

Ansible pour DevOps

Par Dirane TAFEN

Plan

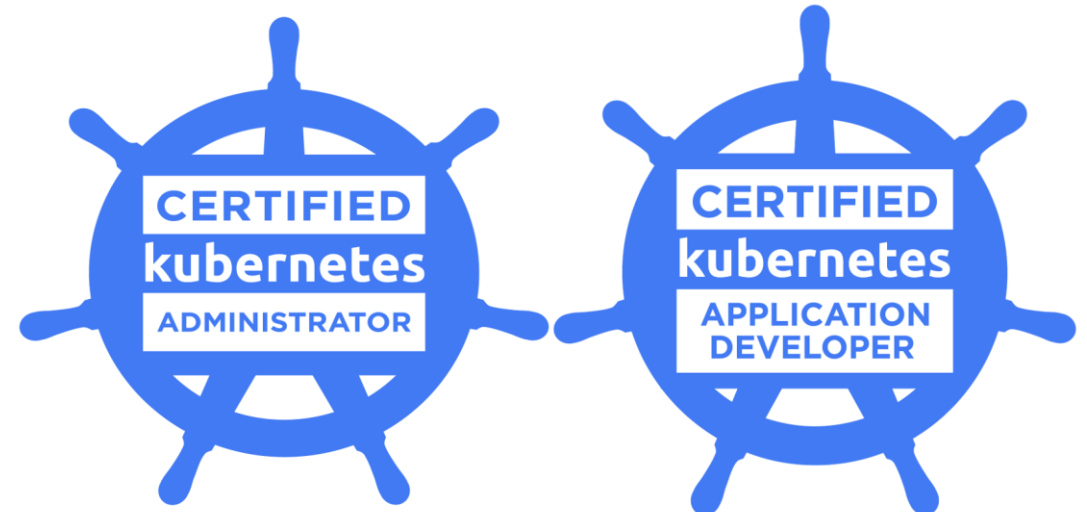
- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating et loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Présentation du formateur

- Dirane TAFEN (formateur et consultant DevOps)
- Capgemini
- Sogeti
- ATOS
- BULL
- AIRBUS
- ENEDIS

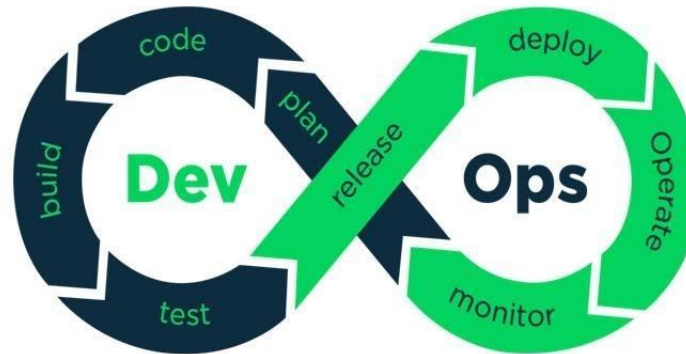


Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Introduction au DevOps et Ansible (1/3): DevOps

