Bats Bash test runner

setup_suite once per run → then **setup_file** once per file → then **setup** each test → @test **teardown** each test → then **teardown_file** once per file

CONCEPTS

A test passes if every command in it succeeds (Bats runs tests with set -e); run never fails, so you assert on $status and $output afterwards.

Each .bats file is evaluated once per test plus once more, so keep top-level code free of output and side effects.

RUNNING

bats *<path>*...
Run files or directories of .bats files

bats -r *<dir>*
Recurse into subdirectories

bats -f *<regex>* *<path>*
Run only tests whose name matches

bats --negative-filter *<regex>* *<path>*
Skip tests whose name matches

bats -c *<path>*
Count tests without running them

bats --abort *<path>*
Stop at the first failing test

RUNNING (cont.)

bats -j *<n>* *<path>*
Run tests in parallel (needs GNU parallel)

bats -x *<path>*
Trace commands like set -x

bats --print-output-on-failure *<path>*
Show $output of failed tests

bats -T *<path>*
Add timing to each result

bats -v / bats -h
Print version / help

OUTPUT

bats -p *<path>*
Pretty output (default on a terminal)

bats -t *<path>*
TAP output (default when piped)

bats -F junit *<path>*
JUnit XML for CI

bats --report-formatter junit -o *<dir>* *<path>*
Write a report file to <dir>

Exit code is 0 when all tests pass, 1 on any failure or when no tests are found.

WRITING TESTS

@test "*<name>***"**
{ *<commands>* **}**
Define a test

load *<file>*
Source a helper relative to the test file

skip ["*<reason>***"]**
Skip this test

bats_require_minimum_version *<ver>*
Fail on a too-old Bats: pass <ver>

bats_load_library *<name>*
Load a library from BATS_LIB_PATH

echo "# *<msg>***" >&3**
Print a message during a test

 # bats test_tags=*<a>***,***<b>*
Tag the next test

RUN AND ASSERT

run *<cmd>* **[args]**
Capture status and output; never fails

run -*<N>* *<cmd>*
Fail unless exit status is N

run ! *<cmd>*
Fail unless exit status is nonzero

RUN AND ASSERT (cont.)

run --separate-stderr *<cmd>*
Split stdout and stderr

["$status" -eq 0]
Assert exit status

["$output" = "*<text>***"]**
Assert combined output

["${lines[0]}" = "*<text>***"]**
Assert one output line

[["$output" == *"*<part>***"*]]**
Assert output contains text

$stderr / ${stderr_lines[@]}
Stderr (with --separate-stderr)

run uses a subshell, so variables set by the command are gone afterwards. Call the function directly if you need them.

SETUP AND TEARDOWN

setup() { ... }
Before each test

teardown() { ... }
After each test, even on failure

SETUP AND TEARDOWN (cont.)

setup_file() { ... }
Once before all tests in a file

teardown_file() { ... }
Once after all tests in a file

$BATS_TEST_TMPDIR
Fresh temp dir for each test

$BATS_FILE_TMPDIR
Temp dir shared within a file

$BATS_TEST_DIRNAME
Directory of the test file

HELPER LIBRARIES

bats-support and bats-assert are separate repos, not part of Bats itself. Check the version your project installs before using them.

load 'test_helper/bats-support/load'
Load bats-support first

load 'test_helper/bats-assert/load'
Then bats-assert

assert_success / assert_failure
Check exit status after run

assert_output "*<text>***"**
Check output (--partial, --regexp)