

Grado de Tecnologías  
de Telecomunicación

Área del Proyecto:  
Administración de redes y  
sistemas operativos

Trabajo Fin de Grado



Introducción a DevOps para la mejora de los  
procesos de desarrollo con herramientas Open  
Source

**Consultor:**  
**Miguel Martín Mateo**  
**Fecha: 09-Junio-2019**

**Autor:**  
**Alberto Élez Villamarín**



Esta obra está sujeta a una licencia de Reconocimiento-  
NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>Introducción a DevOps para la mejora de los procesos de desarrollo con herramientas Open Source</i>
<b>Nombre del autor:</b>	<i>Alberto Élez Villamarín</i>
<b>Nombre del consultor/a:</b>	<i>Miguel Martín Mateo</i>
<b>Fecha de entrega:</b>	06/2019
<b>Titulación:</b>	<i>Grado de Tecnologías de Telecomunicación</i>
<b>Área del Trabajo Final:</b>	<i>Administración de redes y sistemas operativos</i>
<b>Palabras clave</b>	<i>DevOps. gestión, proyectos, metodologías, desarrollo, operaciones, Open Source</i>
<b>Resumen del Trabajo:</b>	
<p>Este documento describe los trabajos realizados en el ámbito del Trabajo Fin de Grado (TFG).</p> <p>El propósito principal de este TFG es describir de una manera clara en qué consiste y cómo funciona DevOps en una organización. Se tratará de explicar de manera detallada el concepto de DevOps, de identificar los beneficios de su implementación en las empresas, los desafíos a los que se enfrentan las organizaciones durante su proceso de adopción, así como analizar su ciclo de vida.</p> <p>También se analizará el estado del arte actual respecto a las herramientas que se utilizan para trabajar con DevOps y se mostrará un ejemplo práctico de implementación de DevOps con herramientas software Open Source.</p>	
<b>Abstract:</b>	
<p>This document describes the work carried out in the scope of the Degree's Thesis.</p> <p>The main purpose of this TFG is to describe in a clear way what is DevOps and how it works in an organization. It will try to explain in detail the concept of DevOps, to identify the benefits of implementing it in companies, the challenges that organizations face during their adoption process, as well as analyzing their life cycle.</p> <p>The current state of the art will also be analyzed with respect to the tools used to work with DevOps and a practical example of DevOps implementation with Open Source software tools will be shown.</p>	

## **Dedicatoria y agradecimientos**

Dedico este trabajo final de carrera a mis padres, las mejores personas que conozco. Son el espejo en el que me gustaría reflejarme.

También a mis suegros, que con su infatigable ayuda y cuidado de sus nietos hacen posible que yo pueda seguir estudiando.

Se lo dedico especialmente a mi mujer Silvia, mi punto de apoyo en esta vida, sin ella a mi lado no podría vivir, y a mis hijos Alberto y Marcos, que son los dos soles que iluminan mi vida.

A todos ellos les pido perdón por las ausencias y por todo el tiempo que le he tenido que dedicar a este proyecto y no a ellos.

## Índice

<b>1</b>	<b>Introducción .....</b>	<b>9</b>
1.1	Descripción del proyecto .....	9
1.2	Justificación del proyecto .....	9
1.3	Motivación para la realización el proyecto .....	9
1.4	Ámbito de aplicación del proyecto.....	9
1.5	Objetivos.....	9
<b>2</b>	<b>Planificación.....</b>	<b>11</b>
2.1	Hitos principales del proyecto y su estado actual .....	11
2.2	Descripción de los hitos principales .....	11
2.3	Productos obtenidos.....	12
2.4	Sumario de los capítulos de la memoria .....	12
2.5	Planificación de las tareas.....	13
2.6	Enfoque y método seguido.....	15
<b>3</b>	<b>Introducción a DevOps.....</b>	<b>16</b>
3.1	¿Qué es DevOps? .....	16
3.2	Historia de DevOps .....	17
3.3	Principios de DevOps .....	18
3.4	Objetivos de DevOps.....	21
3.5	Relación de DevOps con las metodologías ágiles .....	21
3.6	DevOps e ITIL.....	21
3.7	El ciclo de vida de DevOps.....	23
3.8	Utilización de DevOps .....	25
3.9	Costes de DevOps .....	25
<b>4</b>	<b>Estado del arte de las herramientas DevOps.....</b>	<b>26</b>
4.1	Gestión de la configuración del código fuente .....	26
4.2	Integración Continua (CI) .....	27
4.3	Testing .....	27
4.4	Entrega Continua (CD) y Despliegue Continuo.....	28
4.5	Monitorización.....	28
4.6	Alertas.....	28
4.7	Seguridad .....	29
4.8	Virtualización y containerización .....	29
4.9	Orquestación .....	30
4.10	Comunicación y colaboración.....	31
<b>5</b>	<b>Aplicación práctica de implementación en DevOps .....</b>	<b>32</b>
5.1	Entorno de desarrollo .....	32
5.2	Creación de una aplicación Web de ejemplo .....	39
5.3	Instalación de JUnit .....	47
5.4	Instalación de Maven.....	47
5.5	Generación de los ejecutables con Maven .....	49
5.6	Ejecución de pruebas unitarias con JUnit y Maven.....	57
5.7	Código fuente y gestión de su configuración con GIT .....	60

5.8	Instalación de SonarQube .....	67
5.9	Instalación de Nexus Repository OSS .....	70
5.10	Instalación de Jenkins .....	71
5.11	Generación de un pipeline con Jenkins .....	78
5.12	Instalación de Docker .....	78
5.13	Instalación de un servidor web NGINX con Docker .....	81
5.14	Integración de Docker con Eclipse .....	83
5.15	Entorno de desarrollo completo con Docker .....	90
<b>6</b>	<b>Conclusiones.....</b>	<b>91</b>
6.1	Conclusiones y lecciones aprendidas .....	91
6.2	Objetivos alcanzados .....	91
6.3	Líneas de trabajo futuras .....	92
<b>7</b>	<b>Glosario de términos .....</b>	<b>93</b>
<b>8</b>	<b>Bibliografía .....</b>	<b>94</b>

## ÍNDICE DE TABLAS

TABLA 1. HITOS PRINCIPALES DEL PROYECTO .....	11
TABLA 2. PLANIFICACIÓN DE TAREAS .....	14
TABLA 3. GLOSARIO .....	93

## ÍNDICE DE FIGURAS

FIGURA 1. PLANIFICACIÓN GENERAL DE HITOS.....	12
FIGURA 2. PLANIFICACIÓN DETALLADA.....	15
FIGURA 3. TRIÁNGULO DEVOPS.....	17
FIGURA 4. HISTORIA DE DEVOPS.....	18
FIGURA 5. MODELO DEVOPS.....	19
FIGURA 6. MURO DE CONFUSIÓN.....	20
FIGURA 7. ITIL.....	22
FIGURA 8: CICLO DE DEMING.....	23
FIGURA 9. MARCO DE TRABAJO DE SCRUM.....	24
FIGURA 10. CICLO DE VIDA DE DEVOPS.....	24
FIGURA 11. TABLA PERIÓDICA DE LAS HERRAMIENTAS DEVOPS.....	26
FIGURA 12. MÁQUINAS VIRTUALES VS CONTENEDORES.....	29
FIGURA 13. EJEMPLO DE GESTIÓN DE CONTENEDORES CON DOCKSTATION.....	90



## 1 Introducción

---

### 1.1 Descripción del proyecto

---

Actualmente, muchas empresas de desarrollo de software están implementando algún tipo de metodología ágil con el fin de generar software más frecuentemente a través de pequeños incrementos que aporten valor al cliente. Sin embargo, eso no implica que el producto esté listo para entrar en producción y para entregar al cliente, ya que muchas veces pueden aparecer problemas de calidad en el software generado o problemas de ejecución en el entorno de producción, generando a veces conflictos entre el equipo de Desarrollo y el de Operaciones.

Para solucionar estos problemas surgió el movimiento o la cultura “DevOps”, que se basa en una relación de trabajo colaborativo entre el Desarrollo y las Operaciones de TI, lo que posibilita un rápido flujo de trabajo (esto es, altas tasas de despliegue), al tiempo que aumenta la confiabilidad, la estabilidad, la resistencia y la seguridad del entorno de producción.

### 1.2 Justificación del proyecto

---

Aunque DevOps está ampliamente implantado en las empresas del Sector de las TI, todavía existe mucho desconocimiento sobre cómo aplicar esa filosofía a cada empresa.

La idea de este proyecto es ofrecer una visión global de DevOps, mostrando cuales son los beneficios que aporta a la empresa su adopción, así como analizar el software actual disponible para implementar la cultura DevOps y mostrar un ejemplo sencillo de implementación que pueda servir de guía o de punto de apoyo a todo aquél que lo lea para iniciar la adopción de DevOps en sus propios proyectos.

### 1.3 Motivación para la realización el proyecto

---

Los motivos para llevar a cabo este TFG son estudiar y conocer a fondo el movimiento DevOps, así como las herramientas empleadas actualmente para cubrir todo el ciclo de vida de DevOps, haciendo especial hincapié en las herramientas Open Source.

Otro de los motivos principales es poner en marcha un pequeño proyecto, utilizando programas Open Source, de manera que se cubra el ciclo de vida completo y que pueda ser implementado por cualquiera que lea este TFG.

### 1.4 Ámbito de aplicación del proyecto

---

La implantación de la cultura DevOps se puede llevar a cabo en la mayoría de las empresas, así como de cualquier sector y/o tecnología.

Dicha implantación puede ser más sencilla en pequeñas empresas, donde los equipos y los cambios necesarios son menores, o en startups y empresas que se creen desde cero adoptando esa cultura desde el inicio.

### 1.5 Objetivos

---

El primer objetivo que se pretende alcanzar con la realización de este TFG es dar una visión completa de la cultura o el movimiento DevOps: porqué surgió la necesidad de implantarlo en las empresas, sus principios, sus roles, su ciclo de vida, su relación con las metodologías ágiles, sus ventajas e inconvenientes, etc.

Otro de los objetivos es analizar el estado el arte actual de las herramientas de software empleadas en DevOps, dando una visión de las herramientas más importantes para cada una de las áreas involucradas en todas las fases de los proyectos de desarrollo de software, dando especial relevancia a las herramientas Open Source.

El último de los objetivos es desarrollar un ejemplo de implantación de un proyecto de desarrollo de software cubriendo todo el ciclo de vida DevOps, empleando herramientas Open Source.

## 2 Planificación

### 2.1 Hitos principales del proyecto y su estado actual

Los hitos de entrega del proyecto están fijados por el plan docente de la asignatura desde el inicio del curso y se descomponen en cuatro entregas. Además hay otros dos hitos importantes que son la defensa del TFG ante el Tribunal y la publicación de las notas finales.

Las fechas de entrega de los hitos y su estado actual son los siguientes:

HITO	Nombre	FECHA	ESTADO
1.1	PEC1 - Propuesta de plan de trabajo TFG.	09/03/2019	Finalizado
1.2	PEC2 - Entre el 40% y el 60% de todo el TFG.	11/04/2019	Finalizado
1.3	PEC3 - Entre el 80% y el 90% de todo el TFG.	15/05/2019	Finalizado
1.4	PEC4 - Entrega Final.	09/06/2019	Finalizado
1.5	Defensa del TFG ante el Tribunal.	24/06/2019	Pendiente
1.6	Publicación de las notas finales.	24/06/2019	Pendiente

Tabla 1. Hitos principales del proyecto

### 2.2 Descripción de los hitos principales

#### Hito 1.1

Este hito establece el comienzo del desarrollo del TFG. En él se desarrollará el documento actual y se corresponde con la entrega de la PEC1. En este documento se propondrá una planificación inicial que será la guía a seguir para el desarrollo del TFG.

#### Hito 1.2

En este hito se comienza a desarrollar la primera parte de la memoria del TFG, donde se deberá completar entre el 40% y el 60% del trabajo del TFG. Se corresponde con la entrega de la PEC2.

#### Hito 1.3

En este hito se comienza a desarrollar la segunda parte de la memoria del TFG, donde se deberá completar entre el 80% y el 90% del trabajo del TFG. Se corresponde con la entrega de la PEC3.

#### Hito 1.4

En este hito se entrega la memoria final del TFG, junto con una presentación resumen y un video para la defensa del TFG. Este hito se corresponde con la PEC4.

#### Hito 1.5

En este hito se inicia el debate con el tribunal, que revisará todo el trabajo realizado y preguntará lo que estime conveniente.

#### Hito 1.6

En este hito, el tribunal asignará una nota final al proyecto y concluirá el TFG. Para tener una visión más esquemática de los hitos propuestos y del contenido general entregable en cada uno de ellos, se muestra a continuación un diagrama de Gantt en el que se pueden ver los hitos establecidos, sus nombres, el período de duración en días laborables y las fechas de inicio y finalización de cada uno de ellos.

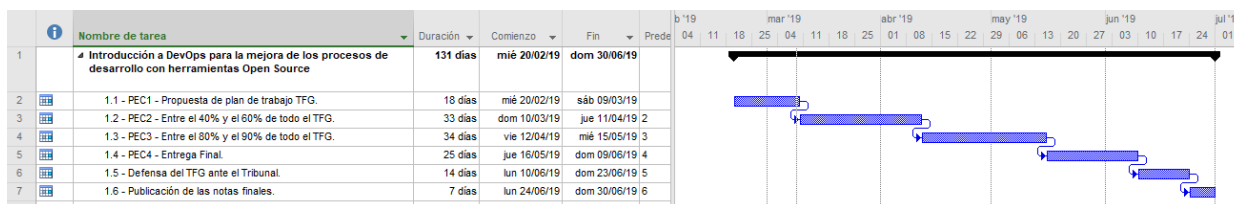


Figura 1. Planificación general de Hitos

## 2.3 Productos obtenidos

Al finalizar el curso, tras realizar las tareas descritas en la planificación, se deberán entregar los siguientes productos:

- **Memoria:** documento con el informe final que recoge todo el trabajo realizado en el TFG.
- **Presentaciones:** presentación multimedia con un resumen del trabajo realizado en el TFG junto con un vídeo con los comentarios correspondientes a cada diapositiva, con una duración máxima de 20 minutos.
- **Código de ejemplo:** se entregará el código fuente y los scripts de configuración empleados en la realización del TFG para que puedan servir de punto de inicio para otros proyectos.

## 2.4 Sumario de los capítulos de la memoria

La lista de capítulos de la memoria del TFG y su relación con el trabajo global es la siguiente:

### 1. Introducción

Presentación del trabajo, justificación y objetivos.

### 2. Planificación

Descripción de los hitos del proyecto y su planificación temporal.

### 3. Introducción a DevOps

Está dedicado a hacer una introducción a DevOps, explicar un poco porqué apareció, en qué consiste y su relación con otras metodologías.

### 4. Estado del arte de las herramientas DevOps

Está dedicado a analizar las herramientas principales que actualmente se emplean para trabajar con DevOps.

### 5. Aplicación práctica implementación en DevOps

Está dedicado a desarrollar un ejemplo práctico de un pipeline completo de Integración Continua (CI) en Jenkins así como a desarrollar un ejemplo de utilización de contenedores Docker.

### 6. Conclusiones

Contiene las conclusiones de la realización del TFG, los objetivos alcanzados y la propuesta de trabajos futuros.

**7. Glosario de términos**

Contiene la lista de acrónimos empleados en la memoria.

**8. Bibliografía**

Contiene la lista con la documentación referenciada en la memoria.

**2.5 Planificación de las tareas**

---

En cada periodo de tiempo asociado a un hito de entrega descrito en el punto 2.1, se han establecido unas tareas a realizar. En este apartado se muestran las subtareas a realizar asociadas a cada uno de esos periodos, de manera que se pueda tener una visión más detallada y estructurada del conjunto total del trabajo que hay que desarrollar en cada periodo y para cada hito de entrega.

A continuación se muestra una tabla en la que se estructura y jerarquiza el trabajo por tareas y subtareas en función de los hitos de entrega establecidos:

Hitos	Descripción de tareas y subtareas	Inicio	Duración	Fin
1.1	PEC1 - Propuesta de plan de trabajo TFG.	20/02/2019	18 días	09/03/2019
	<i>Se realiza la propuesta del TFG y se desarrolla un documento, en el cual se establece la planificación a seguir durante todo el desarrollo del TFG.</i>			
	1.1.1 Preparación de la propuesta del TFG, describiendo el proyecto, sus objetivos y los motivos que impulsan su realización.			
	1.1.2 Realización de la planificación del TFG, describiendo sus hitos principales, descomponiéndolos en tareas y asignándoles un tiempo de ejecución.			
1.2	PEC2 - Entre el 40% y el 60% de todo el TFG.	10/03/2019	33 días	11/04/2019
	<i>Durante esta segunda PEC, se profundizará en el estudio de la cultura DevOps. Se desarrollará el documento correspondiente a esta entrega que contendrá una descripción de DevOps, sus principios, sus roles, su ciclo de vida, sus ventajas e inconvenientes, así como su relación con las metodologías ágiles. También se analizarán las herramientas software empleadas actualmente en DevOps.</i>			
	1.2.1 Descripción de DevOps.	10/03/2019	22 días	31/03/2019
	1.2.2 Análisis de herramientas empleadas en DevOps.	01/04/2019	11 días	11/04/2019
1.3	PEC3 - Entre el 80% y el 90% de todo el TFG.	12/04/2019	34 días	15/05/2019
	<i>Durante esta tercera PEC, se desarrollará un ejemplo de implantación de un proyecto de desarrollo de software cubriendo todo el ciclo de vida DevOps, empleando herramientas Open Source.</i>			
	1.3.1 Desarrollo de ejemplo práctico.	12/04/2019	34 días	15/04/10
1.4	PEC4 – Entrega Final.	16/05/2019	25 días	09/06/2019
	<i>Durante esta tercera PEC, Se preparará la memoria final del TFG junto con una presentación virtual que resuma el trabajo realizado durante el proyecto así como sus resultados y conclusiones. También se entregará un vídeo con la exposición de trabajo realizado durante el TFG.</i>			
	1.4.1 Preparación de la memoria final.	16/05/2019	11 días	26/05/2019
	1.4.2 Preparación de la presentación multimedia.	27/05/2019	7 días	02/06/2019
	1.4.3 Preparación del video.	03/06/2019	7 días	09/06/2019
1.5	Defensa del TFG ante el Tribunal.	10/06/2019	14 días	23/06/2019
	<i>Durante estos 14 días, los miembros del tribunal realizarán las preguntas que estimen oportunas sobre la realización del proyecto a las que se deberá dar respuesta en un plazo máximo de 24 horas.</i>			
	1.5.1 Respuesta a las preguntas formuladas por el tribunal.	10/06/2019	14 días	23/06/2019
1.6	Publicación de las notas finales.	24/06/2019	7 días	30/06/2019
	<i>Se publicarán las notas finales y se dará por finalizado el proyecto.</i>			

Tabla 2. Planificación de tareas

En el siguiente diagrama de Gantt, generado a partir de la tabla anterior, se puede ver toda la planificación de las tareas del proyecto y su estado actual:

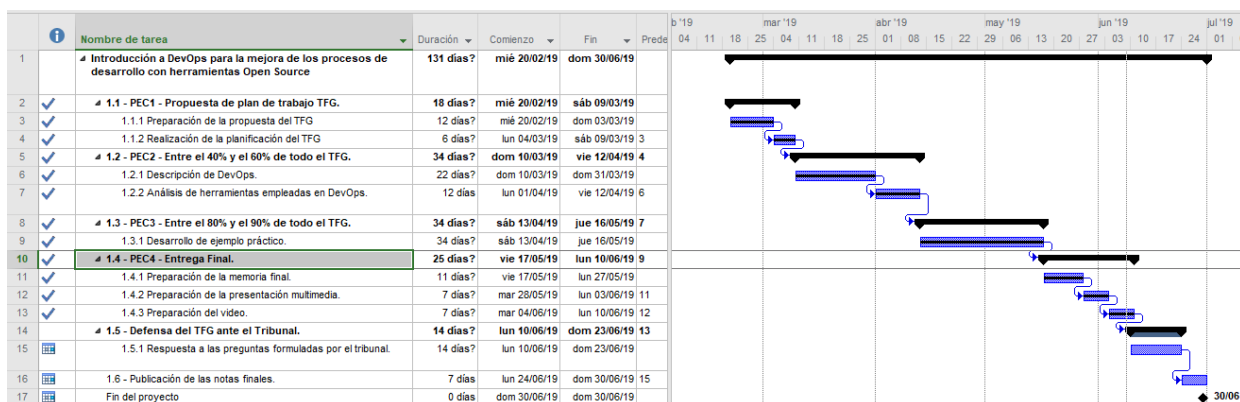


Figura 2. Planificación detallada

## 2.6 Enfoque y método seguido

El enfoque seguido para la realización del TFG es el de ir cumpliendo todos los objetivos del proyecto y en el orden establecido en el capítulo 2.5 de este documento, donde se detalla su planificación.

Para llevar a cabo la planificación, que se deberá cumplir estrictamente para poder terminar el proyecto en la fecha planificada, se dividirá el trabajo total del proyecto en tareas y éstas, a su vez, en subtareas.

- Se empezará describiendo que es DevOps, su historia, su relación con las metodologías ágiles, sus ventajas/inconvenientes, etc.
- A continuación se analizará el estado del arte actual de las herramientas de software empleadas en DevOps, dando una visión de las herramientas más importantes para cada una de las áreas involucradas en todas las fases de los proyectos de desarrollo de software.
- Para terminar, se desarrollará un ejemplo de implantación de un proyecto de desarrollo de software cubriendo todo el ciclo de vida DevOps, empleando herramientas Open Source.

## 3 Introducción a DevOps

---

### 3.1 ¿Qué es DevOps?

---

#### 3.1 ¿Qué es DevOps?

El término DevOps surge de la combinación de las palabras “desarrollo” (Development) y “operaciones” (Operations), ya que se trata de un movimiento profesional y cultural que promueve el trabajo colaborativo, la comunicación y la integración, de las personas que trabajan en el departamento de Desarrollo y las que trabajan en el departamento de Operaciones, para potenciar la entrega rápida, más eficiente, de aplicaciones y servicios, con una alta calidad y que además sean seguros y fáciles de mantener.

Se puede definir DevOps como un conjunto de principios y prácticas que optimizan el tiempo de entrega de un producto software, automatizan todo lo posible, gestionan la infraestructura como código y mejoran la experiencia del usuario en base a la retroalimentación de todo el proceso. DevOps engloba nuevos procesos, nuevas herramientas y nuevas formas de pensar.

La adopción de DevOps implica una nueva forma de trabajar, que trata de que la colaboración impere en el desarrollo, el funcionamiento y las áreas de negocio. Persigue conseguir la eliminación de los silos entre los distintos departamentos (sobre todo entre los de Desarrollo y Operaciones), la participación de las partes interesadas pertinentes y la distribución ágil de resultados de negocio automatizados. Se trata de hacer las cosas mejor con lo que hay disponible, añadir la tecnología apropiada donde el proceso necesita apoyo y conseguir una visibilidad total entre todas las partes implicadas.

En las organizaciones tradicionales, donde las barreras entre los distintos departamentos, con sus correspondientes directores, están muy bien definidas y cada departamento se preocupa únicamente de resolver los problemas que le afectan directamente y se echa la culpa de los posibles errores al resto de departamentos, esta nueva forma de trabajar implica cambios muy importantes, ya que las barreras entre los departamentos se difuminan (por ejemplo se busca formar grupos que asuman funciones de desarrollo y operativas a la vez) y todos se ven involucrados en la generación, de manera continua y en tiempos cortos, del producto y en dar visibilidad total de los procesos a toda la organización.

Cada grupo de trabajo debe romper las barreras existentes entre los departamentos de la organización permitiendo que un producto pase rápidamente desde el departamento de investigación al de diseño, luego al de desarrollo + producción, luego al de test + calidad y por último, a ventas, y con las necesidades de herramientas que permitan desplegar todas estas actividades en cada una de las fases y llevar el control de todos por los responsables de cada uno de los ámbitos, incluidos los funcionales y los de la organización (jefatura y directivos).

En la siguiente imagen se puede apreciar como DevOps se encuentra en la intersección de los equipos de Desarrollo, Operaciones TI y Calidad (QA):



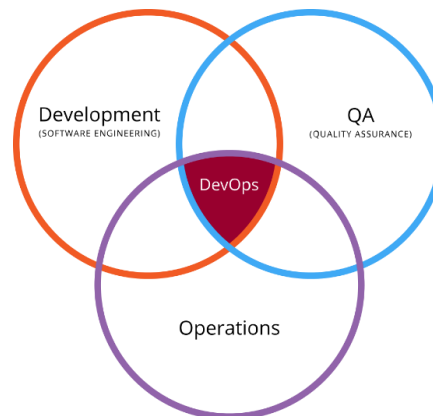


Figura 3. Triángulo DevOps.

Fuente: <https://es.wikipedia.org/wiki/DevOps>

Lo que DevOps pretende es facilitar el desarrollo continuo, la integración continua, la entrega continua y los procesos de monitoreo continuo con el fin de liberar el código más rápido (también reducir los tiempos de paso a producción y de entrega al cliente) y más a menudo para ayudar a la organización a responder de manera más ágil y eficiente a los cambios en los requisitos del negocio.

Los principales motivos estratégicos por los que una organización debe considerar implementar un modelo DevOps son los siguientes:

- **Reducción de costes:** al implementar DevOps se mejoran los procesos del negocio y se reducen por tanto sus costes.
- **Mejora de la calidad:** al estar continuamente generando versiones y probándolas se consigue que el producto final tenga una mejor calidad y se dé un mayor valor al cliente.
- **Mejora de la productividad:** al implementar ciclos más cortos de desarrollo con sus correspondientes pruebas en el entorno de producción se consigue aumentar la productividad y reducir los tiempos de entrega finales.
- **Diferenciación:** al implementar DevOps se reducen las ventajas competitivas de los competidores.
- **Innovación:** al implementar DevOps se consigue innovar introduciendo cambios radicales sobre los procesos del negocio, reduciendo los costes y mejorando la calidad, la eficiencia y el servicio al consumidor (menor time to market).

## 3.2 Historia de DevOps

Todo comenzó en el año 2008, en la conferencia Agile'08 en Toronto (Canadá). Andrew Clay Shafer (creador de Puppet Labs, tenía que dar una charla a la que sólo acudió una persona, un belga llamado Patrick Debois, así que decidió no darla. Sin embargo, pudieron hablar sobre cómo se podría llevar el agilismo al mundo de la infraestructura y la administración de sistemas. Decidieron crear un grupo en Google para abrir la discusión a la comunidad, el Agile System Administrators Group.

Un año después, en el evento Velocity'09, John Allspaw y Paul Hammond expusieron su conferencia "10+ Deploys a Day: Dev and Ops Cooperation at Flickr" a la que el belga Patrick Debois no pudo asistir por la distancia a su casa. Al publicar su desencanto por no haber podido acudir al evento en las redes, recibió la contestación de Paul Nasrat, responsable por entonces del CMS del periódico británico The Guardian y desde 2010 en Google, proponiéndole que organizara un evento similar en Europa.

A Patrick le pareció una idea excelente y sólo cuatro meses después estaba convocado el primer DevOps Day, en Ghent, Bélgica, en Noviembre de 2009. La repercusión fue enorme, y el hashtag creado para la

ocasión, #DevOps, triunfó en las redes sociales de forma viral, dando el nombre definitivo a todo el movimiento.

Desde entonces ha ido ganando adeptos entre todo tipo de compañías, para hacer más y mejor sus productos y servicios y, sobre todo, debido a las sinergias con diferentes tecnologías que se han ido adoptando hasta hoy en día como: el uso de los procesos y metodologías de desarrollo ágil, necesidad de poner una mayor tasa de versiones en producción en menos tiempo, utilización de entornos virtualizados, utilización de la nube y las tecnologías Cloud, mayor automatización de todos los procesos y de los centros de datos y aumento de las herramientas de gestión de configuración.



Figura 4. Historia de DevOps

Fuente: <https://www.slideshare.net/DevOpstastic/where-to-start-with-devops>

### 3.3 Principios de DevOps

DevOps no tiene un manifiesto similar al “manifiesto ágil” adoptado en las distintas metodologías ágiles, aunque en cierta manera también se encuentra incluido implícitamente en él.

Gene Kim en sus libros “DevOps Handbook” y “The Phoenix Project” describe los pilares fundamentales de DevOps como las tres vías (“Three Ways”):

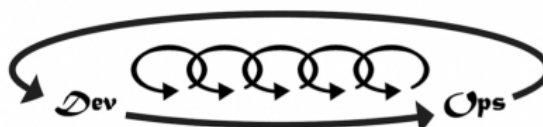
- **La primera vía: Pensamiento sistémico:** se centra en todos los flujos que aportan valor para la entrega al cliente, del sistema completo, no de un solo departamento. Se basa en incrementar valor hasta el final sin dejar que los errores avancen hacia la derecha.



- **La segunda vía: Amplificar los bucles de retroalimentación:** consiste en crear bucles de retroalimentación de derecha a izquierda. Esto es, comprender y responder a las necesidades de los clientes.



- **La tercera vía: Cultura de experimentación y aprendizaje continuos:** consiste en experimentar y asumir riesgos de manera continuada, aprendiendo tanto de los éxitos como de los fracasos. Es necesario para mejorar en el trabajo diario.



DevOps se apoya sobre las siguientes metodologías y modelos:

- **Metodologías ágiles:** permiten incrementar la velocidad de desarrollo de los sistemas y mejorar el Time To Market.
- **Integración continua (Continuous Integration, CI):** consiste en integrar los cambios en el código fuente frecuentemente, con compilaciones automáticas, análisis estático y de cobertura del código y pruebas automáticas, para detectar errores de integración tan pronto como sea posible. Si alguno de estos pasos falla se da por erróneo el cambio y se notifica.
- **Entrega continua (Continuous Delivery, CD):** consiste en automatizar el paso del software a los entornos productivos. Todos los cambios se prueban y envían a un entorno de pruebas y/o producción. La configuración de la infraestructura como código nos permite crear plantillas de servicios complejos de forma fácil y poderlas almacenar bajo control de versiones como el resto del código fuente.
- **Despliegue continuo (Continuous Deployment):** se diferencia de la entrega continua en que no es necesario una aprobación explícita ni ninguna intervención manual.

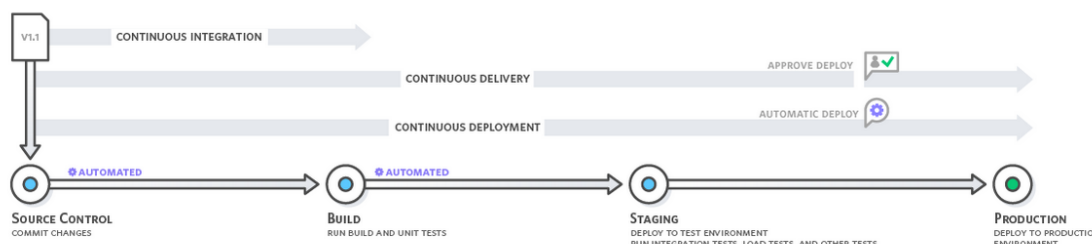


Figura 5. Modelo DevOps.

Fuente: <https://aws.amazon.com/es/devops/continuous-integration/>

Uno de los problemas que soluciona el movimiento DevOps es el llamado “muro de confusión”. Es el conflicto que surge cuando el equipo de Desarrollo hace el traspaso de código al equipo de Operaciones, y éste trabaja en el testing y en los scripts para el pase a producción. Es entonces cuando suelen surgir errores que no se sabe bien a quién atribuir, si a Desarrollo por entregar código con fallos o a Operaciones por realizar una mala configuración.

Cada departamento tiene sus propias prioridades, su forma de trabajar y vela por sus intereses: Desarrollo busca la manera de añadir nuevas funcionalidades y Operaciones busca la estabilidad por encima de todo. Al aparecer los errores surgen las fricciones y la falta de confianza entre los equipos. DevOps pretende aunar los intereses de ambos departamentos y crear un ambiente de confianza y cooperación mediante prácticas que fomentan una actitud positiva respecto al fallo y una cultura sin culpas (blameless culture), colaborativa, en la que la información fluya y el equipo se sienta comprometido con su trabajo.

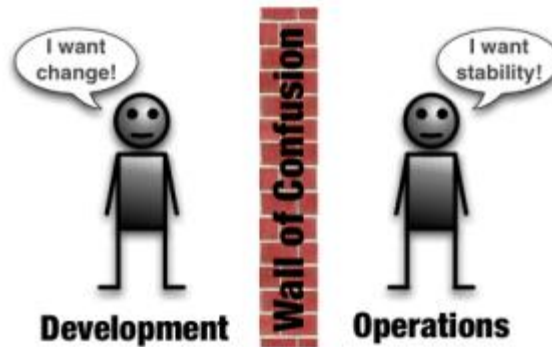


Figura 6. Muro de confusión.

Fuente: <http://dev2ops.org/2010/02/what-is-devops/>

El modelo **CALMS**, acrónimo de Culture, Lean, Automation, Metrics y Sharing, persigue conseguir el cambio de mentalidad necesario en los equipos de Desarrollo y Operaciones para adoptar la cultura DevOps:

- **Cultura:** cambiar la manera de pensar y de comportamiento dentro de la organización. Desaparecen los equipos y se fomenta la cooperación.
- **Automatizar:** automatizar todo lo posible durante todo el ciclo, desde desarrollo hasta producción (CI/CD).
- **Lean:** enfocado en entregar valor al cliente y reduciendo lo que no aporta (los desperdicios).
- **Medir:** monitorizar y medir continuamente. Mostrar las mejoras.
- **Compartir:** compartir conocimiento y colaborar con el resto de trabajadores.

DevOps se fundamenta en los siguientes conceptos principales:

- **Mejora Continua:** DevOps defiende la mejora continua para minimizar el desperdicio.
- **Automatizar todo lo posible:** la automatización es un principio central del proceso DevOps. No solo durante la fase de desarrollo del software, sino también para la gestión de la infraestructura. Se deben automatizar todos los procesos manuales repetitivos y que no aporten valor
- **Trabajar como un solo equipo:** DevOps trata de acabar con los silos y las barreras entre los distintos departamentos. Cada trabajador es responsable de su parte del proyecto.
- **Monitorización y pruebas continuas:** para detectar los errores lo antes posible y mejorar la calidad de los productos generados.