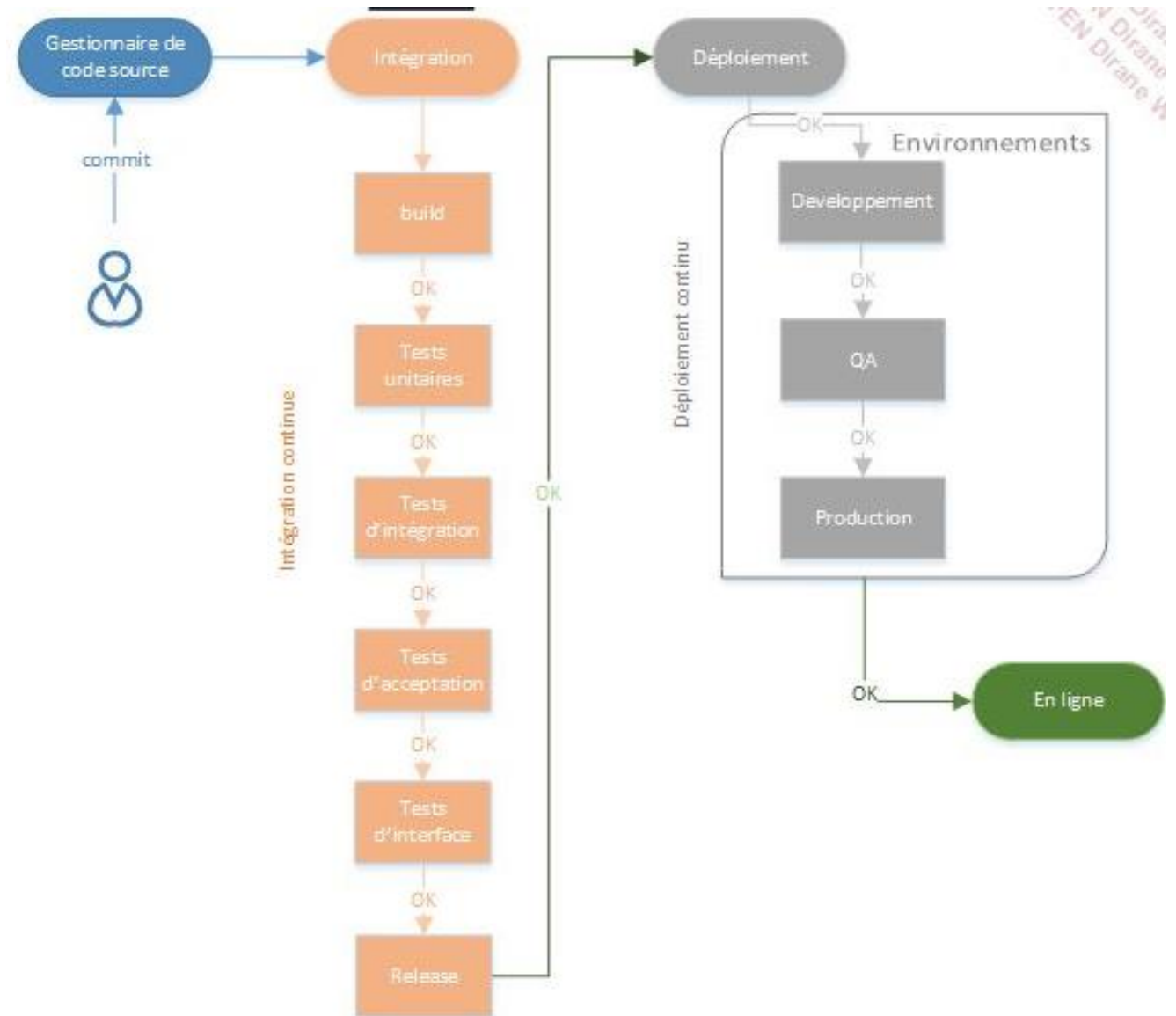
Agile vs. DevOps



- Agile: méthode de développement
- DevOps: agilité dans le Dev et l'Ops = CI + CD

Introduction au DevOps et Ansible (2/3): CI/CD

- Intégration en continu
- Test en continu
- Déploiement en continu
- Automatisation 😊
- Docker



Introduction au DevOps et Ansible (3/3): Ansible

- C'est quoi ?
 - ✓ Cloud provisionneur
 - ✓ Configuration Management
 - ✓ Déploiement d'Application
- Pourquoi ?
 - ✓ Efficace: pas d'agent, copie de petit bout de code sur les machines distantes
 - ✓ Sécurisé: pas d'agent, utilise OpenSSH
 - ✓ Contribution: open-source, python

TP-0: Découvrir la plateforme de TP

- Accès
- Labs
- Durée d'une session
- Données sensibles
- Agrandir la fenêtre du terminal
- Connexion ssh
- Installation de paquet
- Ouverture de port

TP-1: Installation de Ansible

- Installation sur CentOS
- Présentation de l'environnement Ansible d'EAZYTraining
- Toujours travaillez avec admin et pas root pour des raisons de sécurité
- Multi-node
- Connexion SSH

