

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE
AREQUIPA



INGENIERÍA DE SOFTWARE II

Tutorial : Integración Continua con Jenkins

Alumna :

- Chullunquía Rosas, Sharon Rossely

Profesor :

- Sarmiento Calisaya, Edgar

21 de octubre de 2020

Índice

1. Herramienta : Jenkins	2
2. Instalación	2
2.1. Prerrequisitos	2
2.2. Paso 1: Personalización de la imagen de Jenkins Docker	3
2.3. Paso 2: Ejecución de myjenkins:1.0	4
2.4. Paso 3: Desbloqueo de Jenkins	5
2.5. Paso 4: Fork y clone al repositorio de muestra en GitHub	5
3. Integración del Proyecto	6
3.1. Paso 1: Creación de un proyecto Pipeline en Jenkins	6
3.2. Paso 2: Creación del archivo Jenkinsfile	8
3.3. Paso 3: Instalación de Blue Ocean	8
3.4. Paso 4: Usando Blue Ocean en proyecto Pipeline	9
3.5. Paso 5: Agregando una etapa de evaluación a Pipeline	11
3.6. Paso 6: Agregando la etapa final Deliver a Pipeline	13

1. Herramienta : Jenkins



Jenkins es un servidor de automatización de código abierto autónomo que se puede utilizar para automatizar todo tipo de tareas relacionadas con la creación, prueba y entrega o implementación de software.

Jenkins se puede instalar a través de paquetes de sistema nativos, *Docker* o incluso ejecutar de forma independiente en cualquier máquina que tenga instalado *Java Runtime Environment (JRE)*. [1]

Esta herramienta, proviene de otra similar llamada Hudson, ideada por Kohsuke Kawaguchi, que trabajaba en *Sun*. Unos años después de que *Oracle* comprara *Sun*, la comunidad de *Hudson* decidió renombrar el proyecto a *Jenkins*, migrar el código a [Github](#) y continuar el trabajo desde ahí. No obstante, *Oracle* ha seguido desde entonces manteniendo y trabajando en *Hudson*. [2]

2. Instalación

En este tutorial, ejecutaremos *Jenkins* como un contenedor de *Docker* desde la imagen de *Docker* [jenkins/jenkins](#).

2.1. Prerrequisitos

Para este tutorial, necesitaremos:

1. Una máquina macOS, Linux o Windows con:
 - 256 MB de RAM, aunque se recomiendan más de 512 MB.
 - 10 GB de espacio en disco para *Jenkins* y sus imágenes y contenedores de *Docker*.

2. El siguiente software instalado:

- [Docker](#)
- Git y, opcionalmente, *GitHub Desktop* .

2.2. Paso 1: Personalización de la imagen de Jenkins Docker

Abrimos la terminal y personalizamos la imagen oficial de *Jenkins Docker*, ejecutando los siguientes dos pasos:

- Creamos *Dockerfile* con el siguiente contenido:

```
FROM jenkins/jenkins:2.249.1-lts
USER root
RUN apt-get update && apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common
RUN curl -fsSL https://download.docker.com/linux/debian/gpg | \
    apt-key add -
RUN apt-key fingerprint 0EBFCD88
RUN add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    \ $(lsb_release -cs) \
    stable"
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
ENTRYPOINT ["/sbin/tini", "--", "/usr/local/bin/jenkins.sh"]
```

- Creamos una nueva imagen de *Docker* a partir de este *Dockerfile* y asigne un nombre significativo a la imagen, por ejemplo, "*myjenkins: 1.0*":

```
$ sudo docker build -t "myjenkins:1.0" .
```

```

sharon@2d2:~/Documentos/Programs/Jenkins$ sudo docker build -t "myjenkins:1.0" .
Sending build context to Docker daemon 2.56kB
Step 1/9 : FROM jenkins/jenkins:2.249.1-lts
2.249.1-lts: Pulling from jenkins/jenkins
3192219afd04: Pull complete
17c160265e75: Pull complete
cc4fe40d0e61: Pull complete
9d647f502a07: Pull complete
d108b8c498aa: Pull complete
1bfe918b8aa5: Pull complete
daf1a7c0751: Pull complete
242f7ffc3caf: Pull complete
78ed02c828a7: Pull complete
9d35776d2ef1: Pull complete
414082f9c614: Pull complete
2be469abdd47: Pull complete
edb7721f6fb4: Pull complete
cee856ff2763: Pull complete
3ea0bc355208: Pull complete
24e8e0303a63: Pull complete
6f49b4e99d99: Pull complete
c9035cc52593: Pull complete
34159c7dd26f: Pull complete
e32ce11a5e47: Pull complete
a7839bfb1925: Pull complete
Digest: sha256:a3e7b2b6efbc2c252608b028bb844e419d44ad5e3974770c4543ab7ae6e8eb27
Status: Downloaded newer image for jenkins/jenkins:2.249.1-lts
--> 190554e5446b
Step 2/9 : USER root
--> Running in aeadadcbecb32

```

Figura 1: Construcción de una imagen basada en el modelo de Dockerfile

2.3. Paso 2: Ejecución de myjenkins:1.0

Ejecutamos nuestra imagen *myjenkins:1.0* como contenedor en *Docker* con el siguiente *docker run* comando:

```

$ sudo docker run --rm -u root --name jenkins-tutorial \
--volume jenkins-data:/var/jenkins_home \
--volume /var/run/docker.sock:/var/run/docker.sock \
--volume "$HOME":/home --publish 8080:8080 myjenkins:1.0

