Software Quality Management & Assurance

Alin ZAMFIROIU

alin.zamfiroiu@csie.ase.ro

Content

- Quality
- Software quality
- Quality assurance
- Principles of cybersecurity management
- Continuous Integration
- Continuous Integration systems
- Jenkins

Quality

- The degree to which a system, component, or process meets specified requirements.

- The degree to which a system, component, or process meets customer or user needs or expectations.

Quality

- The question of how quality is defined remains subjective, although a variety of definitions are commonly accepted.

https://proqc.com/blog/what-does-quality-mean/

Quality

- Eight dimensions of Quality:
  1. Performance,
  2. Features,
  3. Reliability,
  4. Conformance,
  5. Durability,
  6. Serviceability,
  7. Aesthetics,
  8. Perceived Quality.

https://sloanreview.mit.edu/article/what-does-product-quality-really-mean/

1. Performance

- Performance is the first and the most important aspect for quality.
- The performance is regarding to different characteristics of the product.
- The performance of a product would correspond to its objective characteristics, while the relationship between performance and quality would reflect individual reactions of customers.

2. Features

- Features of the product reflects the second dimension of the quality.

characteristics (performance) **!=** characteristics (features)

- Features, like product performance, involve objective and measurable attributes; their translation into quality differences is equally affected by individual preferences.
- The distinction between the two is primarily one of centrality or degree of importance to the user.

3. Reliability

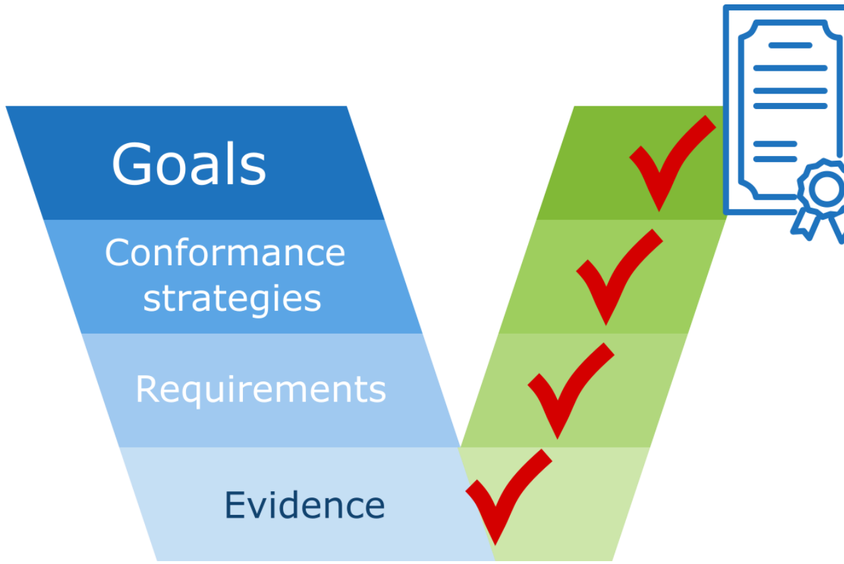
- The third dimension of quality reflects the probability of a product to fail in a period of time.

- The most common measures for reliability are:
  - **MTFF** – Mean Time to First Failure;
  - **MTBF** – Mean Time Between Failures;
  - **FRUT** – Failure Rate per Unit Time.

4. Conformance

- Characteristics of a product should match to preestablished standards.

- Both reliability and conformance are closely tied to the manufacturing-based approach to quality



https://www.argevide.com/conformance/

5. Durability

- Is a measure of product life.
- It can be technical or economical.
- **Durability** and **Sustainability** are closely linked

https://prasadmodakblog.com/2019/09/16/product-durability-repairability-and-sustainability/

6. Serviceability

- The speed, courtesy, and competence of repairing for the product.

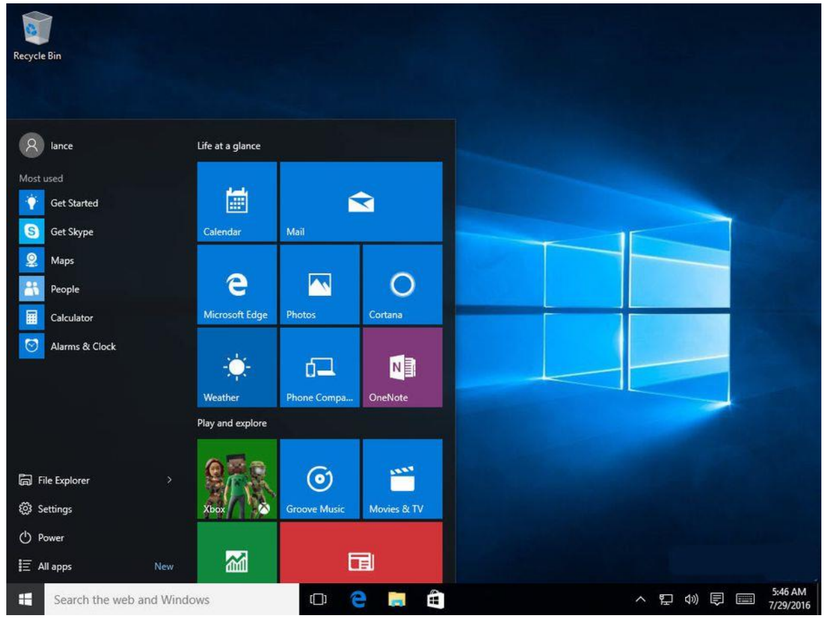
- It is very important for the customer to have a reduced downtime with higher quality.

- For that a lot of companies promise that they will deliver repairs in short time: 24-48 hours .

7. Aesthetics

- Is a subjective characteristic of a product and consist in perception of the consumer/user of that product.

- Aesthetics — how a product looks, feels, sounds, tastes, or smells — is clearly matters of personal judgment, and reflections of individual preferences.



8. Perceived quality

- May be based on images, advertising and brand names.
- Perception is not always the reality.
- Is different from a person to other person.

Software quality

- Conformance to **explicitly stated** functional and performance requirements, **explicitly documented** development standards, and implicit characteristics that are expected of all professionally developed software.

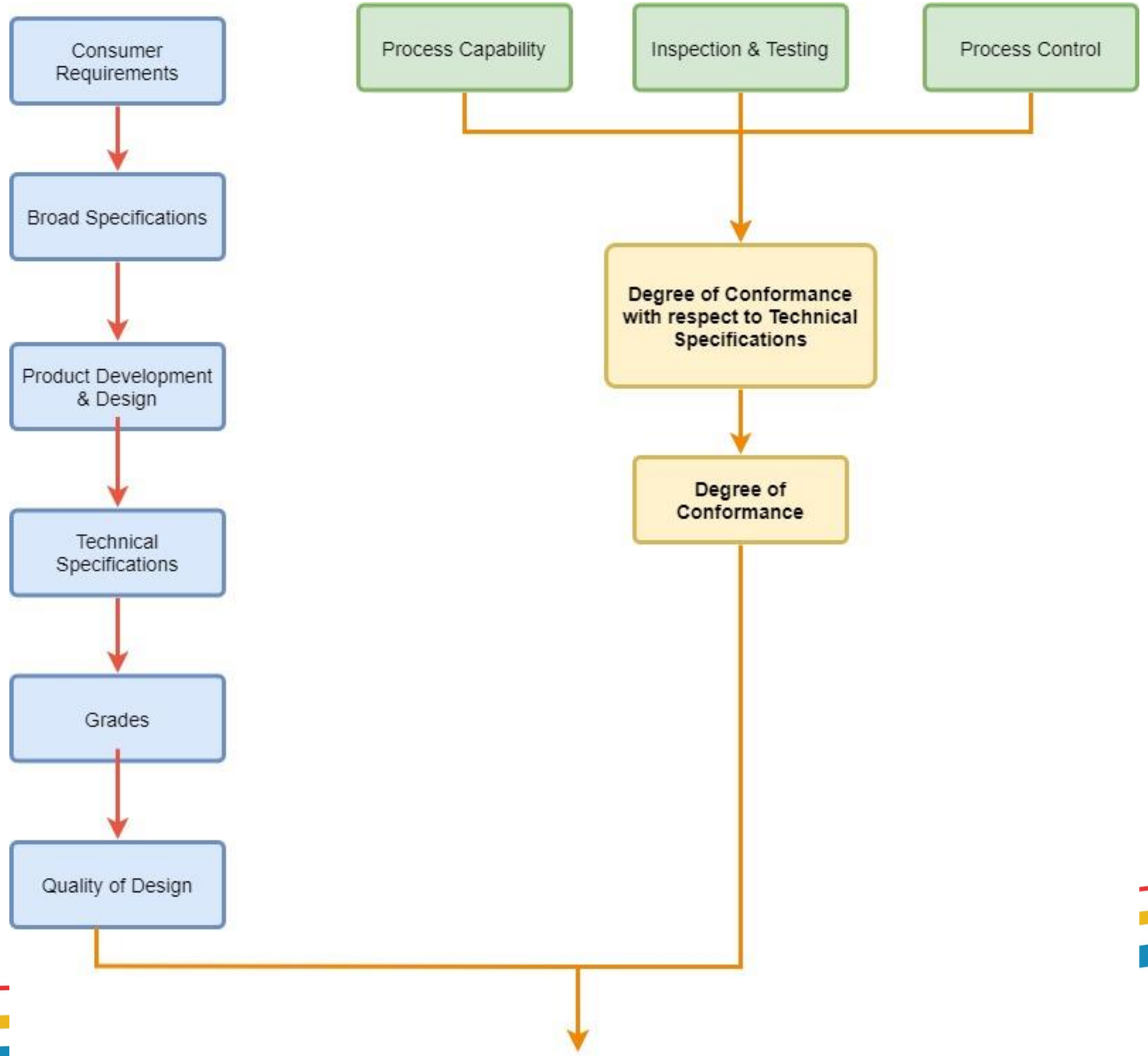
Quality assurance

- A **planned** and **systematic pattern** of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.

- **A set of activities** designed to evaluate the process by which products are developed or manufactured.

Quality assurance

- The process capability, inspection and process control is involved in achieving this conformance so that product/goods produced meet the pre-decided specifications



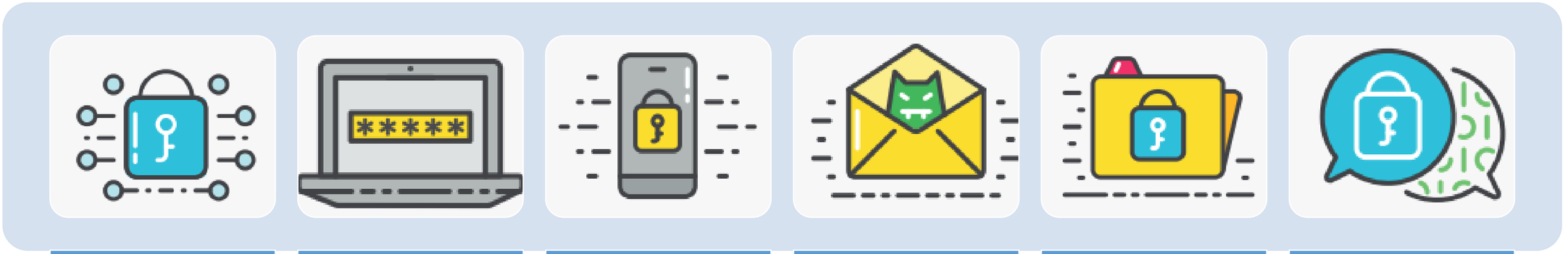
Important Items of Cybersecurity Management

Cybersecurity management

- Cybersecurity is a management issue.

- Usually the information security breaches as caused by human error because:
    - They are greedy and click all e-mails;
    - They don’t keep their computers updated;
    - They don’t use antivirus solutions;
    - etc.

Principles of cybersecurity management



Security beyond firewall

Advanced access management

Enhanced application security

Trusted attack simulation

Data encryption

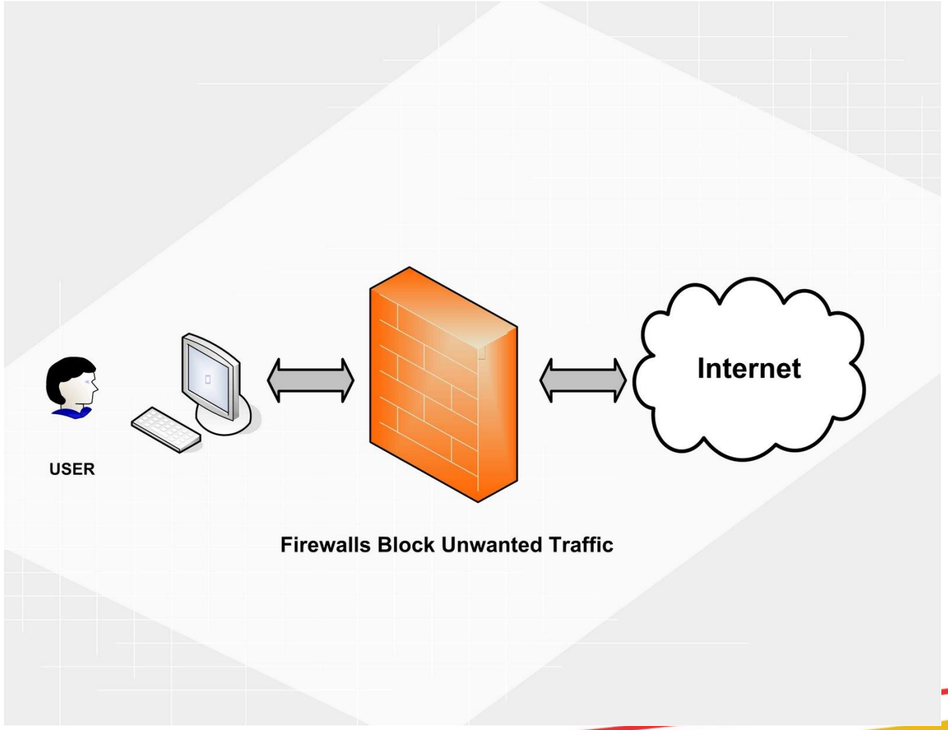
Compliance business framework

Principles of cybersecurity management

#1. SECURITY BEYOND FIREWALL

- Network security used to be achieved by scanning network trac on various OSI layers.

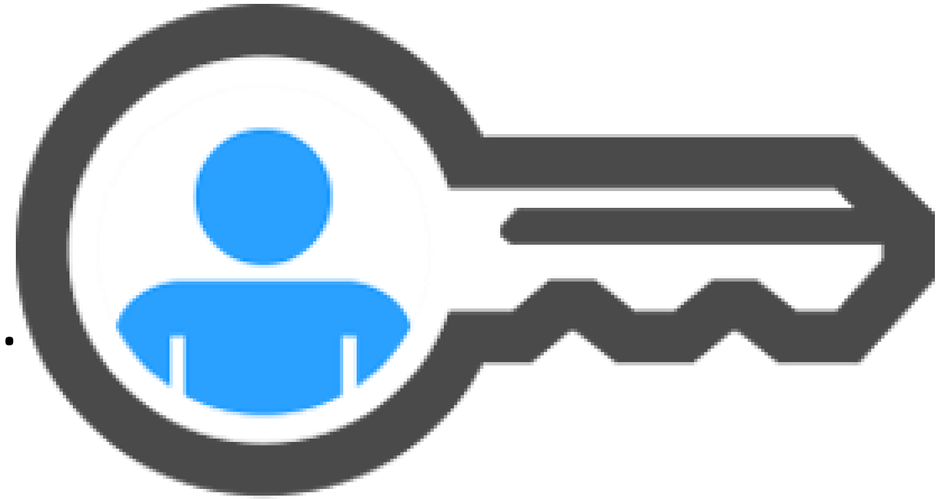
- Some tools are able to look and analyze the suspicious patterns of trac too identify and protect the system against fraud.



Principles of cybersecurity management

#2. ADVANCED ACCESS MANAGEMENT

- If you still use a username and password to access your systems you should seriously consider moving to an advanced access management solution.



- In today’s world, a combination of username and password is no longer secure enough.

Principles of cybersecurity management

#3. ENHANCED APPLICATION SECURITY

- In addition to security measures on the network, most systems are secured with an antivirus solution.

- Enhanced application security consists of two additional measures:
  - 1) security driven release management, where applications, related patches, and service packs are updated for security reasons and not for new functionality;
  - 2) pattern recognition in the application that allows for automatic detection of suspicious behavior. Most of these systems come with a machine learning code.

Principles of cybersecurity management

#4. TRUSTED ATTACK SIMULATION

- One of the most important cyber security principles is to identify security holes before hackers do.

- Trusted Attack Simulation, simulates attacks from outside and inside the system, and gives a report that identifies potential security holes in the system.

Principles of cybersecurity management

#5. DATA ENCRYPTION

- Any data can be stolen, both when it is in transit, or directly from the servers and storage, where the data is at rest.

- The data encryption principle addresses two stages of encryption:
  - 1) Encryption in Transit (EIT);
  - 2) Encryption At Rest (EAR).

Principles of cybersecurity management

#6. COMPLIANCE BUSINESS FRAMEWORK

- Last, but not least, any company that uses data from internal sources, a cloud, or any third party provider, needs to develop its Compliance Business Framework (CBM) for security.

