**Trabajo Práctico 7 - Servidor de Build (de integración continua).**

1- Objetivos de Aprendizaje

- Adquirir conocimientos acerca de las herramientas de integración continua.

- Configurar este tipo de herramientas.

- Implementar procesos de construcción automatizado simples.

2- Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 3 (Libro Continuous Delivery: Cap 3)

3- Consignas a desarrollar en el trabajo práctico:

- Para una mejor evaluación del trabajo práctico, incluir capturas de pantalla de los pasos donde considere necesario.

**4- Desarrollo:**

1- Poniendo en funcionamiento Jenkins

- Bajar la aplicación y ejecutarla (ejemplo para Linux):

export JENKINS\_HOME=~/jenkins

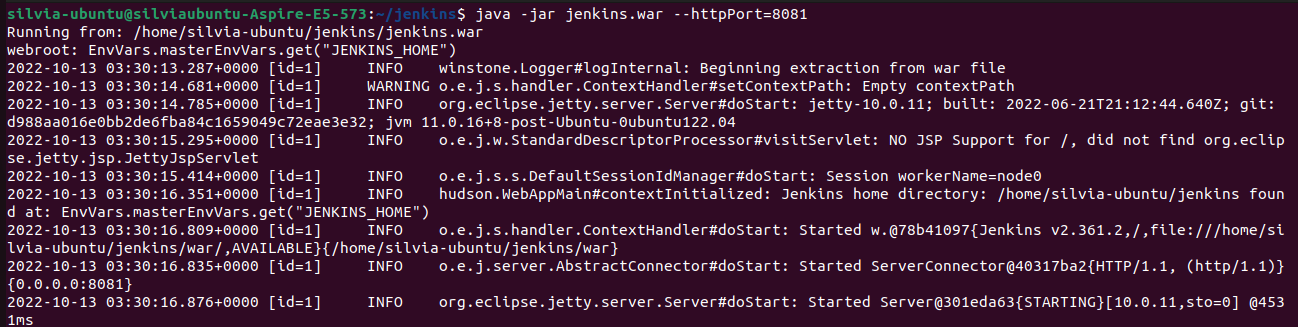
mkdir -p $JENKINS\_HOME

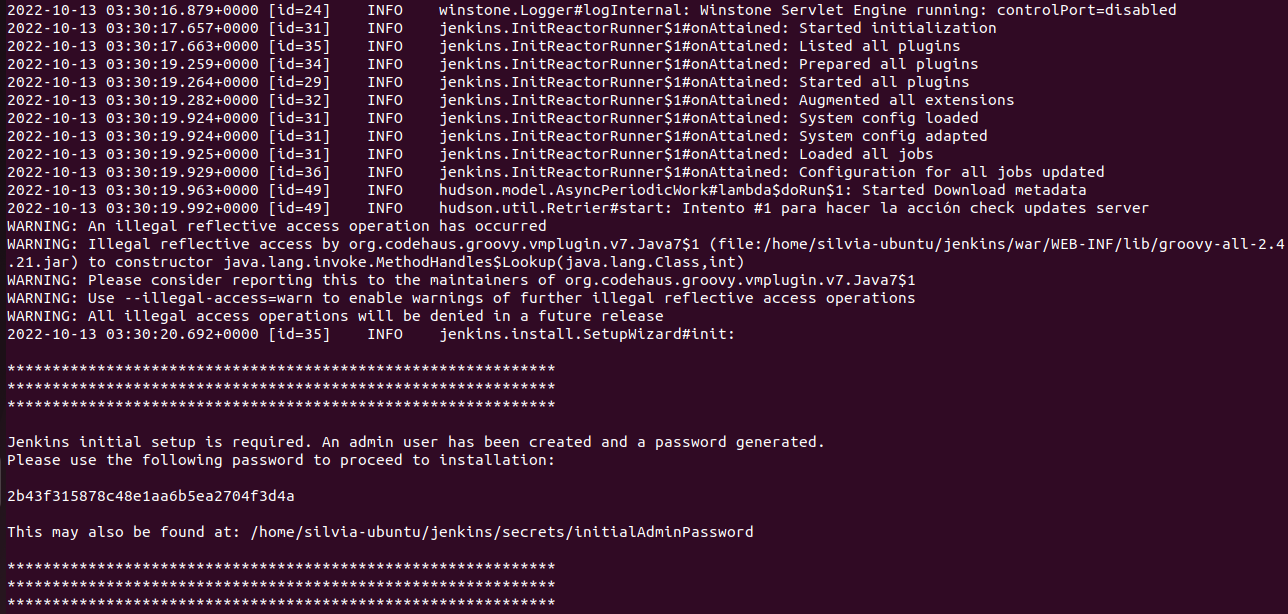
cd $JENKINS\_HOME

wget http://mirrors.jenkins.io/war-stable/latest/jenkins.war

java -jar jenkins.war --httpPort=8081







- Se puede también ejecutar en contenedor de Jenkins (pero para construir imágenes de Docker, el proceso se complica un poco):

# Windows

mkdir -p C:\jenkins

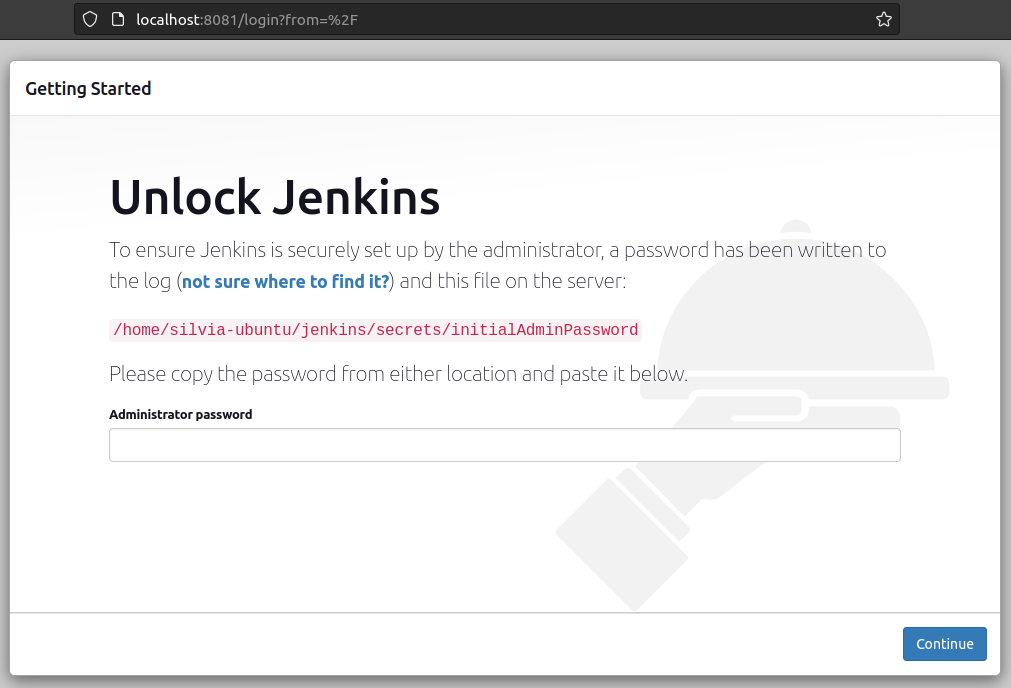
docker run -d -p 8081:8080 -p 50000:50000 -v C:\jenkins:/var/jenkins\_home jenkins/jenkins:lts

