

Tiempo para aprender juntos

Dirección de Servicios de
Infraestructura y Operaciones

Mayo 2021

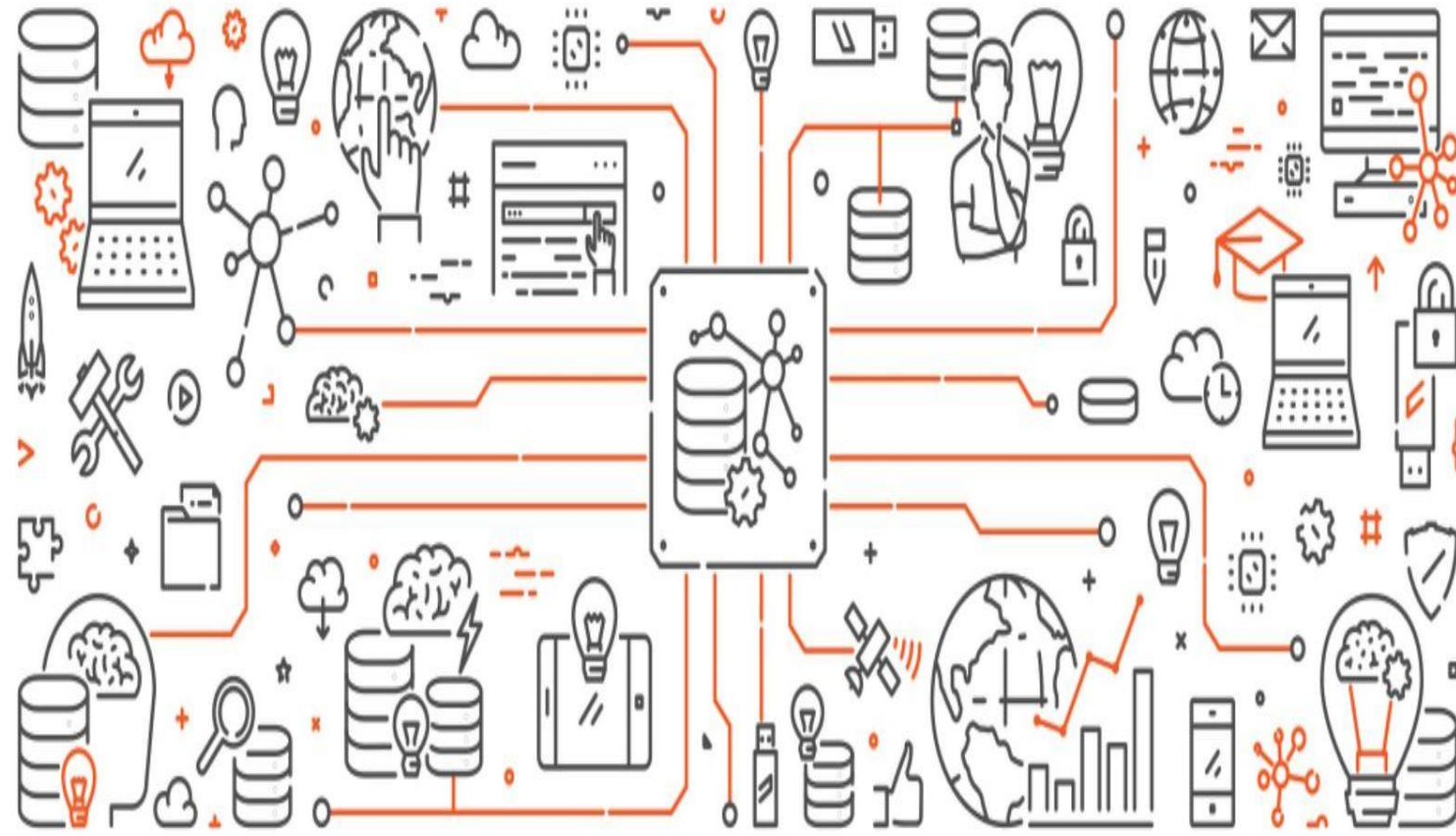
Jenkins



- ¿Qué es la Orquestación?
- Beneficios de la Orquestación
- ¿Qué es Jenkins?
- ¿Cómo aprender Jenkins?
- ¿Cómo funciona Jenkins?
- ¿Cómo se integra con otras herramientas?
- Laboratorios

¿Qué es la Orquestación?

- Es la **configuración, gestión y coordinación automatizadas** de los *sistemas informáticos, las aplicaciones y los servicios*, que ayuda a **IT** a gestionar con mayor facilidad las **tareas complejas** y los **flujos de trabajo**.



- **Ahorro de tiempo**
- **Reducción de errores**
- **Reducción de Time to Market**
- **Reducción de duración de incidentes**
- **Mejor atención al cliente**
- **Incrementa colaboración** entre los **equipos** de trabajo

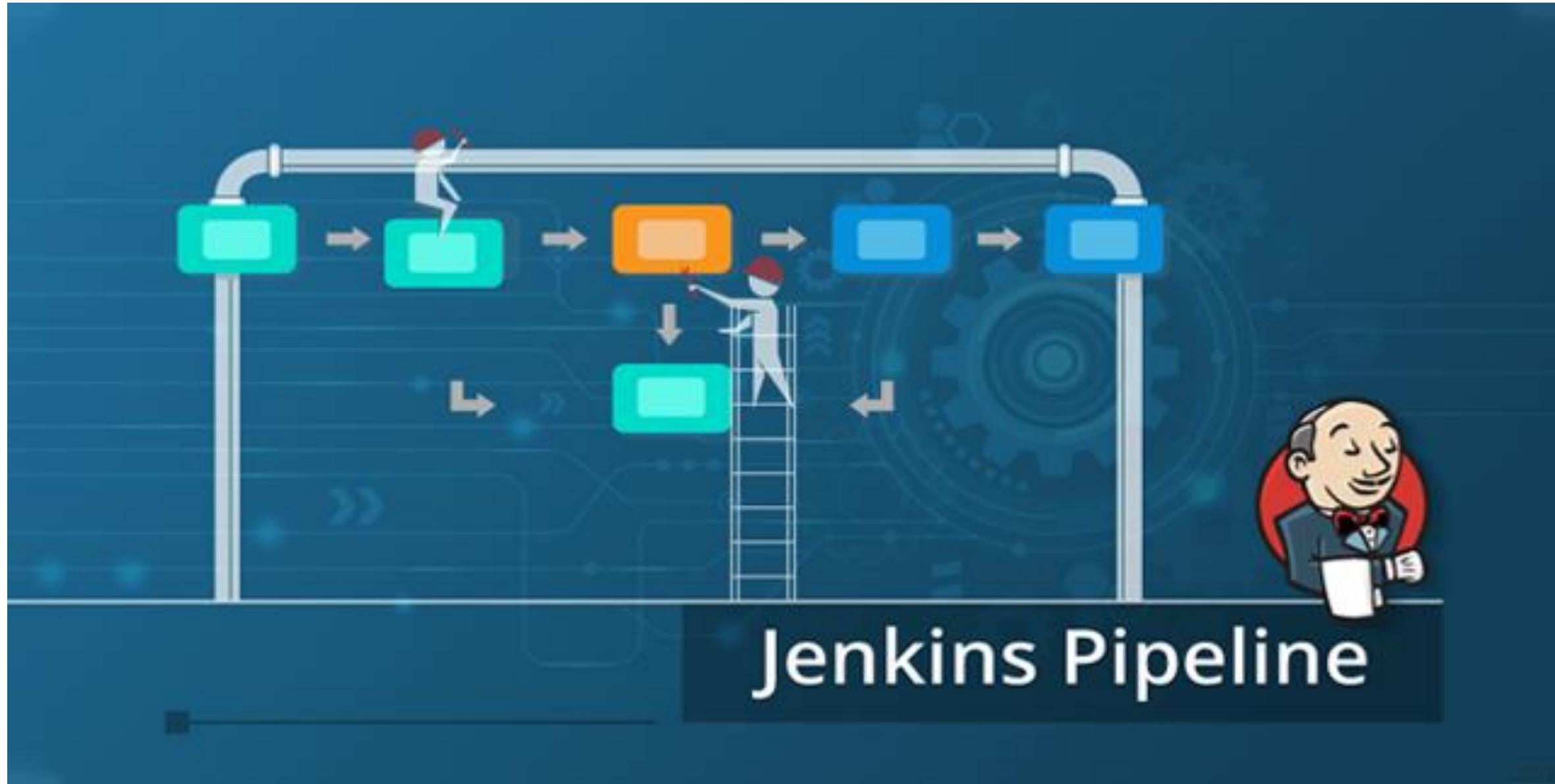
¿Qué es Jenkins?



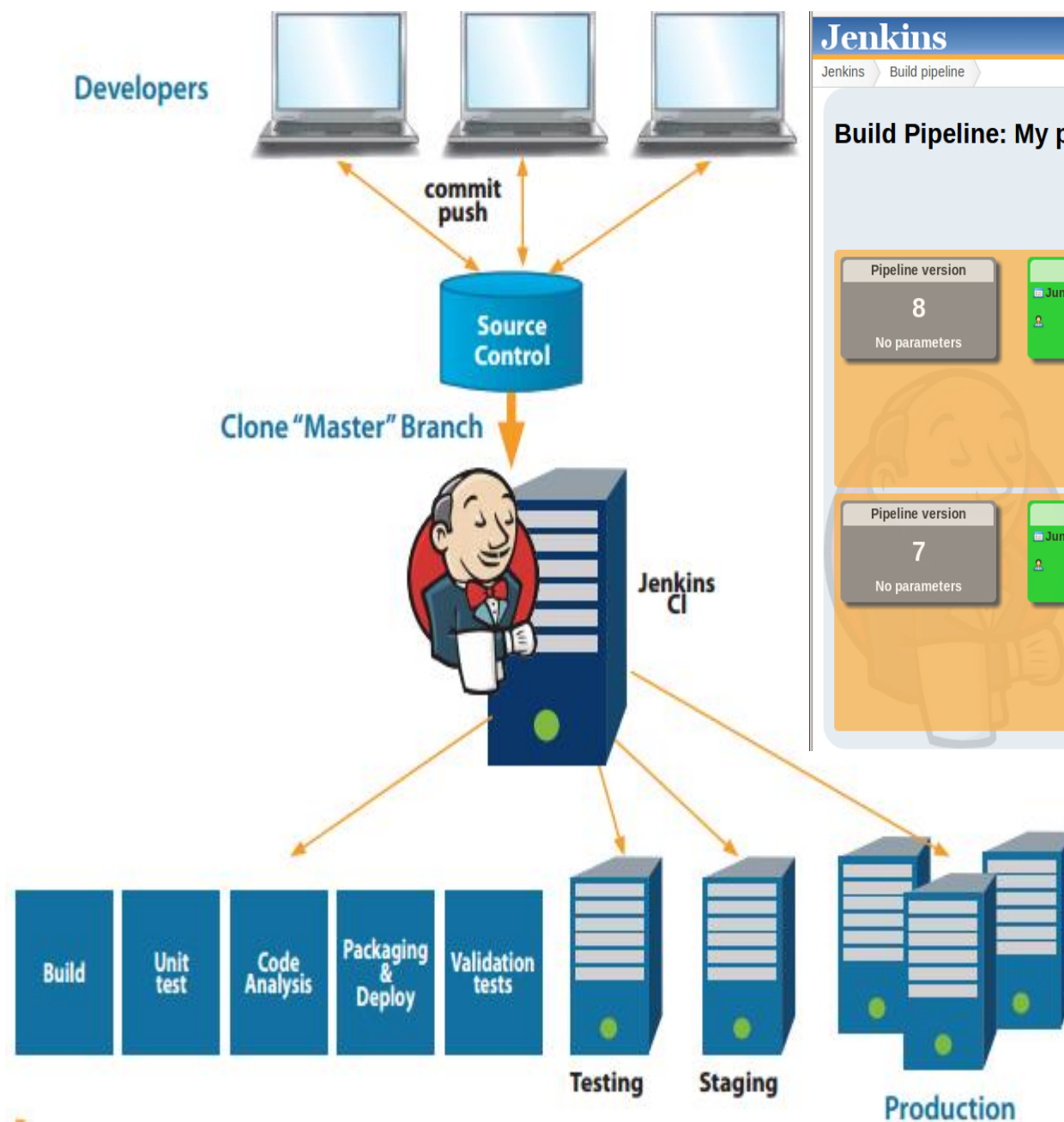
¿Qué es Jenkins?

