ANNÉE PASSERELLE EN INGÉNIERIE

Chapitre 21 — Automatisation

Introduction à la Téléinformatique

Vincent Audergon · HEIA-FR

Année académique 2026-2027

Plan du chapitre

Objectifs du chapitre

HEIA-FR
HTA-FR

À l'issue de ce chapitre, vous saurez :

- **décrire** les enjeux des réseaux actuels et l'importance de l'automatisation ;
- **expliquer** le principe d'une NFV ;
- **expliquer** le principe de SDN et du protocole OpenFlow ;
- **expliquer** le principe d'IaC (Infrastructure as Code) ;
- **lister** quelques outils d'automatisation.

SECTION
01

Pourquoi automatiser ?

Automatiser, pour quels gains ?

DÉFINITION · AUTOMATISATION

Utiliser des logiciels pour exécuter des tâches et processus informatiques répétitifs **presque sans intervention humaine**. (Red Hat)

- **Fiabilité** : moins d'erreurs humaines
- **Scalabilité** : gérer un grand nombre de systèmes
- **Efficacité** : gain de temps
- **Réactivité** : réagir vite aux incidents
- **Reproductibilité** : mêmes étapes à chaque exécution
- **Sécurité** : politiques appliquées de façon cohérente

Automatiser, pour quels gains ?

ASTUCE

Un processus mérite d'être automatisé s'il est **répétitif**, sujet aux **erreurs humaines**, et prend un **temps significatif**. S'il faut plus de temps à automatiser qu'à exécuter manuellement, mieux vaut s'abstenir.

? QUESTION 1

Une tâche ne prend que deux minutes une fois par an. Vaut-il la peine de l'automatiser ?

> **RÉPONSE**
> Probablement **non** : le temps nécessaire pour développer et maintenir l'automatisation dépasse largement le temps gagné sur une tâche aussi rare et courte.

Le rayon d'impact (blast radius)

Automatiser **ne crée pas** le risque : elle le **met à l'échelle**. Une commande exécutée manuellement affecte une machine ; automatisée, elle peut affecter tout un parc.

- **Canary deployment** : déployer sur un petit sous-ensemble d'abord.
- **Double contrôle** : validation par un second opérateur.
- **Dry-run** : simuler sans exécuter réellement.
- **Moindre privilège** : limiter les droits des processus automatisés.
- **Rollback** : revenir rapidement à un état stable.

Exemple de rayon d’impact : AWS S3 (2017)

EXEMPLE · CAS D'ÉCOLE CHEZ AWS (2017)

Une commande d’automatisation mal paramétrée a supprimé bien plus de serveurs que prévu, provoquant une panne majeure touchant Github, Gitlab, Slack et Trello.

InfoQ Homepage > News > A Human Error Took Down AWS S3 US-EAST-1

CLOUD

A Human Error Took Down AWS S3 US-EAST-1

https://www.infoq.com/news/2017/03/aws-s3-disruption/

? QUESTION 2

Quelle est la différence entre un dry-run et un rollback ?

RÉPONSE

Un **dry-run** simule l'exécution d'une commande sans la réaliser, pour vérifier les effets avant de l'exécuter réellement. Un **rollback** est une action corrective qui ramène un système à un état stable après une exécution problématique.

SECTION
02 —

Automatisation, outils et paradigmes

Deux façons de décrire une tâche

HEIA-FR
HTA-FR

Impératif

Décrire **chaque étape**, dans l'ordre : « prends un steak, mets-le sur le grill, attends 10 minutes, retourne-le… »

- Comme une recette de cuisine
- Scripts, langages de programmation impératifs (Java, C, Python, etc.)
- Automatisation : Bash, PowerShell, Python, etc.

Déclaratif

Décrire **l'état final souhaité** : « je veux un steak bien cuit avec des frites », l'outil détermine comment y arriver.

- Comme un menu de restaurant
- Langages déclaratifs (HTML, CSS, Prolog, etc.)
- Automatisation : Ansible, Terraform, OpenTofu, etc.

