

## Team City: CI-CD



### Indice

Introduccion.....	2
Historia:.....	2
Licencia y descarga:.....	2
Instalación:.....	2
Herramientas con las que se relaciona:.....	5
Definición y ejemplo de pipeline:.....	6
Conclusión:.....	9

# Introduccion

TeamCity es una herramienta utilizada para integración continua y manejo de builds. A continuación trataremos sus ventajas, desventajas y particularidades como herramienta de cara al desarrollo.

## Historia:

TeamCity es un servidor de integración continua y manejo de builds desarrollado por JetBrains, empresa detrás de herramientas e IDEs tan conocidos como IntelliJ, PhpStorm y el lenguaje de programación Kotlin. Fue creado en 2006 y desde entonces sigue en continuo desarrollo y mantenimiento.



## Licencia y descarga:

Esta herramienta es utilizable por cualquiera bajo licencia. La principal y que utilizaremos es una licencia “freemium” es decir, licencia gratuita, incluso para uso comercial, con algunos servicios limitados bajo pago. La limitación que recibe esta licencia es 100 configuraciones de build y 3 agentes de compilación como máximo.

Esto suele ser suficiente para desarrolladores en solitario, y para empresas la licencia de pago les permitiría configuraciones y agentes ilimitados.

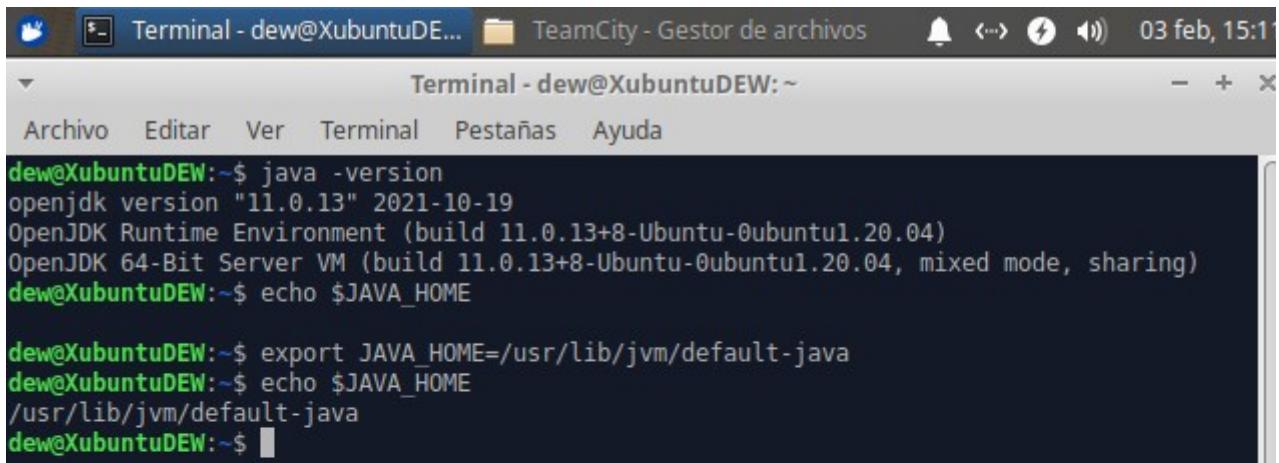
La descarga es sencilla, en nuestro caso utilizaremos Linux, y desde la página oficial de JetBrains podremos descargar el programa comprimido para su uso:

The screenshot shows the JetBrains TeamCity website. The header includes the TeamCity logo and navigation links. The main content area features three pricing plans:

- TeamCity Cloud** (Hosted by JetBrains):
  - For cloud-native teams and newcomers to CI/CD
  - Unlimited web users
  - Unlimited concurrency
  - Fleet of Linux and Windows build agents maintained by JetBrains
  - Supports adding your own build agents
  - Try free for 14 days, then from \$45 per month
  - Learn more about TeamCity Cloud
  - Start Building for Free
- TeamCity Professional** (Hosted by you):
  - For pros who need to control everything
  - Unlimited users
  - Unlimited build time
  - 100 build configurations
  - 3 build agents included
  - All features of TeamCity Enterprise
  - Free forever, even for commercial use
  - Docker image: `docker pull jetbrains/teamcity-server`
  - Free Download | .tar.gz
- TeamCity Enterprise** (CI/CD that works at scale where others fail):
  - Unlimited users
  - Unlimited build time
  - Unlimited build configurations
  - Priority support
  - From \$1,999 per year, with a 50% renewal discount
  - Get Evaluation License | Request Info

