Jenkins

Contents:

1. Introduction 3
1.1 What is the Jenkins 3
1.2 Launching the Test Drive 3
2. Scenario: 3
3. Conclusion 24

1. Introduction

1.1 What is the Jenkins

Jenkins is an open-source continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.

Continuous Integration:

Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently. Every commit made in the repository is then built. This allows the teams to detect the problems early. Apart from this, depending on the Continuous Integration tool, there are several other functions like deploying the build application on the test server, providing the concerned teams with the build and test results etc.

1.2 Launching the Test Drive

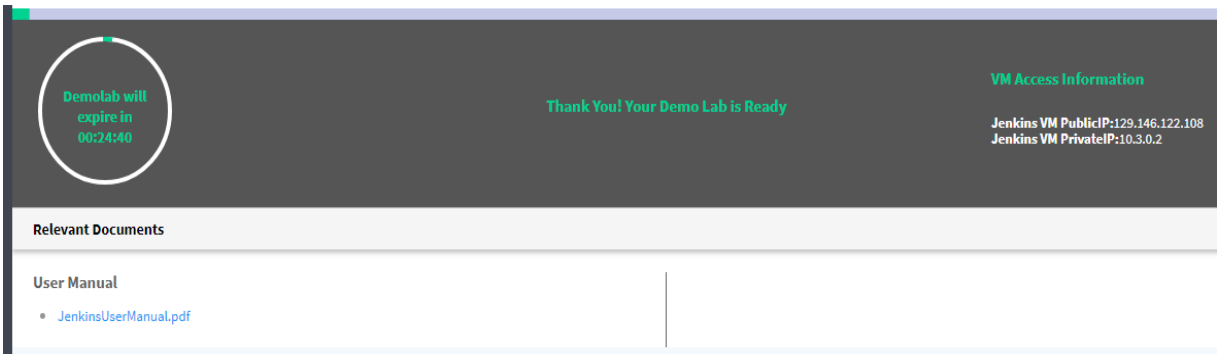
Once the test drive is ready, you should receive the credentials and URL(s) needed to access the test drive in an email sent to the address you used to sign up.

2. Scenario:

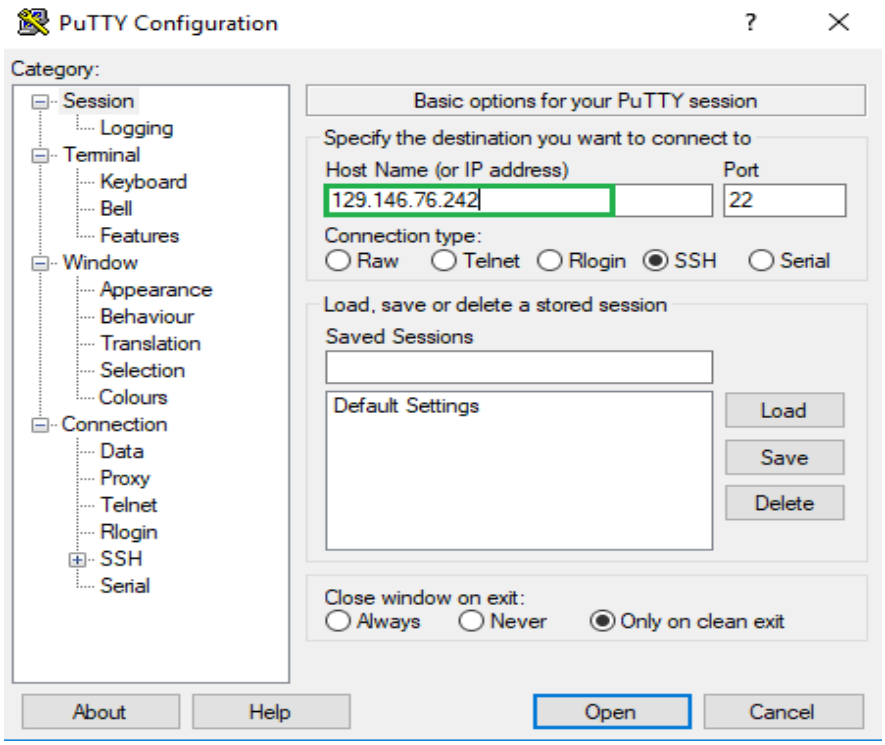
Objectives:

- Launch the test drive.
- Setup and configure the GitHub repository.
- Configure Jenkins job.
- See Jenkins in action.

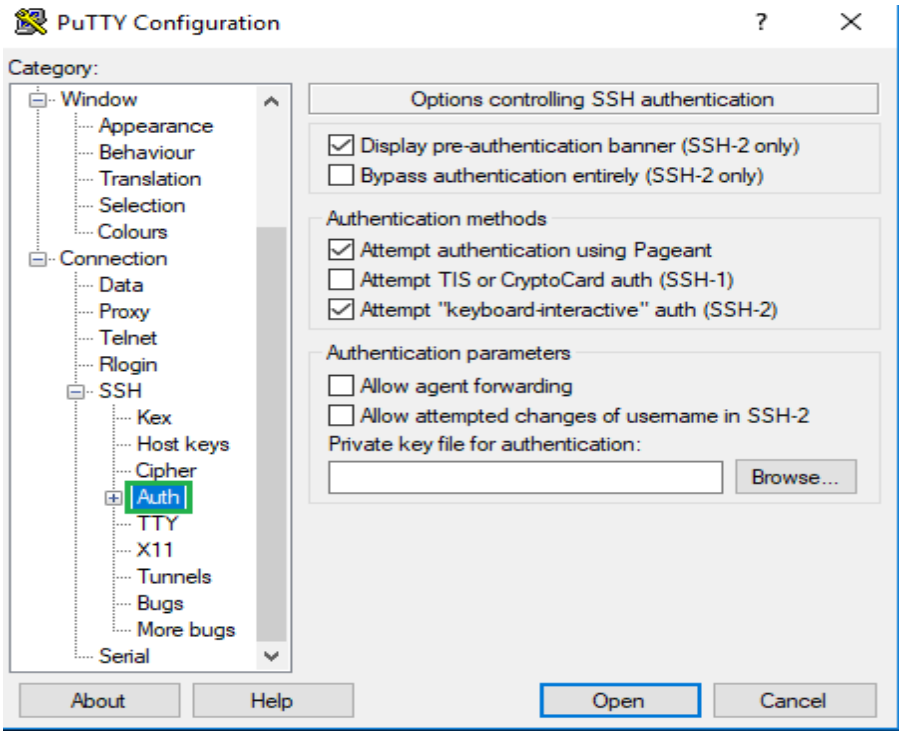
1. Log into your SSH session using client SSH URL provided in test drive launch page.

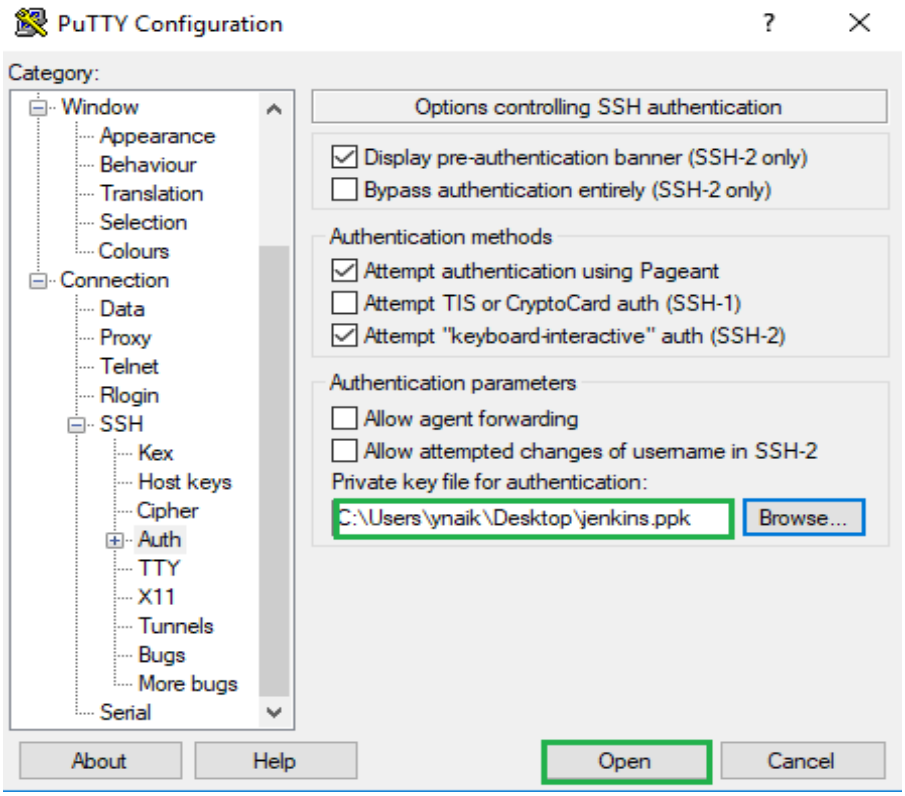


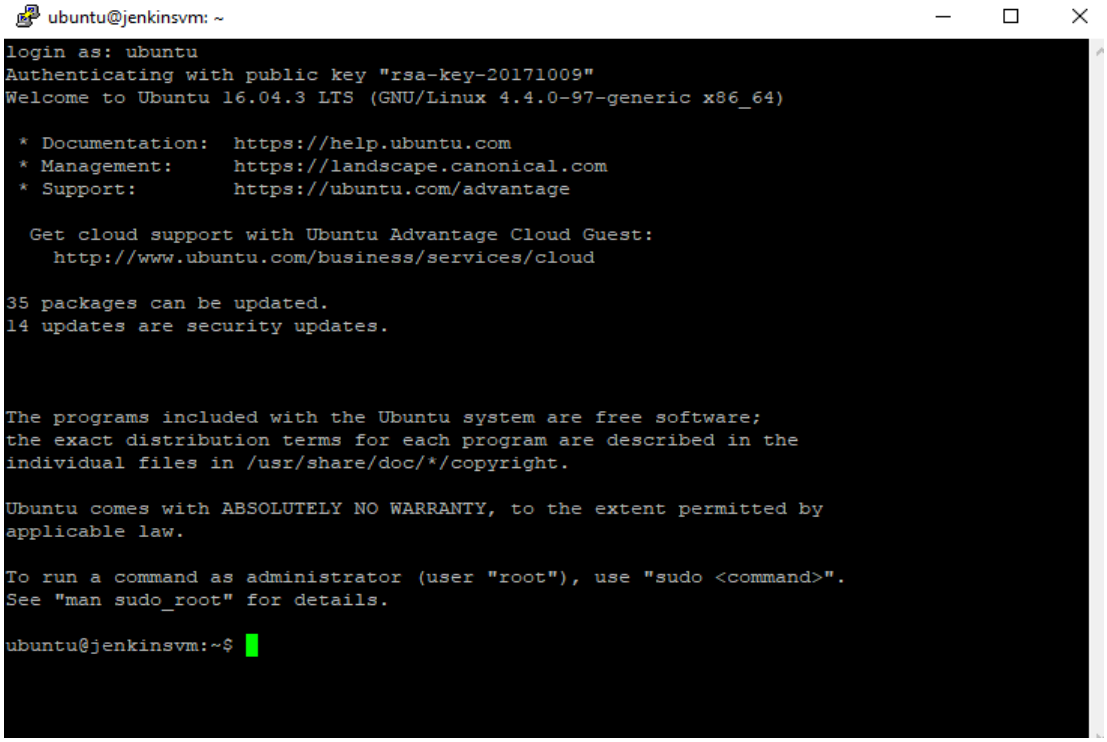
2. Open putty and SSH into the VM using the IP address



