Gradle JVM build tool

START HERE

`./gradlew build`
Compile, test, assemble

`./gradlew build --scan`
Publish a Build Scan

`./gradlew test`
Run the tests

`./gradlew --stop`
Stop the daemons

`./gradlew run`
Run the application

`./gradlew clean`
Delete build output

`./gradlew tasks`
List available tasks

`./gradlew dependencies`
Show dependency tree

CONCEPTS

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

Build settings.gradle(.kts)
Project build.gradle(.kts)
Task build, test, ...

Build structure

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

Prefix a task with : for the root project, or use sub:task to target a subproject.

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

RUN TASKS (cont.)

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

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

`./gradlew <task> -m`
Dry run: show what would execute

`./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

`./gradlew <task> --rerun`
Force only that task to re-run

TESTING

`./gradlew test`
Run all tests

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

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

`./gradlew test --tests '<pkg>*'`
Run a package recursively
Starts with a capital letter = simple class name; otherwise fully qualified

TESTING (cont.)

`./gradlew test --tests <a> --tests <b>`
Combine several filters

`./gradlew test --test-dry-run`
Show which tests would be selected

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

Test results land in build/test-results/<task> as JUnit XML, one file per class.

OPTIONS

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

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

`-S / --full-stacktrace`
Print the full stack trace

`--scan`
Create a Build Scan

`--offline`
Do not use the network

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

`--parallel`
Build projects in parallel
Off by default

`--build-cache`
Enable the build cache
Off by default; --no-build-cache to disable

OPTIONS (cont.)

`--max-workers <n>`
Limit worker count

`--console=plain`
Plain output (no rich console)
Modes: auto, plain, colored, rich, verbose

`-p <dir>`
Use another project directory

`-D<prop>=<value>`
Set a JVM system property

`-P<prop>=<value>`
Set a project property

`-I <script>`
Add an init script

DAEMON

`./gradlew --status`
List running daemons

`./gradlew --stop`
Stop all daemons

`--no-daemon`
Run without the daemon

`--foreground`
Start the daemon in the foreground

The daemon stays alive between builds so later runs start faster. Stop it if memory or stale state is a problem.

INSPECT

`./gradlew tasks`
List main tasks

INSPECT (cont.)

`./gradlew tasks --all`
Include tasks of all groups

`./gradlew tasks --group <name>`
List one task group

`./gradlew help --task <name>`
Show usage of one task

`./gradlew projects`
List subprojects as a tree

`./gradlew dependencies`
Dependency tree per configuration

`./gradlew dependencies --configuration <name>`
Tree for one configuration
e.g. runtimeClasspath

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

`./gradlew buildEnvironment`
Show buildscript dependencies

`./gradlew -q <task>`
Quiet output

SETUP AND WRAPPER

`gradle init`
Create a new build interactively

`gradle init --type <type>`
Pick a project type
e.g. java-library

SETUP AND WRAPPER (cont.)

`gradle wrapper`
Generate gradlew and wrapper files

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

`gradle wrapper --distribution-type all`
Use the all distribution (sources, docs)

`gradle wrapper --gradle-distribution-url <url>`
Download from your own mirror

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

RECIPES

Rebuild and debug a failing build

```
./gradlew clean build --stacktrace --info
```

Why is this version chosen?

```
./gradlew dependencyInsight \
  --dependency guava \
  --configuration runtimeClasspath
```

Fast loop on one test

```
./gradlew test --tests UserServiceTest -t -i
```

Upgrade the wrapper

```
./gradlew wrapper --gradle-version 9
./gradlew --version
```