מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

Real-Time Group

Jenkins

benny@rt-ed.co.il

by Vladimir Levintovich

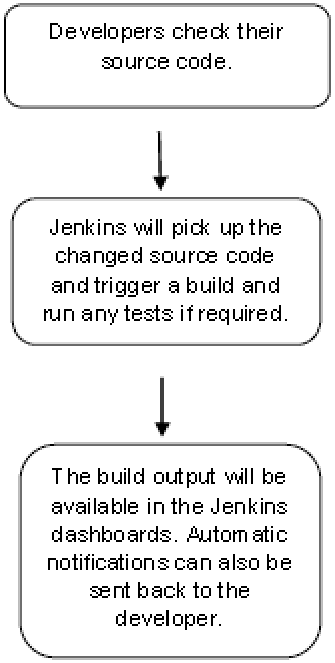
רח` מרדכי רוז'נסקי 14, ראשל"צ טל. 7067057 -077

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

Introduction to Jenkins

Why Jenkins?

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



Developers check their source code.

Jenkins will pick up the changed source code and trigger a build and run any tests if required.

The build output will be available in the Jenkins dashboards. Automatic notifications can also be sent back to the developer.

What is Continuous Integration?

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.

VirtualBox Installation

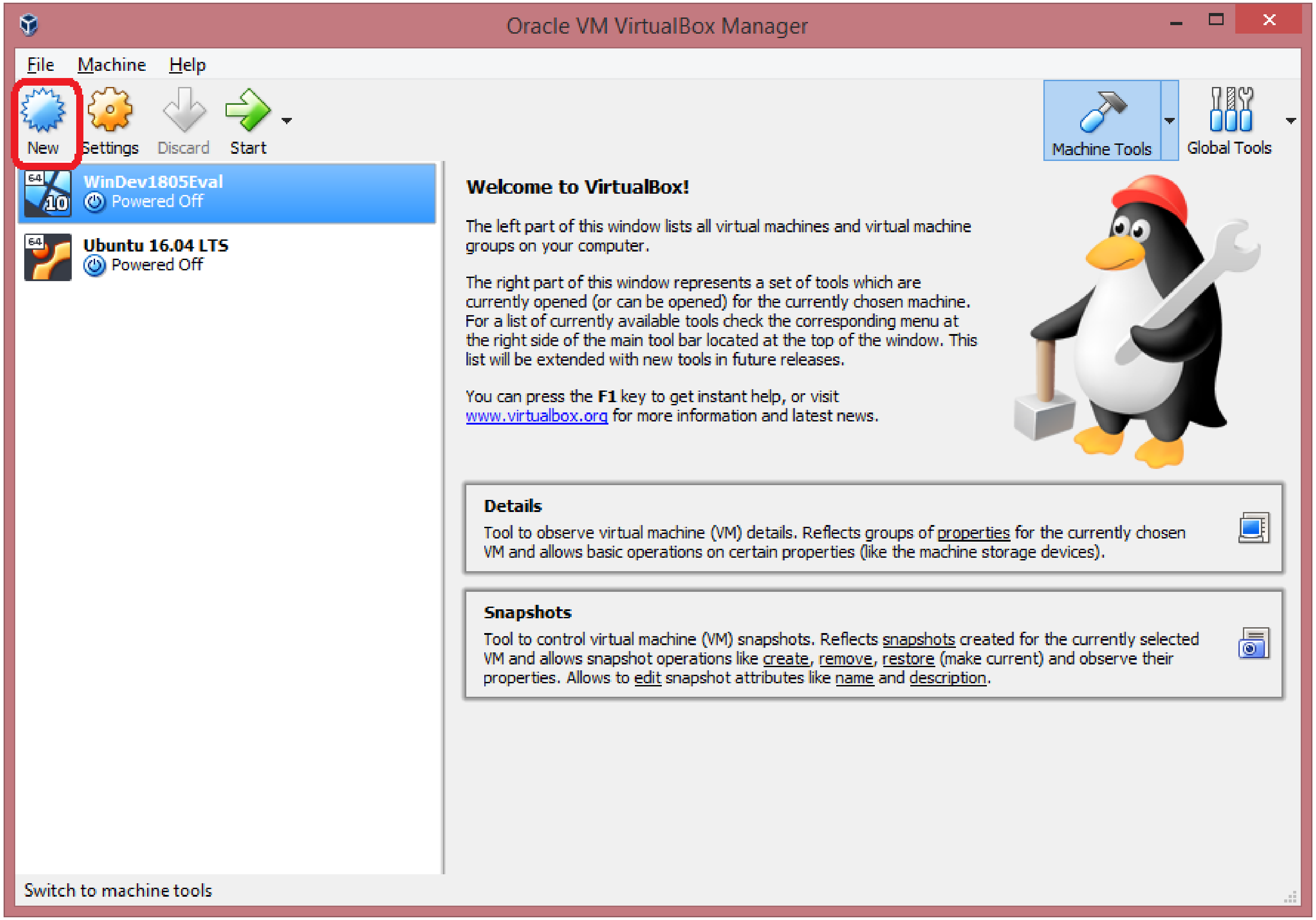
VirtualBox for virtual machine

vmbox installation

- If you using windows this mean that you must install virtual machine
- I’m using virtual box oracle
- First of all ill show you how to configure virtual box and how to install ubuntu
- To install virtual box just go to google type virtual box and download and install it.
- Linux users can skip this topic

vmbox installation

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer.

The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Details

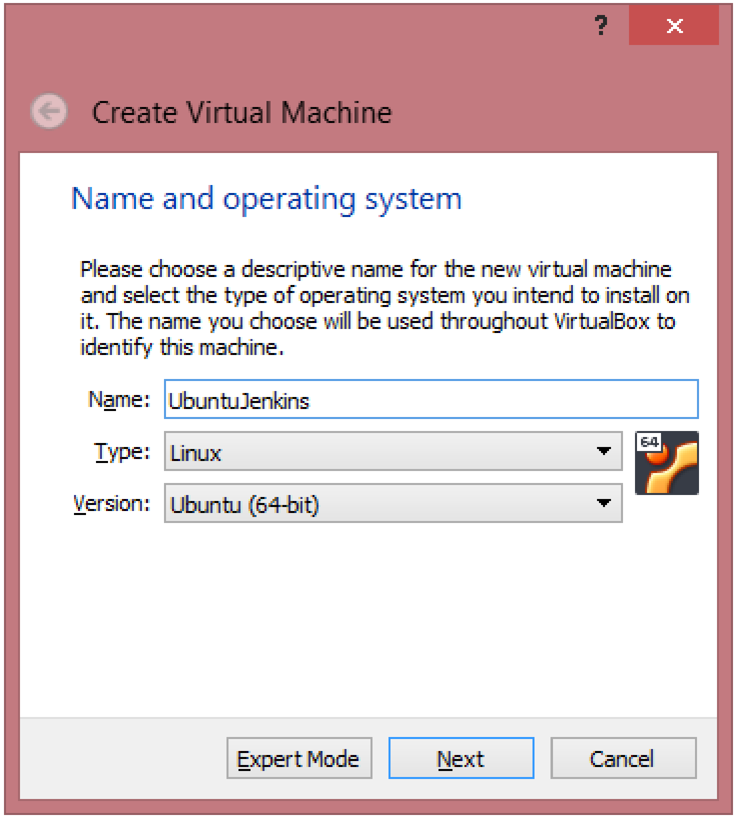
Tool to observe virtual machine (VM) details. Reflects groups of properties for the currently chosen VM and allows basic operations on certain properties (like the machine storage devices).

Snapshots

Tool to control virtual machine (VM) snapshots. Reflects snapshots created for the currently selected VM and allows snapshot operations like create, remove, restore (make current) and observe their properties. Allows to edit snapshot attributes like name and description.

Switch to machine tools

vmbox installation



Create Virtual Machine

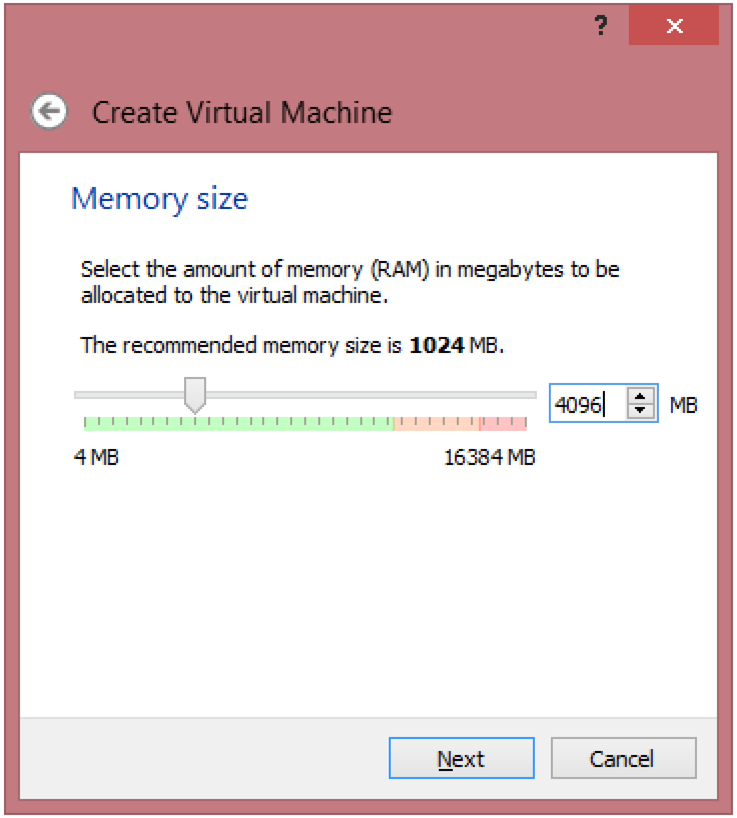
Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: UbuntuJenkins

Type: Linux

Version: Ubuntu (64-bit)



Create Virtual Machine

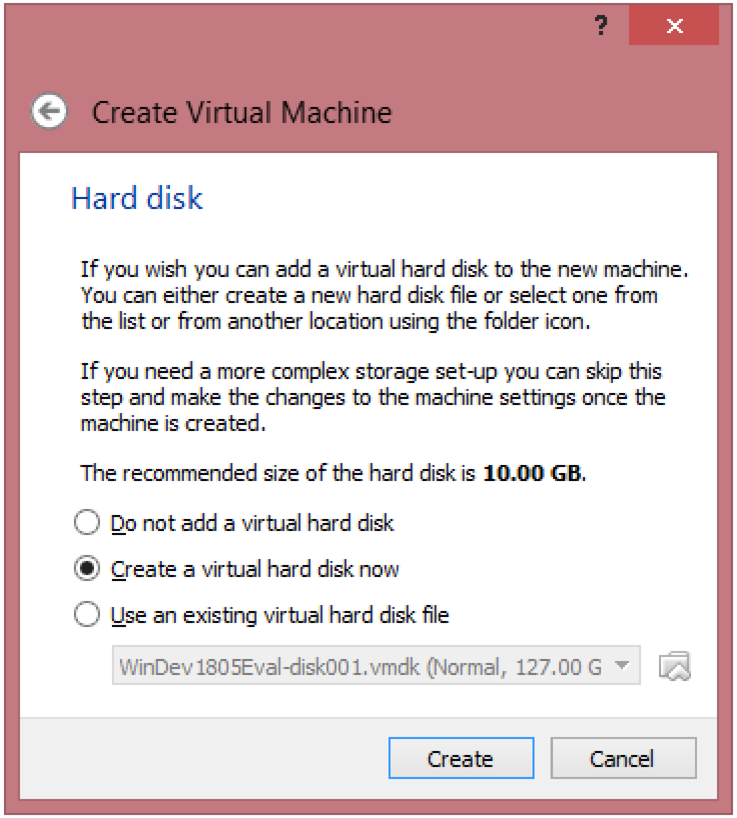
Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

4096 MB

vmbox installation



Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

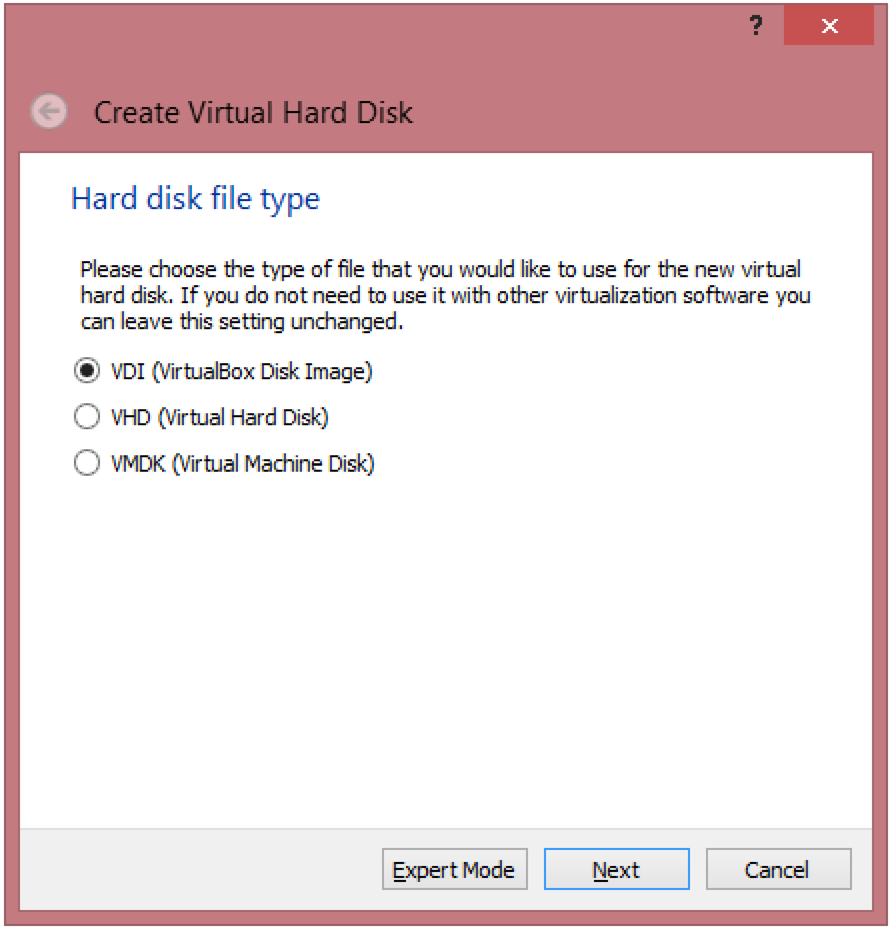
The recommended size of the hard disk is **10.00 GB**.

○ Do not add a virtual hard disk

● Create a virtual hard disk now

○ Use an existing virtual hard disk file

WinDev1805Eval-disk001.vmdk (Normal, 127.00 G



Create Virtual Hard Disk

Hard disk file type

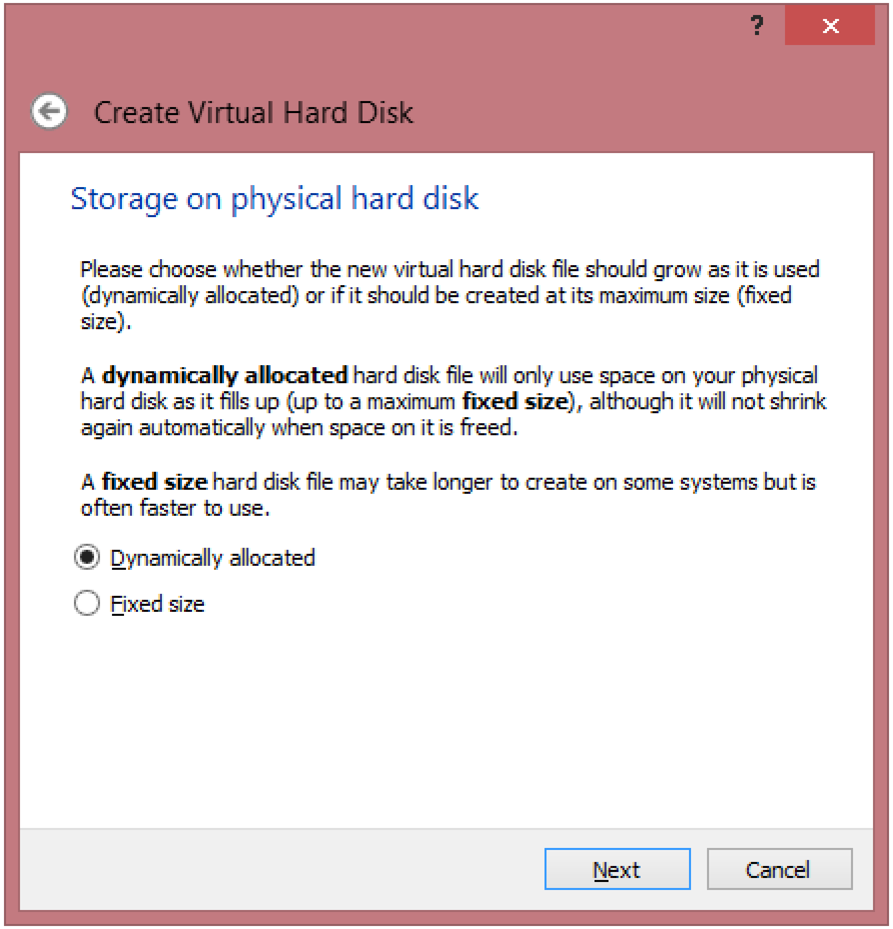
Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

● VDI (VirtualBox Disk Image)

○ VHD (Virtual Hard Disk)

○ VMDK (Virtual Machine Disk)

vmbox installation



Create Virtual Hard Disk

Storage on physical hard disk

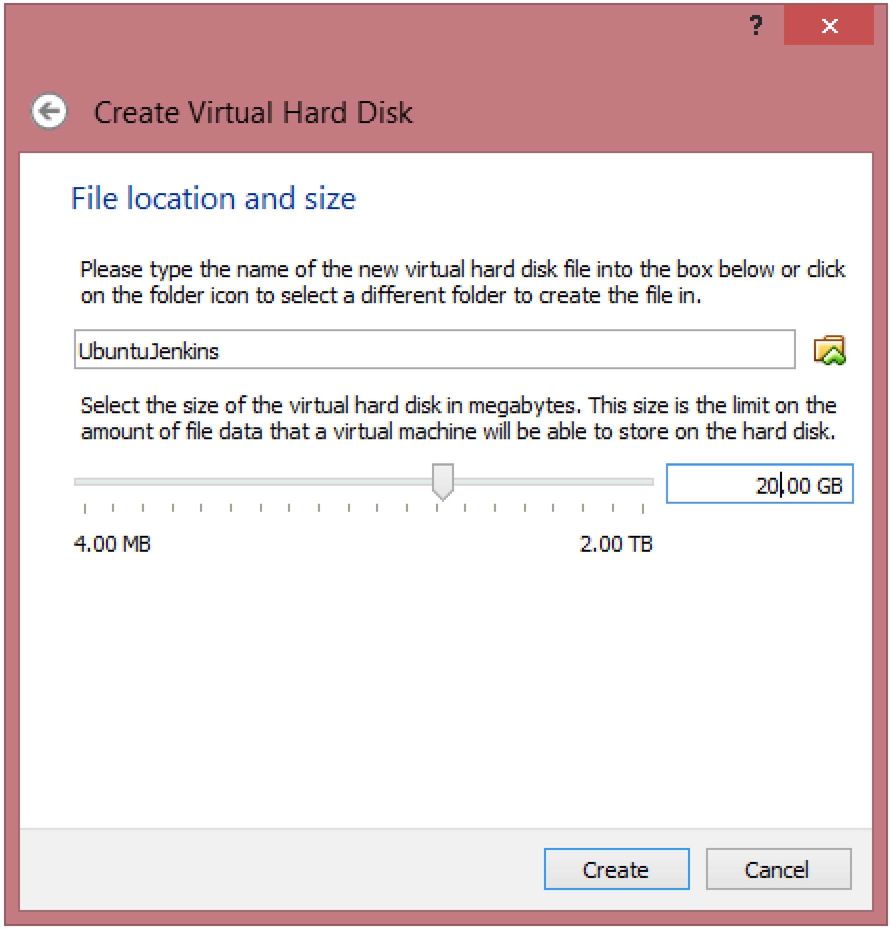
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

● Dynamically allocated

○ Fixed size



Create Virtual Hard Disk

File location and size

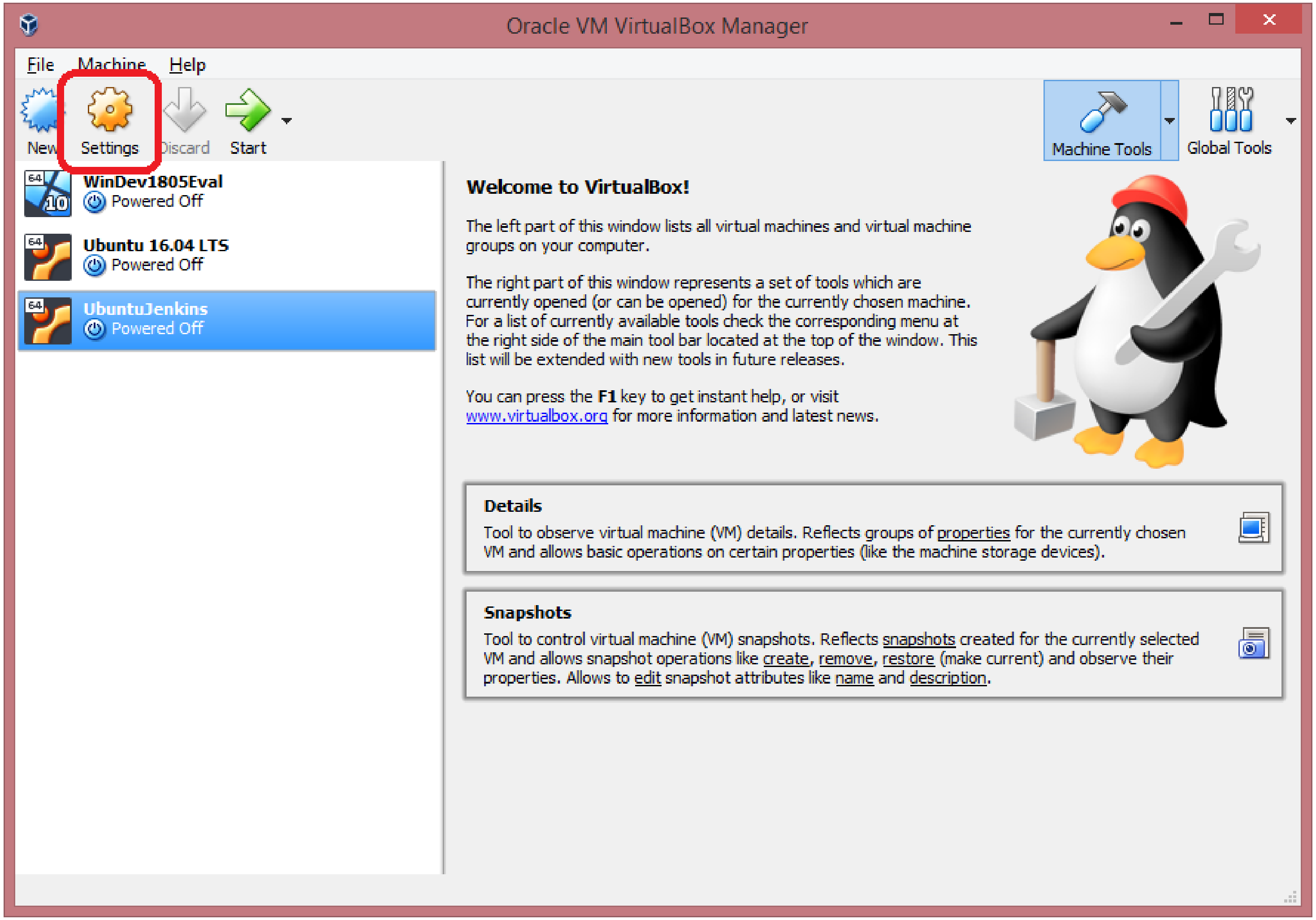
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

UbuntuJenkins

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

20.00 GB

vmbox installation



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer.

The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Details

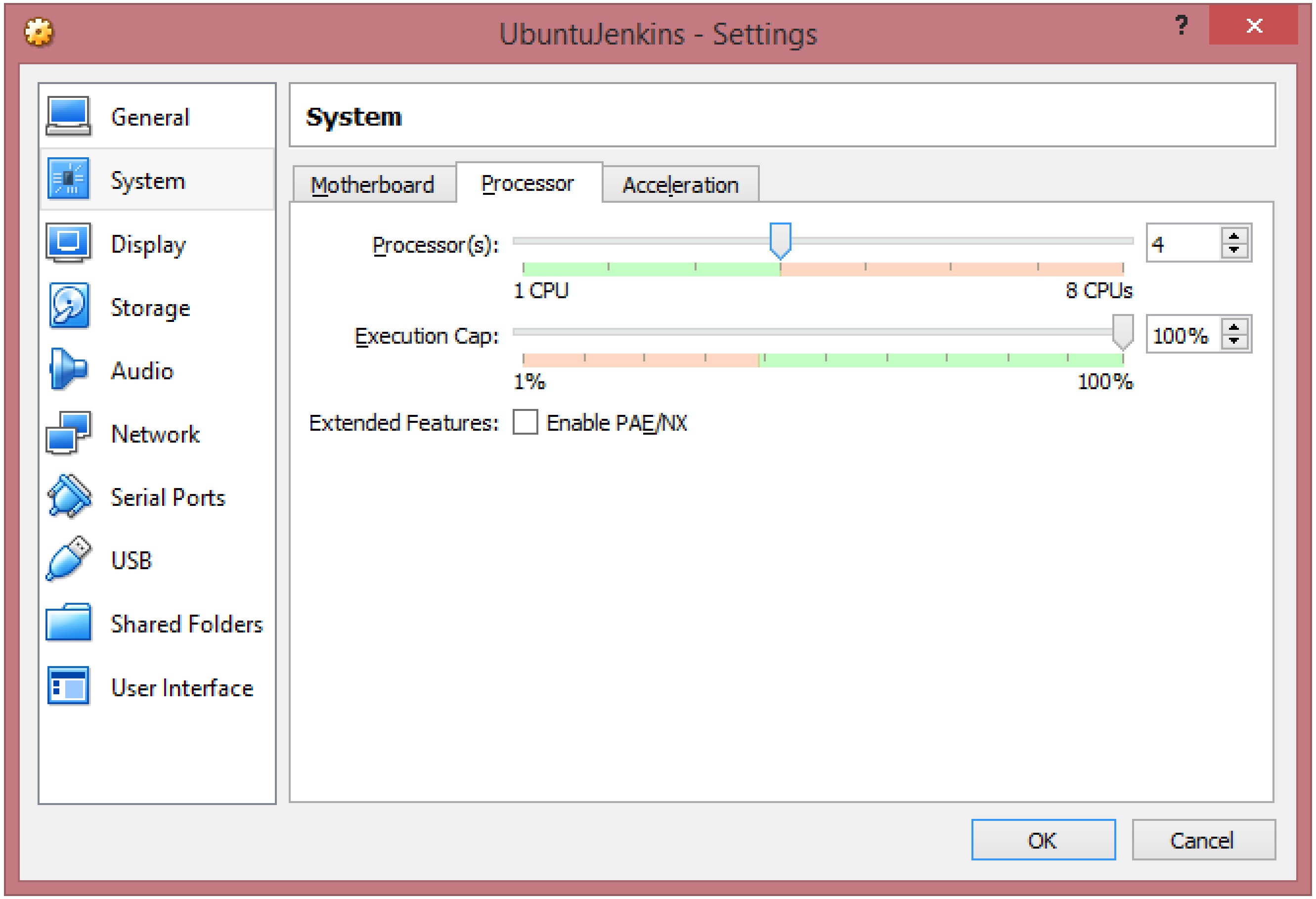
Tool to observe virtual machine (VM) details. Reflects groups of properties for the currently chosen VM and allows basic operations on certain properties (like the machine storage devices).

