

SQA S.A - Innovación

CT - Porvenir

Versión 1.0

Tabla de Contenido

[Descripción General 4](#_Toc81576226)

[Evaluación Histórica 5](#_Toc81576227)

[Accesibilidad y flexibilidad 6](#_Toc81576228)

[¿En qué se diferencian las pruebas continuas de la automatización de pruebas? 7](#_Toc81576229)

[Sistemas involucrados en el proceso 9](#_Toc81576230)

[Flujo Zona Publica 11](#_Toc81576231)

[Configuración de Estructura Screeplay 12](#_Toc81576232)

[Configuración de Estructura Screeplay - Features 14](#_Toc81576233)

[Configuración de Estructura Screeplay – Steps Definitions 15](#_Toc81576234)

[Configuración de Estructura Screeplay – Tareas (Tasks) 15](#_Toc81576235)

[Configuración de Estructura Screenplay – Interaciones (Interactions) 16](#_Toc81576236)

[Configuración de Estructura Screenplay – Ejecutores (Runners) 16](#_Toc81576237)

[Configuración de Estructura Screenplay – Interfaces de Usuario (User interface) 17](#_Toc81576238)

[Configuración de Estructura Screenplay – Ejecutores (Runners) 17](#_Toc81576239)

[Jenkins para proyectos Maven 23](#_Toc81576240)

[Jenkins para proyectos Gradle 24](#_Toc81576241)

[Reportes 24](#_Toc81576242)

[Cucumber 27](#_Toc81576243)

[Serenity BDD 29](#_Toc81576244)

[Ejecutar comando para organizar el reporte de SerenityBDD 32](#_Toc81576245)

[Configuración de Agentes en Jenkins 34](#_Toc81576246)

[Descargar Agente en dispositivo local 41](#_Toc81576247)

[Configuración de Agente con Pipeline 43](#_Toc81576248)

[Ejecutar un Pipeline 46](#_Toc81576249)

[PRINCIPALES BENEFICIOS DE LAS PRUEBAS CONTINUAS 48](#_Toc81576250)

[ASPECTOS A TENER EN CUENTA 49](#_Toc81576251)

Versionamiento

|  |  |  |  |
| --- | --- | --- | --- |
| MODIFICADO POR | VERSIÓN | FECHA | DESCRIPCIÓN |
| Javier Ricardo Charry Barrera | 1.0 | 10/08/2020 | Inicio de documento |

# Descripción General

Implementación de esquema de integración continua y construcción automática en pruebas QA: Revisando la percepción de recibir y generar consistentemente aplicaciones hacia ambientes de QA y producción para evitar errores.

•Automatización de despliegues: Permite eliminar errores en salidas a producción, promueve mayor

•Gestión del Ciclo de Vida de Desarrollo (Requerimientos → Release): Brindamos prácticas para gestionar el ciclo de vida de software, permitiéndoles aprender sobre empoderamiento y así, tener el control de su software. Herramientas para controlar y utilizar sus códigos fuente.

•Automatización de Pruebas Continuas: Automatización de casos de prueba, generación de datos de pruebas, virtualización de servicios.

• Seguridad en el Proceso de Desarrollo

• Monitoreo de Negocio (Ops)

• Implementación de infraestructura de nube híbrida para Contenedores

• Gestión del Cambio en la Organización

Prerrequisitos

Tener elaborado los scripts de automatización en un proyecto (verificar que se ejecuta de forma correcta localmente).

Tener Java instalado, versión 11 o superior. Asegurarse de que las variables de entorno están correctamente configuradas.

Mucha capacidad de investigación y recursividad.

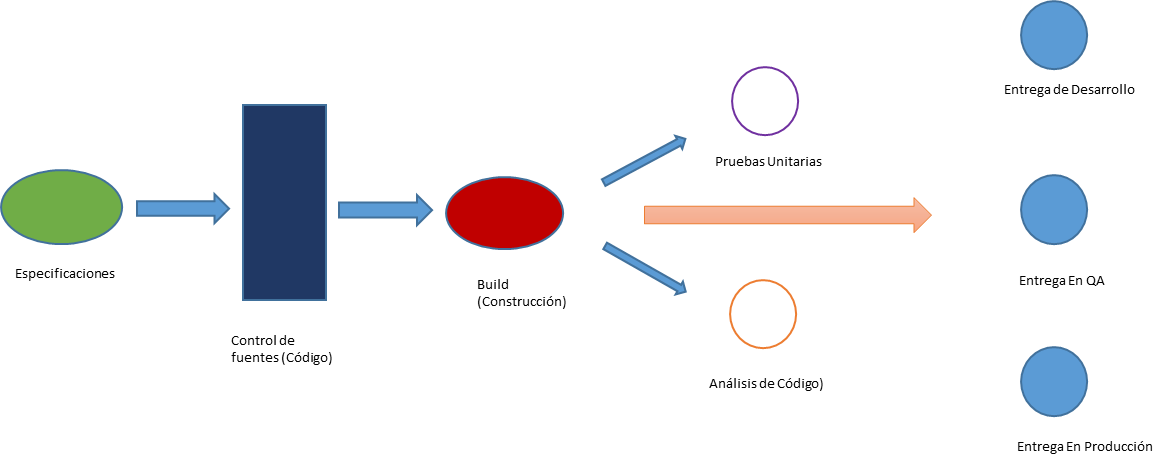
## Evaluación Histórica

La antigua forma de realizar pruebas era centrada en realizar pruebas funcionales de forma manual y convertirlas en transacciones una a una. El software se pasó de un equipo a otro. Un proyecto tendría fases definidas de desarrollo y garantía de calidad. Los equipos de control de calidad siempre querían más tiempo para garantizar la calidad. El objetivo era que la calidad prevaleciera sobre el cronograma del proyecto.

Sin embargo, las empresas o equipos de tecnología quieren una entrega más rápida de software al usuario final. Cuanto más nuevo es el software, mejor se puede distribuir y aumentar el potencial de entradas para el Core del negocio. Por lo tanto, se desarrolló una nueva forma de pruebas.

Continuo significa pruebas ininterrumpidas realizadas de forma continua. En un proceso de DevOps continuo, un cambio de software (candidato de lanzamiento) se mueve continuamente desde el desarrollo hasta las pruebas y la implementación.

## Accesibilidad y flexibilidad



El código se desarrolla, entrega, prueba e implementa continuamente.

Por ejemplo, cada vez que un desarrollador verifica el código en el servidor de código fuente, como Jenkins, se ejecuta un conjunto automatizado de pruebas unitarias en el proceso continuo.

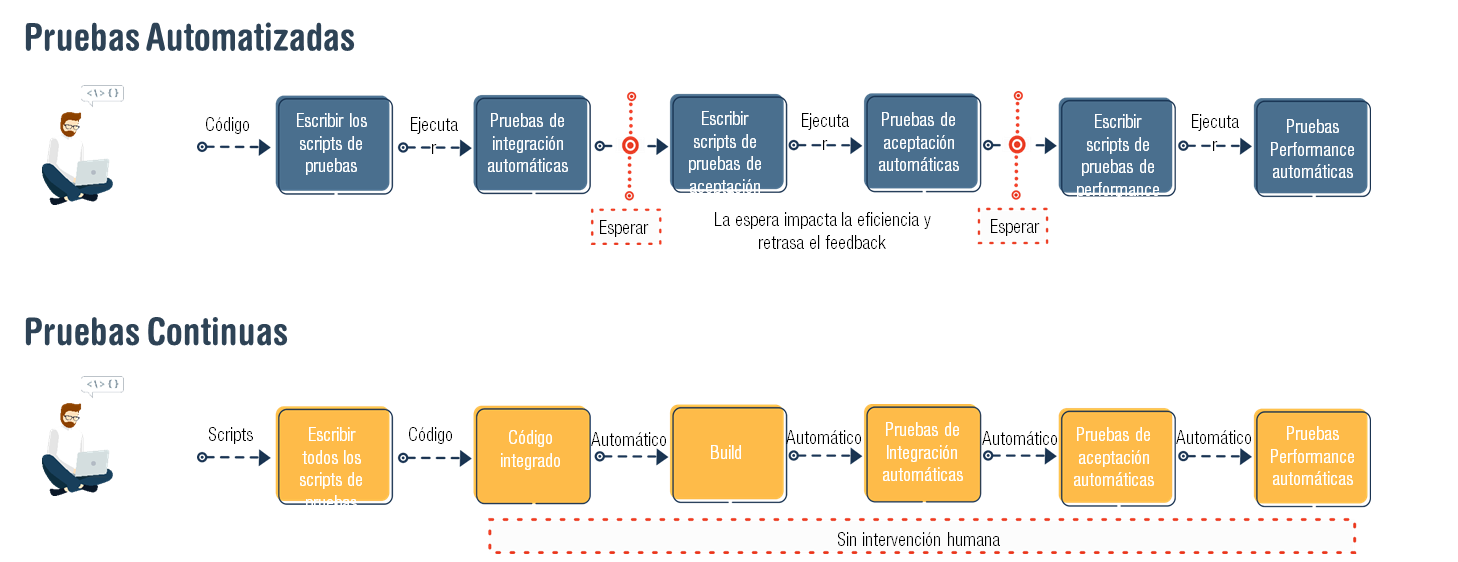
* Si las pruebas fallan, la compilación se rechaza y se notifica al desarrollador.
* Si la compilación pasa la prueba, se implementa en servidores de control de calidad y rendimiento para pruebas funcionales y de carga exhaustivas. Las pruebas se realizan en paralelo.
* Si las pruebas pasan, el software se implementa en producción.

La prueba continua es un pequeño engranaje en el ciclo de desarrollo, integración e implementación continuos.

## ¿En qué se diferencian las pruebas continuas de la automatización de pruebas?

**Automatización de pruebas frente a pruebas continúas**

|  |  |  |
| --- | --- | --- |
| Parámetro | Automatización de pruebas | Prueba continua |
| Definición | La automatización de pruebas es un proceso en el que se utilizan herramientas o software para automatizar tareas. | Es una metodología de prueba de software que se enfoca en lograr una mejora y calidad continuas. |
| Objetivo | Un conjunto de tareas similares o repetitivas que una máquina puede ejecutar, más rápido y con menos errores. | El proceso de prueba continuo ayuda a encontrar el riesgo, abordarlo y mejorar la calidad del producto. |
| Requisito previo | La automatización en las pruebas es posible sin integrar las pruebas continuas. | Las pruebas continuas no se pueden implementar sin la automatización de pruebas. |
| Tiempo | El lanzamiento del software puede tardar de un mes a años. | La versión del software puede publicarse semanalmente o cada hora. |
| Retroalimentación | Comentarios regulares después de probar cada versión. | La retroalimentación en cada etapa debe ser instantánea. |
| Historia | Las pruebas automatizadas se han realizado durante décadas para acelerar el proceso de prueba. | La prueba continua es un concepto relativamente nuevo. |



Cómo hacer pruebas continuas

* Uso de herramientas para generar un conjunto de automatización de pruebas a partir de historias / requisitos de usuario
* Crear entorno de prueba.
* Copie y anonimice los datos de producción para crear un banco de datos de prueba
* Utilice la virtualización de servicios para probar la API
* Pruebas de rendimiento en paralelo

Análisis del proceso actual Porvenir

El objetivo de este esta sección es describir cómo porvenir está manejando el proceso ahora, antes de optimizarlo y automatizarlo. Se busca responder a preguntas como: ¿Cuáles son los pasos que debe seguir alguien dentro de la empresa cada vez para que este proceso funcione?, ¿Qué software tienen que utilizar? ¿Dónde obtienen la información que necesitan? ¿Qué ocurre si algo va mal? Entre otras.

