

# **ANSIBLE PARA DEV+OPS**

**DÍA 5**

# QUE VEREMOS HOY

- Integración continua / Despliegue continuo (CI / CD)
- Interacción con Jenkins
- Labs:
  - TDD + integración continua

# INTEGRACIÓN CONTINUA

- Integración Continua (CI)
- Despliegue Continuo (CD)
- La suma de todo

# INTEGRACIÓN CONTINUA

- Es el proceso de hacer merge a la rama de producción varias veces al día.
- Se basa en introducir pequeños cambios de forma rápida para evitar el “integration hell”
- Estos cambios deben estar probados de antemano para asegurar que no rompen nada

# INTEGRACIÓN CONTINUA

1

- El desarrollador toma una copia del código en el que quiere hacer cambios

2

- Crea una rama específica para su cambio

3

- Implementa el cambio

4

- Prueba su cambio en local

5

- Sube su cambio a la rama de integración

6

- El sistema de CI valida que funciona (mediante tests automáticos) y hace merge a la rama de Producción

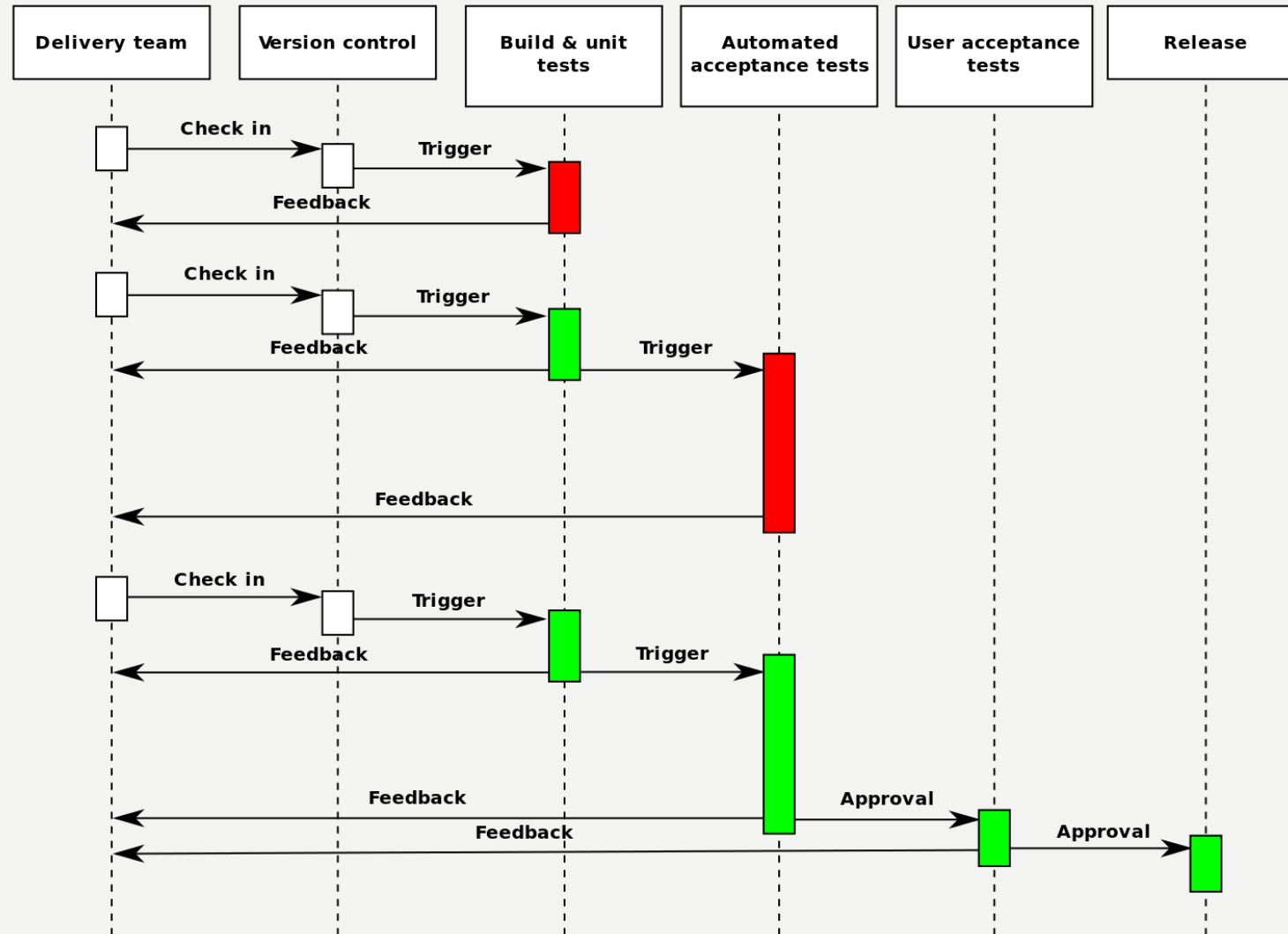
# INTEGRACIÓN CONTINUA

- Es extremadamente importante que el tiempo entre 1 y 5 sea lo más corto posible, de lo contrario pueden haber aparecido nuevos cambios en integración / producción que hagan conflicto con el desarrollo que se quiere implementar