- Mostly the CBM is linked to other compliance policies such as ISO9001, ISO27001 and so forth.

Continuous Integration

Continuous Integration

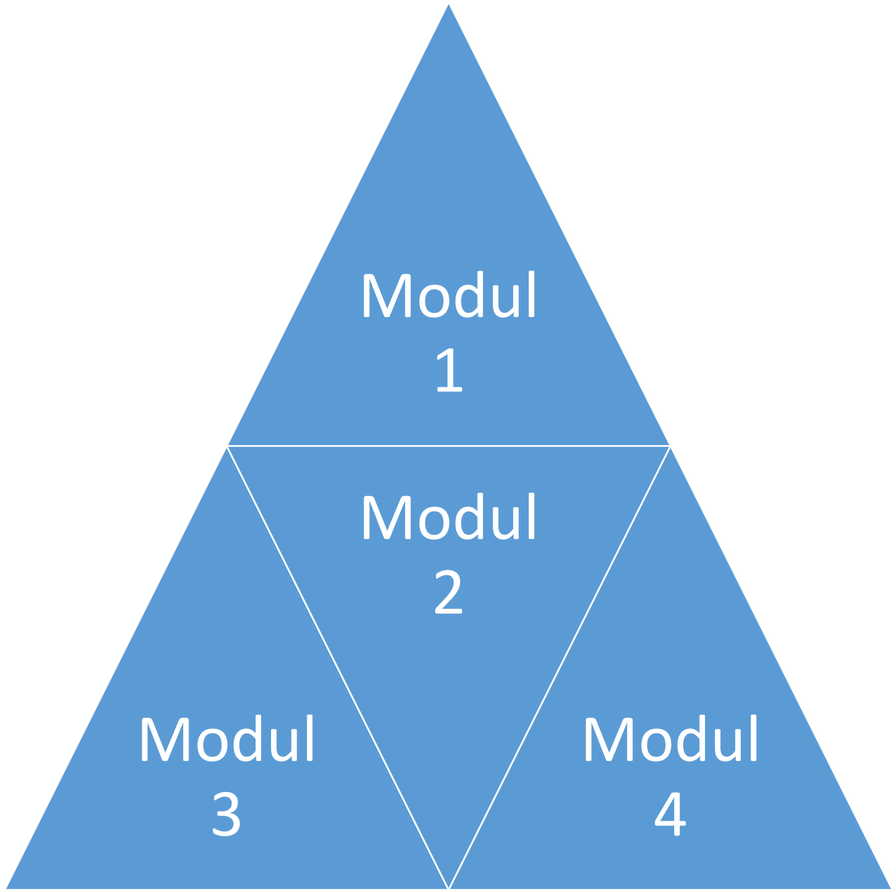
- What is Continuous Integration?

- Developers practicing continuous integration to merge their changes in the main branch as often as possible.

- The developers changes are validated by creating a build and running automated tests.

Continuous Integration

- Why do we need Continuous Integration?



Continuous Integration

- What do we need for Continuous integration
    - Version control;
    - Build automation;
    - Test automation;
    - Often commits;
    - Build on change.

Continuous Integration

- Benefits of continuous integration:
    - Defect early discovery;
    - Automatic application deploy;
    - Transparent health monitor
    - Test pipeline customization;
    - Build-in parallel execution.

https://www.katalon.com/resources-center/blog/benefits-continuous-integration-delivery/

Continuous Delivery

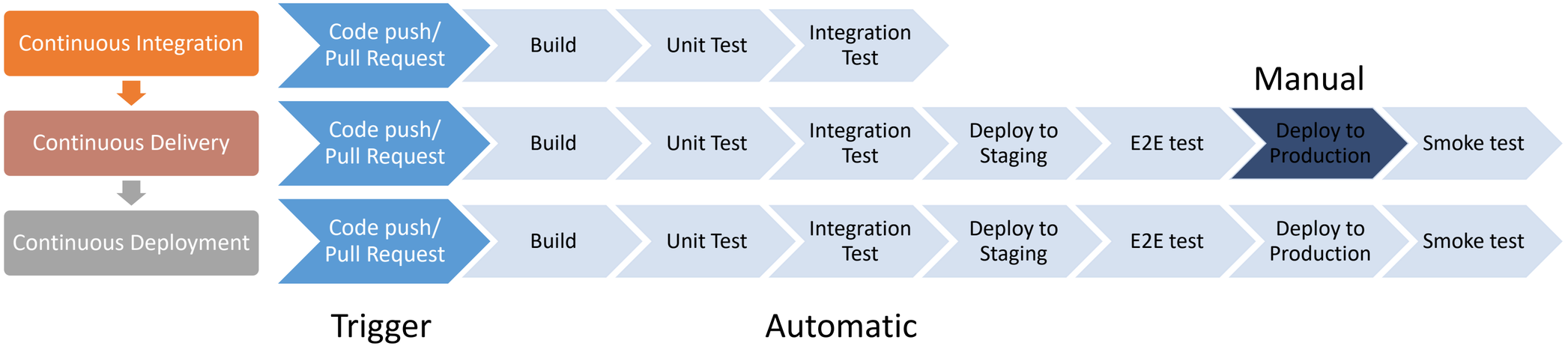
- Is an extension of Continuous integration to make sure that the developer can release new changes to the customers quickly in a sustainable way.

Continuous deployment

- This one goes one step further than the continuous delivery.
- Every change that passes all stages of the production pipeline is released directly to the customers without human intervention.

Continuous Integration

- CI & CDy & CDt

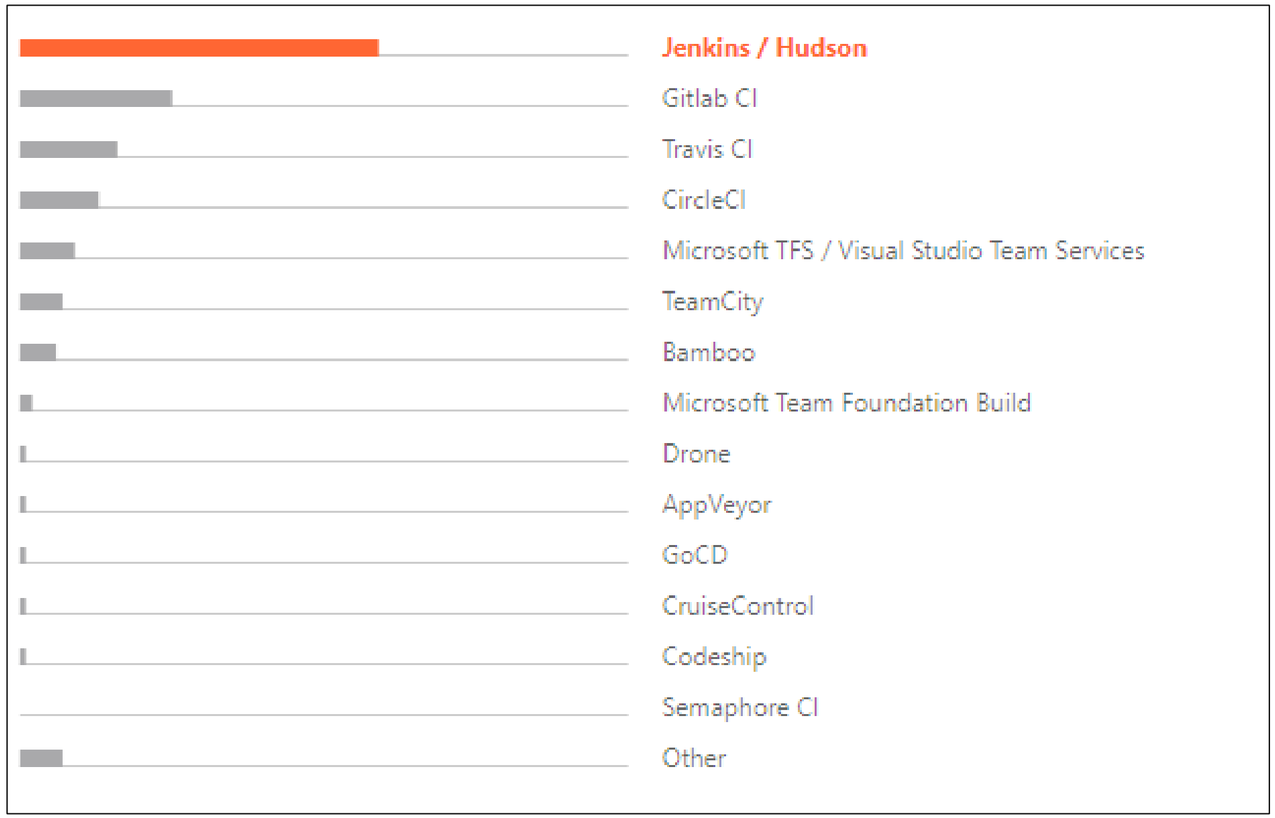


Continuous Integration systems

- The most used CI systems are:
  - Jenkins
  - TeamCity
  - Travis
  - Gitlab
  - CircleCI
  - GoCD
  - Bamboo
  - CodeShip
  - Buldbot
  - Strider CD
  - Drone
  - AppVeyor

Continuous Integration systems

- A ranking of using these systems are presented on JetBrains site:



https://www.jetbrains.com/lp/devecosystem-2019/team-tools/

Continuous Integration systems

- **TeamCity**
  - it is used in industry and supports many powerful features;
  - has many open source plugins;
  - the continuous integration server is always stable;
  - commits can be pre-tested and command remote-run.
- **Travis CI**
  - is very popular for continuous integration that is free tool for open source projects;
  - it is easy to setup without any installation required;
  - it is supported on the most operating systems;
  - is integrated with communication services like slack, hipchat, etc.

Continuous Integration systems

- **GitLab CI**
    - Is a web application with an API that stores its state in a database;
    - provides a friendly and intuitive user interface;
- **Circle CI**
    - Is a flexible tool that runs in any environment
    - It support many languages as Javascript, C++, PHP, python and also supports docker.

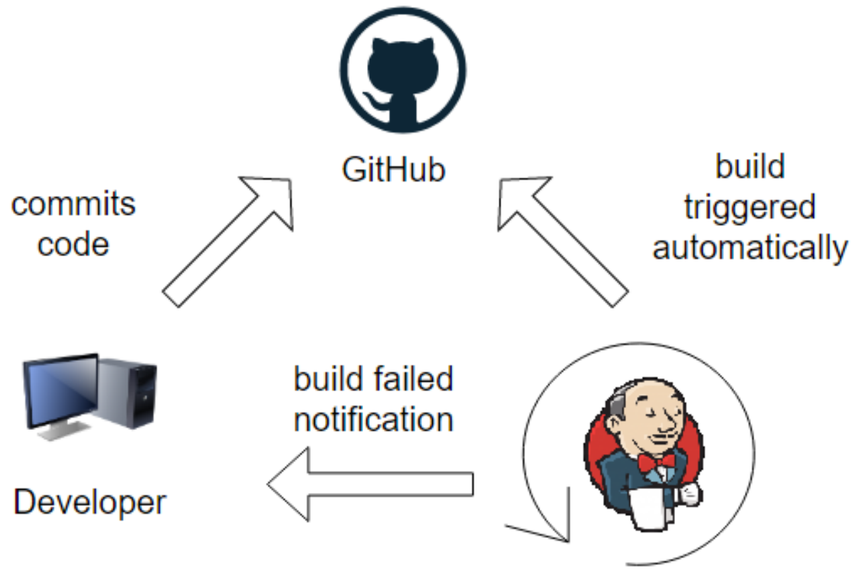
Jenkins

Jenkins - Content

- What is Jenkins
- Advantages & disadvantages
- Install and Running Jenkins
- Jenkins configuration
- Jobs
- Parameterized jobs
- Triggers
- Cron syntax
- Unit testing in Jenkins
- Jenkins with GitHub (Git Integration)
- Jenkins security
- Job Configuration History

What is Jenkins?

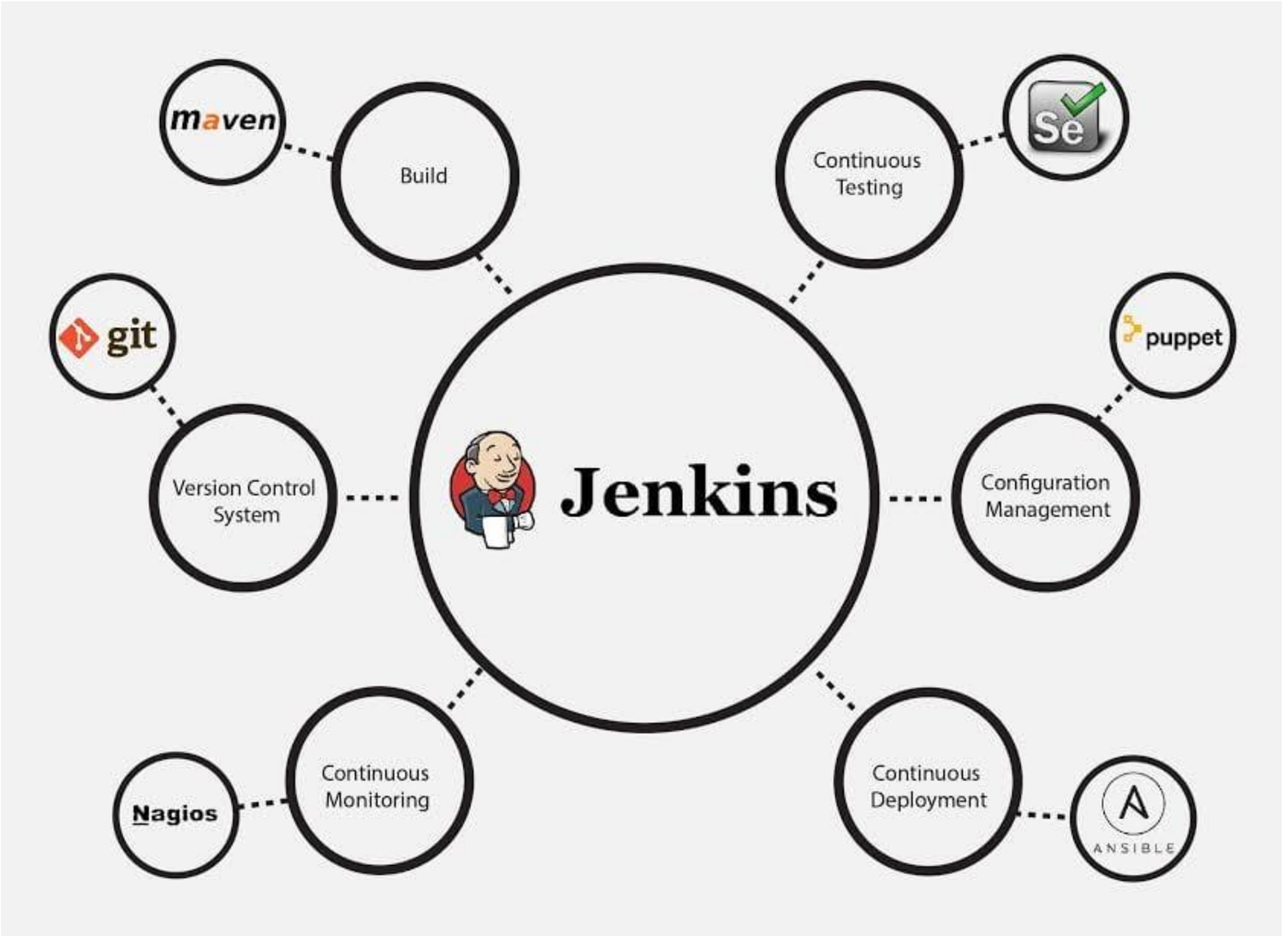
- Jenkins is a Java application used for continous testing and continous delivery for your projects.



https://subscription.packtpub.com/book/networking_and_servers/9781788624480/6/ch06lvl1sec51/continuous-integration-with-github-and-jenkins

What is Jenkins?

- Maven
- GitHub
- Nagios
- Selenium
- Puppet
- Ansible



https://www.edureka.co/blog/what-is-jenkins/

What is Jenkins?

- Jenkins offers a simple way to set up a continuous testing or continuous delivery environment for almost any combination of languages and source code repositories.

- It is capable of orchestrating a chain of actions that help to achieve the Continuous Integration process in an automated fashion.

https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html

What is Jenkins?

- By using Jenkins, you can accelerate the development process for the software.
- Jenkins can automate build and test at a rapid rate.
- Jenkins supports the complete development lifecycle of software from building, testing, documenting the software, deploying and other stages of a software development lifecycle.

Advantages of using Jenkins

- Jenkins is being managed by an open community.
- Jenkins has around 320 plugins published;
- With plugins, Jenkins becomes even more powerful and feature rich.
- Jenkins also supports cloud-based architecture so that you can deploy Jenkins in cloud-based platforms.
- The reason why Jenkins became popular is that it was created by a developers for developers.

