VAGRANT,SSH,DOCKER,JENKINS

Installing a VM with vagrant

On CMD from Windows:

`>vagrant init generic/ubuntu2010` (Create the vagrantfile)

`>vagrant up` (install VM from Vagrant Box)

Change network adapter of the VM on VBOX from NAT to Bridge

Generating SSH KEYS and configuring SSH communication through SSH

On CMD from Windows:

`>ssh-keygen -b 4096` (This will generate the SSH keys on C:/Users/your_username/.ssh)

`>vagrant ssh` (Enter into your VM, user: vagrant, password: vagrant, group: sudoers)

On Bash from Ubuntu:

`>ifconfig` (Get your VM IP)

`>sudo adduser user` (Set username, set password in the form)

`>sudo usermod -aG sudo user` (Add your new user to the sudoers group)

`>exit`

On CMD from Windows:

`>ssh user@ip` (Login to your VM with your new user, then enter the password, just for now)

On Bash from Ubuntu:

In order to login with your private ssh key, you have to make the .ssh dir in your user's home dir and put the public ssh key inside it with the name 'authorized_keys'

`>mkdir ~/.shh` (Make de dir inside your home dir, it has to be hidden)

`>sudo chmod 700 .ssh` (Change the dir permissions)

`>exit`

On CMD from Windows:

>scp “C:/Users/your_username/.ssh/id_rsa.pub” user@ip:~/.ssh (Then enter your new ubuntu user password, not the vagrant user password in order to login to your ubuntu VM, this will upload your ssh public key from your windows host to your VM in the .ssh folder that you just created)

>ssh user@ip (Login to your VM with your new user, then enter the password, just for now)

On Bash from Ubuntu:

>cd .ssh (Go into your .ssh folder)

>mv id_rsa.pub authorized_keys (Move your public key into a file called authorized_keys)

>sudo chmod 700 authorized_keys (For security reasons your file should only be rwx with your current user as well as the .ssh folder that contains it)

>exit

On CMD from Windows:

>ssh user@ip (Now you should be able to login into your VM without entering the password)

INSTALL DOCKER ON UBUNTU

Update apt and install packages that enables your VM use repositories over HTTPS:

```
>sudo apt update

>sudo install \

    ca-certificates \

    curl \

    gnupg \

    lsb-release
```

(The backslash is to enter commands in new lines pressing enter without executing the line before)

Add the GPG official key of docker:

```
>curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg —dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
```

Configure the stable repository:

```
>echo \
```

“deb [arch=$(dpkg —print-architecture) signed-by=/usr/share/keyrings/dockerarchive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable” | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null

Install docker engine:

>sudo apt update

>sudo apt install docker-ce docker-ce-cli containerd.io (Docker engine, docker CLI and containerd)

RUN JENKINS ON A DOCKER CONTAINER

>sudo docker run -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk11

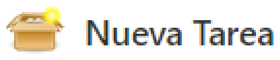
The command above will pull the official image jenkins/jenkins:lts-jdk11 from Docker hub, stablish the exposure of the 8080 port from the docker container to the 8080 port of the guest host, the same for the 50000 port, then it creates a volume called jenkins_home and links it with the /var/jenkins_home dir of the alpine of the image, you can find the volumes created on your VM in /var/lib/docker/volumes/, then it runs the image inside a container. It does not executes the container in the background, if you press ctrl+c you will stop it, you can add a ***-d*** after ***run*** to make it run in the background

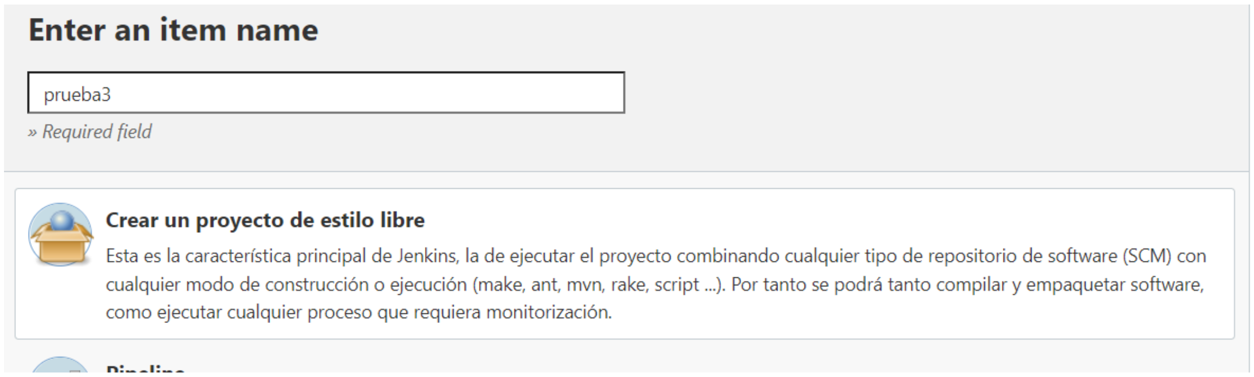
The volume created will help you persist the user you are gonna need to create for Jenkins

After the execution of the command has finished, you can register a user with the GUI of Jenkins on http://vm_ip:8080

