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

Command: taint

The `terraform taint` command informs Terraform that a particular object has become degraded or damaged. Terraform represents this by marking the object as "tainted" in the Terraform state, in which case Terraform will propose to replace it in the next plan you create.

> *Warning:* This command is deprecated, because there are better alternatives available in Terraform v0.15.2 and later. See below for more details.

If your intent is to force replacement of a particular object even though there are no configuration changes that would require it, we recommend instead to use the `-replace` option with `terraform apply` (/docs/cli/commands/apply.html). For example:

```
terraform apply -replace="aws_instance.example[0]"
```

Creating a plan with the "replace" option is superior to using `terraform taint` because it will allow you to see the full effect of that change before you take any externally-visible action. When you use `terraform taint` to get a similar effect, you risk someone else on your team creating a new plan against your tainted object before you've had a chance to review the consequences of that change yourself.

The `-replace=...` option to `terraform apply` is only available from Terraform v0.15.2 onwards, so if you are using an earlier version you will need to use `terraform taint` to force object replacement, while considering the caveats described above.

Usage

Usage: `terraform taint [options] address`

The `address` argument is the address of the resource to mark as tainted. The address is in the resource address syntax (/docs/cli/state/resource-addressing.html) syntax, as shown in the output from other commands, such as:

- `aws_instance.foo`
- `aws_instance.bar[1]`
- `aws_instance.baz[` `aws_instance.baz[\"key\"]` `quot;key` `aws_instance.baz[\"key\"]` `quot;]` (quotes in resource addresses must be escaped on the command line, so that they will not be interpreted by your shell)
- `module.foo.module.bar.aws_instance.qux`

This command accepts the following options:

- `-allow-missing` - If specified, the command will succeed (exit code 0) even if the resource is missing. The command might still return an error for other situations, such as if there is a problem reading or writing the state.
- `-lock=false` - Disables Terraform's default behavior of attempting to take a read/write lock on the state for the duration of the operation.
- `-lock-timeout=DURATION` - Unless locking is disabled with `-lock=false`, instructs Terraform to retry acquiring a lock for a period of time before returning an error. The duration syntax is a number followed by a time unit letter, such as "3s" for three seconds.

For configurations using the `remote` backend (/docs/language/settings/backends/remote.html) only, `terraform taint` also accepts the option `-ignore-remote-version` (/docs/language/settings/backends/remote.html#command-line-arguments).

For configurations using the `local` backend (/docs/language/settings/backends/local.html) only, `terraform taint` also accepts the legacy options `-state`, `-state-out`, and `-backup` (/docs/language/settings/backends/local.html#command-line-arguments).