- Es una **herramienta open source** (desarrollada en *Java*) de **integración continua**, usada principalmente para **orquestrar procesos** en el **desarrollo de software**, sus grandes **capacidades** permiten utilizarlo para mas cosas a través del uso de **pipelines**, como por ejemplo:
 - **Ejecución de procesos secuenciales**
 - **Monitoreo** de procesos
 - **Ejecución a horas determinadas**
 - **Ejecución en base a eventos**
 - **Ejecución distribuida** en varias **máquinas**
 - **Alertas** ante **problemas** en algún proceso

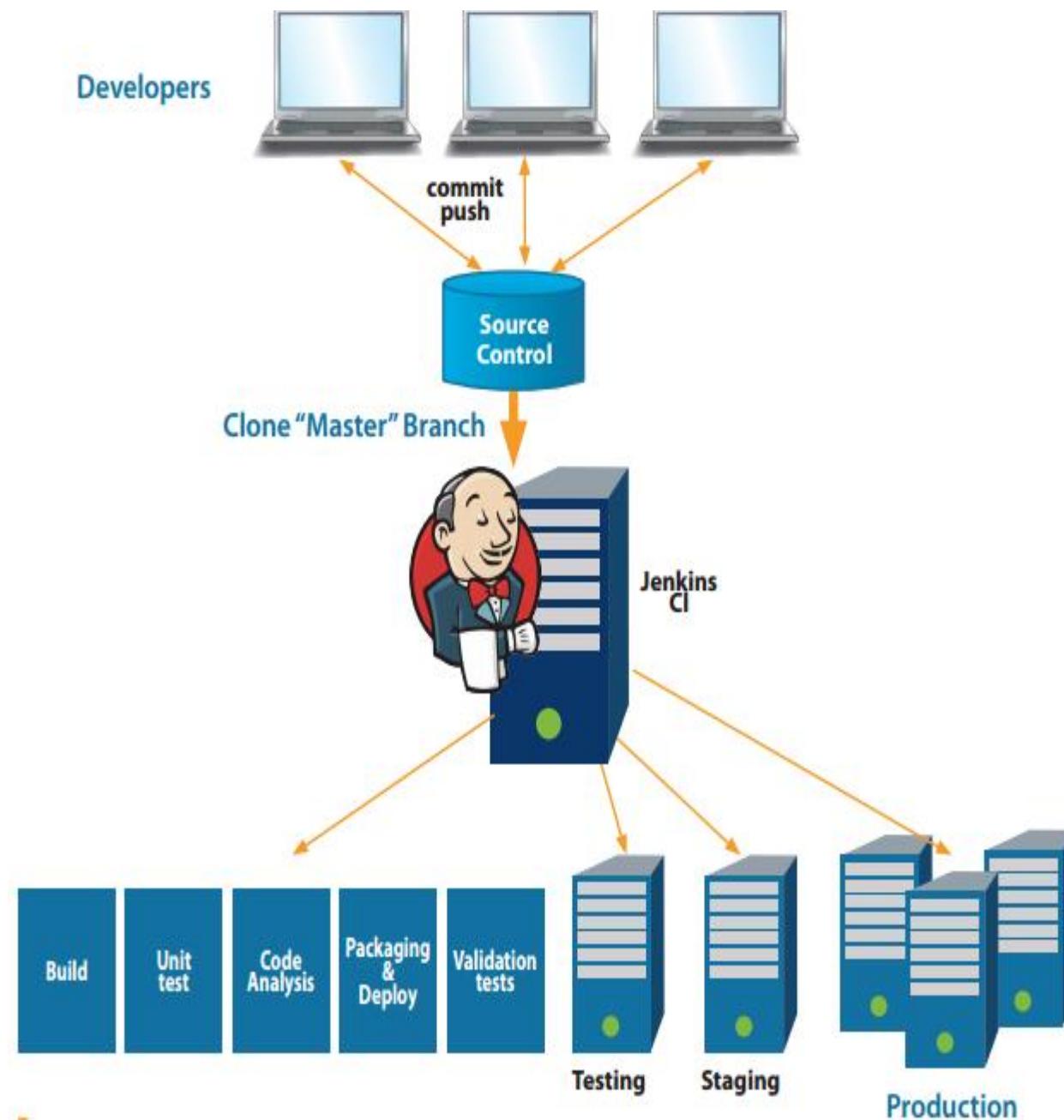
¿Qué es Jenkins?



¿Qué es Jenkins?



¿Qué es Jenkins?



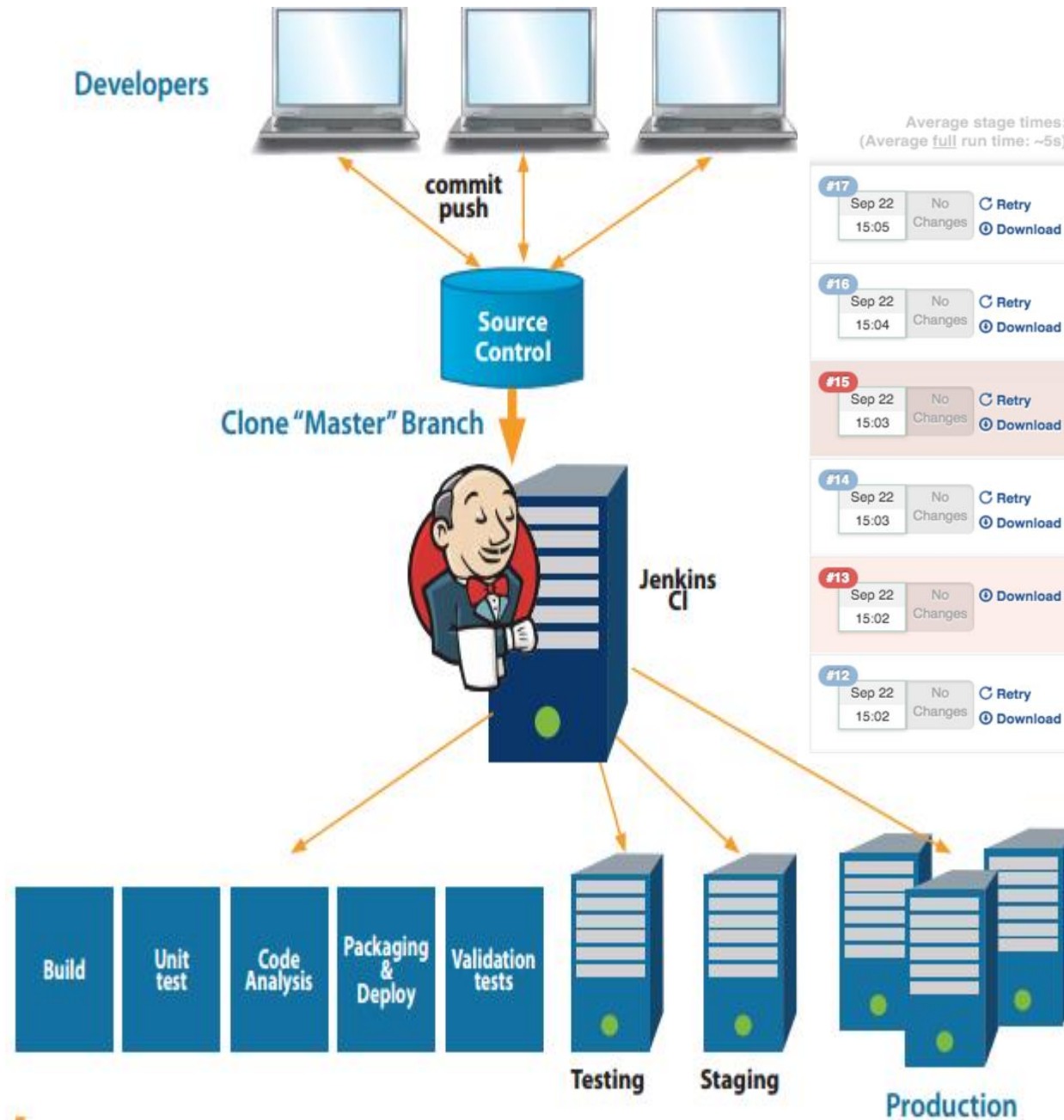
Declarative

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
      }
    }
    stage('Test') {
      steps {
      }
    }
    stage('Deploy') {
      steps {
      }
    }
  }
}
```

Scripted