Disadvantages of using Jenkins

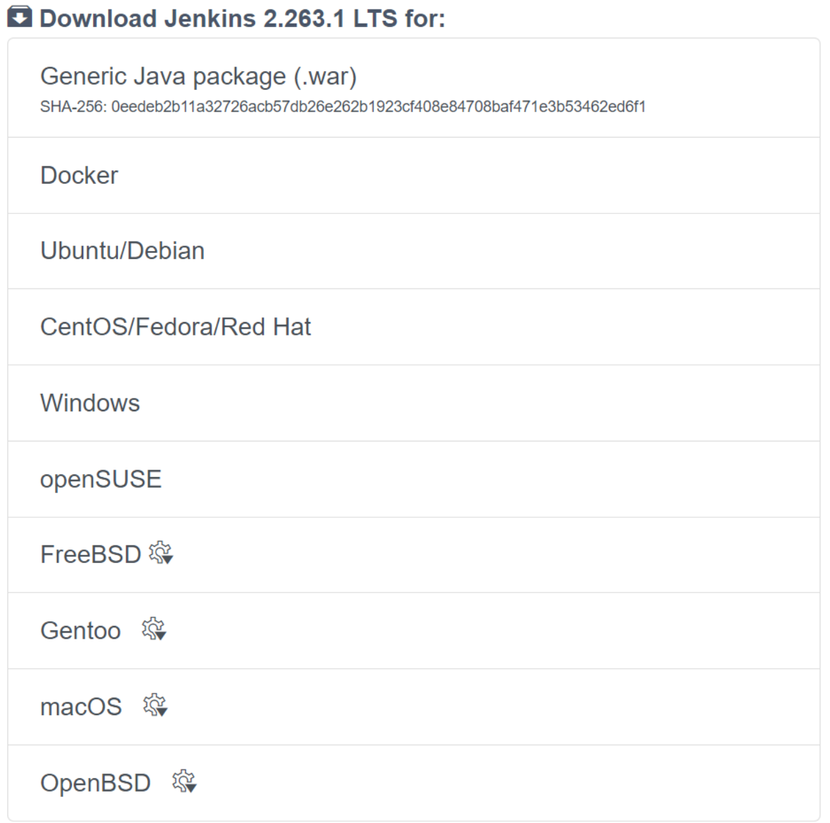
- Its interface is out dated and not user friendly compared to current UI trends.
- Though Jenkins is loved by many developers, it's not that easy to maintain it because Jenkins runs on a server and requires some skills as server administrator to monitor its activity.
- One of the reasons why many people don't implement Jenkins is due to its difficulty in installing and configuring Jenkins.

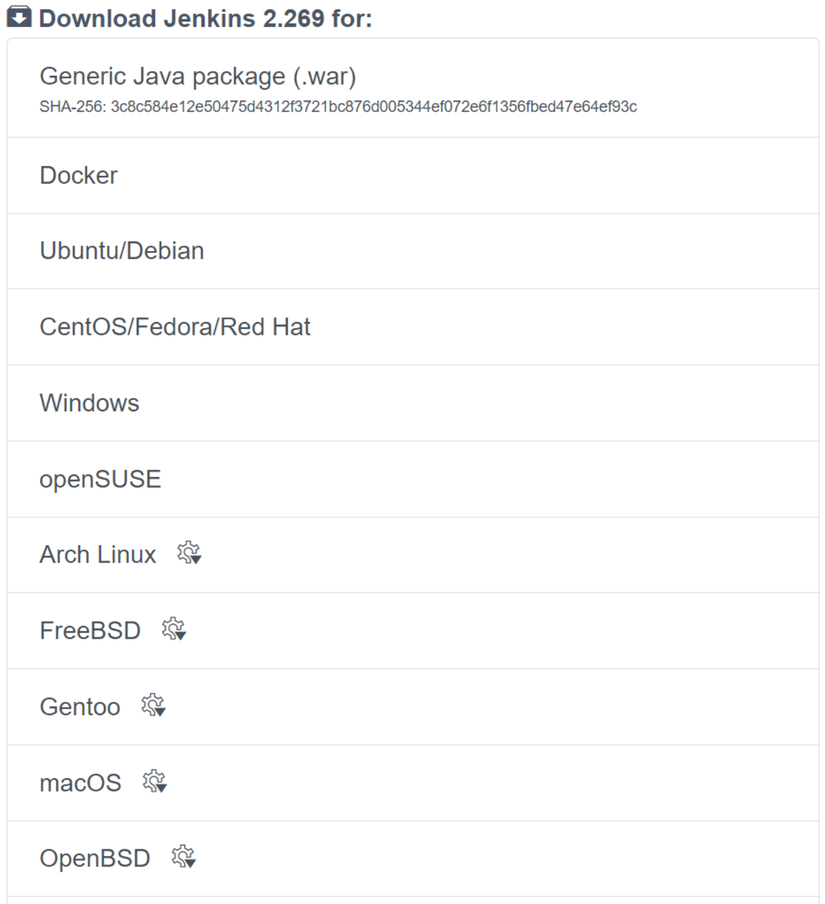
System Requirements

JDK	JDK 1.5 or above
Memory	2 GB RAM (recommended)
Disk Space	No minimum requirement.
Operating System Version	Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS X, openSUSE, FReeBSD, OpenBSD, Gentoo

Install and Running Jenkins

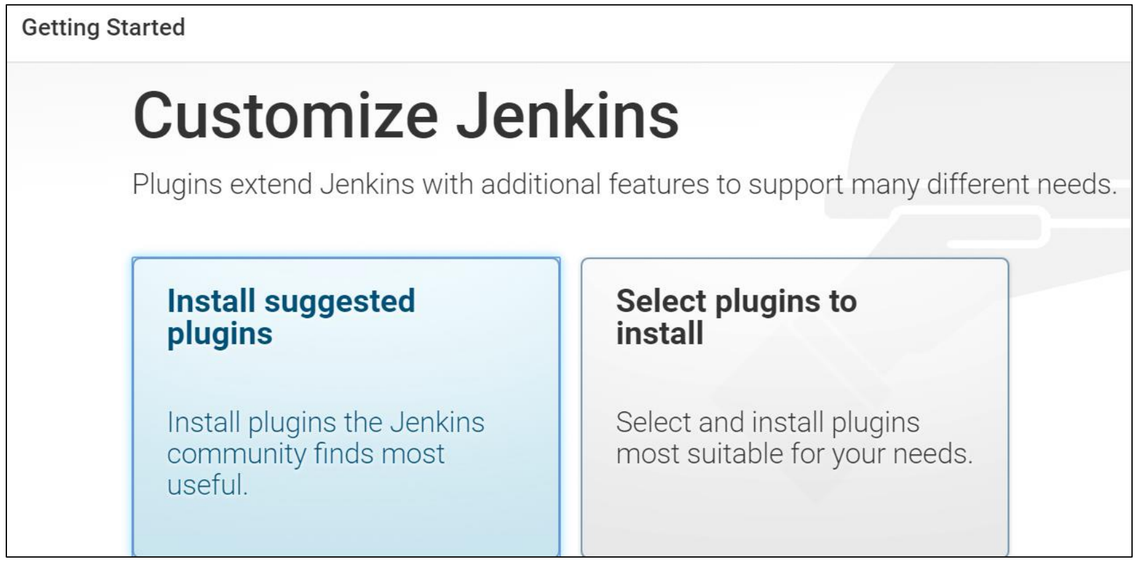
- Go to https://jenkins.io/ and download it.





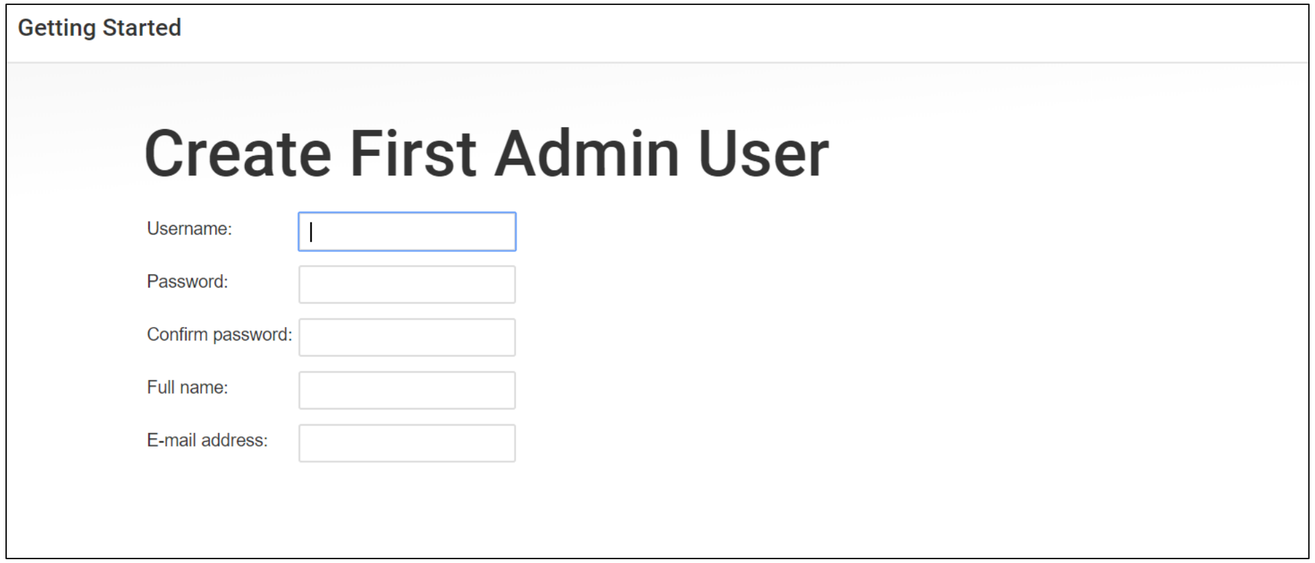
Install and Running Jenkins

- You can select what plugins to install or you can install only suggested plugins and after to install exactly what do you want.



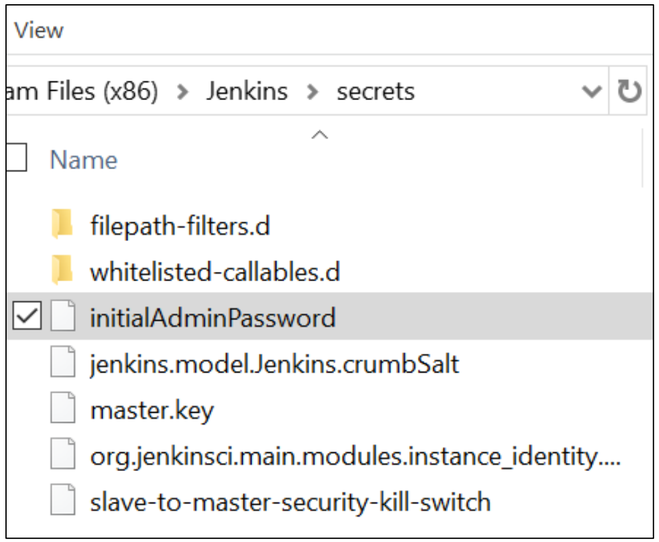
Install and Running Jenkins

- In the installation process you have to create the admin user also.
- Be careful to remember the username and the password for future using of the platform.



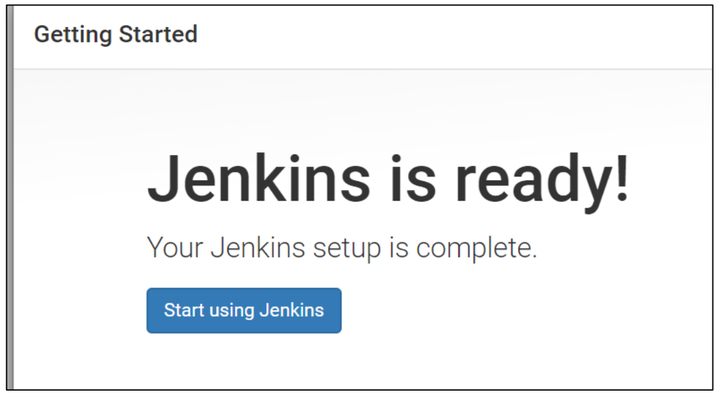
Install and Running Jenkins

- To establish the password you have to access the password file from the install directory



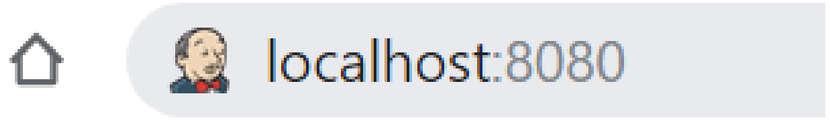
Install and Running Jenkins

- After the confirmation of the password the Jenkins setup is complete and Jenkins is ready to be used.



Access the Jenkins

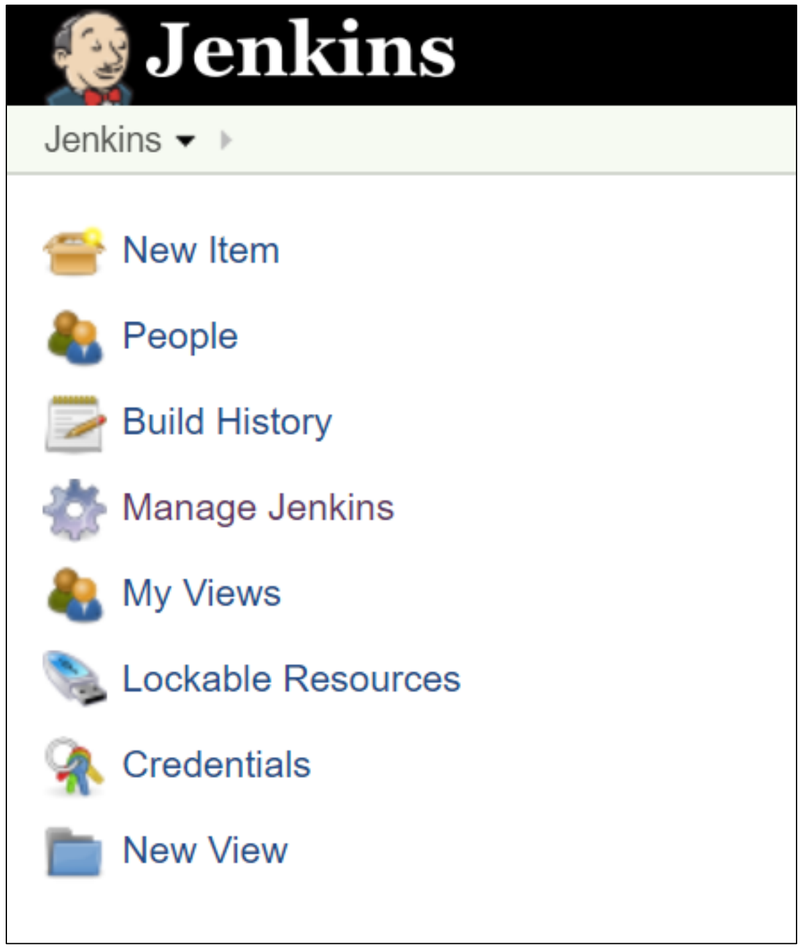
- After the installation you can access the Jenkins at http://localhost:8080



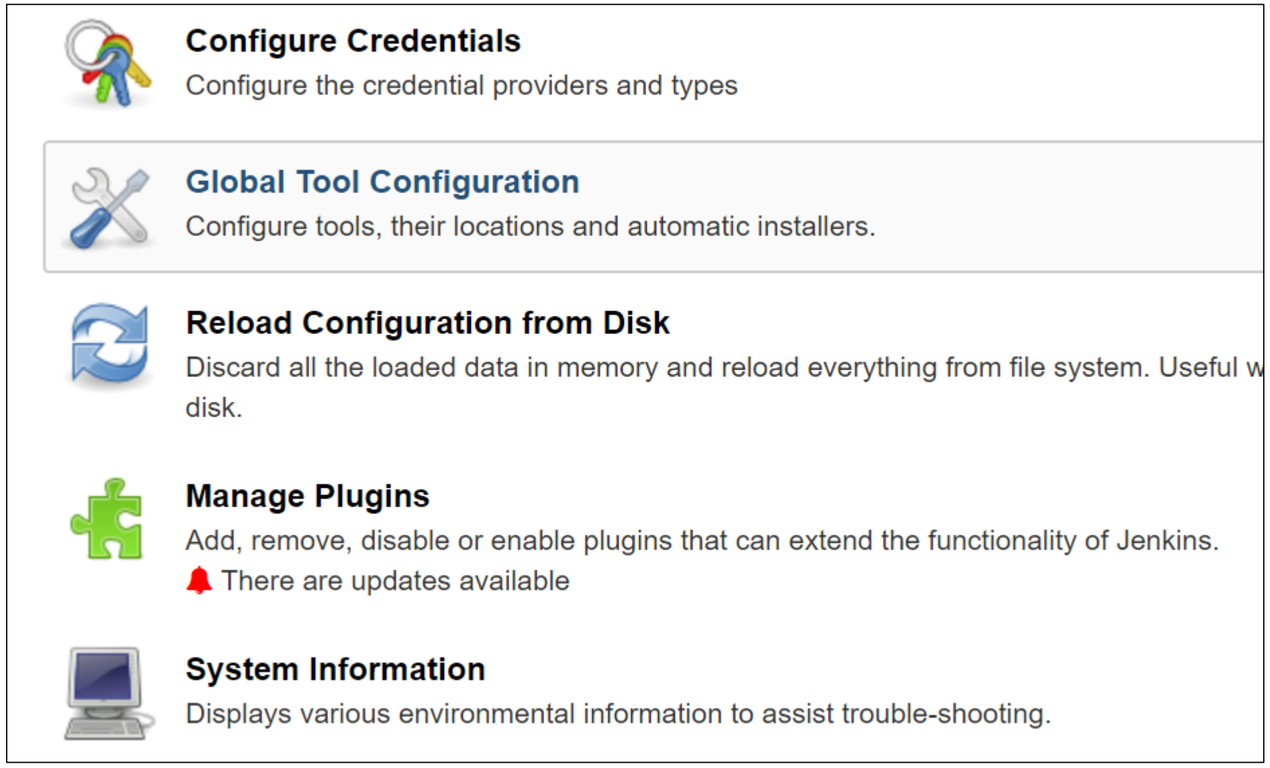
- And you have to login with the username and password established at the previous step.

