



# Proceso de Desarrollo de Software

Andrés Mauricio Martínez Hincapié

---

## Actividad práctica #3: Integración con Git (Webhooks) y Pruebas Automatizadas

---

**Objetivo:** Ajustar el pipeline y comprender cómo funcionan los Webhooks

### 1. Configurar Webhook en GitHub

- A. **Exponer Jenkins al Público (Temporalmente):** Para que GitHub pueda acceder a tu Jenkins local, necesitas usar un **túnel SSH inverso** o un servicio como **ngrok**. (Si no tienes una cuenta en ngrok, ve a su sitio web y crea una cuenta):

# Descarga ngrok (<https://ngrok.com/download>)

- B. Para obtener el **token de autenticación (authtoken)** de **ngrok**, sigue estos pasos:

#### 1. Crear una cuenta en ngrok

Si no tienes una cuenta en ngrok, ve a su sitio web y crea una cuenta:

- **Visita:** <https://ngrok.com>
- Haz clic en **Sign Up** (Registrarse) si aún no tienes una cuenta. Si ya tienes una cuenta, simplemente inicia sesión.

#### 2. Iniciar sesión en tu cuenta de ngrok

Una vez que tengas una cuenta, inicia sesión en el sitio de ngrok.

- **Inicia sesión** en ngrok.com.

#### 3. Obtener tu authtoken

Después de iniciar sesión, sigue estos pasos:

1. En el **panel de control** (dashboard), haz clic en "**Your Authtoken**" en el menú lateral (o visita directamente el link para la página de authtoken). <https://dashboard.ngrok.com/get-started/your-authtoken>
2. En esta página verás un token único que ngrok te proporciona. Este es el **authtoken** que necesitas.
  - El token tiene el siguiente formato:  
`2Lu9Ab4W2g2oYzxx5Mpm2y6LPpGcvLXY8bBY8k8bfh73`  
(ejemplo).
3. **Copia** ese token, ya que lo necesitarás para agregarlo en la configuración de ngrok.

#### 4. Agregar el Authtoken en ngrok

Una vez que tengas tu authtoken, puedes agregarlo a tu configuración local de ngrok. Abre tu terminal y usa el siguiente comando:

```
ngrok config add-authtoken <tu_token_aqui>
```

Reemplaza `<tu_token_aqui>` con el authtoken que copiaste en el paso anterior.

```
bash-3.2$ ngrok config add-authtoken 2xNw9I1qFwtaw6DK7P70V1lNsBi_6dvL9AqFJNhjbmtdNxQwQ
Authtoken saved to configuration file: /Users/andresmauriciomartinezhincapie/Library/Application Support/ngrok/ngrok.yml
bash-3.2$
```

- C. Ejecuta el siguiente comando para obtener la URL:

```
./ngrok http 8080 # Asumiendo que Jenkins corre en el puerto 8080
```

Ngrok generará una URL pública como

`https://174d-161-10-72-43.ngrok-free.app`. Esta será tu **Payload URL**.

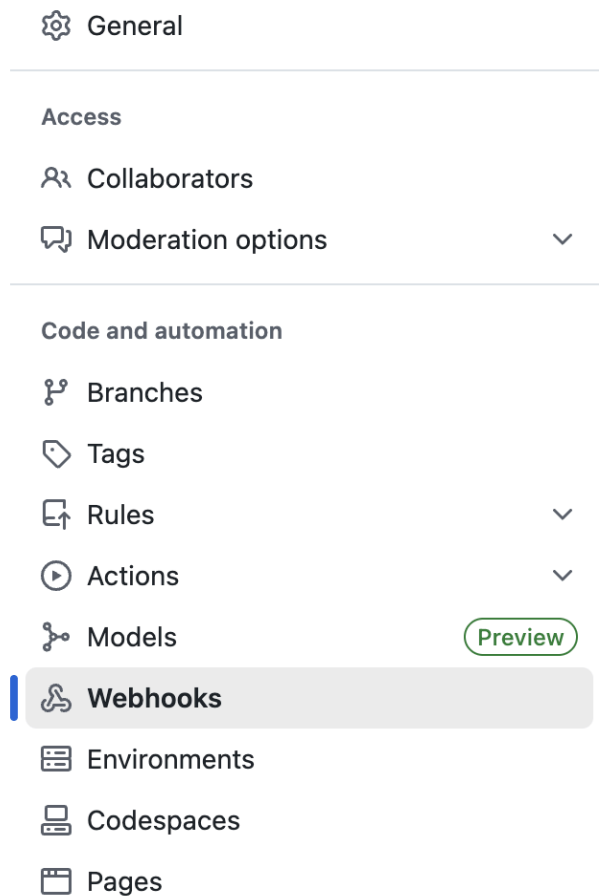
```
ngrok
Take our ngrok in production survey! https://forms.gle/aXiBFWzEA36DudFn6

Session Status      online
Account             Andres Mauricio Martinez Hincapie (Plan: Free)
Version             3.22.1
Region              United States (us)
Latency              97ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://174d-161-10-72-43.ngrok-free.app -> http://localhost:8080

Connections         ttl      opn      rt1      rt5      p50      p90
0                  0        0.00     0.00     0.00     0.00
```

