ANNÉE 2023/2024

TP Déploiements automatisés des configurations

NET4101

Ingénieur généraliste, 2ème année

Rédacteurs

Antoine Lavignotte
Directeur d'études
antoine.lavignotte@telecom-sudparis.eu

Rémy Grünblatt
Maître de Conférences
remy.grunblatt@telecom-sudparis.eu

Équipe enseignante

Andréa Araldo
Maître de Conférences
andrea.araldo@telecom-sudparis.eu

Laurent Bernard
Directeur d'études
laurent.bernard@telecom-sudparis.eu

Franck Gillet
Ingénieur R&D - Plateforme
franck.gillet@telecom-sudparis.eu

Jehan Procaccia
Ingénieur systèmes et réseaux
jehan.procaccia@imtbs-tsp.eu

Objectifs de ce TP et compétences à acquérir

— Comprendre les évolutions réseaux qui amènent au développement de leur programmabilité,
— Connaître les quelques solutions de management de configurations et d'automatisation les plus utilisées sur le marché actuellement,
— Comprendre le fonctionnement de base d' Ansible,
— Être capable de configurer une première architecture réseau simple grâce à Ansible.

1 Vers une programmabilité des réseaux

1.1 Command-Line Interface (CLI)

Il existe de nombreuses façons de se connecter à un réseau et de le gérer. La méthode la plus couramment utilisée au cours des 30 dernières années est l'interface de ligne de commande (CLI). Toutefois, comme presque tout le reste, l'interface CLI présente des avantages et des inconvénients. L'un des défauts les plus flagrants et les plus importants de l'utilisation de l'interface CLI pour gérer un réseau est sans doute la mauvaise configuration. Les entreprises modifient souvent leurs environnements réseau, et certains de ces changements peuvent être extrêmement complexes. Lorsque les entreprises ont une complexité accrue de leurs réseaux, le coût d'une défaillance peut être très élevé en raison du temps supplémentaire nécessaire pour résoudre les problèmes dans un réseau complexe. Cependant, la défaillance d'un réseau ne signifie pas nécessairement qu'un logiciel ou un composant matériel est en cause. La majorité des pannes de réseau sont causées par des êtres humains. De nombreuses pannes surviennent en raison de configurations erronées dues à un manque de compréhension du réseau. Bien que toutes les pannes ou défaillances ne puissent pas être évitées, il existe des outils qui peuvent aider à réduire le nombre de pannes causées par une erreur humaine due à une mauvaise configuration de l'interface CLI. Il existe maintenant des méthodes de déploiement des configurations réseaux via la programmabilité.

1.2 De nombreux outils pour les réseaux

Il existe de nombreux outils capables de gérer un déploiement automatisés des configurations réseaux. On peut les classer en deux grandes catégories :
— Les outils d'automatisation basés sur des agents (Puppet, Chef, SaltStack)
— Les outils sans agent (Ansible, Puppet Bolt, SaltStack SSH)

De nombreuses organisations sont confrontées à des problèmes informatiques et à un taux de rotation élevé, et on demande aux ingénieurs réseau de faire plus avec moins. L'utilisation de certains des outils listés ci-dessus peuvent contribuer à alléger la pression exercée sur le personnel informatique en le déchargeant de certaines des tâches les plus fastidieuses, chronophages et répétitives. Un opérateur réseau peut alors se concentrer davantage sur les responsabilités critiques de la mission, telles que la conception du réseau et la planification de la croissance. La majorité de ces outils fonctionnent de manière très similaire. Le tableau 1.2 ci-dessous présente une comparaison de haut niveau des outils abordés dans ce chapitre.

Facteur	Puppet	Chef	Ansible	SaltStack
Architecture	"Puppet masters" et "Puppet agents"	"Chef server" et "Chef clients"	"Control station" et "remote hosts"	"Salt master" et "Minions"
Language	Puppet DSL	Ruby DSL	YAML	YAML
Terminologie	"Modules" et "Manifests"	"Cookbooks" et "Recipes"	"Playbooks" et "Plays"	"Pillars" et "Grains"
Support pour de grands déploiements	Oui	Oui	Oui	Oui
Version sans agent	Puppet Bolt	/	Oui	Salt SSH

TABLE 1 – Comparaison des outils de management des configurations et d'automatisation

Les facteurs les plus importants dans le choix d'un outil sont la façon dont ils sont utilisés et les compétences du personnel opérationnel qui les adopte. Par exemple, si une équipe maîtrise parfaitement Ruby, il peut être judicieux de se tourner vers Chef. En revanche, si l'équipe est très à l'aise avec la ligne de commande, Ansible ou SaltStack peuvent convenir. Le meilleur outil pour le travail dépend de l'utilisateur, et le choix nécessite une compréhension approfondie des différences entre les outils et une connaissance solide de ce avec quoi l'équipe opérationnelle est à l'aise et qui jouera sur ses points forts.

2 Ansible

Ansible est une plateforme open source d'automatisation et de gestion de configuration largement utilisée dans le domaine de l'informatique. Elle permet aux administrateurs système, réseaux et aux ingénieurs DevOps de simplifier et d'automatiser un large éventail de tâches liées à la gestion des serveurs, des réseaux, des applications et des infrastructures, en particulier (mais pas uniquement) les infrastructures réseaux.