## Instalación:

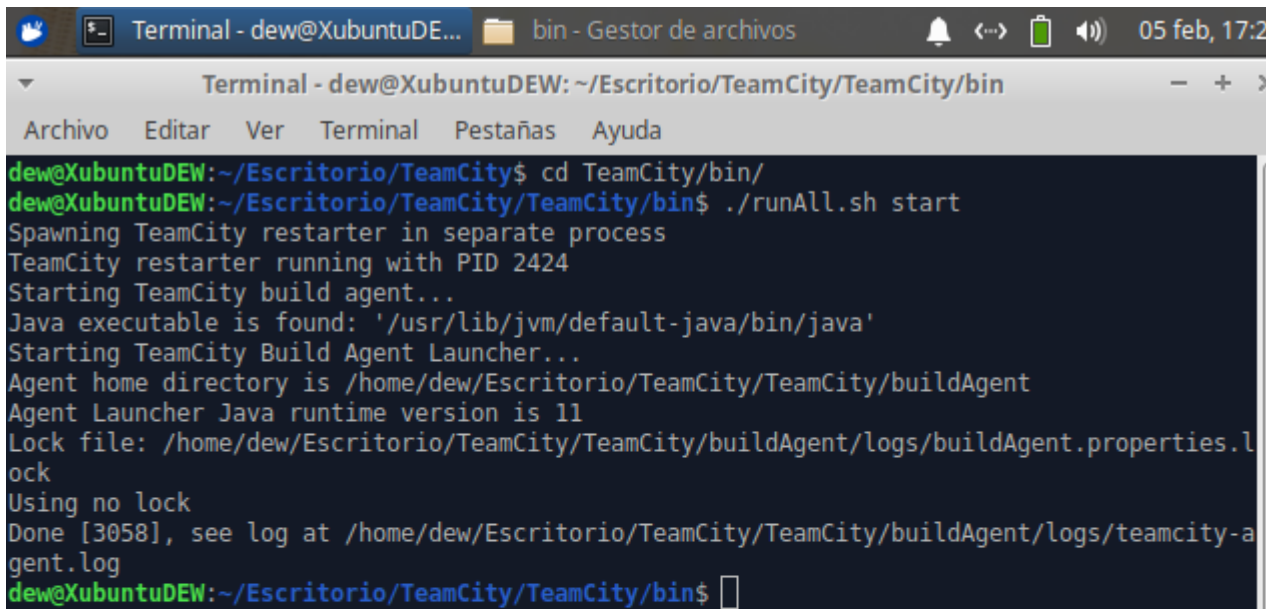
Como requisito previo a la utilización de TeamCity será necesario tener instalado Java JRE o JDK, ya que es una aplicación basada en Java. Comprobamos la versión de java y asignamos un JAVA\_HOME en caso de que no lo haya:

A terminal window titled 'Terminal - dew@XubuntuDE...' with a menu bar (Archivo, Editar, Ver, Terminal, Pestañas, Ayuda). The terminal shows the following commands and output:

```
dew@XubuntuDEW:~$ java -version
openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.20.04)
OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)
dew@XubuntuDEW:~$ echo $JAVA_HOME

dew@XubuntuDEW:~$ export JAVA_HOME=/usr/lib/jvm/default-java
dew@XubuntuDEW:~$ echo $JAVA_HOME
/usr/lib/jvm/default-java
dew@XubuntuDEW:~$
```

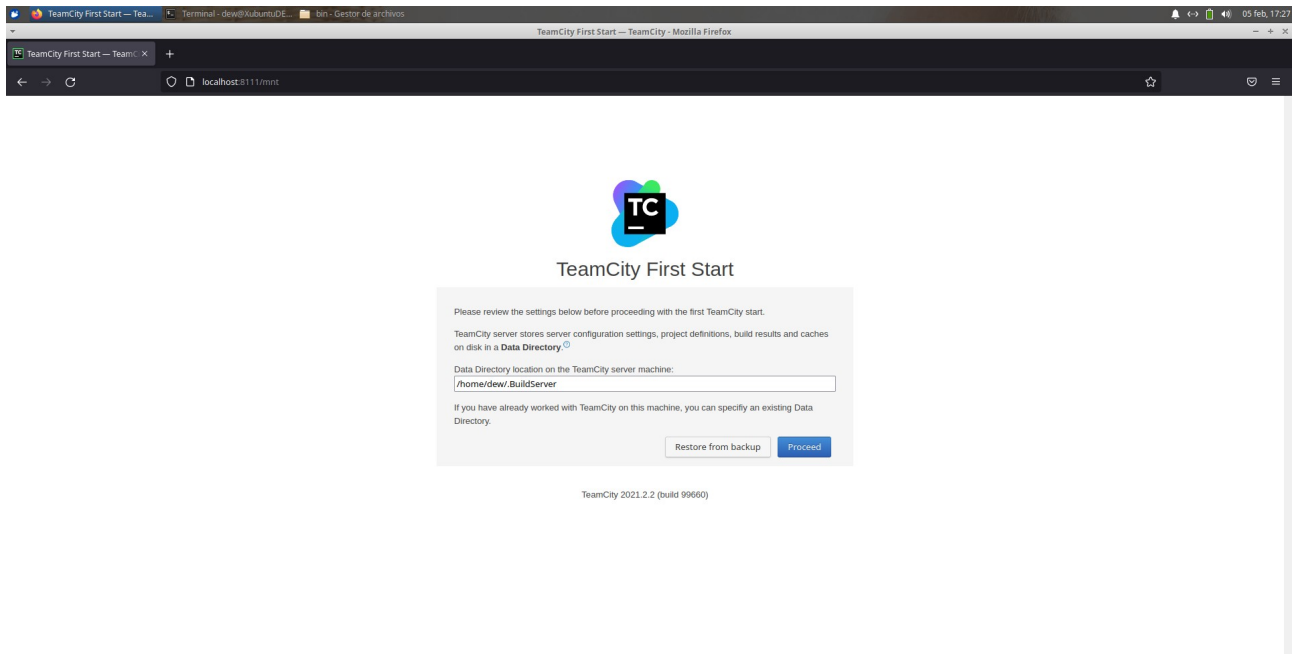
A continuación ya podremos ejecutar el script de inicio del servidor, ya que en Linux no requiere de instalación. Navegamos a la carpeta en la que se encuentra y ejecutamos el script runAll, que ejecuta el servidor con un agente por defecto:

A terminal window titled 'Terminal - dew@XubuntuDE...' with a menu bar (Archivo, Editar, Ver, Terminal, Pestañas, Ayuda). The terminal shows the following commands and output:

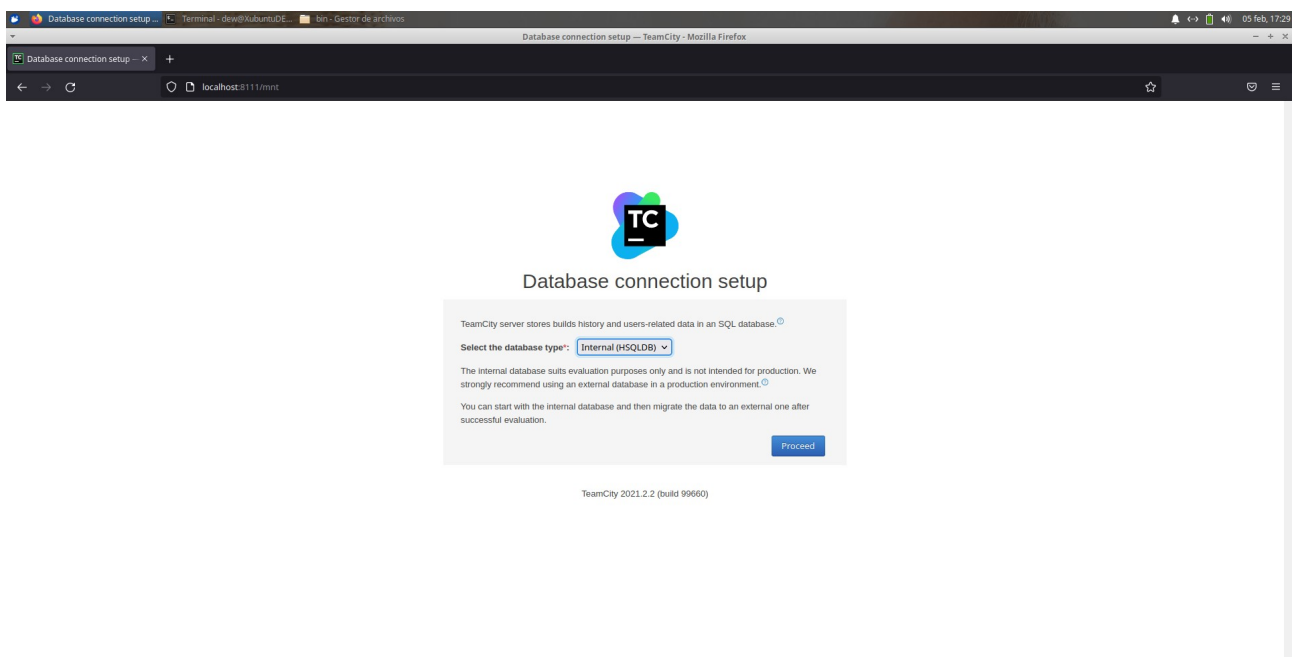
```
dew@XubuntuDEW:~/Escritorio/TeamCity$ cd TeamCity/bin/
dew@XubuntuDEW:~/Escritorio/TeamCity/TeamCity/bin$ ./runAll.sh start
Spawning TeamCity restarter in separate process
TeamCity restarter running with PID 2424
Starting TeamCity build agent...
Java executable is found: '/usr/lib/jvm/default-java/bin/java'
Starting TeamCity Build Agent Launcher...
Agent home directory is /home/dew/Escritorio/TeamCity/TeamCity/buildAgent
Agent Launcher Java runtime version is 11
Lock file: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/logs/buildAgent.properties.lock
Using no lock
Done [3058], see log at /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/logs/teamcity-agent.log
dew@XubuntuDEW:~/Escritorio/TeamCity/TeamCity/bin$
```

Con el parámetro “start” iniciaremos el servidor y cuando queramos cerrarlo lo ejecutaremos con “stop”.