Los beneficios de la utilización de DevOps van orientados a mejorar la calidad del software, reducir el tiempo de entrega y ahorrar esfuerzos/costes en el proceso completo, alineando las estrategias de los equipos de desarrollo y producción/operaciones.

DevOps tiene tres componentes fundamentales: procesos, organización y tecnología. Si alguno de ellos no está suficientemente alineado con los principios de DevOps, no funcionará y no se obtendrán todos sus beneficios.

### 3.4 Objetivos de DevOps

---

Los objetivos de DevOps comprenden todo el proceso de entrega. Los principales son los siguientes:

- **Mejora de la frecuencia de despliegue:** en el mundo del desarrollo de software tradicional se suelen compilar muchas versiones pero se despliegan muy pocas. Uno de los objetivos que se cumplen utilizando metodología DevOps es mejorar la frecuencia y la calidad de los despliegues. Para ello se utilizan herramientas como el Continuous Delivery (entrega continua) y el Continuous Deployment (despliegue continuo), de manera que se compila y testea el código continuamente cada vez que se realiza un cambio y así, por un lado se detectan los errores tempranamente y por otro, siempre se tiene lista una versión entregable del software.
- **Mejora del Time to Market:** al utilizar las herramientas anteriores para mejorar la frecuencia de despliegue permite salir al mercado de manera más ágil, detectar los errores antes y empezar a trabajar con productos terminados más rápidamente.
- **Reducción de la tasa de error:** la aplicación de ciclos de monitorización, testeo continuo y corrección de errores detectados, permiten detectar los errores que pasan al entorno de producción y corregirlos produciendo el menor impacto posible y reduciendo así la tasa y los tiempos de error de los productos.
- **Reducción de los tiempos de corrección:** al integrar de manera continua, se trabaja con menos cantidad de cambios en cada delivery y al emplear automatización y monitorización continuas, las correcciones se realizarán en un tiempo más corto al encontrar con mayor facilidad el origen de los fallos.

### 3.5 Relación de DevOps con las metodologías ágiles

---

DevOps es complementario a las metodologías ágiles de desarrollo de software, lo implementa en las etapas de desarrollo y lo completa con la integración y la entrega continua, asegurando que el código está siempre listo para producción y dando valor al cliente.

Las metodologías ágiles han conseguido que la parte del equipo de Desarrollo mejore notablemente pero se olvida de la parte de Operaciones. DevOps es la herramienta que hace que ambos equipos funcionen como uno solo.

Las metodologías ágiles se centran en el equipo, sus interacciones y sus valores, mientras que DevOps lo hace en los procesos y en el flujo, pero en ambos lo más importante son las personas y el trabajo en equipo orientado a resultados.

### 3.6 DevOps e ITIL

---

La Biblioteca de Infraestructura de Tecnologías de Información (ITIL) es un estándar en la gestión de servicios informáticos que proporciona un conjunto de procedimientos de gestión ideados para ayudar a las organizaciones a lograr mejorar la calidad y la eficiencia en las operaciones de TI.

Su nombre es debido a tener su origen en un conjunto de libros, cada uno dedicado a una práctica específica dentro de la gestión de TI. Actualmente, su última versión, la v3 se centra en el Ciclo de Vida del Servicio y está compuesta por cinco disciplinas o grupos de procesos (libros) principales, en los que la adopción de un modelo DevOps puede afectar de la siguiente manera:

- **Estrategia:** este grupo de procesos (gestión de la demanda de servicios tecnológicos, y elaboración de un catálogo de servicios y diseño y elaboración de un plan tecnológico) se ven poco afectados por la adopción de un modelo DevOps. Únicamente se tiene que tener en cuenta desde el principio y contemplarlo en cada uno de sus procesos.

- **Diseño:** este grupo de procesos sí que se ve afectado por la adopción de un modelo DevOps, ya que en esta fase se debe dar forma a la estrategia definida para los servicios de infraestructura, su estructura interna (arquitectura), su mantenimiento y evolución, tanto a nivel conceptual como físico y por tanto se deben introducir los cambios necesarios para que pueda funcionar empleando el modelo DevOps.
- **Transición:** este grupo de procesos se encarga de la puesta en marcha de los servicios diseñados en la etapa anterior. Por lo tanto se deben poner en marcha los cambios a todos los niveles de adopción de la cultura DevOps.
- **Operación:** este grupo de procesos se ve afectado, en el sentido de que toda la gestión de incidencias y problemas ahora se debe automatizar y mejorar para integrarlo dentro del modelo DevOps, que implica una mayor velocidad en las entregas y por tanto una detección más temprana de los errores que puedan aparecer dentro del entorno final operativo.
- **Mejora continua:** esta es una de las razones por las que se debe implementar un modelo DevOps, por lo tanto todos sus procesos (métodos de medida del rendimiento, planes de acción, procedimientos de evaluación, auditoría, etc.) deben estar integrados dentro de la forma de trabajar del modelo DevOps.

En la siguiente imagen se pueden apreciar los diferentes grupos de procesos de ITIL, con sus componentes principales y sus interrelaciones:

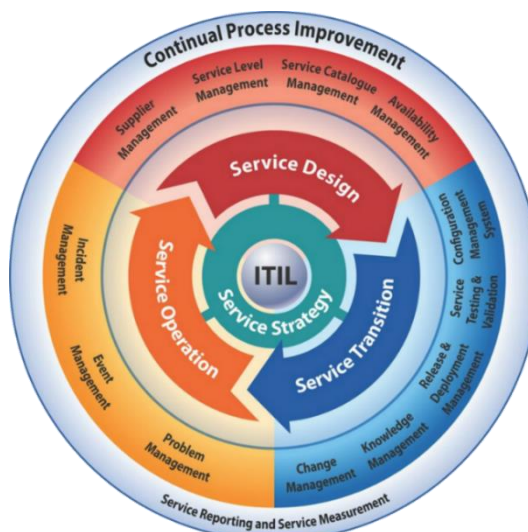


Figura 7. ITIL.

Fuente: <https://www.pdcachome.com/5782/itil-v3-infraestructura-para-tecnologias-de-la-informacion/>

Por lo tanto, ITIL y DevOps no son antagónicos, se pueden utilizar conjuntamente y aprovechar todo el potencial de cada uno en las organizaciones. En principio pueden parecer metodologías incompatibles (tienen fundamentos opuestos) ya que los procesos tradicionales de gestión de servicios y operaciones al estilo ITIL se basan en procesos y procedimientos documentados bien definidos, con enfoque a los mecanismos de planificación y supervisión, y sin embargo DevOps se basa en un enfoque en metodologías ágiles y en entregas continuas (y rápidas) del producto.

Operativamente, al integrar DevOps manteniendo el resto de procedimientos y servicios tradicionales, se añade agilidad a los procesos de ITIL, sin comprometer la seguridad, ya que se consiguen detectar y por tanto corregir los errores que puedan aparecer de una manera más rápida. Es evidente la importancia de adoptar el cambio hacia prácticas como DevOps, que ayuden a optimizar el trabajo para no acabar escribiendo código de forma obsoleta, pero sin olvidar ni eliminar las buenas prácticas de ITIL.

## 3.7 El ciclo de vida de DevOps

### 3.7.1.1 El ciclo de Deming

El ciclo de Deming, es una herramienta de gestión de proyectos que promueve la mejora continua y el aumento de la calidad de los procesos por medio de cuatro fases o etapas que se repiten cíclicamente:

- **Plan:** Es la fase imprescindible y quizás la más importante, donde se deben definir los objetivos a conseguir y cómo se van a alcanzar. Se analizan el presupuesto, los tiempos, el alcance, etc.
- **Do:** En esta fase se implementa todo lo planificado según lo establecido en la fase de planificación, según los plazos, recursos, etc. Ya está claro qué hay que hacer, quién y cómo hacerlo y en esta fase se realiza el trabajo.
- **Check:** En esta fase se analizan los indicadores para ver si lo que se ha planificado se está cumpliendo
- **Act:** La mejora continua permite ajustar los resultados y llegado el caso poder realizar los cambios y ajustes necesarios. Una vez terminado esta fase se vuelve a ejecutar la primera fase de nuevo.



Figura 8: Ciclo de Deming.

Fuente: <https://www.socialmediadol.com>

Este ciclo, también llamado PDCA por las siglas de sus cuatro fases, ayuda en la toma de decisiones, permitiendo una mejora continua en la gestión de proyectos.

### 3.7.1.2 Ciclo de vida de las metodologías ágiles

Es siempre un ciclo de vida **iterativo e incremental**, con iteraciones cortas (semanas) y sin que dentro de cada iteración tenga porque haber fases lineales (tipo cascada), son por tanto muy flexibles. Este tipo de ciclo de vida también se suele llamar **adaptativo u orientado al cambio** ya que responde a niveles altos de cambio y a la participación activa y permanente entre todos los implicados en el proyecto.

Cada tipo de metodología ágil que existe tiene su propio ciclo de vida particularizado, aunque siempre será iterativo e incremental y aumentando la funcionalidad en cada entrega. Existen dos enfoques principales para este los distintos tipos de ciclo de vida ágiles:

- Los **enfocados a las iteraciones:** que se basan en ciclos muy rápidos llamados iteraciones, con una duración de entre 1 a 4 semanas como máximo, en las que se debe realizar el trabajo acordado previamente. En la metodología Scrum, por ejemplo, a cada iteración se le denomina Sprint.



Figura 9. Marco de trabajo de Scrum.

Fuente: <https://www.itnove.com/es/agile/coaching-consultoria-scrum-barcelona>

- Los **centrados al flujo del trabajo**: por ejemplo Kanban, en el que se establecen claramente las limitaciones sobre la secuencia de actividades (concepto de Work In Progress WIP).

### 3.7.1.3 El ciclo de vida DevOps

El ciclo de vida en DevOps se basa en el ciclo de vida de las metodologías ágiles y es por tanto iterativo. Las fases que lo integran forman un bucle infinito donde entran en juego las tareas propias de dos mundos: el de Desarrollo (parte izquierda del bucle) y el de Operaciones (parte derecha del bucle). De esta forma, se inicia un flujo de tareas continuas, que comienzan en la fase de análisis y finalizan en la de aprobación del sistema, repitiéndose de nuevo todo el proceso.

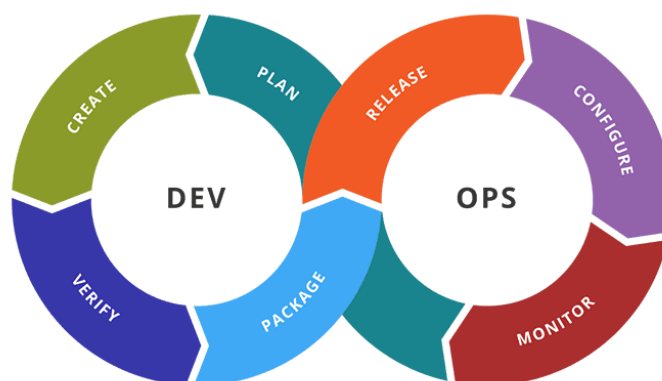


Figura 10. Ciclo de vida de DevOps.

Fuente: <https://www.uxland.es/devops-integracion-continua/>



### 3.8 Utilización de DevOps

---

La utilización de DevOps está especialmente recomendada en los siguientes casos:

- Las empresas con entregas muy frecuentes de software (que utilizan metodologías ágiles).
- Las empresas con un alto componente de TI.
- Startups, que nacen implementado DevOps directamente, de manera natural.
- Las empresas que desarrollan servicios en la web.
- Las empresas que desarrollan servicios o aplicaciones móviles.

Sin embargo no es recomendable su utilización en aplicaciones de misión críticas como bancos, energéticas, militares, etc., ya que por su naturaleza necesitan de estrictos controles de acceso al entorno de producción, una política detallada de gestión de cambios, política de control de acceso a los centros de datos, etc.

### 3.9 Costes de DevOps

---

La adopción de DevOps conlleva unos costes iniciales grandes, en infraestructura y sobre todo en personas. Es muy importante que toda la organización se implique en el cambio de filosofía de trabajo que conlleva DevOps.

Una vez que DevOps está correctamente implantado en la empresa los beneficios que introduce provocan una disminución de los costes incalculable, ya que se consigue reducir el time to market, reducir el tiempo en la detección fallos y por tanto aumentar la eficiencia y la calidad de los productos desarrollados.

## 4 Estado del arte de las herramientas DevOps

DevOps no se trata únicamente de herramientas, pero es cierto que son necesarias herramientas especializadas para facilitar y acelerar todos los procesos del flujo DevOps. Actualmente el número de herramientas relacionadas con DevOps ha crecido espectacularmente y cubren cada una de las categorías involucradas en todo el proceso.

En la siguiente imagen se puede apreciar la Tabla Periódica de las herramientas DevOps que genera la empresa Xebialabs y que mantiene actualizada, donde se pueden ver las herramientas más utilizadas actualmente y su funcionalidad:

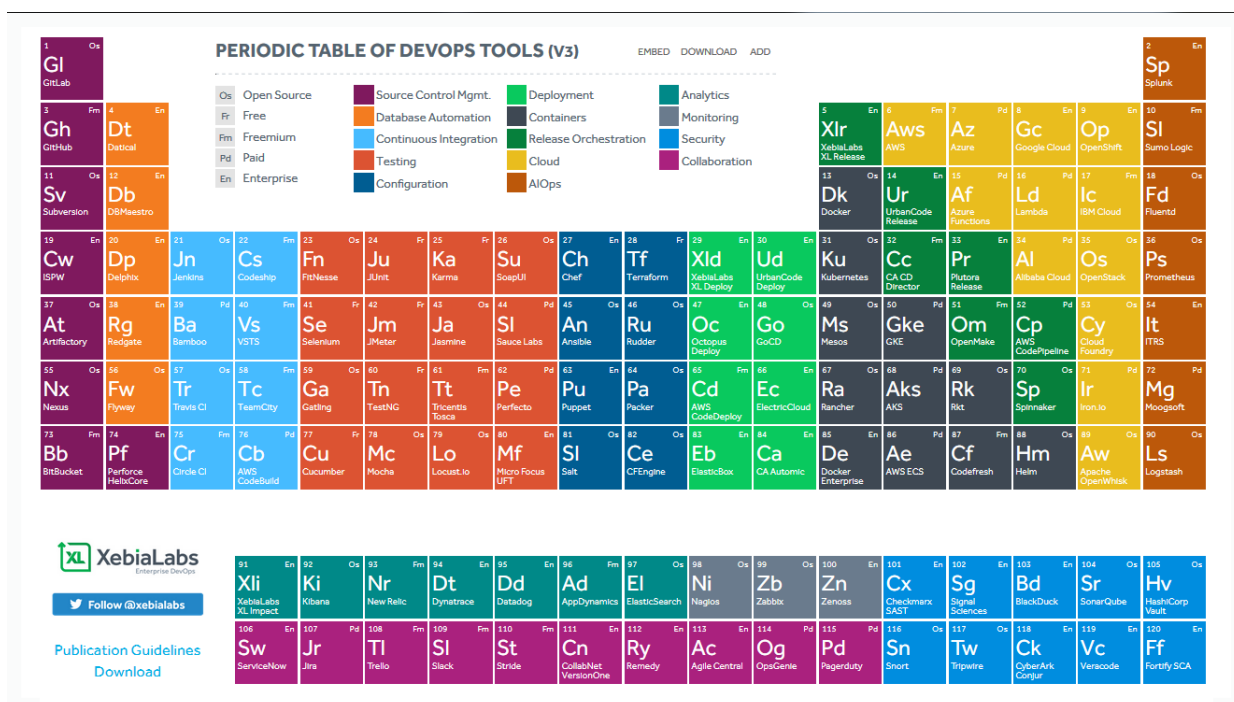


Figura 11. Tabla periódica de las herramientas DevOps.

Fuente: <https://xebialabs.com/periodic-table-of-devops-tools/>

A continuación se muestran algunas de las herramientas más importantes utilizadas actualmente en cada una de las categorías involucradas en los procesos DevOps.

### 4.1 Gestión de la configuración del código fuente

Los sistemas de control de código fuente o SCM (Source Code Management) se utilizan para llevar el control sobre los cambios realizados en el código fuente por los distintos usuarios: modificaciones, creación y merge de ramas, resolución de conflictos entre versiones de ficheros, etc.

Algunos de los SCM más utilizados son los siguientes: Git, CVS, Subversion, SourceSafe, ClearCase, Mercurial, Team Foundation Server, Nexus, Bitbucket.

- **Git:** Fue desarrollado por Linus Torvalds para el desarrollo del kernel de Linux en 2005, y actualmente es el sistema de control de versiones más utilizado para el desarrollo de software. Es un sistema de control de versiones distribuido, cuyas virtudes son la alta velocidad, la integridad de los datos, y soporte para flujos de trabajo no lineales.

- **GitHub:** no es propiamente un SCM, sino que es una plataforma donde alojar los proyectos pero utilizando el sistema de control de versiones Git. Aporta más funcionalidades y proporciona una interfaz gráfica para el control de versiones en vez de ser únicamente por comandos como lo es Git.

Otra herramienta interesante es **Gerrit**, que es una extensión de Git que sirve para establecer un mecanismo de revisión de los cambios en el código antes de la publicación de los mismos. De esta manera solo llega al repositorio final código que ha sido probado y revisado.

## 4.2 Integración Continua (CI)

---

Es una práctica de desarrollo que se basa en la integración frecuente del código fuente en un mismo repositorio compartido. Cada cambio introducido es verificado por una construcción automatizada, permitiendo a los equipos detectar los problemas más rápidamente.

Algunas de las herramientas que facilitan la CI más utilizadas son las siguientes: Jenkins, GitLab CI, Travis CI, Atlassian Bamboo, Circle CI, Ant/Maven/Gradle.

- **Jenkins:** es una de las herramientas más utilizada actualmente. Es de código abierto y escrita en Java. Es compatible con las herramientas de SCM más usuales como Git, Subversion, Mercurial, etc. y puede ejecutar proyectos basados en Ant y Maven, así como las secuencias de comandos shell arbitrarias y comandos por lotes de Windows.
- **GitLab CI:** es de código abierto (muy similar a GitHub pero que es de pago) y es un sistema de gestión completo para el código fuente: sistema de control de versiones basado en Git, sistema de gestión de incidencias, sistema para despliegues automáticos, tiene workers locales y en la nube para lanzar las compilaciones, etc. Se puede instalar en la nube o en local.
- **Travis CI:** es de software propietario, pero a través de la web <https://travis-ci.org/> se pueden gestionar proyectos alojados en GitHub de manera gratuita.
- **Ant/Maven/Gradle:** son tres herramientas para automatizar la construcción de proyectos Java.

En la siguiente página web aparece una comparativa entre Jenkins, Travis CI y Circle CI: <https://djangostars.com/blog/continuous-integration-circleci-vs-travisci-vs-jenkins/>.

## 4.3 Testing

---

Las herramientas de testing son imprescindibles para validar el código antes y después de integrarlo. Uno de los pilares de DevOps es la ejecución automatizada de los tests para detectar errores lo antes posible.

Existe una gran variedad de herramientas de testing, dependiendo del tipo de test a realizar, tipo de aplicación a testear, tipo de lenguaje de desarrollo empleado, etc. Algunas de las más empleadas actualmente son las siguientes: JUnit, JMeter, SoapUI, Selenium, Jasmine, Cucumber, etc.

- **JUnit:** es un framework muy utilizado para pruebas unitarias para código fuente en Java.
- **Selenium:** es un framework muy utilizado para pruebas automáticas de interfaz (UI), muy utilizado para testear aplicaciones web.

## 4.4 Entrega Continua (CD) y Despliegue Continuo

---

La entrega y el despliegue continuo tienen como objetivo hacer llegar al cliente final lo antes posible y con calidad, las modificaciones que se han implementado en desarrollo.

Algunas de las herramientas que facilitan la CD más utilizadas son las siguientes: Ansible, Chef, Puppet, SaltStack, Packer/Terraform/Consul/Vault.

- **Ansible:** es de código abierto y permite automatizar el provisionamiento de software, la gestión de configuración y el despliegue de aplicaciones tanto en máquinas virtuales como físicas. Es parte de RedHat y está basado en Python.
- **Chef:** es de software propietario. Permite automatizar el provisionamiento de software, la gestión de configuración y el despliegue de aplicaciones. Está basado en Ruby y Erlang.
- **Puppet:** es de software propietario aunque tiene versión de código abierto. Está basado en C++ y Clojure. Utiliza su propio lenguaje para describir la configuración del sistema.

En la siguiente página web aparece una comparativa entre Puppet, Chef, Ansible y SaltStack: <https://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack>.

## 4.5 Monitorización

---

La monitorización es esencial para saber el estado de los sistemas y servicios en todo momento. Es necesario recopilar esa información, mostrarla de una manera adecuada y que todos los usuarios tengan acceso a ella.

Algunas de las herramientas más utilizadas en la monitorización son las siguientes: Nagios/Icinga, Telegraf/Statsd, Graphite/Grafana, AWS CloudWatch, Google StackDriver, Pingdom/StatusCake, New Relic/Rollbar, Elasticsearch, etc.

- **Nagios/Icinga:** está orientado a monitorizar el estado del hardware. Tiene agentes NRPE, NRDP, NSClient++ y NCPA. Tiene muchos plugins y se puede conectar por SNMP para obtener informes.
- **Graphite/Grafana:** se emplean principalmente para visualización de datos. Tienen plugins para extraer datos de distintas fuentes y la capacidad de generar alertas.
- **New Relic/Rollbar:** están orientados hacia la gestión de la monitorización del rendimiento de las aplicaciones (Application Performance Management, APM). Monitorizan el rendimiento, tiempos de carga, errores de la aplicación, percepción del rendimiento del usuario, etc. Permiten lanzar alertas cuando el rendimiento es bajo.

## 4.6 Alertas

---

Monitorizar no es suficiente, es necesario contar con un servicio de alertas que informe de los problemas a cualquier hora del día y todos los días del año.

Algunas de las herramientas más utilizadas son las siguientes: Kapacitor, Pushover y OpsGenie/VictorOps/PagerDuty.

## 4.7 Seguridad

La seguridad en DevOps es fundamental, sin embargo al buscar desarrollos ágiles se puede correr el riesgo de dar menos importancia a la seguridad. DevSecOps se ocupa de gestionar la seguridad dentro de todo el proceso DevOps con una serie de herramientas especializadas para esa labor, como por ejemplo: Snort, SonarQube, Veracode, Fortify SCA, etc.

- **Snort:** es un Sistema de Detección de Intrusos (IDS) open source.
- **SonarQube:** es un sistema de aseguramiento de la calidad del código fuente. Es open source. Analiza todo el código y alerta de posibles defectos y errores que puedan provocar problemas de seguridad.

## 4.8 Virtualización y containerización

La virtualización es una tecnología que permite que un equipo anfitrión (host) con un sistema operativo propio pueda ejecutar una o más máquinas virtuales con sistemas operativos clientes. Para ello un programa crea un entorno virtual (máquina virtual) donde se sintetiza un entorno de computación real (NIC virtual, BIOS, tarjeta de sonido, vídeo, etc.). Esto es posible gracias a una capa de software que gestiona estos recursos reales en el equipo anfitrión y los reparte dinámicamente entre las distintas máquinas virtuales hospedadas en dicho equipo. A este software intermedio se le denomina hypervisor.

Los contenedores son también una tecnología de virtualización, pero no necesitan un programa hypervisor ni máquinas virtuales para ser ejecutados sino que se ejecutan directamente sobre el kernel del sistema operativo anfitrión (OS-level virtualization), que se encarga de ejecutar los distintos contenedores y de aislarlos unos de otros.

### VMs vs Docker

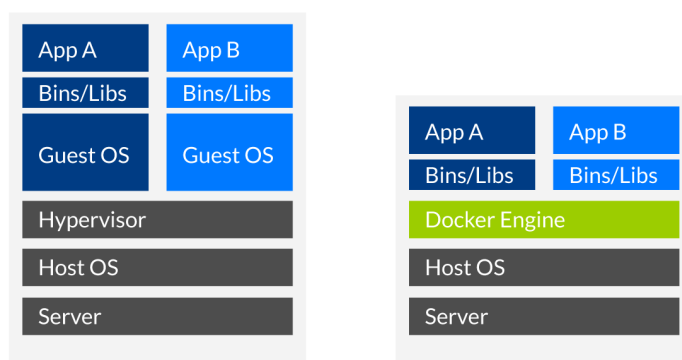


Figura 12. Máquinas virtuales Vs Contenedores

Fuente: <https://www.lomasnuevo.net/cloud/maquinas-virtuales-vs-contenedores/>

El uso de contenedores proporciona una serie de ventajas con respecto al uso de máquinas virtuales:

- **Menor tamaño:** al no contener al sistema operativo son ficheros mucho más pequeños.
- **Menor uso de memoria:** al no tener que ejecutar el sistema operativo necesita menos memoria. Por esta razón (y el punto anterior), se pueden ejecutar de 10 a 100 veces más contenedores que máquinas virtuales en el mismo equipo.
- **Provisión de recursos:** al crear una máquina virtual se debe provisionar el espacio que ocupará en el disco duro, la memoria reservada, el número de CPUs, etc. Este paso no es necesario en los contenedores.

- **Arranque/parada más rápido:** los contenedores pueden iniciarse en segundos mientras que las máquinas virtuales deben arrancar el sistema operativo y pueden tardar minutos.

También alguna desventaja como:

- No pueden ejecutar un programa de un sistema operativo distinto al de la máquina host, que con máquinas virtuales sí se puede.
- Peor aislamiento de ataques: las máquinas virtuales proporcionan abstracción a nivel de hardware, de manera que un ataque puede afectar únicamente a la máquina virtual comprometida, aislando a todas las demás que se están ejecutando en el mismo host.

Algunas de las herramientas más utilizadas para máquinas virtuales las siguientes: VMware, VirtualBox, Hyper-V, Vagrant, etc.

Para la creación de contenedores la herramienta más utilizada es Docker. Realiza virtualización a nivel de sistema operativo, en unidades aisladas llamadas contenedores. Su utilización se ha impuesto en el mercado por su sencillez de uso, su versatilidad, su eficiencia y un repositorio público donde están disponibles multitud de aplicaciones que se pueden descargar y utilizar en cuestión de segundos. También se pueden crear contenedores propios con un lenguaje muy sencillo.

Existen dos versiones: la Docker CE (Community Edition) disponible de manera gratuita y la versión Premium donde se incluyen servicios empresariales y soporte extendido.

## 4.9 Orquestación

---

Actualmente las aplicaciones suelen ser complejas y por regla general no basta con desplegar un solo contenedor en producción, lo habitual es tener que desplegar varios, por ejemplo un contenedor para el Front-End, otro para la base de datos, otro para determinados servicios, etc.

Para solucionar esta necesidad surgió la orquestación de contenedores, esto es, herramientas que automatizan el despliegue, la gestión, el escalado, la interconexión y la disponibilidad de las aplicaciones basadas en contenedores.

Algunas de las herramientas más utilizadas para la orquestación son las siguientes: Kubernetes, Docker Swarm y Compose, Mesosphere DC/OS, Google Container Engine (GKE), Azure Container Service (AKS), Amazon ECS, etc.

- **Kubernetes:** es el más utilizado actualmente. Es propiedad de Google. Funciona agrupando los contenedores que componen una aplicación en unidades lógicas para gestionarlos de una manera sencilla e intuitiva.
- **Docker Swarm:** viene incluido junto con el motor de Docker. Incorpora muchas funciones avanzadas como el descubrimiento de servicios, balanceo de carga, escalado y seguridad. Es más sencillo de usar que Kubernetes pero no es tan potente. Swarm se diferencia de Compose en que es capaz de orquestar contenedores que están ejecutándose en máquinas distribuidas remotamente.
- **Docker Compose:** permite describir la forma en que se pueden orquestar despliegues de Docker a gran escala, dando detalles sobre cómo están relacionados cada uno de los contenedores con los demás. Proporciona un conjunto de instrucciones para desplegar un gran número de contenedores interconectados dentro de Swarm, y proporciona una forma para actualizar los contenedores individuales sin afectar las operaciones de los otros

#### **4.10 Comunicación y colaboración**

---

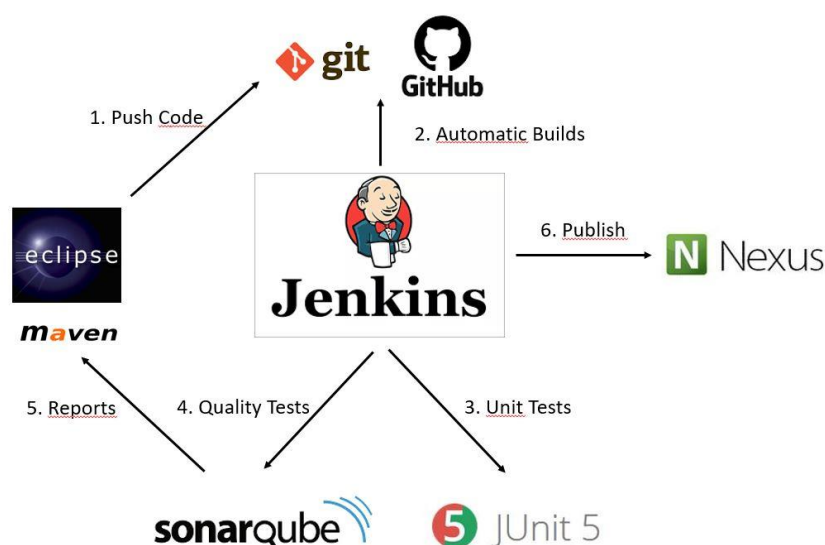
No son estrictamente herramientas de DevOps, sino más bien de gestión de proyectos y de trabajo en equipo, sin embargo son imprescindibles para que la información fluya y se comparta por todos los miembros de la organización involucrados en los procesos DevOps.

Algunas de las herramientas más utilizadas para mejorar la comunicación y fomentar la colaboración son las siguientes: Slack, HipChat, JIRA, Trello, Stride y ServiceNow.

## 5 Aplicación práctica de implementación en DevOps

### 5.1 Entorno de desarrollo

Se va a desarrollar un ejemplo de un proceso de integración continua (CI) completo para un entorno de desarrollo local, esto es, para que cualquier desarrollador se lo pueda instalar en su sistema. El proceso funciona de manera que cada vez que un programador haga un cambio en el código del repositorio Git remoto, Jenkins lo detecta, compila la aplicación con Maven, llama a Junit para pasar los test unitarios y luego a SonarQube para ejecutar los test de calidad del código y, si todo ha ido bien, publica el resultado del proyecto en un repositorio de artefactos como Nexus.



Normalmente en las empresas de desarrollo de software existen al menos 4 entornos de desarrollo diferentes:

- **Entorno de desarrollo local:** es el ordenador de cada programador, donde se instala las herramientas y librerías que considere necesarios.
- **Entorno de desarrollo:** suele ser una máquina distinta, normalmente donde se hace la integración continua y se pasan todos los tests de desarrollo.
- **Entorno de pruebas:** es otra máquina o conjunto de ellas donde se pasan todas las pruebas necesarias, normalmente por un equipo de personas distinto al de desarrollo.
- **Entorno de producción:** es el entorno donde se pasan las pruebas finales en el entorno del cliente final.

El ejemplo de aplicación práctica de este TFG va orientado a la preparación del entorno local de los desarrolladores.

Se ha seleccionado el lenguaje Java por ser uno de los más empleados actualmente. La instalación de Java dependerá de la versión de LINUX que se esté ejecutando, ya que cada una contiene una serie de paquetes distintos, incluso en algunas de ellas ya viene preinstalado por defecto. En cualquier caso la instalación consiste en descargar el paquete correspondiente e instalarlo.