Jenkins Dashboard

- Here you have the Jenkins Dashboard.
- From this dashboard you can manage everything on the Jenkins server.
- The configurations of the Jenkins I one of the mot difficult step of the installing it.

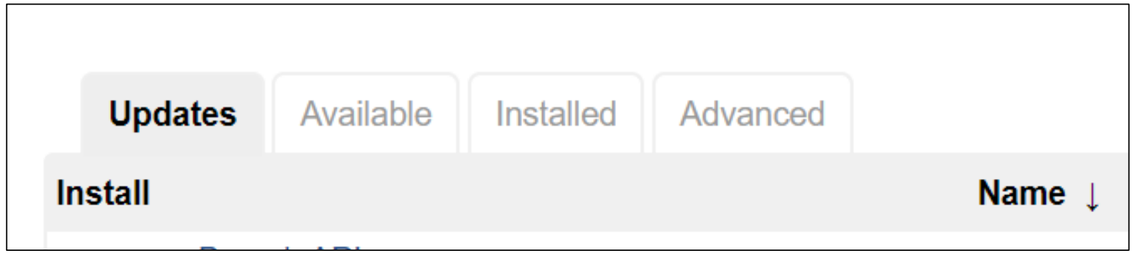


Manage Jenkins



Manage Plugins

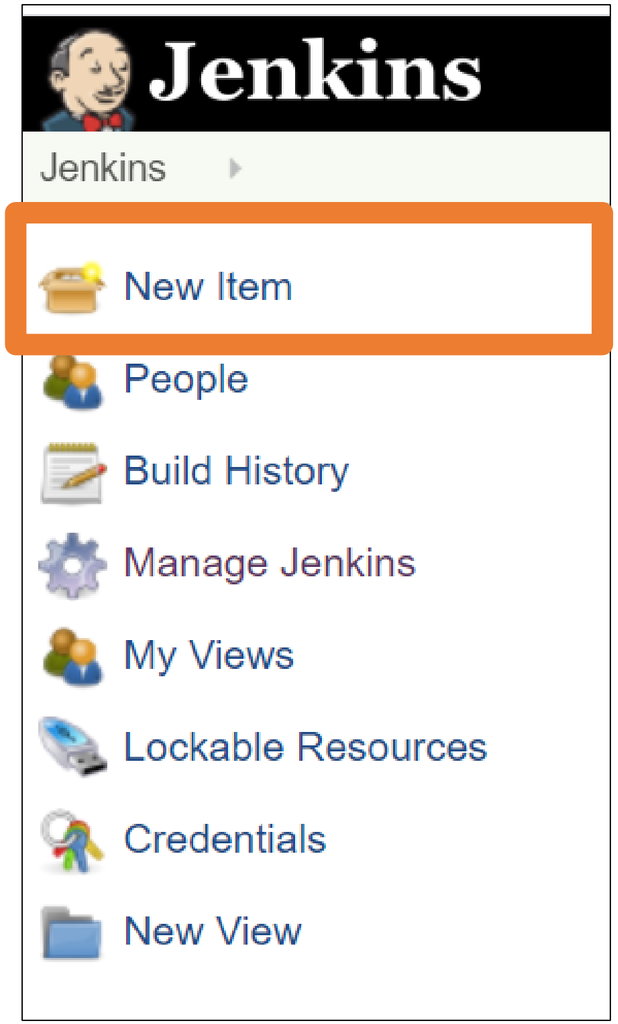
- Here you can manage:
  - available plugins
  - installed pluggins



JOBS

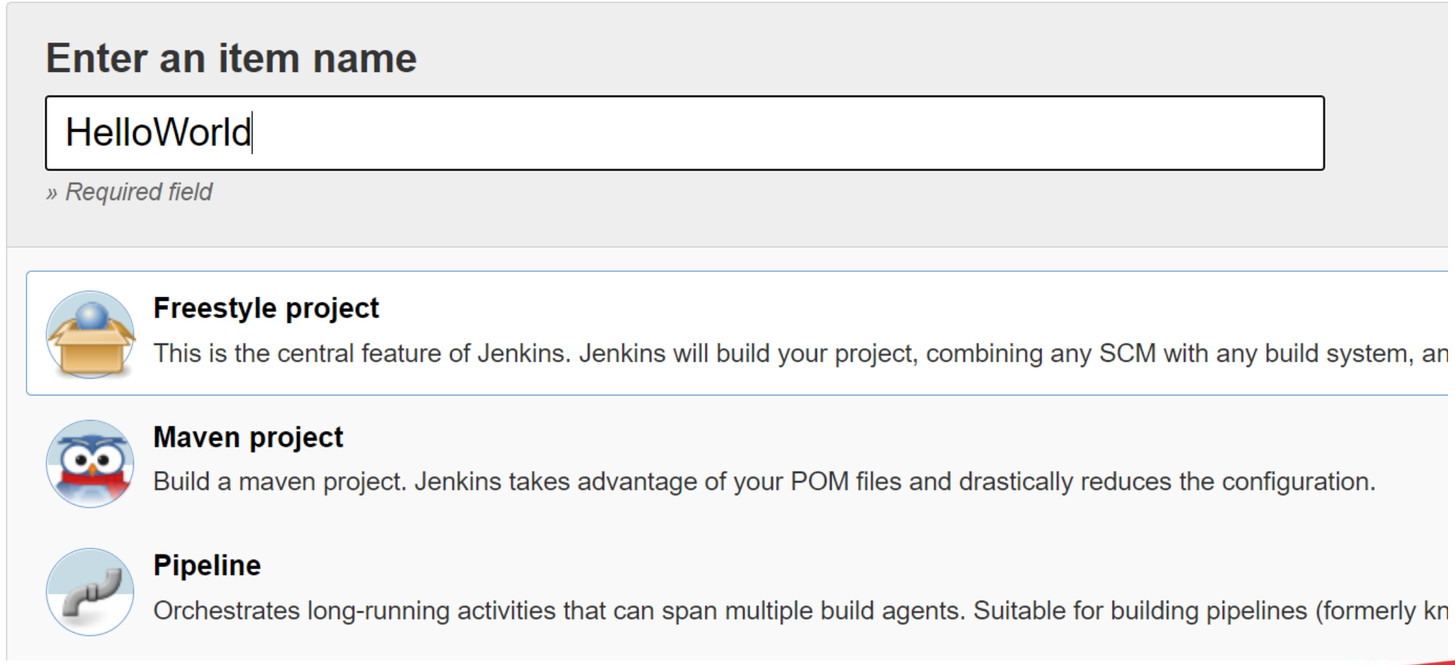
Jenkins – Adding jobs

- To add new jobs in Jenkins, we have few steps.
- After we connect to Jenkins, he have in dashboard the option: **New item**



Jenkins – Adding jobs

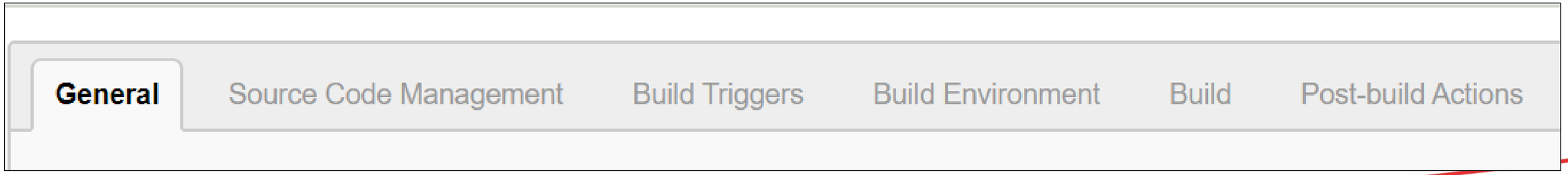
- In next page we have to insert the name of this new job and select **Freestyle project**.



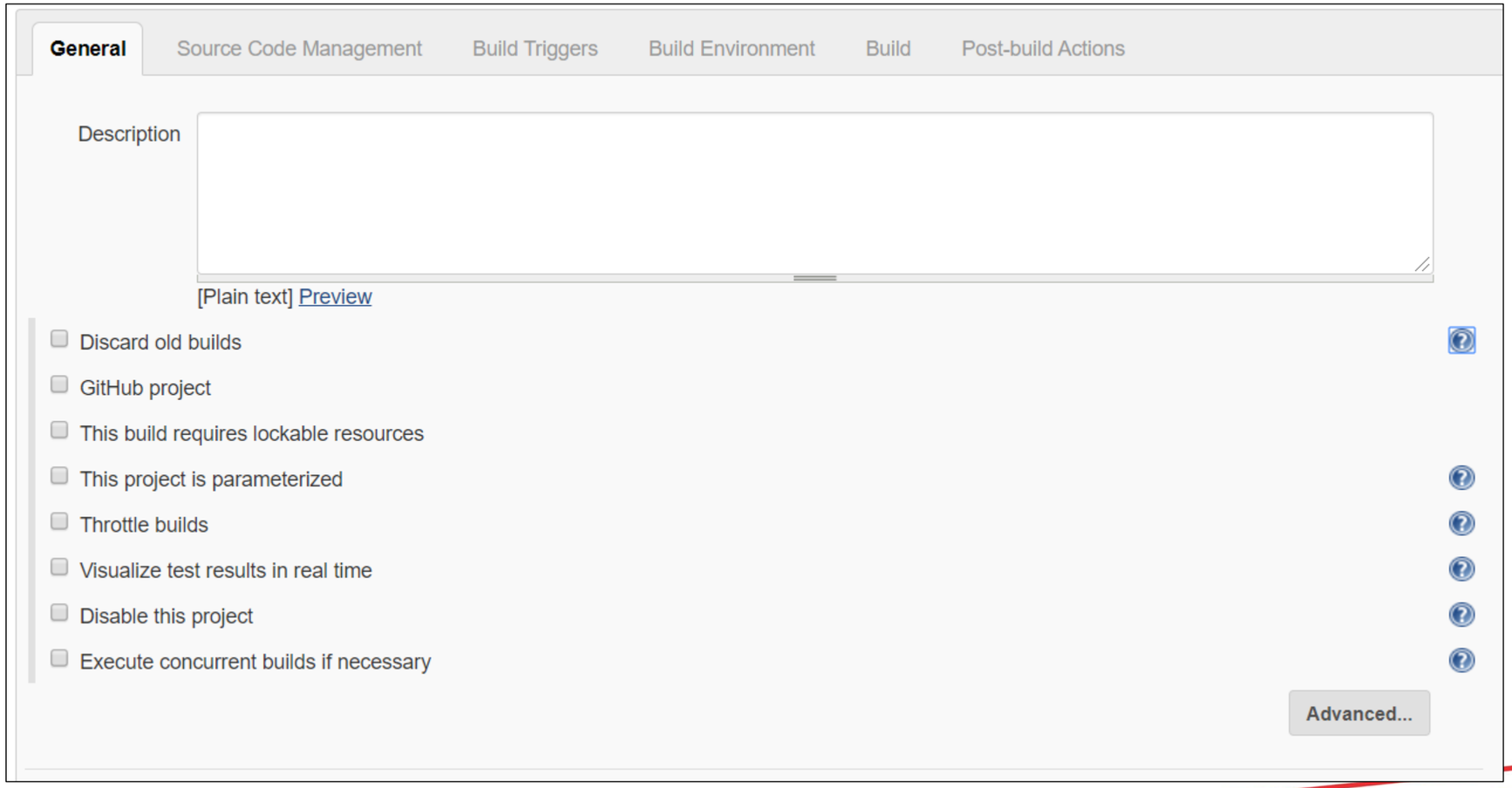
- After you click OK it will be opened a page with more tabs.

Jenkins

- The tabs are:
    - General
    - Source Code Management
    - Build Triggers
    - Build Environment
    - Build
    - Post-Build Actions

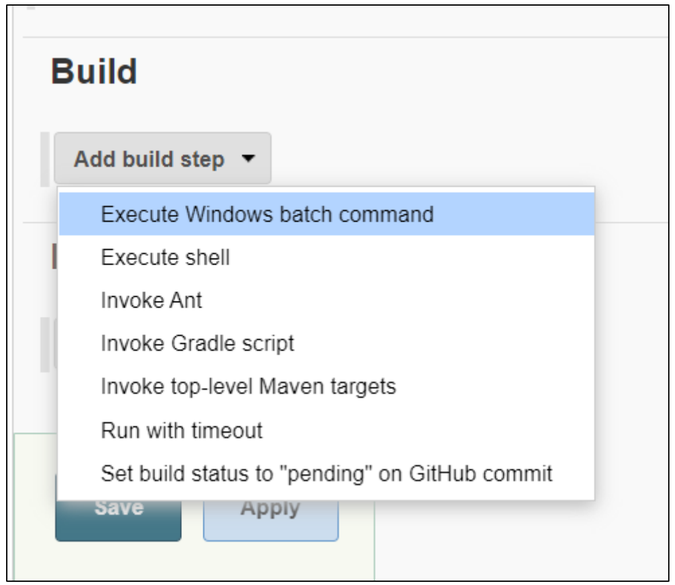


General TAB


General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
[Plain text] Preview
Discard old builds
GitHub project
This build requires lockable resources
This project is parameterized
Throttle builds
Visualize test results in real time
Disable this project
Execute concurrent builds if necessary
Advanced...

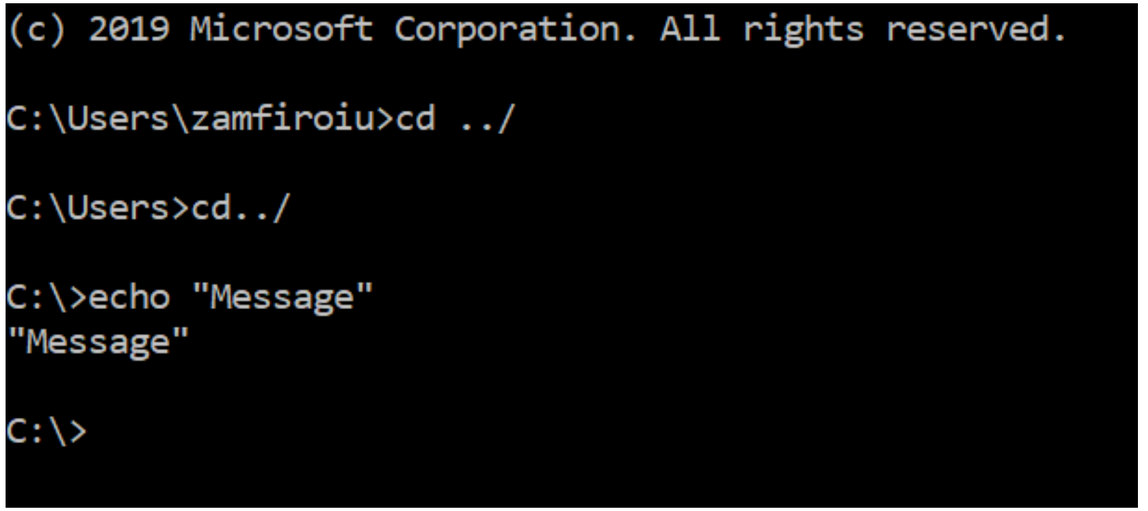
Build TAB

- On build Tab we can manage what we wan to do with this job.
- If we are in Windows OS we can execute a batch command



First job on Jenkins

- For our first job we will use Windows bash commands.



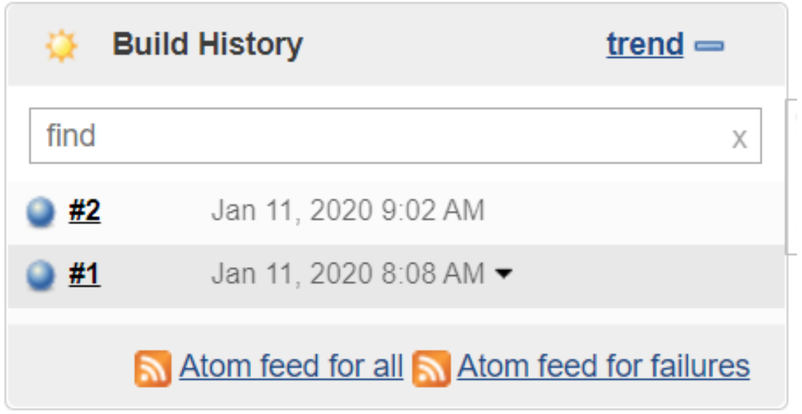
First job on Jenkins

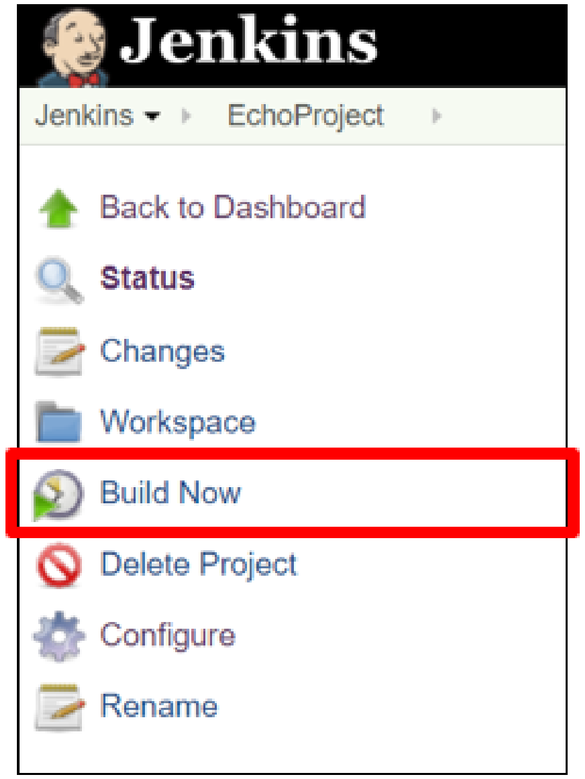
- That assume to put the echo command on Build tab.

First job on Jenkins

- After that on the project we can build the project.

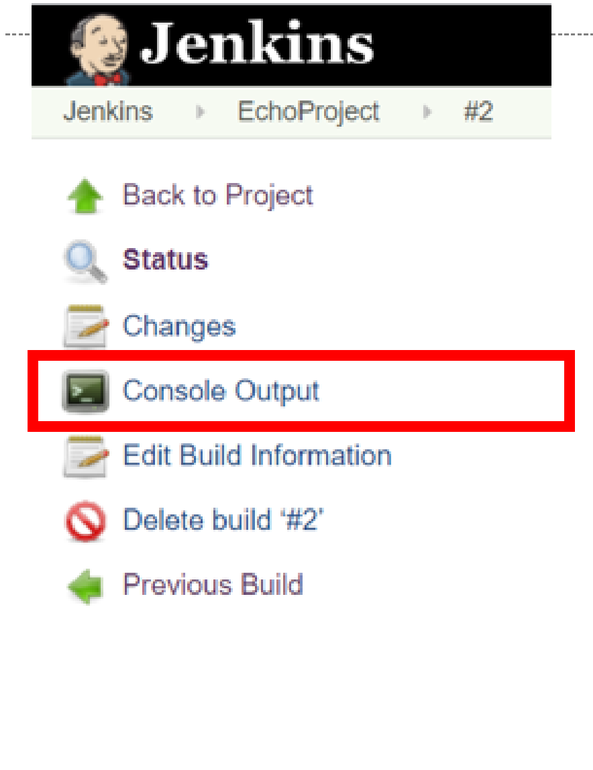
- In Build history we can see all builds.





First job on Jenkins

- On this build we can delete it or we can see the output.

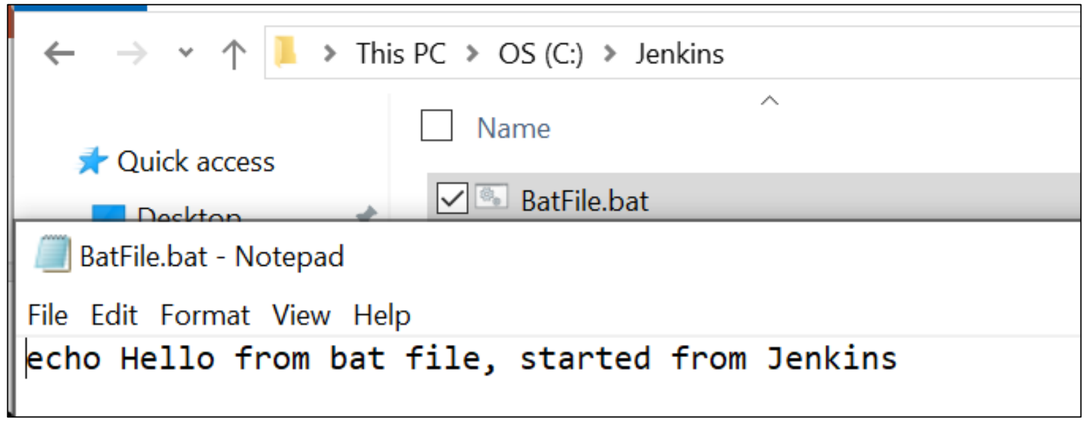


Console Output

```
Started by user Alin Zamfiroiu
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\EchoProject
[EchoProject] $ cmd /c call C:\WINDOWS\TEMP\jenkins515822544073979118.bat

C:\Program Files (x86)\Jenkins\workspace\EchoProject>echo "Hello from jenkins"
"Hello from jenkins"

C:\Program Files (x86)\Jenkins\workspace\EchoProject>exit 0
Finished: SUCCESS
```

Job with Batch file

- We create a bat file in C:/ Jenkins.
- And we test it from Command Line.



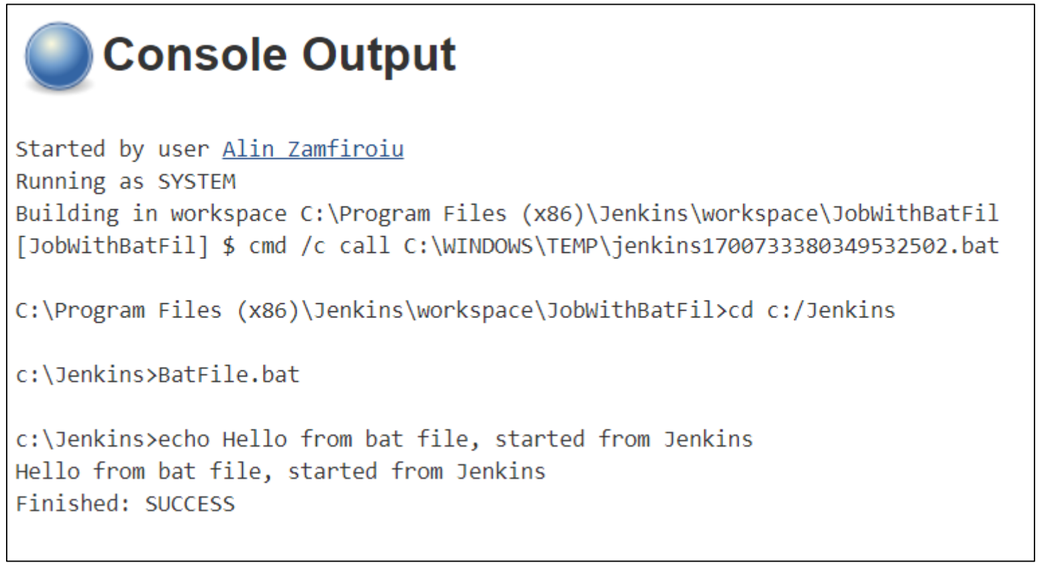

```
C:\Users\zamfiroiu>cd c:/Jenkins

c:\Jenkins>BatFile.bat

c:\Jenkins>echo Hello from bat file, started from Jenkins
Hello from bat file, started from Jenkins

c:\Jenkins>
```

Job with Batch file

- Now we run these two commands in a Jenkins Job.

Job with Batch file

- The result is similar.
- In this way we can create different bat files and run them automatically from Jenkins



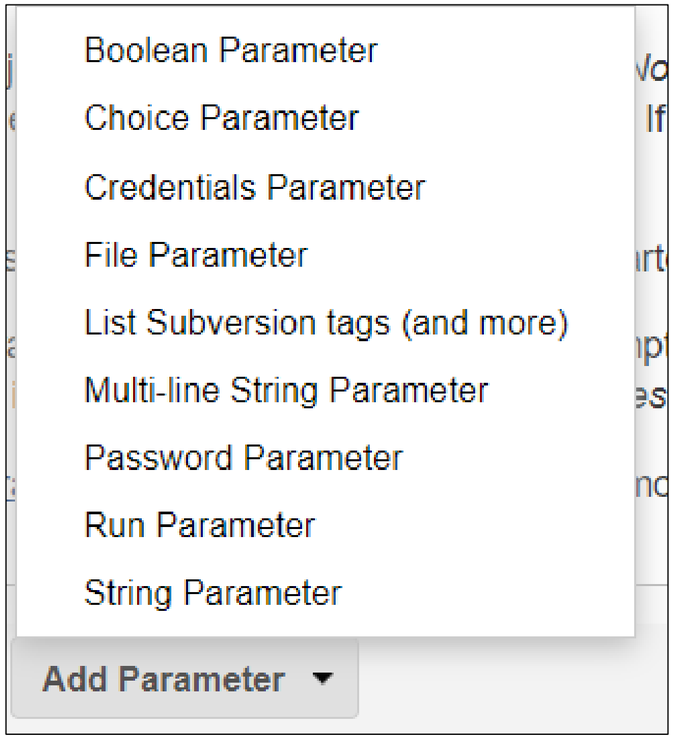
Parameterized jobs

Parametrized Job

- Jenkins allows us to create jobs with different parameters if it necessary.
- For that in the General TAB, we have to select that our project/job is parametrized.

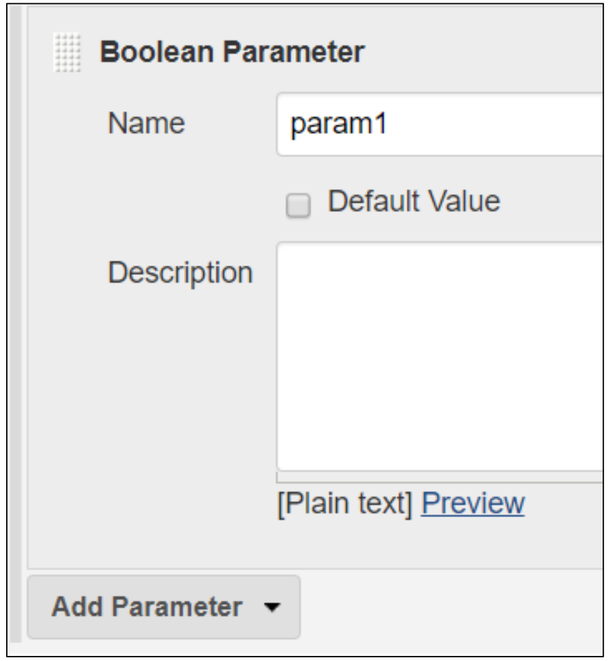
Parametrized Job

- After that, we have to add our parameters.
- Now depends on each job what kind of parameters do we add.



Parametrized Job

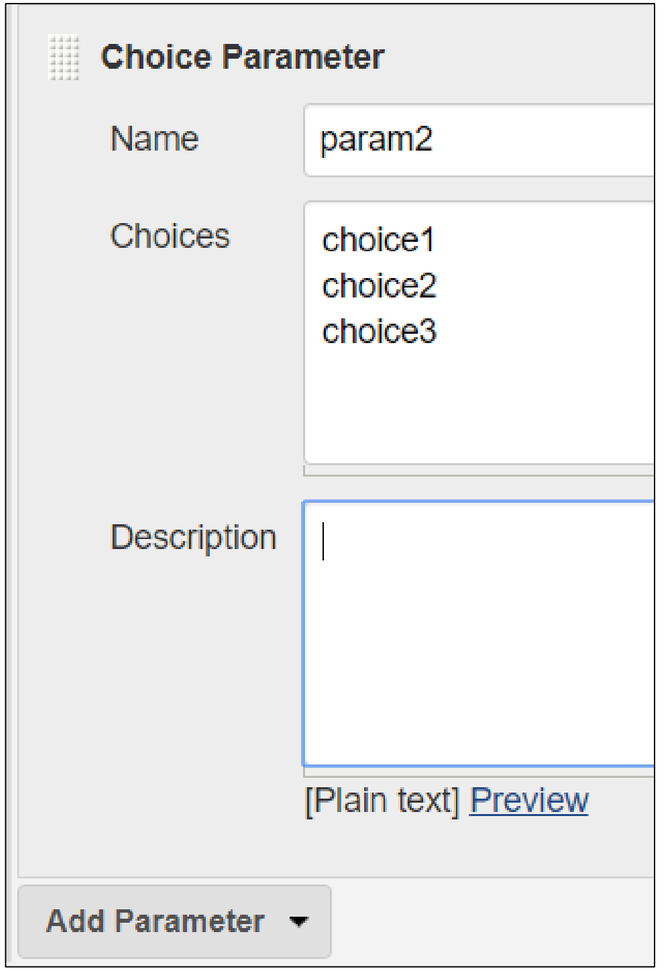
- Boolean parameter it will be use for a TRUE/FALSE value.
- On the description we can set what to be shown to the user on the inserting the value page.
- We can select also the default value to be TRUE.



Parametrized Job

- Choice parameter it is used if the user have to select one of more possible variants.

- For that type of parameter we have to put the name an also the possible choices for the user.

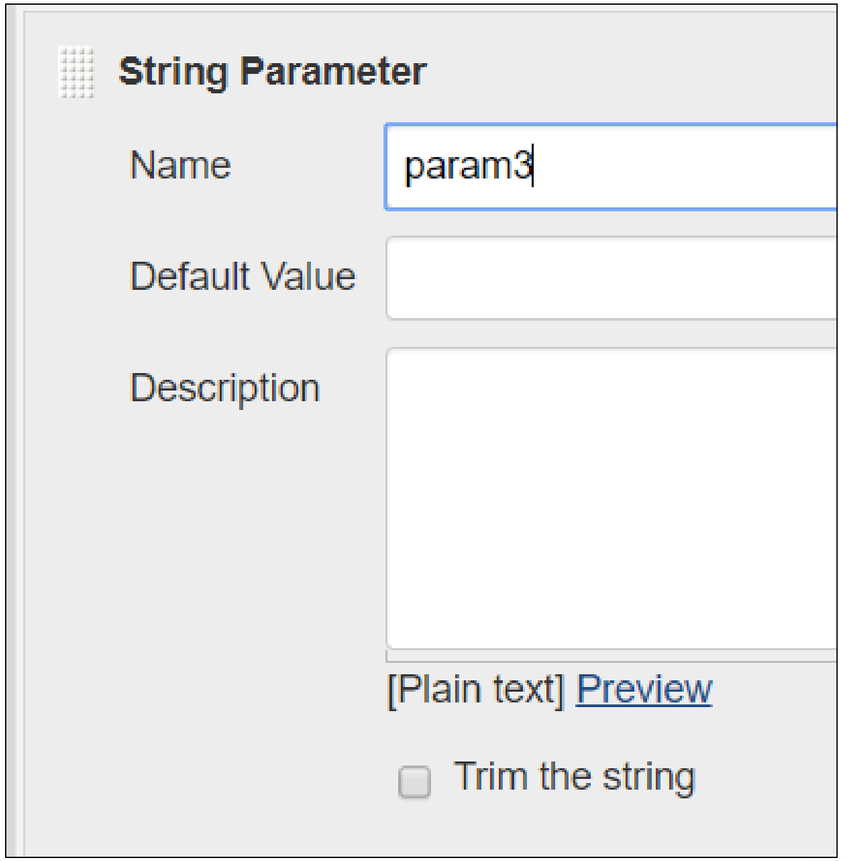


Parametrized Job

- Credential parameter
- File parameter
- List Subversion tags
- Multi-Line String Parameter
- Password Parameter
- Run Parameter

Parametrized Job

- String parameter is used to allow the user to insert any text or value that can be used in the build process.



Parametrized Job

- The name of the parameters should be unique, because these parameters names are used to create environment variables when the build starts.

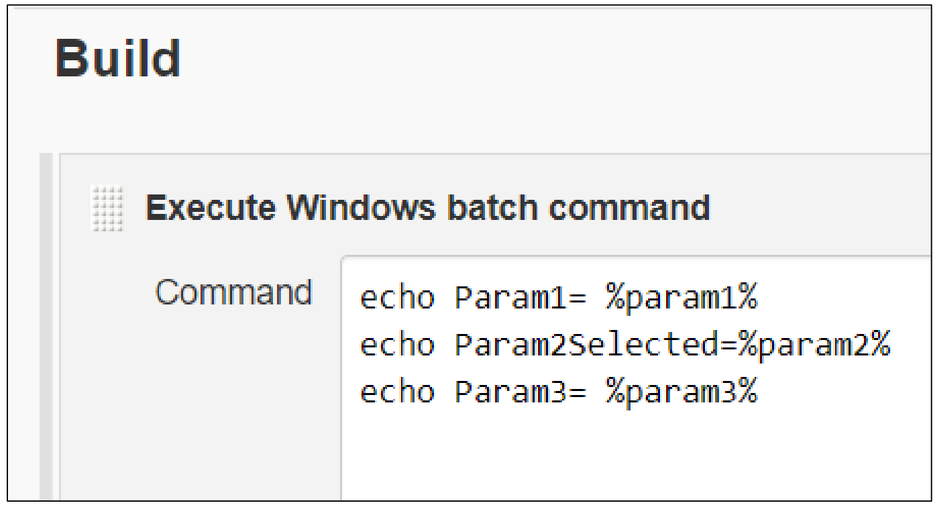
- To use these parameters we have to mention the names:

- Unix: **${Parameter_NAME}**
- Windows: **%Parameter_NAME%**

Parametrized Job

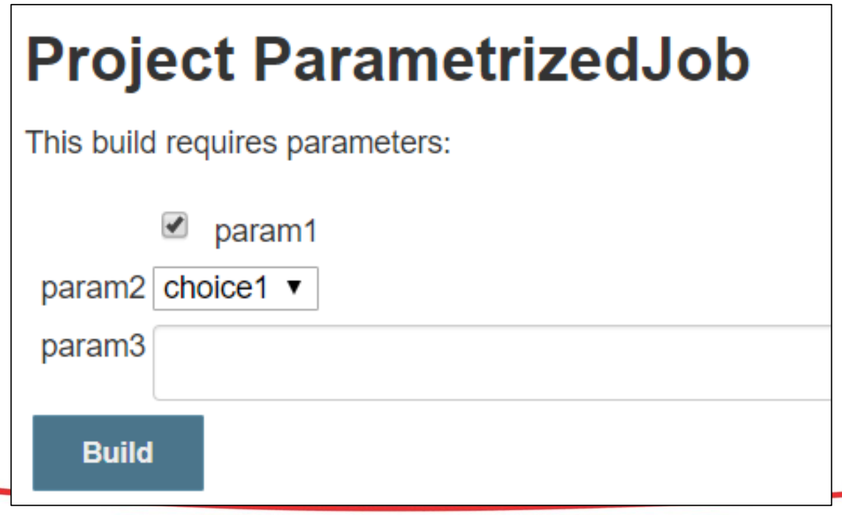
- We will print the inserted values for these parameter.

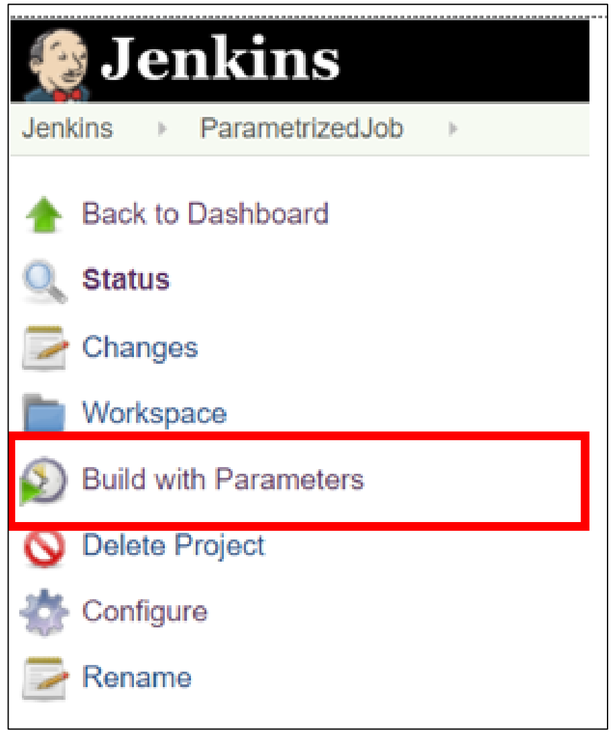
- So the batch commands are:



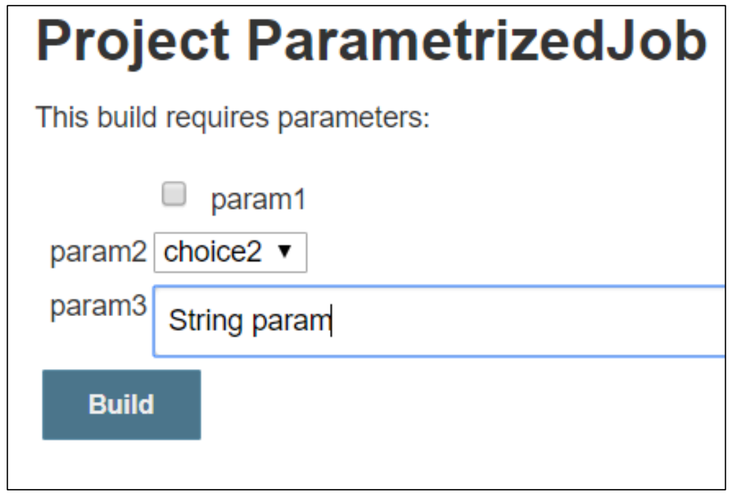
Parametrized Job

- Now, the option “Build now” from the menu is changed in “Build with parameters”, and when it is selected it will be opened a page where the user have to insert the values for all parameters





Parametrized Job

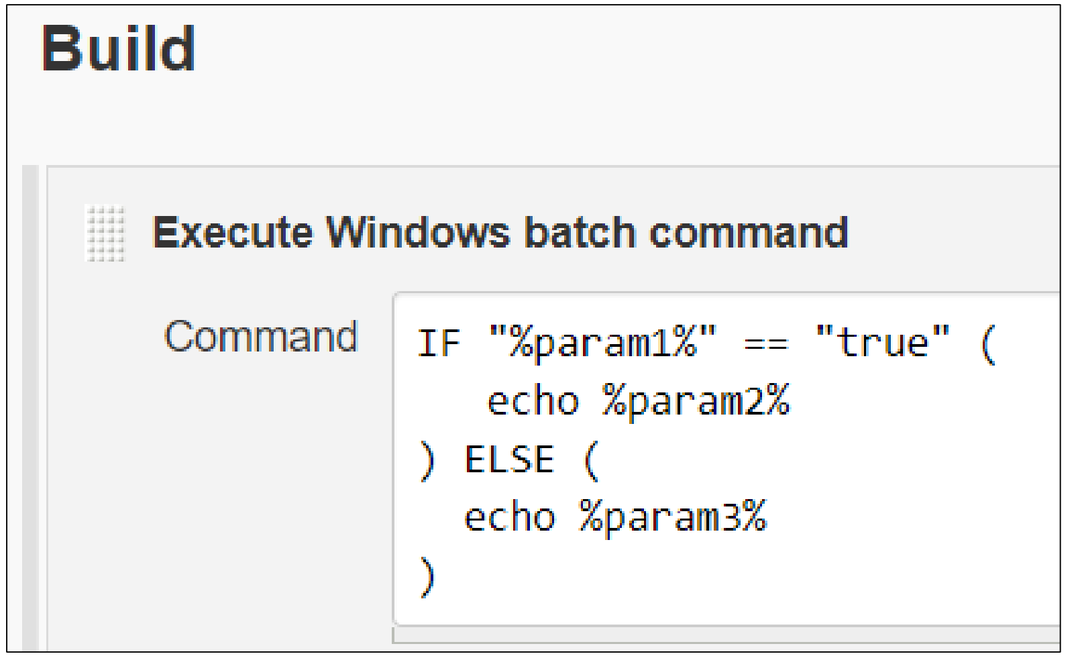


Console Output

```
Started by user Alin Zamfiroiu
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\ParametrizedJob
[ParametrizedJob] $ cmd /c call C:\WINDOWS\TEMP\jenkins7838479397399630871.bat

C:\Program Files (x86)\Jenkins\workspace\ParametrizedJob>echo Param1= false
Param1= false

C:\Program Files (x86)\Jenkins\workspace\ParametrizedJob>echo Param2Selected=choice2
Param2Selected=choice2

C:\Program Files (x86)\Jenkins\workspace\ParametrizedJob>echo Param3= String param
Param3= String param

C:\Program Files (x86)\Jenkins\workspace\ParametrizedJob>exit 0
Finished: SUCCESS
```

Parametrized Job

- Also, these parameters can be used to create an IF statement.



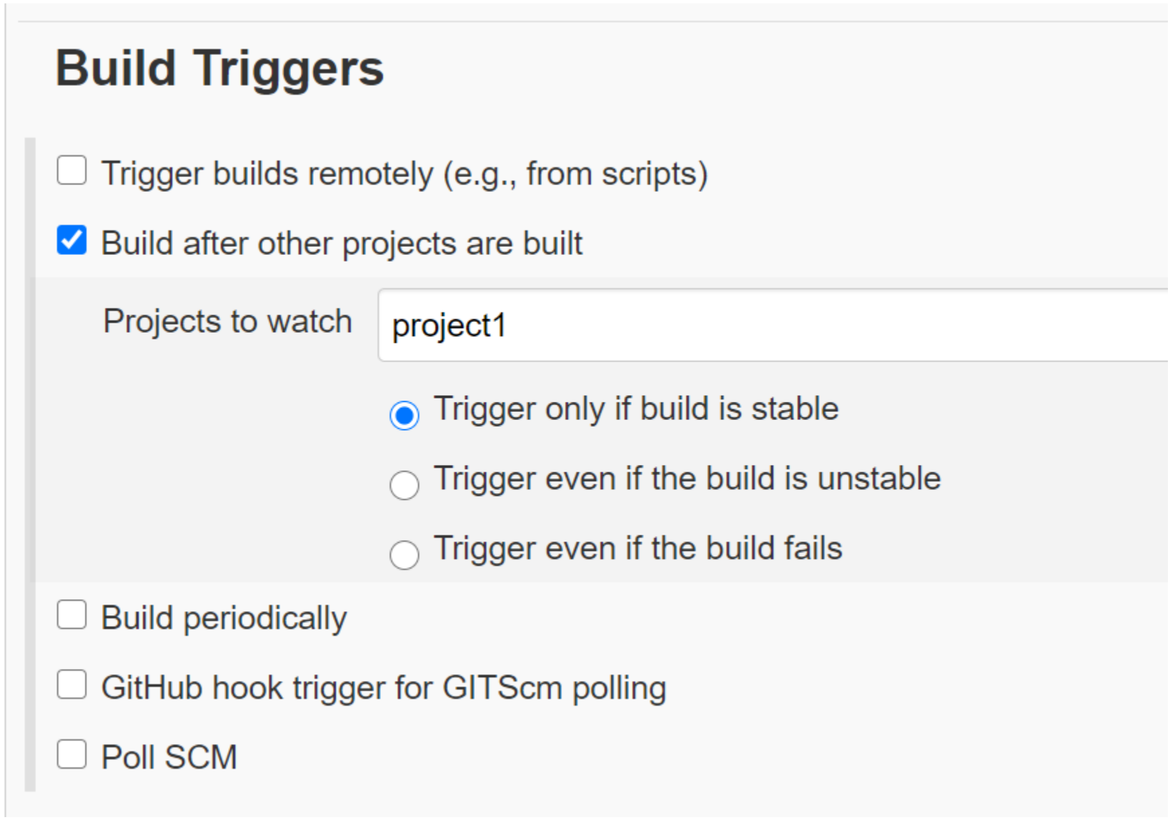
Parametrized Job

Triggers

Triggers

- On-demand run
- Other job finished
- Cron schedule
- SCM check

Triggers


Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Projects to watch
project1
Trigger only if build is stable
Trigger even if the build is unstable
Trigger even if the build fails
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM

Cron syntax

Cron Syntax

- To set the interval in the *Schedule* field you must use five fields for each line that extends through the tab or space :

MINUTE HOUR DOM MONTH DOW

  - MINUTE – the minute from the hour: 0-59;
  - HOUR – the hour of the day: 0-23;
  - DOM – Day Of Month: 1-31;
  - MONTH – Month of the year: 1-12;
  - DOW – Day Of Week: 0-7.

Cron Syntax

- You can use special operator such as:
  - * - for all possible values;
  - M-N – specify the limits of the interval;
  - A,B,..Z – specify multiple values for that field;
  - H – specify one value from the valid interval.
- On Help of the Jenkins you can see different examples.
- Or you can use: https://crontab.guru/

Unit testing in Jenkins

Job to run Unit Tests

- We consider a Java class Matematica.java with four methods:
  - suma;
  - raport;
  - estePar;
  - nNumerePare.

MatematicaTests.java | Matematica.java

```java
import java.util.ArrayList;
import java.util.List;

public class Matematica {

    public int suma(int param1, int param2) {
        return param1 + param2;
    }

    public double raport(int numarator, int numitor) {
        if(numitor==0){
            throw new IllegalArgumentException();
        }
        return (double)numarator / numitor;
    }

    public boolean estePar(int numar) {
        return numar % 2 == 0;
    }

    public List<Integer> nNumerePare(int n) {
        if(n==0) {
            return null;
        }
        if(n<0){
            throw new IllegalArgumentException();
        }
        List<Integer> lista = new ArrayList<Integer>();
        for (int i = 0; i < n; i++)
            lista.add(i * 2);
        return lista;
    }
}

```

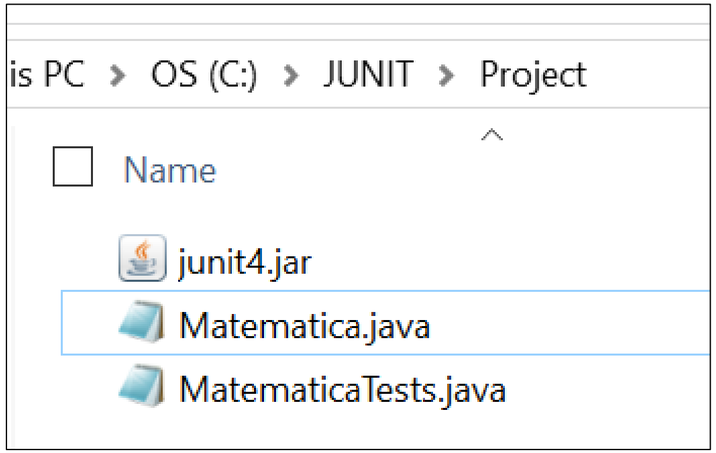
Job to run Unit Tests

- For this class we create a Junit TestCase: MatematicaTests.java
- In this TestCase we have the test for our methods from Matematica class.
- Our propose is to run all these tests.

MatematicaTests.java

```java
import static org.junit.Assert.*;
import java.util.List;
import org.junit.Before;
import org.junit.Test;

public class MatematicaTests {
    Matematica mate;

    @Before
    public void setUp(){

    @Test
    public void testSumaCorectitudine() {
        int rezultat=mate.suma(3, 12);
        int rezultatAsteptat=15;
        assertEquals(rezultatAsteptat, rezultat);
    }

    @Test
    public void testRaportCorect(){

        double rezultat=mate.raport(12, 3);
        double rezultatAsteptat=4;
        assertEquals(rezultatAsteptat, rezultat,0.1)
    }

    @Test
    public void testRaportExceptie(){
        try{
            mate.raport(34, 0);
            fail("Metoda nu arunca exceptie");
        }catch(IllegalArgumentException er){

        }
    }
```

Job to run Unit Tests



We have these files in a project directory. We have putted the junit library also in this directory

To run these test in the command prompt we have these commands:

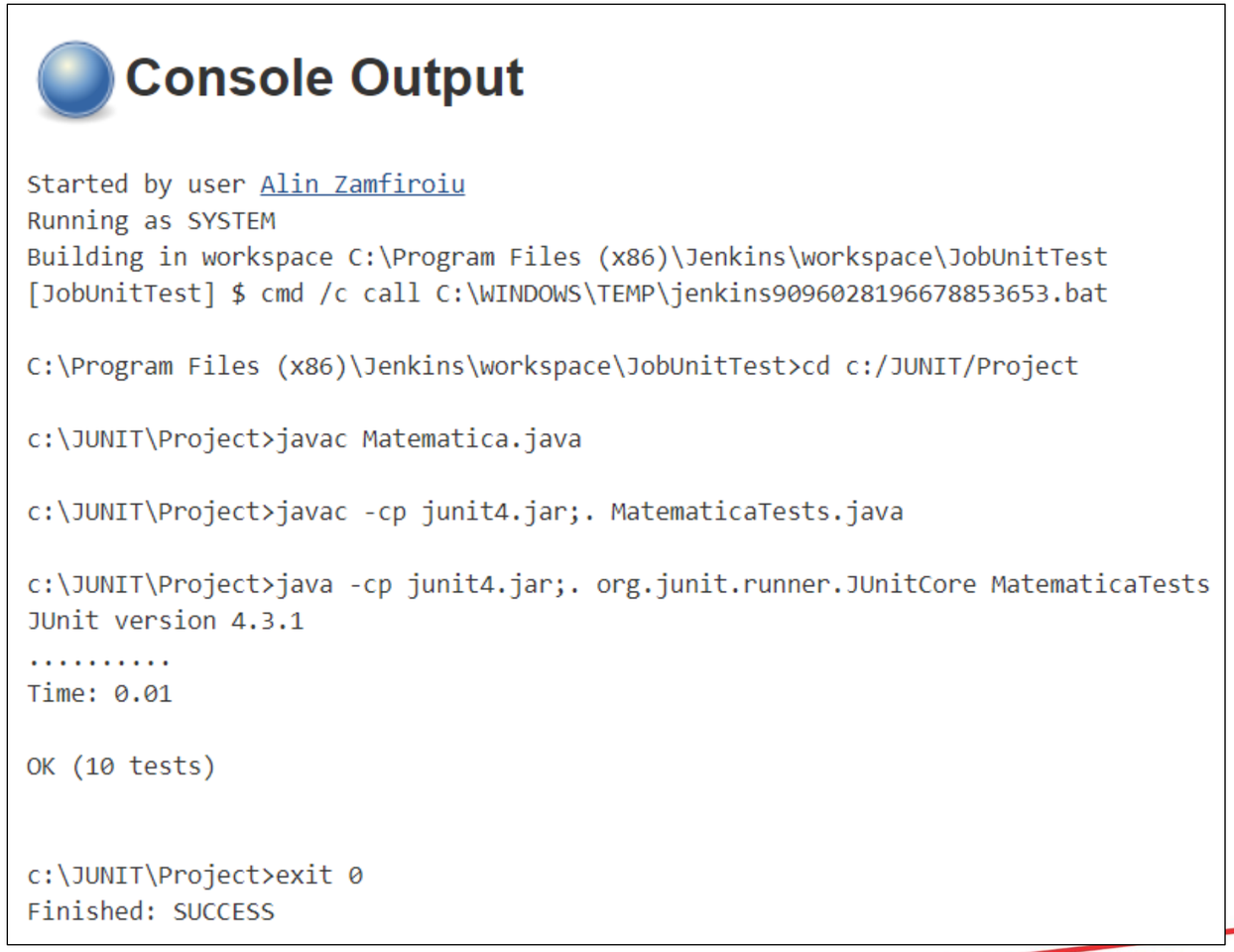
```
C:\Users\zamfiroiu>cd c:/JUNIT/Project

c:\JUNIT\Project>javac Matematica.java

c:\JUNIT\Project>javac -cp junit4.jar;. MatematicaTests.java

c:\JUNIT\Project>java -cp junit4.jar;. org.junit.runner.JUnitCore MatematicaTests
JUnit version 4.3.1
..........
Time: 0.008

OK (10 tests)


c:\JUNIT\Project>
```

Job to run Unit Tests

- To run these tests from Jenkins, we have to create a Job with these commands.

Job to run Unit Tests

- The results show us that we have run 10 tests with success.



Console Output

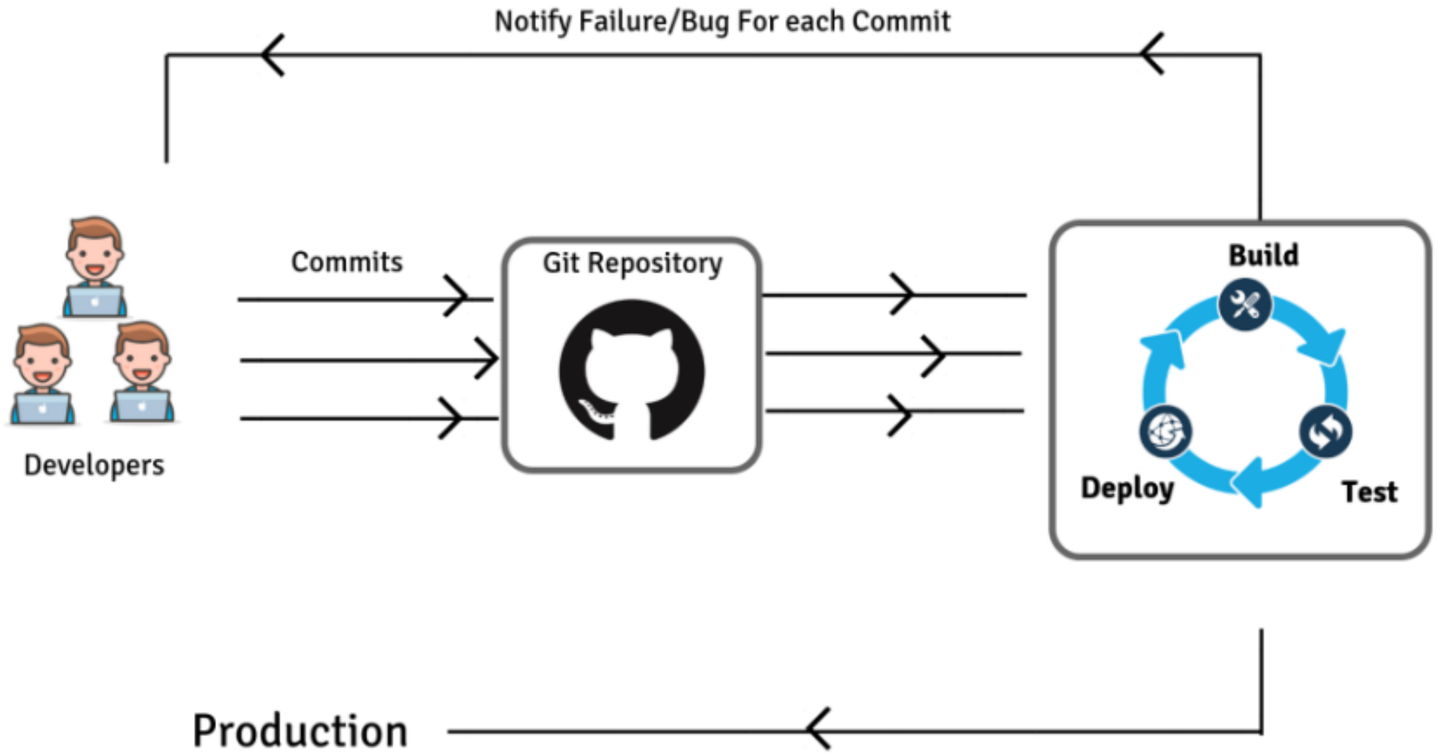
```
Started by user Alin Zamfiroiu
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\JobUnitTest
[JobUnitTest] $ cmd /c call C:\WINDOWS\TEMP\jenkins9096028196678853653.bat

C:\Program Files (x86)\Jenkins\workspace\JobUnitTest>cd c:/JUNIT/Project

c:\JUNIT\Project>javac Matematica.java

c:\JUNIT\Project>javac -cp junit4.jar;. MatematicaTests.java

c:\JUNIT\Project>java -cp junit4.jar;. org.junit.runner.JUnitCore MatematicaTests
JUnit version 4.3.1
..........
Time: 0.01

OK (10 tests)


c:\JUNIT\Project>exit 0
Finished: SUCCESS
```

Job to run Unit Tests

- Observations:
    - To run the test, first you have to compile the java files. In this way you create the class files;
    - You can run more TestCases or…
    - You can create suites and run these suites.

Jenkins with GitHub

Jenkins with GitHUB

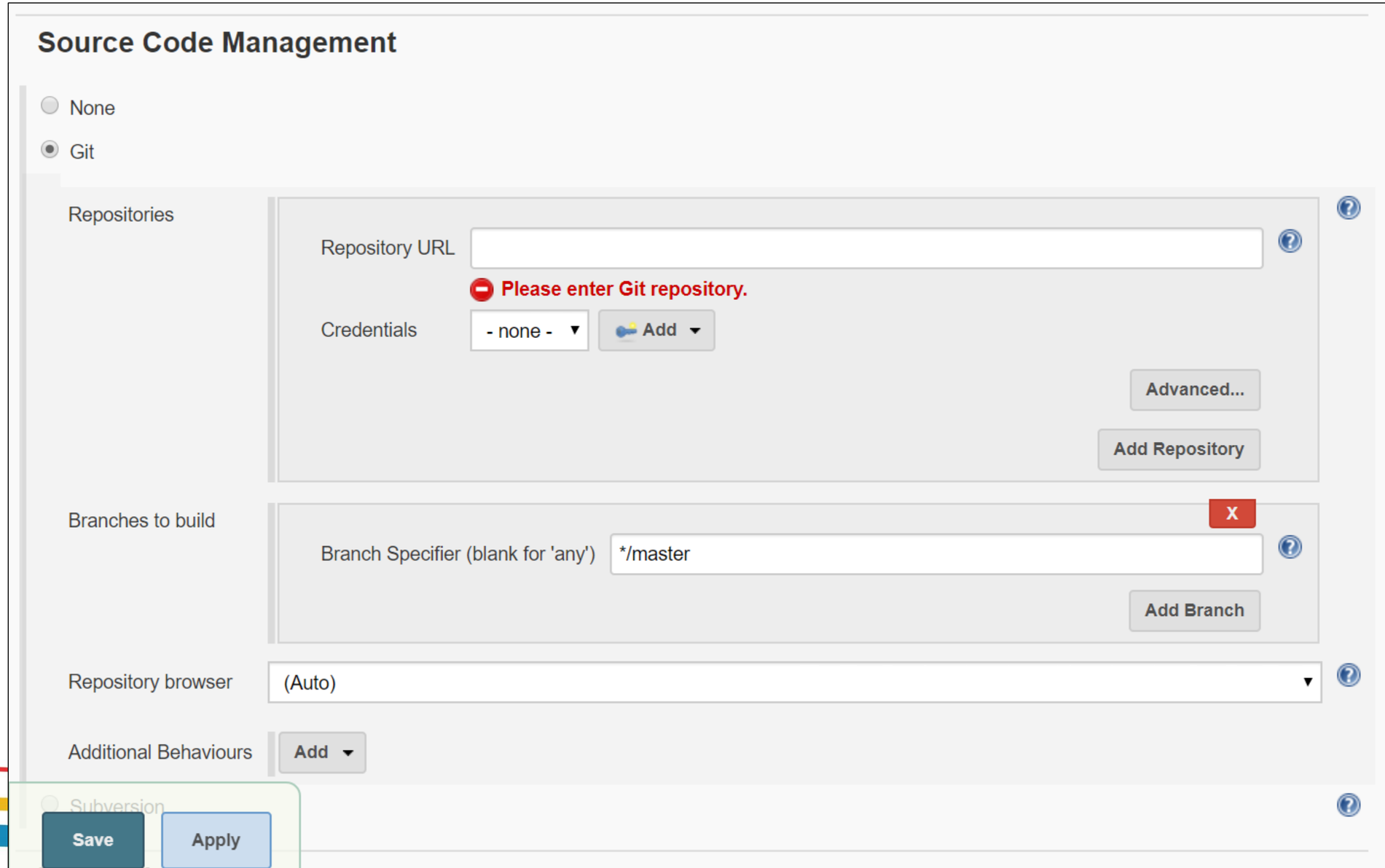


Jenkins with GitHUB

- If we want to test the code from the GitHub we need a repository.
- For this demo we create a repository for test: **TestJenkins**.
- In this repository we have these files: **Matematica.java** and **MatematicaTests.java**

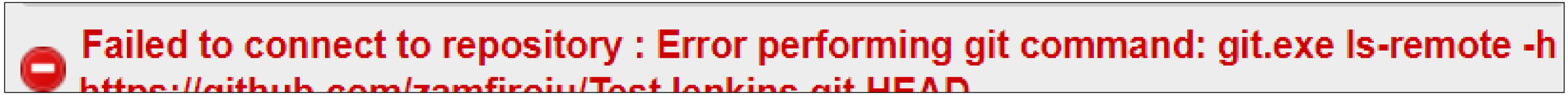
Jenkins with GitHUB

- Under **Source Code Management Tab** we have to select **GitHub**.



Jenkins with GitHUB

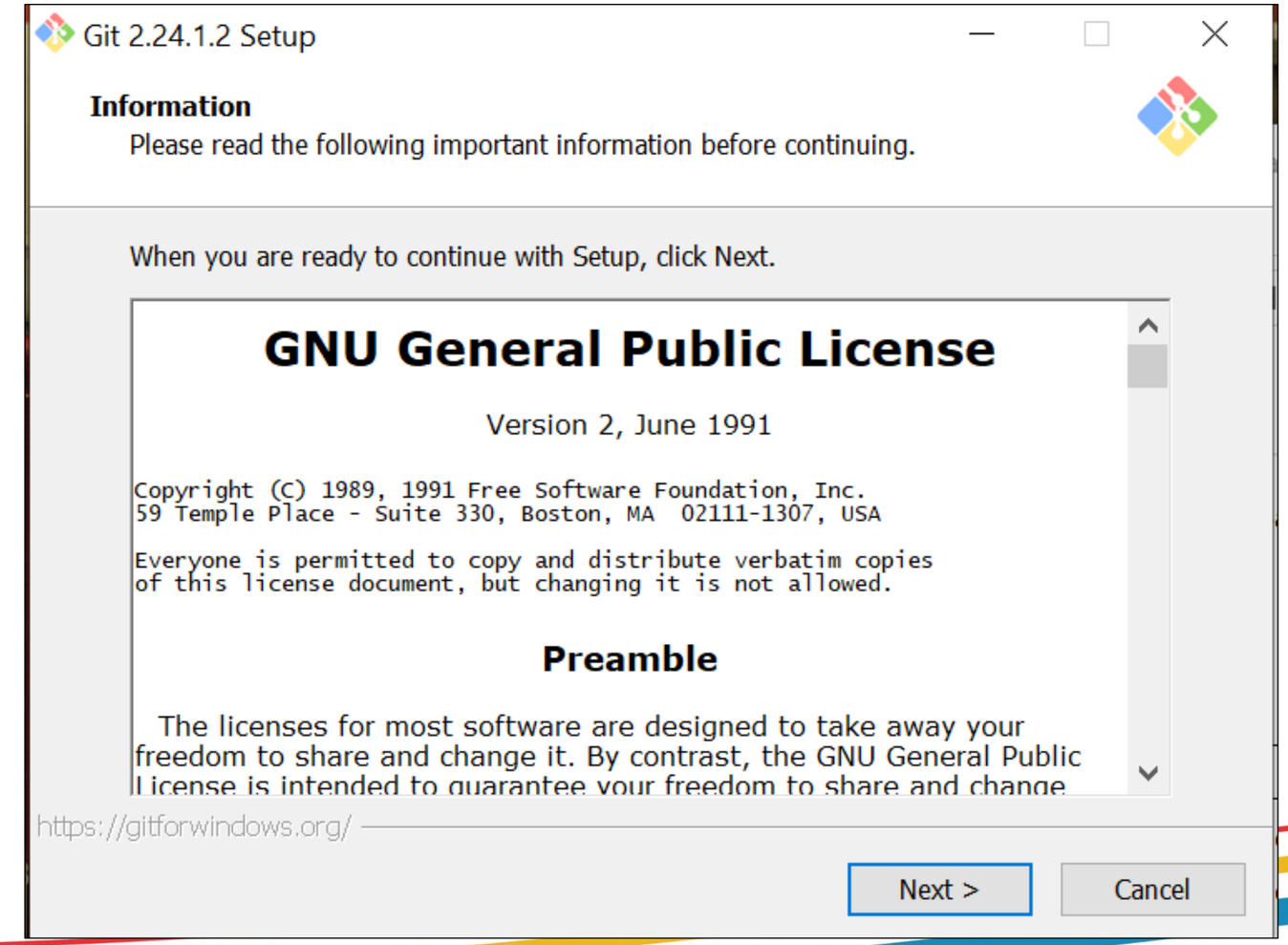
- It is possible when you put your repository to have an error like this one from the image.



- That error is because you have not installed the Git on your machine.

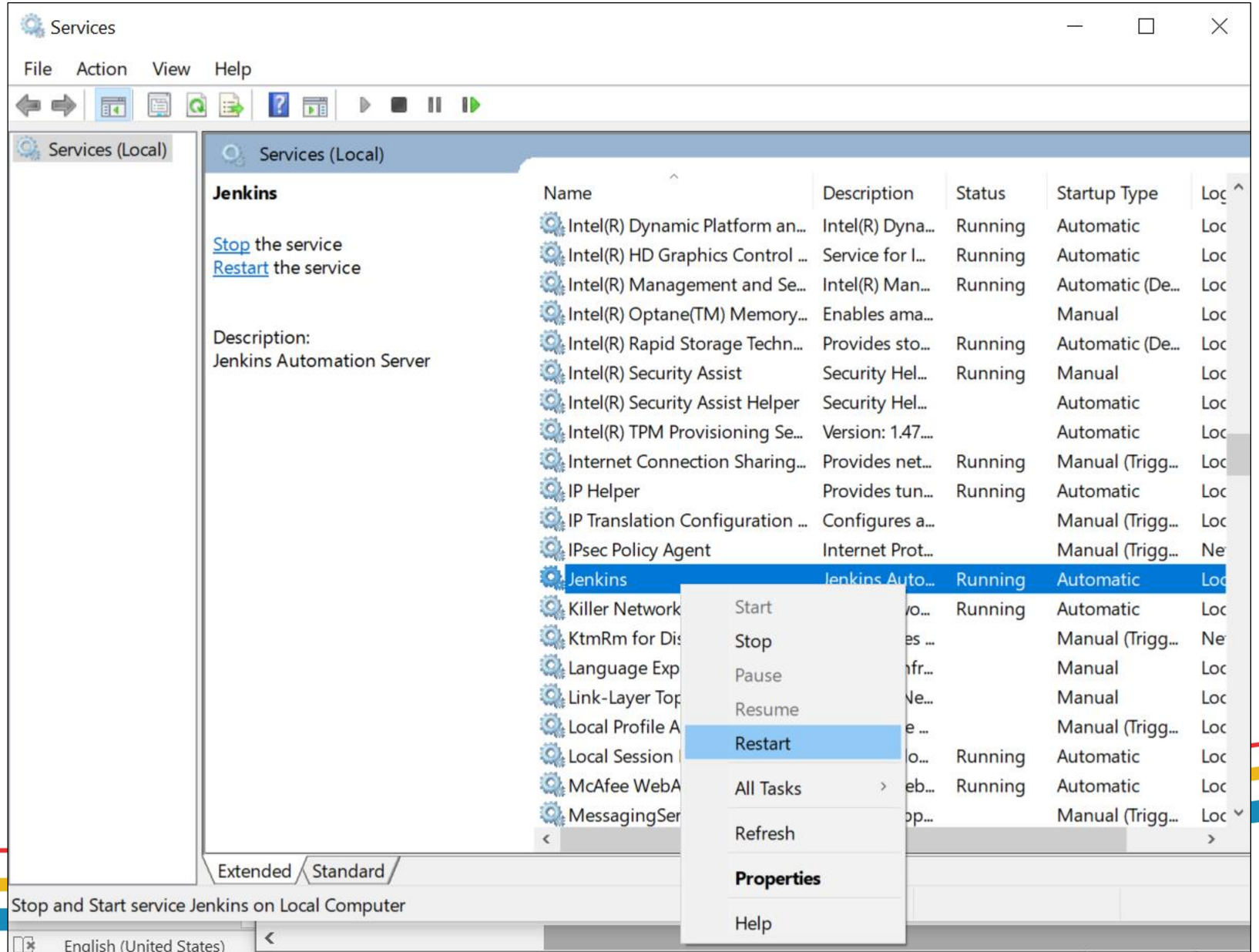
Jenkins with GitHUB

- You have first to install it and after to continue.



