Gradle JVM build tool

CONCEPTS

Use ./gradlew (the wrapper) instead of a global gradle: it pins the version the project was built with.

Tasks run with their dependencies, and up-to-date tasks are skipped. Use --rerun-tasks to force everything.

RUN TASKS

`./gradlew <task>`
Run a task and its dependencies

`./gradlew <task1> <task2>`
Run several tasks in order

`./gradlew build`
Assemble and run all checks

`./gradlew check`
Run all verification tasks

`./gradlew run`
Assemble and run the application

`./gradlew clean`
Delete the build directory contents

`./gradlew clean build`
Rebuild from scratch

`./gradlew :<sub>:<task>`
Run a task in one subproject

`./gradlew <task> -x <name>`
Skip a task, even if others need it

`./gradlew <task> --continue`
Keep going after a failure

`./gradlew <task> -t`
Re-run when inputs change

`./gradlew <task> --rerun-tasks`
Ignore up-to-date checks

TESTING

`./gradlew test`
Run all tests

`./gradlew test --tests <Class>`
Run one test class

TESTING (cont.)

`./gradlew test --tests <Class>.<method>`
Run one test method

`./gradlew test -i`
Show test output in the console

OPTIONS

`-i / --info`
Log at info level

`-s / --stacktrace`
Print stack traces for user errors

`--scan`
Create a Build Scan

`--offline`
Do not use the network

`-U / --refresh-dependencies`
Refresh dependency state

`--parallel`
Build projects in parallel

INSPECT

`./gradlew tasks`
List main tasks

`./gradlew dependencies`
Dependency tree per configuration

`./gradlew dependencyInsight --dependency <name> --configuration <cfg>`
Explain why a dependency is there

SETUP AND WRAPPER

`gradle init`
Create a new build interactively

`gradle wrapper`
Generate gradlew and wrapper files

`gradle wrapper --gradle-version <ver>`
Pin or upgrade the wrapper version

`./gradlew --version`
Show the Gradle version in use