



Certified Tech Developer

The Ultimate Degree

Infraestructura II

Actividad obligatoria e individual

Dificultad: media

Continuous Integration con Jenkinsfile

Parte 4.2

Testing y Nexus upload

¡Continuamos con la práctica en Jenkins! En esta ejercitación vamos a correr unos test a nuestro proyecto para corroborar su calidad y —si logra pasar los chequeos de manera exitosa— vamos a subir nuestro artefacto/aplicación a un repositorio de Nexus3.

Nuestro desafío es:

- Integrar testeo continuo, corriendo el comando de test de Maven y en la etapa del pipeline correspondiente.
- Archivar los resultados de los test en Jenkins.
- Si los test dan bien, vamos a subir el artefacto compilado.

En la siguiente página se encuentra la resolución. Continúa únicamente para realizar una autoevaluación.

Resolución

- **Integrar testeo continuo, corriendo el comando de test de Maven y en la etapa del pipeline correspondiente.**

Para ello, utilizaremos el proyecto de Maven que se encuentra en nuestro [repositorio de github](#).

En la etapa de test de nuestro Jenkins file agregaremos el comando para correr los test. Queda de la siguiente forma:

```
stage('Test') {  
  
    steps {  
  
        dir ('maven-adderapp') {  
  
            sh "mvn test"  
  
        }  
  
    }  
  
}
```

Agregar la directiva dir para trabajar en el directorio donde se encuentra el proyecto Maven en el repo.

Al correr el pipeline podremos observar —luego del paso de **compilación**— el de **testing**. Allí nombra la cantidad de test ejecutados, salteados y el conteo de fallas o errores en los mismos.



```
+ mvn test
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.digitalhouse.examples.maven.java:adderapp:jar:1.0.0
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-jar-plugin is missing. @ line 28, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----< com.digitalhouse.examples.maven.java:adderapp >-----
[INFO] Building adderapp 1.0.0
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ adderapp ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/jenkins_home/workspace/test/maven-adderapp/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ adderapp ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ adderapp ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/jenkins_home/workspace/test/maven-adderapp/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ adderapp ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.0:test (default-test) @ adderapp ---
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.digitalhouse.examples.maven.java.AdderTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.067 s - in com.digitalhouse.examples.maven.java.AdderTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.524 s
[INFO] Finished at: 2021-09-07T20:05:57Z
[INFO] -----
```

- **Archivar los resultados de los test en Jenkins.**

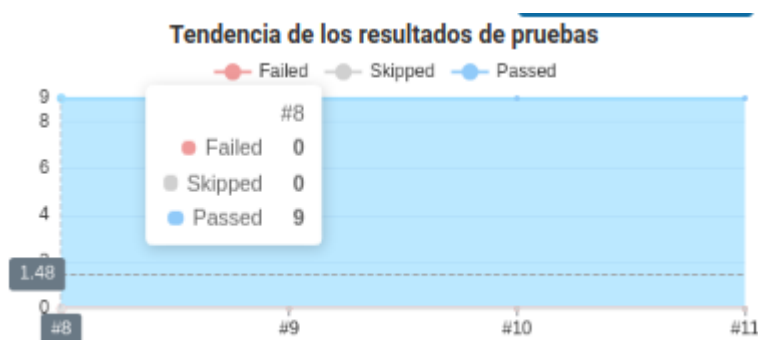
Luego de ejecutar los test se genera —en la mayoría de los casos— un reporte en formato xml. En nuestro caso, este archivo se genera en la carpeta “target/surefire-reports”. Jenkins nos permite archivar estos resultados en su storage para poder visualizarlos y tener un historial de ellos.