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Commandes AD-HOC (1/1): Principes

ANSIBLE AD HOC COMMANDS - SYNTAX			
	Host Group	Module	Arguments to the module
ansible	webserver	-m yum	-a "name=httpd state=latest"
ansible	allservers	-m shell	-a " find /opt/oracle -type f -mtime +10 -name '*.log' "
ansible	appserver	-m user	-a "name=saravak group=admins append=yes shell=bin/bash"

- Test de module
- Lancement de tâche rapidement
- Peut être lancé sur un groupe de machine

TP-2: Utilisation des commandes ad-hoc

- Créez un cluster (1 ansible et 1 client)
- Créez un fichier d'inventaire hosts
- Utilisez une commande ad-hoc pour tentez de ping le client ansible
- Utilisez une commande ad-hoc pour créer un fichier toto.txt avec le contenu « bonjour eazytraining » et qui se trouvera dans le dossier /home/admin/toto.txt sur le client ansible
- Vérifiez que le fichier a bien été créé avec le contenu
- Rajoutez un client et modifier le fichier inventaire afin de rajouter le nouveau client
- Relancez l'action de ping et de création de fichier sur les 2 clients maintenant et vérifiez le resultat
- Testez l'effet du module « setup » sur votre inventaire

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

```
# Un fichier yaml démarre par les 3 tirets ci-dessus
# Déclaration simple
chaine_simple: "Une chaîne simple"
# Ici _42 va contenir un entier :
_42: 42
# _33 va contenir un chiffre à virgule :
_33: 33.333
```

Découverte du yaml (1/5): Déclaration de variables

Version compacte

a: [1, 2, "trois"]

Version très compacte

a: [1,2,"trois"]

Version invalide (manque l'espace après

a:[1,2,"trois"]

a:

- 1

- 2

- "trois"

Découverte du yaml (2/5): les tableaux


```
utilisateur1:  
  nom: pierre  
  prenom: yannig
```

```
utilisateur2:  
  nom: pierre  
  prenom: sarah
```

```
utilisateur1:  
  nom: pierre  
  prenom: yannig  
  date_de_naissance:  
    jour: 7  
    mois: 11  
    annee: 1977
```

Découverte du yaml (3/5): Structures clé/valeur

Découverte du yaml (4/5): Tableau de tables de hachage

```
# Version un peu plus compacte
users:
  - { nom: perre, prenom: yannig }
  - { nom: perre, prenom: sarah }
# Version plus compacte, mais moins lisible
users: [{nom: perre, prenom: yannig},{nom: perre, prenom: sarah}]
```

```
liste_utilisateurs:
  - nom: perre
    prenom: yannig
  - nom: perre
    prenom: sarah
```

Découverte du yaml (5/5): Inventaire

```
all:  
  hosts:  
    rec-apache-1:  
      ansible_user: root
```

- Depuis la version 2.4 de Ansible
- All et Hosts

```
rec-apache-1 ansible_user=root
```

TP-3: Inventaire au format yaml

- Modifiez le fichier hosts que vous avez écrit au format INI afin qu'il soit en au format yaml
- Testez à nouveau vos commandes ad-hoc avec le nouveau fichier d'inventaire au format yaml

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Inventaire ansible (1/5): Problématique

[webservers]

192.168.35.140

192.168.35.141

192.168.35.142

192.168.35.143

[appservers]

192.168.100.1

192.168.100.2

192.168.100.3

[dbservers]

172.35.0.5

```
[all:vars]
ansible_connection=local

[apache]
rec-apache-1 apache_url=rec.wiki.localdomain

[mysql]
rec-mysql-1 mysql_user_password=MyPassword!

[active-directory]
active-directory-1

[microservices]
container-1 ansible_connection=docker

[linux:children]
apache
mysql

[windows:children]
active-directory

[container:children]
microservices

[windows:vars]
ansible_connection=winrm

[container:vars]
ansible_connection=localhost
```