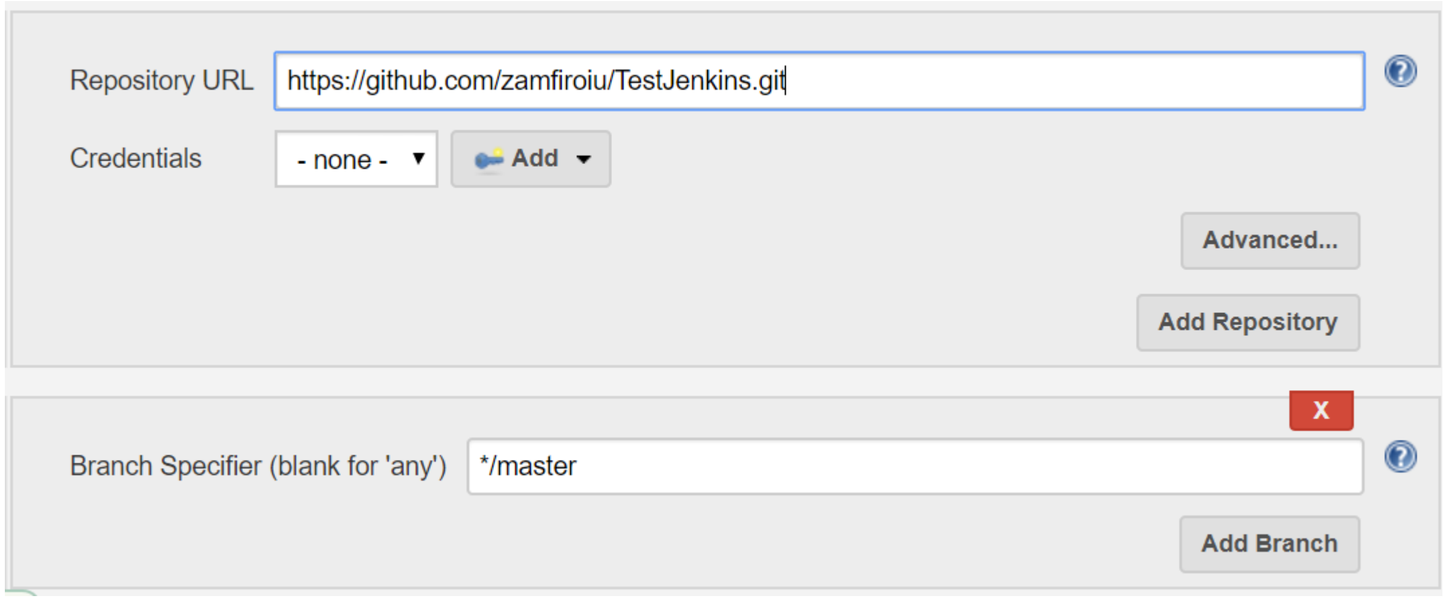
Jenkins with GitHUB

- If the error persists you have to restart the Jenkins from the services



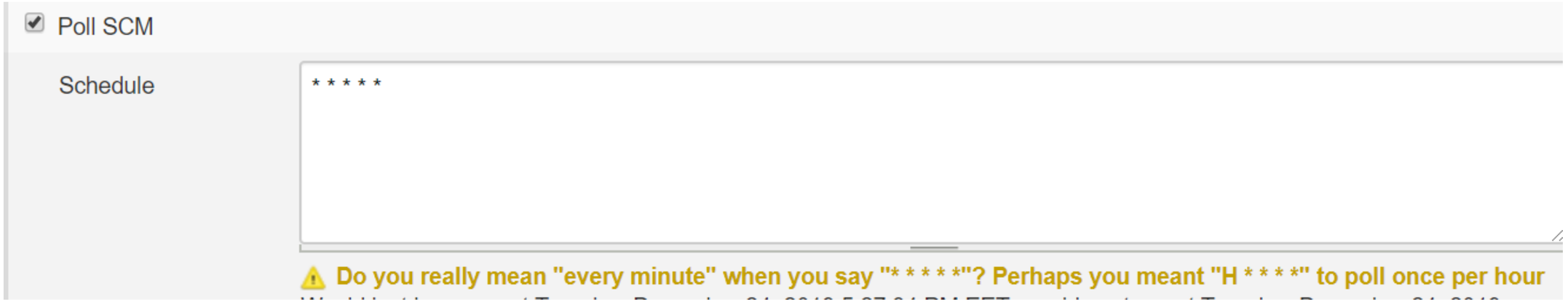
Jenkins with GitHUB

- Now everything should be ok!



Jenkins with GitHUB

- In the Build Triggers TAB we have to select Poll SCM and to define the interval when Jenkins should check the repository for commits.



- In this way Jenkins checks Git repository once every minute for changes and triggers Build if any changes made in the Git repository

Jenkins with GitHUB

- Now Jenkins will check the GitHub repository on every minute if there are any updates.

- When a commit is made on the repository, Jenkins will pull the code, will compile it and run the tests.

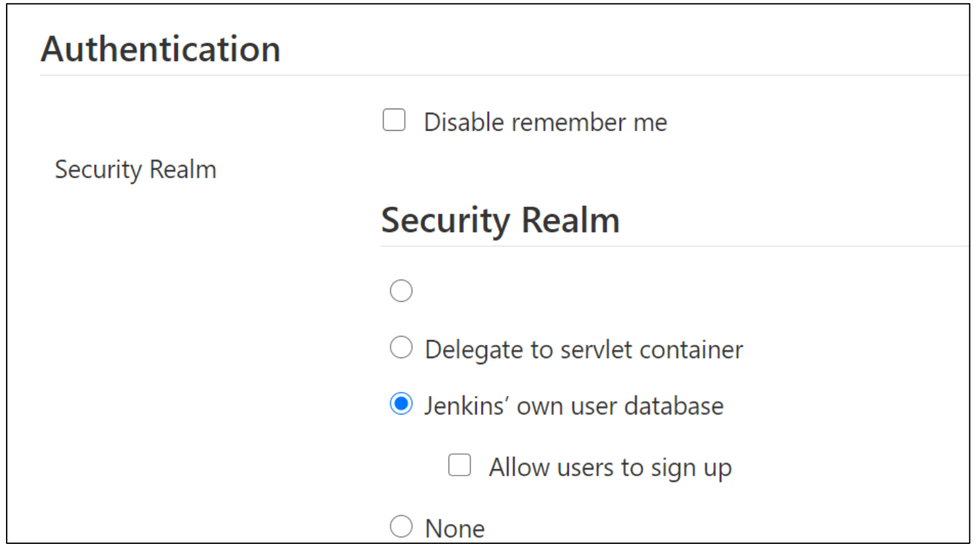
Jenkins with GitHUB

```
[TestJenkins] $ cmd /c call C:\WINDOWS\TEMP\jenkins1378790904753698818.bat

C:\Program Files (x86)\Jenkins\workspace\TestJenkins>javac Matematica.java

C:\Program Files (x86)\Jenkins\workspace\TestJenkins>javac -cp C:/JUNIT/junit4.jar;. MatematicaTests.java

C:\Program Files (x86)\Jenkins\workspace\TestJenkins>java -cp C:/JUNIT/junit4.jar;. org.junit.runner.JUnitCore MatematicaTests
JUnit version 4.3.1
..........
Time: 0.008

OK (10 tests)


C:\Program Files (x86)\Jenkins\workspace\TestJenkins>exit 0
Finished: SUCCESS
```

Jenkins security

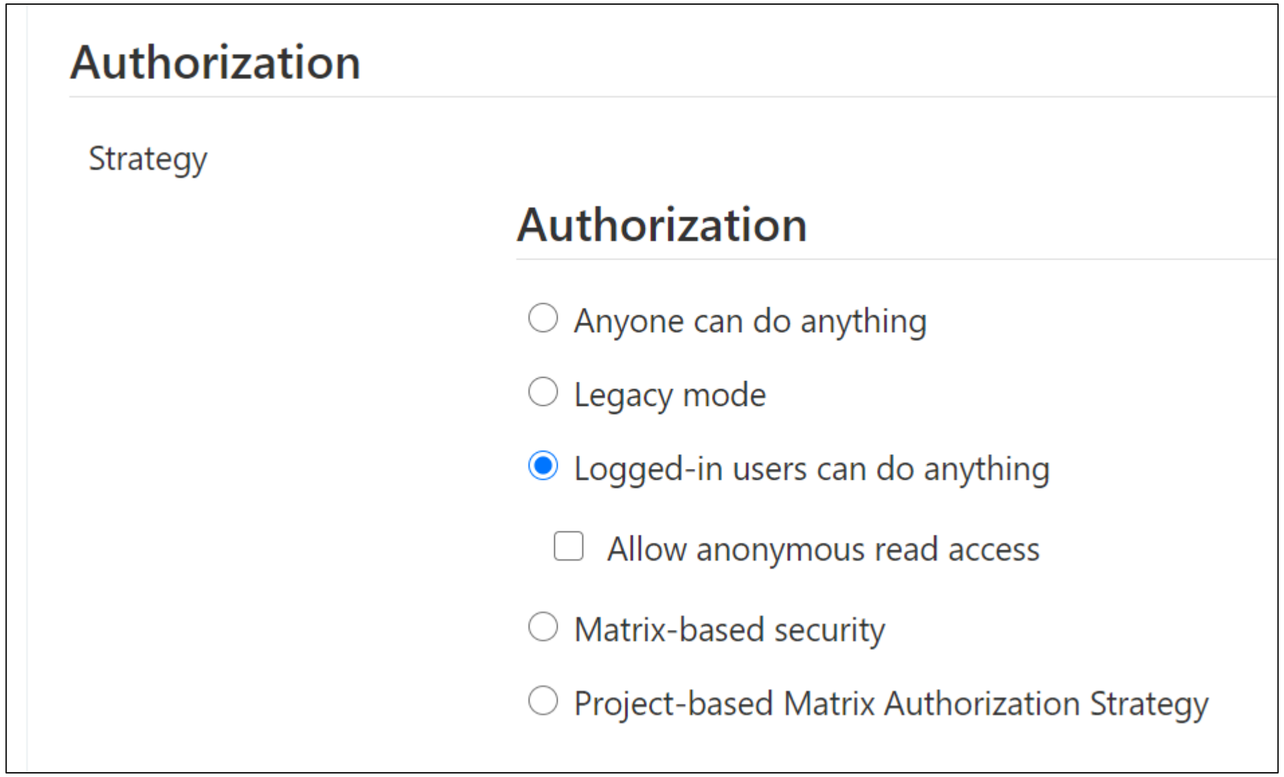
Authentication method

- We can use a LDAP
- We can have own user database
- We can allow to add new users in the database (sign up).



Authorization section

- In the authorization section we can decide for each type of user what it is allowed to do.



If we forgot the admin password

- Got to config.xml file from the Jenkins directory
- Change the value of **useSecurity** element from true to false.

```
<mode>NORMAL</mode>
<useSecurity>true</useSecurity>
<authorizationStrategy class="hu
```

- Remove the **authorizationStrategy** and **securityRealm** elements

```
<authorizationStrategy class="hudson.security.FullControlOnceLoggedInAuthorizationStrategy">
  <denyAnonymousReadAccess>true</denyAnonymousReadAccess>
</authorizationStrategy>
<securityRealm class="hudson.security.HudsonPrivateSecurityRealm">
  <disableSignup>true</disableSignup>
  <enableCaptcha>false</enableCaptcha>
</securityRealm>
```

- Restart Jenkins.

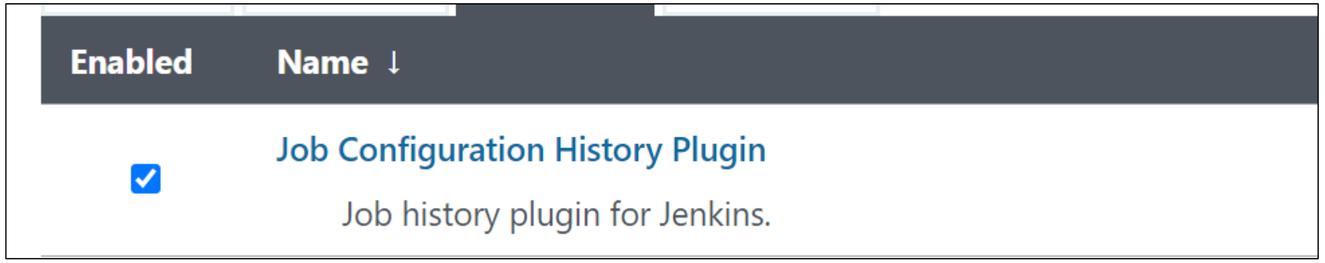
If we forgot the admin password

- After restarting the Jenkins we can enter and allow the sign up
- Create a new user.
- And after we have the user we can use again the security onJenkins.

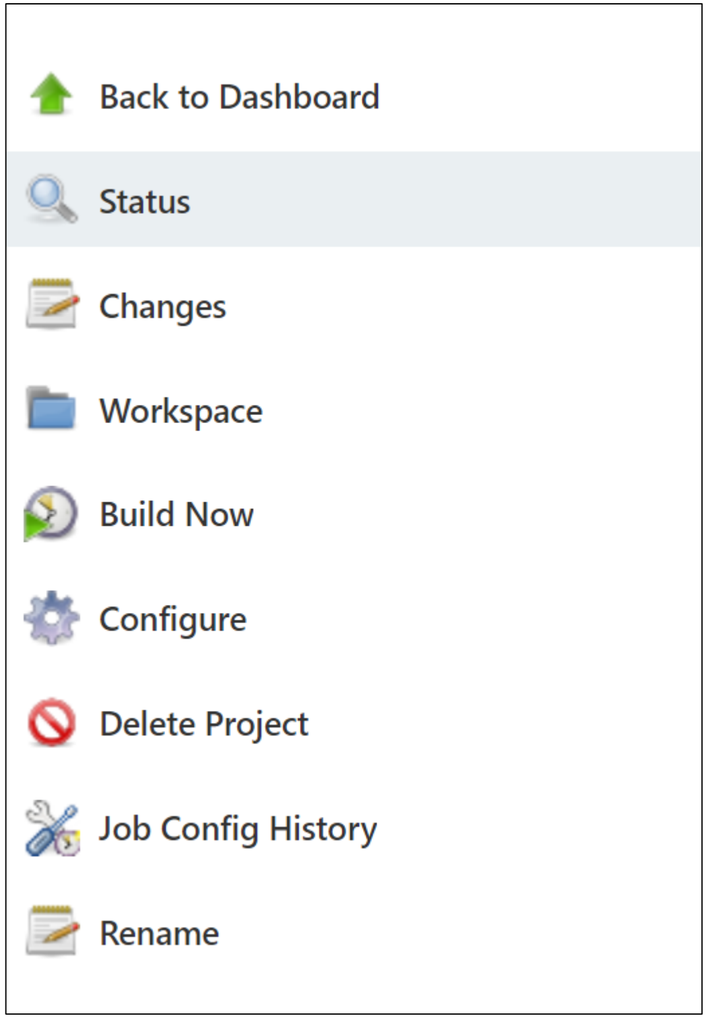
Job Configuration History

Job Configuration History

- We have to install the Job Configuration History.



- Now when someone change any job we can se in the history the changes and the owner of these changes on the “Job Config History” from that job.



Job Configuration History

project1

	Operation	User	Show File	Restore old config	File A	File B	Delete Entry
	Changed	adminzamfiroiu	View as XML (RAW)		○	◉	
	Changed	adminzamfiroiu	View as XML (RAW)	⏮	◉	○	

10 25 100 250 all

Show Diffs

- Also, here we can see the differences between these two versions and we can restore to the old version of our job

Task 1

- Create a Jenkins job that connect to a GitHub repository where you have minimum 2 tests.
- The Jenkins user should decide by a parameter which one of these tests to run. (you can have the tests in different TestCases).

- The repository name should by the format: **SQMA_SecondName_FirstName**. Example: *SQMA_Zamfiroiu_Alin*

- Create a document with every step and the result of running this job.

Task 2

- You have to create more jobs in a pipeline that will run all tests from your repository.
- Create a document with every step and the result of running this pipeline.
- Please put in different sections of this document **Task 1** and **Task 2**.