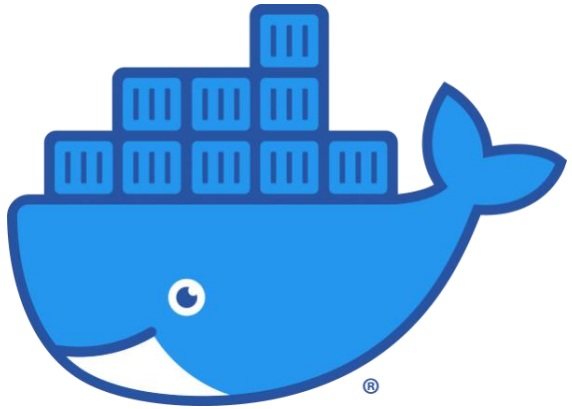


Vagrant pour DevOps

Par Dirane TAFEN

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Plan

- **Présentation de la formation**
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Présentation de la formation (1/2) : Dirane TAFEN

- Dirane TAFEN (formateur et consultant DevOps)
- Capgemini
- Sogeti
- ATOS
- BULL
- AIRBUS
- ENEDIS

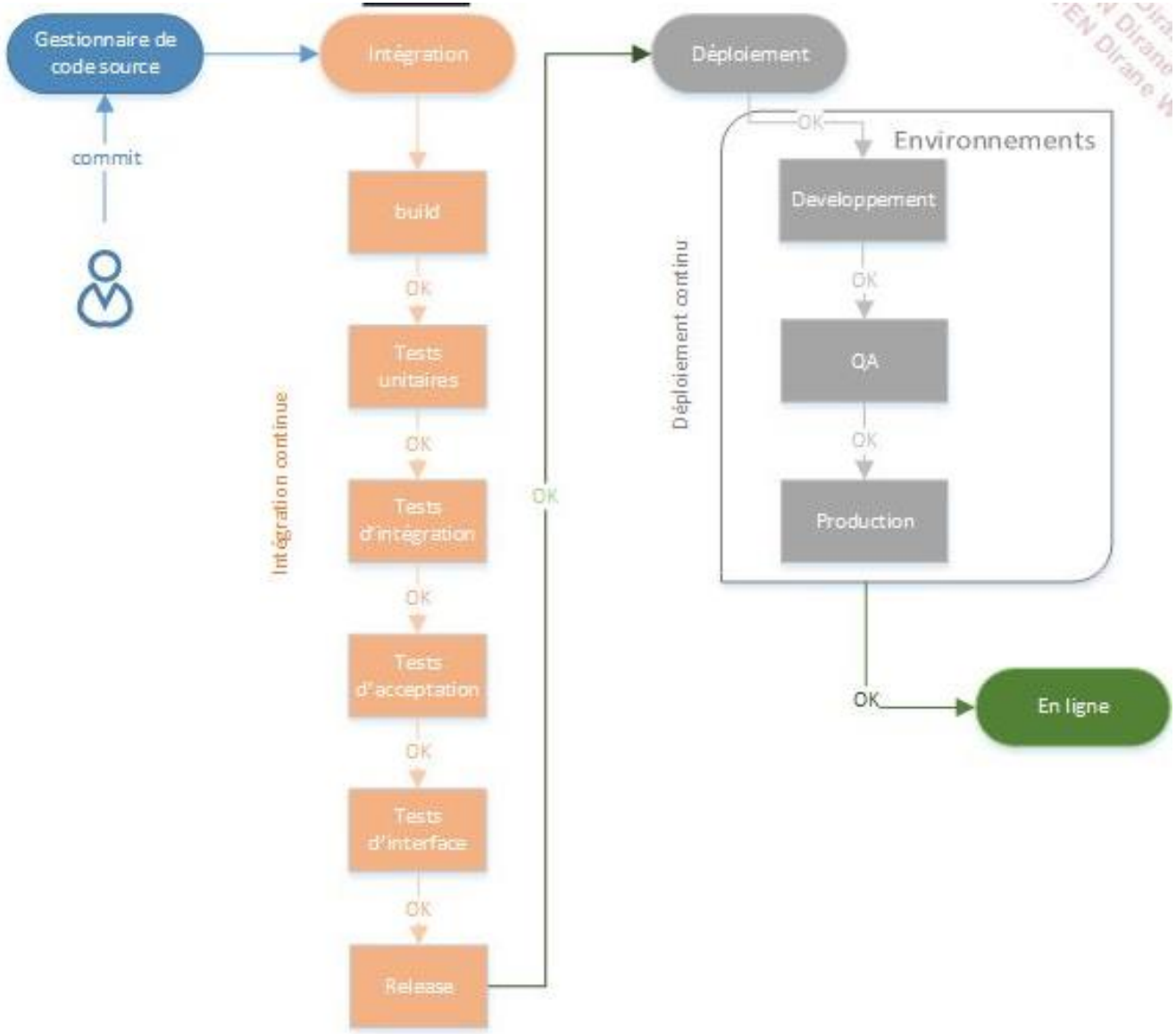
Présentation de la formation (2/2) : Prérequis

Plan

- Présentation de la formation
- **Pourquoi Vagrant ?**
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Pourquoi vagrant (1/3) : Rappel sur le DevOps

- Intégration en continu
- Test en continu
- Déploiement en continu
- Orchestration



Pourquoi Vagrant (2/3) : Vagrant

- Créé par Mitchell Hashimoto
- Ecris en ruby
- Release en Mars 2010
- Hashicorp Company
- Multi-plateforme : Microsoft; MacOS, FreeBSD et Linux
- Programmation du déploiement et de la gestion d'infrastructure virtualisée
- Compatible avec VMWare, Hyper-V, Docker, VirtualBox …

HashiCorp

Vagrant

Pourquoi vagrant (3/3) : Avantages

Development

Developer always look for change

Change

Operations

Change don't like by operation team

Instability

Lab-0 : Plateforme de TP

- Installer Virtuabox
- Installer Vagrant (un redémarage est nécessaire)
- Vérifier que l’installation s’est bien déroulée en utilisant la ligne de commande vagrant

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- **Commandes de Base**
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Commandes de base (1/3) : List-commands