Familles d'outils d'automatisation

- **Scripts** (Bash, Python, PowerShell) : impératifs, tâches simples et séquentielles.
- **Gestion de configuration** (Ansible, Puppet, Chef) : surtout déclaratifs, gèrent l'état de systèmes existants.
- **IaC** (Terraform, OpenTofu) : déclaratifs, font naître l'infrastructure à la demande.
- **SDN** (OpenFlow, ONOS) : Automatisation de réseaux. Séparent le plan de contrôle du plan de données.

Familles d'outils d'automatisation

REMARQUE

Les outils de gestion de configuration sont **hybrides** : principalement déclaratifs, avec parfois des tâches impératives ponctuelles.

? QUESTION 3

Quelle est la différence essentielle entre un outil de **gestion de configuration** et un outil d'**IaC** ?

> **RÉPONSE**
> L'IaC (Terraform) fait **naître** l'infrastructure elle-même (serveurs, réseaux) à la demande, alors que la gestion de configuration (Ansible) se contente de gérer l'**état** de systèmes déjà existants.

SECTION
03

Terraform et Ansible

Terraform, l'infrastructure en code

Terraform (HashiCorp) est un outil d'**Infrastructure as Code** : il crée, modifie ou supprime des ressources (VM, routeurs, switches virtuels…) pour atteindre un état décrit dans un fichier de configuration.

- Système de « providers » : chaque provider (fournisseur) est responsable de la création et gestion des ressources pour un type d'infrastructure.
  - Google Cloud, AWS, Azure, VMware, OpenStack, Kubernetes, etc.
- Déclaratif : on décrit l'état final souhaité, Terraform détermine les actions à entreprendre pour y parvenir.

Terraform : exemple de configuration

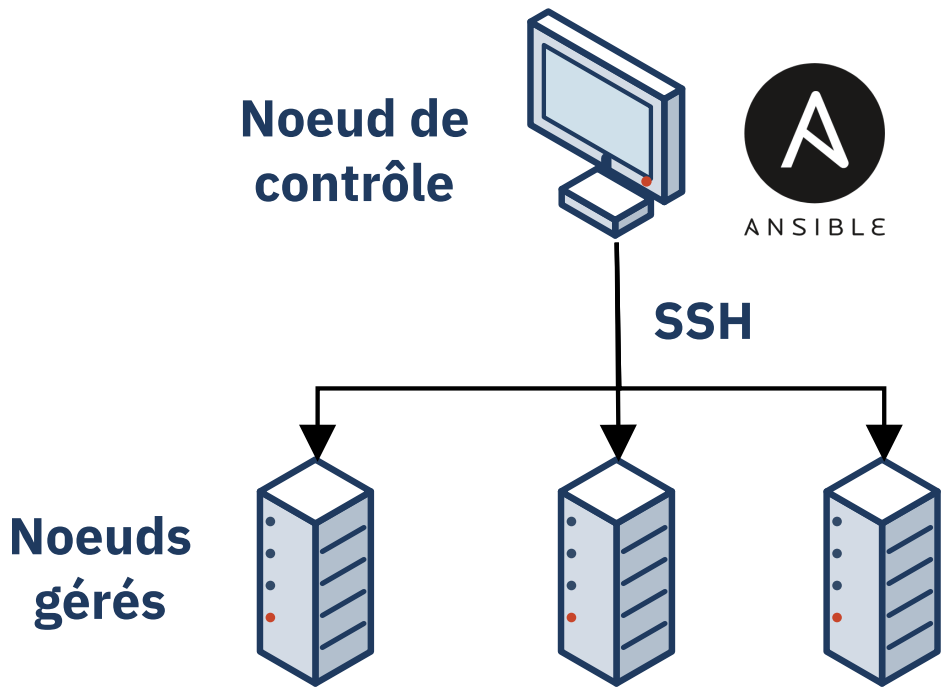

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-xxxxxxxx"
  instance_type = "t2.micro"
}
```

Ansible, control node et managed nodes

- Hybride : principalement déclaratif, impératif ponctuellement
- Gestion de configuration : installe des logiciels, configure des services, etc.
- Système de « notebooks » : chaque playbook décrit un état final souhaité pour un ou plusieurs systèmes.
- Propriété d'**idempotence** : un playbook peut être exécuté plusieurs fois sans provoquer de changement supplémentaire si l'état souhaité est déjà atteint. (non garantit par Ansible, à la charge de l'auteur du playbook)
- Deux entités :
  - **Control node** : exécute les commandes et playbooks Ansible.
  - **Managed nodes** : systèmes gérés par Ansible, accessibles via SSH.

Ansible : architecture



Un playbook Ansible Idempotent