```

```

sharon@2d2:~/Documentos/Programs/Jenkins$ sudo docker run --rm -u root --name jenkins-tutorial --volume jenkins-data:/var/jenkins_home --volume /var/run/docker.sock:/var/run/docker.sock
--volume "$HOME":/home --publish 8080:8080 myjenkins:1.0
Running from: /usr/share/jenkins/jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
2020-10-22 02:58:07.366+0000 [id=1] INFO org.eclipse.jetty.util.log.Log#initialized: Logging initialized @520ms to org.eclipse.jetty.util.log.JavaUtilLog
2020-10-22 02:58:07.559+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2020-10-22 02:58:09.201+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2020-10-22 02:58:09.291+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-9.4.30.v20200611; built: 2020-06-11T12:34:51.929Z; git: 271836e4c1f4612f12b7b13ef5a92a9276348d0;
jvm 1.8.0_242-b08
2020-10-22 02:58:09.684+0000 [id=1] INFO o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServlet
2020-10-22 02:58:09.762+0000 [id=1] INFO o.e.j.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager workerName=node0
2020-10-22 02:58:09.763+0000 [id=1] INFO o.e.j.s.DefaultSessionIdManager#doStart: No SessionScavenger set, using defaults
2020-10-22 02:58:09.768+0000 [id=1] INFO o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every 660000ms
2020-10-22 02:58:10.278+0000 [id=1] INFO hudson.WebAppMain$contextInitialized: Jenkins home directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")
2020-10-22 02:58:10.476+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Started v.0733c423e(jenkins v2.249.1, /file:///var/jenkins_home/war, AVAILABLE) (/var/jenkins_home/war)
2020-10-22 02:58:10.507+0000 [id=1] INFO o.e.j.server.AbstractConnector#doStart: Started ServerConnector@387a8303(HTTP/1.1, (http/1.1)) (0.0.0.0:8080)
2020-10-22 02:58:10.507+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: Started @3070ms
2020-10-22 02:58:10.516+0000 [id=21] INFO winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2020-10-22 02:58:13.362+0000 [id=28] INFO jenkins.InitReactorRunner$1#onAttained: Started initialization
2020-10-22 02:58:13.383+0000 [id=27] INFO jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2020-10-22 02:58:15.501+0000 [id=30] INFO jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2020-10-22 02:58:15.520+0000 [id=26] INFO jenkins.InitReactorRunner$1#onAttained: Started all plugins
2020-10-22 02:58:15.533+0000 [id=27] INFO jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2020-10-22 02:58:16.577+0000 [id=27] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
2020-10-22 02:58:16.578+0000 [id=27] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2020-10-22 02:58:16.578+0000 [id=27] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2020-10-22 02:58:16.588+0000 [id=32] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2020-10-22 02:58:16.625+0000 [id=46] INFO hudson.model.AsyncPeriodicWork$1#lambda$doRun$0: Started Download metadata
2020-10-22 02:58:16.674+0000 [id=46] INFO hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2020-10-22 02:58:18.035+0000 [id=32] INFO o.s.c.s.AbstractApplicationContext#prepareRefresh: Refreshing org.springframework.web.context.support.StaticWebApplicationContext@2177bd8c: displ
ay name [Root WebApplicationContext], startup date [Thu Oct 22 02:58:18 UTC 2020], root of context hierarchy
2020-10-22 02:58:18.035+0000 [id=32] INFO o.s.c.s.AbstractApplicationContext#prepareRefresh: Refreshing org.springframework.web.context.support.StaticWebApplicationContext@2177bd8c: displ
ay name [Root WebApplicationContext], startup date [Thu Oct 22 02:58:18 UTC 2020], root of context hierarchy
2020-10-22 02:58:18.051+0000 [id=32] INFO o.s.b.f.s.DefaultListableBeanFactory#preInstantiateSingletons: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultL
istableBeanFactory@4d54e2b2: defining beans [filter, legacy]; root of factory hierarchy
2020-10-22 02:58:18.051+0000 [id=32] INFO o.s.b.f.s.DefaultListableBeanFactory#preInstantiateSingletons: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultL
istableBeanFactory@4d54e2b2: defining beans [filter, legacy]; root of factory hierarchy
2020-10-22 02:58:18.391+0000 [id=32] INFO o.s.c.s.AbstractApplicationContext#prepareRefresh: Refreshing org.springframework.web.context.support.StaticWebApplicationContext@10731351: displ
ay name [Root WebApplicationContext], startup date [Thu Oct 22 02:58:18 UTC 2020], root of context hierarchy
2020-10-22 02:58:18.391+0000 [id=32] INFO o.s.b.f.s.DefaultListableBeanFactory#preInstantiateSingletons: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultL
istableBeanFactory@4d54e2b2: defining beans [filter, legacy]; root of factory hierarchy
2020-10-22 02:58:18.392+0000 [id=32] INFO o.s.b.f.s.DefaultListableBeanFactory#preInstantiateSingletons: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultL
istableBeanFactory@4d54e2b2: defining beans [filter, legacy]; root of factory hierarchy

```

Figura 2: Creamos un contenedor usando la imagen generada

2.4. Paso 3: Desbloqueo de Jenkins

El comando anterior nos generará un token, el cual usaremos como contraseña para desbloquear *Jenkins*.

Entramos a <http://localhost:8080> y esperamos hasta que aparezca la página *Unlock Jenkins*. Luego, ingresamos nuestro token generado para desbloquear *Jenkins*.

```
*****
*****
*****

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

ce64c6af42b84d7b96525225451a179e

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
```

Figura 3: Token generado ce64c6af42b84d7b96525225451a179e



Figura 4: Iniciar sesión en Jenkins usando token

2.5. Paso 4: Fork y clone al repositorio de muestra en GitHub

Hacemos *fork* a [simple-node-js-react-npm-app](#) en GitHub en su cuenta local de GitHub.

Clonamos el *fork* del repositorio [simple-node-js-react-npm-app](#) (en GitHub) localmente en nuestra maquina. Abrimos la terminal en el siguiente directorio.

```
/home/<your-username>/GitHub/
```

y ejecutamos el siguiente comando.

```
git clone https://github.com/YOUR-GITHUB-ACCOUNT-NAME/simple-node-js-react-npm-app
```

3. Integración del Proyecto

3.1. Paso 1: Creación de un proyecto Pipeline en Jenkins

Iniciamos sesión nuevamente si es necesario y hacemos click en **crear nueva tarea** en **¡ Bienvenido a Jenkins!**.

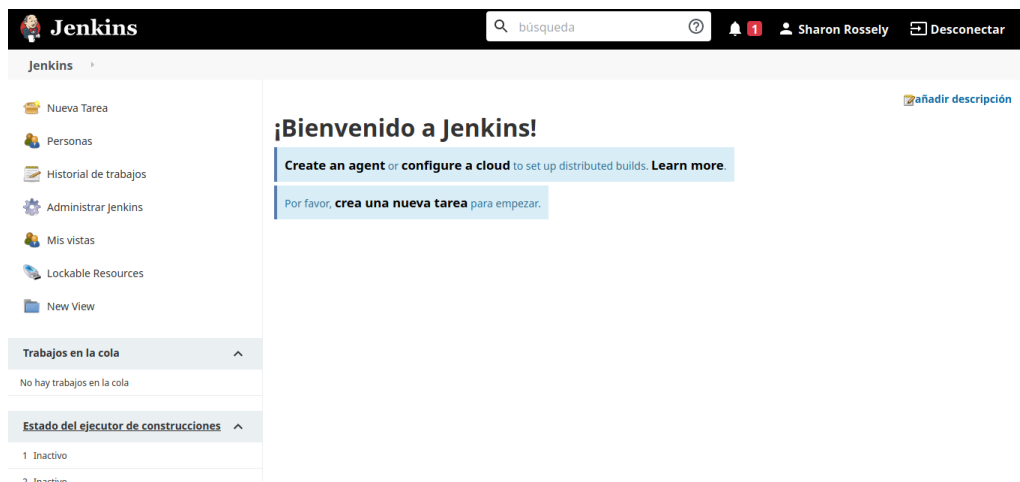


Figura 5: Página de inicio

En el campo Ingresamos un nombre para el elemento , especificamos el nombre del nuevo proyecto pipeline (e.g. simplee-node-js-react-npm-app).Hacemos click en *Pipeline* y presionamos OK.