En este caso en concreto, se está ejecutando un sistema operativo Ubuntu 18.04, donde Java no viene preinstalado, por lo tanto, lo primero que se debe hacer es añadir el paquete OpenJDK builds PPA (mantenido por el equipo de PPA para OpenJDK, un subequipo del equipo de Ubuntu OpenJDK, por lo que los paquetes proceden de una fuente oficial de Ubuntu). Para ello abrir una ventana de consola con permisos de root (ejecutando desde la línea de comandos "sudo -i") y teclear:

```
root@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@ubuntu-18:~# add-apt-repository ppa:openjdk-r/ppa

Más información: https://launchpad.net/~openjdk-r/+archive/ubuntu/ppa
Pulse [ENTRAR] para continuar o Ctrl+C para cancelar la adición.

Obj:1 http://es.archive.ubuntu.com/ubuntu bionic InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu bionic-updates InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu bionic-backports InRelease
Obj:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Des:5 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic InRelease [15,4 kB]
Des:6 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic/main i386 Packages [4
.424 B]
Des:7 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic/main amd64 Packages [
4.428 B]
Des:8 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic/main Translation-en [
1.344 B]
Descargados 25,6 kB en 2s (17,0 kB/s)
Leyendo lista de paquetes... Hecho
root@ubuntu-18:~# █
```

```
root@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@ubuntu-18:~# apt update
Obj:1 http://es.archive.ubuntu.com/ubuntu bionic InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu bionic-updates InRelease
Obj:3 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu bionic-backports InRelease
Obj:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se puede actualizar 1 paquete. Ejecute «apt list --upgradable» para verlo.
root@ubuntu-18:~# █
```

A continuación, se debe instalar el paquete `openjdk-11-jdk`:

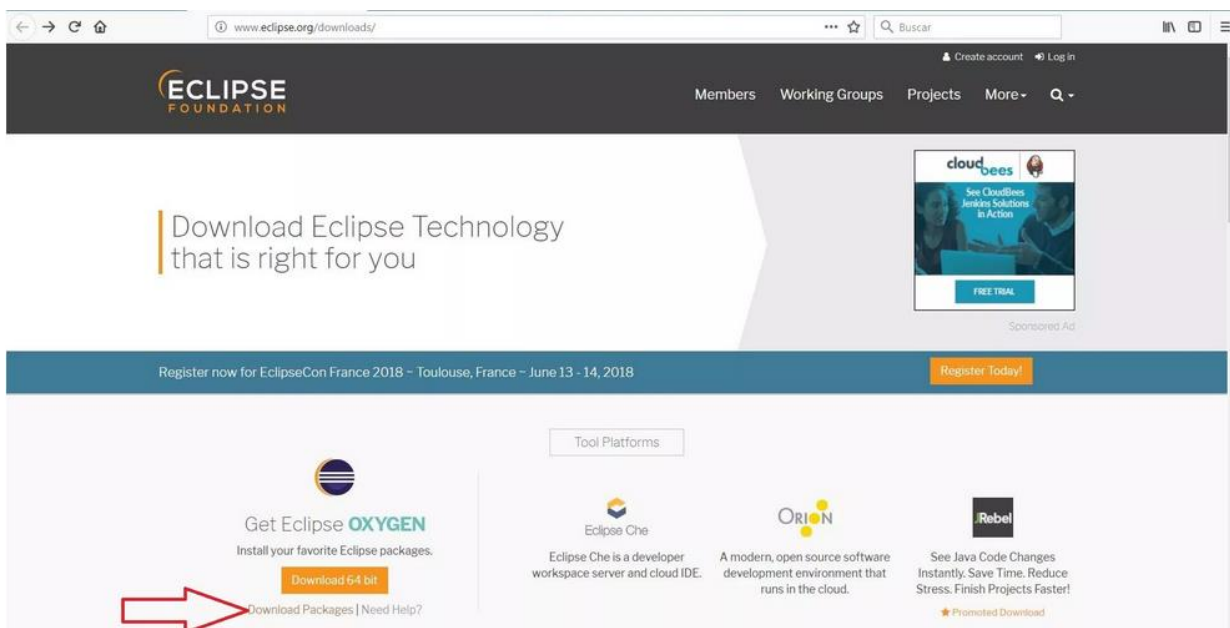
```
root@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@ubuntu-18:~# apt install openjdk-11-jdk
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
libatk-wrapper-java-jni libgif7 libice-dev libpthread-stubs0-dev libsm-dev
libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless
x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
Paquetes sugeridos:
default-jre libice-doc libsm-doc libxcb-doc libxt-doc openjdk-11-demo
openjdk-11-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho
fonts-wqy-microhei | fonts-wqy-zenhei
Se instalarán los siguientes paquetes NUEVOS:
ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
libatk-wrapper-java-jni libgif7 libice-dev libpthread-stubs0-dev libsm-dev
libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
openjdk-11-jre-headless x11proto-core-dev x11proto-dev xorg-sgml-doctools
xtrans-dev
0 actualizados, 23 nuevos se instalarán, 0 para eliminar y 1 no actualizados.
Se necesita descargar 236 MB de archivos.
```

Si todo se ha instalado correctamente se puede verificar la versión que se ha instalado:

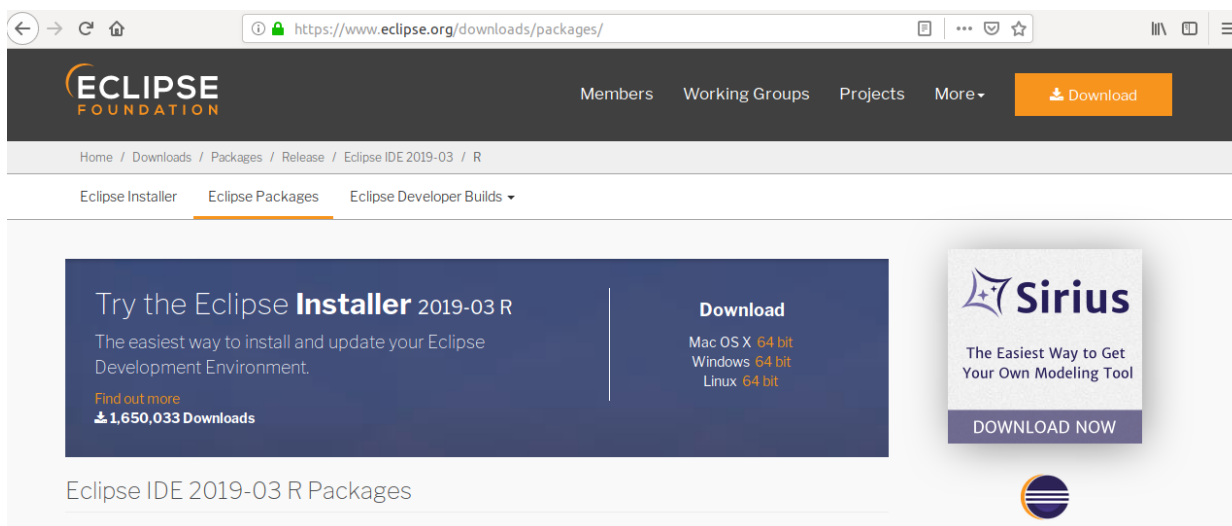
```
root@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@ubuntu-18:~# java -version
openjdk version "11.0.2" 2019-01-15
OpenJDK Runtime Environment (build 11.0.2+9-Ubuntu-3ubuntu118.04.3)
OpenJDK 64-Bit Server VM (build 11.0.2+9-Ubuntu-3ubuntu118.04.3, mixed mode, sha
ring)
```

Para programar en Java no es necesario ningún entorno de programación, con un sencillo editor de textos y el compilador incluido en el JDK sería suficiente, sin embargo, para simplificar un poco las tareas de desarrollo es preferible utilizar un entorno de desarrollo integrado (IDE). Para la realización de este TFG se ha decidido utilizar uno de los IDEs de código abierto más utilizados actualmente: ECLIPSE.

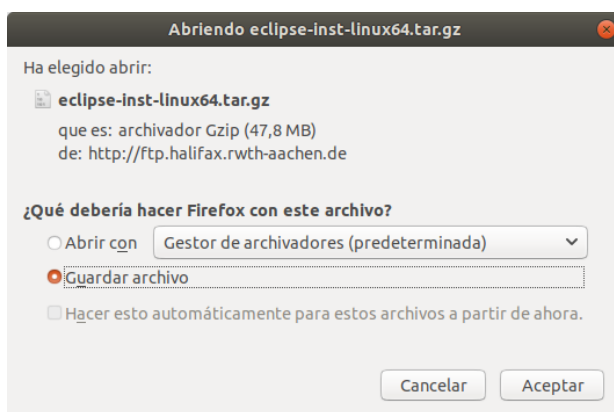
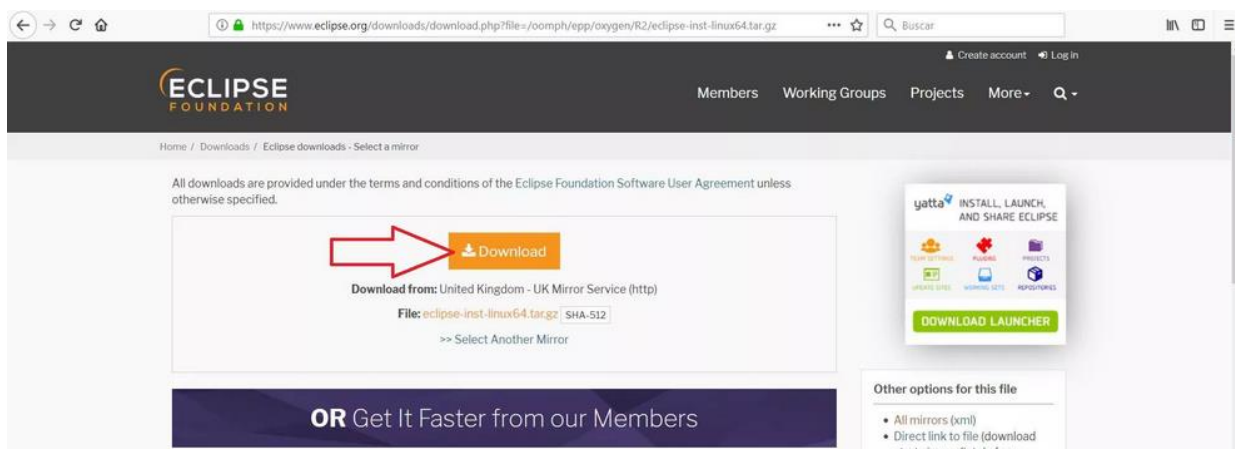
Para ello se debe descargar la última versión disponible desde su página oficial ([www.eclipse.org/downloads/](http://www.eclipse.org/downloads/)) y hacer click en "Download Packages":



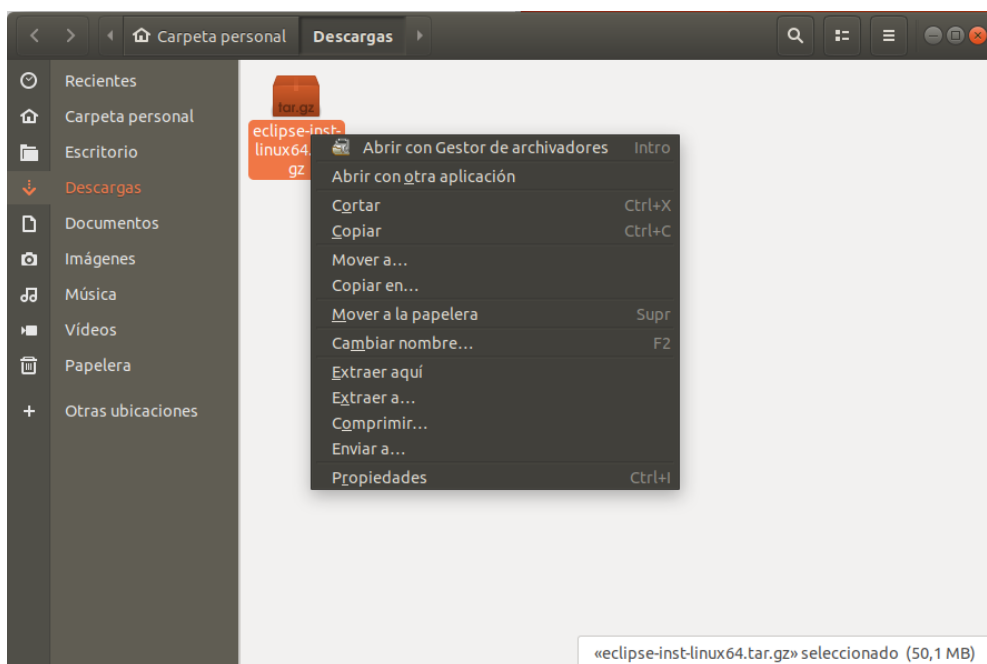
Seleccionar el instalador de Linux 64 bits:



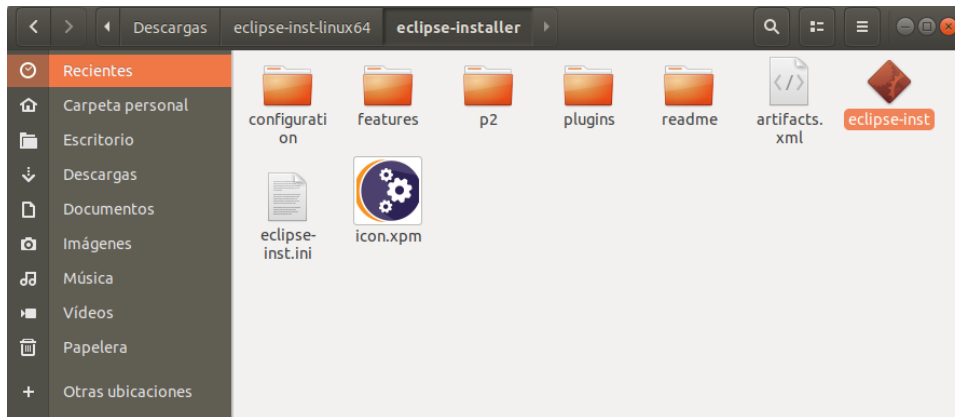
Descargar y guardar el archivo:



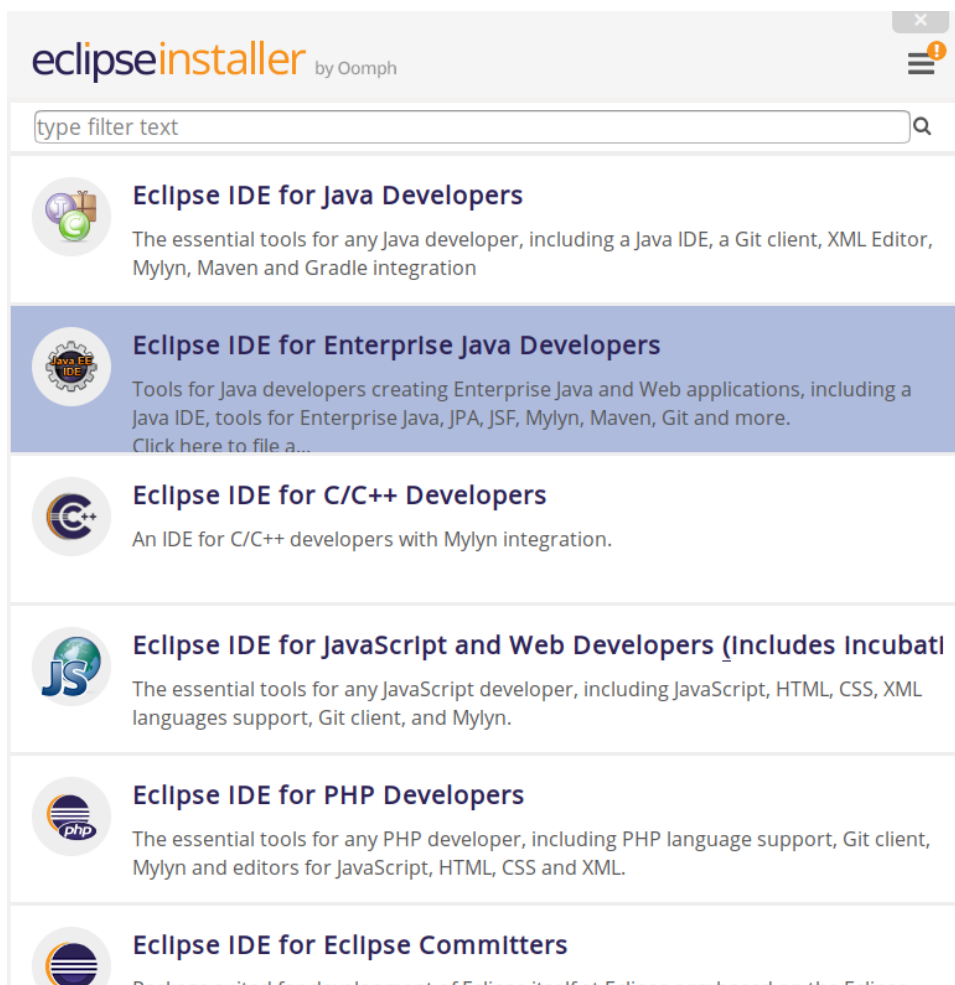
El siguiente paso es descomprimir el archivo de instalación. Ir a la carpeta “Descargas” y hacer click con el botón derecho sobre el instalador y luego en “Extraer aquí”:



Entrar en la carpeta descomprimida y hacer doble click en el archivo “eclipse-inst”:



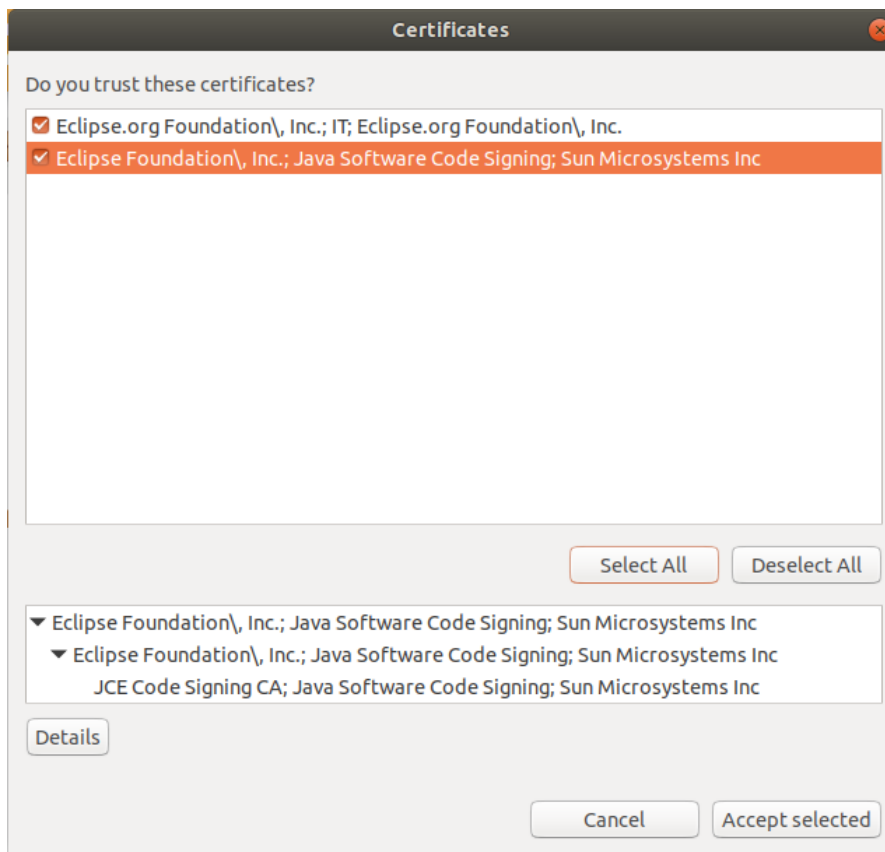
Se abrirá el instalador con todas las opciones. Hacer click en la segunda “Eclipse IDE for Enterprise Java Developers”:



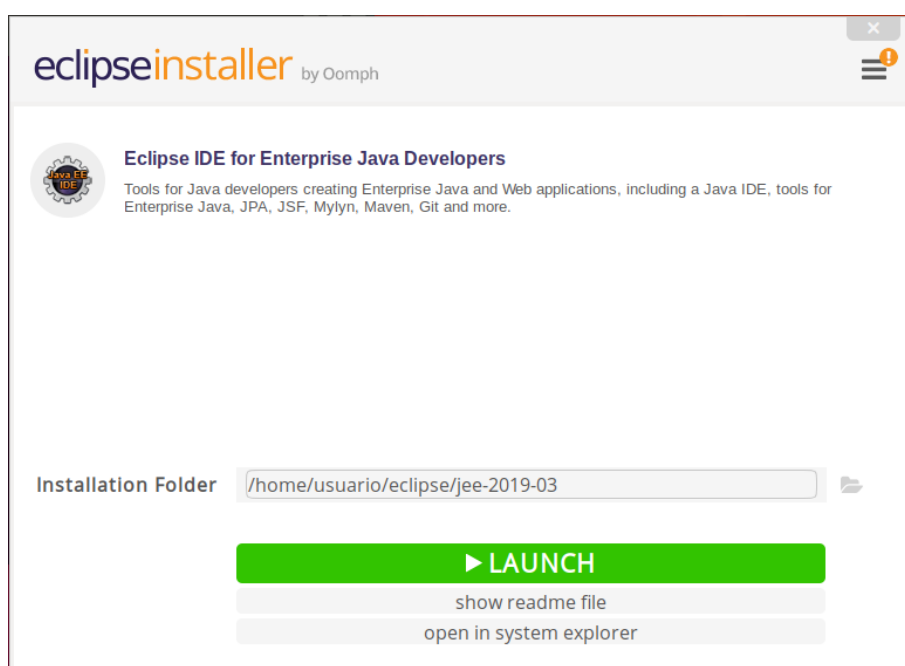
En la siguiente pantalla pulsar sobre "INSTALL":



Aceptar la licencia en la ventana que aparece y después aceptar los dos certificados que aparecen en otra ventana:



En este momento Eclipse ya está instalado. Para iniciarlo hacer click sobre “LAUNCH”:

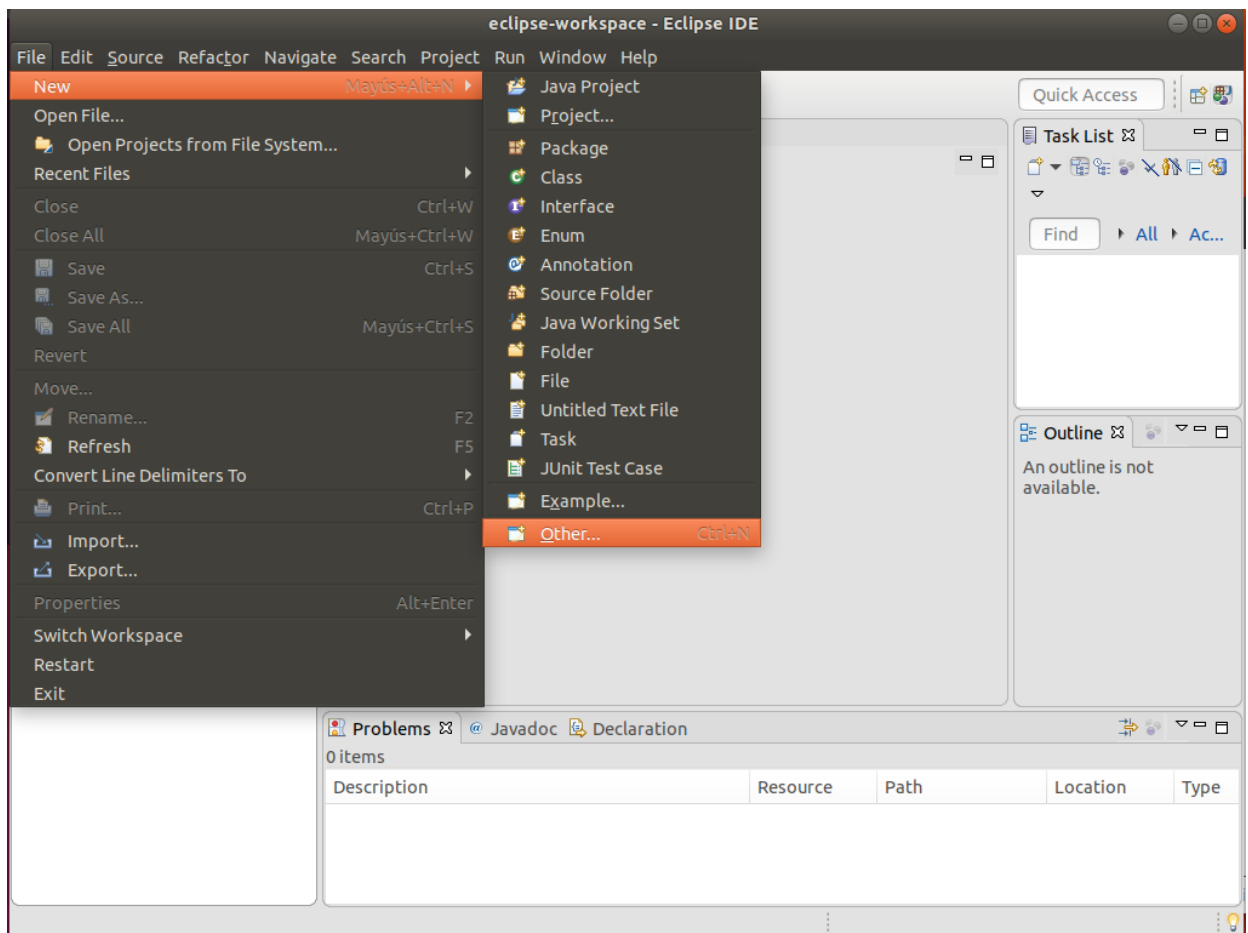


## 5.2 Creación de una aplicación Web de ejemplo

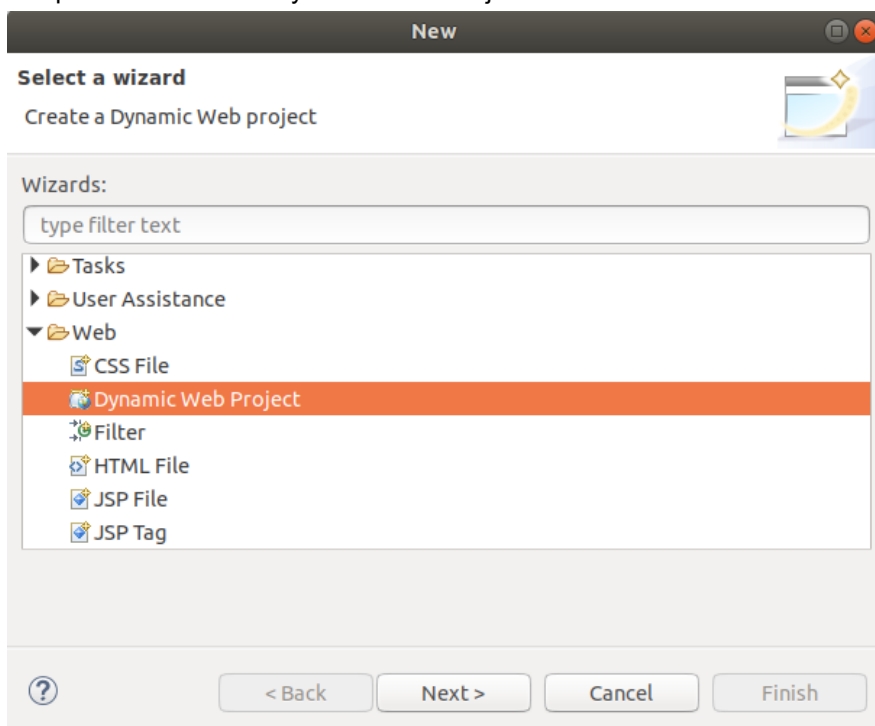
---

Se va a desarrollar una aplicación web de ejemplo para demostrar todo el proceso de integración continua. Con un sencillo programa tipo “Hola mundo” podría valer, sin embargo se va a desarrollar un Servlet, que es una clase que se ejecuta en un servidor web y el resultado de ejecución viaja por Internet para ser mostrado en un navegador web. El Servlet se encargará de mostrar un mensaje de bienvenida y a continuación los números enteros de 1 al 100.

Para crear el proyecto en Eclipse, cerramos la pantalla inicial de Eclipse y seleccionamos la opción “File->New->Other”:



Después seleccionar "Dynamic Web Project":





En el diálogo siguiente especificar el nombre del proyecto (EjemploServlet) y presionar el botón "Finish":

**New Dynamic Web Project**

**Dynamic Web Project**  
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location  
 Use default location  
Location:

Target runtime

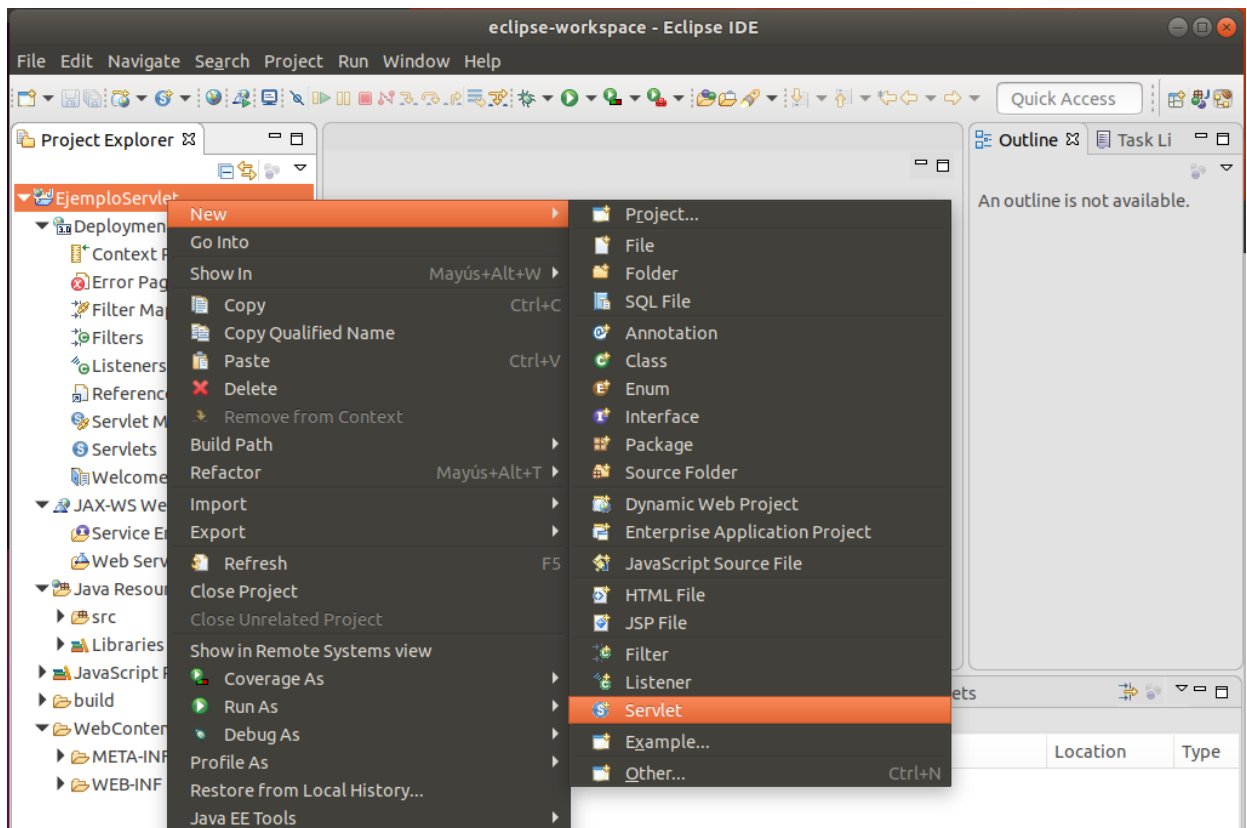
Dynamic web module version

Configuration  
   
The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.

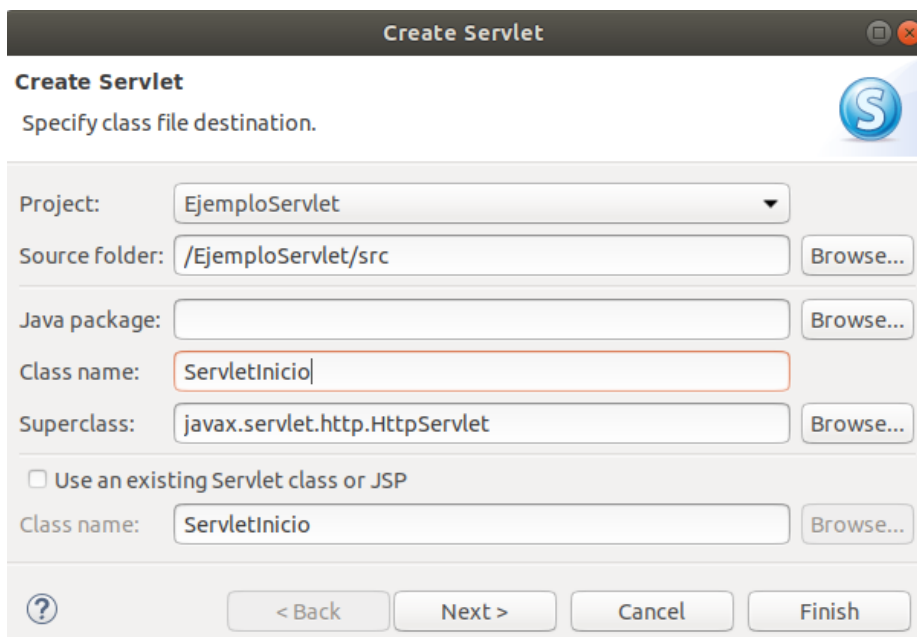
EAR membership  
 Add project to an EAR  
EAR project name:

Working sets  
 Add project to working sets   
Working sets:

Ahora clicar con el botón derecho sobre el nombre del proyecto y seleccionar “New->Servlet”:



En la siguiente pantalla especificar el nombre del Servlet en el campo “Class name” y pulsar el botón “Finish”, así se creará el esqueleto del Servlet:



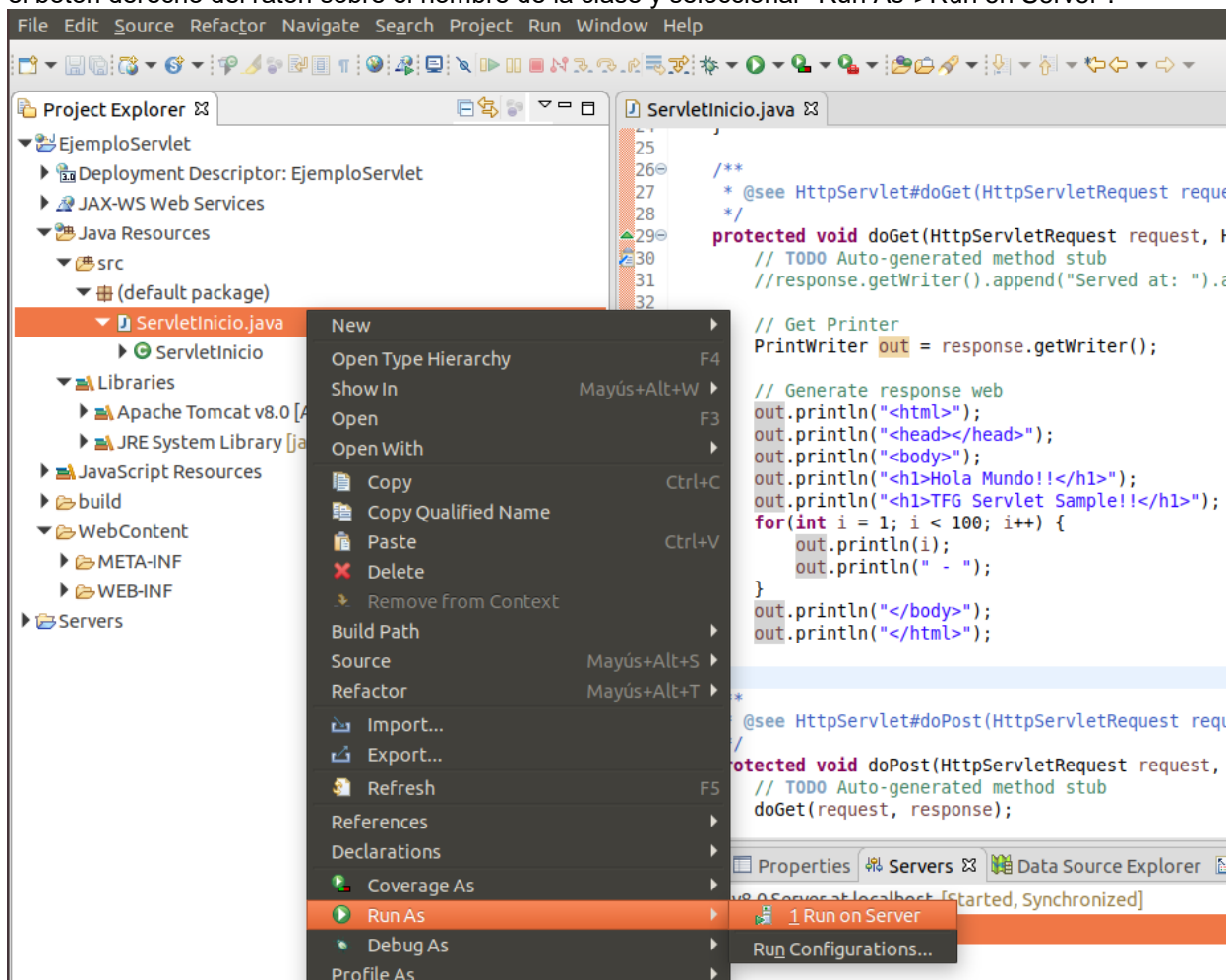
Se debe rellenar el método doGet() que es el que se llamará desde las página web sin formularios (doPost() si tuvieran formulario). El siguiente código crea una página web en HTML con un mensaje de bienvenida "Hola Mundo" y otro "TFG Servlet Sample" y luego escribe los números del 1 al 100:

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    // TODO Auto-generated method stub
    //response.getWriter().append("Served at: ").append(request.getContextPath());

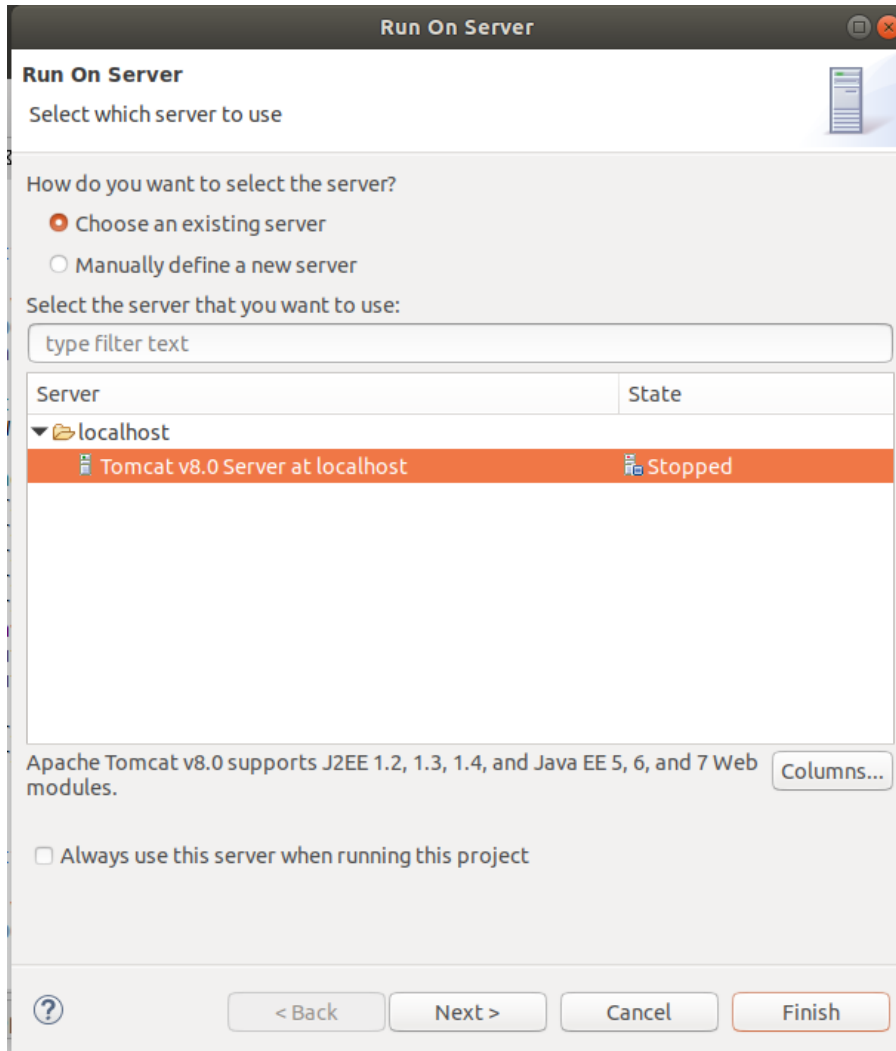
    // Get Printer
    PrintWriter out = response.getWriter();

    // Generate response web
    out.println("<html>");
    out.println("<head></head>");
    out.println("<body>");
    out.println("<h1>Hola Mundo!!</h1>");
    out.println("<h1>TFG Servlet Sample!!</h1>");
    for(int i = 1; i < 100; i++) {
        out.println(i);
        out.println(" - ");
    }
    out.println("</body>");
    out.println("</html>");
}
```

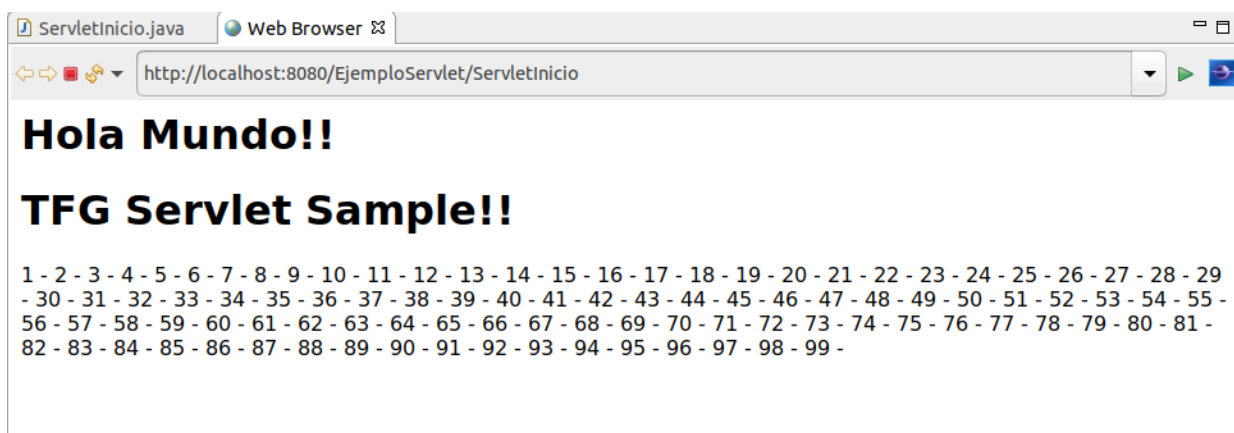
Para probar el Servlet se debe verificar que se ha compilado correctamente sin ningún error y clicar con el botón derecho del ratón sobre el nombre de la clase y seleccionar "Run As->Run on Server".



Se abrirá una ventana donde se deberá seleccionar sobre qué Server se va a ejecutar (el menos debe haber un servidor instalado en el sistema). Seleccionamos uno y pulsamos el botón "Finish" para que se ejecute el Servlet:

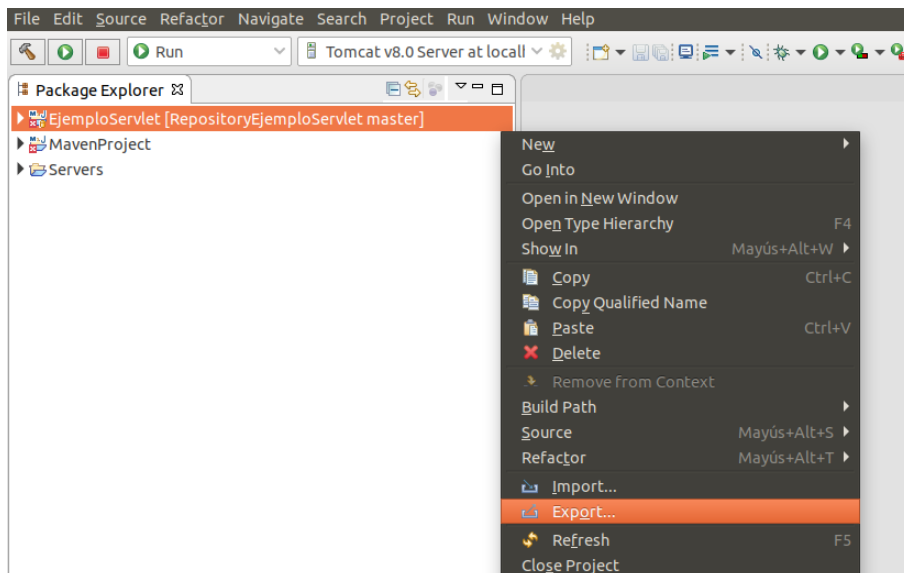


Si todo ha ido bien se abrirá una página web con el contenido HTML generado por el código fuente del Servlet:

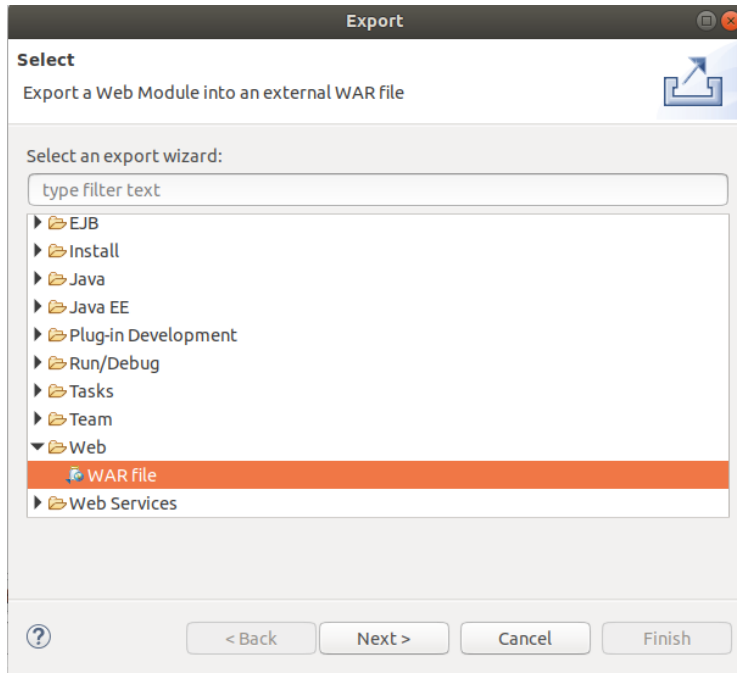


Normalmente, para distribuir las aplicaciones Web y los Servlets Java generados y pasarlos al entorno de pruebas o de producción, se suelen empaquetar en un tipo de especial de archivo JAR, conocido como WAR, que incluye la carpeta META-INF con el manifiesto como en los JAR, pero además incluye la carpeta WEB-INF que contiene la carpeta classes (donde se incluyen archivos .class de la aplicación), la carpeta lib (donde de incluyen otros JAR referenciados por la aplicación), el archivo web.xml y un descriptor de despliegue propio del servidor de aplicaciones donde se va a desplegar la aplicación.

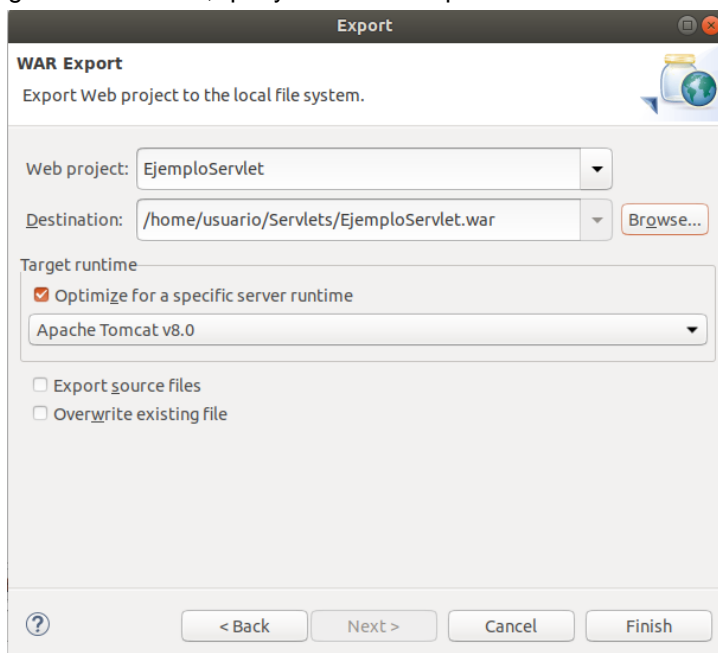
Para crear un archivo WAR desde Eclipse, seleccionar el proyecto que se quiere exportar y con el botón derecho del ratón seleccionar las opciones "Export":



Y en la ventana que se abre, seleccionar "Web->WAR file":



A continuación darle el directorio donde se va a generar y pulsar el botón "Finish" para terminar y que se genere el fichero, que ya estará listo para distribuirlo:



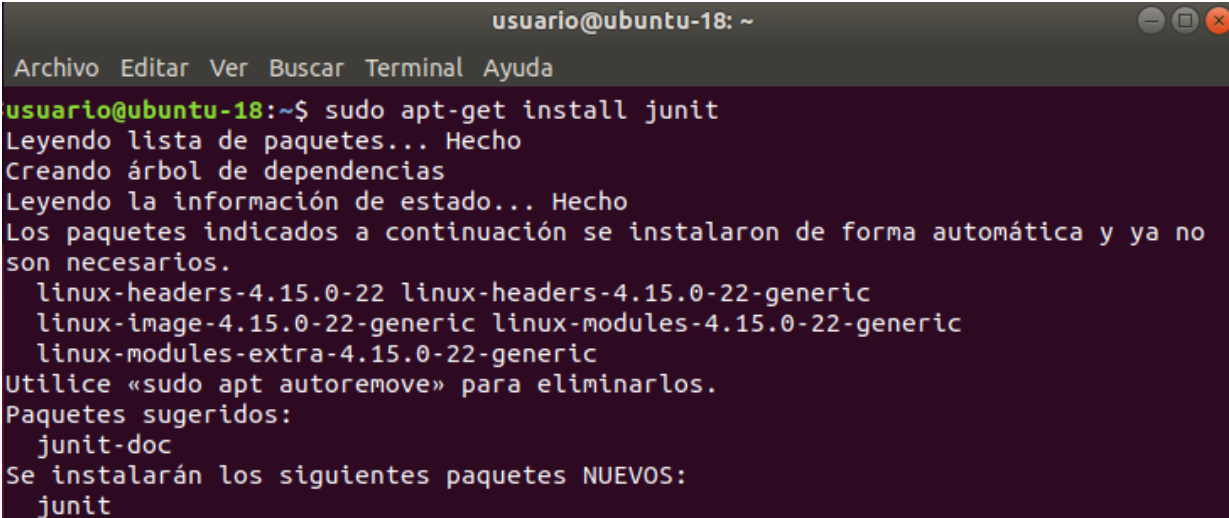
### 5.3 Instalación de JUnit

---

JUnit es un framework para escribir y ejecutar pruebas unitarias en lenguaje Java. Normalmente se utiliza para verificar que dado unos valores de entrada, la función devuelve unos valores de salida esperados. Si efectivamente devuelve un valor correcto entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso contrario JUnit devolverá un fallo. Algunos entornos de desarrollo como NetBeans y Eclipse cuentan con plug-ins para integrar su funcionalidad y con plantillas para generar algunas pruebas automáticamente.

Instalarlo en Ubuntu es muy sencillo, tan solo hay que ejecutar los siguientes comandos en un terminal:

```
sudo apt-get update
sudo apt-get install junit
```



```
usuario@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@ubuntu-18:~$ sudo apt-get install junit
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  linux-headers-4.15.0-22 linux-headers-4.15.0-22-generic
  linux-image-4.15.0-22-generic linux-modules-4.15.0-22-generic
  linux-modules-extra-4.15.0-22-generic
Utilice «sudo apt autoremove» para eliminarlos.
Paquetes sugeridos:
  junit-doc
Se instalarán los siguientes paquetes NUEVOS:
  junit
```

### 5.4 Instalación de Maven

---

Maven es una utilidad que permite controlar el ciclo de vida completo de un programa. Es una herramienta de construcción que permite gestionar las dependencias del proyecto, sus tests, su documentación, compilación, distribución, mailing list, etc. Además está perfectamente integrada con herramientas de control de configuración como Git, SVN, etc.

Maven utiliza un modelo de objeto de proyecto (POM) que es esencialmente un archivo XML que contiene información sobre el proyecto, los detalles de configuración, las dependencias del proyecto, etc.

Para instalarlo en Ubuntu se debe abrir un terminal (Ctrl+Alt+T) e introducir el siguiente comando para actualizando el índice de paquetes:

```
sudo apt update
```

```
usuario@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@ubuntu-18:~$ sudo apt update
Obj:1 http://es.archive.ubuntu.com/ubuntu bionic InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Obj:4 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic InRelease
Ign:5 http://pkg.jenkins.io/debian-stable binary/ InRelease
Des:6 http://pkg.jenkins.io/debian-stable binary/ Release [2.042 B]
Des:7 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Des:8 http://pkg.jenkins.io/debian-stable binary/ Release.gpg [181 B]
Leyendo lista de paquetes... Hecho
```

A continuación, instalar Maven escribiendo el siguiente comando en el terminal:  
sudo apt install Maven

```
usuario@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@ubuntu-18:~$ sudo apt install maven
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 libaopalliance-java libapache-pom-java libatinject-jsr330-api-java
 libcdi-api-java libcommons-cli-java libcommons-io-java libcommons-lang3-java
 libcommons-parent-java libgeronimo-annotation-1.3-spec-java
 libgeronimo-interceptor-3.0-spec-java libguava-java libguice-java
 libhawtjni-runtime-java libjansi-java libjansi-native-java libjsr305-java
```

Terminada su instalación se puede verificar la versión instalada escribiendo el siguiente comando en el terminal:

mvn -version

```
usuario@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@ubuntu-18:~$ mvn -version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 1.8.0_191, vendor: Oracle Corporation, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: es_ES, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-48-generic", arch: "amd64", family: "unix"
```

Para poder utilizarlo desde cualquier directorio es necesario configurar algunas variables de entorno. Para ello, abrir un editor de texto y crear un nuevo archivo llamado maven.sh dentro del directorio /etc/profile.d/ y añadir el siguiente contenido:

```
export JAVA_HOME=/usr/lib/jvm/default-java
export M2_HOME=/opt/maven
export MAVEN_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}
```

Se debe hacer que el script sea ejecutable escribiendo en el terminal:  
sudo chmod +x /etc/profile.d/maven.sh



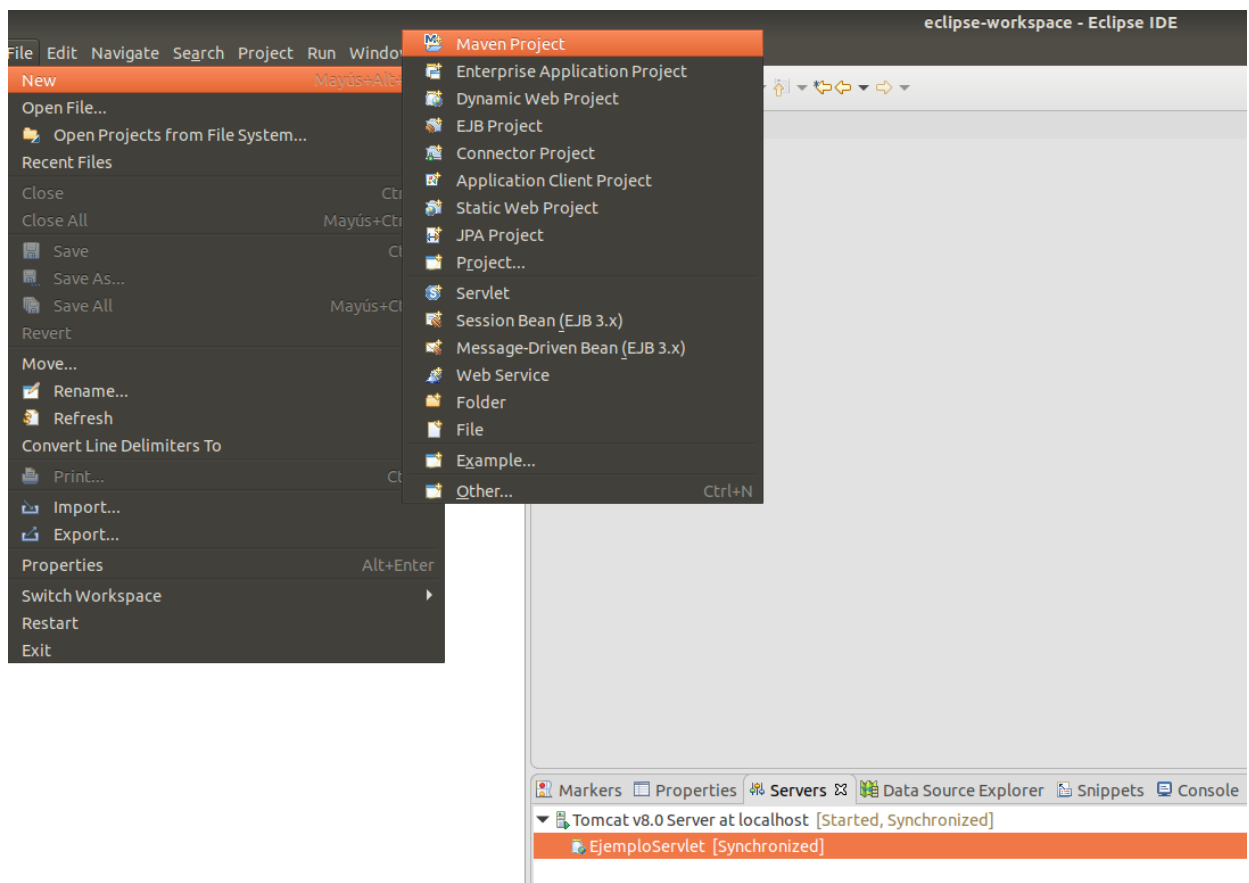
Y finalmente ejecutarlo para cargar las variables de entorno usando el siguiente comando:  
`source /etc/profile.d/maven.sh`

Para verificar que se han cargado correctamente ejecutar el comando:  
`printenv`

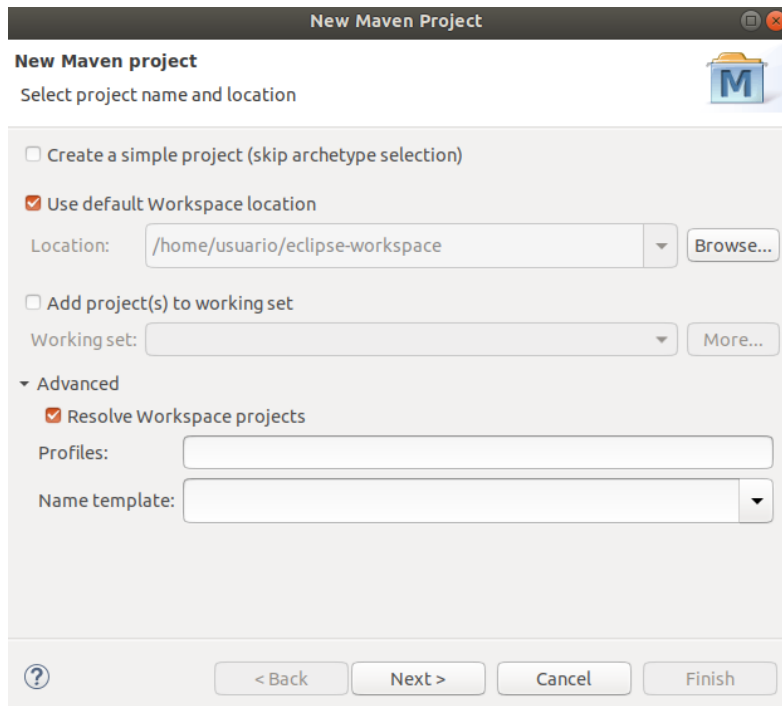
## 5.5 Generación de los ejecutables con Maven

Maven se puede utilizar desde la línea de comandos o desde un IDE como Eclipse, como se va a detallar a continuación.

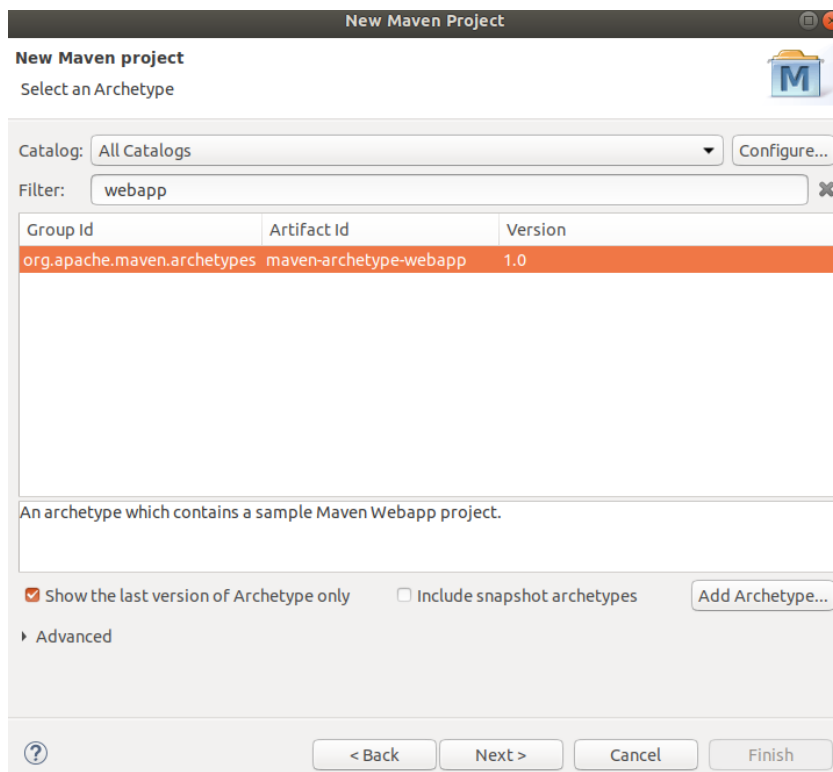
Empezar creando un nuevo proyecto en Eclipse, seleccionando la entrada de menú “File->New->Maven Project”:



En la siguiente pantalla se selecciona la ubicación por defecto y se pulsa el botón “Next”:



En la siguiente ventana filtramos por “webapp” y seleccionamos el arquetipo que aparece:



Pulsamos el botón "Next" y en la siguiente pantalla rellenamos el "Group Id" y el "Artifact Id" y pulsamos el botón "Finish":

New Maven Project

Specify Archetype parameters

Group Id: com.tfg

Artifact Id: MavenProject

Version: 0.0.1-SNAPSHOT

Package: com.tfg.MavenProject

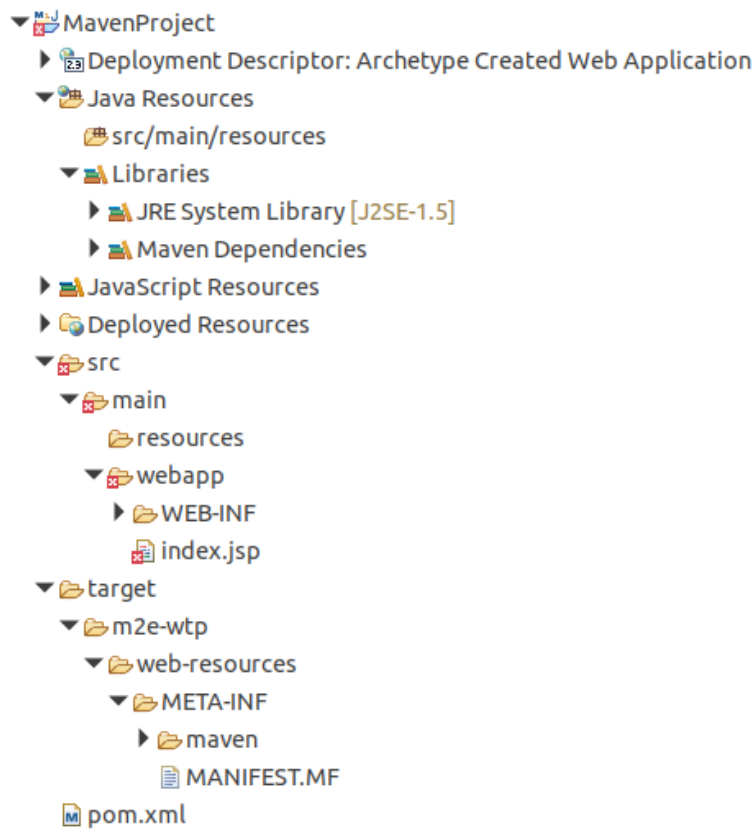
Properties available from archetype:

Name	Value
------	-------

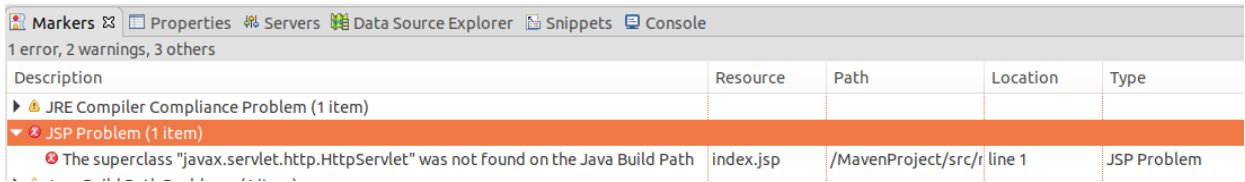
Advanced

< Back Next > Cancel Finish

De esta manera ya está creado todo el esqueleto del proyecto Maven:

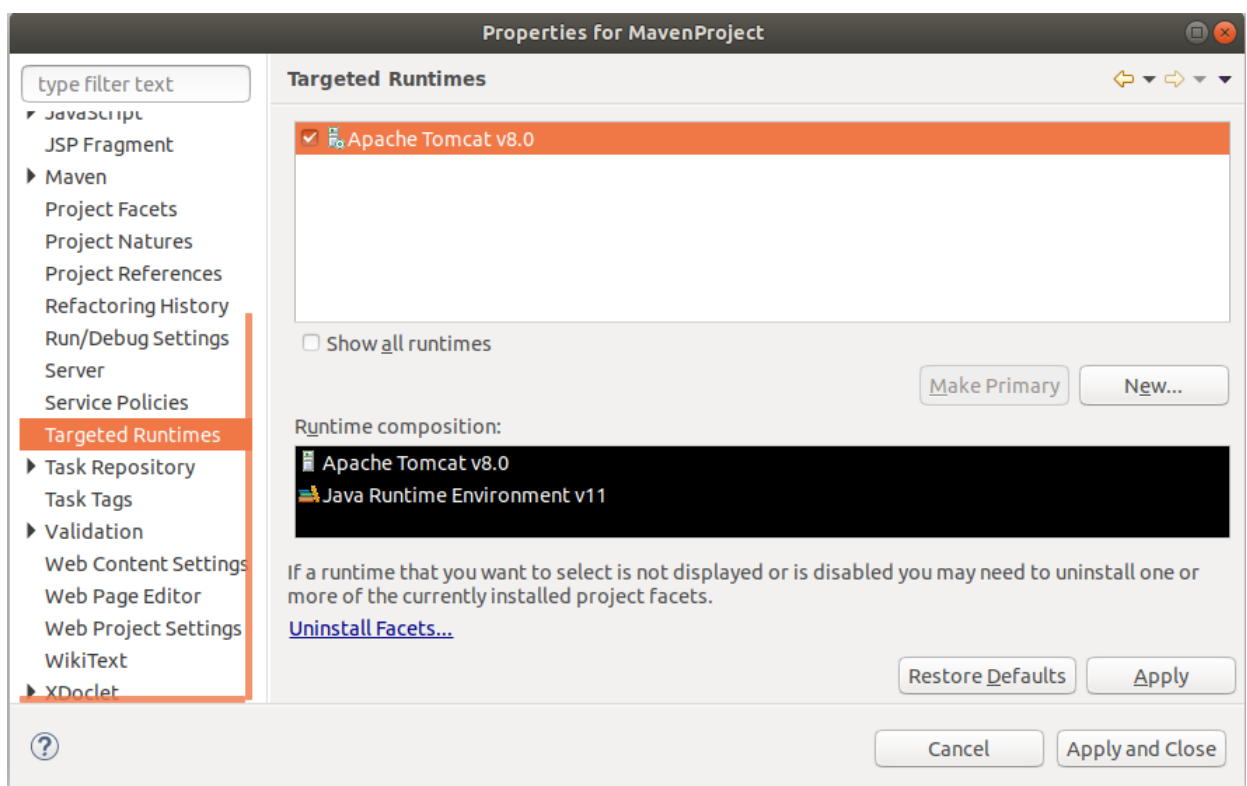


Pueden aparecer errores si no se encuentra alguna dependencia, como por ejemplo:

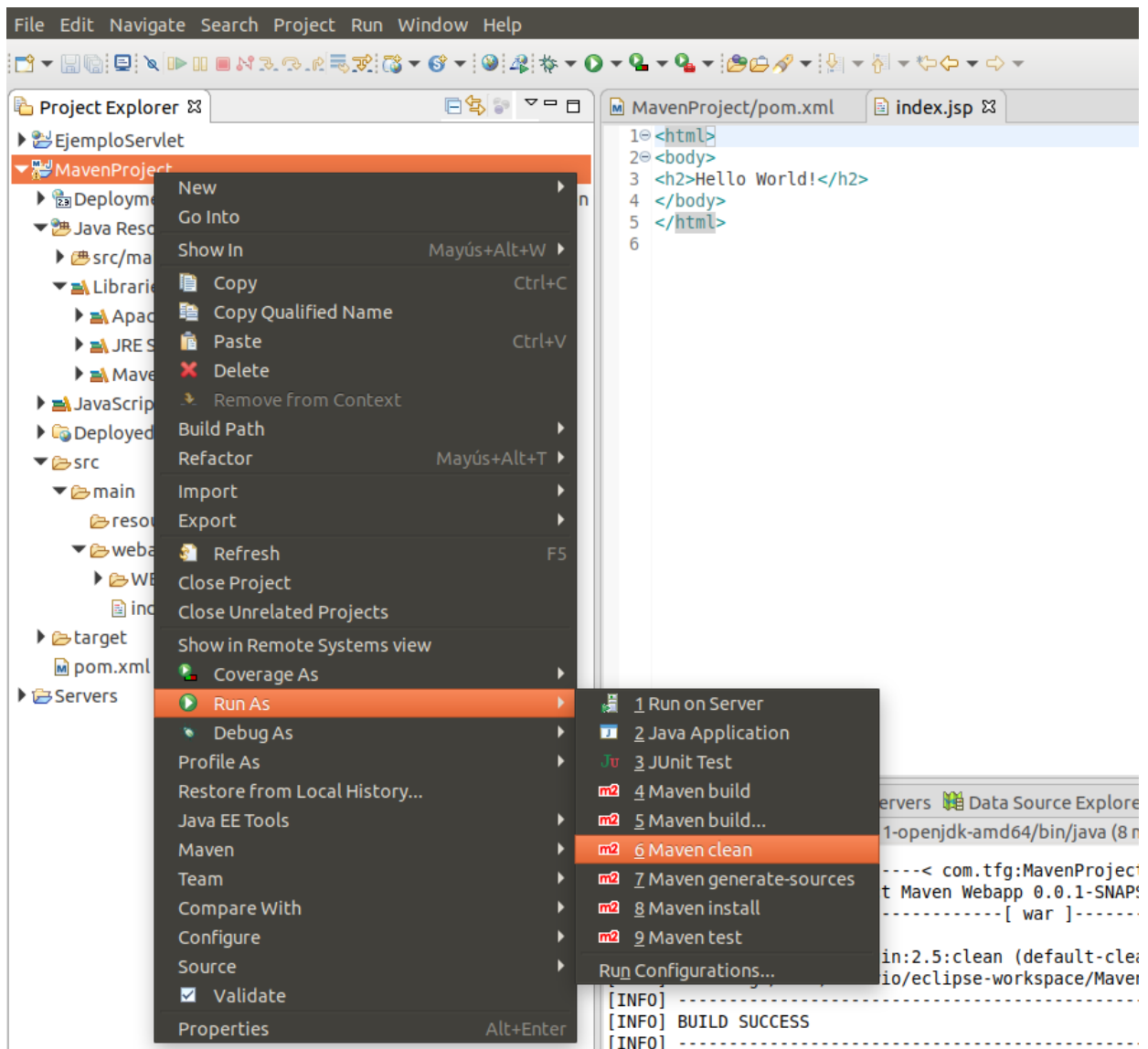


Description	Resource	Path	Location	Type
JRE Compiler Compliance Problem (1 item)				
JSP Problem (1 item)				
The superclass "javax.servlet.http.HttpServlet" was not found on the Java Build Path	index.jsp	/MavenProject/src/r/line 1		JSP Problem

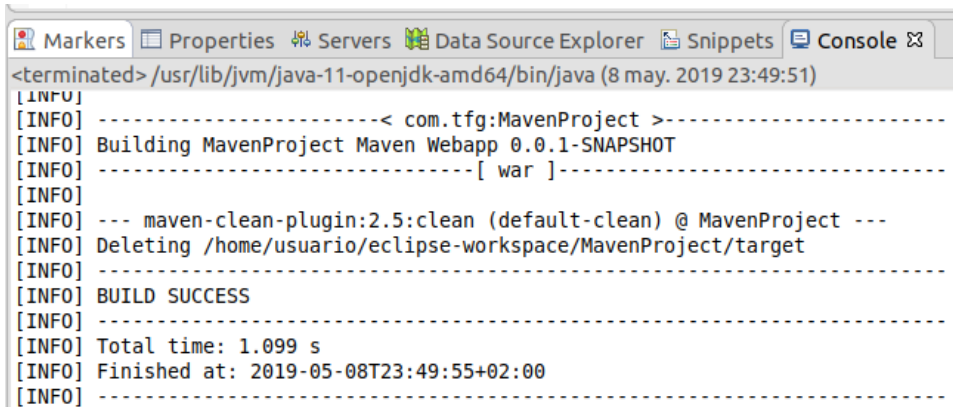
Para solucionarlo, añadir la dependencia manualmente o añadir algún servidor pulsando con el botón derecho del ratón sobre las "Properties" del proyecto y en "Targeted Runtimes" añadir algún servidor de los disponibles:



Hacer una primera compilación completa para asegurar que no hay dependencias y todo está bien construido, pinchando sobre el proyecto y seleccionando “Run As->Maven clean”:

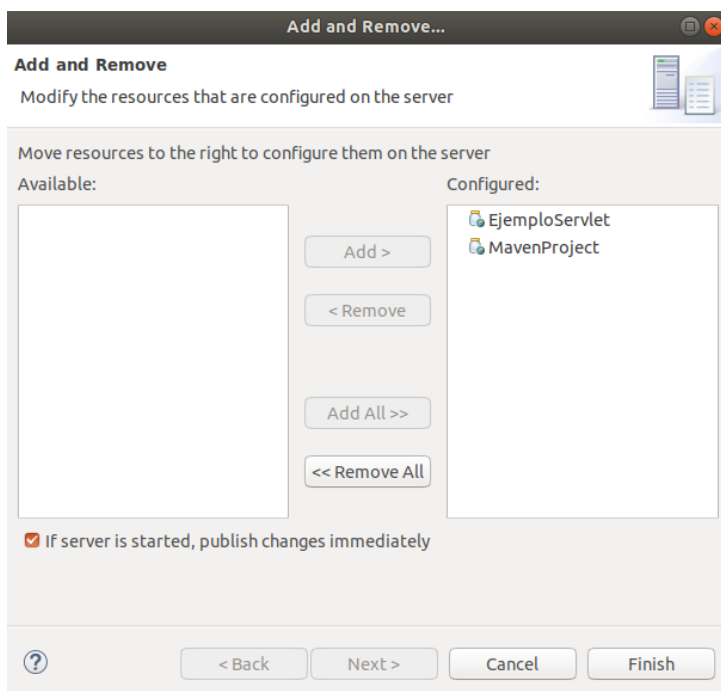


Si todo está bien aparecerá un mensaje de que está bien construido:

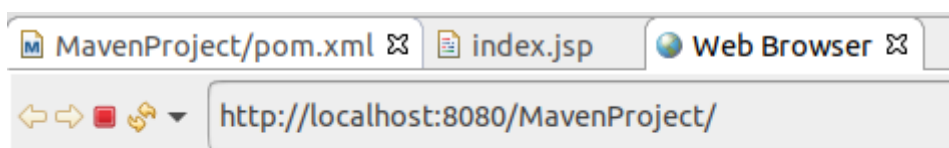


