

**MANUAL DE CI/CD DE WATSON ASSISTANT**

TECNOlOGÍA

**HISTORIAL DE REVISIONES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revisión** | **Fecha** | **Responsable** | **Comentarios** |
| V1 | 10/06/2022 | Diego Guerrón | Creación de documento |

**CONTENIDO**

[1. OBJETIVO 2](#_Toc108521410)

[2. REQUISITOS PREVIOS. 2](#_Toc108521411)

[3. Análisis de despliegue de la Solución 2](#_Toc108521412)

[4. Configuración de entornos de integración 3](#_Toc108521413)

[4.1. IBM Cloud 3](#_Toc108521414)

[4.1.1. Creación de Skills ambiente productivo 3](#_Toc108521415)

[4.1.2. Obtención de Skills details 4](#_Toc108521416)

[4.2. Creación repositorio GitHub 5](#_Toc108521417)

[4.2.1. Edición de config.properties 6](#_Toc108521418)

[4.2.2. Creación del job en Jenkins 6](#_Toc108521419)

[4.3. Integración Jenkins con Github 7](#_Toc108521420)

[4.4. Crear usuarios en Jenkins y Administración de permisos y Roles. 9](#_Toc108521421)

[4.5. Complemento Role Strategy Plugin 10](#_Toc108521422)

[4.6. ScriptApproval 13](#_Toc108521423)

[5. Flujo de ejecución de la solución (pipeline) 14](#_Toc108521424)

**MANUAL DE CI/CD DE WATSON ASSISTANT**

## OBJETIVO

Este documento tiene como objetivos principales:

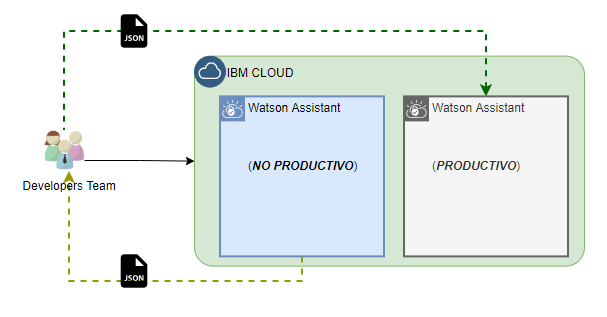
* Dar a conocer el proceso de integración y despliegue continuo de Watson Assistant

## REQUISITOS PREVIOS.

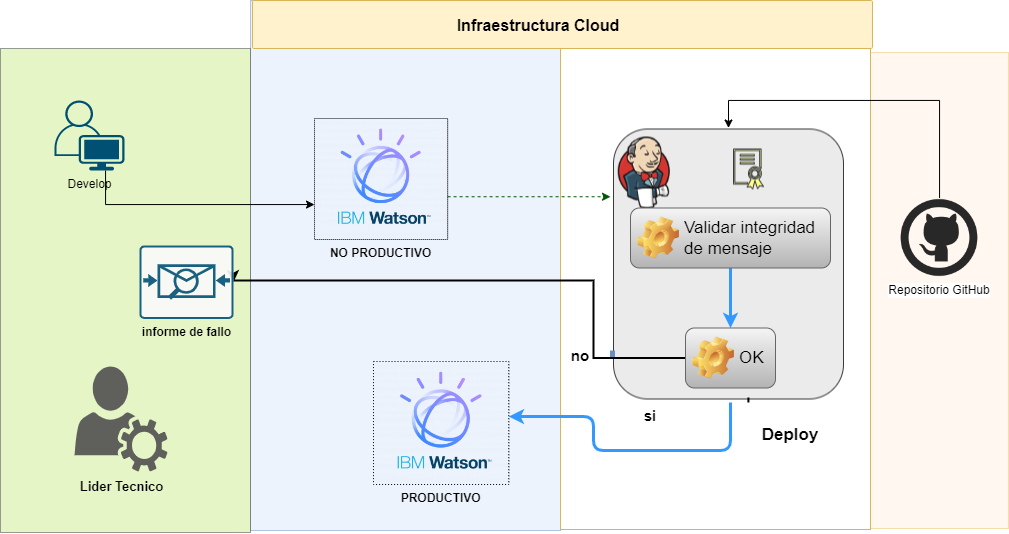
Para poder empezar necesitará una instancia de Watson Assistant en los diferentes ambientes (No Productivo y Productivo).