Inventaire ansible (2/5): Structure INI

```

all :
  vars:
    ansible_connection: local

linux:
  children:
    apache:
      hosts:
        rec-apache-1:
          apache_url: "rec.wiki.localdomain"

    mysql:
      hosts:
        rec-mysql-1:
          mysql_user_password: "MyPassWord!"

windows:
  children:
    active-directory:
      hosts:
        active-directory-1: {}

  vars:
    ansible_connection: "winrm"

container:
  children :
    microservices :
      hosts :
        container-1 :
          ansible_connection : "docker"
  vars :
    ansible_connection : "localhost"

```

Inventaire ansible (3/5): Structure YAML

Inventaire ansible (4/5): Variable d'inventaire

[apache]

rec-apache-1 apache_url=rec.wiki.localdomain

Cette déclaration prendra la forme suivante au format YAML :

```
apache:
  hosts:
    rec-apache-1:
      apache_url: "rec.wiki.localdomain"
```

[mysql:vars]

mysql_user_password=MyPassWord!

Et la même déclaration au format YAML :

```
mysql:
  vars:
    mysql_user_password: "MyPassWord!"
```

- variables du groupe dans le fichier **host** ;
- variables du groupe dans les fichiers **group_vars** ;
- variables de la machine au niveau du fichier **host** ;
- variables de la machine au niveau du fichier **host_vars** ;
- variables se trouvant dans un fichier YAML (`-e @fichier.yml`) ;
- variables directement passées à Ansible (`-e variable=valeur`).

Inventaire ansible (5/5): Application des variables

TP-4: Inventaire et variable

- Créez un cluster (1 ansible et 1 client)
- Créez un fichier hosts.ini au format INI avec les modalités d'inventaire suivant
 - Tous les hôtes via le groupe « all » devront avoir pour login user admin
 - Le client devra faire partie d'un groupe appeler « prod »
 - Le mot de passe à utiliser pour toutes connexion ssh devra être admin pour toutes les machines du groupe « prod »
 - La variable « env » devra être égale à « production » pour toutes les machines du groupe « prod »
- Créez ensuite un fichier hosts.yaml, version yaml du fichier ini
- Testez les commandes ad-hoc de ping et setup avec les deux fichiers d'inventaire

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- **Playbook**
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Playbook (1/3): Structure

```
- name: "Apache Installation"
  hosts: all
  tasks:
    - name: "Install apache package"
      yum:
        name: "httpd"
        state: "present"
    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
    - name: "Allow http connections"
      firewallld:
        service: "http"
        permanent: yes
        state: "enabled"
    - name: "Copy test.html"
      copy:
        src: "test.html"
        dest: "/var/www/html"
        owner: "apache"
        group: "apache"
```

Playbook (2/3): Variables

- Surcharge des host et group vars
- Variables à partir d'un fichier

```
- name: "Generate html file for each host"
hosts: all
gather_facts: yes
vars:
  host_inventory: "central-inventory"
  inventory_path: "/var/www/html/inventory"
tasks:
  - name: "Create template directory"
    file:
      path: "{{inventory_path}}"
      owner: "apache"
      group: "apache"
      mode: "0755"
      state: "directory"
    delegate_to: "{{host_inventory}}"
  - name: "html file generation"
    template:
      src: "machine.html.j2"
      dest: "{{inventory_path}}/{{inventory_hostname}}.html"
    delegate_to: "{{host_inventory}}"
```

Playbook (3/3): Simple Ansible Project

```
production          # inventory file for production servers
staging              # inventory file for staging environment

group_vars/
  group1.yml         # here we assign variables to particular groups
  group2.yml
host_vars/
  hostname1.yml      # here we assign variables to particular hosts
  hostname2.yml

library/             # if any custom modules, put them here
module_utils/        # if any custom module_utils to support modules
filter_plugins/      # if any custom filter plugins, put them here

site.yml             # master playbook
webservers.yml       # playbook for webserver tier
dbservers.yml        # playbook for dbserver tier
```

- Inventaire par environnement
- Utilisation des `group_vars` et `host_vars`
- Un playbook pour chaque grande action à faire