Modificaremos nuestro Jenkinsfile para que en la sección **post** siempre recopilemos los resultados de los test:



```
post {  
  
    always {  
  
        dir('maven-adderapp') {  
  
            junit 'target/surefire-reports/*.xml'  
  
        }  
  
    }  
  
    success {  
  
        dir ('maven-adderapp') {  
  
            archiveArtifacts artifacts: 'target/*.jar', fingerprint:  
true  
  
        }  
  
    }  
  
}
```

Teniendo esto configurado y luego de lanzar el pipeline más de una vez, se genera un gráfico que nos presenta el historial de los resultados de las pruebas ejecutadas:



Si hacemos clic en el gráfico, o si nos dirigimos a un build que haya recolectado estos reportes de test, podremos ver el link **Resultados de los test**:

Panel de Control > test > #9

Back to Project

Status

Changes

Console Output

Edit Build Information

Delete build '#9'

Git Build Data

Resultado de los tests

Ver firmas

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Next Build

Build #9 (7 sep. 2021 20:05:17)

Build Artifacts

adderapp-1.0.0.jar 3.77 KB view

Iniciado por el usuario [jenkins](#)

Revision: cfba582c45a280d0e913f72e137ec2920b7490f1

Repository: <https://github.com/repoinfradh/Infra2.git>

refs/remotes/origin/test+nexus

Resultado de los tests (Sin fallas)

Al hacer clic en él podremos ver el detalle de cada test corrido:

Resultado de tests : AdderTest

0 fallidos (±0)

9 tests (±0)

Tardó 79 Ms.

[añadir descripción](#)

Todos los tests

| Nombre del test | Duración | Estado |
|---|----------|---------|
| whenAddFirstNegativeSecondPositive_ThenSumIsCorrect | 1 Ms | Pasados |
| whenAddFirstNegativeSecondZero_ThenSumIsEqualToFirst | 1 Ms | Pasados |
| whenAddFirstPositiveSecondNegative_ThenSumIsCorrect | 67 Ms | Pasados |
| whenAddFirstPositiveSecondZero_ThenSumIsEqualToFirst | 1 Ms | Pasados |
| whenAddFirstZeroSecondNegative_ThenSumIsEqualToSecond | 0 Ms | Pasados |
| whenAddFirstZeroSecondPositive_ThenSumIsEqualToSecond | 2 Ms | Pasados |
| whenAddTwoZeros_ThenSumIsZero | 1 Ms | Pasados |
| whenTwoNegatives_ThenSumIsCorrect | 1 Ms | Pasados |
| whenTwoPositives_ThenSumIsCorrect | 5 Ms | Pasados |