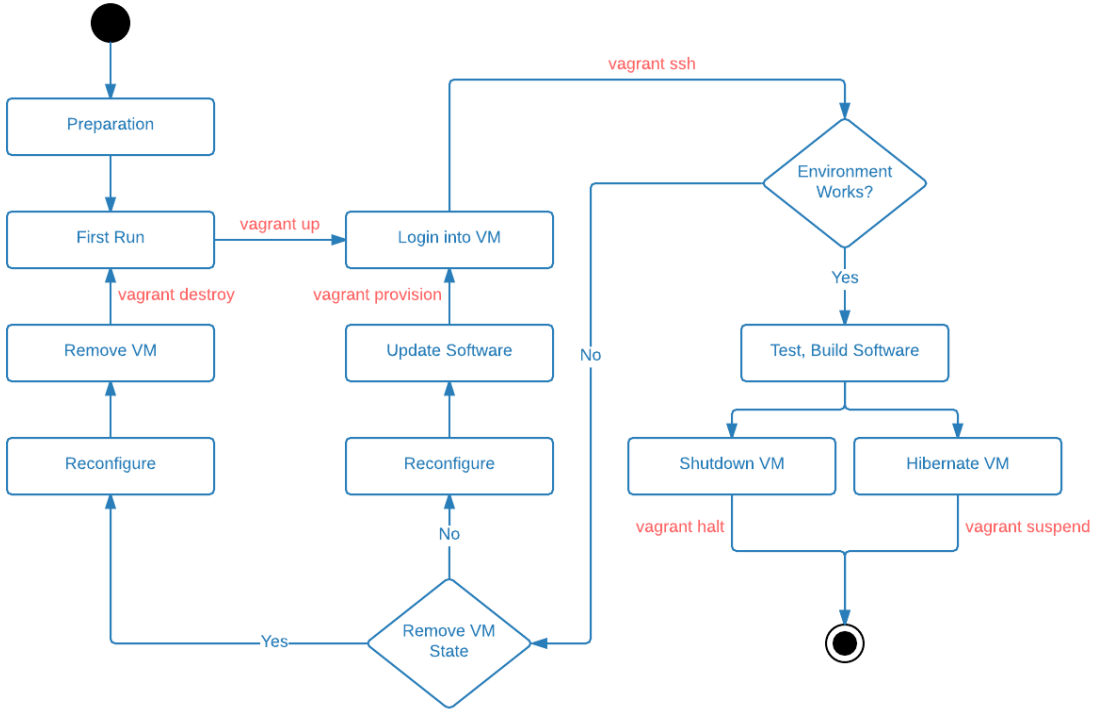
```
PS C:\WINDOWS\system32> vagrant list-commands
Below is a listing of all available Vagrant commands and a brief
description of what they do.

autocomplete    manages autocomplete installation on host
box             manages boxes: installation, removal, etc.
cap             checks and executes capability
cloud           manages everything related to Vagrant Cloud
destroy         stops and deletes all traces of the vagrant machine
docker-exec     attach to an already-running docker container
docker-logs     outputs the logs from the Docker container
docker-run      run a one-off command in the context of a container
global-status   outputs status Vagrant environments for this user
halt            stops the vagrant machine
help            shows the help for a subcommand
hostsupdater
init            initializes a new Vagrant environment by creating a Vagrantfile
list-commands   outputs all available Vagrant subcommands, even non-primary ones
login
package         packages a running vagrant environment into a box
plugin          manages plugins: install, uninstall, update, etc.
port            displays information about guest port mappings
powershell      connects to machine via powershell remoting
provider        show provider for this environment
provision       provisions the vagrant machine
push            deploys code in this environment to a configured destination
rdp             connects to machine via RDP
reload          restarts vagrant machine, loads new Vagrantfile configuration
resume          resume a suspended vagrant machine
rsync           syncs rsync synced folders to remote machine
rsync-auto      syncs rsync synced folders automatically when files change
snapshot        manages snapshots: saving, restoring, etc.
ssh             connects to machine via SSH
ssh-config      outputs OpenSSH valid configuration to connect to the machine
status          outputs status of the vagrant machine
suspend         suspends the machine
up              starts and provisions the vagrant environment
upload          upload to machine via communicator
validate        validates the Vagrantfile
version         prints current and latest Vagrant version
winrm           executes commands on a machine via WinRM
winrm-config    outputs WinRM configuration to connect to the machine
```

```
PS C:\WINDOWS\system32> vagrant global-status
id       name    provider   state  directory
```

```
PS C:\WINDOWS\system32> vagrant global-status -h
Usage: vagrant global-status

        --prune                     Prune invalid entries.
        --[no-]color                Enable or disable color output
        --machine-readable          Enable machine readable output
    -v, --version                   Display Vagrant version
        --debug                     Enable debug output
        --timestamp                 Enable timestamps on log output
        --debug-timestamp           Enable debug output with timestamps
        --no-tty                    Enable non-interactive output
    -h, --help                      Print this help
```

Commandes de Base (2/3) : global-status

Commandes de Base (3/3) : VM workflow



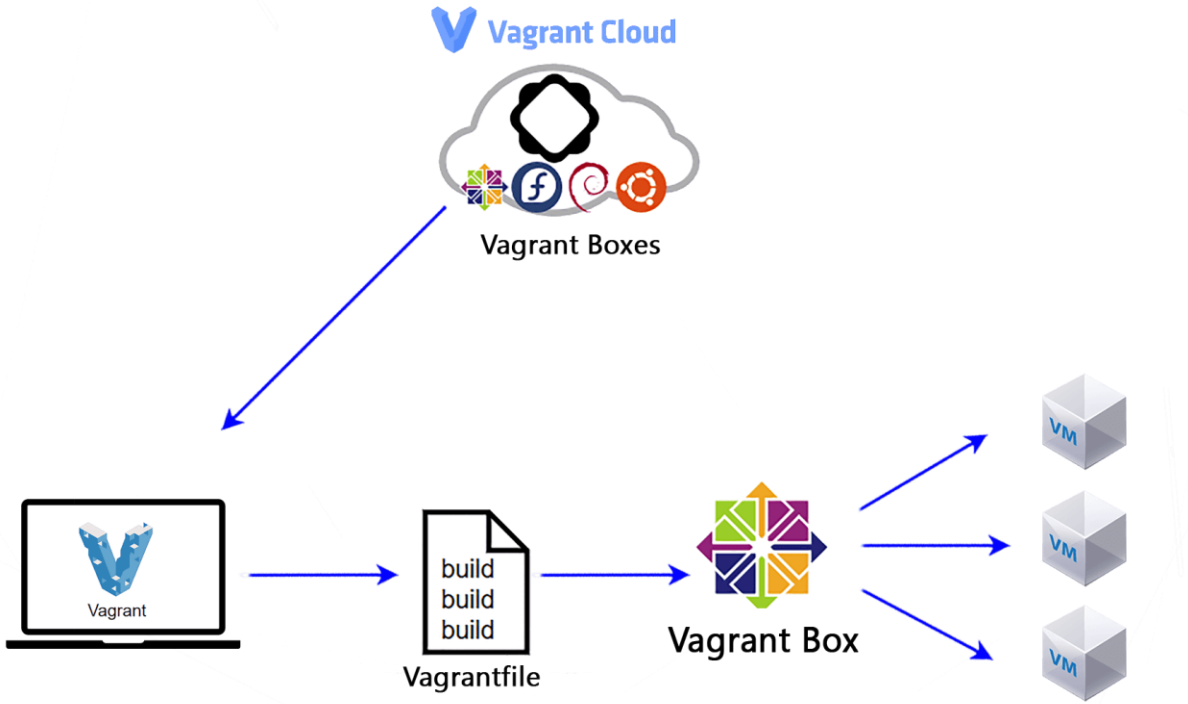
Lab-1 : Votre première VM ubuntu

- Créez un dossier lab-1
- Déplacez-vous dans ce dossier
- Initialisez vagrant (vagrant init ubuntu/trusty64)
- Démarrez la VM
- Connectez-vous en ssh
- Arretez la VM puis supprimez là

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- **Vagrant Boxes**
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Vagrant Boxes (1/5) : Principe



Vagrant Boxes (2/5) : Box Components

- Box File - This is a compressed (`tar` , `tar.gz` , `zip`) file that is specific to a single provider and can contain anything. Vagrant core does not ever use the contents of this file. Instead, they are passed to the provider. Therefore, a VirtualBox box file has different contents from a VMware box file and so on.

- Box Catalog Metadata - This is a JSON document (typically exchanged during interactions with HashiCorp's Vagrant Cloud) that specifies the name of the box, a description, available versions, available providers, and URLs to the actual box files (next component) for each provider and version. If this catalog metadata does not exist, a box file can still be added directly, but it will not support versioning and updating.