## Análisis de despliegue de la Solución

Situación actual del proceso de despliegue



Situación propuesta para el despliegue



## Configuración de entornos de integración

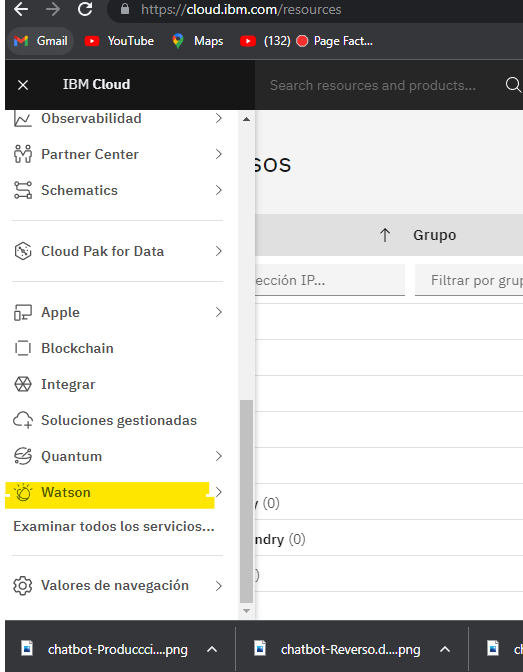
## IBM Cloud

Si no se dispone el Skills de producción creado dentro de Watson Assistant seguir los siguientes pasos: caso contrario se deberá obtener los valores del workspace [ver 4.1.2]

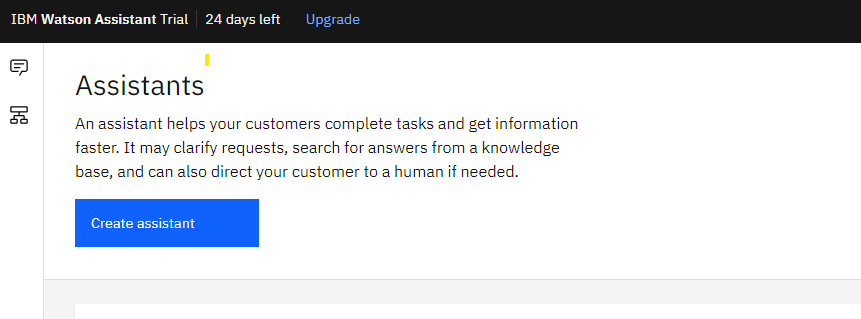
## Creación de Skills ambiente productivo

Este paso se lo debe realizar cuando no exista el skill en la instancia de Watson Assistant del en el ambiente destino.

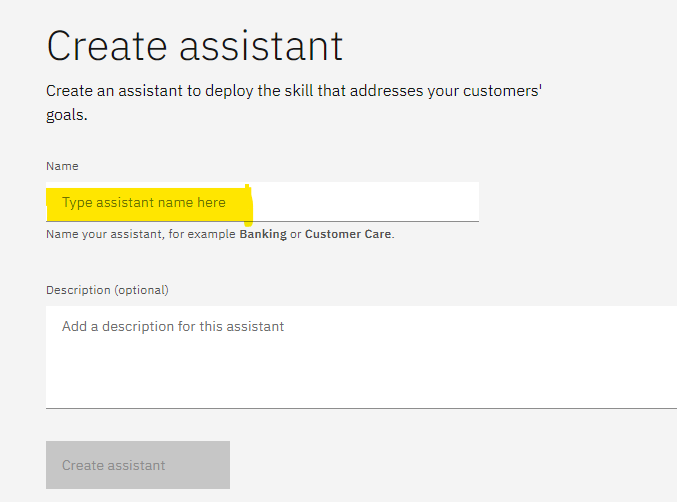
Pulse Iniciar Watson Assistant. Si se le solicita que inicie una sesión, proporcione sus credenciales de IBM Cloud.

Se abre una nueva ventana de navegador y se muestra Watson Assistant.



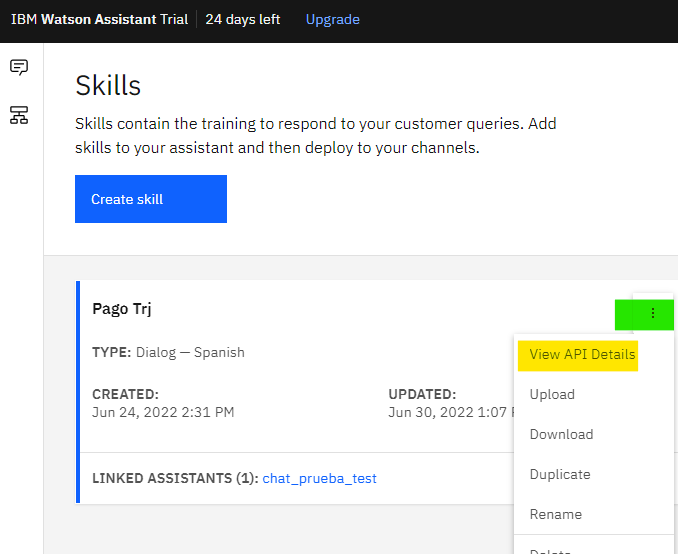
En la cual se procede a ingresar la información solicitada y presionar continuar



## Obtención de Skills details

Se debe proceder a la obtención de la información del workspace que se genera tanto en la instancia de QA como en la de producción.