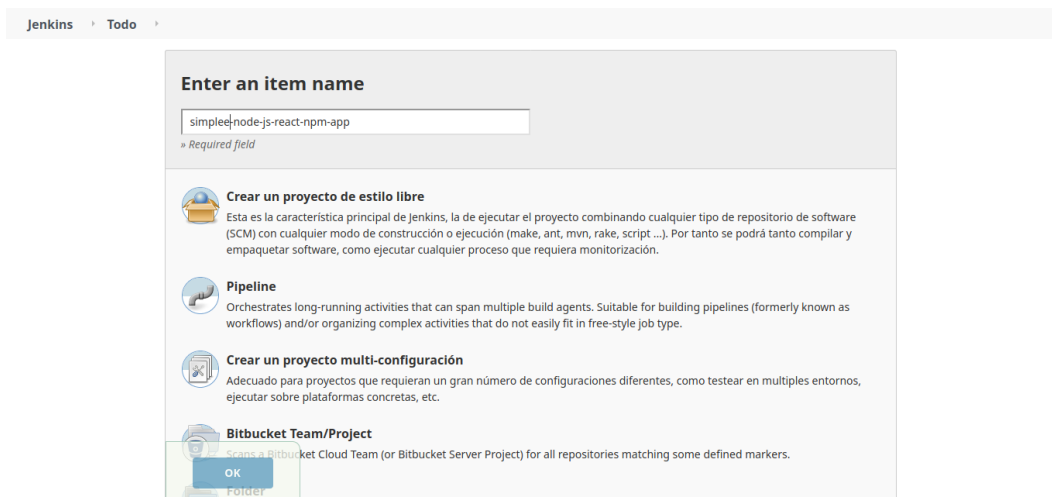


Figura 6: Nuevo proyecto pipeline

Hacemos clic en la pestaña Pipeline en la parte superior de la página para desplazarse hacia abajo hasta la sección *Pipeline*.

En el campo de definición, elija la opción *Pipeline script from SCM*. Esta opción le indica a *Jenkins* que obtenga su canalización de Source Control Management (SCM), que será su repositorio de Git clonado localmente.

En el campo SCM, elija Git. En el campo URL del repositorio, especifique la ruta del directorio de su repositorio clonado localmente, para Linux */home/GitHub/simple-node-js-react-npm-app*.

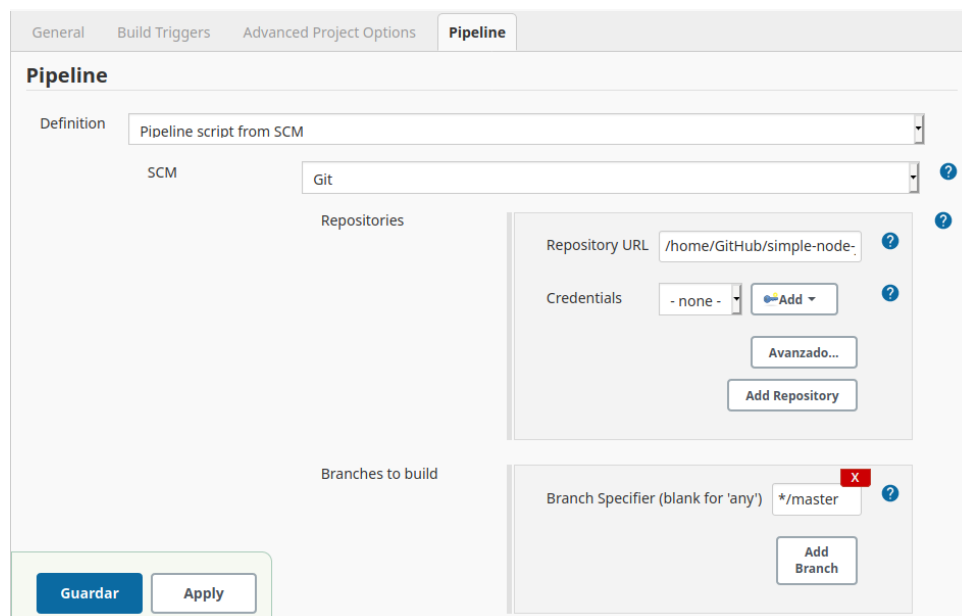
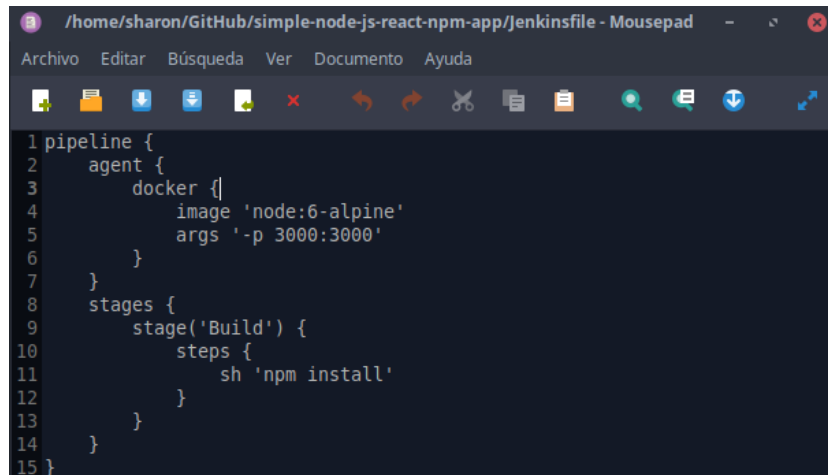


Figura 7: Agregando ajustes al proyecto Pipeline

3.2. Paso 2: Creación del archivo Jenkinsfile

Creamos el archivo *Jenkinsfile*, el cual tendrá el siguiente contenido.

A screenshot of a code editor window titled "/home/sharon/GitHub/simple-node-js-react-npm-app/Jenkinsfile - Mousepad". The editor shows a Jenkinsfile with the following content:

```
1 pipeline {
2   agent {
3     docker {
4       image 'node:6-alpine'
5       args '-p 3000:3000'
6     }
7   }
8   stages {
9     stage('Build') {
10      steps {
11        sh 'npm install'
12      }
13    }
14  }
15 }
```

Figura 8: Estructura

Luego, hacemos commit en el repositorio de Git clonado localmente.(simple-node-js-react-npm-app).

3.3. Paso 3: Instalación de Blue Ocean

Instalamos el plugin *Jenkins Blue Ocean*, este plugin nos servirá como interfaz para la integración automática. Hacemos click en **Administrar Jenkins**, luego click en **Administrar plugins**

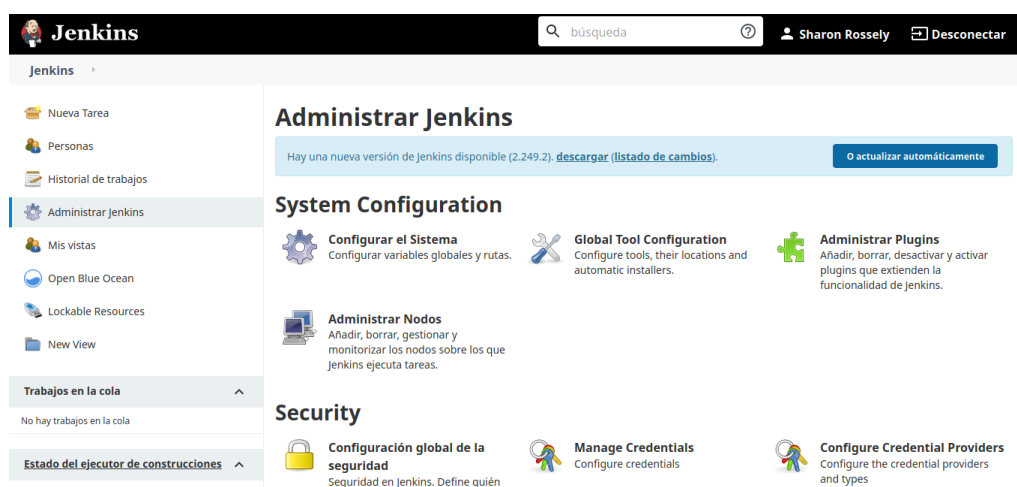


Figura 9: Página de inicio

Buscamos el plugin y lo instalamos.

