

UD3 - Despliegue y Automatización con Jenkins

Cristóbal Suárez Abad

2º ASIR

Instrucciones Generales

Entregar:

- Un documento PDF con las capturas de pantalla que evidencien el funcionamiento de cada apartado
 - Los ficheros de configuración utilizados (Jenkinsfile, scripts, etc.).
 - **Muestra capturas y explicaciones de la configuración realizada durante todo el proceso, además demuestra que todo funciona como esperamos.**
-

Parte 1: Gestión de Usuarios y Seguridad (1,5 puntos)

Objetivo: Aplicar el modelo de seguridad basado en roles.

1. Cree un primer usuario administrador con el nombre **admin_asir** y una contraseña segura.

Creamos el usuario: Administrar Jenkins → Users → Crear usuario.

Create User

Username

Password

Confirm password

Full name

E-mail address



admin_asir

Administrador ASIR Examen



Le damos privilegios de Administrador: **Administrar Jenkins → Security → Authentication:**

Authorization

Matrix-based security

User/group	Overall			CI		
	Administrator	Read	Create	Delete	Read	Create
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Administrador	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Administrador ASIR Examen	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Add user... Add group... ?

Cerramos la sesión actual y entramos con el nuevo Administrador:

The sidebar menu includes the following items:

- Theme (Light)
- My Views
- Account
- Appearance
- Preferences
- Security
- Experiments
- Credentials
- Sign out

2. Creación de Usuarios:

- Cree dos nuevos usuarios: desarrollador1 y jefe_proyecto.

- desarrollador1

Create User

Username

Password

Confirm password

Full name

E-mail address

Create User

- jefe_proyecto

Create User

Username

Password

Confirm password

Full name

E-mail address

Create User

User ID	Name	Action
 admin	Administrador	 
 admin_asir	Administrador ASIR Examen	 
 desarrollador1	Desarrollador 01	 
 jefe_proyecto	Jefe Proyecto	 

3. Matriz de Autorización:

- Configure la seguridad global para que:
 - **Usuarios anónimos:** No tengan acceso de lectura ni ejecución.
 - **jefe_proyecto:** Tenga control total (Admin) sobre Jenkins.
 - **desarrollador1:** Solo tenga permisos para **Ver** (Read) y **Construir** (Build) tareas, pero NO para configurar Jenkins ni gestionar plugins.

Authorization

Matrix-based security

	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics	
User/group									View
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> 
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> 
Administrador	<input checked="" type="checkbox"/>								<input checked="" type="checkbox"/> 
Administrador ASIR Examen	<input checked="" type="checkbox"/>								<input checked="" type="checkbox"/> 
Desarrollador 01	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> 
Jefe Proyecto	<input checked="" type="checkbox"/>								<input checked="" type="checkbox"/> 

Add user... Add group... ?

Parte 2: Freestyle Jobs y Shell Scripting (1,5 puntos)

Objetivo: Crear tareas básicas de automatización.

1. Job "Sistema_Info":

New Item

Enter an item name

Sistema_Info

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one S steps like archiving artifacts and sending email notifications.

- Añade un script de shell que realice lo siguiente:
 - Muestre la fecha y hora actual.
 - Liste los archivos del directorio de trabajo (workspace).
 - Muestre el usuario del sistema que está ejecutando el job (`whoami`).

Usamos comandos de Linux:

“date” nos da la hora.

“ls” lista objetos en un directorio.

“`whoami`” indica el usuario que tiene la sesión.

Build Steps

Automate your build process with ordered tasks like c

≡ Execute shell ?

Command

See [the list of available environment variables](#)

```
date
ls -l
whoami
```

 **Console Output**

```
Started by user Administrador ASIR Examen
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/Sistema_Info
[Sistema_Info] $ /bin/sh -xe /tmp/jenkins15838757071345453215.sh
+ date
Fri Dec 12 08:49:32 UTC 2025
+ ls -l
total 0
+ whoami
jenkins
Finished: SUCCESS
```

2. Parámetros:

- Modifique el job para que acepte un **parámetro de texto** llamado **ENTORNO** (con valor por defecto "TEST").
- Haga que el script imprima: "*Desplegando en el entorno: \$ENTORNO*".

Tenemos que crearlo:

Podemos crear variables de entorno personalizadas:

Administrar Jenkins → System → Propiedades Globales: Marca la casilla de “Variables de entorno” → Añadir.

Global properties

Disable deferred wipeout on this node ?

Disk Space Monitoring Thresholds

Environment variables

List of variables ?

Name	ENTORNO
Value	TEST

+ Add

Tool Locations

Ahora incluimos un “echo” con la sentencia que se especifica en el enunciado.

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Execute shell ?

Command

See the [list of available environment variables](#)

```
date
ls -l
whoami
echo "Desplegando en el entorno: $ENTORNO"
```

Console Output

```
Started by user Administrador ASIR Examen
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/Sistema_Info
[Sistema_Info] $ /bin/sh -xe /tmp/jenkins10331023766246928213.sh
+ date
Fri Dec 12 08:54:59 UTC 2025
+ ls -l
total 0
+ whoami
jenkins
+ echo Desplegando en el entorno: TEST
Desplegando en el entorno: TEST
Finished: SUCCESS
```

Parte 3: Pipelines y Git (3 puntos)

Objetivo: Implementar un pipeline llamado “Despliegue”

New Item

Enter an item name

Despliegue

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one or more branches, runs a build, and performs post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build and post-build steps (such as publishing artifacts) and/or organizing complex activities that do not easily fit into a single stage.

1. Usa el siguiente repositorio <https://github.com/octocat>Hello-World.git>. No es necesario credenciales
2. Diseñe un pipeline con las siguientes etapas (*Stages*):
 - **Stage 'Build'**: Imprima un mensaje "Compilando aplicación..." y descarga el repositorio indicado en la carpeta de trabajo.

Los pipelines dividen sus acciones en “stage” y dentro de estas están los pasos o “steps”.

En este caso se hace un “echo” y un comando simple de git para descargarnos el repositorio, al no necesitar credenciales.

```
stage('Build') {
    steps {
        echo "Compilando aplicación..."
        // Clona el repositorio en el workspace
        git url: 'https://github.com/octocat>Hello-World.git'
    }
}
```

- **Stage 'Test'**: Compruebe si el archivo README existe. Si existe, el paso es exitoso; si no, debe fallar.

```
stage('Test') {
    steps {
        script {
            echo "Verificando existencia de README..."
            if (fileExists('README')) {
                echo "README encontrado. Test OK."
            } else {
                error("ERROR: No se encontró el archivo README. Falla la etapa Test.")
            }
        }
    }
}
```

- **Stage 'Deploy'**: Imprima "Desplegando a producción...".

```
stage('Deploy') {
    steps {
        echo "Desplegando a producción..."
    }
}
```

El script lo he puesto en: **Script Pipeline Punto 3.txt**

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
1v pipeline {
2    agent any
3
4v   stages {
5
6v     stage('Build') {
7v         steps {
8            echo "Compilando aplicación..."
9            // Clona el repositorio en el workspace
10           git url: 'https://github.com/octocat/Hello-World.git'
11       }
12     }
13
14v     stage('Test') {
15v         steps {
```

Use Groovy Sandbox ?

[Pipeline Syntax](#)

Advanced

Ha sido exitoso:

S	W	Name ↓	Last Success
		Despliegue	2 min 3 sec #1

```
[Pipeline] echo  
Verificando existencia de README...  
[Pipeline] fileExists  
[Pipeline] echo  
README encontrado. Test OK.  
[Pipeline] }  
[Pipeline] // script  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Deploy)  
[Pipeline] echo  
Desplegando a producción...  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

El output completo lo he puesto en: **Output - Script Pipeline Punto 3.txt**

Si vamos al Workspace de **Despliegue**, podemos ver como se ha descargado el repositorio (directorio “.git”) con todos sus archivos (archivo “README”).

The screenshot shows the Jenkins workspace interface. At the top, there is a URL bar with the address <http://localhost:8080/job/Despliegue/lastSuccessfulBuild/execution/node/3/ws/>. Below the URL bar, the breadcrumb navigation shows: Despliegue / #1 / Allocate node : Start / Workspace / Workspace. The main area is titled "Workspace". A search bar with a placeholder "/ .git" and a search icon is visible. Underneath the search bar, the ".git" directory is listed, containing a single file named "README". The "README" file is shown with its details: Dec 12, 2025, 9:02:02 AM, 13 B, and a refresh icon. At the bottom of the list, there is a link labeled "↳ (all files in zip)".

Parte 4: Pipelines y Git (4 puntos)

1. Creación de Credenciales

- Usa el almacén de credenciales de Jenkins para crear un texto secreto `SUPER_CLAVE_PRODUCCION_2025` y el id `token-despliegue`

Vamos a: **Administrar Jenkins** → **Credentials** → **System** → **Global Credentials**

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

token-despliegue

Description ?

Credencial Texto Secreto Examen

Create

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description	
 token-despliegue	Credencial Texto Secreto Examen	Secret text	Credencial Texto Secreto Examen	

Icon: S M L

2. Crear el Pipeline "Hijo" (Backend_Deploy)

Este será el pipeline que es *llamado* por el principal. Simulará el despliegue final.

