SUHANA Sutra 51701858
FIDALGO Alex 71600135

PROGRAMMATION COMPARÉE

TERRAFORM 

vs

PUPPET 

vs

CHEF 

Sous la direction de Guatto Adrien, Letouzey Pierre & Férée Hugo

SOMMAIRE

1 Introduction

Pour commencer, ces trois logiciels ont un but commun : ils permettent de faire de **l'infrastructure en tant que code**. Néamoins, il est possible de les comparer en explorant certains de leurs aspects qui se révèlent parfois très différents.
Principalement utilisés dans le monde de l'industrie informatique, ils ont connu un essor important depuis l'utilisation répandue de l'informatique dans le nuage (cloud computing).
Ils permettent notamment d'automatiser certains processus qui autrefrois nécessitait des interventions mannuelles, de réduire les coûts de déploiement où encore d'étendre plus simplement des infrastructures informatiques.
Puppet et **Chef**, tous deux écrits en Ruby sont respectivement sortis en 2005 et 2009.
Terraform quant à lui est sorti en 2014 et est écrit en Golang.

2 Infrastructure en tant que code

L'**infrastructure en tant que code** est le processus qui rassemble ces trois outils. Il consiste à gérer où provisionner des infrastructures informatiques à l'aide de lignes de codes.
Avant l'apparition de ces outils, il existait bien entendu d'autres moyens permettant de gérer une infrastructure informatique, mais ceux-ci étaient manuels où bien "semi-automatiques".
L'**automatisation** est l'avantage le plus significatif qui démarque ces outils.
L'infrastructure en tant que code présente bien entendu d'autres avantages : la **diminution potentielle du nombre d'erreurs** (moins de scripts à écrire qu'auparavant donc moins de sources d'erreurs), une **infrastructure plus cohérente** (les différents scripts de configurations sont rassemblés/partagés), la possibilité de **versionner** ces scripts de configurations (permettant un retour à une version antérieure en cas de problème).
Ce processus peut-être utilisé dans deux catégories que sont le **provisionnement** et la **gestion de configuration**. Celà constitue la première différence entre ces trois outils.

3 Provisionnement et gestion de configuration

Tout d'abord, le **provisionnement** est la première étape lorsque l'on parle d'infrastructure en tant que code.
C'est le processus qui va permettre de mettre en place différentes infrastructures informatiques telles que des **serveurs**, des **bases de données**, des **services** où encore des interfaces **réseau**, de façon automatisée.

Ce processus constitue la fonctionnalité principale de **Terraform** et est généralement suivi du processus de **gestion de configuration** qui quant à lui est laissé aux deux autres outils : **Puppet** et **Chef**.
En effet, ces deux derniers permettent, une fois la mise en place effectuée, de maintenir où de gérer une où plusieurs infrastructures : **installation de paquets**, **correction d'erreurs**, **retours en arrière**.
Ces deux aspects ne sont pas mutuellement exclusifs. En effet, **Terraform** permet de faire de la **gestion de configuration** mais de façon partielle. Réciproquement, **Puppet** et **Chef** permettent dans certains cas de provisionner des infrastructures mais de façon plus limité que **Terraform**.
HashiCorp, entreprise qui à mis **Terraform** en production mentionne notamment : " Les outils de gestion de configuration installent et maintiennent des logiciels sur des machines déjà existantes. Terraform n'est pas un outil de gestion de configuration, et autorise des outils existants de gestion de configuration à se concentrer sur leurs forces. "

4 Procédural vs déclaratif

Terraform, **Puppet** et **Chef** fonctionnent tous les trois via des scripts appelés respectivement des **scripts Terraform**, des **manifests** et des **recettes**.
Les scripts **Terraform** peuvent être écrits en **HCL** (langage spécifique à Terraform) où bien en **Json**. **Puppet** impose quant à lui d'utiliser un langage spécifique **Puppet language** et **Chef** impose **Ruby** comme langage de script. Hormis une différence plutôt habituelle d'appelation où de langage de script, la façon d'analyser ces scripts constitue la différence notable entre ces trois logiciels.
En effet, **Chef** utilise un langage de script dit **procédural** : afin d'atteindre l'état final voulu, il est nécéssaire de spécifier chacunes des étapes (comme dans un programme **C** par exemple).
Terraform et **Puppet** quant à eux, utilise un langage dit **déclaratif** : il suffit de spécifier l'état final voulu et le logiciel fera en sorte de nous fournir l'état souhaité.
Ces deux approches présente des avantages et des inconvénients. Un langage **procédural** permet une meilleure **expréssivité**. Dans certains cas, l'état final peut s'avérer être complexe à décrire de façon déclarative alors que de façon procédurale, une énumeration d'étapes permet d'être plus précis.
Par contre, un script **déclaratif Terraform** où **Puppet** est plus simple à modifier et à maintenir. En réalité, ces deux outils ont une connaissance de l'**état courant** de l'infrastructure, ce qui permet à l'utilisateur d'effectuer une modification sans avoir à se soucier des états précédents.