### Sistemas involucrados en el proceso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Descripción** | **Sistema** | **Se necesita rol de usuario** | **Tecnología usada por el sistema** | **Versión** |
| Qué **Plataforma**está usando: | Azure. |  | Portal Zona Franca | 2020 |
| ¿Qué **servicio**de control de versiones usa? | GitLab | Usuario de auxiliar solo para aprobación y eliminación. | N/A | N/A |
| ¿Qué herramienta de **compilación**de código utiliza? | Maven | N/A | N/A | N/A |
| ¿Qué herramienta utiliza para gestionar los **pipelines**? | Jenkins  y está montado en un servidor de la nube (Azure) | Usuario y Contraseña | N/A | (2.277.1) |
| En qué **tecnologías**de desarrollo están construida la **aplicación web**con la realizará el**piloto de CT**? | **Front**    Java (Freamework Liferay ) | No requiere usuario | N/A | N/A |
| **Back** | Java (EJB’s) | N/A | N/A |
| Base de datos de salida | Oracle | N/A | N/A | N/A |

Automatización Zona Publica

Dentro de la propuesta de lo evaluado para Porvenir, se realizo dos tipos de automatización una con el modelo de Maven con el lineamiento interno, sin embargo, este esquema requiere todo el conjunto de librerías para cualquier tipo de proyecto, por lo cual es necesario tener los permisos de Maven desde su repositorio central expuesto en internet e ser tener configurados todo los artefactos que se requieran.

Dentro de este documento vamos de dejar descrito el modelo utilizado con modelo final de automatización para Zona Publica y el paso a paso de uso.

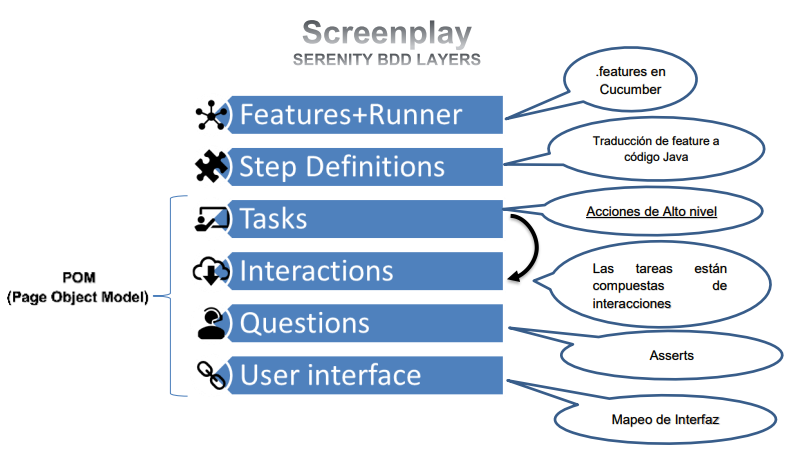
Automatización ScreenPlay

Para la mejora de automatización lo que definitivo parte de la siguiente estructura: SERENITY BDD + SCREENPLAY con CUCUMBER Patrón Screenplay. Trabajábamos en el Modelo Page Object (POM), y el cambio que pretendemos realizar viene de analizar la programación orientada a objetos (POO). Esta se soporta en 4 pilares.

* Abstracción
* Encapsulamiento
* Herencia
* Polimorfismo

Entre otras definiciones que no se detallan en este documento, solo queda el modelo de automatización.

La linea base de automatización de Zona Publica es la siguiente:



Se creo un proyecto base con el siguiente flujo automatizado en zona publica:

### Flujo Zona Publica

Ingreso con usuario y password

Seleccionar Empresa

1. Seleccionar empleados

Seleccionar Retiro de Cesantías

Aceptar condiciones

Ingreso de datos y Selección de tipo de cc y documento (Registrados para tal fin)

Continuar proceso

1. Grabar Retiro

Seleccionar Medio de Pago

Concepto

Medio de Pago

Valor de retiro

Continuar proceso

1. Completar información

Confirmar información

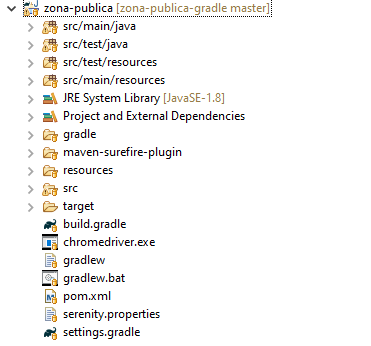
Continuar proceso

1. Autorizar el retiro
2. Fin de la transacción

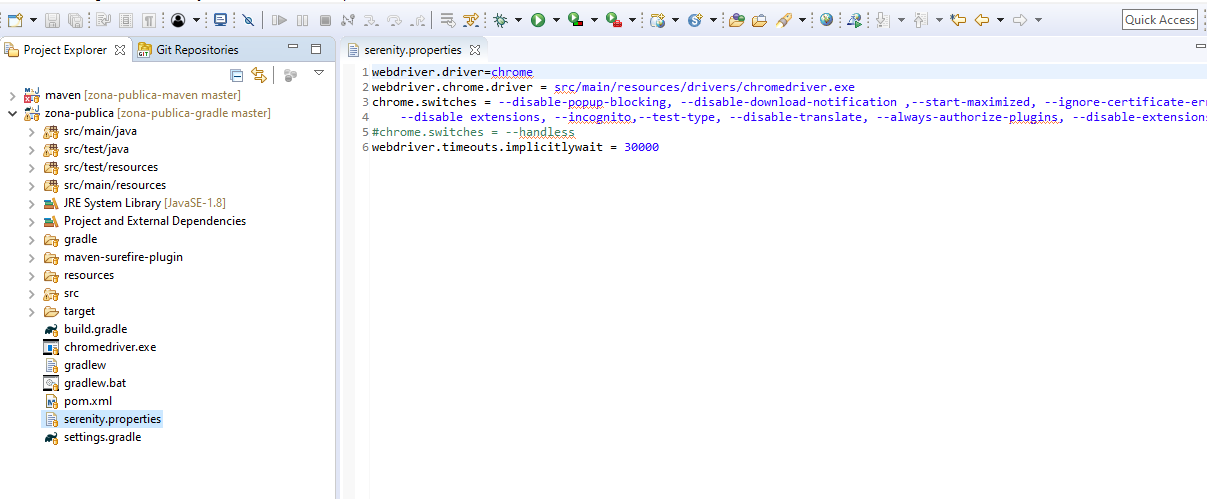
### Configuración de Estructura Screeplay

En la maquina asignada para la automatización se crea estructura de la siguiente forma:

1. Carpetas principales main/java, test/java, main resources



1. Archivo de configuración serenity.propierties



Esta es la base de configuración

webdriver.driver=chrome

webdriver.chrome.driver = src/main/resources/drivers/chromedriver.exe

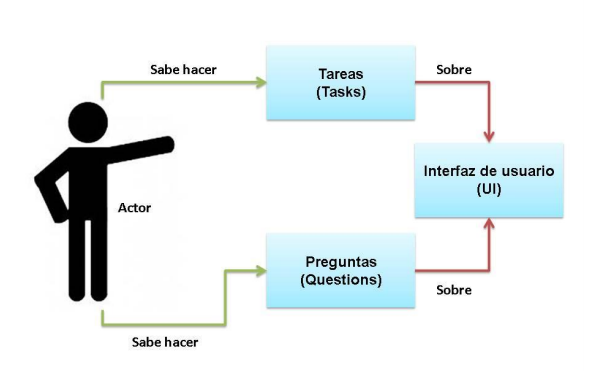
chrome.switches = --disable-popup-blocking, --disable-download-notification ,--start-maximized, --ignore-certificate-errors, --allow-running-insecure-content, \

--disable extensions, --incognito,--test-type, --disable-translate, --always-authorize-plugins, --disable-extensions, --disable-dev-shm-usage,

#chrome.switches = --handless

webdriver.timeouts.implicitlywait = 30000

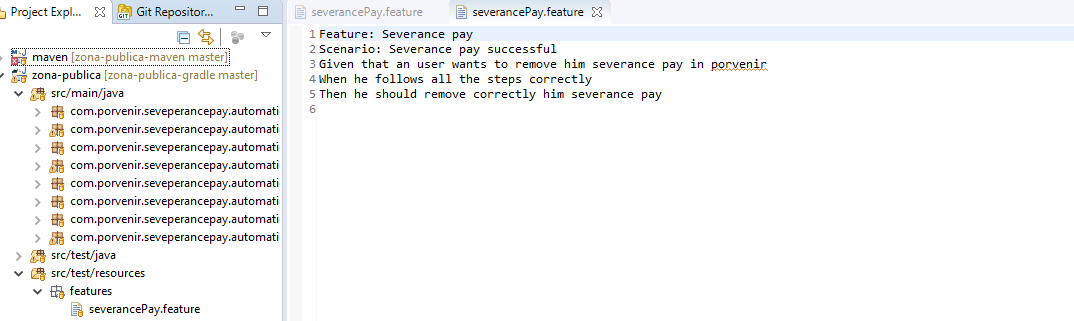
De acuerdo a al patrón utilizado se maneja la siguiente estandarización:



Dentro del proyecto se crea una historia de usuario con modelo Cucumber base en eclipse, procederemos donde se crea la el Feature (característica) crear una historia de usuario.

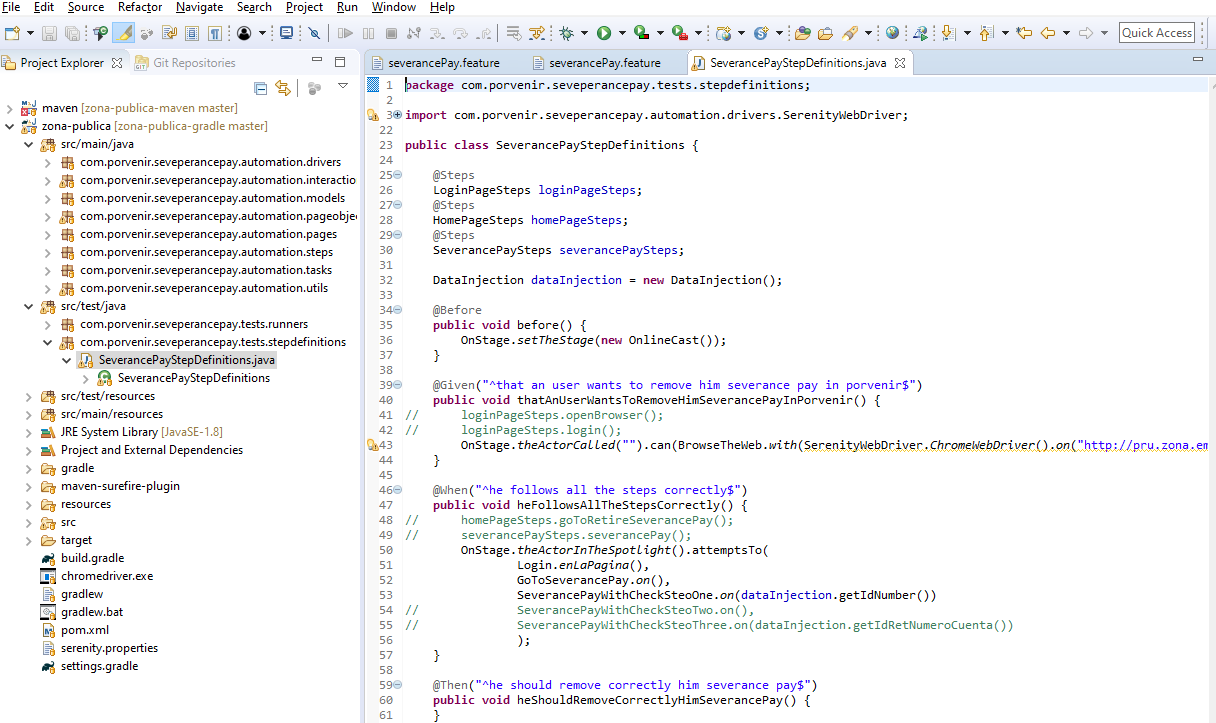
### Configuración de Estructura Screeplay - Features

1. Para esto, iremos a nuestro proyecto y en la carpeta “src/test/resources” encontraremos un paquete llamado “features”.