```
<terminated> /usr/lib/jvm/java-11-openjdk-amd64/bin/java (8 may. 2019 23:49:51)
[INFO]
[INFO] -----< com.tfg:MavenProject >-----
[INFO] Building MavenProject Maven Webapp 0.0.1-SNAPSHOT
[INFO] -----[ war ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ MavenProject ---
[INFO] Deleting /home/usuario/eclipse-workspace/MavenProject/target
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 1.099 s
[INFO] Finished at: 2019-05-08T23:49:55+02:00
[INFO] -----
```

Para probarlo, arrancar el servidor en la pestaña “Servers”, pulsando sobre él con el botón derecho del ratón y seleccionando “Start” y después pinchar de nuevo sobre él y seleccionar “Add and remove”, para después añadir el proyecto “MavenProject”:



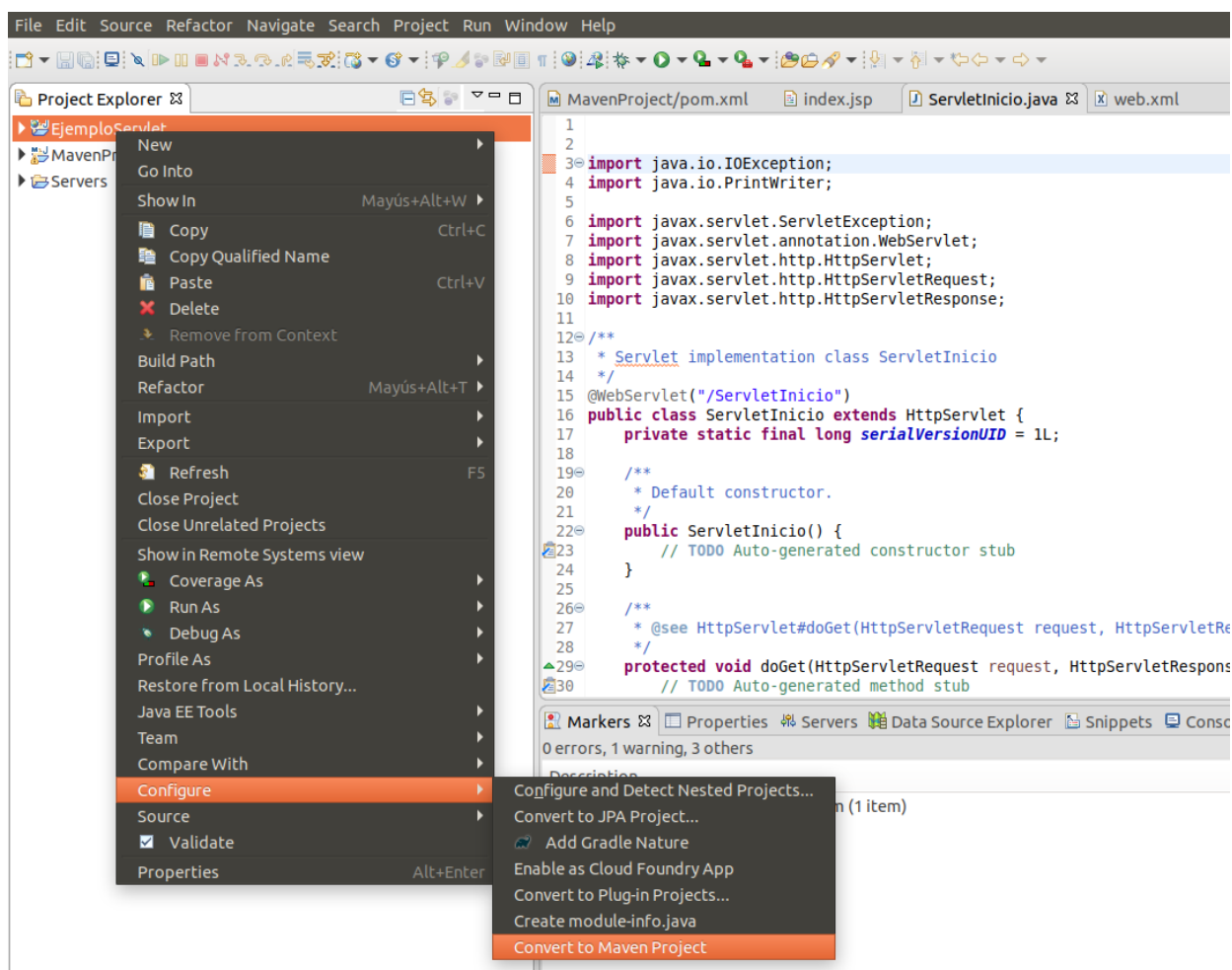
Para ejecutarlo clicar con el botón derecho del ratón sobre el nombre “MavenProject” y seleccionar “Run As->Run on Server” y se ejecutará el código que tengamos, que de momento solo muestra un mensaje de “Hello World!”:



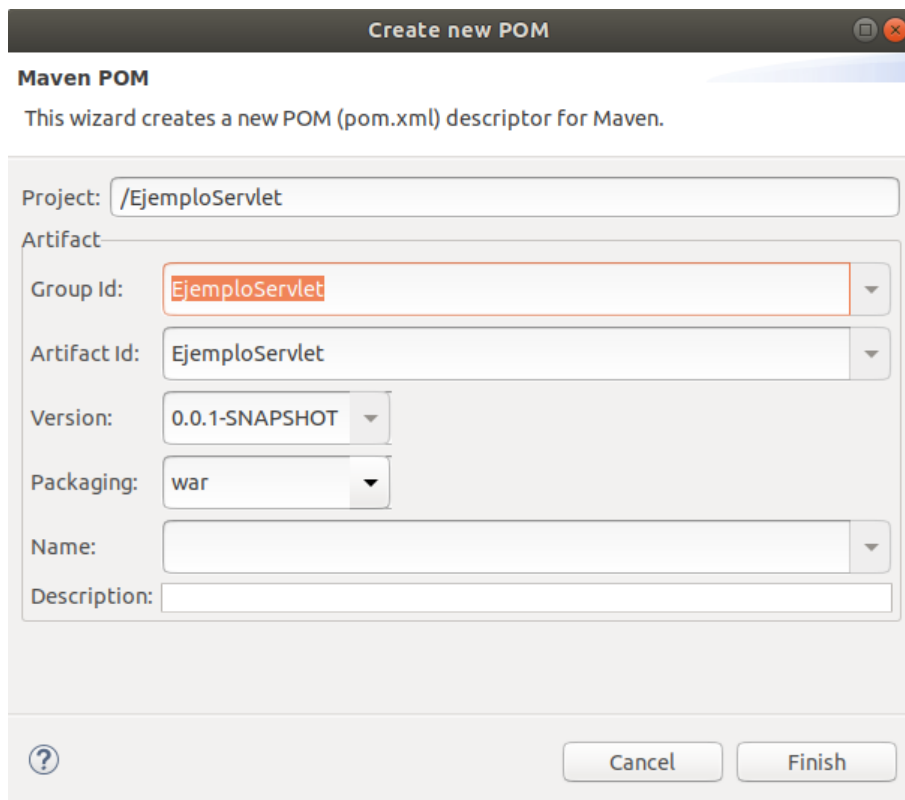
# Hello World!

Todo este procedimiento es para crear un proyecto Maven desde cero, pero si lo que se busca es convertir un proyecto ya creado a Maven se puede hacer directamente utilizando plugins de Eclipse como por ejemplo m2eclipse o el que viene por defecto integrado en la versión JEE de Eclipse.

Por ejemplo, para convertir el proyecto creado anteriormente “EjemploServlet” a Maven, hay que pinchar con el botón derecho del ratón sobre el nombre del proyecto y seleccionar “Configure->Convert to Maven Project”:



Se abrirá una ventana para la creación del POM, donde los datos que aparecen por defecto suelen valer, pulsamos “Finish” y se creará el fichero POM para Maven:



**Create new POM**

**Maven POM**

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /EjemploServlet

Artifact

Group Id: EjemploServlet

Artifact Id: EjemploServlet

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:

Cancel Finish

El fichero POM generado es muy sencillo y contiene lo siguiente:

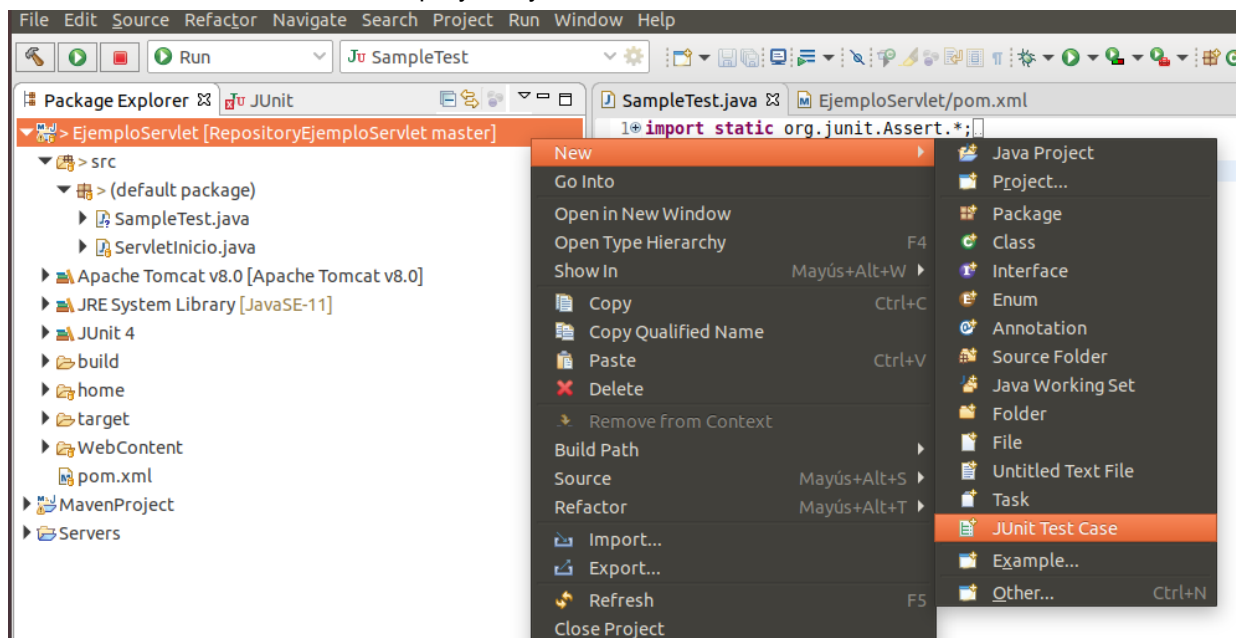


```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sche
2 <modelVersion>4.0.0</modelVersion>
3 <groupId>EjemploServlet</groupId>
4 <artifactId>EjemploServlet</artifactId>
5 <version>0.0.1-SNAPSHOT</version>
6 <packaging>war</packaging>
7 <build>
8 <sourceDirectory>src</sourceDirectory>
9 <plugins>
10 <plugin>
11 <artifactId>maven-compiler-plugin</artifactId>
12 <version>3.8.0</version>
13 <configuration>
14 <release>11</release>
15 </configuration>
16 </plugin>
17 <plugin>
18 <artifactId>maven-war-plugin</artifactId>
19 <version>3.2.1</version>
20 <configuration>
21 <warSourceDirectory>WebContent</warSourceDirectory>
22 </configuration>
23 </plugin>
24 </plugins>
25 </build>
26 </project>
```

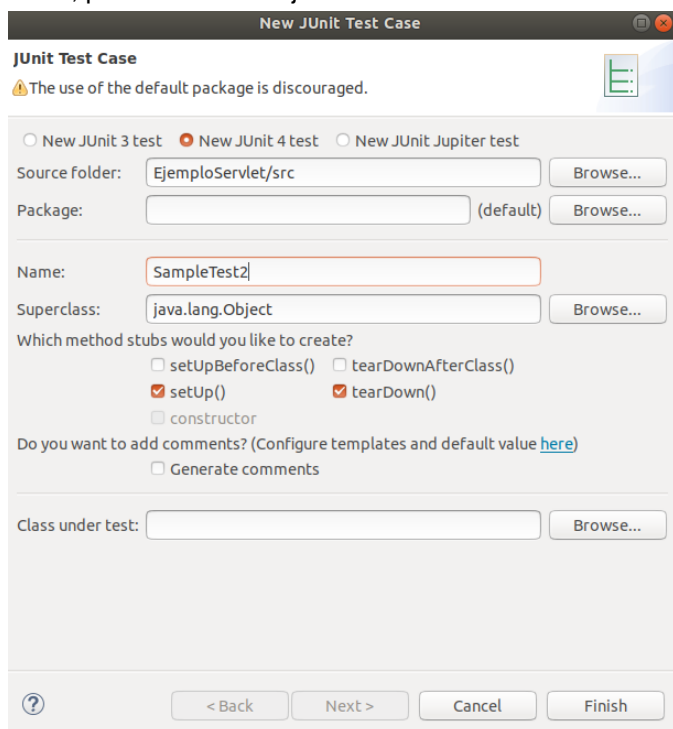


## 5.6 Ejecución de pruebas unitarias con JUnit y Maven

Para ejecutar las pruebas desde Eclipse, añadir una nueva clase que contendrá los tests, pinchando con el botón derecho del ratón sobre el proyecto y seleccionando la entrada de menú “New->JUnit Test Case”:



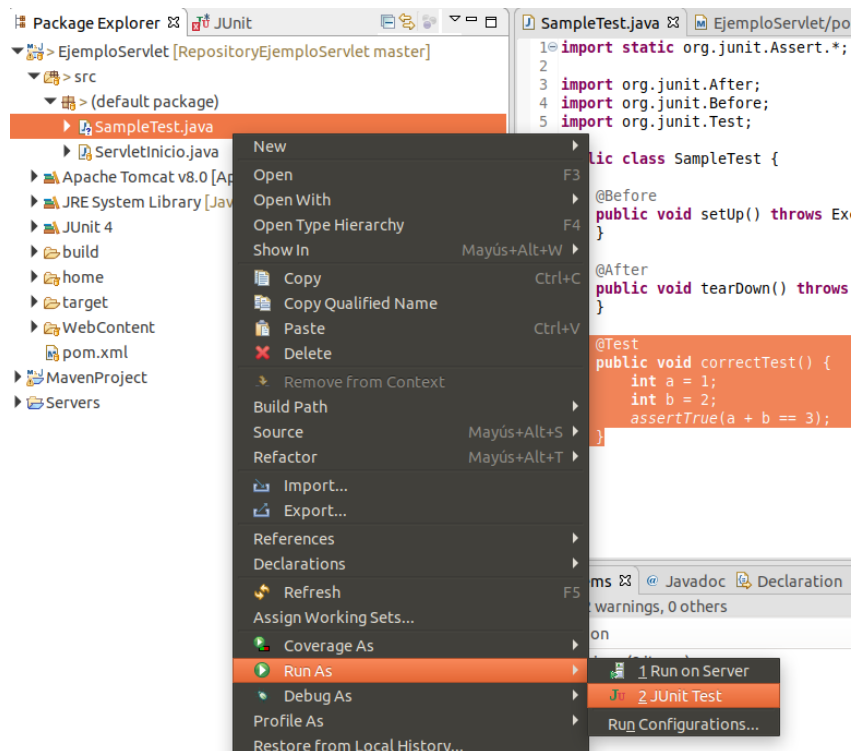
En la ventana que se abre darle un nombre, por ejemplo “SampleTest” y seleccionar si va a realizar tests sobre alguna clase ya creada. En este caso se va a hacer un ejemplo sencillo, que no pruebe ninguna clase, por lo tanto se deja vacío:



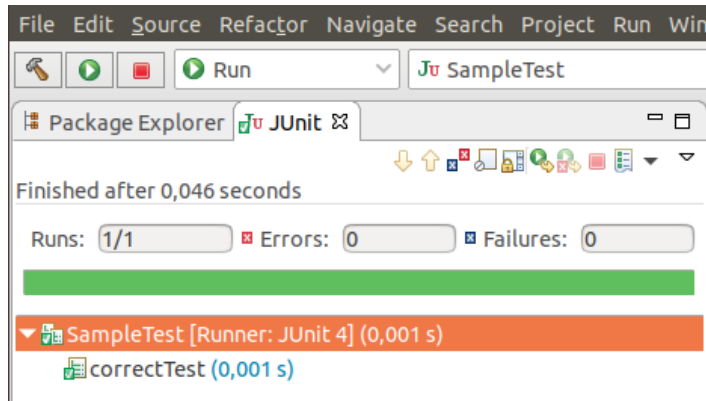
A continuación se rellena el código de la prueba, que en este caso va a ser muy sencillo y va a consistir en verificar el resultado de una suma:

```
SampleTest.java EjemploServlet/pom.xml
1 import static org.junit.Assert.*;
2
3 import org.junit.After;
4 import org.junit.Before;
5 import org.junit.Test;
6
7 public class SampleTest {
8
9     @Before
10    public void setUp() throws Exception {
11    }
12
13    @After
14    public void tearDown() throws Exception {
15    }
16
17    @Test
18    public void correctTest() {
19        int a = 1;
20        int b = 2;
21        assertTrue(a + b == 3);
22    }
23 }
```

Ahora pulsando con el botón derecho del ratón sobre la clase y seleccionando “Runs As->JUnit Test”:



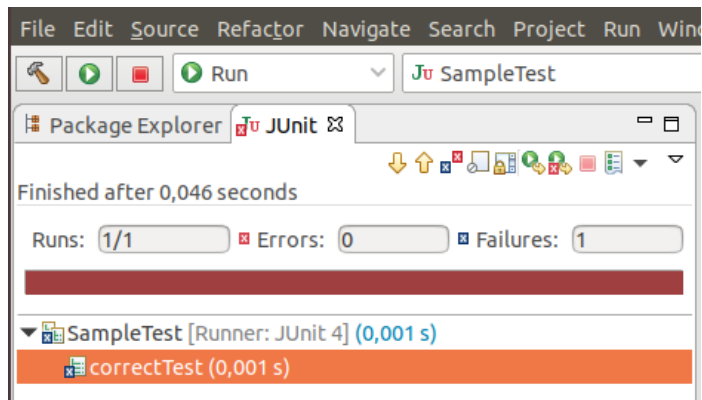
Se ejecutará el test y nos mostrará el resultado que en este caso es satisfactorio:



Pero si cambiamos el código del test por este otro:

```
SampleTest.java x EjemploServlet/pom.xml
1 import static org.junit.Assert.*;
2
3 import org.junit.After;
4 import org.junit.Before;
5 import org.junit.Test;
6
7 public class SampleTest {
8
9     @Before
10    public void setUp() throws Exception {
11    }
12
13    @After
14    public void tearDown() throws Exception {
15    }
16
17    @Test
18    public void correctTest() {
19        int a = 1;
20        int b = 2;
21        assertTrue(a + b == 4);
22    }
23
24 }
```

El resultado de la ejecución de las pruebas será un fallo:



Para ejecutar las pruebas desde Maven es necesario modificar el fichero pom.xml y añadirle la dependencia a JUnit, introduciendo las siguientes líneas:

```
ServletInicio.java EjemploServlet/pom.xml SampleTest.java
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http:
2 <modelVersion>4.0.0</modelVersion>
3 <groupId>EjemploServlet</groupId>
4 <artifactId>EjemploServlet</artifactId>
5 <version>0.0.1-SNAPSHOT</version>
6 <packaging>war</packaging>
7 <dependencies>
8 <dependency>
9 <groupId>junit</groupId>
10 <artifactId>junit</artifactId>
11 <version>4.12</version>
12 </dependency>
13 </dependencies>
14 <build>
15 <sourceDirectory>src</sourceDirectory>
16 <plugins>
17 <plugin>
18 <artifactId>maven-compiler-plugin</artifactId>
19 <version>3.8.0</version>
20 <configuration>
21 <release>11</release>
22 </configuration>
23 </plugin>
24 <plugin>
25 <artifactId>maven-war-plugin</artifactId>
26 <version>3.2.1</version>
27 <configuration>
28 <warSourceDirectory>WebContent</warSourceDirectory>
29 </configuration>
30 </plugin>
31 </plugins>
32 </build>
33 </project>
```


## 5.7 Código fuente y gestión de su configuración con GIT

A continuación se detalla el proceso para subir el proyecto Java anterior a GitHub utilizando Eclipse. Para ello es necesario tener instalado un plugin en Eclipse como eGit, aunque en algunas versiones del IDE ya viene instalado. También es necesario tener un repositorio creado en GitHub. Para la realización de este proyecto se ha creado el siguiente repositorio:

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

 albertoelez ▾



Repository name \*

/ DevOps\_TFG 

Great repository names are short and memorable. Need inspiration? How about **musical-broccoli**?

Description (optional)

DevOps TFG Repository

-  **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.

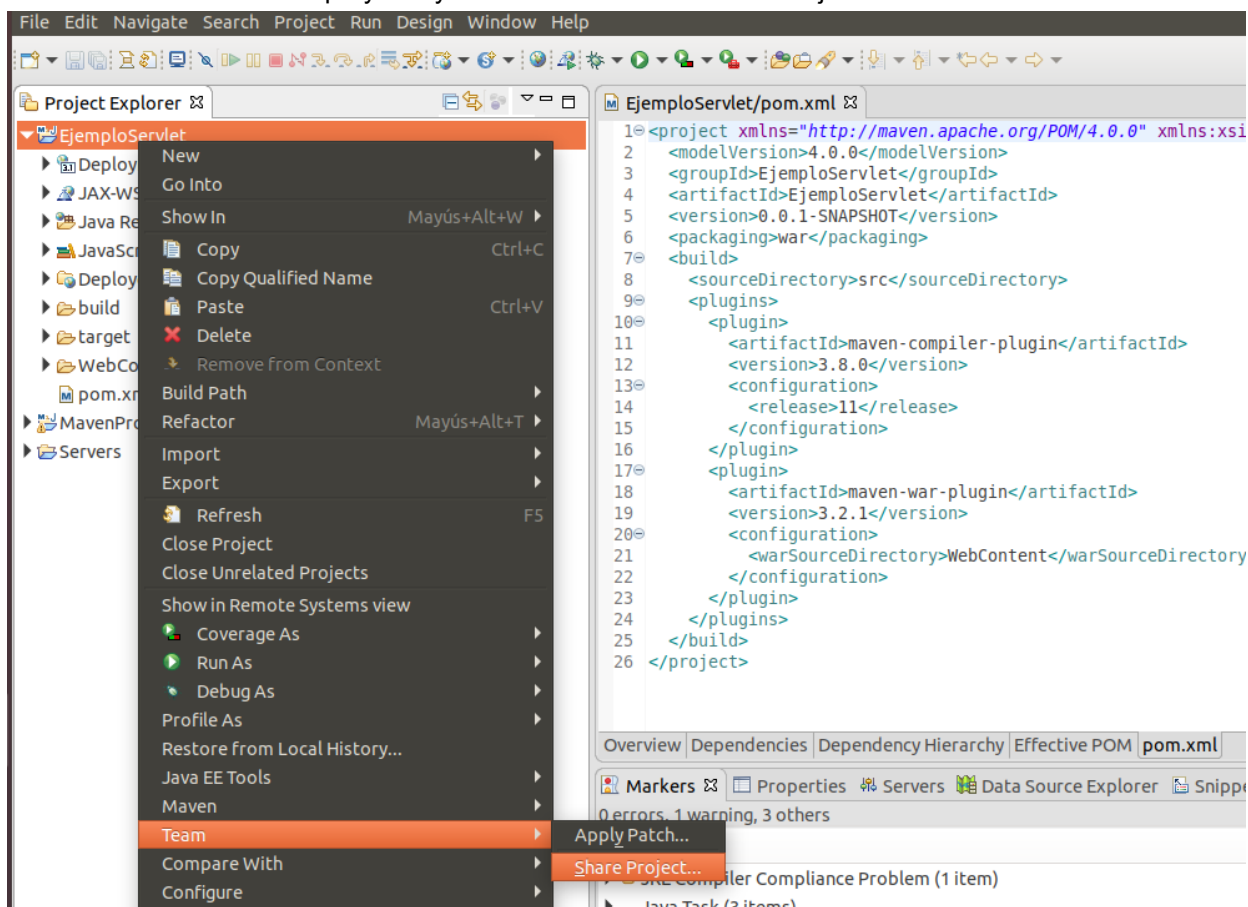
Add .gitignore: **None** ▾

Add a license: **None** ▾

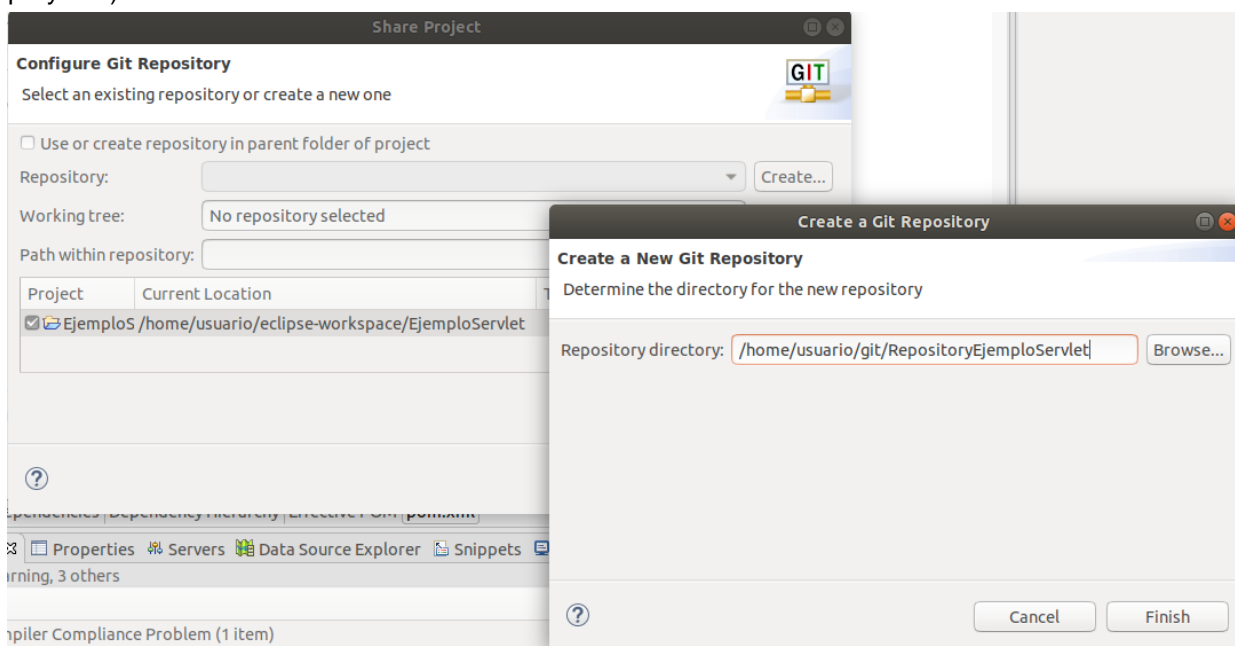


Create repository

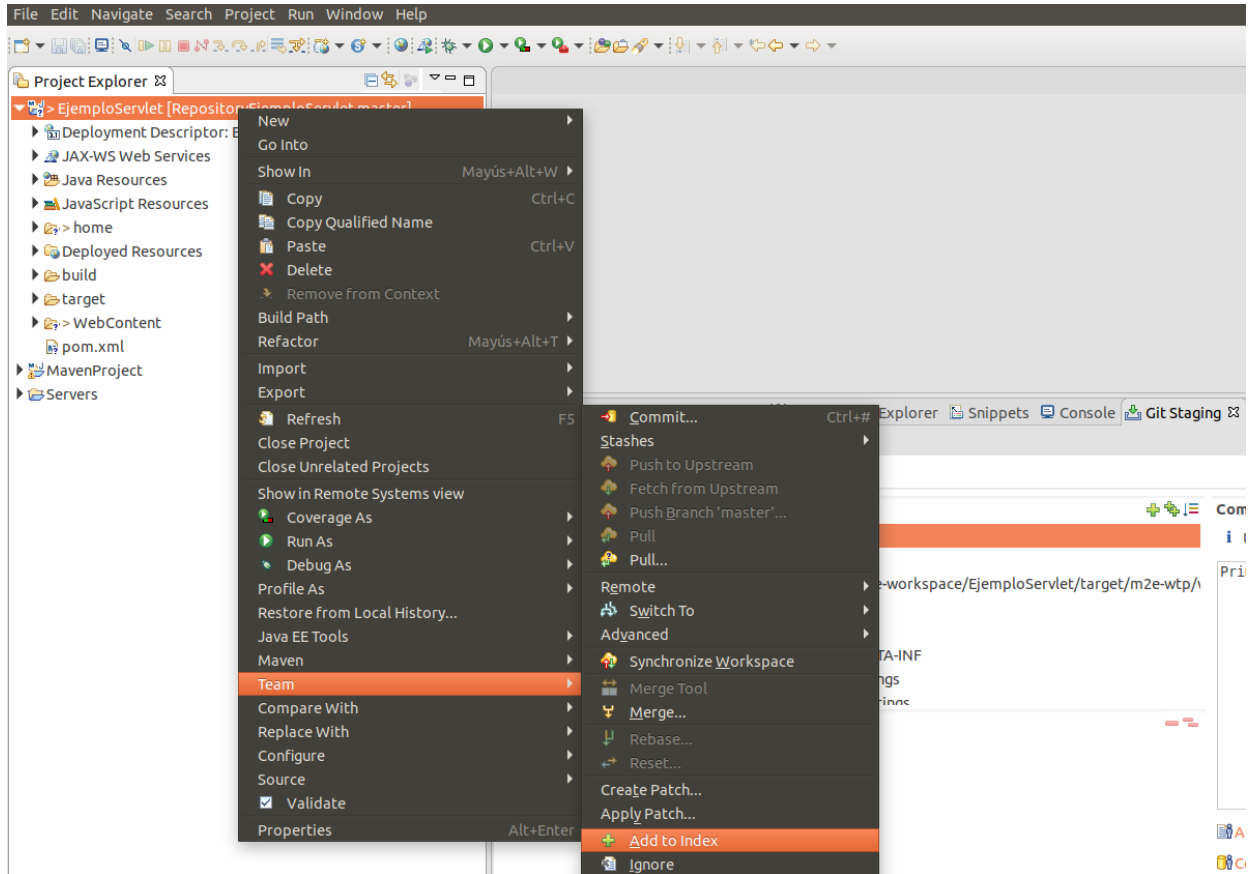
Primero se debe conectar el proyecto con un repositorio local de Git. Para ello, hacer click con el botón derecho del ratón sobre el proyecto y seleccionar "Team->Share Project":



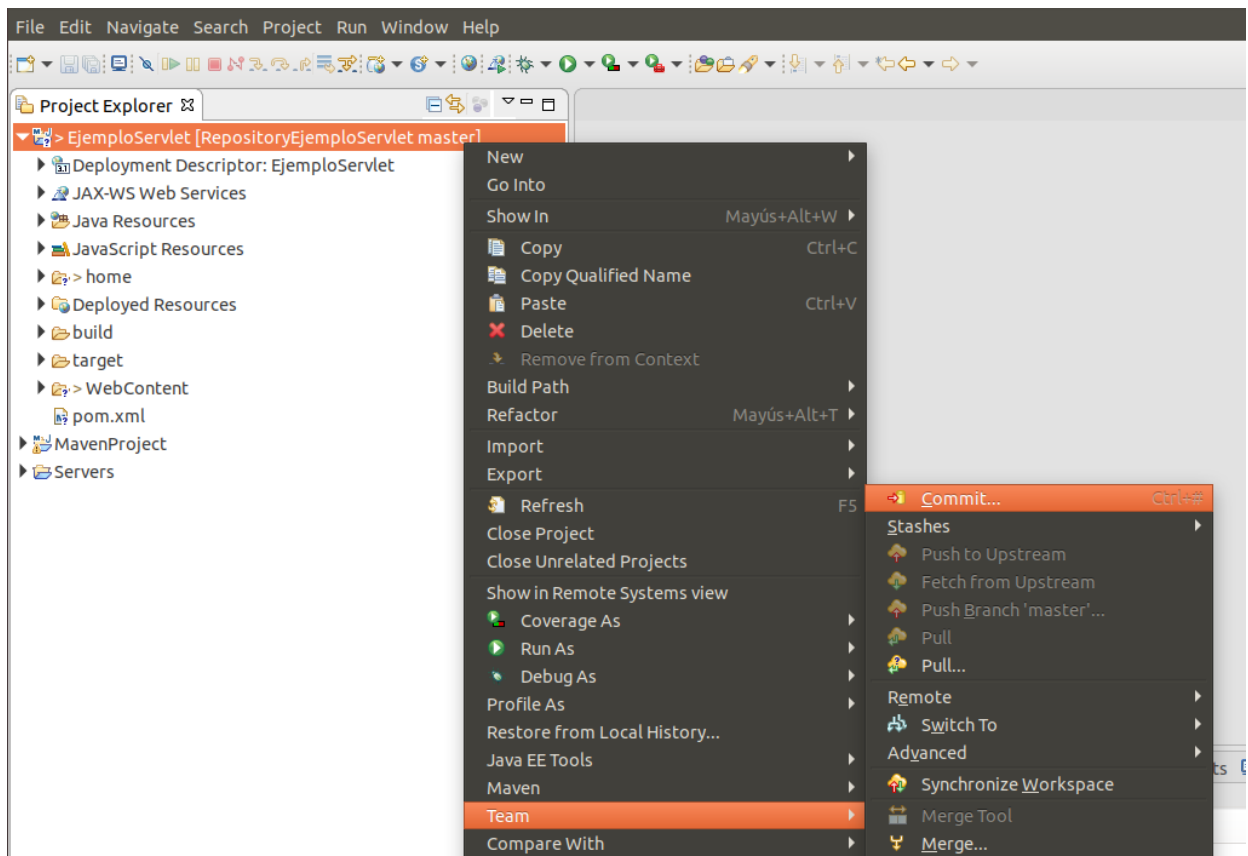
En la nueva ventana que se abre crear un nuevo repositorio Git (preferiblemente fuera del workspace del proyecto):



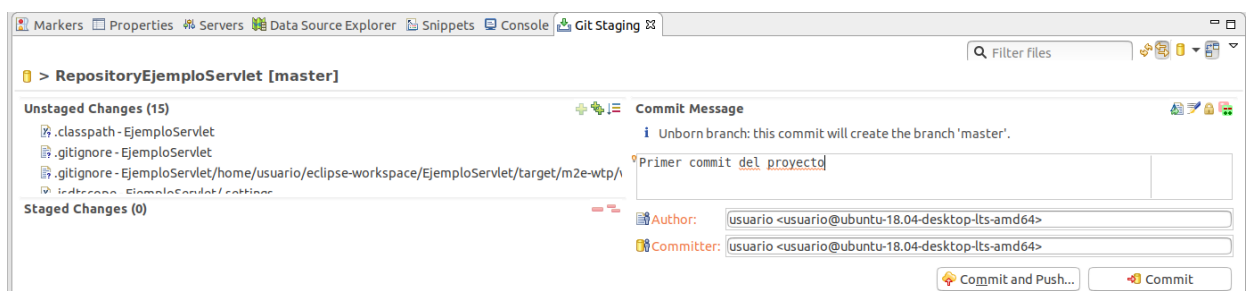
Ahora se deben añadir todos los ficheros del proyecto a Git, para ello hacer click con el botón derecho del ratón sobre el proyecto y seleccionando "Team->Add to Index":



Ahora subir el proyecto, haciendo click con el botón derecho del ratón sobre el proyecto y seleccionando "Team->Commit":

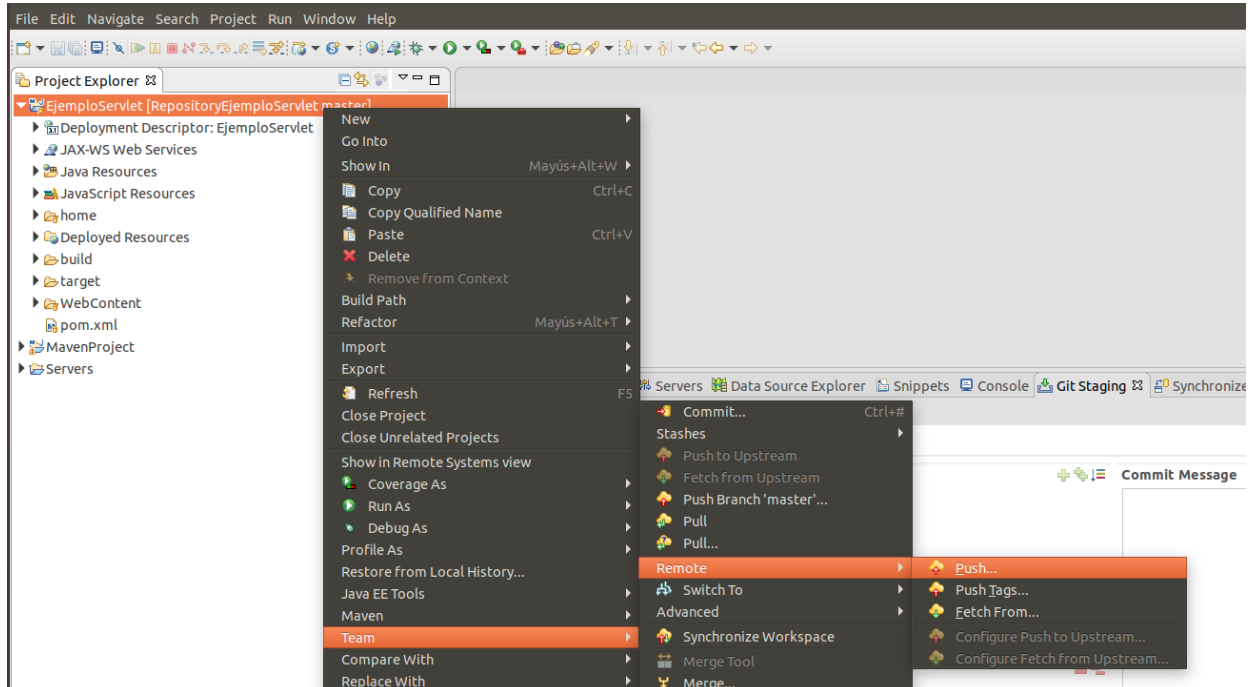


Añadir una descripción al commit y pulsar el botón "Commit" para que se añadan definitivamente todos los ficheros al repositorio local:

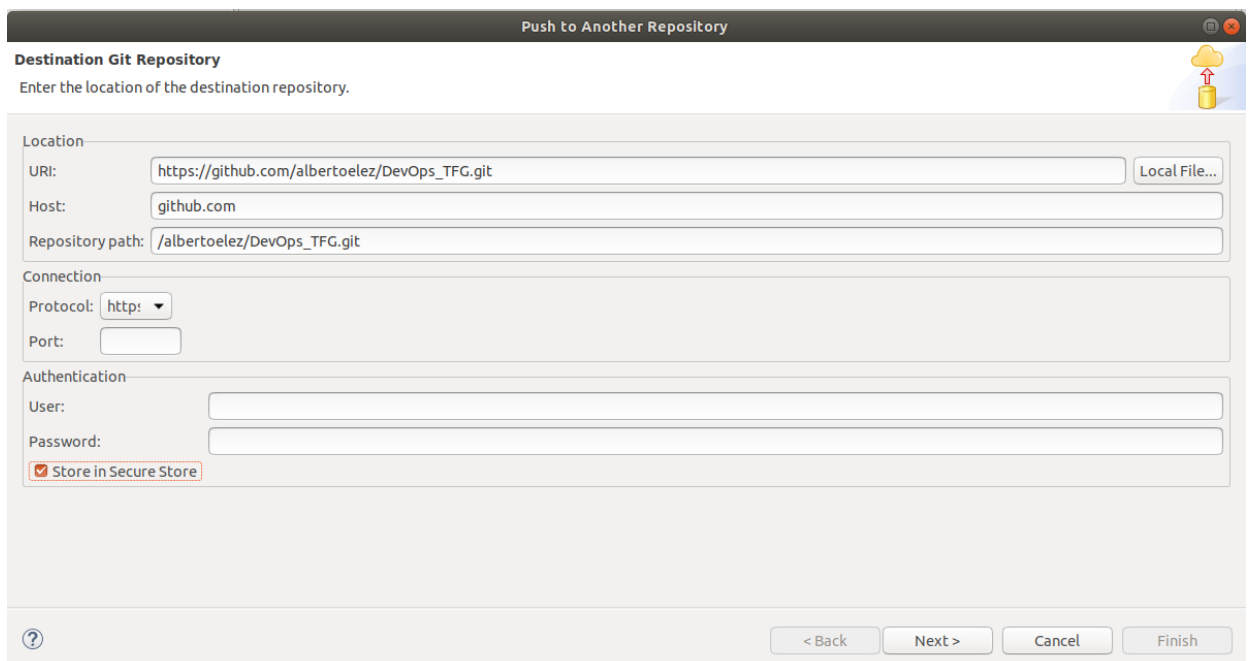




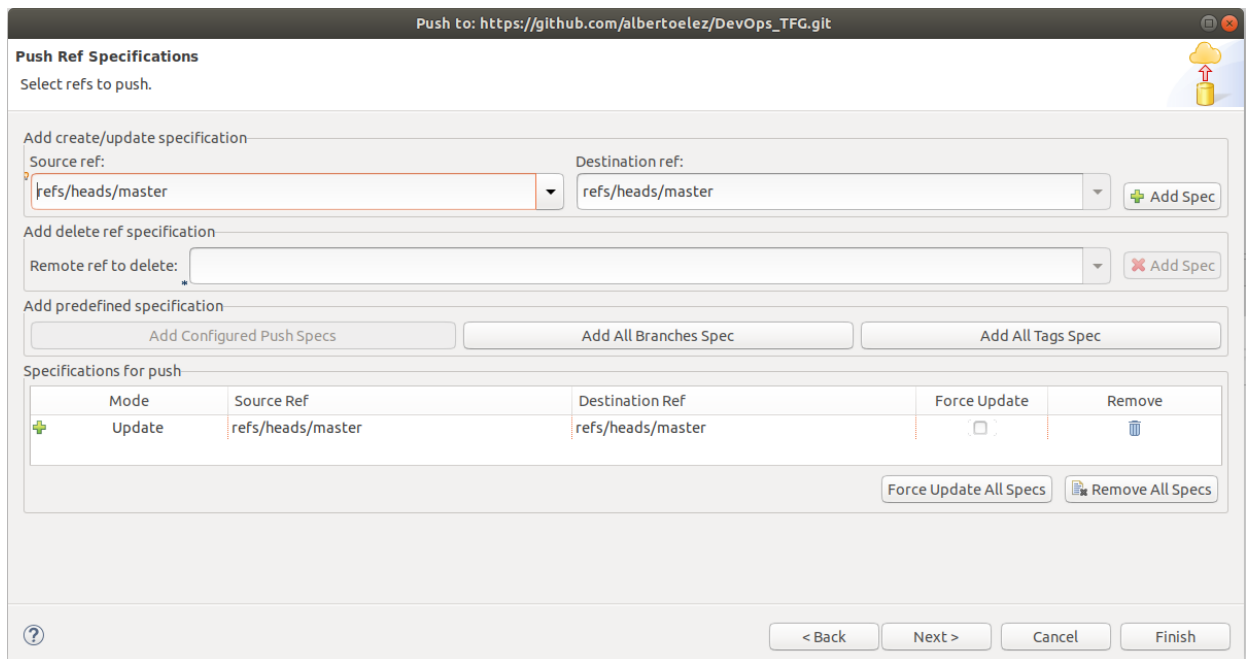
El siguiente paso es subir el repositorio a la cuenta de GitHub, al repositorio remoto, y de esa manera quedará compartido con otros desarrolladores. Para ello pulsar el botón derecho del ratón sobre el proyecto y seleccionar “Team->Remote->Push”:



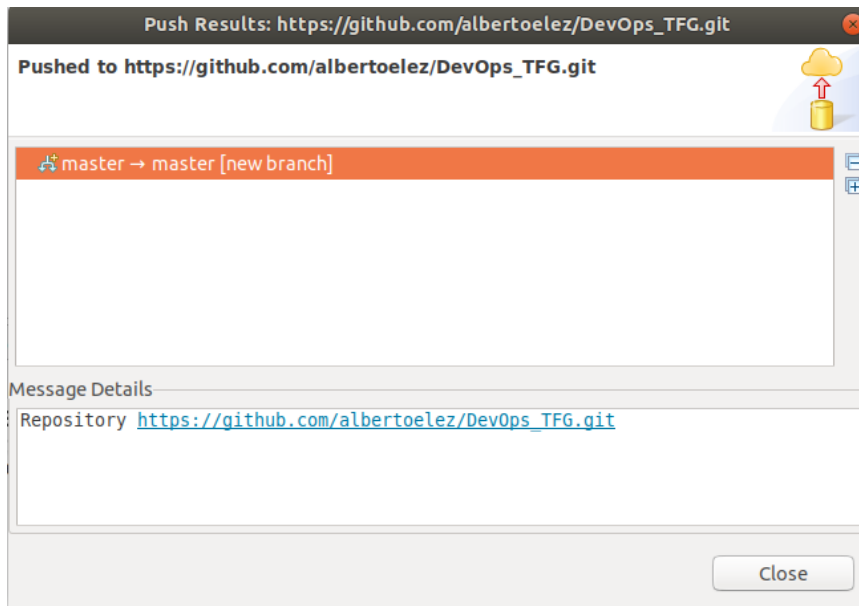
En la siguiente pantalla se deben introducir los datos del repositorio remoto en GitHub, así como el usuario y la password:



En la siguiente pantalla seleccionar las opciones de master y pulsar sobre “Add Spec”:



Y pulsando el botón “Finish”, el proyecto se subirá al repositorio en la nube de GitHub:



Y ya se pueden ver los ficheros por Internet desde GitHub:

The screenshot shows a GitHub repository page for 'albertoalez / DevOps\_TFG'. At the top, there are buttons for 'Watch' (0), 'Star' (0), and 'Fork' (0). Below this, there are navigation tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Insights', and 'Settings'. The main content area shows the repository structure for the 'master' branch. It lists several files and folders, all of which were committed by 'usuario' 17 minutes ago. The files include '.settings', 'WebContent/META-INF', 'home/usuario/eclipse-workspace/EjemploServlet/target/m2e-wt...', 'src', '.classpath', '.gitignore', '.project', and 'pom.xml'.

File/Folder	Commit Message	Time
..	Primer commit del proyecto	17 minutes ago
.settings	Primer commit del proyecto	17 minutes ago
WebContent/META-INF	Primer commit del proyecto	17 minutes ago
home/usuario/eclipse-workspace/EjemploServlet/target/m2e-wt...	Primer commit del proyecto	17 minutes ago
src	Primer commit del proyecto	17 minutes ago
.classpath	Primer commit del proyecto	17 minutes ago
.gitignore	Primer commit del proyecto	17 minutes ago
.project	Primer commit del proyecto	17 minutes ago
pom.xml	Primer commit del proyecto	17 minutes ago

## 5.8 Instalación de SonarQube

SonarQube es una herramienta que se utiliza para revisar el código fuente y analizar la calidad del software en desarrollo. Necesita una base de datos para funcionar, aunque cuenta con una base de datos embebida llamada H2 pero es muy básica y solo se recomienda para pruebas rápidas. Por tanto lo más recomendable es instalar otra base de datos como por ejemplo MySQL, PostgreSQL, Oracle, etc.

Para instalar MySQL en Ubuntu se puede hacer descargando los binarios desde su página oficial <https://dev.mysql.com/downloads/mysql/> o instalando los paquetes desde Ubuntu. Para hacerlo de esta segunda manera, se debe abrir un terminal con privilegios de root y teclear:

```
root@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@ubuntu-18:~# apt install mysql-server
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 libaio1 libevent-core-2.1-6 libhtml-template-perl mysql-client-5.7
 mysql-client-core-5.7 mysql-common mysql-server-5.7 mysql-server-core-5.7
Paquetes sugeridos:
 libipc-sharedcache-perl mailx tinycb
Se instalarán los siguientes paquetes NUEVOS:
 libaio1 libevent-core-2.1-6 libhtml-template-perl mysql-client-5.7
 mysql-client-core-5.7 mysql-common mysql-server mysql-server-5.7
 mysql-server-core-5.7
0 actualizados, 9 nuevos se instalarán, 0 para eliminar y 3 no actualizados.
Se necesita descargar 20,5 MB de archivos.
Se utilizarán 160 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 mysql-common all 5.8
+1.0.4 [7.308 B]
```

A continuación se debe ejecutar el archivo de instalación segura, donde se debe configurar la fortaleza de las passwords, meter una password para el root y otra serie de opciones que se pueden ir contestando como "yes" por defecto:

```
root@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@ubuntu-18:~# mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No: n
Please set the password for root here.

New password:

Re-enter new password:
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.
```

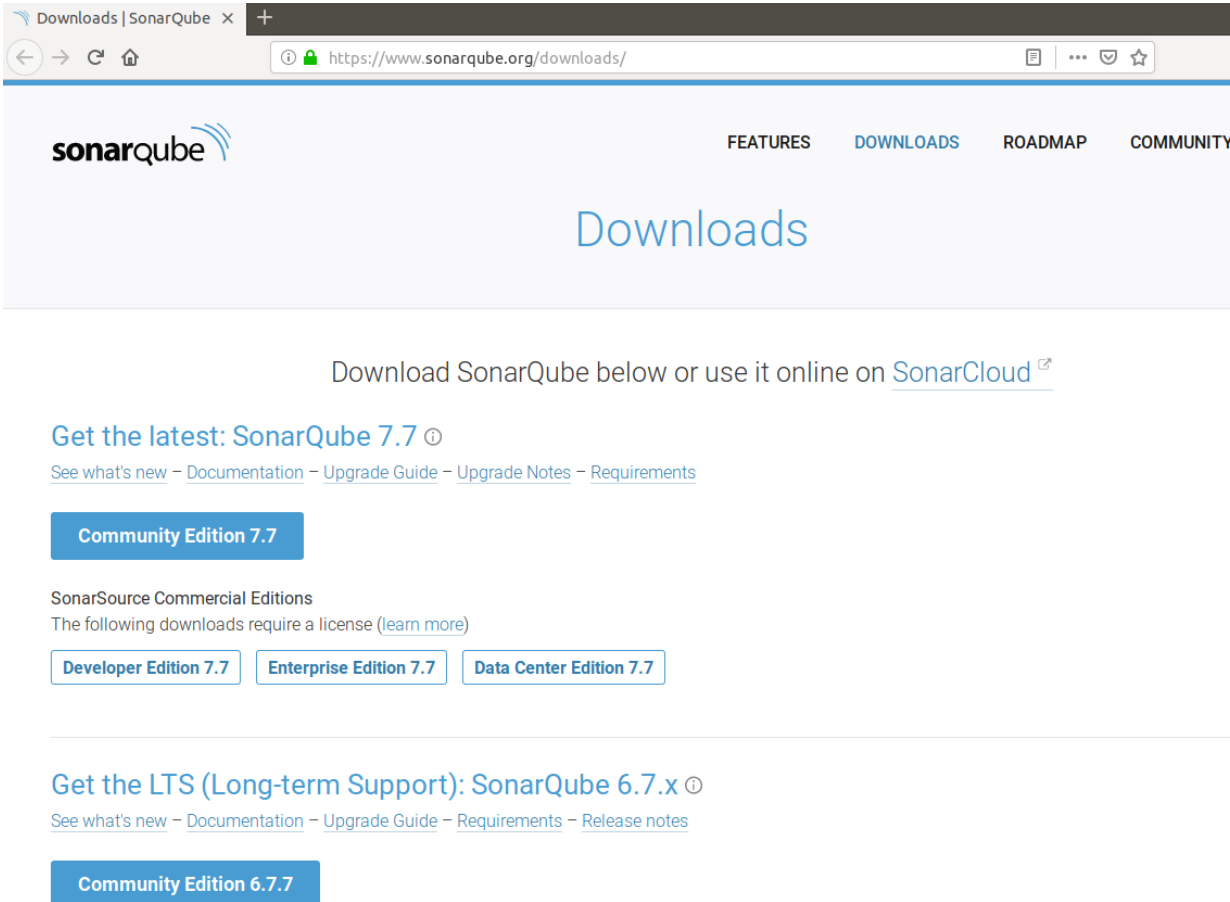
Para poder trabajar con SonarQube es necesario crear un esquema y asignarle un usuario que permita crear, modificar y eliminar datos en dicho esquema. El charset de la base de datos debe ser UTF-8.

Para ello entramos en la base de datos tecleando en el terminal:  
mysql

A continuación se deben teclear las siguientes sentencias SQL para crear el esquema y el usuario:  
CREATE DATABASE sonarqube CHARACTER SET utf8 COLLATE utf8\_general\_ci;  
CREATE USER 'sonarqube' IDENTIFIED BY 'sonarqube';  
GRANT ALL ON sonarqube.\* TO 'sonarqube'@'%' IDENTIFIED BY 'sonarqube';  
GRANT ALL ON sonarqube.\* TO 'sonarqube'@'localhost' IDENTIFIED BY 'sonarqube';  
FLUSH PRIVILEGES;

Para salir de la base de datos, ejecutar en el terminal el comando:  
exit

A continuación ya se puede instalar SonarQube. Para ello se debe descargar la última versión Community Edition disponible en su página web:



The screenshot shows a web browser window with the URL <https://www.sonarqube.org/downloads/>. The page features the SonarQube logo and navigation links for FEATURES, DOWNLOADS, ROADMAP, and COMMUNITY. The main heading is "Downloads". Below this, there is a link to "Download SonarQube below or use it online on [SonarCloud](#)".

**Get the latest: SonarQube 7.7** ⓘ  
[See what's new](#) - [Documentation](#) - [Upgrade Guide](#) - [Upgrade Notes](#) - [Requirements](#)

**Community Edition 7.7**

SonarSource Commercial Editions  
The following downloads require a license ([learn more](#))

[Developer Edition 7.7](#) [Enterprise Edition 7.7](#) [Data Center Edition 7.7](#)

---

**Get the LTS (Long-term Support): SonarQube 6.7.x** ⓘ  
[See what's new](#) - [Documentation](#) - [Upgrade Guide](#) - [Requirements](#) - [Release notes](#)

**Community Edition 6.7.7**

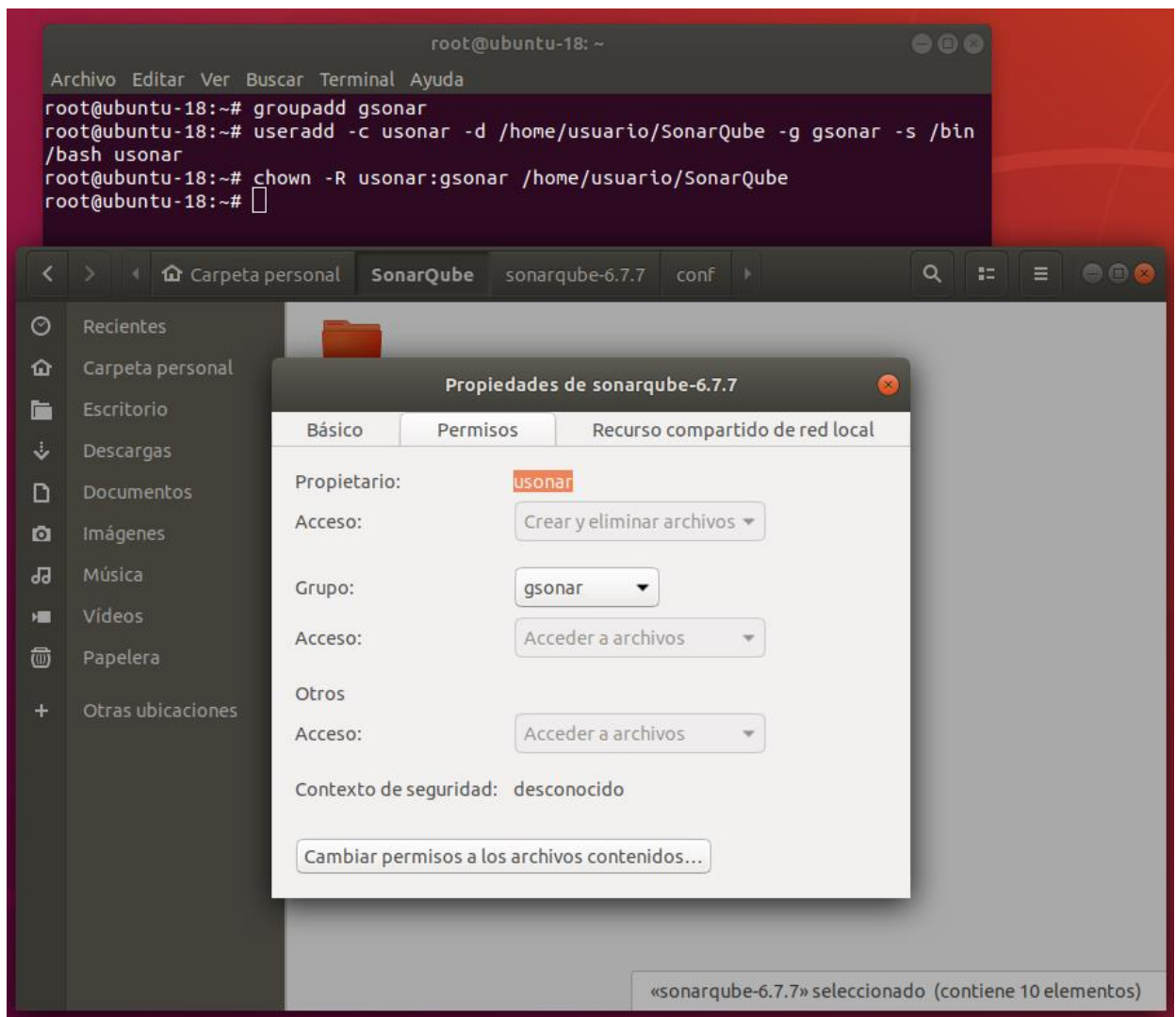
Descargar el fichero, copiarlo y descomprimirlo en la carpeta donde se quiera ejecutar SonarQube.

Desde las últimas versiones de SonarQube, éste no se puede ejecutar como root, por lo que se debe crear un usuario que será el que se utilice para ejecutar el script de SonarQube. Por ejemplo se puede crear un usuario sonar de la siguiente manera:

```
groupadd gsonar
```

```
useradd -c usonar -d home/usuario/SonarQube -g gsonar -s /bin/bash usonar
```

```
chown -R usonar:gsonar home/usuario/SonarQube
```



Después abrir un terminal con el usuario "usonar" y ejecutar el siguiente comando:  
./sonar.sh console

Con el servidor iniciado se podrá acceder a SonarQube en la URL <http://localhost:9000/>

## 5.9 Instalación de Nexus Repository OSS

Nexus Repository OSS es un repositorio de artefactos. Los artefactos son el resultado de la construcción de una aplicación, es decir, los ejecutables (por ejemplo un War o un Jar en Java). Se identifican por 3 propiedades: groupId, artifactId y versión. Esta información por ejemplo es utilizada por Maven para identificar las dependencias de un proyecto, necesarias a la hora de compilar y ejecutar la aplicación.

Un repositorio de artefactos es útil tanto para subir y compartir librerías de terceros (las dependencias de nuestros proyectos) como para publicar los ejecutables generados en el desarrollo y que posteriormente serán desplegados en los entornos de certificación, pre-producción y producción.

Para instalarlo se debe descargar desde su página web, guardar en una carpeta local y descomprimir en la carpeta donde se quiera ejecutar.

No se puede iniciar como usuario root, así que se debe editar el fichero "nexus.rc" y establecer manualmente el usuario que lo ejecutará:

```
run_as_user="usuario"
```

Después ir a la carpeta bin donde esté descomprimido y ejecutar "./nexus start" para arrancar el servidor de Nexus.

## 5.10 Instalación de Jenkins

Antes de instalar Jenkins hay que tener en cuenta que actualmente solo funciona con algunas versiones de Java, por lo tanto hay que revisar su página web (<https://jenkins.io/doc/administration/requirements/java/>) para saber qué versión de Java es compatible.

En este ejemplo se estaba utilizando openjdk11 que no es compatible con Jenkins, por lo tanto se debe instalar adicionalmente openjdk8 que sí que es compatible y seleccionar la versión de Java por defecto utilizando el siguiente comando:

```
usuario@ubuntu-18:~$ sudo update-alternatives --config java
Existen 2 opciones para la alternativa java (que provee /usr/bin/java).