5 Infrastructure : Mutable vs Immutable

Ces logiciels sont également différents d'un point de vue **infrastructure**. Quand **Terraform** dispose d'une infrastructure dite **immutable**, l'infrastructure de **Puppet** et **Chef** est quant à elle **mutable**.
Pour illustrer cette différence, imaginons la situation suivante : nous disposons de 10 machines que nous voulons mettre à jour. Dans le cas de **Terraform**, ces 10 machines seront écrasées et re-crées en disposant de la nouvelle version voulue. Par contre, **Puppet** et **Chef** modifieront simplement les 10 machines en effectuant une mise à jour sur chacunes d'elles.
Ceci peut amener un problème appelé **dérive de configuration**.
Le but de ces logiciels est d'effectuer des modifications sur une application existente au cours du temps. Néanmoins, une infrastructure mutable autorise une différentiation sur des infrastructures censées fonctionner de la même façon. En effet, à un certain moment, si deux infrastructures sont modifiées d'une façon différente, il se peut que des bugs apparaissent lors de la mise en production (où même avant).

6 Client vs Client / Serveur & Agent vs Sans agent

Nous allons maintenant aborder l'aspect communication de ces logiciels.
Chef et **Puppet** fonctionnent tous deux sur le même modèle de communication. Ils possèdent tous les deux un serveur central (où plusieurs en cas de problème) qui va servir de point de **convergence**.
Les étapes sont les suivantes :

- → L'utilisateur écrit ses scripts de configuration, potentiellement à l'aide de librairie externes.
- → Il pousse ses scripts sur le serveur central.
- → Les différents noeuds cibles font des requêtes périodiques au serveur central afin de prendre en compte des modifications potentielles.

De plus, **Puppet** et **Chef** nécéssite pour fonctionner une installation au préalable de leur client sur les noeuds cibles. Chacun des noeuds cibles possèble donc le client du logiciel, c'est lui qui va permettre une communication avec le serveur central.

Terraform est différent de ces deux derniers. Il n'est pas seulement client bien-sûr car celà n'aurait pas de sens, mais ne possède pas de serveur central comme **Puppet** et **Chef**. En fait, il communique via des **interfaces de programmation applicatives** (API), qui feront en quelque sorte office de serveur central. De plus, les fournisseurs de ces API (AWS, Kubernetes, VMWARE etc ...) s'occuperont d'installer tout ce qui est nécessaire sur les machines cibles.

7 Communauté

Pour terminer, lorsque l'on s'intéresse à un logiciel, la communauté autour de celui-ci est quelque chose d'important à prendre en compte.
Voici donc quelques chiffres concernant **Terraform**, **Puppet** et **Chef** :

	Étoiles sur Github	**Contributeurs**	**Librairies**
Puppet	6 500	568	7206 (catalogues)
Chef	6 800	632	3996 (livres de cuisines)
Terraform	31 300	1588	8239 (modules)

8 Conclusion

Bien que **Terraform** soit orienté **provisionnement** et que **Chef** et **Puppet** soient orientés **gestion de configuration**, ils aspirent tous les trois à un but commun : **faciliter** et **automatiser** des processus qui étaient autrefois manuels et répétitifs.

9 Questions et suggestions

9.1 Questions

→ Question : **Sawssen** : **Terraform** peut fonctionner de paire avec **Puppet** et **Chef**, pourquoi **Chef** et **Puppet** ne fonctionnent pas ensemble ?

→ Réponse : **Terraform** est axé provisionnement, c'est à dire mise en place des infrastructures. **Puppet** et **Chef** sont tous les deux responsables de la partie **gestion de configuration** et font donc parti de la même "famille" d'outils. C'est pourquoi ont choisit généralement soit l'un soit l'autre.

→ Question : **Mr. Guatto** : Savez-vous comment **Terraform** maintien l'état d'une infrastructure ?

→ Réponse : Non, mais je suppose que **Terraform** interroge l'API afin d'avoir des informations sur l'état de l'infrastructure.

→ Correction : En fait, **Terraform** utilise des fichiers de configuration en local qui enregistre toutes les informations sur l'état courant des différentes infrastructures.

→ Question : **Mr. Guatto** : Savez-vous comment peut fonctionner la gestion des machines de cette salle de TP ?

→ Réponse : Non.

9.2 Suggestions

→ **Mr. Guatto** : Il aurait été pertinent de mentionner ce qui existait avant ces logiciels.

→ **Mr. Guatto** : Présenter d'abord à quoi servent ces logiciels avant de les présenter en tant que tel (langage de programmation etc..).

Ressources

→ https://www.redhat.com/fr/topics/automation/what-is-infrastructure-as-code-iac#approches-déclarative-et-impérative

→ https://www.redhat.com/fr/topics/automation/what-is-provisioning

→ https://blog.devgenius.io/provisioning-vs-configuration-management-with-terraform-4bf07b9c79db

→ https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c#c7ba

→ https://www.terraform.io

→ https://www.chef.io

→ https://martezreed.medium.com/terraform-is-not-ansible-or-puppet-f5a222b8460

→ https://www.oreilly.com/library/view/terraform-up-and/9781491977071/ch01.html

→ https://www.youtube.com/watch?v=l5k1ai_GBDE

→ https://www.youtube.com/watch?v=llcjg1R0DdM&t=242s

→ https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code

→ https://logz.io/blog/chef-vs-puppet/