### Configuración de Estructura Screeplay – Steps Definitions

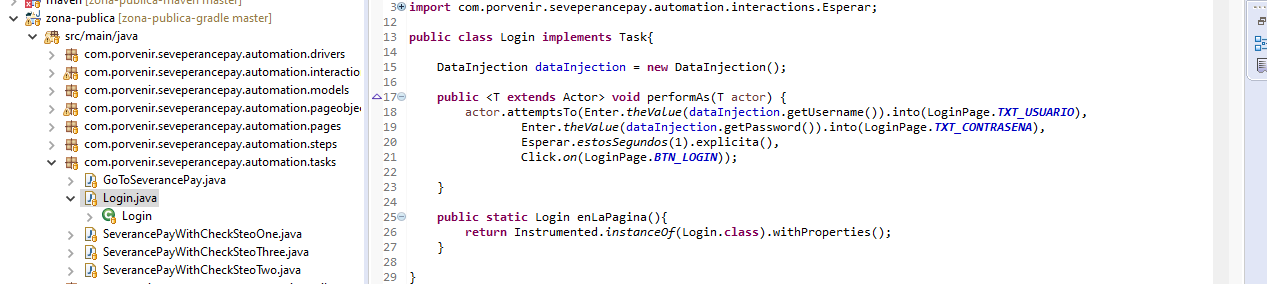
1. Se crearon StepDefinition con los métodos sugeridos, e iremos a nuestra carpeta “src/test/java”



En lo anterior ser ven las configuraciones iniciales y pasos a seguir en la automatización

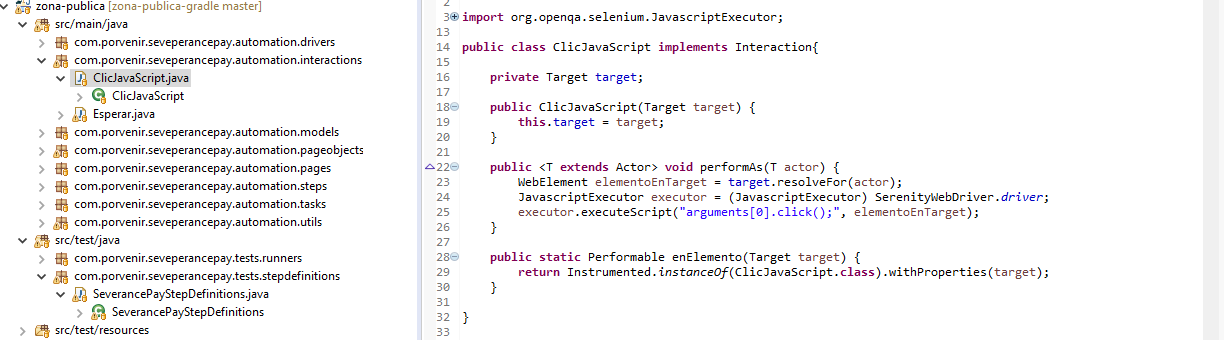
### Configuración de Estructura Screeplay – Tareas (Tasks)

1. Creación de tareas (Tasks), en acciones se creó la interfaz de usuario (UI) para el desarrollo de cada paso descrito en la historia de usuario. En este paquete encontraremos 4 tareas que se realizan dentro del modelo de la historia de usuario.



### Configuración de Estructura Screenplay – Interaciones (Interactions)

1. En este paquete se van a encontrar el mapeo de la interfaz, donde capturamos todos los elementos con los que se interactuaron durante la automatización.



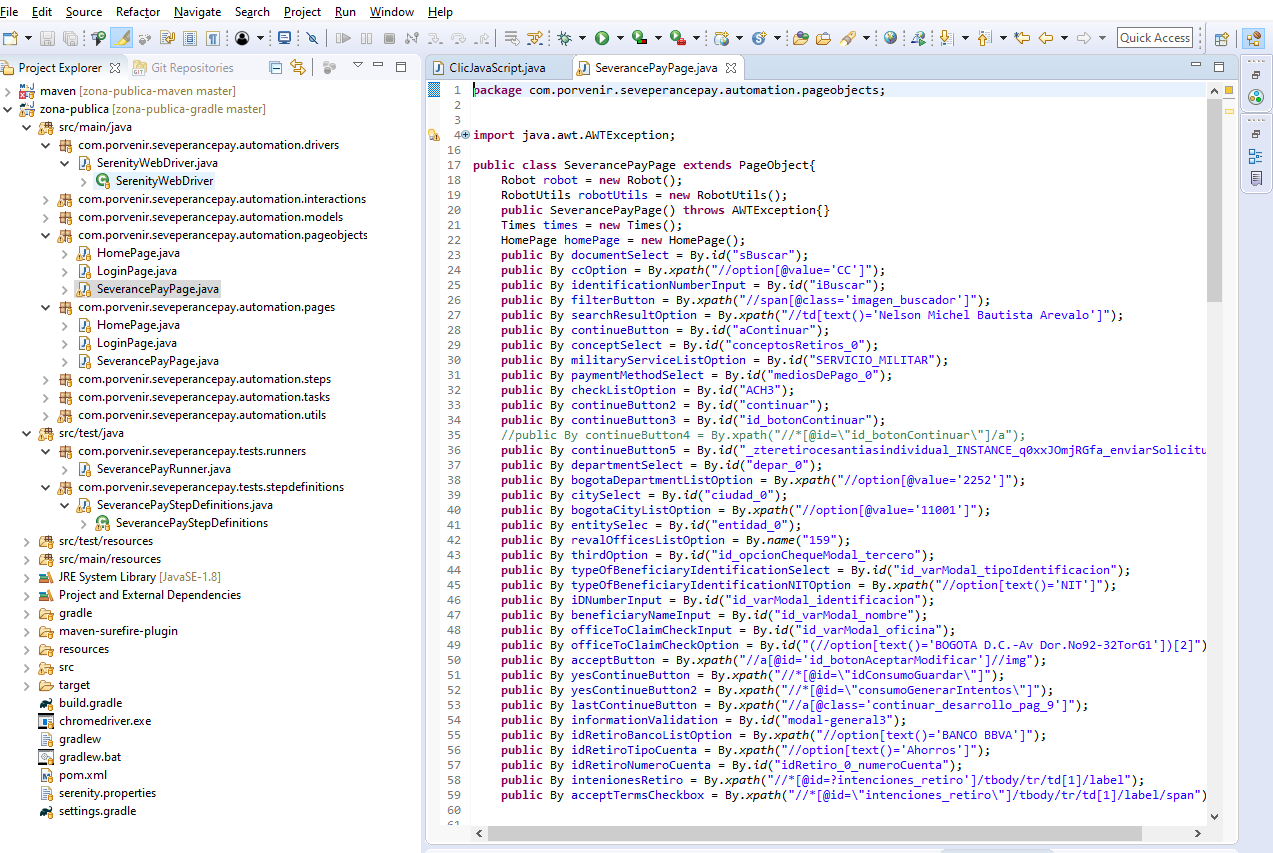
### Configuración de Estructura Screenplay – Ejecutores (Runners)

1. Acá se dejan validaciones pertinentes a la automatización

Acá se encuentra los ejecutores de los features y tags, glue (donde se encuentra el step definición) y conexiones de URL del driver y las conexiones pertinentes.

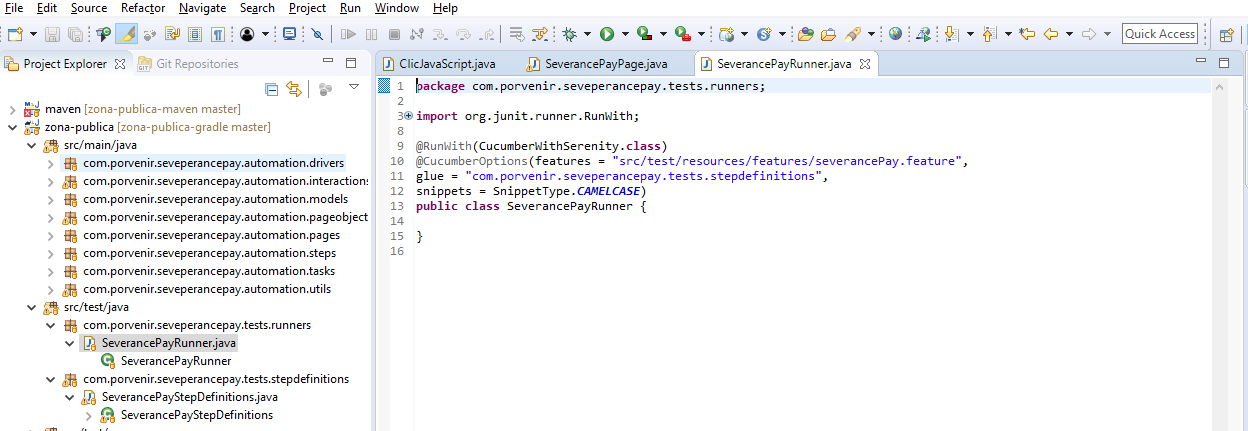
### Configuración de Estructura Screenplay – Interfaces de Usuario (User interface)

1. Las UI son el mapeo de la interfaz donde se capturaron todos los elementos con los cuales se interactuaron durante la automatización



### Configuración de Estructura Screenplay – Ejecutores (Runners)

1. En este método se encontrará el ejecutor de los features, tags, glue (donde se encuentra el steps definitions.



Configuraciones sugeridas de Maven

Para configurar Maven ingresaremos los siguientes comandos:

* sudo yum install maven

Nos pedirá ingresar la letra ‘y’ para permitir instalaciones o actualizaciones

* mvn –version

Al final debemos ver el mensaje en pantalla de PuTTY

Apache Maven 3.0.5 (Red Hat 3.0.5-17)

Maven home: /usr/share/maven

Java version: 1.8.0\_282, vendor: Red Hat, Inc.

Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86\_64/jre

Default locale: en\_US, platform encoding: UTF-8

En la respuesta anterior podremos ver el valor del Maven home el cual vamos a copiar y pegar en el siguiente comando

* export MAVEN\_HOME=”/usr/share/maven”
* echo $MAVEN\_HOME

Se nos mostrara la ruta que hemos ingresado

Configuración de Gradle

Para configurar Gradle ingresaremos los siguientes comandos:

* wget https://services.gradle.org/distributions/gradle-5.0-bin.zip -P /tmp
* sudo unzip -d /opt/gradle /tmp/gradle-5.0-bin.zip
* export GRADLE\_HOME=/opt/gradle/gradle-5.0
* export PATH=${GRADLE\_HOME}/bin:${PATH}
* gradle -v

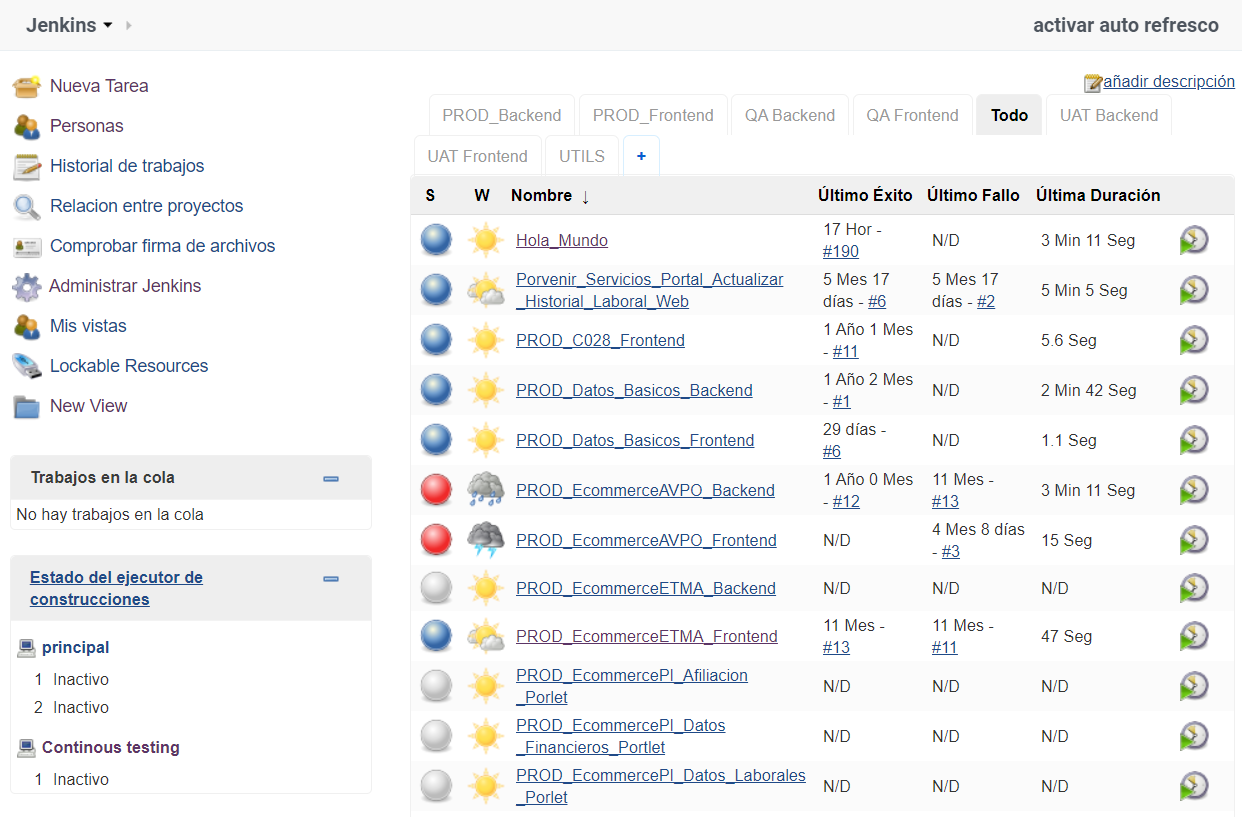
Al final debemos ver el mensaje en pantalla de PuTTY



Configuración de Gradle

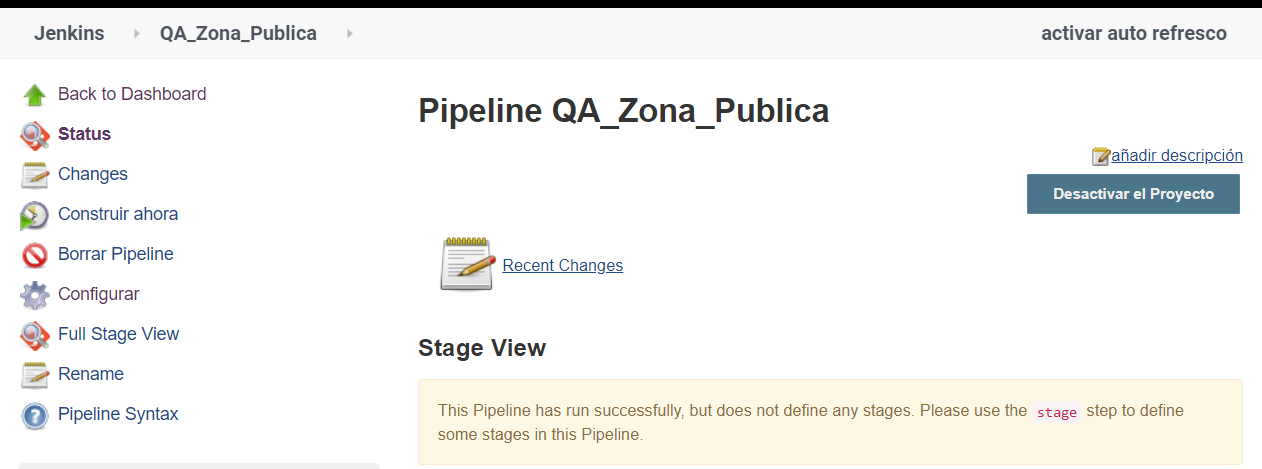
Finalmente veremos la interfaz de Jenkins donde fue configurado el Pipeline de Jenkins para Zona Publica, el cual se debe contar con los accesos pertinentes al Jenkins de provenir con usuario y contraseña.

En este se verán múltiples proyectos donde debemos ingresar a la búsqueda del proyecto con el siguiente nombre **“QA\_Zona\_Publica”.**

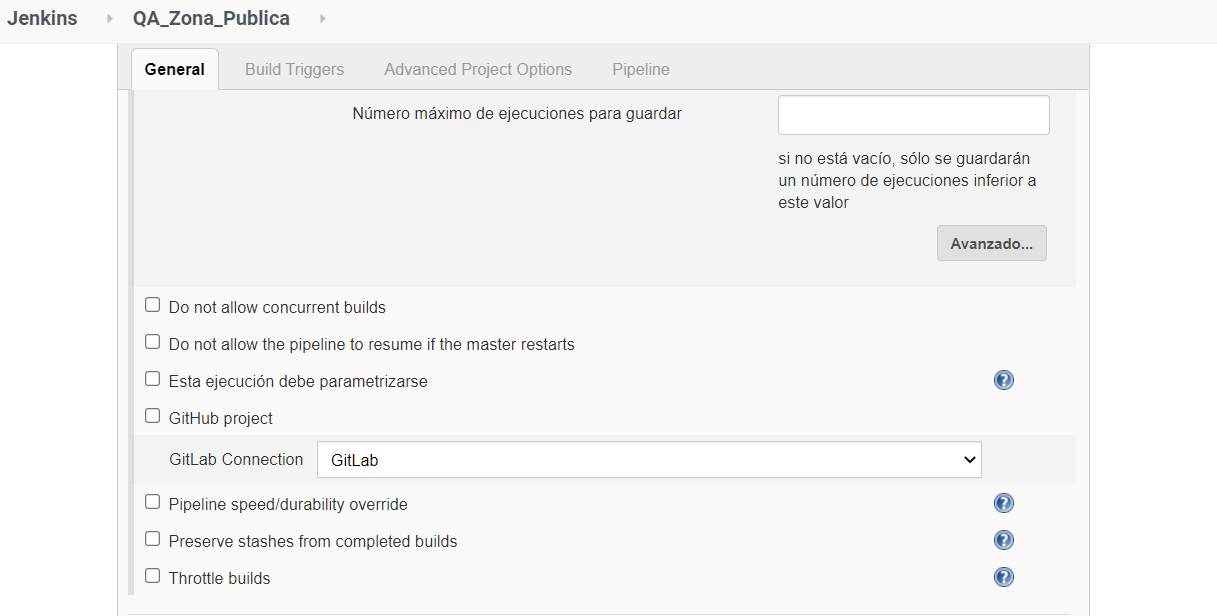


Jenkins configuración base del proyecto Pipeline Zona Publica

Dentro este vamos a ubicarnos en Configurar para ver las configuraciones del proyecto ingresar en la opcion de configurar



En genera se esta ingresado esta relacinado con el repositorio de GibLab de porvenir



GitLab de repositorio ubicado en la siguiente dirección

http://10.190.40.132:8080/project/QA\_Zona\_Publica

Jenkins configuración base del proyecto Pipeline Zona Publica – Pipeline Scritp

En este se encutra el script de integracion continua utilizado como base para la ejecución de la automatizacion de finido para esta actividad, este realiza la descarga de codigo, compilacion del test mediante codigo y su estructura es la siguiente:

node() {

cleanWs()

checkout([$class: 'GitSCM', branches: [[name: '\*/master']], doGenerateSubmoduleConfigurations: false, submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'Git', url: 'http://10.110.30.81:9090/Celula\_Automatizacion/zona-publica.git']]])

stash name: 'SourceCode', includes: "\*\*"

}

node('Continous testing'){

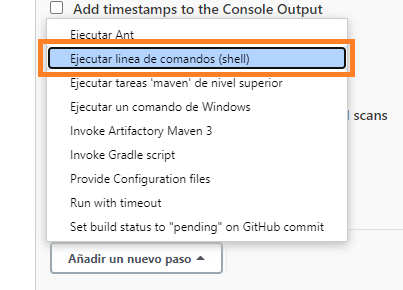
unstash name: 'SourceCode'

bat "gradle clean test aggregate"

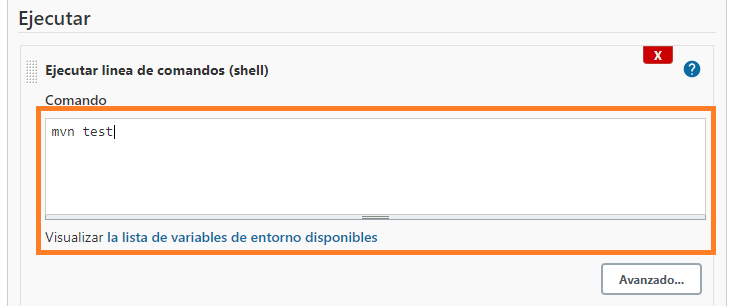
}

## Jenkins para proyectos Maven

En caso para el ejecicio de la automatizacion se sugiere utililzar sea maven o gradle y la fomra para este primero se puede realizar con la siguiente opción Ejecutar línea de comandos (shell)