- Box Information - This is a JSON document that can provide additional information about the box that displays when a user runs `vagrant box list -i` . More information is provided here.

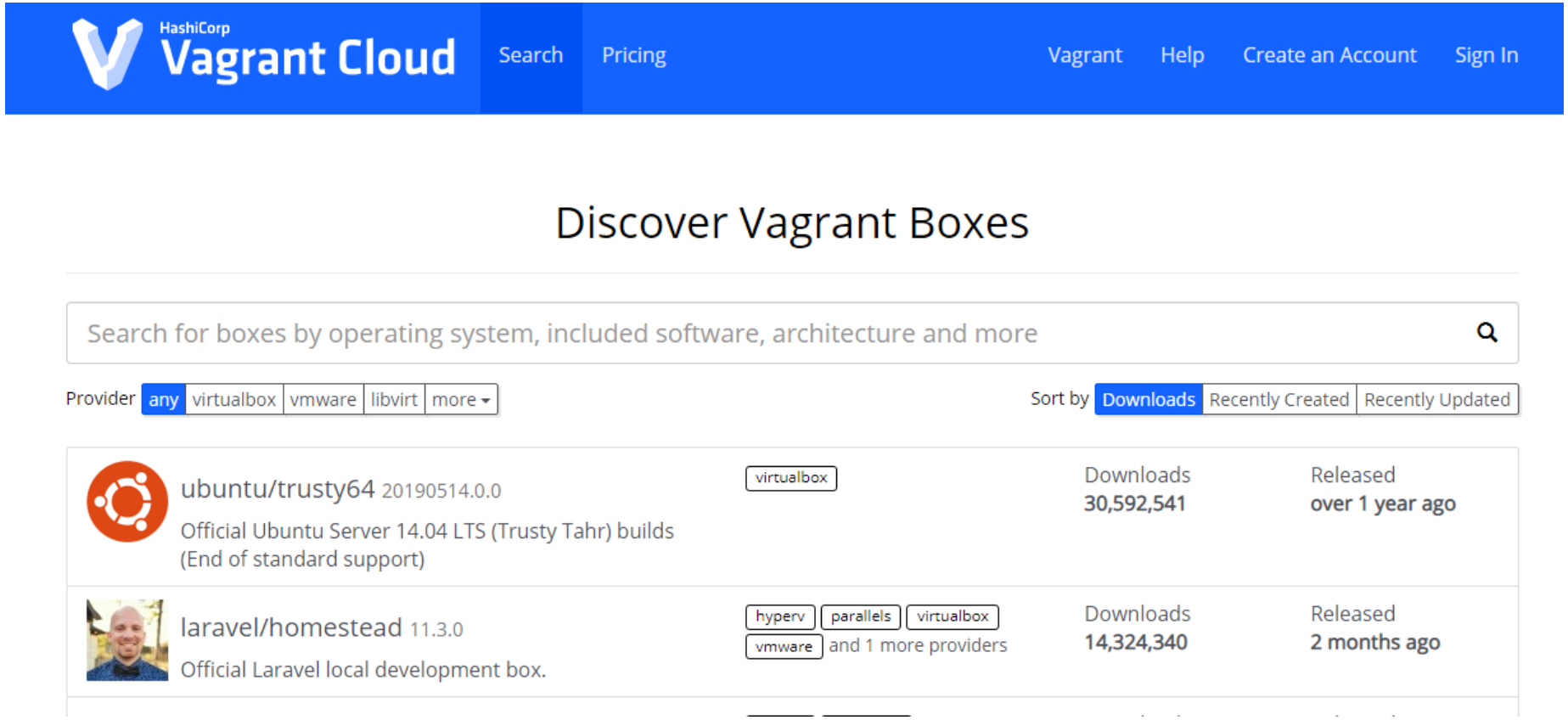
Vagrant Boxes (3/5): Install Vagrant Box

```
PS C:\WINDOWS\system32> vagrant box add --name ubuntu "https://app.vagrantup.com/ubuntu/boxes/trusty64/versions/20190514
.0.0/providers/virtualbox.box"
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'ubuntu' (v0) for provider:
    box: Downloading: https://app.vagrantup.com/ubuntu/boxes/trusty64/versions/20190514.0.0/providers/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
    box:
==> box: Successfully added box 'ubuntu' (v0) for 'virtualbox'!
PS C:\WINDOWS\system32> vagrant box add ubuntu/trusty64
==> box: Loading metadata for box 'ubuntu/trusty64'
    box: URL: https://vagrantcloud.com/ubuntu/trusty64
==> box: Adding box 'ubuntu/trusty64' (v20190514.0.0) for provider: virtualbox
    box: Downloading: https://vagrantcloud.com/ubuntu/boxes/trusty64/versions/20190514.0.0/providers/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
    box:
==> box: Successfully added box 'ubuntu/trusty64' (v20190514.0.0) for 'virtualbox'!
PS C:\WINDOWS\system32> vagrant box add https://app.vagrantup.com/ubuntu/boxes/trusty64
==> box: Loading metadata for box 'https://app.vagrantup.com/ubuntu/boxes/trusty64'
==> box: Adding box 'ubuntu/trusty64' (v20190514.0.0) for provider: virtualbox
The box you're attempting to add already exists. Remove it before
adding it again or add it with the `--force` flag.

Name: ubuntu/trusty64
Provider: virtualbox
Version: 20190514.0.0
```

```
PS C:\WINDOWS\system32> vagrant box add ubuntu/trusty64 --box-version=20190429.0.1
==> box: Loading metadata for box 'ubuntu/trusty64'
    box: URL: https://vagrantcloud.com/ubuntu/trusty64
==> box: Adding box 'ubuntu/trusty64' (v20190429.0.1) for provider: virtualbox
    box: Downloading: https://vagrantcloud.com/ubuntu/boxes/trusty64/versions/20190429.0.1/providers/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
    box:
==> box: Successfully added box 'ubuntu/trusty64' (v20190429.0.1) for 'virtualbox'!
```

Vagrant Boxes (4/5) : Box Version

Vagrant Boxes (5/5) : Vagrant Cloud



Lab-2 : Créer un vbox

- Créez un dossier lab-2
- Déplacez-vous dans ce dossier
- Initialisez vagrant en utilisant la version 20190425.0.0 ubuntu/trusty64
- Démarrez la VM
- Connectez-vous en ssh
- Installez nginx
- Démarrez le service et activez son démarage au lancement de la VM
- Créez une vbox à partir de cette VM
- Créez un compte sur vagrant cloud
- Publiez votre vbox sur vagrant cloud sous le nom <username>/nginx en version v1

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- **Vagrantfile**
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Vagrantfile (1/11) : pourquoi le vagrantfile ?