- D. En tu repositorio (`amartinezh/ucp-app-react`):

- a. Ve a **Settings > Webhooks > Add webhook**.



- b. **Payload URL:** `http://<IP-o-DOMINIO-JENKINS>/github-webhook/` (ej: `https://174d-161-10-72-43.ngrok-free.app`).
- c. **Content type:** `application/json`.
- d. **Secret:** Déjalo vacío (a menos que hayas configurado autenticación en Jenkins).
- e. **Events:** Selecciona **"Just the push event"**.
- E. En caso particular de ir al menu Settings > Webhook y no tener el botón verde para agregar un hook, procedemos con la API de la siguiente manera:

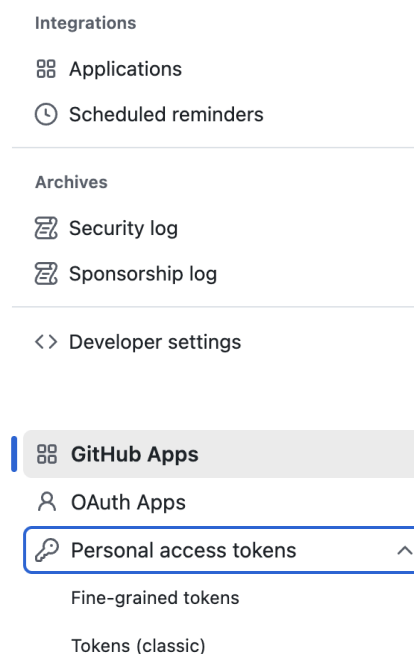
```
curl -L \
-X POST \
-H "Accept: application/vnd.github+json" \
-H "Authorization: Bearer ghp_C9URDfzm4sGDV4pdDxAJTziaOvtSIh0rFhWQ" \
-H "X-GitHub-API-Version: 2022-11-28" \
https://api.github.com/repos/amartinezh/ucp-app-react/hooks\
-d
'{"name":"web","active":true,"events":["push","pull_request"],"config":{"url":"https://174d-161-10-72-43.ngrok-free.app","content_type":"json"}}'
```

## F. Configura el servidor de gitHub en Jenkins

### Crear un Token de Acceso en GitHub

#### Paso a Paso:

1. **Inicia sesión** en tu cuenta de [GitHub](#).
2. **Ve a Settings** (<https://github.com/settings/profile>):
  - Haz clic en tu foto de perfil (esquina superior derecha) > **Settings > Developer settings > Personal access tokens > Tokens (classic)**.



3. **Genera un nuevo token:**
  - Haz clic en **Generate new token > Generate new token (classic)**.
  - **Nombre del token:** Ej. Jenkins-Access.
  - **Expiración:** Elige una fecha (ej. 30 días) o "No expiration" (no recomendado para producción).
  - **Selecciona los scopes (permisos):**
    - repo (control total sobre repos privados y públicos).
    - admin:repo\_hook (gestionar webhooks).
    - Opcional: workflow si usas GitHub Actions.

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

4. **Genera el token:**

- Haz clic en **Generate token**.
- **¡Copia el token inmediatamente!** (No se mostrará nuevamente).

---

## 2. Configurar el Token en Jenkins

### G. Configurar el Token en Jenkins

Ve a **Jenkins > Manage Jenkins > Configure System**.

Panel de Control > Administrar Jenkins > System >

GitHub

GitHub Servers ?

Add GitHub Server ▾

Busca **GitHub** en la sección *GitHub Servers*.

Agrega un servidor GitHub con:

**Name:** `GitHub` (o cualquier nombre).

**API URL:** `https://api.github.com`.

**Credentials:** Haz clic en **Add > Jenkins**.

**Kind:** *Secret text*.

**Secret:** Pega el token.

**ID:** `github-global-token`.

Marca **Manage hooks**.