Click on browse and select the private key you have stored in your local system







3. Once you login into the VM you need to run a script to configure the job for the Test Drive

Run the below command:

sudo wget https://raw.githubusercontent.com/yougandar/test/master/script.sh

This command basically downloads the script from the given location

```
ubuntu@jenkinsvm: ~
ubuntu@jenkinsvm:~$ sudo wget https://raw.githubusercontent.com/yougandar/test/master/script.sh
--2017-11-02 10:40:48--  https://raw.githubusercontent.com/yougandar/test/master/script.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.200.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.200.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1607 (1.6K) [text/plain]
Saving to: 'script.sh'

script.sh                   100%[===========================================>]   1.57K  --.-KB/s    in 0s

2017-11-02 10:40:48 (446 MB/s) - 'script.sh' saved [1607/1607]

ubuntu@jenkinsvm:~$
```

To run the script run the following command

sudo sh script.sh

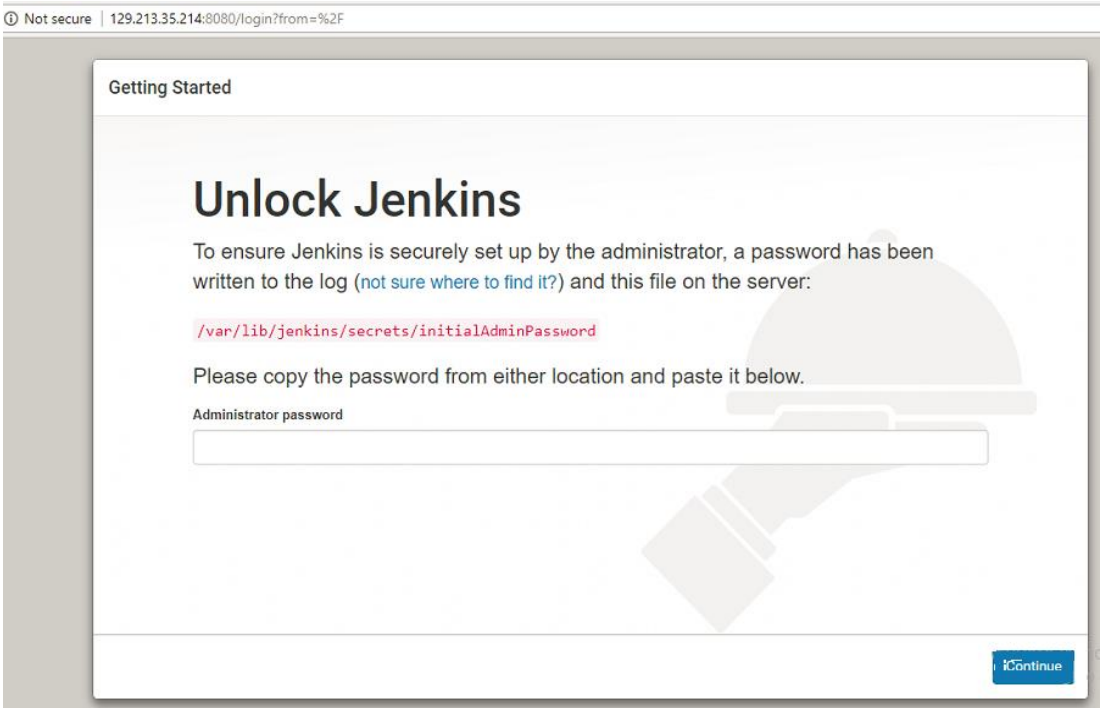
```
ubuntu@jenkinsvm:~$ sudo sh script.sh
Ign:1 http://pkg.jenkins-ci.org/debian-stable binary/ InRelease
Hit:2 http://pkg.jenkins-ci.org/debian-stable binary/ Release
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:5 http://iad-ad-2.clouds.archive.ubuntu.com/ubuntu xenial InRelease
Hit:6 http://iad-ad-2.clouds.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:7 http://iad-ad-2.clouds.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-headers-virtual linux-image-virtual
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  html-xml-utils
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 231 kB of archives.
After this operation, 2,691 kB of additional disk space will be used.
Get:1 http://iad-ad-2.clouds.archive.ubuntu.com/ubuntu xenial/universe amd64 html-xml-utils amd64 6.9-1 [231 kB
]
Fetched 231 kB in 0s (3,175 kB/s)
Selecting previously unselected package html-xml-utils.
(Reading database ... 63619 files and directories currently installed.)
Preparing to unpack .../html-xml-utils_6.9-1_amd64.deb ...
Unpacking html-xml-utils (6.9-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up html-xml-utils (6.9-1) ...
```

```
Saving to: '/usr/share/jenkins/job-configfile.xml'

job-configfile.xml          100%[===========================================>]   1.66K  --.-KB/s    in 0s

2017-11-02 10:41:39 (502 MB/s) - '/usr/share/jenkins/job-configfile.xml' saved [1698/1698]

---Configuring Jenkins---
--2017-11-02 10:41:59--  http://localhost:8080/jnlpJars/jenkins-cli.jar
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2972859 (2.8M) [application/java-archive]
Saving to: '/usr/share/jenkins/jenkins-cli.jar'

jenkins-cli.jar             100%[===========================================>]   2.83M  --.-KB/s    in 0.02s

2017-11-02 10:41:59 (125 MB/s) - '/usr/share/jenkins/jenkins-cli.jar' saved [2972859/2972859]

Authenticated as: admin
Authorities:
  authenticated
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    46  100    46    0     0    972      0 --:--:-- --:--:-- --:--:--   978
00e37fef3980f722957057ec0f639180
Jenkins-Crumb:261ca6fb27dc951ec1790044 1a27a7f7
ubuntu@jenkinsvm:~$ ls
script.sh
ubuntu@jenkinsvm:~$
```

4. Now open any browser and paste the Jenkins IP address that we get in the access information to login to the Jenkins dashboard

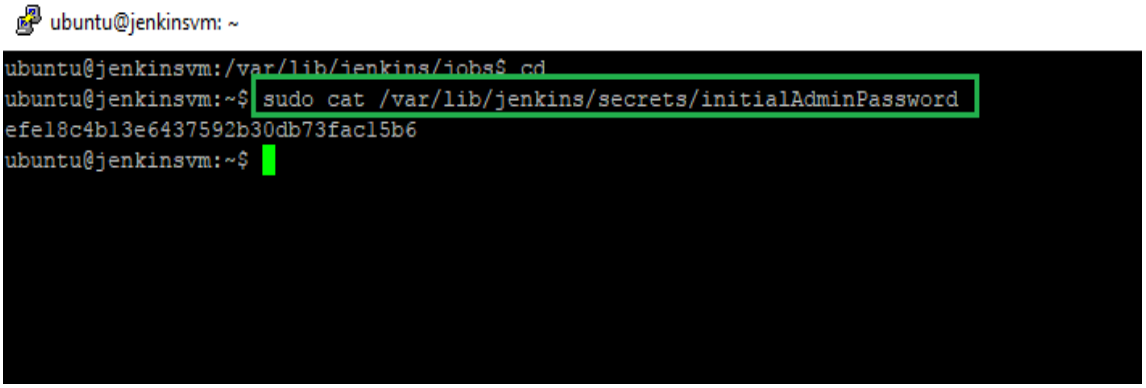
Ex: ***http://JenkinsVMIP:8080***

http://129.146.76.242:8080

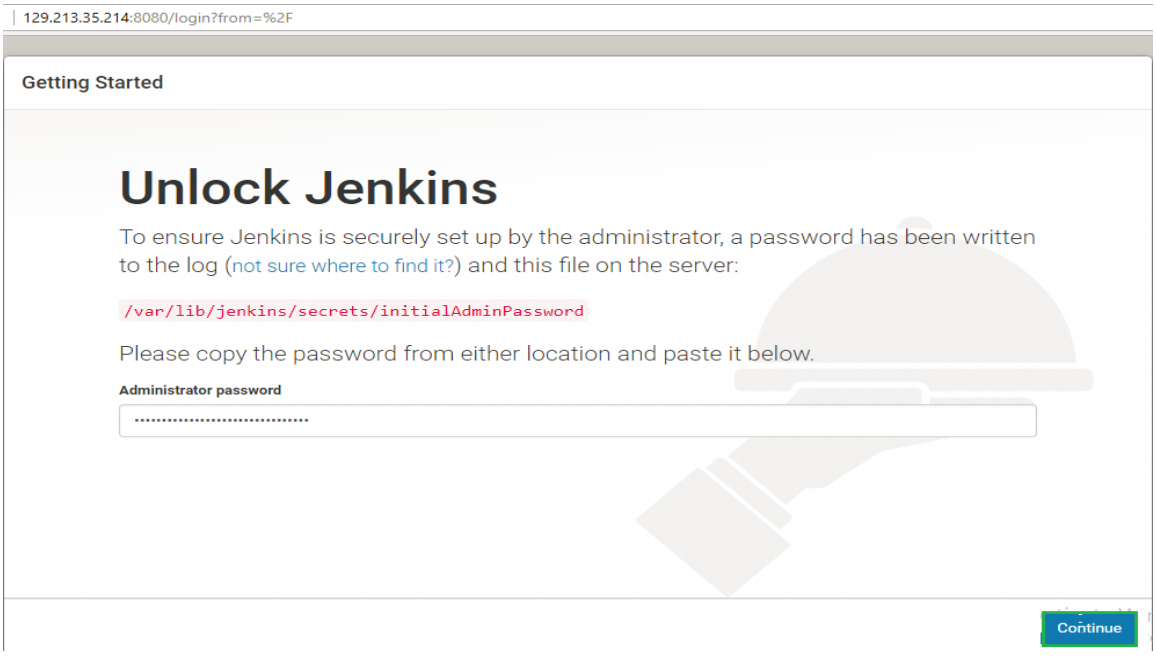


5. To unlock Jenkins you need to login to the Jenkins machine and use the following command to access the initialAdminPassword