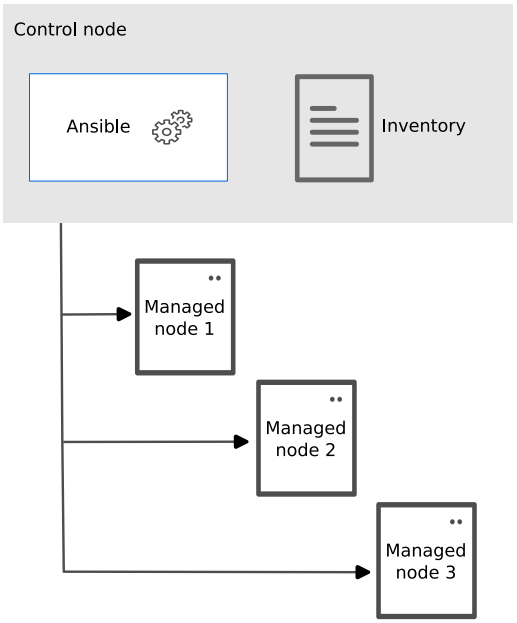

FIGURE 1 – Synoptique du fonctionnement d'Ansible. Source : https://docs.ansible.com/

Agentless Ansible se distingue par son caractère « sans agent », éliminant ainsi la nécessité d'installer des agents sur les machines cibles à administrer (autres que des protocoles de connexions à distance type SSH). Au lieu de définir des séquences d'actions à exécuter, Ansible adopte une approche déclarative, où l'administrateur spécifie l'état souhaité du système, et Ansible se charge de faire correspondre l'état du système sur cette déclaration, en prenant l'ensemble des opérations nécessaires.

Sans base de données Ansible n'utilise pas de « base de données » pour stocker l'état courant des déploiements de configurations : tout est décrit dans les modules et les playbooks Ansible, qui consistent simplement en du texte. Cela facilite la collaboration autour des playbooks : il n'y a pas besoin de partager une base de données centralisée pour s'assurer qu'Ansible fonctionne correctement ; plusieurs administrateurs peuvent *théoriquement* intervenir en parallèle sans avoir besoin de se coordonner ou de se synchroniser.

Playbooks La structure fondamentale d'Ansible repose sur des « playbooks », des fichiers YAML qui décrivent de manière formelle les tâches à effectuer sur un ensemble d'entités. Ces tâches englobent diverses opérations telles que l'installation de logiciels (peu utile dans le cas d'équipements réseaux physiques), la configuration de services, la gestion de fichiers, etc.

Modules Ansible fournit de nombreux modules prêts à l'emploi pour effectuer diverses opérations sur les systèmes cibles. Par exemple, il existe des modules pour gérer les paquets, les services, les utilisateurs, les fichiers, les bases de données, et bien d'autres. Dans un contexte réseau, certains modules peuvent gérer l'état d'interfaces au niveau 2, ou au niveau 3, le routage, etc.

Inventaire Pour organiser la gestion, Ansible utilise un « inventaire », un répertoire d'entités répertoriées par groupe et agrémenté de variables d'inventaire pour personnaliser les opérations. On pourrait par exemple imaginer un groupe « switch », un groupe « routeur », et si l'on configure plusieurs sites, des groupes permettant de spécifier la localisation géographique des équipements (« Paris », « Tokyo », …).

Gestion des secrets Ansible propose des mécanismes pour gérer en toute sécurité les informations sensibles, telles que les mots de passe et les clés SSH, en utilisant des variables chiffrées, des coffres-forts (Vaults), et des gestionnaires d'identités externes.

Attention : L'automatisation, par nature, réduit le risque d'erreur humaine en reproduisant automatiquement les meilleures pratiques connues qui ont été testées en profondeur dans un environnement donné. Cependant, l'automatisation peut être dangereuse si elle reproduit un mauvais processus ou une configuration erronée (cela s'applique à n'importe quel outil d'automatisation, et pas seulement à Ansible).

Lorsque l'on se prépare à automatiser une tâche ou un ensemble de tâches, il est important de commencer par définir le résultat souhaité de l'automatisation, pour ensuite créer un plan pour atteindre ce résultat. Une méthodologie couramment utilisée pour ce processus est le cycle de vie PPDIOO (Prepare, Plan, Design, Implement, Observe, Optimize), illustré à la figure 2.

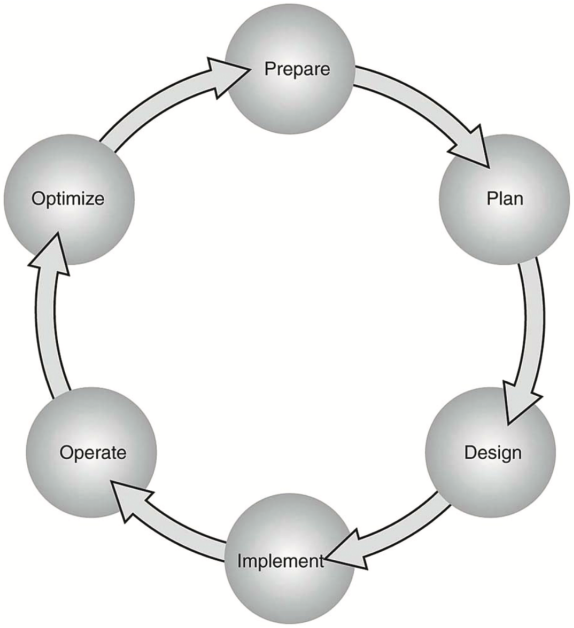

FIGURE 2 – Cycle de vie PPDIOO

2.1 Playbooks et YAML

Ansible utilise des « playbooks » pour déployer des changements de configuration ou récupérer des informations sur les hôtes d'un réseau. Un « playbook » Ansible est un ensemble structuré d'instructions. Un « playbook » Ansible contient plusieurs jeux, et chaque jeu contient les tâches que chaque joueur doit accomplir pour que le jeu particulier soit réussi. Le Tableau 2 décrit les composants utilisés dans Ansible et fournit quelques exemples couramment utilisés.

Composants	Description	Cas d'utilisation
Playbook	Une collection de "Plays" pour un hôte ou un groupe d'hôtes	Appliquer les étapes de configuration et/ou de déploiement
Play	Un ensemble de "tasks" appliqués à un seul hôte ou à un groupe d'hôtes.	Regroupement d'un ensemble d'hôtes pour leur appliquer une politique ou une configuration.
Task	Un appel à un module Ansible	Connexion à un dispositif pour lancer une commande d'affichage afin de récupérer les données de sortie.

TABLE 2 – Structure d'un « playbook » Ansible

Les playbooks Ansible sont écrits en utilisant YAML (Yet Another Markup Language). Les fichiers YAML Ansible commencent généralement par une série de trois tirets (- - -) et se terminent par une série de trois points (...). Bien que

cette structure soit facultative, elle est courante. Les fichiers YAML contiennent également des listes et des dictionnaires. L'exemple ci-dessous présente un fichier YAML contenant une liste de genres musicaux.

Exemple d'une liste YAML

```yaml
---
# List of music genres
Music:
  - Metal
  - Rock
  - Rap
  - Country
...
```

Attention : En YAML, l'indentation est très importante (comme en python). Elle s'effectue en général avec deux espaces (pas de tabulations). Des erreurs d'indentations peuvent silencieusement changer la structure de vos données, et donc le comportement de vos playbooks Ansible !

Les listes YAML sont très faciles à lire et à utiliser. Il est possible d'ajouter des commentaires dans YAML en commençant les lignes par un dièse (#). Comme mentionné précédemment, un fichier YAML commence souvent par - - - et se termine par ... ; en outre, chaque ligne d'une liste peut commencer par un tiret et un espace (-), et l'indentation rend le fichier YAML lisible. YAML utilise des dictionnaires similaires aux dictionnaires JSON, car ils utilisent également des paires clé/valeur. Une paire clé/valeur JSON se présente sous la forme " clé " : "value" ; une paire clé/valeur YAML est similaire mais ne nécessite pas de guillemets : key : value. “ L'exemple ci-dessous présente un dictionnaire YAML contenant un enregistrement d'employé :

Exemple d'un dictionnaire YAML

```yaml
---
# HR Employee record
Employee1:
  Name: John Dough
  Title: Developer
  Nickname: Mr. DBug
```

Les listes et les dictionnaires peuvent être utilisés ensemble dans YAML. L'exemple ci-dessous montre un dictionnaire avec une liste dans un seul fichier YAML.

Exemple d'un dictionnaire et d'une liste YAML

```yaml
---
# HR Employee records
- Employee1:
    Name: John Dough
    Title: Developer
    Nickname: Mr. DBug
    Skills:
      - Python
      - YAML
      - JSON
- Employee2:
    Name: Jane Dough
    Title: Network Architect
    Nickname: Lay DBug
    Skills:
      - CLI
      - Security
      - Automation
```

Ansible dispose d'un outil CLI qui peut être utilisé pour exécuter des playbooks ou des commandes CLI ad hoc sur des hôtes ciblés. Cet outil comporte des commandes très spécifiques que vous devez utiliser pour activer l'automatisation. Le tableau ci-dessous présente les commandes CLI Ansible les plus courantes et les cas d'utilisation associés.

Commande CLI	Cas d'utilisation
`ansible`	Exécution des modules sur les hôtes ciblés
`ansible-playbook`	Exécution de « playbooks »
`ansible-docs`	Fournit une documentation sur la syntaxe et les paramètres dans le CLI.
`ansible-pull`	Fait passer les clients Ansible du modèle "push" par défaut au modèle "pull"
`ansible-vault`	Chiffre les fichiers YAML qui contiennent des données sensibles.

TABLE 3 – Commandes CLI Ansible

Ansible utilise un fichier d'inventaire pour garder la trace des hôtes qu'il gère. L'inventaire peut être un groupe d'hôtes nommé ou une simple liste d'hôtes individuels. Un hôte peut appartenir à plusieurs groupes et peut être représenté par une adresse IP ou un nom DNS résoluble. L'exemple ci-dessous montre le contenu d'un fichier d'inventaire d'hôtes avec l'hôte 192.168.10.1 dans deux groupes différents.

Exemple d'inventaire Ansible au format YAML

```yaml
routers:
  hosts:
    192.168.10.1:
    192.168.20.1:
switches:
  hosts:
    192.168.10.25:
    192.168.10.26:
primary-gateway:
  hosts:
    192.168.10.1:
```

Voyons maintenant quelques exemples de « playbooks » Ansible utilisés pour accomplir des tâches courantes. Imaginez utiliser un « playbook » pour déployer la configuration d'une interface sur un périphérique sans avoir à la configurer manuellement. Vous pourriez pousser cette idée un peu plus loin et utiliser un « playbook » pour configurer une interface et déployer un processus de routage EIGRP. L'exemple suivant présente le contenu d'un « playbook » Ansible appelé "ConfigureInterface.yaml", que vous pouvez utiliser pour configurer l'interface GigabitEthernet2 sur un routeur CSR 1000V. En exploitant le module Ansible "ios_config", ce « playbook » ajoute la configuration suivante à l'interface GigabitEthernet2 sur le routeur CSR1KV-1 :

Show interface GigabitEthernet2

```
description Configured by ANSIBLE!!!
ip address 10.1.1.1
subnet mask 255.255.255.0
no shutdown
```

ConfigureInterface.yaml Playbook

```yaml
---
- hosts: CSR1KV-1
  gather_facts: false
  connection: local
  tasks:
  - name: Configure GigabitEthernet2 Interface
    ios_config:
      lines:
        - description Configured by ANSIBLE!!!
        - ip address 10.1.1.1 255.255.255.0
        - no shutdown
      parents: interface GigabitEthernet2
      host: "{{ ansible_host }}"
      username: cisco
      password: testtest
```

Pour exécuter ce « playbook », la commande `ansible-playbook` est utilisée pour appeler le fichier YAML du

« playbook » spécifique (`ConfigureInterace.yaml`). La figure suivante montre la sortie de l'appel du « playbook » à partir du shell Linux. Les éléments importants à noter dans la sortie sont les sections PLAY, TASK et PLAY RECAP, qui listent le nom du jeu et chaque tâche individuelle exécutée dans chaque jeu. La section PLAY RECAP montre l'état du « playbook » qui est exécuté. La sortie de la Figure suivante montre qu'un "play", nommé CSR1KV-1, a été lancé, suivi d'une "task" appelée "Configure GigabitEthernet2 Interface". D'après le statut "ok=1", vous savez que la modification a réussi ; le statut "changed=1" signifie qu'une seule modification a été effectuée sur le routeur CSR1KV-1.

Exécution du Playbook ConfigureInterface.yaml

```
$ ansible-playbook ConfigureInterface.yaml

PLAY [CSR1KV-1] ****************************************************************************

TASK [Configure GigabitEthernet2 Interface ***********************************************]




changed: [CSR1KV-1]

PLAY RECAP *********************************************************************************
CSR1KV-1                   : ok=1   changed=1    unreacheable=0   failed=0
```

Préambule : De l'importance d'un réseau Out-Of-Band

Lorsque l'on configure des équipements réseaux, il est bien évidemment possible d'utiliser des connexions console sur le port série de ces équipements, mais, très souvent, la première tâche est d'activer un réseau de management dit « out-of-band » sur ces équipements réseaux, ainsi qu'un serveur SSH, pour gérer à distance ces équipements.

Ce réseau est dit « out-of-band » car il offre une voie de communication **distincte et indépendante** de la connectivité principale du réseau, ce qui garantit la disponibilité permanente pour la gestion, même en cas de défaillance du réseau principal (qui sera plutôt appelé « inband »). Cette redondance est vitale pour maintenir la continuité des opérations réseau : on imagine très bien (et on l'a vu plusieurs fois pendant les TPs de NET4101) se tromper sur un masque réseau ou une adresse IP lorsque que l'on configure un équipement réseau : si notre connexion SSH dépend de la connexion principale de l'équipement réseau, et que cette erreur casse cette connexion réseau, on se retrouve coupé de l'équipement réseau à configurer et on doit le reconfigurer via un accès physique, par connexion console / série.

Un réseau de management « out-of-band » permet donc un accès distant sécurisé (via SSH) aux équipements réseau, ce qui facilite leur configuration, leur surveillance, ainsi que leur dépannage à distance, sans interagir a priori avec le réseau principal de production. Cette séparation permet aussi une isolation et une protection accrues contre les attaques potentielles : si le réseau principal abrite une machine cliente compromise (par exemple, un poste Windows non mis à jour), celui-ci ne devrait pas avoir accès au réseau de management, et ne devrait pas pouvoir se connecter aux équipements réseaux qui, théoriquement, ne devraient exposer leur serveur SSH que sur le réseau de management « out-of-band ».

3 Réalisation du TP

Objectif : L'objet de ce TP est de vous permettre de découvrir comment utiliser un outil de gestion de configuration pour gérer un ensemble d'équipements réseau. Nous avons choisi une des solutions les plus utilisées : Ansible.

Ce TP est constitué de deux parties utilisant la même architecture réseau. La première partie présente une à une les principales fonctionnalités et notions d'Ansible, et se concentre donc sur le « pilotage » d'un seul équipement réseau.

La seconde partie vous permet de piloter une maquette de réseau dans son ensemble, et d'utiliser ce que vous venez d'apprendre pour la configurer. Elle est volontairement beaucoup moins directive. Une correction vous sera proposée, mais il ne s'agit que d'une des nombreuses solutions possibles. Vous êtes donc invités à expérimenter et à essayer de monter la maquette par vous-même avant de regarder la correction.

3.1 Découverte d'Eve-NG et d'Ansible

Pendant cette session, nous allons utiliser une machine virtuelle hébergée sur les serveurs de l'équipe THD, serveurs hébergés dans le mini-datacenter du troisième étage du bâtiment étoile que vous avez visité au début de l'année. Cette machine virtuelle utilise le logiciel **EVE-NG** qui permet d'héberger et de virtualiser des équipements réseaux et les câbles réseaux les reliant ensemble. Cette machine virtuelle héberge une topologie réseau elle aussi virtuelle, basée sur des routeurs cisco virtuels (Cisco Cloud Services Router 1000V Series), ainsi que sur des switchs virtuels (IOL), qui se comportent à peu de chose près comme les routeurs et les switchs que vous avez déjà eu l'occasion de manipuler. Dans ce TP, **l'ensemble des opérations s'effectueront depuis la machine hôte**, à travers sa console et à travers son navigateur web, sans utiliser de machines virtuelles dans VirtualBox.

Question 1 : Accéder à l'interface web de EVE-NG, qui est disponible à l'adresse `http://192.168.0.254/`. Le nom d'utilisateur à utiliser est `admin` et le mot de passe `eve`. Choisissez l'option « Native console ». Si le *Lab* intitulé « NET4101 - Automatisation » n'est pas encore ouvert, ouvrez le en le sélectionnant et en cliquant sur « Open ». Sinon, vous devriez voir une architecture virtuelle composée de deux réseaux, trois routeurs, deux switchs, et de 4 PCs, tous éteints (grisés).

Important : Si vous n'arrivez pas à ouvrir l'interface web, prenez contact avec vos chargés de TPs.

Question 2 : Ajoutez un lien entre le switch S4 et le switch S5 (i.e. entre les deux switchs). Pour cela, effectuez un clic gauche (maintenu) sur la petite prise orange apparaissant sur l'un des switch, et relachez la souris après l'avoir déplacée sur le second switch. Choisissez ensuite sur quel port, sur chaque switch, brancher ce « câble » virtuel.

Important : Pour ajouter des câbles « virtuels », tous les nœuds doivent être éteint !

Question 3 : Lancez l'ensemble des machines du TP (clic-droit, « Start », ou alors « More actions » > « Start all nodes » dans le menu de gauche). Comme vous pouvez le constater, l'ensemble des entités du réseau sont reliées à un même réseau de *Management* nommé *NetManagement*, sur au moins l'une de leur interface : il s'agit du réseau *Out-Of-Band* mentionnée au début de la partie. Nous avons déjà préconfiguré ce réseau de management : il utilise le réseau IP `10.0.0.0/24`, réseau qui devrait être directement accessible depuis votre machine hôte (vérifiez avec `ip route`).

Les préconfigurations qui ont été appliquées sont les suivantes :

Configuration des routeurs

```
enable
configure terminal
    hostname Router1 ! ou Router2, Router3. . .
    ip domain name tp.fr
    no ip domain lookup
    crypto key generate rsa modulus 2048
    ip ssh version 2
    line vty 0 4
        transport input ssh
        login local
        exit
    username utilisateur secret motdepasse
    enable secret int
    interface GigabitEthernet1
        ip address 10.0.0.??? 255.255.255.0
        no shutdown
        exit
    exit
copy running-config startup-config
```

Configuration des switchs

```
enable
configure terminal
    hostname Switch1 ! ou Switch2, Switch3. . .
    ip domain-name tp.fr
    no ip domain-lookup
    crypto key generate rsa modulus 2048
    ip ssh version 2
    line vty 0 4
        transport input ssh
        login local
        exit
    username utilisateur secret motdepasse
    enable secret int
    interface vlan 1
        ip address 10.0.0.??? 255.255.255.0
        no shutdown
        exit
    exit
copy running-config startup-config
```

Question 4 : À quoi servent chacune des commandes dans les configurations ci-dessus ?

Question 5 : Reconstruisez sur votre cahier de manipulation un schéma de l'architecture présentée et spécifiez les adresses IP de management des différents équipements. Vous pouvez accéder à des terminaux vers ces équipements en double-cliquant sur leurs icones, et en acceptant, le cas échéant, d'utiliser gnome-terminal pour afficher la connexion *telnet* si Firefox vous le demande.

3.2 Découverte d'Ansible

3.2.1 Se connecter à une machine avec Ansible (SSH)

Pour qu'un contrôleur puisse gérer la configuration d'une machine, il faut évidemment qu'il puisse s'y connecter et y exécuter des commandes. La plupart des systèmes de gestion de configuration utilisent un client dédié à installer sur la machine. Avec Ansible, on utilise simplement SSH.

Question 6 : Quel est l'intérêt d'utiliser SSH au lieu d'un client dédié pour gérer les équipements réseau ? Vérifiez que vous arrivez à vous connecter aux différents équipements réseaux depuis votre terminal en utilisant des commandes du type `ssh <utilisateur>@<Adresse IP>`.

Attention : Pour se connecter aux équipements réseaux de type *switch*, il est nécessaire d'ajouter des options à la commande `ssh`, qui sont les options suivantes :
`-oKexAlgorithms=+diffie-hellman-group1-sha1 -oHostKeyAlgorithms=+ssh-rsa`.
Ces options sont nécessaires car le serveur SSH tournant sur les switchs est trop vieux vis à vis du client SSH qui est disponible sur votre machine hôte.

Question 7 : Lorsqu'on se connecte pour la première fois en SSH à une nouvelle machine, un prompt s'affiche et nous demande quelque chose... Pourquoi ? (indice : rechercher `SSH TOFU` sur le web). Effectuez cette manipulation (TOFU) sur l'ensemble des équipements réseaux de la maquette.

3.3 Inventaire et premières commandes Ansible

Maintenant que l'on s'est assuré que la connexion SSH aux équipements fonctionne, le moment est venu de préparer notre environnement Ansible et d'exécuter nos premières commandes.

Information : L'éditeur de texte `codium`, qui est peu ou prou l'équivalent libre de `vscode` est installé sur vos machines hôtes. N'hésitez pas à vous en servir pour l'édition des fichiers textes, avec de la coloration syntaxique !

Question 7 : Créer un dossier dans votre home nommé `NET4101-<Date du jour>` et déplacez vous, en console, vers ce dossier. Créez ensuite un fichier `ansible.cfg` le texte suivant :

```
[defaults]
inventory = ./inventory.yaml
```

Question 7 : Toujours dans ce dossier, créez un fichier nommé `inventory.yaml` contenant l'inventaire des équipements, au format YAML, tel que décrit à la page 5. À la fin de l'inventaire, ajoutez la section suivante en adaptant le contenu :

```yaml
all:
  vars:
    ansible_user: <UTILISATEUR SSH>
    ansible_ssh_pass: <MOT DE PASSE SSH>
    ansible_ssh_common_args: -oKexAlgorithms=+diffie-hellman-gro ... # Continuer la ligne
    ansible_connection: network_cli
    ansible_network_os: ios
    ansible_become: yes # On passe « enable » sur l'ensemble des commandes / playbooks
    ansible_become_method: enable
    ansible_become_pass: <MOT DE PASSE BECOME>
```

Cette section permet de définir des variables sur l'ensemble des machines de l'inventaire (d'où le mot clef `[all]`) et de spécifier, entre autres, le nom d'utilisateur à utiliser pour se connecter en SSH aux équipements, le mot de passe, des options supplémentaires pour SSH (utile pour les switchs…), et des informations sur le mode de connexion aux équipements (il s'agit d'équipements Cisco utilisant le système d'exploitation `ios`).

Question 8 : Vérifiez avec la commande `ansible-inventory --list` que votre inventaire est bien compris par Ansible. Entre autres, vous devriez pouvoir observer que les variables ajoutées à la question précédente sont bien « activées » sur l'ensemble des hôtes.

Question 9 : Utilisez la commande suivante pour vérifier que les connexions s'effectuent bien vers les équipements réseaux : `ansible all -m ios_command -a "commands='show version'"`. Dans cette commande, à quoi sert le mot clef `all` ?

Question 10 : Quelle commande faut-il utiliser pour afficher la liste des VLANs des différents switchs (et uniquement les switchs) ? Le cas échéant, modifiez votre inventaire pour rendre cette opération plus simple.

3.4 Écrire des séquences de commandes en YAML avec les Playbooks

Pour l'instant, nous n'avons utilisé Ansible que pour exécuter des commandes uniques. Ce mode d'utilisation est appelé Ad hoc et si ce mode est bien pratique pour exécuter rapidement une commande ou faire du debug, l'intérêt principal d'Ansible se trouve dans l'utilisation des playbooks.

Les playbooks sont des séquences d'instructions (ou tasks) exécutées de manière séquentielle sur un ou plusieurs hôtes. Ils sont définis au format YAML, qui est un format de représentation de données comparable à JSON ou XML, qui se veut facile à lire par des êtres humains.

Important : Attention, tout comme pour le Python, l'interprétation d'un fichier YAML repose sur l'indentation. Les espaces sont donc importants.

Nous allons redéfinir notre commande précédente sous forme de playbook. Commencez par créer un nouveau fichier que vous nommerez `display_config.yaml`. Voici son contenu :

```
- hosts: all
  tasks:
    - ios_command:
        commands: show running-config
```

Pour exécuter notre playbook, nous allons utiliser la commande `ansible-playbook display_config.yaml`.

A l'exécution de notre playbook, on se rend compte qu'il ne se passe pas grand-chose... C'est tout à fait normal car nous avons demandé à Ansible d'exécuter la commande show running-config mais pas de nous afficher le résultat. Corrigeons cela.

```
- hosts: all
  tasks:
    - ios_command:
        commands: show running-config
      register: sortie_commande
    - debug:
        msg: "{{ sortie_commande }}"
```

3.5 Modification de la configuration et utilisation de modules Ansible

Jusqu'ici, nous avons utilisé le module `ios_command` qui permet d'exécuter une commande simple sur notre switch. Évidemment, Ansible dispose de toute une gamme de modules, listés dans la documentation officielle. Pour ce TP nous nous intéresserons principalement aux modules réseau pour IOS.

Question 11 : Assurez vous que les modules `cisco.ios` sont bien installés en tapant la commande `ansible-galaxy collection install cisco.ios` sur votre machine hôte.

Remarquez que dans la partie précédente, nous avons utilisé le module debug pour afficher la valeur d'une variable conf qui stocke le résultat de l'exécution de l'instruction précédente. Utilisons maintenant un nouveau module permettant de modifier la configuration de nos switches : `ios_config`. Pour premier exemple, essayons de modifier le hostname du switch S5, qui est théoriquement `Switch2` en `Switch5`.

```
- hosts: 10.0.0.. . .
  tasks:
    - name: Change hostname
      ios_config:
        lines:
          - hostname Switch5
```

On utilise l'IP dans cet exemple, mais si votre fichier d'inventaire défini un nom spécifique pour le switch, il est aussi possible d'utiliser ce nom. Une fois que le playbook à été exécuté, connectez vous en SSH sur le switch pour constater le changement.

Pour continuer sur notre exemple, partons maintenant sur l'exécution d'une suite de taches : en plus de changer le hostname, nous allons ajouter une bannière MOTD et créer un nouvel utilisateur.

```
- hosts: 10.0.0.. . .
  tasks:
    - name: Another way to change the hostname
      cisco.ios.ios_hostname:
        config:
          hostname: Switch1337
        state: merged
    - name: Add MOTD banner
      ios_banner:
```

```
      banner: motd
      text: |
        Hey ! I am a switch !
        and I have a nice
        multiline
        banner !
      state: present
  - name: Add a new user
    ios_user:
      name: remy # ou antoine, ou . . .
      configured_password: 'hunter2'
```

Question 12 : Quelles modifications devriez-vous apporter au playbook pour sauvegarder le changement de configuration ?

Question 13 : Dans les questions précédentes, on a vu qu'il y avait plusieurs moyens de changer le *hostname* des équipements ciscos, par le biais de `ios_hostname` (commande spécialisée) et par le biais de `ios_config` (commande générique).
Quel intérêt peut-il y avoir à utiliser une commande spécialisée plutôt que la commande générique ?
Allez consoluter la liste des modules disponibles à l'adresse de la documentation Ansible pour les équipements ciscos https://docs.ansible.com/ansible/latest/collections/cisco/ios/index.html, certains vous seront peut être utiles plus tard…

3.6 Récupérer des informations avec les facts Ansible

Jusqu'ici, nous avons exécuté explicitement la commande show running-config pour récupérer des informations sur la configuration de notre switch. Cette Méthode fonctionne, mais il y a plus simple. Par défaut lorsqu'un playbook est exécuté, Ansible récupère ce que l'on appelle des facts sur chaque hôte (il s'agit de la première tache dans l'affichage de sortie à l'exécution de nos playbooks, Gathering Facts). Ces facts sont un ensemble de variables système. Si nous souhaitons voir les facts qu'Ansible récupère sur nos switches, nous pouvons utiliser le playbook suivant, qui fait appel au module debug.

```
- hosts: all
  tasks:
    - name: display facts
      debug:
        msg: "{{ ansible_facts }}"
```

On remarque que pour accéder à la valeur d'une variable, on utilise la syntaxe suivante :{{ nom_variable }}. La variable `ansible_facts` contient l'ensemble de nos facts. On peut ainsi utiliser les facts pour générer rapidement toute sortes de rapport. Par exemple, si on veut connaître la version d'IOS et le numéro de série de nos switches, on peut utiliser le playbook suivant.

```
- hosts: all
  tasks:
    - name: display facts
      debug:
        msg: "La version d'IOS est {{ ansible_facts.net_version }} et le numero de série est
              {{ ansible_facts.net_serialnum }}"
```

4 Configuration d'un réseau avec Ansible

Maintenant que nous nous sommes familiarisés avec les concepts de base d'Ansible, il est temps de les mettre en pratique pour construire et configurer un réseau.

Vous êtes un(e) consultant(e) ayant pour mission de mettre en place le réseau d'une startup qui vient d'acquérir de nouveaux locaux. Elle comprend deux catégories d'utilisateurs : les développeurs et les agents marketing (que nous désignerons respectivement par Dev et Comm).

Étant donné que l'entreprise ne dispose pas de SI propre, mais uniquement d'une équipe de développeurs, vous décidez d'aborder une approche IaC (Infrastructure as Code). La configuration des équipements sera donc définie par un ensemble de playbooks Ansible.

Vous avez décidé de mettre en place le réseau suivant :

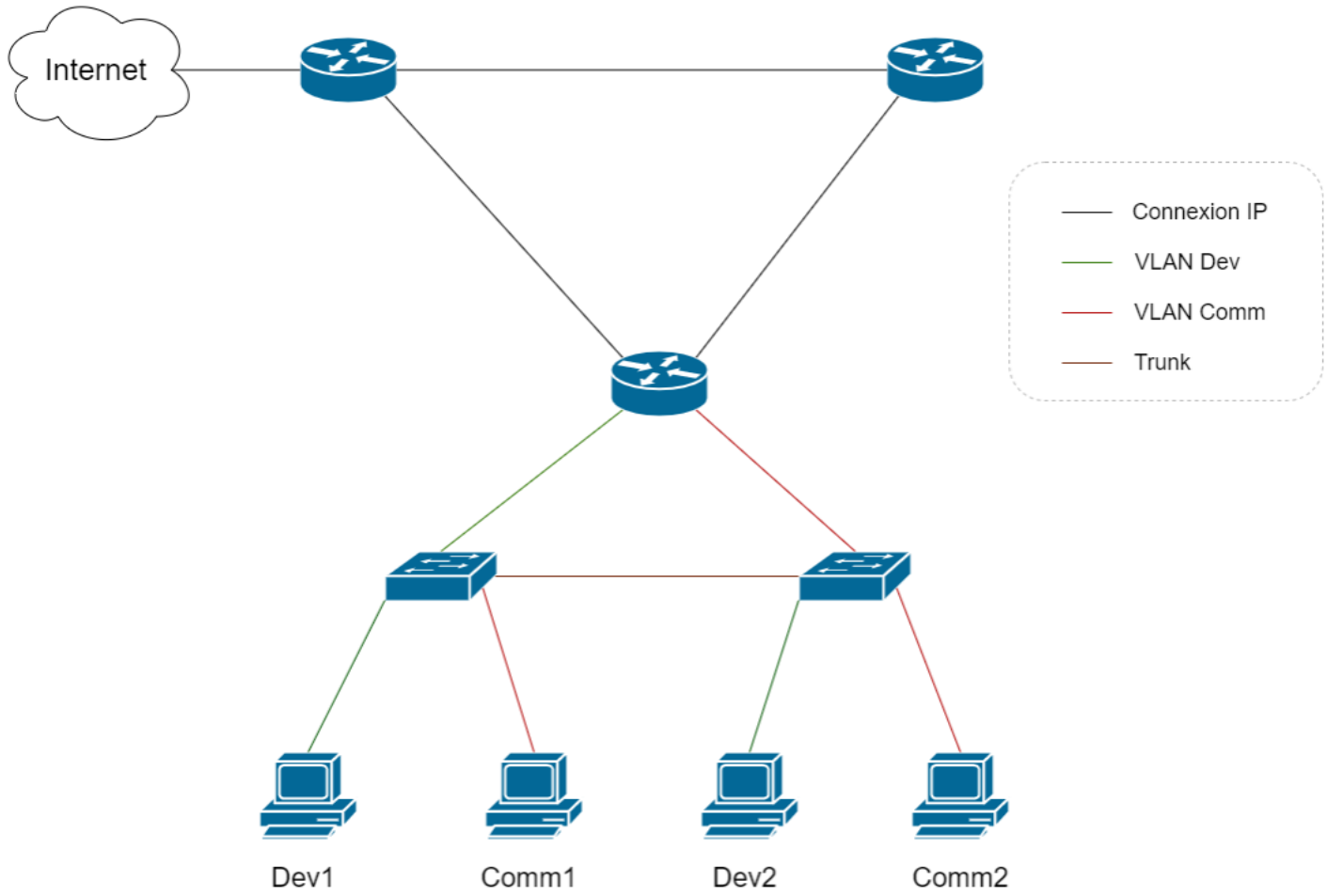

FIGURE 3 – Topologie physique du réseau à configurer

Ce réseau est composé de trois routeurs échangeant leurs routes via OSPF.

La configuration d'un routeur OSPF a été vue en première année pendant le TP de Net3602. On rappelle ici les commandes à taper pour configurer un routeur dont l'IP est `W.X.Y.Z` dans un réseau `/24`, accessible via son interface `Ethernet42`, pour qu'il annonce le réseau `W.X.Y.Z/24` en OSFP :

```
routeur-ABCD(config)# interface Ethernet42
routeur-ABCD(config-if)# ip address W.X.Y.Z 255.255.255.0
routeur-ABCD(config-if)# exit
routeur-ABCD(config)# router ospf 1
routeur-ABCD(config-router)# router-id W.X.Y.Z
routeur-ABCD(config-router)# network W.X.Y.Z 255.255.255.0 area 0 !
routeur-ABCD(config-router)# exit
```

On rappellera aussi que le réseau `172.16.0.0/12`, peut être utilisé pour faire de l'adressage privé, au même titre que `10.0.0.0/8` ou `192.168.0.0/16`. Attention cependant à ne pas créer de collisions avec le réseau de management !

L'un des routeurs permet à l'entreprise de sortir sur Internet. Pour ce TP, nous ferons l'impasse sur la configuration de l'interface de sortie vers Internet. Un autre routeur va raccorder le réseau local du site principal de l'entreprise. Un troisième routeur à été installé afin de prévoir l'installation ultérieure d'un second site.

Sur l'un des routeurs est connecté une paire de switches, auxquels sont raccordés des postes utilisateur.

Pour ce qui est des postes utilisateur, ils seront séparés en deux VLANs. Pour chaque VLAN, sa passerelle sera définie sur une des interfaces de routeur. Pour mieux répartir le trafic, un lien trunk à été établi entre les deux switch.

A vous maintenant de déployer ce réseau, en utilisant Ansible !

Note : N'hésitez pas à vous référer à la documentation officielle d'Ansible
Documentation générale
Documentation réseau
Documentation des modules cisco de Ansible

N'oubliez pas : il n'y a pas de solution unique ! Bonne chance !