Packer Machine image builder

START HERE

`packer init .`
Install plugins

`packer build -var-file=<file> .`
Build with values

`packer fmt .`
Format templates

`packer build -only=<name> .`
Build one source

`packer validate .`
Check template

`packer inspect .`
Show template layout

`packer build .`
Build the image

`PACKER_LOG=1 packer build .`
Debug logging

Template *.pkr.hcl → `packer init` → **Plugins** builders, provisioners → `packer build` → **Temp machine** source + provisioners → `post-processors` → **Image** AMI, box, template

CONCEPTS

A template is HCL files: a source says what to boot, a build runs provisioners on it and saves an image.

Point commands at a directory (.) to load every *.pkr.hcl in it, not just one file.

Packer builds a throwaway machine, then deletes it; only the resulting image remains.

INIT & PLUGINS

`packer init <dir>`
Install plugins the template requires

`packer init -upgrade <dir>`
Upgrade plugins within constraints

`packer plugins installed`
List installed plugin binaries

`packer plugins required <dir>`
List plugins a template needs

`packer plugins install <source> [version]`
Install a plugin by hand

`packer plugins remove <source> [version]`
Remove an installed plugin

`packer init -force <dir>`
Reinstall plugins

CHECK & FORMAT

`packer validate <dir>`
Check syntax and configuration

`packer validate -syntax-only <dir>`
Check syntax only

`packer validate -evaluate-datasources <dir>`
Also run data sources

Data sources may call paid external services.

`packer fmt <dir>`
Rewrite files in canonical style

CHECK & FORMAT (cont.)

`packer fmt -check <dir>`
Fail if unformatted (CI)

`packer fmt -diff <dir>`
Show formatting changes

`packer fmt -recursive <dir>`
Include subdirectories

`packer inspect <dir>`
List variables, sources, provisioners

BUILD

`packer build <dir>`
Build all sources

`packer build <file.pkr.hcl>`
Build from a single file

`packer build -only=<a>,<b> <dir>`
Run only the named builds

`packer build -except=<a>,<b> <dir>`
Skip the named builds

`packer build -force <dir>`
Overwrite earlier artifacts

`packer build -parallel-builds=<n> <dir>`
Limit concurrent builds

`packer build -timestamp-ui <dir>`
Timestamp each output line

`packer build -color=false <dir>`
Plain output for logs

`packer build -machine-readable <dir>`
Comma-delimited output for scripts

VARIABLES

`packer build -var <key>=<value> <dir>`
Set one variable

`packer build -var-file=<file> <dir>`
Load variables from a file

`PKR_VAR_<name>=<value>`
Set a variable via environment

VARIABLES (cont.)

`*.auto.pkrvars.hcl`
Variable file loaded automatically

`packer validate -var-file=<file> <dir>`
Validate with the same values

`-no-warn-undeclared-var`
Silence unused var-file warnings

Later wins: environment, then auto files, then -var and -var-file in the order given.

Mark secrets with sensitive = true so Packer masks them in its output.

DEBUG & ERRORS

`packer build -on-error=ask <dir>`
Prompt to clean up, abort or retry

`packer build -on-error=abort <dir>`
Keep the failed machine

May need -force on the next build.

`packer build -on-error=cleanup <dir>`
Delete machine on failure (default)

`packer build -on-error=run-cleanup-provisioner <dir>`
Keep machine, run error cleanup

`packer build -debug <dir>`
Step by step, pauses between steps

`PACKER_LOG=1`
Verbose logs on stderr

`PACKER_LOG_PATH=<file>`
Write the log to a file

`packer console <dir>`
Evaluate HCL expressions interactively

`TMPDIR=<short path>`
Fix 'plugin exited before we could connect'

TEMPLATE SKELETON

Plugins and a variable

```
packer {
  required_plugins {
    docker = {
      version = "⩾ 1.0.0"
      source  = "github.com/
hashicorp/docker"
    }
  }
}

variable "tag" {
  type    = string
  default = "latest"
}
```

Source and build

```
source "docker" "app" {
  image  = "ubuntu:24.04"
  commit = true
}

build {
  sources =
["source.docker.app"]
  provisioner "shell" {
    inline = ["apt-get update"]
  }
}
```

`packer hcl2_upgrade <file.json>`
Convert a JSON template to HCL2 (beta)

`packer fix <file.json>`
Update an old JSON template

RECIPES

Pre-flight before CI build

```
packer init .
packer fmt -check -recursive .
packer validate -var-
file=prod.pkrvars.hcl .
packer build -var-
file=prod.pkrvars.hcl .
```

Debug a failing provisioner

```
# keeps the machine so you can
SSH in and retry
PACKER_LOG=1
PACKER_LOG_PATH=packer.log \
  packer build -on-error=ask .
```

Rebuild a single source

```
packer build -
only='app.docker.app' -force .
```

Try an expression

```
echo 'upper("web")' | packer
console .
```