Haz clic en **Test connection** para verificar.

### Jenkins Credentials Provider: Jenkins

Global credentials (unrestricted) ▼

Kind

Secret text ▼

Scope ?

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

Secret

.....

ID ?

github-global-token

Description ?

Cancel Add

Panel de Control > Administrar Jenkins > System >

### GitHub

#### GitHub Servers ?

GitHub Server ?

Name ?

GitHub

API URL ?

https://api.github.com

Credentials ?

- none - ▼

+ Add

Test connection

#### Credentials ?

github-global-token ▼

+ Add

Credentials verified for user amartinez, rate limit: 4999

Test connection

☒ Manage hooks

Avanzado ▼

## 2. Crear Pipeline en Jenkins

Ve al pipeline creado en el Taller\_2 al pipeline

✓ ☁ react-pipeline-demo 15 Min #12 15 Hor #9 33 Seg ▶

Ingresa en configurar:

Panel de Control > react-pipeline-demo >



Status

</> Changes



Construir ahora



Configurar



Borrar Pipeline



Rename



Pipeline Syntax



GitHub Hook Log

Activa: GitHub hook trigger for GITScm polling

### Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Construir tras otros proyectos ?
- ☐ Ejecutar periódicamente ?
- ☐ GitHub Branches
- ☐ GitHub Pull Requests ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Consultar repositorio (SCM) ?

Y luego actualiza tu activo de configuración en React

**Jenkinsfile** (Colócalo en la raíz de tu repositorio):

```
pipeline {
  agent any

  tools {
    nodejs 'Node_24' // Nombre definido en Global Tool Configuration
  }

  stages {
    // Etapa 1: Checkout del código desde GitHub
    stage('Checkout') {
      steps {
        git branch: 'main', url:
'https://github.com/amartinezh/ucp-app-react.git'
      }
    }
  }
}
```

```

// Etapa 2: Instalar dependencias y build del proyecto
stage('Build') {
    steps {
        sh 'npm install'
        sh 'npm run build' // Ejecuta el build de React
    }
}

// Etapa 3: Ejecutar pruebas unitarias
stage('Pruebas Unitarias') {
    steps {
        sh 'npm test -- --watchAll=false --ci --reporters=default
--reporters=jest-junit' // Genera reporte JUnit
    }
    post {
        always {
            junit 'junit.xml' // Publica reporte en Jenkins
            archiveArtifacts artifacts: 'junit.xml', allowEmptyArchive: true
        }
    }
}

// Post-actions (opcional)
post {
    always {
        emailx (
            subject: "Pipeline ${currentBuild.result}: ucp-app-react
#${env.BUILD_NUMBER}",
            body: """
                Estado: ${currentBuild.result}
                URL Build: ${env.BUILD_URL}
                Detalles de Pruebas: ${env.BUILD_URL}testReport/
            """,
            to: 'tu-email@example.com' // Reemplaza con tu email
        )
    }
}
}

```

---

### 3. Verifica la Configuración

#### 1. Instalar Plugins en Jenkins:

- **GitHub Integration:** Para webhooks.
  - **JUnit:** Para reportes de pruebas.
  - **Email Extension:** Para notificaciones.
- 

### 4. Configurar Jest para Reportes JUnit

En tu proyecto React (`package.json`):

```

"devDependencies": {
  "jest-junit": "^15.0.0"
}

```



```
}
```

Crea/edita `jest.config.js`:

```
module.exports = {  
  reporters: [  
    'default',  
    ['jest-junit', { outputDirectory: './', outputName: 'junit.xml' }]  
  ],  
};
```

## 5. Probar el Flujo

1. Haz un cambio y ejecuta:

```
git add .  
git commit -m "Trigger Jenkins"  
  
git push origin main
```

2. **Verifica en Jenkins:**

- El pipeline se iniciará automáticamente.
- Los reportes de pruebas estarán en la pestaña **"Test Result"**.

**Builds** ... 🔗

/

**Today**

✓ #13 20:35

✗

▼

### Resultados de los tests

0 fallidos (±0)

1 tests (±0)

Tardó 0.18 Seg.

[añadir descripción](#)

### Todos los tests

Paquete	Duración	Fallidos	(diferencias)	Omitir	(diferencias)	Pass	(diferencias)	Total	(diferencias)
(root)	14 Ms	0		0		1		1	