# DESPLIEGUE CONTINUO

- Es la capacidad de poder desplegar nuevas versiones de un desarrollo en cualquier momento
- Requiere un proceso de despliegue repetible y confiable para ser implementado

# DESPLIEGUE CONTINUO





# + Y – DE CI/CD

- +
  - Se reduce el tiempo de salida al mercado
  - Se construye el producto que se necesita (cada funcionalidad es una petición y cada petición se traduce en una parte de código individual)
  - Ahorro de tiempo gracias a la automatización
  - Mejora de calidad de los entregables (al usar procesos repetitivos validados)

# + Y – DE CI/CD

- -

- Algunas organizaciones no pueden tolerar cambios constantes (especialmente en algunas épocas del año)
- Algunos sectores requieren de validaciones muy burocráticas que pueden entorpecer el proceso
- La falta de automatización en los tests impide la existencia de CI/CD
- Diferencia entre entornos: CI/CD requiere que todos los entornos sean iguales para evitar que problemas no detectados acaben en producción
- Tests no automatizables: si existen tests no automatizables habrá que integrar validación humana en el pipeline , lo que retrasará el proceso

# INTERACCIÓN CON JENKINS

- Introducción a Jenkins
- Tipos de Jobs
- Pipelines de Jenkins

# INTRODUCCIÓN A JENKINS

- Sistema de CI/CD escrito en Java
- Muy extendido
- Con gran cantidad de plugins
- Funciona definiendo jobs que pueden realizar tareas variadas
- Las tareas se ejecutan en Build Servers, que ejecutan un agente de Jenkins



# TIPOS DE JOB

- Freestyle
- Maven Project
- Pipeline
- External job
- Multi conf project
- Multi branch pipeline

# PIPELINES DE JENKINS

- Jobs definibles programáticamente con una DSL en Groovy  
<https://jenkins.io/doc/pipeline/steps/>
- Todos los pasos que se pueden hacer en jobs freestyle de Jenkins se pueden implementar en pipelines
- Ciertos plugins pueden extender la DSL
- Permiten definir procesos de CI / CD de forma programática
- Permiten pasos en paralelo

# POR SI SOBRA TIEMPO...

- Alternativas a Jenkins
  - TravisCI
  - GitlabCI
  - AWS CodePipeline
- Alternativas a Github
  - Gitlab
  - gogs.io
- El CI/CD en el mundo del IaaS

# LABS

- Instalación de Jenkins
- Integración con github
- Preparación de Jenkins para poder ejecutar nuestros comandos de test
- Desarrollo del pipeline de CI/CD para nuestros playbooks
- Prueba

No subestimes este lab ;)

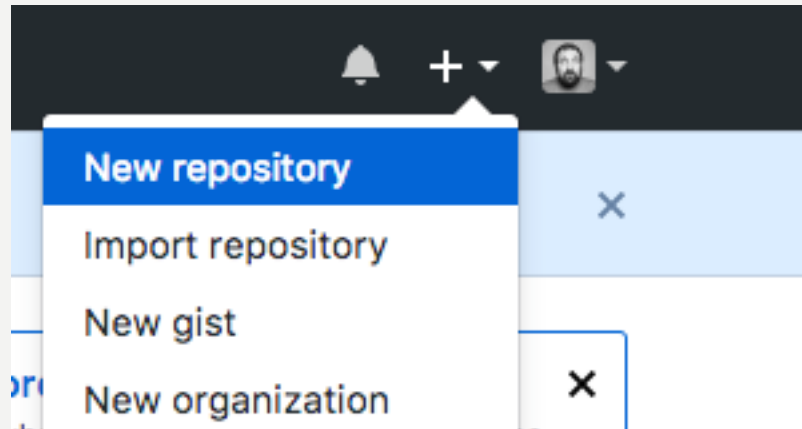


# OBJETIVO DEL LAB

- Tener un repositorio en github para nuestro proyecto tal que
  - Tenga un Jenkinsfile que lance kitchen test en Jenkins
  - Dispare ese Jenkinsfile de forma automática en Jenkins cuando se genera una pull request

# PASO 1: CREAR REPO EN GITHUB

- Creamos repositorio en github que se llame “apache-webpage”



# PASO 1: CREAR REPO EN GITHUB


- Creamos repositorio en github que se llame “apache-webpage”

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name


 warp3r ▾

 / 


apache-webpage

Great repository names are short and memorable. Need inspiration? How about [urban-eureka](#).