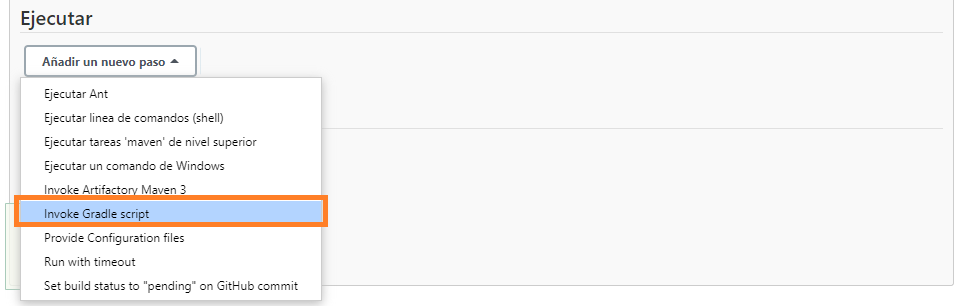


Escribimos el comando **mvn test** en la caja de texto

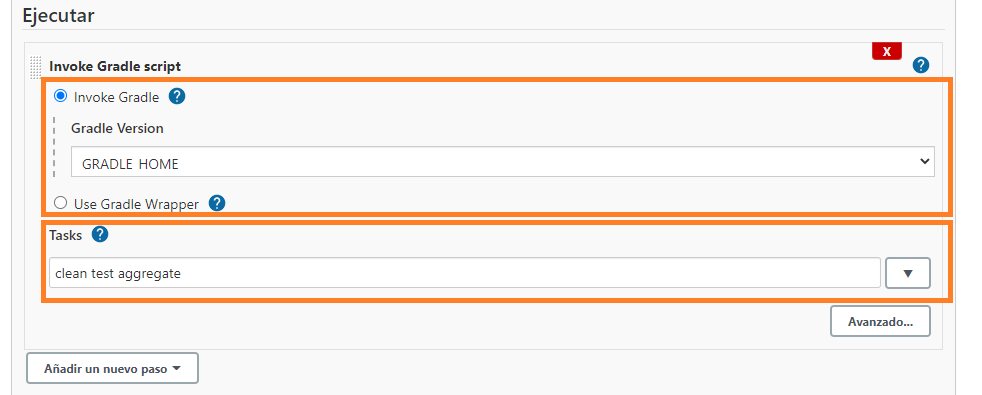


## Jenkins para proyectos Gradle

Para esta opción podremos utilizar la siguiente opción de comando **Invoke Gradle script**



Seleccionamos la opción **Invoke Gradle** y escogemos nuestro **GRADLE\_HOME** en la opción **Gradle Version** y finalizamos agregando el comando **clean test aggregate**

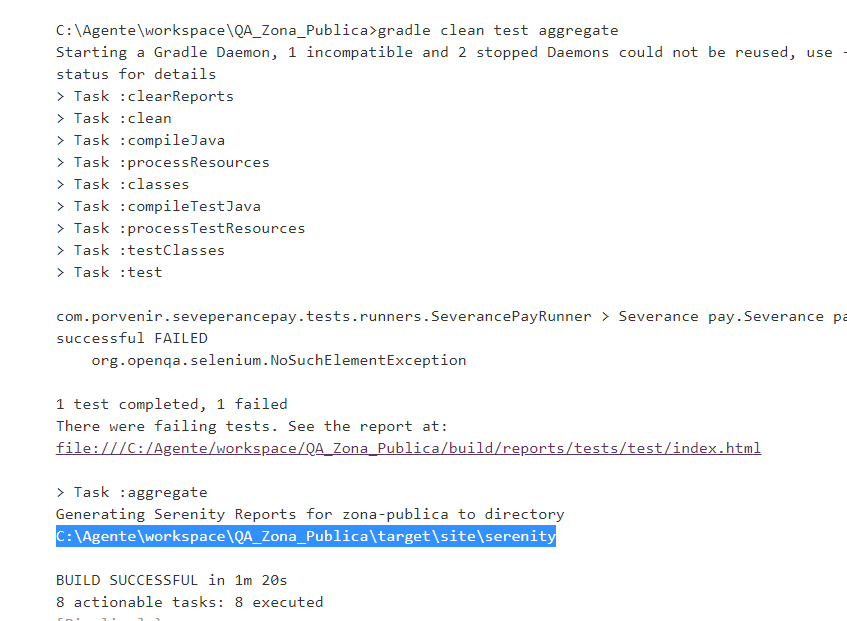


## Reportes

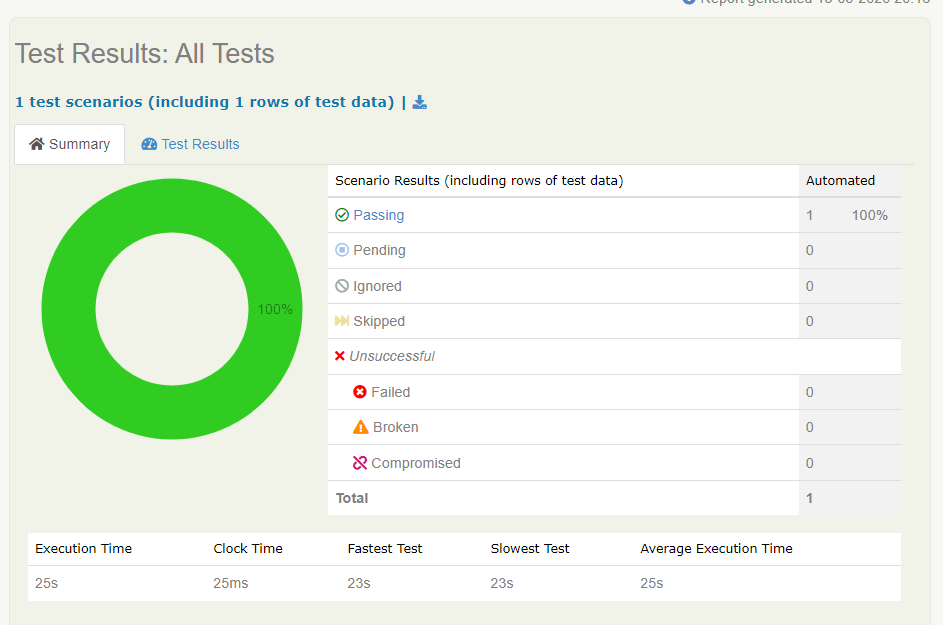
Los reportes es el resultado en estadísticas al final de nuestras ejecuciones, donde podemos observar el paso a paso realizado por nuestras pruebas automatizadas y tomas de evidencias.

Una vez ejecutado el pipeline se genera en el log en la siguiente ruta:

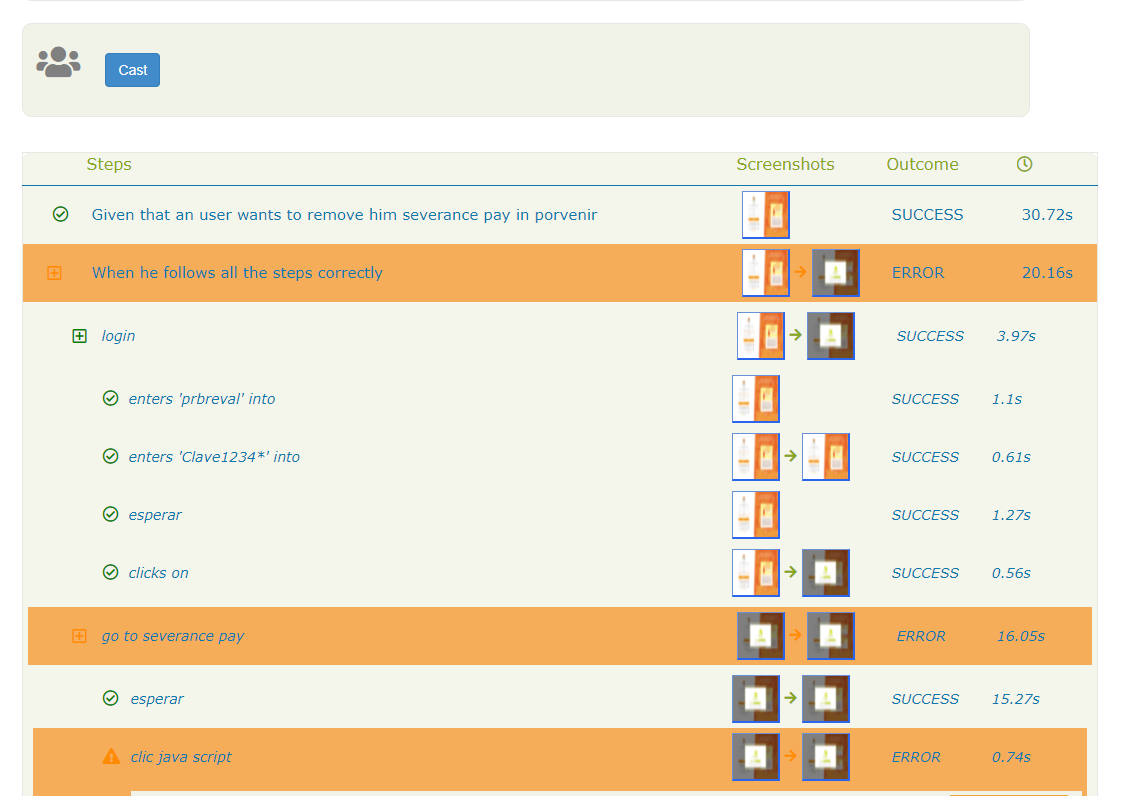
C:\Agente\workspace\QA\_Zona\_Publica\target\site\serenity



Y ser muestra de la siguiente forma:



Este contiene el resumen de las pruebas y el estado de las mismas, y contendrá las imágenes de lo eventos paso a paso como se comportó la automatización

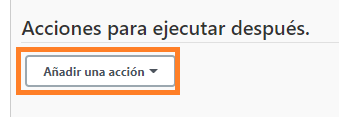
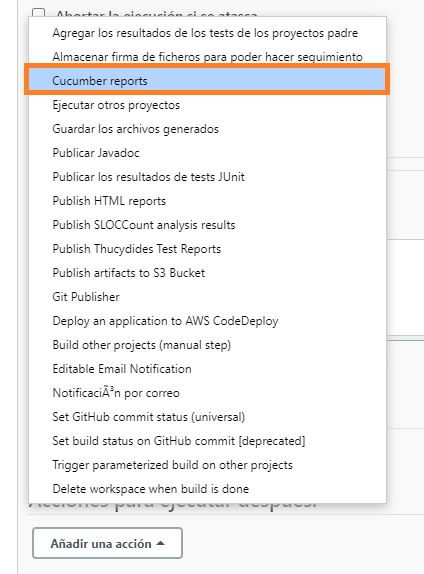


### Cucumber

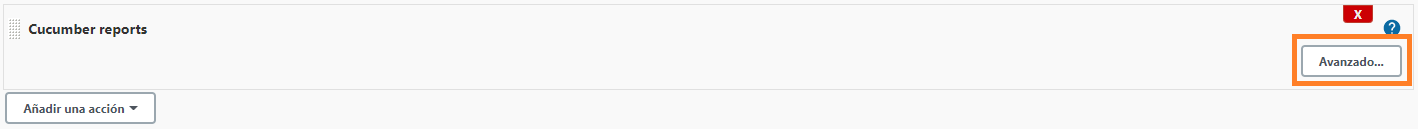
Si se requiere modificar el **Runner** de nuestro proyecto insertando en los **CucumberOptions** la siguiente línea de código.

