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

ACCESS

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

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

PROVISIONING

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

BOXES

`vagrant box list`
List installed boxes

`vagrant box add <name>`
Download a box

SNAPSHOTS

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

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

PROJECT AND ENVIRONMENT

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

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

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"
```

Shell provisioner

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