```
node {
  stage('Build') {
  }
  stage('Test') {
  }
  stage('Deploy') {
  }
}
```

¿Qué es Jenkins?



Average stage times:
(Average full run time: ~5s)

	build	test: integration-&-quality	test: functional	test: load-&-security	approval	deploy: prod
#17	538ms	10s	10ms	8ms	72ms (paused for 7s)	4ms
#16	479ms	6s	9ms	9ms	74ms (paused for 6s)	5ms
#15	922ms	6s	10ms	9ms failed		
#14	1s	8s	12ms	9ms	80ms (paused for 5s)	5ms
#13	942ms	9s	13ms failed			
#12	1s	6s	13ms	11ms	111ms (paused for 5s) aborted	

¿Qué es Jenkins?

jenkins / Grpc Demo #1

✓

Branch
Commit
No changes

master
9da17cc

🕒 9 hours 31 minutes 59 seconds

🕒 a few seconds ago

Pipeline

Changes

Tests

Artifacts

⚙️

Checkout

Build Custom Environment

Build

Test

Prepare Docker Image

Build and Push Docker Image

Deploy to DEV

Verify Deployment

Deploy to LIVE

✓

✓

✓

✓

✓

✓

✓

✓

✓

Test Sample Client

Test Server

Steps - Deploy to LIVE

🔗 ⬇️

✓ > Wait for interactive input

9 hours 25 minutes 36 seconds

✓ > Shell Script

0 seconds

✓ > Shell Script

0 seconds

✓ ▾ Shell Script

1 seconds

1 [Grpc_Demo_master-NWYJ7AR6IHZBLYJKMZMD6I27AIHJXZ2LXQCPBCI6W3POFL6JSGOQ] Running shell script

2 + kubectl --kubeconfig=**** --context=LIVE rollout status deployment/grpc-demo

3 Waiting for rollout to finish: 0 of 1 updated replicas are available...

4 deployment "grpc-demo" successfully rolled out

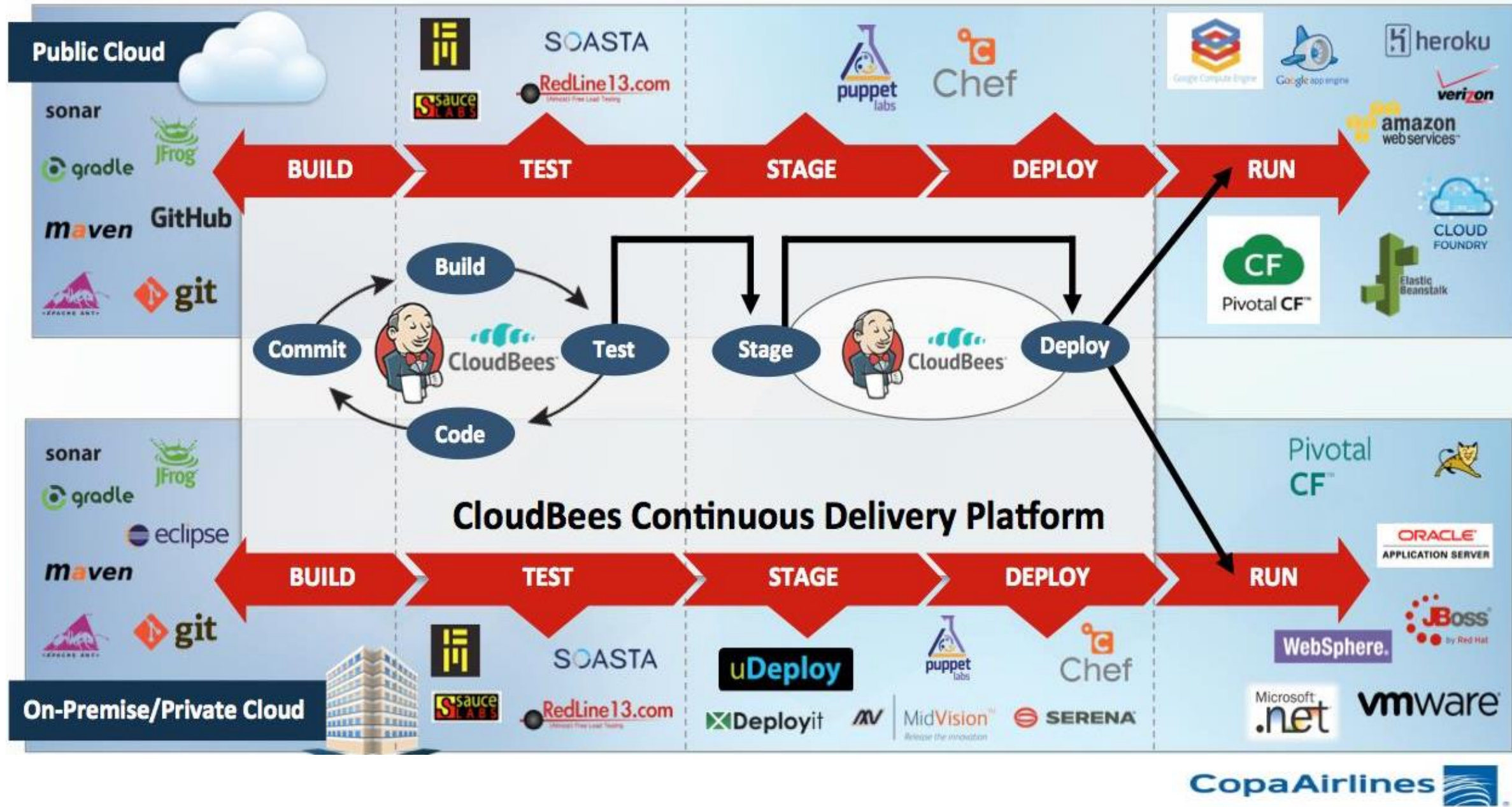
✓ > Shell Script

0 seconds

¿Cómo aprender Jenkins?

- En el sitio de Jenkins (<https://www.jenkins.io/>)
- Sobre los pipelines:
 - <https://www.jenkins.io/doc/book/pipeline/>
- **Buscando** en foros (<https://stackoverflow.com/>) o sitios (<https://medium.com/>).
- **Aplicarlo** a una **situación** en **2021** que consideras puedas resolver a través de la **orquestación**.

¿Cómo se integra con otras herramientas?



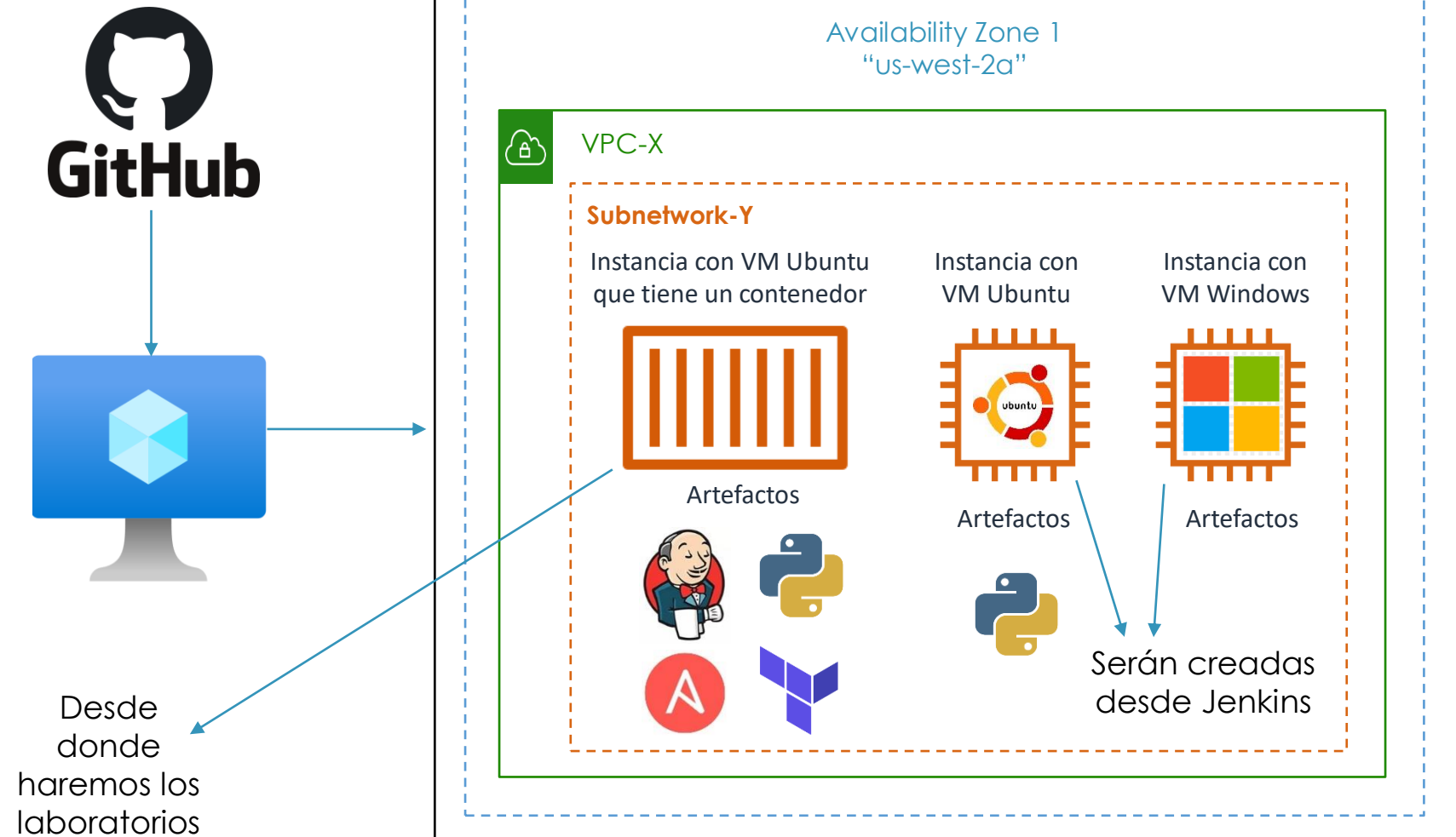


Laboratorios – Comienza la diversión

- En estos laboratorios vamos a aprender a:
 - Crear **pipelines** usando la interfaz gráfica de Jenkins:
 - Integrarlo con **GitHub**
 - Usar **Webhooks**
 - Crear **Jenkinfiles**
 - Ejecutarlos desde la línea de comando usando **curl**:
- **Nota: reutilizaremos códigos de los repositorios vistos en los cursos anteriores**



Laboratorios – Comienza la diversión





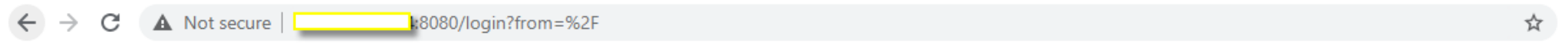
- En **Sharepoint** se encuentra el archivo con todo el inventario de VMs creadas, selecciona **1 VM** (registra en el Excel tu **nombre**).
- La VM ya tiene instalado **Jenkins 2.285**, el cual corre usando **Docker** y dentro del contenedor donde se ejecuta Jenkins se instaló **Docker**.
- Conéctate por SSH a la VM y ejecuta el siguiente comando para subir Jenkins:
 - `ssh -v -l ubuntu -i <llave> <ip_publica_vm>`
 - `docker start jenkins`
- Verifica que Jenkins esté corriendo:
 - `docker ps`

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
[REDACTED]	jenkins/jenkins:2.285-slim	jenkins	"/sbin/tini -- /usr/..."	11 days ago	Up 18 seconds	0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:49153->50000/tcp, :::49153->50000/tcp



Laboratorios – Comienza la diversión

- Conéctate usando el navegador a Jenkins:
 - http://<ip_publica_vm>:8080

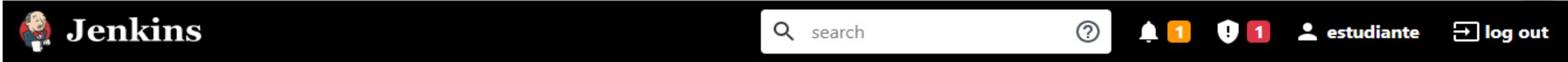


Welcome to Jenkins!

Sign in

☐ Keep me signed in

- En la sesión se facilitará la información para acceder a Jenkins

 New View

2 Idle

[Learn more about distributed builds](#)

Copa Airlines 



Laboratorios – Comienza la diversión

- Vamos a crear nuestro **primer job**

Jenkins search ? 1 1 estudiante log out

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue ^

No builds in the queue.

Build Executor Status ^

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

- El cual se llamará “**job-01**” y será del tipo “**Freestyle project**”

Enter an item name

job-01

» 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 not easily fit in free-style job type.



Laboratorios – Comienza la diversión

- Agrégale un comentario en el campo “**Description**” del tab “**General**”

The screenshot shows the Jenkins configuration interface with the 'General' tab selected. The 'Description' field is highlighted in yellow and contains the text 'Primer job en Jenkins'. Other tabs visible include 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'.

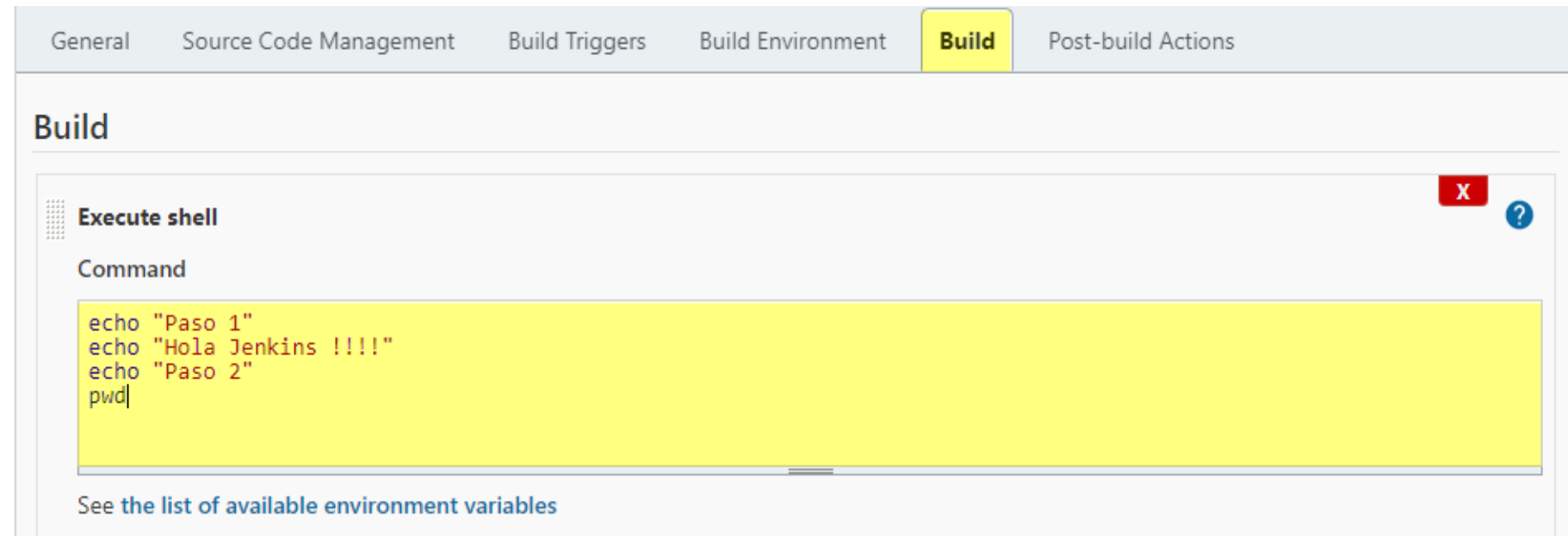
- En el tab “**Build**” selecciona “**Execute shell**”

The screenshot shows the Jenkins configuration interface with the 'Build' tab selected. Under the 'Build' section, the 'Add build step' dropdown menu is open, and 'Execute shell' is highlighted. Other options in the dropdown include 'Execute Windows batch command', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'. The 'With Ant' checkbox is also visible under the 'Build' section.

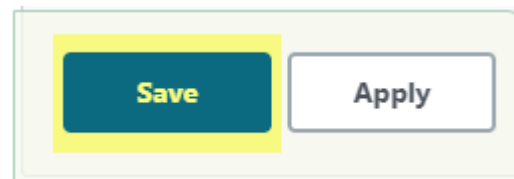


Laboratorios – Comienza la diversión

- El job ejecutará estos comandos:
 - `echo "Paso 1"`
 - `echo "Hola Jenkins !!!!!"`
 - `echo "Paso 2"`
 - `pwd`



- Salva para seguir con la ejecución





Laboratorios – Comienza la diversión

- Así aparece el job creado

[Dashboard](#) > [job-01](#) >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Configure](#)

[Delete Project](#)

[Rename](#)

[Build History](#) [trend](#) ^

Project job-01

Primer job en Jenkins

[Workspace](#)

[Recent Changes](#)

Permalinks



Laboratorios – Comienza la diversión

- Da clic en “**Build Now**” para ejecutarlo, en la sección “**Build History**” aparece un número que identifica la ejecución del job (en este caso **#1**), da clic en ese número para ver el resultado, después da clic en “**Console Output**”

Dashboard > job-01 >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now**
- Configure
- Delete Project
- Rename

Build History trend ^

find

#1 Apr 17, 2021 9:48 PM

Dashboard > job-01 > #1

- Back to Project
- Status
- Changes
- Console Output**
- View as plain text
- Edit Build Information
- Delete build '#1'

Console Output

Started by user [estudiante](#)
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/job-01
[job-01] \$ /bin/sh -xe /tmp/jenkins2369512647895429433.sh
+ echo Paso 1
Paso 1
+ echo Hola Jenkins !!!!
Hola Jenkins !!!!
+ echo Paso 2
Paso 2
+ pwd
/var/jenkins_home/workspace/job-01
Finished: SUCCESS



Laboratorios – Comienza la diversión

- Vamos a crear nuestro **segundo job**

Jenkins search ? 1 1 estudiante log out

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue ^

No builds in the queue.

Build Executor Status ^

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

- El cual se llamará “**job-02**” y será del tipo “**Freestyle project**”

Enter an item name

job-01

» 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 not easily fit in free-style job type.



Laboratorios – Comienza la diversión

- Agrégale un comentario en el campo “**Description**” del tab “**General**”

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

Description

Segundo job que imprime variables de ambiente de Jenkins así como crea un archivo en el directorio del workplace

- En el tab “**Build**” selecciona “**Execute shell**”

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

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s) ?

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant ?

Build

Add build step ^

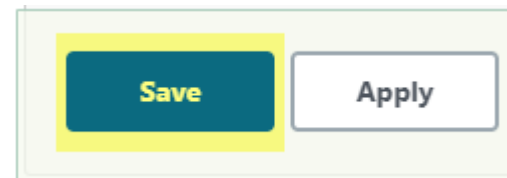
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit



Laboratorios – Comienza la diversión

- El job ejecutará estos comandos:
 - echo Paso 1 del job-02
 - echo \$BUILD_ID
 - echo Paso 2 del job-02
 - echo \$WORKSPACE
 - echo Paso 3 del job-02
 - ls -lR \$WORKSPACE
 - echo Paso 4 del job-02
 - echo "Que ondas aprendiendo Jenkins !!!" > \$WORKSPACE/file.txt
 - echo Paso 5 del job-02
 - ls -lR \$WORKSPACE
 - echo Paso 6 del job-02
 - cat \$WORKSPACE/file.txt

- Salva para seguir con la ejecución





Laboratorios – Comienza la diversión

- Así aparece el job creado

Dashboard > job-02 >

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Build History trend ^

Project job-02

Segundo job que imprime variables de ambiente de Jenkins así como crea un archivo en el directorio del workplace

Workspace

Recent Changes

Permalinks



Laboratorios – Comienza la diversión

- Da clic en “**Build Now**” para ejecutarlo, en la sección “**Build History**” aparece un número que identifica la ejecución del job (en este caso **#1**), da clic en ese número para ver el resultado, después da clic en “**Console Output**”

Dashboard > job-02 >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Configure](#)

[Delete Project](#)

[Rename](#)

[Build History](#) [trend](#) ^

find

[#1](#) Apr 17, 2021 9:57 PM

Dashboard > job-02 > #1

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#1'](#)

Console Output

Started by user [estudiante](#)
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/job-02
[job-02] \$ /bin/sh -xe /tmp/jenkins6365979626420522284.sh
+ echo Paso 1 del job-02
Paso 1 del job-02
+ echo 1
1
+ echo Paso 2 del job-02
Paso 2 del job-02
+ echo /var/jenkins_home/workspace/job-02
/var/jenkins_home/workspace/job-02
+ echo Paso 3 del job-02
Paso 3 del job-02
+ ls -lR /var/jenkins_home/workspace/job-02
/var/jenkins_home/workspace/job-02:
total 0
+ echo Paso 4 del job-02
Paso 4 del job-02
+ echo Que ondas aprendiendo Jenkins !!!
+ echo Paso 5 del job-02
Paso 5 del job-02
+ ls -lR /var/jenkins_home/workspace/job-02
/var/jenkins_home/workspace/job-02:
total 4
-rw-r--r-- 1 jenkins jenkins 34 Apr 17 22:06 file.txt

Paso 6 del job-02
+ cat /var/jenkins_home/workspace/job-02/file.txt
Que ondas aprendiendo Jenkins !!!
Finished: SUCCESS



- Vamos a crear nuestro **tercer job**
 - Nombre: “**job-03**”
 - Descripción: “**Job para obtener información de docker, ansible, python y terraform así como los contenedores corriendo**”
 - Tipo “**Freestyle project**”
 - Ejecutará:
 - docker version
 - docker ps
 - docker image ls
 - ansible --version
 - python3 --version
 - terraform --version



- **Reto 1 – Crea un job**
 - Nombre “**reto_1**”
 - Tipo: Free Style
 - Que imprima
 - El día del año en curso
 - El nombre del día de la semana
 - El nombre completo del mes
 - El año



Laboratorios – Comienza la diversión

- Vamos a crear un **cuarto job**
 - Nombre: “**job-04**”
 - Descripción: “**Job para clonar el repositorio de GitHub del curso de bash y posteriormente ejecuta 3 scripts**”
 - Tipo “**Freestyle project**”



Laboratorios – Comienza la diversión

- En la sección de “**Source Code Management**” en “Repository URL” escribe “**https://github.com/HugoAquinoNavarrete/bash_scripting**” y en “**Branch Specifier**” déjalo en blanco.

The screenshot shows the Jenkins configuration interface for 'Source Code Management'. The tabs at the top are 'General', 'Source Code Management' (selected), 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/HugoAquinoNavarrete/bash_scripting' and a 'Credentials' dropdown set to '- none -'. There is an 'Add' button next to the credentials dropdown. The 'Branches to build' section contains a 'Branch Specifier (blank for 'any')' field which is empty. The 'Add Repository' button is located at the bottom right of the 'Repositories' section.



Laboratorios – Comienza la diversión

- En la sección de “**Build**” copia estos comandos:
 - `pwd`
 - `cd bin`
 - `ls -l`
 - `bash 00-primer_script.sh`
 - `bash 01-imprime_variables.sh`
 - `bash 06-mi_entorno.sh`



- Sálvalo y córralo



Laboratorios – Comienza la diversión

Dashboard ▶ job-04 ▶ #1

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#1'

Git Build Data

Console Output

Started by user [estudiante](#)

Running as SYSTEM

Building in workspace /var/jenkins_home/workspace/job-04

The recommended git tool is: NONE

No credentials specified