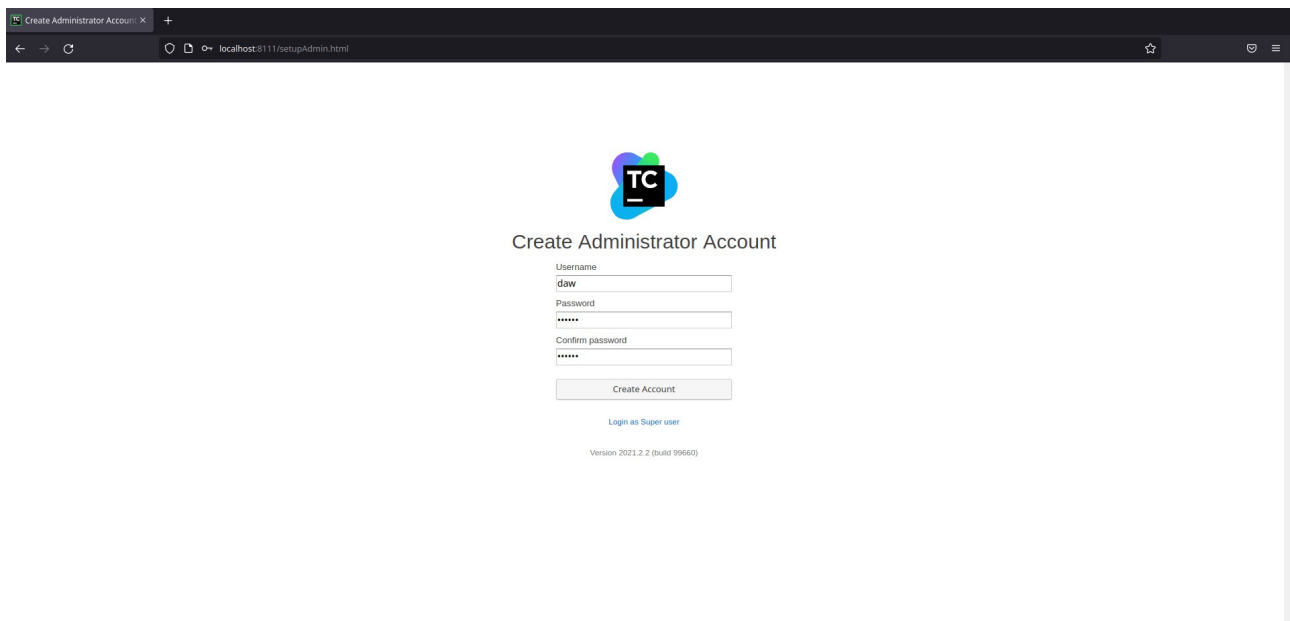
Ahora podremos acceder a la interfaz desde el navegador en local usando el puerto **8111**.



En primera ejecución habrá que realizar una configuración. Especificaremos el directorio de datos.

