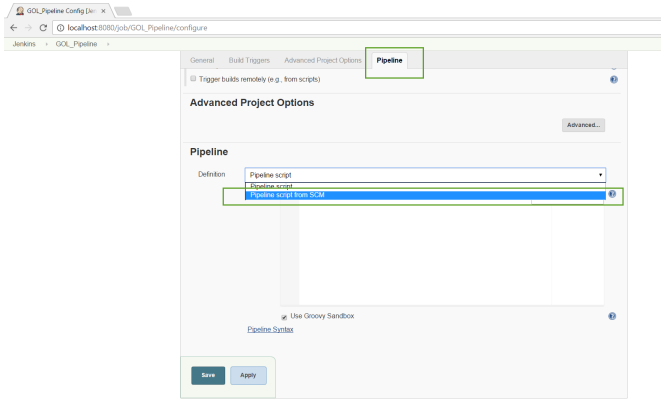
From job dashboard GOL_Pipeline, click to **Configure**



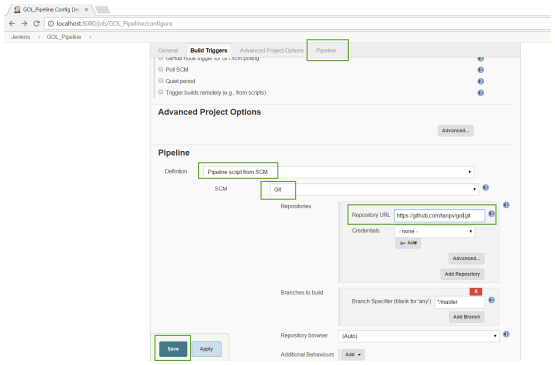
Select **Build Triggers** tab, select **Post SCM** check box, then put in following string to check change every minute on github **“* * * * *”**



Select **Pipeline** tab, then select **Pipeline script from SCM**



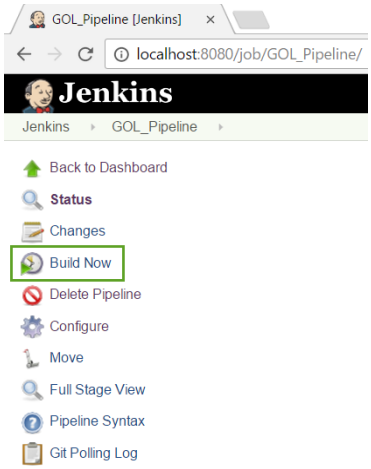
From **SCM** dropbox, select **Git,** then put in GOL project repository url https://github.com/tanpv/gol.git , finally click to **Save** button.



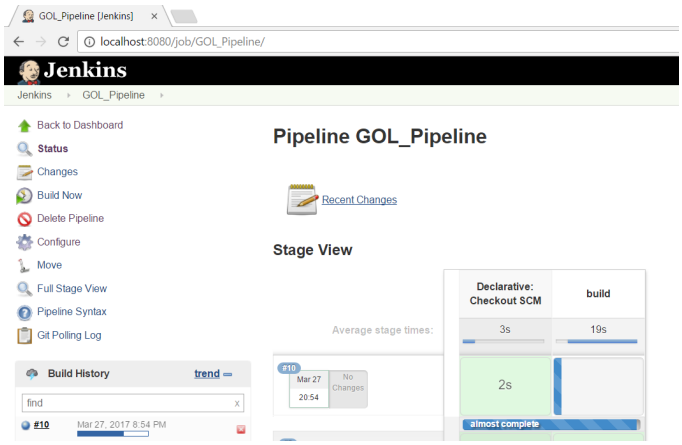
That it, we already complete create the first pipeline job with Jenkins!

Build with Pipeline

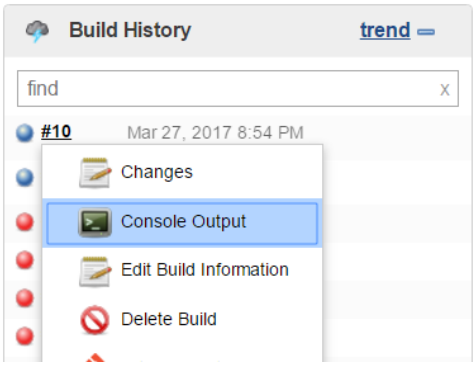
From GOL dashboard, click to **Build Now**



You will see job run with quite nice progress show up



Click to **Console Log,** build log will show up with build success for GOL project.



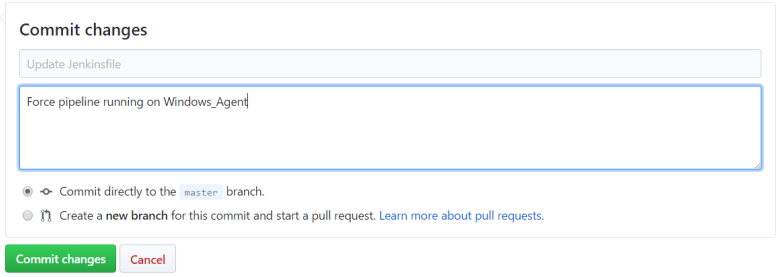

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] gameoflife ......................................... SUCCESS [  2.677 s]
[INFO] gameoflife-build ................................... SUCCESS [  1.581 s]
[INFO] gameoflife-core .................................... SUCCESS [  5.542 s]
[INFO] gameoflife-web ..................................... SUCCESS [  5.501 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 15.575 s
[INFO] Finished at: 2017-03-27T20:55:16+07:00
[INFO] Final Memory: 39M/441M
[INFO] ------------------------------------------------------------------------
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Running Pipeline On Specific Agent

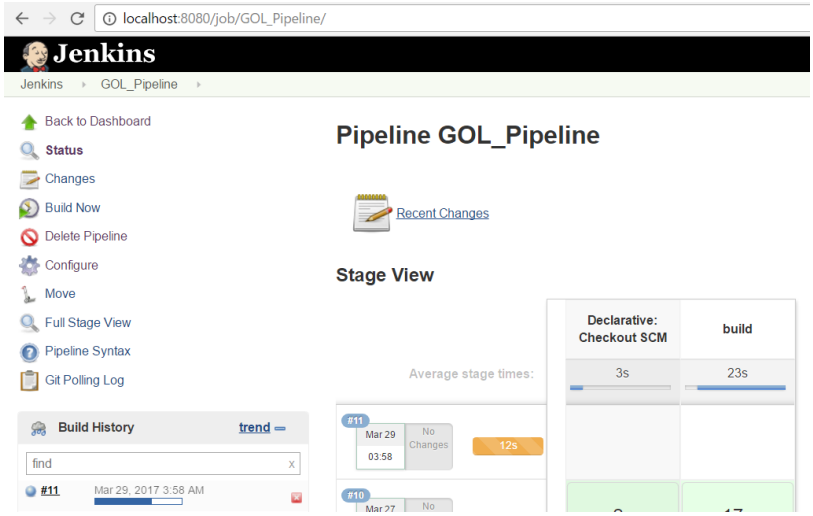
Edit **Jenkinsfile** on Github with following content, basically this code will force pipeline to only running on Windows_Agent

```
// start of pipeline
pipeline {
  // where pipeline job will run
  agent {
    // force pipeline job to running on windows_agent
    label "Windows_Agent"
  }
  // start of stages : build, test, deploy ...
  stages {
    // start of stage : build
    stage('build') {
      // start of running steps inside one stage
      steps {
        // invoke command to build with maven
        bat 'mvn clean install'
      }
    }
  }
}
```

Click to **Commit changes**



You will see GOL_Pipeline automatically run due to it saw code change from Github



And from **Console Output,** you can see that GOL_Pipeline job is running with "Windows_Agent"

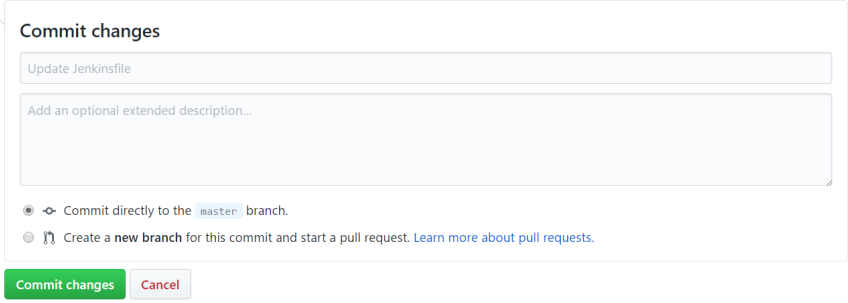
Deploy to Tomcat with Pipeline

Edit **Jenkinsfile** on GOL repository with following code, this code add deployment to Tomcat server step.