Vagrantfile (2/11) : Création du vagrantfile

```
PS C:\Users\dtafen\test> vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

```
PS C:\Users\dtafen\test> vagrant init -m
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

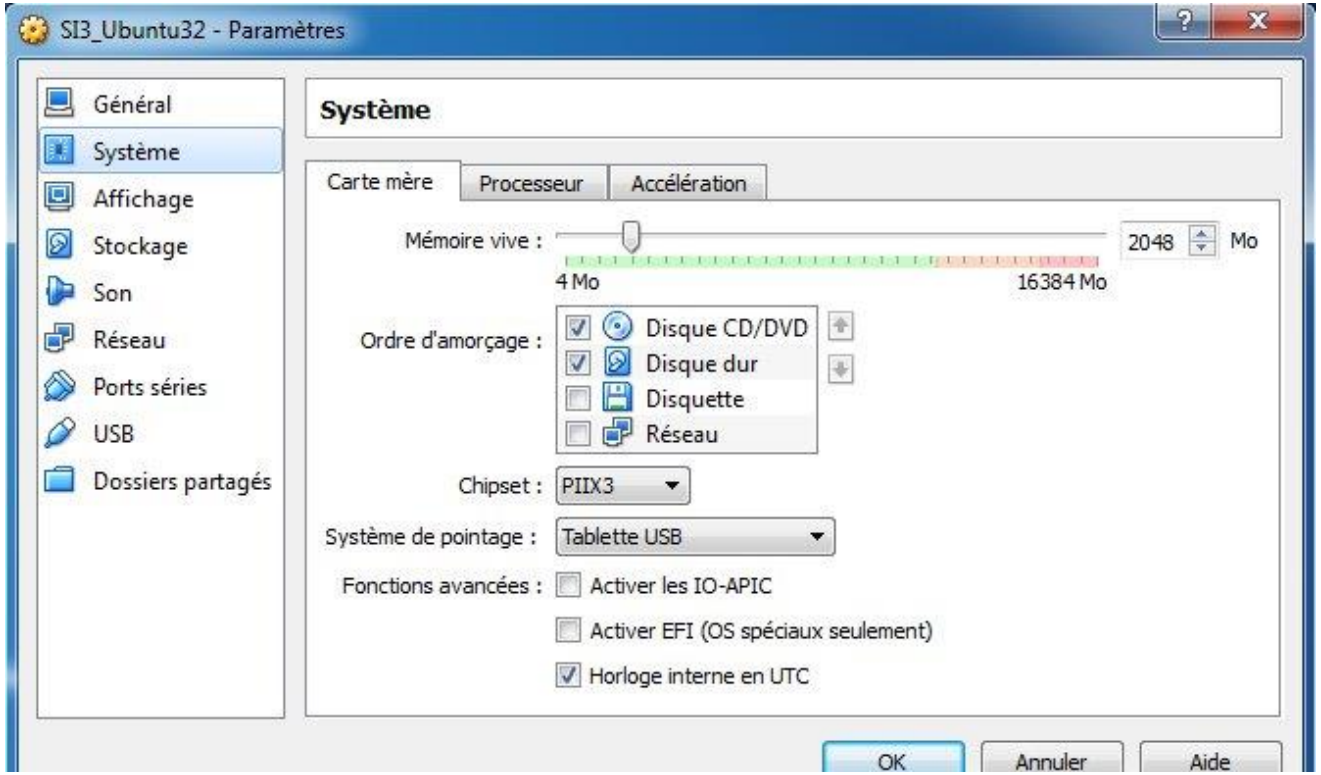
```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "base"
end
```

Vagrantfile (3/11): Système d’arborescence

Vagrantfile (4/11) : Config

```
Vagrant.configure("1") do |config|
  # v1 configs...
end

Vagrant.configure("2") do |config|
  # v2 configs...
end
```

Vagrantfile (5/11): Config.vm



- Voir documentation

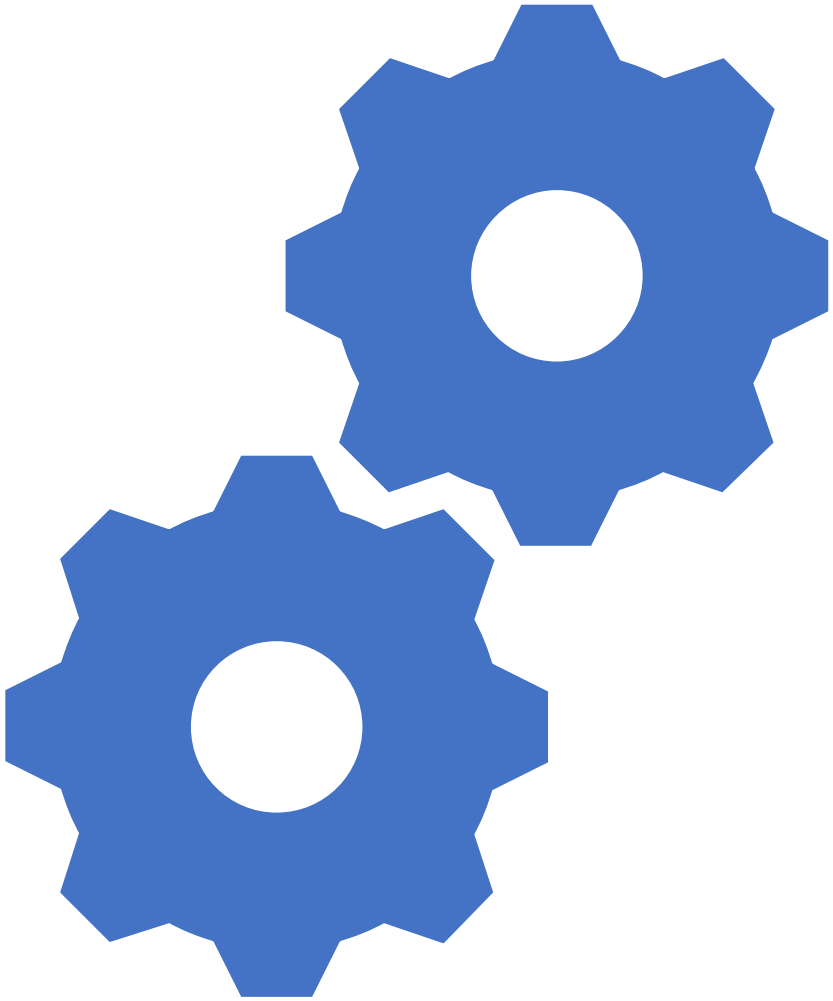
Vagrantfile (6/11): Config.ssh

- Voir [Documentation]

```
  GNU nano 2.5.3                File: /etc/ssh/sshd_config

# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin prohibit-password
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
                              [ Read 88 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   ^Y Prev Page
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line ^V Next Page
```