Seleccionar el Skills a versionar



Copiar los datos como: ***Skill ID, Legacy v1 workspace URL: API key:*** *De cada ambiente*

***QA****. - ambiente origen*

***Producción****. - ambiente destino.*

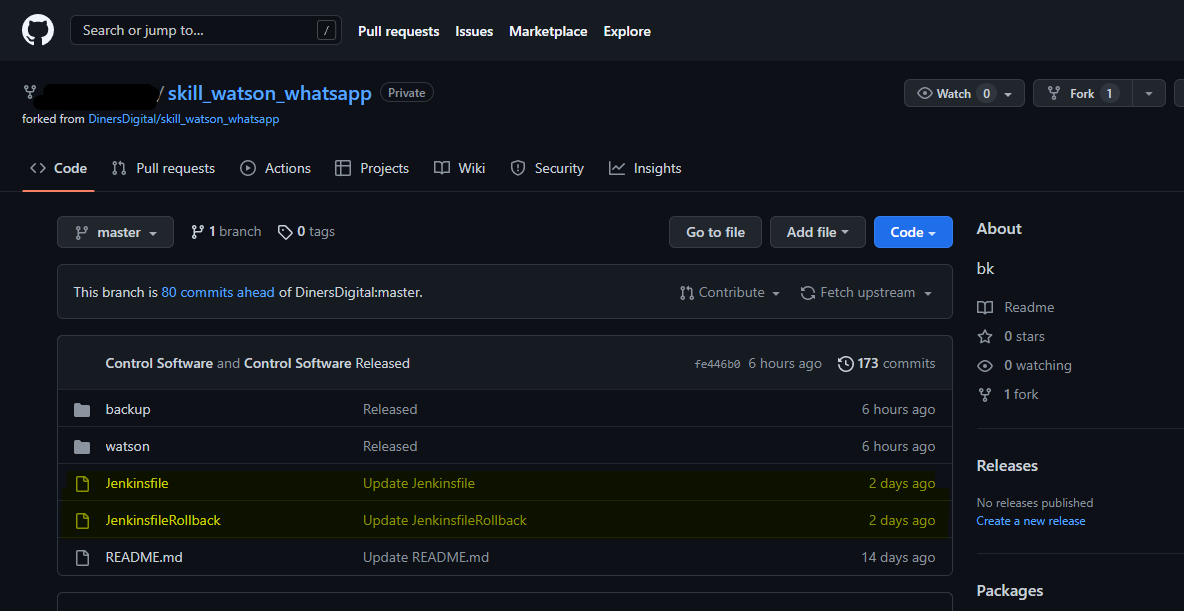


Guardar dicha información para ser configurada en el archivo config.properties

## Creación repositorio GitHub

Se debe solicitar la creación del proyecto a control de software dentro del repositorio GitHub en base al proyecto **skill\_watson\_whatsapp.** Se puede realizar un clone local del proyecto o a su vez se puede editar online.

Si no existen los archivos **jenkinsfile.json** y **jenkinsfileRollback.json**, **config.properties** proceder a realizar una copia de dichos archivos del proyecto **skill\_watson\_whatsapp** que se encuentra en github:



## Edición de config.properties

Se procede a modificar el archivo con la información de los worksapace obtenidos en el paso **4.1.2**

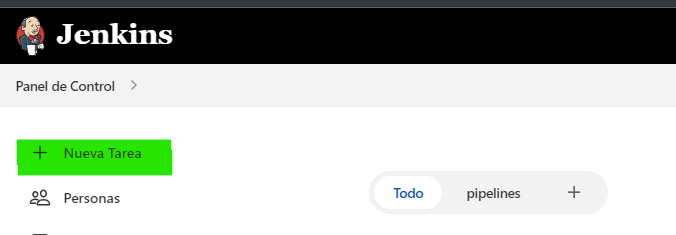


|  |  |
| --- | --- |
| **release.file** | Nombre que se le da al archivo.json a ser versionado |
| **branch.name** | Branch en el que se va ha trabajar en el caso es develop |
| **api.key.prod** | Código de Api obtenido en el workspace de Watson Assistant de PROD. |
| **url.workspace.prod** | Url del workspace de Watson Assistant de PROD |
| **api.key.test** | Código de Api obtenido en el workspace de Watson Assistant de TEST |
| **url.workspace.test** | Url del workspace de Watson Assistant de TEST |

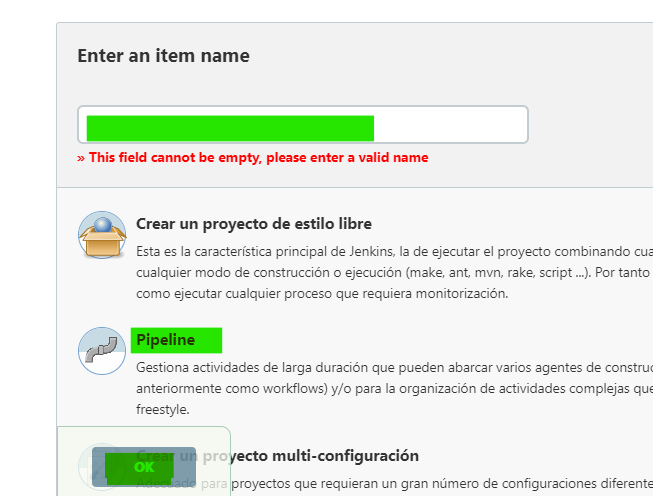
## Creación del job en Jenkins

Una vez tenemos el repositorio Git, para poder avanzar tenemos que definir y configurar el pipeline en Jenkins tanto de ejecución como de reverso.