1. Cree una **Nueva Tarea** tipo Pipeline llamada **Backend_Deploy**.

El script de este Pipeline está en: **Backend_Deploy.txt**

New Item

Enter an item name
Backend_Deploy

Select an item type

-  Freestyle project
Classic, general-purpose job type that checks out from up to steps like archiving artifacts and sending email notifications.
-  Pipeline
Orchestrates long-running activities that can span multiple build workflows (e.g., parallel builds) and/or organizing complex activities that do not

2. En la configuración del Pipeline, inserte estos dos parámetros que será VERSION y AUTOR, ambos strings

```
parameters {
    string(name: 'VERSION', description: 'Versión de la aplicación')
    string(name: 'AUTOR', description: 'Persona que solicita el despliegue')
}
```

stages {

3. Tendrá los siguientes stages:

- a. **Stage Verificación del entorno**: mostrará los mensajes siguientes y esperará 2 segundos:

- Iniciando despliegue de la versión [VERSIÓN]
- Solicitado por [AUTOR]

stages {

```
    stage('Verificación del entorno') {
        steps {
            echo "Iniciando despliegue de la versión ${params.VERSION}"
            echo "Solicitado por ${params.AUTOR}"
        }
    }
```

```
    // Esperar 2 segundos
    sleep(time: 2, unit: 'SECONDS')
```

}

b. **Stage Despliegue Exitoso:** mostrará el mensaje:

- La aplicación ha sido desplegada correctamente en el puerto 8080.

```
stage('Despliegue Exitoso') {
    steps {
        echo "La aplicación ha sido desplegada correctamente en el puerto 8080."
    }
}
```

Comprobación:

Console Output

```
Started by user Administrador ASIR Examen
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Backend_Deploy
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Verificación del entorno)
[Pipeline] echo
Iniciando despliegue de la versión
[Pipeline] echo
Solicitado por
[Pipeline] sleep
Sleeping for 2 sec
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Despliegue Exitoso)
[Pipeline] echo
La aplicación ha sido desplegada correctamente en el puerto 8080.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

El output está en: **Output - Backend_Deploy.txt**

Ahora, cuando vayamos construir de nuevo el Pipeline, nos saldrá que se construye con parámetros; y veremos la opción de usar los parámetros que hemos puesto antes. Ahora mismo cuando vayamos a ejecutarlo, podemos dejarlos vacíos, porque en el script no hay nada que haga uso de ellos y por lo tanto no dará error.

The screenshot shows the Jenkins Pipeline configuration page for the 'Backend_Deploy' pipeline. On the left, there is a sidebar with various options: Status, Changes, Build with Parameters (selected), Configure, Delete Pipeline, Stages, Rename, Pipeline Syntax, and Credentials. Below the sidebar, there is a 'Builds' section with a 'Filter' input field and a 'Build' button. The main area is titled 'Pipeline Backend_Deploy' and displays the message 'This build requires parameters:'. It shows two parameter fields: 'VERSION' (labeled 'Versión de la aplicación') and 'AUTOR' (labeled 'Persona que solicita el despliegue'). Both fields have empty input boxes. At the bottom right, there are 'Build' and 'Cancel' buttons.

3. Crear el Pipeline "Padre" (Orquestador_Master)

1. Cree una **Nueva Tarea** tipo Pipeline llamada **Orquestador_Master**.

New Item

Enter an item name

Orquestador_Master

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one step like archiving artifacts and sending email notifications.



Pipeline

Pipeline
Orchestrates long-running activities that can span multiple builds (multiple workflows) and/or organizing complex activities that do not easily fit into a workflow.

2. Tendrá los siguientes stages:

- a. Stage 'Build & Test': Mostrará dos mensajes "Compilando código fuente..." y "Tests pasados correctamente."

```
stages {
```

```
stage('Build & Test') {
    steps {
        echo "Compilando código fuente..."
        echo "Tests pasados correctamente."
    }
}
```

- b. Stage 'Aprobación de Pase a Producción': Muestra un mensaje '¿Deseas desplegar a Producción?' que debe ser aceptado para continuar el proceso. Si no contesta en 1 minuto el proceso debe ser abortado

```
stage('Aprobación de Pase a Producción') {
    steps {
        timeout(time: 1, unit: 'MINUTES') {
            input message: "{Deseas desplegar a Producción?}"
        }
    }
}
```

c. Stage 'Llamada a Despliegue':

- Muestra mensaje "Autorizando con token seguro:[API_TOKEN]"
- Llamamos al job hijo pasándole los parámetros que deseemos

"build job": llama al otro pipeline para que se ejecute. En nuestro caso "Backend_Deploy".

"wait: true": hace que se espere a la ejecución del otro script y que este termine antes de continuar.

```
stage('Llamada a Despliegue') {
    steps {
        echo "Autorizando con token seguro: ${API_TOKEN}"

        // Llamada al JOB HIJO Backend_Deploy
        build job: 'Backend_Deploy',
            wait: true,
            parameters: [
                string(name: 'VERSION', value: '1.0.0'),
                string(name: 'AUTOR', value: 'Administrador')
            ]
    }
}
```

3. Si el pipeline finaliza correctamente mostrará "El orquestador finalizó con éxito. El job hijo debería haberse ejecutado."
4. Si el pipeline no finaliza correctamente mostrará "El orquestador finalizó con éxito. El job hijo debería haberse ejecutado."

```
9
0^  post {
1^      success {
2          echo "El orquestador finalizó con éxito. El job hijo debería haberse ejecutado."
3      }
4^      failure {
5          echo "El orquestador falló. El job hijo no se ejecutó o falló en su ejecución."
6      }
7^      aborted {
8          echo "El orquestador fue abortado. No se completó el proceso."
9      }
0  }
```

El script del pipeline se encuentra en: **Orquestador_Master.txt**

Ha finalizado con éxito.

```
¿Deseas desplegar a Producción?  
Proceed or Abort  
Approved by Administrador ASIR Examen  
[Pipeline] }  
[Pipeline] // timeout  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Llamada a Despliegue)  
[Pipeline] echo  
Autorizando con token seguro: 12345-SECURE-TOKEN  
[Pipeline] build (Building Backend_Deploy)  
Scheduling project: Backend_Deploy  
Starting building: Backend_Deploy #3  
Build Backend_Deploy #3 completed: SUCCESS  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Declarative: Post Actions)  
[Pipeline] echo  
El orquestador finalizó con éxito. El job hijo debería haberse ejecutado.  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

¿Qué ocurre si falla el Pipeline Hijo? (no lo encuentra o cualquier otro motivo).

“El orquestador falló. El job hijo no se ejecutó o falló en su ejecución.”

¿Deseas desplegar a Producción?

Proceed or Abort

Approved by Administrador ASIR Examen

[Pipeline] }

[Pipeline] // timeout

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage (hide)

[Pipeline] { (Llamada a Despliegue)

[Pipeline] echo

Autorizando con token seguro: 12345-SECURE-TOKEN

[Pipeline] build

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Declarative: Post Actions)

[Pipeline] echo

El orquestador falló. El job hijo no se ejecutó o falló en su ejecución.

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

ERROR: No item named Backend_Deployjasjasjas found

Finished: FAILURE

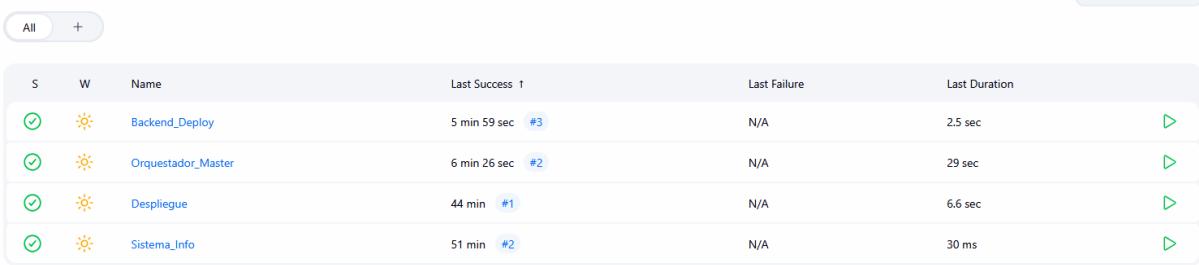
El output completo está en: **Output Proceed - Orquestador_Master.txt**

También: **Output Fail - Orquestador_Master.txt**

Para una versión en la que no se contesta a la pregunta, usamos:

Output Abort - Orquestador_Master.txt

En el Dashboard principal podemos ver como tras la ejecución de **Orquestador_Master**, también se ejecuta **Backend_Deploy**.



S	W	Name	Last Success	Last Failure	Last Duration	
✓	☀	Backend_Deploy	5 min 59 sec #3	N/A	2.5 sec	▶
✓	☀	Orquestador_Master	6 min 26 sec #2	N/A	29 sec	▶
✓	☀	Despliegue	44 min #1	N/A	6.6 sec	▶
✓	☀	Sistema_Info	51 min #2	N/A	30 ms	▶

Al principio del “Console Output” de “Backend_Deploy”, podemos ver como se inicia su ejecución desde “Orquestador_Master”.

✓ Console Output

```

Started by upstream project "Orquestador_Master" build number 2
originally caused by:
  Started by user Administrador ASIR Examen
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Backend_Deploy
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Verificación del entorno)
[Pipeline] echo
Iniciando despliegue de la versión 1.0.0
[Pipeline] echo
Solicitado por Administrador
[Pipeline] sleep
Sleeping for 2 sec
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Despliegue Exitoso)
[Pipeline] echo
La aplicación ha sido desplegada correctamente en el puerto 8080.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```