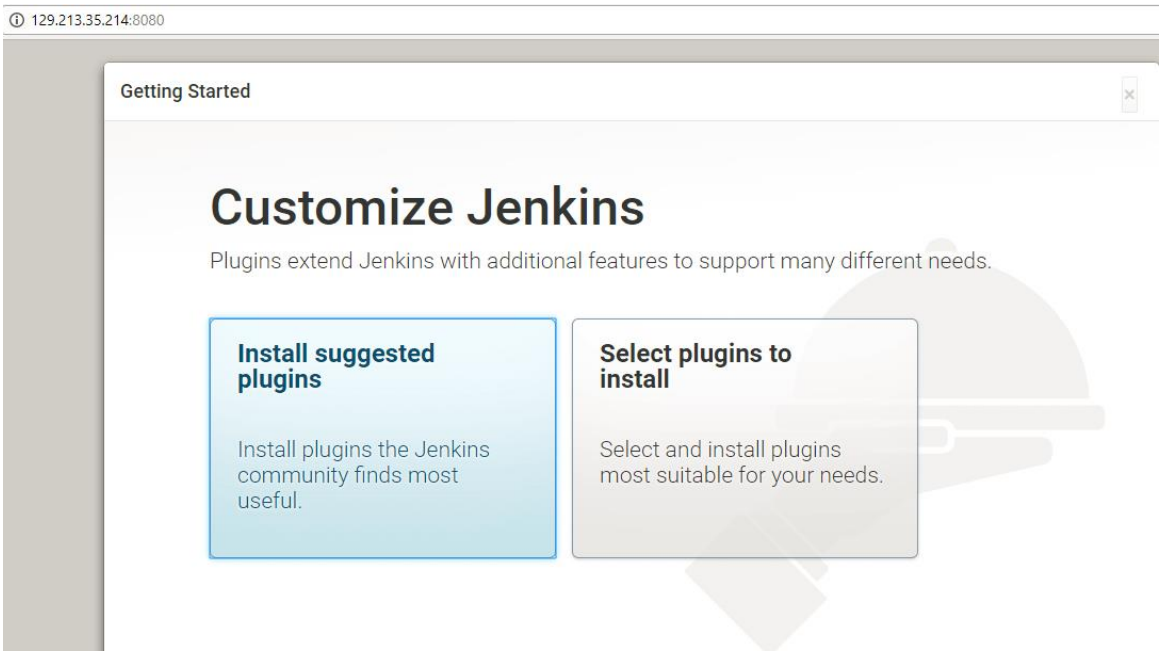
sudo cat/var/lib/Jenkins/secrets/initialAdminPassword

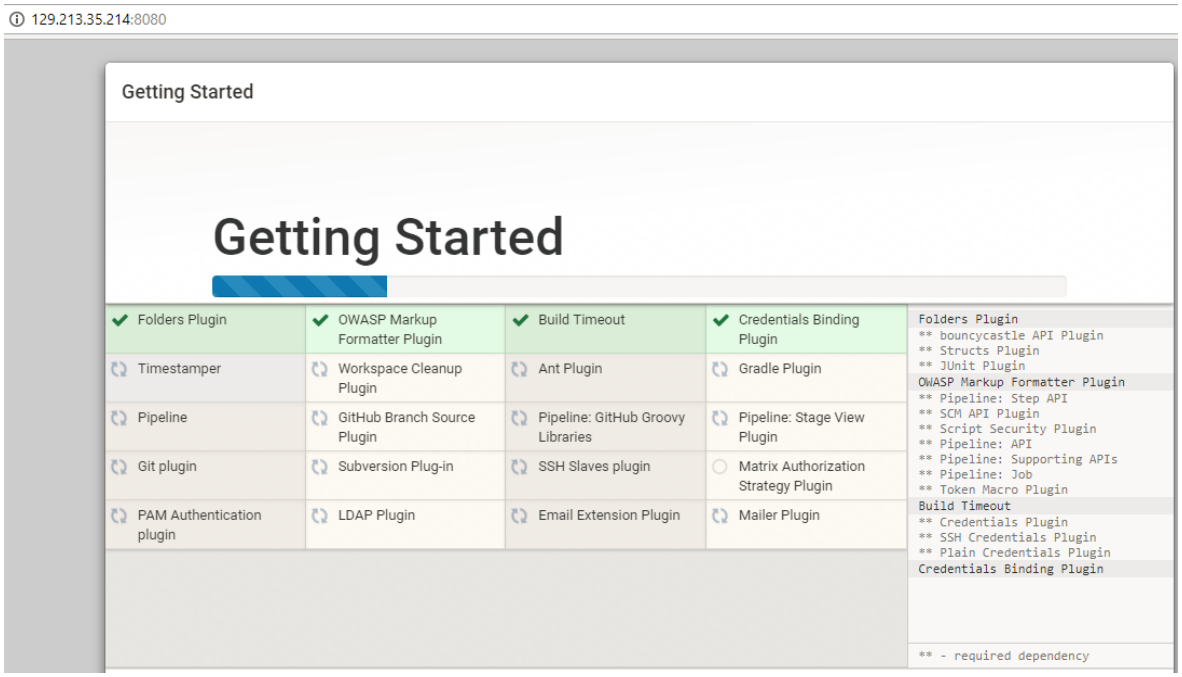


Paste the password you get as shown below



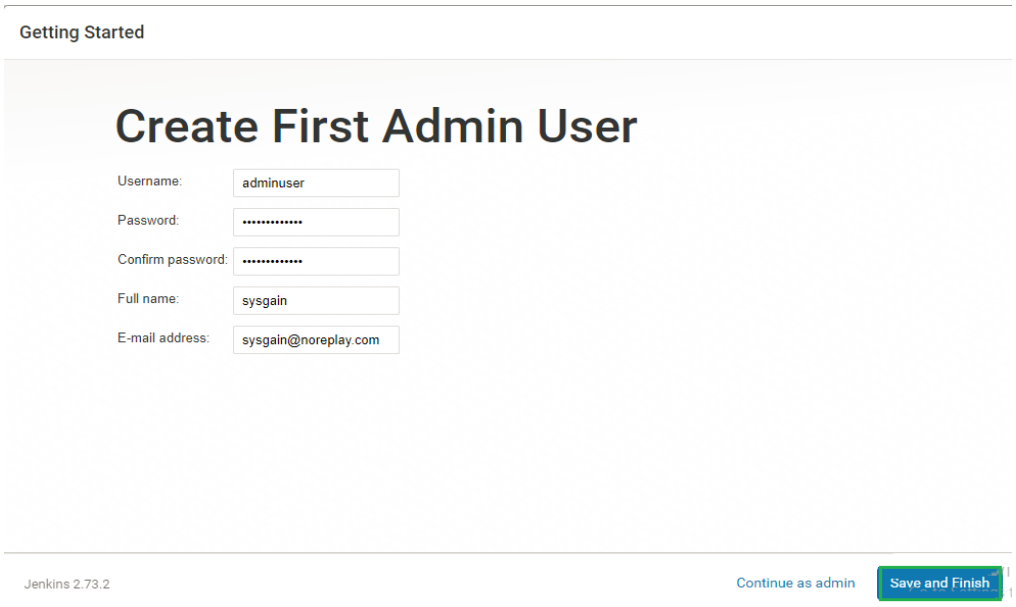
When prompted please click on install suggested plugins, so Jenkins automatically installs the necessary plugins