# Linux / Mac OS

mkdir -p ~/jenkins

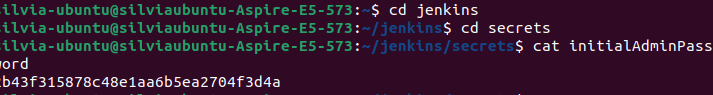
docker run -d -p 8081:8080 -p 50000:50000 -v ~/jenkins:/var/jenkins\_home jenkins/jenkins:lts

- Una vez en ejecución, abrir [http://localhost:8081](http://localhost:8081/)



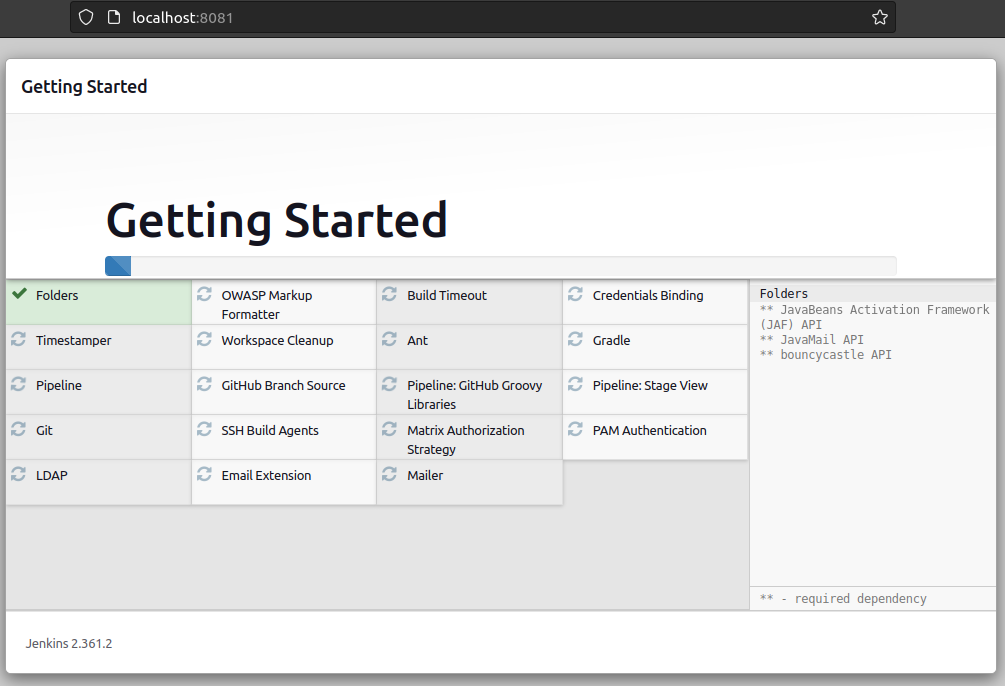
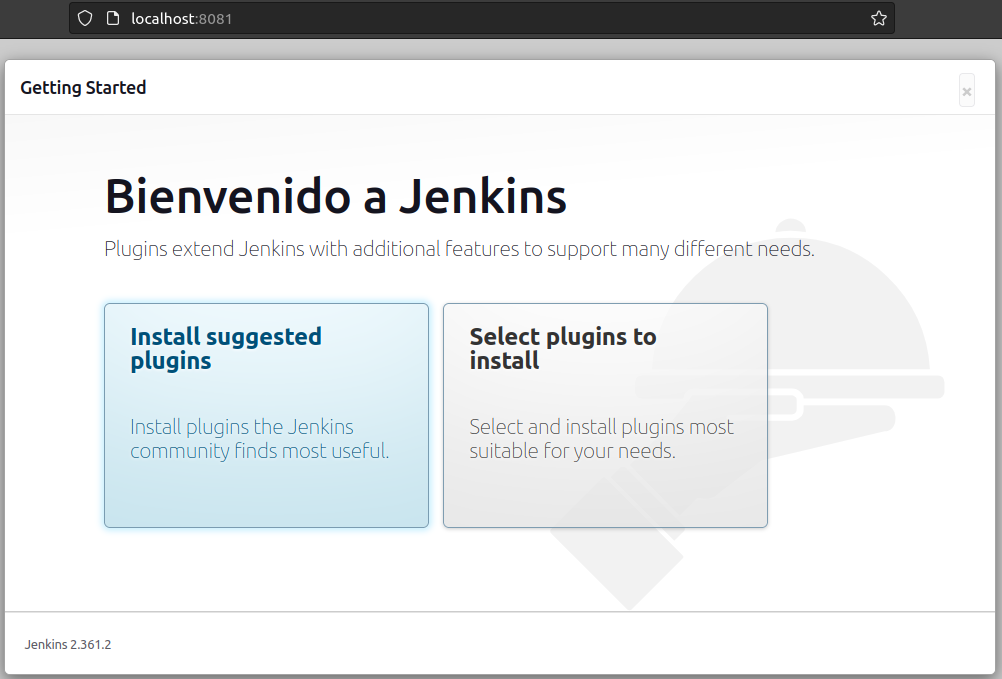
- Inicialmente deberá especificar el texto dentro del archivo ~/jenkins/secrets/initialAdminPassword

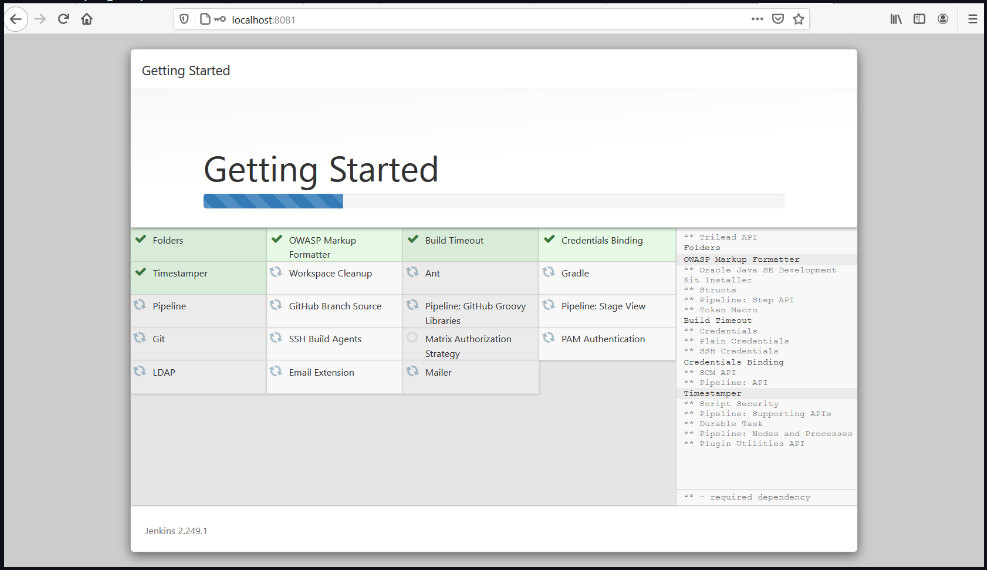
cat ~/jenkins/secrets/initialAdminPassword



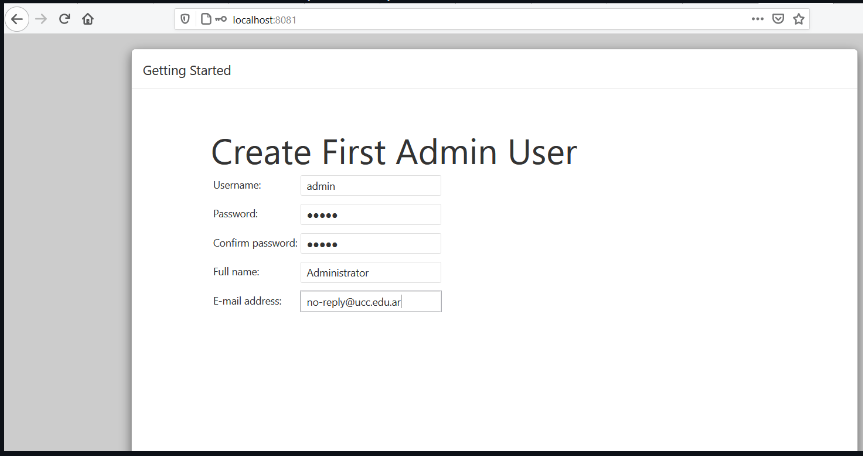
- Instalar los plugins por defecto

Selecciono la opción Suggested



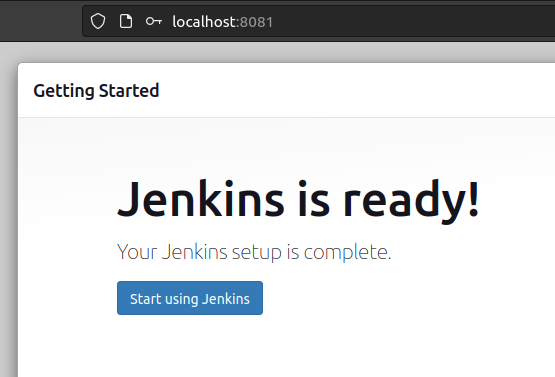


- Crear el usuario admin inicial. Colocar cualquier valor que considere adecuado.

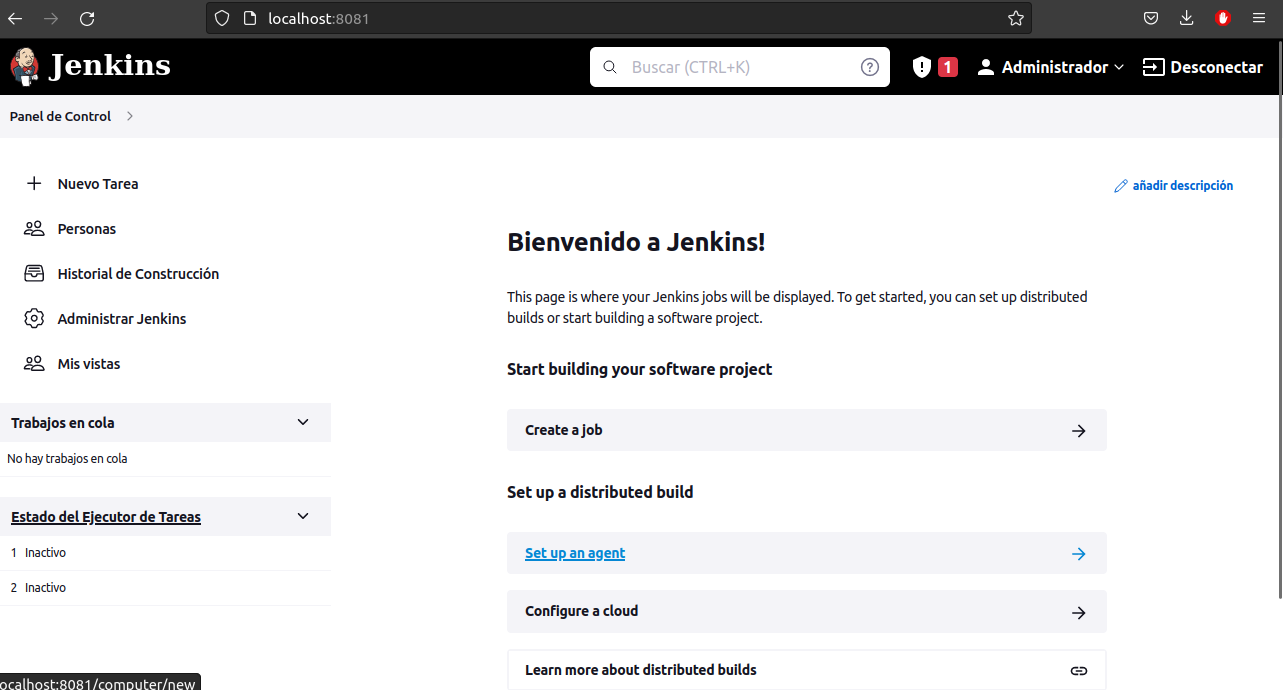


username: silviac

password: password



- Se aconseja persistir la variable \*\*JENKINS\_HOME\*\*, ya sea por ejemplo en .bashrc o en las variables de entorno de Windows.



2- Conceptos generales

- Junto al Jefe de trabajos prácticos:

- Explicamos los diferentes componentes que vemos en la página principal

- Analizamos las opciones de administración de Jenkins

**Jenkins**: servidor open source para IC. Tool que se usa para compilar y probar proyectos de sw de forma continua, lo que facilita a los devs integrar cambios en un proyecto y entregar nuevas versiones a los users. Escrito en java, es multiplataforma y accesible mediante interfaz web. Es el sw mas utilizado en la actualidad para este proósito. Con Jenkins, las organizaciones aceleran el proceso de desarrollo y entrega de sw a través de la automatización. Mediante sus centenares de pluggins, se puede implementar en diferentes etapas del ciclo de vida del desarrollo, como la compilación, la documentación, el testeo o el despliegue.

Con Jenkins, podemos automatizar multitud de tareas que nos ayudarán a reducir el time to market de nuestros productos digitales o de nuevas versiones de ellos, concretamente, con esta tool podemos:

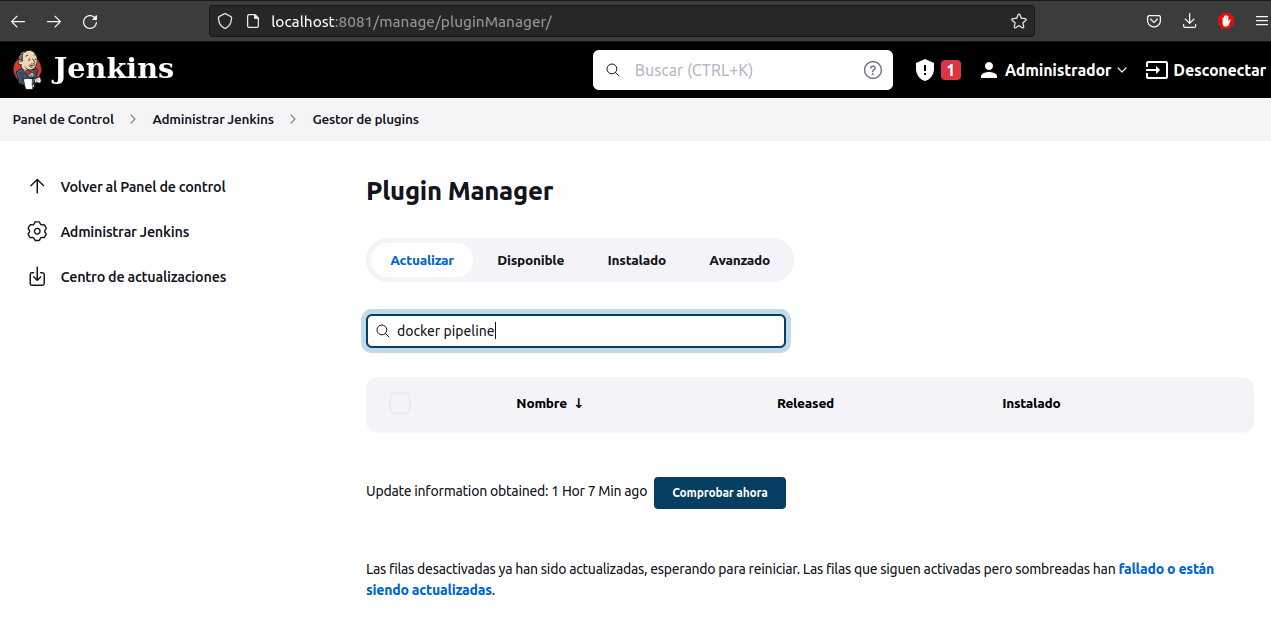
* automatizar la compilación y testeo de sw
* notificar a los equipos correspondientes la detección de errores
* desplegar los cambios en el código que hayan sido validados
* hacer un seguimiento de la calidad del código y de la cobertura de las pruebas
* generar la documentación del proyectos

Podemos ampliar las funcionalidades de Jenkins a través de múltiples pluggins creados por la comunidad, diseñados para ayudarnos en muchas tareas a lo largo de las diferentes etapas del proceso de desarrollo

3- Instalando Plugins y configurando herramientas

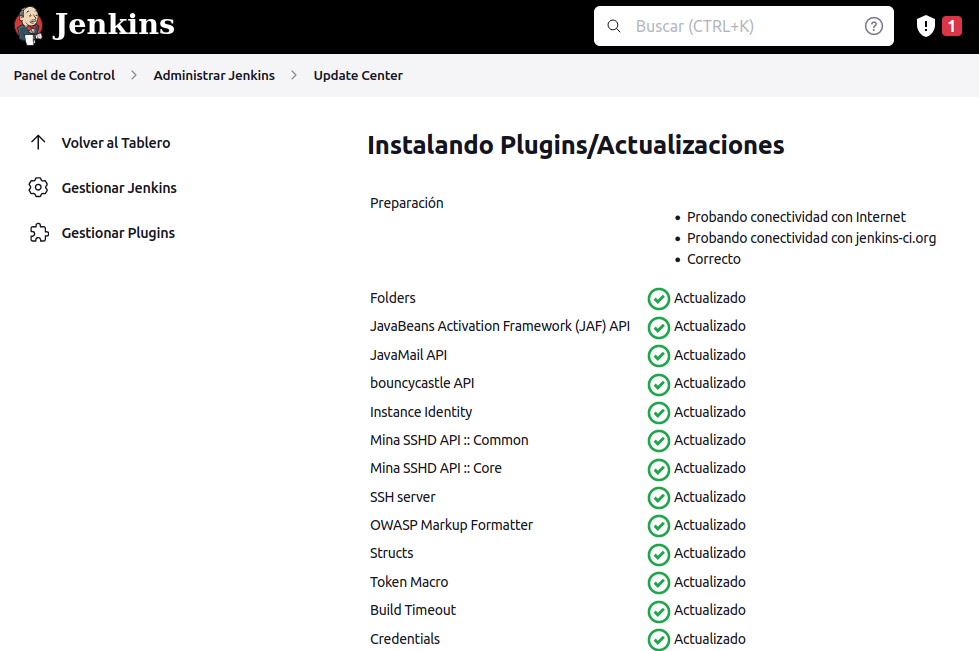
- En Administrar Jenkins vamos a la sección de Administrar Plugins

- De la lista de plugins disponibles instalamos \*\*Docker Pipeline\*\*



- Instalamos sin reiniciar el servidor.

- Abrir nuevamente página de Plugins y explorar la lista, para familiarizarse qué tipo de plugins hay disponibles.



- En la sección de administración abrir la opción de configuración de herramientas

- Agregar maven con el nombre de \*\*M3\*\* y que se instale automáticamente.

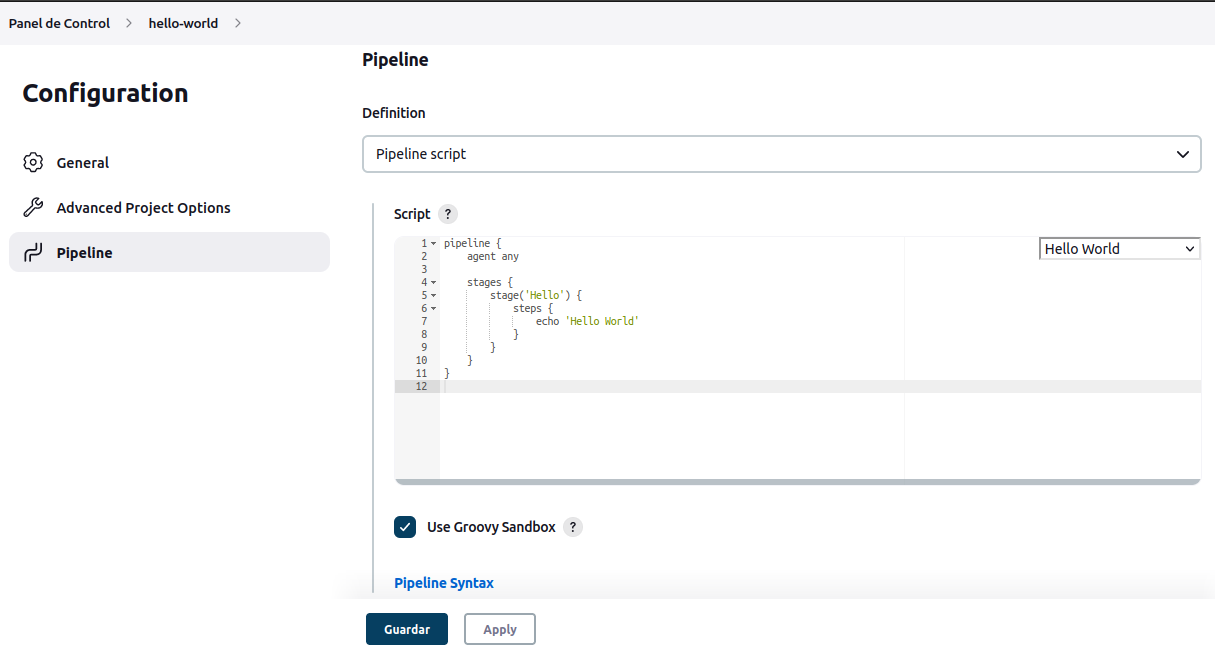


4- Creando el primer Pipeline Job

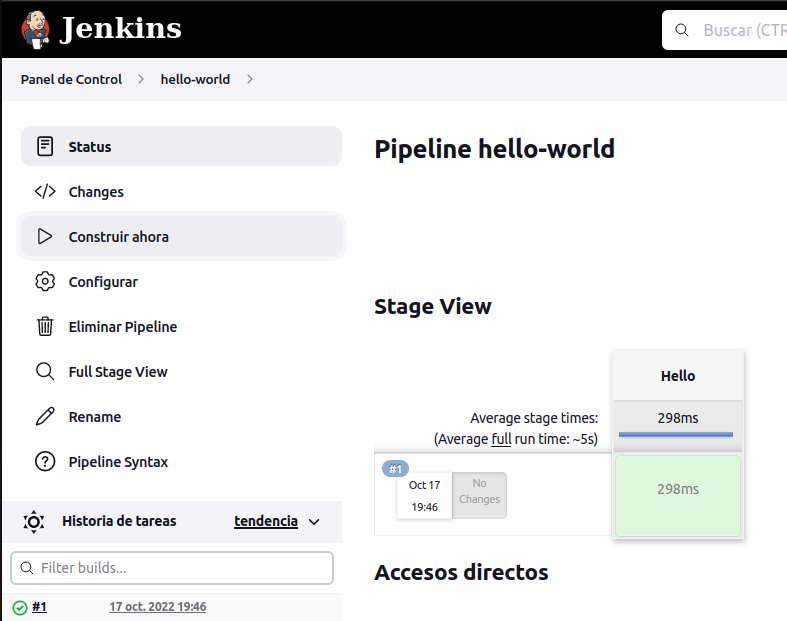
- Crear un nuevo item, del tipo Pipeline con nombre \*\*hello-world\*\*

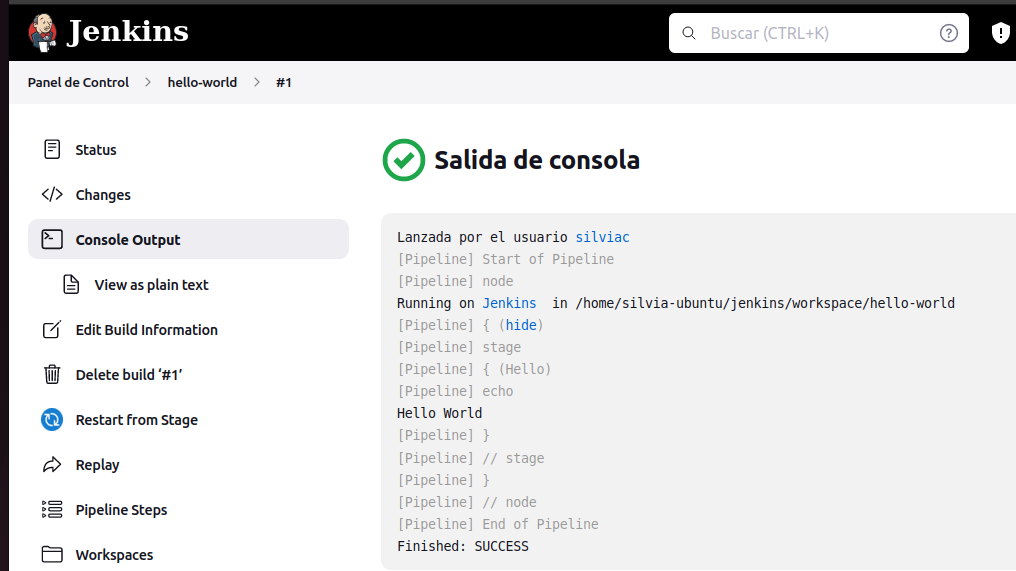


- Una vez creado el job, en la sección Pipeline seleccionamos \*\*try sample Pipeline\*\* y luego \*\*Hello World\*\*



- Guardamos y ejecutamos el Job





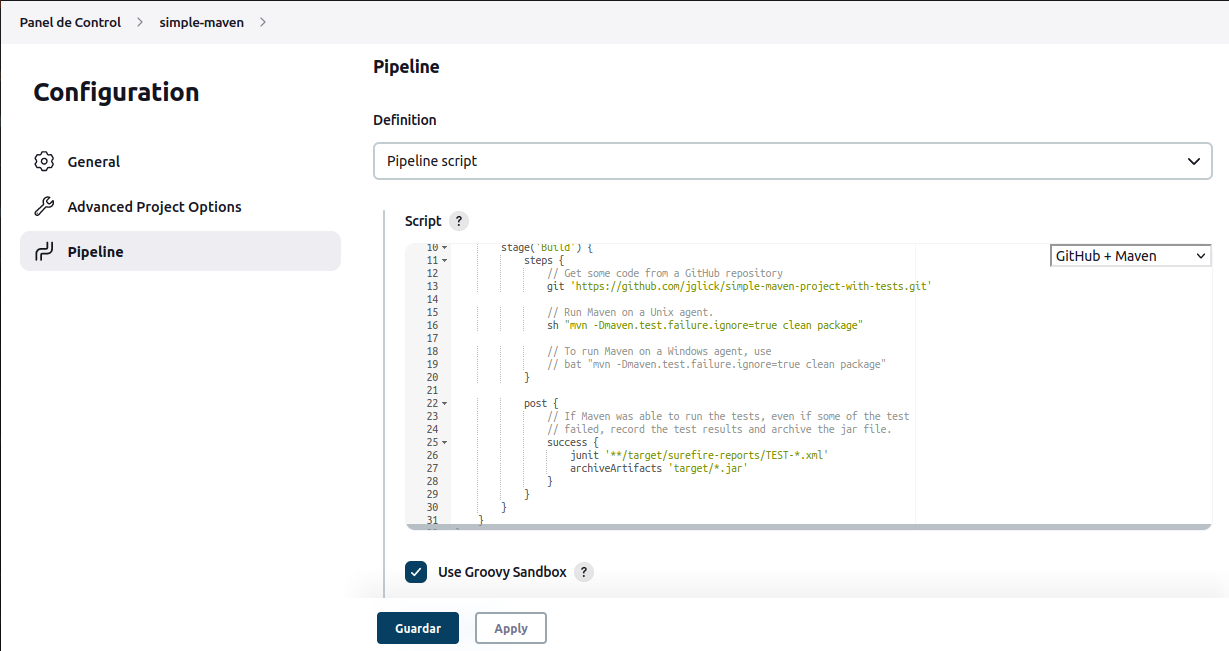
- Analizar la salida del mismo

Jenkins recorre y ejecuta linea por linea el código hasta la realizar un echo y mostrar el mensaje “Hello World” para luego finalizar. Todo esto, de manera automática.