```
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/job-04/.git # timeout=10
```

Fetching changes from the remote Git repository

```
> git config remote.origin.url https://github.com/HugoAquinoNavarrete/bash_scripting # timeout=10
```

Fetching upstream changes from https://github.com/HugoAquinoNavarrete/bash_scripting

```
> git --version # timeout=10
```

```
> git --version # 'git version 2.20.1'
```

```
> git fetch --tags --force --progress -- https://github.com/HugoAquinoNavarrete/bash_scripting +refs/heads/*:refs/remotes/origin/* # timeout=10
```

Seen branch in repository origin/main

Seen 1 remote branch

```
> git show-ref --tags -d # timeout=10
```

Checking out Revision 025e07bfb4b0cf4b4edd69f6789825c549643ab8 (origin/main)

```
> git config core.sparsecheckout # timeout=10
```

```
> git checkout -f 025e07bfb4b0cf4b4edd69f6789825c549643ab8 # timeout=10
```

Commit message: "Ajuste a presentación para reflejar cambio en sitio para hacer fork al repositorio"

First time build. Skipping changelog.

```
[job-04] $ /bin/sh -xe /tmp/jenkins2143680279553034611.sh
```

```
+ pwd
```

```
/var/jenkins_home/workspace/job-04
```

```
+ cd bin
```

```
+ ls -l
```

```
+ bash 06-mi_entorno.sh
```

```
El nombre del usuario es:
```

```
El nombre del equipo es: d7201c6c9a88
```

```
El shell que usamos es: /bin/bash
```

```
El directorio donde nos encontramos es: /var/jenkins_home/workspace/job-04/bin
```

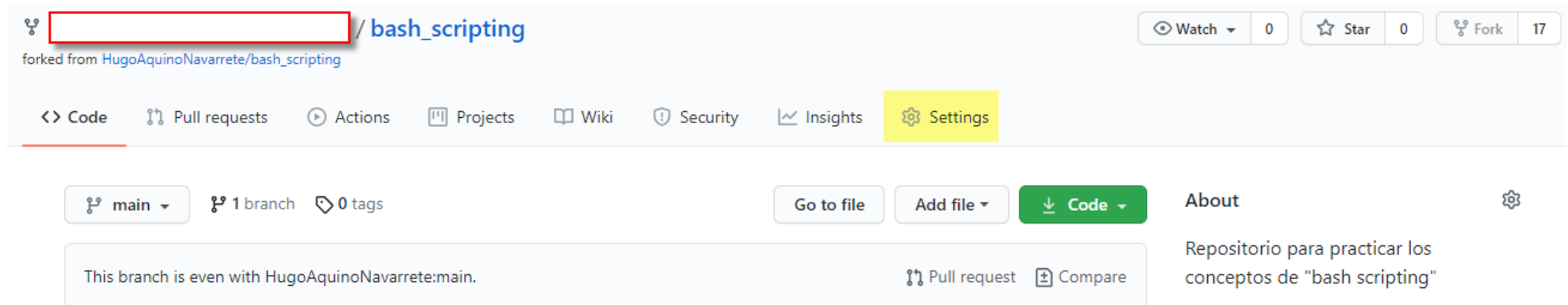
```
La ruta de búsqueda de comandos es: /opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

```
Finished: SUCCESS
```

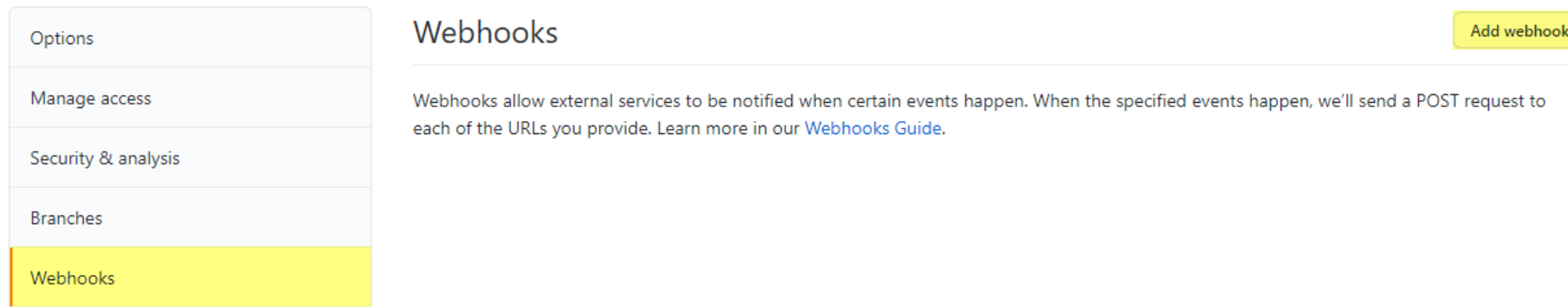


Laboratorios – Comienza la diversión

- En tu cuenta de **GitHub**, haz un **fork** del siguiente repositorio:
 - https://github.com/HugoAquinoNavarrete/bash_scripting
- Dentro del repositorio, da clic en **Settings**:



- Da clic en **Webhooks** y después en “**Add webhook**”





Laboratorios – Comienza la diversión

- En “**Payload URL**” completa esta información:
 - `http://<ip_publica_vm>:8080/github-webhook/`

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

`http://34.214.103.247:8080/github-webhook/`

Content type

`application/json`

Secret

Which events would you like to trigger this webhook?

☐ Just the push event.

☐ Send me everything.

☒ Let me select individual events.

- Selecciona “**Pull requests**” y “**Pushes**”, finalmente da clic en “**Add webhook**”

☒ **Pull requests**
Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, synchronized, ready for review, converted to draft, locked, unlocked, auto merge enabled, auto merge disabled, milestone, or demilestone.

☒ **Pushes**
Git push to a repository.

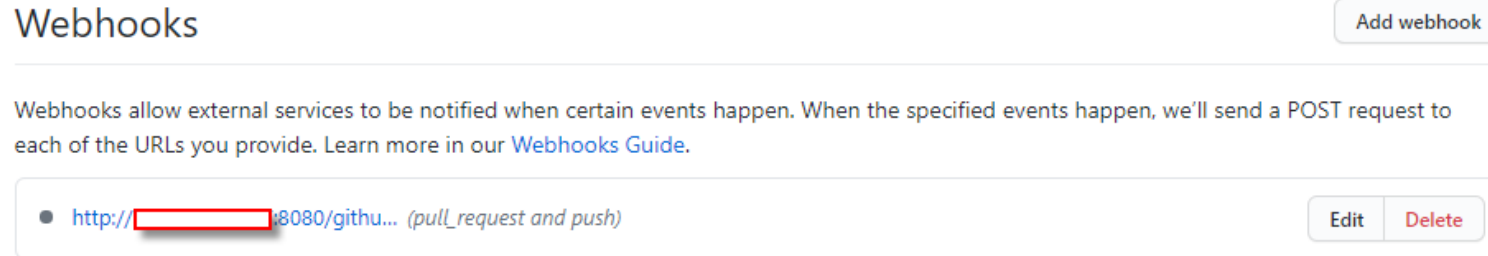
☒ **Active**
We will deliver event details when this hook is triggered.

Add webhook



Laboratorios – Comienza la diversión

- En la pantalla aparecerá el **webhook** creado:



- En **Jenkins** vamos a crear un **quinto job**
 - Nombre: “**job-05**”
 - Tipo “**Freestyle project**”
 - Descripción: “**Job para verificar el uso de un webhook**”
 - Source Code Management: “**https://github.com/<usuario>/bash_scripting**”
 - Branch Specifier: dejarlo vacío
 - En Build Triggers selecciona “**GitHub hook trigger for GITScm polling**”

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☒ GitHub hook trigger for GITScm polling
- ☐ Poll SCM





Laboratorios – Comienza la diversión

- En **Build**, selecciona “**Execute a script**” que contenga
 - echo “Aprendiendo el uso de los webhooks”
 - echo “Al hacer un commit al repositorio”
 - echo “Se ejecutará este job”
 - echo “Este es el ID:” \$BUILD_ID
 - date
- En tu laptop clona el repositorio
 - git clone https://github.com/<usuario_git_hub>/bash_scripting
- Edita el archivo “**README.md**”, agrega al final del archivo una línea en blanco y posteriormente haz un commit al repositorio:
 - nano README.md
 - git add README.md
 - git commit -m “Ajuste en README.md para ver funcionamiento de un webhook”
 - git push



- En **Jenkins** vamos a crear un **sexto job**
 - Nombre: “**job-06**”
 - Tipo “**Freestyle project**”
 - Copy from “**job-05**”
 - Descripción: “**Job para verificar el uso de un webhook**”
- En **Build**, dentro de “**Execute a script**” borra lo anterior y agrega
 - `cd bin`
 - `bash 11-mlb_equipos.sh`
- En el repositorio clonado en tu laptop edita el archivo **year.txt** y cámbiale el valor y una vez que hagas **commit** mira lo que aparece en Jenkins
 - `nano input/year.txt`
 - `git add input/year.txt`
 - `git commit -m “Ajuste en year.txt para ver funcionamiento de un webhook”`
 - `git push`



- **Reto 2 – Crea un job**

- Nombre “reto_2”

- Tipo: Free Style

- Haz “**Fork**” al siguiente repositorio:

- https://github.com/HugoAquinoNavarrete/python_scripting

- Haz lo siguiente:

- Configura un “**webhook**” (reutiliza el que ya creaste)

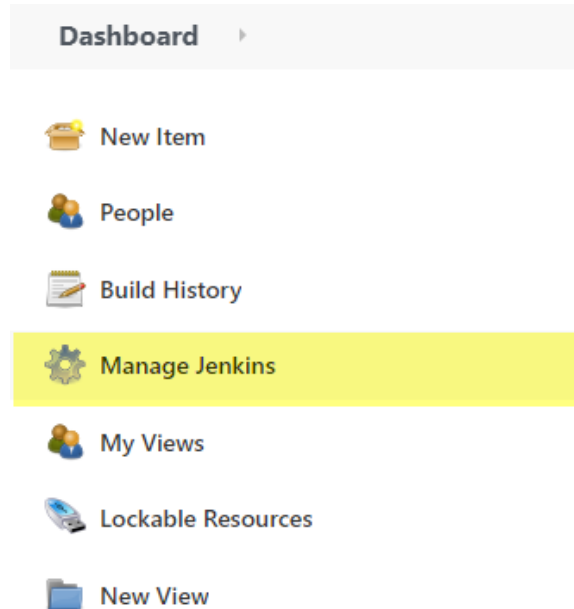
- En “**Build**” se tienen que ejecutar 2 scripts del directorio “**bin**”

- Clona el repositorio en tu laptop, edita el archivo “**README.md**” y haz un commit en git para lograr una ejecución automática del job.