* Accedemos a Jenkins y pulsamos sobre el enlace nueva tarea en el panel lateral.

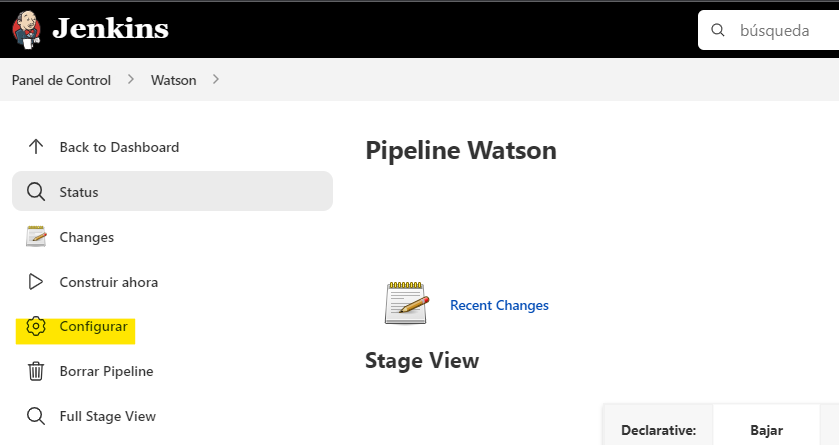
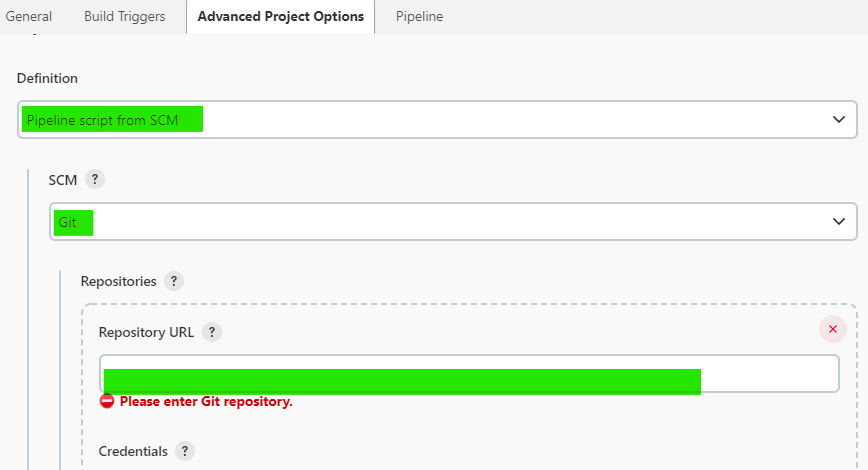


* Especificamos un nombre para el nuevo job; por ejemplo, (**skill\_watson\_whatsapp**. **skill\_watson\_whatsapp**\_**reverso**) (Depende del proyecto)
* Seleccionamos Pipeline en la lista de tipos de proyectos.
* Pulsamos Ok para crear el job.
* No configuramos nada y pulsamos **Save**.



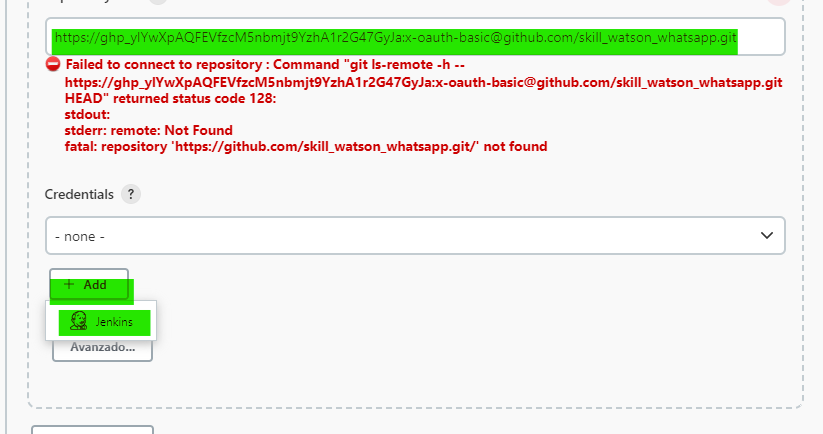
## Integración Jenkins con Github

Configuramos las credenciales que necesitamos para conectar con Git

Desde la página principal de Jenkins, pulsamos sobre Credentials en el panel lateral.

