

# Gérez les images de vos machines virtuelles

# Mathieu Corbin, Ingénieur **@Exoscale**

## Développeur

- Clojure
- Golang
- Emacs
- ...

## Sysadmin

- Automatisation
- CI/CD
- Monitoring
- Systèmes distribués
- ...



@\_mcorbin

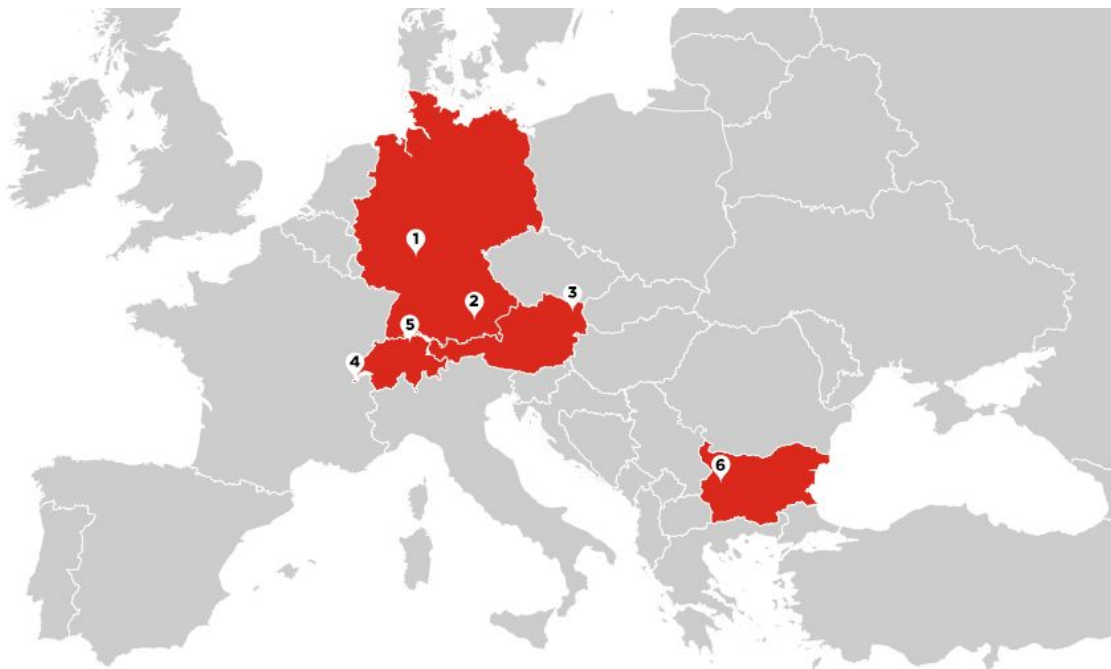
mcorbin.fr



# EXOSCALE

Un Cloud Provider Européen

- |    |           |          |
|----|-----------|----------|
| 1. | FRANKFURT | DE-FRA-1 |
| 2. | MUNICH    | DE-MUC-1 |
| 3. | VIENNA    | AT-VIE-1 |
| 4. | GENEVA    | CH-GVA-2 |
| 5. | ZURICH    | CH-DK-2  |
| 6. | SOFIA     | BG-SOF-1 |





# Le cloud

Hostname

hello

This will be the hostname and the display name of your instance.

Template



☒ Linux Debian 10 (Buster) 64-bit

☐ Linux Debian 9 64-bit

☐ Linux Debian 8 64-bit

Sélection d'une image

Création

Your instance:

hello



DE-FRA-1



Linux Debian 10 (Buster) 64-bit

2 x 2198 MHz

4 GB RAM

50 GB

Keypair

dell

Security Groups

default









CREATE

# Le cloud

Hostname

This will be the hostname and the display name of your instance.

Template



☒ Linux Debian 10 (Buster) 64-bit

☐ Linux Debian 9 64-bit


☐ Linux Debian 8 64-bit


Sélection d'une image

Création

Your instance:


**hello**

 DE-FRA-1

 Linux Debian 10 (Buster) 64-bit

2 x 2198 MHz  
4 GB RAM  
50 GB

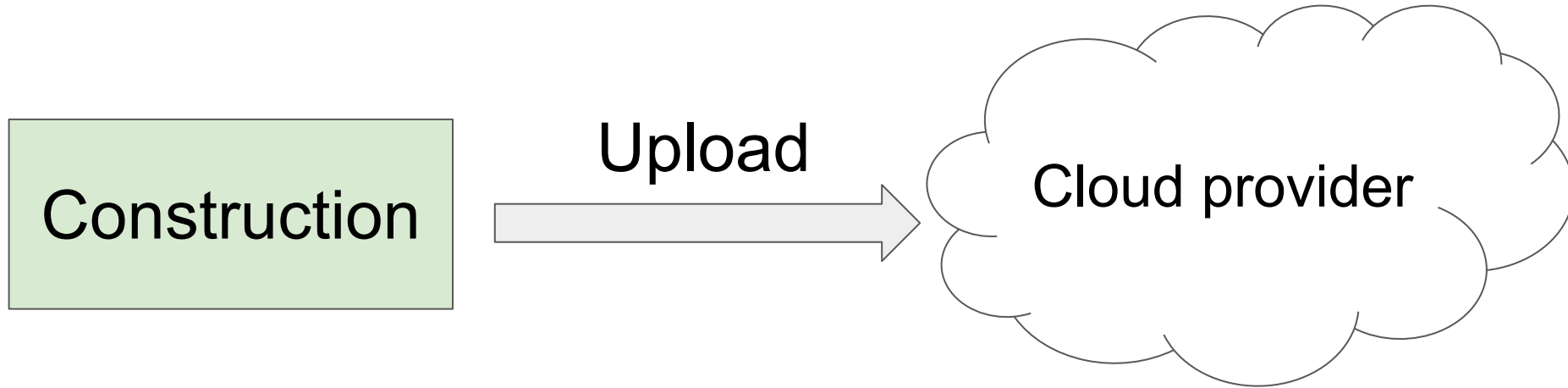
Keypair  
dell

Security Groups   
default

**CREATE**

# Images “custom”

Utilisation d’images que **vous** avez créé



# Images “custom”: pourquoi ?

- Images non disponibles sur votre cloud
  - OS, distributions spécifiques
- Multi Cloud
  - Déployer la même image sur différents Cloud



# Images “custom”: pourquoi ?

- Images répondant à un besoin spécifique
  - Logiciels pré-installés
  - Gain de temps
- Gérer les mises à jour
  - Sécurité
  - Mises à jour
- Immutable infrastructure

# La VM est la brique de base d'une infrastructure

Souvent négligée au profit des conteneurs

# Construire une image

Shell scripts + qemu + chroot + ...

# Construire une image

Shell scripts + qemu + chroot + ...



# Construire une image



HashiCorp

# Packer

# Packer: avantages

- Façon standard de construire des images
- Configuration des constructions en json (versioning)
- Support pour de nombreux Cloud/outils

# Cloud init

Une collection de scripts permettant de configurer une machine au boot



# Cloud init

- Chaque Cloud Provider maintient une “Datasource” dans Cloud Init
- Les Cloud providers exposent des services qui seront contactés par Cloud Init
- Au boot, Cloud Init détecte le Cloud Provider et va chercher des informations sur la machine



# La datasource Exoscale

- Expose des informations sur la machine (offering, ip, hostname, instance ID, zone)...
- Gère l'authentification (clé publique et mot de passe)
- Va chercher les **user data** fournies par le client

# User data

Un fichier YAML passé à la création de la machine

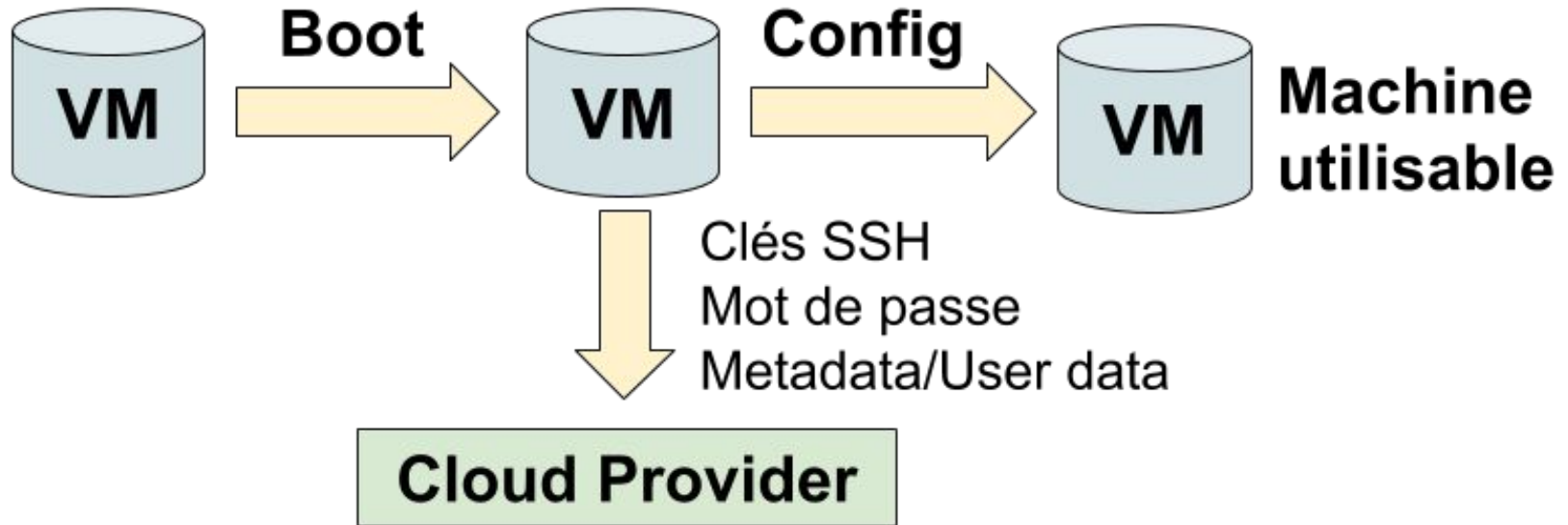
```
users:
  - name: fizzbuzz
    sudo: False
    ssh_authorized_keys:
      - <ssh pub key 1>
      - <ssh pub key 2>

runcmd:
  - [ ls, -l, / ]

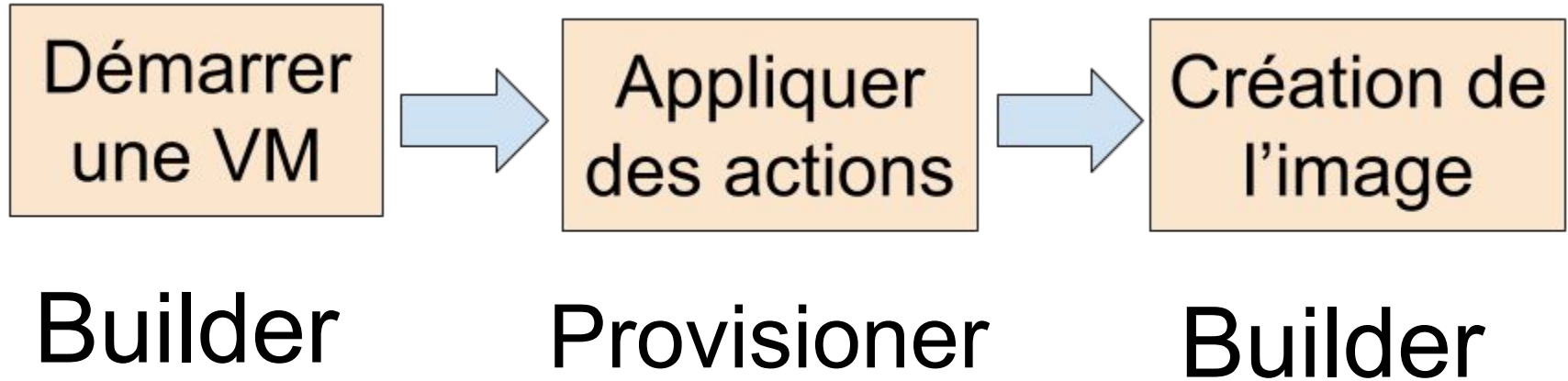
write_files:
  - path: /tmp/foo
    content: "my file content"
```

# Cloud init est obligatoire

Sans Cloud Init, la machine n'est pas configurée



# Packer: fonctionnement



# Builder

Démarre une machine virtuelle

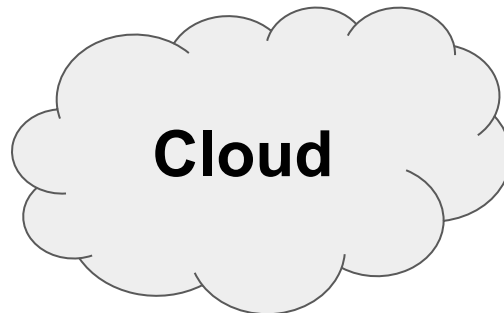


HashiCorp

**Vagrant**



**QEMU**



# Image de base

Les builders partent d'une image de base

# Images de votre Cloud Provider

- Vous démarrez une image disponible sur votre Cloud Provider, si ce dernier possède un Builder Packer
- **Cloud Init** généralement pré-installé sur ces images
- La machine virtuelle est transformée en template

# Images “Cloud” des distributions

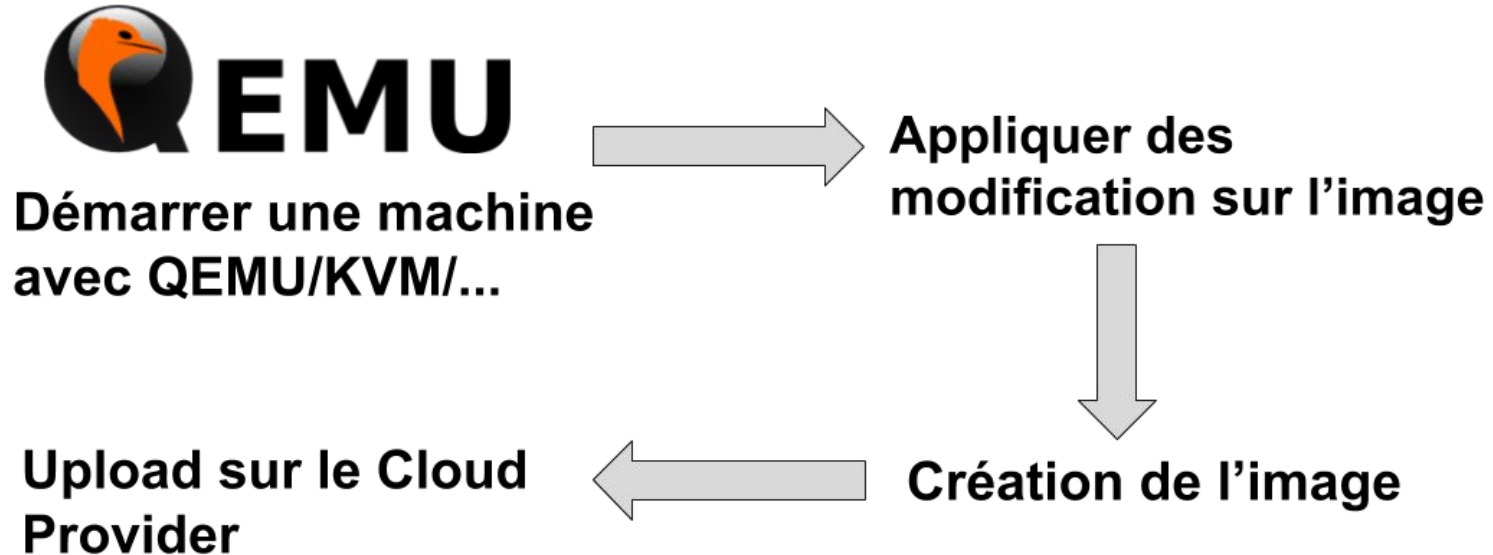
Les distributions Linux (Debian, Ubuntu, Centos...) fournissent des images “Cloud”

- **Cloud Init** pré-installé
- Prêtes à être utilisées sur un Cloud Provider



# Images “Cloud” des distributions

On peut vouloir personnaliser ces images avant de les déployer chez notre Cloud Provider.



# Authentication: Cloud init

Il faut pouvoir se connecter sur la machine que l'on démarre  
Cloud init fournit deux datasources utilisables “hors ligne”

- No Cloud
- No Cloud Net

# Datasource No Cloud

Un disque contenant des user data est passé à la machine

Cloud init détectera ce disque et configurera la machine

# Datasource No Cloud

## Fichier “user-data”

```
#cloud-config
ssh_authorized_keys:
- "<your_public_key>"
```

**cloud-localds seed.img user-data**

# Datasource No Cloud Net

Packer peut exposer à la machine virtuelle un dossier en  
HTTP

La datasource “No Cloud Net” va lire et utiliser ces fichiers

# Un exemple: Debian 10



**debian**

# Variables

```
"variables": {  
  "image_url": "https://cdimage.debian.org/cdimage/openstack/  
               current-10/debian-10-openstack-amd64.qcow2",  
  "image_checksum_url": "https://cdimage.debian.org/cdimage/  
                        openstack/current-10/SHA256SUMS",  
  "image_checksum_type": "sha256",  
  "image_name": "debian-buster",  
  "ssh_private_key_file": "{{env `PACKER_PRIVATE_KEY`}}"  
},
```

# Builder

```
"type": "qemu",
"qemuargs": [
    ["-cpu", "qemu64,rdrand=on"],
    ["-drive", "file=output-qemu/{{user `image_name`}}.qcow2,format=qcow2,if=virtio"],
    ["-drive", "file=seed.img,format=raw,if=virtio"]
],
"vm_name": "{{user `image_name`}}.qcow2",
"iso_url": "{{user `image_url`}}",
"iso_checksum_url": "{{user `image_checksum_url`}}",
"iso_checksum_type": "{{user `image_checksum_type`}}",
"disk_image": true,
"disk_size": 3000,
"disk_compression": true,
"communicator": "ssh",
"ssh_username": "debian",
"ssh_private_key_file": "{{user `ssh_private_key_file`}},
```



# Compression des images

## Taille virtuelle

```
qemu-img info debian-buster.qcow2  
  
image: debian-buster.qcow2  
file format: qcow2  
virtual size: 10G (10737418240 bytes)  
disk size: 471M  
cluster_size: 65536  
Format specific information:  
  compat: 1.1  
  lazy refcounts: false  
  refcount bits: 16  
  corrupt: false
```

# Provisioners

```
"provisioners": [  
  {"type": "shell",  
    "execute_command": "chmod +x {{.Path}}; sudo {{.Path}}",  
    "scripts": [  
      "scripts/cloud-cleanup.sh",  
      "scripts/cloud-password-module.sh",  
      "scripts/grub-exoscale.sh",  
      "{{user `image_name`}}/script.sh"  
    ]  
  }  
]
```

# Provisioners

- Shell: exécute des scripts shells
- File: copie des fichiers sur la machine
- Ansible: lance des playbooks Ansible sur la machine
- Custom: possibilité d'étendre Packer
- ...

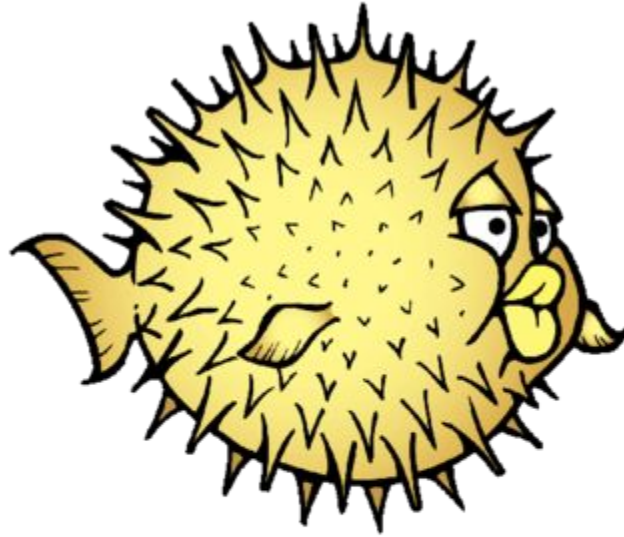
# Créer une image à partir d'un installer

# Créer une image à partir d'un installer

Installation interactive: répondre aux “questions” lors de l'installation

- Permet de construire plus de distributions
- Plus d'options lors de l'installation (choix des partitions par exemple)
- On simule le clavier avec Packer

# Un exemple: OpenBSD



***Open*BSD**

# Variables

```
"variables": {  
  "image_url": "http://ftp.fr.openbsd.org/pub/OpenBSD/  
               6.4/amd64/install64.iso",  
  "image_checksum_url": "http://ftp.fr.openbsd.org/pub/  
                        OpenBSD/6.4/amd64/SHA256",  
  "image_checksum_type": "sha256",  
  "image_name": "openbsd-6.4",  
  "ssh_password": "{{uuid}}"  
},
```

# Builder

```
"builders": [  
  {  
    "type": "qemu",  
    "iso_url": "{{user `image_url`}}",  
    "iso_checksum_url": "{{user `image_checksum_url`}}",  
    "iso_checksum_type": "{{user `image_checksum_type`}}",  
    "shutdown_command": "/sbin/halt -p",  
    "disk_size": 12000,  
    "format": "qcow2",  
    "ssh_username": "root",  
    "ssh_password": "{{user `ssh_password`}}",  
  }  
]
```

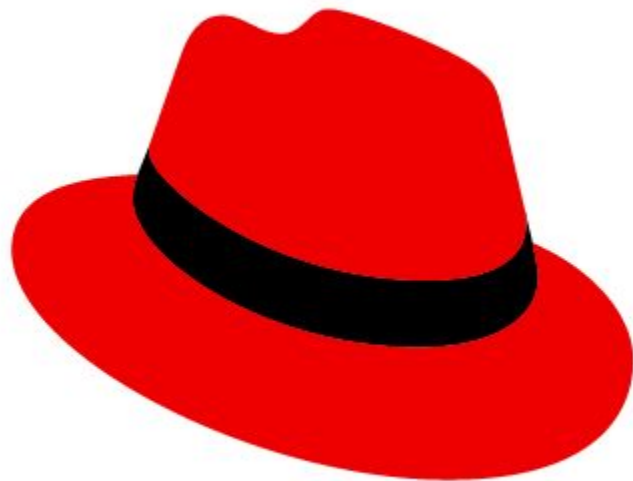


```
"boot_wait": "30s",
"boot_command": [
    "I<enter><wait>",
    "<enter><wait>",
    "openbsd<enter><wait>",
    "vio0<enter><wait>",
    "dhcp<enter><wait>",
    "none<enter><wait>",
    "<enter><wait>",
    "<enter><wait>",
    "{{user `ssh_password`}}<enter><wait>",
    "{{user `ssh_password`}}<enter><wait>",
    "yes<enter><wait>",
    "no<enter><wait>",
    "no<enter><wait>",
    "no<enter><wait>",
    "yes<enter><wait>",
    "Europe/Zurich<enter><wait>",
    "sd0<enter><wait>",
    "whole<enter><wait>",
```

# Une fois l'installation terminée...

- La machine redémarrera
- Configuration de la machine via ssh + mot de passe
  - Désactiver le mot de passe à la fin de l'installation

# RHEL/Centos: Fichiers kickstart (ou équivalent chez Debian etc...)



# Fichier kickstart

- Packer expose un fichier “kickstart.cfg” via HTTP
- Ce fichier décrit comment construire l'image
- Technique utilisable pour d'autres distributions (mais format différent)

# Fichier kickstart: exemple

```
url --mirrorlist="http://mirrorlist.centos.org/  
    ?release=$releasever&arch=$basearch&repo=os"  
  
install  
keyboard 'us'  
reboot  
  
timezone Europe/Zurich  
  
network --bootproto=dhcp --device=eth0 --ipv6=auto --no-activate  
network --hostname=localhost.localdomain  
lang en_US  
  
auth --useshadow --passalgo=sha512  
ignoredisk --only-use=vda  
# System bootloader configuration  
bootloader --append=" crashkernel=auto" --location=mbr --boot-drive=vda
```

```
clearpart --all --initlabel  
zerombr  
  
part /boot/efi --fstype="efi" --ondisk=vda --size=300  
    --fsoptions="umask=0077,shortname=winnt"  
part / --fstype="xfs" --ondisk=vda --grow  
  
skipx  
  
eula --agreed
```

```
%packages --excludedocs
@^minimal
@core
chrony
cloud-init
cloud-utils-growpart
gdisk
dracut-config-generic
dracut-norecue
firewalld
kexec-tools
openssh-clients
openssh-server
sudo
wget
curl
rsync
tar
```

```
%post
```

```
# configure ssh keys
```

```
mkdir /home/centos/.ssh
```

```
cat <<EOF >/home/centos/.ssh/authorized_keys
```

```
ssh-rsa <PUBLIC_KEY>
```

```
EOF
```

```
chown -R centos:centos /home/centos/.ssh
```

```
chmod 0600 /home/centos/.ssh/authorized_keys
```

```
chmod 0600 /home/centos/.ssh/
```

```
# unlock centos user
```

```
passwd -u centos
```



# Fichier kickstart: le lire depuis packer

```
"http_directory": "{{user `image_name`}}/http",
"boot_command": [
    "<leftCtrlOn>e<leftCtrlOff>",
    "<spacebar>",
    "inst.text<spacebar>inst.ks=http://{{ .HTTPIP }}:{{ .HTTPPort }}
    /kickstart.cfg<wait>",
    "<leftCtrlOn>x<leftCtrlOff>"
]
```

# Gérer l'UEFI

Construire/démarrer des images UEFI  
UEFI pour des disques > 2 TB

<input checked="" type="radio"/>	Storage-jumbo	24x 2198 MHz	225 GB RAM
<input type="radio"/>	Storage-huge	8x 2198 MHz	32 GB RAM
	Jumbo	24x 2198 MHz	225 GB RAM
<a href="#">Request access</a> to this instance type.			

Disk

10 TB

20 TB

# Création d'une partition EFI

```
# Disk partitioning information
part /boot/efi --fstype="efi" --ondisk=vda --size=300
    --fsoptions="umask=0077,shortname=winnt"
part / --fstype="xfs" --ondisk=vda --grow
```

# Utilisation du firmware UEFI

```
"qemuargs": [  
    ["-drive", "file=output-qemu/{{user `image_name`}}.qcow2,  
                if=virtio,cache=writeback,discard=ignore,  
                format=qcow2"],  
    ["-drive", "file=/usr/share/OVMF/OVMF_CODE.fd,  
                if=pflash,format=raw,unit=0,readonly=on"],  
    ["-drive", "file=/usr/share/OVMF/OVMF_VARS.fd,if=pflash,  
                format=raw,unit=1"],  
]
```

# Post processors

Réalise une action après la construction

- Publier l'image sur un cloud
- Compression du résultat du build
- Exécution de scripts shell
- ...

# Test d'une image

# Pourquoi tester une image ?

- Tests de sécurité (clés publiques existantes par exemple)
- Tester si l'image/certains services démarrent correctement
- Tests “techniques” (reboot, cloud init...)

# Comment tester ?

- Server spec / Goss
  - S'installent sur la machine
  - Permettent de définir des checks à exécuter
- Outil maison



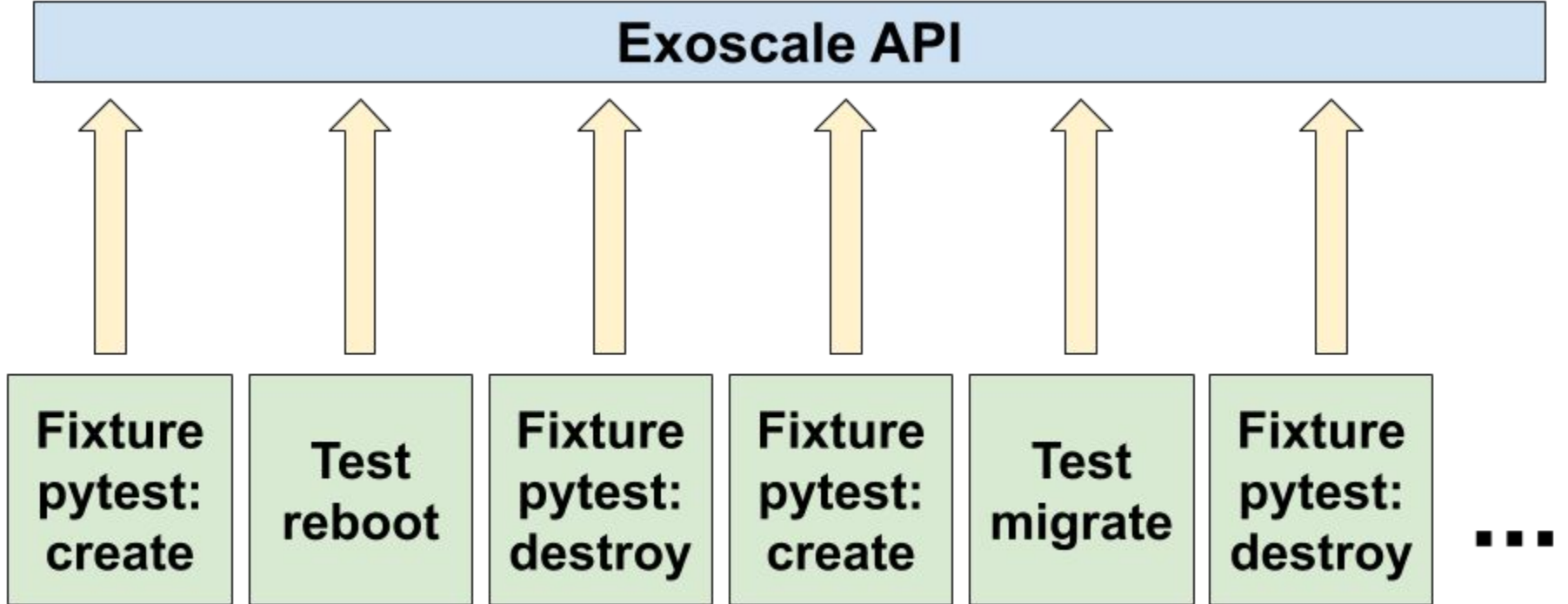
# Les tests à Exoscale: pytest (Python)

- Création d'une fixture pytest créant une machine à partir d'un template
- Exécution du test
- Destruction de la machine

# Les tests à Exoscale

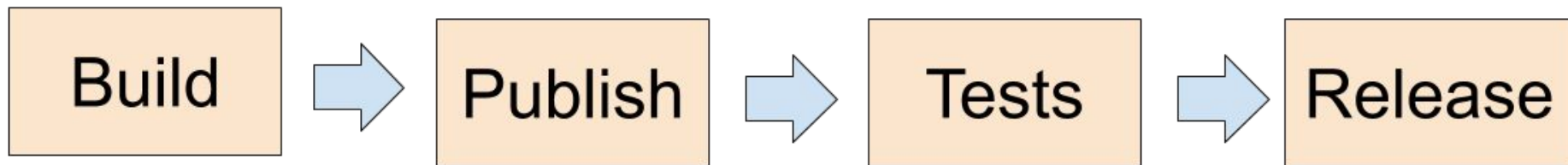
- Test de Cloud Init
- Test de la connexion via clé/mot de passe
- Test du changement de mot de passe
- Test de migration “live” d’un hyperviseur à un autre
- ...

# Les tests à Exoscale



# Intégration continue

Tout cela s'intègre très bien dans un outil de CI



# Intégration continue

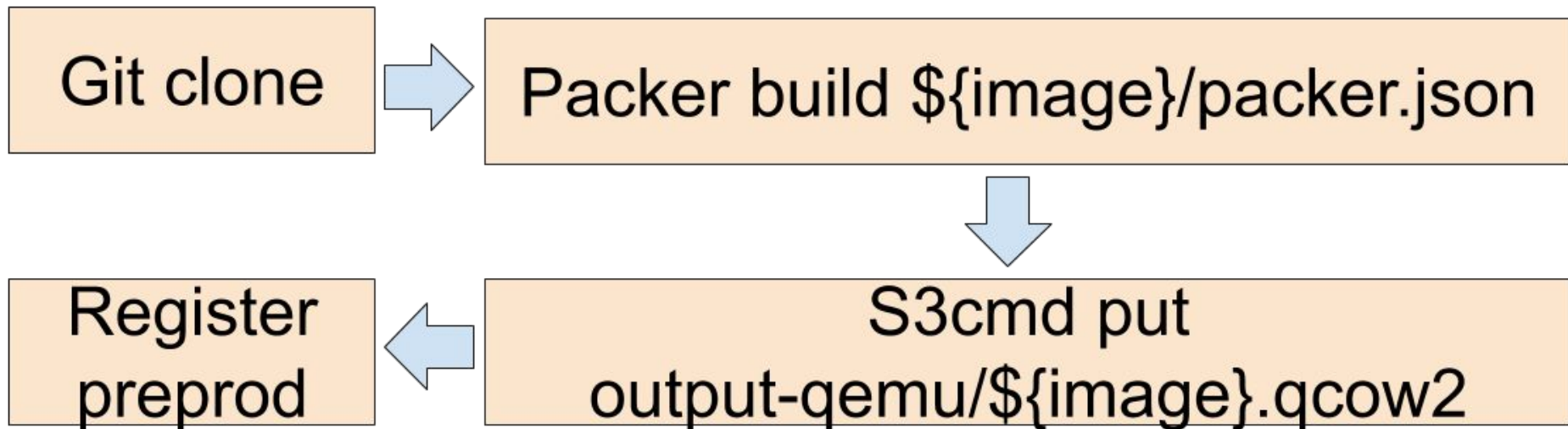
Exoscale utilise Jenkins (Jenkinsfile + Job DSL)



# Intégration continue: build + publish

- Un dépôt git contenant un dossier par image à construire
- Un job Jenkins prenant en paramètre le nom de l'image à construire
- Le job construit l'image, l'upload sur SOS (S3-like),
- L'image est ensuite publiée en préprod

# Intégration continue: build + publish



# Intégration continue: Tests

- Un job prenant un template ID et qui lance les tests sur ce dernier

# Intégration continue: Release

- Un job prenant un template ID et l'enregistre sur l'environnement de production



Questions ?