Laboratorios – Comienza la diversión

- Vamos a instalar un **plugin**. Da clic en “**Dashboard**” -> “**Manage Jenkins**”



- En “**System Configuration**” da clic en “**Manage Plugins**”

System Configuration



Configure System

Configure global settings and paths.



Global Tool Configuration

Configure tools, their locations and automatic installers.



Manage Plugins

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

🔴 There are updates available

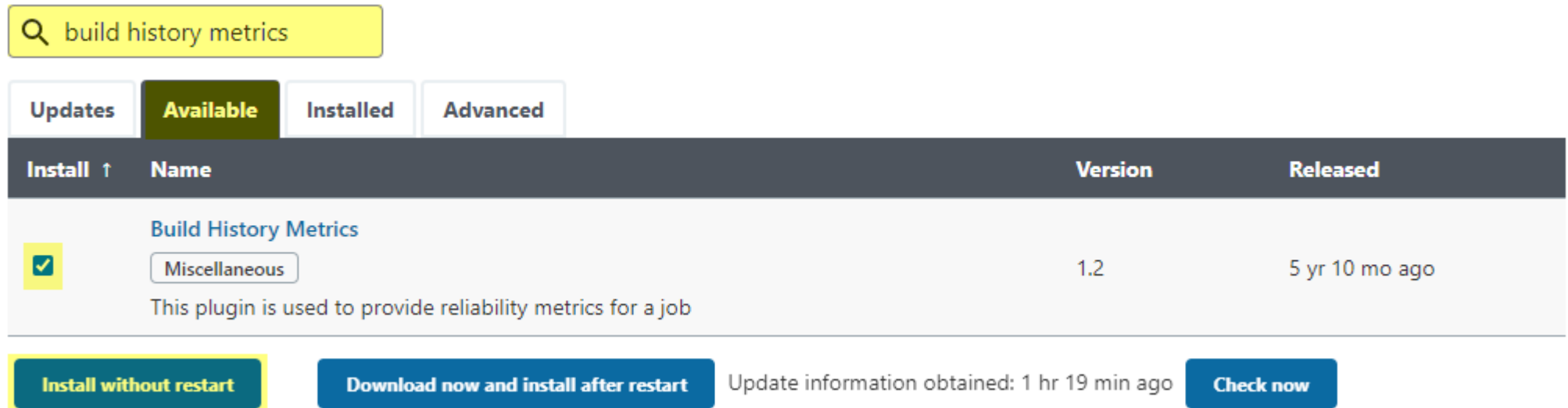


Laboratorios – Comienza la diversión

- Da clic en **“Available”**



- En **“Search”** escribe **“build history metrics”**, **“build with parameters”**, **“build pipeline”**, **“ansi color”** y **“build monitor view”** selecciónalo y da clic en **“Install without restart”**.





Laboratorios – Comienza la diversión

- Después de unos segundos el estado pasará de “**Pending**” a “**Success**” y da clic en “**Restart Jenkins when installation is complete and no Jobs are running**”, espera unos segundos mientras carga Jenkins para que te conectes nuevamente.

Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity

Build History Metrics

⋯ Pending

Loading plugin extensions

⋯ Pending

➡ [Go back to the top page](#)
(you can start using the installed plugins right away)

➡ ☐ Restart Jenkins when installation is complete and no jobs are running

Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Build History Metrics

✓ Success

Loading plugin extensions

✓ Success

➡ [Go back to the top page](#)
(you can start using the installed plugins right away)

➡ ☐ Restart Jenkins when installation is complete and no jobs are running



Please wait while Jenkins is restarting ...

Your browser will reload automatically when Jenkins is ready.



Laboratorios – Comienza la diversión

- Ve a alguno de los Jobs que se han creado para que veas las estadísticas.

Project job-05

Job para verificar el uso de un webhook



Workspace



Recent Changes

MTTR	Last 7 Days	0 ms
	Last 30 Days	0 ms
	All Time	0 ms
MTTF	Last 7 Days	0 ms
	Last 30 Days	0 ms
	All Time	0 ms
Standard Deviation	Last 7 Days	5.1 sec
	Last 30 Days	5.1 sec
	All Time	5.1 sec



Laboratorios – Comienza la diversión

- Da clic en el signo “+” para agregar una nueva vista.

Dashboard ▾

New Item

People

Build History

Manage Jenkins

My Views

add description

All	+	S	W	Name ↓	Last Success	Last Failure	Last Duration	
		job-01	44 min - #4	N/A	11 ms			
		job-02	1 hr 41 min - #1	N/A	9 ms			
		job-03	1 hr 39 min - #1	N/A	16 sec			

- En “**View name**” agrega “**Jenkins – Lab**” y selecciona “**Build Monitor View**”

View name

Jenkins - Lab

☒ **Build Monitor View**

Shows a highly visible status of selected jobs. Ideal as an Extreme Feedback Device to be displayed on a screen on your office wall.

☐ **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.

OK



Laboratorios – Comienza la diversión

- Da clic en “**Use a regular expresión to include jobs into the view**” y escribe “**(job-[0-9]+) | (pipeline-[0-9]+) | (jenkinsfile-[0-9]+)**”

☒ Use a regular expression to include jobs into the view



Regular expression

```
(job-[0-9]+)|(pipeline-[0-9]+)|(jenkinsfile-[0-9]+)
```

- Obsérvalos en la vista



Jenkins - Lab



job-01 #4 54 minutes ago	job-02 #1 2 hours ago
job-03 #1 2 hours ago	job-04 #1 a moment ago
job-05 #3 54 minutes ago	job-06 #1 2 hours ago

Build Monitor version 1.12+build.201809061734 brought to you by Jan Molak



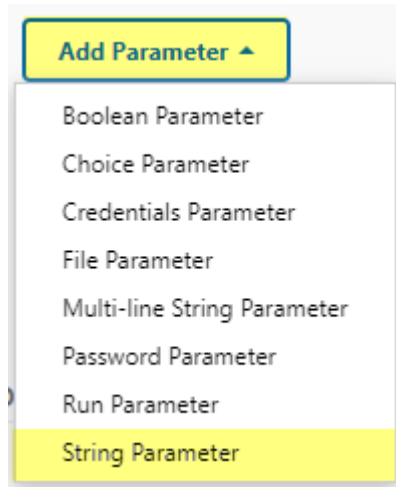
- En **Jenkins** vamos a crear un **séptimo job**
 - Nombre: “**job-07**”
 - Tipo “**Freestyle project**”
 - Descripción: “**Job para verificar el uso de argumentos**”
 - Da clic en “**This Project is parametrized**”

The screenshot shows the Jenkins job configuration interface. At the top, there are tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is selected. Below the tabs, there is a 'Description' section with a text area containing the text 'Job para verificar el uso de argumentos'. Below the description, there is a section with several checkboxes: 'Discard old builds', 'GitHub project', 'This build requires lockable resources', and 'This project is parameterized'. The 'This project is parameterized' checkbox is checked. To the right of each checkbox is a blue question mark icon. At the bottom of the configuration area, there is a yellow button labeled 'Add Parameter' with a dropdown arrow.



Laboratorios – Comienza la diversión

- En **“Add Parameter”** selecciona **“String Parameter”** en **Name** ponle **nombre**



String Parameter

Name ?

nombre

Default Value ?

Description ?

[Plain text] Preview

- En **“Source Code Management”** deja **None**
- En **“Build Environment”** selecciona **“Color ANSI Console Output”**, dejando **“xterm”**
- En **“Build”** selecciona **“Execute shell”** escribe:
 - `echo "Imprimiendo el valor de la variable \"nombre\":\" ${nombre}"`
- Ejecútalo (**“Build with Parameters”**)



- En **Jenkins** vamos a crear un **octavo job**
 - Nombre: “**job-08**”
 - Tipo “**Freestyle project**”
 - Copy from “**job-07**”
 - En “**This project is parameterized**”
 - “String Parameter” -> Name: “palabra”
 - “String Parameter” -> Name: “veces”
 - En “**Source Code Management**” seleccionar **Git** con el siguiente URL:
 - https://github.com/HugoAquinoNavarrete/bash_scripting
 - En “**Branch Specifier**” dejar vacío
 - En “**Build**”
 - `bash bin/07-imprime_palabras_n_veces.sh ${palabra} ${veces}`
 - Ejecútalo



- En **Jenkins** vamos a crear un **noveno job**
 - Nombre: “**job-09**”
 - Tipo “**Freestyle project**”
 - Copy from “**job-08**”
 - En “**This project is parameterized**”
 - “String Parameter” -> Name: “year”
 - En “**Source Code Management**” deberá aparecer **Git** con el siguiente URL:
 - https://github.com/HugoAquinoNavarrete/bash_scripting
 - En “**Branch Specifier**” dejar vacío
 - En “**Build**”
 - `cd bin`
 - `bash 10-mlb_equipos.sh ${year}`
 - Ejecútalo



Laboratorios – Comienza la diversión

- En **GitHub** hagan un **fork** del siguiente repositorio
 - https://github.com/HugoAquinoNavarrete/terraform_jenkins_aws
- En **Jenkins** vamos a crear un **décimo job**
 - Nombre: “**job-10**”
 - Tipo “**Freestyle project**”
 - Copy from “**job-09**”
 - En “**This project is parameterized**”
 - “String Parameter” -> Name: “AWS_ACCESS_KEY_ID”
 - “String Parameter” -> Name: “AWS_SECRET_ACCESS_KEY”
 - “String Parameter” -> Name: “nombre_instancia”
 - “String Parameter” -> Name: “cantidad_vm_ubuntu”
 - “String Parameter” -> Name: “cantidad_vm_windows”
 - “String Parameter” -> Name: “subred_id”
 - “String Parameter” -> Name: “sg_id”
 - “String Parameter” -> Name: “nombre_llave”
 - “String Parameter” -> Name: “usuario_github”
 - “Choice Parameter” -> Name: “accion” -> Choices: “crea” “destruye”



Laboratorios – Comienza la diversión

- En **GitHub** hagan un **fork** en su laptop del siguiente repositorio:
 - https://github.com/HugoAquinoNavarrete/terraform_jenkins_aws
- Edita el archivo “**main.tf**” y ajusta las siguientes líneas
 - **Línea 55:** aparece default = "<nombre_llave>", cambia <nombre_llave> con el nombre (sin espacios) que deseas tenga la llave
- Haz un “**commit**” al repositorio
 - git add main.tf
 - git commit -m "Ajustes en main.tf para reflejar cambios en el nombre de la llave privada"
 - git push



Laboratorios – Comienza la diversión

- En **Jenkins** vamos a crear un **décimo job**
 - En “**Source Code Management**” deberá aparecer **Git** con el siguiente URL:
 - [https://github.com/\\${usuario_github}/terraform_jenkins_aws](https://github.com/${usuario_github}/terraform_jenkins_aws)
 - En “**Branch Specifier**” dejar vacío, continua en el siguiente slide



- En **Jenkins** vamos a crear un **décimo job**
 - En “**Build**”
 - `#!/bin/bash`
 - `terraform init`
 - `export AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}`
 - `export AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}`
 - `if [["${accion}" == "crea"]]; then`
 - `terraform apply -var nombre_instancia=${nombre_instancia} -var cantidad_instancias_ubuntu=${cantidad_vm_ubuntu} -var cantidad_instancias_windows=${cantidad_vm_windows} -var subred_id=${subred_id} -var sg_id=${sg_id} -var key_name=${nombre_llave} -auto-approve`
 - `else`
 - `terraform destroy -var subred_id=${subred_id} -var sg_id=${sg_id} -auto-approve`
 - `fi`



Laboratorios – Comienza la diversión

- En **Jenkins** vamos a crear un **décimo job**
 - Ejecútalo con la opción “**crea**” habilitando **1 servidor Ubuntu** y **1 servidor Windows**. El resto de la información aparece en el **archivo de Excel**.