Snapshots

Tool to control virtual machine (VM) snapshots. Reflects snapshots created for the currently selected VM and allows snapshot operations like create, remove, restore (make current) and observe their properties. Allows to edit snapshot attributes like name and description.

vmbox installation

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



UbuntuJenkins - Settings

System

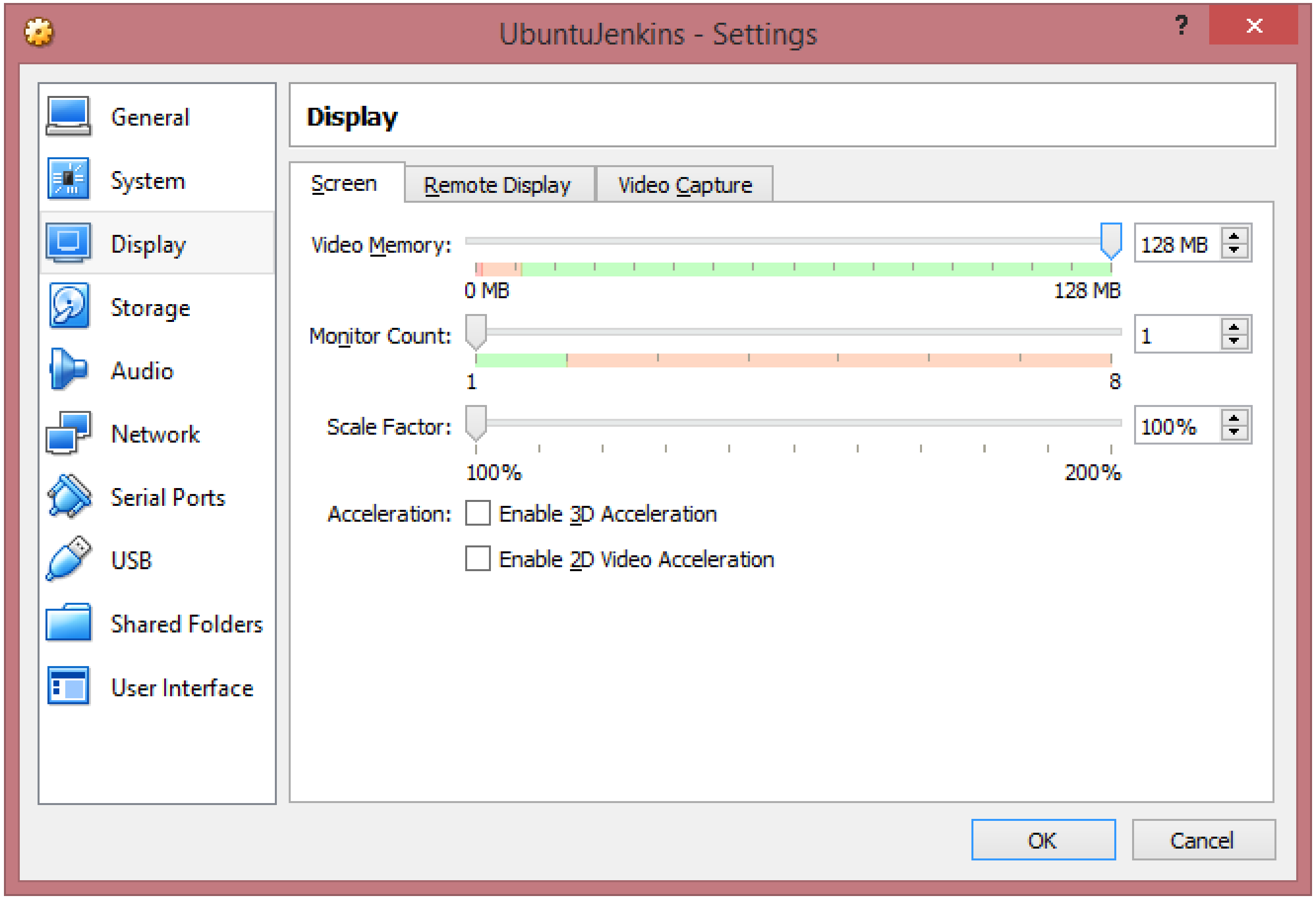
Processor(s): 4

Execution Cap: 100%

Extended Features: □ Enable PAE/NX

vmbox installation

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



UbuntuJenkins - Settings

General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface

Display

Screen | Remote Display | Video Capture

Video Memory: 128 MB (0 MB – 128 MB)

Monitor Count: 1 (1 – 8)

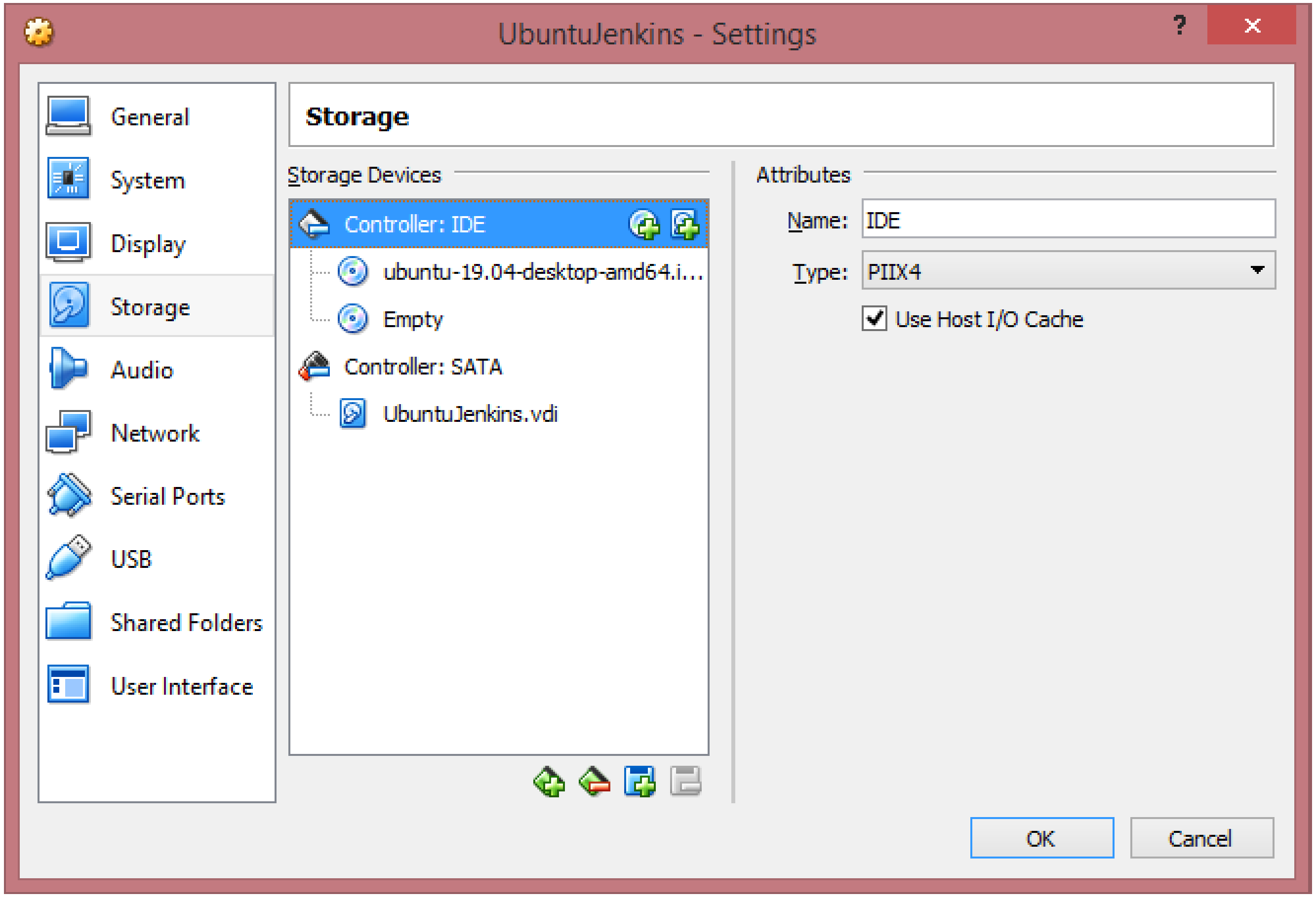
Scale Factor: 100% (100% – 200%)

Acceleration: □ Enable 3D Acceleration

□ Enable 2D Video Acceleration

OK | Cancel

vmbox installation



UbuntuJenkins - Settings

Storage

Storage Devices

- Controller: IDE
  - ubuntu-19.04-desktop-amd64.i...
  - Empty
- Controller: SATA
  - UbuntuJenkins.vdi

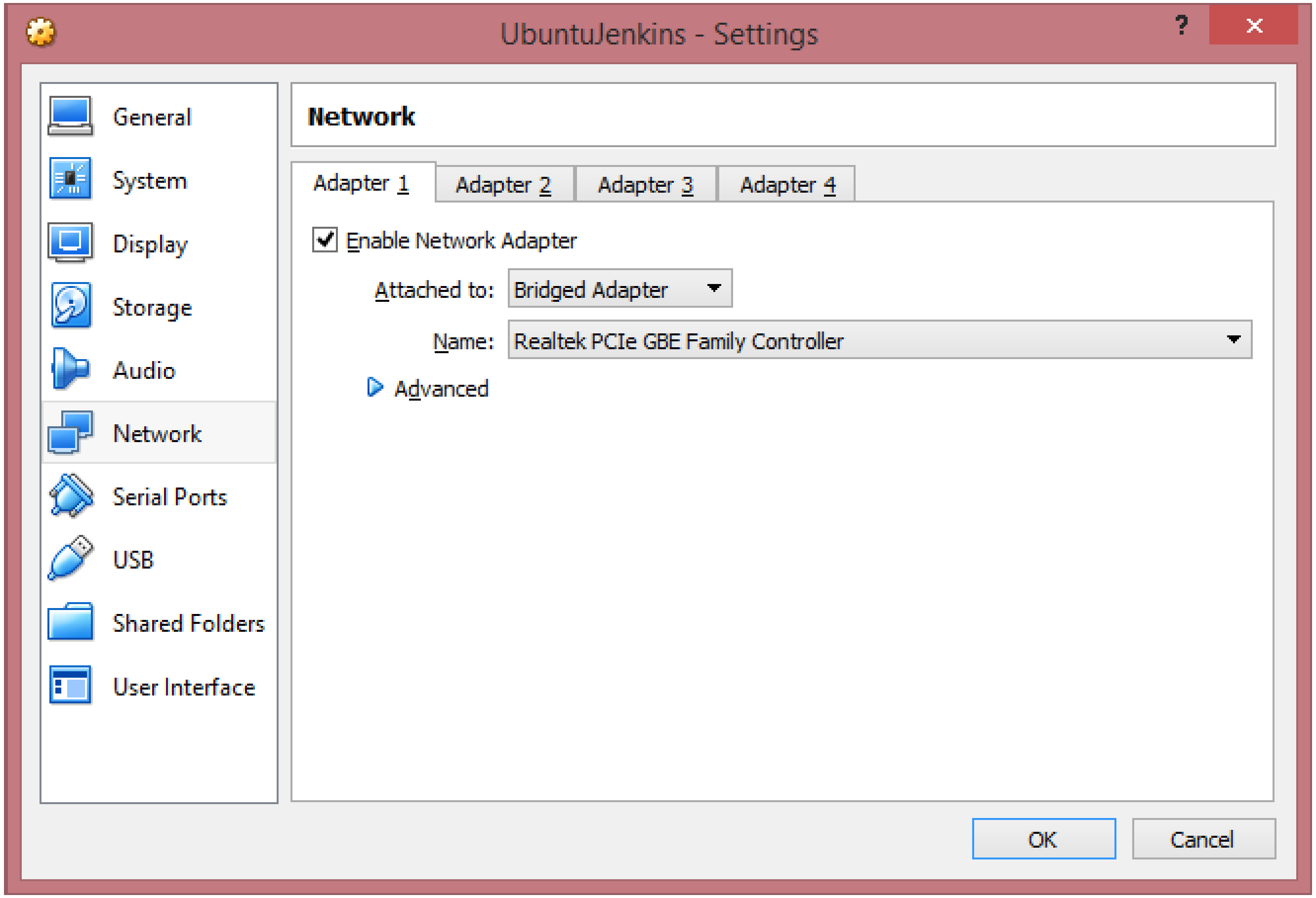
Attributes

Name: IDE

Type: PIIX4

☑ Use Host I/O Cache

vmbox installation


UbuntuJenkins - Settings

Network

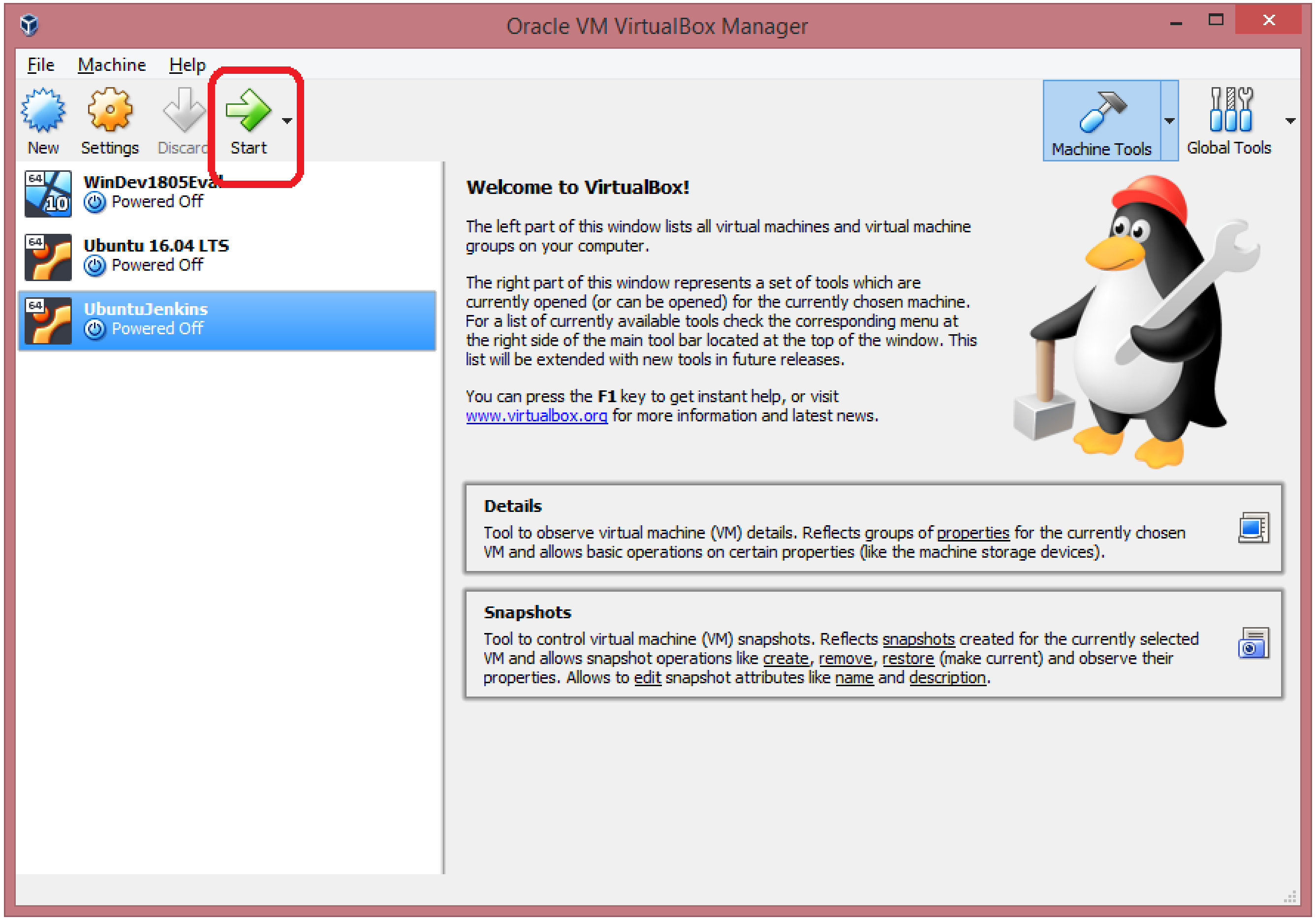
☑ Enable Network Adapter

Attached to: Bridged Adapter

Name: Realtek PCIe GBE Family Controller

vmbox installation

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer.

The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

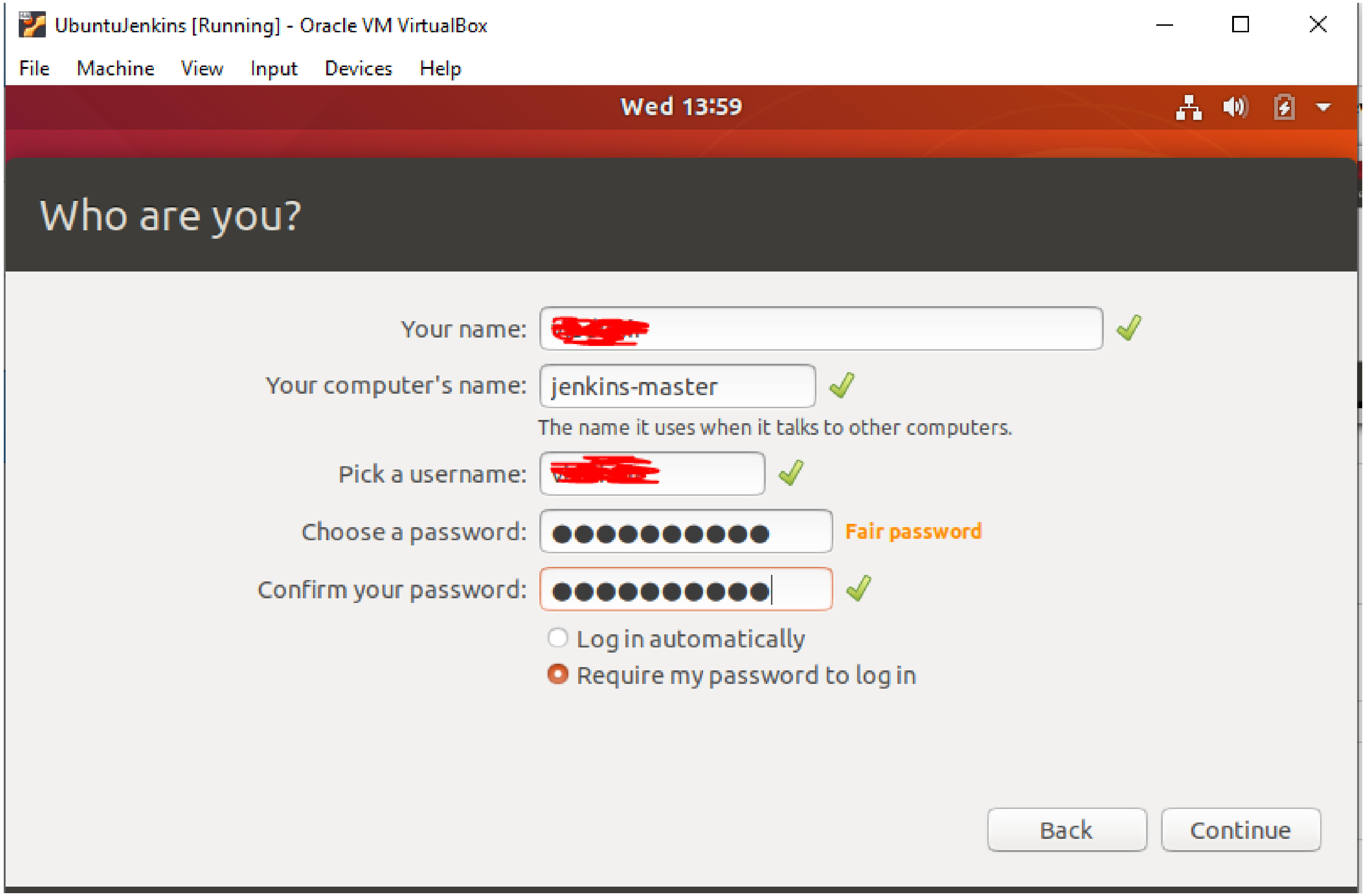
Details

Tool to observe virtual machine (VM) details. Reflects groups of properties for the currently chosen VM and allows basic operations on certain properties (like the machine storage devices).

Snapshots

Tool to control virtual machine (VM) snapshots. Reflects snapshots created for the currently selected VM and allows snapshot operations like create, remove, restore (make current) and observe their properties. Allows to edit snapshot attributes like name and description.

vmbox installation



Who are you?

Your name: [redacted] ✓

Your computer's name: jenkins-master ✓

The name it uses when it talks to other computers.

Pick a username: [redacted] ✓

Choose a password: ●●●●●●●●●●● Fair password

Confirm your password: ●●●●●●●●●●● ✓

○ Log in automatically

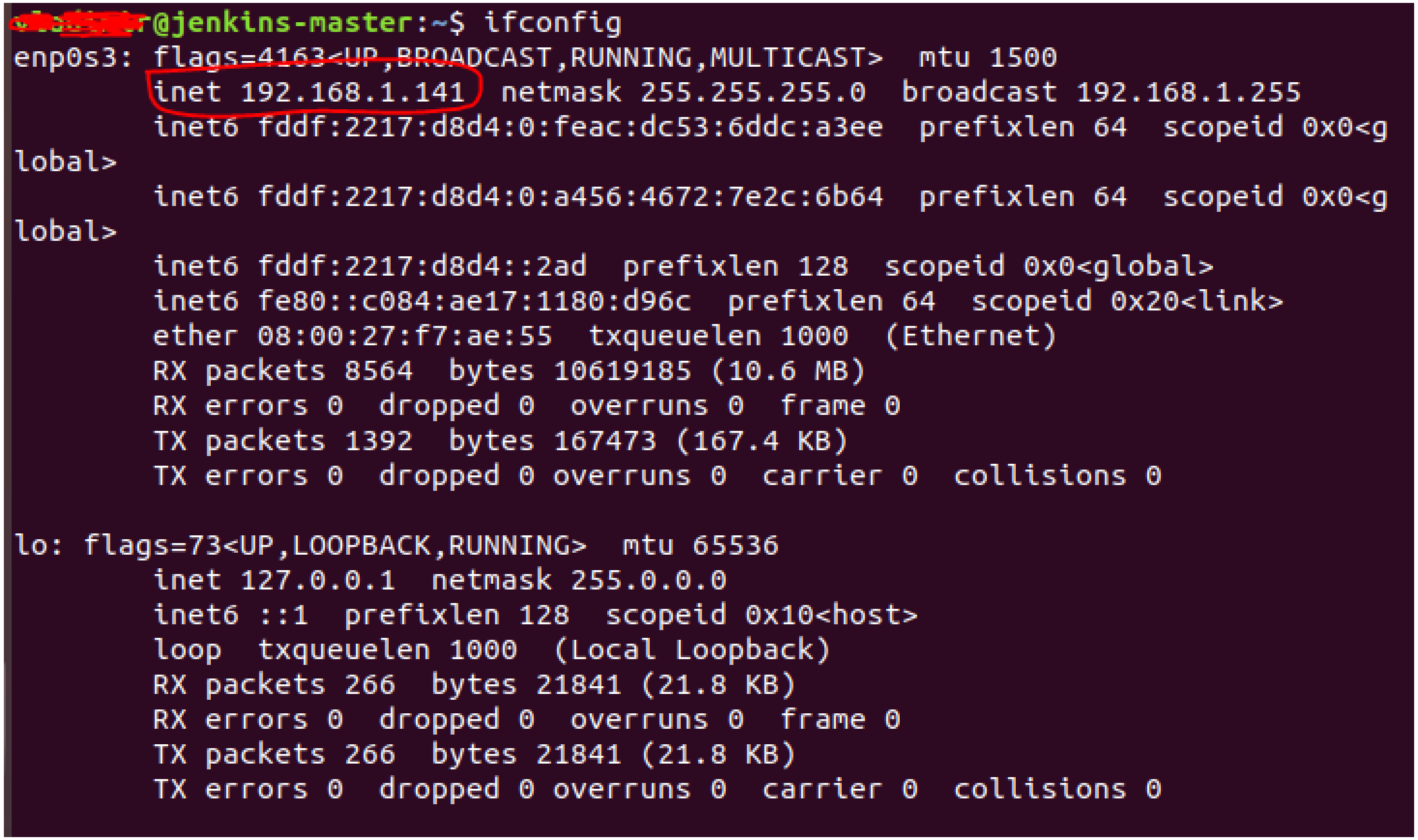
● Require my password to log in

What is Putty?

- PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

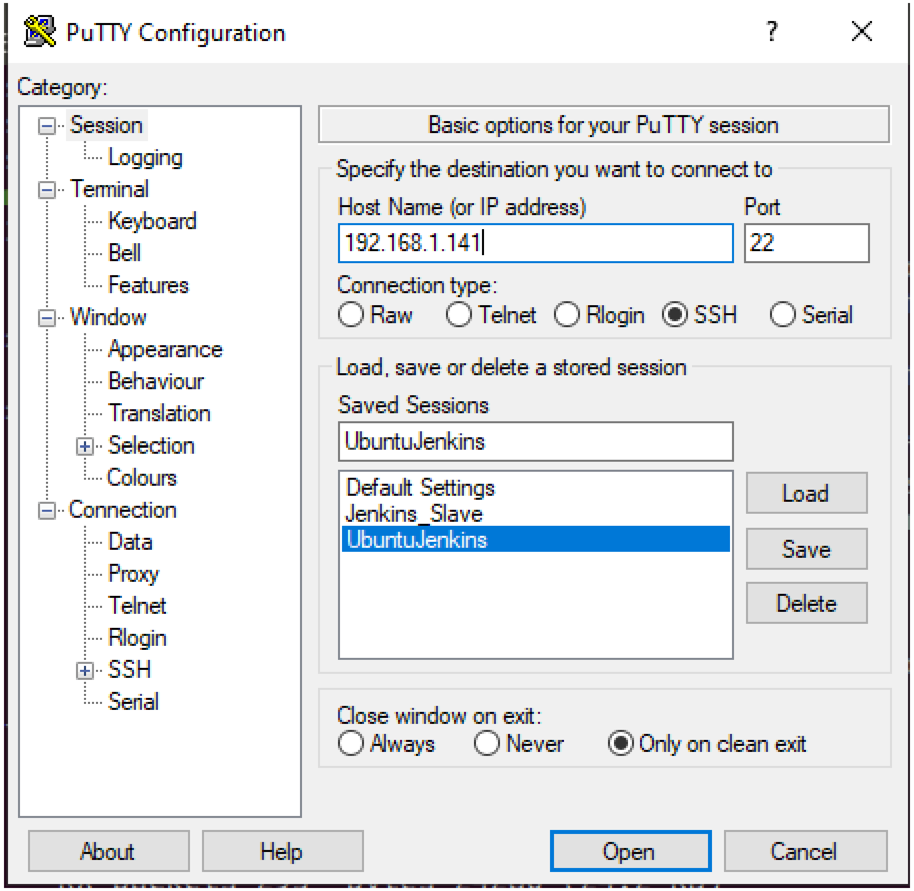
- To download putty just go to https://www.putty.org/
- then click on here and install it

vmbox installation



```
[redacted]@jenkins-master:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.141  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fddf:2217:d8d4:0:feac:dc53:6ddc:a3ee  prefixlen 64  scopeid 0x0<g
lobal>
        inet6 fddf:2217:d8d4:0:a456:4672:7e2c:6b64  prefixlen 64  scopeid 0x0<g
lobal>
        inet6 fddf:2217:d8d4::2ad  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::c084:ae17:1180:d96c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:f7:ae:55  txqueuelen 1000  (Ethernet)
        RX packets 8564  bytes 10619185 (10.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1392  bytes 167473 (167.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 266  bytes 21841 (21.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 266  bytes 21841 (21.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

vmbox installation



PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.141

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: UbuntuJenkins

Default Settings
Jenkins_Slave
UbuntuJenkins

Close window on exit: ○ Always ○ Never ● Only on clean exit

vmbox installation

- Just enter your username and password
- and then we will work with putty to connect to the virtual machine
- Note that the vmbox must work all the time
- the vmbox must be connected to the internet
- you can test it by using ping google.com

- Note: if you get fatal error, type this
- in your linux machine
- sudo apt-get install openssh-server openssh-client

Jenkins Installation

Jenkins installation

- First need to install Java

```
$ sudo apt install openjdk-8-jdk
```

- Check your Java version

```
$ java –version
```

- Implement the commands below:

```
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list’
$ sudo apt-get update
$ sudo apt-get install jenkins
```

Jenkins installation

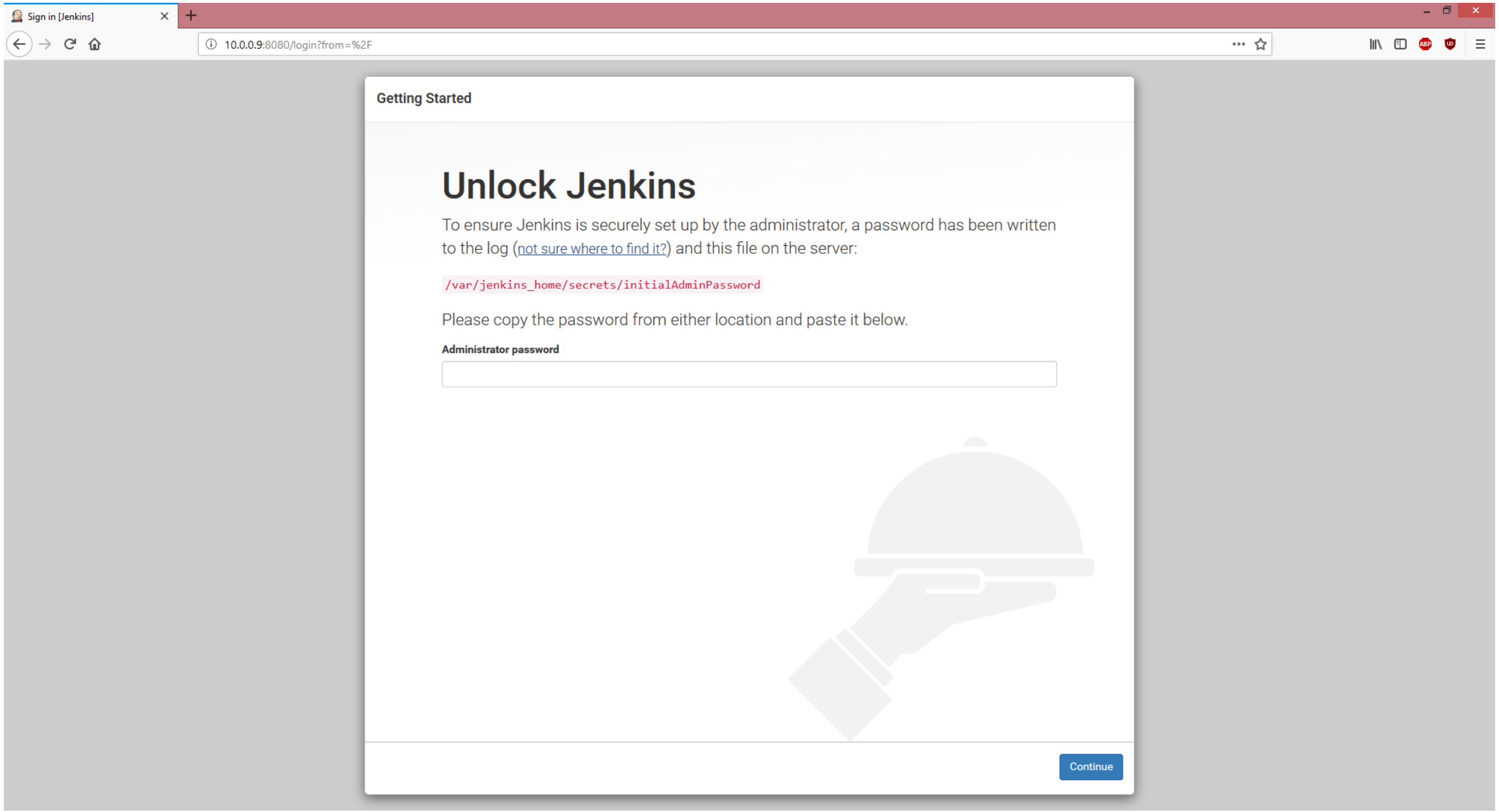
- **Power Off** your VM and then **Power On**
- Find your VM IP by command **ifconfig**
- Browse in Chrome

  http://<your_IP_address>:8080

  Example: *http://192.168.1.221:8080/*

Jenkins config

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



Getting Started

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:

```
/var/jenkins_home/secrets/initialAdminPassword
```

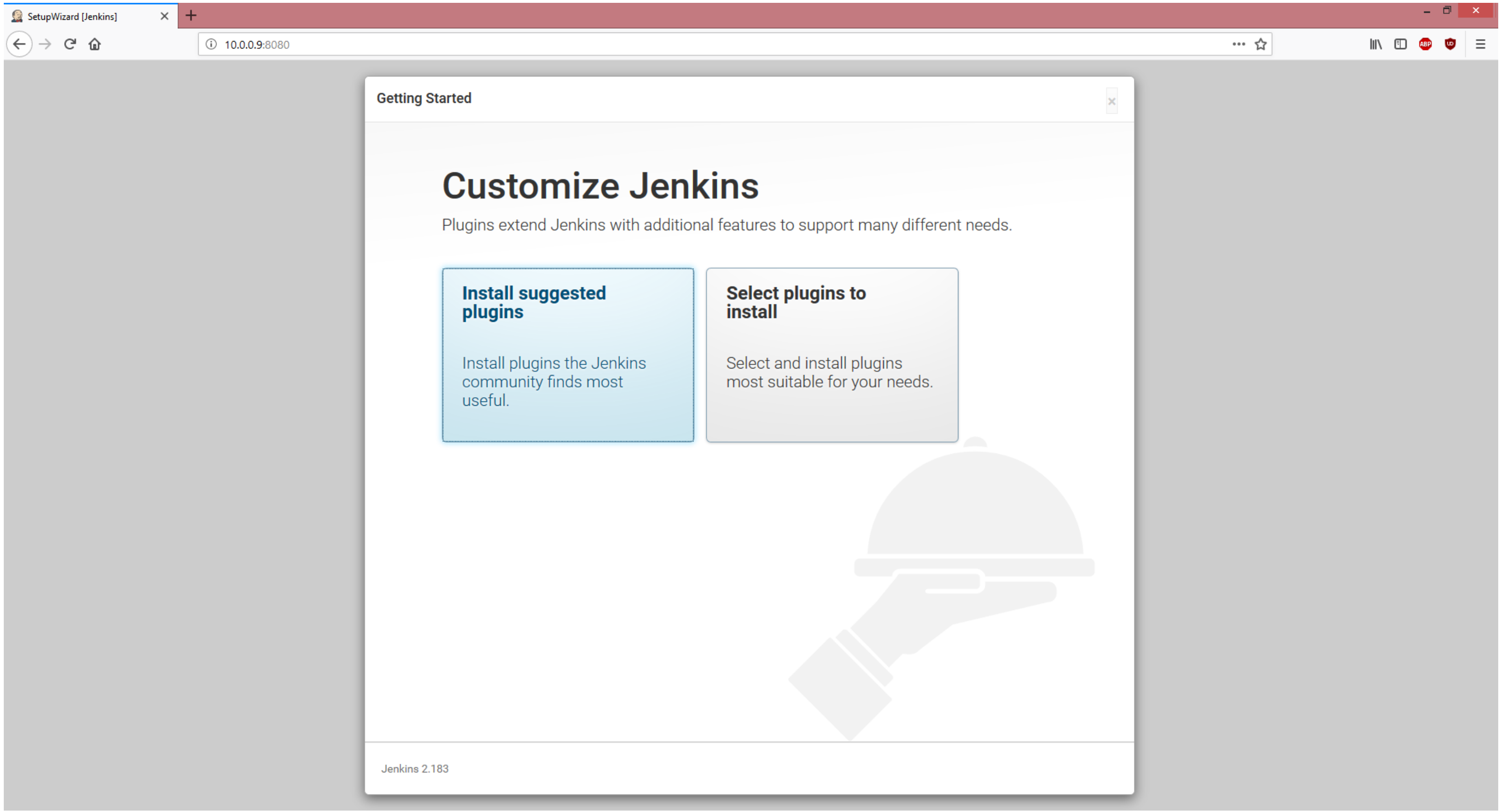
Please copy the password from either location and paste it below.

Administrator password

Continue

Jenkins config

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

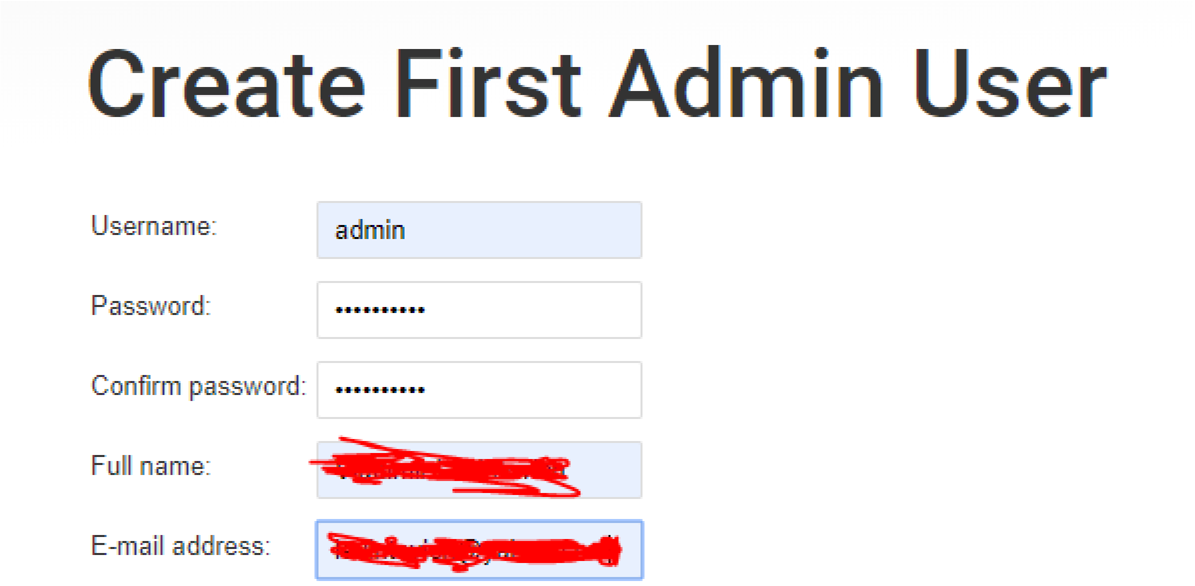
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.183

Jenkins config

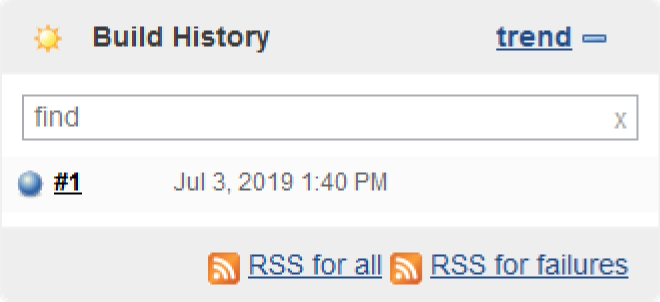


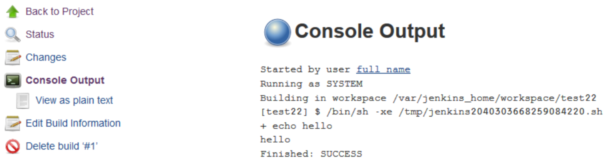
Create First Admin User

Username:	admin
Password:	•••••••••••
Confirm password:	•••••••••••
Full name:	[redacted]
E-mail address:	[redacted]

Jenkins first job

- go to dashboard > new item > fill the name of your job > freestyle project > ok
- build > add build step > execute shell > write your command in the text box > save
- when you in your project after you write some command to execute go to the navigation bar > build now then go to build history tab and choose the last execution
- then go to
- console output

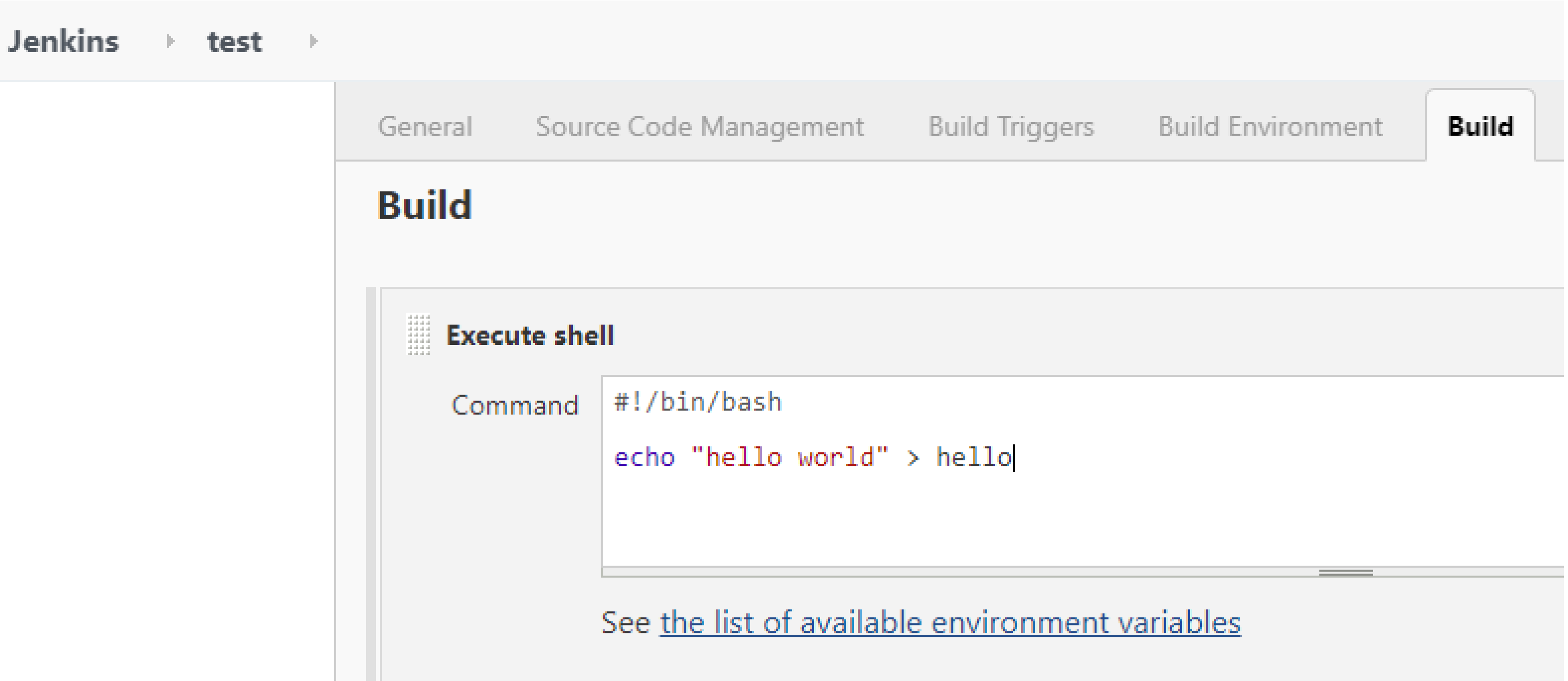




```
Started by user full name
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/test22
[test22] $ /bin/sh -xe /tmp/jenkins2040303668259084220.sh
+ echo hello
hello
Finished: SUCCESS
```

- to edit your job go to Configure

Hello World Job



Build

Execute shell

Command

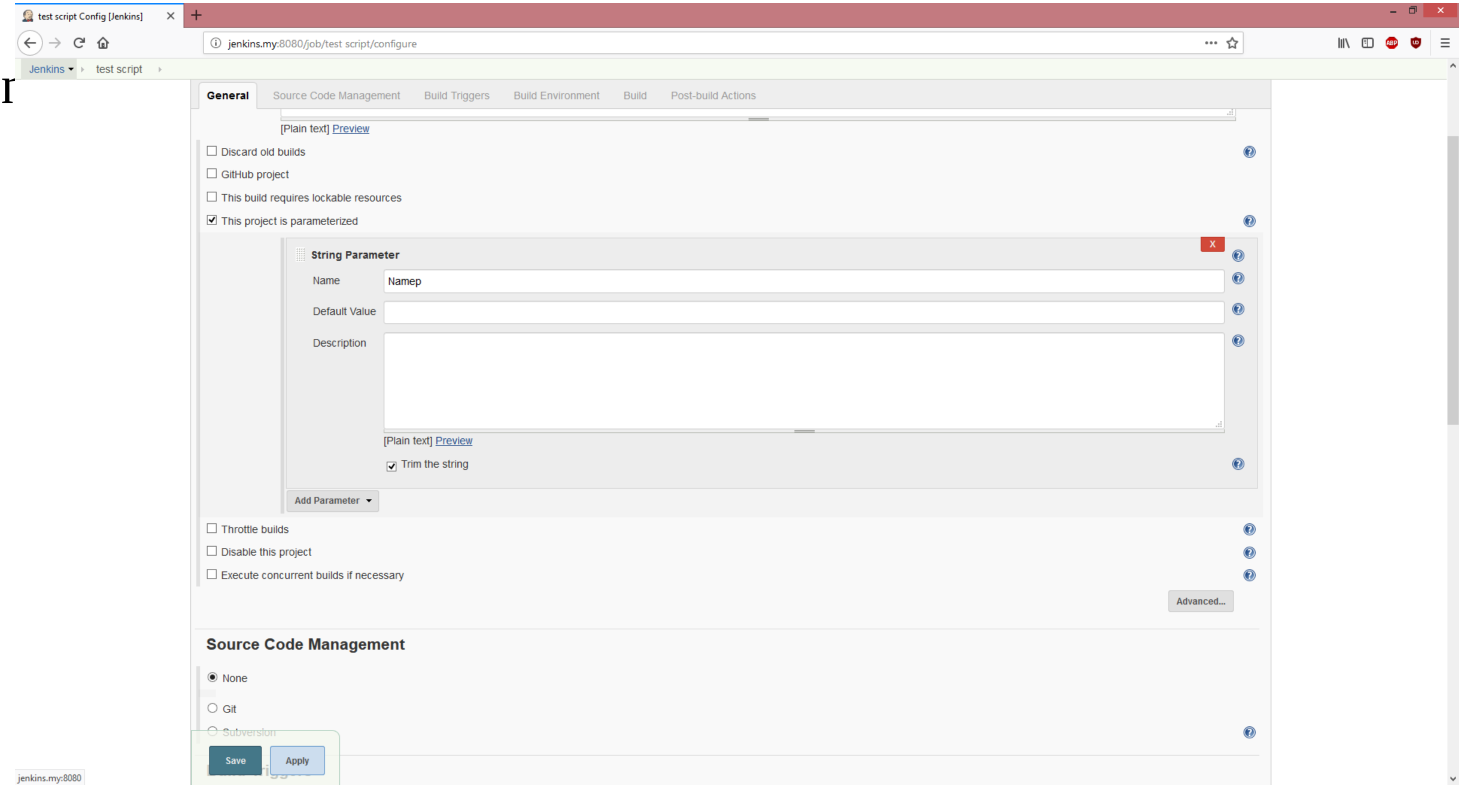
```
#!/bin/bash
echo "hello world" > hello
```

See the list of available environment variables

Jenkins adding parameters

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- go to Configure > general > this project is parameterized > add parameter
- after you fir
- press save



test script Config [Jenkins]

jenkins.my:8080/job/test script/configure

Jenkins ▸ test script ▸

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

[Plain text] Preview

☐ Discard old builds

☐ GitHub project

☐ This build requires lockable resources

☑ This project is parameterized

String Parameter

Name: Namep

Default Value:

Description:

[Plain text] Preview

☑ Trim the string

Add Parameter ▾

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

Advanced...

Source Code Management

● None

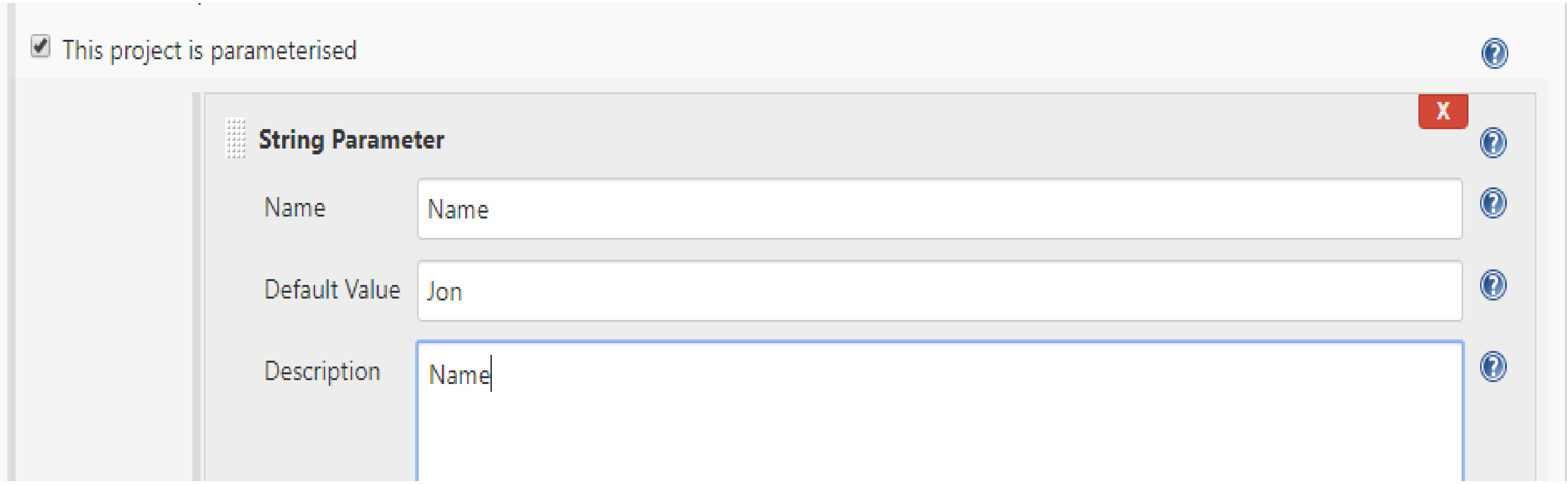
○ Git

○ Subversion

Save | Apply

jenkins.my:8080

Jenkins adding parameters



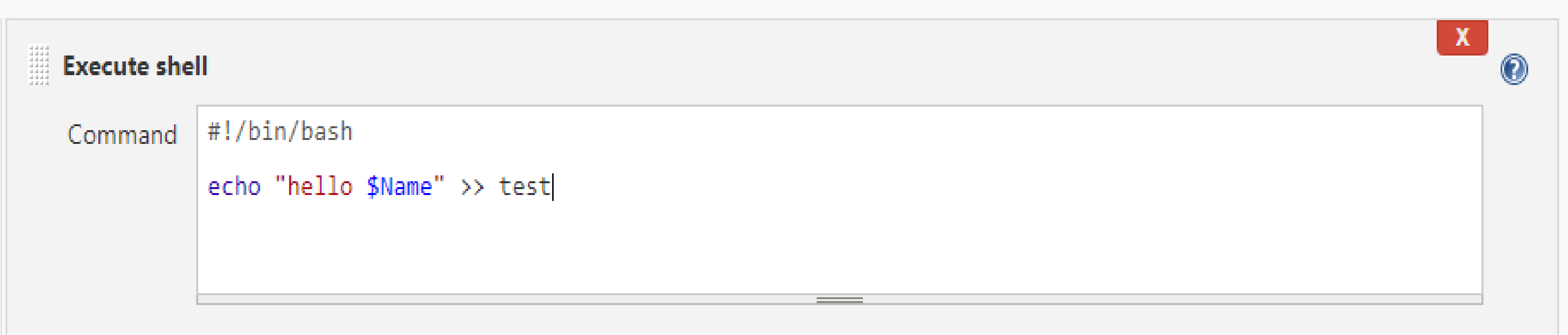
☑ This project is parameterised

String Parameter

Name: Name

Default Value: Jon

Description: Name



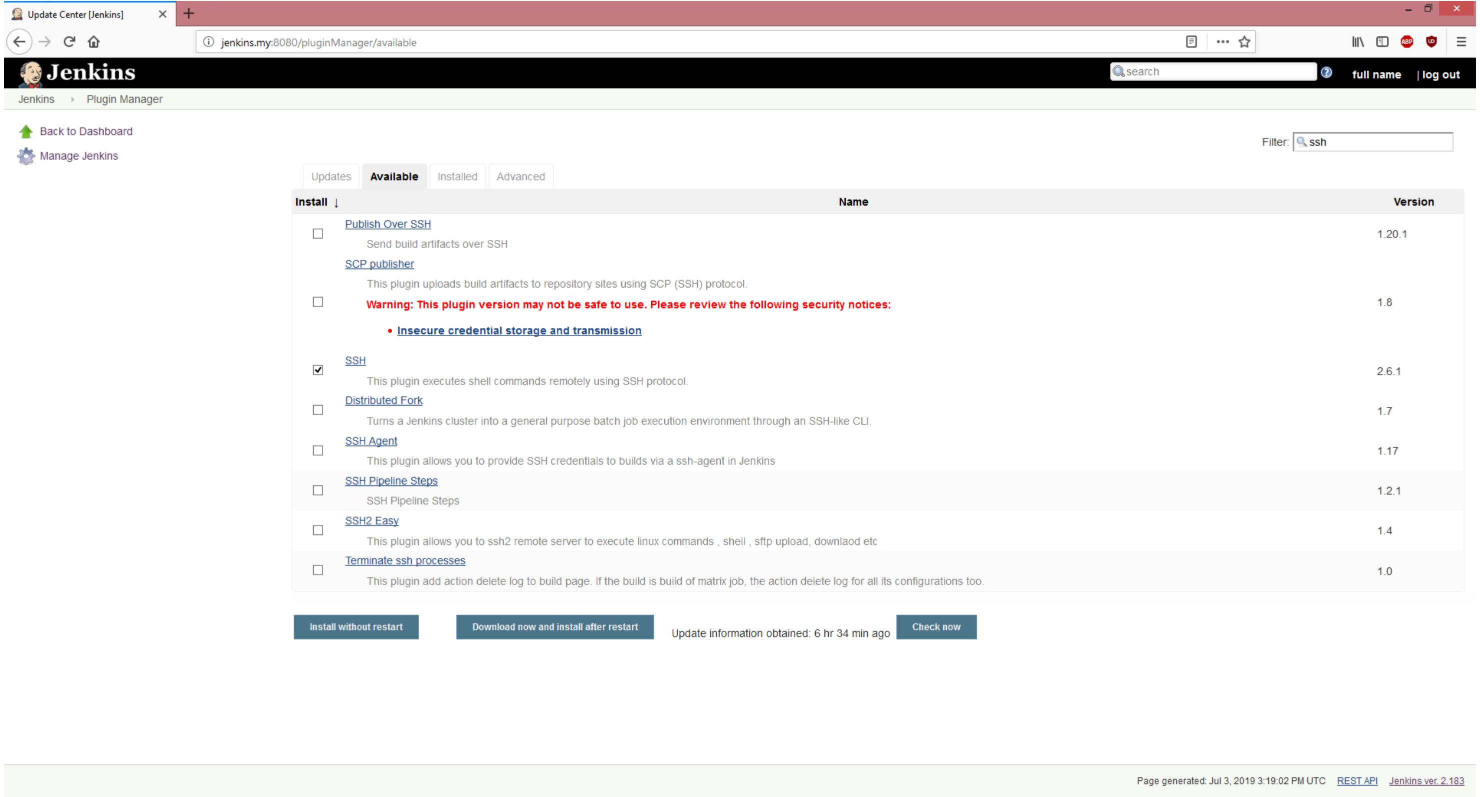
Execute shell

Command:

```
#!/bin/bash
echo "hello $Name" >> test
```

Jenkins installing plugins

- go to dashboard > manage Jenkins > manage plugins > available > choose the plugin that you need to install and then choose > install without restart



Filter: ssh

Install ↓	Name	Version
□	Publish Over SSH Send build artifacts over SSH	1.20.1
□	SCP publisher This plugin uploads build artifacts to repository sites using SCP (SSH) protocol. **Warning: This plugin version may not be safe to use. Please review the following security notices:** • Insecure credential storage and transmission	1.8
☑	SSH This plugin executes shell commands remotely using SSH protocol.	2.6.1
□	Distributed Fork Turns a Jenkins cluster into a general purpose batch job execution environment through an SSH-like CLI.	1.7
□	SSH Agent This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins	1.17
□	SSH Pipeline Steps SSH Pipeline Steps	1.2.1
□	SSH2 Easy This plugin allows you to ssh2 remote server to execute linux commands , shell , sftp upload, downlaod etc	1.4
□	Terminate ssh processes This plugin add action delete log to build page. If the build is build of matrix job, the action delete log for all its configurations too.	1.0

Update information obtained: 6 hr 34 min ago

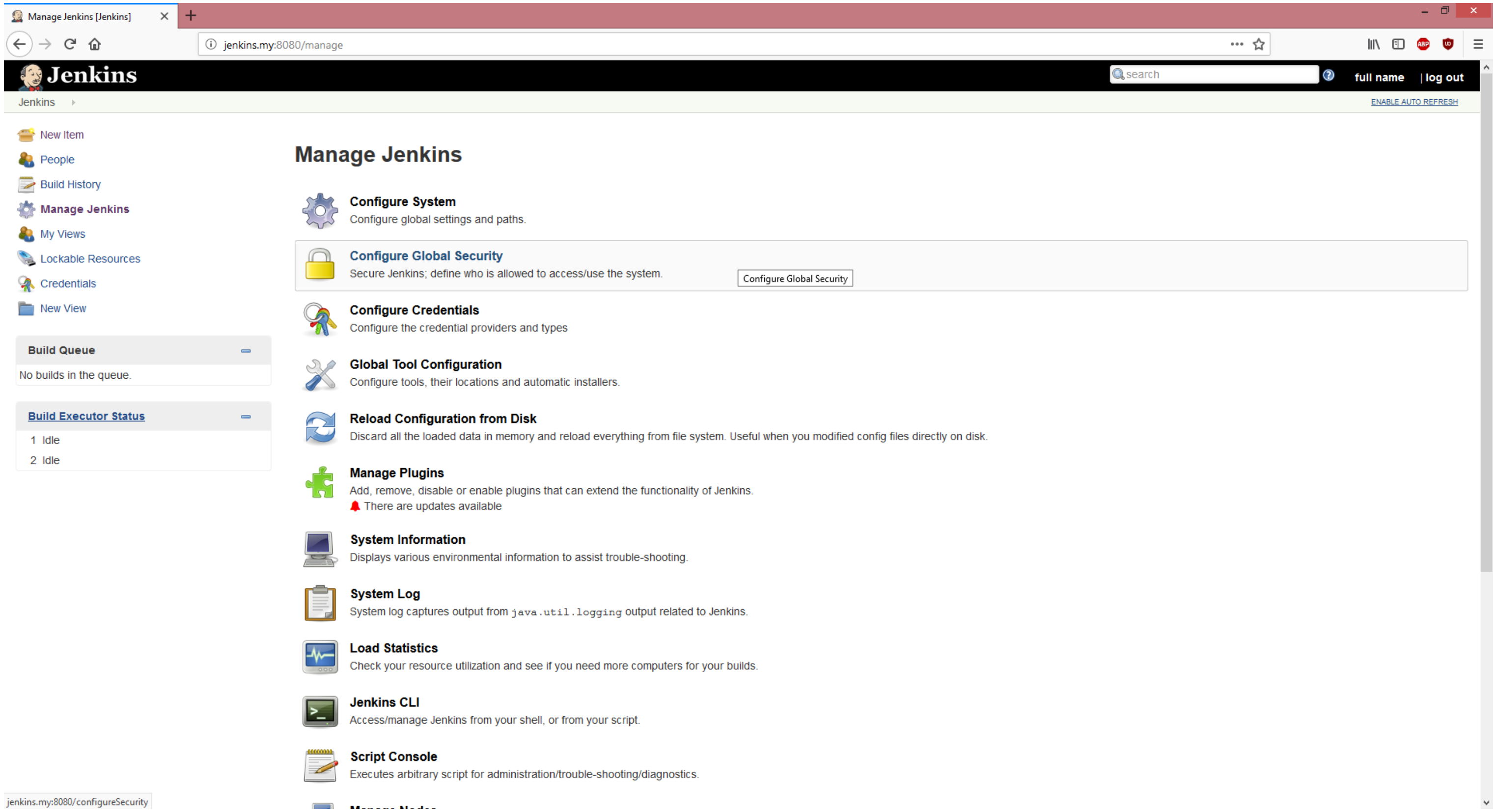
Page generated: Jul 3, 2019 3:19:02 PM UTC

- make sure that you restart the server after this

Jenkins Security

Jenkins Security

- go to dashboard > Manage Jenkins > Configure Global Security



No builds in the queue.

1 Idle
2 Idle

Manage Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

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 you modified config files directly 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.

System Log
System log captures output from `java.util.logging` output related to Jenkins.

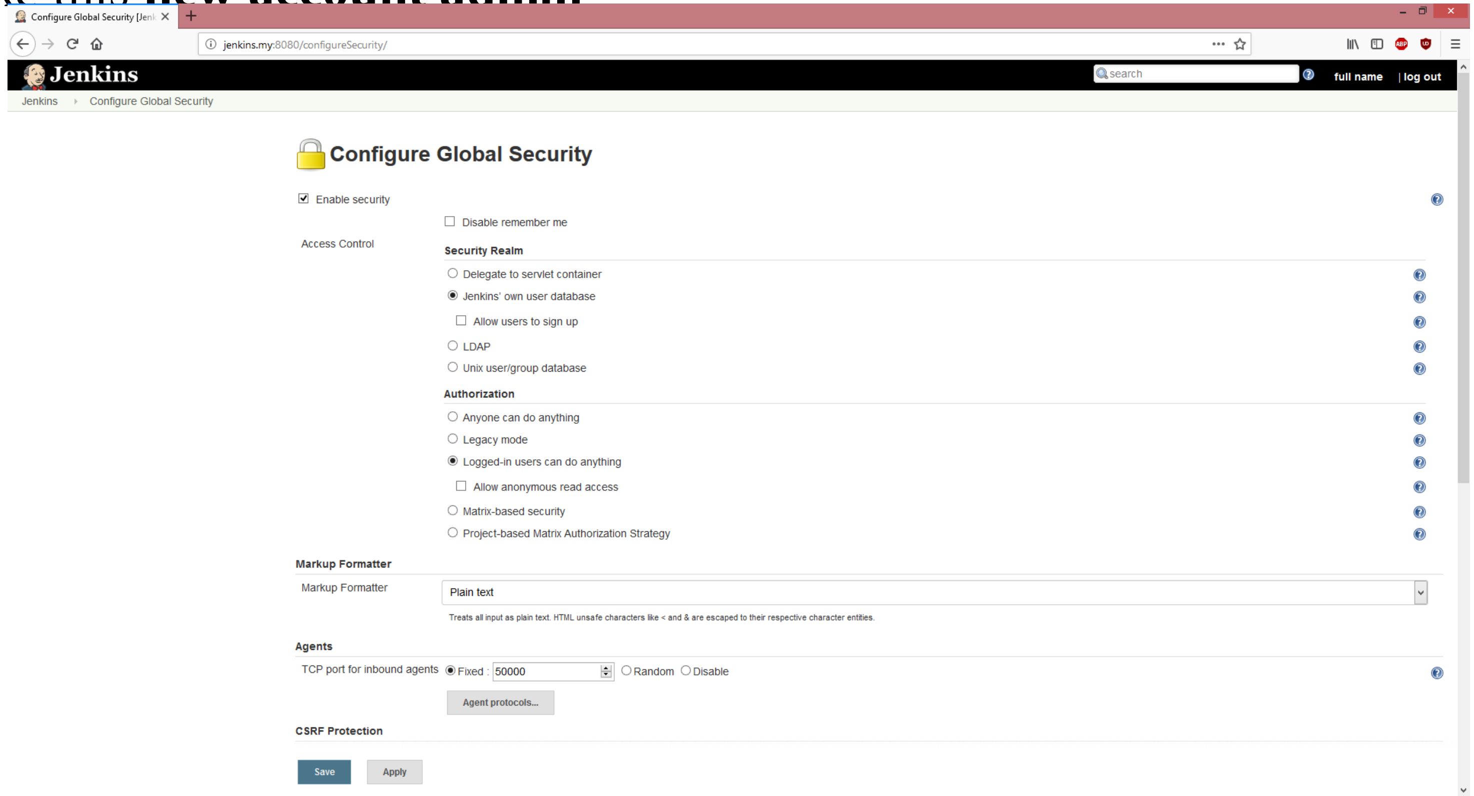
Load Statistics
Check your resource utilization and see if you need more computers for your builds.

Jenkins CLI
Access/manage Jenkins from your shell, or from your script.

Script Console
Executes arbitrary script for administration/trouble-shooting/diagnostics.

Jenkins Security

- You have the option *Allow users to sign up* this option will give anyone who want to connect to your Jenkins create an account and by default it will make this **new account admin**



Configure Global Security [Jenk

jenkins.my:8080/configureSecurity/

Jenkins › Configure Global Security

Configure Global Security

☑ Enable security

☐ Disable remember me

Access Control

Security Realm

○ Delegate to servlet container
● Jenkins' own user database
☐ Allow users to sign up
○ LDAP
○ Unix user/group database

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

Markup Formatter

Markup Formatter: Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

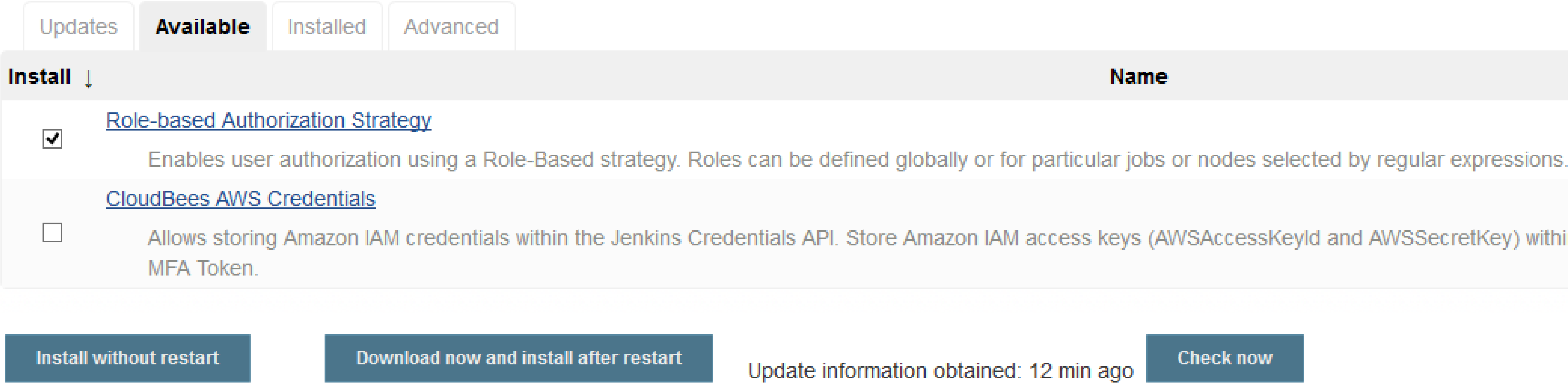
Agents

TCP port for inbound agents ● Fixed : 50000 ○ Random ○ Disable

CSRF Protection

Jenkins Security

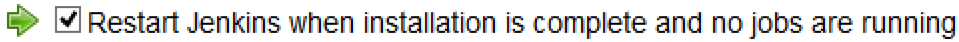
- This is why we want to install plugin that will help us to fix this problem
- go to your dashboard > Manage Jenkins > Manage Plugins > Available
- find plugin name *Role-based Authorization Strategy* > install without restart



Install ↓	Name
☑	Role-based Authorization Strategy Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.
□	CloudBees AWS Credentials Allows storing Amazon IAM credentials within the Jenkins Credentials API. Store Amazon IAM access keys (AWSAccessKeyId and AWSSecretKey) withi MFA Token.

Update information obtained: 12 min ago

- in the next screen make sure to choose to restart

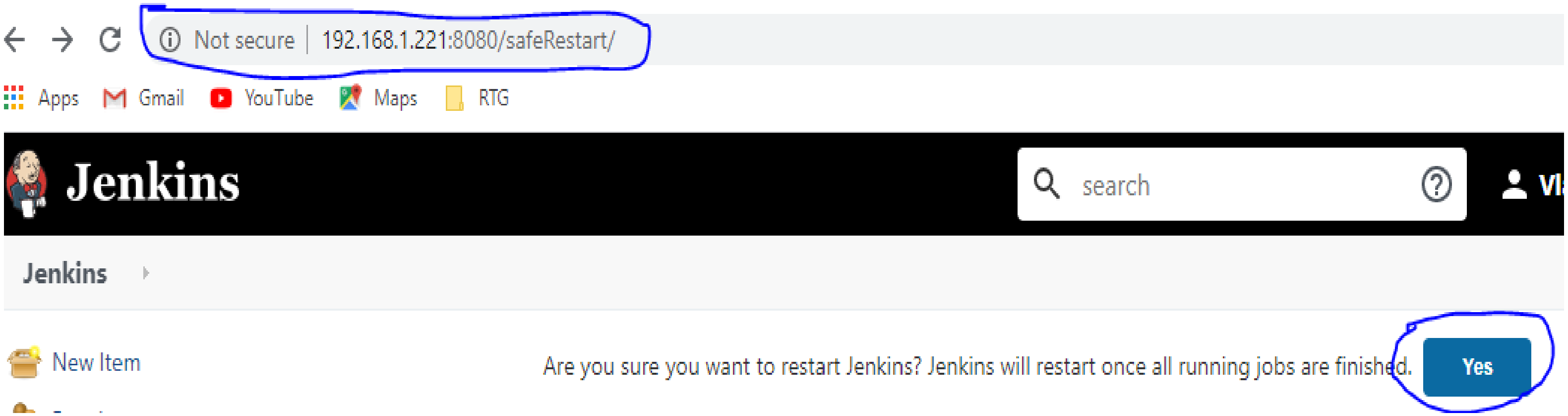


☑ Restart Jenkins when installation is complete and no jobs are running

Jenkins Restart

There are 2 options to restart Jenkins:

1) (jenkins_url)/safeRestart - Allows all running jobs to complete
2) (jenkins_url)/restart - Forces a restart without waiting for builds to complete

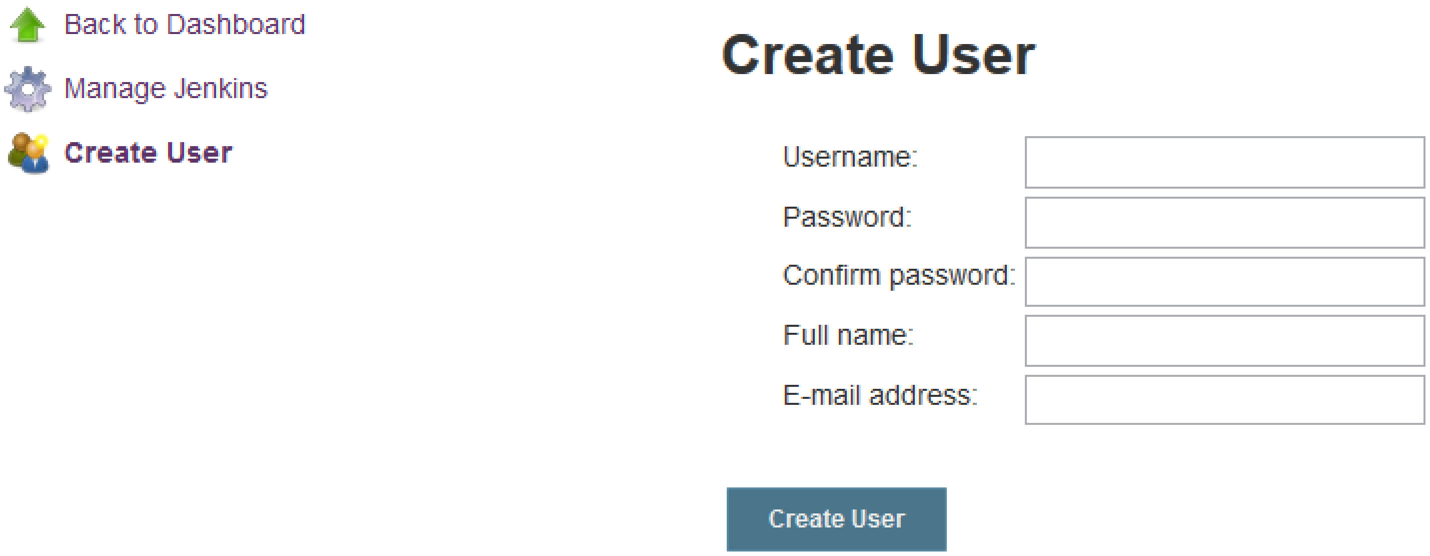


Not secure | 192.168.1.221:8080/safeRestart/

Are you sure you want to restart Jenkins? Jenkins will restart once all running jobs are finished.

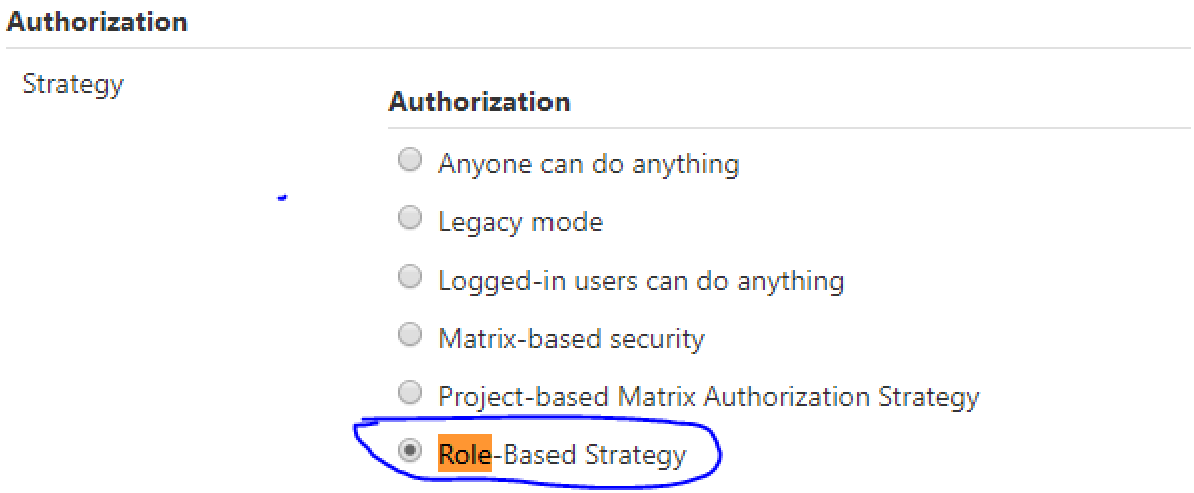
Jenkins Security

- Go to dashboard > manage Jenkins > Manage users
- first we must create new user on your left side you will see create user press on it then you will get new screen fill the form and press on create user



Jenkins Security

- Go to dashboard > manage Jenkins > Configure Global Security
- Find and select **Role-Based Strategy**



Authorization

Strategy

Authorization

○ Anyone can do anything

○ Legacy mode

○ Logged-in users can do anything

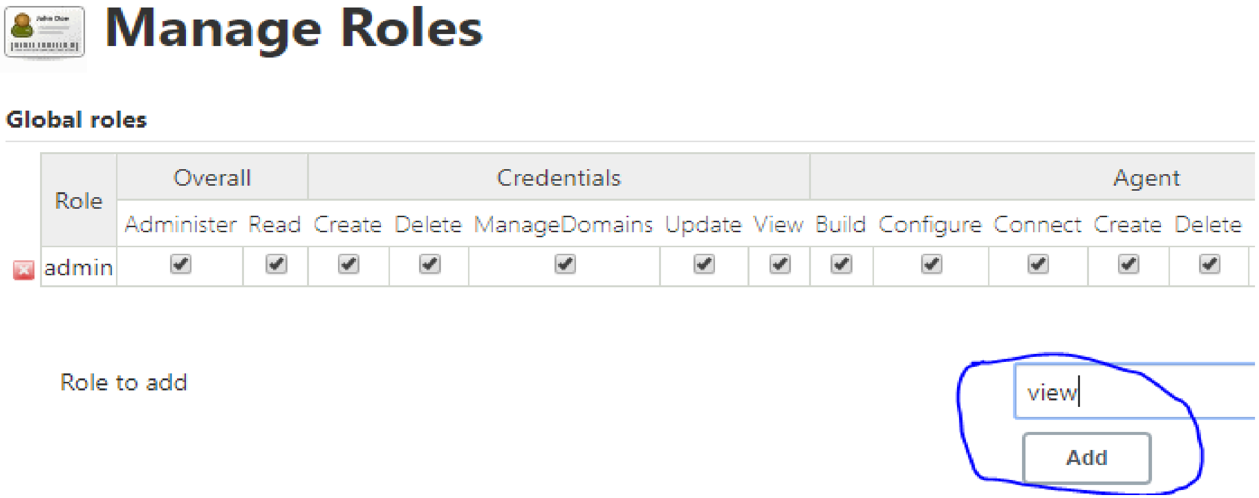
○ Matrix-based security

○ Project-based Matrix Authorization Strategy

● Role-Based Strategy

Jenkins Security

- to configure user permissions go to dashboard > manage Jenkins > manage and assign roles > manage roles
- in this screen you can add some rules
- then back to manage and assign roles then add the user that we just created and give him the permissions that you need
- Don’t forget to save!



Manage Roles

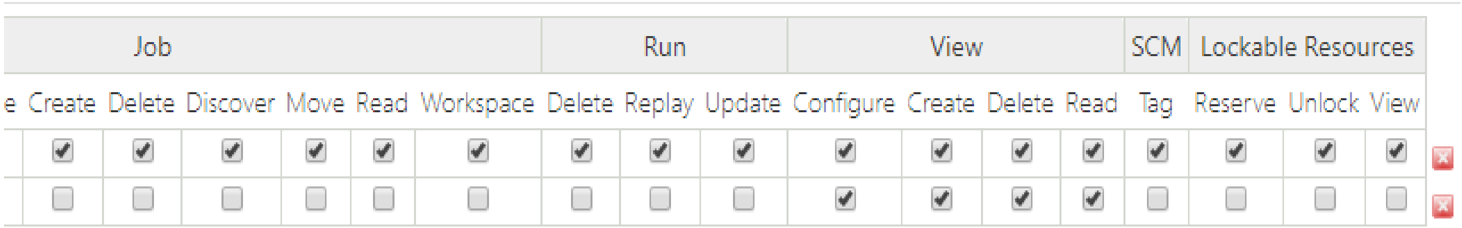
Global roles

Role	Overall		Credentials					Agent				
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete
☒ admin	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

Role to add: view

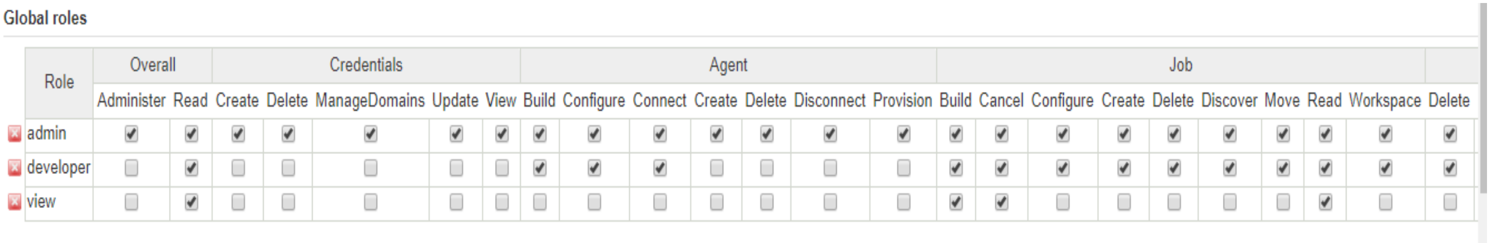
Jenkins Security

- Tick appropriate values for your new role.
- Creating a role to view only.



Job							Run			View				SCM	Lockable Resources			
e	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock	View	
	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒
	□	□	□	□	□	□	□	□	□	☑	☑	☑	☑	□	□	□	□	☒

- Create role **developer** to enable create jobs and configure them.

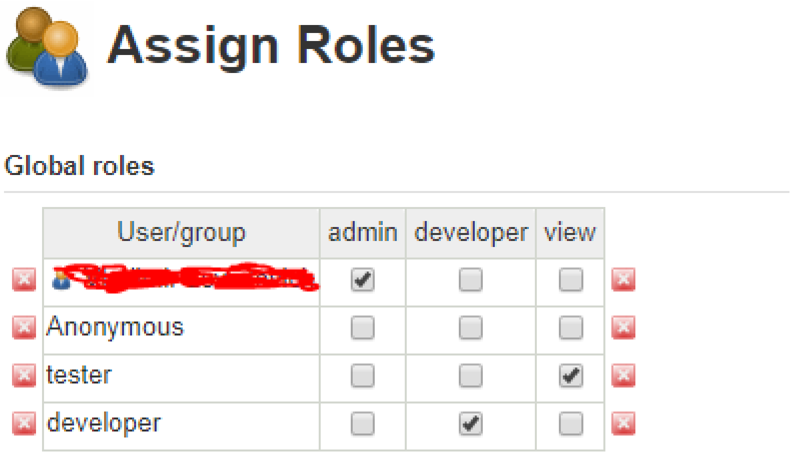


Global roles

	Role	Overall		Credentials					Agent							Job									
		Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete
☒	admin	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
☒	developer	□	☑	□	□	□	□	□	☑	☑	☑	□	□	□	□	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
☒	view	□	☑	□	□	□	□	□	□	□	□	□	□	□	□	☑	☑	□	□	□	□	□	☑	□	□

Jenkins Security

- Go to **Manage and Assign Roles**
- Add user **test** to role **view**
- Add user **developer** to role



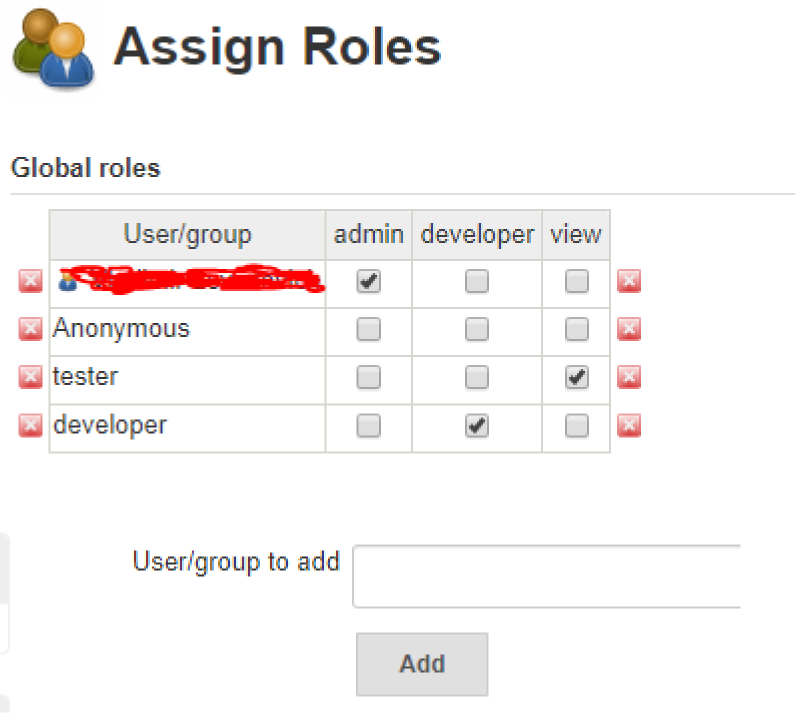
Assign Roles

Global roles

	User/group	admin	developer	view	
☒	[redacted]	☑	□	□	☒
☒	Anonymous	□	□	□	☒
☒	tester	□	□	☑	☒
☒	developer	□	☑	□	☒

Jenkins Security

- Create user **developer** and add to role **developer**



Assign Roles

Global roles

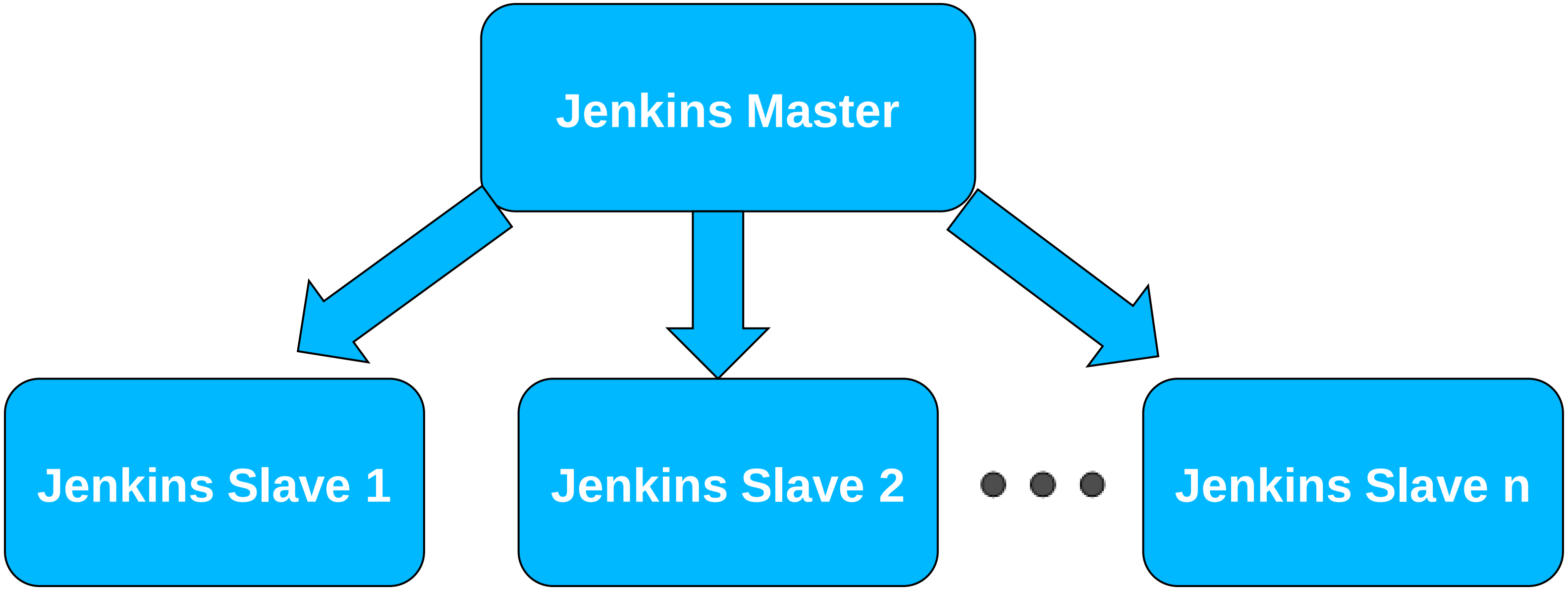
User/group	admin	developer	view
[redacted]	☑	☐	☐
Anonymous	☐	☐	☐
tester	☐	☐	☑
developer	☐	☑	☐

User/group to add

Jenkins Master and Slave

remote machines

Jenkins Master and Slave



touch /var/lib/jenkins/.ssh/authorized_keys

Configuring master node

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- You need to create user **jenkins** on the master node.

