sbt Scala build tool

START HERE

sbt
Open the interactive shell

reload
Reload build.sbt

compile
Compile main sources

console
Scala REPL on classpath

test
Run all tests

run
Run the main class

~test
Re-run on file change

projects
List subprojects

CONCEPTS

Start sbt once and stay in its shell: later commands skip the JVM startup and reuse compiled state.

build.sbt settings and tasks
↓ reload
sbt shell keeps state warm
↓ compile / test
target/ classes and reports

Tasks take a scope prefix: Test/compile compiles test sources, subproj/test runs one subproject.

RUN AND BUILD

sbt
Start the interactive shell

sbt *<command>*
Run one command and exit (batch mode)
Quote several commands: sbt "clean" "test"

compile
Compile main sources

run *<args>*
Run the main class in sbt's JVM

runMain *<class>* *<args>*
Run a specific main class

clean
Delete target directories

Test/compile
Compile test sources

package
Build the binary jar

RUN AND BUILD (cont.)

doc
Generate Scaladoc

update
Resolve dependencies

exit
Leave the shell (or Ctrl+D)

help *<command>*
Show help for a command

TESTING

test
Run all tests

testOnly *<Suite>*
Run named suites, * works

testOnly *<Suite>* **-- -z** "*<text>*"
Run tests matching a name
Everything after -- goes to the test framework; flags differ per framework

testQuick
Run new, failed or affected tests

Test/console
REPL with test classpath

IntegrationTest/test
Run the IntegrationTest config

WATCH MODE

~compile
Recompile on source change

~test
Re-run tests on change

~testQuick
Re-run only affected tests

WATCH MODE (cont.)

~run
Restart the app on change

Enter
Stop a ~ watch and return to shell

PROJECTS

projects
List all subprojects

project *<id>*
Switch the current project

*<id>***/test**
Run a task in one subproject

*<id>***/compile**
Compile one subproject

project /
Switch back to the root project

clean test
Run tasks one after another

; *<a>* **;** *<b>*
Run b only if a succeeds

BUILD DEFINITION

reload
Reload build files after editing

reload plugins
Switch to the build-definition project

reload return
Return to the main build

set *<key>* **:=** *<value>*
Change a setting until reload

BUILD DEFINITION (cont.)

session save
Write set changes to build.sbt

show *<key>*
Print a setting or task value

inspect *<key>*
Show scope, value and dependencies

inspect tree *<key>*
Show the dependency tree of a key

tasks
List available tasks

settings
List available settings

eval *<expr>*
Evaluate a Scala expression

consoleProject
REPL on the build definition

CROSS BUILDING AND REPL

console
Scala REPL with project classpath

consoleQuick
REPL without compiling the project

+test
Run for every crossScalaVersions entry

++*<version>* **test**
Switch Scala version for one command

CROSS BUILDING AND REPL (cont.)

+compile
Compile for all Scala versions

crossScalaVersions
Show the versions used by +

PUBLISH AND OPTIONS

publishLocal
Publish to the local Ivy repository

publish
Publish to the publishTo repository

publishM2
Publish to the local Maven repo

dependencyTree
Print the dependency tree
Needs sbt 1.4+; in older builds add the plugin

sbt -J-Xmx2G
Pass a JVM option to sbt

sbt -Dsbt.offline=true
Do not fetch from repositories

sbt -Dsbt.version=*<v>*
Use another sbt version for a run

sbt -Dsbt.ci=true
No color or supershell for CI

sbt --client *<command>*
Send a command to a running server

sbt -Dsbt.task.timings=true
Print how long tasks take

BUILD.SBT BASICS

Minimal build.sbt

```
ThisBuild / scalaVersion := "3.3.4"

lazy val root = (project in file("."))
  .settings(name := "app")
```

Add a dependency

```
libraryDependencies += "org.scalatest" %% "scalatest" % "3.2.19" % Test
```

Operators

Operator	Meaning
`:=`	Set a value
`+=`	Append one element
`++=`	Append a collection
`%`	Group % artifact % version
`%%`	Add the Scala suffix to the artifact

project/build.properties

```
sbt.version=1.10.10
```

RECIPES

Fast TDD loop on one suite

```
sbt
~testOnly com.example.UserSpec
```

Multi-module build, one module

```
sbt "api/clean" "api/test"
```

Cross-build and publish locally

```
sbt +test +publishLocal
```

CI run without color

```
sbt -Dsbt.ci=true -J-Xmx2G
clean test
```