Pulsamos sobre el almacén global de credencials (llamado Jenkins).



Pulsamos sobre Global credentials (unrestricted) y seleccionamos la opción Add credentials.

En el desplegable para el tipo de credencial, seleccionamos *Username with password* (es la opción por defecto).

En el ***scope***, seleccionamos *Global*.

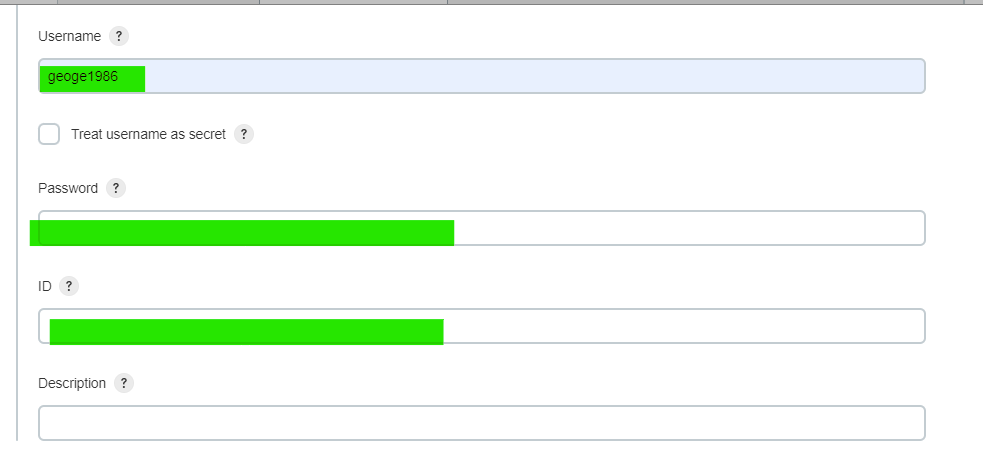


En el campo ***Username***, especificamos el usuario con permisos de acceso al repositorio donde se encuentra el código. En nuestro caso, el usuario operador.

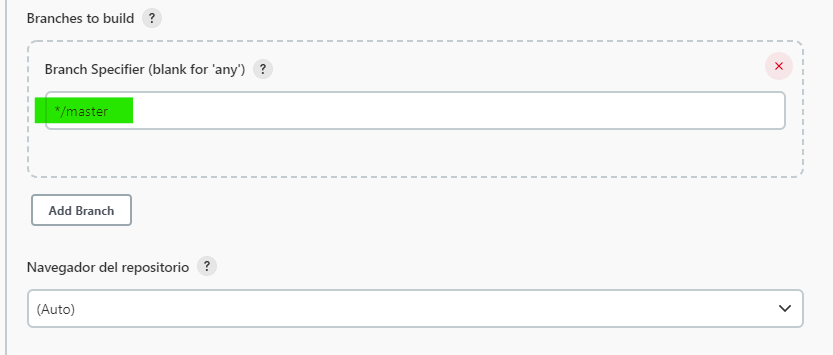
En el campo ***Password***, introducimos la contraseña del usuario.

Podemos dejar el campo ***ID*** en blanco; Jenkins genera un ID aleatorio. Para facilitarnos la tarea de identificar las credenciales guardadas -para su uso en scripts, por ejemplo, podemos especificar un *ID* específico (siempre que sea **único** en Jenkins).

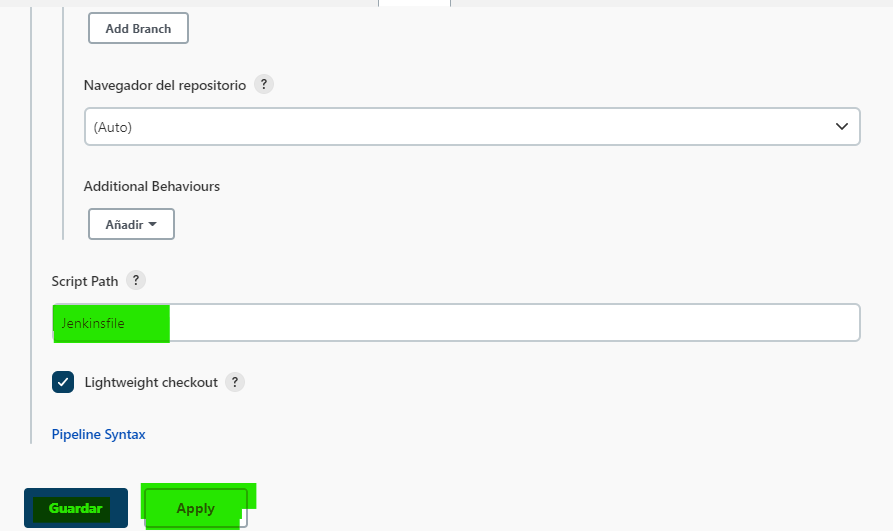
En el campo ***Description*** podemos dar una descripción detallada de las credenciales, normas de uso, etc.



En el campo Branch seleccionar \*/master



Seguidamente ingresar en script path el nombre del archivo de configuración del pipeline esto según el flujo ingresar jenkinsfile o jenkinsfileRollback



Tras especificar los campos necesarios para definir las credenciales, pulsamos el botón Ok. Esta descripción se muestra junto al nombre de usuario de las credenciales guardadas.

## Crear usuarios en Jenkins y Administración de permisos y Roles.

Normalmente en una organización, hay varios equipos separados para administrar y ejecutar trabajos en Jenkins, pero administrar una gran cantidad de usuarios y asignarles roles puede resultar engorroso.

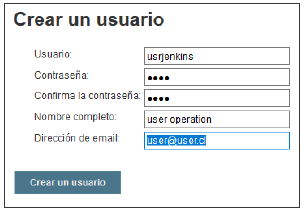
Por defecto, Jenkins viene con opciones básicas de creación de usuarios, se pueden crear varios usuarios, pero solo se pueden asignar los mismos roles y privilegios globales.

Para resolver esto, existe un complemento llamado Role Strategy Plugin permite asignar diferentes roles y privilegios a diferentes usuarios.

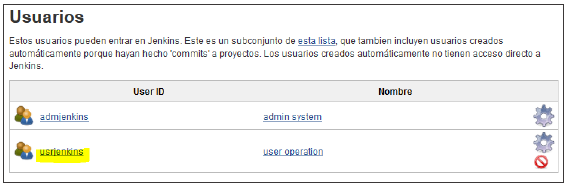
En consola de administración clic sobre administrar Jenkins, luego gestión de usuarios y finalmente crear usuario



Ingresar datos solicitados; nombre, contraseña y correo, finalizar presionando el botón crear usuario.



El usuario es creado



## Complemento Role Strategy Plugin

Ir a administrar Jenkins, clic en administrar plugins.



En sección todos los complementos, buscar por la palabra “role” Seleccionar Role-based Authorization Strategy, clic en “Instalar sin reiniciar”.

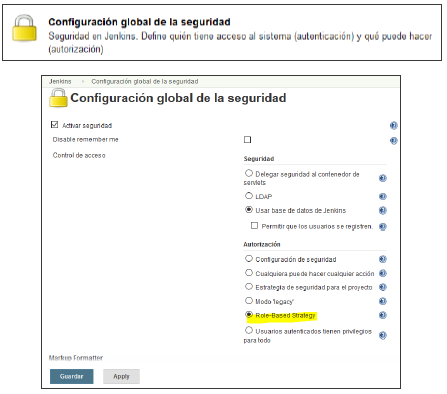


Una vez que el complemento está instalado, se desplegara “Actualizado”



Clic en volver a la página anterior.

Ir a Administrar Jenkins – Configuración Global de la seguridad – bajo Autorización, seleccionar Role Based Strategy, finalizar con clic en Guardar.

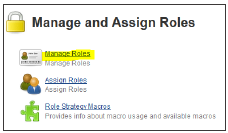


**Crear roles**

En este paso, se verán roles para un usuario o un grupo de usuarios.

Clic en Administrar Jenkins, Seleccionar Manage and Assign Roles.





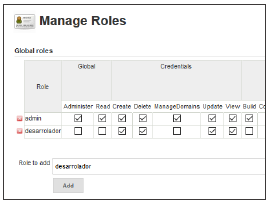
Crear un nuevo rol llamado **aprobadorRol**.

Escribir **aprobadorRol** bajo rol.

Clic en agregar.

Ahora seleccionar los permisos que se requieren para el rol de autorizador.

Clic en guardar.

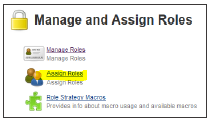


**Asignar un rol**

Ahora que ha sido creado el rol, este se asignará a un usuario (o grupo de usuarios) especifico.

Clic en administrar Jenkins

Seleccionar Manage and Assign Roles

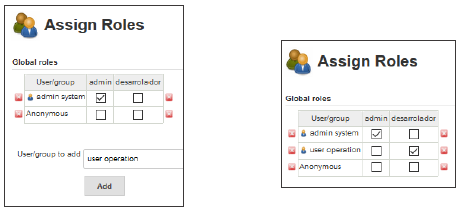


Agregaremos el nuevo rol **aprobadorRol** al usuario user operation.

Buscar el usuario creado y agregar.

Seleccionar el rol desarrollador del checkbox y asignar a usuario.

Clic en guardar.