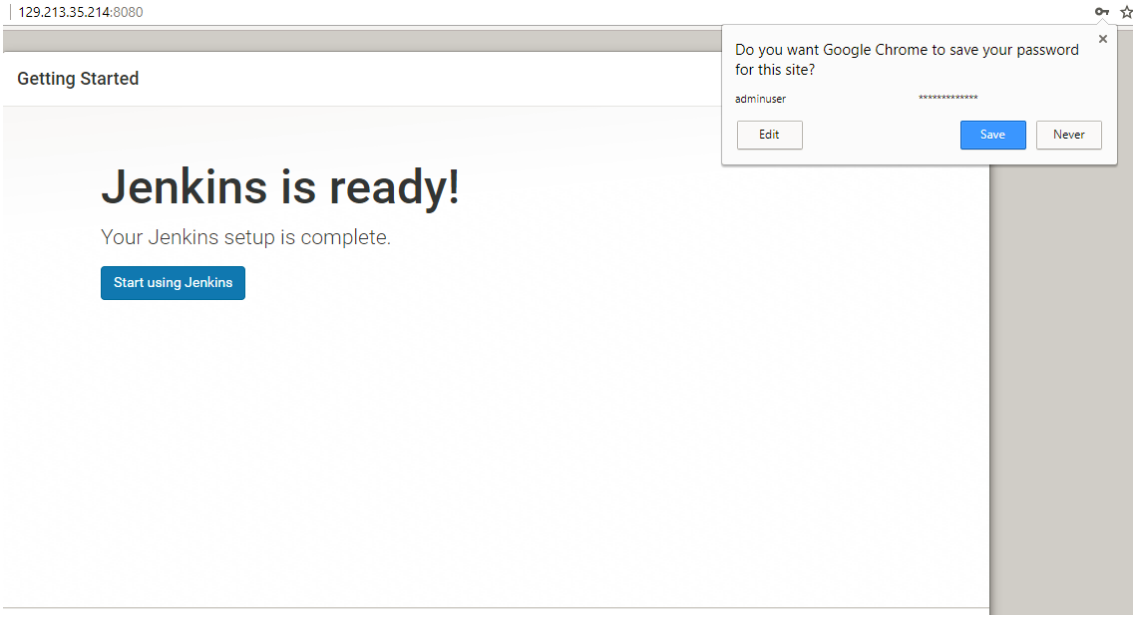




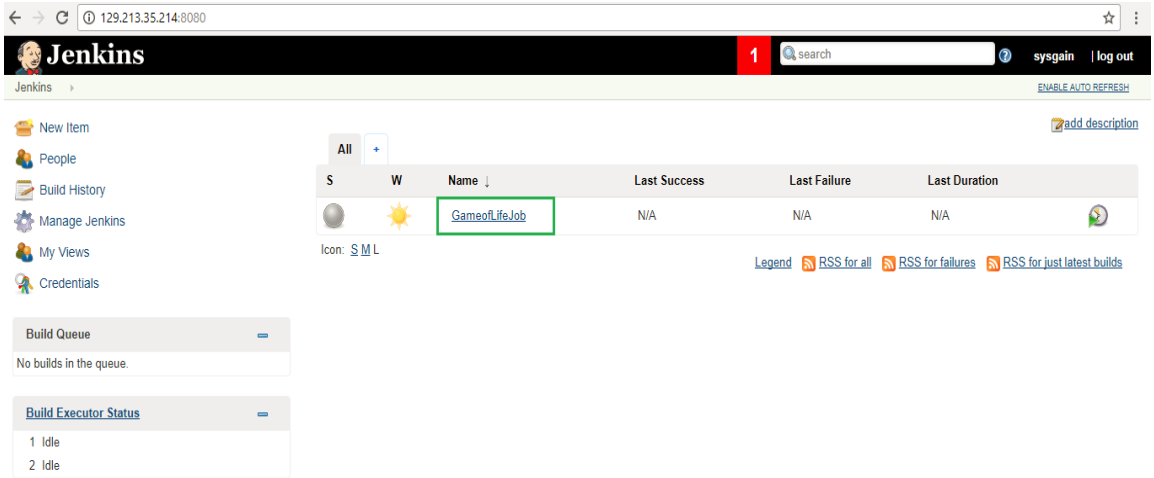
Please wait while the plugins get installed

6. When prompted create a new admin username and password which you can use to access the dashboard.

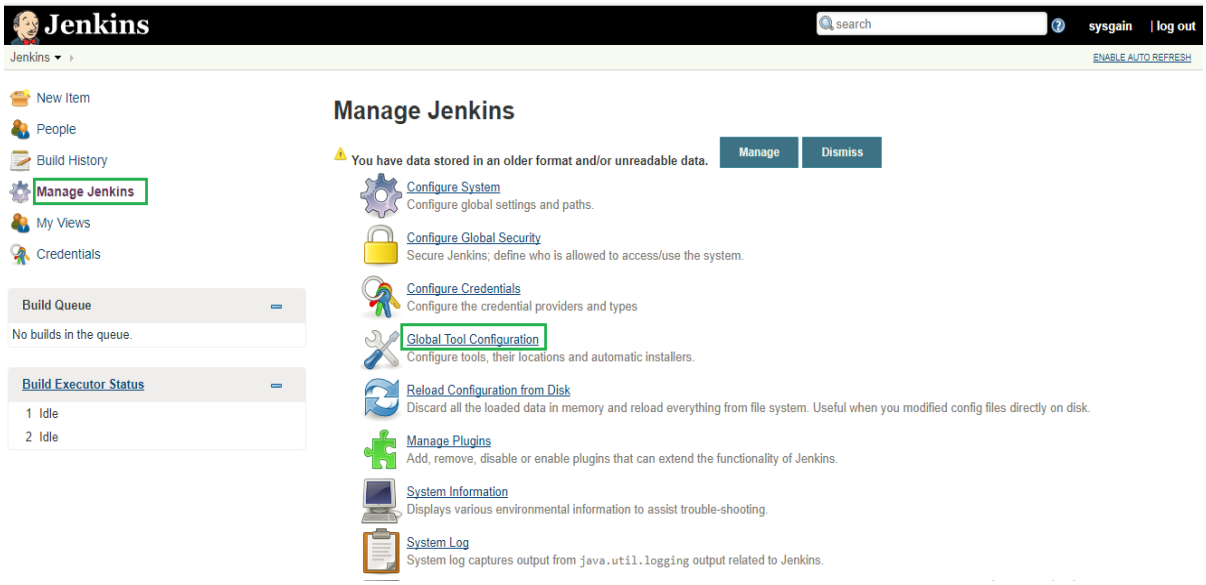


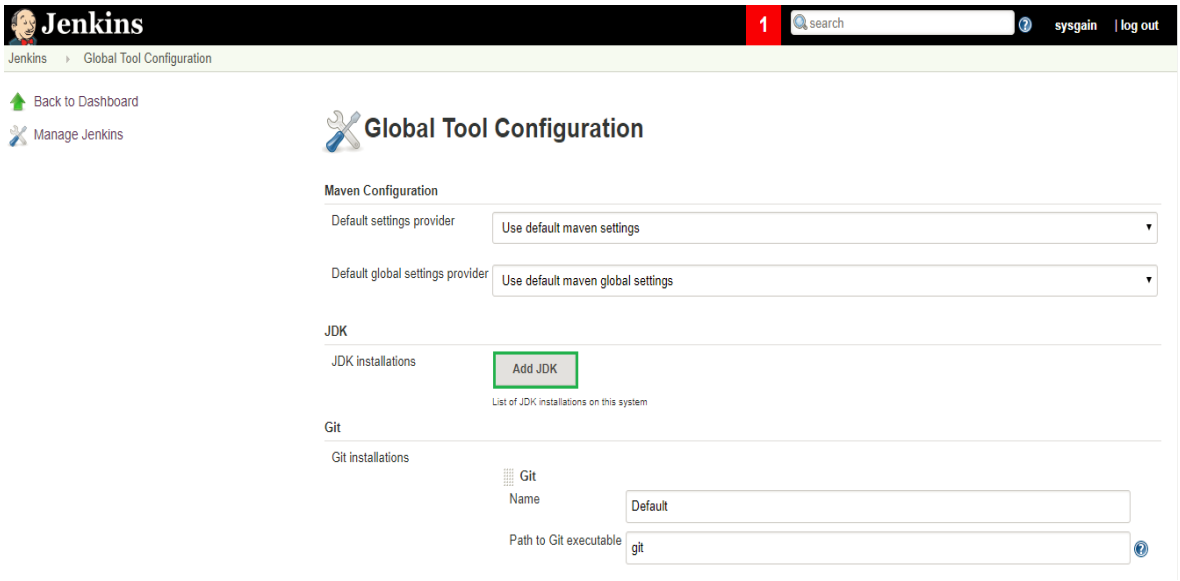


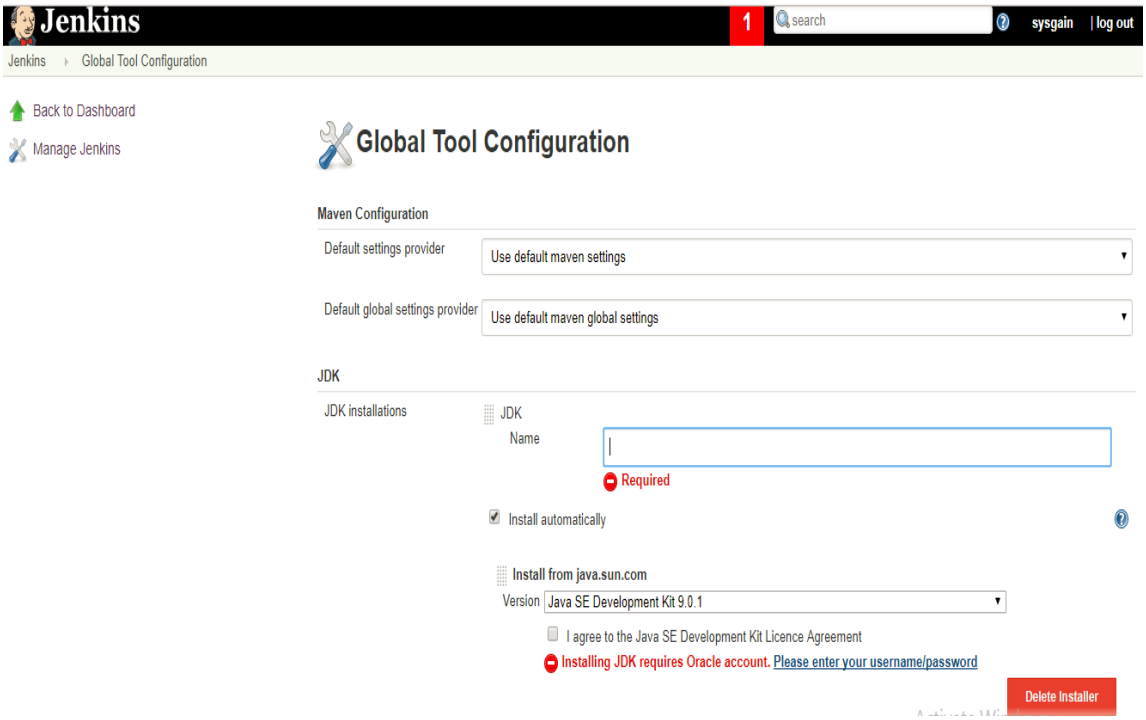
7. When you access the dashboard you can see a job configured: i.e. **GameofLifeJob**



8. Go to manage Jenkins click on global tool configuration, here you need to add the JDK path as shown in the below screenshot

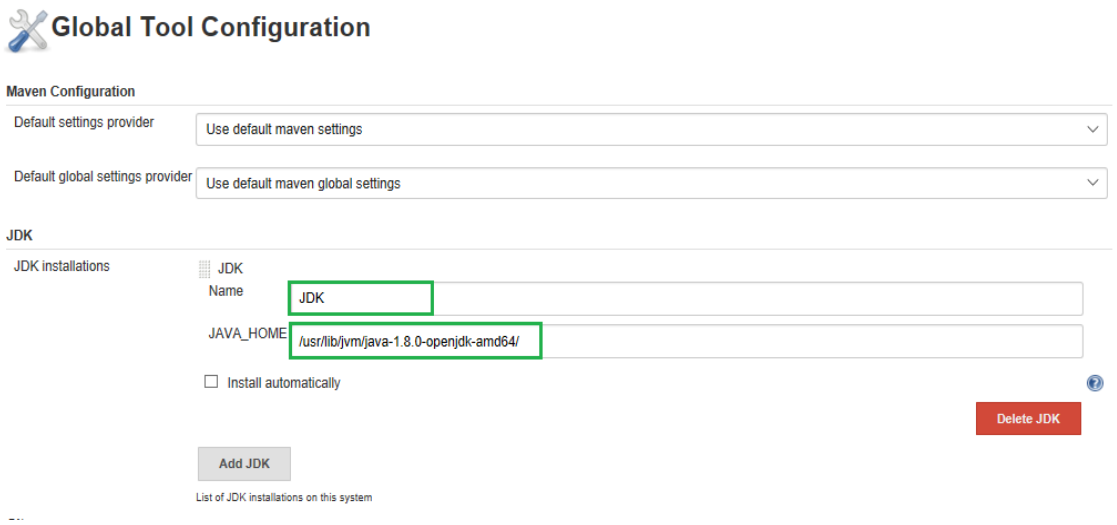






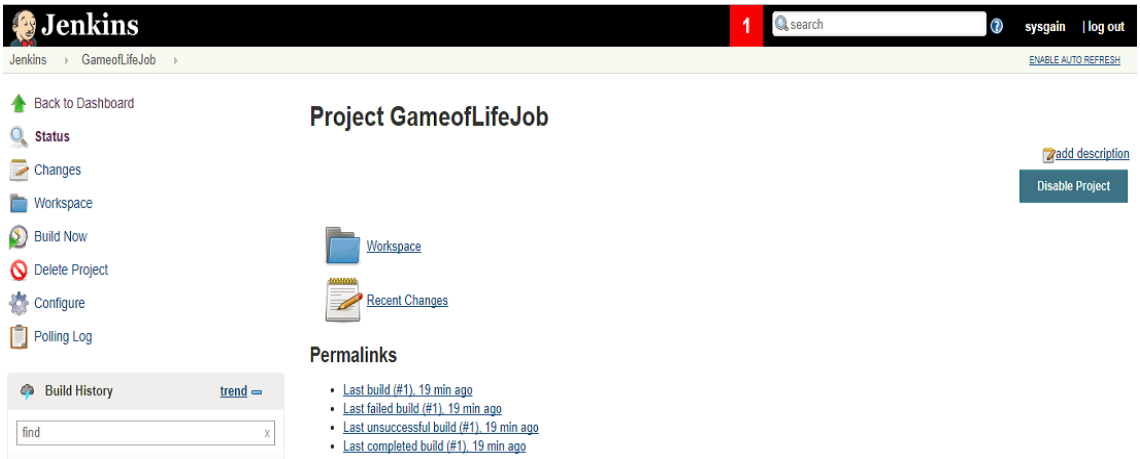
9. Please paste the path as **/usr/lib/jvm/java-1.8.0-openjdk-amd64.**Choose any name

```
ubuntu@jen-jenkinsvm:~$ cd /usr/lib/jvm
ubuntu@jen-jenkinsvm:/usr/lib/jvm$ ls
default-java  java-1.8.0-openjdk-amd64  java-8-openjdk-amd64
ubuntu@jen-jenkinsvm:/usr/lib/jvm$ cd java-1.8.0-openjdk-amd64
ubuntu@jen-jenkinsvm:/usr/lib/jvm/java-1.8.0-openjdk-amd64$ ls
ASSEMBLY_EXCEPTION  bin  docs  include  jre  lib  man  src.zip  THIRD_PARTY_README
ubuntu@jen-jenkinsvm:/usr/lib/jvm/java-1.8.0-openjdk-amd64$
```



10. Under maven give any name and select the checkbox to install automatically

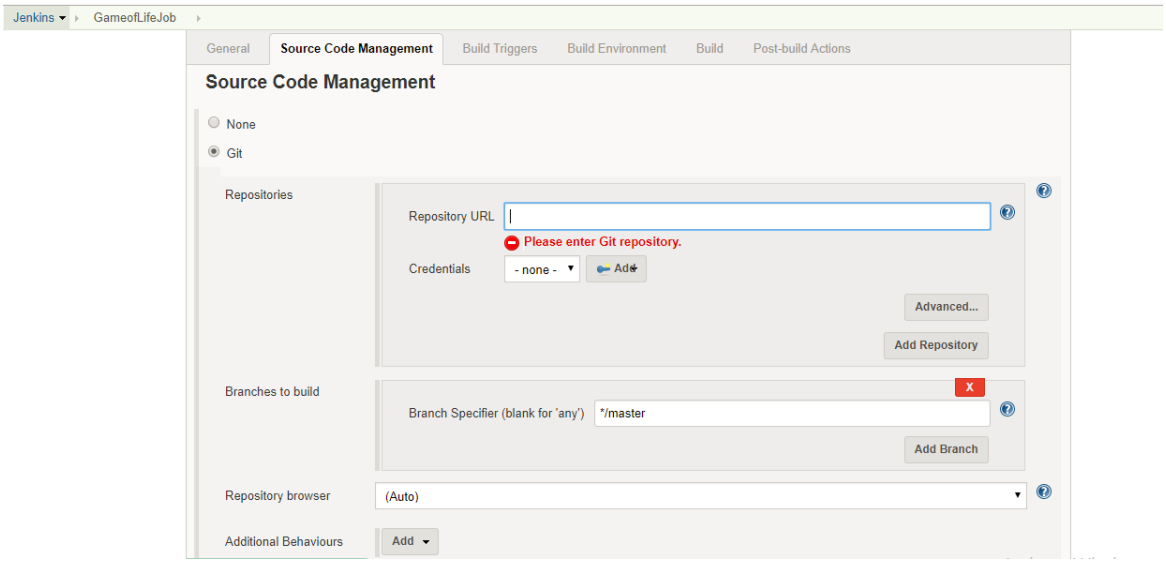
11. Now click on the job and click on configure.

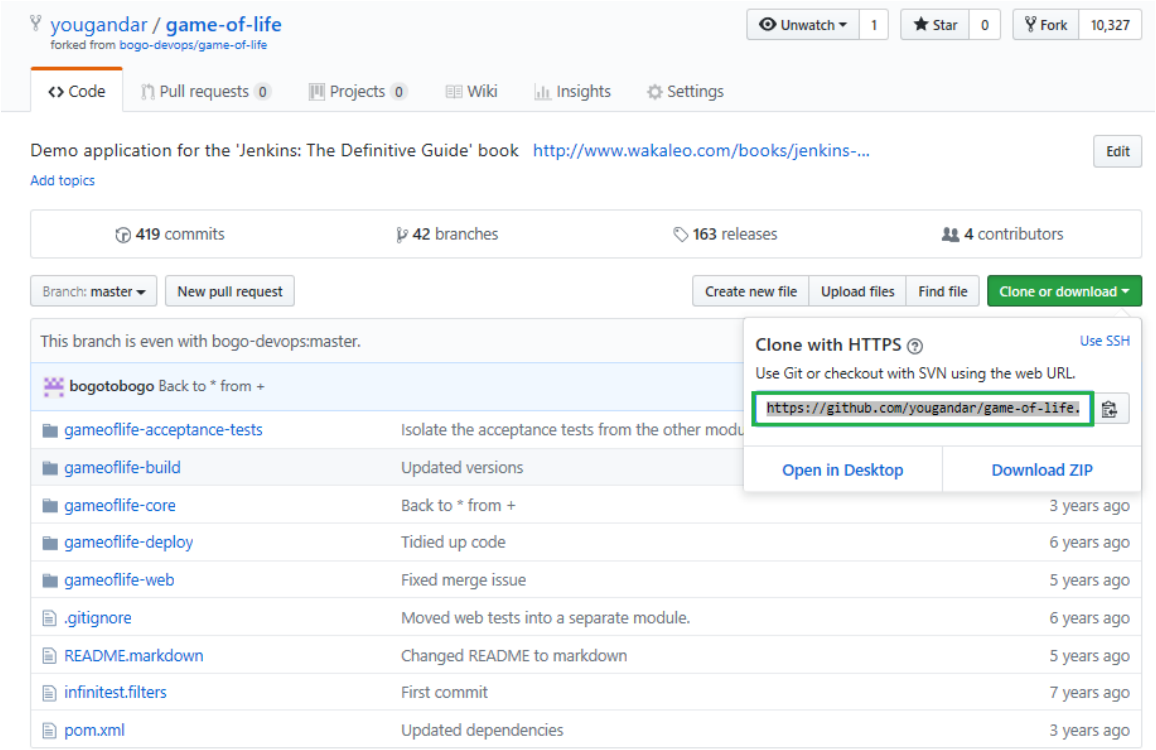


12. Under SourceCodeManagement give the GitHub repository URL (the game of life repository which you have forked earlier)