Project job-10

This build requires parameters:

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY

nombre_instancia

cantidad_vm_ubuntu

cantidad_vm_windows

subred_id

sg_id

nombre_llave

usuario_github

accion

crea ▼

Build



Laboratorios – Comienza la diversión

- En **Jenkins** vamos a crear un **décimo job**
 - Ahora vuelve a ejecutar el **job-10** solamente facilitando la información de **“AWS_ACCESS_KEY_ID”**, **“AWS_SECRET_ACCESS_KEY”**, **“subred_id”**, **“sd_id”** y **“usuario_github”** seleccionando la opción **“destruye”**

Project job-10

This build requires parameters:

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY

nombre_instancia

cantidad_vm_ubuntu

cantidad_vm_windows

subred_id

sg_id

nombre_llave

usuario_github

accion

destruye ▼



Laboratorios – Comienza la diversión

- Ahora vamos a aprender a usar otra manera de ejecutar un job desde tu laptop usando **curl** y un **token**.
- Da clic en “**estudiante**” y posteriormente en “**Configure**”

The screenshot shows the Jenkins web interface. At the top, there's a black header with the Jenkins logo on the left, a search bar in the center, and notification icons on the right. The user 'estudiante' is logged in, as indicated by the profile icon and name in the top right corner. Below the header, a breadcrumb trail shows 'Dashboard' and 'estudiante'. The left sidebar contains a menu with options: 'People', 'Status', 'Builds', 'Configure' (highlighted in yellow), 'My Views', and 'Credentials'. The main content area displays the title 'estudiante' and the text 'Jenkins User ID: copa_lab'.



Laboratorios – Comienza la diversión

- En “**API Token**” da clic en “**Add new Token**”

API Token

Current token(s)

Add new Token

- Ponle como nombre “**token_jobs**” y da clic en “**Generate**”

token_jobs

Generate

Cancel

- Copia el valor en un lugar seguro y has clic en “**Save**”

token_jobs

⚠ Copy this token now, because it cannot be recovered in the future.





Laboratorios – Comienza la diversión

- Ahora vamos a usarlo ejecutando el “**job-01**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-01/build --user copa_lab:<token_recien_creado>`
- Para el “**job-02**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-02/build --user copa_lab:<token_recien_creado>`
- Para el “**job-03**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-03/build --user copa_lab:<token_recien_creado>`
- Para el “**job-04**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-04/build --user copa_lab:<token_recien_creado>`



Laboratorios – Comienza la diversión

- Para el “**job-05**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-05/build --user copa_lab:<token_recien_creado>`
- Para el “**job-06**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-06/build --user copa_lab:<token_recien_creado>`
- Para el “**job-07**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-07/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre", "value":"mensaje_prueba"}]}'`



Laboratorios – Comienza la diversión

- Para el “**job-08**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-08/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"palabra", "value":"mensaje_prueba"}, {"name":"veces", "value":"10"}]}'`
- Para el “**job-09**”. Desde tu laptop ejecuta esto:
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-09/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"year", "value":"1921"}]}'`



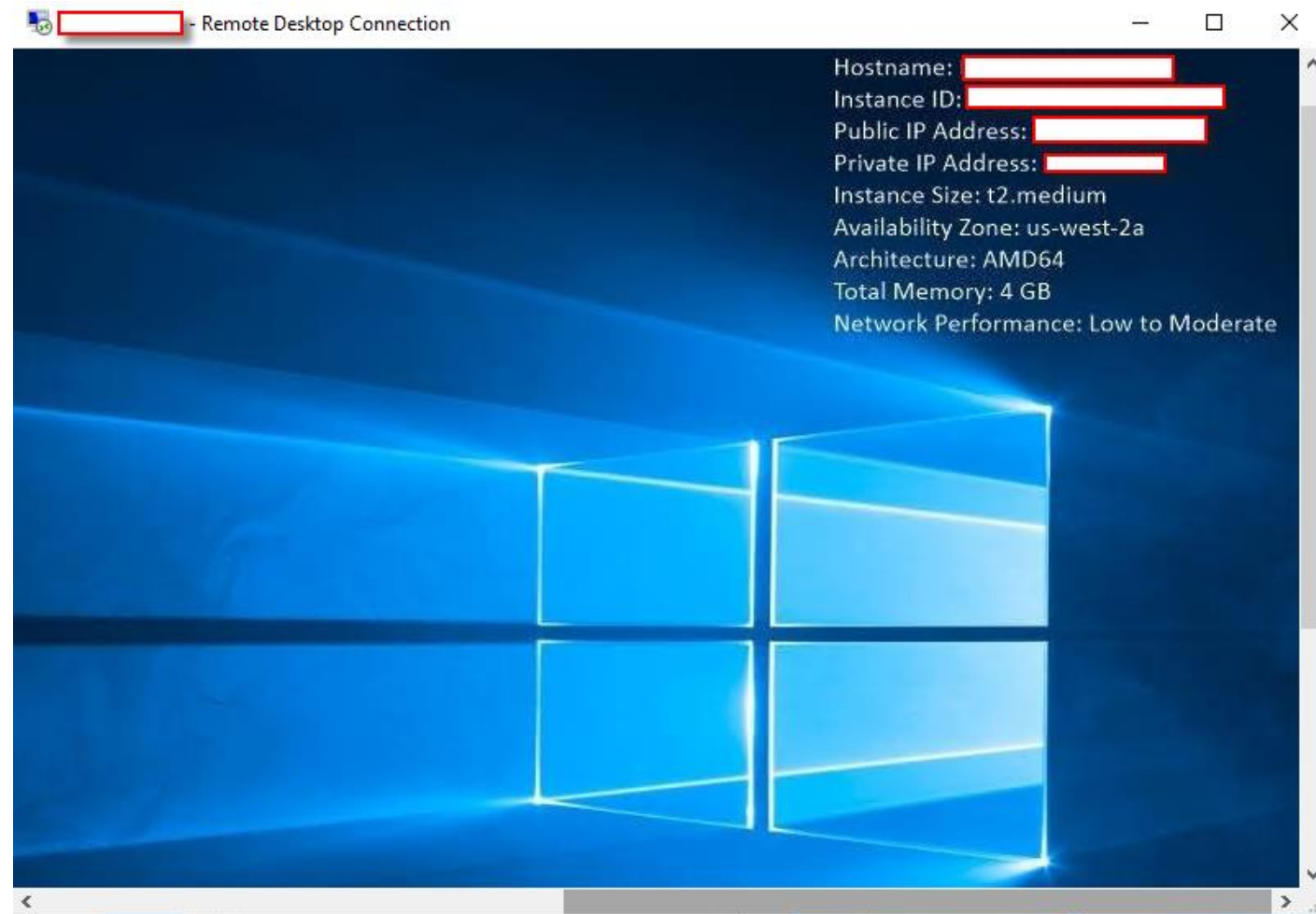
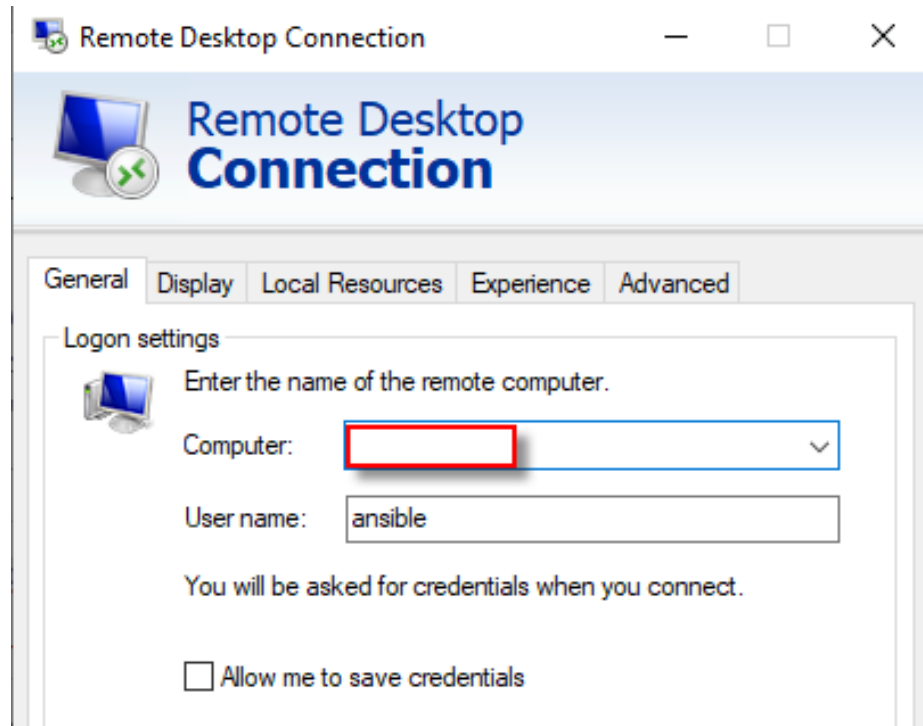
Laboratorios – Comienza la diversión

- Para el “**job-10**”. Desde tu laptop ejecuta esto:
 - Para **crear** las VMs
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-10/build --user copa_lab:<valor_dado_en_clase> --data-urlencode json='{"parameter": [{"name": "AWS_ACCESS_KEY_ID", "value": "<valor_dado_en_clase>"}, {"name": "AWS_SECRET_ACCESS_KEY", "value": "<valor_dado_en_clase>"}, {"name": "nombre_instancia", "value": "lab"}, {"name": "cantidad_vm_ubuntu", "value": "2"}, {"name": "cantidad_vm_windows", "value": "2"}, {"name": "subred_id", "value": "<valor_dado_en_clase>"}, {"name": "sg_id", "value": "<valor_dado_en_clase>"}, {"name": "nombre_llave", "value": "<nombre_llave>"}, {"name": "usuario_github", "value": "<usuario_github>"}, {"name": "accion", "value": "crea"}]}'`



Laboratorios – Comienza la diversión

- Para el “**job-10**”. Desde tu laptop ejecuta esto:
 - Intenta **conectarte** a una de las **VM Windows** usando **RDP** (**IP pública** que aparece en la ejecución del job, usuario “**ansible**” y contraseña el campo “**password_data**”)





Laboratorios – Comienza la diversión

- Para el “**job-10**”. Desde tu laptop ejecuta esto:
 - Para **destruir** las VMs
 - `curl -X POST http://<ip_publica_vm>:8080/job/job-10/build --user copa_lab:<valor_dado_en_clase> --data-urlencode json='{"parameter": [{"name": "AWS_ACCESS_KEY_ID", "value": "<valor_dado_en_clase>"}, {"name": "AWS_SECRET_ACCESS_KEY", "value": "<valor_dado_en_clase>"}, {"name": "subred_id", "value": "<valor_dado_en_clase>"}, {"name": "sg_id", "value": "<valor_dado_en_clase>"}, {"name": "accion", "value": "destruye"}]}'`



Laboratorios – Comienza la diversión

- Ahora vamos a crear el **primer pipeline**.
 - Nombre: “**pipeline-01**”
 - Tipo “**Pipeline**”
 - En “**Description**” pon “**Pipeline como código**”
 - En “**Pipeline**” escoge “**Pipeline script**” da clic en “**try simple Pipeline**” y selecciona “**Hello World**”

Pipeline

Definition

Pipeline script

Script

1

try sample Pipeline...
try sample Pipeline...
Hello World
GitHub + Maven
Scripted Pipeline

Script

```
1 pipeline {  
2   agent any  
3  
4   stages {  
5     stage('Hello') {  
6       steps {  
7         echo 'Hello World'  
8       }  
9     }  
10  }  
11 }  
12
```

- Haz clic en “**Save**” y ejecútalo desde tu laptop de la siguiente manera:

• curl -X POST http://<ip_publica_vm>:8080/job/pipeline-01/build --user copa_lab:<token_recien_creado>



- Ahora vamos a crear el **segundo pipeline**.
 - Nombre: “**pipeline-02**”
 - Copy from “**pipeline-01**”
 - Tipo “**Pipeline**”
 - En “**Description**” pon “**Pipeline como código**”
 - En “**Pipeline**” hay que editar las líneas de código que aparecen con las que aparecen en el siguiente slide
 - Haz clic en **Save** y ejecútalo desde tu laptop de la siguiente manera:
 - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-02/build --user copa_lab:<token_recien_creado>`



Laboratorios – Comienza la diversión

```
pipeline {
  agent any
  stages {
    stage('# BUILD_ID') {
      steps {
        sh 'echo $BUILD_ID'
      }
    }
    stage('WORKSPACE') {
      steps {
        sh 'echo $WORKSPACE'
      }
    }
    stage('Contenido inicial del WORKSPACE') {
      steps {
        sh 'ls -lR $WORKSPACE'
      }
    }
    stage('Crea archivo file.txt') {
      steps {
        sh 'echo "Que ondas aprendiendo Jenkins !!!" > $WORKSPACE/file.txt'
      }
    }
    stage('Contenido final del WORKSPACE') {
      steps {
        sh 'ls -lR $WORKSPACE'
      }
    }
    stage('Contenido archivo file.txt') {
      steps {
        sh 'cat $WORKSPACE/file.txt'
      }
    }
  }
}
```