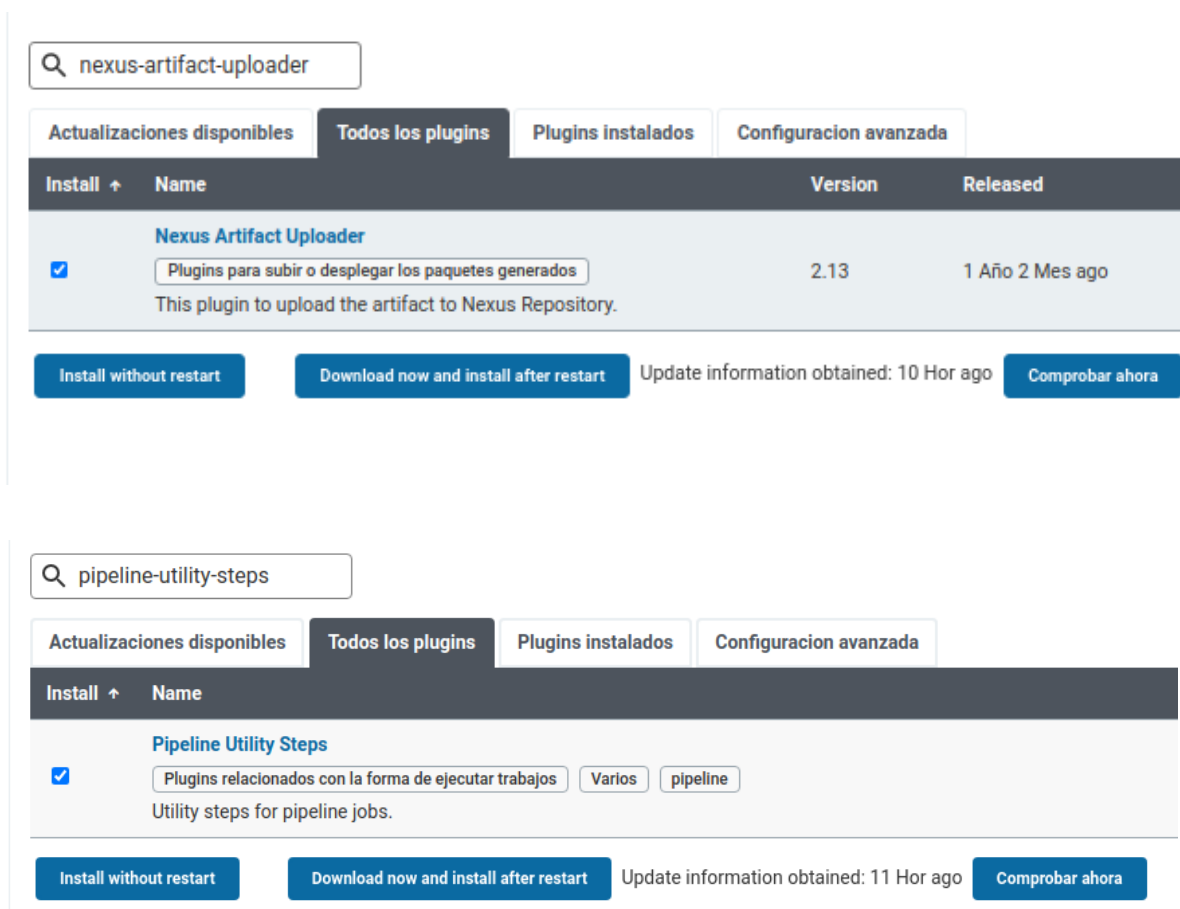
- **Si los test dan bien, vamos a subir el artefacto compilado**

En caso de que los test fallen, esto provocará una parada en el pipeline: se aborta su ejecución. Pero si los test pasan de manera exitosa, solo nos queda pendiente subir nuestro artefacto.

Subiremos nuestro artefacto a Nexus 3 (podemos usar la instalación realizada en la práctica 4.1 o la que se ofreció en ejercitaciones anteriores).

Debemos integrar mediante un plugin Nexus con Jenkins. Buscamos y encontramos el siguiente: <https://www.jenkins.io/doc/pipeline/steps/nexus-artifact-uploader/>

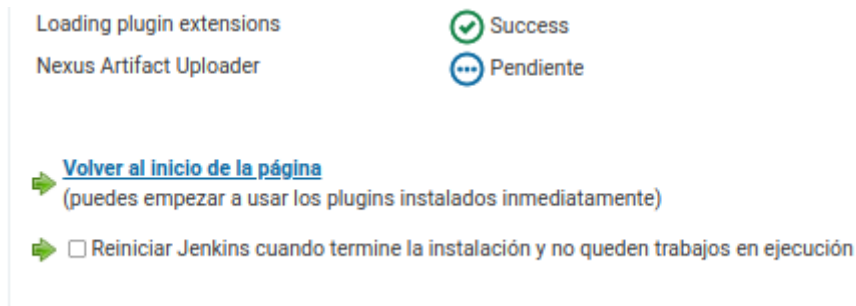
Para instalarlo iremos al administrador de plugins de Jenkins "<http://jenkins/pluginManager/available>" y lo buscamos:



The screenshot shows the Jenkins Plugin Manager interface. The search bar contains 'nexus-artifact-uploader'. The 'Todos los plugins' tab is selected. The table lists the 'Nexus Artifact Uploader' plugin, version 2.13, released 1 year and 2 months ago. Below the table are buttons for 'Install without restart', 'Download now and install after restart', and 'Comprobar ahora'. The update information is '10 Hor ago'.