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.


☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

Create repository

# PASO 2: PASAR CÓDIGO A GITHUB

- En la máquina de control, desde la carpeta /root

```
[root@localhost ~]# git clone http://gitlab.teradisk.net/trainings/apache-webpage.git
Cloning into 'apache-webpage'...
remote: Counting objects: 55, done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 55 (delta 21), reused 0 (delta 0)
Unpacking objects: 100% (55/55), done.
```

# PASO 2: PASAR CÓDIGO A GITHUB

- Quitamos el origen del gitlab

```
[root@localhost ~]# cd apache-webpage/  
[root@localhost apache-webpage]# git remote -v  
origin  http://gitlab.teradisk.net/trainings/apache-webpage.git (fetch)  
origin  http://gitlab.teradisk.net/trainings/apache-webpage.git (push)  
[root@localhost apache-webpage]# git remote remove origin
```

# PASO 2: PASAR CÓDIGO A GITHUB

- Añadimos el origen de github (ojo, cada uno tiene el suyo)

```
[root@localhost apache-webpage]# git remote add origin https://github.com/warp3r/apache-webpage.git
[root@localhost apache-webpage]# git push -u origin master
Username for 'https://github.com': warp3r
Password for 'https://warp3r@github.com':
Counting objects: 55, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (43/43), done.
Writing objects: 100% (55/55), 8.50 KiB | 0 bytes/s, done.
Total 55 (delta 21), reused 0 (delta 0)
remote: Resolving deltas: 100% (21/21), done.
To https://github.com/warp3r/apache-webpage.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

# PASO 2: PASAR CÓDIGO A GITHUB

- Validamos en github que vemos el código

The screenshot shows the GitHub interface for the repository 'warp3r / apache-webpage'. At the top, there are buttons for 'Unwatch', 'Star' (0), and 'Fork' (0). Below this is a navigation bar with links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Insights', and 'Settings'. The main content area shows 'No description, website, or topics provided.' with an 'Edit' button and a link to 'Add topics'. Below this, statistics show '11 commits', '1 branch', '0 releases', and '0 contributors'. A bar at the top of the file list shows 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The file list itself shows a merge by 'Jordi Molina' and a table of files and folders with their commit history.

File/Folder	Commit	Time
defaults	commit inicial	14 hours ago
files	commit inicial	14 hours ago
handlers	commit inicial	14 hours ago
meta	commit inicial	14 hours ago
tasks	commit inicial	14 hours ago
templates	commit inicial	14 hours ago
test/integration/default	commit inicial	14 hours ago
vars	commit inicial	14 hours ago
.gitignore	commit inicial	14 hours ago
.kitchen.yml	commit inicial	14 hours ago
Gemfile	commit inicial	14 hours ago
Gemfile.lock	commit inicial	14 hours ago
Jenkinsfile	test pr	11 hours ago
README.md	merge	11 hours ago

# PASO 3: CONFIGURAR JENKINS

- Lanzamos el playbook de creación de máquina de jenkins, desde la carpeta root del controller (OJO CADA UNO SU NOMBRE):
  - `cd /root`
  - `git clone http://gitlab.teradisk.net/trainings/laboratorio-jenkins.git`
  - `cd laboratorio-jenkins`
  - `ansible-playbook crear-jenkins-aws.yml -e "NOMBRE_ALUMNO=Jordi"`



# PASO 3: CONFIGURAR JENKINS

- La ejecución actual muestra la IP del servidor jenkins

```
PLAY RECAP *****
34.241.90.202      : ok=18    changed=16    unreachable=0    failed=0
localhost         : ok=8     changed=3     unreachable=0    failed=0
```

- Abrimos en un navegador web la url `https://<ip>`

# PASO 3: CONFIGURAR JENKINS

- Hay que entrar por ssh a la máquina jenkins y ver el contenido del fichero que indica

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

**Administrator password**

# PASO 3: CONFIGURAR JENKINS

- Hay que entrar por ssh a la máquina jenkins y ver el contenido del fichero que indica