  ```
  $ sudo su
  $ useradd -d /var/lib/jenkins jenkins
  ```

- Create ssh keys on the new user

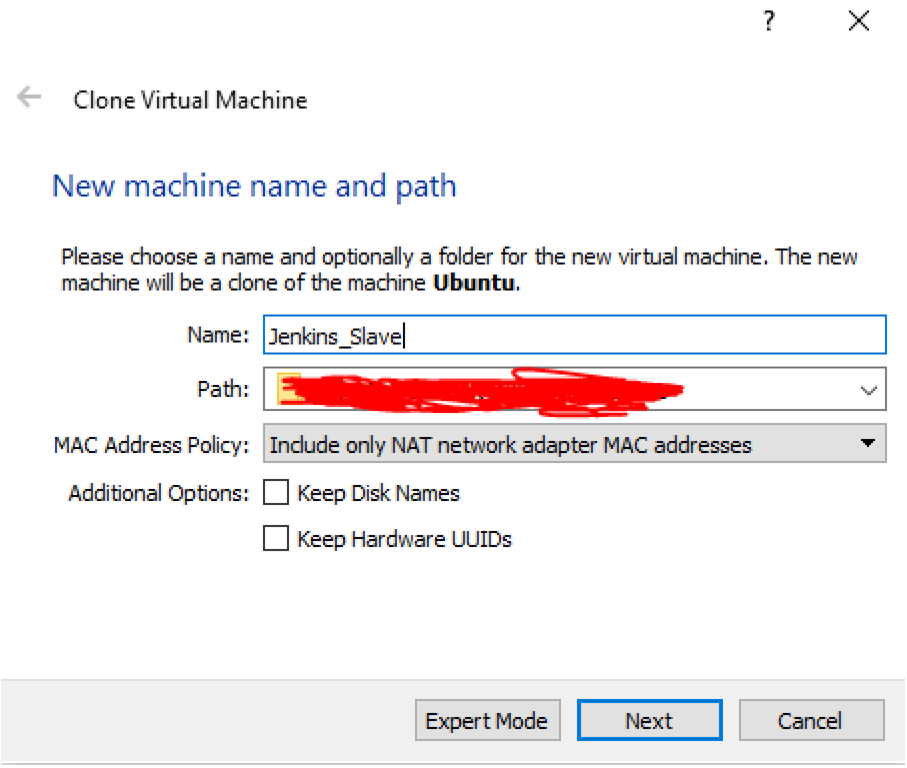
  ```
  $ su jenkins
  $ cd ~
  $ ssh-keygen
  ```

- Copy a public key of master

  ```
  $ cat /var/lib/jenkins/.ssh/id_rsa.pub
  ```

Creating a slave node

- Clone your VM from your initial Linux Machine



New machine name and path

Please choose a name and optionally a folder for the new virtual machine. The new machine will be a clone of the machine **Ubuntu**.

Name: Jenkins_Slave

Path: [redacted]

MAC Address Policy: Include only NAT network adapter MAC addresses

Additional Options: □ Keep Disk Names

□ Keep Hardware UUIDs

Configure a slave node

```
$ sudo apt-get install openssh-server openssh-client
```

- Connect to the slave via Putty
- Install OpenJDK

```
$ sudo apt install openjdk-8-jdk
```

- Create an ssh key-pair

```
$ cd ~
$ ssh-keygen
$ touch ~/.ssh/authorized_keys
```

- Copy a content a public key of master into the slave by

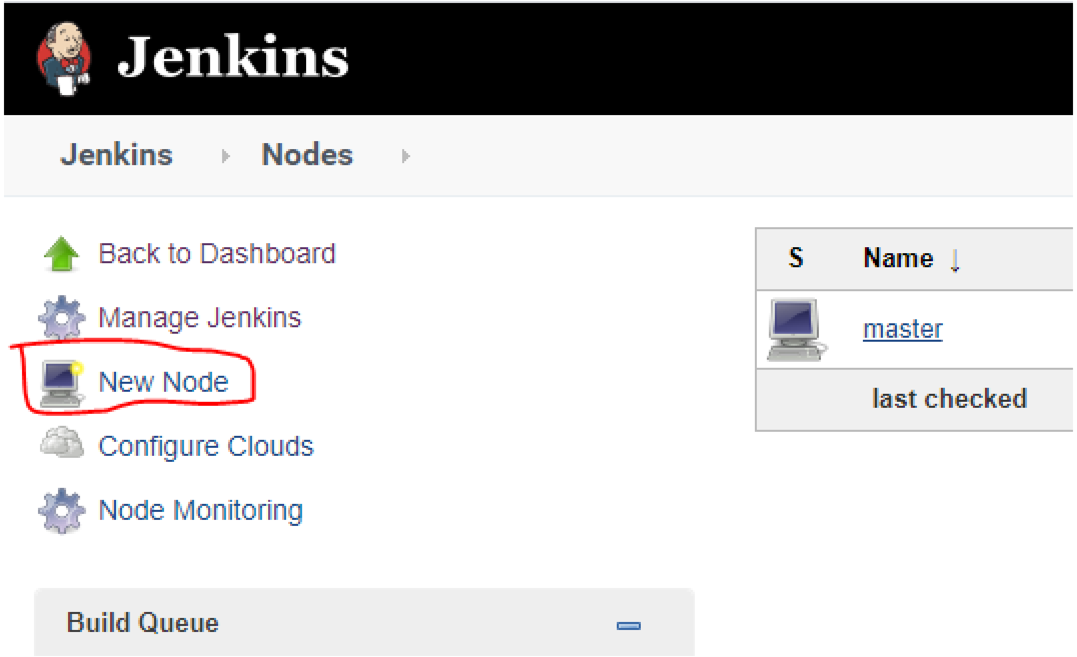
```
$ vim ~/.ssh/authorized_keys
```

- Login to the slave node from master without entering a password.

```
 $ ssh <slave_username>@<slave_IP>
```

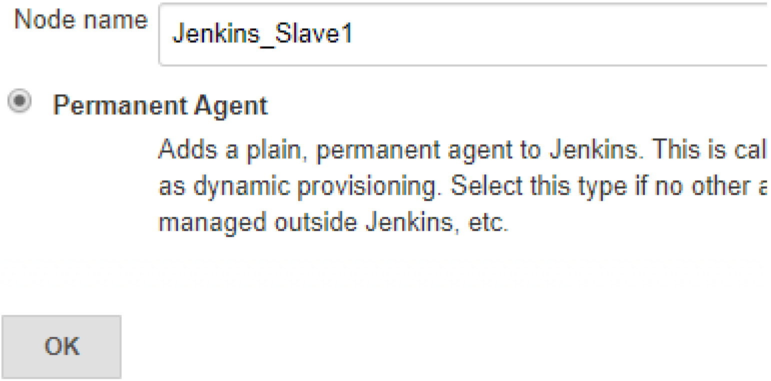
Adding a slave to master

- Go to **Dashboard** > **Manage Jenkins** > **Manage Nodes and Clouds**
- Click on **"New Node"**



Adding a slave to master

- Give a name of the node and select "**Permanent Agent**"

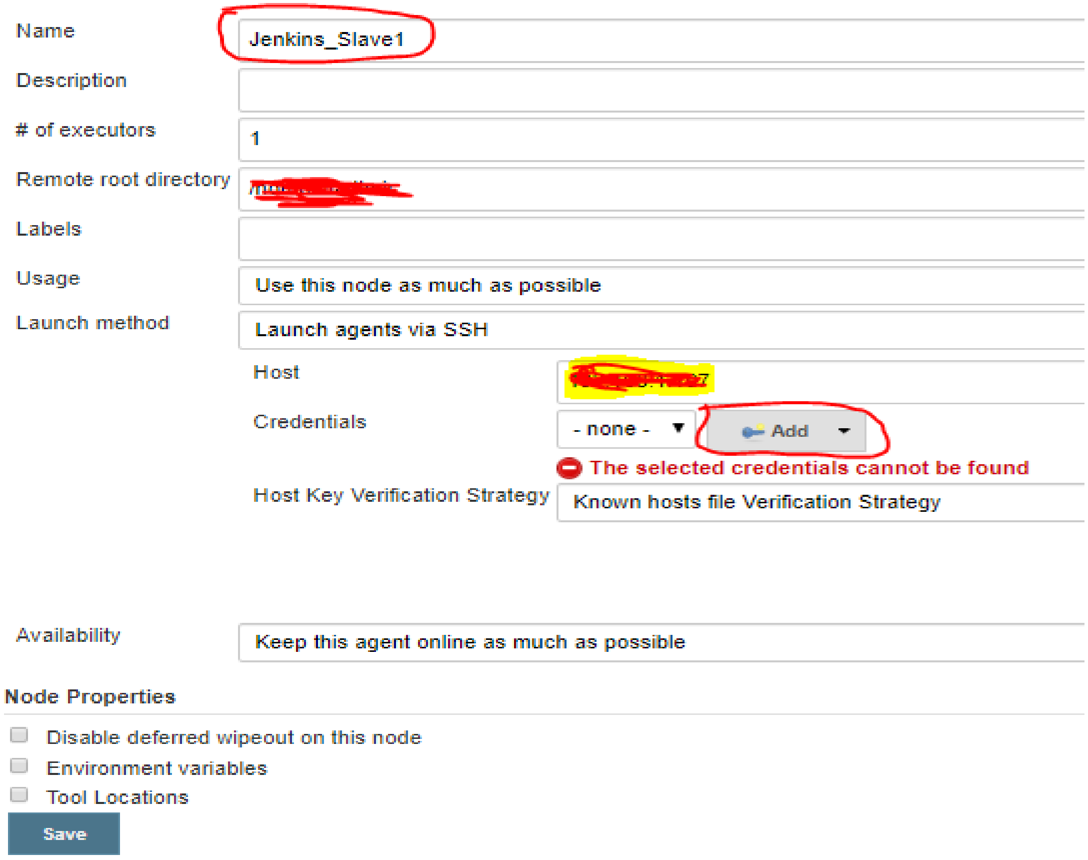


Node name: Jenkins_Slave1

● Permanent Agent

Adds a plain, permanent agent to Jenkins. This is cal
as dynamic provisioning. Select this type if no other a
managed outside Jenkins, etc.

Adding a slave to master



Name	Jenkins_Slave1	
Description		
# of executors	1	
Remote root directory	[redacted]	
Labels		
Usage	Use this node as much as possible	
Launch method	Launch agents via SSH	
	Host	[redacted]
	Credentials	- none - ▼ Add ▾
		⛔ The selected credentials cannot be found
	Host Key Verification Strategy	Known hosts file Verification Strategy
Availability	Keep this agent online as much as possible	

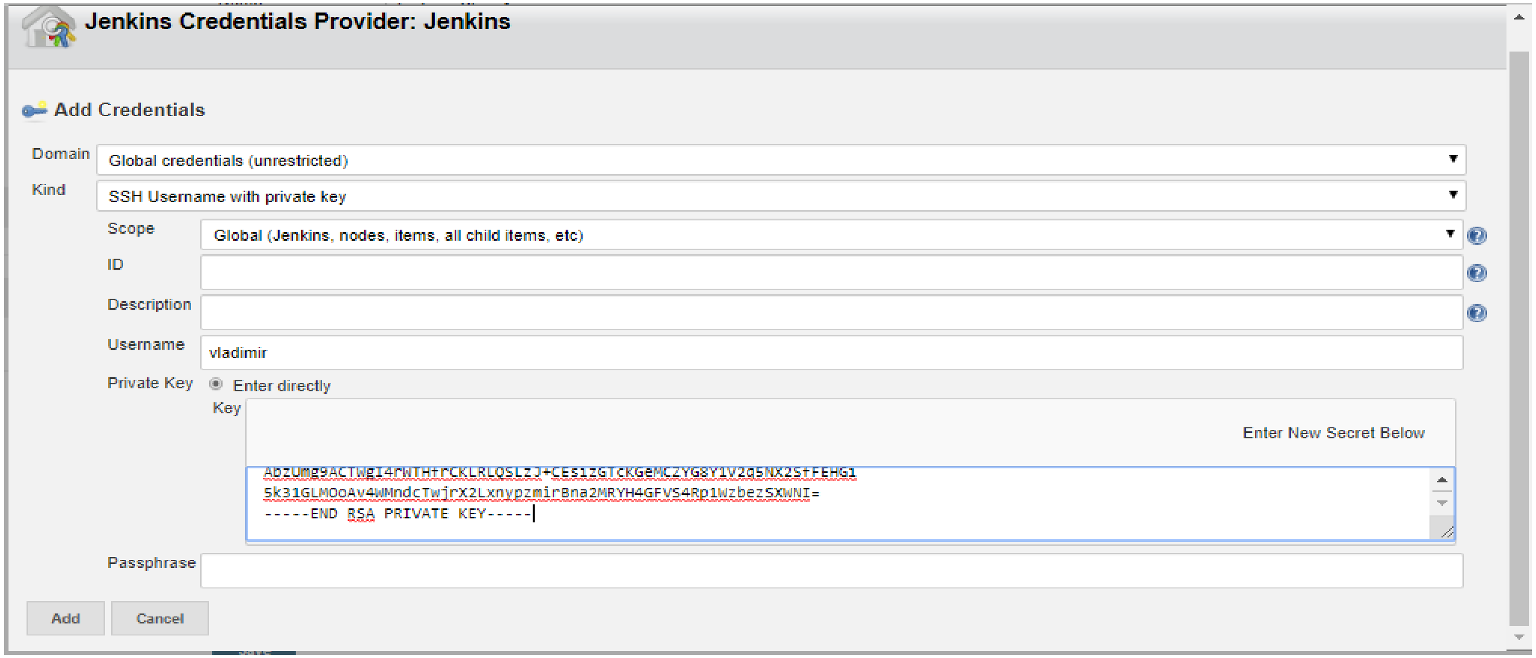
Node Properties

- ☐ Disable deferred wipeout on this node
- ☐ Environment variables
- ☐ Tool Locations

Adding a slave to master

- Select “**SSH Username with with private key**”
- Copy the private key from the **master** node by

```
$ ssh-keygen
$ cat .ssh/id_rsa.pub
```



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID:

Description:

Username: vladimir

Private Key: ● Enter directly

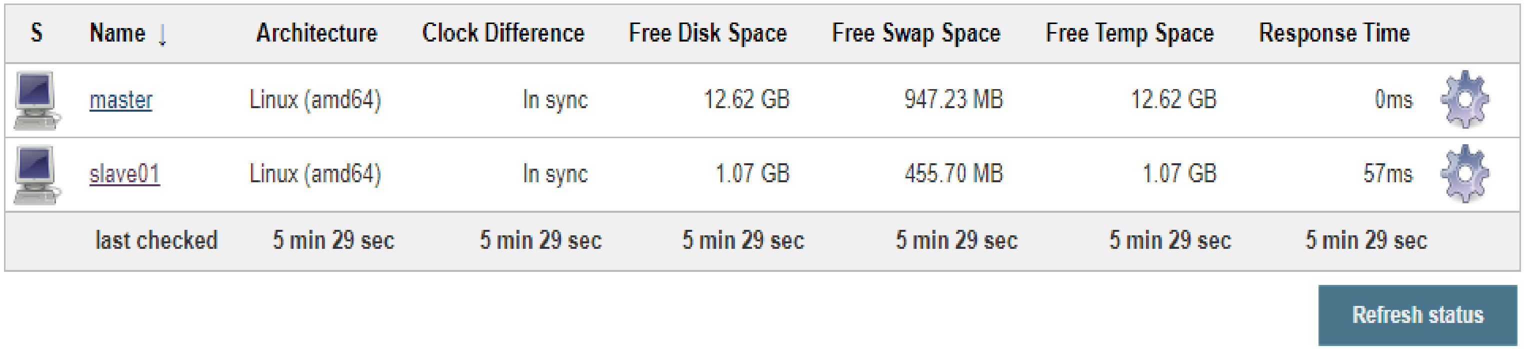
Key: Enter New Secret Below

```
AbzUmg9ACTWgI4rWTH+rCKLRLQSLzJ+CEs1zGTcKGeMCZYG8Y1V2q5NX2S+FEHG1
5k31GLMOoAv4WMndcTwjrX2LxnypzmirBna2MRYH4GFVS4Rp1WzbezSXWNI=
-----END RSA PRIVATE KEY-----
```

Passphrase:

Adding a slave to master

- Save the settings and check your node

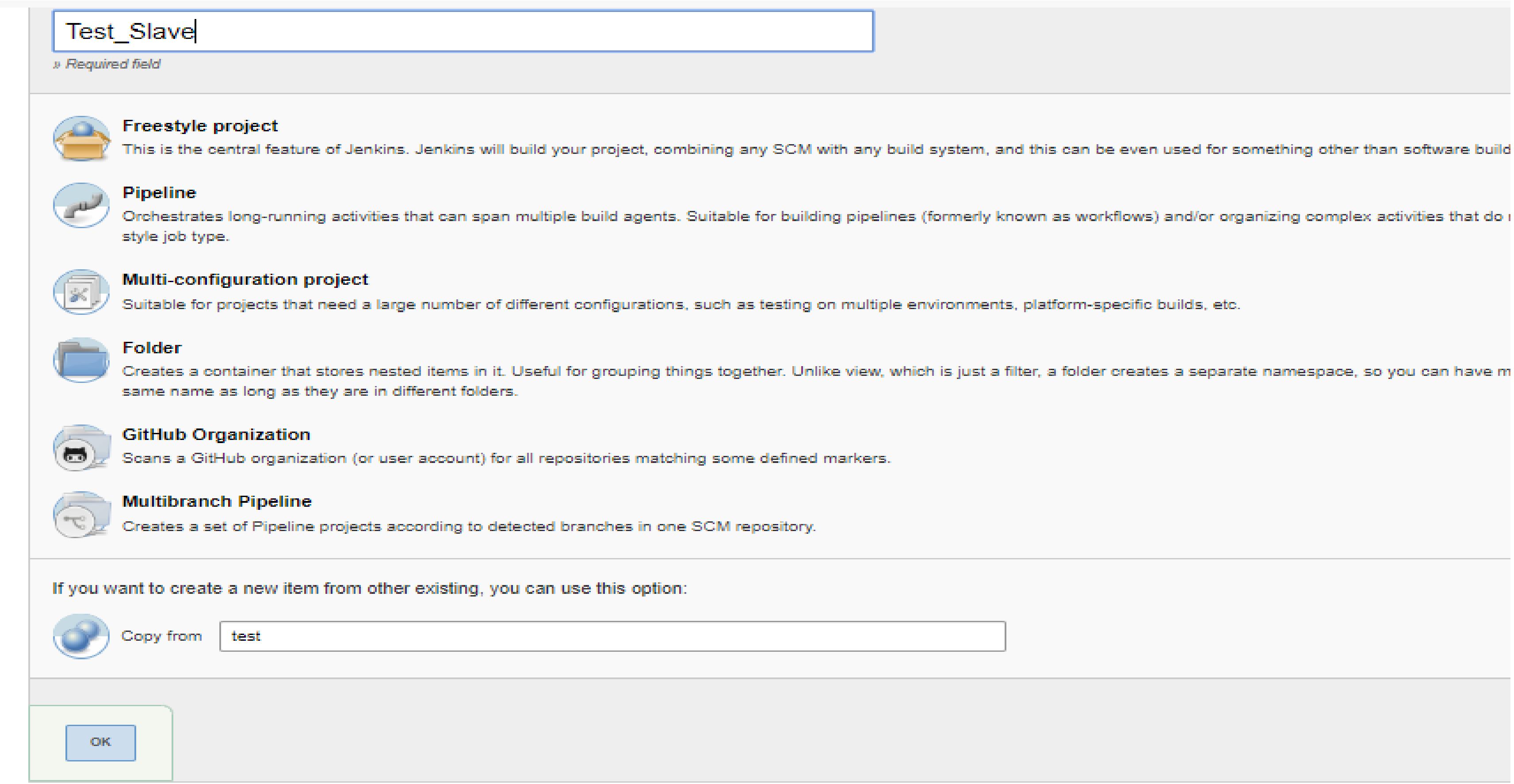


S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	master	Linux (amd64)	In sync	12.62 GB	947.23 MB	12.62 GB	0ms	
	slave01	Linux (amd64)	In sync	1.07 GB	455.70 MB	1.07 GB	57ms	
	last checked	5 min 29 sec	5 min 29 sec	5 min 29 sec	5 min 29 sec	5 min 29 sec	5 min 29 sec	

Running Slave node

remote machines

A new job with a slave



Test_Slave

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do
style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have m
same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

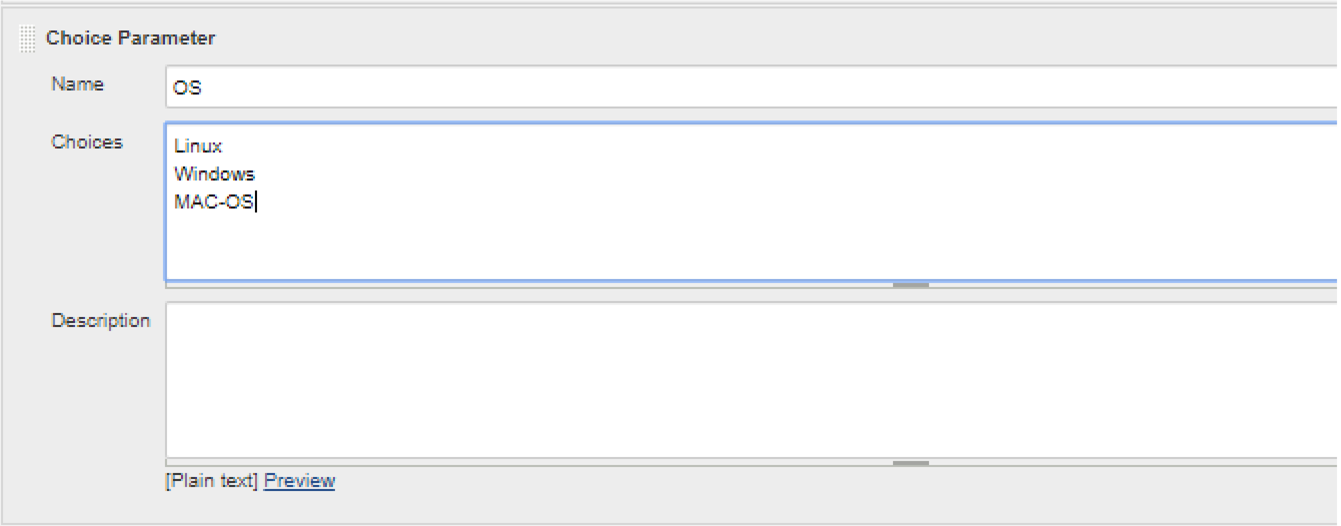
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Copy from: test

A new job with a slave

- Use a new **Choice parameter**



Choice Parameter

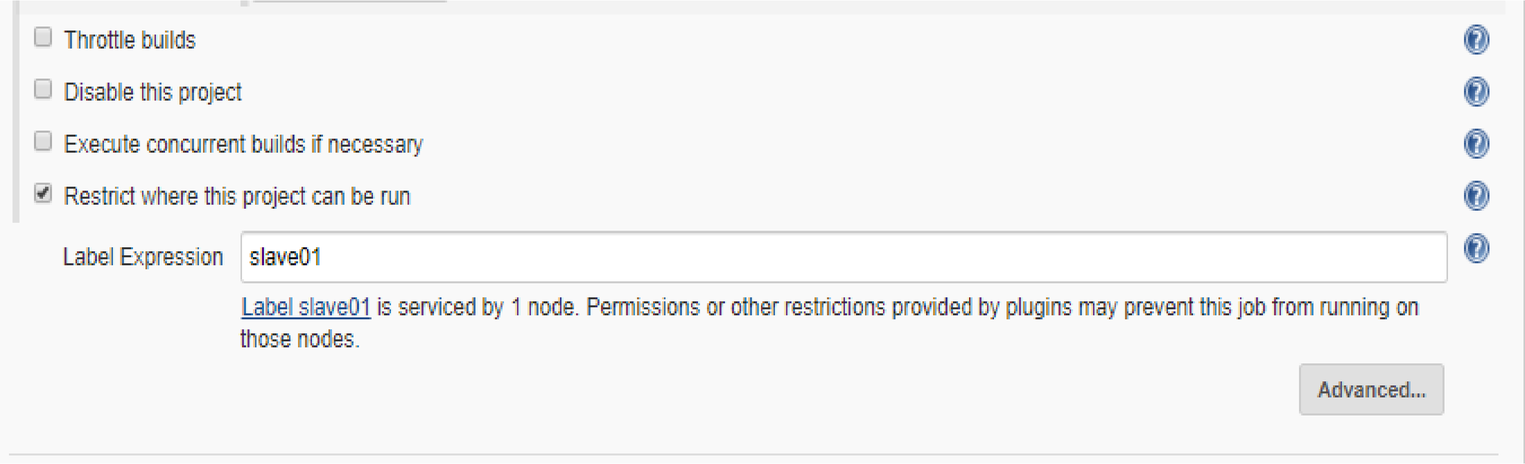
Name: OS

Choices:
Linux
Windows
MAC-OS

Description:

A new job with a slave

- Add the new slave node



□ Throttle builds

□ Disable this project

□ Execute concurrent builds if necessary

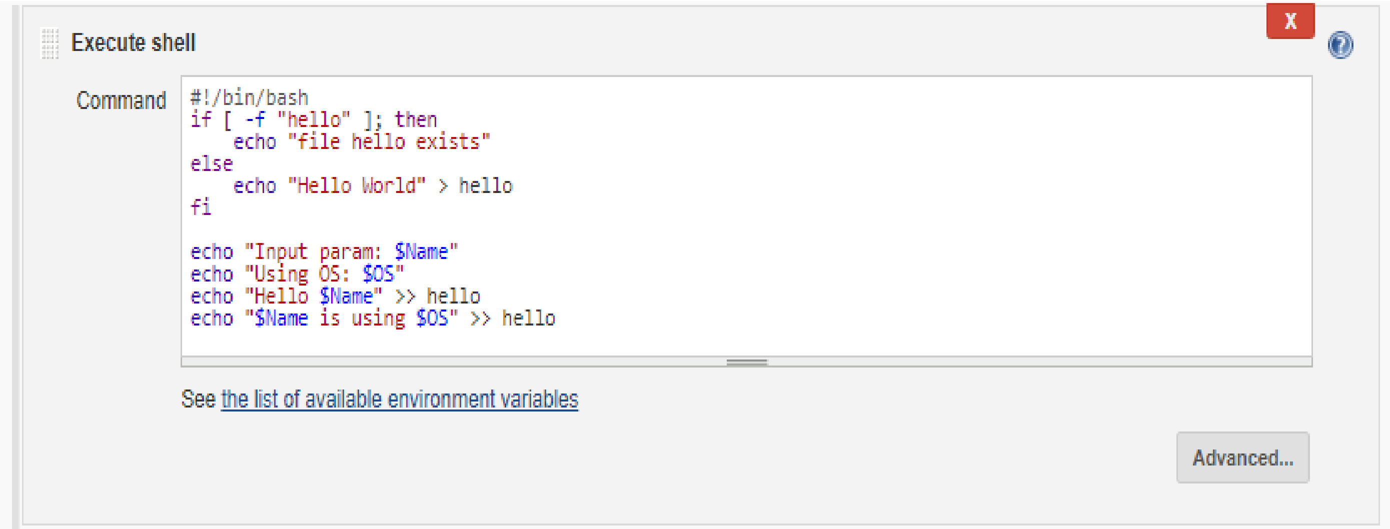
☑ Restrict where this project can be run

Label Expression: slave01

Label slave01 is serviced by 1 node. Permissions or other restrictions provided by plugins may prevent this job from running on those nodes.

A new job with a slave

- Update your shell script



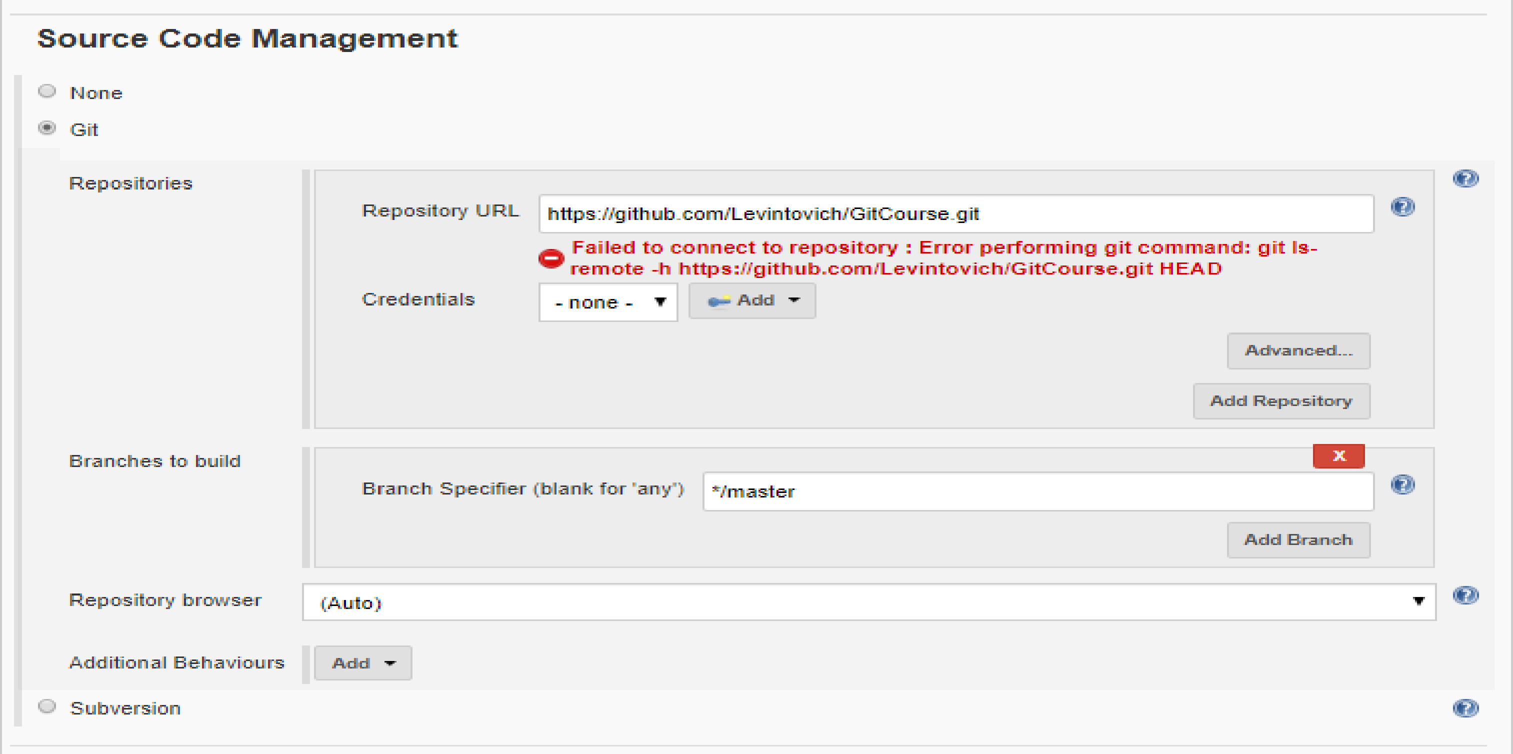
```
#!/bin/bash
if [ -f "hello" ]; then
    echo "file hello exists"
else
    echo "Hello World" > hello
fi