TP-5: Déployez un conteneur apache

- Créez un cluster (1 ansible et 1 client)
- Créez un dossier webapp qui va contenir tous les fichiers de notre projet
- Créez un fichier d'inventaire appelé prod.yml (au format yaml) contenant un groupe prod avec comme seul membre notre client
- Créez un dossier group_vars qui va contenir un fichier nommé prod qui contiendra les informations de connexion à utiliser par ansible (login et mot de passe)
- Créez un playbook nommé deploy.yml permettant de déployez apache à l'aide de docker sur le client (l'image à utiliser est httpd et le port à exposer à l'extérieur est le 80)
- Vous avez le droit d'installer tout prérequis que vous jugerez nécessaire à l'aide du module yum
- Vérifiez la syntaxe de votre playbook avec la commande ansible-lint (installez là si elle n'est pas disponible)
- Vérifiez qu'après l'exécution de votre playbook le site par défaut de apache est bien disponible sur le port 80
- Explorez les options de debug de ansible
- Afin de conserver votre travail, poussez le sur votre github

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Templating, loop, condition (1/3): Templating

```
<html>
  <head>
    <title>Machine {{inventory_hostname}}</title>
  </head>
  <body>
    <p>Cette machine s'appelle {{inventory_hostname}}</p>
  </body>
</html>
```

```
- name: "Generate html file for each host"
  hosts: all
  connection: local
  tasks:
    - name: "html file generation"
      template:
        src: "machine.html.j2"
        dest: "{{playbook_dir}}/{{inventory_hostname}}.html"
```

```
---  
- hosts: ubuntu_webserver  
  become: yes  
  tasks:  
    - name: Create new users  
      user:  
        name: '{{ item }}'  
        state: present  
  
    loop:  
      - john  
      - mike  
      - andrew
```

Templating,
loop,
condition
(2/3): loop

```
---  
  
- hosts: group1  
  tasks:  
    - name: Enable Selinux  
      selinux:  
        state: enabled  
1 when: ansible_os_family == 'Debian'  
  register: enable_selinux 2  
  
    - debug:  
      msg: "Selinux Enabled. Please restart the server to apply changes."  
      when: enable_selinux.changed == true 3  
  
- hosts: group2  
  tasks:  
    - name: Install apache  
      yum:  
        name: httpd  
        state: present  
4 when: ansible_system_vendor == 'HP' and ansible_os_family == 'RedHat'
```