```
[root@localhost ~]# ssh -i /root/curso-itnow.pem centos@34.241.90.202
Last login: Wed Oct 25 11:43:27 2017 from 213.143.49.176
[centos@ip-10-0-0-125 ~]$ sudo su -
Last login: Sat Oct 14 17:38:20 UTC 2017 on pts/0
[root@ip-10-0-0-125 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
XXXXXXXXXX
```

# PASO 3: CONFIGURAR JENKINS

- Seleccionamos “Install suggested pluggins”

## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

# PASO 3: CONFIGURAR JENKINS

- Definimos el user administrador.
- Importante: RECORDADLO, nos hará falta
- Hacer click en SAVE AND FINISH

## Create First Admin User

Username:	<input type="text" value="jordi"/>
Password:	<input type="password" value="....."/>
Confirm password:	<input type="password" value="....."/>
Full name:	<input type="text" value="Jordi"/>
E-mail address:	<input type="text" value="jordi@teradisk.com"/>

# PASO 3: CONFIGURAR JENKINS

- Click en start using jenkins

## Jenkins is ready!

Your Jenkins setup is complete.

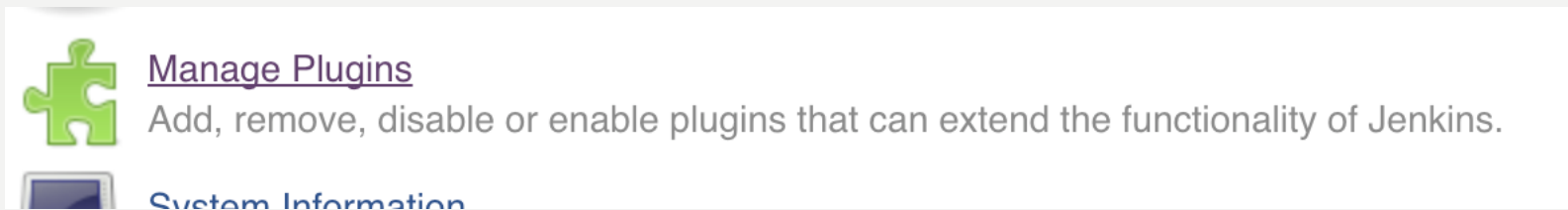
Start using Jenkins

# PASO 3: CONFIGURAR JENKINS

- Click en Manage Jenkins ->

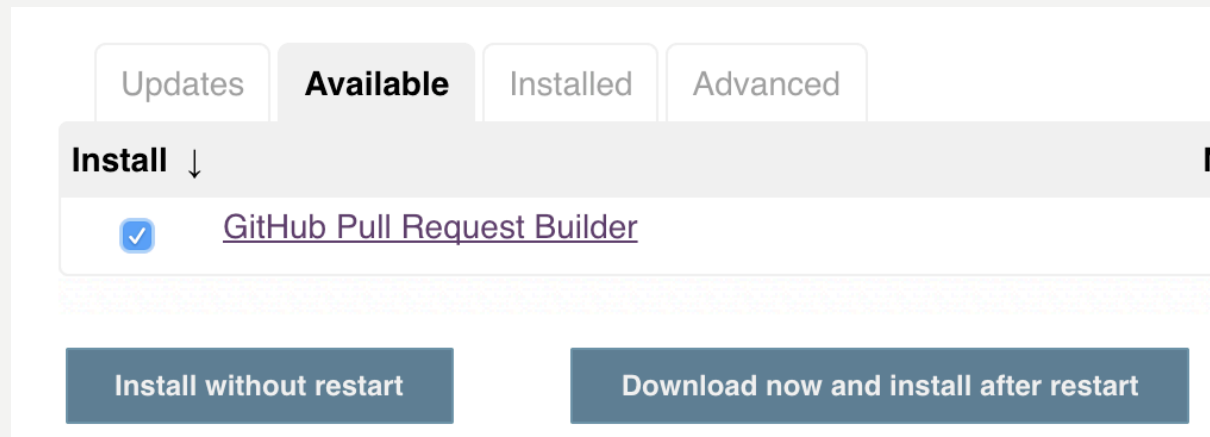


- Click en Manage Plugins ->



# PASO 3: CONFIGURAR JENKINS

- Ir a la pestaña available
- En el filtro escribir “github pull request builder”
- Marcar el plugin, y seleccionar ”Download now and restart after install”





# PASO 3: CONFIGURAR JENKINS

- Cuando aparezca lo siguiente, hacer click en la checkbox “Restart jenkins when installation is complete...”

SSH Agent Plugin


 Downloaded Successfully. Will be activated during the next boot

GitHub Pull Request Builder

 Downloaded Successfully. Will be activated during the next boot

 [Go back to the top page](#)

(you can start using the installed plugins right away)

 ☐ Restart Jenkins when installation is complete and no jobs are running

# PASO 3: CONFIGURAR JENKINS

- Autenticarse

A screenshot of the Jenkins login interface. It features a white background with a light gray border. The 'User' field contains the text 'jordi'. The 'Password' field is highlighted with a blue border and contains ten dots, indicating a masked password. Below the password field is an unchecked checkbox labeled 'Remember me on this computer'. At the bottom is a blue 'log in' button.