echo "Input param: $Name"
echo "Using OS: $OS"
echo "Hello $Name" >> hello
echo "$Name is using $OS" >> hello
```

- Run the job and check the **Console Output**

Integration with GitHub

Checkout SCM

- Create a new job and add the GitHub repository



Source Code Management

○ None

● Git

Repositories

Repository URL: https://github.com/Levintovich/GitCourse.git

Failed to connect to repository : Error performing git command: git ls-remote -h https://github.com/Levintovich/GitCourse.git HEAD

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

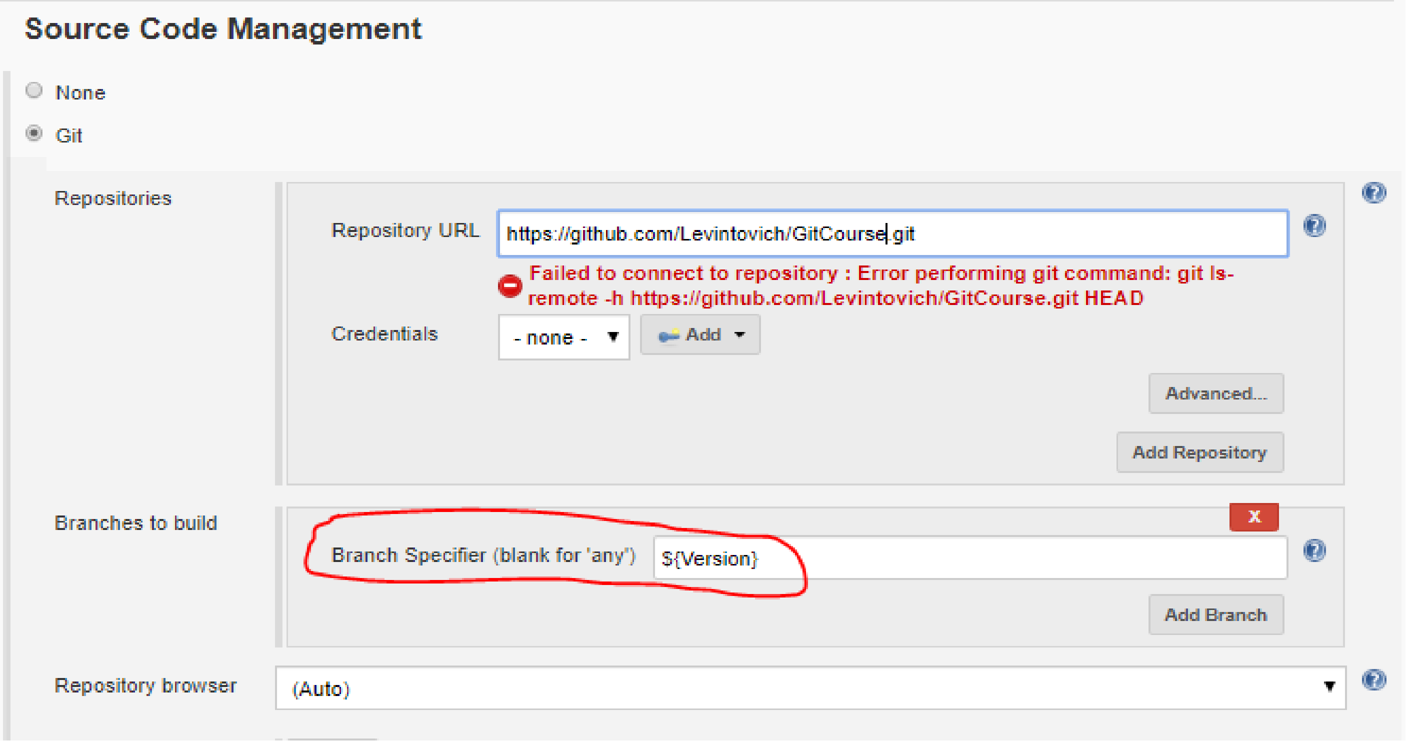
Repository browser: (Auto)

Additional Behaviours

○ Subversion

Checkout SCM

- Clone the particular release by choosing the required version (add a **Choise Parameter**)



Source Code Management

○ None

● Git

Repositories

Repository URL: https://github.com/Levintovich/GitCourse.git

Failed to connect to repository : Error performing git command: git ls-remote -h https://github.com/Levintovich/GitCourse.git HEAD

Credentials: - none -

Branches to build

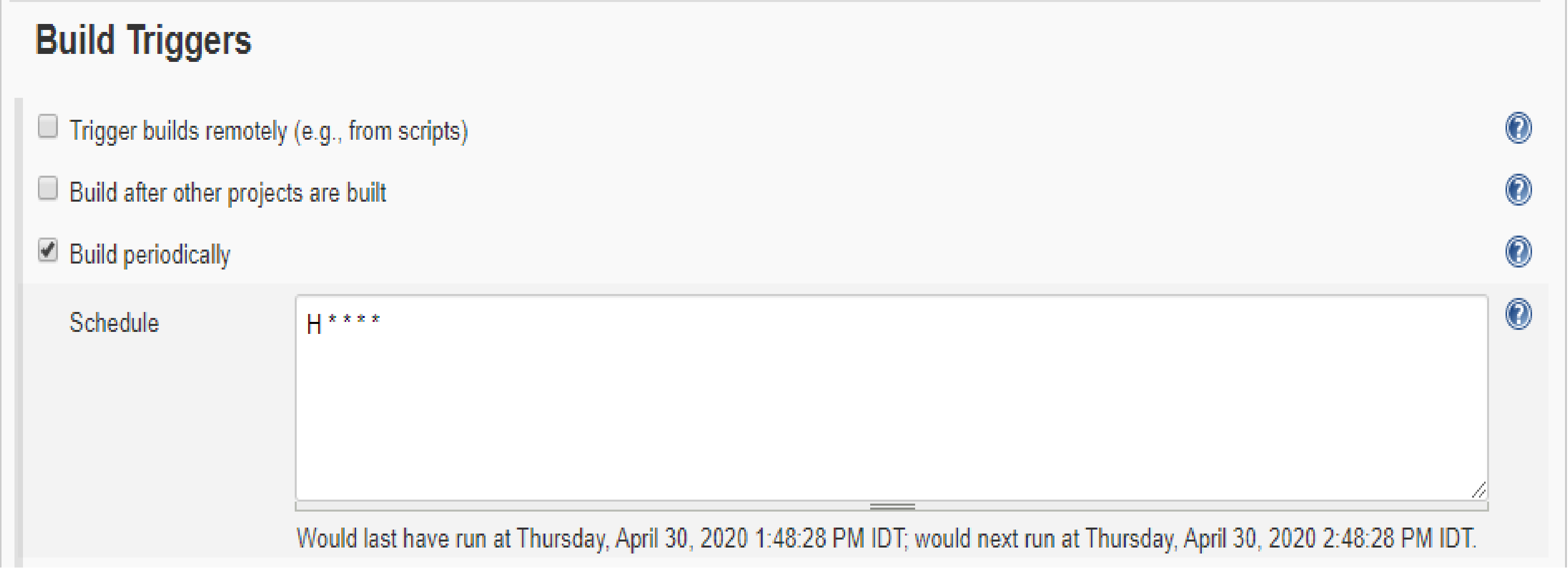
Branch Specifier (blank for 'any'): ${Version}

Repository browser: (Auto)

Build Triggers

Build Periodically

- Automatically triggers periodically – **scheduling**.
- **H * * * *** triggers every hour



Build Triggers

□ Trigger builds remotely (e.g., from scripts)

□ Build after other projects are built

☑ Build periodically

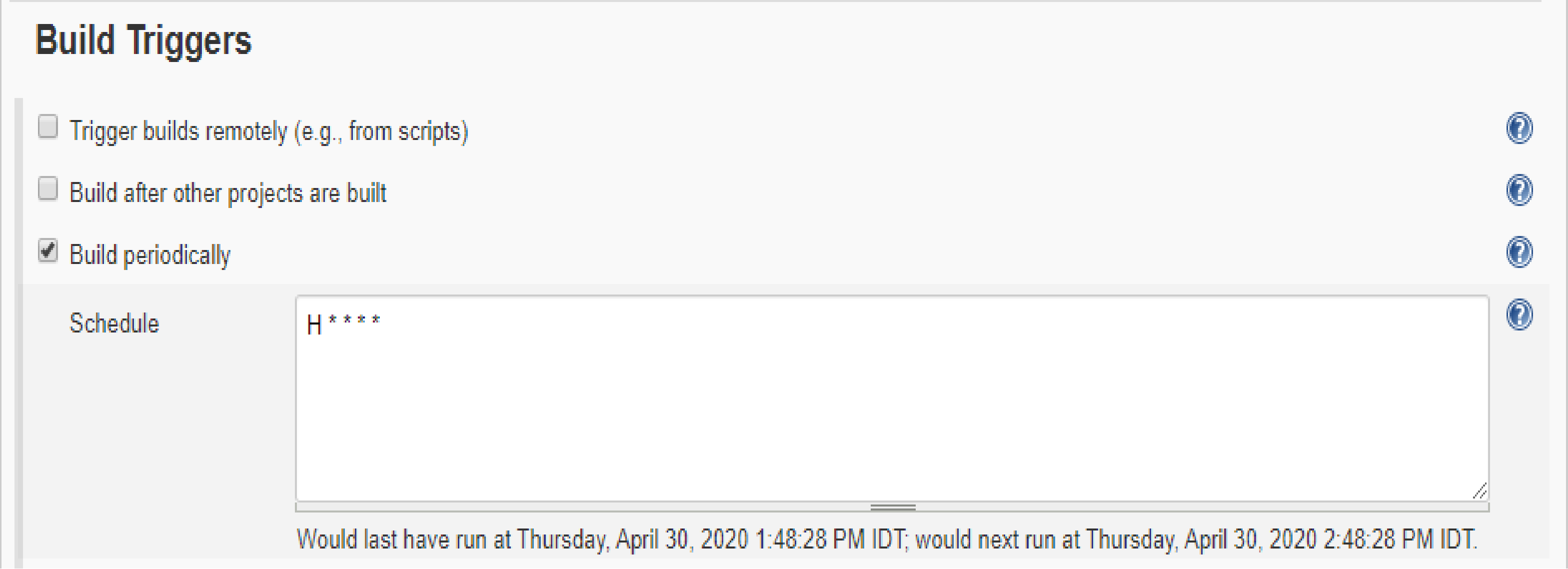
Schedule

```
H * * * *
```

Would last have run at Thursday, April 30, 2020 1:48:28 PM IDT; would next run at Thursday, April 30, 2020 2:48:28 PM IDT.

Build Periodically

- Automatically triggers periodically – **scheduling**.
- **H * * * *** triggers every hour



Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☑ Build periodically

Schedule