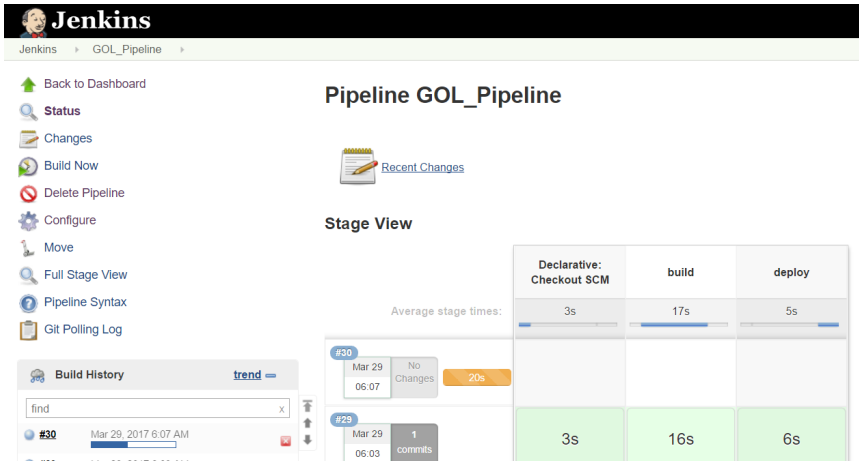
```
// start of pipeline
pipeline {
```

```
  // where pipeline job will run
  agent {
    // force pipeline job to running on windows_agent
    label "Windows_Agent"
  }


  // start of stages : build, test, deploy ...
  stages {

    // start of build stage
    stage('build') {
      // define step to run
      steps {
        // invoke command to build with maven
        bat 'mvn clean install'
      }
    }

    // start of deploy state
    stage('deploy') {
      // define step to run
      steps {
        //invoke command to stop tomcat service
        bat 'sc stop Tomcat7'
        bat 'ping 127.0.0.1 -n 6'
        // copy war file from build target to webapp Tomcat folder
        bat 'xcopy /y
C:\\jenkins_slave\\workspace\\GOL_Pipeline\\gameoflife-
web\\target\\gameoflife.war "C:\\Program Files\\Apache Software
Foundation\\Tomcat 7.0\\webapps"'
        //invoke command to start tomcat service
        bat 'sc start Tomcat7'
      }
    }
  }
}
```

Commit change of **Jenkinsfile** to Github right from browser



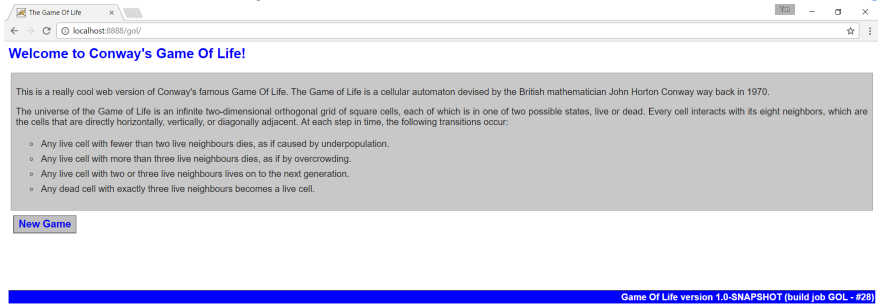
Wait for about 1 minute and GOL_Pipeline job will automatically build



From **Console Output** will see deploy to Tomcat7 step

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (deploy)
[Pipeline] bat
[GOL_Pipeline] Running batch script
```

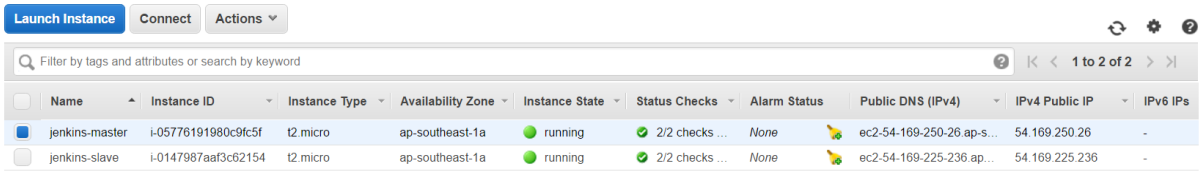
Check the site and you will see it work fine at http://localhost:8888/gol



Note for auto build trigger after commit

Jenkins in the Cloud

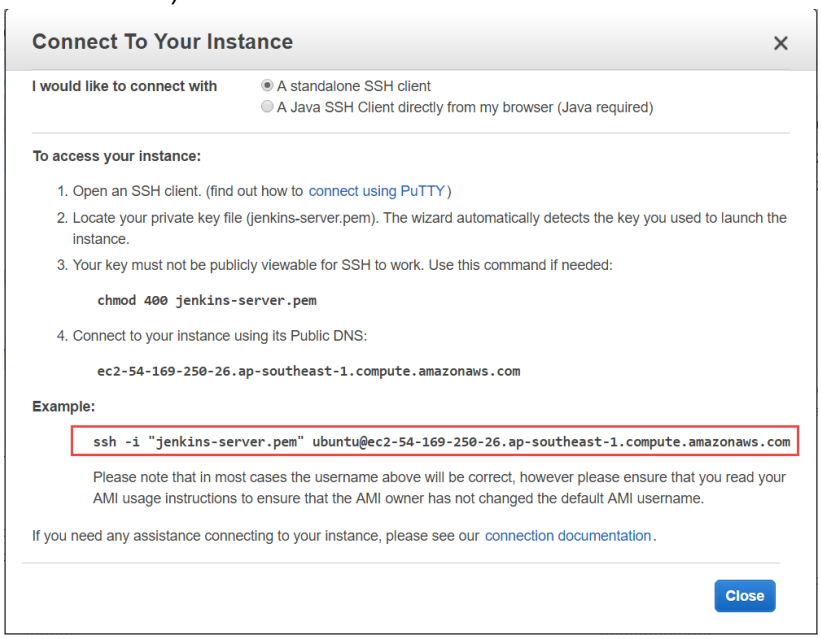
From AWS console creae 2 Ubuntu machine, one machine will be Jenkins master and one machine will be Jenkins slave



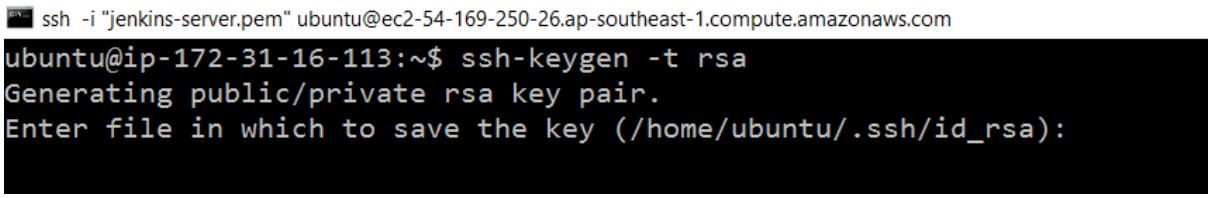
Create a Key to Connect between Master and Slave

The purpose is master should successfully connect to slave with ssh command.

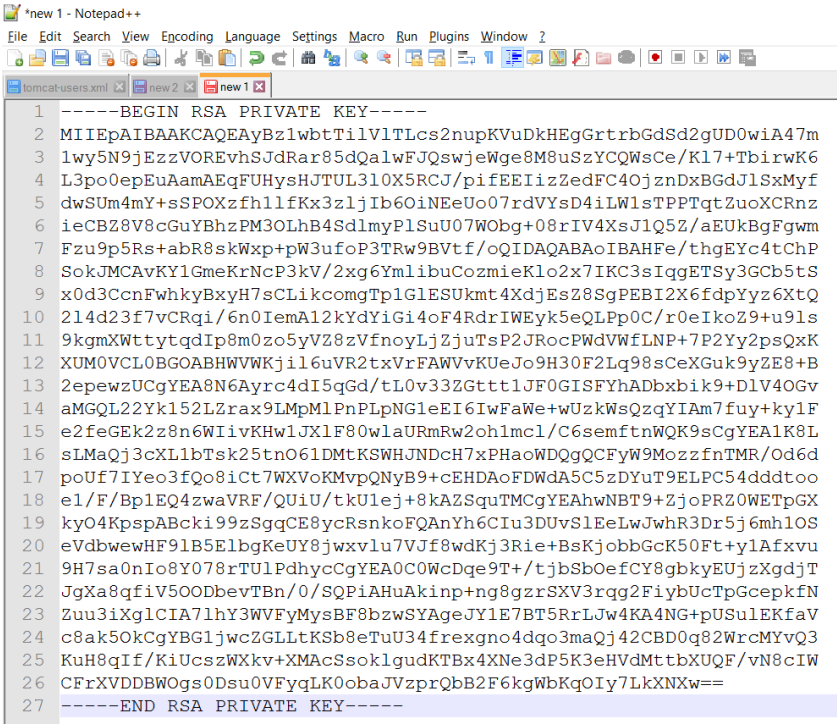
Using ssh and key pair file in order to connect to master machine (I already install ssh on my local window).



Create a publish key from master machine with command
ssh-keygen -t rsa the private key will be store at ***/home/ubuntu/.ssh/id_rsa***



Store the private key to a file for later use



Open the publish key on master machine with command ***cat ~/.ssh/id_rsa.pub***