User: jordi

Password: .....|

☐ Remember me on this computer

log in

# PASO 3: CONFIGURAR JENKINS

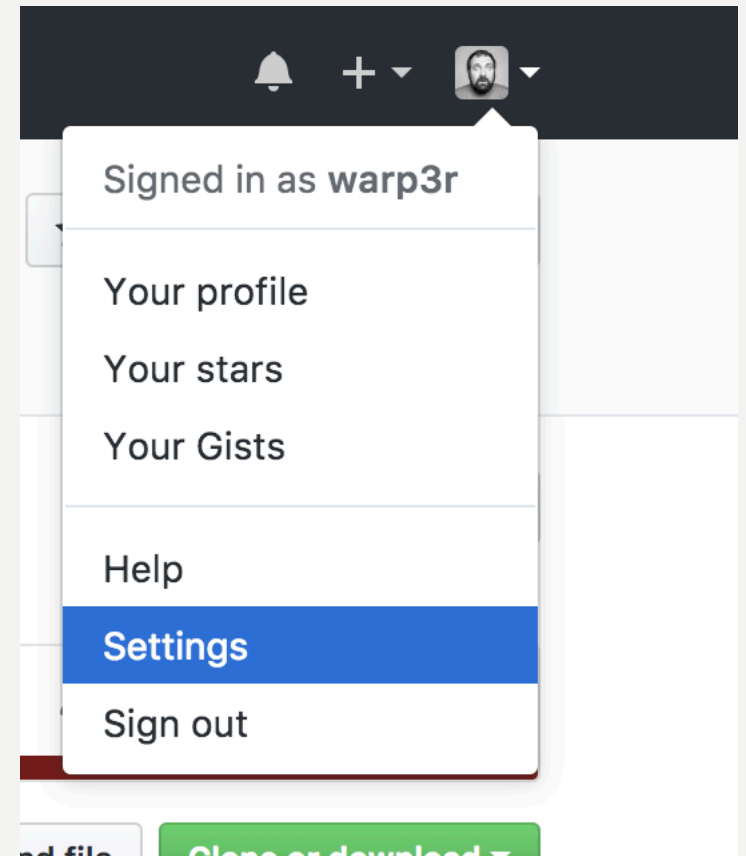
- Back to dashboard



Back to Dashboard

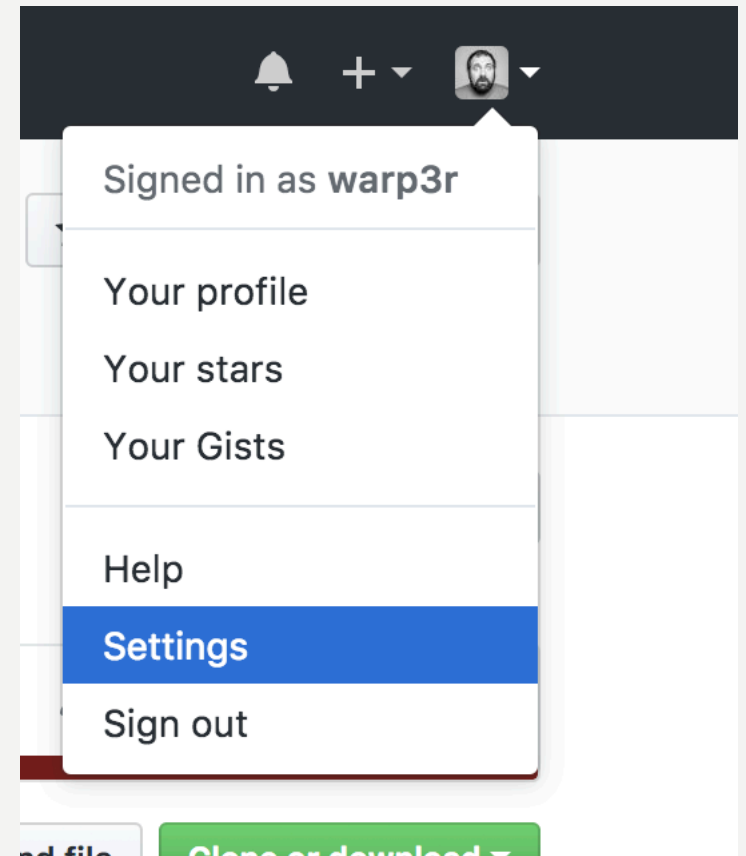
# PASO 4: GENERAR TOKEN DE GITHUB

- En GitHub, en nuestro avatar, seleccionamos Settings



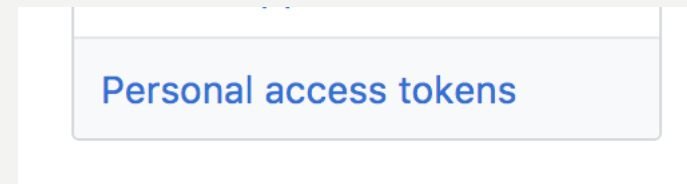
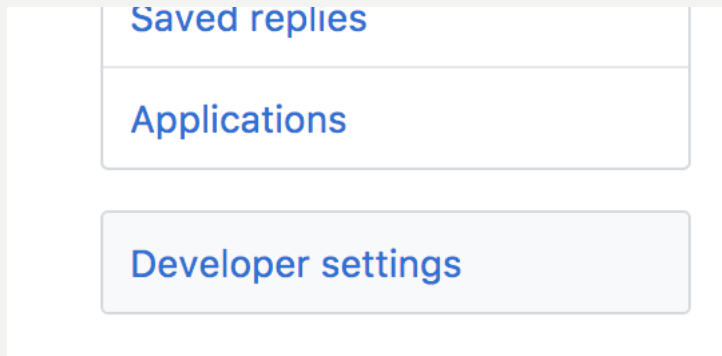
# PASO 4: GENERAR TOKEN DE GITHUB

- En GitHub, en nuestro avatar, seleccionamos Settings




# PASO 4: GENERAR TOKEN DE GITHUB

- Hacemos click en Developer Settings y Personal Access Tokens



# PASO 4: GENERAR TOKEN DE GITHUB

- Hacemos click en Generate New Token



Generate new token

# PASO 4: GENERAR TOKEN DE GITHUB

- Seleccionamos “Repo” y hacemos click en “Generate Token”
- Como nombre indicamos “build\_token”

## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Token description

What's this token for?

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	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 type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams
<input type="checkbox"/> write:org	Read and write org and team membership



# PASO 4: GENERAR TOKEN DE GITHUB


- Copiamos el Token en un sitio fácil de acceder

## Personal access tokens

Generate new tokenRevoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 18afd6d5d676a15b16e1227eba634b18532969e0 	EditDelete
brew osx — public access	Last used within the last 11 months EditDelete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS.

# PASO 5: CONFIGURAR INTEGRACIÓN JENKINS <-> GITHUB

- En Jenkins
  - Manage Jenkins ->
  - Configure System ->



[Manage Jenkins](#)

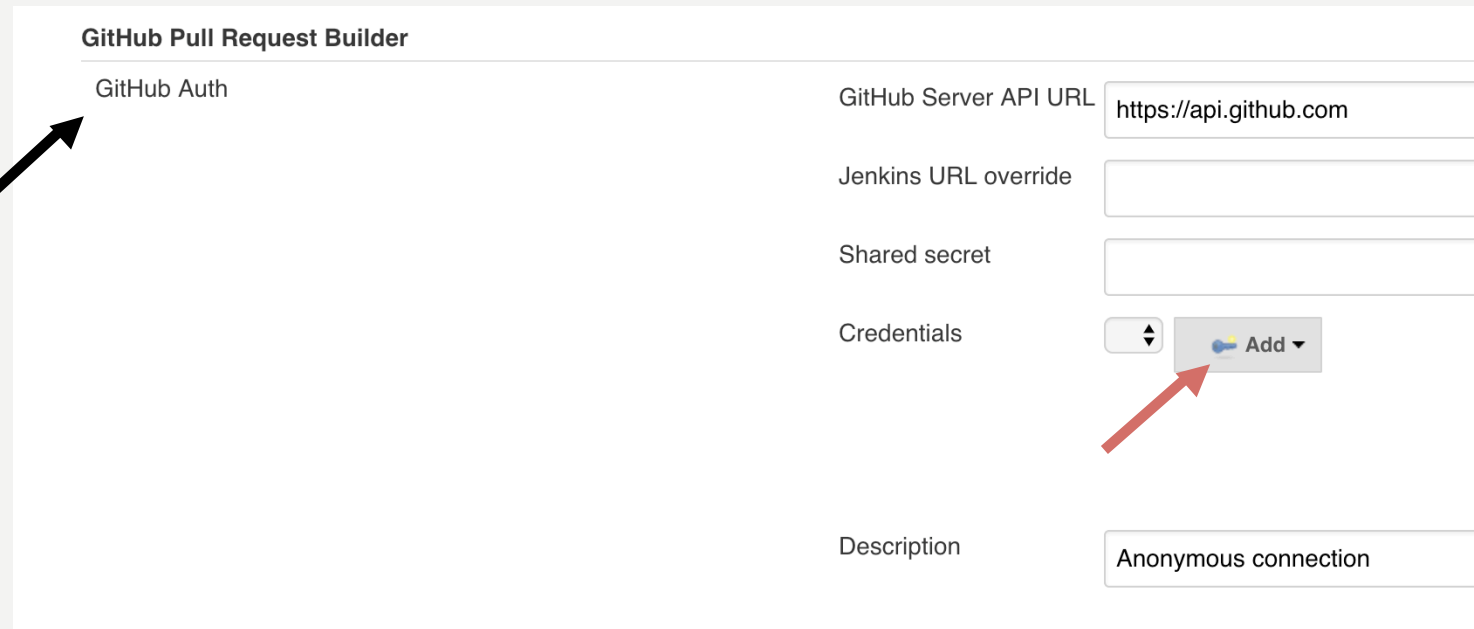


[Configure System](#)

Configure global settings and paths.

# PASO 5: CONFIGURAR INTEGRACIÓN JENKINS <-> GITHUB

- Buscamos



