HashiCorp
(https://www.hashicorp.com)

State Command

JUMP TO SECTION

The `terraform state` command is used for advanced state management. As your Terraform usage becomes more advanced, there are some cases where you may need to modify the Terraform state (/docs/language/state/index.html). Rather than modify the state directly, the `terraform state` commands can be used in many cases instead.

This command is a nested subcommand, meaning that it has further subcommands. These subcommands are listed to the left.

Usage

Usage: `terraform state <subcommand> [options] [args]`

Please click a subcommand to the left for more information.

Remote State

The Terraform state subcommands all work with remote state just as if it was local state. Reads and writes may take longer than normal as each read and each write do a full network roundtrip. Otherwise, backups are still written to disk and the CLI usage is the same as if it were local state.

Backups

All `terraform state` subcommands that modify the state write backup files. The path of these backup file can be controlled with `-backup`.

Subcommands that are read-only (such as list (/docs/cli/commands/state/list.html)) do not write any backup files since they aren't modifying the state.

Note that backups for state modification *can not be disabled*. Due to the sensitivity of the state file, Terraform forces every state modification command to write a backup file. You'll have to remove these files manually if you don't want to keep them around.

Command-Line Friendly

The output and command-line structure of the state subcommands is designed to be usable with Unix command-line tools such as grep, awk, and similar PowerShell commands.

For advanced filtering and modification, we recommend piping Terraform state subcommands together with other command line tools.