

Instalar las Herramientas de Desarrollo



Xubuntu

1. Instalación de Java 11 mediante el OpenJDK:

<https://jdk.java.net/java-se-ri/11>

La última versión ahora es:

https://download.java.net/openjdk/jdk11/ri/openjdk-11+28_linux-x64_bin.tar.gz

Copiar el tar.gz a la carpeta /opt (sudo)

Extraer el tar

```
sudo tar -xvf openjdk-11+28_linux-x64_bin.tar.gz
```

Esto crea una carpeta /opt/jdk-11

Ejecutamos /opt/jdk-11/bin/java -version para identificar la versión de java que acabamos de desempaquetar

```
openjdk 11 2018-09-25
OpenJDK Runtime Environment 18.9 (build 11+28)
OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)
```

2. Instalación de Maven:

<https://maven.apache.org/download.cgi>

La última versión ahora es:

<https://ftp.cixug.es/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

Copiar el tar.gz a la carpeta /opt (sudo)

Extraer el tar

```
sudo tar -xvf apache-maven-3.6.3-bin.tar.gz
```

Esto crea una carpeta /opt/apache-maven-3.6.3/

3. Configurar Java y Maven para la ejecución

Configurar las variables de entorno JAVA_HOME y PATH para la ejecución de las dos herramientas desde nuestro usuario añadiendo al final del fichero ~/.profile los comandos:

```
# User specific environment variables
export JAVA_HOME=/opt/jdk-11
PATH=$JAVA_HOME/bin:$PATH

# Add maven directory to PATH
PATH=/opt/apache-maven-3.6.3/bin:$PATH

export PATH
```

Para hacerlo efectivo es necesario hacer logout y login de nuevo.

Se comprueban las versiones:

```
drdel@drd-xubul:~$ java --version
openjdk 11 2018-09-25
OpenJDK Runtime Environment 18.9 (build 11+28)
OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)
drdel@drd-xubul:~$ mvn --version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /opt/apache-maven-3.6.3
Java version: 11, vendor: Oracle Corporation, runtime: /opt/jdk-11
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.8.0-44-generic", arch: "amd64", family: "unix"
```

4. Instalación de Git

Se instala con el comando apt directamente

```
sudo apt-get install git
```

Se comprueba la version:

```
drdel@drd-xubul:~$ git --version
git version 2.25.1
```

5. Instalación de Docker

Es conveniente hacer un update y upgrade previo al sistema

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

A continuación, se instalan los paquetes de requisitos previos que permitan a apt usar paquetes a través de HTTPS:

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

Luego, se añade la clave de GPG para el repositorio oficial de Docker en su sistema:

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

Agregue el repositorio de Docker a las fuentes de APT ya que no está en el repositorio de Ubuntu:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

A continuación, se actualiza el paquete de base de datos con los paquetes de Docker del repositorio recién agregado:

```
sudo apt update
```

Nos aseguramos que la instalación de Docker-ce se realizará desde el repositorio de Docker (recién añadido) en lugar del repositorio predeterminado de Ubuntu:

```
apt-cache policy docker-ce
```

Si bien el número de versión de Docker puede ser distinto, se verá un resultado como el siguiente:

```
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.3~3-0~ubuntu-focal
  Version table:
     5:20.10.3~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

Se observa que docker-ce no está instalado, pero la opción más viable para la instalación es del repositorio de Docker para Ubuntu 20.10 (focal).

Por último, instalamos Docker:

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

Comprobamos la versión instalada

```
drdel@drd-xubul:~$ docker --version
Docker version 20.10.3, build 48d30b5
```

Con esto, Docker quedará instalado, el demonio se iniciará y el proceso se habilitará para ejecutarse en el inicio. Compruebe que funcione:

```
sudo systemctl status docker
```

El resultado debe ser similar al siguiente, y mostrar que el servicio está activo y en ejecución:

```
drdel@drd-xubul:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Thu 2021-02-25 17:09:18 CET; 1min 59s ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 9322 (dockerd)
     Tasks: 8
    Memory: 40.0M
    CGroup: /system.slice/docker.service
            └─9322 /usr/bin/dockerd -H fd:// --
   containerd=/run/containerd/containerd.sock
```

La instalación de Docker proporciona no solo el servicio de Docker (demonio) sino también la utilidad de línea de comandos docker o el cliente de Docker.

Con el fin de evitar tener que lanzar docker con un sudo vamos a añadir nuestro usuario al grupo de Docker:

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

Para hacerlo efectivo es necesario hacer logout y login de nuevo.

Unos primeros pasos con Docker para ver su correcto funcionamiento

Descarga de la imagen hello-world de Docker HUB, comprobación de que la imagen está creada, ejecución del container y comprobación del proceso del container finalizado. A continuación se borra el container (cada uno puede tener su nombre diferente) y se comprueba que ha dejado de existir.

https://hub.docker.com/_/hello-world

```
drdel@drd-xubul:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
drdel@drd-xubul:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:7e02330c713f93b1d3e4c5003350d0dbe215ca269dd1d84a4abc577908344b30
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
drdel@drd-xubul:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-world latest bf756fb1ae65 13 months ago 13.3kB
drdel@drd-xubul:~$ docker run hello-world
```

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/>

```
drdel@drd-xubul:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
976c681f1c80   hello-world    "/hello"                20 seconds ago Exited (0) 19 seconds ago
mystifying_chatelet
drdel@drd-xubul:~$ docker rm mystifying_chatelet
mystifying_chatelet
drdel@drd-xubul:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS         NAMES
```

6. Visual Studio Code

Desde la pagina de download del vscode nos descargamos la última versión para Linux 64bits en paquete .deb

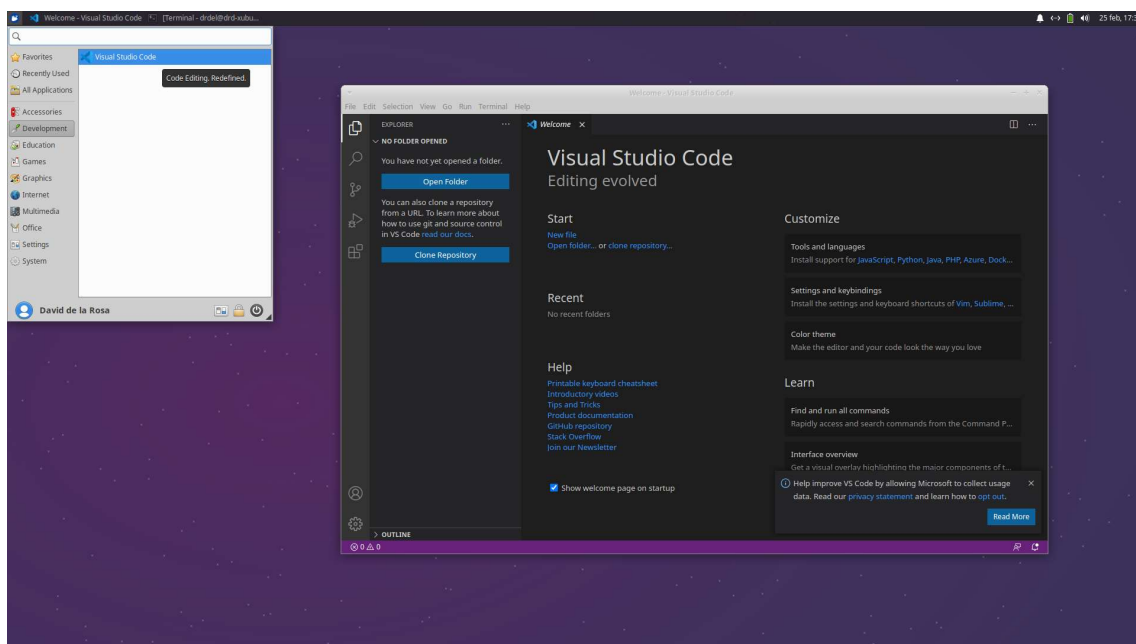
<https://code.visualstudio.com/Download>

En este momento es https://code.visualstudio.com/docs/?dv=linux64_deb

Esto nos dejara en el directorio ~/Downloads el fichero code_1.53.2-1613044664_amd64.deb ejecutamos el comando para instalarlo:

```
sudo dpkg -i ~/Downloads/code_1.53.2-1613044664_amd64.deb
```

Maravilloso, ya tenemos nuestro VSCode en un entorno virtualizado, tenemos Git, tenemos Java, maven.. podemos incluir las extensiones que queramos y a codificar.




1.1. Visual Studio Code Java Extensions

Java Extension Pack es uno de los paquetes más usados para Java en VSCode y está esponsorizado por Red Hat, para instalarlo hay que buscarlo en extensiones e instalarlo



<https://marketplace.visualstudio.com/items?vscjava.vscode-java-pack>

En la web del market place se recomiendan otras extensiones para Spring, Quarkus, linting de Sonar, Microservices, etc que podemos instalar según nuestras necesidades



Java Extension Pack

Preview

Microsoft |  6,986,201 installs |  (30) | Free

Popular extensions for Java development and more.

[Install](#) [Trouble Installing?](#)

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

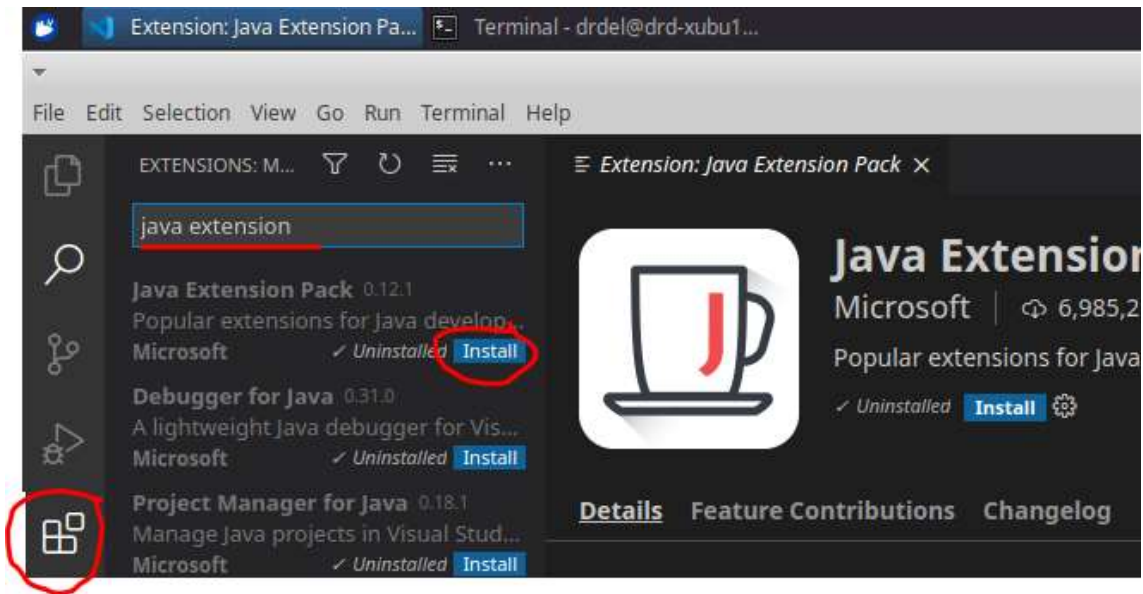
Java Extension Pack

Java Extension Pack is a collection of popular extensions that can help write, test and debug Java applications in Visual Studio Code. Check out [Java in VS Code](#) to get started.

Extensions Included

By installing Java Extension Pack, the following extensions are installed:

-  [Language Support for Java™ by Red Hat](#)
 - Code Navigation
 - Auto Completion
 - Refactoring
 - Code Snippets
-  [Debugger for Java](#)
 - Debugging
-  [Java Test Runner](#)
 - Run & Debug JUnit/TestNG Test Cases
-  [Maven for Java](#)
 - Project Scaffolding
 - Custom Goals
-  [Project Manager for Java](#)
 - Manage Java projects, referenced libraries, resource files, packages, classes, and class members
-  [Visual Studio IntelliCode](#)
 - AI-assisted development
 - Completion list ranked by AI



Para Git son recomendables **Git Lens** y **Git Graph** el primero está muy extendido y el segundo es para los que echamos de menos el Git Extensions de entorno Windows en VSCode

<https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens>

<https://marketplace.visualstudio.com/items?itemName=mhutchie.git-graph>