Bats Bash test runner

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.

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

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

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

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

bats -v / bats -h
Print version / help

OUTPUT

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

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

WRITING TESTS

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

WRITING TESTS (cont.)

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

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

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

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

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

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

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

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

$BATS_TEST_TMPDIR
Fresh temp dir for each test

$BATS_TEST_DIRNAME
Directory of the test file