Templating,
loop,
condition
(3/3):
Condition

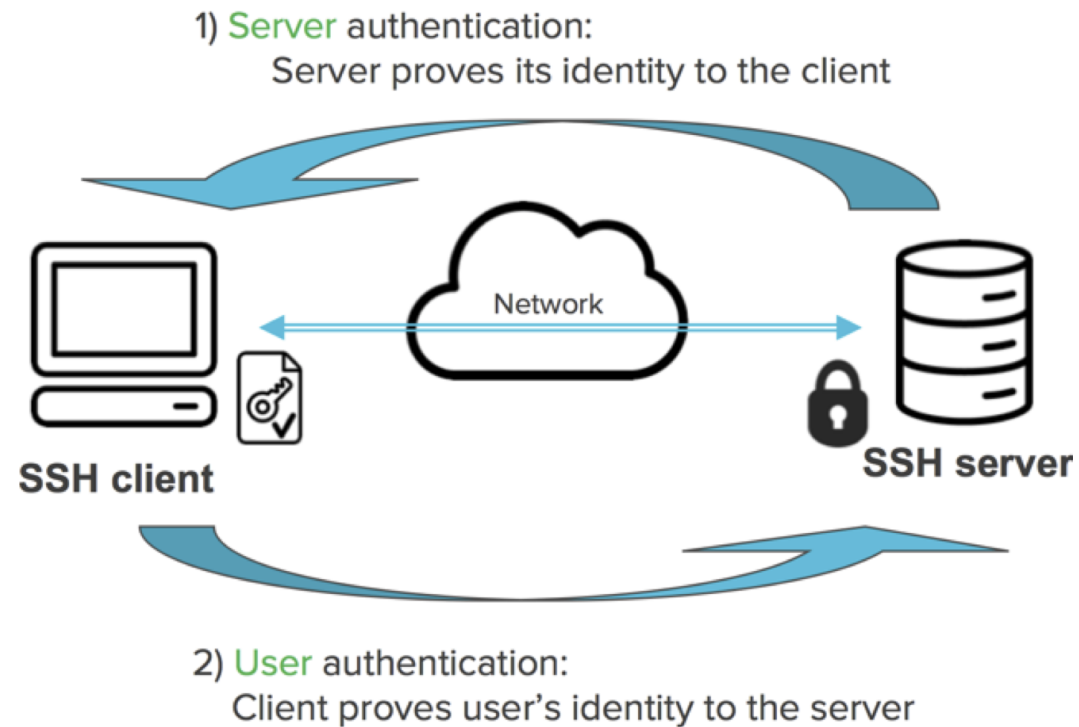
TP-6: Déployez votre propre site

- Créez un cluster (1 ansible et 1 client) et récupérez le code webapp stocké sur github
- Créez un dossier templates avec comme contenu un fichier index.html.j2 qui devra permettre d'afficher sur le site "bienvenue sur -nom de l'hôte-"
- Modifiez le playbook deploy.yml afin:
 - D'utiliser les loop pour installer git et wget à l'aide du module yum
 - D'utiliser une condition afin de n'installer wget et git uniquement si nous sommes sur un système CentOS
 - Modifiez le site internet par défaut, en copiant un template index.html.j2 qui devra afficher "bienvenue sur -nom de l'hôte-" –nom de l'hôte- étant récupéré dans les variables fournies par ansible
 - Modifiez le déploiement du conteneur apache pour qu'il monte le fichier index.html dans le repertoire dans lequel il lit le site internet par défaut (consulter la Doc de apache sur le Dockerhub), vous utiliserez la notion de montage de volume
- Vérifiez que votre site est bien conforme aux attentes
- Poussez votre travail que votre github privé car l'environnement dans lequel vous travaillez est éphémère

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

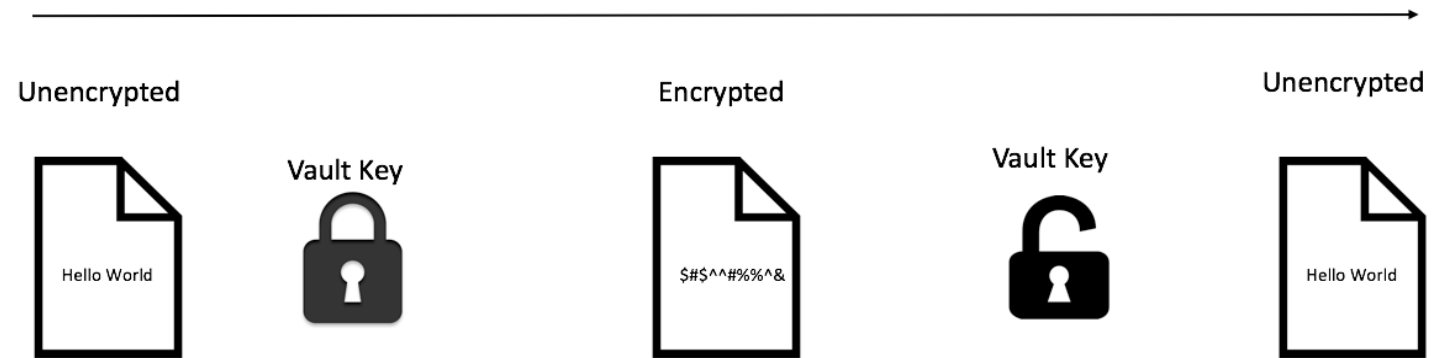
Sécurité (1/2): Paire de clé



- ssh-keygen
- authorized_keys
- ansible_ssh_common_args='-o StrictHostKeyChecking=no'