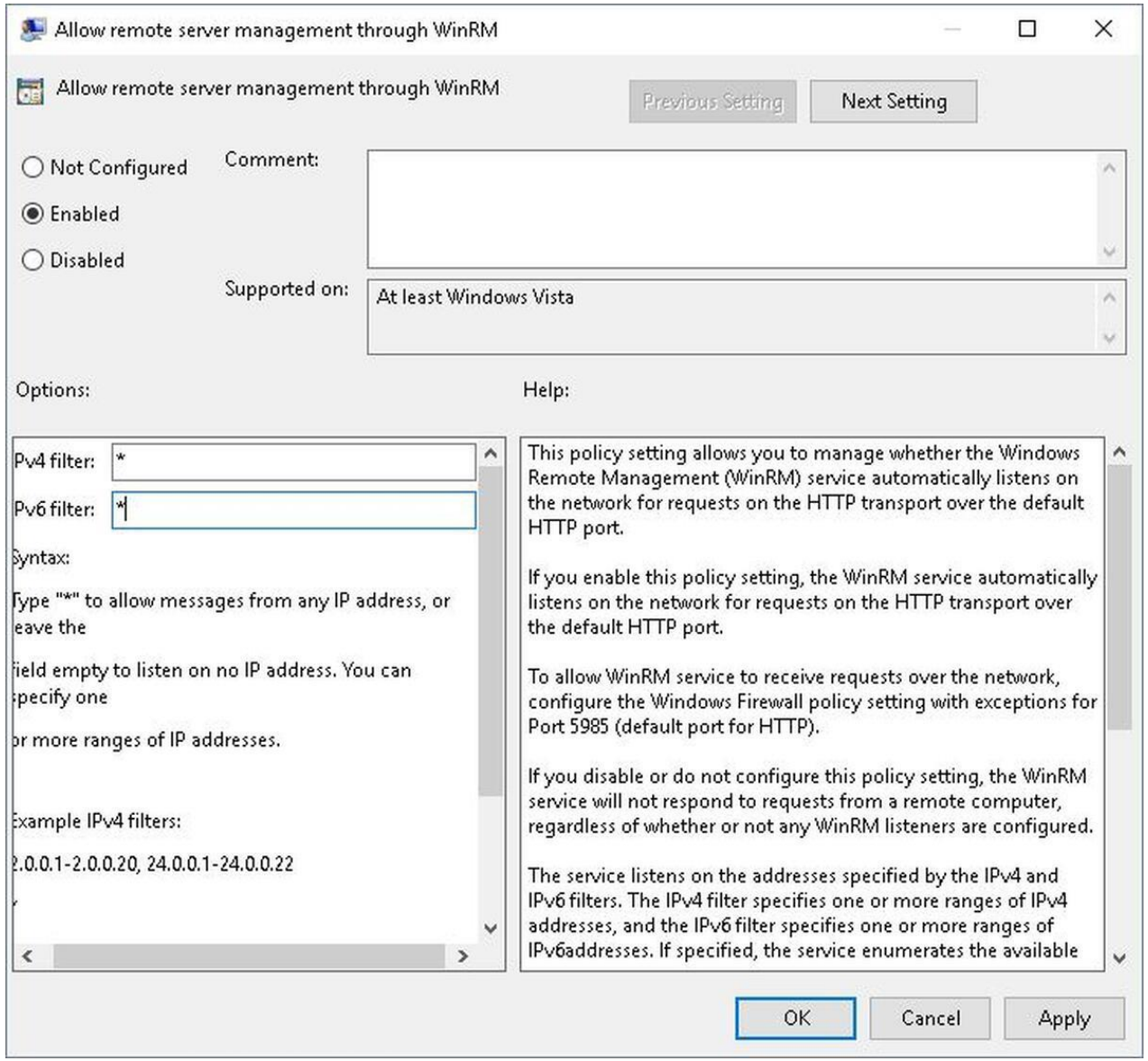
Vagrantfile (7/11) : Config.vagrant

- Voir Documentation
- Config.vagrant.host
- Config.vagrant.plugins
- Config.vagrant.sensitive



Vagrantfile (8/11) : config.winrm

Voir [documentation]



Vagrantfile (9/11): config.winssh

Voir documentation

Vagrantfile (10/11): Variable

- Variables d'environnement

```
### configuration parameters ###
# which host port to forward box SSH port to
LOCAL_SSH_PORT = "2222"
# Vagrant base box to use
BOX_BASE = "puppetlabs/centos-6.6-64-nocm"
# amount of RAM for Vagrant box
BOX_RAM_MB = "2048"
# number of CPUs for Vagrant box
BOX_CPU_COUNT = "1"
### /configuration parameters ###
```

Vagrantfile (11/11): loop

```
(1..3).each do |i|
  config.vm.define "node-#{i}" do |node|
    node.vm.provision "shell",
      inline: "echo hello from node #{i}"
  end
end
```

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

VAGRANTFILE_API_VERSION = "2"

cluster = {
  "master" => { :ip => "192.168.33.10", :cpus => 1, :mem => 1024 },
  "slave" => { :ip => "192.168.33.11", :cpus => 1, :mem => 1024 }
}

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  cluster.each_with_index do |(hostname, info), index|

    config.vm.define hostname do |cfg|
      cfg.vm.provider :virtualbox do |vb, override|
        config.vm.box = "ubuntu/trusty64"
        override.vm.network :private_network, ip: "#{info[:ip]}"
        override.vm.hostname = hostname
        vb.name = hostname
        vb.customize ["modifyvm", :id, "--memory", info[:mem], "--cpus", info[:cpus], "--hwvirtex", "on"]
      end # end provider
    end # end config

  end # end cluster
end
```

Lab-3 : Création dun Vagrantfile

- Créez un dossier lab-3
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer la VM à partir des informations suivantes
  - Image de base : Centos 7 By Geerlingguy
  - CPU : 2
  - RAM : 2 Go
- Variabilisez les paramètres indiquées ci-dessus
- Connectez-vous en ssh
- Installez nginx
- Arretez la VM puis supprimez la

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- **Networking**
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Networking (1/3): Port-forwarding

Networking (2/3): Private Network

Networking (3/3): public Network

Lab-4 : Déploiement d’un serveur Web

- Créez un dossier lab-4
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer la VM à partir des informations suivantes
  - Image de base : Centos 7 By Geerlingguy
  - CPU : 2
  - RAM : 2 Go
  - IP fixe privée : 10.0.0.10
- Variabilisez les paramètres indiquées ci-dessus
- Connectez-vous en ssh
- Installez nginx
- Vérifiez que vous avez accès à l’application depuis le PC local via http://10.0.0.10
- Arretez la VM puis supprimez la

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- **Multi-machine**
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

- Accurately modeling a multi-server production topology, such as separating a web and database server.
- Modeling a distributed system and how they interact with each other.
- Testing an interface, such as an API to a service component.
- Disaster-case testing: machines dying, network partitions, slow networks, inconsistent world views, etc.

Multi-machine (1/3): Pourquoi ?

Multi-machine (2/3): config.vm.define

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: "echo Hello"

  config.vm.define "web" do |web|
    web.vm.box = "apache"
  end

  config.vm.define "db" do |db|
    db.vm.box = "mysql"
  end
end
```

Multi-machine (3/3): Autres considérations

- Controlling Multiple Machines
- Communication Between Machines
- Specifying a Primary Machine
- Autostart Machines

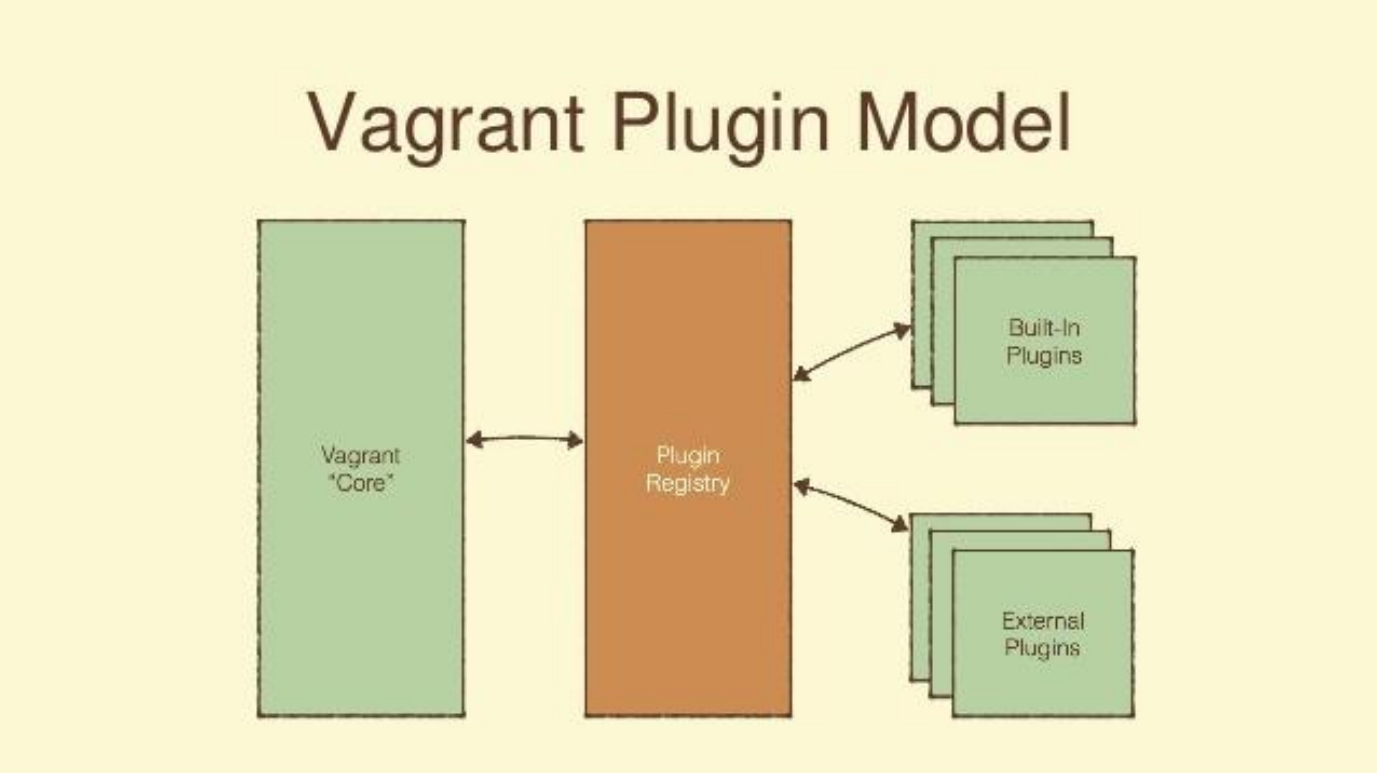
Lab-5 : Déploiement d’un serveur Web

- Créez un dossier lab-5
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer 3 VMs à partir des informations suivantes
  - Image de base : ubuntu/xenial64
  - CPU : 1
  - RAM : 1 Go
  - VM1: lb , ip: 10.0.0.10
  - VM2: web1, ip: 10.0.0.11
  - VM3: web2, ip: 10.0.0.12
- Connectez-vous en ssh sur chacunes d’elles
- Arretez les VMs puis supprimez les

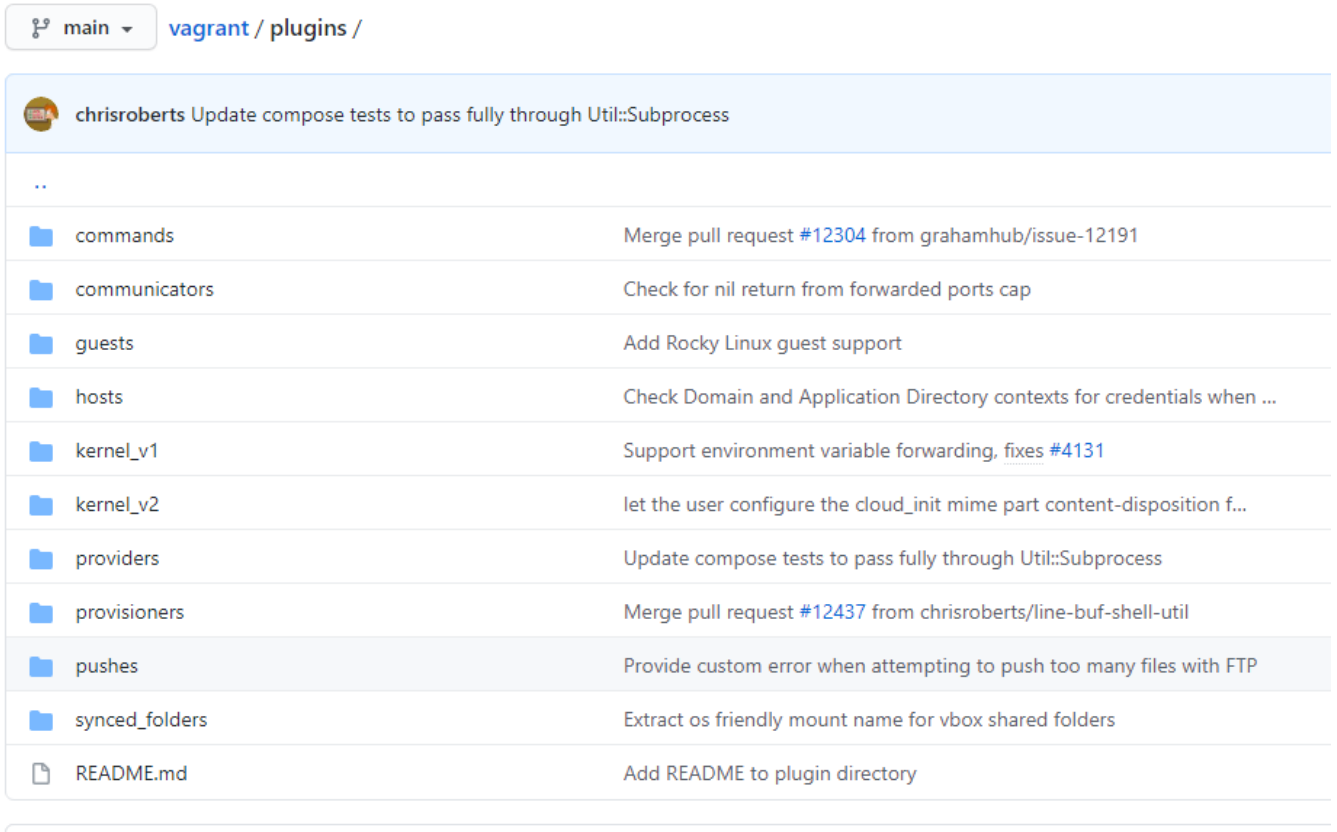
Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- **Plugins**
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