**format = {"json:target/cucumber.json"}**

Volviendo a Jenkins en la configuración de nuestro Pipeline en la sección de **Acciones para ejecutar después** damos clic en **Añadir una acción** y escogemos la opción de **Cucumber Report**

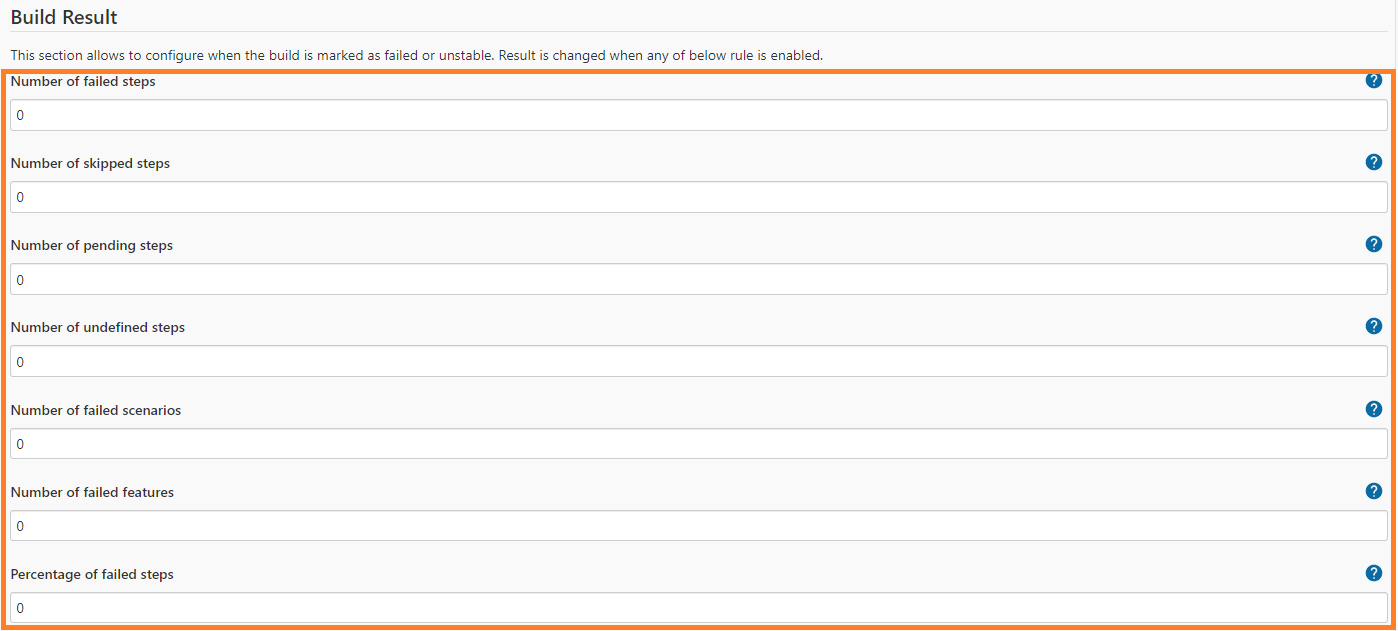
****

Damos clic en **Avanzado**



En el campo **Report title** dámos el nombre a nuestro reporte y en los campos de **Build Result** donde aparece el valor de **-1** lo reemplazamos por el valor de **0**



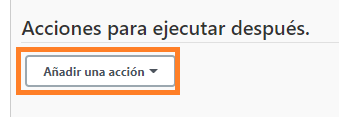


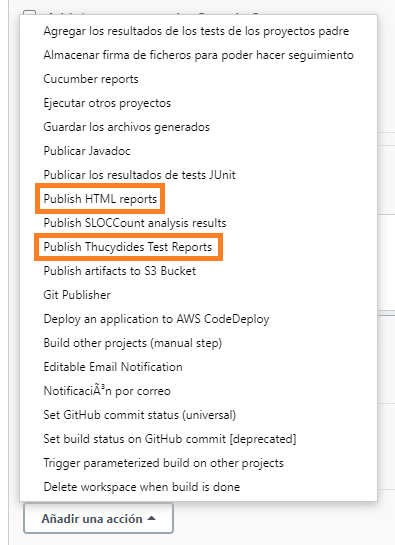
**Nota:** Para conocer más sobre el reporte de Cucumber con Jenkins, lo pueden hacer viendo los siguientes links.

* <https://www.lambdatest.com/blog/cucumber-with-jenkins-integration/>
* <https://www.jenkins.io/doc/pipeline/steps/cucumber-reports/>
* <https://plugins.jenkins.io/cucumber-reports/>

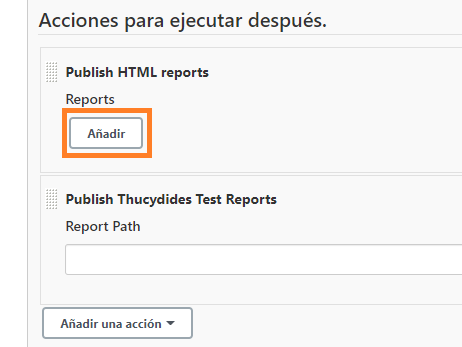
### Serenity BDD

En la sección de **Acciones para ejecutar después** damos clic en **Añadir una acción** y escogemos la opción de **Publish HTML reports** y **Publish Thucydides Test Reports**

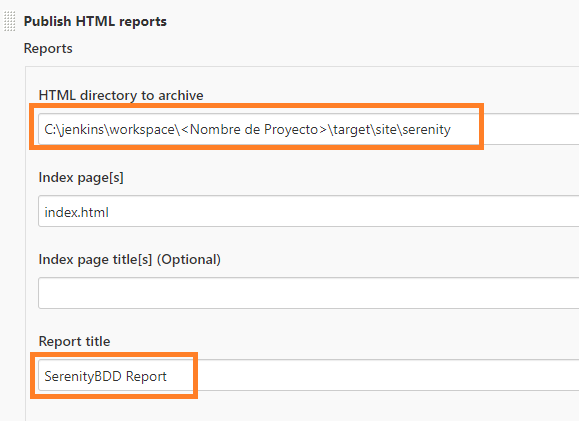
****



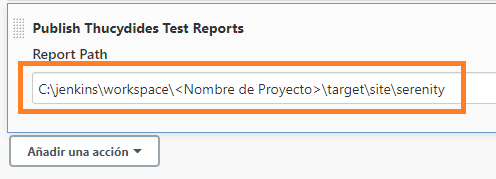
En **Publish HTML reports** damos clic en el botón **Añadir**

****

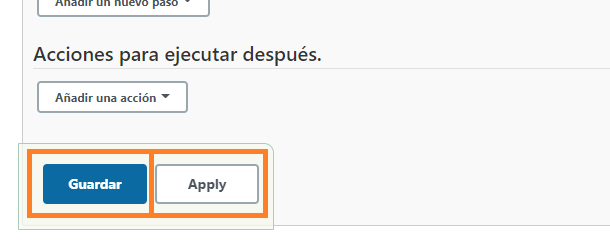
En **Publish HTML reports** ingresamos la dirección de donde se encuentra nuestro proyecto en nuestra maquina local y le agregamos al final **\target\site\serenity** en caso tal que sea en un servidor en la nube seria **/var/lib/jenkins/workspace/<Nombre de Proyecto> /target/site/serenity** y también agregamos en la opción **de Report title** el título que deseemos para nuestro reporte



En **Publish Thucydides Test Reports** solo ingresamos la ruta de nuestro proyecto que ingresamos en **Publish HTML reports**



Damos clic al botón **Apply** y luego al botón **Guardar**

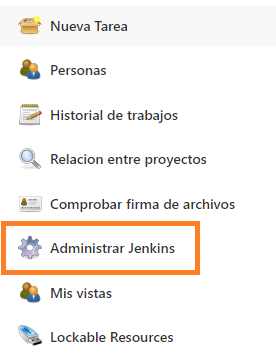
****

### Ejecutar comando para organizar el reporte de SerenityBDD

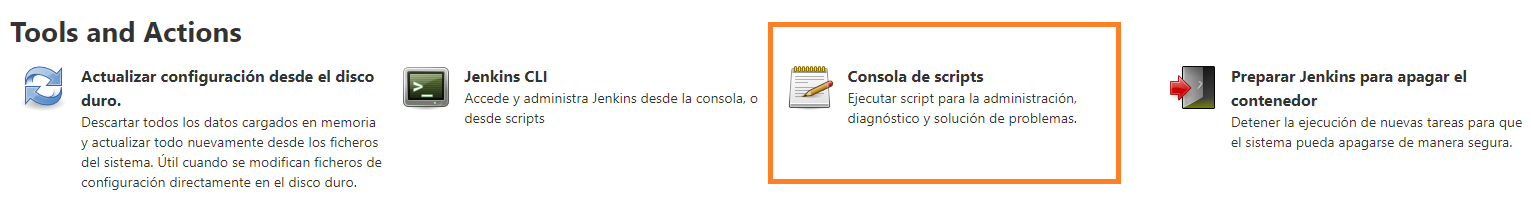
**Paso 1:** Damos clic en el logo de **Jenkins** en la parte superior izquierda



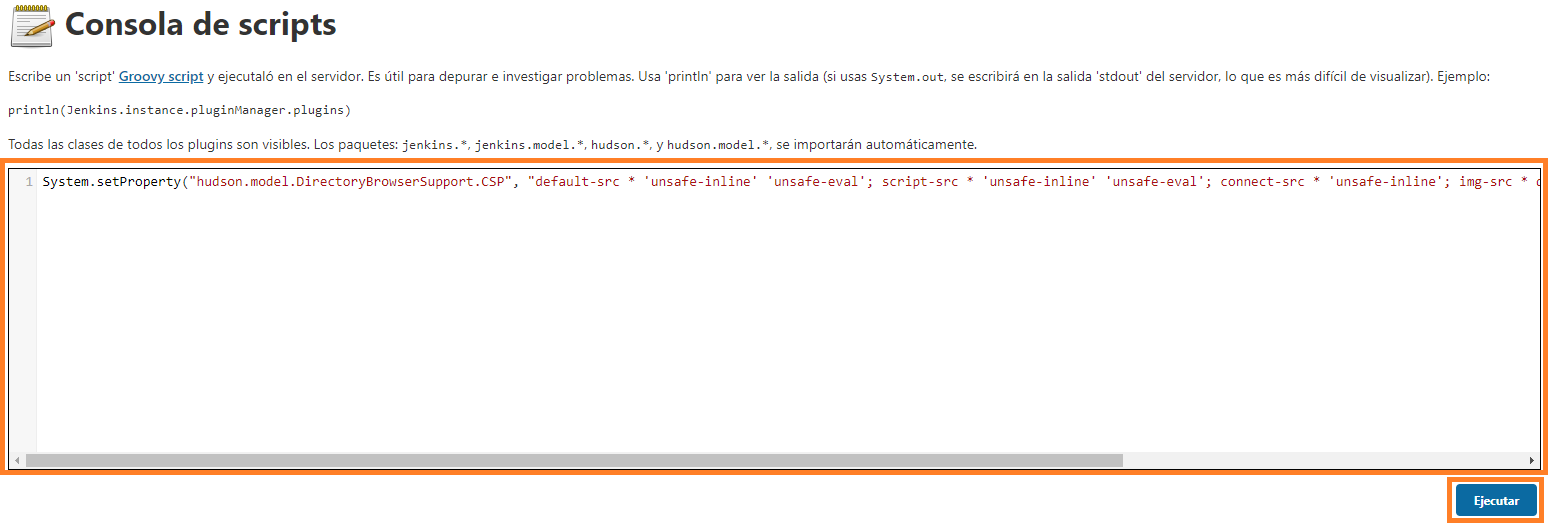
Damos clic en **Administrar Jenkins**



Bajamos hasta **Tools and Actions** y damos clic en **Consola de scripts**



**Paso 4:** Ingresamos en la **Consola de scripts** el siguiente comando y damos clic en **Ejecutar** System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "default-src \* 'unsafe-inline' 'unsafe-eval'; script-src \* 'unsafe-inline' 'unsafe-eval'; connect-src \* 'unsafe-inline'; img-src \* data: blob: 'unsafe-inline'; frame-src \*; style-src \* 'unsafe-inline';")

****

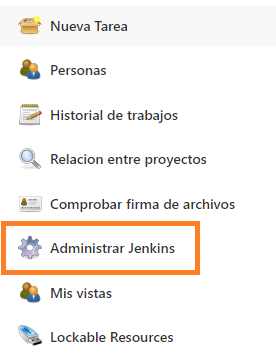
**Nota:** Este comando nos organiza el reporte de **SerenityBDD** de nuestro pipeline cuando termine la ejecución de la prueba