The second screenshot shows the same interface with the search bar containing 'pipeline-utility-steps'. The 'Pipeline Utility Steps' plugin is listed, version 'Varios', released '11 Hor ago'. Below the table are buttons for 'Install without restart', 'Download now and install after restart', and 'Comprobar ahora'.

Adicionalmente, buscamos y seleccionamos **pipeline-utility-steps** que nos brinda la posibilidad de trabajar con el pom de nuestro proyecto. Hacemos clic en **Download now and install after restart**:



Seleccionamos "Reiniciar Jenkins cuando termine...". Mientras esperamos podemos modificar el pipeline para usar este plugin.

Para hacerlo, agregaremos en la directiva **script** una línea que nos permitirá obtener información del archivo pom.xml de nuestro proyecto con el objetivo de subir el artefacto con su metadata correspondiente.

Luego de obtener esta información, haremos uso del plugin instalado pasándole todo lo necesario. Dejaremos entonces la sección **success** de la **post** ejecución del pipeline de la siguiente forma:

```
success {  
  
    dir ('maven-adderapp') {  
  
        script {  
  
            pom = readMavenPom file: "pom.xml";  
  
            files = findFiles(glob: "target/*.${pom.packaging}");  
            filePath = files[0].path;  
  
            nexusArtifactUploader(  
  
                nexusVersion: "3.33.1-01",
```



```
        protocol: "http",

        nexusUrl: "${env.NEXUS}:8081",

        groupId: pom.groupId,

        version: pom.version,

        repository: "maven-jenkins",

        credentialsId: "nexus",

        artifacts: [

            [artifactId: pom.artifactId,

            classifier: '',

            file: filePath,

            type: pom.packaging],

            [artifactId: pom.artifactId,

            classifier: '',

            file: "pom.xml",

            type: "pom"]

        ]

    );

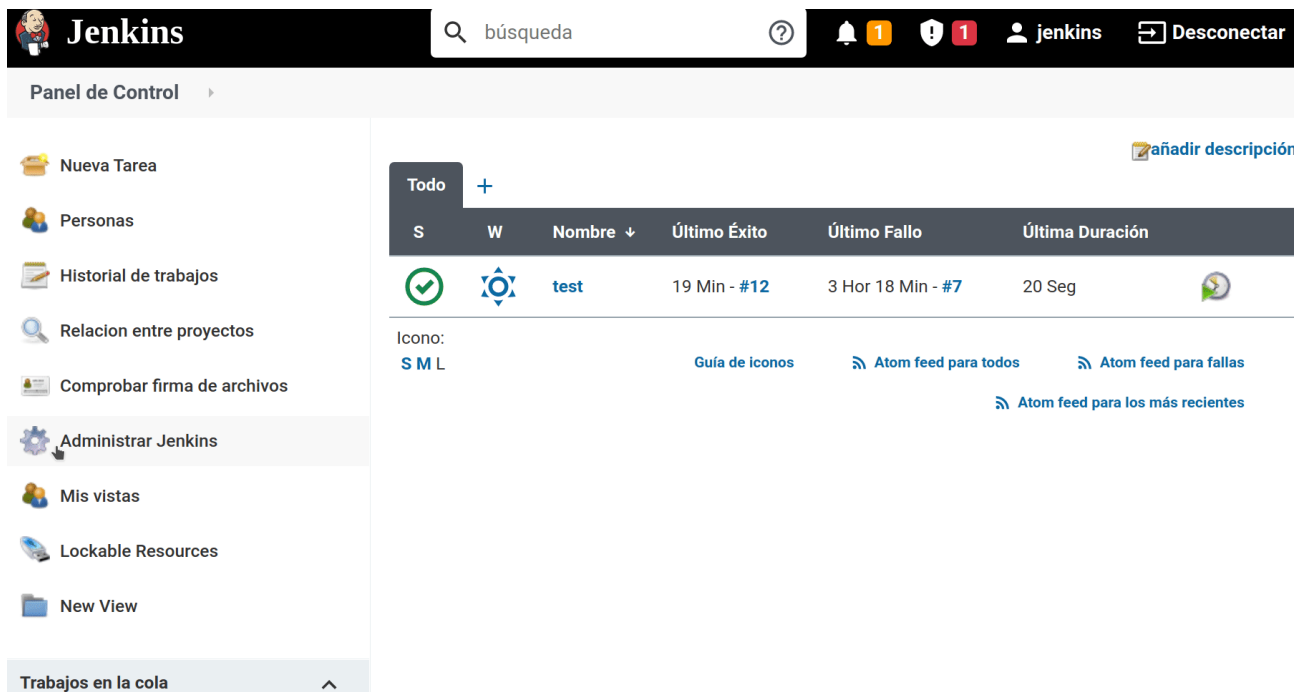
}

}
```