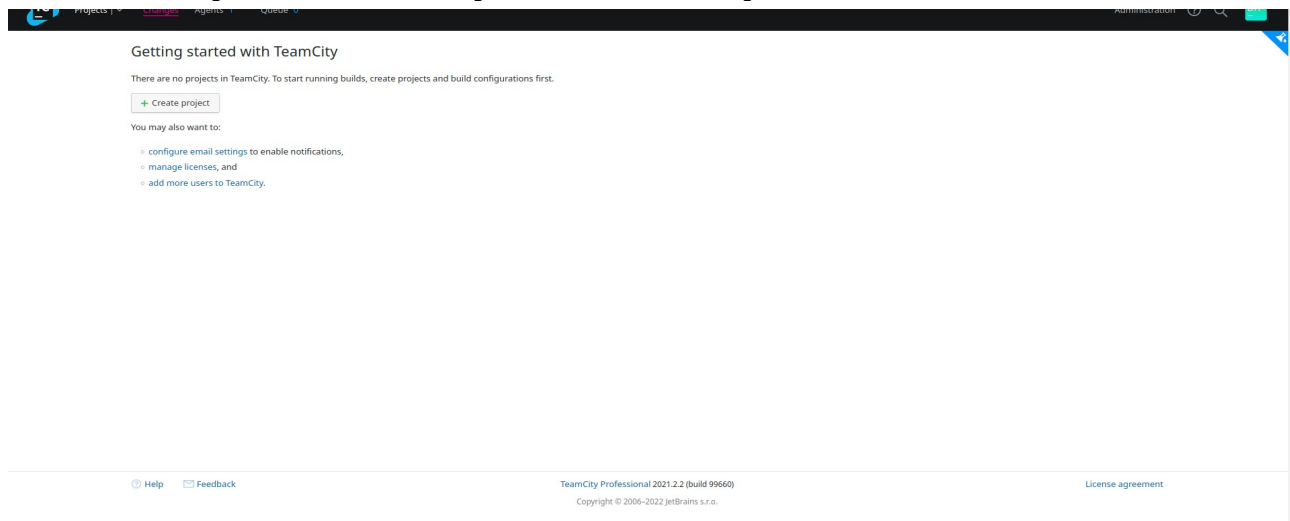


Especificaremos la base de datos (En este caso la interna, para hacer pruebas).



Y creamos una cuenta de administrador.

Con todos estos pasos realizados la aplicación estará lista para usar:



## Herramientas con las que se relaciona:

Al tratarse TeamCity de una herramienta creada por JetBrains está en continuo desarrollo y permite con el uso de plugins la integración con los principales IDEs, por ejemplo:



IDEs JetBrains: IDEA, PyCharm, PhpStorm... Todos estos IDEs al ser desarrollados por la misma empresa poseen plugins que integran directamente con TeamCity, permitiendo ver los datos de builds y pipelines, editarlos, lanzar builds etc. Todo desde el propio IDE, sin necesidad de acceder a la propia herramienta.



Eclipse: Eclipse posee un plugin diseñado para permitir lanzar builds y ver los resultados de builds anteriores desde su propia interfaz.



Visual Studio Code: Este programa también incluye un plugin que permite utilizar toda la potencia de TeamCity desde el IDE, permitiendo al usuario hacer todas sus operaciones desde el mismo programa.



Git: TeamCity permite al usuario la creación de proyectos desde repositorios de Git, permitiendo al usuario hacer pruebas sobre proyectos remotos.

## Definición y ejemplo de pipeline:

Para demostrar la utilización de esta herramienta crearemos un proyecto y un pipeline de prueba.

En primer lugar crearemos un proyecto de prueba desde la URL de un repositorio de Git:

Este contendrá una app de ejemplo sencilla de Maven.

<https://github.com/mkjetbrains/SimpleMavenSample>

Administration / <Root project>

### Create Project

From a repository URL

Manually

Parent project: \* <Root project>

Repository URL: \*

A VCS repository URL. Supported formats: `http(s)://`, `svn://`, `git://`, etc. as well as URLs in Maven format.

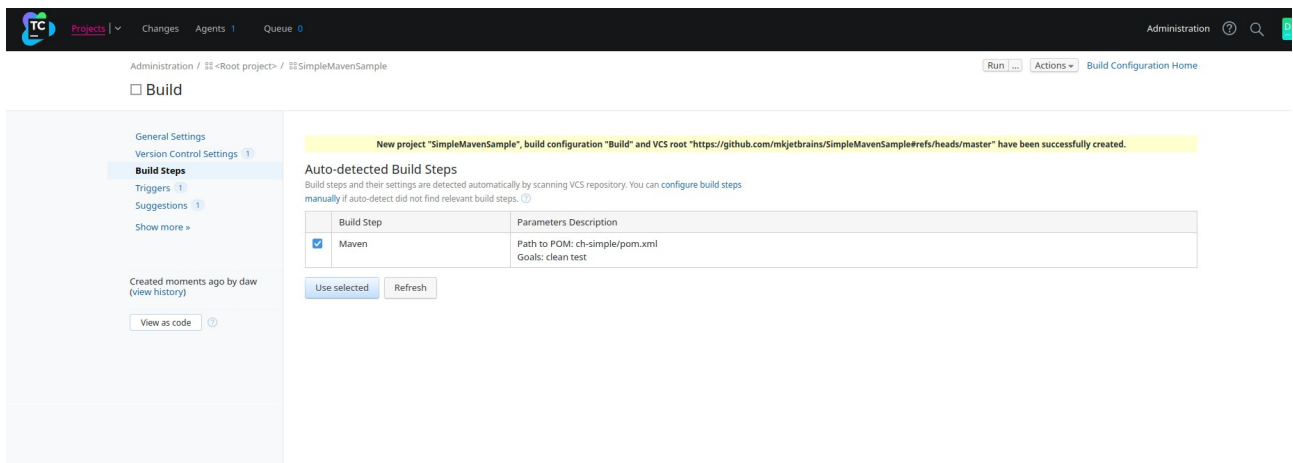
Username:

Provide a username if access to the repository requires authentication.

Password / access token:

Provide a password or a personal access token if access to the repository requires authentication.

En este caso no será necesaria la identificación de usuario, pero en caso de ser un repositorio privado habría que introducirla.



Administration / <<Root project> / SimpleMavenSample

Run Actions Build Configuration Home

Build

General Settings  
Version Control Settings  
**Build Steps**  
Triggers  
Suggestions  
Show more »

Created moments ago by daw  
(view history)

View as code

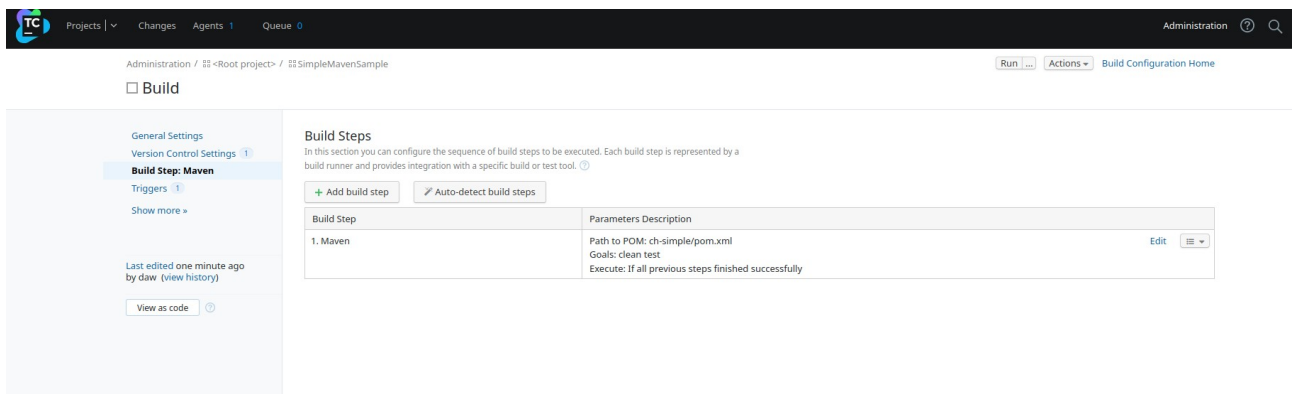
New project "SimpleMavenSample", build configuration "Build" and VCS root "https://github.com/mkjetbrains/SimpleMavenSample#refs/heads/master" have been successfully created.

**Auto-detected Build Steps**  
Build steps and their settings are detected automatically by scanning VCS repository. You can configure build steps manually if auto-detect did not find relevant build steps.

Build Step	Parameters Description
<input checked="" type="checkbox"/> Maven	Path to POM: ch-simple/pom.xml Goals: clean test

Use selected Refresh

TeamCity escanea el repositorio y detecta automáticamente pasos para la build. En este caso al ser un proyecto Maven nos sugiere como primer paso realizar un Maven clean test. Podemos aceptar los pasos sugeridos o añadirlos posteriormente si queremos. En este caso aceptaremos este primer paso.



Administration / <<Root project> / SimpleMavenSample

Run Actions Build Configuration Home

Build

General Settings  
Version Control Settings  
**Build Step: Maven**  
Triggers  
Show more »

Last edited one minute ago by daw  
(view history)

View as code

**Build Steps**  
In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build runner and provides integration with a specific build or test tool.

+ Add build step Auto-detect build steps

Build Step	Parameters Description
1. Maven	Path to POM: ch-simple/pom.xml Goals: clean test Execute: If all previous steps finished successfully

Edit

En la página de la build podremos añadirle nuevos pasos a la build o modificar los que tenemos, además de ejecutar los pasos actuales.

Vamos a probar a añadirle un nuevo paso a nuestra build, antes de ejecutar el test de maven comprobaremos la versión de maven:

TC

Projects |

Changes

Agents 1

Queue 0

Administration

Administration / <Root project> / SimpleMavenSample

Run | Actions | Build Configuration Home

Build

General Settings

Version Control Settings 1

**Build Step: Maven**

Triggers 1

Show more >

Last edited 5 minutes ago by daw (view history)

View as code

New Build Step

Runner type: Command Line

Simple command execution

Step name: Check Maven version

Optional, specify to distinguish this build step from other steps.

Run: Custom script

Custom script: \*

Enter build script content:

1 mvn --version

A platform-specific script, which will be executed as a .cmd file on Windows or as a shell script in Unix-like environments.

Docker Settings

Run step within Docker container:

E.g. ruby:2.4. TeamCity will start a container from the specified image and will try to run this build step within this container.

Show advanced options

Save

Cancel

-- Choose build runner type --

-- Choose build runner type --  
.NET  
Ant  
C# Script  
Command Line  
Container Deployer  
Docker  
Docker Compose  
Duplicates finder (Java)  
Duplicates finder (ReSharper)  
FTP Upload  
FxCop  
Gradle  
Inspections (IntelliJ IDEA)  
Inspections (ReSharper)  
IntelliJ IDEA Project

Como podemos ver podemos elegir con que “Runner” se ejecutará nuestro paso, permitiendo utilizar gran cantidad de herramientas como Docker, Maven e incluso proyectos de IDEs. En este caso utilizaremos la linea de comandos y colocaremos el script deseado en el campo de texto.

## Build Steps

In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build runner and provides integration with a specific build or test tool.

+ Add build step

Reorder build steps

Auto-detect build steps

Build Step	Parameters Description	
1. Check Maven version	Command Line Custom script: mvn --version Execute: If all previous steps finished successfully	<a>Edit</a> <div></div>
2. Maven	Path to POM: ch-simple/pom.xml Goals: clean test Execute: If all previous steps finished successfully	<a>Edit</a> <div></div>

Comprobamos que los pasos solo se ejecutarán si el anterior tuvo exito y ejecutamos nuestra build.



SimpleMavenSample / Build

Run Actions Edit Configuration Settings

#5 (05 Feb 22 18:15)

OverviewChangesTestsBuild LogParametersArtifactsMaven Build Info

4 | All history | Last recorded build

Tree view | Tail

Download full build log (~30.63 KB) | .zip

Repeat block names

View: All messages Console view

18:14:55 The build is removed from the queue to be prepared for the start

18:14:55 Collecting changes in 1 VCS root

18:14:56 Starting the build on the agent "Default Agent"

18:15:01 Updating tools for build

18:15:01 Clearing temporary directory: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/temp/buildTemp

18:15:01 Using vcs information from agent file: 4622d330a39c0740.xml

18:15:01 Checkout directory: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/work/4622d330a39c0740

18:15:01 Updating sources: auto checkout (on agent) (1s)

18:15:02 Step 1/2: Check Maven version (Command Line)

18:15:02 Step 1/2 Starting: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/temp/agentTemp/custom\_script17728358502613539266

18:15:02 Step 1/2 in directory: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/work/4622d330a39c0740

18:15:02 Apache Maven 3.6.3 [m

18:15:02 Step 1/2 Maven home: /usr/share/maven

18:15:02 Step 1/2 Java version: 11.0.13, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64

18:15:02 Step 1/2 Default locale: es ES, platform encoding: UTF-8

18:15:02 Step 1/2 OS name: "linux", version: "5.13.0-28-generic", arch: "amd64", family: "unix"

18:15:02 Step 1/2 Process exited with code 0

18:15:02 Step 2/2: Maven (6s)

18:15:02 Step 2/2 Initial M2\_HOME not set

18:15:02 Step 2/2 Current M2\_HOME = /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/tools/maven3.6

18:15:02 Step 2/2 PATH = /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/tools/maven3.6/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

18:15:02 Step 2/2 Using watcher: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/plugins/mavenPlugin/maven-watcher-jdk16/maven-watcher-agent.jar

18:15:02 Step 2/2 Using agent local repository at /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/system/jetbrains-maven-runner/maven.repo.local

18:15:02 Step 2/2 \*\*\* Start reading the project structure \*\*\*

18:15:04 Step 2/2 Initial MAVEN\_OPTS not set

18:15:04 Step 2/2 Current MAVEN\_OPTS not set

18:15:04 Step 2/2 Starting: /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dagent.home.dir=/home/dew/Escritorio/TeamCity/TeamCity/buildAgent -Dagent.name=Default Agent -Dagent.ownPort=9090 -Dagent.work.dir=/home/dew/Escritorio

18:15:04 Step 2/2 in directory: /home/dew/Escritorio/TeamCity/TeamCity/buildAgent/work/4622d330a39c0740

18:15:06 Step 2/2 [INFO] Scanning for projects...

18:15:06 Step 2/2 [INFO] .....

18:15:06 Step 2/2 [INFO] Reactor Build Order:

18:15:06 Step 2/2 [INFO]

El resultado de la build habrá tenido éxito y podremos revisar paso por paso el resultado.

TeamCity permite además generar y guardar los archivos que puedan producir nuestras pipeline y builds, por ejemplo .war o .zip:

Build

General Settings

Version Control Settings 1

Build Steps 2

Triggers 1

Show more »

Last edited 6 minutes ago by daw (view history)

View as code

Name: \*

Build

Build configuration

SimpleMavenSample\_Build Regenerate ID

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

Description:

Publish artifacts: ?

Even if build fails Specify the artifacts publishing policy.

Artifact paths: ?

Newline- or comma-separated paths in the form of [+:]source [ => target] to exclude files or directories to publish as build artifacts, e.g. use \*\*/\* => target\_directory, -: \*\*/folder1 => target\_directory except for folder1 into the target\_directory.

Show advanced options

The latest finished build is SimpleMavenSample / Build #5

Select files to be published as artifacts from its checkout directory:

.git

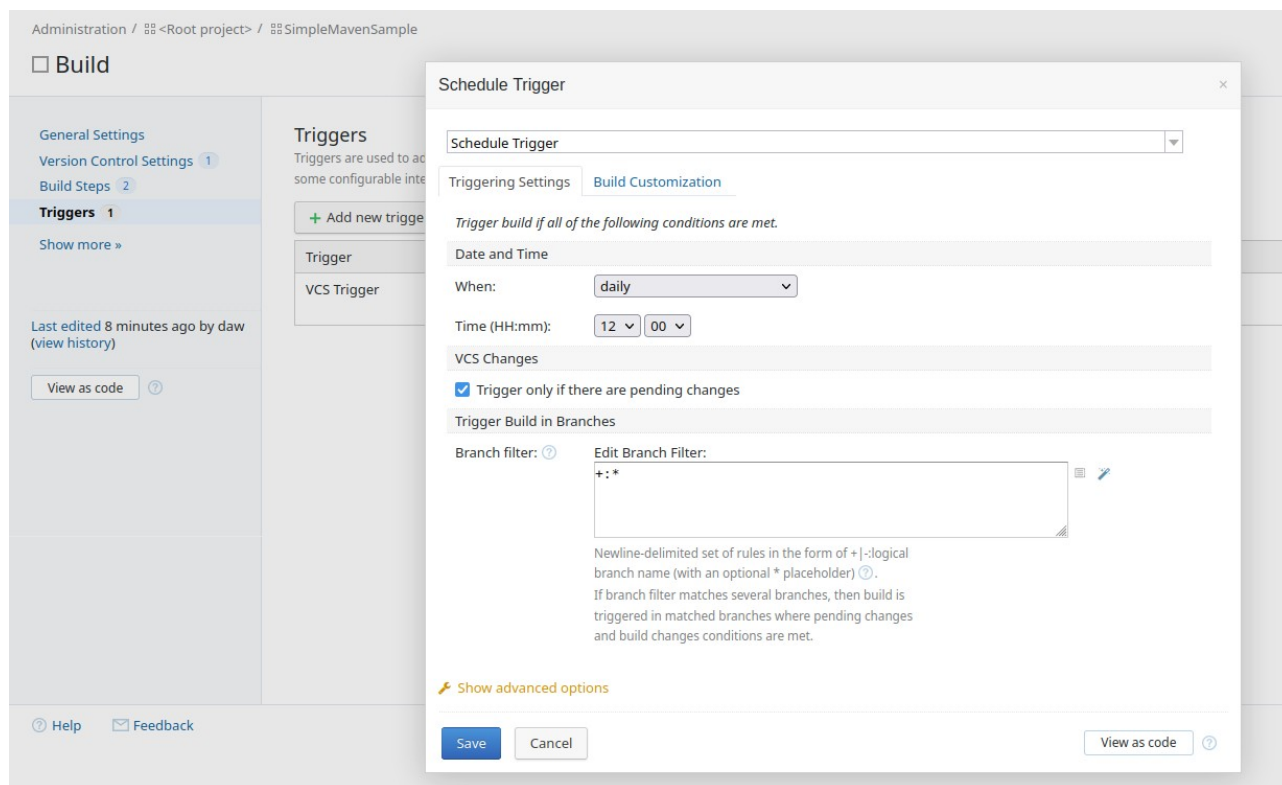
ch-simple

gitattributes (378 B)

gitignore (574 B)

En la página de build abajo a la derecha podremos publicar por ejemplo la carpeta target y TeamCity nos permitirá tener los .war publicados durante la ejecución.

Además en la sección triggers podremos programar ejecuciones con diferentes disparadores, por ejemplo en determinadas fechas o cuando ocurren determinadas acciones, a continuación ejemplo de una ejecución diaria:



Estas dos particularidades le proporcionan a TeamCity una gran potencia ya que al automatizar tanto la ejecución como la recogida de resultados (.war, .zip, logs) y poder acceder a ellos desde la propia plataforma ahorrará al usuario gran cantidad de tiempo y trabajo.

## Conclusión:

TeamCity es una herramienta para la integración continua con una gran potencia, respaldada por una empresa líder en el mercado. Esto hace que sea una herramienta ideal, ya que recibirá soporte durante años, además de estar fuertemente integrada con muchas herramientas de uso común hoy en día (Git, Docker, IDEs). A pesar de sus ventajas y potencia es una herramienta compleja, que requiere tiempo para adquirir soltura y en el caso de empresas grandes tiene un coste asociado a su uso.