```
ubuntu@ip-172-31-16-113:~$ cat  ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDIHPXBu1OKVWVMtyzae6kpW4OQcSAau2tsZ1J3aBQPTCIDjubXDLk32MTPNU5ES+FIl1Fqvzl1BqXAUlCzCN5aB7wzy5LNgJBawJ78qXv5NuKvArovemjR6k
S4BqYASoVQfKwclNQveXRflEIn+mJ8QQiLNl50ULg6POcPEEZ0mVLEzJ93BJSbiZj6xI85fN+HWV8rHfOWMhvo6I0R5SjTut1ViwPiItbWxM89Oq1m6hcJGfOJ4IFnxXxwa5gGHM8zc4uEHhJ2WbI+VK5TTtY5
uD7TyshXhewnVDln9oRSQGAWDCYXO72nlGz5ptHyyRbGn6lbe5+g/dNHD0FW1/+h ubuntu@ip-172-31-16-113
ubuntu@ip-172-31-16-113:~$
```

Save this content to file ***~/.ssh/authorized_keys*** on slave machine with ***echo*** command

```
ubuntu@ip-172-31-22-93:~$ echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDIHPXBu1OKVWVMtyzae6kpW4OQcSAau2tsZ1J3aBQPTCIDjubXDLk32MTPNU5ES+FIl1Fqvzl1BqXAUlCzCN5aB
7wzy5LNgJBawJ78qXv5NuKvArovemjR6kS4BqYASoVQfKwclNQveXRflEIn+mJ8QQiLNl50ULg6POcPEEZ0mVLEzJ93BJSbiZj6xI85fN+HWV8rHfOWMhvo6I0R5SjTut1ViwPiItbWxM89Oq1m6hcJGfOJ4I
FnxXxwa5gGHM8zc4uEHhJ2WbI+VK5TTtY5uD7TyshXhewnVDln9oRSQGAWDCYXO72nlGz5ptHyyRbGn6lbe5+g/dNHD0FW1/+h ubuntu@ip-172-31-16-113" >> ~/.ssh/authorized_keys
```

Now from master machine, try to ssh connection to slave machine. It should be successfully connectd.

```
ubuntu@ip-172-31-16-113:~$ ssh ubuntu@172.31.22.93
The authenticity of host '172.31.22.93 (172.31.22.93)' can't be established.
ECDSA key fingerprint is SHA256:OmzfpJbCxhUGo2I28WyDA8CVZKwCZ/JgchzcYNixSsE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.22.93' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

20 packages can be updated.
0 updates are security updates.
```

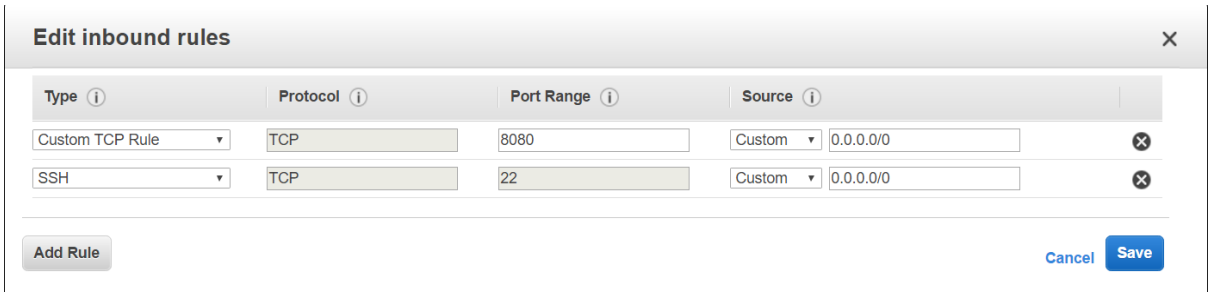
Install Jenkins Server on Cloud

Running following command from master ssh console

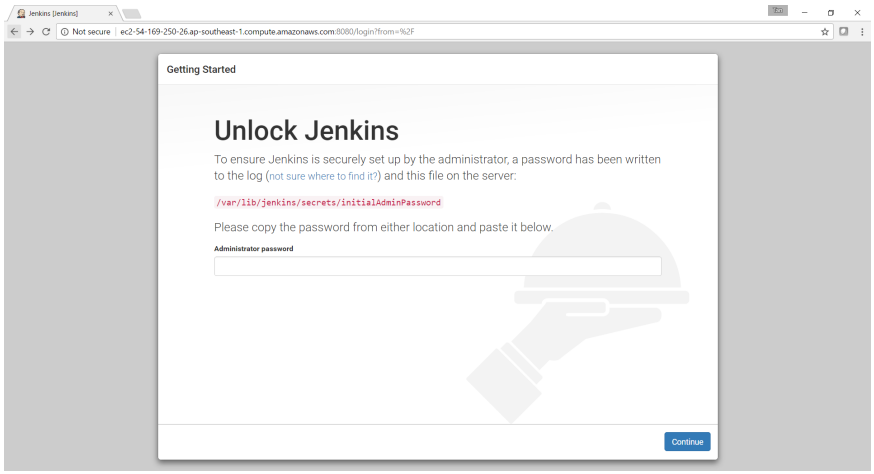
```
$ wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo
apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ >
```