The screenshot shows the 'GitHub Pull Request Builder' configuration page in Jenkins. On the left, under the 'GitHub Auth' section, there is a list of configured authentication methods. A black arrow points to this section. On the right, there are several input fields: 'GitHub Server API URL' (pre-filled with 'https://api.github.com'), 'Jenkins URL override', 'Shared secret', and 'Credentials'. The 'Credentials' section has a dropdown menu with an 'Add' button. A red arrow points to this 'Add' button. At the bottom, there is a 'Description' field with the text 'Anonymous connection'.

- Hacemos click en el Add -> Jenkins

# PASO 5: CONFIGURAR INTEGRACIÓN JENKINS <-> GITHUB

- Kind: Secret Text
- Ponemos el token en el campo SECRET y damos a ADD



The screenshot shows the 'Add Credentials' dialog in Jenkins. It has a title bar with a key icon and the text 'Add Credentials'. Below the title bar are several input fields: 'Domain' with a dropdown menu showing 'Global credentials (unrestricted)', 'Kind' with a dropdown menu showing 'Secret text', 'Scope' with a dropdown menu showing 'Global (Jenkins, nodes, items, all child items, etc)', 'Secret' with a text input field containing a series of dots, 'ID' with a text input field containing 'github\_token', and 'Description' with an empty text input field. At the bottom of the dialog are two buttons: 'Add' and 'Cancel'.

**Add Credentials**

Domain: Global credentials (unrestricted)

Kind: Secret text

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

Secret: .....

ID: github\_token

Description:

Add Cancel

# PASO 5: CONFIGURAR INTEGRACIÓN JENKINS <-> GITHUB

- Hacemos click en test credentials
- Activamos “Test basic Connection to Github”
- Hacemos click en Connect to Api, debería decir “Connected”

☒ Test basic connection to GitHub

Connect to API

- Hacemos click en Save

# PASO 6: CONFIGURAR JOB


- Click en New Item




# PASO 6: CONFIGURAR JOB

- Rellenamos nombre, seleccionamos pipeline y damos click a OK

**Enter an item name**  
  
» Required field

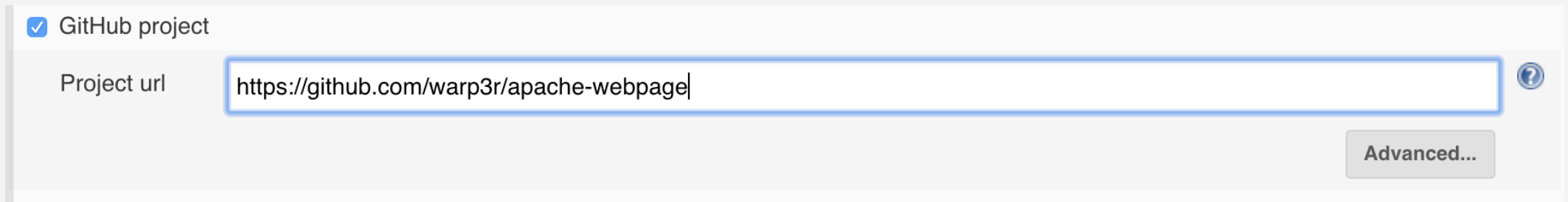
**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**  
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

