Vagrant Reproducible dev VMs

CONCEPTS

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

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

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

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

`vagrant up --no-destroy-on-error`
Keep a half-built machine to debug it

ACCESS

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

`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

`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

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Fetch a newer version of the project box

`vagrant box outdated`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

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

PROJECT AND ENVIRONMENT

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

`vagrant validate`
Check the Vagrantfile for errors

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

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

`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"
```

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
```