Bazel Fast, hermetic builds

Load read BUILD files → packages → **Analyze** build the action graph → actions → **Execute** run actions, use cache → outputs → **bazel-bin/** results

CONCEPTS

A label is //package:target. The package is a directory with a BUILD file; //pkg alone means //pkg:pkg.

Builds are hermetic and cached: a rerun with no input changes does nothing. --nobuild stops after analysis.

Bazel 9 uses Bzlmod only: dependencies are declared in MODULE.bazel, WORKSPACE is gone.

BUILD

bazel build *<targets>*
Build the given targets

bazel build //...
Build every target in the main repo

bazel build //*<pkg>*/...
Build a package and all below it

bazel build //*<pkg>*:all
Build all rule targets in one package

bazel build -- //... -//*<pkg>*/...
Build all except a subtree

BUILD (cont.)

bazel build -c *<mode>*
fastbuild, dbg or opt

bazel build -k
Keep going after errors

bazel build --config=*<name>*
Apply a named .bazelrc config

TEST

bazel test *<targets>*
Build and run tests

bazel test --test_output=errors
Print output of failed tests only

bazel test --test_output=all
Print all test output

bazel test --test_filter=*<expr>*
Select tests inside a target

bazel test --cache_test_results=no
Force tests to rerun

RUN

bazel run *<target>*
Build then run a target

bazel run *<target>* **--** *<args>*
Pass args to the program

QUERY

bazel query "deps(*<t>***)"**
All dependencies of a target

bazel query "rdeps(//..., *<t>*)"
Who depends on this target

bazel query "somepath(*<a>***, *<b>*)"**
One dependency path from a to b

bazel query "kind(*<regex>***, //...)"**
Targets by rule kind

bazel query //*<pkg>*/... --output=label_kind
List targets with their rule

bazel cquery *<expr>*
Query with configurations applied

DEPENDENCIES (BZLMOD)

bazel mod graph
Show the module dependency graph

bazel mod explain *<module>*
Why is this module here?

bazel mod tidy
Format MODULE.bazel, fix use_repo

bazel fetch //...
Pre-fetch external repos

SERVER AND CLEANUP

bazel clean
Delete build outputs

bazel clean --expunge
Delete the whole output base

bazel info *<key>*
One value (bazel-bin, output_base)

bazel version
Print Bazel version

bazel help *<command>*
Help for one command

DEBUGGING BUILDS

bazel build --verbose_failures
Print the failing command in full

bazel build -s
Print every command as it runs

.BAZELRC

build --*<flag>*
Default flag for build and its children

common --*<flag>*
Flag for every command that supports it

.BAZELRC (cont.)

build:*<name>* **--***<flag>*
Flag group used by --config=<name>

try-import %workspace%/user.bazelrc
Optional personal overrides

Command-line flags beat rc files; more specific commands (test) beat build, which beats common.

RECIPES

Test one target and see why it fails

```
bazel test //<pkg>:<test> \
  --test_output=errors --
test_filter=<name> \
  --cache_test_results=no --
verbose_failures
```

Why does A depend on B?

```
bazel query "somepath(//
<pkg>:<a>, //<pkg>:<b>)"
bazel query "rdeps(//..., //
<pkg>:<b>)" --
output=label_kind
```

Release and debug configs

```
# .bazelrc
build --keep_going
build:release -c opt
build:debug -c dbg --
strip=never
# bazel build --
config=release //...
```