- Ahora vamos a crear el **tercer pipeline**.
 - Nombre: “**pipeline-03**”
 - Copy from “**pipeline-02**”
 - Tipo “**Pipeline**”
 - En “**Description**” pon “**Pipeline como código**”
 - En “**Pipeline**” hay que editar las líneas de código que aparecen con las que aparecen en el siguiente slide
 - Haz clic en **Save** y ejecútalo desde tu laptop de la siguiente manera:
 - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-03/build --user copa_lab:<token_recien_creado>`



Laboratorios – Comienza la diversión

```
pipeline {
  agent any

  stages {
    stage('Clona repositorio') {
      steps {
        git branch: 'main', url: 'https://github.com/HugoAquinoNavarrete/bash_scripting'
      }
    }

    stage('Imprime directorios') {
      steps {
        sh 'ls -lR'
      }
    }

    stage('Imprime variables') {
      steps {
        sh '/bin/bash ./bin/01-imprime_variables.sh'
      }
    }

    stage('Imprime variables de ambiente') {
      steps {
        sh '/bin/bash ./bin/06-mi_entorno.sh'
      }
    }
  }
}
```



- Ahora vamos a crear el **cuarto pipeline**.
 - Nombre: “**pipeline-04**”
 - Copy from “**pipeline-03**”
 - Tipo “**Pipeline**”
 - En “**Description**” pon “**Pipeline como código**”
 - En “**This project is parameterized**”
 - “String Parameter” -> Name: “palabra”
 - “String Parameter” -> Name: “veces”
 - En “**Pipeline**” hay que editar las líneas de código que aparecen con las que aparecen en el siguiente slide
 - Haz clic en **Save** y ejecútalo desde tu laptop de la siguiente manera:
 - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-04/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"palabra", "value":"ejecutando_pipeline_desde_curl"}, {"name":"veces", "value":"25"}]}'`



Laboratorios – Comienza la diversión

```
pipeline {  
  agent any  
  
  stages {  
    stage('Clona repositorio') {  
      steps {  
        git branch: 'main', url: 'https://github.com/HugoAquinoNavarrete/bash_scripting'  
      }  
    }  
  
    stage('Imprime un mensaje n veces') {  
      steps {  
        sh '/bin/bash ./bin/07-imprime_palabras_n_veces.sh ${palabra} ${veces}'  
      }  
    }  
  }  
}
```




- Ahora vamos a crear el **quinto pipeline**.
 - Nombre: “**pipeline-05**”
 - Copy from “**pipeline-04**”
 - Tipo “**Pipeline**”
 - En “**Description**” pon “**Pipeline como código**”
 - En “**This project is parameterized**”
 - “String Parameter” -> Name: “AWS_ACCESS_KEY_ID”
 - “String Parameter” -> Name: “AWS_SECRET_ACCESS_KEY”
 - “String Parameter” -> Name: “nombre_instancia”
 - “String Parameter” -> Name: “cantidad_vm_ubuntu”
 - “String Parameter” -> Name: “cantidad_vm_windows”
 - “String Parameter” -> Name: “subred_id”
 - “String Parameter” -> Name: “sg_id”
 - “String Parameter” -> Name: “nombre_llave”
 - “String Parameter” -> Name: “usuario_github”
 - “Choice Parameter” -> Name: “accion” -> Choices: “crea” “destruye”
 - En “**Pipeline**” hay que editar las líneas de código que aparecen con las que aparecen en el siguiente slide, después haz clic en **Save**



Laboratorios – Comienza la diversión

```
pipeline {
  agent any
  options {
    ansiColor('xterm')
  }
  stages {
    stage('Clona repositorio') {
      steps {
        git branch: 'main', url: 'https://github.com/${usuario_github}/terraform_jenkins_aws'
      }
    }
    stage('Exporta variables de entorno') {
      steps {
        sh 'export AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}'
        sh 'export AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}'
      }
    }
    stage('Inicializa terraform') {
      steps {
        sh 'terraform init'
      }
    }
    stage('Crea infraestructura') {
      when {
        equals expected: "crea", actual: "${accion}"
      }
      steps {
        sh 'terraform apply -var nombre_instancia=${nombre_instancia} -var cantidad_instancias_ubuntu=${cantidad_vm_ubuntu} -var cantidad_instancias_windows=${cantidad_vm_windows} -var subred_id=${subred_id} -var sg_id=${sg_id} -var key_name=${nombre_llave} -var usuario_github=${usuario_github} -auto-approve'
        sh 'echo "Inventario VM Ubuntu"'
        sh 'cat ansible_inventario.txt'
        sh 'echo "Inventario VM Windows"'
        sh 'cat ansible_inventario_win.txt'
      }
    }
    stage('Destruye infraestructura') {
      when {
        equals expected: "destruye", actual: "${accion}"
      }
      steps {
        sh 'terraform destroy -var subred_id=${subred_id} -var sg_id=${sg_id} -auto-approve'
      }
    }
  }
}
```



- Ahora vamos a crear el **quinto pipeline**.
 - Crea **1 VM Ubuntu** y **1 VMs Windows**
 - Para **crear** las VMs
 - ```
curl -X POST http://<ip_publica_vm>:8080/job/pipeline-05/build --user
copa_lab:<valor_dado_en_clase> --data-urlencode json='{"parameter":
[{"name":"AWS_ACCESS_KEY_ID",
"value":"<valor_dado_en_clase>"}, {"name":"AWS_SECRET_ACCESS_KEY",
"value":"<valor_dado_en_clase>"}, {"name":"nombre_instancia",
"value":"lab"}, {"name":"cantidad_vm_ubuntu", "value":"2"}, {"name":"cantidad_vm_windows",
"value":"2"}, {"name":"subred_id", "value":"<valor_dado_en_clase>"}, {"name":"sg_id",
"value":"<valor_dado_en_clase>"}, {"name":"nombre_llave", "value":"<nombre_llave>"},
{"name":"usuario_github", "value":"<usuario_github>"}, {"name":"accion", "value":"crea"}]}'
```



- Ahora vamos a crear el **quinto pipeline**.
  - Para **destruir** las VMs
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-05/build --user copa_lab:<valor_dado_en_clase> --data-urlencode json='{"parameter": [{"name": "AWS_ACCESS_KEY_ID", "value": "<valor_dado_en_clase>"}, {"name": "AWS_SECRET_ACCESS_KEY", "value": "<valor_dado_en_clase>"}, {"name": "usuario_github", "value": "<usuario_github>"}, {"name": "accion", "value": "destruye"}]}'`



- Ahora vamos a crear el **sexto pipeline**.
  - Nombre: “**pipeline-06**”
  - Copy from “**pipeline-05**”
  - Tipo “**Pipeline**”
  - En “**Description**” pon “**Pipeline como código**”
  - En “**This project is parameterized**”
    - “String Parameter” -> Name: “nombre\_llave”
    - “Choice Parameter” -> Name: “accion” -> Choices: “lista\_archivos” “haz\_ping” “imprime\_mensaje\_ubuntu” “imprime\_mensaje\_windows”
  - En “**Pipeline**” hay que editar las líneas de código que aparecen con las que aparecen en el siguiente slide, después haz clic en **Save**



# Laboratorios – Comienza la diversión

```
pipeline {
 agent any
 options {
 ansiColor('xterm')
 }
 stages {
 stage('Clona repositorio') {
 steps {
 git branch: 'main', url: 'https://github.com/HugoAquinoNavarrete/ansible_scripting'
 }
 }
 stage('Copia llave e inventarios del pipeline-05') {
 steps {
 sh 'cp ../pipeline-05/${nombre_llave} .'
 sh 'cp ../pipeline-05/ansible_inventario.txt .'
 sh 'cp ../pipeline-05/ansible_inventario_win.txt .'
 }
 }
 stage('Lista archivos del repositorio clonado') {
 when {
 equals expected: "lista_archivos", actual: "${accion}"
 }
 steps {
 sh 'ls -l'
 sh 'cat ansible_inventario.txt'
 sh 'cat ansible_inventario_win.txt'
 }
 }
 stage('Haz ping con Ansible a las VMs creadas') {
 when {
 equals expected: "haz_ping", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/01-ping.sh ${nombre_llave}'
 sh '/bin/bash ./bin/01-ping_win.sh ${nombre_llave}'
 }
 }
 stage('Imprime mensaje con Ansible a las VMs Ubuntu creadas') {
 when {
 equals expected: "imprime_mensaje_ubuntu", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/02-mensaje.sh ${nombre_llave}'
 }
 }
 stage('Imprime mensaje con Ansible a las VMs Windows creadas') {
 when {
 equals expected: "imprime_mensaje_windows", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/02-mensaje_win.sh ${nombre_llave}'
 }
 }
 }
}
```



- Ahora vamos a crear el **sexto pipeline**.
  - Ejecútalo una vez seleccionando la **acción “lista\_archivos”**
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-06/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [ {"name":"accion", "value":"lista_archivos"}, {"name":"nombre_llave", "value":"<nombre_llave>"}]}'`
  - Para poder ejecutar las acciones **“haz\_ping”** e **“imprime\_mensajes”** hay que pasar la llave **“key”** y el inventario **“ansible\_inventario.txt”** creados en el **“pipeline-05”** al workspace del **“pipeline-06”**, para hacerlo vamos a conectarnos desde tu laptop usando la llave que se facilitó al inicio del curso a la VM en la cual está corriendo **Jenkins**:
    - `ssh -v -l ubuntu -i <llave> <ip_pública_vm>`



- Ahora vamos a crear el **sexto pipeline**.
  - Para **VM Ubuntu**, ejecuta el script “**./bin/01-ping.sh**” para habilitar el “**fingerprint**”,
    - `./bin/01-ping.sh <nombre_llave>`
  - A la pregunta “**Are you sure you want to continue connecting (yes/no)?**”, contesta “**yes**”, cancela la ejecución del script “**Control-C**” y vuélvelo a ejecutar tantas veces como VMs hayas creado o teclea “**Enter**” para ir avanzando en la ejecución e ir contestando “**yes**”
  - Para **VM Windows**, ejecuta el script “**./bin/01-ping\_win.sh**” para habilitar el “**fingerprint**”
    - `./bin/01-ping_win.sh <nombre_llave>`
  - Instala el siguiente componente de **ansible-galaxy** para los siguientes laboratorios asociados a windows
    - `ansible-galaxy collection install ansible.windows`





- Ahora vamos a crear el **sexto pipeline**.
  - La ejecución final debería verse así:

```
jenkins@[REDACTED]:~/workspace/pipeline-06$./bin/01-ping.sh
Using /etc/ansible/ansible.cfg as config file
/var/jenkins_home/workspace/pipeline-06/ansible_inventario.txt did not meet host_list requirements, check plugin documentation if this is unexpected
/var/jenkins_home/workspace/pipeline-06/ansible_inventario.txt did not meet script requirements, check plugin documentation if this is unexpected
/var/jenkins_home/workspace/pipeline-06/ansible_inventario.txt did not meet yaml requirements, check plugin documentation if this is unexpected

PLAY [servers] *****

TASK [ping] *****
ok: [REDACTED] => {"changed": false, "ping": "pong"}
ok: [REDACTED] => {"changed": false, "ping": "pong"}

PLAY RECAP *****
[REDACTED] : ok=1 changed=0 unreachable=0 failed=0
[REDACTED] : ok=1 changed=0 unreachable=0 failed=0

jenkins@d7201c6c9a88:~/workspace/pipeline-06$./bin/01-ping_win.sh
Using /etc/ansible/ansible.cfg as config file
/var/jenkins_home/workspace/pipeline-06/ansible_inventario_win.txt did not meet host_list requirements, check plugin documentation if this is unexpected
/var/jenkins_home/workspace/pipeline-06/ansible_inventario_win.txt did not meet script requirements, check plugin documentation if this is unexpected
/var/jenkins_home/workspace/pipeline-06/ansible_inventario_win.txt did not meet yaml requirements, check plugin documentation if this is unexpected

PLAY [servers_win] *****

TASK [win_ping] *****
ok: [REDACTED] => {"changed": false, "ping": "pong"}
ok: [REDACTED] => {"changed": false, "ping": "pong"}

PLAY RECAP *****
[REDACTED] : ok=1 changed=0 unreachable=0 failed=0
[REDACTED] : ok=1 changed=0 unreachable=0 failed=0
```



- Ahora vamos a crear el **sexto pipeline**.
  - Ejecútalo una vez seleccionando la **acción “haz\_ping”**
  - `curl -X POST http://<ip_pública_vm>:8080/job/pipeline-06/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [ {"name":"accion", "value":"haz_ping"}, {"name":"nombre_llave", "value":"<nombre_llave>"}]}'`
  - Ejecútalo una vez seleccionando la **acción “imprime\_mensaje\_ubuntu”**
  - `curl -X POST http://<ip_pública_vm>:8080/job/pipeline-06/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [ {"name":"accion", "value":"imprime_mensaje_ubuntu"}, {"name":"nombre_llave", "value":"<nombre_llave>"}]}'`
  - Ejecútalo una vez seleccionando la **acción “imprime\_mensaje\_windows”**
  - `curl -X POST http://<ip_pública_vm>:8080/job/pipeline-06/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [ {"name":"accion", "value":"imprime_mensaje_windows"}, {"name":"nombre_llave", "value":"<nombre_llave>"}]}'`