## Configuración de Agentes en Jenkins

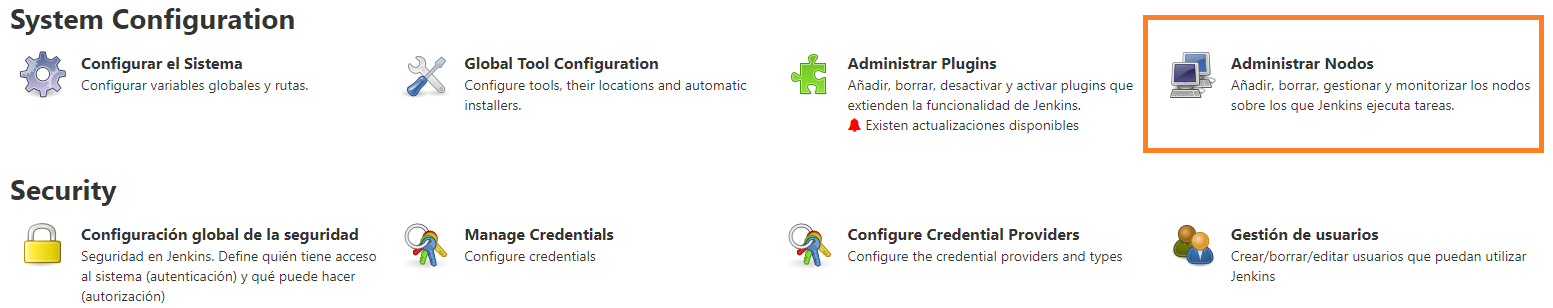
Los agentes de Jenkins son conectores que descargamos en nuestros computadores locales o servidores en la nube para conectar Jenkins con el dispositivo y realizar las pruebas en este.

Para configurar un agente debemos realizar los siguientes pasos:

Damos clic en **Administrar Jenkins**



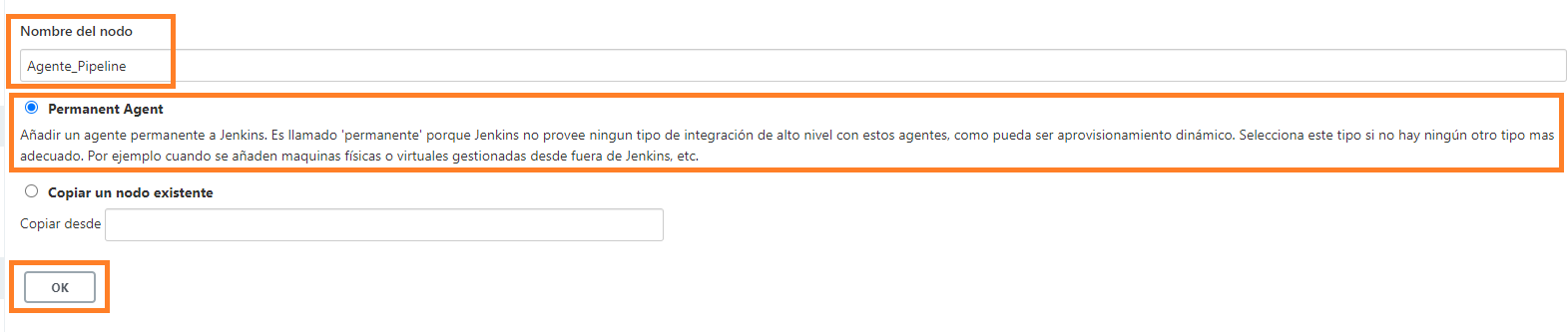
Nos ubicamos en **System Configuration** y damos clic en **Administrar Nodos**



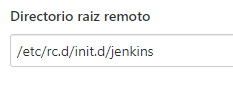
**Paso3:** Damos clic en **Nuevo Nodo**



Ingresamos el nombre a nuestro agente, damos clic en **Permanent Agent**  y damos clic en el botón **OK**

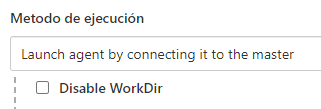
****

En el campo **Directorio raíz remoto** ingresamos la dirección de donde tenemos instalado **Jenkins** en este caso como es en un **servidor de AWS** ponemos la ruta donde se encuentra instalado Jenkins mediante **Putty** en caso de ser instalado en un computador local la ruta es donde se indique la instalación de Jenkins, normalmente queda en la ruta **C:\Program Files\Jenkins**

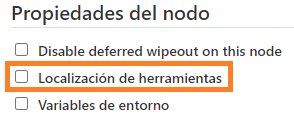


**Nota:** Normalmente cuando hacemos la instalación de Jenkins en un servidor en la nube la ruta donde queda instalado es la de la imagen sin importar el servidor

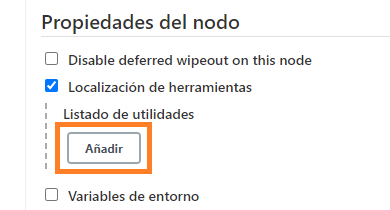
**Paso 6:** En el campo **Metodo de ejecución** escogemos la opción **Launxh agent by connecting it to the master**



**Paso 7:** En **Propiedades del nodo** seleccionamos la casilla **Localización de herramientas**

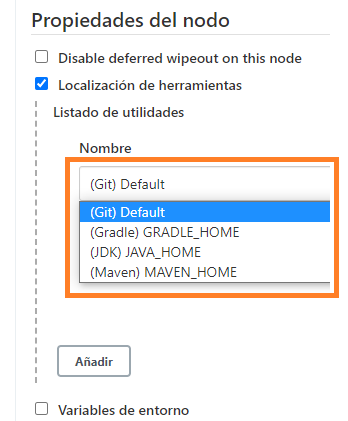
****

**Paso 8:** Damos clic en el botón **Añadir**

****

Nos aparecerá dos campos **Nombre** el cual contiene una lista de utilidades y **Directorio** un campo donde pondremos las direcciones de donde se encuentran las utilidades donde se ejecutara el agente

Al darle clic en el campo **Nombre** nos aparecerá una lista con las utilidades necesaria para una ejecución, al escoger una debemos ingresar la dirección de donde se encuentra instalada o ubicada la utilidad seleccionada. Para añadir cada utilidad debemos darle clic en el botón **Añadir** para que nos vuelva aparecer los campos para ingresarlas

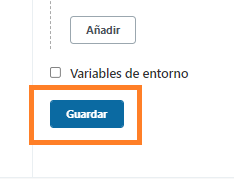


Al final debe verse como la siguiente imagen

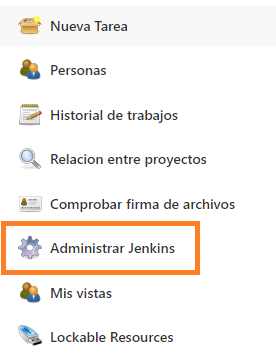


**Nota:** En este caso se está manejando para que el agente sea ejecutado en un computador local.

Damos clic en el botón **Guardar**



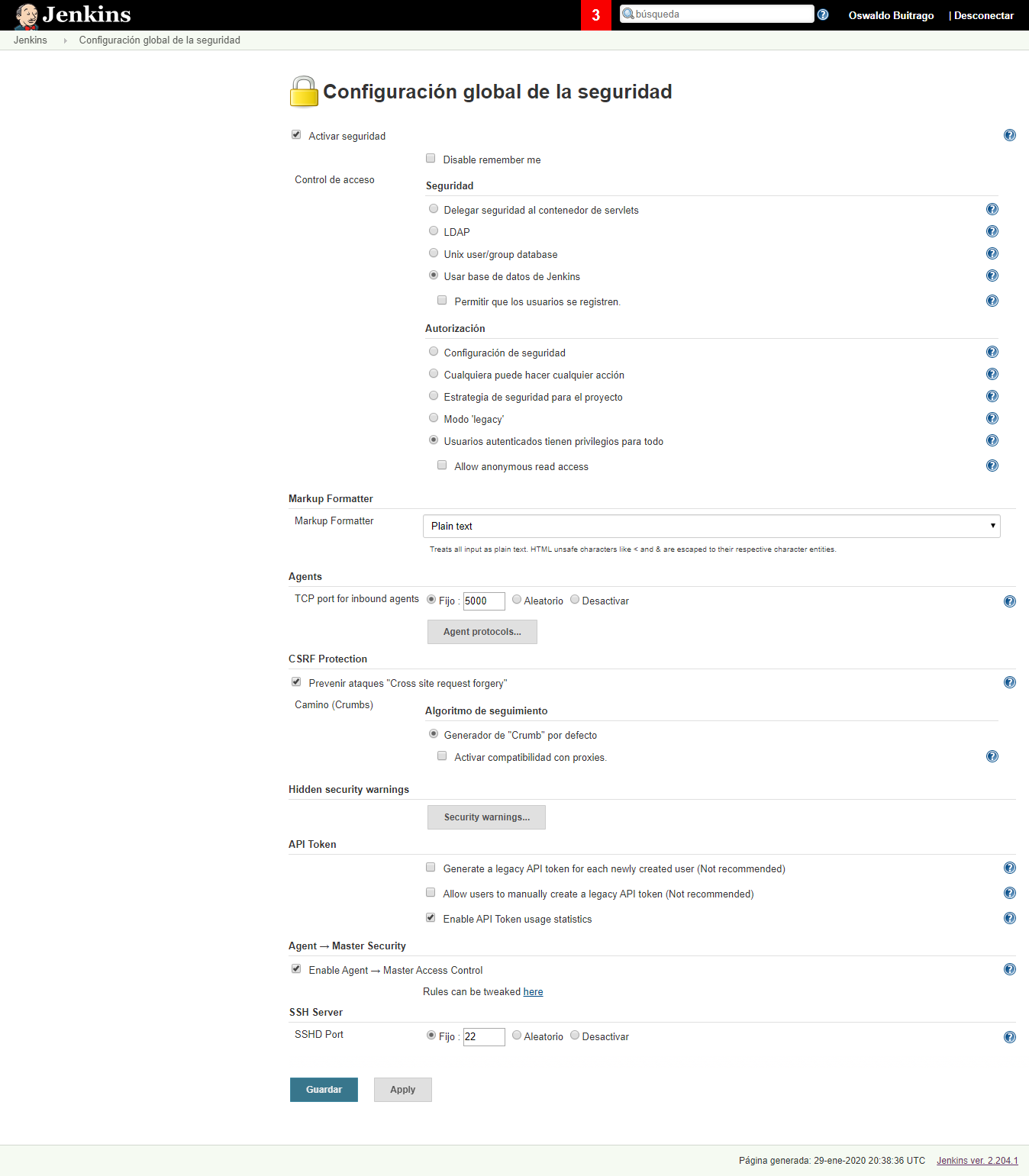
**Paso 13:** Damos clic en **Administrar Jenkins**



**Paso 14:** Damos clic en la opción **Configuración global de la seguridad**

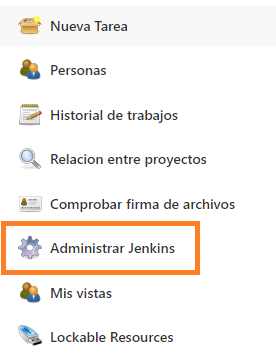


**Paso 15:** Cuando estemos en la **Configuración global de la seguridad** debemos configurar los campos tal y como se muestra en la siguiente imagen y damos clic al botón **Guardar** al final

