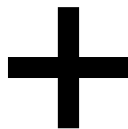


# Jenkins Pipeline



<https://www.youtube.com/watch?v=vATPHy0GXaE>

PEDRO AMADOR RODRÍGUEZ - QA Manager @gigigoapps

**gigigo** pedro.amador@gigigo.com

 @gigigolabs @pedro\_amador\_r






<https://es.linkedin.com/in/pamador>








pedroamador.rodriguez@gmail.com

## 1 - Al principio...

- Programadores  x 1
- Servidores  x 1
- Aplicaciones  x 1

- 
- **No SCM:** el código, en el PC del programador.
  - **No pruebas:** se programa y se prueba “a ojo”.
  - **No automatización:** se sube todo en un zip por FTP. O se toca a mano.




## 2 - La empresa crece

- Desarrolladores             x 2 iOS,  x 2 Android,  x 1 Servidor
- Servidores                     x 5
- Aplicaciones                 x 10

- 
- **Uso de SVN:** el código se mantiene en un repositorio central.
  - **No pruebas:** se programa, se prueba “a ojo” y se sube al repositorio (svn ci).
  - **No automatización:** se sube todo en un zip por FTP. O se toca a mano.

...y descubrimos: SVN en el servidor. Despliegue con “svn co” y primeros scripts.

## 3 - Se sigue creciendo

- Personal  x 30: iOS, Android, Backend, Infraestructura.
- Plataformas  x 50: múltiples servicios.
- Proyectos  x 20: aplicaciones grandes y complejas.

- 
- **Uso de GIT:** bitbucket, cientos de repositorios.
  - **Algunas pruebas:** TDD, Behat, phpspec.
  - **Automatización:** scripts, versionado de configuraciones.

...y descubrimos: la complejidad y diversidad de los despliegues.

## 4 - Nos adaptamos al cambio

- **Entornos:** vagrant en desarrollo, múltiples entornos (D,S,Q,L) misma config.
- **Configuración:** IaC, se versiona config con el proyecto, usamos puppet.
- **Automatización:** Fabric, evitamos repetir algunas tareas manualmente.
- **Deploys:** nueva palabra, desplegar código comienza a tener entidad propia.

## PERO

- Las medidas tomadas no parecen suficientes.
- Equipo de Infraestructura/Sistemas toma rol de Operaciones.

...y descubrimos: si aumenta el “delivery”, aumenta la presión y los fallos.

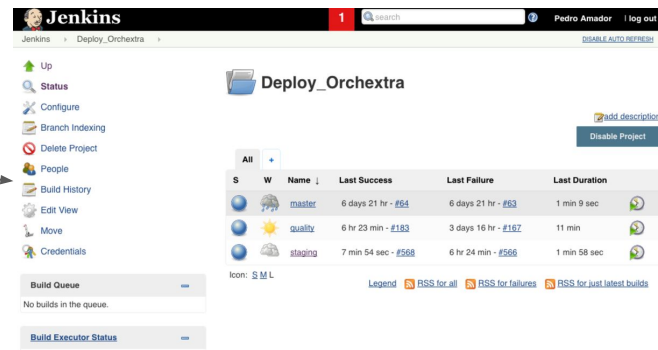
Entonces alguien se trae Jenkins en su mochila...



...y todo empieza a cambiar

## Tácticas:

- Diseño de “Jobs” de Jenkins.
- Uso de Webhooks en Bitbucket.
- Automatización de despliegues con Fabric.
- Análisis de código con SonarQube.
- Ejecución de test con cada subida.
- Cerca de 10k “builds” en poco más de 1 año



## Resultados:

- Aumenta el “delivery”, aumenta la calidad, disminuyen los errores.

# Great!

# PERO

Ah, ¿hay un pero?



¡Sólo aplicaciones de servidor!

PHP, NodeJS, aplicaciones de servidor (api, web, backends, landings)

Pero entonces,

¿Qué pasa con las aplicaciones móviles (iOS, Android)?

¿Y con las híbridas (phonegap, cordova, ionic)?

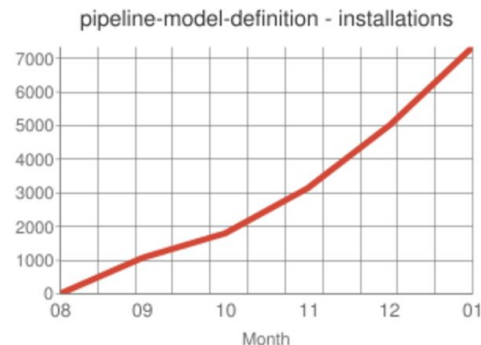
## iOS:

- Solución oficial: XCode Server.

## Android. Clonamos Jenkins de back/front y:

- Diseñamos Jobs de Jenkins.
- Ponemos webhooks en bitbucket.
- Automatizamos build de APK's (gradle) y testing (Jacoco).
- Añadimos análisis de código con SonarQube.

De los Jobs pasamos a Jenkins Pipeline  
...y de ahí a Declarative Pipeline.