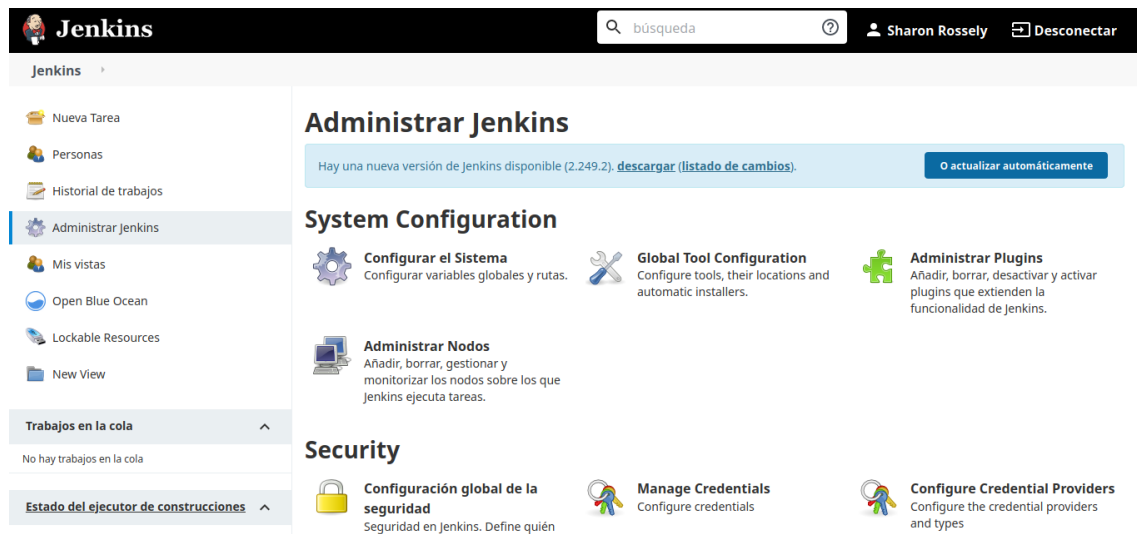


Figura 10: Buscando el plugin Blue Ocean

<input checked="" type="checkbox"/>	Bitbucket Pipeline for Blue Ocean BlueOcean Bitbucket pipeline creator	1.24.1	Desinstalar
<input checked="" type="checkbox"/>	Blue Ocean BlueOcean Aggregator	1.24.1	Desinstalar
<input checked="" type="checkbox"/>	Blue Ocean Core JS The Jenkins Plugins Parent POM Project	1.24.1	Desinstalar

Figura 11: Plugin Blue Ocean ya instalado

3.4. Paso 4: Usando Blue Ocean en proyecto Pipeline

En el cuadro de mensaje **This job has not been run**, haga clic en **Iniciar**, luego haga clic rápidamente en el enlace **ABRIR** que aparece brevemente en la parte inferior derecha para ver a *Jenkins* construyendo su proyecto Pipeline.

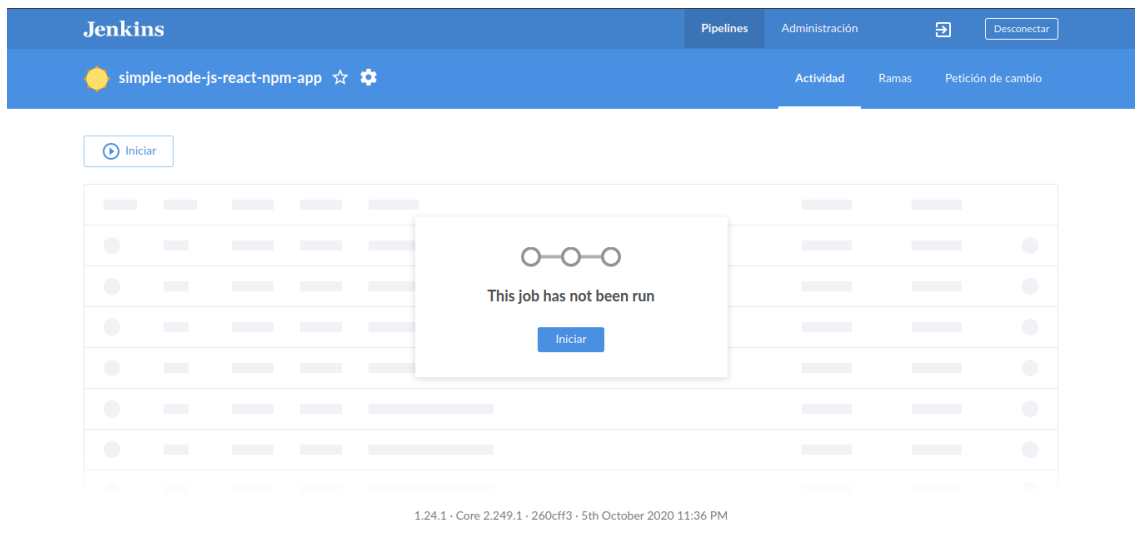


Figura 12: Proyecto en ejecución

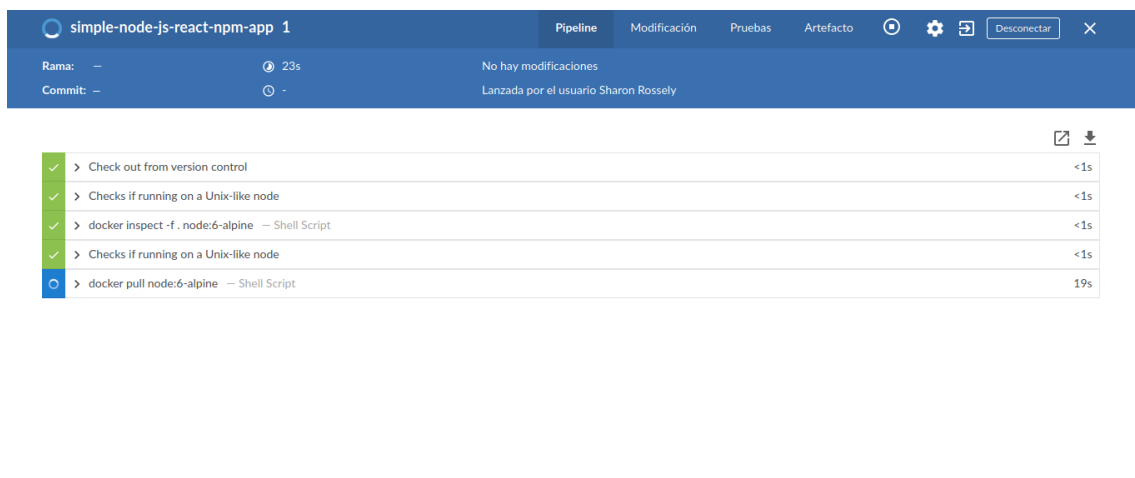


Figura 13: Proyecto en ejecución

Luego, se da la ejecución de la etapa Build (definida en *Jenkinsfile*) en el contenedor Node. Durante este tiempo, npm descarga muchas dependencias necesarias para ejecutar su aplicación Node.js y React, que finalmente se almacenarán en el `node_modules` directorio del espacio de trabajo (dentro del directorio de inicio de *Jenkins*).