Selección   Ruta
-----
* 0          /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111   modo automático
1           /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111   modo manual
2           /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java    1081   modo manual

Pulse <Intro> para mantener el valor por omisión [*] o pulse un número de selección: 2
update-alternatives: utilizando /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java para proveer /usr/bin/java (java) en modo manual
```

A continuación instalamos Jenkins, tecleando los siguientes comandos en un terminal:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install Jenkins
```

```
usuario@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@ubuntu-18:~$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key ad
d -
OK
usuario@ubuntu-18:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt
/sources.list.d/jenkins.list'
usuario@ubuntu-18:~$ sudo apt update
Obj:1 http://es.archive.ubuntu.com/ubuntu bionic InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Des:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Obj:4 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic InRelease
Des:5 http://es.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Ign:6 http://pkg.jenkins.io/debian-stable binary/ InRelease
Obj:7 http://pkg.jenkins.io/debian-stable binary/ Release
Descargados 252 kB en 1s (202 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se pueden actualizar 3 paquetes. Ejecute «apt list --upgradable» para verlos.
usuario@ubuntu-18:~$ sudo apt install jenkins
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

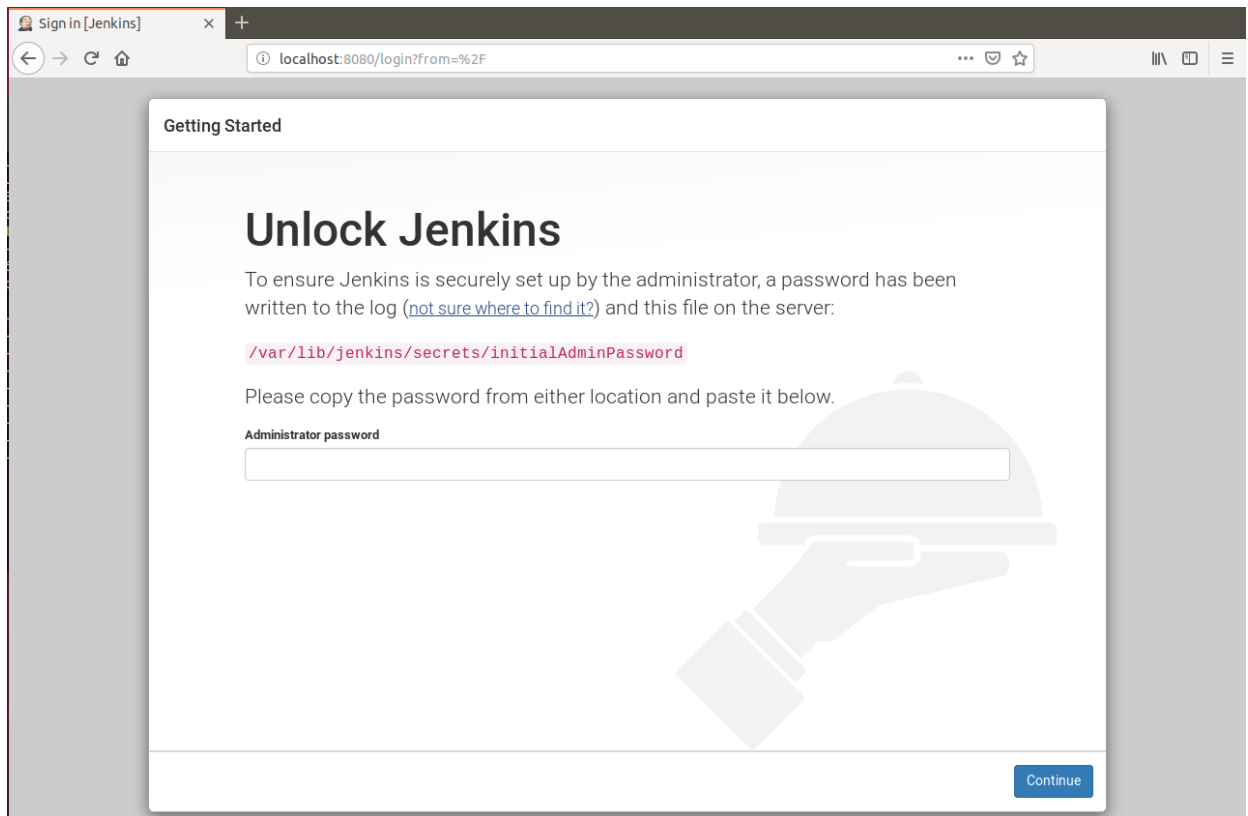
Para iniciar Jenkins utilizar los siguientes comandos para iniciarlo y luego comprobar su estado:

```
usuario@ubuntu-18: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@ubuntu-18:~$ sudo systemctl start jenkins
usuario@ubuntu-18:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; generated)
   Active: active (exited) since Thu 2019-05-09 16:46:28 CEST; 3min 38s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 4683)
   CGroup: /system.slice/jenkins.service

may 09 16:46:27 ubuntu-18 systemd[1]: Starting LSB: Start Jenkins at boot time...
may 09 16:46:27 ubuntu-18 jenkins[4760]: Correct java version found
may 09 16:46:27 ubuntu-18 jenkins[4760]: * Starting Jenkins Automation Server jenkins
may 09 16:46:27 ubuntu-18 su[4793]: Successful su for jenkins by root
may 09 16:46:27 ubuntu-18 su[4793]: + ??? root:jenkins
may 09 16:46:27 ubuntu-18 su[4793]: pam_unix(su:session): session opened for user jenkins by (ui
may 09 16:46:27 ubuntu-18 su[4793]: pam_unix(su:session): session closed for user jenkins
may 09 16:46:28 ubuntu-18 jenkins[4760]: ...done.
may 09 16:46:28 ubuntu-18 systemd[1]: Started LSB: Start Jenkins at boot time.
lines 1-16/16 (END)
```

Ahora ya se podrá acceder a Jenkins en la URL <http://localhost:8080>:

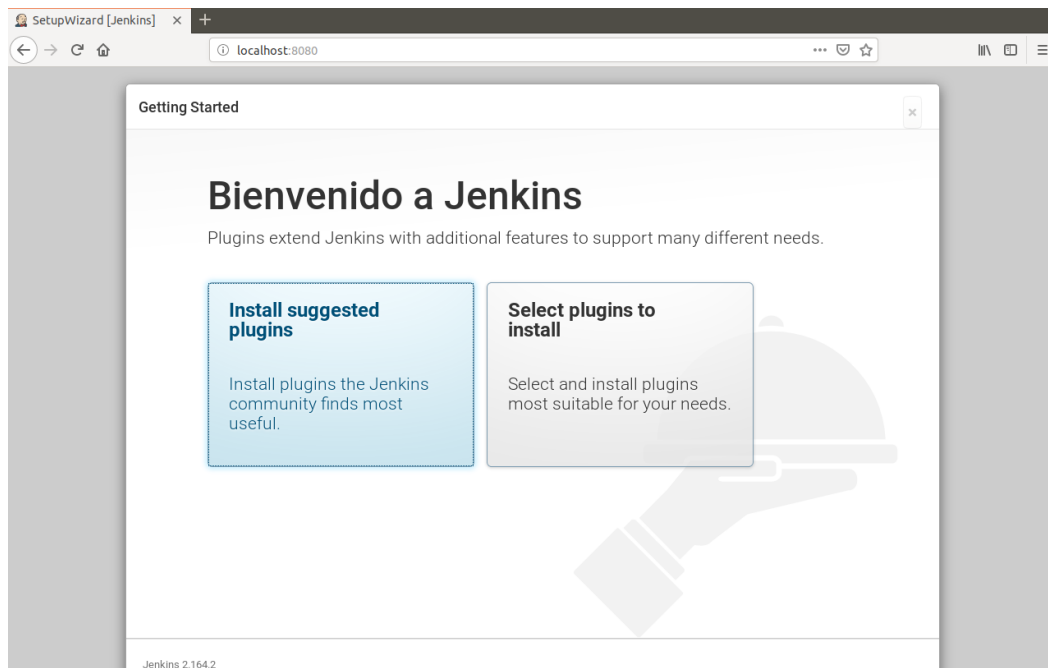




Como la instalación de Jenkins viene securizada, se debe desbloquear metiendo la password guardada en el fichero `/var/lib/jenkins/secrets/initialAdminPassword`. Se puede obtener con el siguiente comando:

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

A continuación se muestra la pantalla de bienvenida de Jenkins:



Seleccionar "Install suggested plugins" y se irán instalando los plugins por defecto. Cuando termine la instalación se mostrará un formulario para la creación de un usuario administrador, rellenar toda la información y pulsar el botón "Save and Continue":

A screenshot of the Jenkins Setup Wizard 'Create First Admin User' form. The form is titled 'Create First Admin User' and contains several input fields. The fields are: 'Usuario:' with the value 'admin'; 'Contraseña:' with masked characters '\*\*\*\*'; 'Confirma la contraseña:' with masked characters '\*\*\*\*'; 'Nombre completo:' with the value 'Administrador'; and 'Dirección de email:' with the value 'admin@localhost'. At the bottom of the form, there are two buttons: 'Continue as admin' and 'Save and Continue'. The version 'Jenkins 2.164.2' is displayed at the bottom left of the page.

Aparece una nueva ventana, cuyos valores por defecto son correctos, pulsar el botón “Save and Finish” y después “Start using Jenkins” y ya estará Jenkins terminado de instalar:

### Getting Started

# Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.164.2 Not now [Save and Finish](#)

A continuación aparecerá su panel de control:

The screenshot shows the Jenkins web interface in a Mozilla Firefox browser. The address bar shows 'localhost:8080'. The page title is 'Panel de control [Jenkins] - Mozilla Firefox'. The Jenkins logo is visible in the top left. The main content area displays a welcome message: '¡Bienvenido a Jenkins!' with a prompt to 'crea una nueva tarea' (create a new task). On the left sidebar, there are navigation links: Nueva Tarea, Personas, Historial de trabajos, Administrar Jenkins, Mis vistas, Credentials, Lockable Resources, and New View. Below the sidebar, there are two sections: 'Trabajos en la cola' (Jobs in queue) showing 'No hay trabajos en la cola' (No jobs in queue), and 'Estado del ejecutor de construcciones' (Build executor status) showing two 'Inactivo' (Inactive) executors.

Es necesario instalar los siguientes plugins para el proceso completo de integración continua:

Maven Integration

Actualizado

SonarQube Scanner

Actualizado

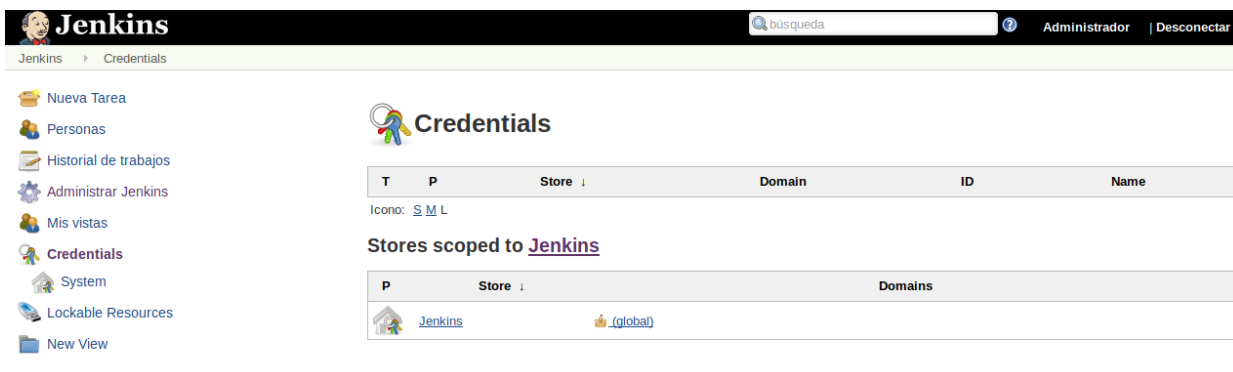
Para ello seleccionar Administrar Jenkins->Administrar Plugins:

The screenshot shows the Jenkins Administration page. On the left is a navigation sidebar with options like 'Nueva Tarea', 'Personas', 'Historial de trabajos', 'Administrar Jenkins', 'Mis vistas', 'Credentials', 'Lockable Resources', and 'New View'. Below the sidebar are sections for 'Trabajos en la cola' (empty) and 'Estado del ejecutor de construcciones' (two inactive executors). The main content area is titled 'Administrar Jenkins' and contains several configuration options: 'Configurar el Sistema' (Configure global variables and paths), 'Configuración global de la seguridad' (Global security configuration), 'Configure Credentials', 'Global Tool Configuration', 'Actualizar configuración desde el disco duro' (Update configuration from disk), 'Administrar Plugins' (highlighted with a red box), and 'Información del sistema' (System information).

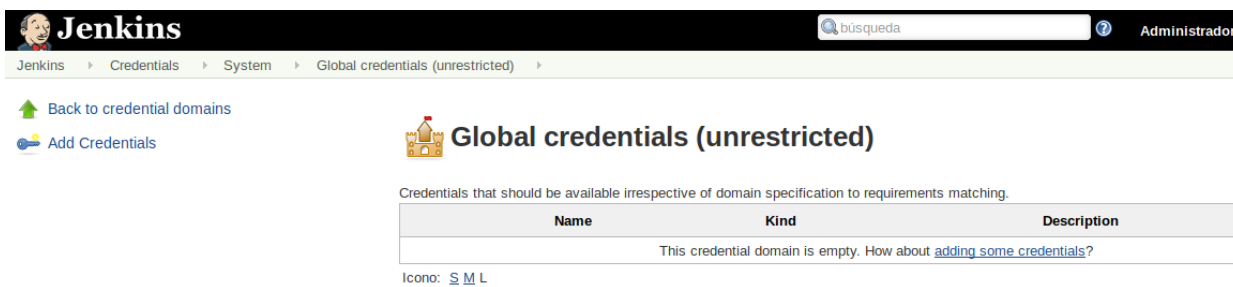
Después hay que configurar Jenkins para que localice las rutas de instalación de los distintos componentes que forman parte del entorno de integración continua. Seleccionar “Administrar Jenkins->Configurar el sistema”:

This screenshot shows the same Jenkins Administration page, but with the 'Configurar el Sistema' option highlighted by a red box. The browser's address bar shows 'localhost:8080/manage'.

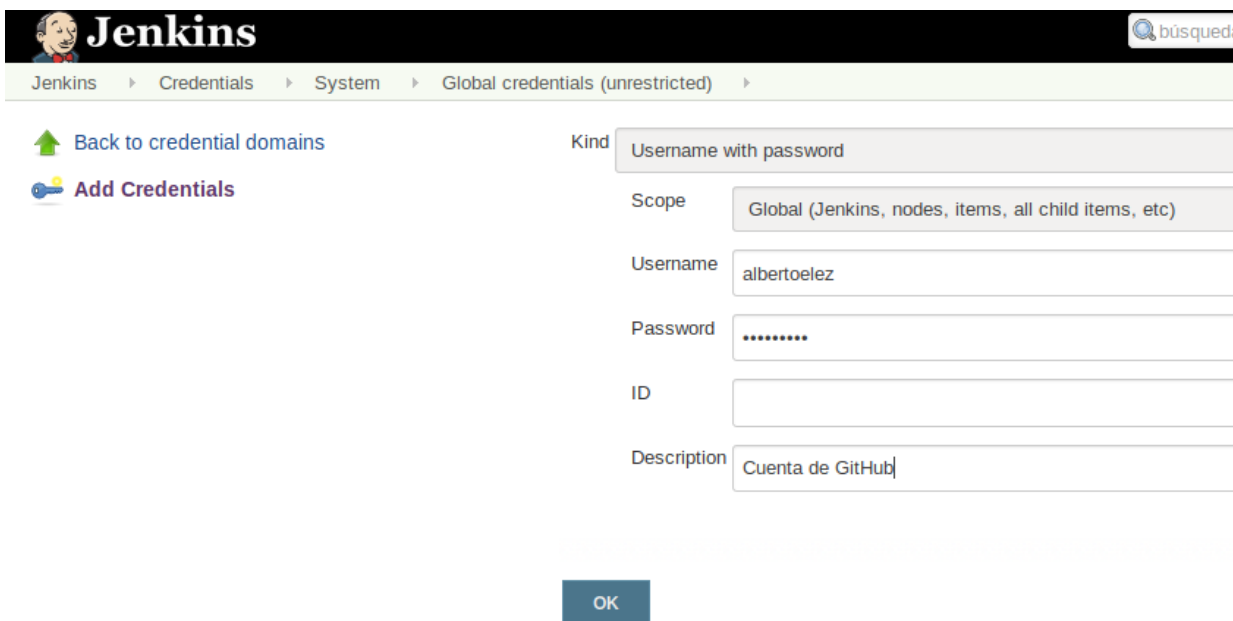
Para poder acceder a los proyectos guardados en GitHub es necesario configurar las credenciales de acceso. Para ello, en el menú principal de Jenkins pinchar sobre la opción “Credentials”:



En la tabla que aparece seleccionar el dominio existente “global”, hacer click sobre él y en la ventana que aparece pulsar sobre “adding some credentials”:



Rellenar con el usuario y la password del repositorio de GitHub:

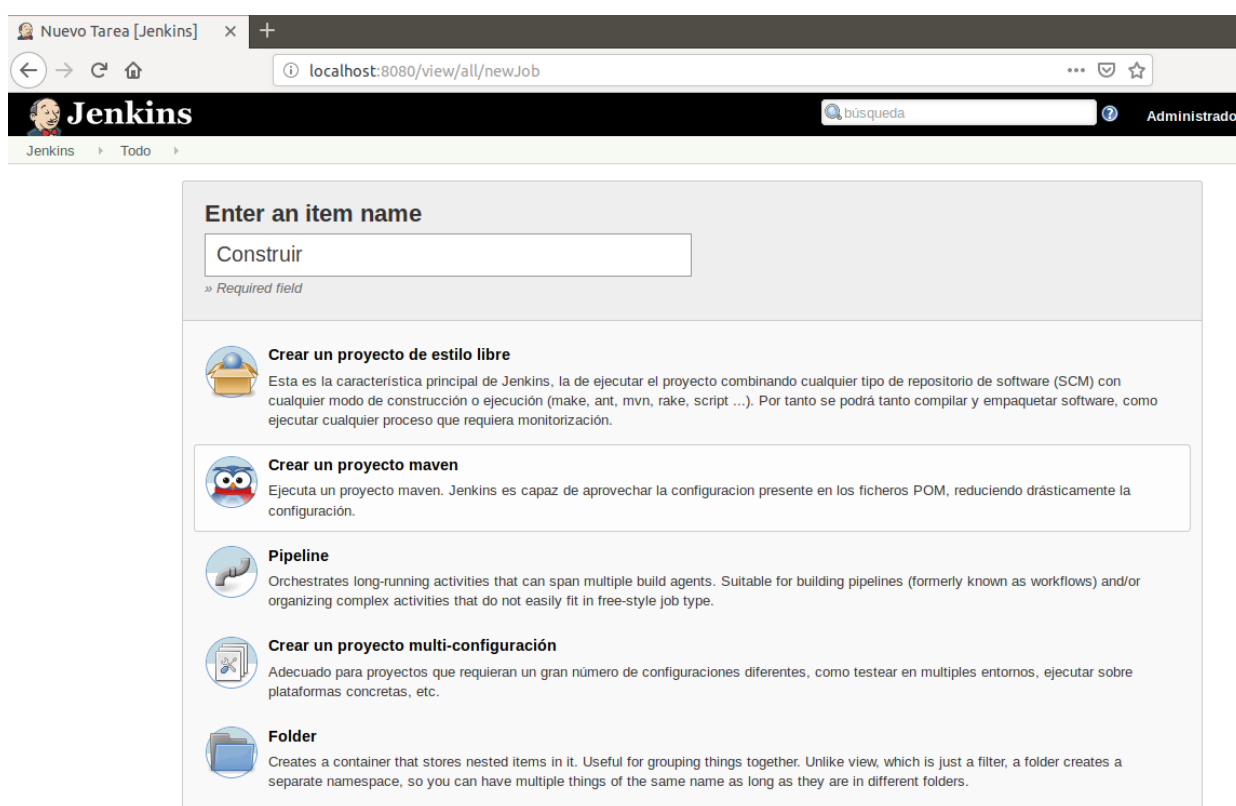


## 5.11 Generación de un pipeline con Jenkins

Se va a desarrollar un pipeline de Jenkins que contemple las fases principales de un proyecto de software. Los procesos que va a contemplar son los siguientes:

1. Compilación automática del código fuente cada vez que se detecte un cambio en el repositorio Git.
2. Chequeo del código fuente y ejecución de pruebas automáticas.
3. Si las pruebas son correctas publicación de los artefactos en un repositorio.

Para los dos primeros procesos se puede crear una tarea (job) dentro del pipeline de Jenkins. Para ello pulsar sobre "Nueva tarea" y se abrirá una ventana donde se debe dar un nombre a la tarea y elegir un tipo de tarea que en este caso será "Crear un proyecto Maven":



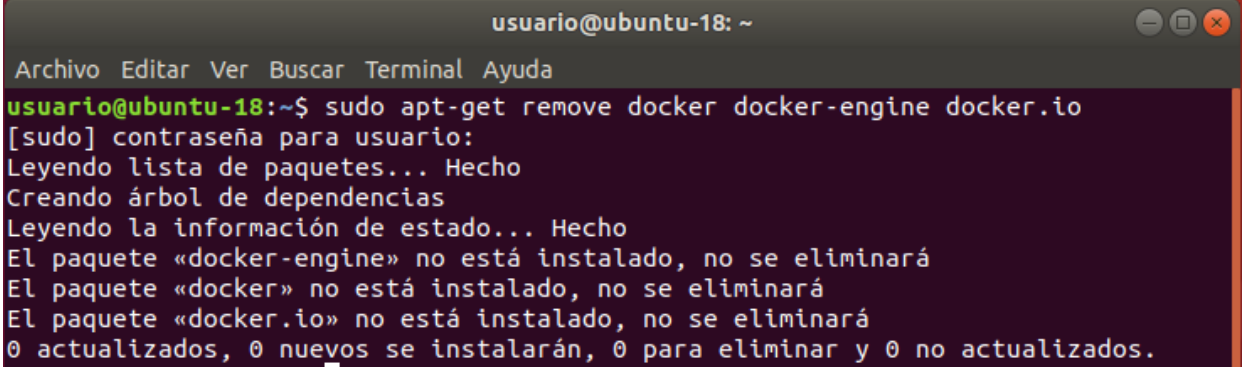
## 5.12 Instalación de Docker

Como ya se comentó en el apartado 4.8, Docker es una herramienta que realiza virtualización a nivel de sistema operativo, en unidades aisladas llamadas contenedores. Esto permite poder probar cualquier software sin tener que instalar nada en el sistema operativo, ya que todo lo que necesite para correr las aplicaciones estará dentro del propio contenedor. Se pueden tener corriendo varios contenedores simultáneamente, pararlos, borrarlos y volverlos a lanzar de una manera muy rápida y consumiendo pocos recursos hardware de la máquina anfitriona.

La instalación de Docker en la máquina anfitriona es la siguiente. Primero se deben eliminar todas las instancias anteriores de Docker si las hubiera, sino omitir este paso, para ello teclear en una ventana de terminal el siguiente comando:

```
sudo apt-get remove docker docker-engine docker.io
```

Si no estaba instalado en la máquina dará un mensaje como este:

A terminal window titled 'usuario@ubuntu-18: ~' showing the execution of the command 'sudo apt-get remove docker docker-engine docker.io'. The output indicates that the packages 'docker-engine', 'docker', and 'docker.io' are not installed and therefore will not be removed. The terminal shows the following text:

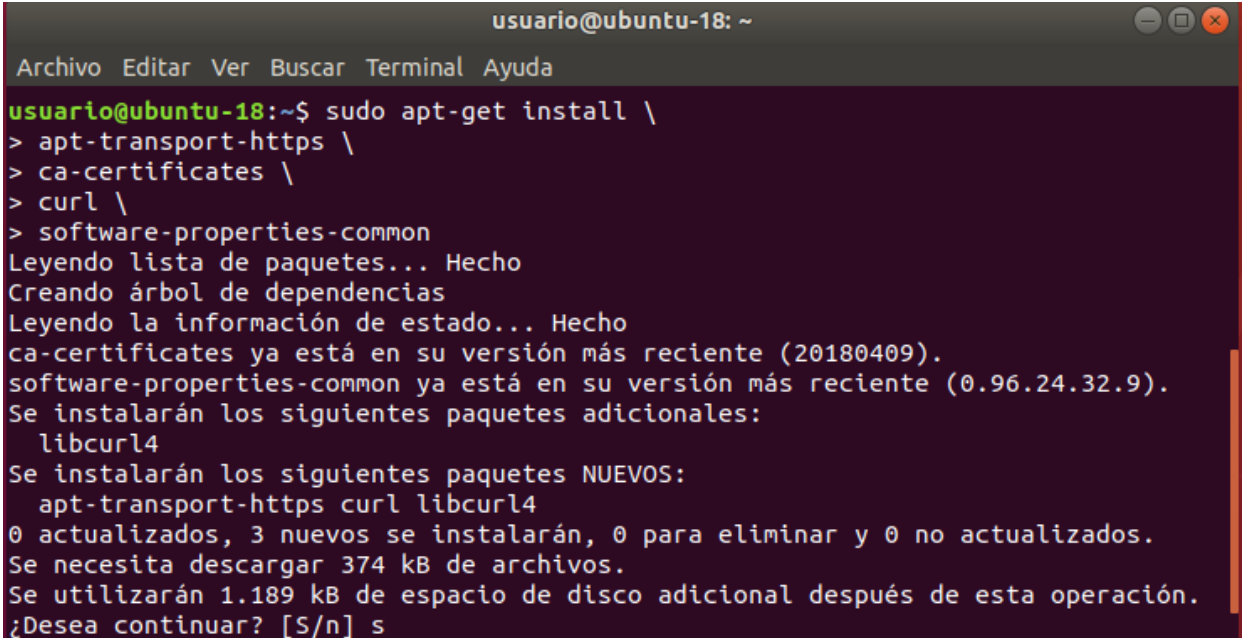
```
usuario@ubuntu-18:~$ sudo apt-get remove docker docker-engine docker.io
[sudo] contraseña para usuario:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
El paquete «docker-engine» no está instalado, no se eliminará
El paquete «docker» no está instalado, no se eliminará
El paquete «docker.io» no está instalado, no se eliminará
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
```

A continuación se deben actualizar los repositorios y los paquetes con los siguientes comandos:

```
sudo apt-get update
sudo apt-get upgrade
```

A continuación se deben instalar algunas dependencias necesarias para Docker con estos comandos:

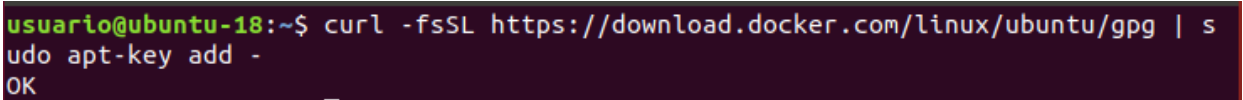
```
sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
software-properties-common
```

A terminal window titled 'usuario@ubuntu-18: ~' showing the execution of the command 'sudo apt-get install apt-transport-https ca-certificates curl software-properties-common'. The output shows that 'ca-certificates' and 'software-properties-common' are already up-to-date, while 'apt-transport-https', 'curl', and 'libcurl4' are new packages that will be installed. The terminal shows the following text:

```
usuario@ubuntu-18:~$ sudo apt-get install \
> apt-transport-https \
> ca-certificates \
> curl \
> software-properties-common
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20180409).
software-properties-common ya está en su versión más reciente (0.96.24.32.9).
Se instalarán los siguientes paquetes adicionales:
 libcurl4
Se instalarán los siguientes paquetes NUEVOS:
 apt-transport-https curl libcurl4
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 374 kB de archivos.
Se utilizarán 1.189 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
```

Ahora se debe importar la clave GPG oficial con el siguiente comando:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

A terminal window showing the execution of the command 'curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -'. The output shows that the command was successful and the key was added. The terminal shows the following text:

```
usuario@ubuntu-18:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | s
udo apt-key add -
OK
```

Se debe verificar que la huella digital sea 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, buscando los últimos 8 caracteres de la huella digital con el siguiente comando:

```
sudo apt-key fingerprint 0EBFCD88
```

```
usuario@ubuntu-18:~$ sudo apt-key fingerprint 0EBFCD88
pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid   [desconocida] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]
```

Se debe añadir manualmente el repositorio al sistema con el siguiente comando:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

```
usuario@ubuntu-18:~$ sudo add-apt-repository "deb [arch=amd64] https://download.
docker.com/linux/ubuntu $(lsb_release -cs) stable"
Obj:1 http://es.archive.ubuntu.com/ubuntu bionic InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu bionic-updates InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu bionic-backports InRelease
Obj:4 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu bionic InRelease
Des:5 https://download.docker.com/linux/ubuntu bionic InRelease [64,4 kB]
Obj:6 http://security.ubuntu.com/ubuntu bionic-security InRelease
Ign:7 http://pkg.jenkins.io/debian-stable binary/ InRelease
Obj:8 http://pkg.jenkins.io/debian-stable binary/ Release
Des:9 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [6.4
26 B]
Descargados 70,9 kB en 1s (49,0 kB/s)
Leyendo lista de paquetes... Hecho
```

Se debe actualizar la lista de repositorios mediante el comando:

```
sudo apt-get update
```

Ahora ya se puede proceder a instalar Docker en el sistema con el siguiente comando:

```
sudo apt-get install docker-ce
```

```
usuario@ubuntu-18:~$ sudo apt-get install docker-ce
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  aufs-tools cgroupfs-mount containerd.io docker-ce-cli git git-man
  liberror-perl pigz
Paquetes sugeridos:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
Se instalarán los siguientes paquetes NUEVOS:
  aufs-tools cgroupfs-mount containerd.io docker-ce docker-ce-cli git git-man
  liberror-perl pigz
0 actualizados, 9 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 55,4 MB de archivos.
Se utilizarán 277 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
```

Tras instalar Docker es recomendable reiniciar el equipo (o máquina virtual), ya que los servicios de Docker inician de manera automática al arrancar el sistema.



Es recomendable añadir el grupo de Docker a nuestro usuario ya que este se crea en el sistema, pero no se añade automáticamente, para ello sobre la terminal ejecutar el siguiente comando cambiando \$USER por el usuario concreto y acto seguido reiniciar el sistema para que se ejecute el cambio:

```
sudo usermod -aG docker $USER
```

Para verificar que se ha instalado correctamente y que se está ejecutando en el sistema se puede realizar una sencilla prueba, ejecutando un contenedor sencillo. Para ello ejecutar el siguiente comando:

```
sudo docker run hello-world
```

```
usuario@ubuntu-18:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:0e11c388b664df8a27a901dce21eb89f11d8292f7fca1b3e3c4321bf7897bffe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

### 5.13 Instalación de un servidor web NGINX con Docker

Nginx (leído como Engine X) es un servidor web ligero de alto rendimiento, opensource y multiplataforma. También es un excelente proxy inverso para contenido web o para protocolos de correo electrónico como por ejemplo IMAP o POP3.

La principal ventaja de Nginx como servidor web es que consume muchos menos recursos al servir contenido estático, y esto le convierte en una excelente opción para funcionar como proxy inverso o como balanceador de carga para otros servidores como Apache, optimizando la entrega de contenidos. Permite responder a millones de peticiones por segundo aprovechando al máximo los núcleos o hilos de ejecución del servidor con una configuración muy simple. La mejora de rendimiento con respecto a otros servidores web como Apache es notable, consiguiendo un tiempo de respuesta de casi un 150% más rápido que en el caso de Apache. También hace un uso más eficiente del consumo de memoria RAM del servidor.

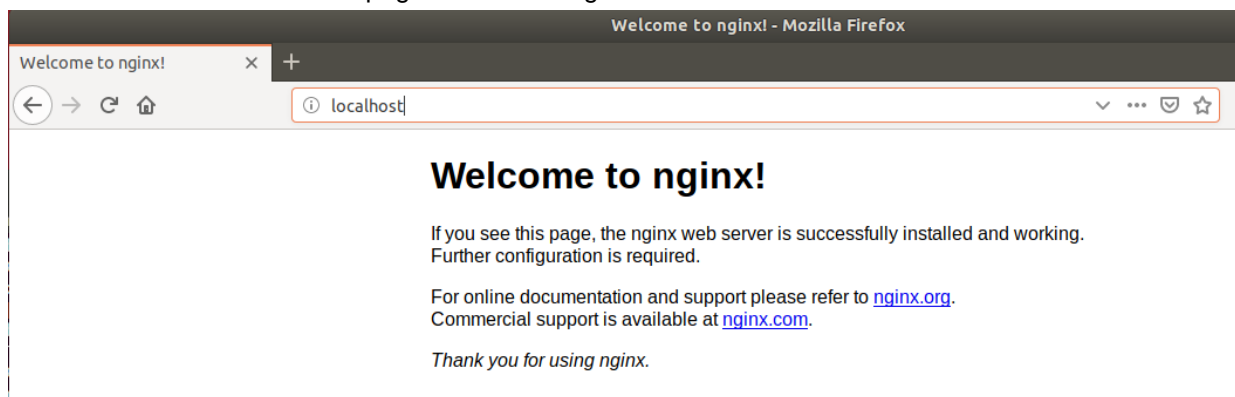
A continuación se va a detallar el proceso de ejecución de un servidor web Nginx desde Docker. Lo primero que hay que hacer es descargar la última imagen Docker de Nginx con el siguiente comando (no es necesario anteponer el sudo si ya hemos añadido el usuario al grupo de docker):

```
sudo docker pull nginx
```

```
usuario@ubuntu-18:~$ sudo docker pull nginx
[sudo] contraseña para usuario:
Using default tag: latest
latest: Pulling from library/nginx
743f2d6c1f65: Pull complete
6bfc4ec4420a: Pull complete
688a776db95f: Pull complete
Digest: sha256:23b4dcdf0d34d4a129755fc6f52e1c6e23bb34ea011b315d87e193033bcd1b68
Status: Downloaded newer image for nginx:latest
```

A continuación ya podemos ejecutar el contenedor con Nginx mediante el siguiente comando:  
sudo docker run --name docker-nginx -p 80:80 nginx

A continuación abrimos una ventana del navegador web (Firefox, Chrome, etc.) y ponemos la dirección Localhost:80 esto mostrará la página inicial de Nginx:



Normalmente el contenido de los contenedores Docker no se suele modificar, porque si se modifica su contenido y después se borra ese contenedor se perderían esas modificaciones. Por lo tanto si se deben ejecutar pruebas sobre ficheros generados por el desarrollador, lo que se suele hacer es linkar un directorio interno del contenedor con un directorio local del sistema de ficheros del ordenador anfitrión.

Para mostrar esa funcionalidad se va a crear una página web de inicio, que se va a alojar en un directorio local del usuario y se va a linkar con la carpeta del contenedor de Nginx donde se guarda la página de inicio. Para ello primero se debe abrir un editor de texto en el directorio local ~/docker-nginx/html y crear el fichero index.html con el siguiente contenido de ejemplo:

```
<html>
  <head>
    <title>Running NGINX inside Docker</title>
  </head>
  <body>
    <div class="container">
      <h1>Hello World!</h1>
      <p>This is a sample web page served by Nginx</p>
    </div>
  </body>
</html>
```

A continuación se debe linkar el directorio de la máquina local con el directorio deseado de dentro del contenedor, para ello se utiliza el siguiente comando:

```
sudo docker run --name docker-nginx -p 80:80 -d -v ~/docker-nginx/html:/usr/share/nginx/html nginx
```

Al abrir un navegador y meter la dirección local se mostrará la página recién creada y a partir de ahora, todas las páginas que se creen en el directorio local se actualizarán automáticamente también en el directorio interno del contenedor (por haberse quedado linkadas):



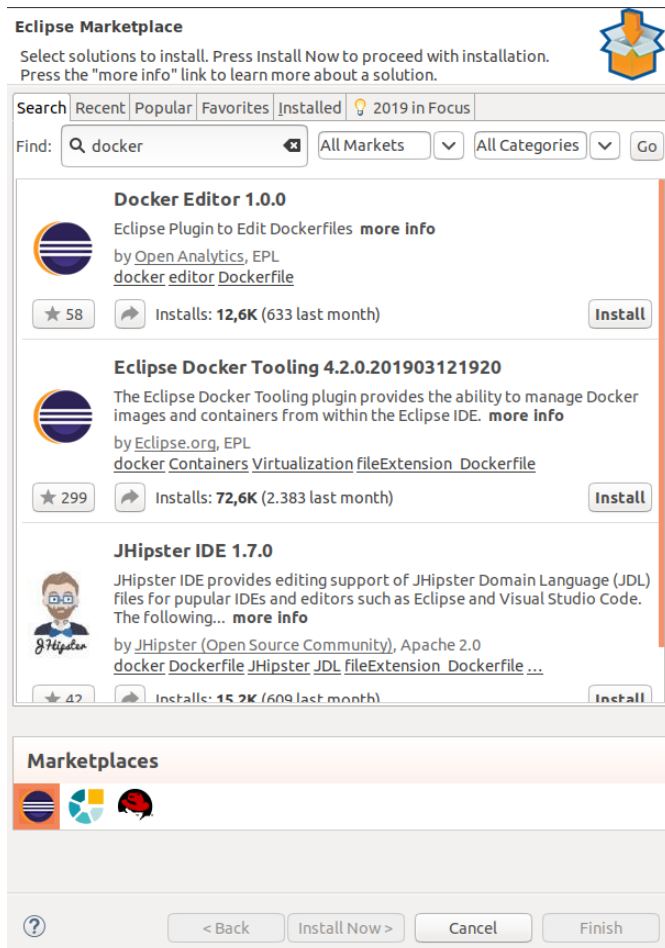
## Hello World!