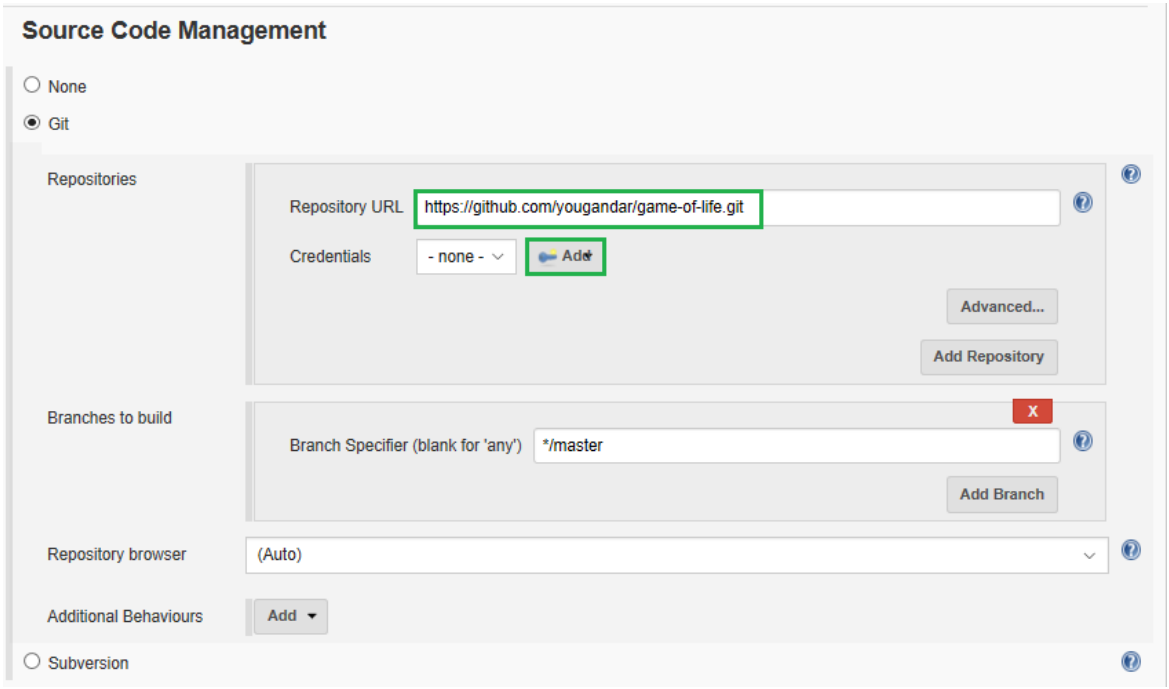
EX: ***https://github.com/yougandar/game-of-life.git***

Under branch specifier select */master

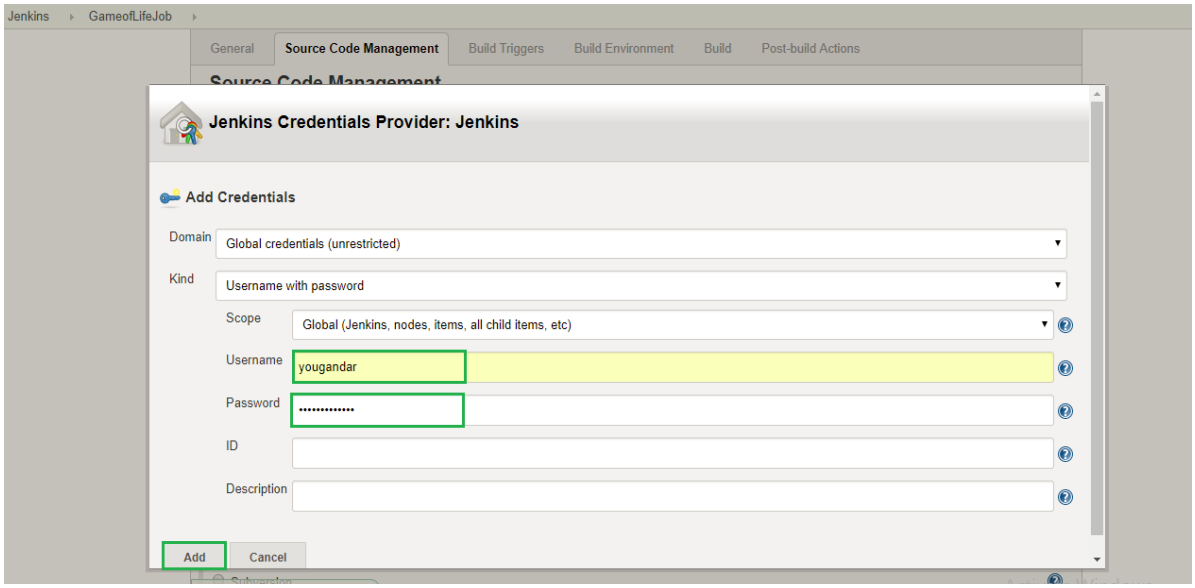


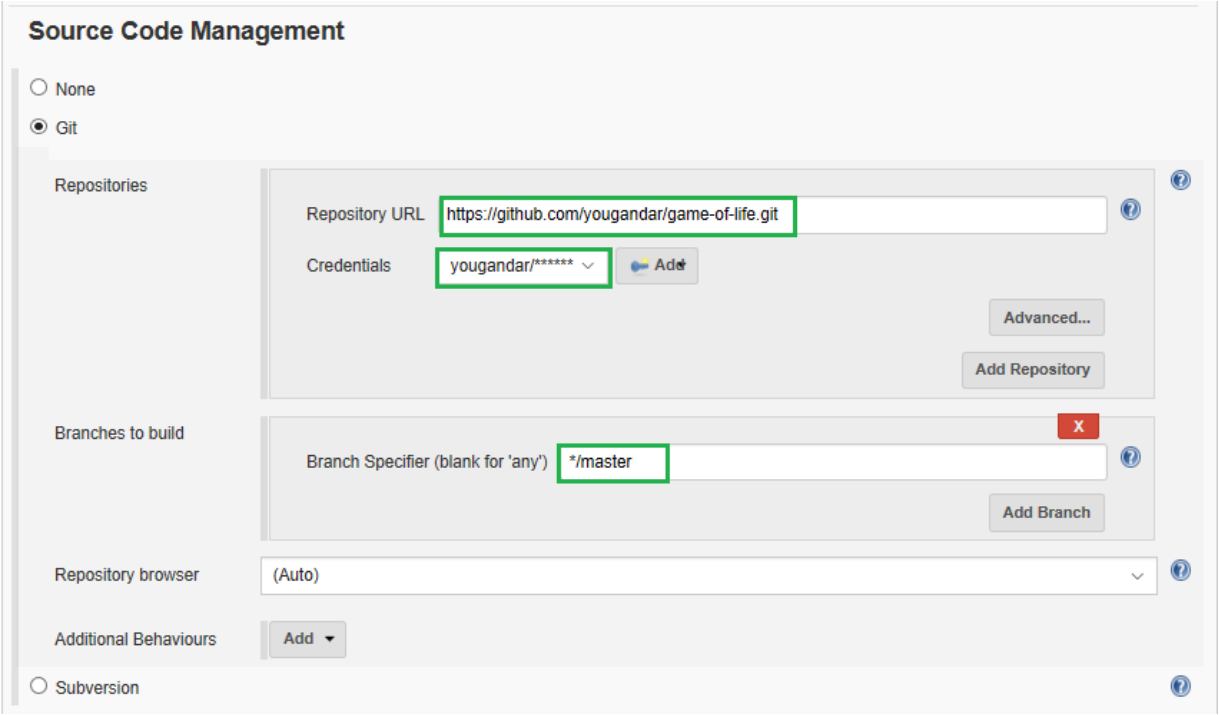


Click on Add -> Jenkins ->It will open the Jenkins Credentials provider



13. Provide the credentials of your git account so Jenkins can access your repository

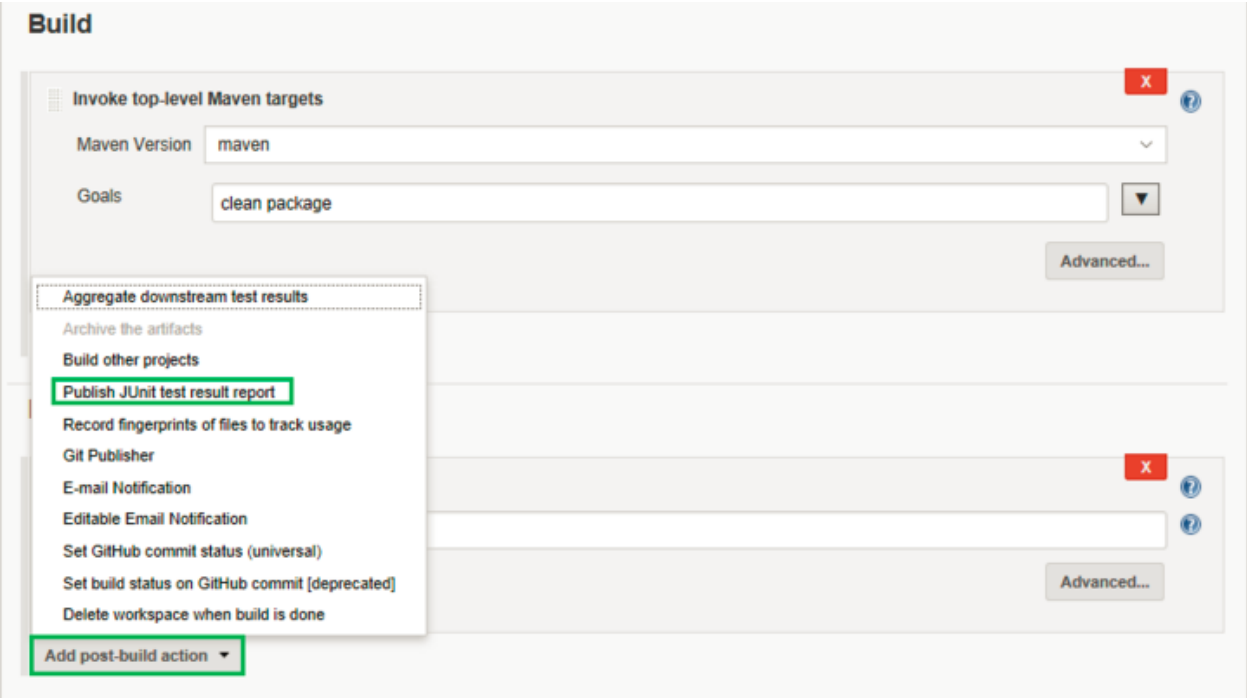




The standard for test reporting in Java is an XML format used by JUnit.

Jenkins understands this format, so if our build produces JUnit XML test results, Jenkins can generate nice graphical test reports and statistics on test results over time, and also let us view the details of any test failures.

Jenkins also keeps track of how long our tests take to run, both globally, and per test-this can come in handy if we need to track down performance issues.

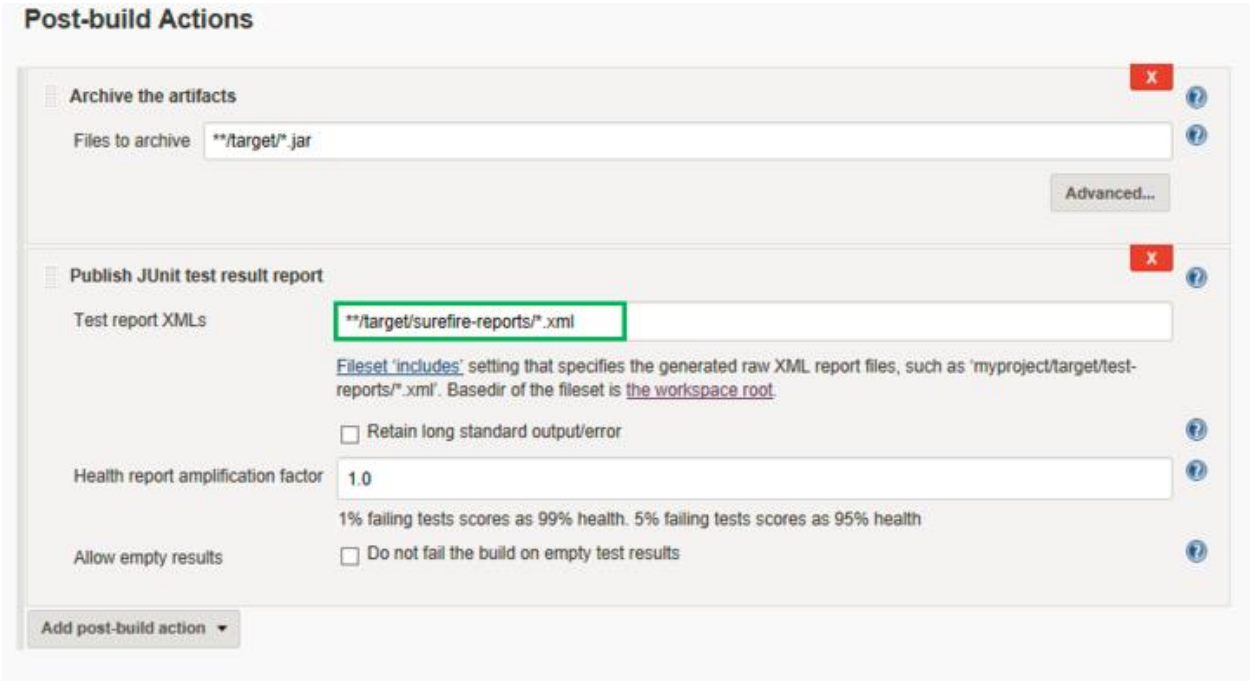


14) Go to the Post-build Actions section and check "Publish JUnit test result report" checkbox.

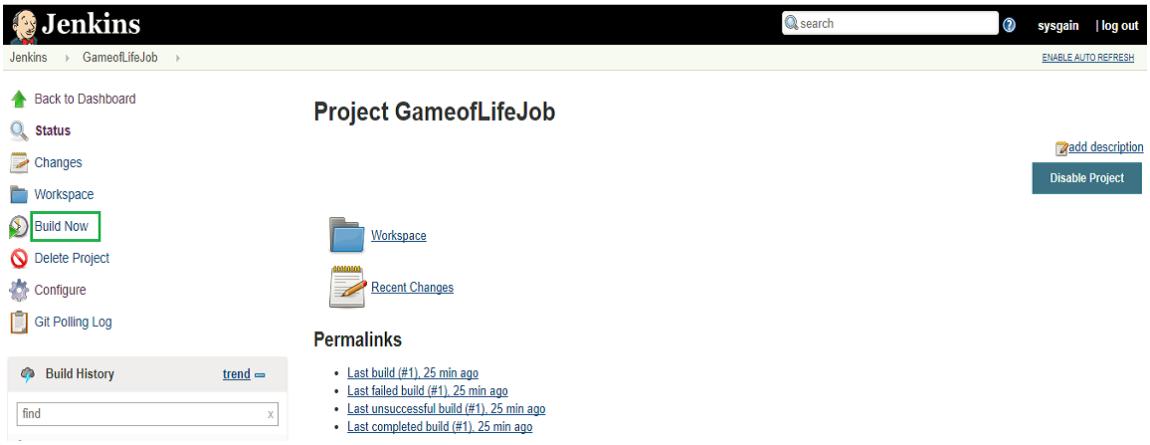
When Maven runs unit tests in a project, it automatically generates the XML test reports in a directory called surefire-reports in the target directory.

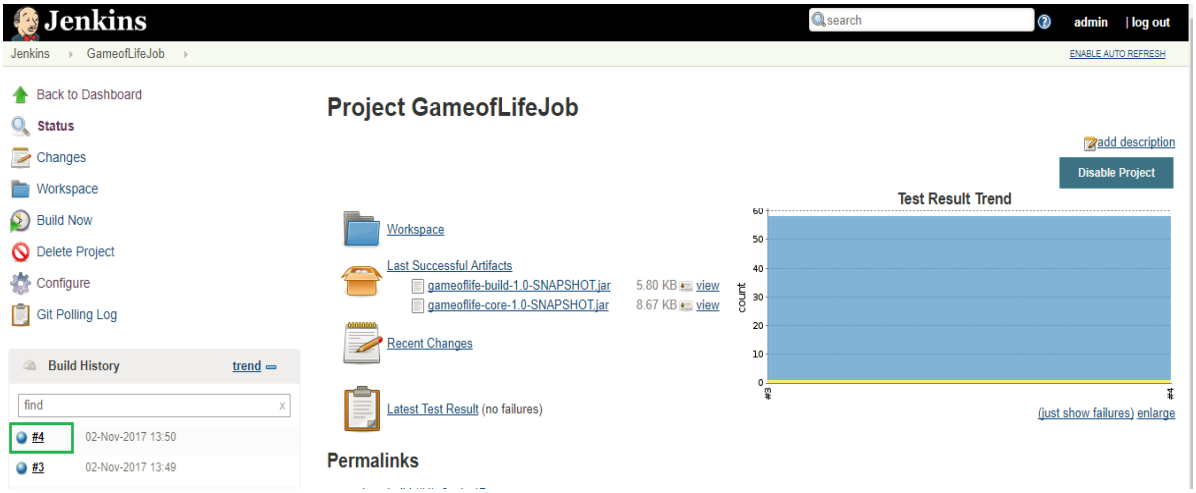
So, enter "**/target/surefire-reports/*.xml" in the "Test report XMLs" field. The two asterisks at the start of the path ("**") are a best practice to make the configuration a bit more robust:

they allow Jenkins to find the target directory no matter how we have configured Jenkins to check out the source code.

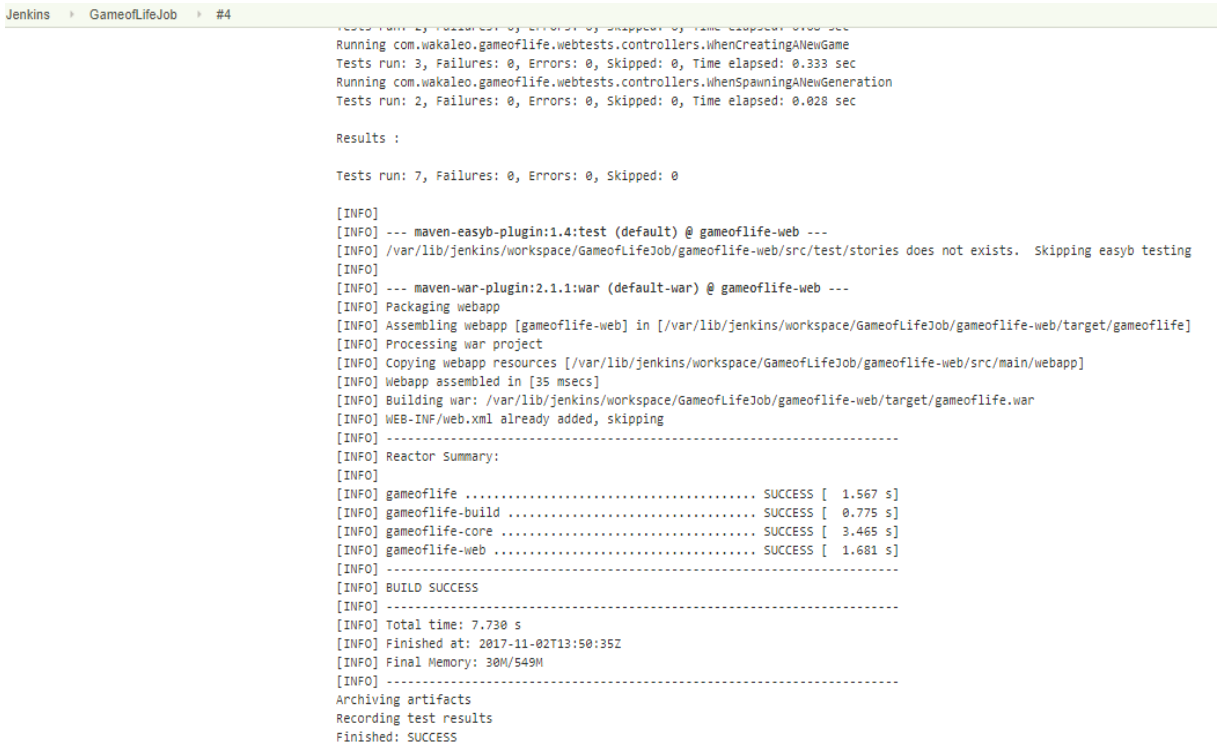


15. Now when you build the project Jenkins performs the tests and give the test results accordingly indicating success or failure.

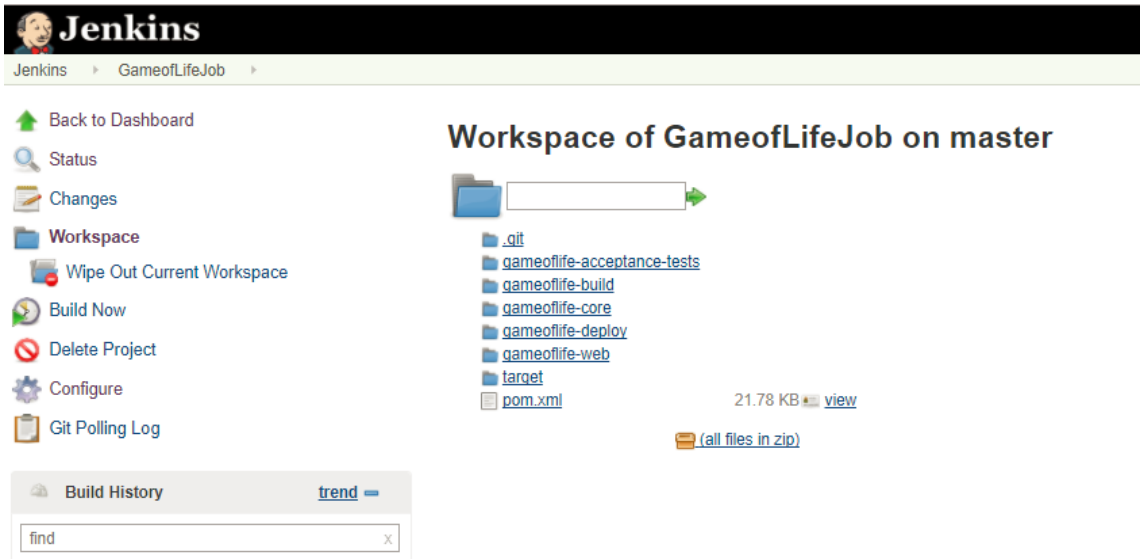




In the below screenshot we can see that the build was successful in the console output.

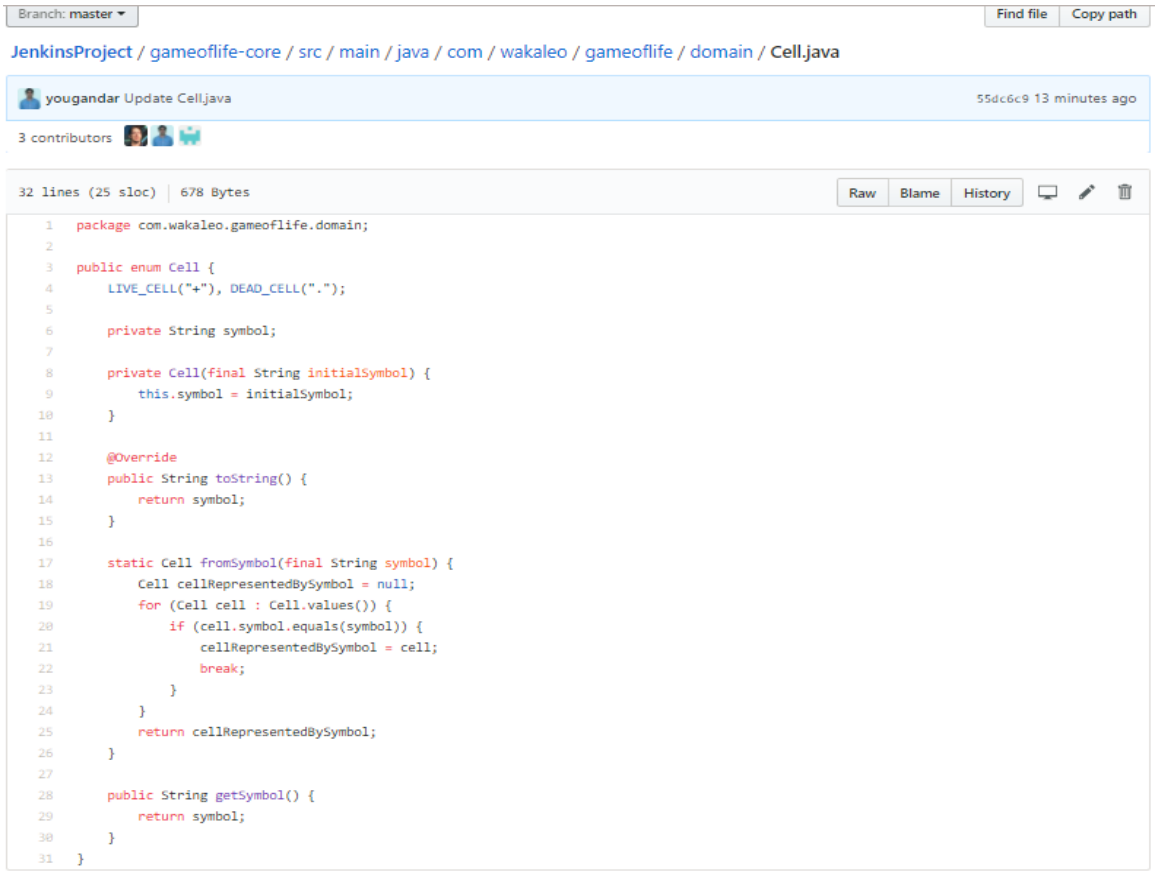


16. Click on the Workspace of project as show in the below screenshot.

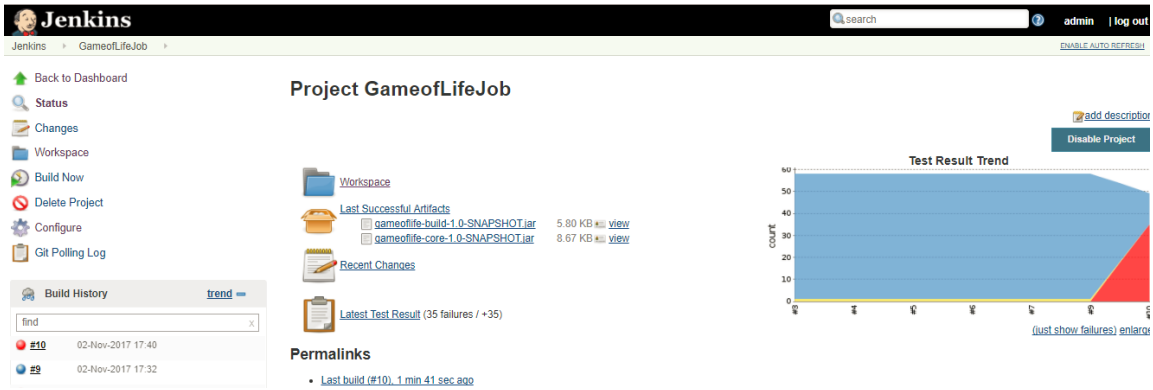


17. Now, what we can do is change to source code to demonstrate Jenkins failed test results.

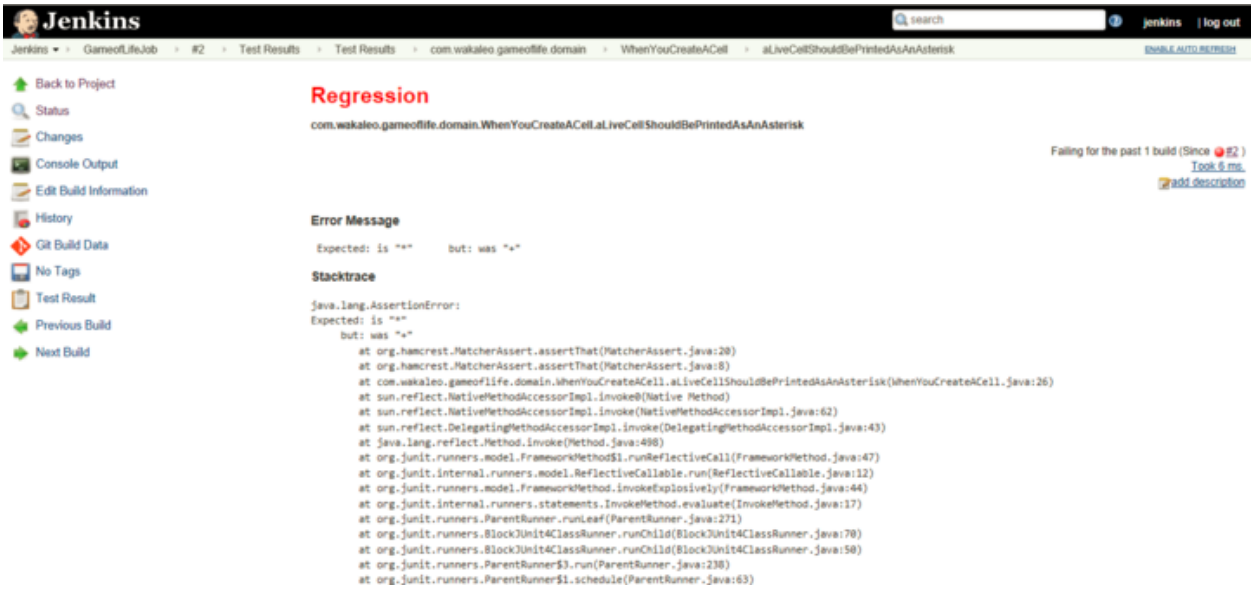
In the source code if we change the LIVE_CELL (*) to LIVE_CELL (+) and build the job again.



18. Here we can see that the build has failed and also review the results as what has caused the failure.



As you can see in the screenshot below we can check the failed results and figure put what has caused the error (as shown in the error message below : Expected is “*” but: was “+”



19. Now once we revert back to the original source code, we can see that build is successful.

And also we can see the test results trend.

Branch: master ▾ | Find file | Copy path

JenkinsProject / gameoflife-core / src / main / java / com / wakaleo / gameoflife / domain / **Cell.java**

wakaleo Restored working tests — 118e088 on Apr 12, 2013

2 contributors

32 lines (25 sloc) 678 Bytes — Raw | Blame | History

```java
package com.wakaleo.gameoflife.domain;

public enum Cell {
    LIVE_CELL("*"), DEAD_CELL(".");

    private String symbol;

    private Cell(final String initialSymbol) {
        this.symbol = initialSymbol;
    }

    @Override
    public String toString() {
        return symbol;
    }

    static Cell fromSymbol(final String symbol) {
        Cell cellRepresentedBySymbol = null;
        for (Cell cell : Cell.values()) {
            if (cell.symbol.equals(symbol)) {
                cellRepresentedBySymbol = cell;
                break;
            }
        }
        return cellRepresentedBySymbol;
    }

    public String getSymbol() {
        return symbol;
    }
}
```

3. Conclusion

Now that you've had the opportunity to try Jenkins for yourself and perform a pipeline deployment job, you should have a good understanding on how easy it is to incorporate a Continues Integration process into your existing delivery workflow.