OK

# PASO 6: CONFIGURAR JOB

- Marcamos Github Project y ponemos la URL de nuestro repo



A screenshot of a web form for configuring a job. At the top, there is a checkbox labeled "GitHub project" which is checked. Below this, there is a text input field labeled "Project url" containing the text "https://github.com/warp3r/apache-webpage". To the right of the input field is a small blue circular help icon with a question mark. At the bottom right of the form is a button labeled "Advanced...".

☒ GitHub project

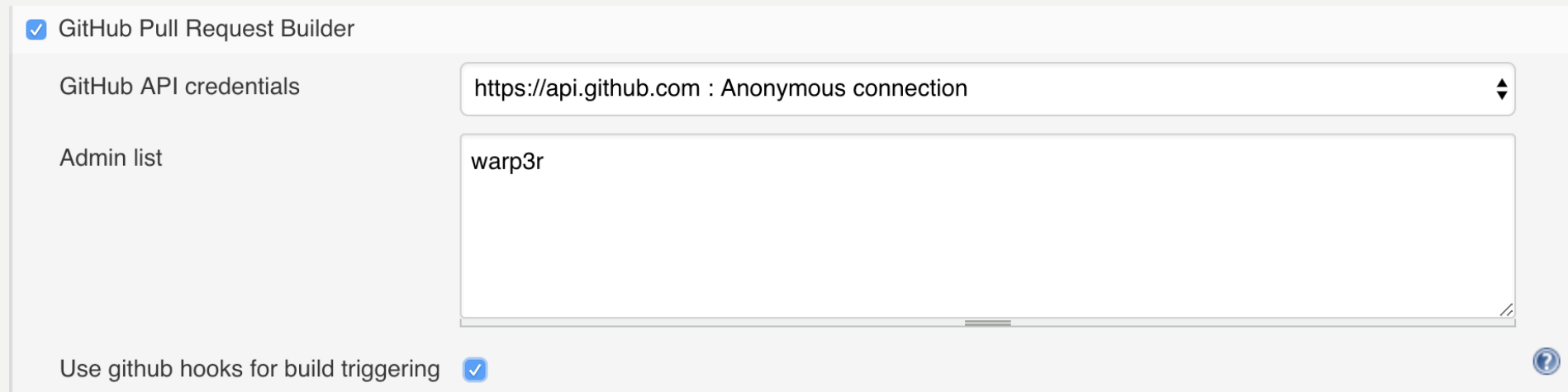
Project url

Advanced...



# PASO 6: CONFIGURAR JOB

- En Build Triggers, marcamos “Github Pull Request Builder” y rellenamos
  - En admin nuestro user de github
  - Activamos el flag “Use github hooks for build triggering”



The screenshot shows a configuration panel for the 'GitHub Pull Request Builder'. At the top, there is a checked checkbox labeled 'GitHub Pull Request Builder'. Below this, there are three main fields: 'GitHub API credentials' with a dropdown menu showing 'https://api.github.com : Anonymous connection'; 'Admin list' with a text area containing 'warp3r'; and 'Use github hooks for build triggering' with a checked checkbox. A help icon (question mark) is located at the bottom right of the panel.

☒ GitHub Pull Request Builder

GitHub API credentials:

Admin list:

Use github hooks for build triggering ☒

# PASO 6: CONFIGURAR JOB

- En Pipeline
  - Definition -> Pipeline Script from SCM
  - SCM -> Git
  - Repository URL -> URL del repo (acabada en .git)
    - Advanced
      - Refspec: +refs/pull/\*:refs/remotes/origin/pr/\*
    - Branch: \${sha1}

# PASO 6: CONFIGURAR JOB

- En Pipeline

**Pipeline**

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/warp3r/

Credentials

- none -

Add

Name

Refspec

+refs/pull/\*:refs/remotes/o

Add Repository

Branches to build

Branch Specifier (blank for 'any')

\$(sha1)

Add

Branch

# PASO 6: CONFIGURAR JOB

- Desmarcamos “Lightweight checkout”

Lightweight checkout ☐

- Click Save

Save

# PASO 7: CREAR PR

- En controller, directorio /root/apache-webpage