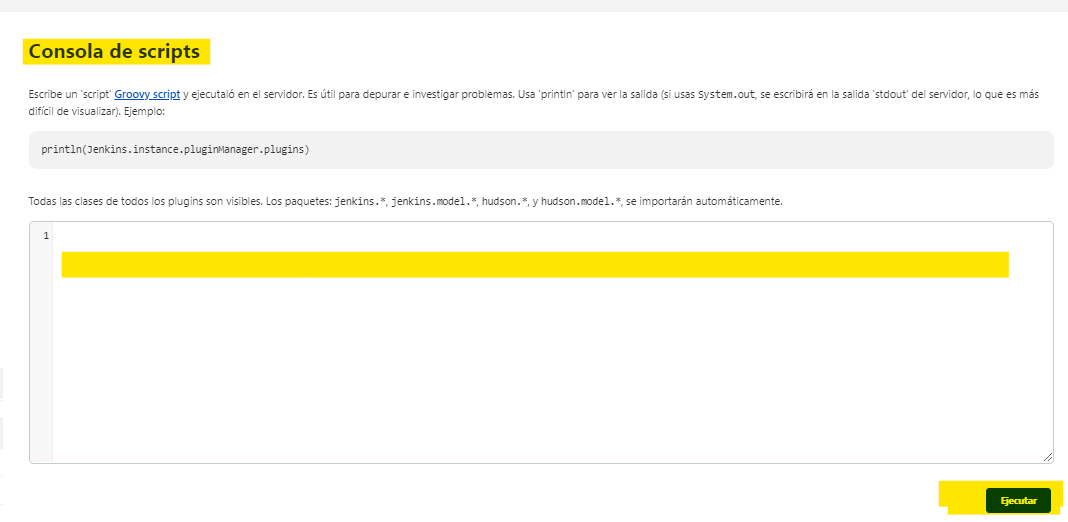


Finalmente, conectarse con nuevo usuario y validar permisos y vistas.

## ScriptApproval

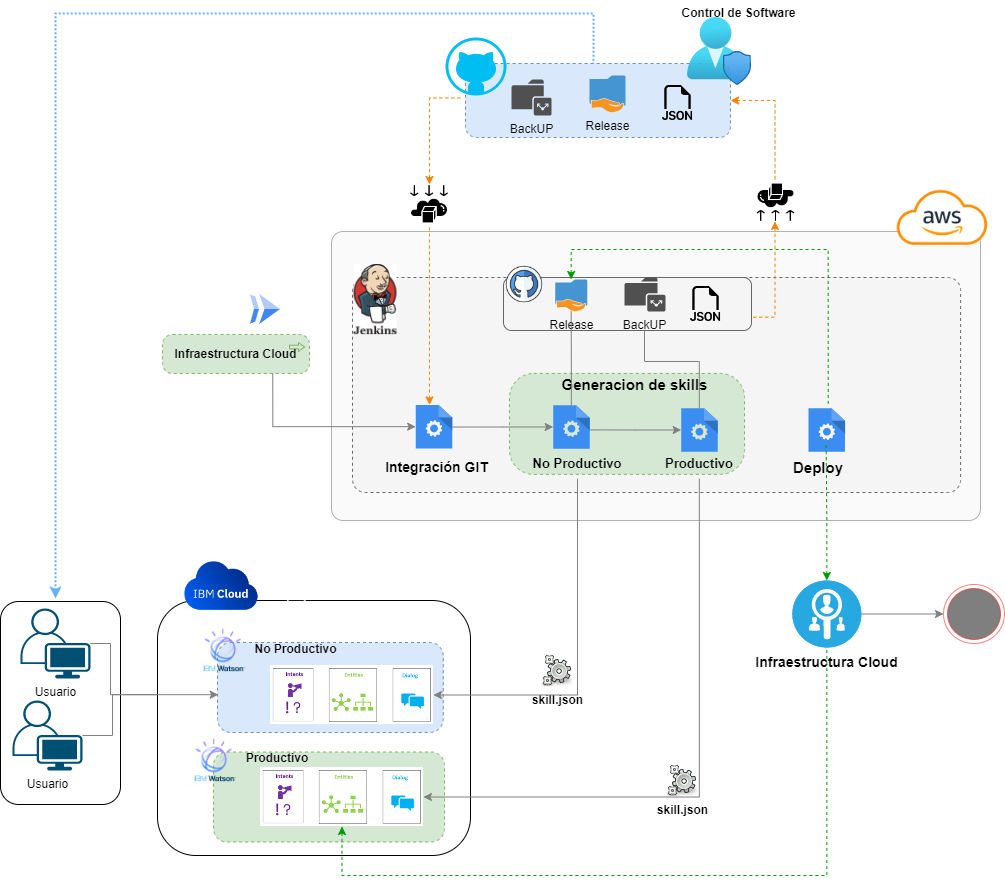
Es necesario aprobar la ejecución de varios complementos que se hacen uso dentro del archivo jenkinsfile para realizar este proceso ingresar a la consola de scripts y se ejecutará las siguientes sentencias.