```
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install jenkins
$ sudo service jenkins start
```

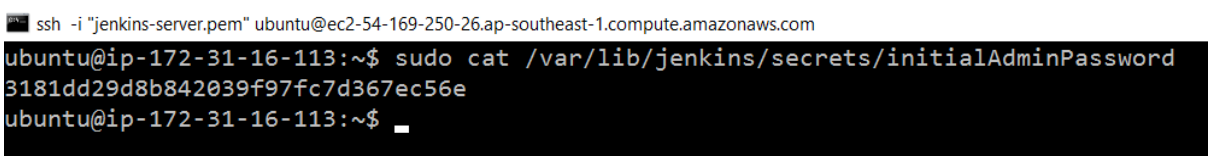
Add port 8080 to inbound rules to master machine



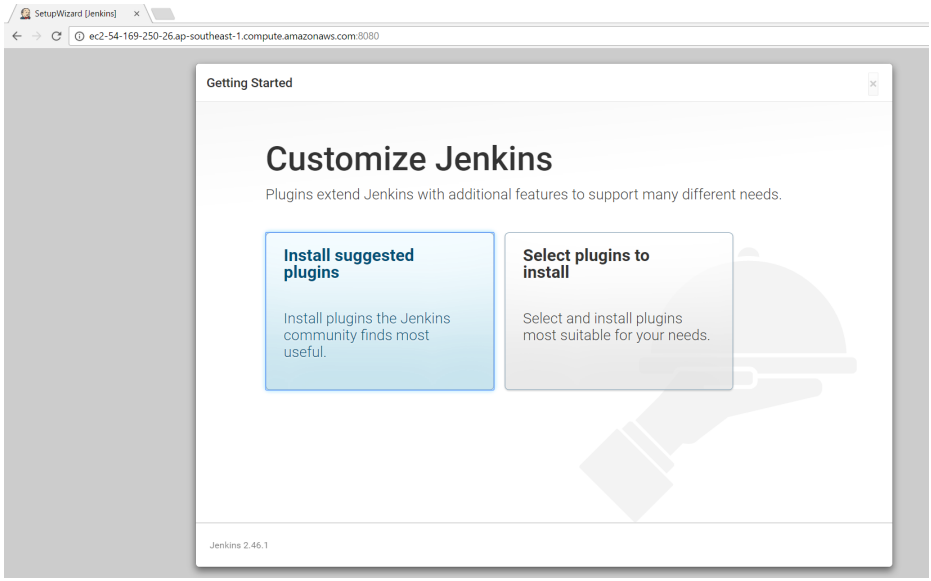
Access to Jenkins from local browser with port 8080



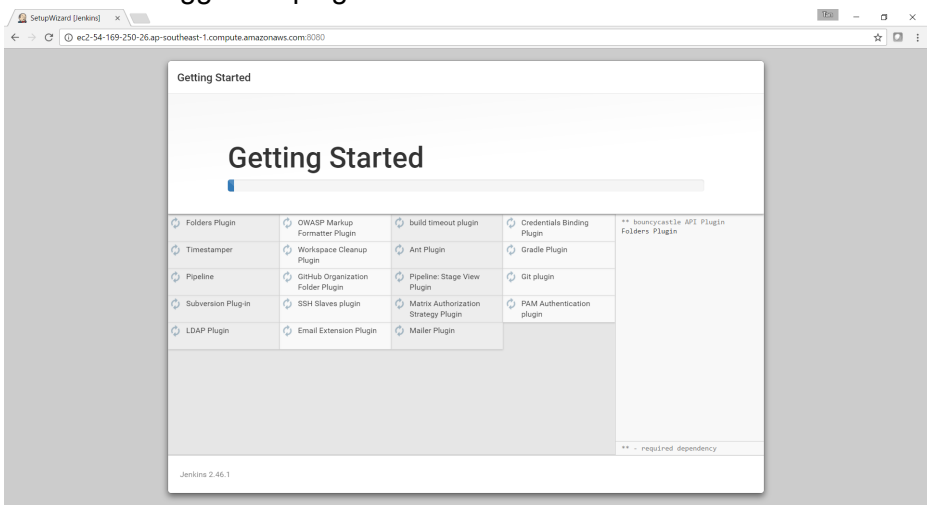
From command prompt using command to get default admin pass word
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

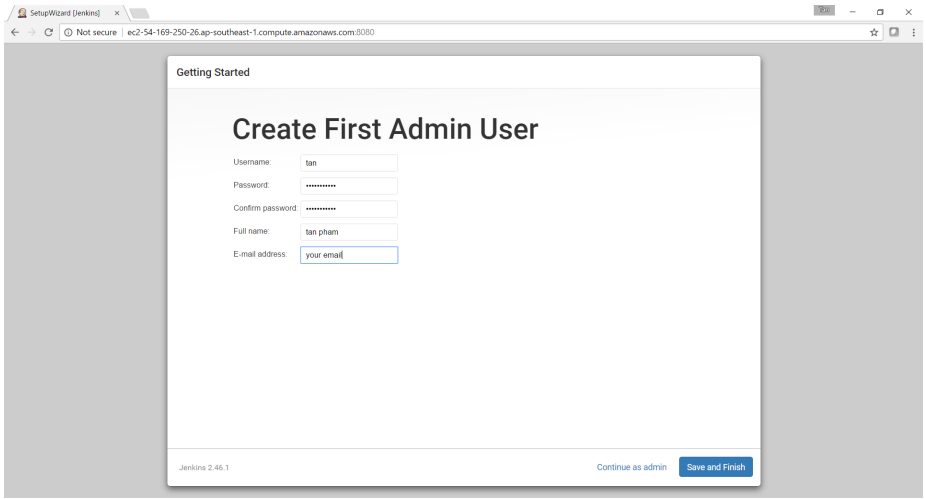


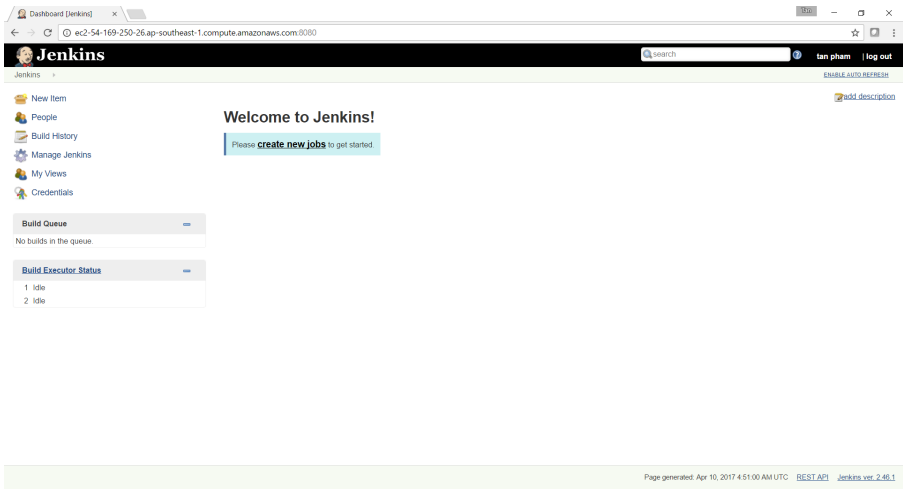
Put default password to browser to unlock Jenkins



Install with suggested plugins







Install Java on Slave Machine

SSH into slave nodes, upgrade the packages, and install a Java Runtime Environment:

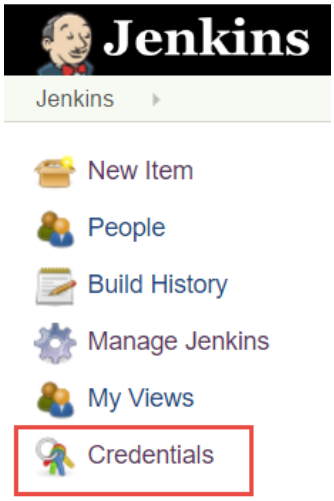
```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install default-jre
$ java -version
```

After install successfully, java version should be show correctly. That's the only package Jenkins needs for slaves by default.

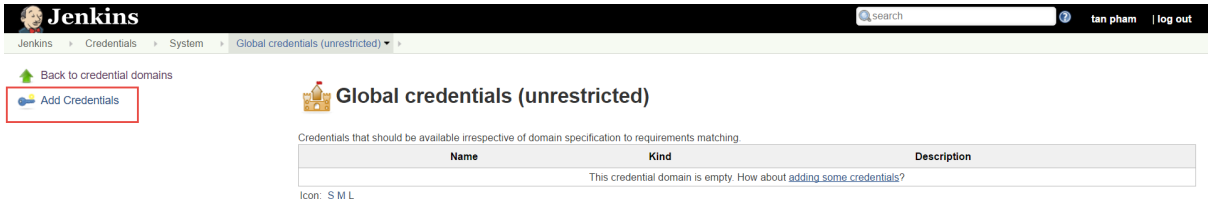
Configure a New Slave Agent from Jenkins

Create a new credential

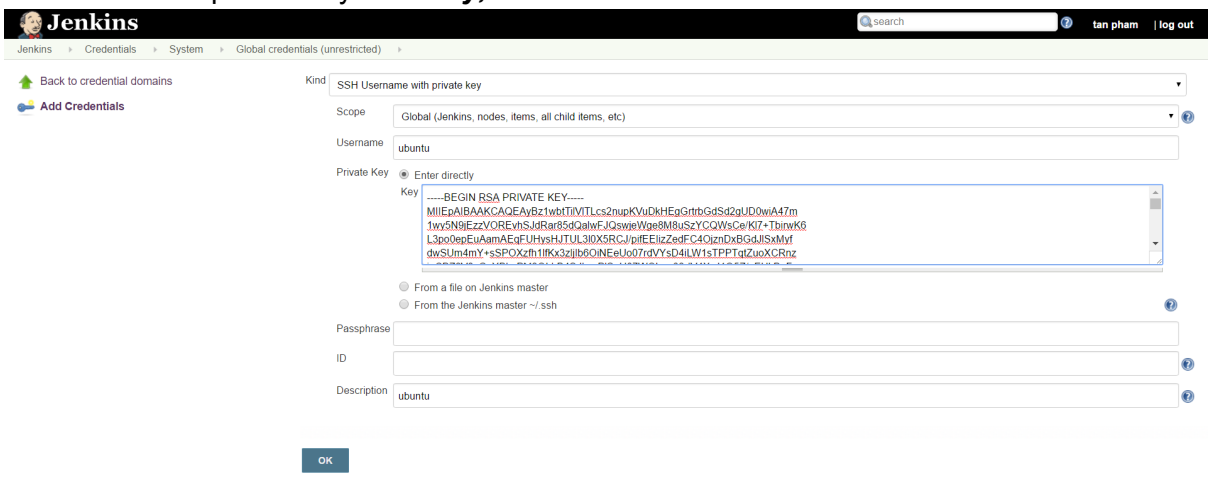
Click to ***Credential*** from Jenkins home page



From ***Global credentials*** click to ***Add Credential***

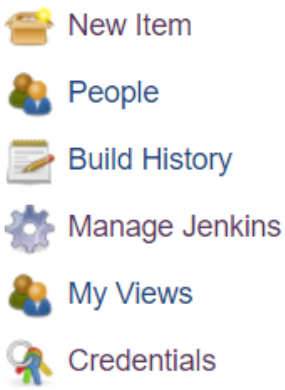


Add the master private key in to ***Key,*** then ***OK***

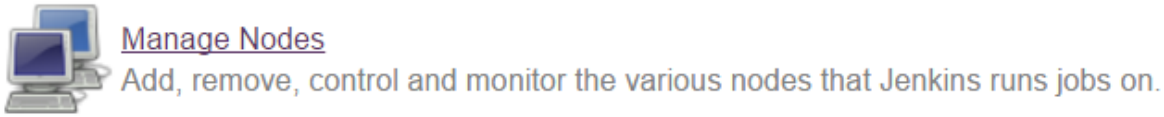


Create a new node

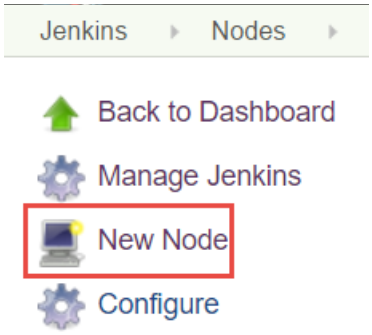
Click to ***Manage Jenkins***



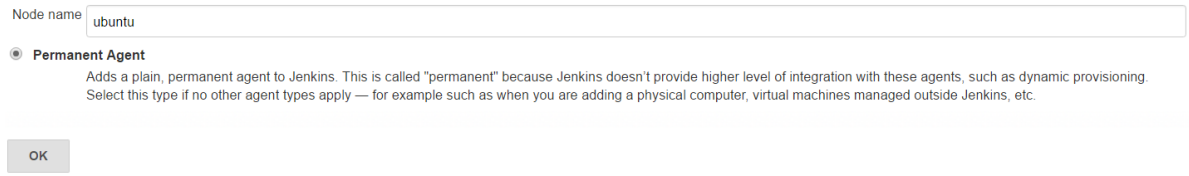
Click to ***Manage Node***



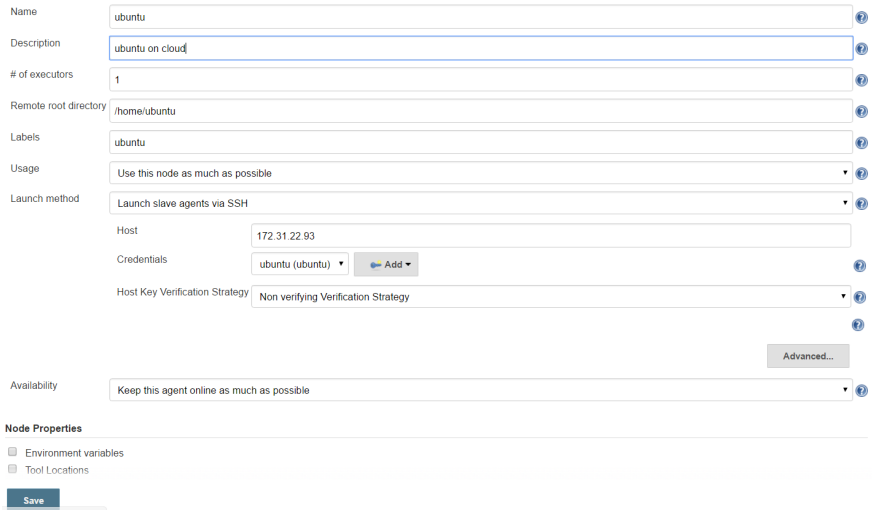
Then ***New Node***



Then put in node name



Put slave ip at ***Host*** and use all option as selected, finally click to ***Save***



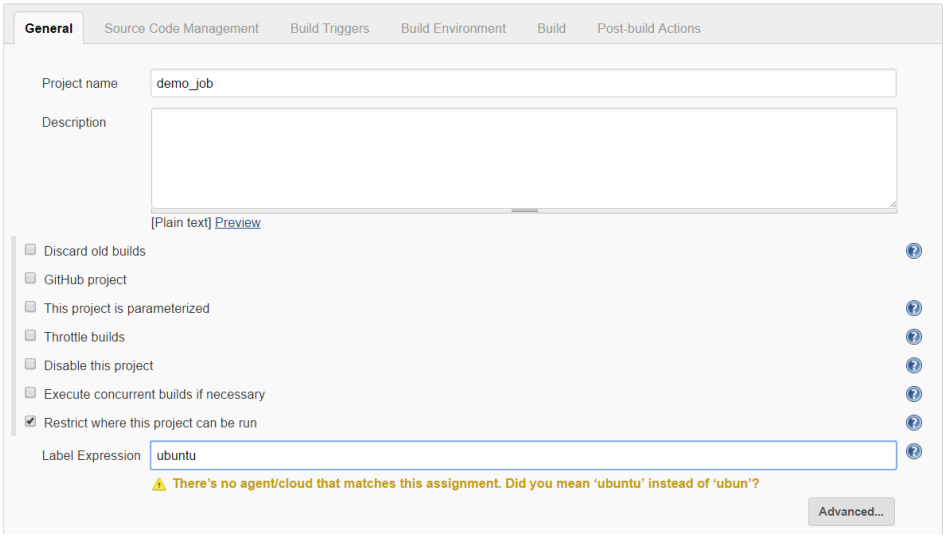
Click to Lanch Node, and connect successfully

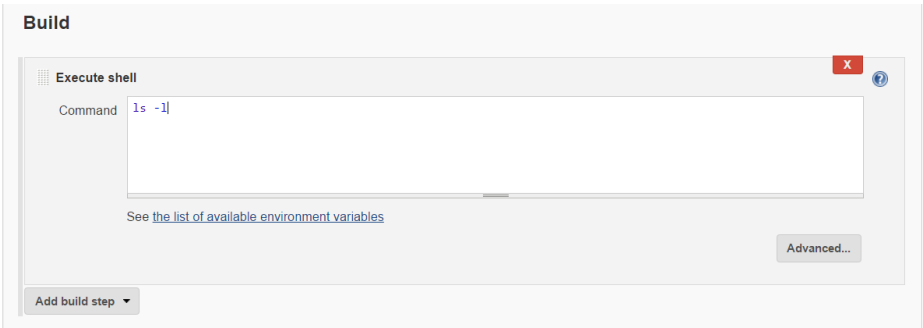
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.64 GB	0 B	5.64 GB	0ms
	ubuntu	Linux (amd64)	In sync	5.88 GB	0 B	5.88 GB	3012ms
	Data obtained	41 sec	41 sec	41 sec	41 sec	41 sec	41 sec

Refresh status

Create a Job Using Running on Cloud

Create a job which restricted node running with ***ubuntu***



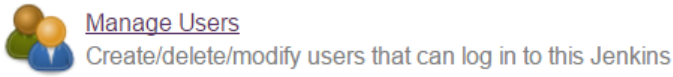


Run the job and see if the job running

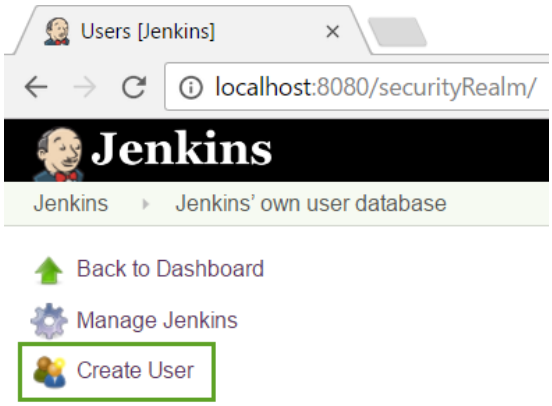
Jenkins Security

Create gol_developer user