¡Vamos bien, pero tenemos algunos pendientes para resolver! Sobre el usuario de Nexus –como se puede ver en el código– pasamos un **credentialsId**:

```
credentialsId: "nexus",
```

Para generarlo, nos dirigimos a **Administrar Jenkins** → **Manage Credentials** → **Store** → **Jenkins** → **Global** → **Add Credentials**. En este punto agregamos usuario y contraseña de nuestro Nexus. Para terminar, hacemos clic en ok y tendremos lista nuestra credencial:



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and user information (jenkins) with a 'Desconectar' button. The left sidebar contains a 'Panel de Control' with links to 'Nueva Tarea', 'Personas', 'Historial de trabajos', 'Relacion entre proyectos', 'Comprobar firma de archivos', 'Administrar Jenkins' (highlighted), 'Mis vistas', 'Lockable Resources', and 'New View'. The main content area displays a table of build jobs. The first job is named 'test' and has a status of 'S' (Success). Below the table, there are links for 'Icono: S M L', 'Guía de iconos', and 'Atom feed' links for all builds, failures, and recent builds.

| S | W | Nombre ↓ | Último Éxito | Último Fallo | Última Duración |
|---|----|----------|--------------|-------------------|-----------------|
| ✓ | ⚙️ | test | 19 Min - #12 | 3 Hor 18 Min - #7 | 20 Seg |

Para tener en cuenta: el **ID** es el **credentialsId** que pasamos en el Jenkinsfile.

