UST4HPC
Openstack / Ansible / Packer / Terraform

Nicolas Gibelin

UAR GRICAD - CNRS

19 - 20 juin 2023

Section 1

Introduction

UST4HPC Cloud

?

- Qui à déjà utilisé
  - Openstack
  - Packer
  - Terraform
  - Ansible

!

- Les pdf contiennent des liens cliquable vers de la documentation
- Introduction à tous ces outils
  - 3 TPs
  - Ansible en fil rouge
- Corrigés à la fin

Hashicorp

Société spécialisée dans les domaines du cloud et de l'automatisation.

Hashicorp

- Infrastructure
  - Terraform
  - Packer
- Sécurité
  - Vault
- Applications
  - Vagrant
- Et d'autres

Pour les developpeurs

- developer.hashicorp.com

Section 2

Openstack

Historique

- 2010 : début du projet Rackspace / Nasa
- Plateforme de Cloud computing sur matériel standard
- Développé en python
- Communautaire et Opensource
- Un version tous les 6 mois : Releases
  - Austin, Bexar, Cactus, ... Yoga, Zed (alphabet terminé)
  - 2023.1 Antelope, 2023.2 Bobcat ...
- Une multitude de projets
  - “Core” et “Big Tent”

Chiffres

- Pour Wallaby
  - Commits: 11507
  - Patch Sets: 48423
  - Resolved Bugs: 1403
- Developpement
  - Ouvert à tous
  - Cycles court de 6 mois
  - Tres actif
- Stackalytics

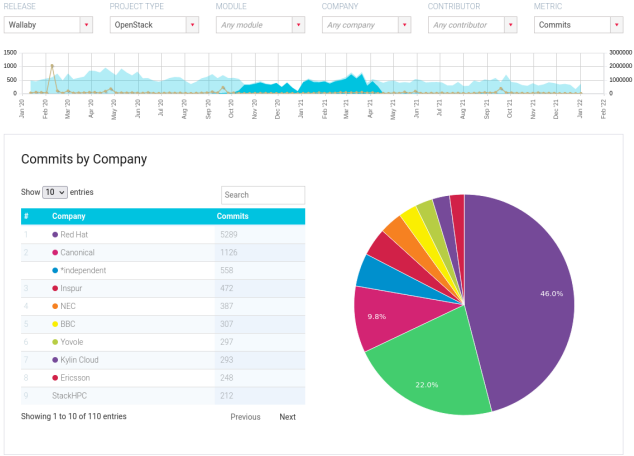

Figure 1: Contributeurs

Stack logicielle

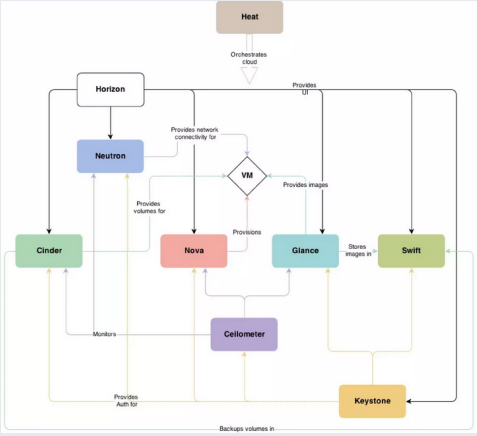

Figure 2: Services de base

Composants de base

- Nova : Calcul
- Cinder : Stockage bloc
- Neutron : Réseau (SDN)
- Glance : Images VM
- Keystone : Identité
- Horizon : UI web
- ...

Autres composants

- Ceilometer : Métrologie
- Heat : Orchestration
- Ironic : Bare-metal
- Designate : DNS
- Watcher : Service d'optimisation des ressources
- Octavia : Load balancer
- ...

Composants

- Découpé en services
- Bus à message : rabbitMQ, ...
- Base de donnée : MariaDB, MySql, ...
- API
  - Ports spécifiques
  - Python
  - REST

Nova

- Virtualisation : libvirt, KVM
- Instances éphémères
- Gabarits : flavor
- Glance : images
- Neutron : réseaux
- Cinder : stockage bloc, volumes
- Migration à chaud ou à froid
- ...

Neutron : Network as a Service

- IP flottantes
- Groupes de sécurité (firewall)
- Agents DHCP pour les instances
- OpenVSwitch, DRS,
- L2, L3
- HAProxy
- Namespaces réseau linux
- ...

Avantages / Inconveniants

- rapide, élastique
- libre et gratuit =! vmware
- facile à déployer : kolla
- fullvirtu versus conteneurs
- vrai cloud
- grosse communauté

Section 3

Packer

C'est quoi ?

- Un outil opensource de Hashicorp
- Images as code
- Construction d'images pour systemes ou conteneurs
- Multi-cloud : AWS, Docker, Openstack, ...
- Un langage de description
  - Packer $<$ 1.5.0 : json
  - Packer $>=$ 1.5.0 : json et HCL2 en beta
  - Packer $>=$ 1.7.0 : HCL2 et json en legacy
    - `packer hcl2_upgrade my-packer.json`

Modulaire

- un fichier tout en un
- plusieurs fichiers. Exemple :
  - sources.pkr.hcl
  - build.pkr.hcl
  - variables.pkr.hcl
  - data.pkr.hcl
  - ...

Le langage

Arguments et Blocs

```
# bloc
source "openstack" "mon_nom" {

    # Argument
    ssh_username = "ubuntu"
}
```

Commentaires

- Simple ligne
  - #
  - //
- Multi ligne
  - /* et */

Chaines de caractères multilignes

```
variable "longue_chaine" {
  type = "string"
  default = <<EOF
Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Sed feugiat mollis purus nec dignissim.
EOF
}
```

Blocs racines

- `source` : bloc réutilisable de configuration pour les `builder`
- `build` :
  - quel builder (docker, openstack, vagrant, AWS ...)
  - provisioners (shell, powershell, ...)
  - post-processors (checksum, compress)

Bloc packer (depuis v1.6.5) configurations

```
# Exemple pour l'utilisation d'openstack
packer {
  required_plugins {
    openstack = {
      version = ">= 1.1.0"
      source  = "github.com/hashicorp/openstack"
    }
  }
}
```

https://developer.hashicorp.com/packer/plugins

- Openstack
  - builder
- Docker
  - builder
  - post-processors:
    - Docker import
    - ...

Variables : définition

```
variable "test_name" {
  type    = string
  default = "default_varlue"
}
variables {
  var1 = "foo"
  var2 = "bar"
}
```

Arguments

- type
- default
- description
- validation : en très gros regex
- sensitive : sensible ?

Type simple

- string
- number
- bool

Type complexe

- list()
- set()
- map()
- object({ = , ... })
- tuple([, ...])

Appel d'une variable

```
var.test_name
```

Validation des variables

```
variable "external_net" {
  type    = string
  default = null

  validation {
    condition     = can(index(["dmz", "public"], var.external_net))
    error_message = "Only 'dmz' or 'public' networks allowed."
  }
}
```

Des provisioners

- Outils pour installer et configurer la machine après son démarrage.

Exemples

- file
- ansible-local
- ansible
- shell
- PowerShell
- ...

Des fonctions

- Chaines : format, split, ...
- Collections : index, element, sort, ...
- Nombres : min, max, ...
- Date : timestamp, ...
- Convertion de type : can
- ...

Section 4

Terraform

C'est quoi

- Infrastructure As Code
- Outil de Hashcorp
- Multi-cloud / conteneurs
- Gestion du cycle de vie d'une infrastructure
- Un langage de configuration de ressources
  - openstack
  - Cisco ACI
  - AWS
  - Et plein d'autres

Concepts

Provider

Plugin d'interaction avec les APIs des services (openstack, AWS, docker, ...)

Module

Un répertoire contenant des configurations et réutilisable

Ressources

Ce sont des blocs permettant de décrire les objets des infrastructures (VM, réseau, FW, IP, ...) qui seront gérés par Terraform.

Sources de données

Implémentées dans le provider, permet de récupérer des informations sur des objets non gérés par Terraform

Variables d'entrée

Des variables au format clef - valeur pour customiser les configurations

Valeurs de sortie

Permet de récupérer des informations sur les objets gérés par Terraform et de les afficher, ou de les enregistrer pour être utilisés par d'autres outils.

Import

Permet d'importer une ressource déjà présente

Un état

Conserve l'état des objets gérés par Terraform

Plan

Utilise l'état pour déterminer ce qui doit être modifié, créé, détruit, déplacé.

Mise en œuvre : apply

Applique le plan

Cycle de vie

- Init : Installe les providers et les plugins
  - –upgrade : en cas d'ajout, modification
- Plan : Calcul les modifications à appliquer
- Apply : Appliques les modifications
- Destroy : supprime des ressources

Le langage HCL

Configuration de Terraform

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    openstack = {
      source  = "terraform-provider-openstack/openstack"
      version = "~> 1.48.0"
    }
  }
}
```

Ressources

```
resource "type_resource" "mon_nom" {
  conf1 = "value"
  conf2 = 13
}
```

Données

```
data "type_resource" "mon_nom" {
  conf1 = "value"
  conf2 = 13
}
```

Providers

```
provider "openstack" {
  tenant_name = "default"
  username = "admin"
  ...
}
```

Variables

```
variable "http_ports" {
  type = list(object({
    http = number
    https = number
    protocol = string
  }))
  default = [
    {
      http = 80
      https = 443
      protocol = "tcp"
    }
  ]
  validation {
    condition     = var.http_ports.http < var.http_ports.https
    error_message = "port http doit etre plus petit que https"
  }
}
```

Méta arguments

- providers
- depends_on
- count
- for_each

Fonctions

https://developer.hashicorp.com/terraform/language/functions

Autre

- check
- import
- local
- Output
- Modules

Section 5

Ansible

C'est quoi

Moteur d'automatisation qui passe par une connexion `ssh` (ou pas).

- Provisionning : VMs, baremetal, réseau ...
- Configuration
  - Gestion d'utilisateurs
  - Politique de sécurité
  - Services
- Déploiement d'applications : devops ...
- Déploiement d'infrastructures : openstack, kubernetes ...

Concepts

- Nœud de contrôle
- Les nœuds (Hosts)
- Inventaire
- Playbooks
  - Plays
    - Roles
    - Tasks
    - Handlers (exec si tache précédente : 'changed')
- Modules
- Plugins
  - connexion
  - filtres
  - callbacks
- Collections
  - Ansible Galaxy

Modules

Des programmes qui sont poussés sur les `hosts` depuis la machine de contrôle.

- Ansible.Builtin
  - user
  - add_host
  - file
  - iptables
  - debug
- Collections
  - openstack.cloud
  - ...

Commandes

- ansible
  - ansible localhost -m ping
  - ansible localhost -a “uname -a” -u john
  - ansible localhost -a “cat /etc/shadow” -u ubuntu –become
  - ansible localhost -m ansible.builtin.setup
  - ansible localhost -m ansible.builtin.setup -a “filter=ansible_all_ipv4_addresses”
  - ansible localhost -m ansible.builtin.setup -a “filter=*ip*”
- ansible-playbook
  - ansible-playbook myplay.yml
  - ansible-playbook [options] myplay.yml
- ansible-vault
- ansible-config
  - ansible-config init –disabled -t all > ansible.cfg

Inventaire

- Définition des `hosts` de votre infrastructure
- Plusieurs formats

INI

```
[compute]
compute-01.exemple.fr
compute-02.exemple.fr

[controller]
controller-0[1:3].exemple.fr
```

Yaml

```
compute:
  hosts:
    compute-01.exemple.fr
    compute-02.exemple.fr
controller:
  hosts:
    controller-0[1:3].exemple.fr
```

Inventaires dynamiques

- scripts python, hotes statiques, plugins, ...

Utilisation

- 'ansible-playbook myplay.yml -i inv01.yml -i inv02.yml
- `ansible-playbook myplay.yml -i inventory/`
  - avec `inventory` un répertoire qui contient des inventaires

Variables

Il est possible d'ajouter des variables à l'inventaire.

- Aleatoires
  - `maVar=42`
- De connexion
  - `ansible_connection=[local|ssh]`
  - `ansible_user=ubuntu`

INI

```
[web]
serveur-[01:03].example.fr     ansible_user=root mon_interface=eno1
```

Variables de groupes

INI

```
[dev]
host1
host2

[dev:vars]
ansible_user=debian
```

YAML

```
dev:
  hosts:
    host1:
    host2:
  vars:
    ansible_user=debian
```

Héritage / hiérarchie

INI

```
[test1]
host1

[test2]
host2

[test:children]
test1
test2

[test:vars]
...
```

YAML

```
all:
  children:
    test:
      children:
        test1:
          hosts:
            host1
        test2:
          hosts:
            hosts2
      vars:
        ...
```

Facts

Variables relatives à l'hôte

Dans un playbook

```
- name: Afficher toutes les variables fact
  ansible.builtin.debug:
    var: ansible_facts
```

CLI

```
ansible <hostname> -m ansible.builtin.setup
```

Désactiver

```
- hosts: whatever
  gather_facts: false
```

Ajouter

- `ansible.builtin.set_fact`
- `/etc/ansible/facts.d`
- `fact_path`
- `ansible all -m setup -a "filter=ansible_local"`

Types et manipulation

Manipulation de données plus ou mois complexes.

- int
- float
- string
- list
- dict

Afficher le type d'une donnée

```
{{ myvar | type_debug }}
```

Forcer le type donnée

```
- debug:
  msg: test
  when: some_string_value | bool
```

Debug

```
tasks:
  - name: My debug
    debug:
      msg: "{{ var | list }}"
```

Dictionnaire

```
user:
  nom: john
  prenom: doe
```

Liste

```
- key: nom
  value: john
- key: prenom
  value: doe
```

Conversion

```
{{ user | dict2items }}
```

Valeur par défaut

```
{{ ma_variable | default("toto") }}
```

Variable facultative

```
{{ ma_variable | default(omit) }}
```

Variable obligatoire

```
{{ ma_variable | mandatory }}
```

Filtres

- listes : intersect, union, uniq, min, ...
- random
- hash, password_hash, ...
- math: log, pow, root, ...
- json_query
- ...

Templating : ansible.builtin.template

```
---
- name:
  hosts: all
  tasks:
  - name: Exemple jinja2
    ansible.builtin.template:
       src: fichier.conf.j2
       dest: /tmp/fichier.conf
```

fichier.j2

```
Adresse IP {{ ansible_facts['ipv4'] }}
```

Connection aux hôtes

- ansible_host
- ansible_port
- ansible_user
- ansible_password
- ansible_ssh_extra_args
- ansible_become
- ansible_connection
  - ssh : smart, ssh, paramiko
  - autre :
    - local
    - docker
    - ...
- ...

Roles

Créer un rôle

```
ansible-galaxy init [ROLE NAME]
```

Exemple

- test-role
  - defaults
    - main.yml
  - files
  - handlers
    - main.yml
  - meta
    - main.yml
  - tasks
    - main.yml
  - templates
  - tests

Debug : inline

Debug inline

- Tout le playbook

```
ansible-playbook main.yml --check
```

- Une tache en particulier dans le playbook

```
check_mode: yes
```

- Option `--check` et `ignore_errors`: ignorer les erreurs

```
ignore_errors: '{{ ansible_check_mode }}'
```

- Option –diff : avertit des modifications
- Dans le playbook

```
diff: no
```

Une tache

```
- name: Ma tache
  command: ls /tmp
  debugger: on_failed
```

Toutes les taches

```
- hosts: all
  debugger: on_failed
  tasks:
    - name: Ma tache
      command: ls /tmp
```

debugger

- always
- never
- on_failed
- on_unreachable
- on_skipped

Mode debug

- `p task/task_vars/host/result`: affiche des informations utilisées pendant l'exécution par votre tâche module (p = print).
- `task.args[key] = value`: met à jour l'argument du module.
- `task_vars[key] = value`: met à jour les variables de votre playbook.
- `update_task`: si modif des task_vars
- `redo`: exécutez à nouveau la tâche.
- `continue`: passe à la tâche suivante.
- `quit`

Verbose CLI

4 niveaux de verbosité :

- `-v`
- `-vv`
- `-vvv`
- `-vvvv`

Section 6

Kolla

Openstack pour les admins

C'est quoi ?

Un projet qui permet de déployer et mettre à jour une plateforme Openstack, simplement, rapidement grâce à des outils simples et performants : Ansible et Docker

Des images Docker

- Les principaux projets (38 actuellement)
- Une image par *microservice*, ex: neutron_server, neutron_dhcp_agent, neutron_l3_agent, ...
- Image de base au choix (CentOS, Debian, RHEL, Ubuntu)
- Disponible en mode *binaire* (apt/dnf) ou *source* (pip depuis les sources)
- Personnalisation possible

GRICAD

Exemple de configuration

nova-compute

```
[DEFAULT]
vnc_keymap=fr
[libvirt]
cpu_mode = custom
cpu_model = Broadwell-noTSX-IBRS
images_rbd_pool=ephemeral-vms
images_type=rbd
images_rbd_ceph_conf=/etc/ceph/ceph.conf
rbd_user=nova
disk_cachemodes='"network=writeback"'
{% if gpu is defined %}
[devices]
enabled_vgpu_types = {{ gpu }}
{% endif %}
```

Une commande pour tout faire

kolla-ansible

- kolla-ansible pre-checks
- kolla-ansible pull
- kolla-ansible deploy
- kolla-ansible reconfigure
- kolla-ansible upgrade
- ...

En conclusion

- Documentation pas toujours présente
- Connaissance de Docker et d'Ansible indispensable
- Lecture du code Ansible

- Configuration centralisée et facilement versionable
- Dépendances minimales
- Procédure de mise à jour facilitées
- Facilité de personnalisation
- Reproductibilité = test de déploiement facile
- Ansible/Jinga2 facile à lire
- Projet actif et qui évolue
- Montée/saut de version Openstack simple

Pour aller plus loin

- 2022 : Kolla
  - https://wiki.openstack.org/wiki/Kolla
- 2019 : Nova - Un nuage arc-en-ciel au-dessus des Alpes
  - https://hal.archives-ouvertes.fr/hal-02387868
- 2017 : Production Deployment Tools for IaaSes: an Overall Model and Survey
  - https://doi.org/10.1109/FiCloud.2017.51