From **Manage Jenkins,** click to **Manage Users**



Click to **Create User**



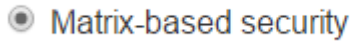
Add information for new user

Then new user is created

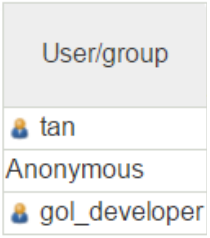
Matrix-based Security

Click **Configure Global Security**

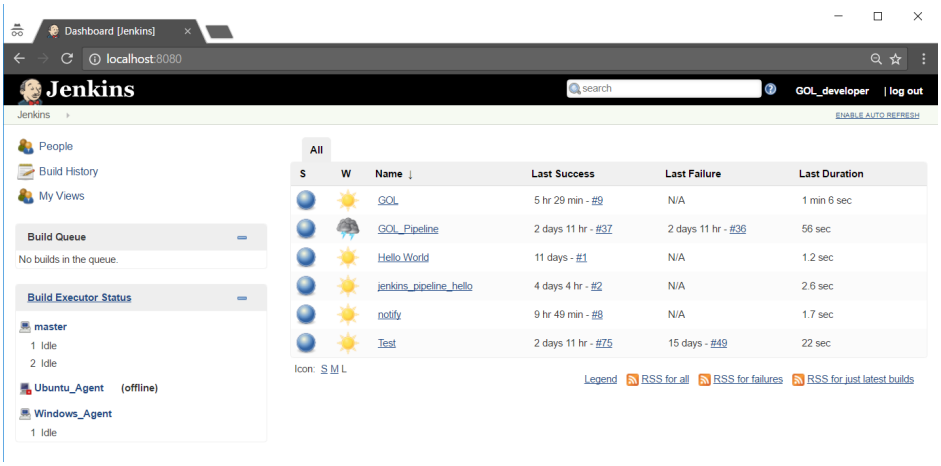
Click to **Matrix-based security**



Setting all permission for **your user,** and try to setting for user **gol_developer** overall read and job read permission. Finally click to **Save** button.



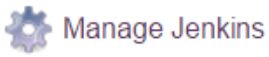
Logout from your account, then try to login with **gol_developer** user. Now GOL_developer could read the jobs only.

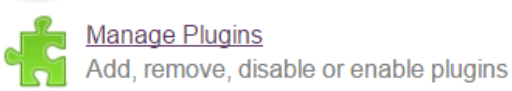


Role-based Security

Install Role-based Plugin

From home page click to **Manage Jenkins,** then **Manage Plugin.**

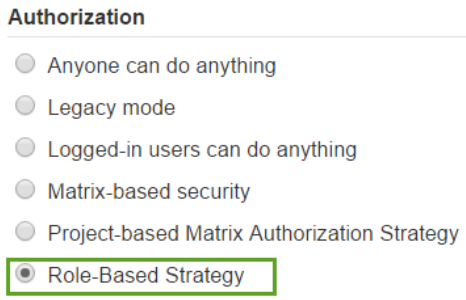




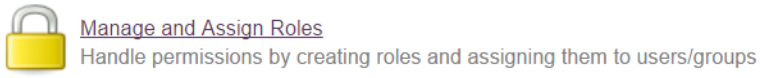
At **Manage Plugin** page, select tab **Available,** then typing into filter **Role-based Authorization Strategy**, click to **Install without restart**

Configure Global Role

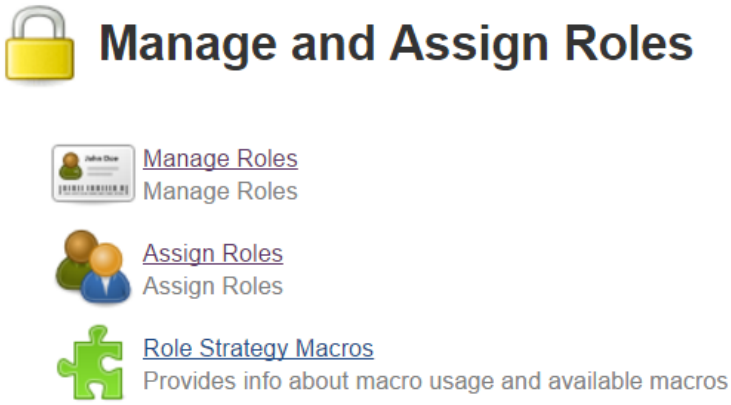
From **Manage Jenkins,** configure to **Configure Global Security,** and then at session **Authorization** select **Role-based Strategy** option. Then click to **Save** button.



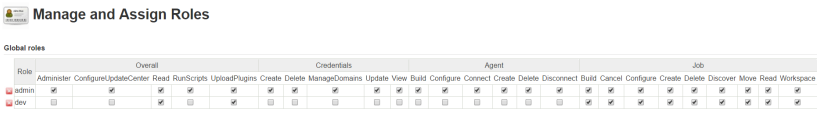
Now from **Manage Jenkins,** you will see new item call **Manage and Assign Roles**



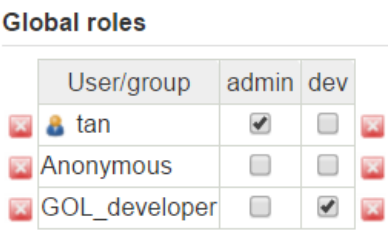
Click to **Manage and Assign Roles,** UI allow to **Manage Roles and Assign Roles** will show up



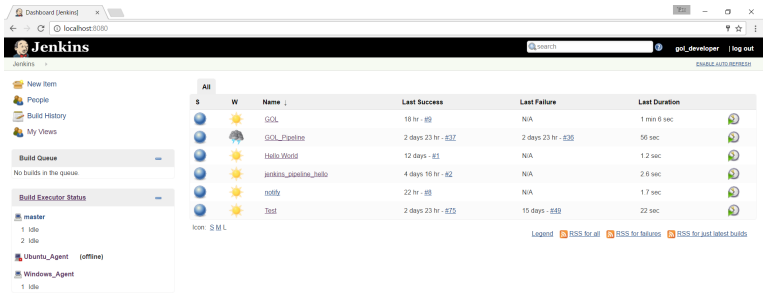
Click to **Manage Roles,** at **Global roles,** add **dev** role with read permission at **Overall** and full permission with Job, then click to **Save**



Back to screen **Manage Roles and Assign Roles,** then click to **Assign Roles.** Now assign GOL_Developer to **dev** role. Click to **Save**

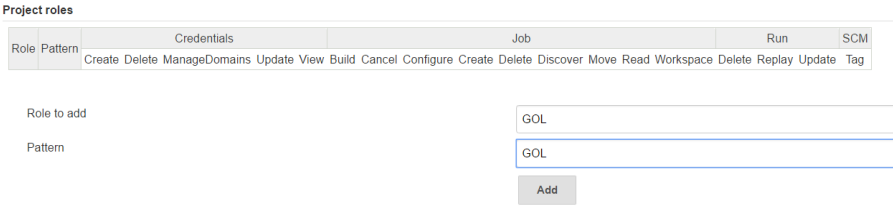


That it, now try to logout from Jenkins and login with user GOL_developer. You will see GOL_developer could see all the job, but could not see **Manage Jenkins**

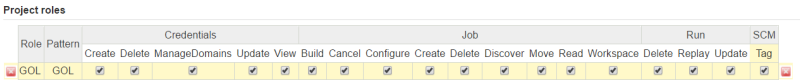


Configure Project Role

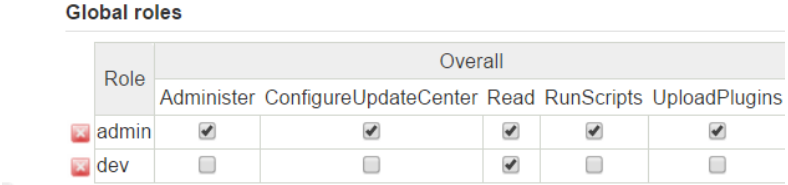
Currently, GOL_developer could see and access all jobs, You want to configure so GOL_developer only could see GOL project.