- Ahora vamos a crear el **séptimo pipeline**.
  - Nombre: “**pipeline-07**”
  - Copy from “**pipeline-06**”
  - Tipo “**Pipeline**”
  - En “**Description**” pon “**Pipeline como código**”
  - En “**This project is parameterized**”
    - “String Parameter” -> Name: “nombre\_llave”
    - “Choice Parameter” -> Name: “accion” -> Choices: “crea\_directorios” “borra\_directorios” “clona\_repositorio” “copia\_archivos” “borra\_archivos”
  - En “**Pipeline**” hay que editar las líneas de código que aparecen con las que aparecen en el siguiente slide, después haz clic en **Save**



```
pipeline {
 agent any
 options {
 ansiColor('xterm')
 }
 stages {
 stage('Clona repositorio') {
 steps {
 git branch: 'main', url: 'https://github.com/HugoAquinoNavarrete/ansible_scripting'
 }
 }
 stage('Copia llave e inventario del pipeline-05') {
 steps {
 sh 'cp ../pipeline-05/${nombre_llave} .'
 sh 'cp ../pipeline-05/ansible_inventario.txt .'
 sh 'cp ../pipeline-05/ansible_inventario_win.txt .'
 }
 }
 stage('Crea directorios') {
 when {
 equals expected: "crea_directorios", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/08-directorios_crea.sh ${nombre_llave}'
 sh '/bin/bash ./bin/08-directorios_crea_win.sh ${nombre_llave}'
 }
 }
 stage('Borra directorios') {
 when {
 equals expected: "borra_directorios", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/09-directorios_borra.sh ${nombre_llave}'
 sh '/bin/bash ./bin/09-directorios_borra_win.sh ${nombre_llave}'
 }
 }
 stage('Clona repositorio git') {
 when {
 equals expected: "clona_repositorio", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/10-git_clona.sh ${nombre_llave}'
 }
 }
 stage('Copia archivos') {
 when {
 equals expected: "copia_archivos", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/13-archivos_copia.sh ${nombre_llave}'
 sh '/bin/bash ./bin/13-archivos_copia_win.sh ${nombre_llave}'
 }
 }
 stage('Borra archivos') {
 when {
 equals expected: "borra_archivos", actual: "${accion}"
 }
 steps {
 sh '/bin/bash ./bin/22-archivos_borra.sh ${nombre_llave}'
 sh '/bin/bash ./bin/22-archivos_borra_win.sh ${nombre_llave}'
 }
 }
 }
}
```

# Laboratorios – Comienza la diversión



- Ahora vamos a crear el **séptimo pipeline**.
  - Ejecútalo una vez seleccionando la **acción “crea\_directorios”**
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-07/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"accion", "value":"crea_directorios"}]}'`
  - Ejecútalo una vez seleccionando la **acción “borra\_directorios”**
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-07/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"accion", "value":"borra_directorios"}]}'`
  - Ejecútalo una vez seleccionando la **acción “clona\_repositorio”**
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-07/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"accion", "value":"clona_repositorio"}]}'`



- Ahora vamos a crear el **séptimo pipeline**.
  - Ejecútalo una vez seleccionando la **acción “copia\_archivos”**
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-07/build --user copa_lab:<token_recien_creado> --data-urlencode json='{\"parameter\": [{\"name\": \"nombre_llave\", \"value\": \"<nombre_llave>\"}, {\"name\": \"accion\", \"value\": \"copia_archivos\"}]}'`
  - Ejecútalo una vez seleccionando la **acción “borra\_archivos”**
  - `curl -X POST http://<ip_publica_vm>:8080/job/pipeline-07/build --user copa_lab:<token_recien_creado> --data-urlencode json='{\"parameter\": [{\"name\": \"nombre_llave\", \"value\": \"<nombre_llave>\"}, {\"name\": \"accion\", \"value\": \"borra_archivos\"}]}'`



## Laboratorios – Comienza la diversión

- Ahora vamos a crear el **primer jenkinsfile**.
  - En tu cuenta de GitHub haz un **fork** del siguiente repositorio:
    - [https://github.com/HugoAquinoNavarrete/jenkins\\_scripting](https://github.com/HugoAquinoNavarrete/jenkins_scripting)
  - Habilita un **webhook** para el repositorio al cual le hiciste **fork** (slides 36 y 37)
  - Nombre: “**jenkinsfile-01**”
  - Tipo “**Pipeline**”
  - En “**Description**” pon “**Jenkinsfile como código**”
  - En “**Build Triggers**” selecciona “**GitHub hook trigger for GITScm polling**”



## Laboratorios – Comienza la diversión

- Ahora vamos a crear el **primer jenkinsfile**.
  - En “**Pipeline**” selecciona “**Pipeline script from SCM**”, en “**SCM**” escoge “**Git**” y en “**Repository URL**” pon el URL del sitio al cual le hiciste fork:

**Pipeline**

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

[https://github.com/HugoAquinoNavarrete/jenkins\\_scripting](https://github.com/HugoAquinoNavarrete/jenkins_scripting)

Credentials

- none - Add





## Laboratorios – Comienza la diversión

- Ahora vamos a crear el **primer jenkinsfile**.
  - En “**Branches to build**” escribe “\*/main”

Branches to build ?

Branch Specifier (blank for 'any') X ?

\*/main

- En “**Script Path**” escribe “jenkinsfiles/Jenkinsfile-01.txt” y después da clic en “**Save**”

Script Path ?

jenkinsfiles/Jenkinsfile-01.txt

☒ Lightweight checkout ?

Pipeline Syntax

**Save** Apply



## Laboratorios – Comienza la diversión

- Ahora vamos a crear el **primer jenkinsfile**.
  - Ejecútalo una vez
  - `curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-01/build --user copa_lab:<token_recien_creado>`



## Laboratorios – Comienza la diversión

- Ahora vamos a crear el **segundo jenkinsfile**.
  - Nombre: “**jenkinsfile-02**”
  - Copy from “**jenkinsfile-01**”
  - En “**Script Path**” escribe “**jenkinsfiles/Jenkinsfile-02.txt**” y después da clic en “**Save**”



- Ahora vamos a crear el **segundo jenkinsfile**.
  - Ejecútalo una vez seleccionando las **acciones** en **directorios** “**Crear**” y en **archivos** “**Copia**”
  - ```
curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-02/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"directorios", "value":"Crear"}, {"name":"archivos", "value":"Copia"}, {"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"nombre_llave", "value":"<nombre_llave>"}]}'
```
 - Ejecútalo una vez seleccionando las **acciones** en **directorios** “**Borrar**” y en **archivos** “**Nada**”
 - ```
curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-02/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"directorios", "value":"Borrar"}, {"name":"archivos", "value":"Nada"}, {"name":"nombre_llave", "value":"<nombre_llave>"}]}'
```



- Ahora vamos a crear el **tercer jenkinsfile**.
  - Nombre: “**jenkinsfile-03**”
  - Copy from “**jenkinsfile-02**”
  - En “**Script Path**” escribe “**jenkinsfiles/Jenkinsfile-03.txt**” y después da clic en “**Save**”
  - Ejecútalo una vez seleccionando la **acción** en **nginx** “**Instalar**”
  - `curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-03/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"nginx", "value":"Instala"}]}'`
  - Intenta cargar a través de la IP pública de una de las VMs que creaste
  - `http://<ip_vm_creada_Jenkins_pipeline>`



- Ahora vamos a crear el **tercer jenkinsfile**.
  - Ejecútalo una vez seleccionando la **acción** en **nginx** “**Sube**”
  - `curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-03/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"nginx", "value":"Sube"}]}'`
  - Intenta cargar a través de la IP pública de una de las VMs que creaste
  - `http://<ip_vm_creada_Jenkins_pipeline>`
  - Ejecútalo una vez seleccionando la **acción** en **nginx** “**Baja**”
  - `curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-03/build --user copa_lab:<token_recien_creado> --data-urlencode json='{"parameter": [{"name":"nombre_llave", "value":"<nombre_llave>"}, {"name":"nginx", "value":"Baja"}]}'`
  - Intenta cargar a través de la IP pública de una de las VMs que creaste
  - `http://<ip_vm_creada_Jenkins_pipeline>`



- Ahora vamos a crear el **tercer jenkinsfile**.
  - Ejecútalo una vez seleccionando la **acción** en **nginx** “**Despliega**”
  - `curl -X POST http://<ip_publica_vm>:8080/job/jenkinsfile-03/build --user copa_lab:<token_recien_creado> --data-urlencode json='{ "parameter": [ { "name": "nombre_llave", "value": "<nombre_llave>" }, { "name": "nginx", "value": "Despliega" } ] }'`
  - Intenta cargar a través de la IP pública de una de las VMs que creaste
  - `http://<ip_vm_creada_Jenkins_pipeline>`



- Para respaldar la información, ve a “**Manage Jenkins**” -> “**Manage Plugins**” -> “**Available**” y escribe “**thinbackup**”, instálalo sin reiniciar Jenkins
- Ve a “**Manage Jenkins**”, ve al final en “**ThinBackup**” da clic:

### Uncategorized



#### ThinBackup

Backup your global and job specific configuration.

- Da clic en “**Settings**” y en “**Backup directory**” escribe “**/var/jenkins\_home/**” y da clic en “**Save**”

### Backup settings

Backup directory

/var/jenkins\_home/





- Da clic en “**Backup Now**”



- Conéctate a la VM que está corriendo Jenkins
  - `ssh -v -l ubuntu -i <llave> <ip_pública_vm>`
- Conéctate al contenedor donde está corriendo Jenkins
  - `docker exec -it jenkins /bin/bash`



- Ve al directorio “/var/jenkins\_home” y lista el contenido
  - `cd /var/jenkins_home`
  - `ls`
- Los respaldos tienen este formato “**FULL-aaaa-mm-dd-hh-mm**”
- Comprime el directorio que tiene el respaldo
  - `tar -zcvf jenkins.tar.gz <directorio>`
- Sal del contenedor
  - `exit`



- Identifica el ID del contenedor

- `docker ps`

| CONTAINER ID | IMAGE                      | COMMAND                  | CREATED     | STATUS     | PORTS                                                                                    |
|--------------|----------------------------|--------------------------|-------------|------------|------------------------------------------------------------------------------------------|
|              |                            |                          |             | NAMES      |                                                                                          |
| d7201c6c9a88 | jenkins/jenkins:2.285-slim | "/sbin/tini -- /usr/..." | 5 weeks ago | Up 12 days | 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:49154->50000/tcp, :::49154->50000/tcp |
|              |                            |                          |             | jenkins    |                                                                                          |

- Copia el archivo del contenedor a la VM
  - `docker cp <id_contenedor>:/var/jenkins_home/jenkins.tar.gz .`
- Desde tu laptop y teniendo la llave de la VM en la que corre Jenkins, copia el respaldo a un directorio de la laptop
  - `scp -i <llave> ubuntu@<ip_pública_vm>:/home/ubuntu/jenkins.tar.gz <directorio_laptop>`



## Restaurar información

- Desde tu laptop y teniendo la llave de la VM en la que corre Jenkins, copia el respaldo a la VM en la que corre Jenkins
  - `scp -i <llave> <directorio_laptop>/jenkins.tar.gz ubuntu@<ip_publica_vm>:/home/ubuntu/jenkins.tar.gz`
- Conéctate a la VM que está corriendo Jenkins
  - `ssh -v -l ubuntu -i <llave> <ip_pública_vm>`
- Copia el archivo del respaldo al contenedor
  - `docker cp /home/ubuntu/jenkins.tar.gz <id_contenedor>:/var/jenkins_home`
- Conéctate al contenedor donde está corriendo Jenkins
  - `docker exec -it jenkins /bin/bash`



- Ve al directorio “/var/jenkins\_home” y lista el contenido
  - `cd /var/jenkins_home`
  - `ls`
- El respaldo aparecerá con el nombre “**jenkins.tar.gz**”
- Descomprime el directorio que tiene el respaldo
  - `tar -xvf jenkins.tar.gz`
- Sal del contenedor
  - `exit`
- Abre Jenkins desde el navegador
  - `http://<ip_pública_vm>:8080`



## Restaurar información

- Para restaurar la información, ve a “**Manage Jenkins**” -> “**Manage Plugins**” -> “**Available**” y escribe “**thinbackup**”, instálalo sin reiniciar Jenkins
- Ve a “**Manage Jenkins**”, ve al final en “**ThinBackup**” da clic:

### Uncategorized



#### ThinBackup

Backup your global and job specific configuration.

- Da clic en “**Settings**” y en “**Backup directory**” escribe “**/var/jenkins\_home/**” y da clic en “**Save**”

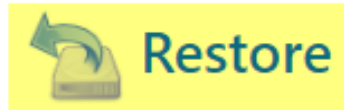
### Backup settings

Backup directory

/var/jenkins\_home/



- Da clic en “**Restore**”



- Selecciona el respaldo que deseas restaurar y da clic en “**Restore**”



## Restore options

restore backup from

2021-05-20 04:11 ▼

- ☐ Restore next build number file (if found in backup)
- ☐ Restore plugins



## Restaurar información

- En la VM baja y sube el contenedor Jenkins
  - `docker stop jenkins`
  - `docker stop jenkins`
- Abre Jenkins desde el navegador y deberá aparecerte los jobs, pipelines y demás información respaldada



- Tiempo para repasar jugando



Gracias