```yaml
---
- name: Installer et configurer un serveur web
  hosts: serveurs_web
  become: true
  tasks:
    - name: Installer Nginx
      ansible.builtin.apt:
        name: nginx
        state: present
    - name: Démarrer Nginx
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true
```

? QUESTION 4

Vous exécutez deux fois de suite le même playbook Ansible ci-dessus, sans rien changer entre-temps. Que se passe-t-il la deuxième fois ?

RÉPONSE

Rien de nouveau : Nginx est déjà installé et démarré, donc Ansible ne modifie rien. C’est le principe d’**idempotence** : un pilier des outils déclaratifs.

SECTION
04 —

NFV et SDN

La virtualisation des fonctions réseau (NFV)

DÉFINITION · NFV

Network Function Virtualization : Virtualiser des fonctions réseau (pare-feu, routeur, load balancer, IDS/IPS) traditionnellement exécutées sur du matériel dédié, pour les exécuter en logiciel.

- + économie de coûts, flexibilité, scalabilité
- + tests jetables, automatisation facilitée
- – performance parfois inférieure au matériel dédié
- – surface d'attaque et complexité accrues

La virtualisation des fonctions réseau (NFV)

HEIA-FR
HTA-FR

EXEMPLE · PARE-FEUX VIRTUELS

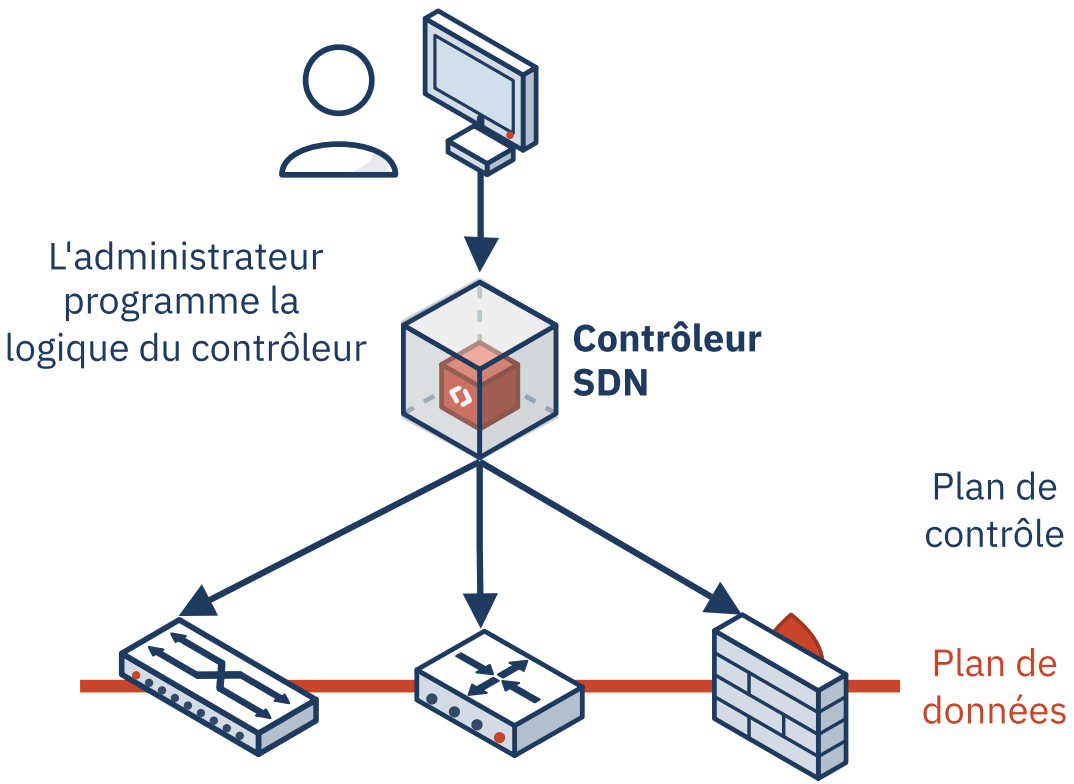
pfSense, OPNsense, VyOS : des NFV populaires offrant les fonctionnalités d’un pare-feu matériel, avec la flexibilité du logiciel.

SDN : séparer contrôle et données

DÉFINITION · SDN (SOFTWARE DEFINED NETWORKING)

Séparer le **plan de contrôle** (décisions de routage, centralisées) du **plan de données** (exécution par les équipements réseau).

- **Contrôleur** SDN : communique les ordres via des protocoles SDN (comme **OpenFlow**)



? QUESTION 5

Dans une architecture SDN, un switch reçoit un paquet pour lequel aucune règle locale n'existe. Que se passe-t-il ?

RÉPONSE

Le switch interroge le **contrôleur SDN centralisé**, qui décide comment traiter le paquet et peut ajouter une nouvelle règle dans la **table de flux** de l'équipement.

OpenFlow : les tables de flux

OpenFlow définit comment un contrôleur SDN communique avec les équipements, via des **tables de flux** : des règles (priorité + critères + action).

Priorité	Correspond à	Action
30	MAC destination = 00:11:22:33:44:55	Port 1
20	IP destination = 10.2.0.0/24	Port 2
10	Port TCP destination = 80	Port 3

Trois modes de configuration : **réactif** (règle créée à la demande), **proactif** (règles préconfigurées) ou **hybride**.

Ryu : un framework OpenFlow en Python