```
[root@localhost apache-webpage]# git checkout -b nuevo-readme  
Switched to a new branch 'nuevo-readme'
```

# PASO 7: CREAR PR

- Añadimos una línea al fichero README.md y enviamos a github

```
[root@localhost apache-webpage]# vi README.md
[root@localhost apache-webpage]# git add README.md
[root@localhost apache-webpage]# git commit -m "Nueva linea en readme"
[nuevo-readme 46c6a0d] Nueva linea en readme
```

# PASO 7: CREAR PR


- Añadimos una línea al fichero README.md y enviamos a github

```
[root@localhost apache-webpage]# git push -u origin nuevo-readme
Username for 'https://github.com': warp3r
Password for 'https://warp3r@github.com':
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 306 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/warp3r/apache-webpage.git
 * [new branch]      nuevo-readme -> nuevo-readme
Branch nuevo-readme set up to track remote branch nuevo-readme from origin.
```

# PASO 7: CREAR PR

- Si vamos a Github, a la web de nuestro proyecto, veremos lo siguiente

Your recently pushed branches:

 **nuevo-readme** (less than a minute ago)

 **Compare & pull request**

- Hacemos click en Compare y Pull Request





# PASO 7: CREAR PR

- Rellenamos y damos a Create Pull Request

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across](#)

 base: **master** ... compare: **nuevo-readme** ✓ **Able to merge.** These branches can be automatically merged



Nueva linea en readme


Write

Preview

AA ▾ B i “ <> 🔗 ☰ ☷ ☑ ↶ @ ★

Esta linea es esencial


Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.


 Styling with Markdown is supported

Create pull request


# PASO 7: CREAR PR

- Si todo ha ido bien, habrá lanzado el check en GitHub




**Some checks haven't completed yet**[Hide all checks](#)

1 pending check

 **default** — Build started sha1 is merged.

[Details](#)



**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

Merge pull request

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# PASO 7: CREAR PR

- Y en Jenkins

## Stage View

Average stage times:

