


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
- 



- 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.






Quality

- Eight dimensions of Quality:

1. Performance,
2. Features,
3. Reliability,
4. Conformance,
5. Durability,
6. Serviceability,
7. Aesthetics,
8. Perceived Quality.



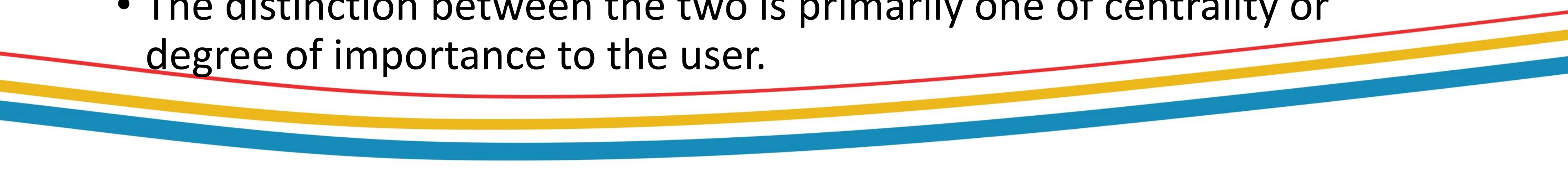
1. Performance

- 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



5. Durability

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



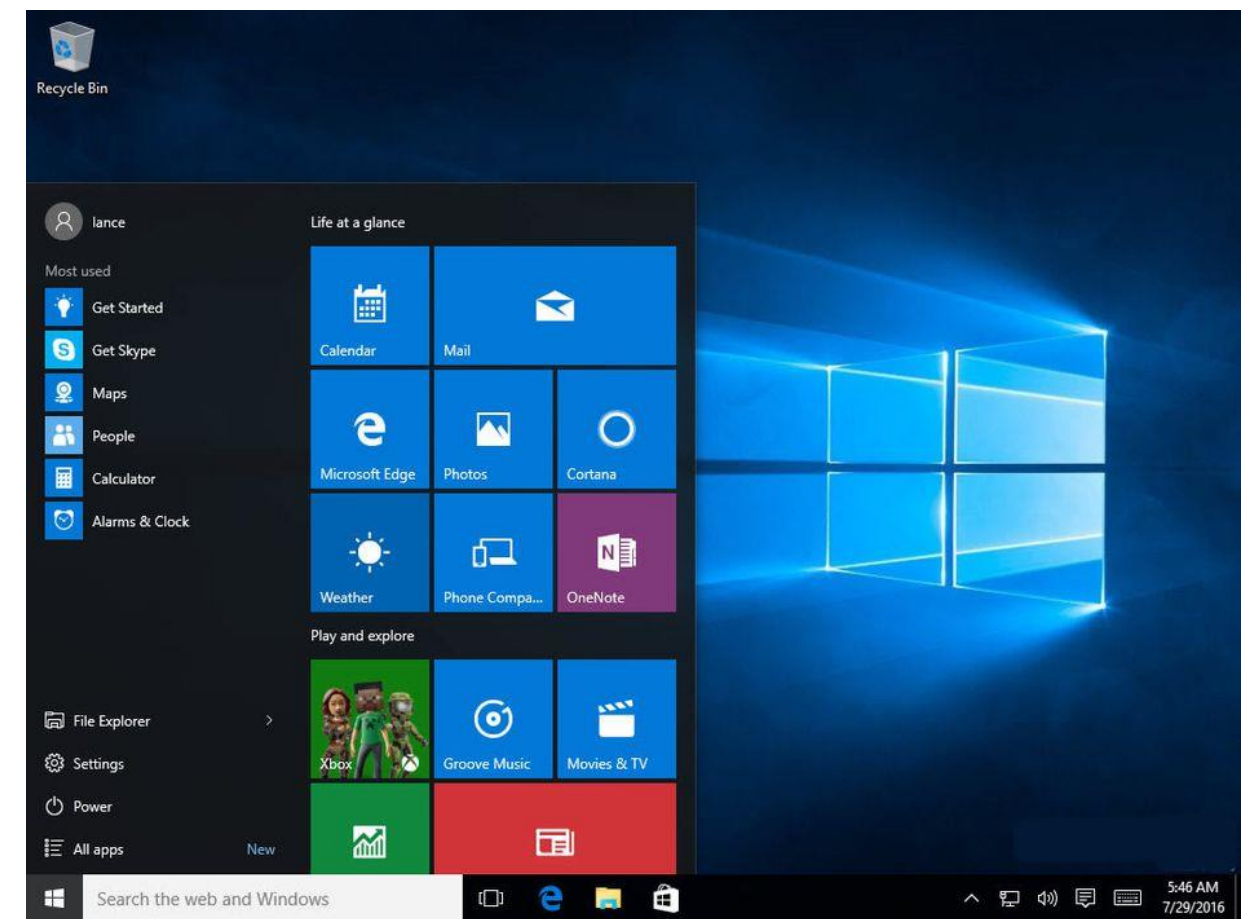
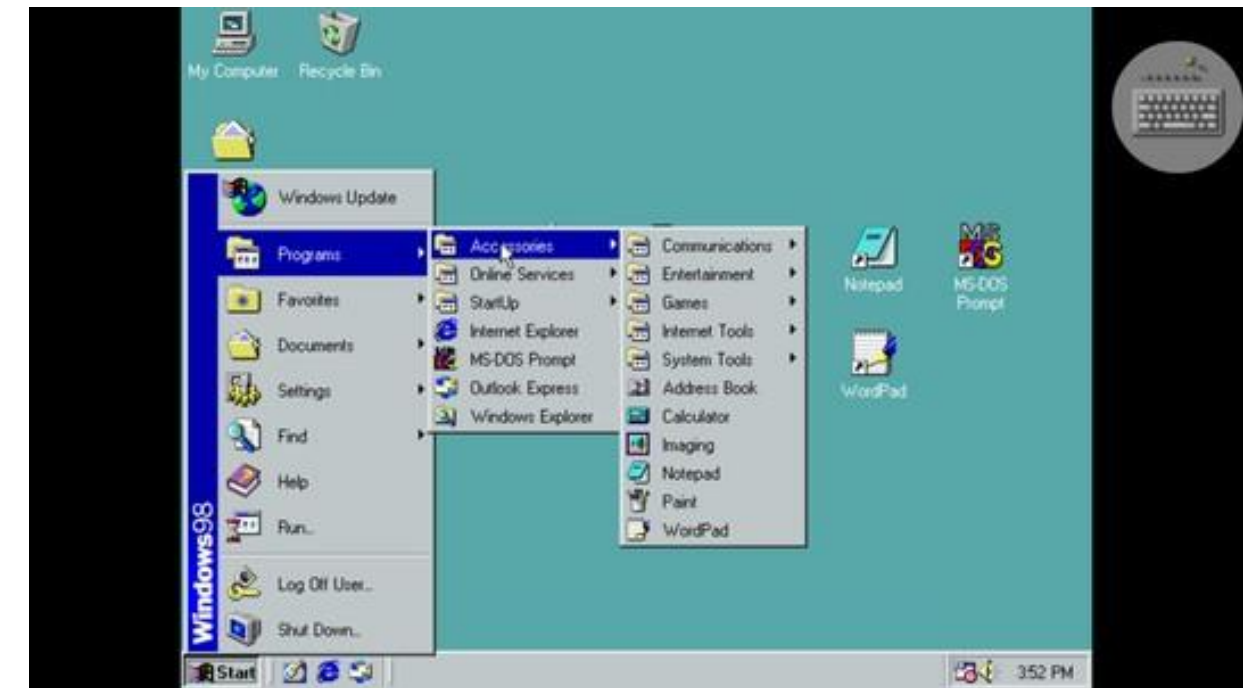
6. Serviceability

- 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.



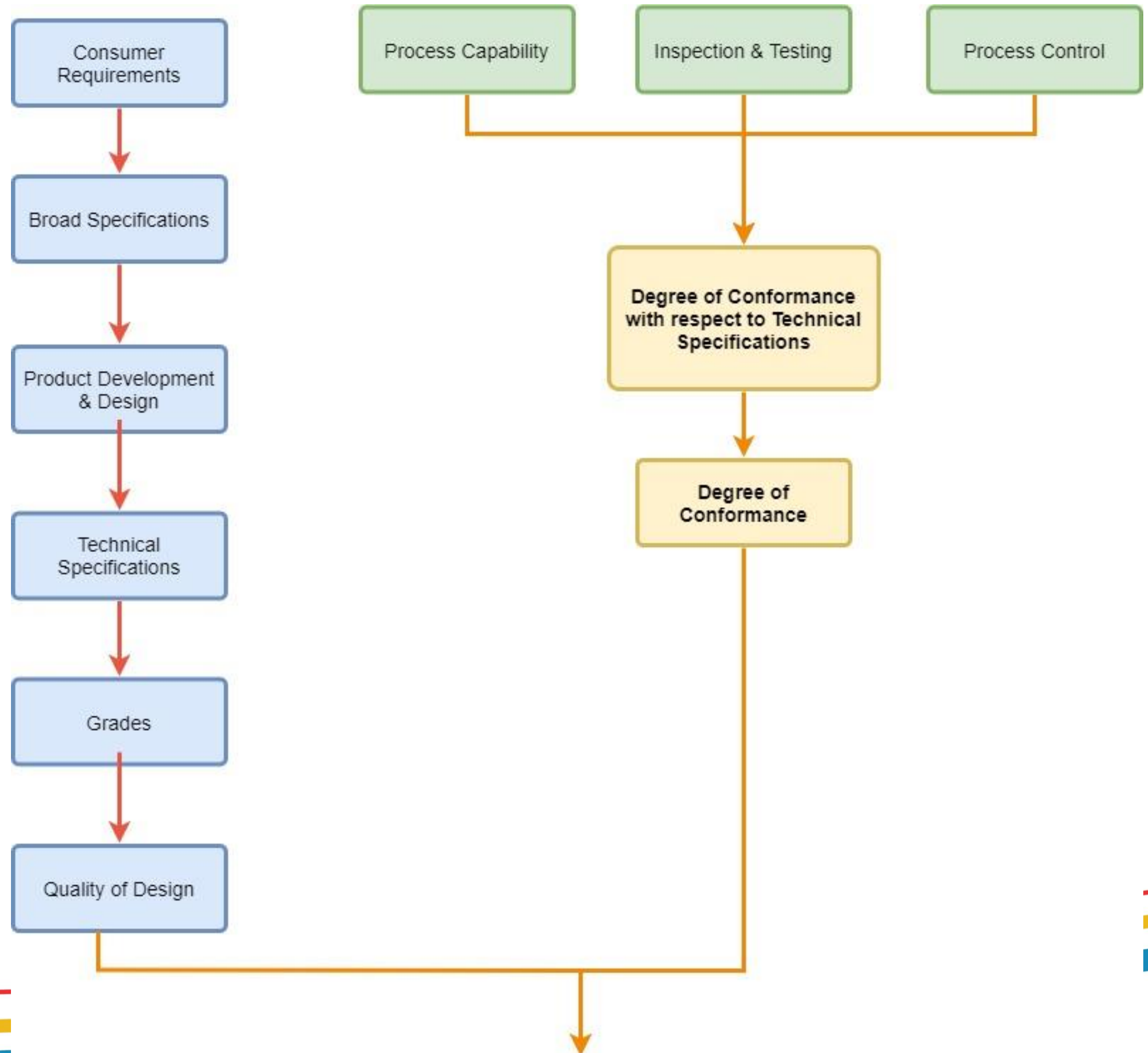
Quality assurance

- 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

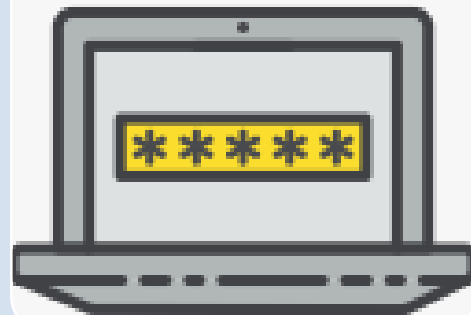
Cybersecurity management

- Cybersecurity is a management issue.
 - Usually the information security breaches are 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

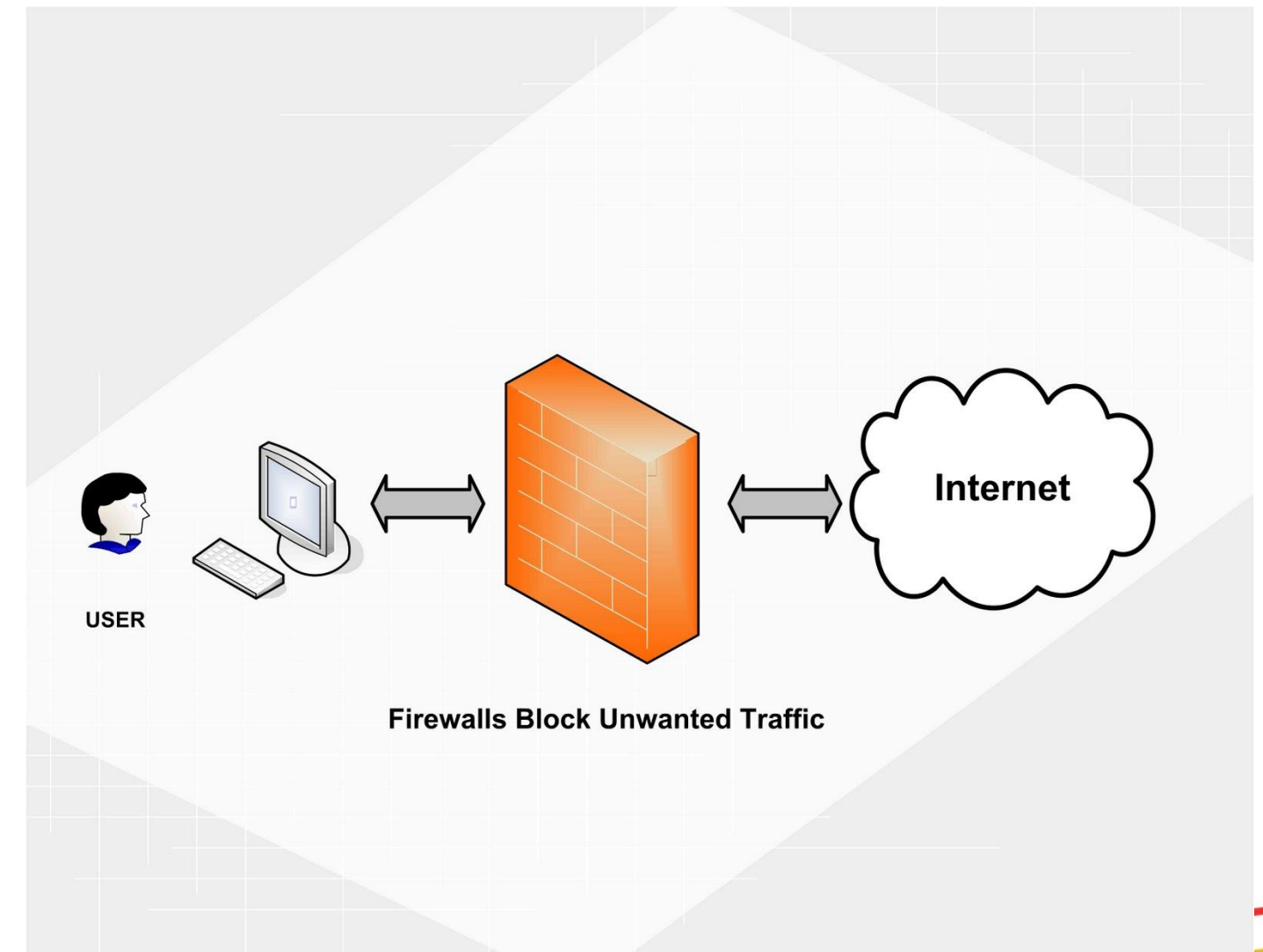


Compliance
business
framework

Principles of cybersecurity management

#1. SECURITY BEYOND FIREWALL

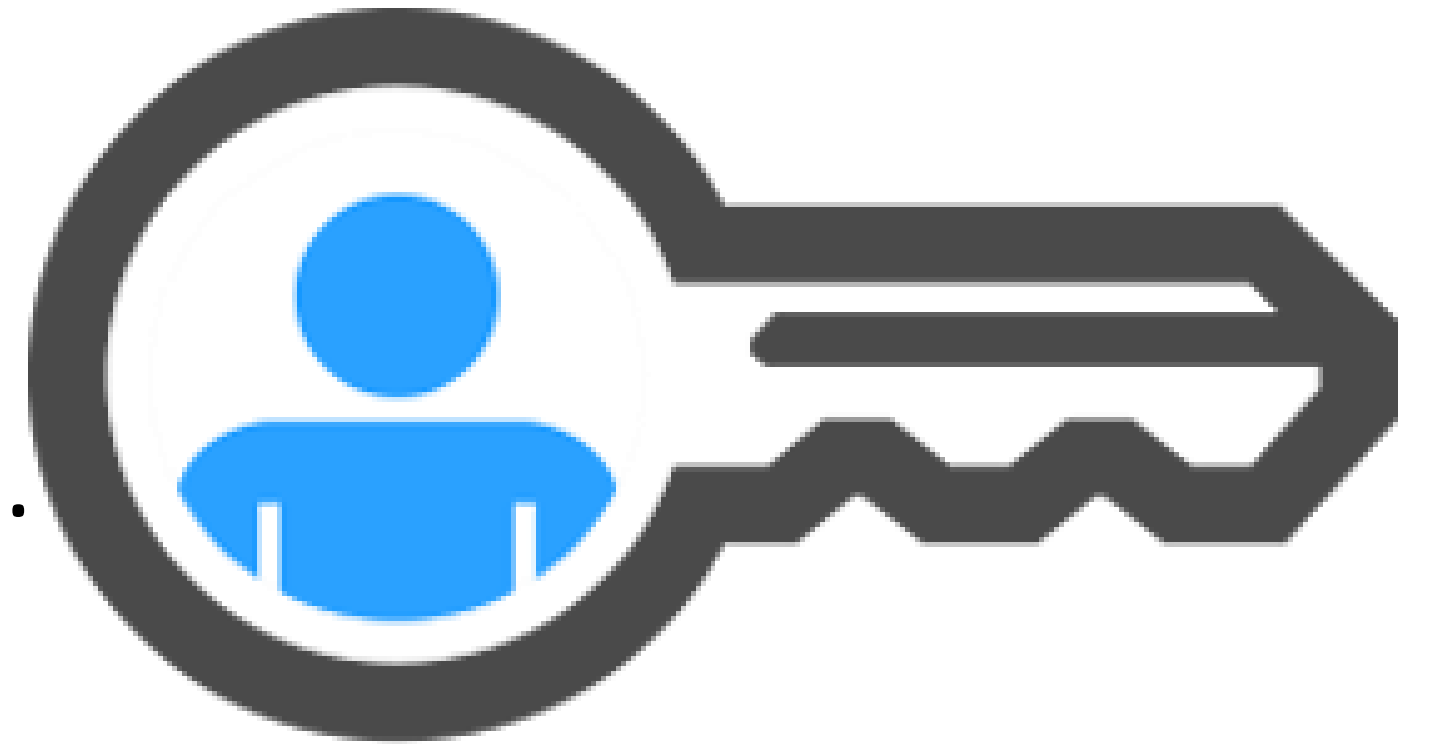
- Network security used to be achieved by scanning network traffic on various OSI layers.
- Some tools are able to look and analyze the suspicious patterns of traffic to identify and protect the system against fraud.



Principles of cybersecurity management

#2. ADVANCED ACCESS MANAGEMENT

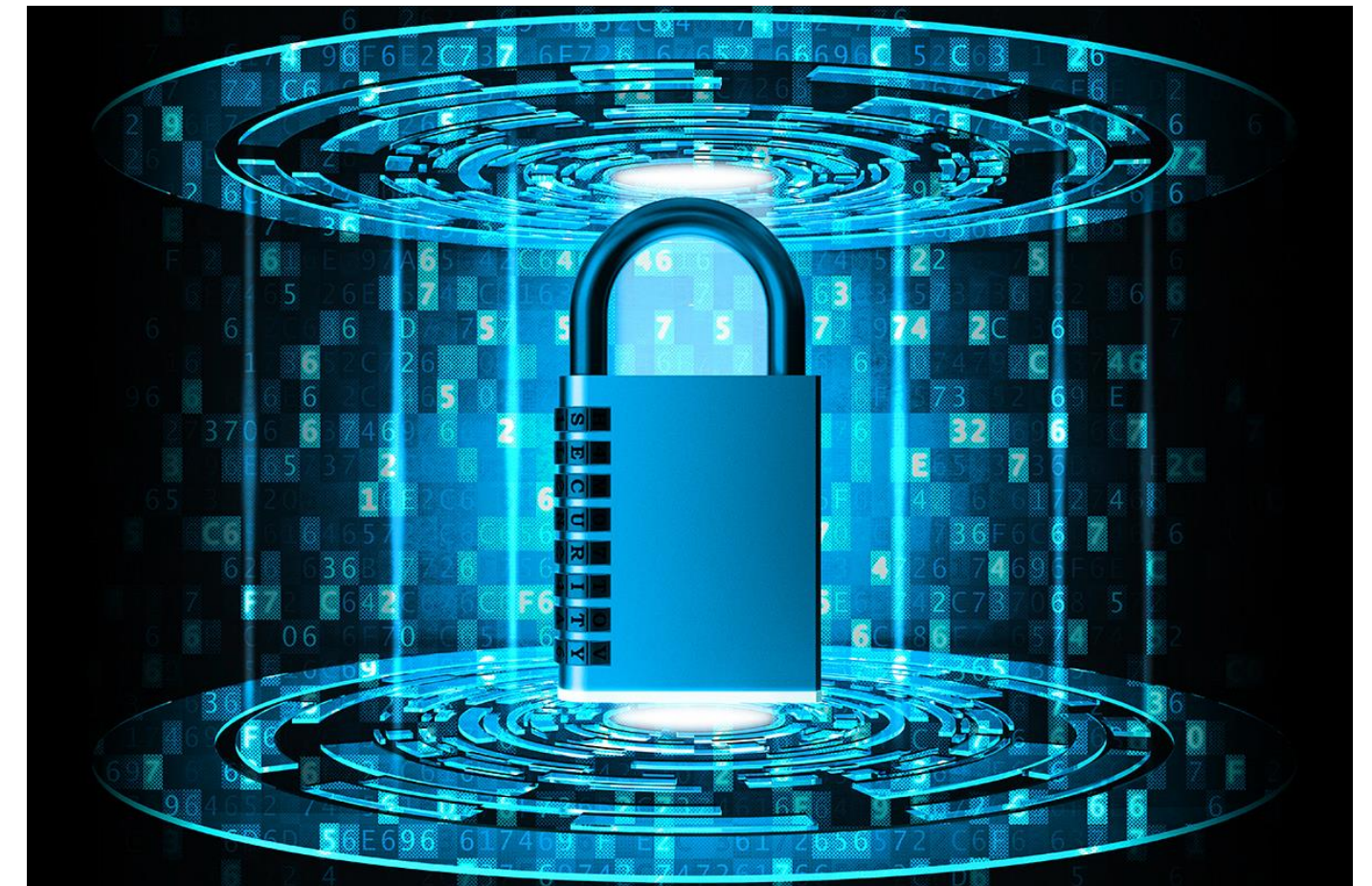
- If you still use a username and password to access your systems you should seriously consider moving to an advanced access management solution.
- In today's world, a combination of username and password is no longer secure enough.



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

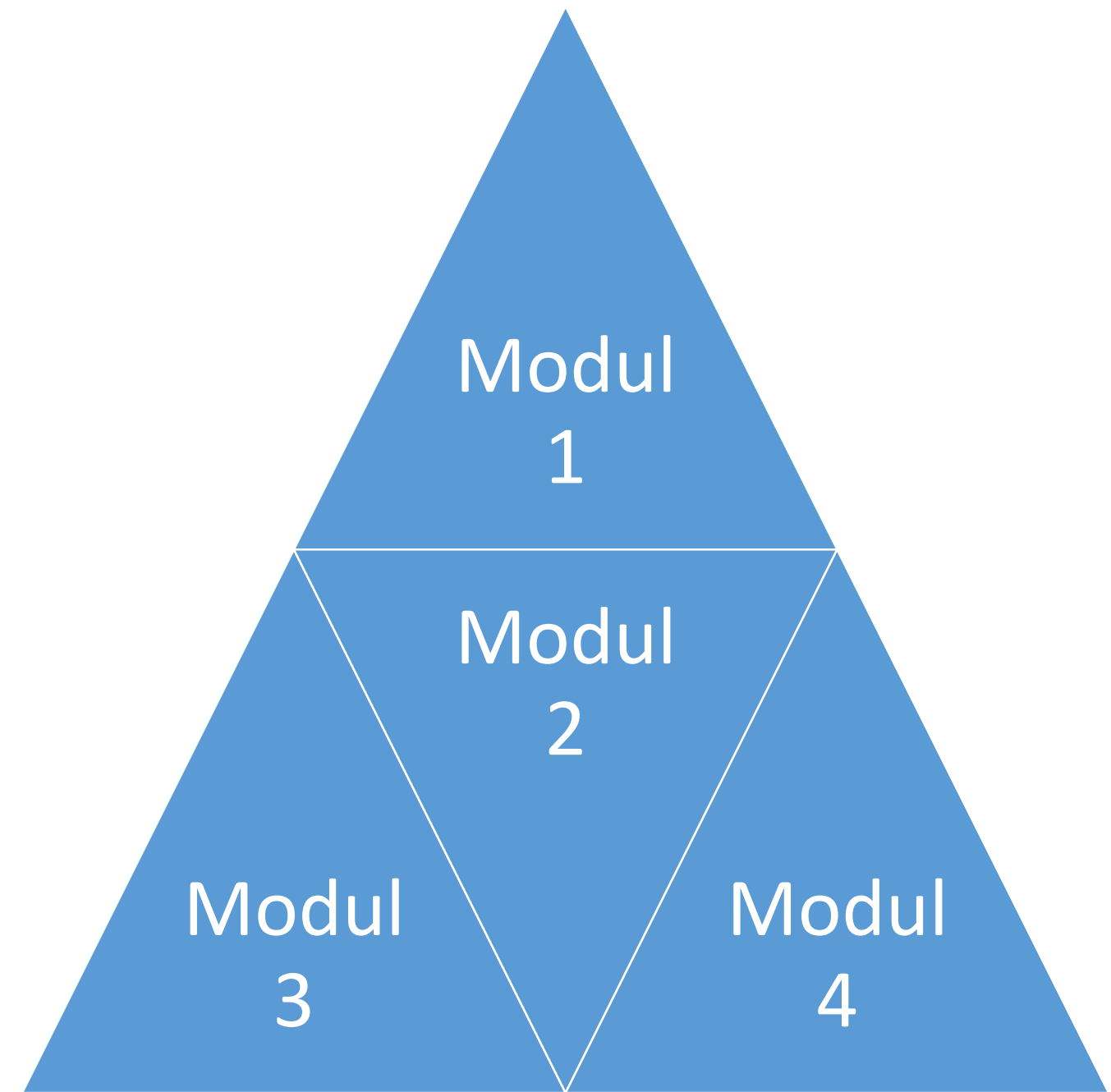
Continuous Integration

- 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.

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.



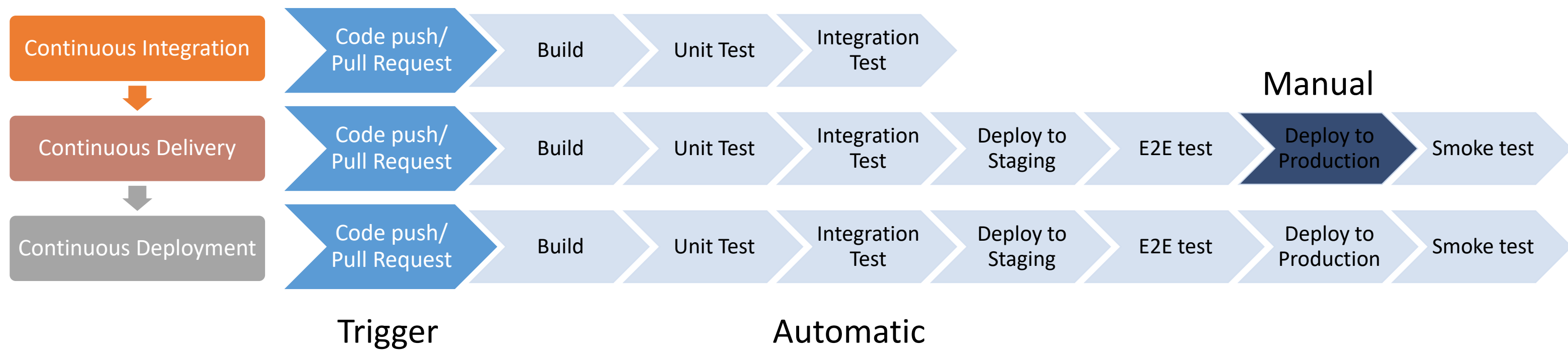
Continuous deployment

- 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



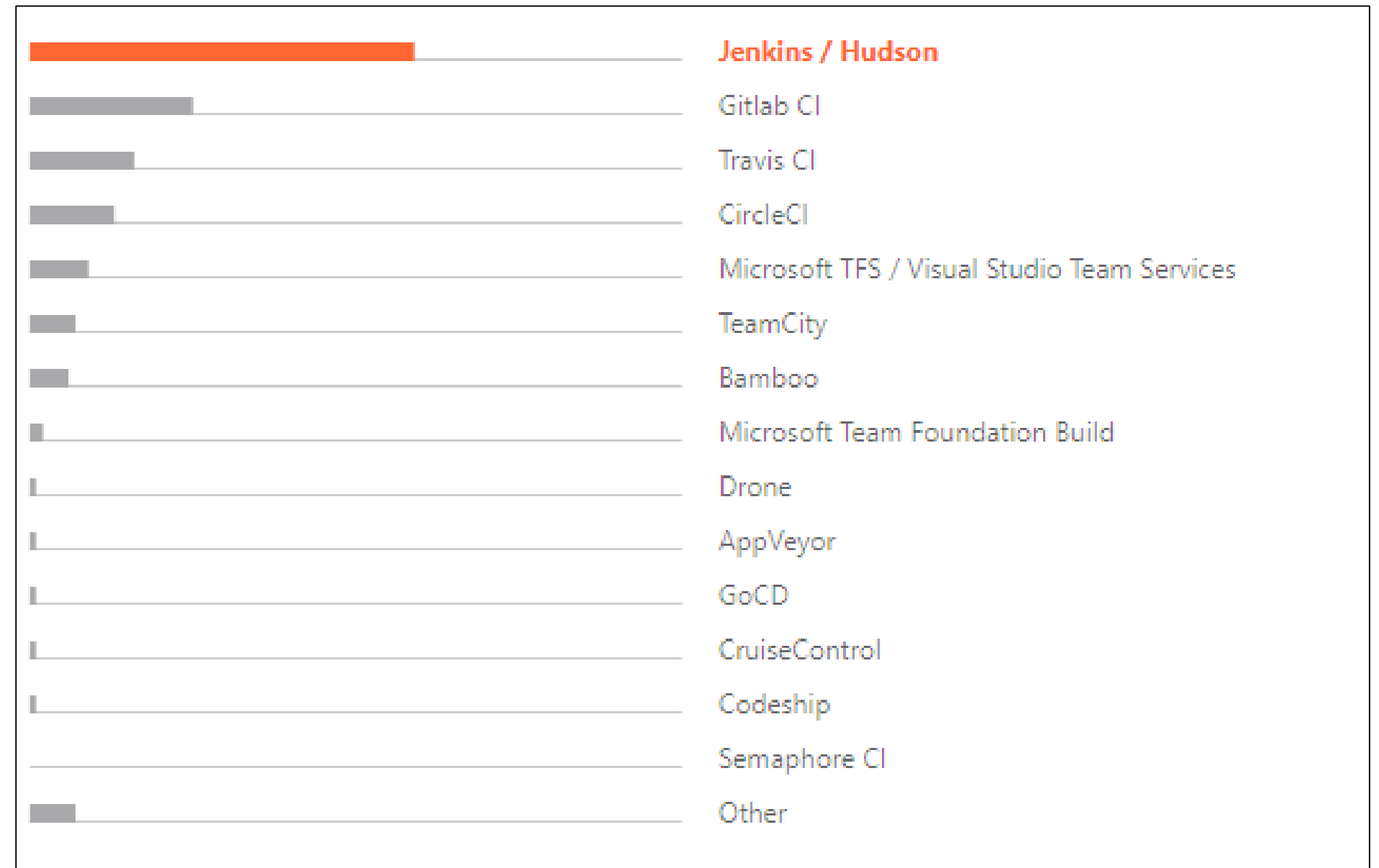
Continuous Integration systems

- 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:




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

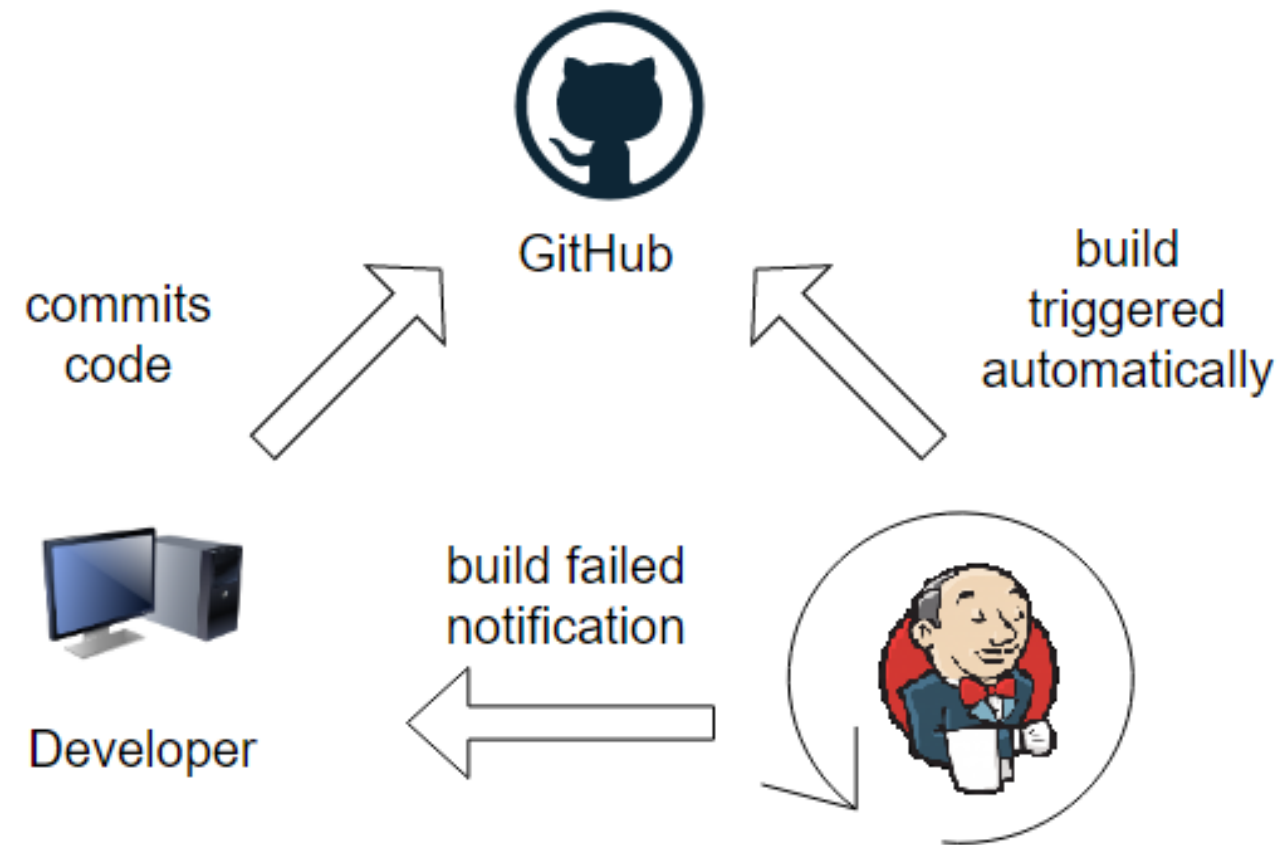


Jenkins - Content

- 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
- 

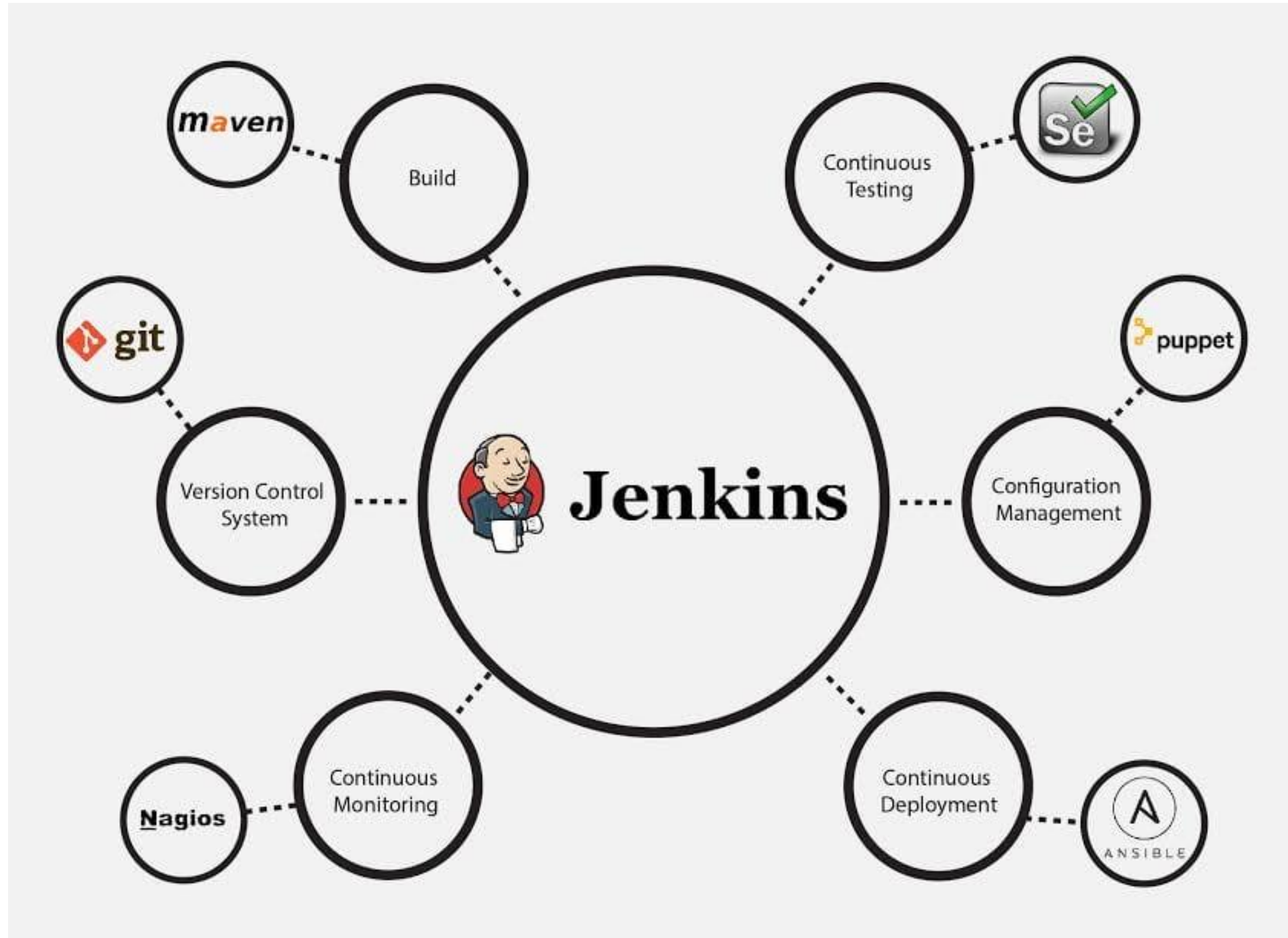
What is Jenkins?

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



What is Jenkins?


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




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.


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.

Download Jenkins 2.263.1 LTS for:

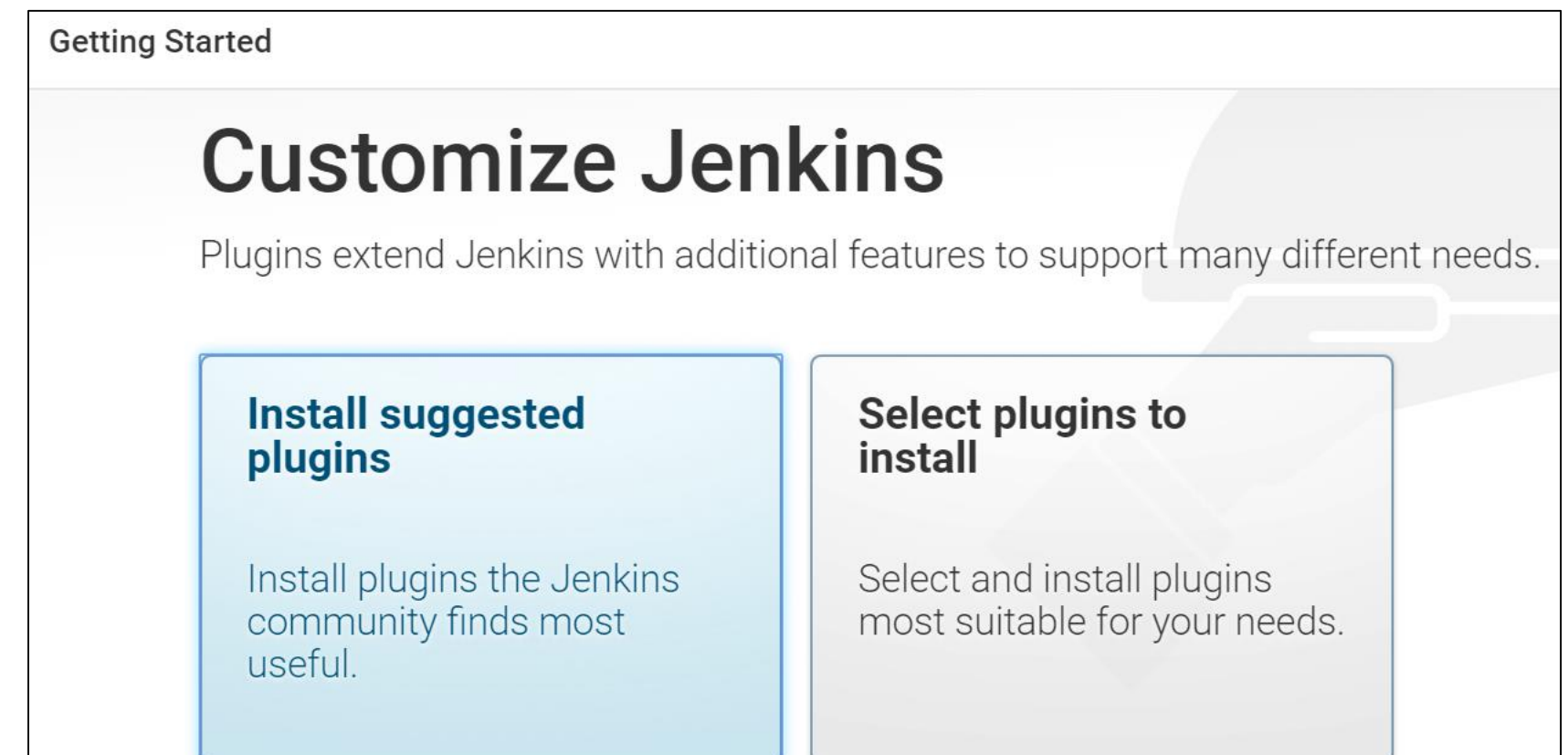
Generic Java package (.war) SHA-256: 0eedeb2b11a32726acb57db26e262b1923cf408e84708baf471e3b53462ed6f1
Docker
Ubuntu/Debian
CentOS/Fedora/Red Hat
Windows
openSUSE
FreeBSD ⚙️
Gentoo ⚙️
macOS ⚙️
OpenBSD ⚙️

Download Jenkins 2.269 for:

Generic Java package (.war) SHA-256: 3c8c584e12e50475d4312f3721bc876d005344ef072e6f1356fbed47e64ef93c
Docker
Ubuntu/Debian
CentOS/Fedora/Red Hat
Windows
openSUSE
Arch Linux ⚙️
FreeBSD ⚙️
Gentoo ⚙️
macOS ⚙️
OpenBSD ⚙️

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.

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Install and Running Jenkins

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

Unlock Jenkins

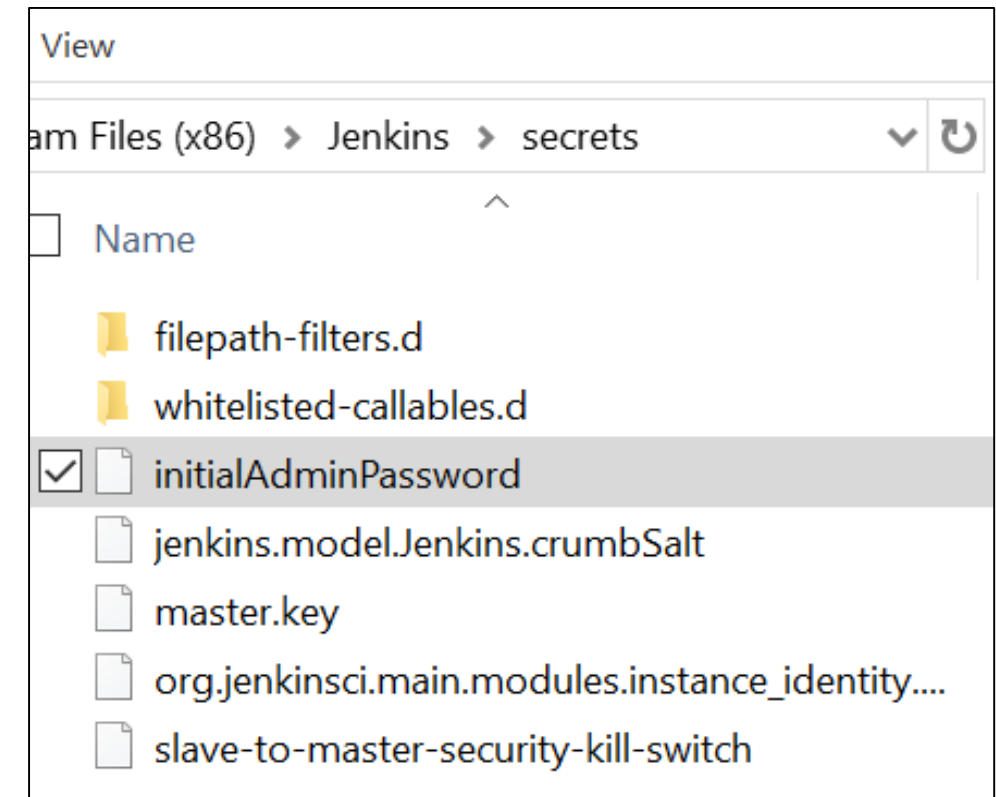
To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword`

Please copy the password from either location and paste it below.