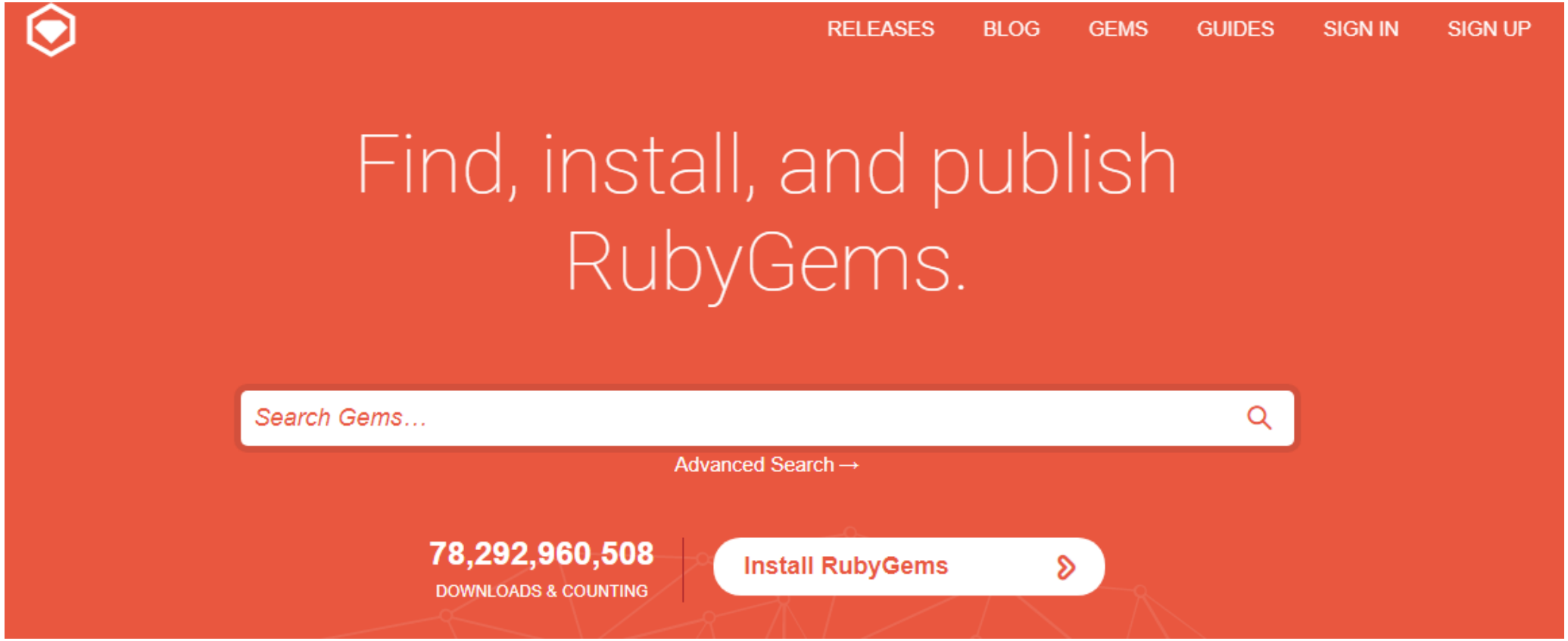
Plugin (1/4): Plugin Model



Plugin (2/4): Built-in plugin



Plugin (3/4): RubyGems source



Plugin (4/4): Gestion des plugins

```
PS C:\Users\dtafen\test> vagrant plugin -h
Usage: vagrant plugin <command> [<args>]

Available subcommands:
     expunge
     install
     license
     list
     repair
     uninstall
     update

For help on any individual command run `vagrant plugin COMMAND -h`
        --[no-]color                 Enable or disable color output
        --machine-readable           Enable machine readable output
    -v, --version                    Display Vagrant version
        --debug                      Enable debug output
        --timestamp                  Enable timestamps on log output
        --debug-timestamp            Enable debug output with timestamps
        --no-tty                     Enable non-interactive output
```

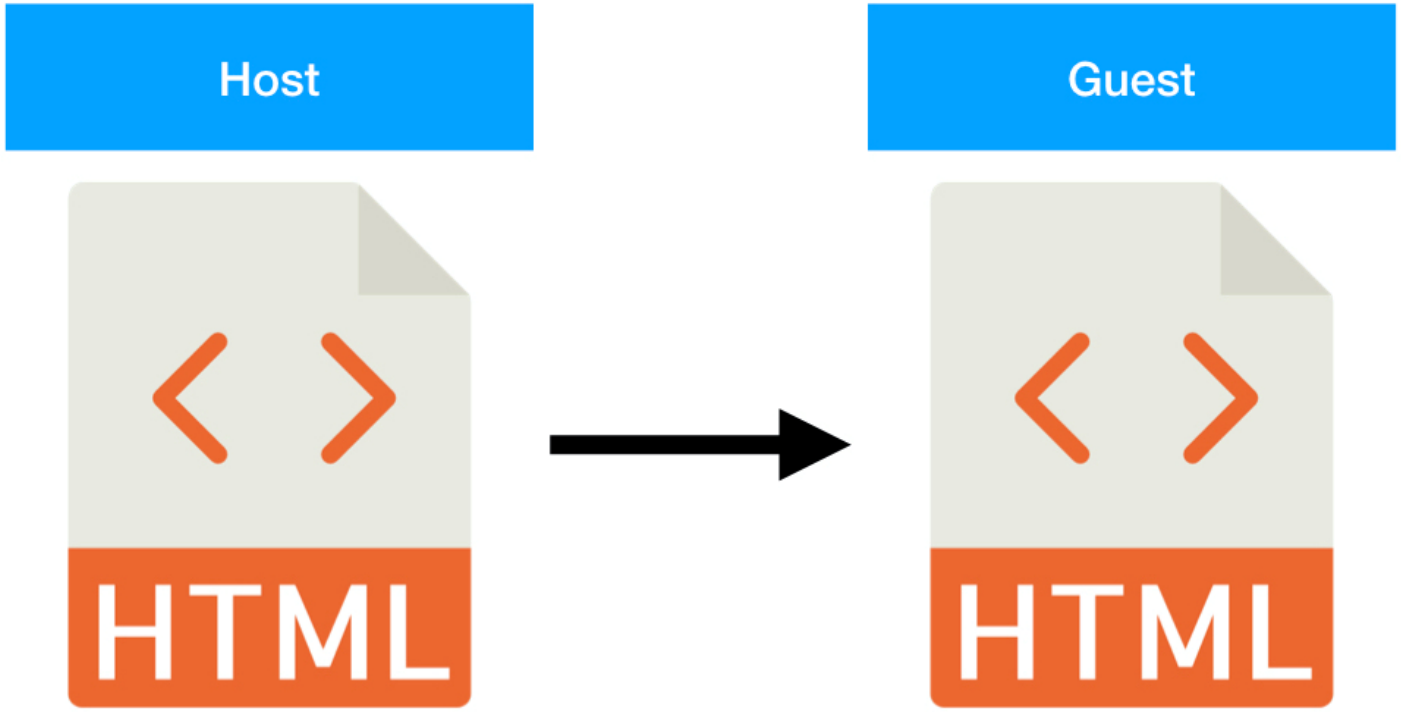
Lab-6 : Plugin

- Créez un dossier lab-6
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer 3 VMs à partir des informations suivantes
  - Image de base : ubuntu/xenial64
  - CPU : 1
  - RAM : 1 Go
  - VM1: lb , ip: 10.0.0.10
  - VM2: web1, ip: 10.0.0.11
  - VM3: web2, ip: 10.0.0.12
- Installez et Utilisez le plugin vagrant-hostsupdater afin que le fichier hosts de chaque machine corresponde à la réalité de leur nom et de leur IP malgré les actions UP, RESUME et RELAOD (lisez la Doc pour mieux comprendre son utilité)
- Lors du vagrant up, verifiez que le plugin est bien appelé
- Connectez-vous en ssh sur chacunes d'elles
- Arretez les VM puis supprimez les

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- **Files**
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Files (1/5): config.vm.synced_folder



```
Vagrant.configure("2") do |config|
  # other config here

  config.vm.synced_folder "src/", "/srv/website"
end
```

Files (2/5): Basic Syncing

Files (3/5): NFS

```
Vagrant.configure("2") do |config|
  config.vm.synced_folder ".", "/vagrant", type: "nfs"
end
```

Files (4/5): Rsync

```
Vagrant.configure("2") do |config|
  config.vm.synced_folder ".", "/vagrant", type: "rsync",
    rsync__exclude: ".git/"
end
```

```
Vagrant.configure("2") do |config|
  # ... other configuration

  config.vm.provision "file", source: "~/.gitconfig", destination: ".gitconfig"
end
```

```
Vagrant.configure("2") do |config|                                    Copy
  # ... other configuration

  config.vm.provision "file", source: "~/path/to/host/folder", destination: "$HOME/remote/newfolder"
end
```

Files (5/5): Upload File and Folder

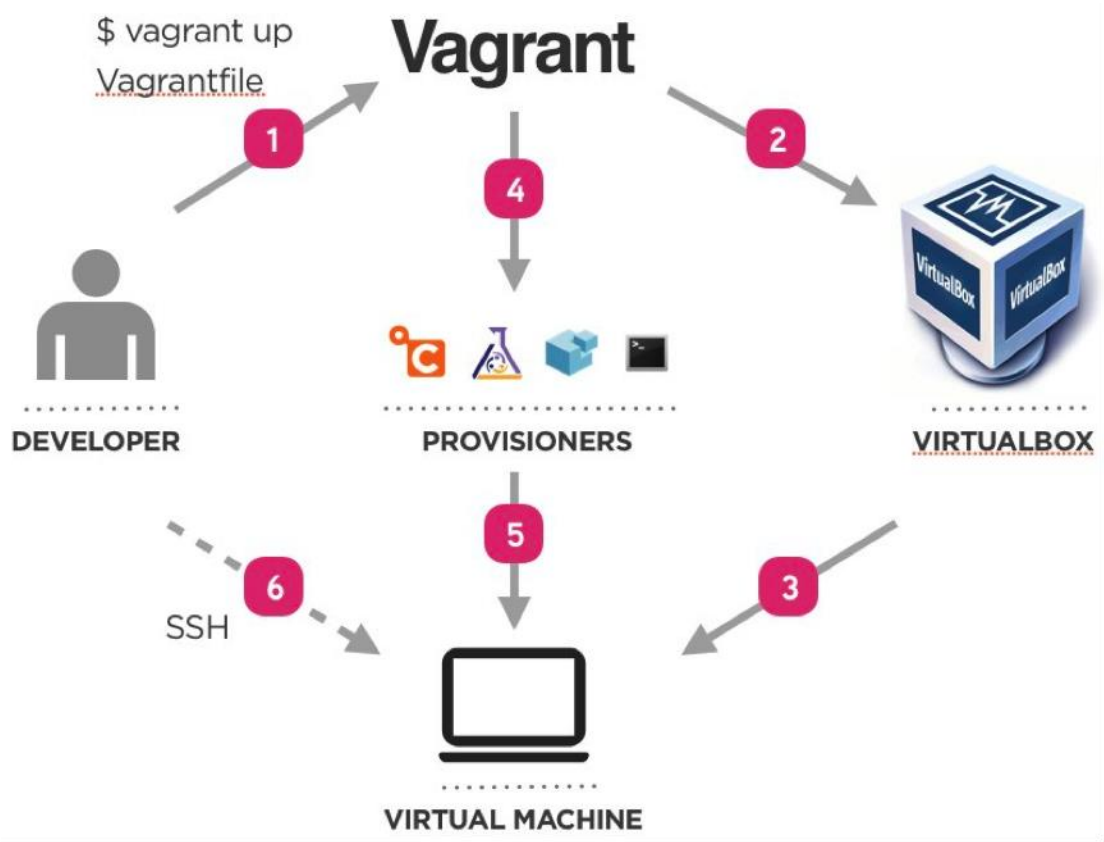
Lab-7 : Webapp Folder

- Créez un dossier lab-7
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer 1 VMs à partir des informations suivantes
  - Image de base : <username>/nginx (vagrant box créée au lab-2)
  - CPU : 1
  - RAM : 1 Go
  - IP fixe : 10.0.0.10
- Recupérez en local dans votre dossier lab-7 l'application static-website-example
- Montez le code de l'application par le moyen de votre choix dans le dossier **/usr/share/nginx/html** de la VM
- Vérifiez que le site recupéré depuis github est bien accéssible
- Arretez la VM puis supprimez la

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- **Shell Provisioning**
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- Conclusion

Shell Provisioner (1/5): config.vm.provisioner



- Install software
- Alter configurations
- Operating-system-level changes
- System settings

Shell Provisioner (2/5): Provision option

vagrant up

vagrant up --provision

vagrant provision

vagrant reload --provision

Shell Provisioner (3/5): Inline Shell

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell",
    inline: "echo Hello, World"
end
```

```
$script = <<-SCRIPT
echo I am provisioning...
date > /etc/vagrant_provisioned_at
SCRIPT

Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: $script
end
```

Shell Provisioner (4/5): External Script

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", path: "script.sh"
end
```

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", path: "https://example.com/provisioner.sh"
end
```

Shell Provisioner (5/5): Arguments

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell" do |s|
    s.inline = "echo $1"
    s.args   = "'hello, world!'"
  end
end
```

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell" do |s|
    s.inline = "echo $1"
    s.args   = ["hello, world!"]
  end
end
```

Lab-8 : Shell

- Créez un dossier lab-8
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer 3 VMs à partir des informations suivantes
  - Image de base : ubuntu/xenial64
  - CPU : 1
  - RAM : 1 Go
  - VM1: lb , ip: 10.0.0.10, exécutez le script suivant
  - VM2: web1, ip: 10.0.0.11, exécutez le script suivant
  - VM3: web2, ip: 10.0.0.12, exécutez le script suivant
- N'hésitez pas à lire le script pour mieux comprendre les operations exécutées
- Accédez à l'interface web de chaque serveur et confirmez que l'application a bien été déployée
- Arretez les VM puis supprimez les

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- **Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)**
- Exemples pratique
- Mini-projet
- Conclusion

Autres provisioners (1/1): Lisons la documentation

https://www.vagrantup.com/docs/provisioning

Lab-9 : Installation de nginx à l'aide d'ansible

- Créez un dossier lab-9
- Déplacez-vous dans ce dossier
- Créez un vagrantfile afin de configurer 1 VMs à partir des informations suivantes
  - Image de base : ubuntu/xenial64 ou tout OS de votre choix
  - CPU : 1
  - RAM : 1 Go
  - IP fixe : 10.0.0.10
- Ecrivez un playbook s'appellant nginx.yaml qui permettra d'installer nginx et le démarrer
- Vous devrez vous baser sur ansible-local pour que le playbook soit execute dans la VM
- Vérifiez que la page par defaut de nginx est bien accessible via le navigateur
- Arretez la VM puis supprimez la

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- **Exemples pratique**
- Mini-projet
- Conclusion

Exemples pratiques (1/2): Lab-10 - Jenkins deployment with docker, ansible and bash

Exemples pratiques (2/2): Lab-11 - Gitlab Deployment with docker, ansible and bash

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- **Mini-projet**
- Conclusion

Mini-projet

minikube

Plan

- Présentation de la formation
- Pourquoi Vagrant ?
- Commandes de Base
- Vagrant Boxes
- Vagrantfile
- Networking
- Multi-machine
- Plugins
- Files
- Shell Provisioning
- Autres provisioner (Ansible, Chef, Puppet, Docker,Salt)
- Exemples pratique
- Mini-projet
- **Conclusion**

Conclusion : Merci à vous