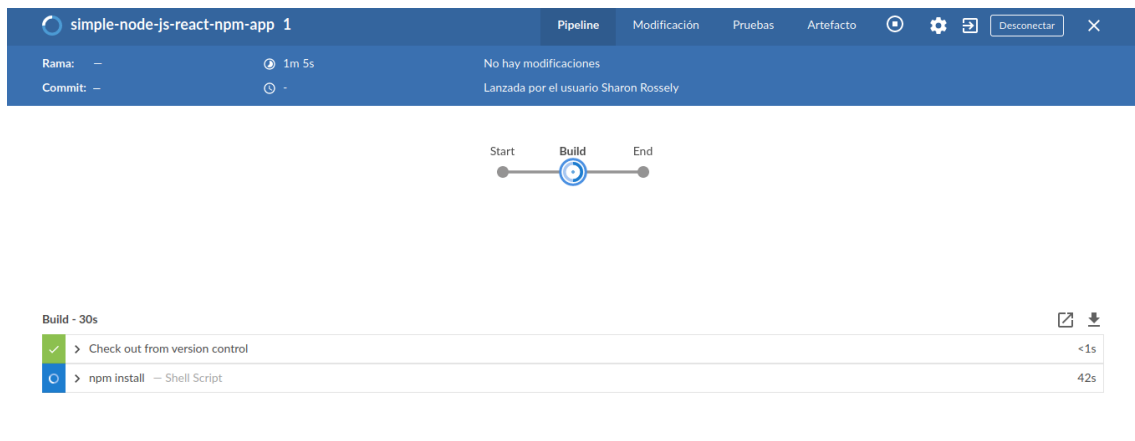


Figura 14: Construcción de proyecto Pipeline

La interfaz de Blue Ocean se vuelve verde si Jenkins construyó su aplicación Node.js y React correctamente.

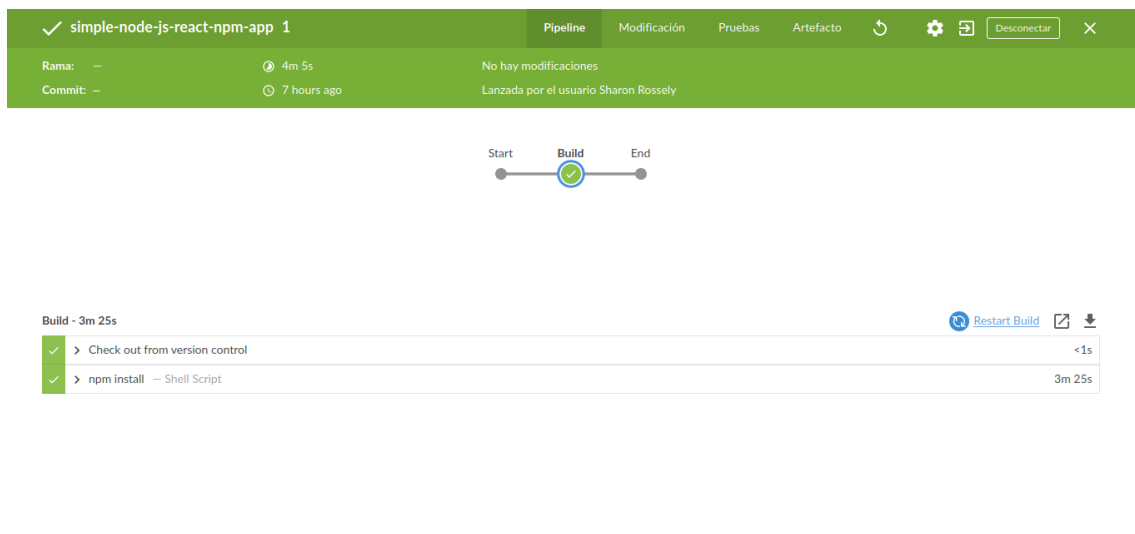


Figura 15: Construcción exitosa

Hacemos click en la X de la parte superior derecha para volver a la interfaz principal de Blue Ocean.

3.5. Paso 5: Agregando una etapa de evaluación a Pipeline

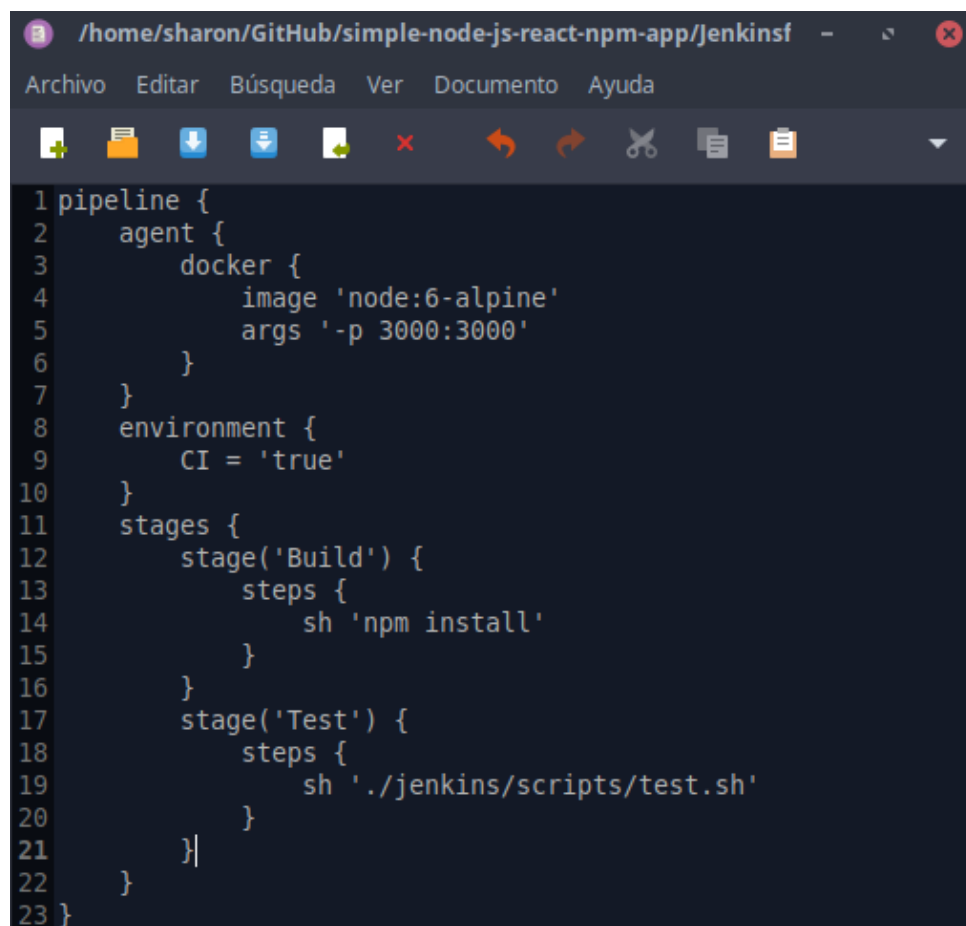
Copiamos y pegamos la siguiente sintaxis declarativa de Pipeline inmediatamente debajo de la sección **agent** del archivo *Jenkinsfile*:

```
environment {  
    CI = 'true'  
}
```

