Vagrant Reproducible dev VMs

START HERE

`vagrant init <box>`
Create a Vagrantfile

`vagrant up`
Create and start the VM

`vagrant ssh`
Log into the VM

`vagrant status`
State of this project

`vagrant halt`
Shut the VM down

`vagrant provision`
Re-run provisioners

`vagrant destroy -f`
Delete the VM

`vagrant global-status`
All VMs on this host

Box base image → `vagrant up` → **Provider** VirtualBox, libvirt → `provision` → **Guest VM** configured → `vagrant ssh` → **You** shell in the VM

CONCEPTS

The Vagrantfile in your project folder defines the VM; commit it, and anyone can recreate the same machine with vagrant up.

Your project folder is mounted in the VM as /vagrant, so you edit on the host and run inside the guest.

halt keeps the disk, destroy deletes it. Provisioners run on the first up only, unless you ask again.

LIFECYCLE

`vagrant up [<name>]`
Create and start the machine

`vagrant halt`
Shut down the machine gracefully

`vagrant destroy -f`
Delete the machine without prompting

`vagrant status`
Show the machine state of this project

`vagrant reload`
Restart and re-read the Vagrantfile

`vagrant suspend`
Save state to disk and pause

`vagrant resume`
Wake a suspended machine

`vagrant destroy -g`
Shut down gracefully, then delete

`vagrant up --provider <name>`
Pick a provider other than VirtualBox

`vagrant up --no-provision`
Start without running provisioners

`vagrant up --parallel`
Start several machines at once
Only works if the provider supports it.

`vagrant up --no-destroy-on-error`
Keep a half-built machine to debug it
By default a fatal error on the first up destroys the new machine.

ACCESS

`vagrant ssh [<name>]`
Open a shell in the machine

ACCESS (cont.)

`vagrant ssh -c '<cmd>'`
Run one command and exit

`vagrant ssh-config`
Print an OpenSSH config for the machine

`vagrant ssh -- <ssh args>`
Pass extra arguments to ssh
Handy for tunnels, for example -- -L 8080:localhost:80.

`vagrant port`
List forwarded ports

PROVISIONING

`vagrant provision`
Re-run provisioners on a running machine

`vagrant reload --provision`
Restart and run provisioners

`vagrant provision --provision-with <name>`
Run only the named provisioners

`vagrant up --provision`
Force provisioners on up

BOXES

`vagrant box list`
List installed boxes

`vagrant box add <name>`
Download a box

`vagrant box remove <name>`
Delete an installed box

`vagrant box update`
Fetch a newer version of the project box
A running machine keeps the old box; destroy and up again to use the new one.

`vagrant box outdated`
Check if the project box has an update

`vagrant box outdated --global`
Check every installed box

`vagrant box prune --dry-run`
Preview removal of old box versions

`vagrant box prune`
Remove old box versions

`vagrant box add <name> --box-version <ver>`
Add a specific version

`vagrant box add <file-or-url> --name <name>`
Add a box from a file or URL

BOXES (cont.)

`vagrant package`
Package the running machine as a box

SNAPSHOTS

`vagrant snapshot save <name>`
Save a named snapshot

`vagrant snapshot restore <name>`
Roll back to a named snapshot

`vagrant snapshot list`
List snapshots

`vagrant snapshot delete <name>`
Delete a snapshot

`vagrant snapshot push`
Push an unnamed snapshot on the stack

`vagrant snapshot pop`
Restore and drop the latest pushed one
Do not mix push/pop with save/restore.

Snapshot support depends on the provider, and deleting is faster on a halted machine.

PROJECT AND ENVIRONMENT

`vagrant init -m <box>`
Write a minimal Vagrantfile

`vagrant init <box> --box-version <ver>`
Pin the box version in the file

`vagrant validate`
Check the Vagrantfile for errors

`vagrant global-status`
List all known machines with their IDs

`vagrant global-status --prune`
Drop stale entries from the list
Slow; the list is cached and can show stale state.

`vagrant destroy <id>`
Control a machine by ID from any directory

`vagrant plugin list`
List installed plugins

`vagrant plugin install <name>`
Install a plugin

`vagrant version`
Show installed and latest version

`vagrant <cmd> -h`
Help for any subcommand

VAGRANTFILE

Minimal file

```
Vagrant.configure("2") do |config|
  config.vm.box = "<box>"
  config.vm.hostname = "dev"
end
```

Network and folders

```
# in the configure block
config.vm.network "forwarded_port", guest: 80, host: 8080
config.vm.network "private_network", ip: "192.168.56.10"
config.vm.synced_folder "./data", "/srv/data"
```

Provider resources

```
config.vm.provider "virtualbox" do |vb|
  vb.memory = 2048
  vb.cpus = 2
end
```

Shell provisioner

```
config.vm.provision "shell", inline: "apt-get update"
config.vm.provision "shell", path: "setup.sh"
```

Ansible provisioner

```
config.vm.provision "ansible" do |a|
  a.playbook = "site.yml"
end
```

Two machines

```
config.vm.define "web" do |web|
  web.vm.box = "<box>"
end
config.vm.define "db" do |db|
  db.vm.box = "<box>"
end
```

Changed the Vagrantfile? Run vagrant reload (or halt then up) so it takes effect.

RECIPES

New project in a minute

```
vagrant init -m <box>
vagrant up
vagrant ssh
```

Fresh start

```
vagrant destroy -f
vagrant up
```

Safe experiment

```
vagrant snapshot save clean
vagrant ssh -c 'sudo <risky-command>'
vagrant snapshot restore clean
```

Tidy up the host

```
vagrant global-status --prune
vagrant box outdated --global
vagrant box prune --dry-run
```