## ¿Qué son los Jobs de Jenkins?



















Forma tradicional de diseñar tareas en Jenkins:

- Configurados con interface Web.
- Eliges el SCM, los “build steps”, las “post-build actions”, ...
- Se ejecutan en un nodo: checkout del código, ...

*Veremos un Job en la parte práctica*

 Build History trend ▾

x

 <a href="#">#550</a>	Feb 23, 2017 11:03 AM	
 <a href="#">#549</a>	Feb 22, 2017 3:27 PM	
 <a href="#">#548</a>	Feb 22, 2017 10:13 AM	
 <a href="#">#547</a>	Feb 21, 2017 5:28 PM	
 <a href="#">#546</a>	Feb 21, 2017 9:30 AM	
 <a href="#">#545</a>	Feb 17, 2017 10:00 AM	
 <a href="#">#544</a>	Feb 16, 2017 8:12 PM	
 <a href="#">#543</a>	Feb 14, 2017 3:50 PM	
 <a href="#">#542</a>	Feb 13, 2017 5:54 PM	

↑

↑

↓

|

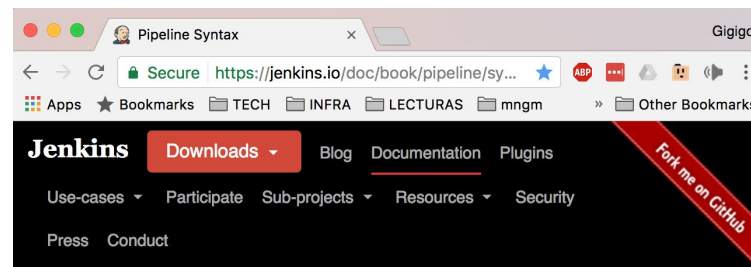
## ¿Qué es Jenkins Pipeline?

(también llamado “Scripted Pipeline”)

- DSL basado en groovy.
- Pipeline-as-Code: script como código groovy en un fichero “Jenkinsfile”.
- El script viaja con el repositorio.
- Jenkins ejecuta el script definido en el “Jenkinsfile”.

Similar a concepto Travis CI.

*Veremos un Scripted Pipeline en la parte práctica*



## Scripted Pipeline

Scripted Pipeline, like [Declarative Pipeline](#), is built on top of the underlying Pipeline sub-system. Unlike Declarative, Scripted Pipeline is effectively a general purpose DSL <sup>[2]</sup> built with [Groovy](#). Most functionality provided by the Groovy language is made available to users of Scripted Pipeline, which means it can be a very expressive and flexible tool with which one can author continuous delivery pipelines.

## Flow Control

Scripted Pipeline is serially executed from the top of a [Jenkinsfile](#) downwards, like most traditional scripts in Groovy or other languages. Providing flow control therefore rests on Groovy expressions, such as the [if/else](#) conditionals, for example:

```
Jenkinsfile (Scripted Pipeline)
node {
    stage('Example') {
        if (env.BRANCH_NAME == 'master') {
            echo 'I only execute on the master branch'
        } else {
            echo 'I execute elsewhere'
        }
    }
}
```

## ¿Qué es Declarative Pipeline?

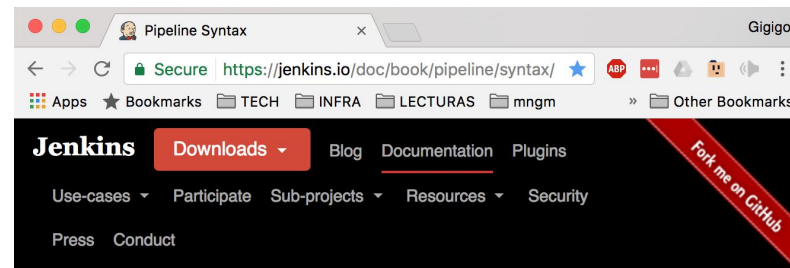
- Evolución de Jenkins Pipeline.
- Sintaxis por bloques más clara.
- DSL ampliado, específico.
- Manejo de docker incluido.

Versión 1.0 liberada el 1 de febrero de 2017

## Más información y referencia

<https://jenkins.io/doc/book/pipeline/syntax/>

*Veremos un Declarative Pipeline en la parte práctica*



### Example

```
Jenkinsfile (Declarative Pipeline)
pipeline {
  agent any
  stages {
    stage('Example') {
      steps {
        echo 'Hello World'
      }
    }
  }
  post {
    1 always {
      2 echo 'I will always say Hello again!'
    }
  }
}
```

1 Conventionally, the `post` section should be placed at the end of the Pipeline.

# workshop time

<https://bitbucket.org/pedroamador/android-testing-workshop>

<https://bitbucket.org/pedroamador/android-testing-workshop-fork>


<https://github.com/pedroamador/android-testing>



<https://github.com/pedroamador/ci-scripts>

# Jenkins Pipeline

Gracias

PEDRO AMADOR RODRÍGUEZ - QA Manager @gigigoapps

**gigigo** pedro.amador@gigigo.com  
 @gigigolabs @pedro\_amador\_r

 <https://es.linkedin.com/in/pamador>  
 pedroamador.rodriquez@gmail.com