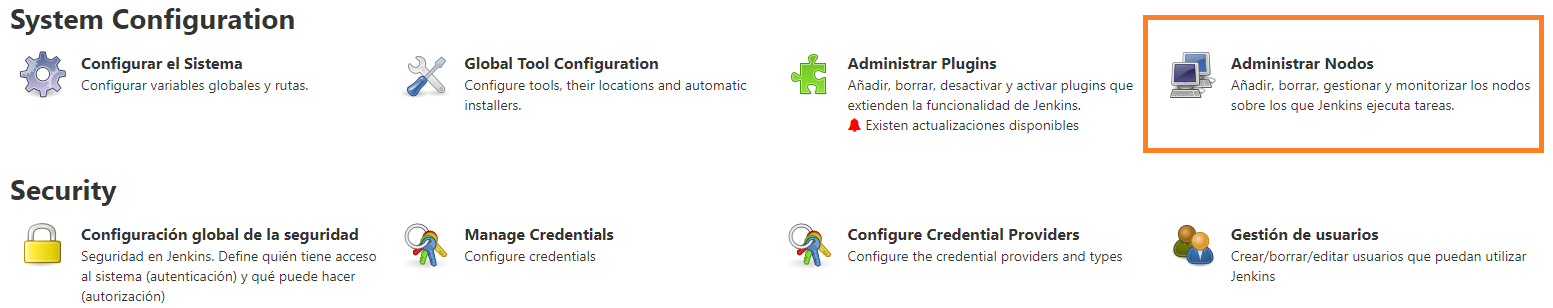


## Descargar Agente en dispositivo local

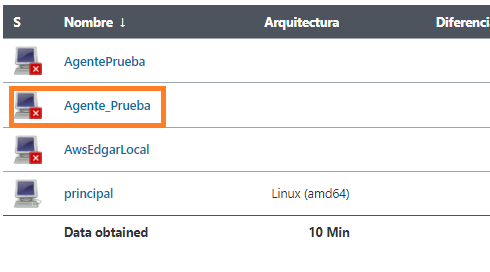
**Paso 1:** Damos clic en **Administrar Jenkins**



**Paso 2:** Nos ubicamos en **System Configuration** y damos clic en **Administrar Nodos**



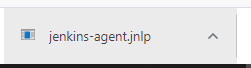
**Paso 3:** Damos clic al agente que creamos.



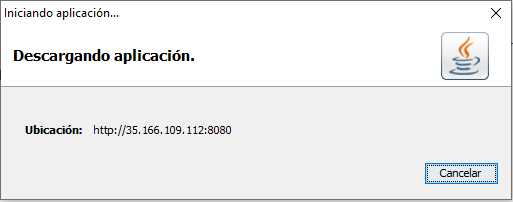
**Paso 4:** Damos clic al botón **Launch** para descargar el agente

****

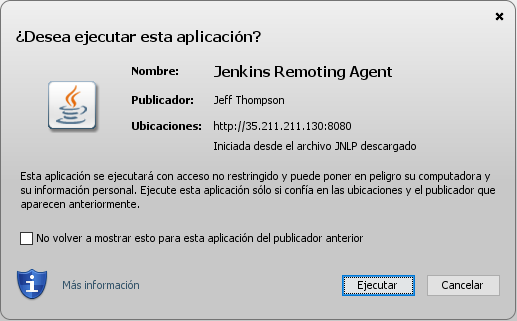
**Paso 5:** Damos clic en el ejecutable que descargamos



**Paso 6:** Dejamos que se inicialice



**Paso 7:** Damos clic en **Ejecutar**



**Paso 8:** Nos debe aparecer esta ventana que nos indica que ya está conectado nuestro agente



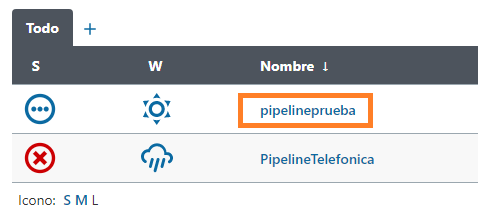
## Configuración de Agente con Pipeline

Una vez tenemos el Agente creado debemos asignarlo a un Pipeline para su ejecución. Para asociar un agente debemos realizar los siguientes pasos:

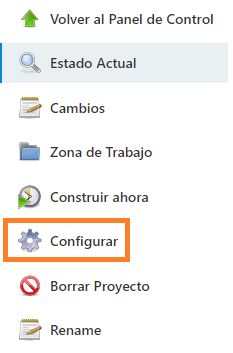
Hacer clic en el logo de **Jenkins** en la parte superior izquierda



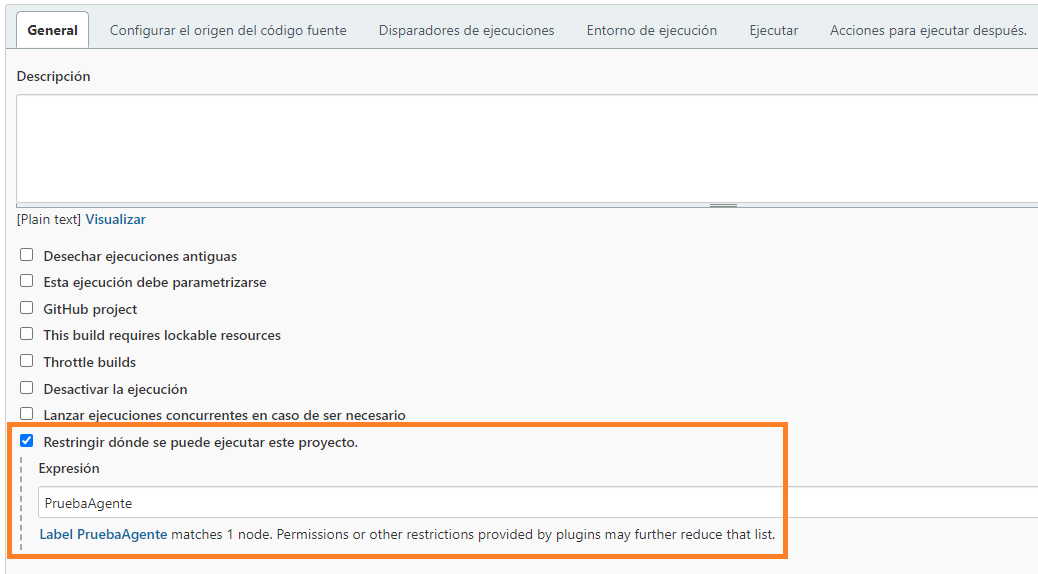
Hacer clic al pipeline que queremos asignarle el agente. (No es necesario escoger el Pipeline que aparece en la imagen si no el que desee)



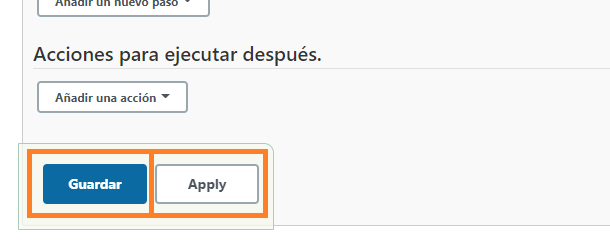
**Hacer** Damos clic en **Configurar**

****

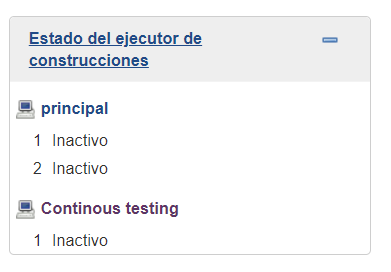
En la pestaña **General** seleccionamos la opción **Restringir donde se puede ejecutar este proyecto** y en el campo **Expresión** ingresamos el nombre que le hemos dado al agente

****

**Paso 5:** Damos clic al botón **Apply** y luego al botón **Guardar**

****

**Para el caso de Pipeline de Zona Publica el agente se debe ver en la siguiente forma:**



**Y después de decargar el .jar se debe ejecutar el siguiente comando desde la ruta de ubicación por medio del cmd.**

**java -jar agent.jar -jnlpUrl http://10.190.40.132:8080/computer/Continous%20testing/slave-agent.jnlp -secret 80106fab708b61a1cc02910b19be97182d8f7633e524f0141825812547d83700 -workDir "C:/Agente"**

## Ejecutar un Pipeline

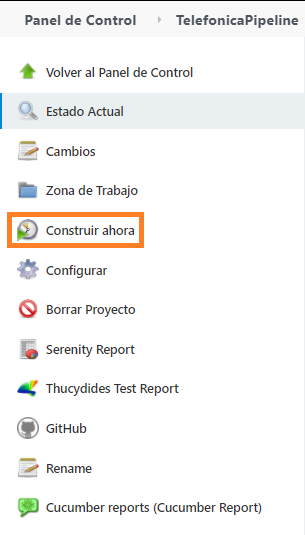
Para ejecutar un Pipeline ya sea en un servidor en la nube, en el mismo Jenkins o un computador local mediante un agente sin importar el modo se ejecuta de la misma forma. Para ejecutar un Pipeline debemos realizar los siguientes pasos:

**Paso 1:** Damos clic en el logo de **Jenkins** en la parte superior izquierda



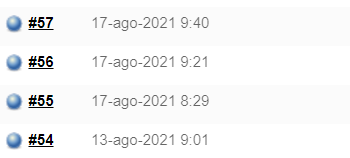
**Paso 2:** Damos clic al pipeline que queremos ejecutar. (No es necesario escoger el Pipeline que aparece en la imagen si no el que desee)

**Paso 3:** Damos clic en Construir ahora

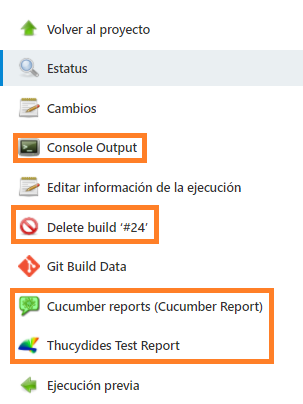
****

En este momento el pipeline está ejecutando, pero si quisiéramos ver el log de la ejecución debemos hacerlos los siguientes pasos:

**Paso 1:** En la parte inferior izquierda nos encontraremos con el listado de ejecuciones que hemos realizado y al darle clic a cualquiera podremos ver el resultado de las ejecuciones.



**Paso 2:** Una vez le demos clic a alguna ejecución podremos ver nuevas opciones al lado izquierdo, la cuales el **Console Output** permite ver en consola de Jenkins la ejecución de nuestro Pipeline, **Delete build** eliminar la ejecución el cual hemos ingresado y **Cucumber Report** y **Thucydides Test Report (solo si está configurado en el pipeline)** permiten ver los reportes de Cucumber y SerenityBDD

****

## PRINCIPALES BENEFICIOS DE LAS PRUEBAS CONTINUAS

A continuación, se menciona algunos de los principales beneficios:

* Integración con ecosistema DevOps-CT
* Validar continuamente las aplicaciones para identificar posibles fallos lo antes posible.
* Mejora el time-to-market
* Estandarización del Proceso de Pruebas y Automatización

## ASPECTOS A TENER EN CUENTA

* Realizar evaluación para identificar tipos de pruebas y herramientas de pruebas continuas.
* A nivel de pruebas de Software, identificar:
  + Los casos de prueba de las rutas críticas y si hay que crear o adaptar la automatización para que sea ejecutada desde el servidor de integración continua.
* Apoyar en el diseño del pipeline que se desea integrar al ecosistema devops.
* Analistas con conocimiento en pruebas continuas.
* Equipo elite para el apoyo en la preparación del ambiente para pruebas continuas.