```
self.add_flow(dp, priority=30,
              match=parser.OFPMatch(
                eth_dst="00:11:22:33:44:55"
              ),
              out_port=1)
self.add_flow(dp, priority=20,
              match=parser.OFPMatch(
                eth_type=0x0800,
                ipv4_dst=(
                  "10.2.0.0",
                  "255.255.255.0"
              )),
              out_port=2)
self.add_flow(dp, priority=10,
              match=parser.OFPMatch(
                eth_type=0x0800,
                ip_proto=6,
                tcp_dst=80
              ),
              out_port=3)
```

? QUESTION 6

1. Que font les trois instructions `self.add_flow` ?
2. Que représente l'argument `eth_type=0x0800` dans les deux dernières instructions ? Pourquoi n'est-il pas présent dans la première instruction ?
3. Que représente l'argument `ip_proto=6` dans la troisième instruction ?

RÉPONSE

1. Les trois instructions `self.add_flow` ajoutent chacune une règle dans la table de flux d'un switch OpenFlow. Chaque règle a une priorité, un critère de correspondance (match) et une action (out_port) qui indique vers quel port le paquet doit être envoyé si les critères sont remplis.
2. L'argument `eth_type=0x0800` indique que la règle s'applique aux paquets IPv4, 0x0800 étant le code hexadécimal (EtherType) pour le protocole IPv4. Il n'est pas présent dans la première instruction car celle-ci n'inspecte aucun champ de niveau supérieur (IP ou TCP) : elle se base uniquement sur l'adresse MAC de destination, un champ de couche 2. Il devient en revanche obligatoire dès qu'on veut matcher un champ IP.
3. L'argument `ip_proto=6` indique que la règle s'applique aux paquets TCP, 6 étant le numéro de protocole pour TCP (rien à voir avec IPv6 !).

Synthèse

CE QU'IL FAUT RETENIR

- L’automatisation gagne en **fiabilité, scalabilité, efficacité** — mais amplifie aussi le **rayon d’impact** des erreurs.
- **Impératif** (étapes) vs **déclaratif** (état final) : scripts, gestion de configuration (Ansible), IaC (Terraform).
- **NFV** : fonctions réseau virtualisées en logiciel.
- **SDN** : plan de contrôle centralisé, plan de données exécuté par les équipements, communication via **OpenFlow**.

Des questions ?

Sources

Les sources utilisées pour ce cours (normes, documentation officielle, articles, ouvrages de référence) sont citées et listées intégralement dans la **bibliographie du support de cours écrit**, disponible sur le site du cours :

https://api-teleinfo.isc.heia-fr.ch