```
H * * * *
```

Would last have run at Thursday, April 30, 2020 1:48:28 PM IDT; would next run at Thursday, April 30, 2020 2:48:28 PM IDT.

Build Periodically

- ***/5 * * * *** to schedule the build every 5 minutes
- **15 13 * * *** to schedule the build every day at 13:15

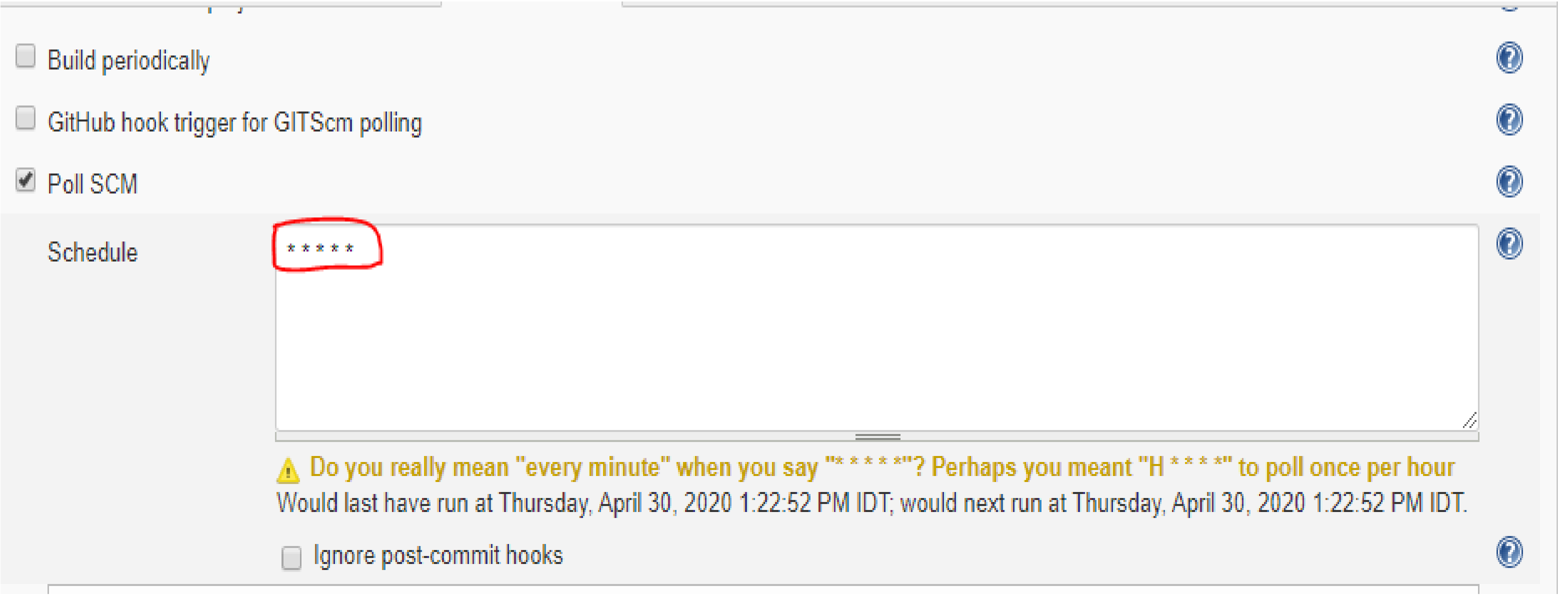
Syntax

- MINUTES Minutes in one hour (0-59)
- HOURS Hours in one day (0-23)
- DAYMONTH Day in a month (1-31)
- MONTH Month in a year (1-12)
- DAYWEEK Day of the week (0-7) where 0 and 7 are Sunday
- For further information you can use:

https://en.wikipedia.org/wiki/Cron#CRON_expression

Poll SCM

- Automatically triggers after each commit on GitHub.
- But before that install **git** on **master**



☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☑ Poll SCM

Schedule

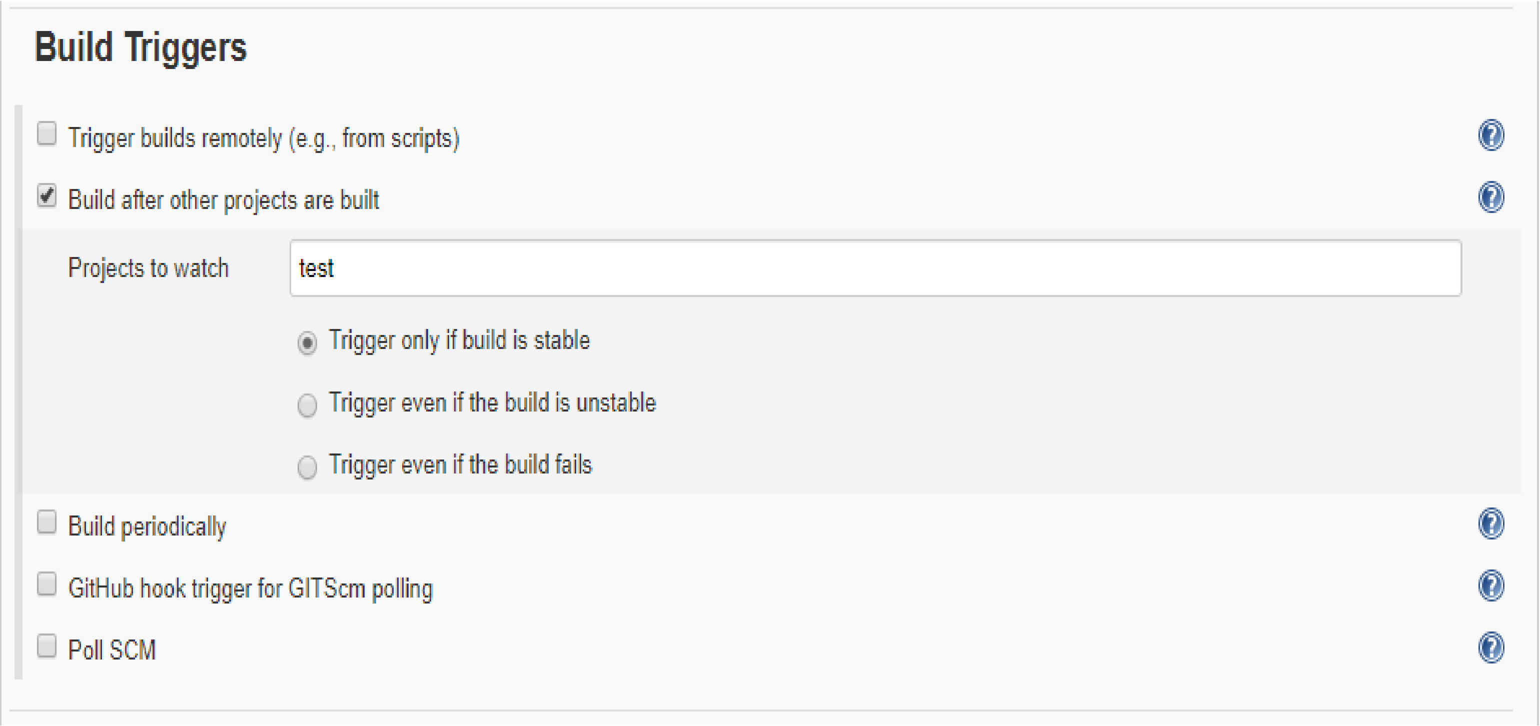
```
* * * * *
```

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Thursday, April 30, 2020 1:22:52 PM IDT; would next run at Thursday, April 30, 2020 1:22:52 PM IDT.

☐ Ignore post-commit hooks

Trigger by another job



Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☑ Build after other projects are built

Projects to watch: test

● Trigger only if build is stable

○ Trigger even if the build is unstable

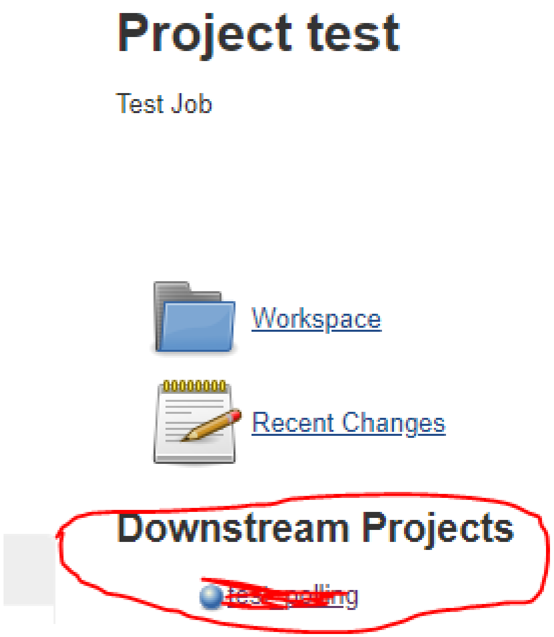
○ Trigger even if the build fails

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Trigger by another job



Project test

Test Job

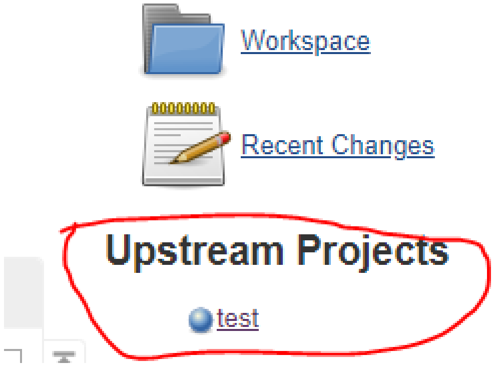
Workspace

Recent Changes

Downstream Projects

[redacted]

Trigger by another job



Workspace

Recent Changes

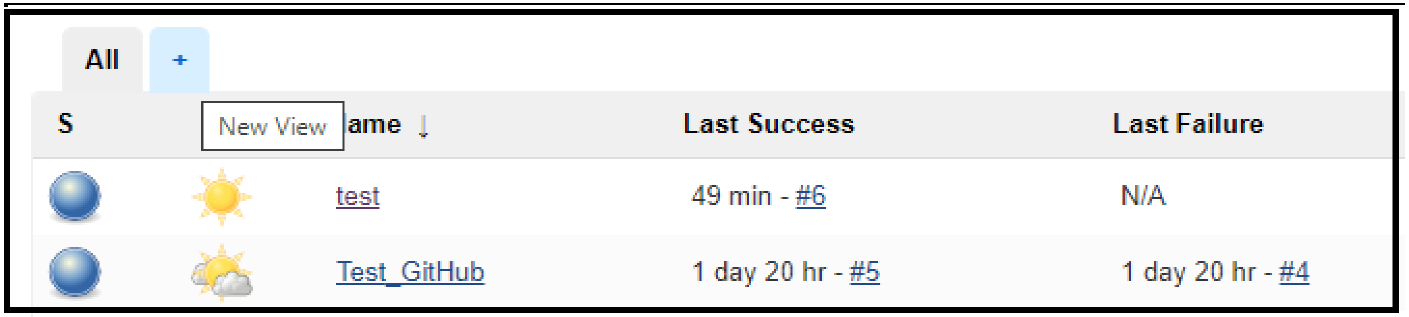
Upstream Projects

test

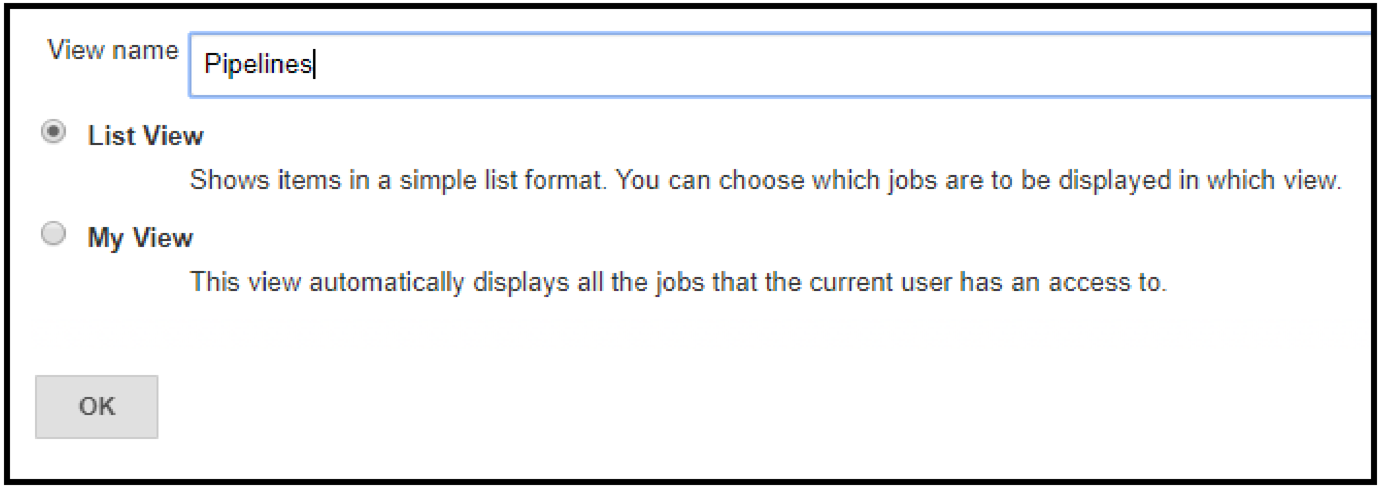
Jenkins View

Provides a view of the selected jobs

Create a new view



S		ame ↓	Last Success	Last Failure
●	☀	test	49 min - #6	N/A
●	⛅	Test_GitHub	1 day 20 hr - #5	1 day 20 hr - #4



View name: Pipelines

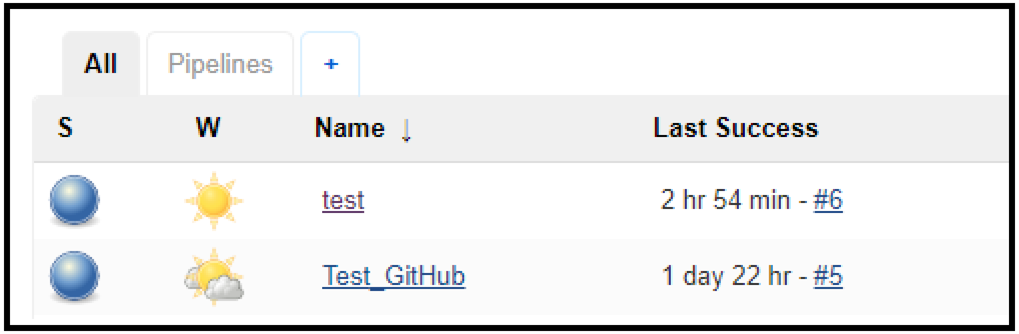
● **List View**

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

○ **My View**

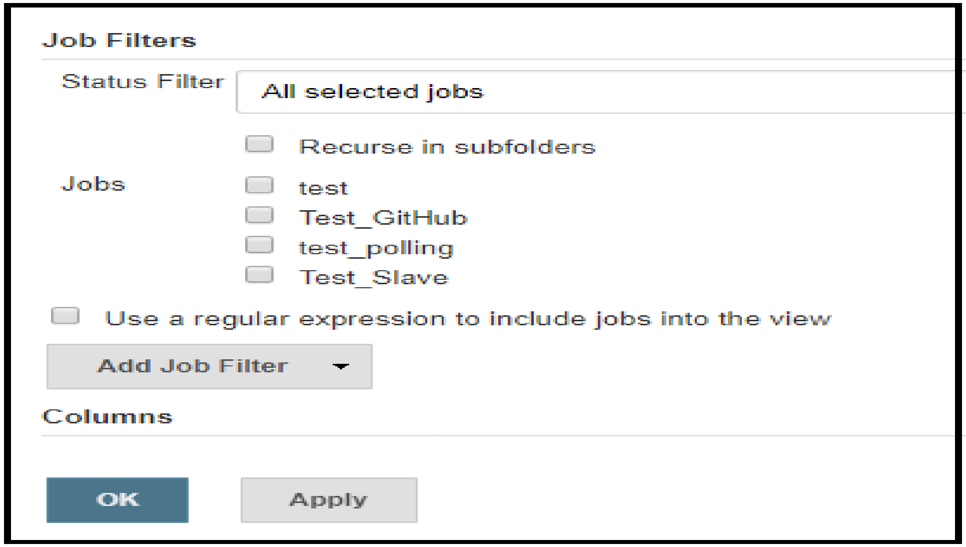
This view automatically displays all the jobs that the current user has an access to.

Create a new view



All | Pipelines | +

S	W	Name ↓	Last Success
●	☀	test	2 hr 54 min - #6
●	☀	Test_GitHub	1 day 22 hr - #5



Job Filters

Status Filter: All selected jobs

Jobs
- □ Recurse in subfolders
- □ test
- □ Test_GitHub
- □ test_polling
- □ Test_Slave

□ Use a regular expression to include jobs into the view

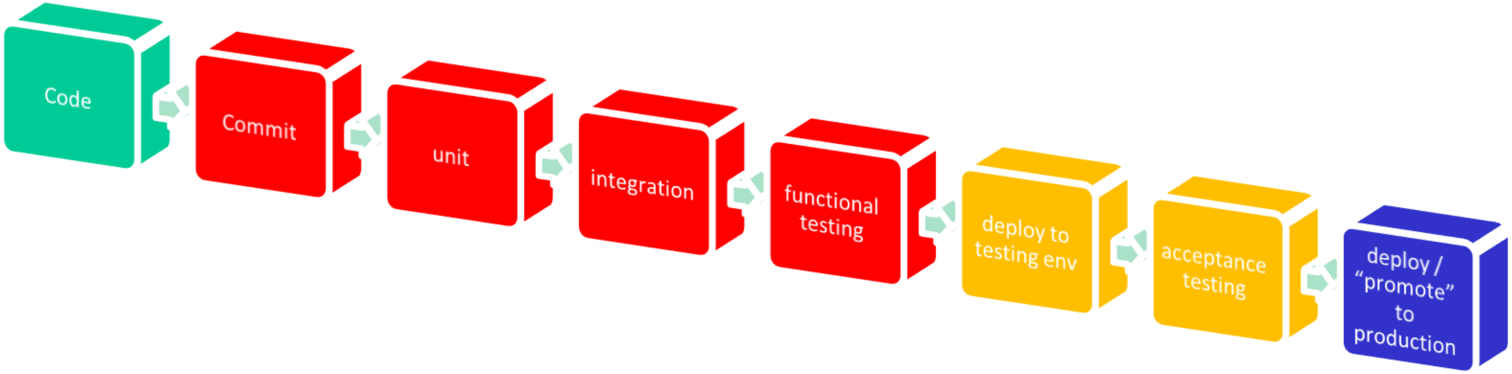
Add Job Filter ▾

Columns

OK | Apply

Jenkins CI/CD

Release Pipeline



Continuous Integration

- The CI part of CICD can be summarized with: you want all parts of what goes into making your application go to the same place and run through the same processes with results published to an easy to access place.

- The simplest example of continuous integration is something you might not have even thought of being significant: committing all your application code in a single repository! While that may seem like a no-brainer, having a single place where you “integrate” all your code is the foundation for extending other, more advanced practices.

- Once you have all your code and changes going to the same place, you can run some processes on that repository every time something changes. This could include:

Continuous Integration

- Run automatic code quality scans on it and generate a report of how well your latest changes adhere to good coding practices
- Build the code and run any automated tests that you might have written to make sure your changes didn’t break any functionality
- Generate and publish a test coverage report to get an idea of how thorough your automated tests are

- These simple additions (made easy with tooling that will be mentioned later) allows you, the developer, to focus on writing the code. Your central repository of code is there to receive your changes while your automated processes can build, test, and scan your code while providing reports.

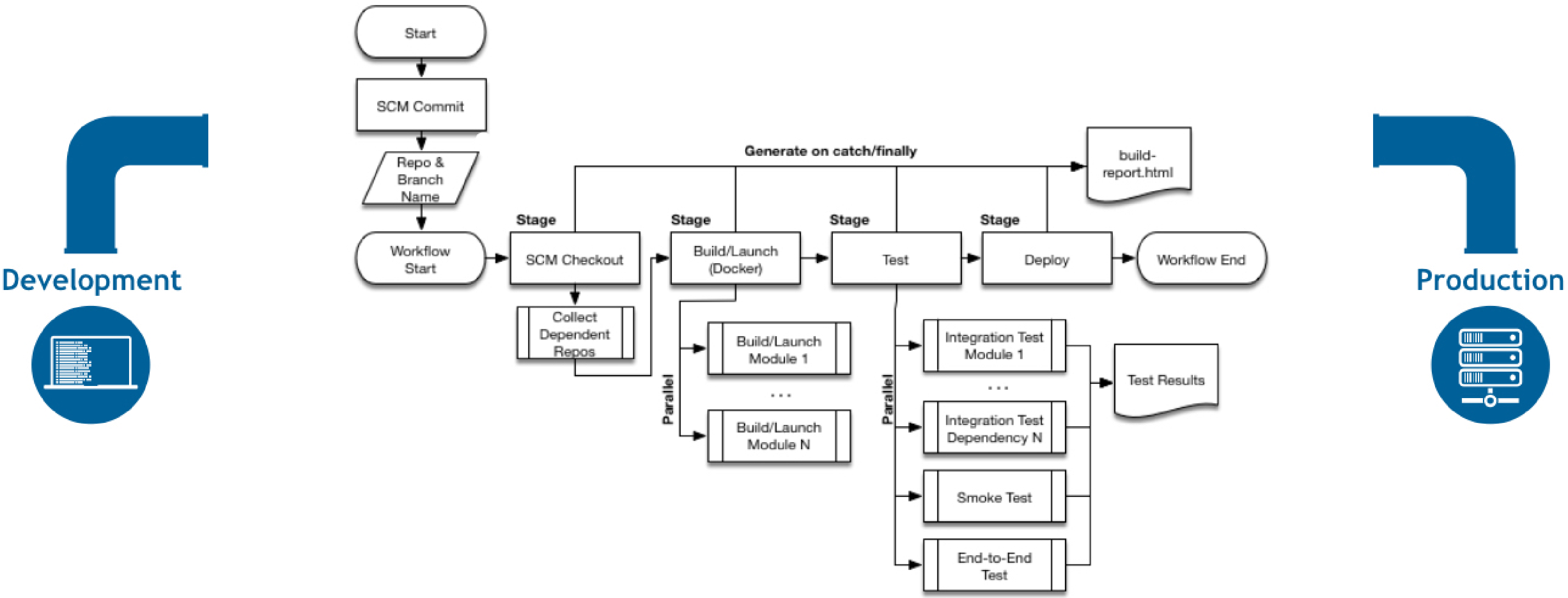
Continuous Deployment

- Deploying code can be hard. If you’ve ever been jamming on building a project for a while, shifting your mindset to getting it ready to be deployed can be jarring.

- One of the best things you can do to avoid this, much like other things in software, is to automate it! Make it so that your code gets automatically deployed to wherever you or your users can get to it.

- There are many freely available tools to let you do this easily. One popular example is Travis CI, which integrates directly with Github. You can configure Travis to automatically run CI tasks like unit tests and push your code to a hosting platform like Heroku every time you push new changes to a branch.

Release Pipeline

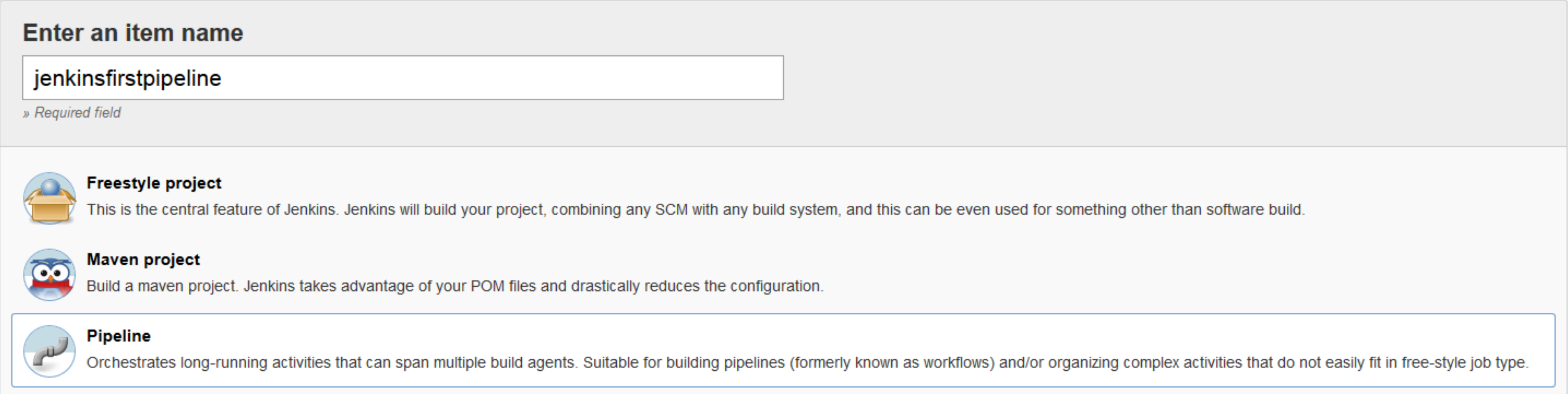


Jenkins file

- Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.
- for more information go to https://jenkins.io/doc/book/pipeline/jenkinsfile/
- for syntax go to https://jenkins.io/doc/book/pipeline/syntax/

Jenkins first pipeline

- Before we start make sure that the pipeline plugin installed
- now create new Jenkins project give it a name and then choose Pipeline



Enter an item name

jenkinsfirstpipeline

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

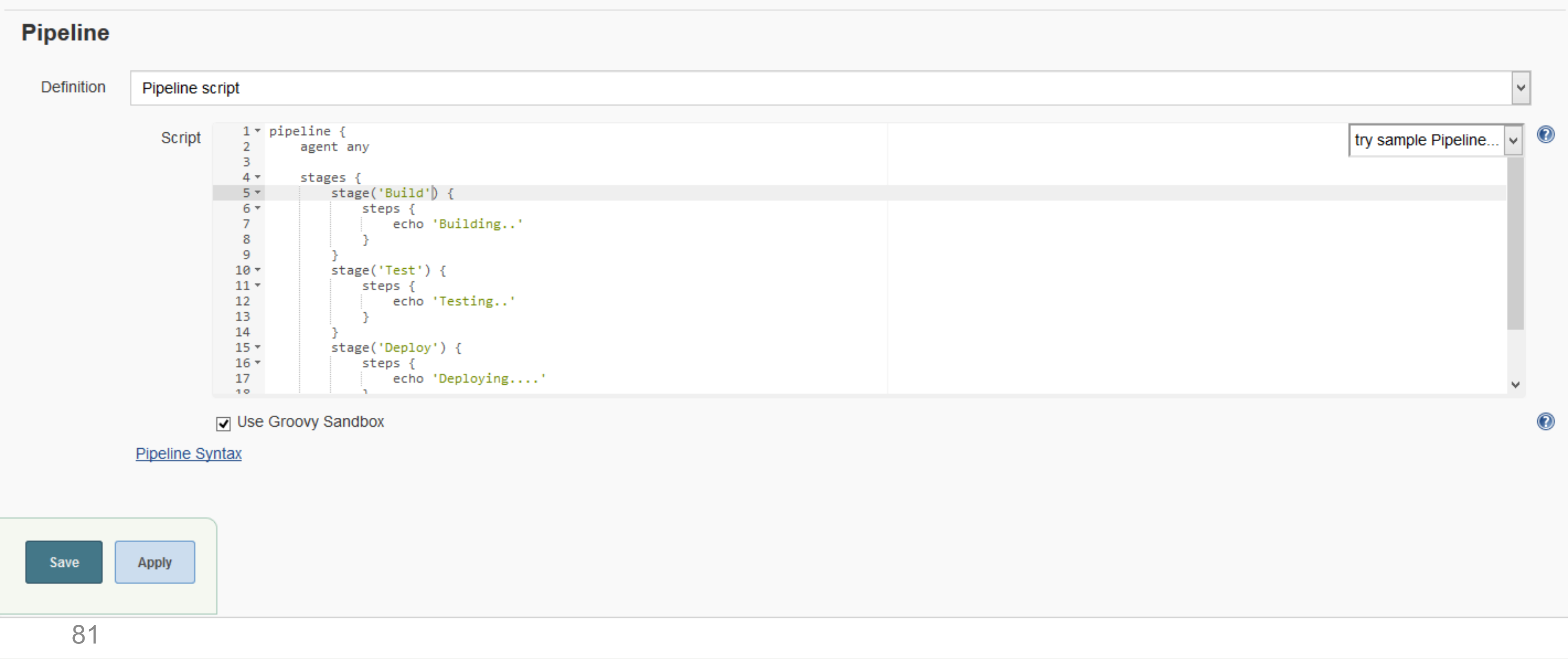
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

- then click OK

Jenkins first pipeline

- Go to Pipeline and paste the script that you got press save and build the job



Definition: Pipeline script

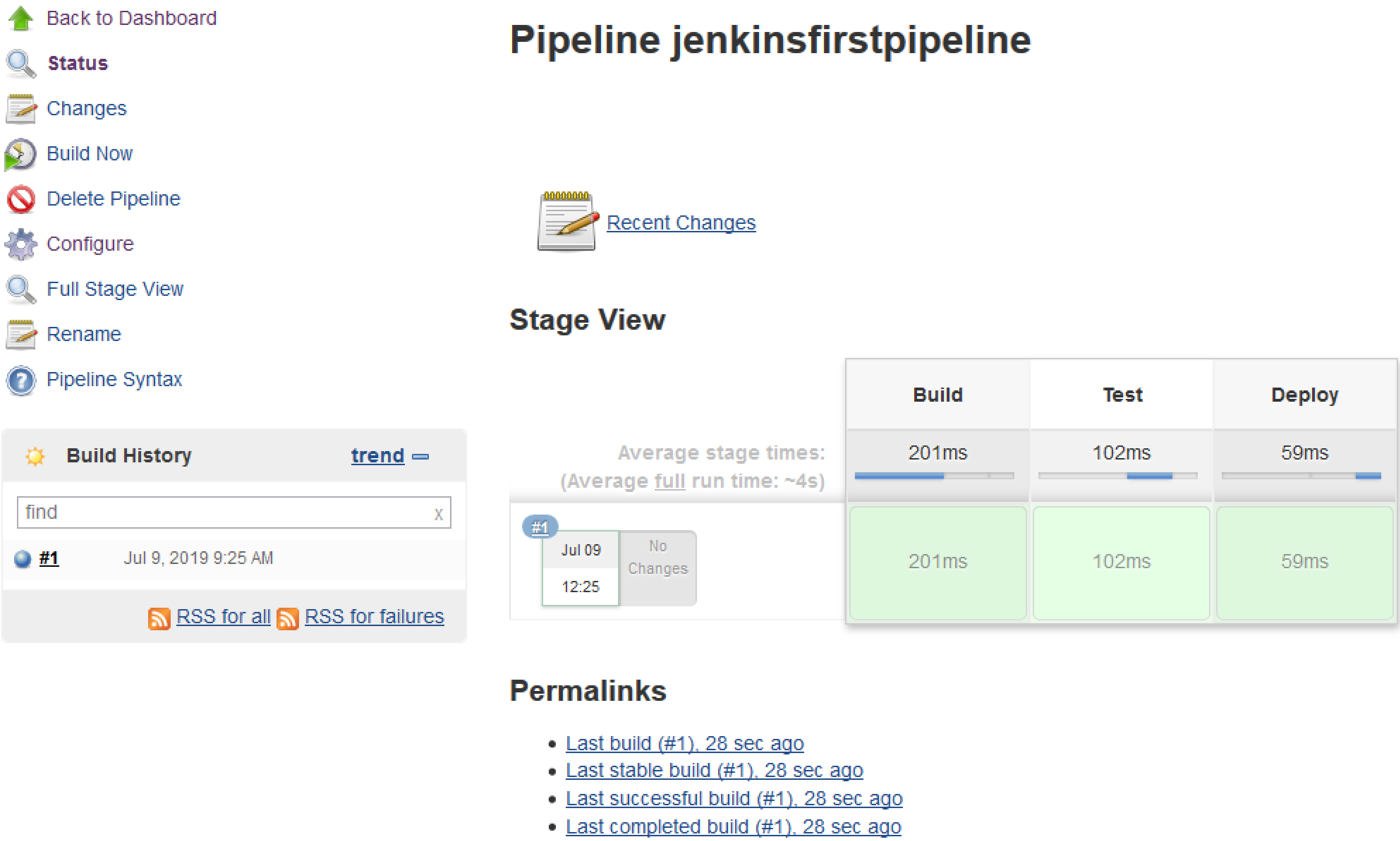
Script

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
```

☑ Use Groovy Sandbox

Jenkins first pipeline

- After you build your job you will see this screen



- Back to Dashboard
- **Status**
- Changes
- Build Now
- Delete Pipeline
- Configure
- Full Stage View
- Rename
- Pipeline Syntax

Build History trend

find x

#1 Jul 9, 2019 9:25 AM

RSS for all RSS for failures

Pipeline jenkinsfirstpipeline

Recent Changes

Stage View

	Build	Test	Deploy
Average stage times: (Average full run time: ~4s)	201ms	102ms	59ms
#1 Jul 09 12:25 No Changes	201ms	102ms	59ms

Permalinks

- Last build (#1), 28 sec ago
- Last stable build (#1), 28 sec ago
- Last successful build (#1), 28 sec ago
- Last completed build (#1), 28 sec ago

Jenkins first pipeline

- After we finish we can choose the latest item from build history and then we will go to console output
- In this screen we will see that Jenkins print what we write in the script this is very important because we will see how it execute the file and if it fail we will have more information about the fail

Jenkins multi-steps

- Jenkins let us do more then one command so lets add

```
• stage('Build') {
•         steps {
•             sh 'echo "My first pipeline"'
•             sh '''
•                 echo "By the way, I can do more stuff in here"
•                 ls –la ~
•             '''
•         }
•     }
```

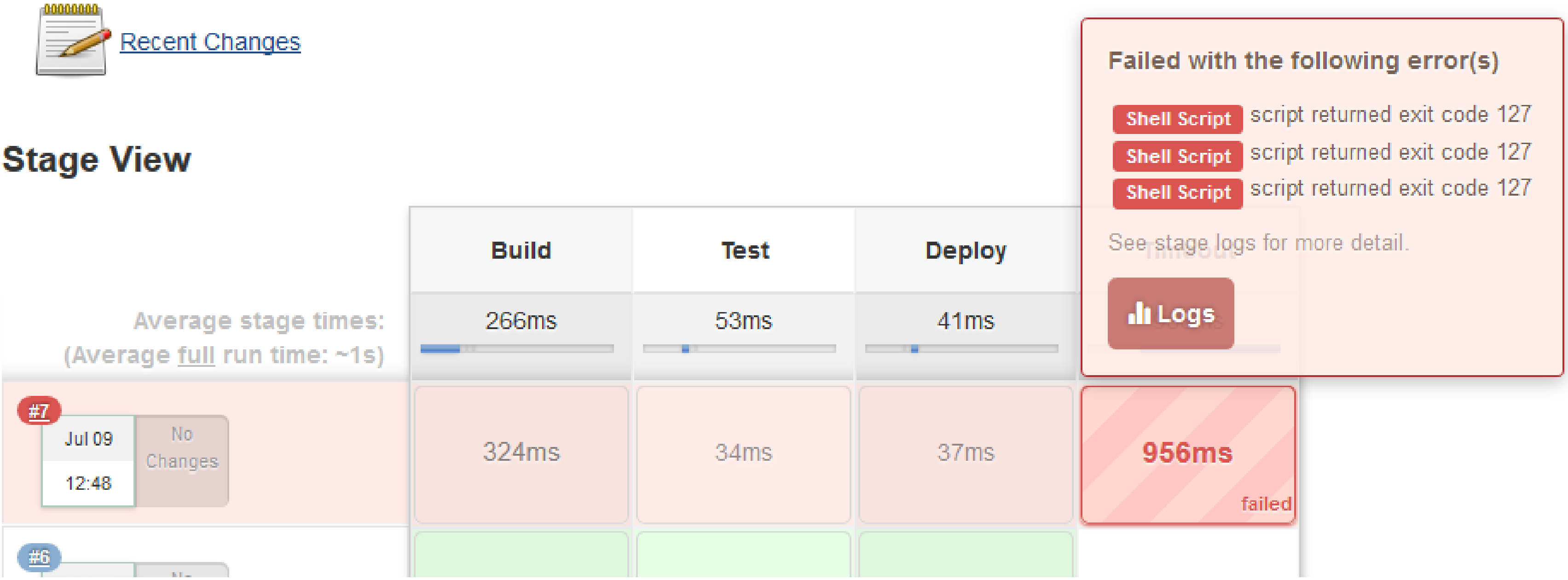
- to our Jenkins file

Jenkins retry

- Sometimes when we execute a command it will fail example connect to remote server it can fail and then we want to execute the command again because this time the server running, lets add this script to our file
- stage('Timeout') {
- steps {
- retry(3) {
- sh 'I am not going to work :c'
- }
- }
- }
- this will stage will fail because I it not a command

Jenkins retry

- After we added the script we build the project and as expected the build fails
- the popup appears if we will stand with the mouse on the error



Recent Changes

Stage View

Average stage times: (Average full run time: ~1s)	Build	Test	Deploy	
	266ms	53ms	41ms	
#7 Jul 09 12:48 No Changes	324ms	34ms	37ms	956ms failed
#6				

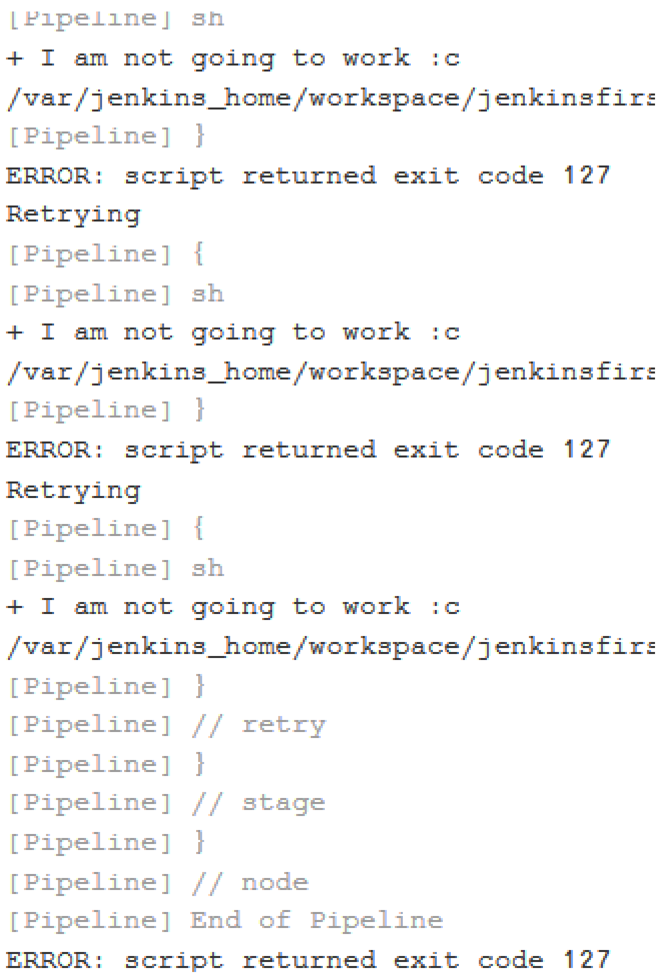
Failed with the following error(s)

Shell Script script returned exit code 127
Shell Script script returned exit code 127
Shell Script script returned exit code 127

See stage logs for more detail.