Así como lo siguiente inmediatamente debajo de la etapa Build:

```
stage('Test') {  
    steps {  
        sh './jenkins/scripts/test.sh'  
    }  
}
```

para que termines con:

A screenshot of a code editor window titled '/home/sharon/GitHub/simple-node-js-react-npm-app/Jenkinsf'. The editor shows a Jenkinsfile script with the following content:

```
1 pipeline {  
2   agent {  
3     docker {  
4       image 'node:6-alpine'  
5       args '-p 3000:3000'  
6     }  
7   }  
8   environment {  
9     CI = 'true'  
10  }  
11  stages {  
12    stage('Build') {  
13      steps {  
14        sh 'npm install'  
15      }  
16    }  
17    stage('Test') {  
18      steps {  
19        sh './jenkins/scripts/test.sh'  
20      }  
21    }  
22  }  
23 }
```

Figura 16: Archivo Jenkinsfile

Luego, hacemos click en **Iniciar** en la parte superior izquierda, luego haga click rápidamente en el enlace ABRIR que aparece brevemente en la parte inferior derecha para ver a *Jenkins* ejecutando el proyecto de Pipeline modificado.

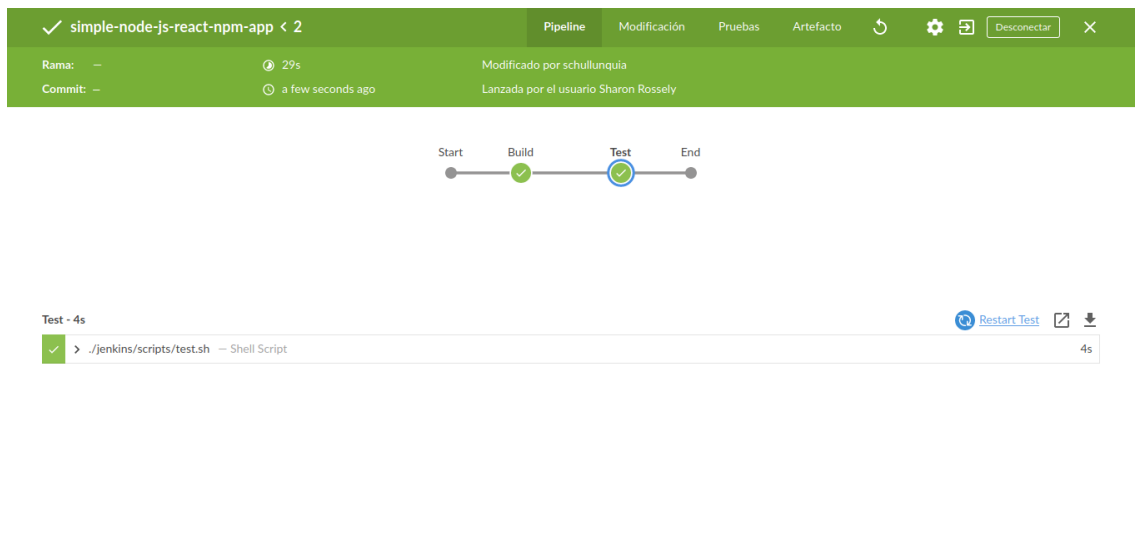


Figura 17: Construcción exitosa

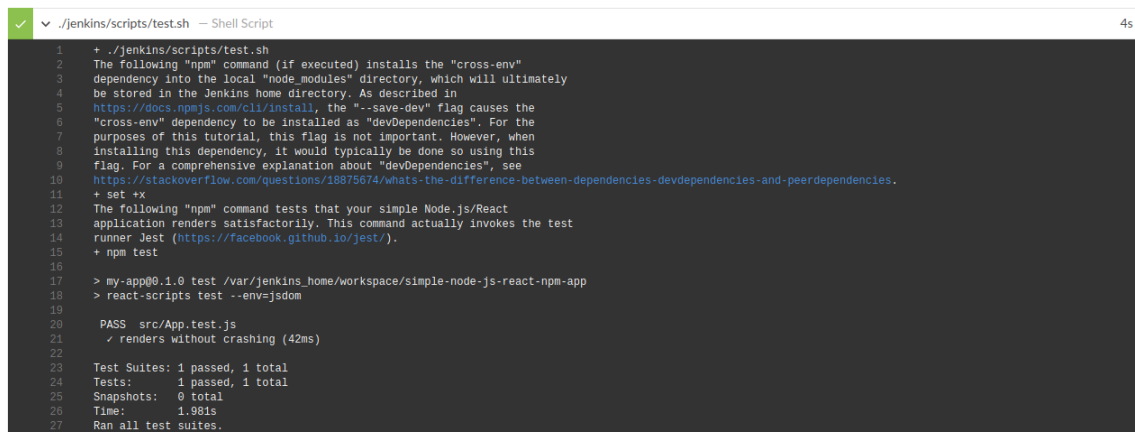
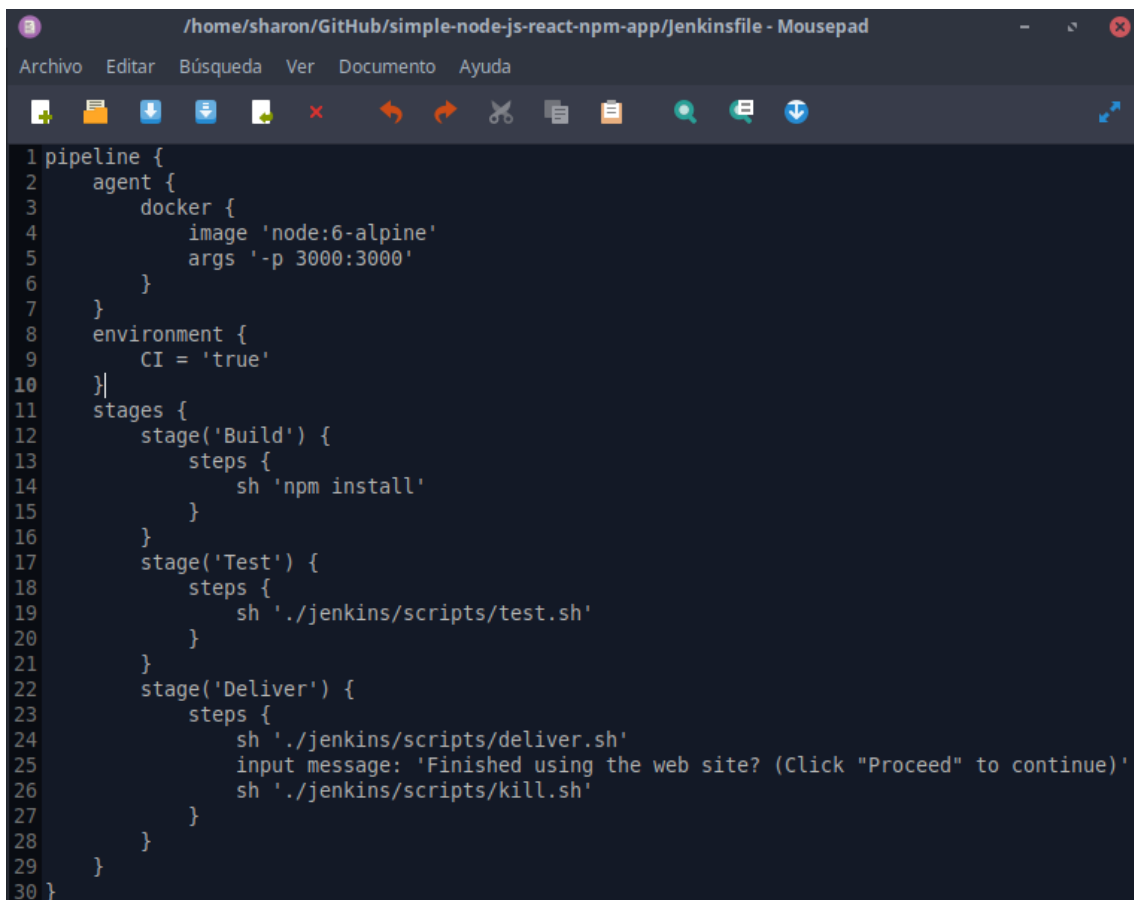