From **Manage and Assign Roles** screen, click to **Manage Roles**, at **Project roles** section, add GOL



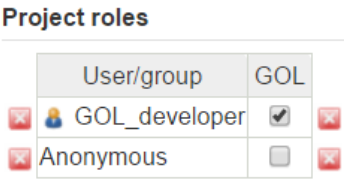
And then select all permission for GOL role. Click **Save** button.



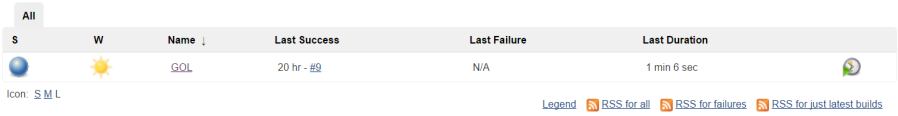
From **Global roles** section, remove all permission related to job and only keep overall read permission.



Now come back to **Assign Role** screen, then at the **Project Role** section, add GOL_developer user, and select GOL role for this user. Save configure.

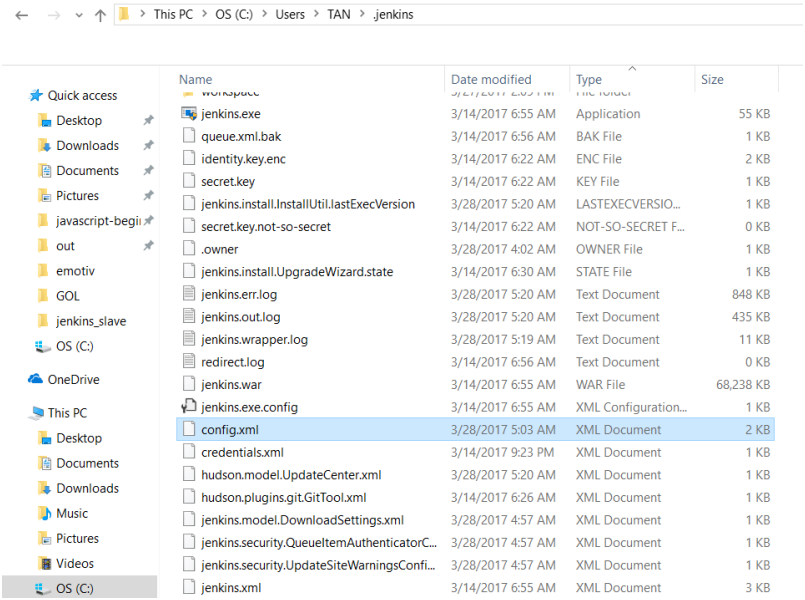


Logout from your admin and login back with GOL_developer. Now you can see that GOL_developer only see GOL project.

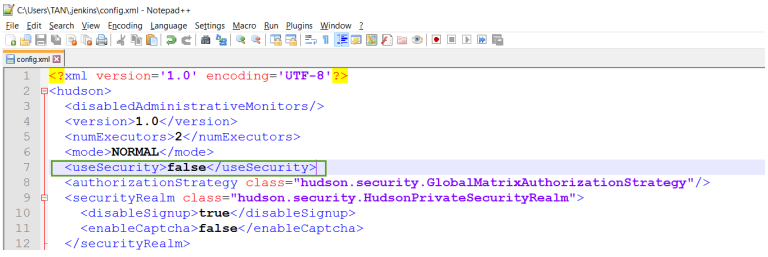


Disable Security

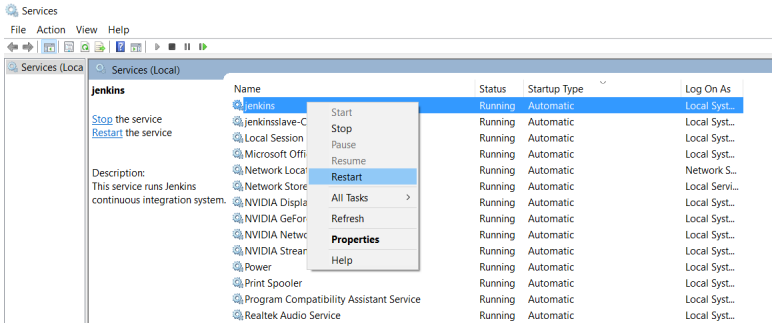
In some case, you do not need any security (like a personal build system), you just want to disable Jenkins security. To do this, go to place where Jenkins is installed, edit file **config.xml**



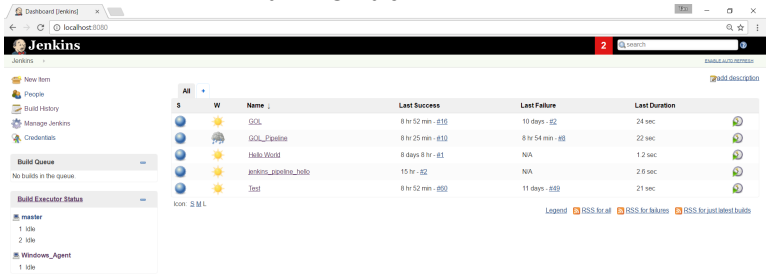
Change value of attribute **useSecurity** to **false**



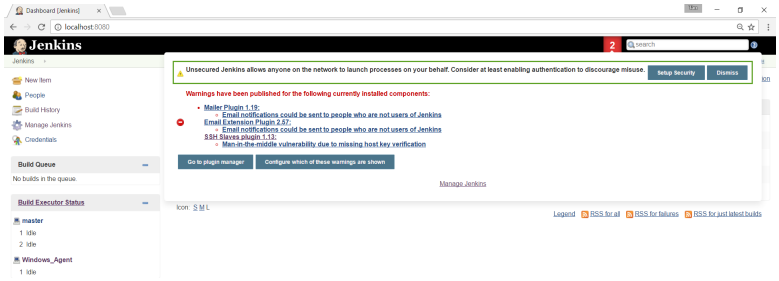
Restart Jenkins by service after change security configure



Now, You can do anything by just access to Jenkins without of need to login



And You can see a warning message from Jenkins show that security is disabled. And suggest You go for configure security

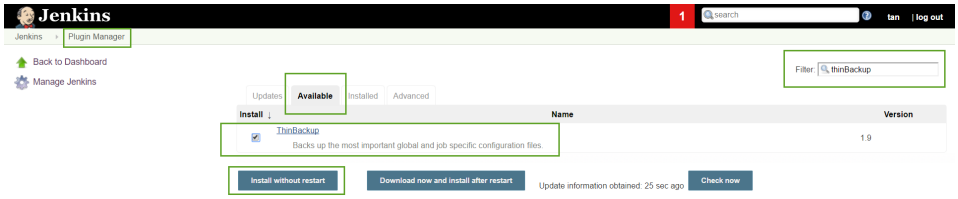


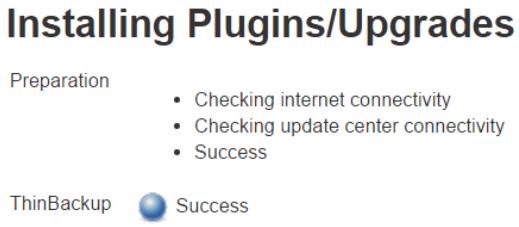
https://www.youtube.com/watch?v=rdypdeG_idE

Jenkins Backup and Restore

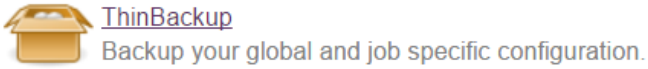
Install and Configure ThinBackup Plugin

From **Plugin Manager,** select **Available** tab, search for **thinBackup** plugin, select check box, then click to **Install without restart**

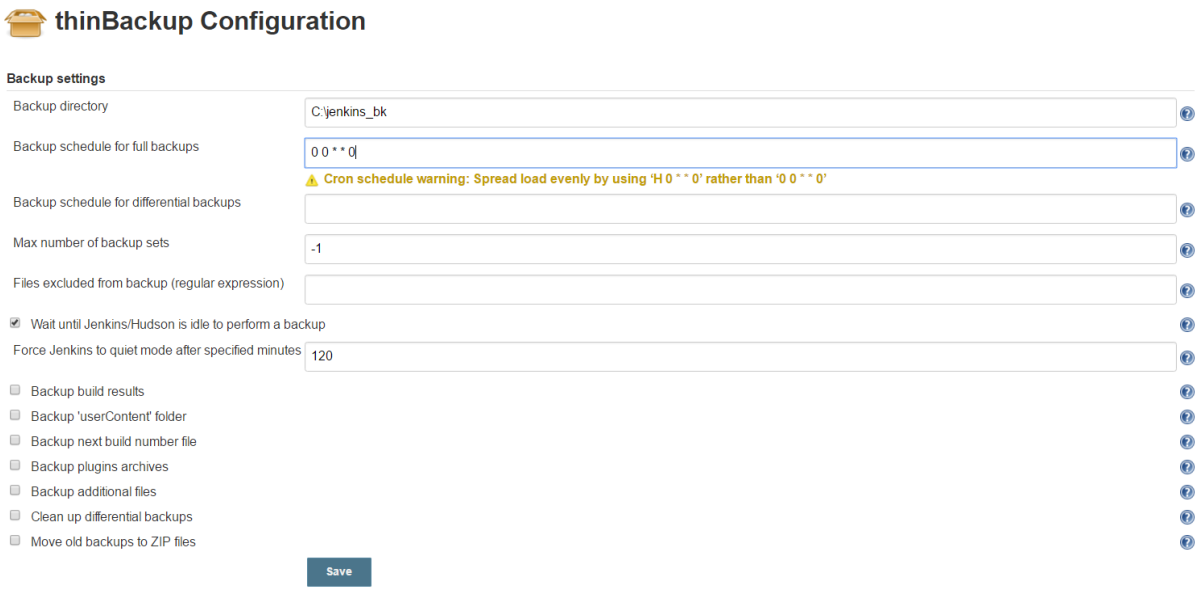




From **Manage Jenkins,** click to **ThinBackup,** then click to **Settings**

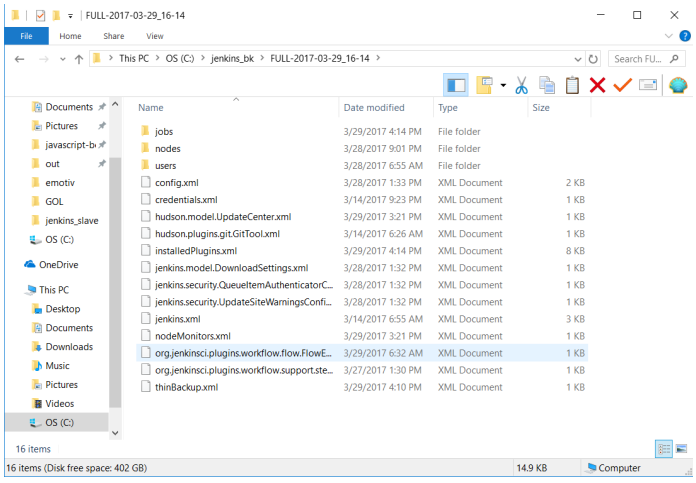


Normally we not need to backup build result, so uncheck **Backup build results.** Specify **0 0 * * 0** mean do full backup every 7 days (This is for demo, you can specify timeline that suite with your jenkins system). Finally, click to **Save** button



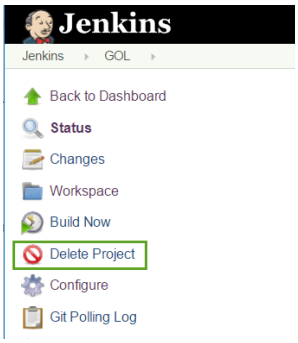
Manually Backup

Now click to **Backup Now** you will see full backup folder is created inside C:\jenkins_bk