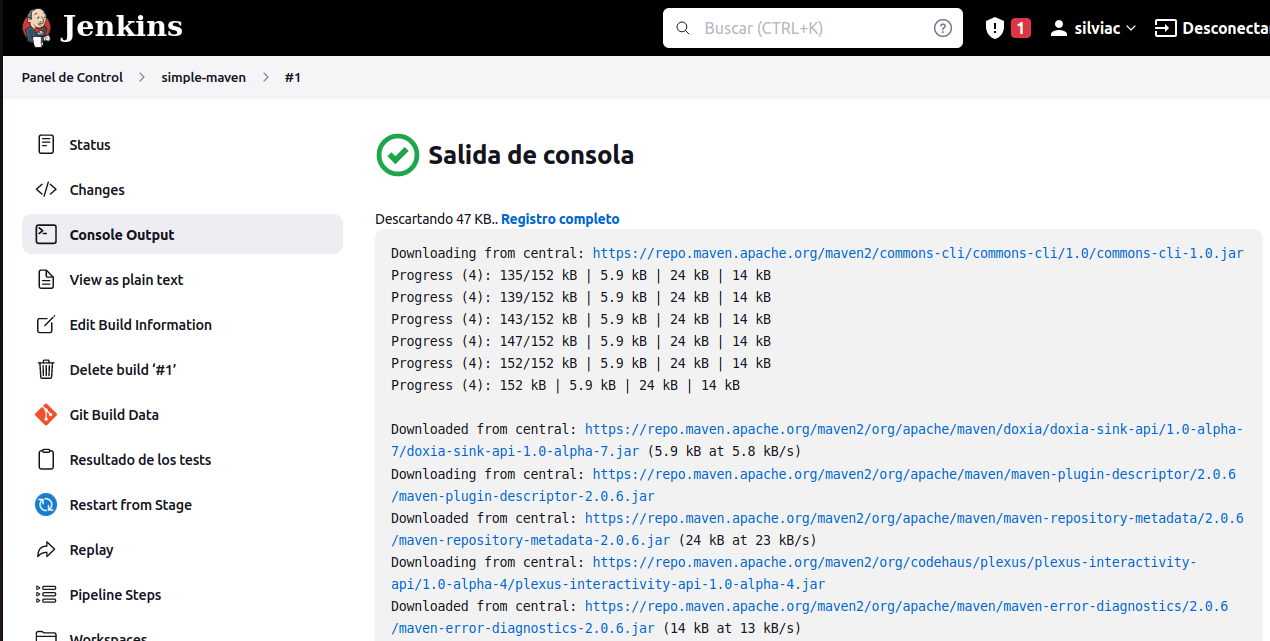
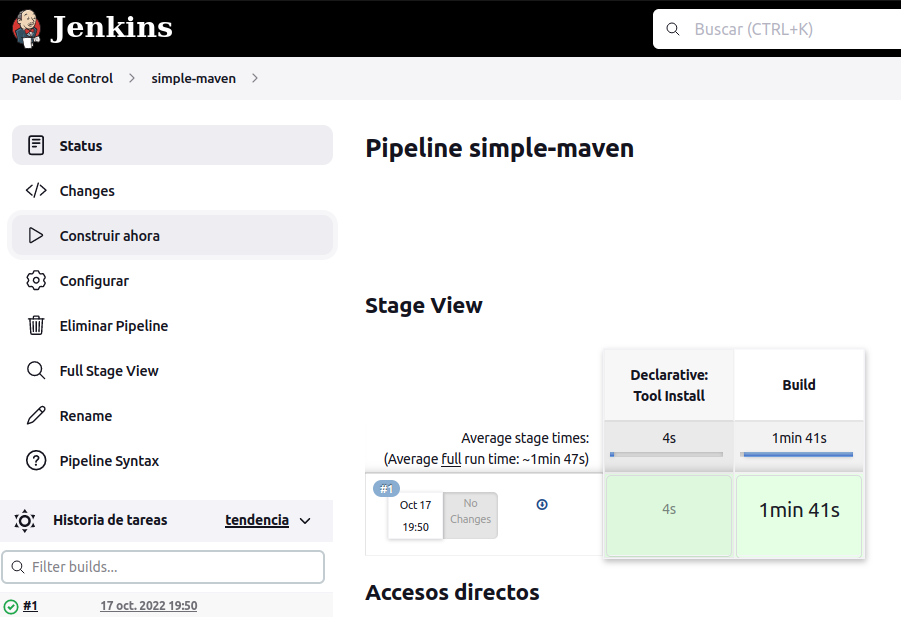
5- Creando un Pipeline Job con Git y Maven

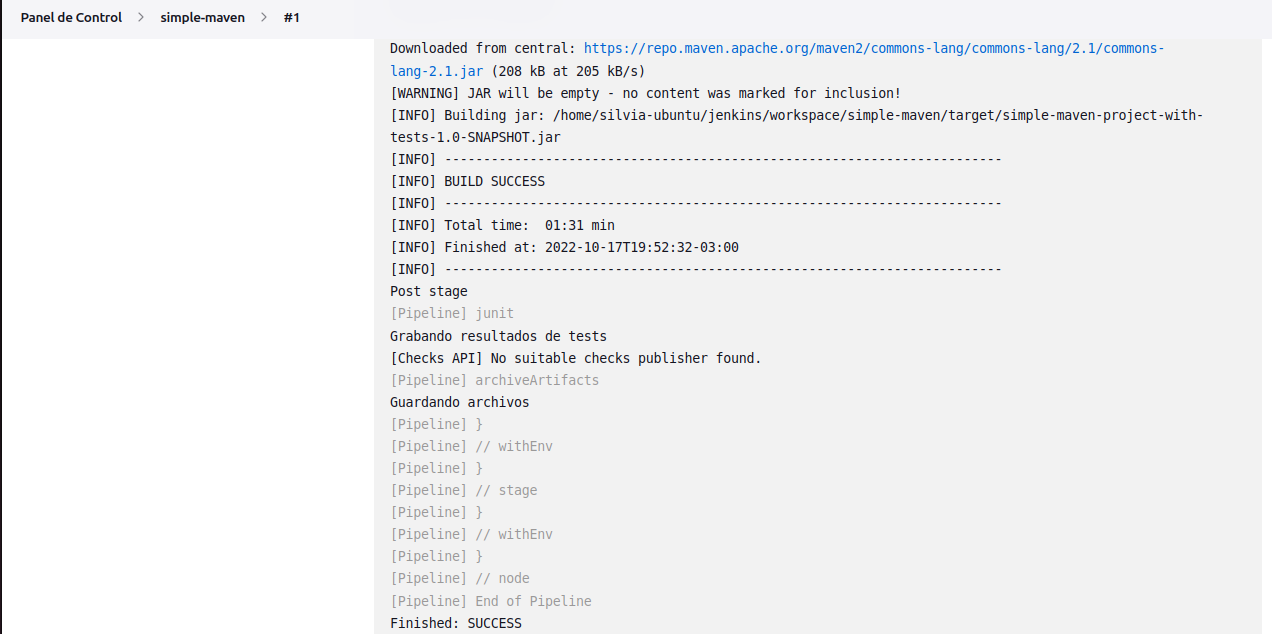
- Similar al paso anterior creamos un ítem con el nombre \*\*simple-maven\*\*

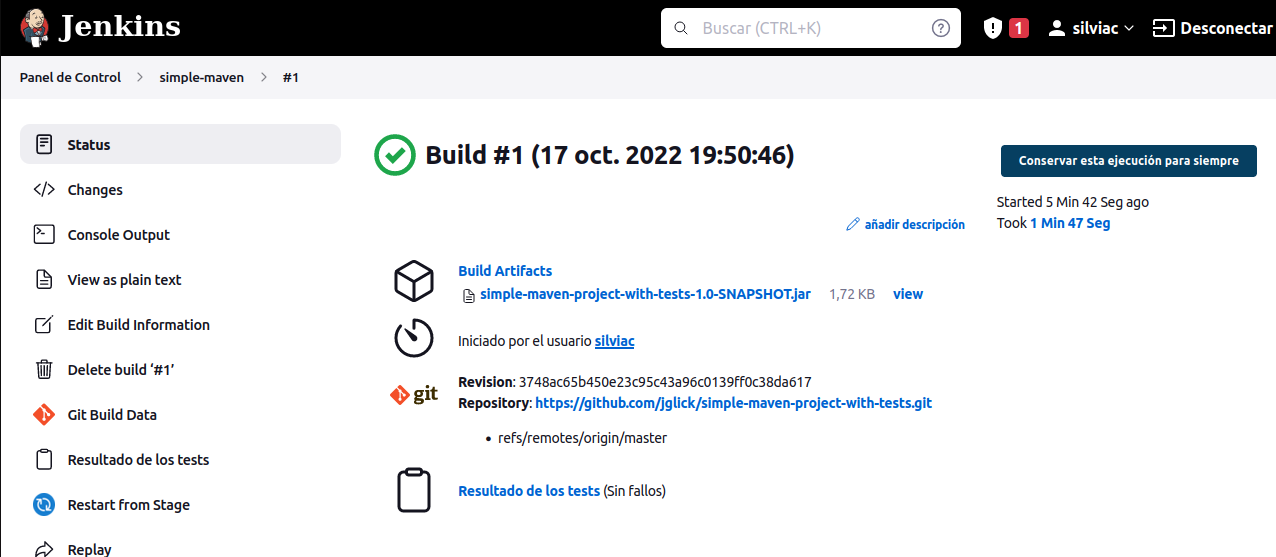
- Elegir \*\*Git + Maven\*\* en la sección \*\*try sample Pipeline\*\*



- Guardar y ejecutar el Job







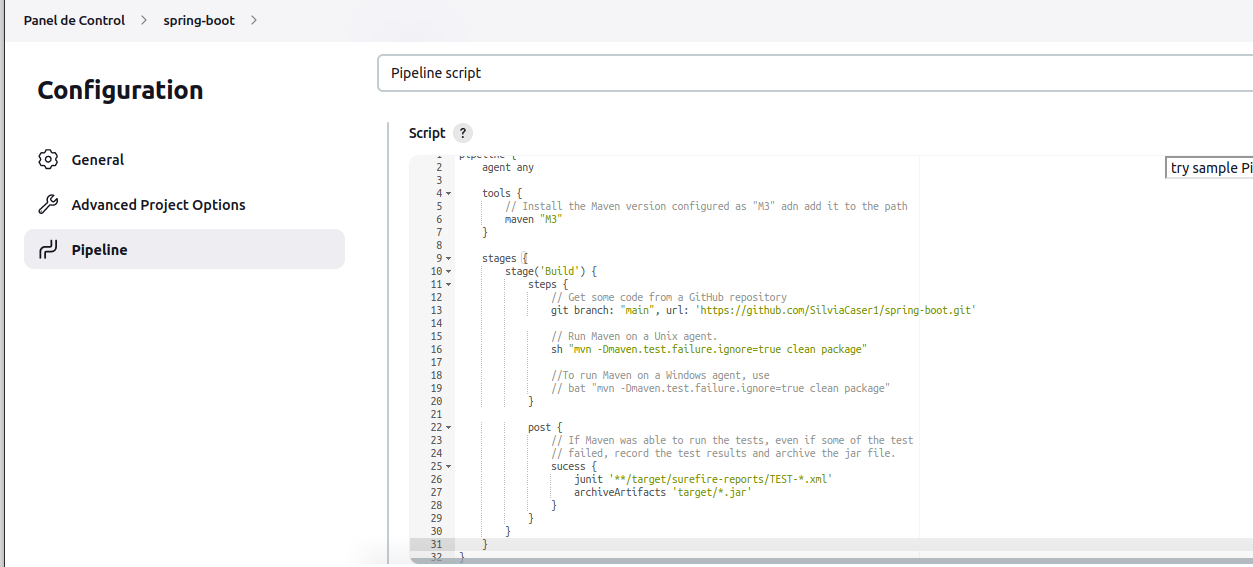
- Analizar el script, para identificar los diferentes pasos definidos y correlacionarlos con lo que se ejecuta en el Job y se visualiza en la página del Job.

Jenkins descarga el proyecto maven del repo corre los tests correspondientes.

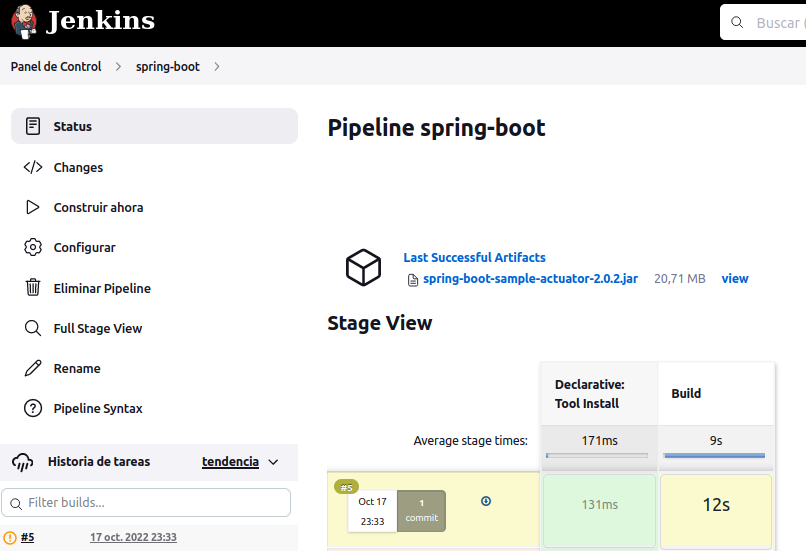
**6- Utilizando nuestros proyectos**

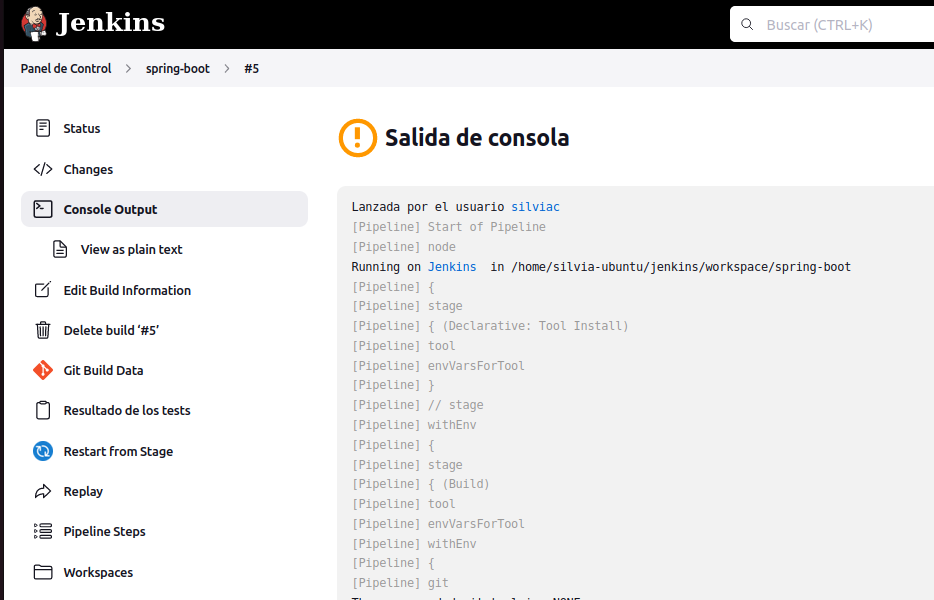
- Utilizando lo aprendido en el ejercicio 5

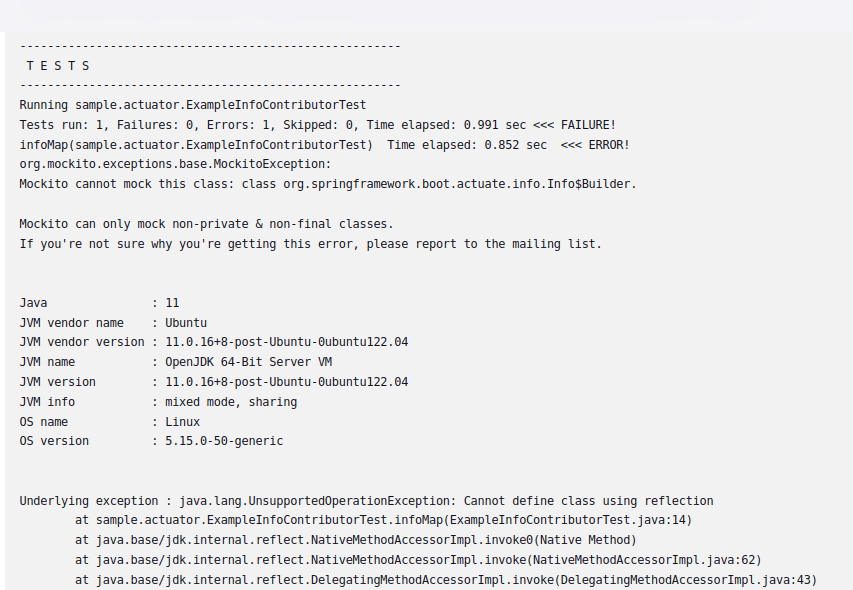
- Crear un Job que construya el proyecto \*\*spring-boot\*\* del [trabajo práctico 6](06-construccion-imagenes-docker.md).



- Obtener el código desde el repositorio de cada alumno (se puede crear un repositorio nuevo en github que contenga solamente el proyecto maven).









- Generar y publicar los artefactos que se producen.

- Como resultado de este ejercicio proveer el script en un archivo \*\*spring-boot/Jenkinsfile\*\*

**7- Utilizando nuestros proyectos con Docker**

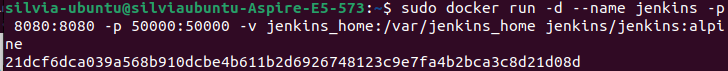
- Extender el ejercicio 6

Se debe crear una imagen de jenkins e instalar docker dentro de esta. La razón es que jenkins no reconoce docker al momento de correr el script. A continuación, se muestra el Dockerfile dentro de la carpeta jenkins para crear la imagen:

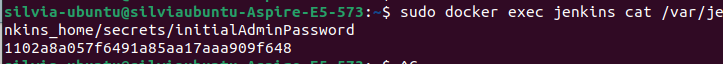
a) descargar la imagen de jenkins desde dockerhub



b) crear el contenedor jenkins en docker

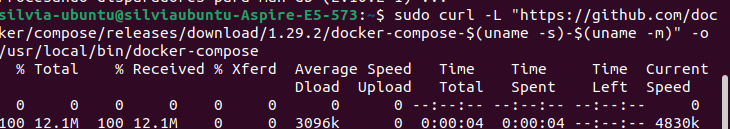


c) configurar el contenedor de jenkins

Iniciar docker en localhost;8080 e instalar los pluggins necesarios. Crear el usuario y loguearse en jenkins.

d) crear contenedor jenkins en docker con docker compose

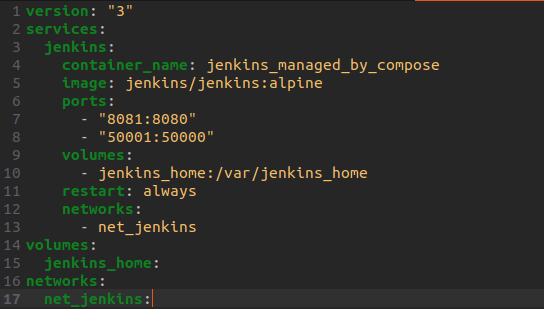
d -1) Instalar docker (descargar binario)

(dar permisos de ejecución)

(versión instalada)



e) crear archivo docker-compose.yaml



f) levantar contenedor jenkins con compose



- Generar y publicar en Dockerhub la imagen de docker ademas del Jar.

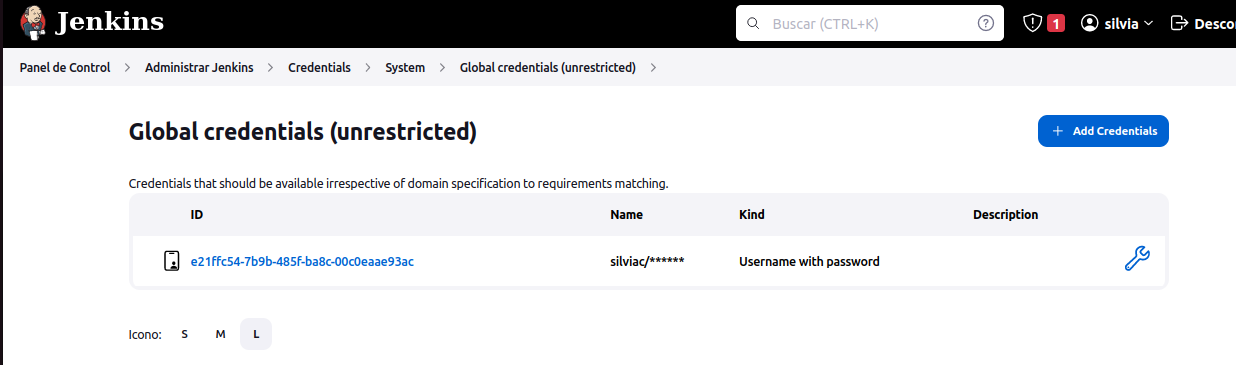
- Se puede utilizar el [plugin de docker](https://docs.cloudbees.com/docs/admin-resources/latest/plugins/docker-workflow) o comandos de shell.

- No poner usuario y password en el pipeline en texto plano, por ejemplo para conectarse a DockerHub, utilizar [credenciales de jenkins](https://github.com/jenkinsci/credentials-plugin/blob/master/docs/user.adoc) en su lugar.

- Como resultado de este ejercicio proveer el script en un archivo \*\*spring-boot/Jenkinsfile\*\*

- Referencia: https://tutorials.releaseworksacademy.com/learn/building-your-first-docker-image-with-jenkins-2-guide-for-developers

Para subir el proyecto a docker-hub, primero se crea la credencial con los datos de docker-hub. Se usa el id que se genera en el script para que sea validado el usr y pwd, para luego, subirse el proyecto,



El siguiente paso es crear el item spring-boot-docker:

