Gradle

The Gradle Build Runner runs Gradle projects.

> ⚠️ To run builds with Gradle, you need to have Gradle 0.9-rc-1 or higher installed on all the agent machines. Alternatively, if you use the Gradle wrapper, you need to have properly configured Gradle Wrapper scripts checked in to your Version Control.

In this section:

- Gradle Parameters
- Run Parameters
- Java Parameters
- Build properties
- Docker Settings
- Code Coverage

Gradle Parameters

Option	Description
Gradle tasks	Specify Gradle task names separated by spaces. For example: `:myproject:clean :myproject:build` or `clean build`. If this field is left blank, the 'default' task is used. Note that TeamCity currently supports building Java projects with Gradle. Building Groovy/Scala/etc. projects has not been tested.
Incremental building	TeamCity can make use of the Gradle `:buildDependents` feature. If the Incremental building checkbox is enabled, TeamCity will detect Gradle modules affected by changes in the build, and start the `:buildDependents` command for them only. This will cause Gradle to fully build and test only the modules affected by changes.
Gradle home path	Specify here the path to the Gradle home directory (the parent of the `bin` directory). If not specified, TeamCity will use the Gradle from an agent's `GRADLE_HOME` environment variable. If you don't have Gradle installed on agents, you can use Gradle wrapper instead.
Additional Gradle command line parameters	Optionally, specify the space-separated list of command line parameters to be passed to Gradle.
Gradle Wrapper	If this checkbox is selected, TeamCity will look for Gradle Wrapper scripts in the checkout directory, and launch the appropriate script with Gradle tasks and additional command line parameters specified in the fields above. In this case, the Gradle specified in Gradle home path and the one installed on agent, are ignored.

Run Parameters

Option	Description
Debug	Selecting the Log debug messages check box is equivalent to adding the `-d` Gradle command line parameter.
Stacktrace	Selecting the Print stacktrace check box is equivalent to adding the `-s` Gradle command line parameter.

Java Parameters

Option	Description
JDK	Select a JDK. This section details the available options. The default is JAVA_HOME environment variable or the agent's own Java.

JDK home path	The option is available when <Custom> is selected above. Use this field to specify the path to your custom JDK used to run the build. If the field is left blank, the path to JDK Home is read either from the JAVA_HOME environment variable on agent the computer, or from the env.JAVA_HOME property specified in the build agent configuration file (buildAgent.properties). If these values are not specified, TeamCity uses the Java home of the build agent process itself.
JVM command line parameters	You can specify such JVM command line parameters, e.g. maximum heap size or parameters enabling remote debugging. These values are passed by the JVM used to run your build. Example: `-Xmx512m -Xms256m`

Build properties

The TeamCity system parameters can be accessed in Gradle build scripts in the same way as Gradle properties. The recommended way to reference properties is as follows:

```
task printProperty << {
    println "${project.ext['teamcity.build.id']}"
}
```

or if the system property's name is a legal Groovy name identifier (e.g. `system.myPropertyName = myPropertyValue`):

```
task printProperty << {
    println "$myPropertyName"
}
```

Docker Settings

In this section, you can specify a Docker image which will be used to run the build step.

Setting	Description
Run step within Docker container	Specify a Docker image here. TeamCity will start a container from the specified image and will try to run this build step within this container.
Pull image explicitly (since TeamCity 2017.2)	If the checkbox is enabled, `docker pull <imageName>` will be run before the `docker run` command.
Additional docker run arguments	The Edit arguments field allows specifying additional options for `docker run`. The default argument is `--rm`.

Technically, the command of the build runner is wrapped in a shell script, and this script is executed inside a Docker container with the `docker run` command. All the details about the started process, text of the script etc. are written into the build log (the Verbose mode enables viewing them).

The checkout directory and most build agent directories are mapped inside the Docker process, and TeamCity passes most environment variables from the build agent into the docker process.

After the build step with the Docker wrapper, a build agent will run the **chown** command to restore access of the buildAgent user to the checkout directory. This mitigates a possible problem when the files from a Docker container are created with the 'root' ownership and cannot be removed by the build agent later.

If the process environment contains the TEAMCITY_DOCKER_NETWORK variable, this network is passed to the started `docker run` command with `--network` switch.

It is possible to provide extra parameters for the `docker run` command, for instance, provide an additional volume mapping.

Code Coverage

Code coverage with IDEA code coverage engine and JaCoCo is supported.

See also:

Administrator's Guide: IntelliJ IDEA Code Coverage