This is a sample web page served by Nginx

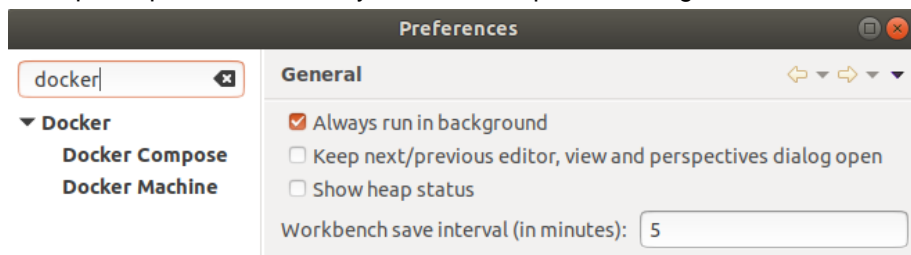
### 5.14 Integración de Docker con Eclipse

Como se comentó anteriormente, el ejemplo de aplicación práctica de este TFG va orientado a la preparación del entorno local de los desarrolladores, por lo tanto se va a explicar cómo se puede tener integrado Docker dentro del entorno de desarrollo Eclipse.

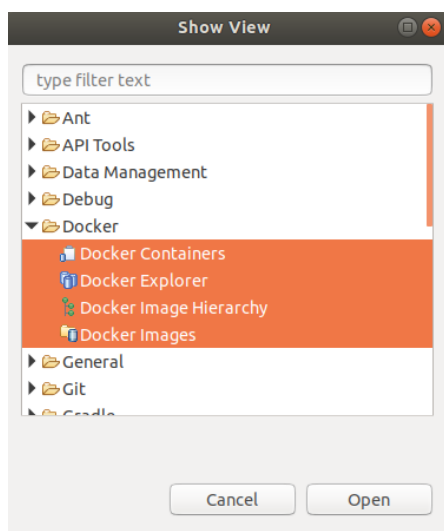
Para verificar si está instalado Docker dentro de Eclipse, ejecutar Eclipse y acceder al menú "Window->Preferences", filtrar por la palabra "Docker", si no aparece nada es que es necesario instalarlo. Para ello hay que seleccionar la entrada de menú "Help->Eclipse Marketplace" y en la ventana que se abre, en el campo "Find" teclear "docker" y nos aparecerán una serie de complementos, donde debemos seleccionar e instalar "Eclipse Docker Tooling xxxxx" pulsando sobre su botón "install":



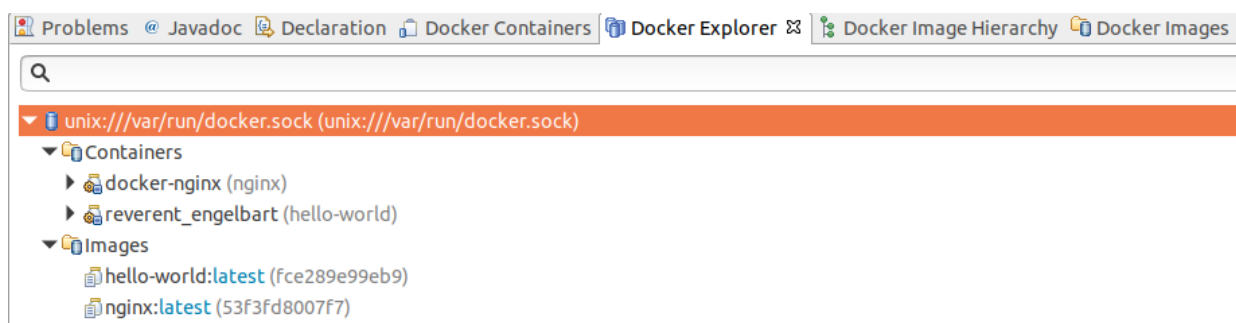
Para verificar que se ha instalado correctamente volvemos a acceder al menú “Window->Preferences”, filtrar por la palabra “Docker” y ahora debe aparecer lo siguiente:



A continuación se deben habilitar todas las pestañas de Docker en Eclipse, para ello acceder al menú “Window->Show View->Others”, seleccionar todas las de Docker y pulsar el botón “Open”:



De todas las pestañas que aparecen la más útil es “Docker Explorer”, ya que en ella aparecen todas las imágenes y contenedores instalados en la máquina:



Es importante entender ambos conceptos:

- **Imagen:** es una plantilla, esto es una captura del estado de un contenedor, con una serie de instrucciones para construir contenedores a partir de ella. Hay muchas imágenes públicas disponibles en Internet con las herramientas y sistemas operativos más utilizados.
- **Repositorio:** son sitios web públicos, donde se almacenan las imágenes con sus diferentes versiones. Un ejemplo es el Hub de Docker (<https://hub.docker.com/>). Las imágenes se identifican por un ID y un par nombre-versión.

- **Contenedor:** es una instancia en ejecución de una imagen, esto es, la ejecución de las aplicaciones. A partir de una única imagen se pueden ejecutar varios contenedores, de manera que se pueden tener varias copias de una aplicación ejecutándose simultáneamente y accediendo a ellos a través de balanceadores de carga, etc. Los cambios ejecutados dentro de un contenedor se pierden al borrar ese contenedor, por lo tanto, no se suelen modificar, pero si alguna vez es necesario, se puede hacer un commit de ese contenedor para generar una imagen nueva que contenga los cambios. De esta manera se pueden versionar los contenedores y siempre se puede volver a una versión anterior del mismo.

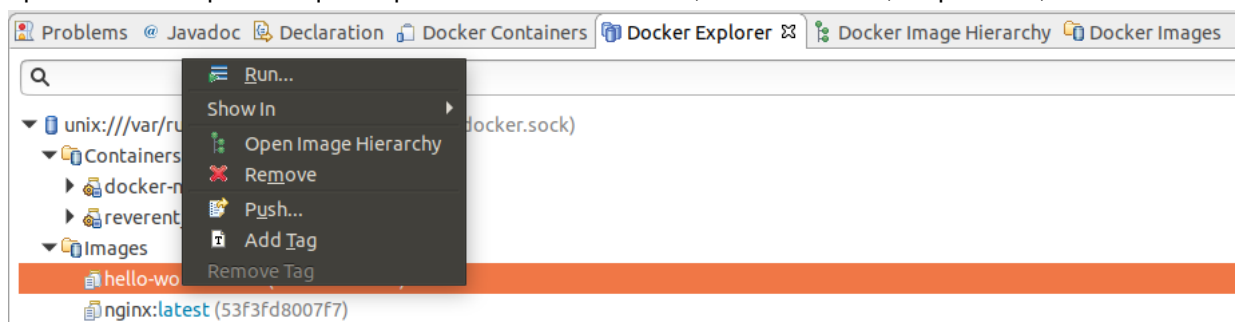
Otro concepto importante es el del Dockerfile, que es un archivo de configuración que se utiliza para crear imágenes. En dicho archivo se especifica qué es lo que debe tener la imagen, y los distintos comandos para instalar las herramientas necesarias. Por ejemplo, este sería el contenido de un DockerFile para tener una imagen de Ubuntu con Git instalado:

```
FROM Ubuntu:14.04
RUN apt-get update
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get -qq Install git
```

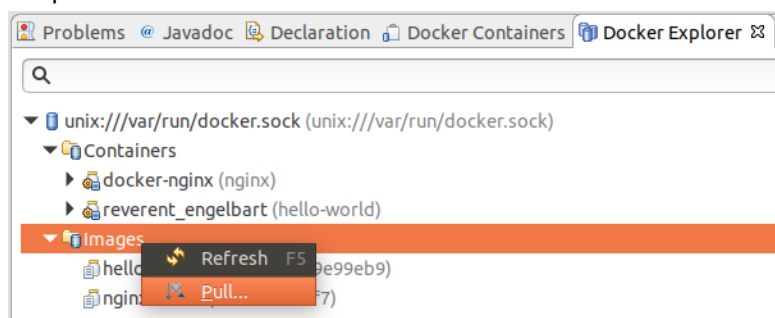
Normalmente se crean las imágenes partiendo de otras imágenes padre, en este caso, Docker debe partir de la imagen ubuntu: 14.04 y sobre ella crear la nueva, instalando posteriormente git por consola. Posteriormente, ejecutando el comando “docker build” sobre este DockerFile, se creará la imagen correspondiente, lista para crear contenedores a partir de ella.

Estos Dockerfiles se pueden almacenar en un sistema de control de versiones como Git/GitHub implementando por tanto el concepto de DevOps de **Infraestructure as Code (Iac)**.

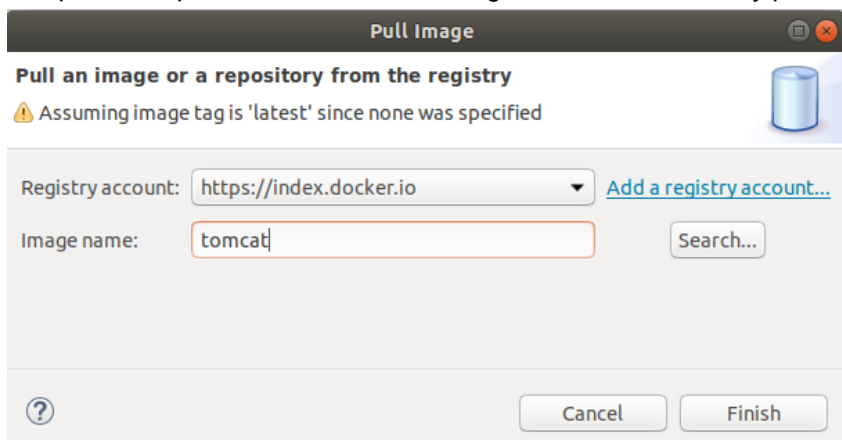
Dentro del entorno de Eclipse, se puede pulsar sobre una imagen con el botón derecho del ratón y aparecerán las opciones que se pueden hacer sobre ellas, como borrarlas, etiquetarlas, etc:



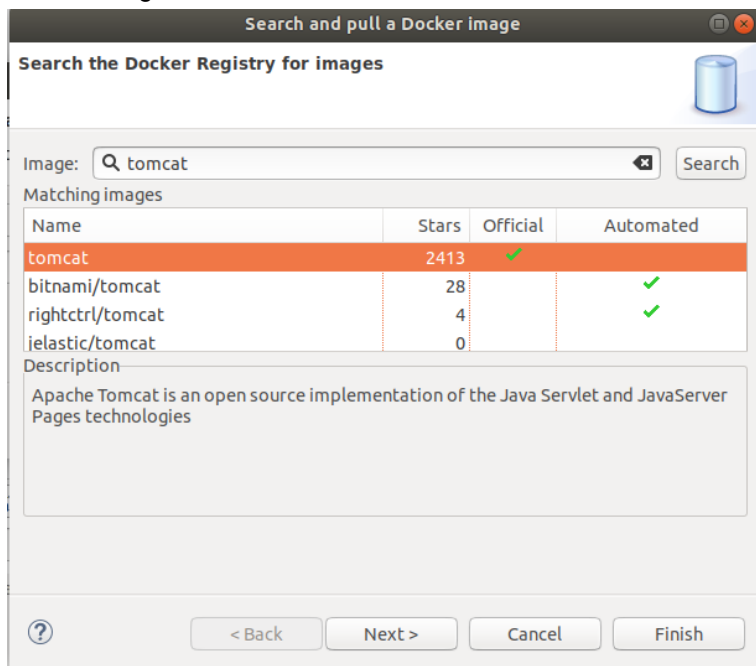
Si se desea cargar una nueva imagen se debe pinchar con el botón derecho del ratón sobre “Images” y después seleccionar “Pull”:



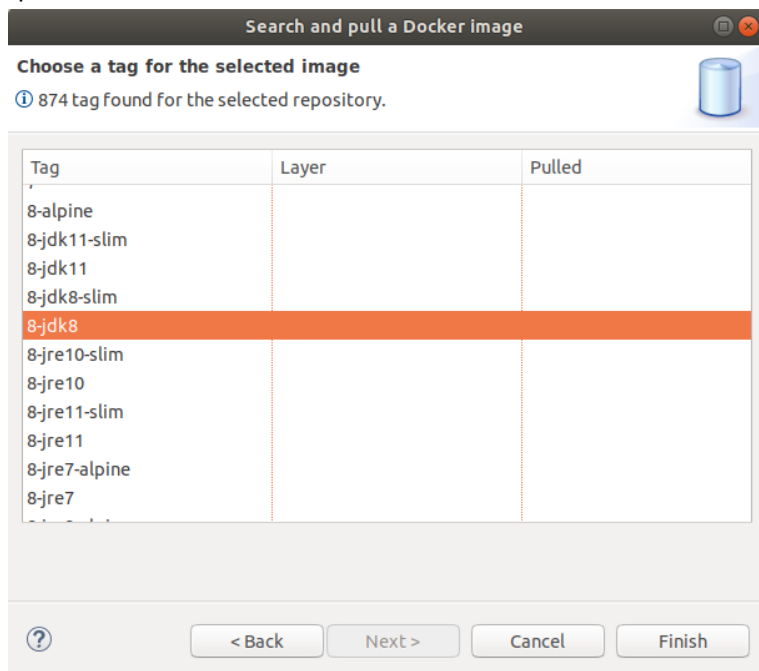
A continuación se abrirá una ventana donde aparece el HUB donde se van a buscar las imágenes y un campo donde poner el nombre de la imagen a buscar, rellenar y pulsar "Search":



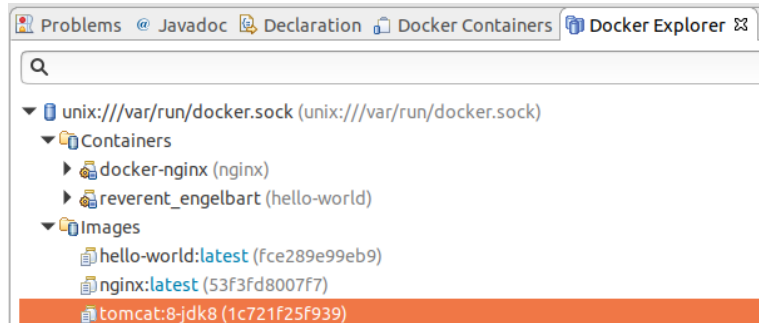
Se abre una nueva ventana con todas las imágenes encontradas. Es preferible seleccionar las que tienen el tick "Official" activado, y si no hay disponibles al menos que tengan el tick "Automated" activado que son las que se han generado a partir de un script que se puede revisar para verificar lo que se ha instalado en esa imagen:



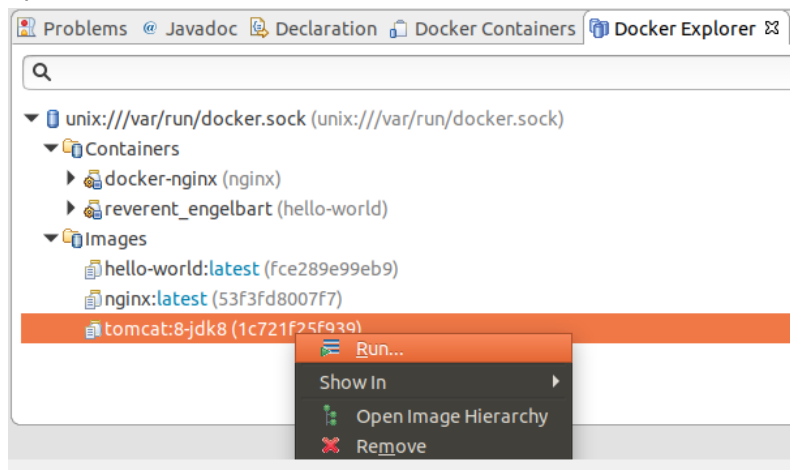
Seleccionar una, pulsar el botón “Finish”, que abrirá una ventana para seleccionar la versión (con su tag) que se desea:



Seleccionar una y pulsar el botón “Finish” en las 2 ventanas que hará que se descargue (pull) y se instale en la máquina y a continuación ya aparecerá en la lista de imágenes:



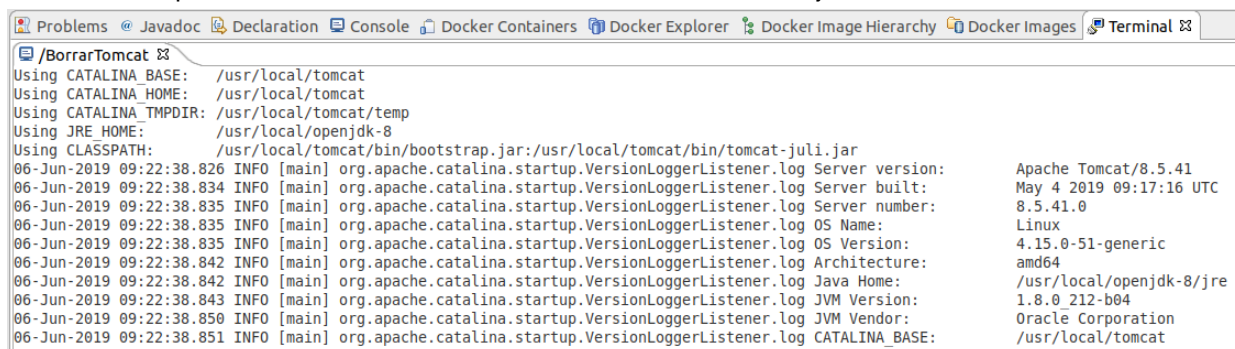
En este ejemplo hemos descargado la imagen de un servidor Apache Tomcat 8. Para ejecutarla se debe crear un contenedor a partir de ella, para ello, pulsar con el botón derecho sobre ella y seleccionar la opción “Run”:



En la ventana que se abre darle un nombre, configurar los puertos y la IP donde se va a ejecutar, y seleccionar los 3 ticks de la imagen, para que se borre el contenedor después de ejecutarse, finalmente pulsar el botón "Finish" para ejecutarlo:

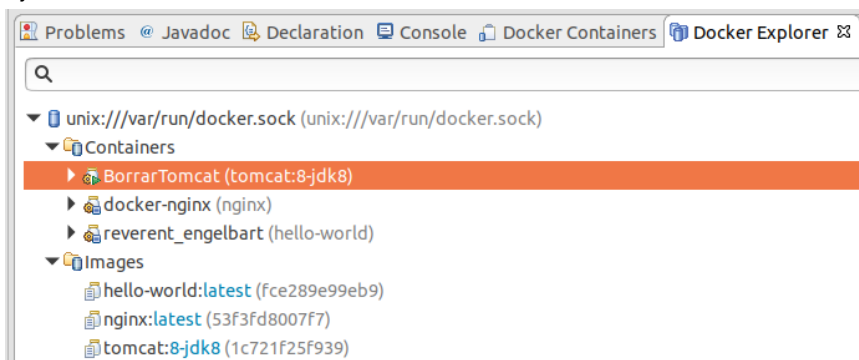


Se abrirá una pestaña con el terminal mostrando el resultado de la ejecución:

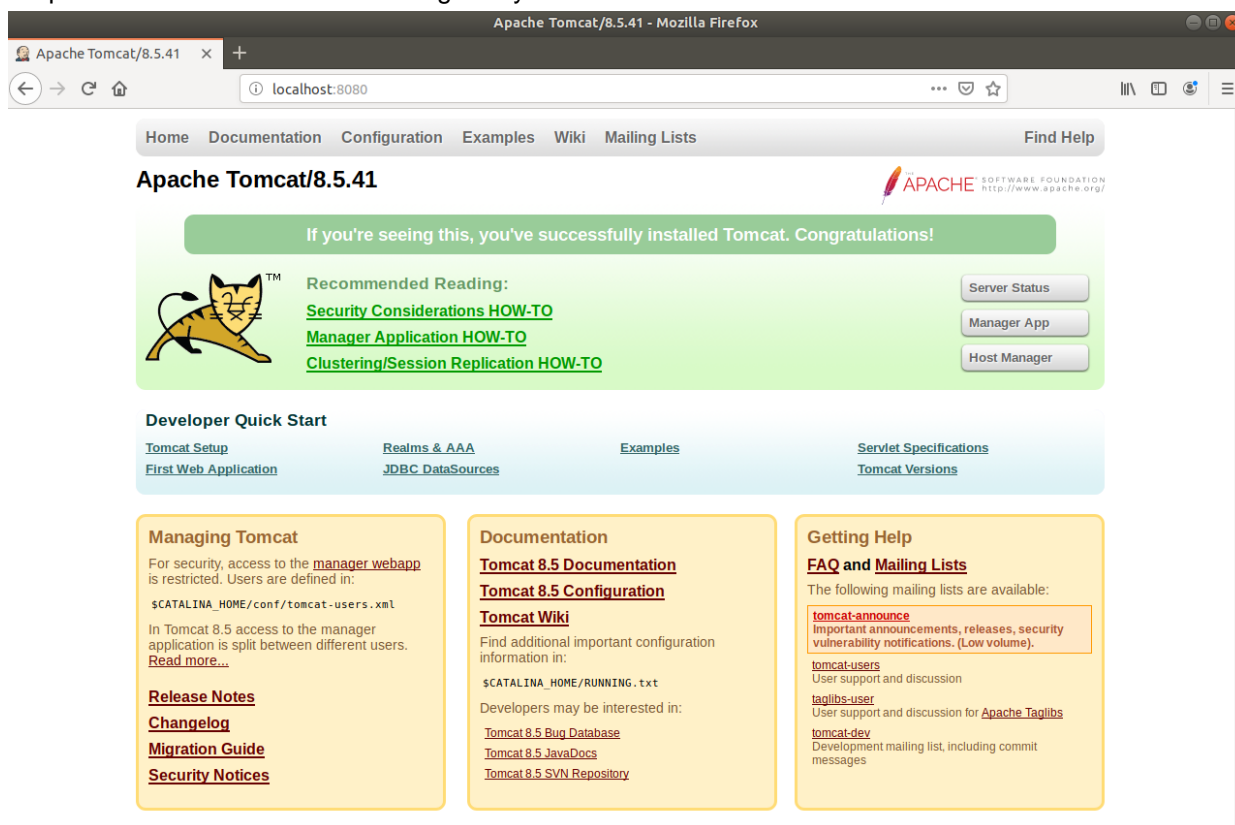




Y en la pestaña “Docker Explorer” se podrá ver un icono verde indicando que el contenedor se está ejecutando:



Se puede verificar abriendo un navegador y accediendo a su dirección:



También se puede verificar que funciona correctamente ejecutando el fichero War del Servlet generado en el apartado 5.2.

También existen otras herramientas como Kitematic y DockStation, que son interfaces gráficas más amigables e intuitivas que nos permiten trabajar más cómodamente con los contenedores Docker.

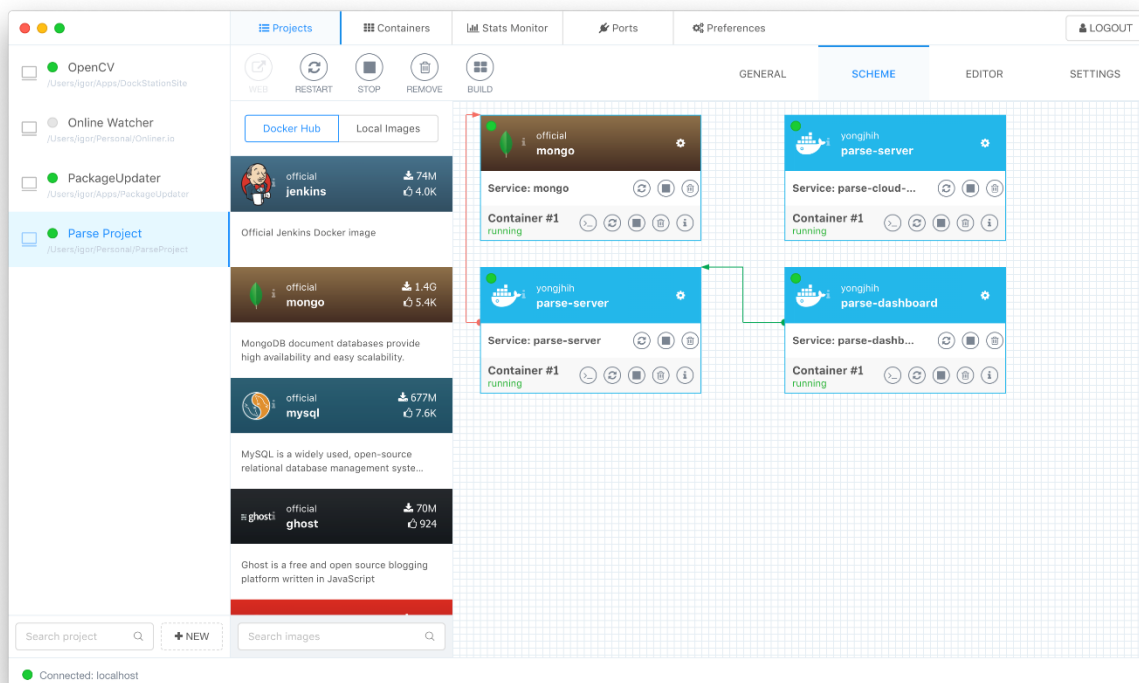


Figura 13. Ejemplo de gestión de contenedores con DockStation.

Fuente: <https://dockstation.io/>

## 5.15 Entorno de desarrollo completo con Docker

Ahora que ya se ha mostrado un entorno de desarrollo completo basado en la filosofía DevOps, con un pipeline completo con Jenkins y la utilización de Docker, se puede mejorar aún más el proceso de manera que todo el entorno de desarrollo esté formado por contenedores Docker, ya que todas las herramientas que se han utilizado se pueden incluir en contenedores Docker (de hecho, al ser herramientas Open Source, hay imágenes disponibles para todas ellas).

Todo esto se puede hacer con la herramienta **Docker Compose**, que se basa en un fichero en formato Yaml, el `docker-compose.yml`, donde se incluyen las definiciones de los servicios necesarios para crear los contenedores con las herramientas necesarias.

De esta manera se puede tener el entorno completo de desarrollo guardado bajo control de versiones (Git/GitHub) y desplegarlo en cualquier máquina en cuestión de minutos no de días como solía hacerse antes y sin tener que instalar ninguna aplicación en la máquina anfitriona evitando así problemas de librerías, configuraciones, etc. Es una forma de asegurar que todos los desarrolladores estén trabajando exactamente en el mismo entorno. Además, se puede facilitar el entorno al equipo de pruebas y también al de producción para que puedan montar el suyo propio a partir de él.

## 6 Conclusiones

---

### 6.1 Conclusiones y lecciones aprendidas

---

DevOps tiene sus raíces en metodologías iterativas, y más específicamente en la evolución de las herramientas para desarrollo ágiles. Intenta coordinar en un mismo ciclo iterativo el trabajo de dos departamentos (potencia la comunicación y el trabajo colaborativo) que hasta ahora estaban totalmente separados y que tradicionalmente han sido fuente de rivalidades y discusiones: Desarrollo y Operaciones. No es una moda pasajera sino que cada vez se va a ir implantando más fuertemente en las empresas de desarrollo de software por los múltiples beneficios que conlleva.

La gran cantidad de herramientas Open Source que han surgido para trabajar en este tipo de proyectos hace que se puedan implementar en cualquier empresa con un coste de software muy bajo. Sin embargo la complejidad de los procesos para configurar el ciclo de desarrollo completo, así como la dificultad de involucrar a todas las personas necesarias y de derribar las paredes entre los distintos departamentos hace que sea una tarea muy compleja, pero que merece la pena realizar por los beneficios que aporta con el paso del tiempo.

Este TFG contribuye a dar una visión general de lo que esta filosofía aporta, tanto a nivel teórico como a nivel práctico. A nivel teórico se hace una breve introducción a todo lo que implica y a nivel práctico se implementa una solución completa de integración y entrega continua con herramientas Open Source de manera que cualquiera pueda implementarla sin tener que invertir en software específico. En la parte práctica también se explica el manejo de contenedores con Docker y su integración con el entorno de desarrollo.

A nivel personal, el trabajo realizado durante este TFG ha sido muy satisfactorio, ya que me ha permitido ampliar mis conocimientos sobre DevOps y trabajar con herramientas con las que nunca antes había trabajado. Espero poder aplicar todo lo que he aprendido en los proyectos en los que actualmente trabajo.

Para todos los que lean este TFG espero que su lectura sea didáctica y les ayude a adoptar los principios de DevOps, y que la parte práctica desarrollada les sirva como punto de partida para sus propios procesos de desarrollo internos.

### 6.2 Objetivos alcanzados

---

Con la entrega de esta memoria se puede considerar que todos los objetivos propuestos al inicio del TFG, definidos en la sección 1.5, se han cumplido:

- El primer objetivo se ha cumplido describiendo lo que es DevOps, así como su historia, sus principios, objetivos, costes, etc., detallados en el capítulo 3.
- El segundo objetivo se ha cumplido describiendo el estado del arte de las herramientas empleadas en DevOps, poniendo mayor énfasis en las herramientas Open Source, en el capítulo 4.
- El tercer objetivo se ha cumplido con el ejemplo práctico descrito en el capítulo 5, donde se ha desarrollado un pipeline completo de DevOps utilizando Jenkins y además se ha mostrado cómo trabajar con contenedores utilizando Docker.

Respecto al cumplimiento de la planificación inicial, en todo momento se ha seguido la misma, sin desviaciones relevantes que destacar. Se ha realizado un trabajo constante durante todo el tiempo de desarrollo del TFG.

### **6.3 Líneas de trabajo futuras**

---

Como líneas futuras de desarrollo sería interesante realizar toda la parte práctica con el sistema operativo Microsoft Windows 10, ya que recientemente ha aparecido la versión de Docker para dicho sistema operativo, de manera que la parte práctica de este trabajo le pueda ser útil también a los que trabajen con Windows.

También sería interesante generar un entorno de desarrollo completo, incluyendo un pipeline completo de Jenkins, en Docker Compose y compartirlo en un repositorio GitHub, para que cualquiera se lo pueda descargar y pueda tener un entorno de desarrollo completo listo para empezar a trabajar con él en cuestión de minutos. Se podría generar uno para cada uno de los lenguajes de programación más demandados actualmente, por ejemplo para: Java, C++, Python y PHP.

## 7 Glosario de términos

---

Termino	Significado
<b>AWS</b>	Amazon Web Services
<b>BIOS</b>	Basic Input/Output System
<b>CD</b>	Entrega Continua
<b>CI</b>	Integración Continua
<b>DevOps</b>	Development & Operations
<b>ITIL</b>	Information Technology Infrastructure Library
<b>OS</b>	Sistema Operativo
<b>PEC</b>	Prueba de Evaluación Continua
<b>QA</b>	Calidad
<b>RAM</b>	Random Access Memory
<b>SCM</b>	Source Code Management
<b>SNMP</b>	Simple Network Management Protocol
<b>TFG</b>	Trabajo Final de Grado
<b>TI</b>	Tecnologías de la Información
<b>UI</b>	User Interface
<b>UOC</b>	Universitat Oberta de Catalunya
<b>VM</b>	Virtual Machine
<b>WIP</b>	Work In Progress

Tabla 3. Glosario

## 8 Bibliografía

---

- [1] <https://www.devopsdays.org/>, [Fecha de consulta 10/03/2019]
- [2] <https://www.amazon.com/Gene-Kim/e/B00AERCJ9E>, [Fecha de consulta 10/03/2019]
- [3] <http://sunqu.net/es-compatible-devops-con-itil/>, [Fecha de consulta 11/03/2019]
- [4] <http://pmcgrupo.com/2017/04/18/el-impacto-de-devops-en-su-implementacion-itil/>, [Fecha de consulta 11/03/2019]
- [5] [https://es.wikipedia.org/wiki/Information\\_Technology\\_Infrastructure\\_Library](https://es.wikipedia.org/wiki/Information_Technology_Infrastructure_Library), [Fecha de consulta 12/03/2019]
- [6] [https://es.wikipedia.org/wiki/C%C3%ADrculo\\_de\\_Deming](https://es.wikipedia.org/wiki/C%C3%ADrculo_de_Deming), [Fecha de consulta 14/03/2019]
- [7] <https://jbravomontero.wordpress.com/2015/04/29/31-ejemplos-de-arquitectura-para-devops-y-entrega-continua/>, [Fecha de consulta 21/03/2019]
- [8] <https://xebialabs.com/periodic-table-of-devops-tools/>, [Fecha de consulta 01/04/2019]
- [9] Gene Kim, "DevOps distilled, Pat 1: The three underlying principles", <https://www.ibm.com/developerworks/library/se-devops/>, [Fecha de consulta 10/03/2019]
- [10] <https://travis-ci.org/>, [Fecha de consulta 10/04/2019]
- [11] <https://djangostars.com/blog/continuous-integration-circleci-vs-travisci-vs-jenkins/>, [Fecha de consulta 10/04/2019]
- [12] <https://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack>, [Fecha de consulta 10/04/2019]
- [13] <https://dev.mysql.com/downloads/mysql/>, [Fecha de consulta 05/05/2019]
- [14] <https://jenkins.io/doc/administration/requirements/java/>, [Fecha de consulta 10/05/2019]
- [15] <https://hub.docker.com/>, [Fecha de consulta 21/05/2019]