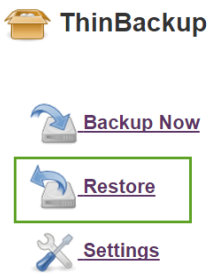


Restore from Backup

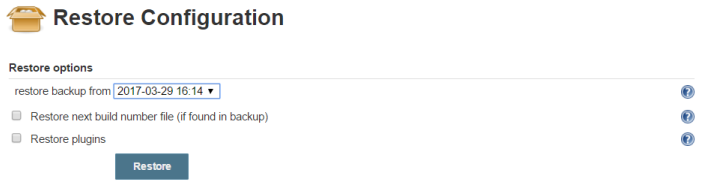
Try restore function by delete a job and then restore it. From home page click to GOL job, then click to **Delete Project**



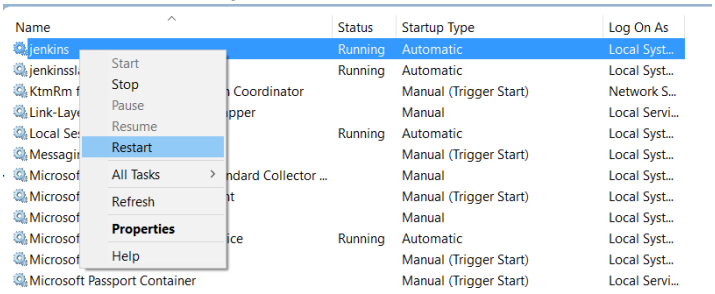
From **ThinBackup** click to **Restore**



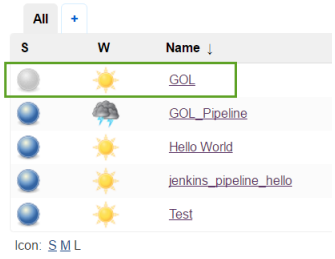
Select version want to restore and click to **Restore** button



Restart Jenkins by window service



Now back to Jenkins home page, You will see GOL job restored successfully.



Jenkins Interview Questions

Jenkins Interview Question

What is Jenkins?

Jenkins is an open source tool with plugin built for continuous integration purpose. The principle functionality of Jenkins is to keep a track of version control system and to initiate and monitor a build system if changes occur. It monitors the whole process and provides reports and notifications to alert.

What is the requirement for using Jenkins?

To use Jenkins you require

- A source code repository which is accessible, for instance, a Git repository
- A working build script, e.g., a Maven script, checked into the repository

What are the advantages of Jenkins?

Advantage of Jenkins include

- At integration stage, build failures are cached
- For each code commit changes an automatic build report notification generates
- To notify developers about build report success or failure, it is integrated with LDAP mail server
- Achieves continuous integration agile development and test driven development
- With simple steps, maven release project is automated
- Easy tracking of bugs at early stage in development environment than production

How you can move or copy Jenkins from one server to another?

- Slide a job from one installation of Jenkins to another by copying the related job directory
- Make a copy of an already existing job by making clone of a job directory by a different name
- Renaming an existing job by renaming a directory.

What are the commands you can use to start Jenkins manually?

To start Jenkins manually, you can use either of the following

- (Jenkins_url)/restart: Forces a restart without waiting for builds to complete
- (Jenkin_url)/safeRestart: Allows all running builds to complete

How can create a backup and copy files in Jenkins?