Sécurité (2/2): Vault

AES Symmetrical Key Encryption



```
ubuntu@ip-10-0-2-54:~$ cat secrets.txt
$ANSIBLE_VAULT;1.1;AES256
64633735613636656665343436316337626635316161323130323236343039303935656132613937
3031353664346635613431616631663731313339346231390a343233666163633433613232613631
32353163313033323739656664376536333038633038326639653738333435383961666233333661
3633363037366533610a663565646230353239353462333338623164393361386431316330343962
65643839663737623134306366653239626636303866323030323634656365306165373730353935
31333465323839656564633464663962386666663130373032396363323863633936316264663439
653764323731653964653332366564633739
```

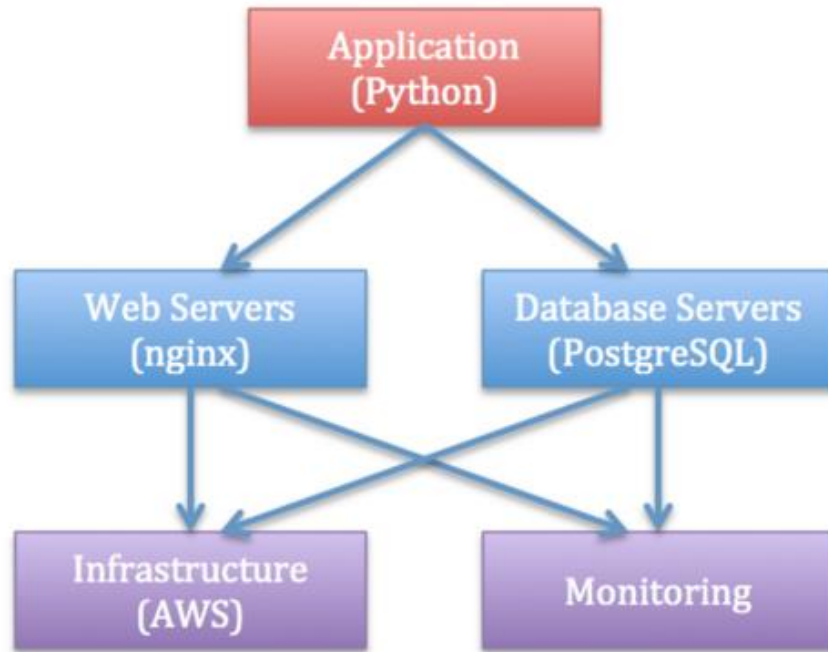

TP-7: Key pair et Vault

- Créez un cluster (1 ansible et 1 client) et récupérez votre code du TP-6 depuis votre github
- Créez un repertoire files qui va lui-même contenir un dossier secrets qui lui-même va contenir un fichier credentials.vault
- Déplacez la variable contenant la variable du mot de passe admin dans ce fichier
- Ensuite vous allez encryptez ce fichier à l'aide de la commande ansible-vault encrypt
- Vous allez modifier le playbook deploy.yml afin de lui demander de charger le fichier vaulté en tant que vars_files
- Générez une paire de clé (ssh-keygen -t rsa) en laissant tous les paramètres par défaut
- Copiez le contenu de la clé publique (id_rsa.pub) dans le fichier /home/admin/.ssh/authorized_host du client
- Modifiez le fichier d'inventaire afin de rajouter cette variable à tous les hôtes (all) ansible_ssh_common_args='-o StrictHostKeyChecking=no
- Lancez votre playbook en rajoutant le paramètre qui vous permettra de fournir la clé vault
- Vérifiez que tout s'est bien passé
- Afin de conserver votre travail, poussez le sur votre github

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- [Rôle ansible](#)
- Mini-projet