***http://<servidorJenkins>:<puerto>/script***



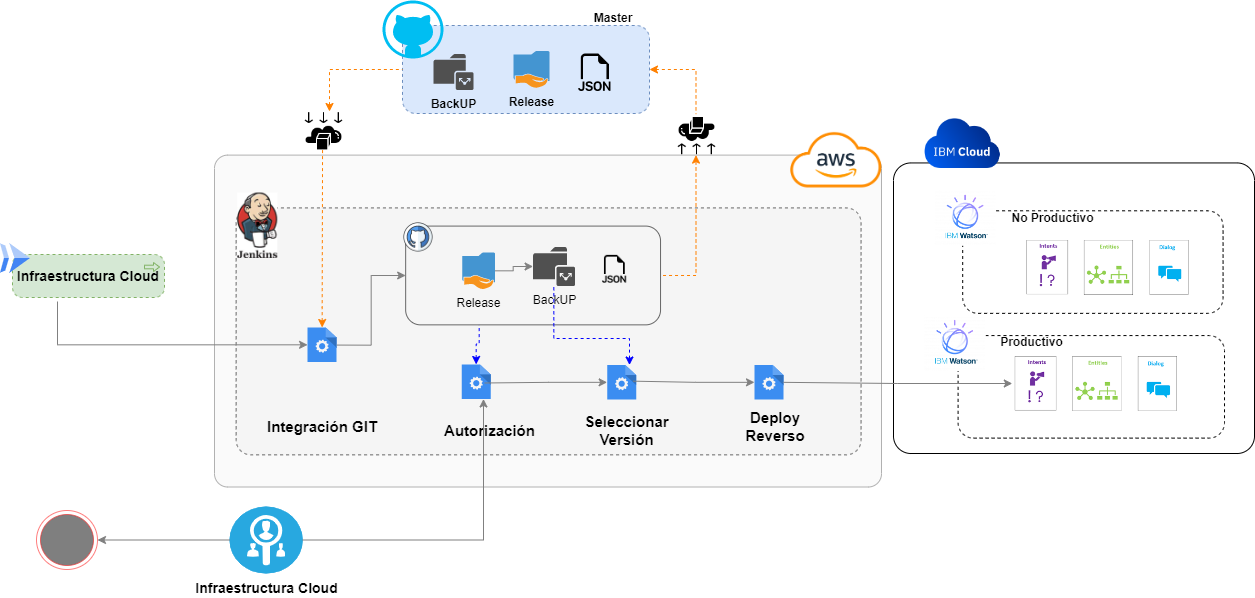
|  |
| --- |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('staticMethod jenkins.model.Jenkins getInstance') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('new java.io.File java.lang.String') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('method java.io.File getName') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('staticMethod org.codehaus.groovy.runtime.DefaultGroovyMethods reverse java.util.List boolean') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('staticMethod org.codehaus.groovy.runtime.DefaultGroovyMethods eachFileRecurse java.io.File groovy.io.FileType groovy.lang.Closure') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('staticField groovy.io.FileType FILES') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('method jenkins.model.Jenkins getAuthorizationStrategy') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('method com.michelin.cio.hudson.plugins.rolestrategy.RoleMap getSidsForRole java.lang.String') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('method com.michelin.cio.hudson.plugins.rolestrategy.RoleBasedAuthorizationStrategy getRoleMap com.synopsys.arc.jenkins.plugins.rolestrategy.RoleType') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('staticField com.synopsys.arc.jenkins.plugins.rolestrategy.RoleType Global') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('staticMethod org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval get') |
| org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval.get().approveSignature('method org.jenkinsci.plugins.scriptsecurity.scripts.ScriptApproval approveSignature java.lang.String') |

## Flujo de ejecución de la solución (pipeline)



**Roles y responsabilidades flujo de versionamiento**

|  |  |  |  |
| --- | --- | --- | --- |
|  | DESARROLLO | REPOSITORIO GIT | JENKINS |
| **Develop CDI** | * Generar el cambio en la instancia de QA de Watson Assistant * Realizar los cambios necesarios en el archivo **config.properties** en base a la información revisada en el punto 4.2.1 | * Solicitar la creación de repositorio del proyecto en github al control de software * Generar el push a la rama de develop de los cambios sobre el archivo de configuraciones |  |
| **Control de Software** |  | Crear repositorio generado solicitado en base al proyecto **skill\_watson\_whatsapp**. |  |
| **Infraestructura Cloud** |  | En conjunto con Develop se debe modificar el archivo **config.properties** con la información de los workspace de producción. Según el lo revisado en el punto 4.2.1 | * Creación de Skills en instancia de ibm cloud productivo si es necesario * Creación y configuración del job en Jenkins (flujo versionamiento, flujo de reverso) * Ejecución y aprobación de pipeline de versionamiento |



**Roles y responsabilidades flujo de reverso**

|  |  |
| --- | --- |
|  | JENKINS |
| **Develop CDI** | Soporte para selección de versión a reversar |
| **Infraestructura Cloud** | Ejecución del pipeline de reversos según lo indique el equipo de Develop CDI |