Jenkins saves all the setting, build artifacts and logs in its home directory, to create a backup of your Jenkins setup, just copy this directory. You can also copy a job directory to clone or replicate a job or rename the directory.

How you can set up Jenkins job?

To create a project that is handled via jobs in Jenkins. Select New item from the menu, once this done enter a name for the job and select free-style job. Then click OK to create new job in Jenkins. The next page enables you to configure your job.

What are the two components Jenkins is mainly integrated with?

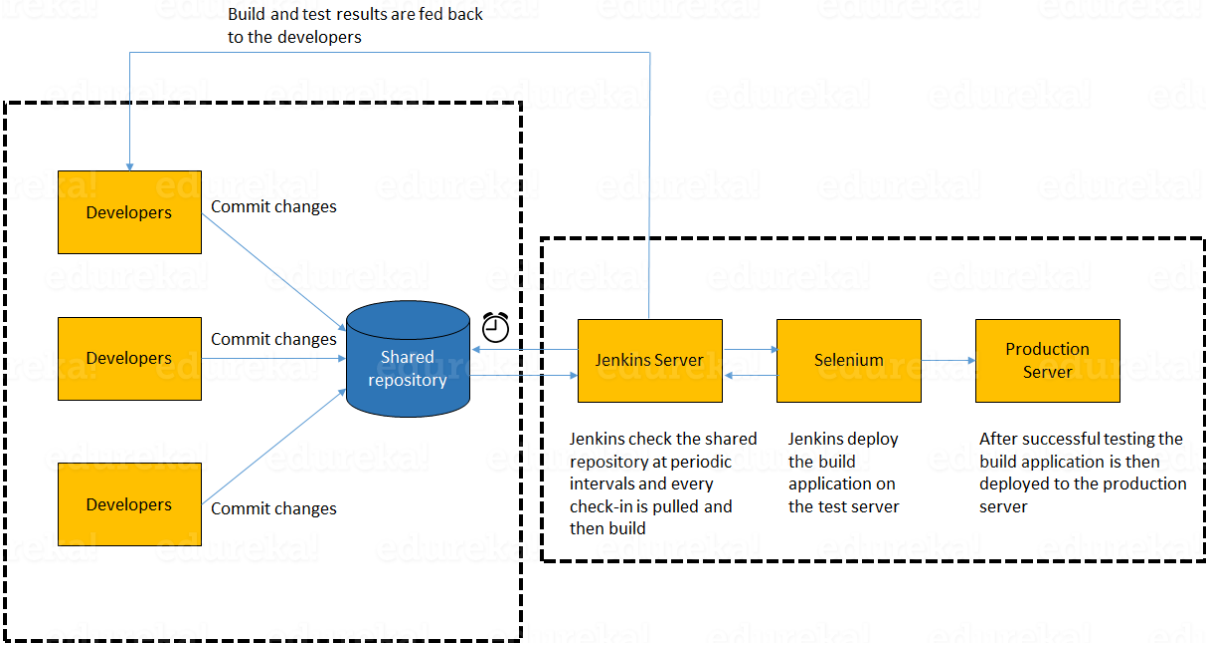
Jenkin is mainly integrated with two components

- Version Control system like GIT, SVN
- And build tools like Apache Maven.

CI/CD Interview Question

What is continuous integration?

It is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.



In the diagram shown above:

1. Developers check out code into their private workspaces.
2. When they are done with it they commit the changes to the shared repository (Version Control Repository).
3. The CI server monitors the repository and checks out changes when they occur.
4. The CI server then pulls these changes and builds the system and also runs unit and integration tests.
5. The CI server will now inform the team of the successful build.
6. If the build or tests fails, the CI server will alert the team.
7. The team will try to fix the issue at the earliest opportunity.
8. This process keeps on repeating.

Why is Continuous Integration important?

Two important reasons:

- Defects found early cost less to fix : When a defect is found immediately after a developer codes it, it takes 10x times less time to fix it compared to finding the defect a month later.
- Reduced Time to Market : Software is always tested. So, it is always ready to move to further environments.

What are the success factors for Continuous Integration?

Here you have to mention the requirements for Continuous Integration. You could include the following points in your answer:

- Maintain a code repository
- Automate the build

- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

Special Bonus

List of BEST DEVELOPMENT COURSE on Udemy with cheapest coupon from author!
http://devcourses.co

- Web development bootcamp
- Game development with C# and Unity
- Game, app development for iOS
- Game, app development for Android
- Data science with Python
- SQL and database