Jenkins retry

- now lets go to the console output
- we can see that the script
- executed 3 time



```
[Pipeline] sh
+ I am not going to work :c
/var/jenkins_home/workspace/jenkinsfirs
[Pipeline] }
ERROR: script returned exit code 127
Retrying
[Pipeline] {
[Pipeline] sh
+ I am not going to work :c
/var/jenkins_home/workspace/jenkinsfirs
[Pipeline] }
ERROR: script returned exit code 127
Retrying
[Pipeline] {
[Pipeline] sh
+ I am not going to work :c
/var/jenkins_home/workspace/jenkinsfirs
[Pipeline] }
[Pipeline] // retry
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 127
```

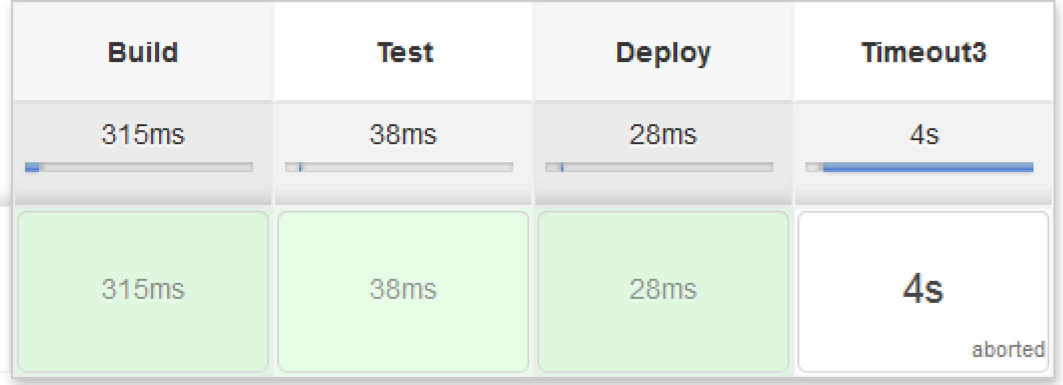
Jenkins timeout

- Let add this code

```
stage('Timeout3') {
      steps {
        retry(3) {
          sh 'echo hello'
        }

        timeout(time: 3, unit: 'SECONDS') {
          sh 'sleep 5'
        }
      }
    }
```

Jenkins timeout

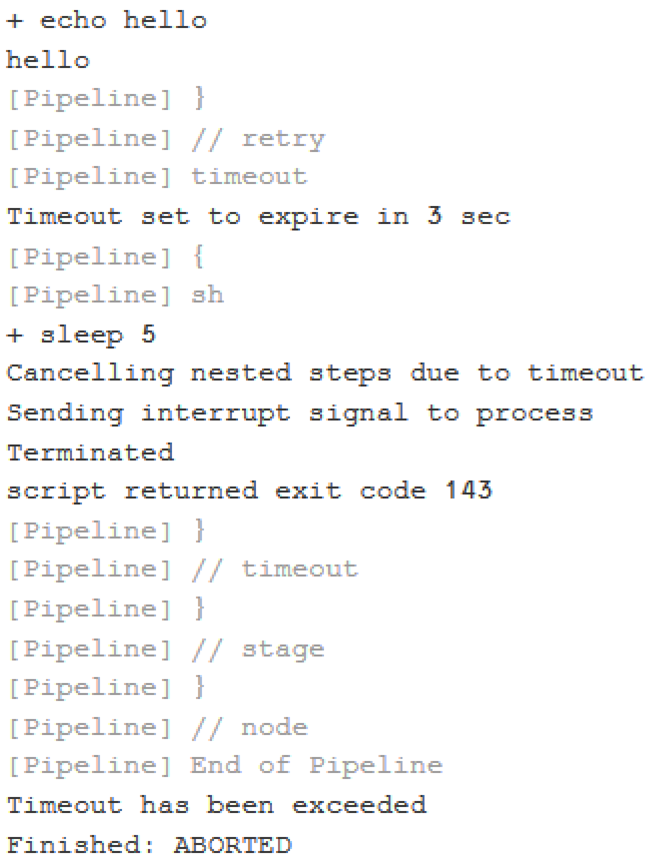
- we can see that we set the timeout to 3 second but the sleep command is 5 second
- so after we build the job



Build	Test	Deploy	Timeout3
315ms	38ms	28ms	4s
315ms	38ms	28ms	4s aborted

Jenkins timeout

- we can see that Jenkins abort
- the job because we wait more
- then 3 sec



```
+ echo hello
hello
[Pipeline] }
[Pipeline] // retry
[Pipeline] timeout
Timeout set to expire in 3 sec
[Pipeline] {
[Pipeline] sh
+ sleep 5
Cancelling nested steps due to timeout
Sending interrupt signal to process
Terminated
script returned exit code 143
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Timeout has been exceeded
Finished: ABORTED
```

Jenkins Environment variables

- We can add envs to our Jenkins file to use it in the stages
- pipeline {
- agent any

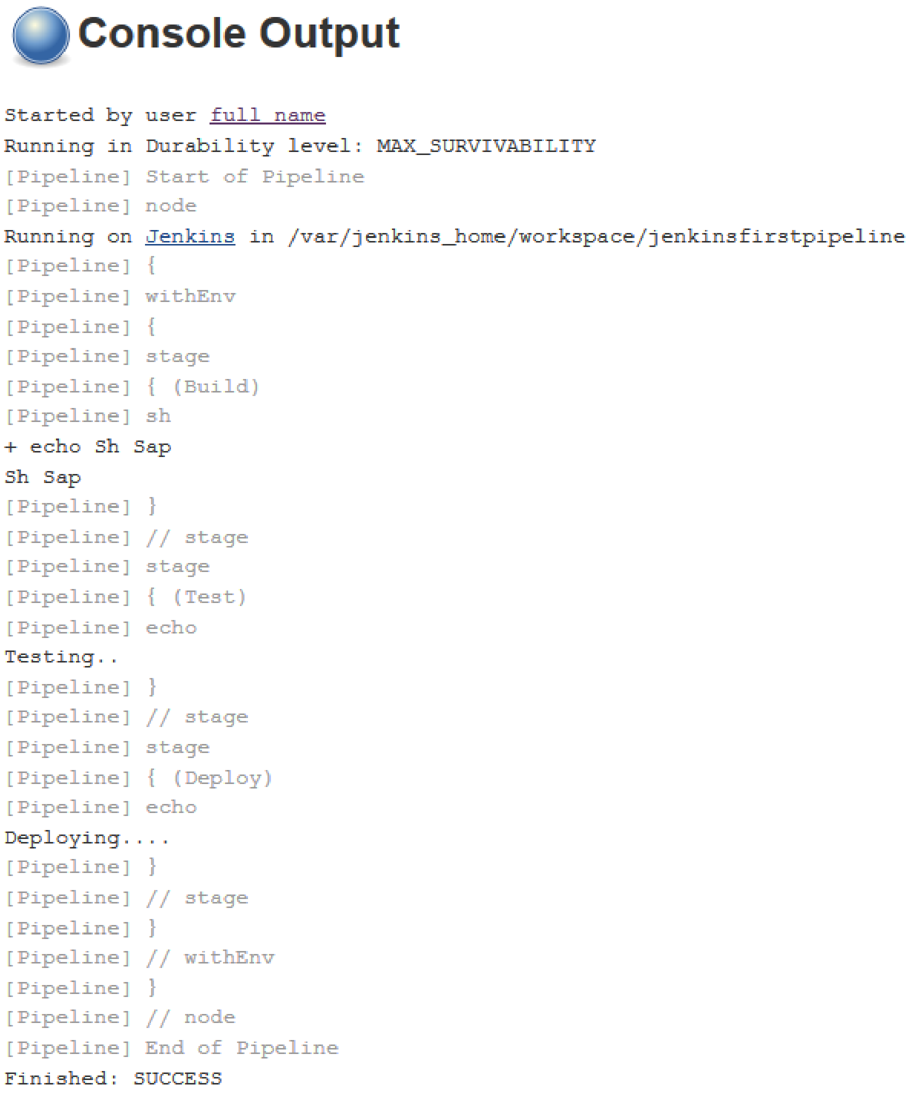
- environment {
- NAME = ‘Vl’
- LASTNAME = ‘Lev'
- }

Jenkins Environment variables

```
    stages {
        stage('Build') {
            steps {
                sh 'echo $NAME $LASTNAME'
            }
        }
    }
}
```

- now lets build the job and lets see what we got

Jenkins Environment variables

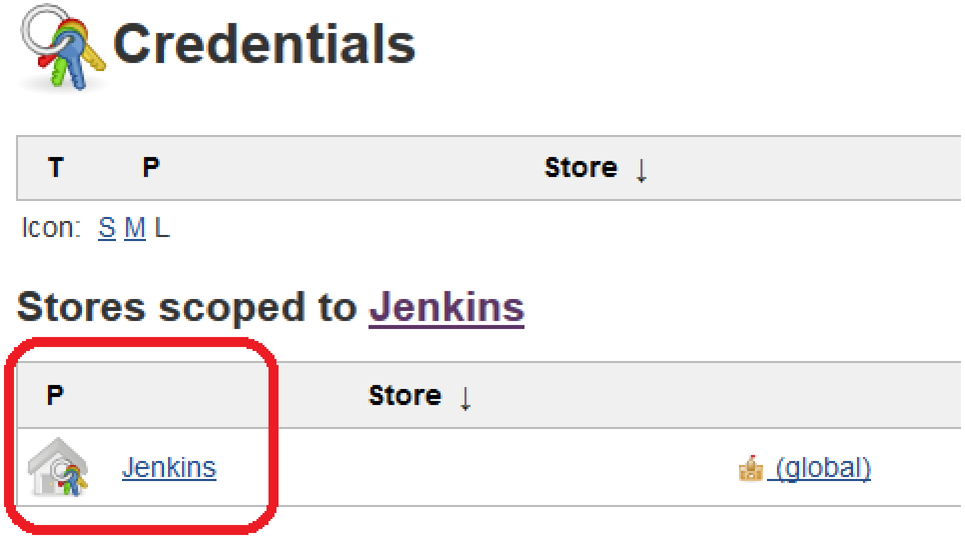


Console Output

```
Started by user full name
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/jenkinsfirstpipeline
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
+ echo Sh Sap
Sh Sap
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Testing..
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying....
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Jenkins credentials

- Let go to dashboard > Credentials > jenkins



Credentials

T	P	Store ↓

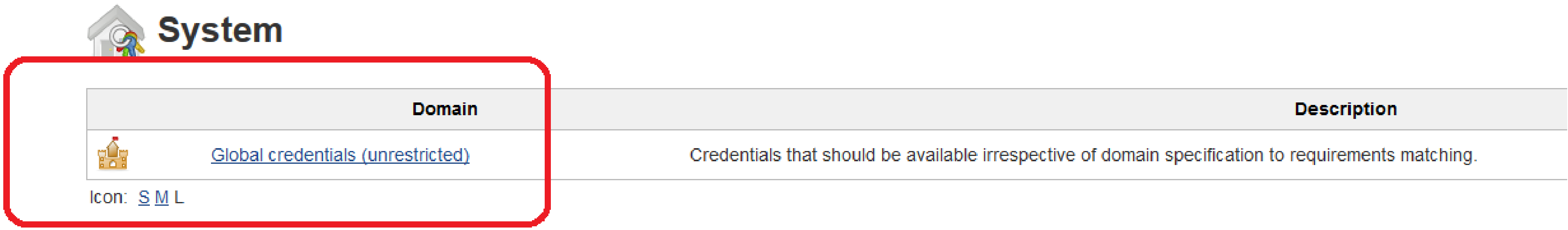
Icon: S M L

Stores scoped to Jenkins

P	Store ↓
Jenkins	(global)

Jenkins credentials

- Let go to dashboard > Credentials > Jenkins > Global …

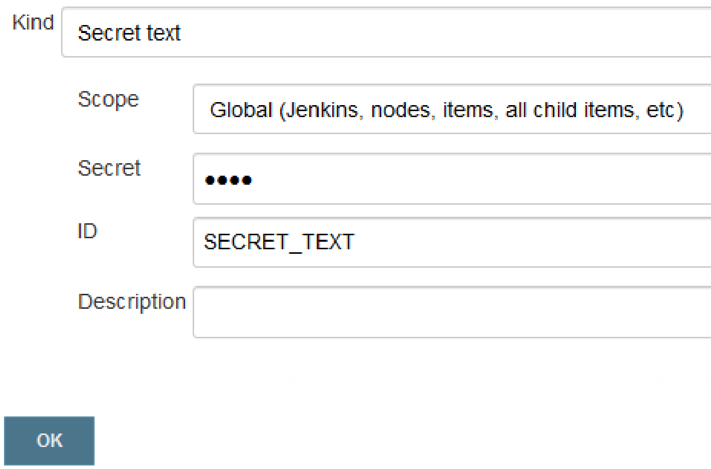


System

	Domain	Description
	Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Icon: S M L

> > choose in kind the Secret Text option



Kind	Secret text
Scope	Global (Jenkins, nodes, items, all child items, etc)
Secret	••••
ID	SECRET_TEXT
Description	

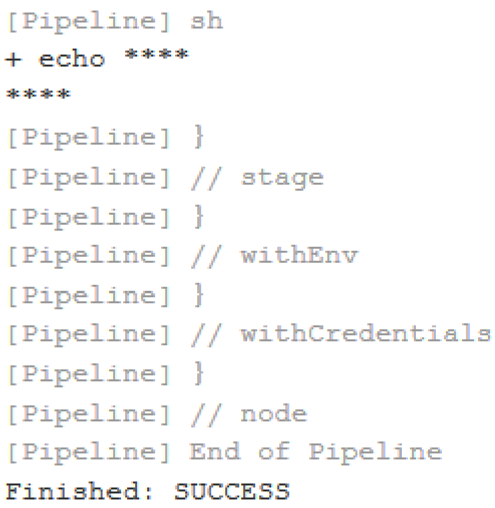
Jenkins credentials

- Now lets go to our job and add the next script

```
environment {
        secret = credentials(‘SECRET_TEXT')
    }
    stages {
        stage('Example stage 1') {
            steps {
                sh 'echo $secret'
            }
        }
    }
```

Jenkins credentials

- After we finish to build the job the we will see this in the console output



```
[Pipeline] sh
+ echo ****
****
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Jenkins Post actions

- Now lets add this script to our job

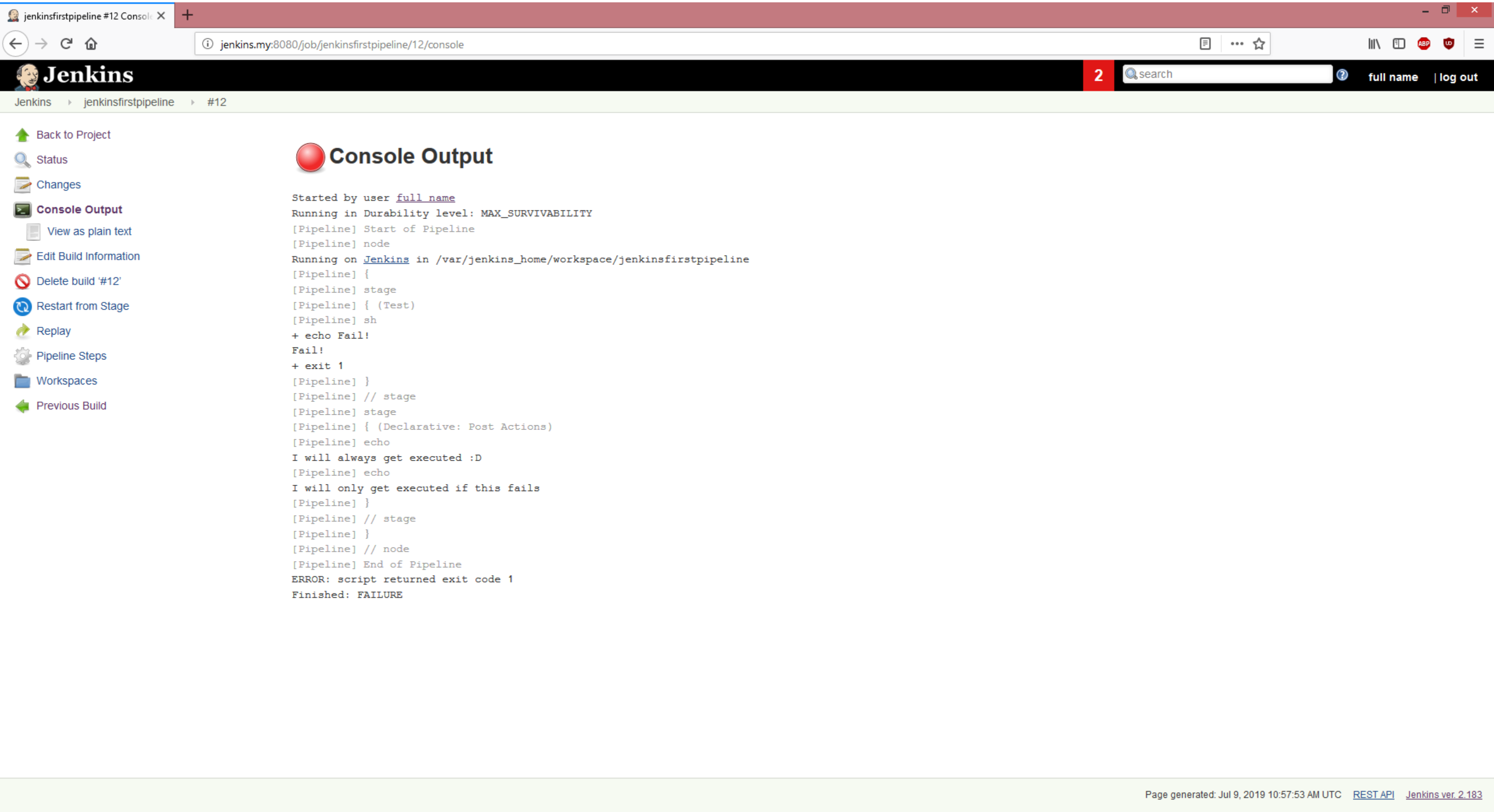
```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                sh 'echo "Fail!"; exit 1'
            }
        }
    }
```

Jenkins Post actions

```
post {
        always {
            echo 'I will always get executed :D'
        }
        success {
            echo 'I will only get executed if this success'
        }
        failure {
            echo 'I will only get executed if this fails'
        }
        unstable {
            echo 'I will only get executed if this is unstable'
        }
    }
}
```

Jenkins Post actions

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



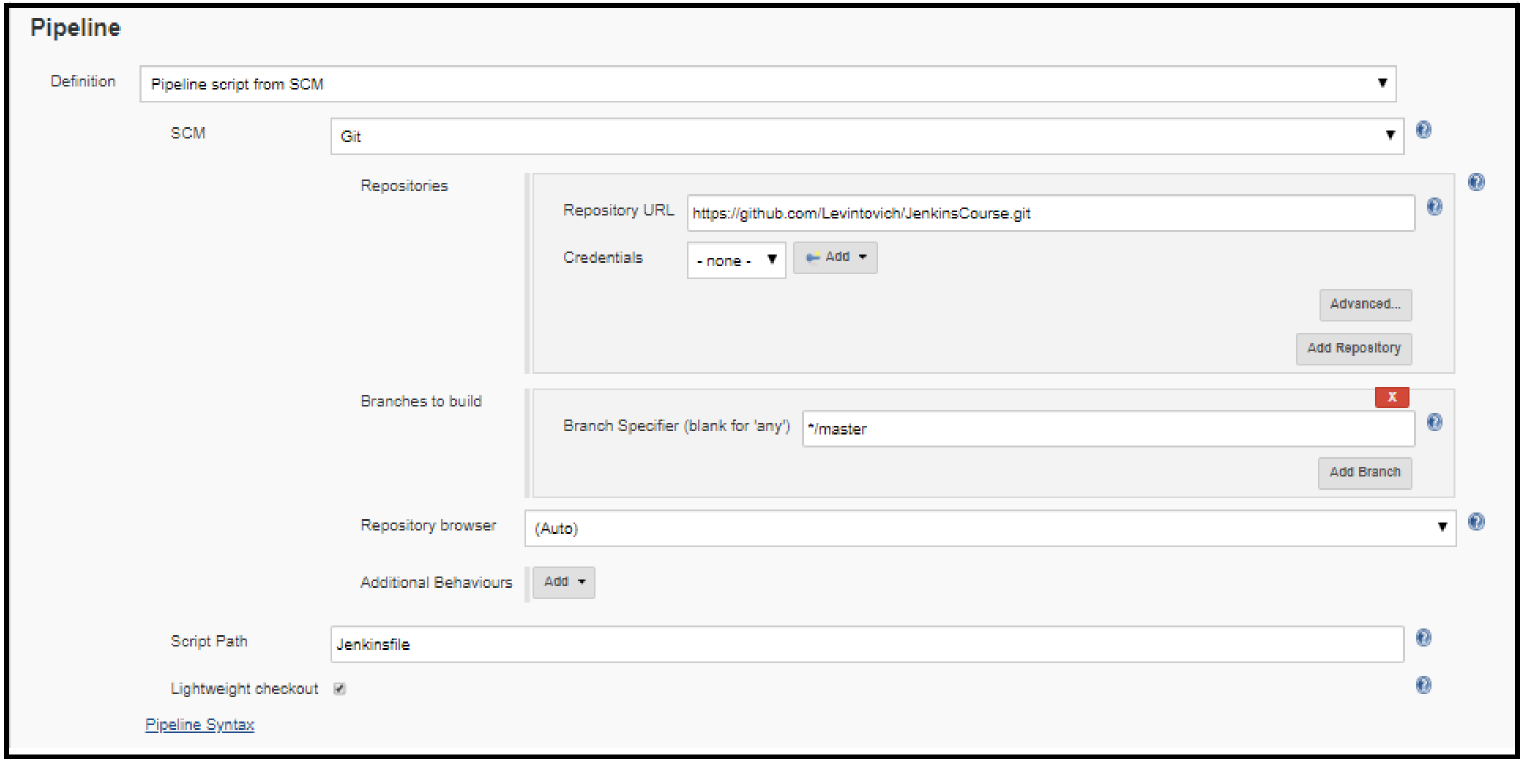
Console Output

```
Started by user full name
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/jenkinsfirstpipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh
+ echo Fail!
Fail!
+ exit 1
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
I will always get executed :D
[Pipeline] echo
I will only get executed if this fails
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

Pipeline and GitHub

Use a pipeline from GitHub

- Using the source code create and configure a new pipeline job. https://github.com/Levintovich/JenkinsCourse



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/Levintovich/JenkinsCourse.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

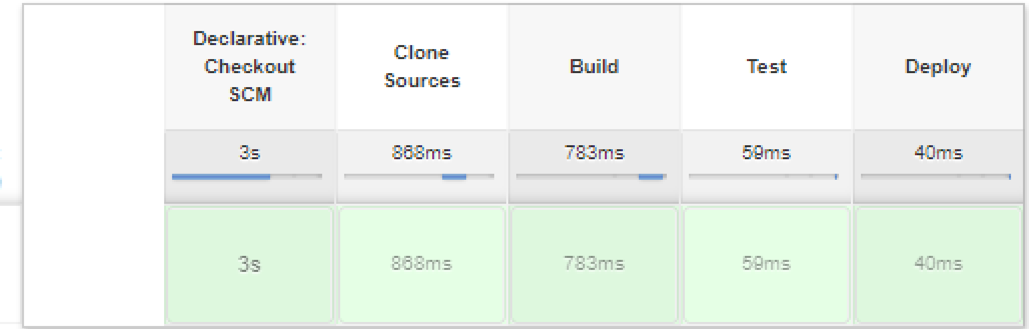
Repository browser: (Auto)

Script Path: Jenkinsfile

Lightweight checkout ☑

Use a pipeline from GitHub

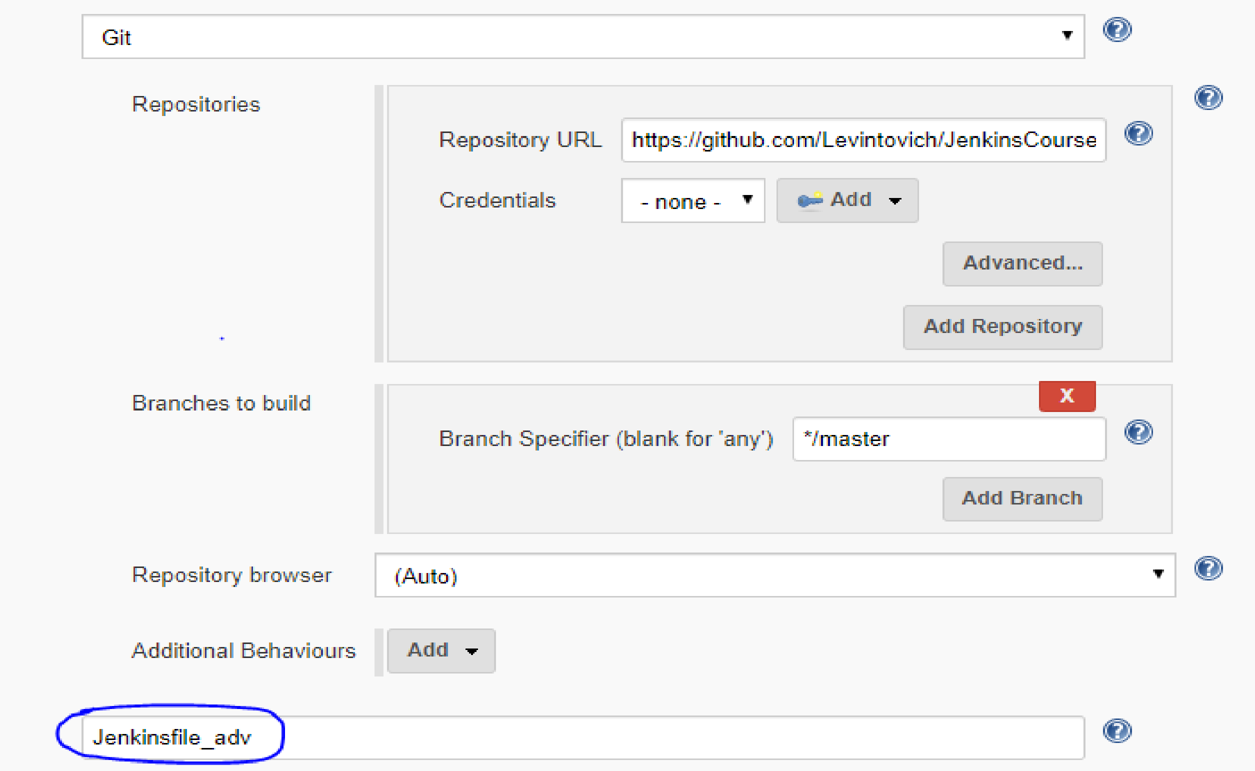
- Using the source code create and configure a new pipeline job.



Declarative: Checkout SCM	Clone Sources	Build	Test	Deploy
3s	868ms	783ms	59ms	40ms
3s	868ms	783ms	59ms	40ms

Advanced Pipeline

- Create a new pipeline job using another file **Jenkinsfile_adv**



Git

Repositories

Repository URL: https://github.com/Levintovich/JenkinsCourse

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

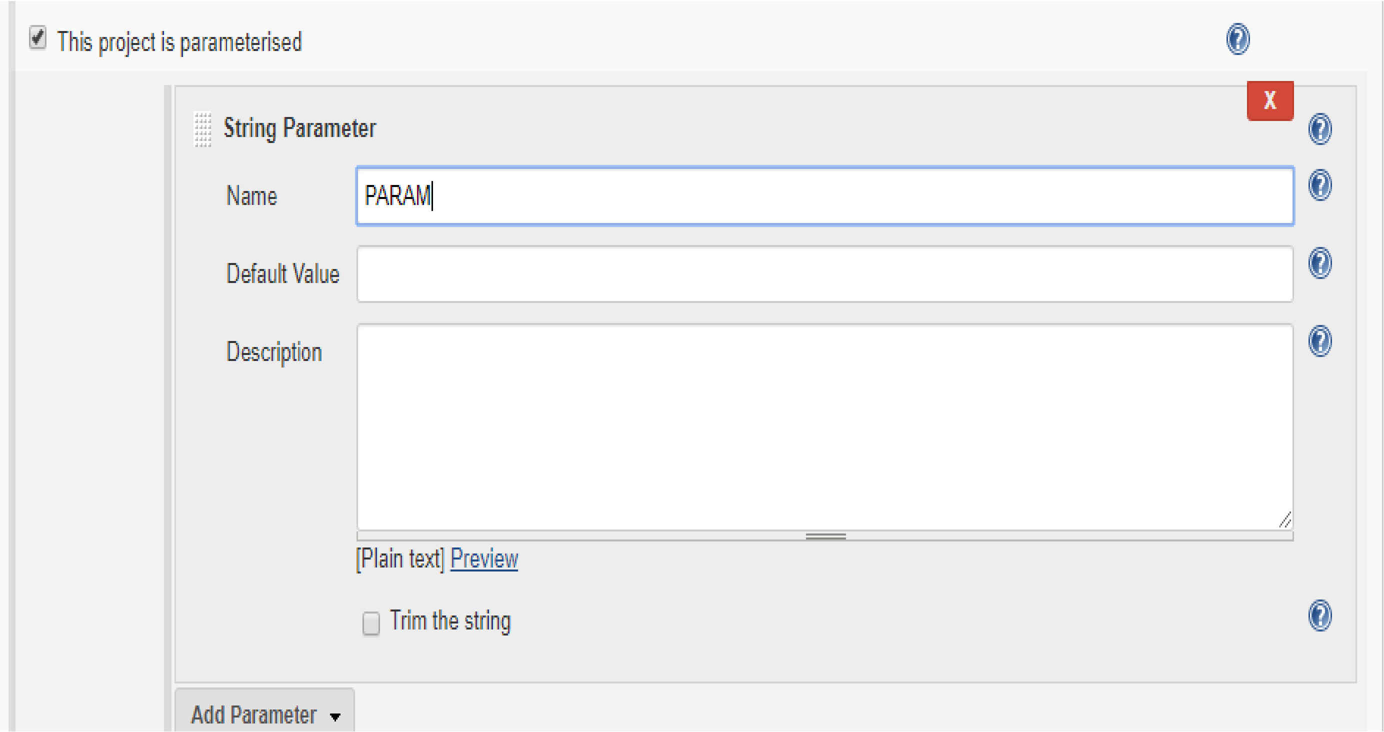
Repository browser: (Auto)

Additional Behaviours

Jenkinsfile_adv

Advanced Pipeline

- Add a new string parameter, save and then run this pipeline.



☑ This project is parameterised

String Parameter

Name: PARAM

Default Value:

Description:

[Plain text] Preview

□ Trim the string

Hope you like it ;)