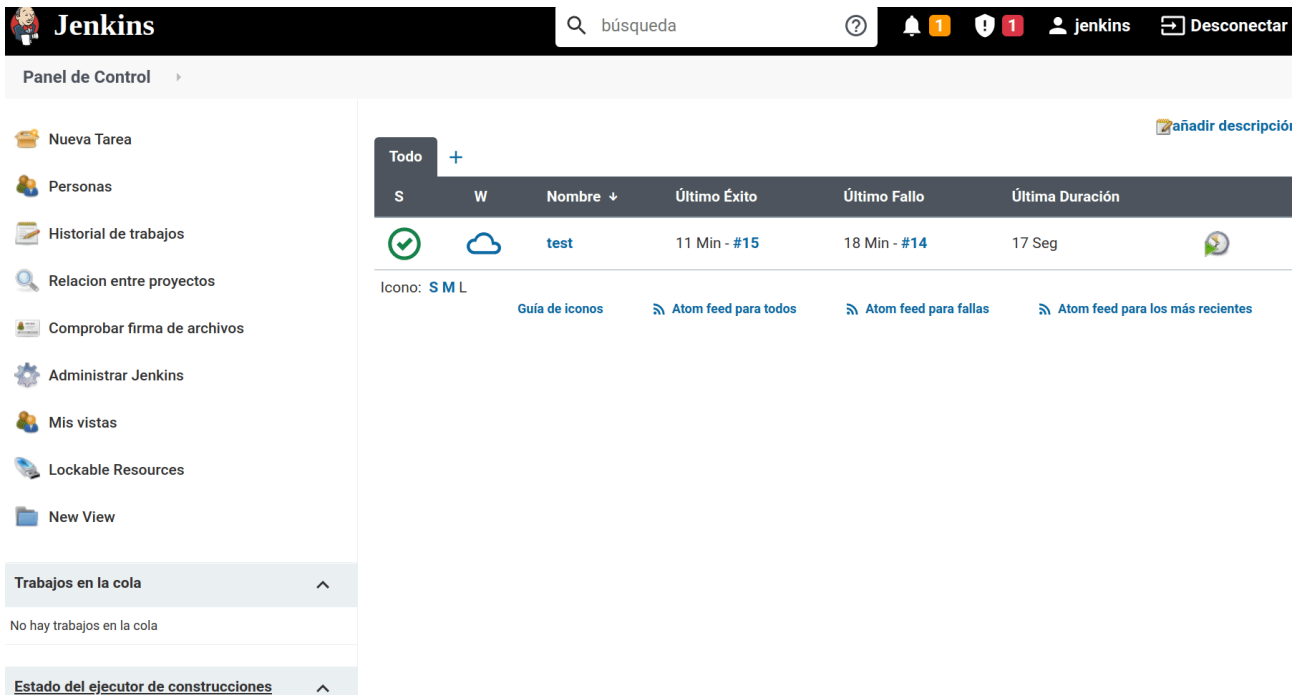
También vemos:

```
nexusUrl: "${env.NEXUS}:8081",
```

De esta forma podemos tomar parámetros en un Jenkins file. En este caso enviaremos el host de nuestro servidor Nexus 3.

Para crear el parámetro en el pipeline nos dirigimos a nuestro job/pipeline. Vamos a configuración, marcamos la opción **Esta ejecución debe parametrizarse** y añadimos

un parámetro de tipo “cadena” con el nombre **NEXUS** y el valor de la IP donde tenemos corriendo Nexus. Luego, aplicamos y guardamos.



The screenshot shows the Jenkins web interface. At the top is a navigation bar with the Jenkins logo, a search bar, and user information. Below this is a 'Panel de Control' (Control Panel) on the left with various links like 'Nueva Tarea', 'Personas', and 'Historial de trabajos'. The main area displays a table of jobs. The first job, 'test', is shown with a green checkmark icon, indicating it is successful. Below the table, there are links for 'Guía de iconos' and 'Atom feed' for different views.

| S | W | Nombre | Último Éxito | Último Fallo | Última Duración |
|---|---|--------|--------------|--------------|-----------------|
| ✓ | ☁ | test | 11 Min - #15 | 18 Min - #14 | 17 Seg |

Para terminar corremos el pipeline. Nos mostrará un cuadro de diálogo donde debemos ingresar el parámetro **NEXUS**. Hacemos clic en Ejecución.

Pipeline test

Esta ejecución requiere parámetros adicionales:

NEXUS

Nexus Server url

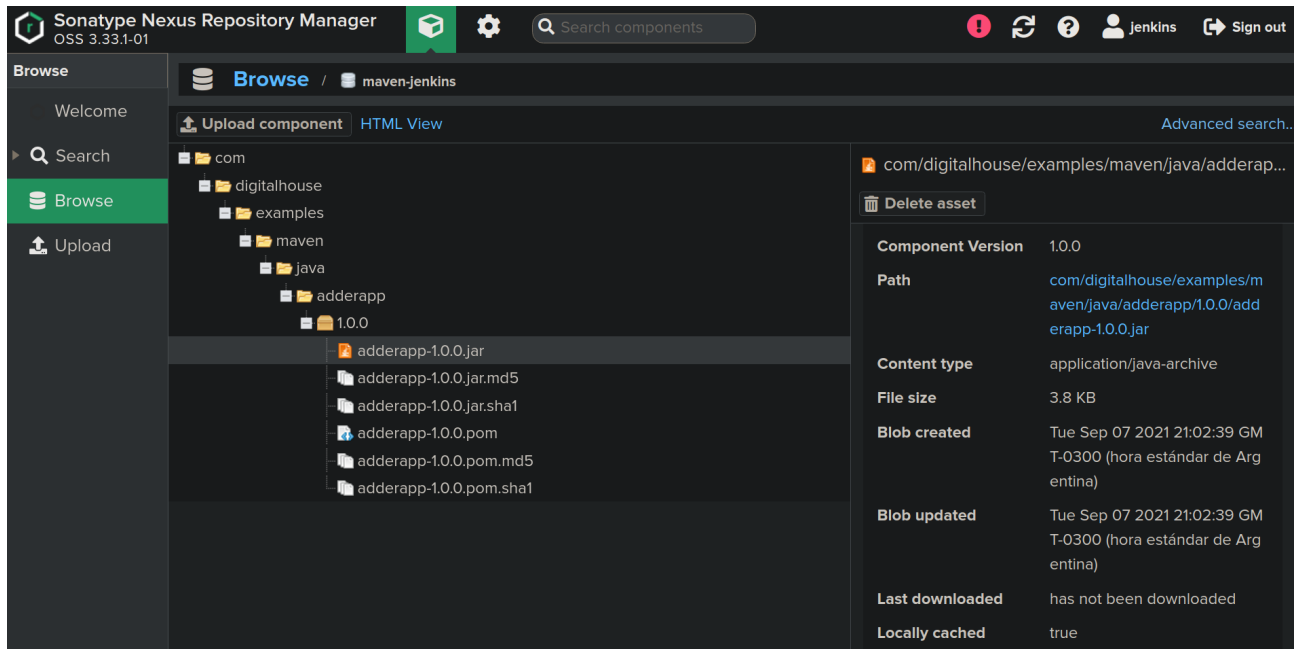
Ejecución

Esto iniciará nuestro pipeline corriendo todos los procesos y subiendo nuestro artefacto a Nexus. Veremos el recorrido en verde:

Stage View

| | Declarative: Checkout SCM | Declarative: Tool Install | Build | Test | Deploy | Declarative: Post Actions |
|--|---------------------------|---------------------------|-------|------|--------|---------------------------|
| Average stage times: (Average <u>full</u> run time: ~19s) | 1s | 365ms | 5s | 5s | 653ms | 1s |
| #19 Sep 07 21:06 No Changes | 813ms | 319ms | 4s | 4s | 620ms | 1s |

En este momento podremos revisar el repositorio en nuestro Nexus y corroborar la subida del artefacto compilado.



The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar has a 'Browse' menu. The main area displays a tree view of the repository structure: com > digitalhouse > examples > maven > java > adderapp > 1.0.0. Under the 1.0.0 version, several files are listed: adderapp-1.0.0.jar, adderapp-1.0.0.jar.md5, adderapp-1.0.0.jar.sha1, adderapp-1.0.0.pom, adderapp-1.0.0.pom.md5, and adderapp-1.0.0.pom.sha1. The right sidebar shows the details for the selected artifact, com/digitalhouse/examples/maven/java/adderapp-1.0.0.jar, including its version (1.0.0), path, content type (application/java-archive), file size (3.8 KB), and creation/update timestamps.

Conclusión

En esta práctica se integró paso a paso Nexus con Jenkins dándonos la posibilidad de tener un registro de las compilaciones realizadas como Releases. Tener este proceso es de vital importancia porque nos genera un historial de todas las versiones de nuestros productos listas para ser desplegadas.

¡Nos volvemos a encontrar en la próxima práctica!