HELLO WORLD ON JENKINS

Create a Job





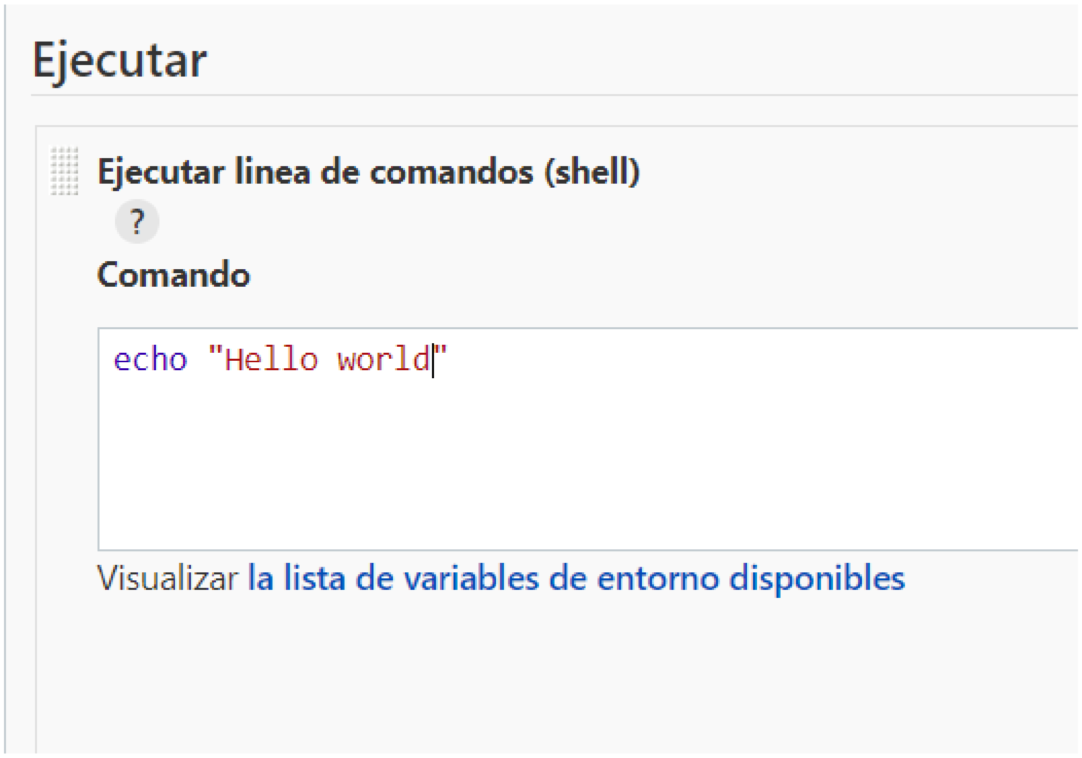
Enter an item name

prueba3

» Required field

Crear un proyecto de estilo libre

Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.



Ejecutar

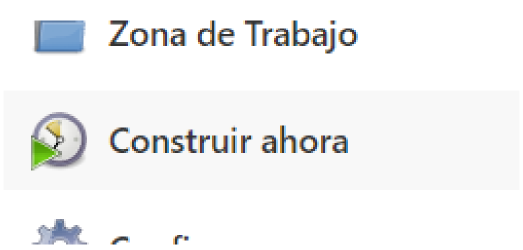
Ejecutar linea de comandos (shell)

Comando

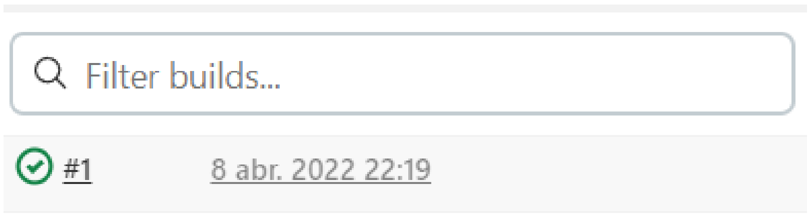
```
echo "Hello world"
```

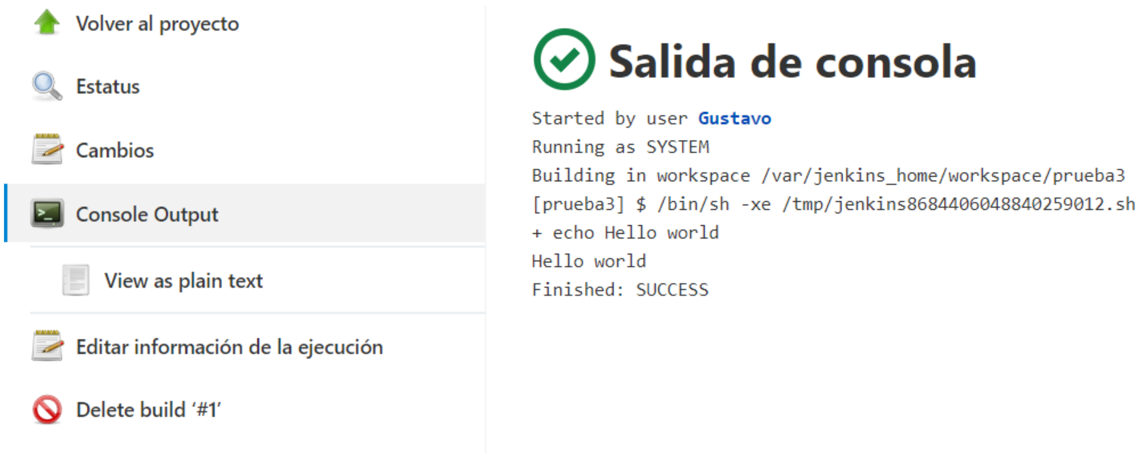
Visualizar la lista de variables de entorno disponibles

Build the Job



Watch the console output





```
Started by user Gustavo
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/prueba3
[prueba3] $ /bin/sh -xe /tmp/jenkins8684406048840259012.sh
+ echo Hello world
Hello world
Finished: SUCCESS
```