Figura 18: Shell script

3.6. Paso 6: Agregando la etapa final Deliver a Pipeline

Copiamos y pegamos la siguiente sintaxis declarativa de Pipeline inmediatamente debajo de stage Test de *Jenkinsfile*:

```
stage('Deliver') {  
  steps {  
    sh './jenkins/scripts/deliver.sh'  
    input message: 'Finished using the web site? (Click "Proceed"  
    to continue)'  
    sh './jenkins/scripts/kill.sh'  
  }  
}
```

A screenshot of a text editor window titled '/home/sharon/GitHub/simple-node-js-react-npm-app/Jenkinsfile - Mousepad'. The editor shows a Jenkinsfile with the following content:

```
1 pipeline {  
2   agent {  
3     docker {  
4       image 'node:6-alpine'  
5       args '-p 3000:3000'  
6     }  
7   }  
8   environment {  
9     CI = 'true'  
10  }  
11  stages {  
12    stage('Build') {  
13      steps {  
14        sh 'npm install'  
15      }  
16    }  
17    stage('Test') {  
18      steps {  
19        sh './jenkins/scripts/test.sh'  
20      }  
21    }  
22    stage('Deliver') {  
23      steps {  
24        sh './jenkins/scripts/deliver.sh'  
25        input message: 'Finished using the web site? (Click "Proceed"  
26        to continue)'  
27        sh './jenkins/scripts/kill.sh'  
28      }  
29    }  
30  }  
}
```

Figura 19: Archivo Jenkinsfile

Guarde los cambios en *Jenkinsfile* y hacemos un commit en **simple-node-js-react-npm-app**, repositorio local de Git. Para eso, dentro del `simple-node-js-react-npm-app` directorio, ejecutamos los comandos:

```
git stage .
```

,luego

```
git commit -m "Add 'Deliver' stage"
```

Volvemos a *Jenkins* y ejecutamos nuevamente el proyecto Pipeline

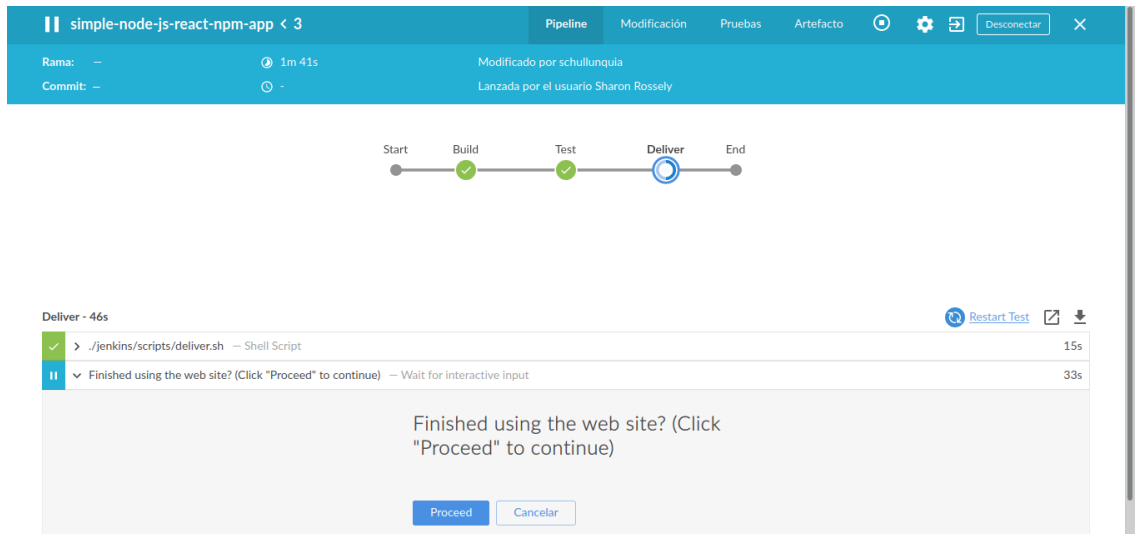


Figura 20: Proyecto en ejecución

Nos aseguramos de estar viendo la etapa "Deliver", luego hacemos click en el `./jenkins/scripts/deliver.sh` para expandir su contenido y nos desplazamos hacia abajo hasta que veamos el enlace <http://localhost:3000>.

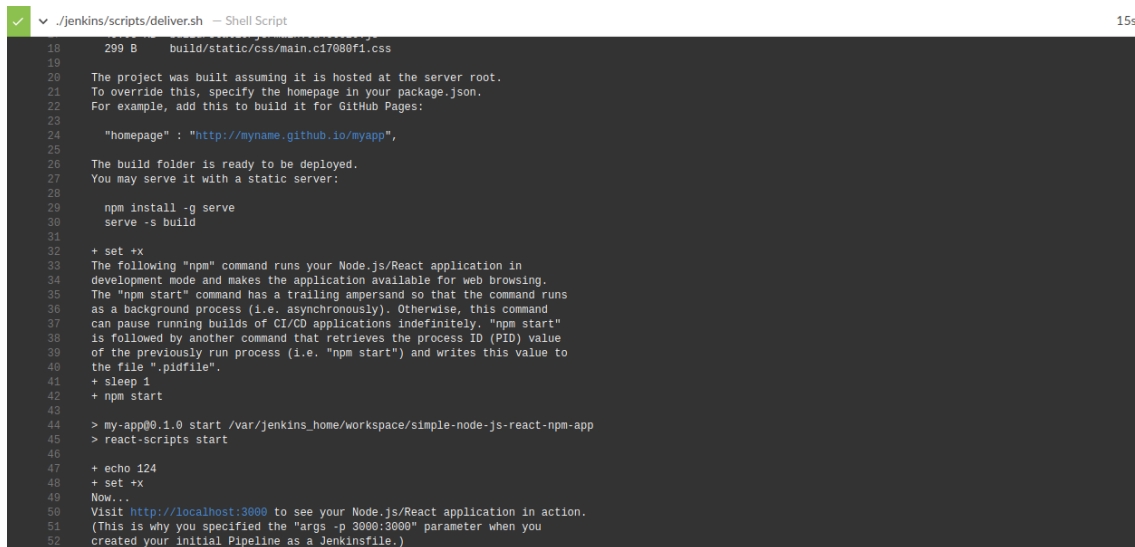


Figura 21: Shell script

Hacemos click en el enlace <http://localhost:3000> para ver su aplicación Node.js y React ejecutándose (en modo de desarrollo) en una nueva pestaña del navegador web. Debería verse así:



Figura 22: Aplicación en ejecución

Cuando hayamos terminado de ver la página/sitio, hacemos click en el botón **Proceed** para completar la ejecución de Pipeline.

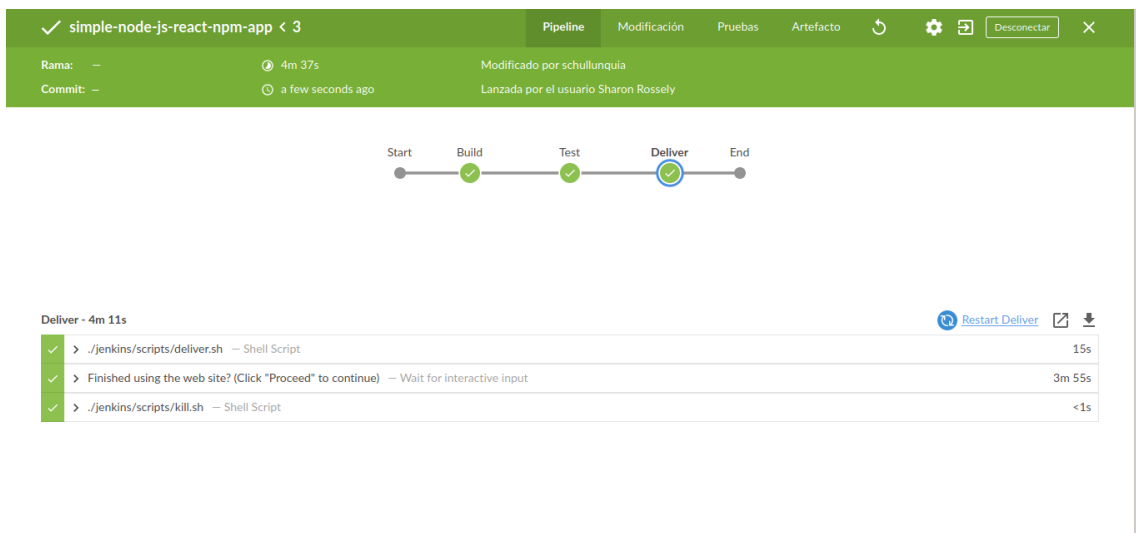
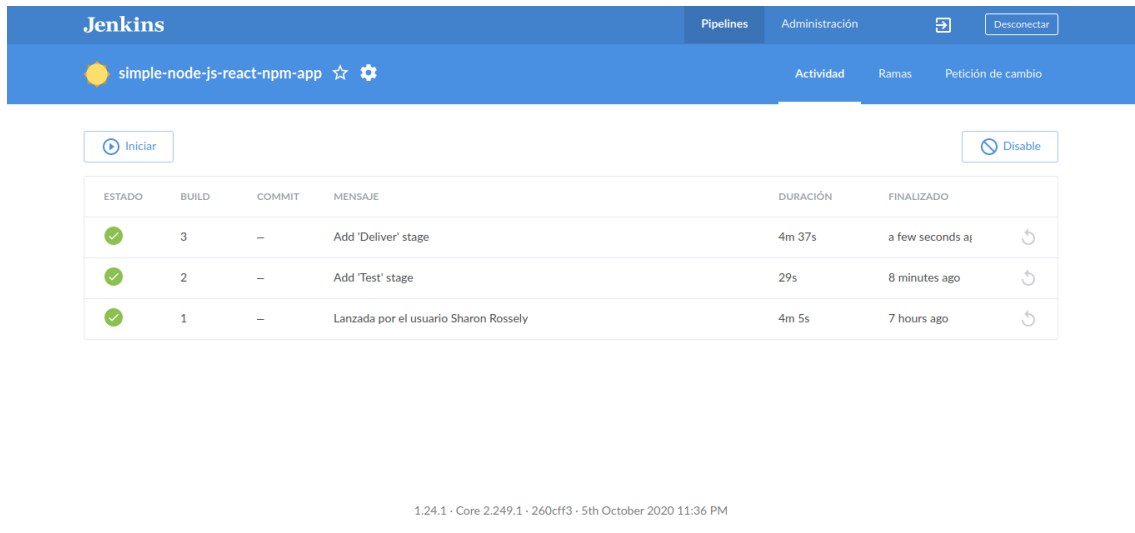


Figura 23: Ejecución exitosa

Hacemos click en la X en la parte superior derecha para volver a la interfaz principal de Blue Ocean, y podemos observar las anteriores ejecuciones de Pipeline en orden cronológico inverso.



The screenshot shows the Jenkins web interface for a pipeline named 'simple-node-js-react-npm-app'. The interface includes a top navigation bar with 'Pipelines' and 'Administración' tabs, and a 'Desconectar' button. Below the navigation bar, there are buttons for 'Iniciar' and 'Disable'. The main content area displays a table of pipeline runs, ordered from newest to oldest (reverse chronological order). The table has columns for 'ESTADO', 'BUILD', 'COMMIT', 'MENSAJE', 'DURACIÓN', and 'FINALIZADO'. Three runs are visible, all with a green checkmark indicating success. The first run (BUILD 3) was completed 'a few seconds ago', the second (BUILD 2) '8 minutes ago', and the third (BUILD 1) '7 hours ago'. At the bottom of the interface, the Jenkins version '1.24.1' and core version '2.249.1' are displayed along with the timestamp '5th October 2020 11:36 PM'.

ESTADO	BUILD	COMMIT	MENSAJE	DURACIÓN	FINALIZADO
✓	3	—	Add 'Deliver' stage	4m 37s	a few seconds ago
✓	2	—	Add 'Test' stage	29s	8 minutes ago
✓	1	—	Lanzada por el usuario Sharon Rossely	4m 5s	7 hours ago

1.24.1 - Core 2.249.1 - 260cff3 - 5th October 2020 11:36 PM

Figura 24: Ejecuciones de Pipeline en orden cronológico inverso

Referencias

- [1] Jenkins User Documentation, *What is Jenkins?*. <https://www.jenkins.io/doc/>.
- [2] Javier Garzas, *¿Qué es Jenkins?*. <https://www.javiergarzas.com/2014/05/jenkins.html>.