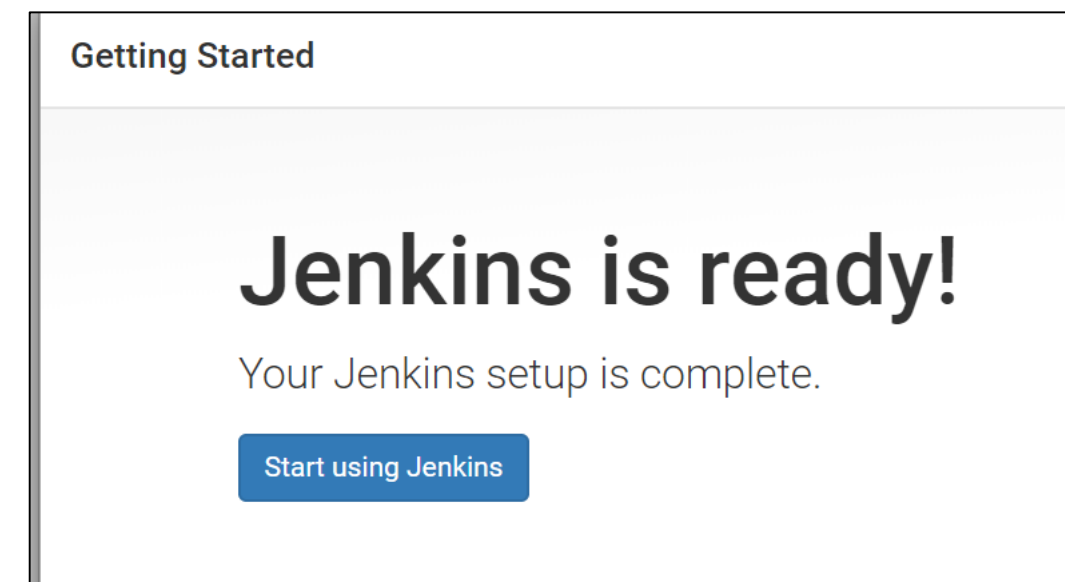
Administrator password

Continue



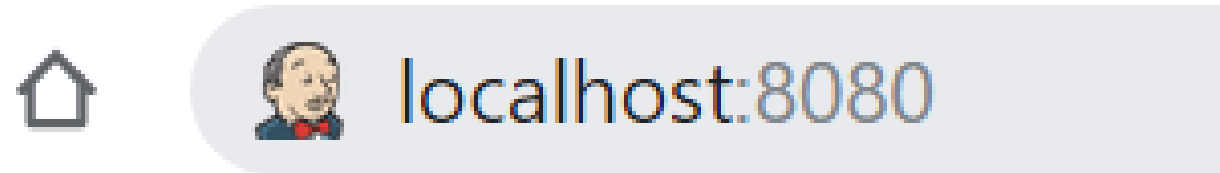
Install and Running Jenkins

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

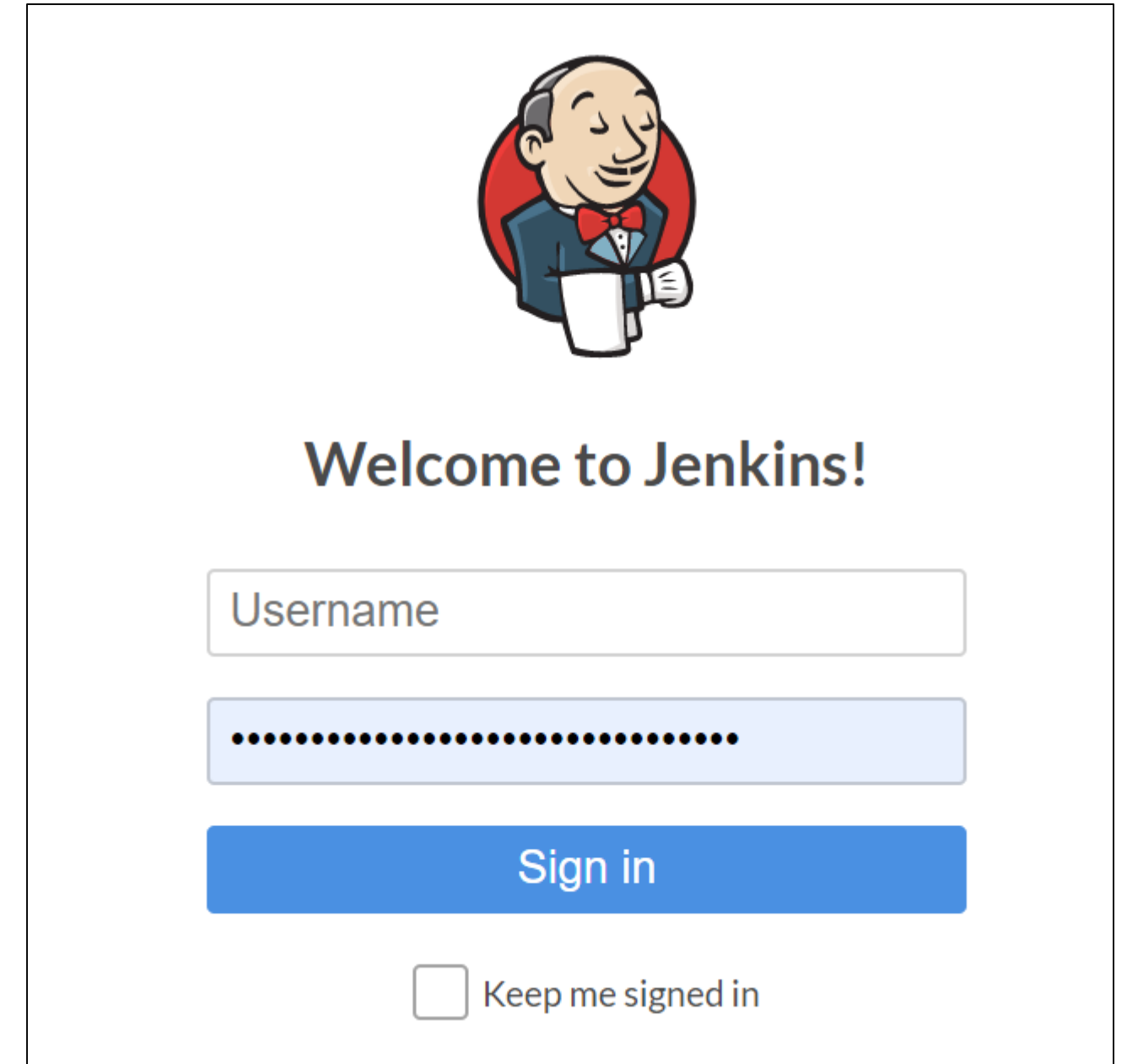


Access the Jenkins

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

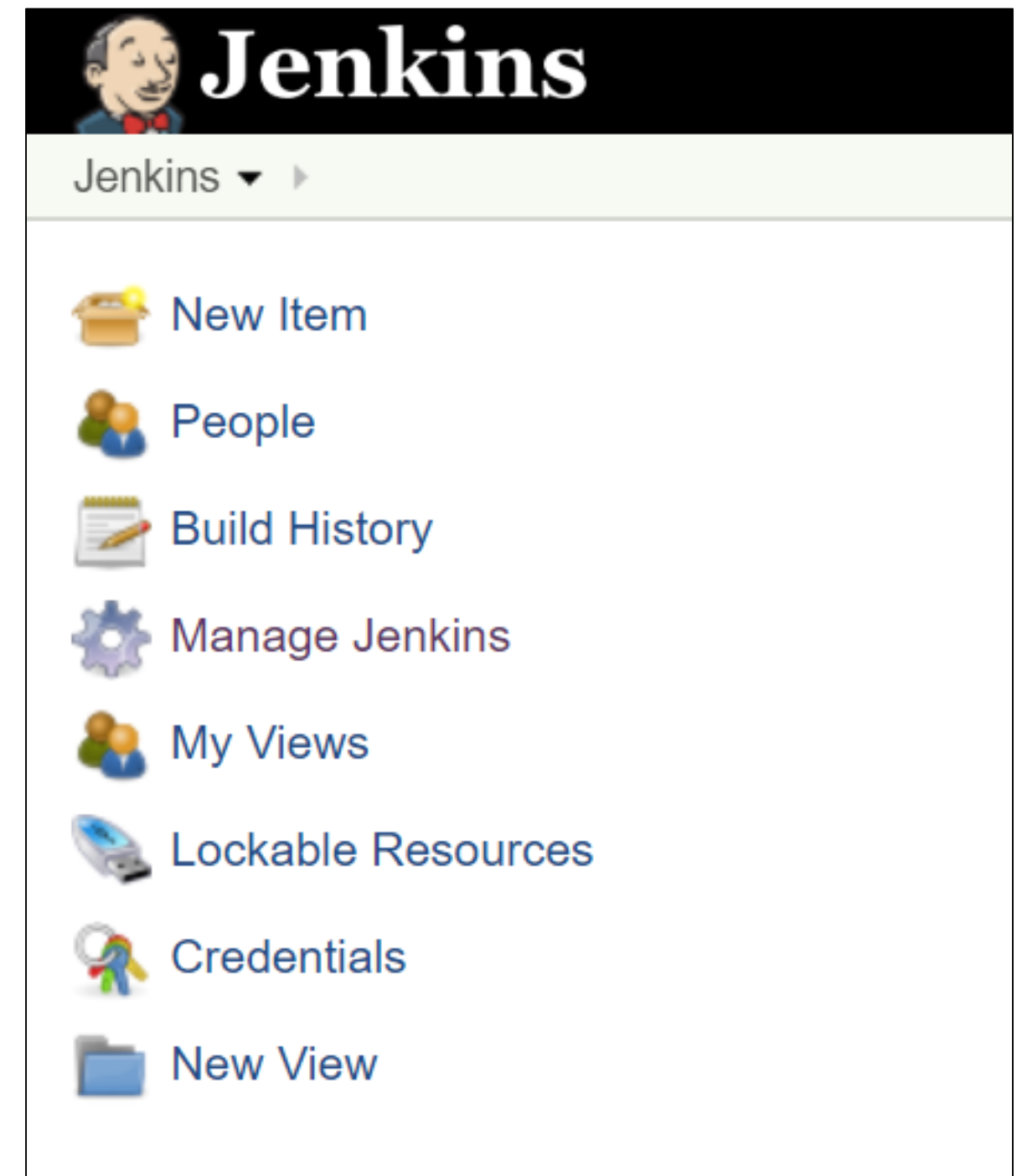


- And you have to login with the username and password established at the previous step.

The image shows the Jenkins login interface. At the top is the Jenkins logo (a cartoon man in a suit). Below it is the text 'Welcome to Jenkins!'. There are two input fields: the first is labeled 'Username' and the second is a password field with dots. Below the password field is a blue 'Sign in' button. At the bottom is a checkbox labeled 'Keep me signed in'.

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 most difficult step of the installing it.



Manage Jenkins



Configure Credentials

Configure the credential providers and types



Global Tool Configuration

Configure tools, their locations and automatic installers.



Reload Configuration from Disk

Discard all the loaded data in memory and reload everything from file system. Useful when configuration changes are made on disk.



Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

🔔 There are updates available

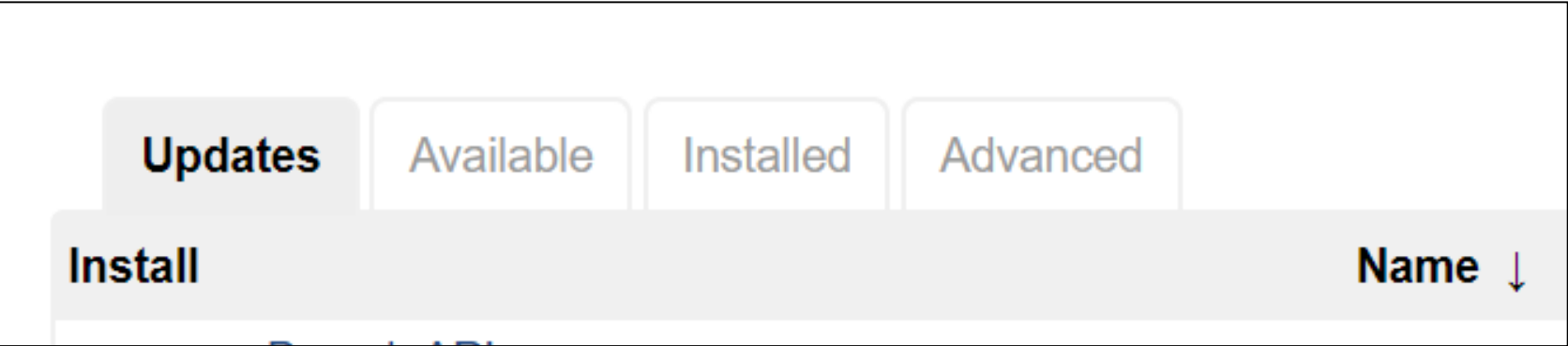


System Information

Displays various environmental information to assist trouble-shooting.

Manage Plugins

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



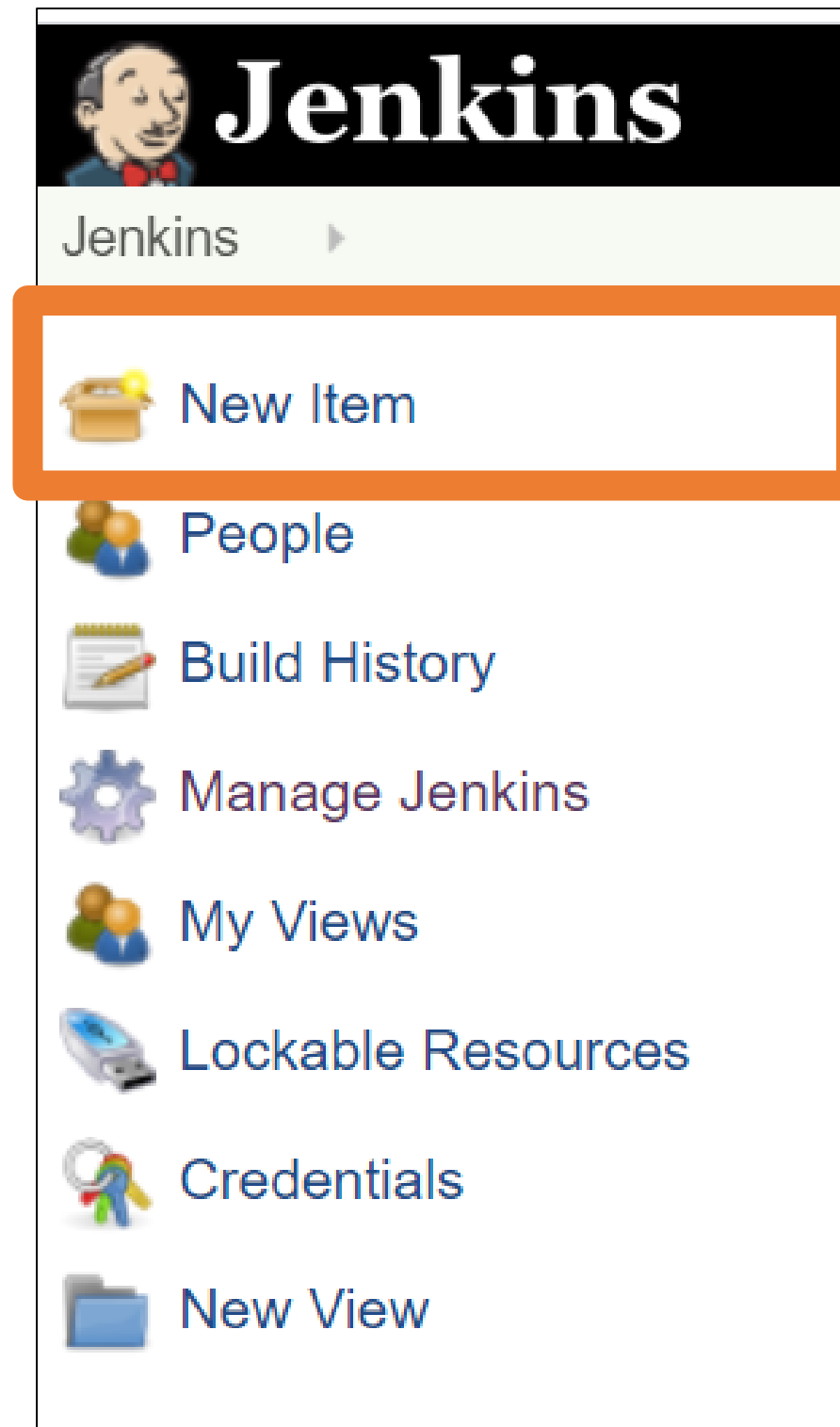
<input checked="" type="checkbox"/>	Git client plugin Utility plugin for Git support in Jenkins	3.0.0	<button>Uninstall</button>
<input checked="" type="checkbox"/>	Git plugin This plugin integrates Git with Jenkins.	4.0.0	<button>Uninstall</button>
<input checked="" type="checkbox"/>	GIT server Allows Jenkins to act as a Git server.	1.8	<button>Uninstall</button>
<input checked="" type="checkbox"/>	GitHub API Plugin This plugin provides GitHub API for other plugins.	1.95	<button>Uninstall</button>
<input checked="" type="checkbox"/>	GitHub Branch Source Plugin Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	2.5.8	<button>Uninstall</button>
<input checked="" type="checkbox"/>	GitHub plugin This plugin integrates GitHub to Jenkins.	1.29.5	<button>Uninstall</button>

JOBS



Jenkins – Adding jobs

- To add new jobs in Jenkins, we have few steps.
- After we connect to Jenkins, we 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.

Enter an item name


» Required field



Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, an



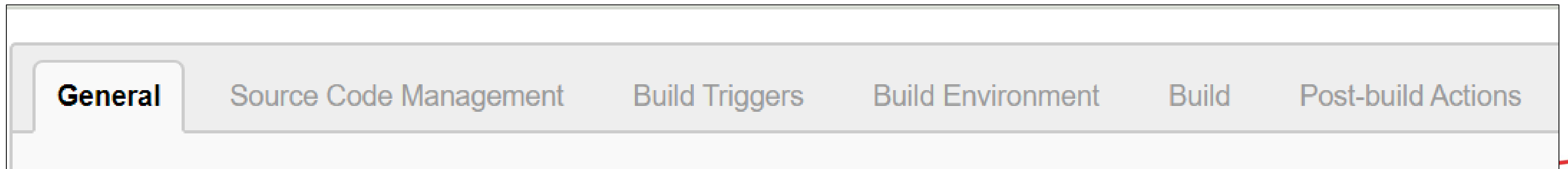
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly kn

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

?

?

?

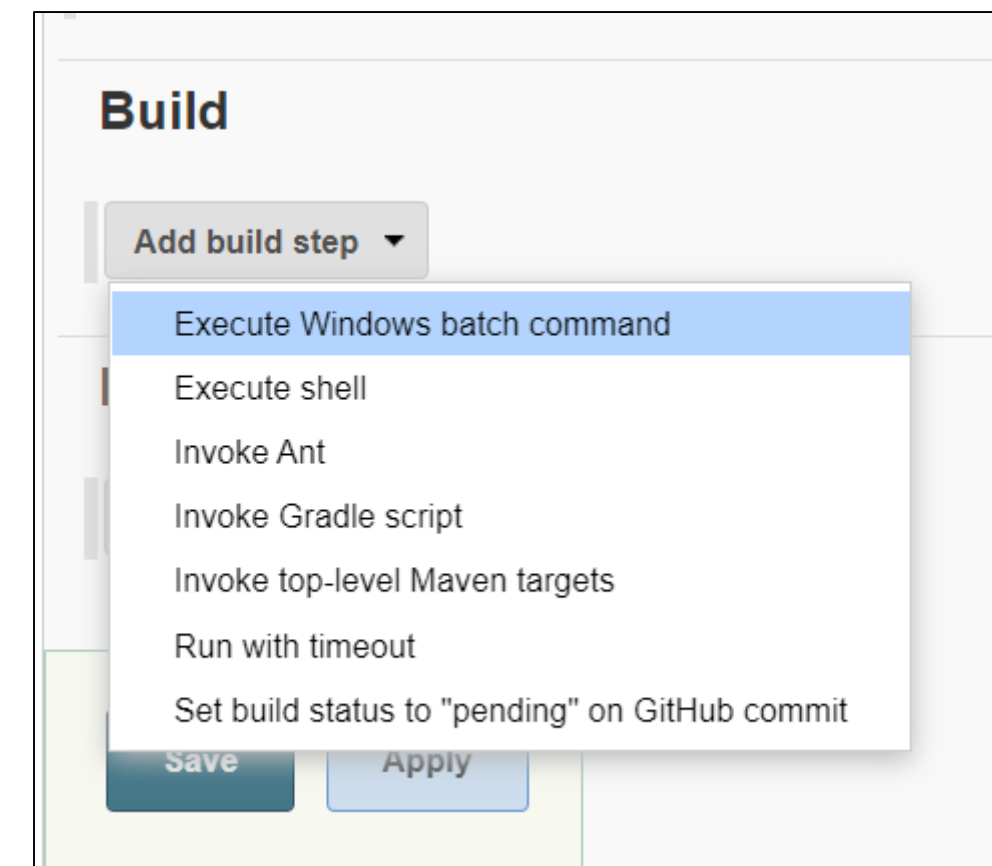
?

?

Advanced...

Build TAB

- On build Tab we can manage what we want 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.

```
(c) 2019 Microsoft Corporation. All rights reserved.  
C:\Users\zamfiroiu>cd ../  
C:\Users>cd../  
C:\>echo "Message"  
"Message"  
C:\>
```

First job on Jenkins

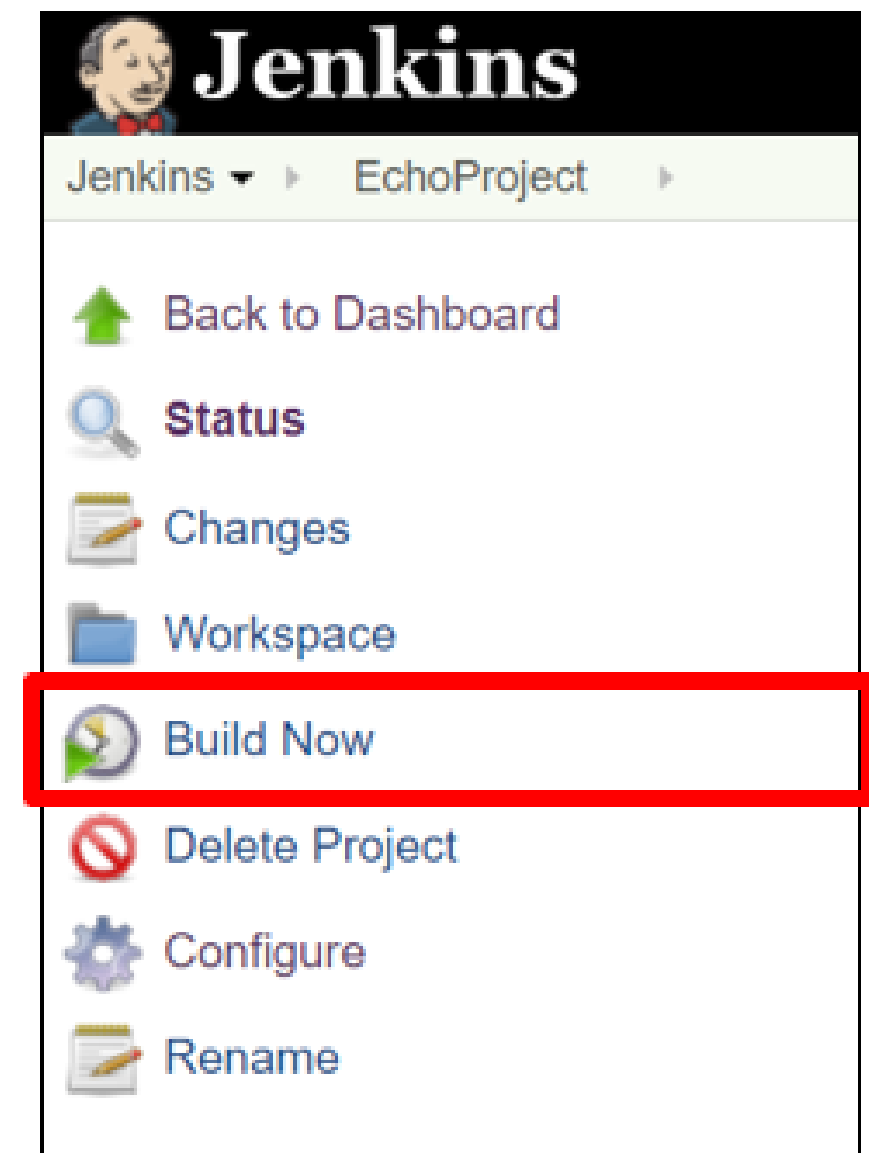
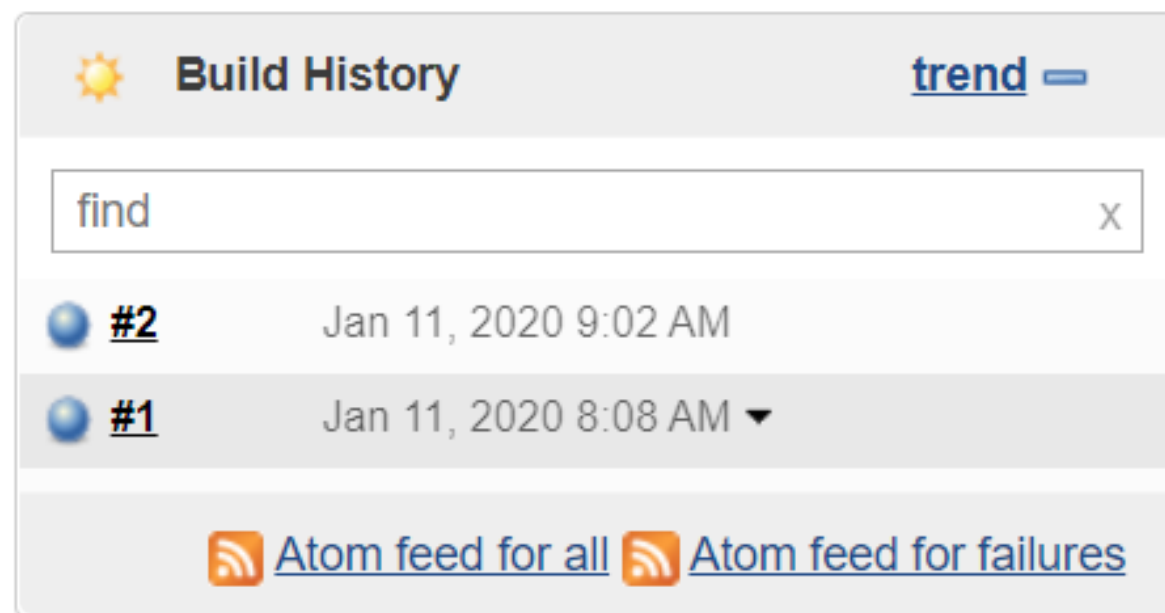
- That assume to put the echo command on Build tab.



The screenshot shows the 'Build' tab configuration in Jenkins. The main section is titled 'Execute Windows batch command'. Below this title is a text area labeled 'Command' containing the text 'echo "Hello from jenkins"'. To the right of the text area are a red 'X' button and a blue question mark icon. Below the text area is a link that says 'See [the list of available environment variables](#)'. At the bottom right of the configuration area is a button labeled 'Advanced...'. At the bottom left of the entire configuration area is a button labeled 'Add build step' with a downward arrow.

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

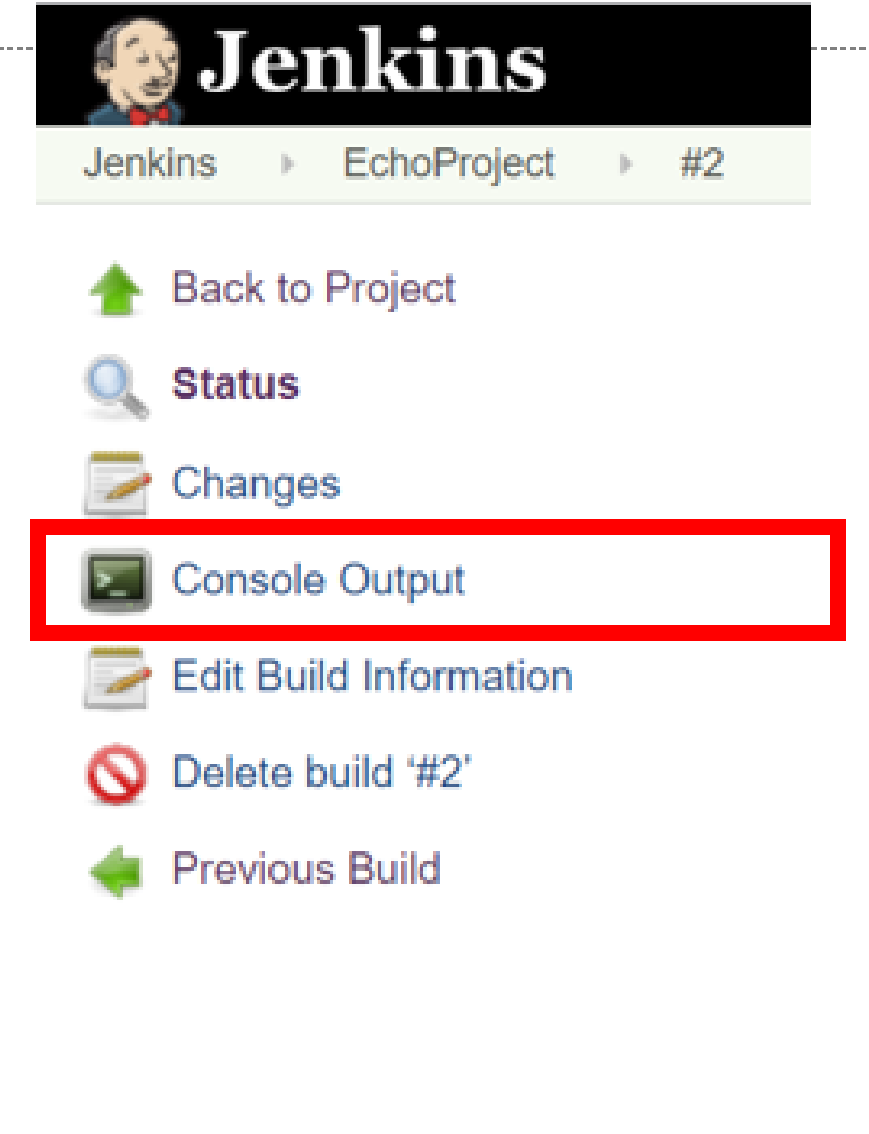
Started by user [Alin Zamfiroiu](#)

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"
```

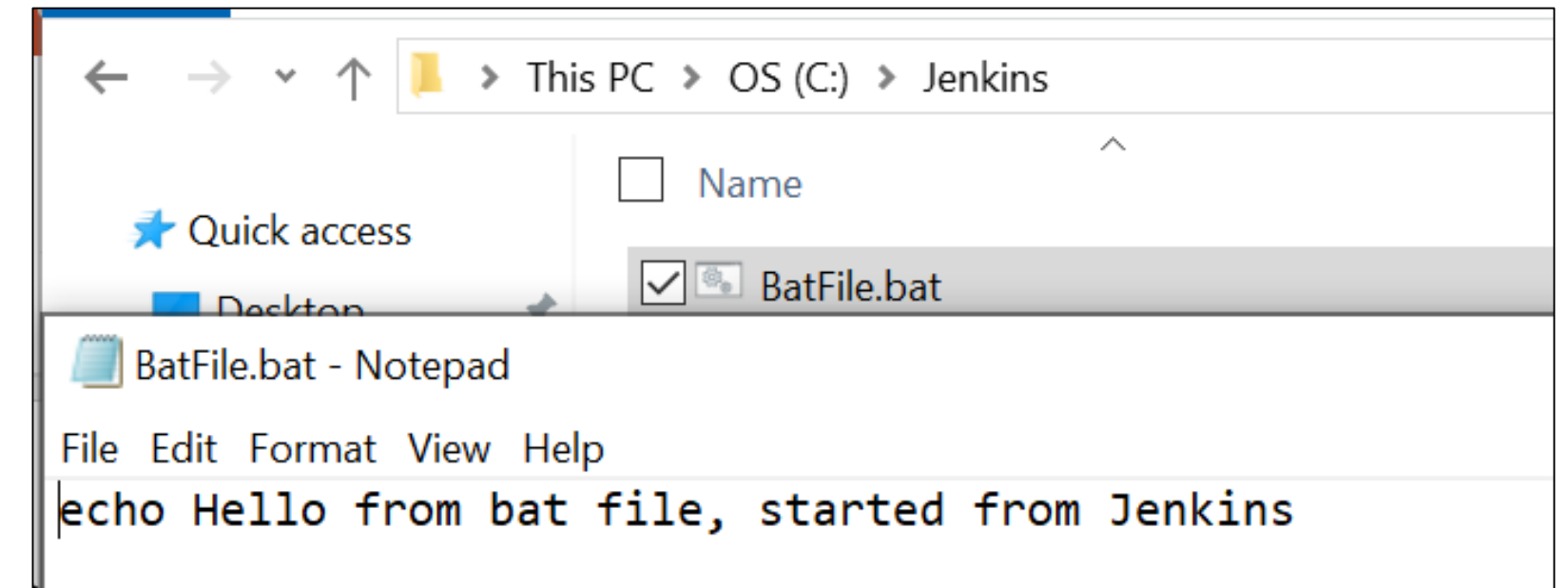
```
C:\Program Files (x86)\Jenkins\workspace\EchoProject>exit 0  
Finished: SUCCESS
```



The screenshot shows the Jenkins web interface for a build named '#2' of the 'EchoProject'. The top navigation bar includes the Jenkins logo and the breadcrumb 'Jenkins > EchoProject > #2'. On the left sidebar, several links are visible: 'Back to Project' (with a green up arrow), 'Status' (with a magnifying glass), 'Changes' (with a document icon), 'Console Output' (with a terminal icon and highlighted by a red rectangle), 'Edit Build Information' (with a pencil icon), 'Delete build '#2'' (with a red prohibition sign), and 'Previous Build' (with a green left arrow).

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.

Build

 **Execute Windows batch command**

Command

```
cd c:/Jenkins  
BatFile.bat
```

See [the list of available environment variables](#)

Advanced...

Job with Batch file

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



Console Output

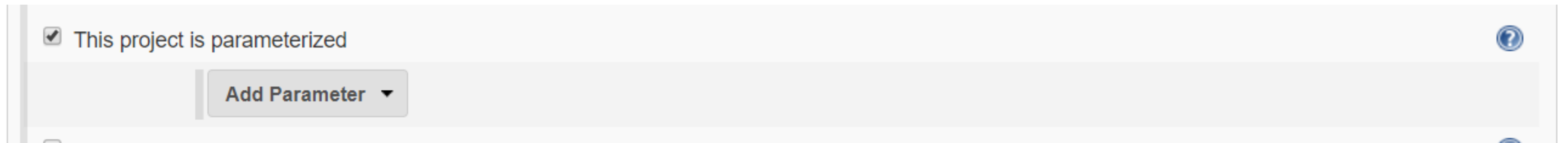
```
Started by user Alin Zamfiroiu  
Running as SYSTEM  
Building in workspace C:\Program Files (x86)\Jenkins\workspace\JobWithBatFil  
[JobWithBatFil] $ cmd /c call C:\WINDOWS\TEMP\jenkins1700733380349532502.bat  
  
C:\Program Files (x86)\Jenkins\workspace\JobWithBatFil>cd c:/Jenkins  
  
c:\Jenkins>BatFile.bat  
  
c:\Jenkins>echo Hello from bat file, started from Jenkins  
Hello from bat file, started from Jenkins  
Finished: SUCCESS
```

Parameterized jobs



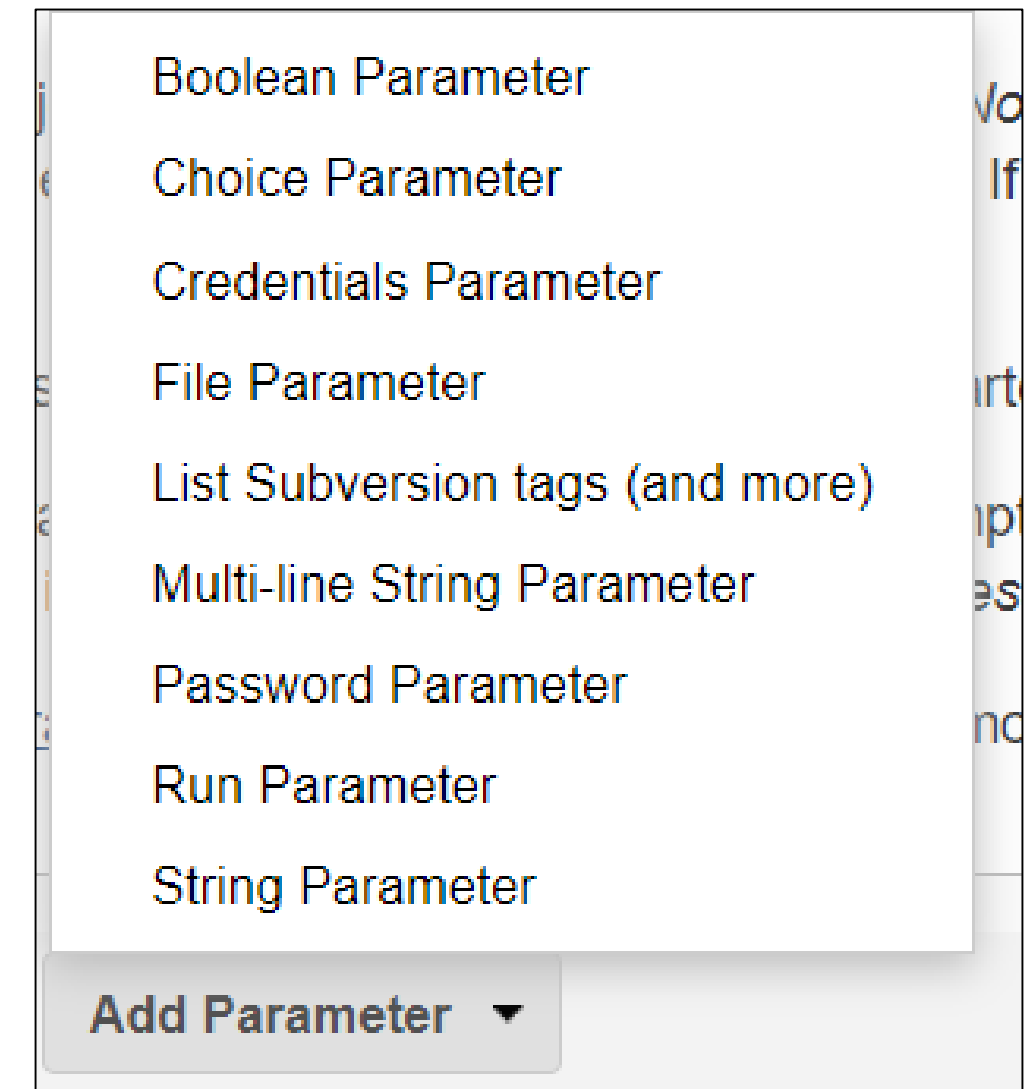
Parametrized Job

- 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.



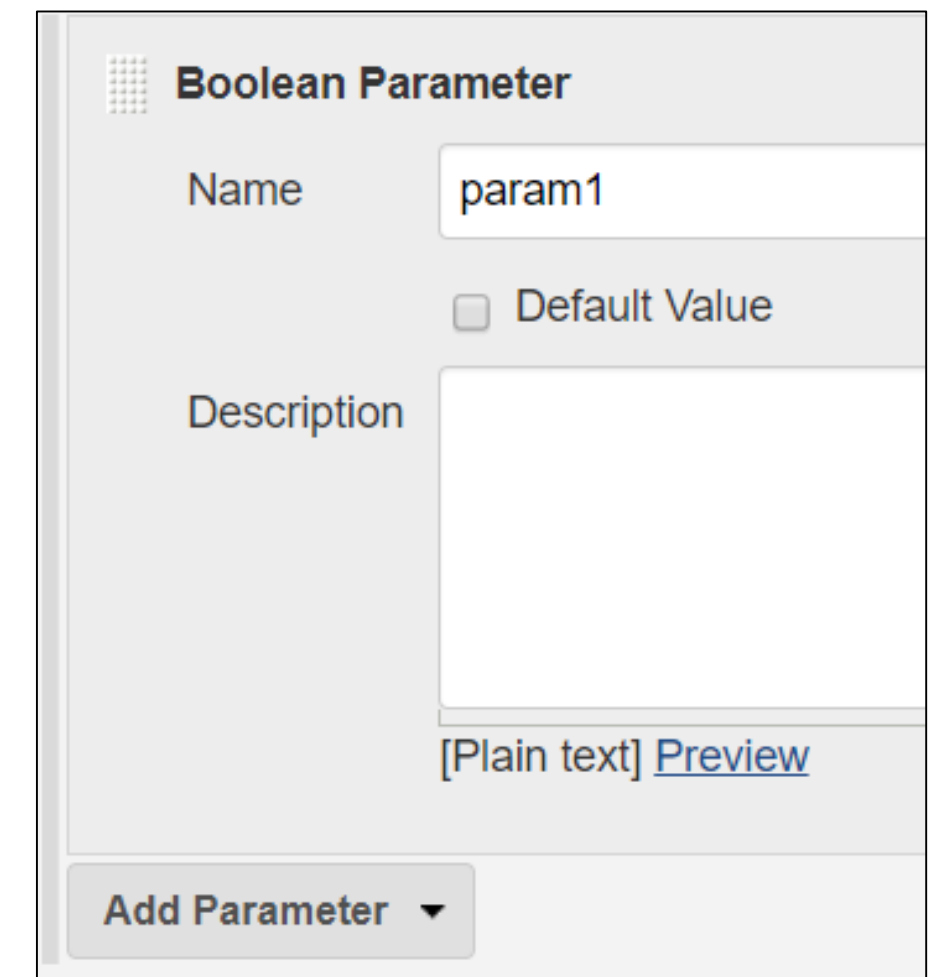
Parametrized Job

- 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.

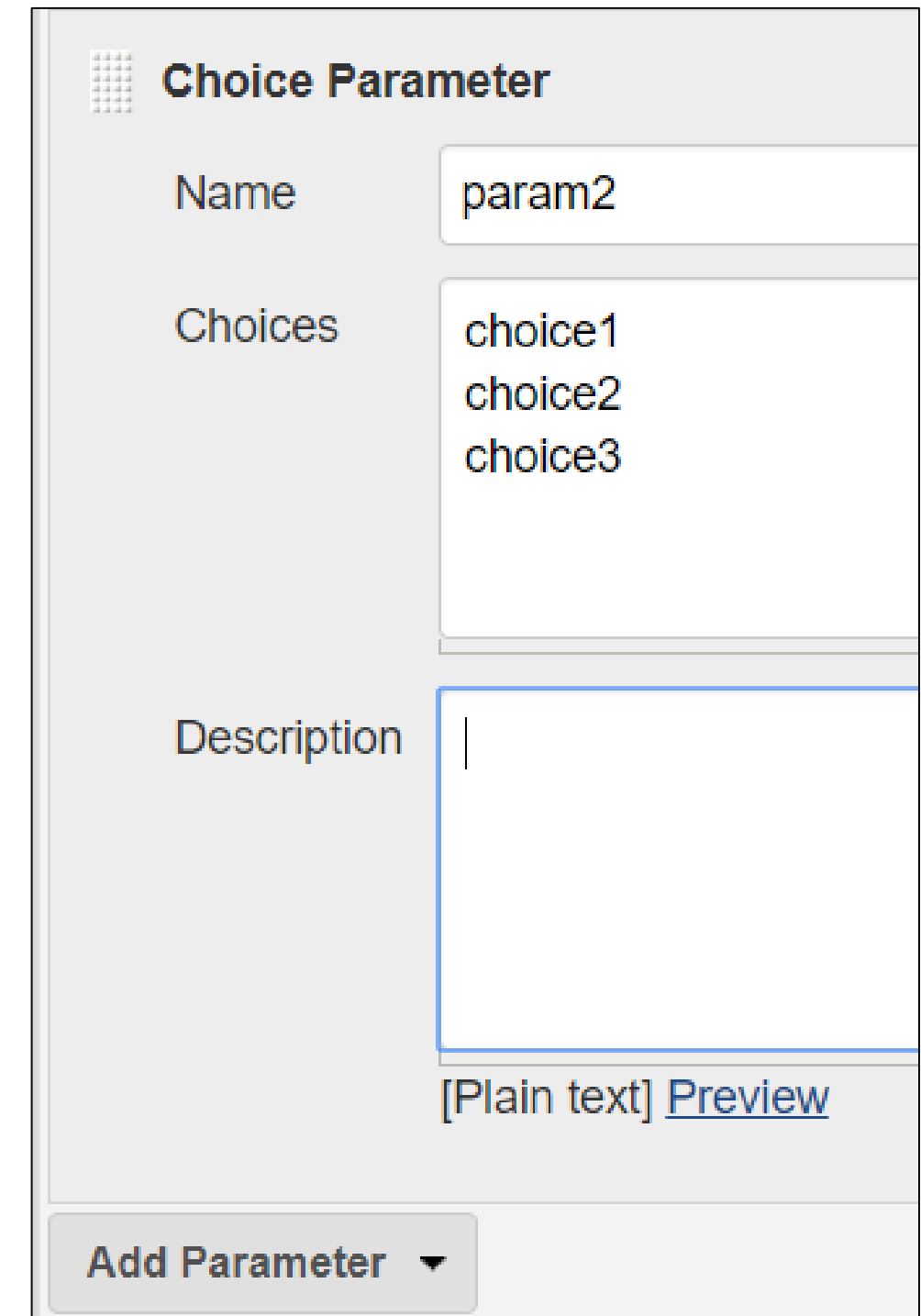


The image shows a configuration window titled "Boolean Parameter". It contains the following fields and controls:

- Name:** A text input field containing "param1".
- Default Value:** A checkbox that is currently unchecked.
- Description:** A large text area for user input.
- Preview:** A small preview area at the bottom of the description field showing "[Plain text]" and a "Preview" link.
- Add Parameter:** A button with a dropdown arrow at the bottom left of the window.

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.



The image shows a configuration form for a 'Choice Parameter'. The form has a title bar with a grid icon and the text 'Choice Parameter'. Below the title bar, there are three main sections: 'Name', 'Choices', and 'Description'. The 'Name' section has a text input field containing 'param2'. The 'Choices' section has a text area containing three lines of text: 'choice1', 'choice2', and 'choice3'. The 'Description' section has a text area with a vertical cursor. Below the text area, there is a label '[Plain text]' and a link 'Preview'. At the bottom of the form, there is a button labeled 'Add Parameter' with a downward arrow.

Choice Parameter	
Name	param2
Choices	choice1 choice2 choice3
Description	<div></div> <div>[Plain text] Preview</div>
Add Parameter ▼	

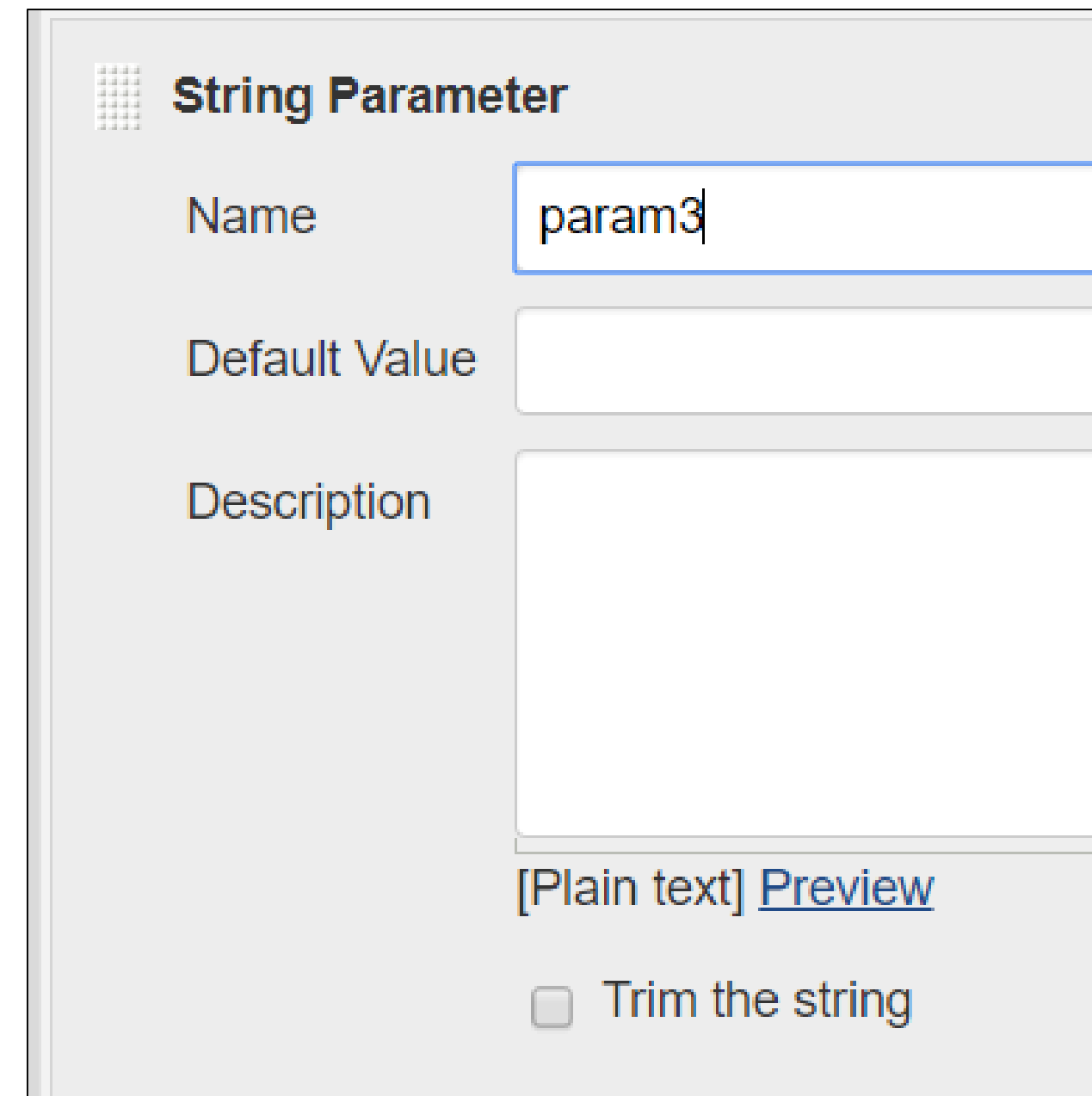
Parametrized Job

- 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.



The image shows a configuration window titled "String Parameter". It contains three input fields: "Name" with the value "param3", "Default Value" (empty), and "Description" (empty). Below the "Description" field, there is a link "[Plain text] [Preview](#)". At the bottom, there is a checkbox labeled "Trim the string" which is currently unchecked.

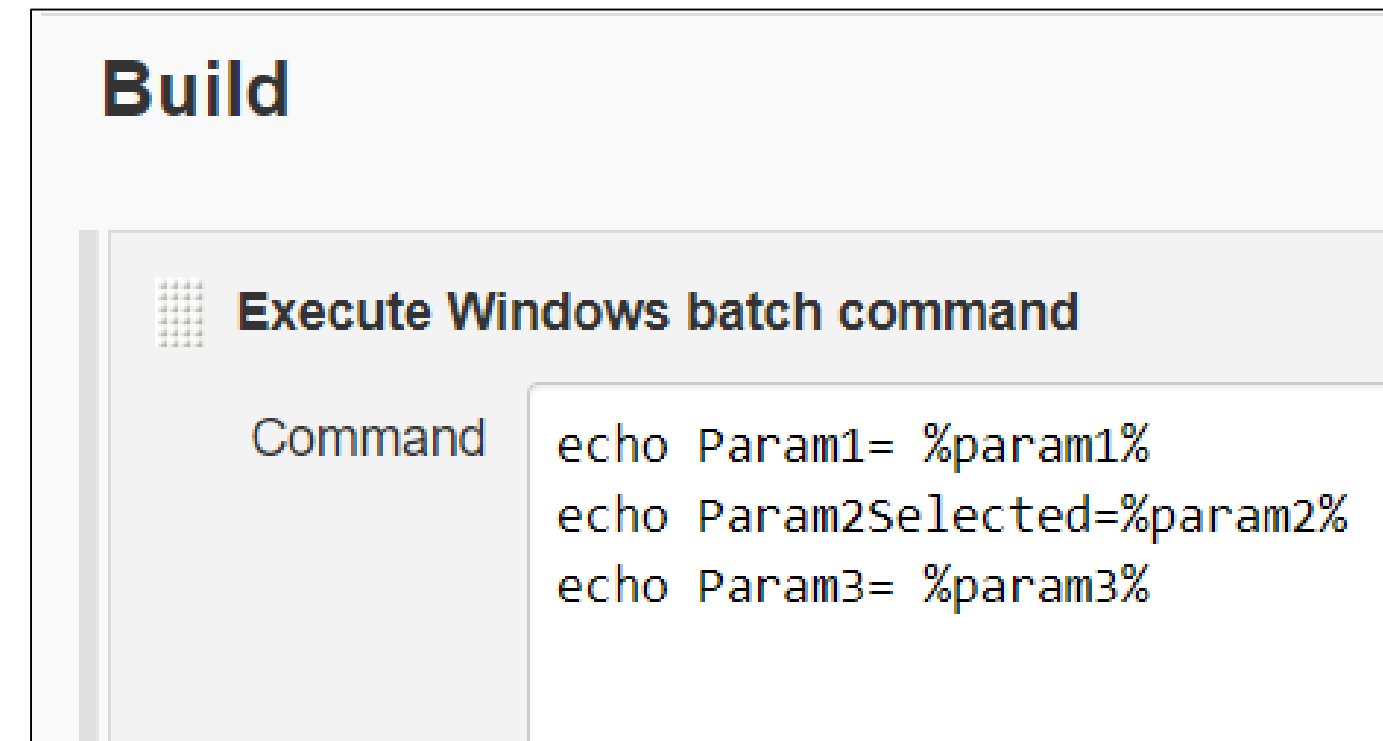
String Parameter	
Name	param3
Default Value	
Description	
[Plain text] Preview	
<input type="checkbox"/> Trim the string	

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

Project ParametrizedJob

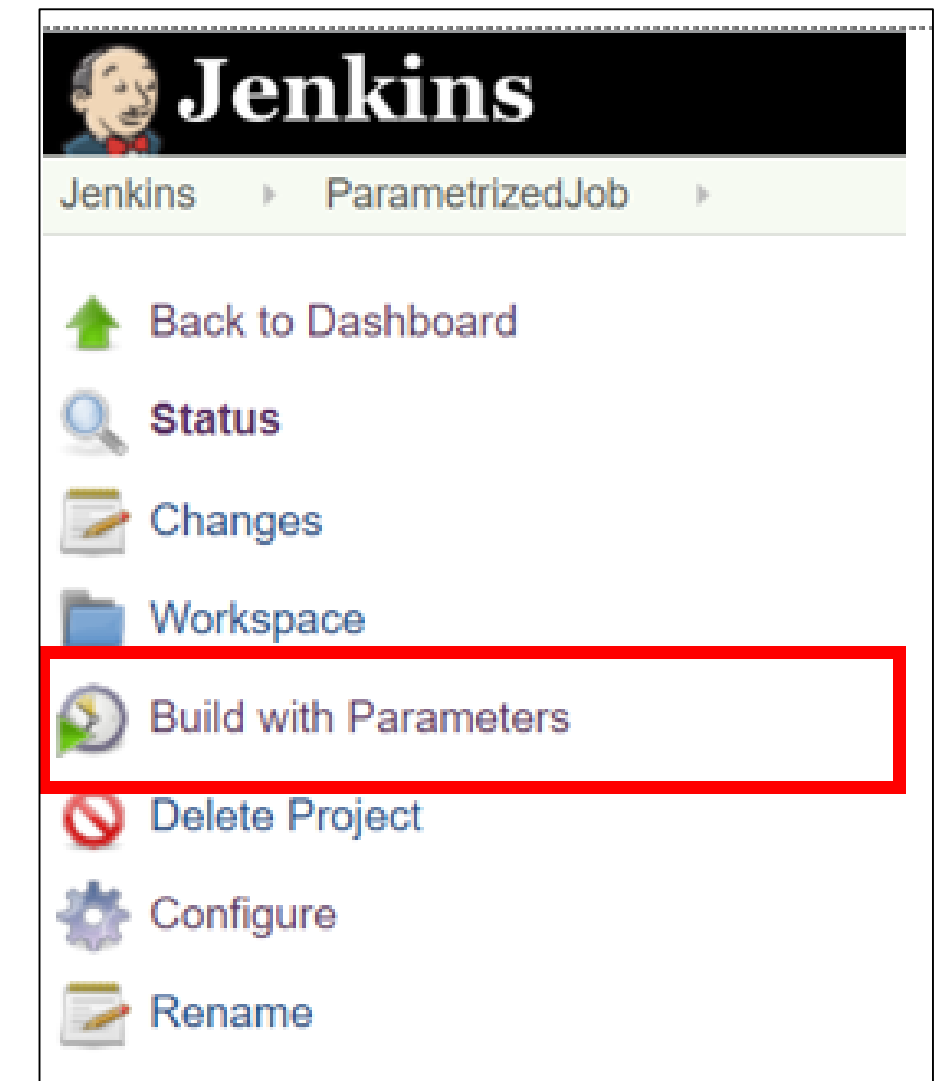
This build requires parameters:

☒ param1

param2

param3

Build



Parametrized Job

Project ParametrizedJob

This build requires parameters:

☐ param1

param2

param3

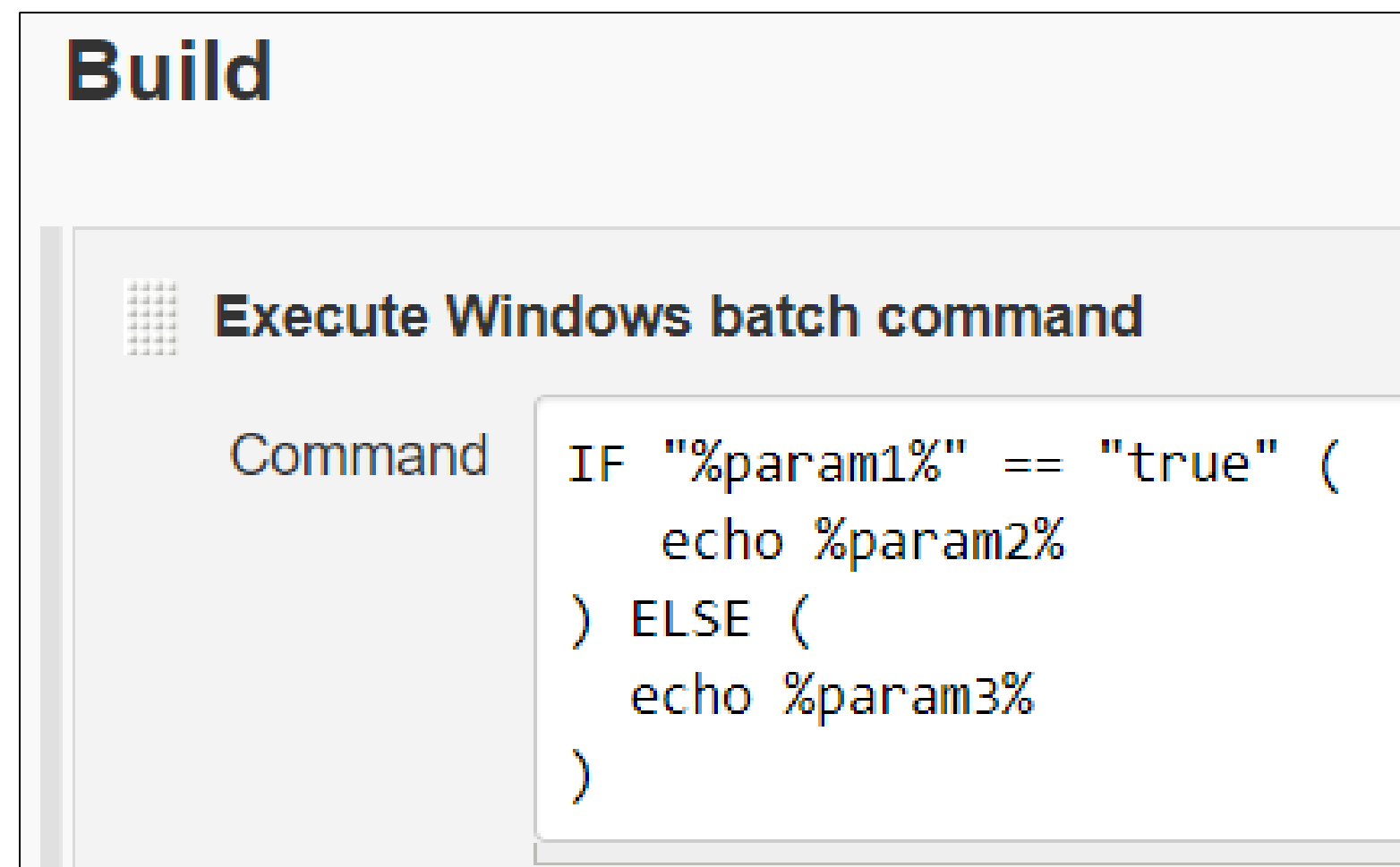
Build

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



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\jenkins6078657781819056225.bat

```
C:\Program Files (x86)\Jenkins\workspace\ParametrizedJob>IF "true" == "true" (echo choice2 ) ELSE (echo test )
choice2
```

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

Finished: SUCCESS

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

☒ 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.

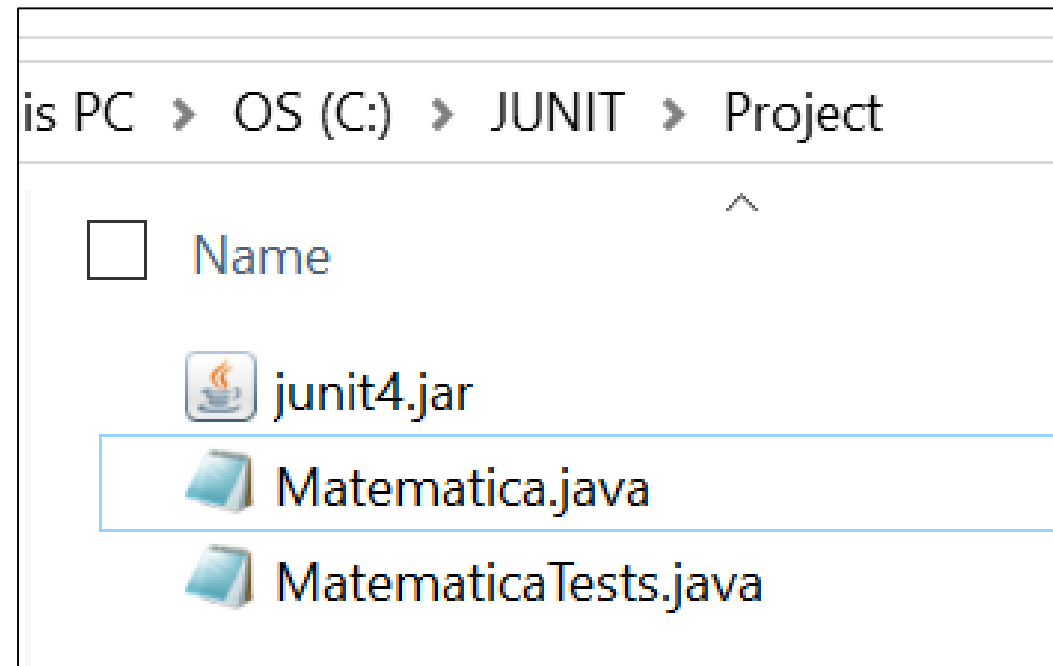
```
MatematicaTests.java x Matematica.java x
1  import java.util.ArrayList;
2  import java.util.List;
3
4  public class Matematica {
5
6      public int suma(int param1, int param2) {
7          return param1 + param2;
8      }
9
10     public double raport(int numarator, int numitor) {
11         if(numitor==0){
12             throw new IllegalArgumentException();
13         }
14         return (double)numarator / numitor;
15     }
16
17     public boolean estePar(int numar) {
18         return numar % 2 == 0;
19     }
20
21     public List<Integer> nNumerePare(int n) {
22         if(n==0) {
23             return null;
24         }
25         if(n<0){
26             throw new IllegalArgumentException();
27         }
28         List<Integer> lista = new ArrayList<Integer>();
29         for (int i = 0; i < n; i++)
30             lista.add(i * 2);
31         return lista;
32     }
33 }
34
```

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
1  import static org.junit.Assert.*;
2  import java.util.List;
3  import org.junit.Before;
4  import org.junit.Test;
5
6  public class MatematicaTests {
7      Matematica mate;
8
9      @Before
10     public void setUp() {
11
12     }
13
14     @Test
15     public void testSumaCorectitudine() {
16         int rezultat=mate.suma(3, 12);
17         int rezultatAsteptat=15;
18         assertEquals(rezultatAsteptat, rezultat);
19     }
20
21     @Test
22     public void testRaportCorect() {
23
24         double rezultat=mate.raport(12, 3);
25         double rezultatAsteptat=4;
26         assertEquals(rezultatAsteptat, rezultat, 0.1);
27     }
28
29     @Test
30     public void testRaportExceptie() {
31         try{
32             mate.raport(34, 0);
33             fail("Metoda nu arunca exceptie");
34         }catch(IllegalArgumentException er){
35
36         }
37     }
38 }
```


Job to run Unit Tests



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

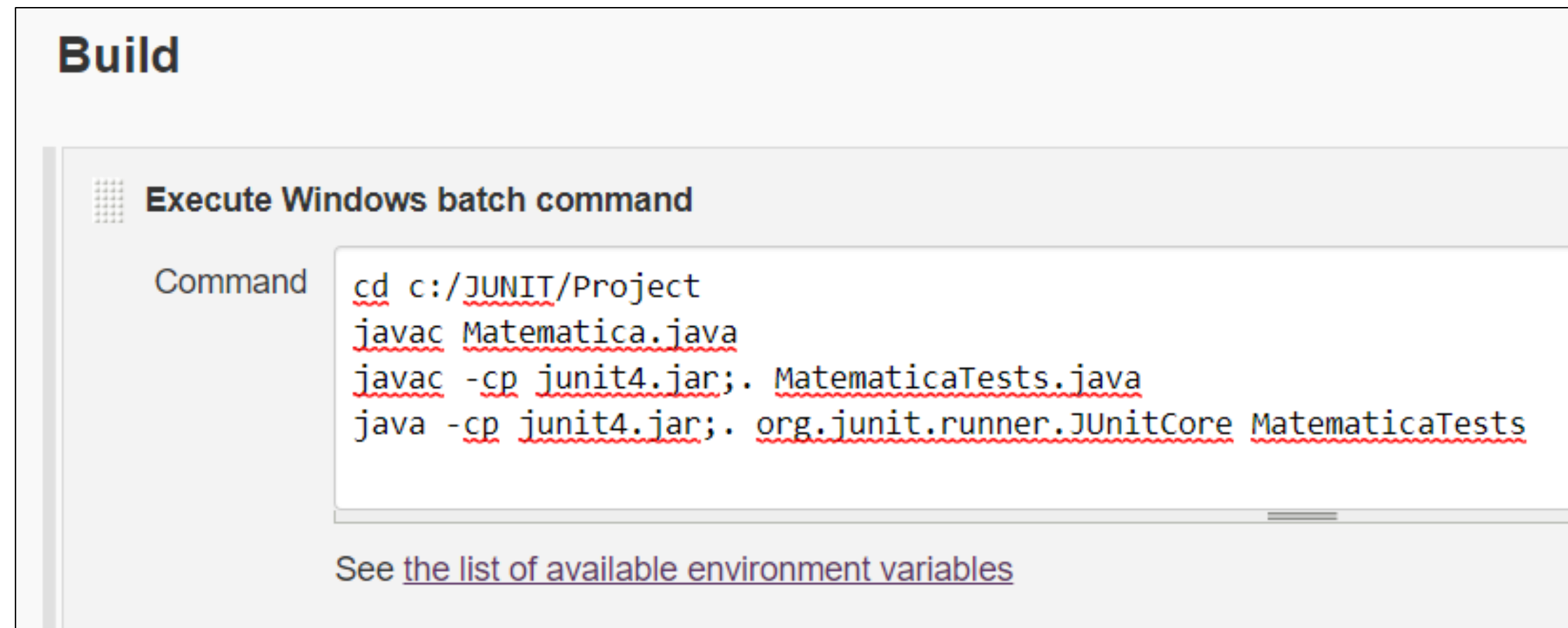
We have these files in a project directory.
We have putted the junit library also in this directory

```
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

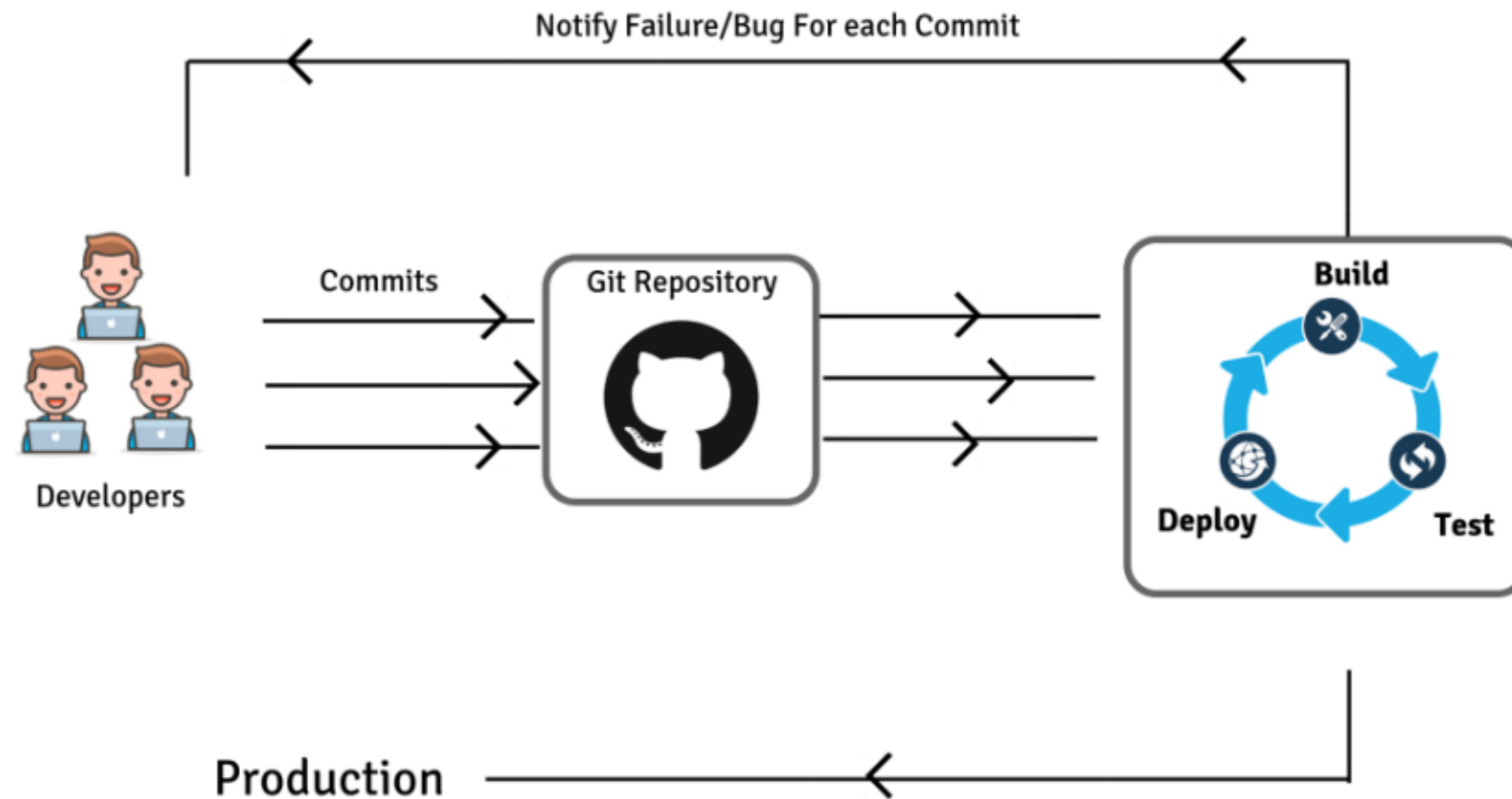
Job to run Unit Tests

- 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



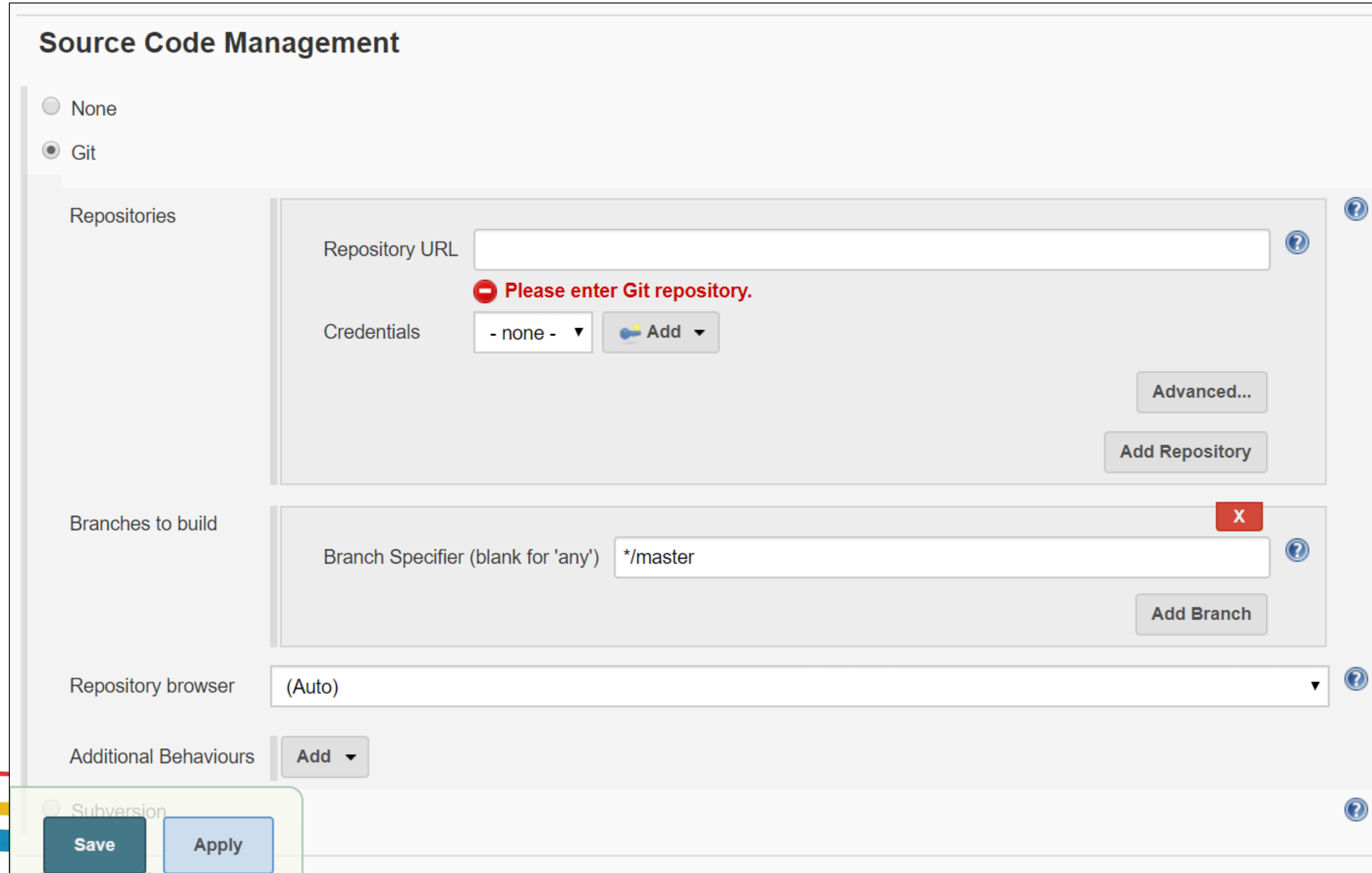
Jenkins with GitHub

- 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](#)

 Matematica.java	Add files via upload
 MatematicaTests.java	Add files via upload
 README.md	Create README.md

Jenkins with GitHub

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



The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' radio button is selected under the 'Source Code Management' section. The 'Repositories' section contains a 'Repository URL' field, a 'Credentials' dropdown menu (currently set to '- none -'), and an 'Add' button. A red error message 'Please enter Git repository.' is displayed below the 'Repository URL' field. The 'Advanced...' button is visible. The 'Add Repository' button is at the bottom right of the 'Repositories' section. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field containing '*/master' and an 'Add Branch' button. The 'Repository browser' dropdown menu is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. At the bottom, the 'Subversion' radio button is visible, and the 'Save' and 'Apply' buttons are at the bottom center.

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

Credentials

Please enter Git repository.

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Additional Behaviours

☐ Subversion

Jenkins with GitHub

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

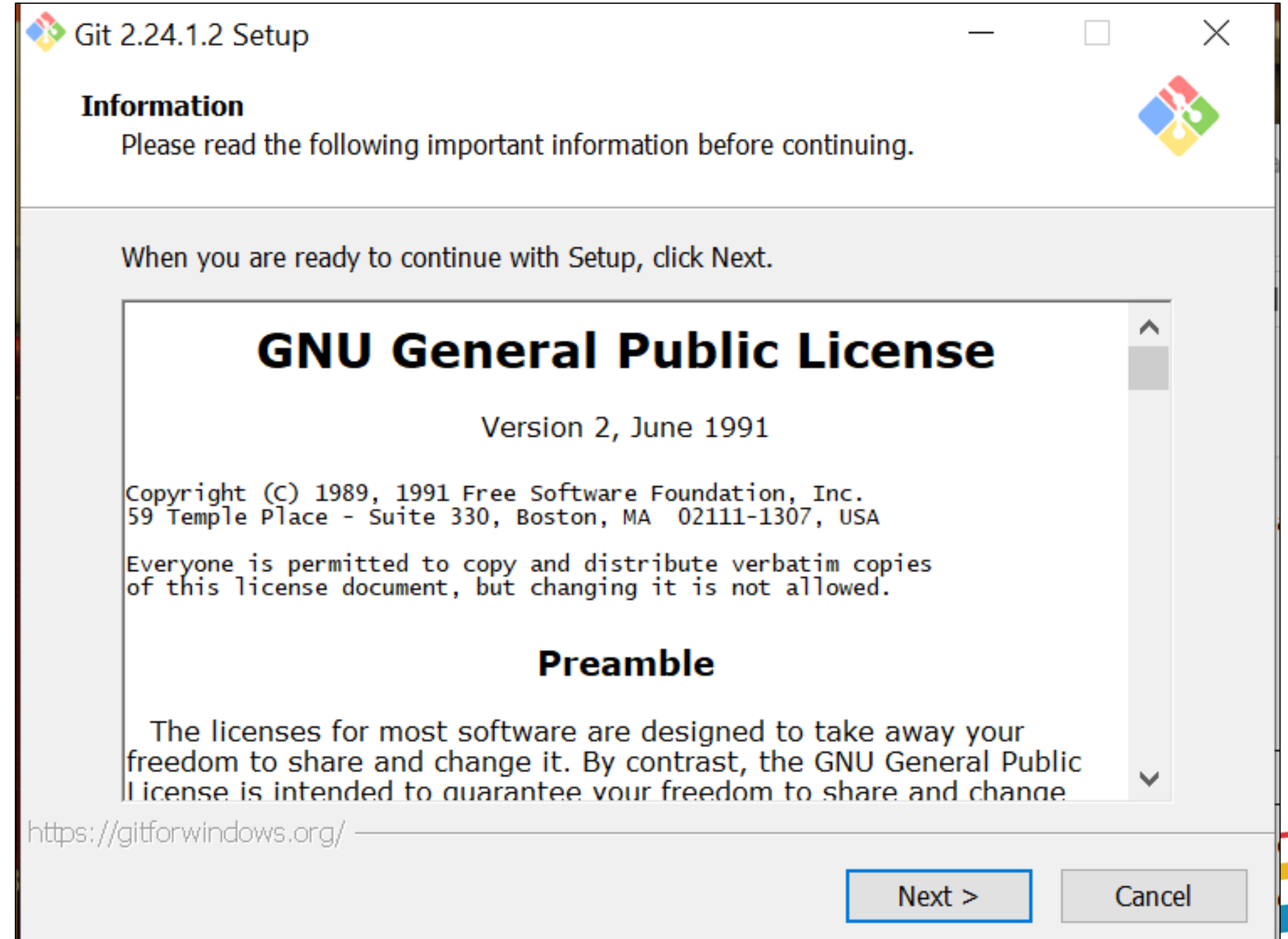
A screenshot of a Jenkins console output showing a red error message. The message is: "Failed to connect to repository : Error performing git command: git.exe ls-remote -h https://github.com/zamfirou/Test Jenkins git HEAD". The text is red and preceded by a red circle with a white minus sign icon.

Failed to connect to repository : Error performing git command: git.exe ls-remote -h https://github.com/zamfirou/Test Jenkins git HEAD

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

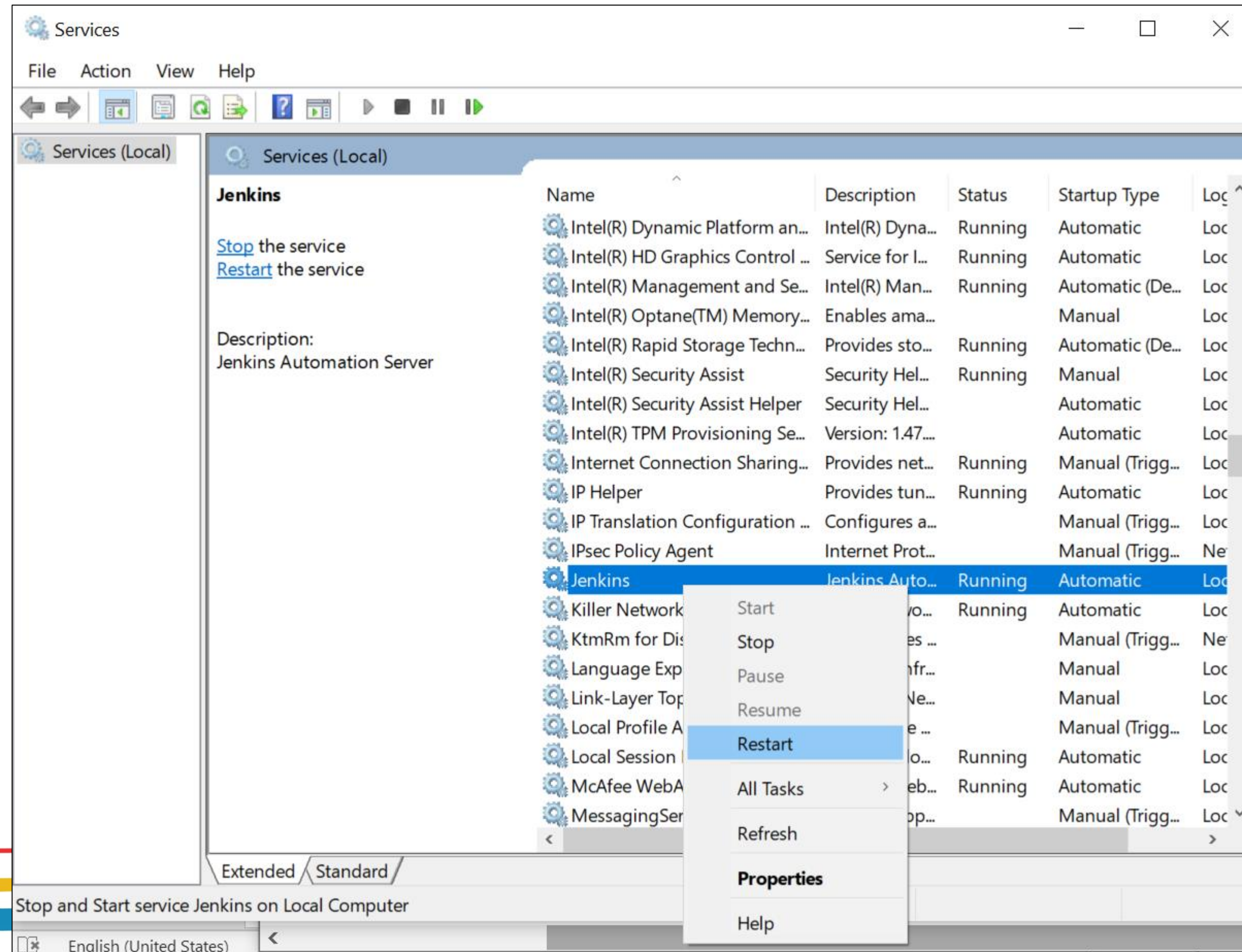
Jenkins with GitHub

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



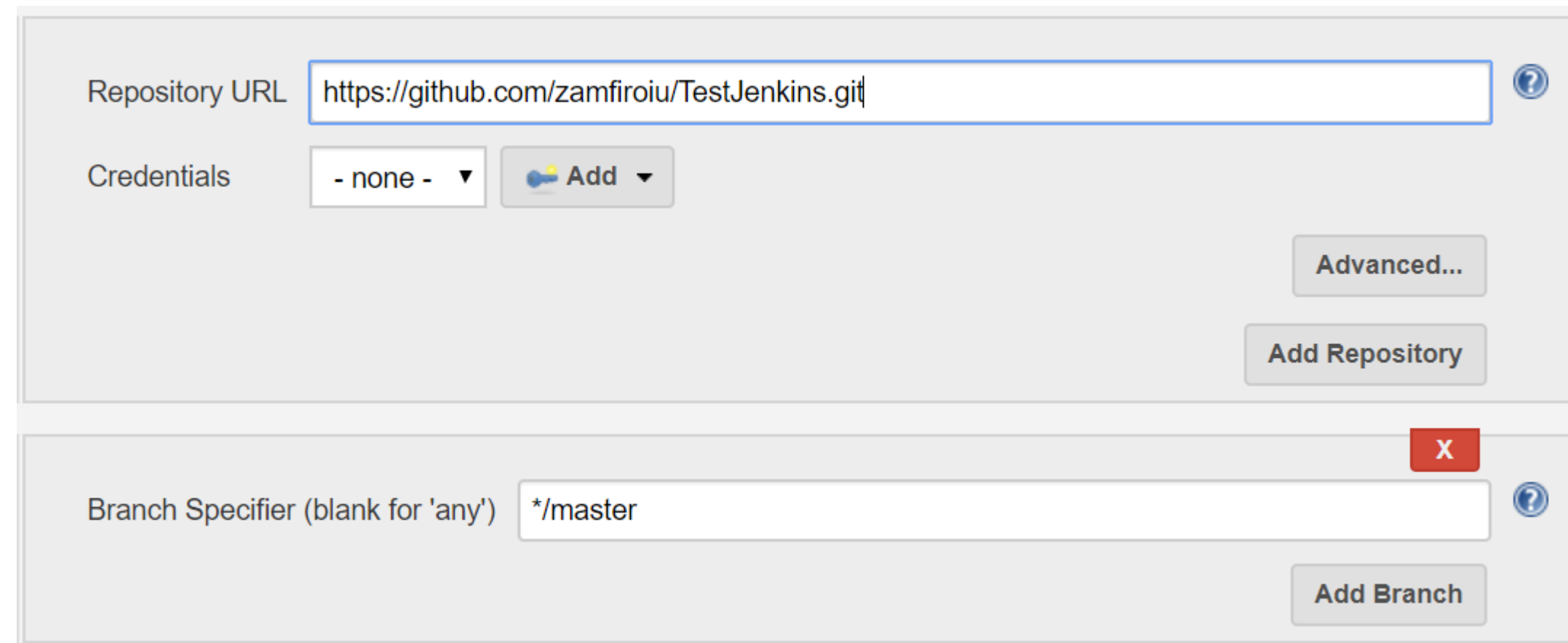
Jenkins with GitHUB

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



Jenkins with GitHub

- Now everything should be ok!



The image shows the Jenkins configuration page for the GitHub plugin. It is divided into two main sections. The top section is for repository configuration, featuring a 'Repository URL' text box with the value 'https://github.com/zamfiroiu/TestJenkins.git', a 'Credentials' dropdown menu set to '- none -' with an 'Add' button, and buttons for 'Advanced...', 'Add Repository', and a help icon. The bottom section is for branch configuration, featuring a 'Branch Specifier (blank for 'any')' text box with the value '*/master', an 'Add Branch' button, a red 'X' icon, and a help icon.

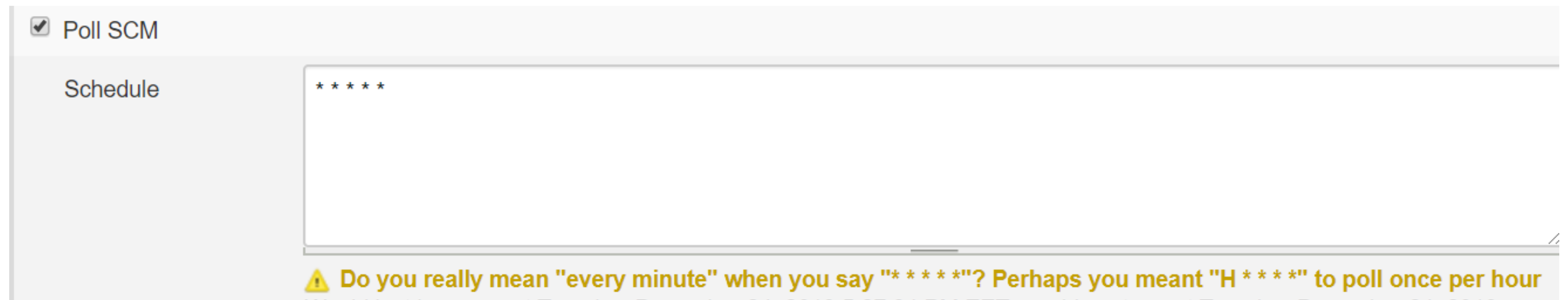
Repository URL

Credentials

Branch Specifier (blank for 'any')

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).

Authentication

☐ Disable remember me

Security Realm

Security Realm

☐

☐ Delegate to servlet container

☒ Jenkins' own user database

☐ Allow users to sign up

☐ None

Authorization section

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

Authorization

Strategy

Authorization

- ☐ Anyone can do anything
- ☐ Legacy mode
- ☒ Logged-in users can do anything
 - ☐ Allow anonymous read access
- ☐ Matrix-based security
- ☐ Project-based Matrix Authorization Strategy

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 on Jenkins.



Job Configuration History

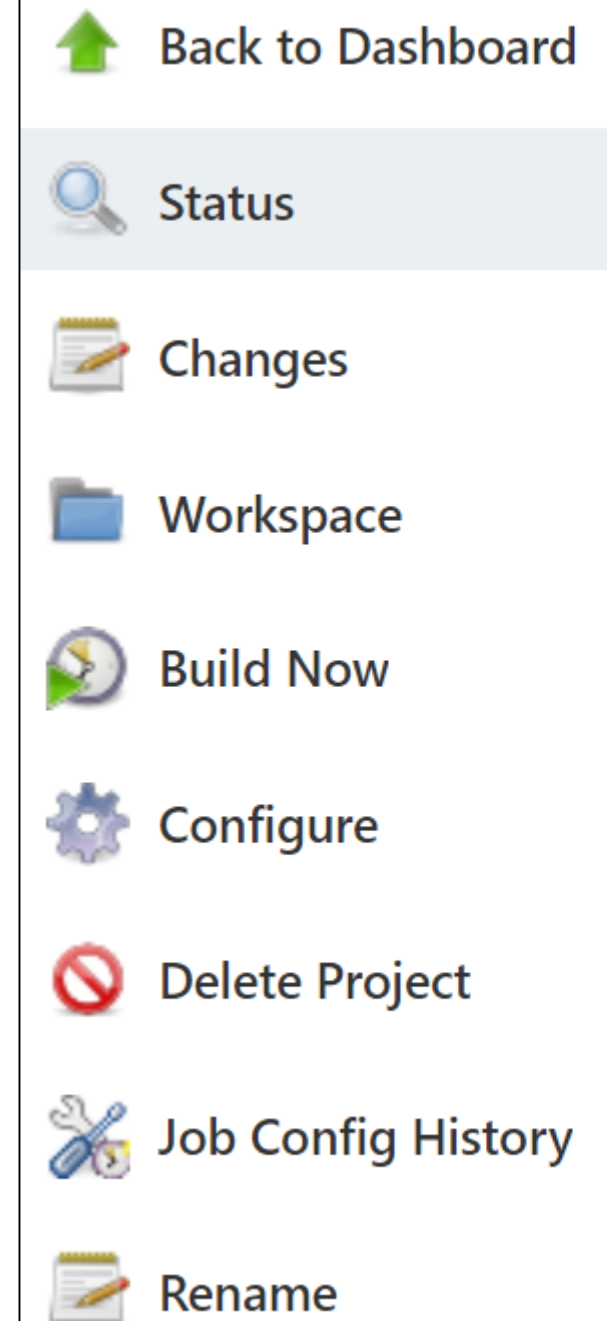


Job Configuration History

- We have to install the Job Configuration History.






Enabled	Name ↓
<input checked="" type="checkbox"/>	Job Configuration History Plugin Job history plugin for Jenkins.

- 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)		<input type="radio"/>	<input checked="" type="radio"/>	
	Changed	adminzamfiroiu	 View as XML (RAW)		<input checked="" type="radio"/>	<input type="radio"/>	

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**.