Rôle ansible (1/3): Objectif



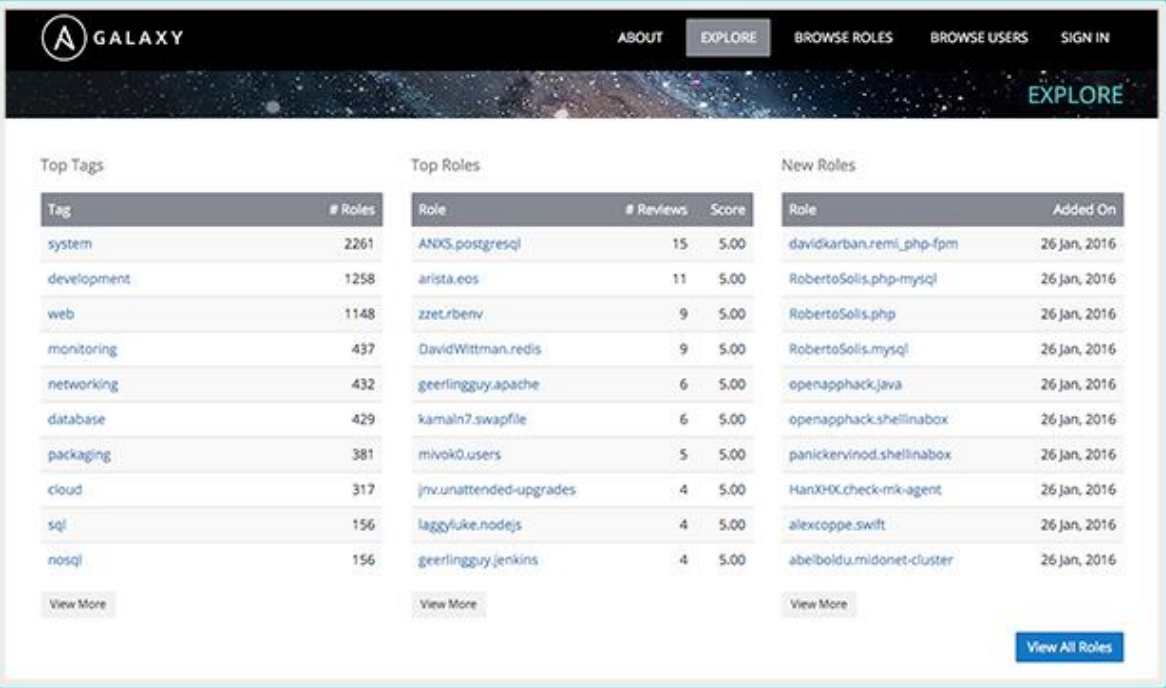
```
roles/  
  app/  
    files/  
    tasks/  
  nginx/  
    templa  
    tasks/  
    vars/  
  postgresql  
    templa  
    tasks/  
    vars/  
  aws/  
    tasks/  
    vars/  
  monitoring  
    tasks/
```

- Templating
- Réutilisable
- Evolutif
- Adaptable

```
├── ansible.cfg
├── hosts
├── roles
│   ├── mongodb
│   │   ├── defaults
│   │   │   └── main.yml
│   │   ├── files
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── meta
│   │   │   └── main.yml
│   │   ├── README.md
│   │   ├── tasks
│   │   │   └── main.yml
│   │   ├── templates
│   │   ├── tests
│   │   │   ├── inventory
│   │   │   └── test.yml
│   │   └── vars
│   │       └── main.yml
```

Rôle ansible
(2/3):
Structure

Rôle ansible (3/3): Galaxy Ansible



The screenshot shows the Ansible Galaxy website interface. At the top, there is a navigation bar with links: ABOUT, EXPLORE (active), BROWSE ROLES, BROWSE USERS, and SIGN IN. Below the navigation bar is a header image with the word "EXPLORE" in green. The main content area is divided into three columns: Top Tags, Top Roles, and New Roles.

Top Tags

Tag	# Roles
system	2261
development	1258
web	1148
monitoring	437
networking	432
database	429
packaging	381
cloud	317
sql	156
nosql	156

[View More](#)

Top Roles

Role	# Reviews	Score
ANXS.postgresql	15	5.00
arista.eos	11	5.00
zzet.rbnb	9	5.00
DavidWittman.redis	9	5.00
geerlingguy.apache	6	5.00
kamain7.swapfile	6	5.00
mivoki0.users	5	5.00
jnv.unattended-upgrades	4	5.00
laggyluke.nodejs	4	5.00
geerlingguy.jenkins	4	5.00

[View More](#)

New Roles

Role	Added On
davidkarban.remi_php-fpm	26 Jan, 2016
RobertoSolis.php-mysql	26 Jan, 2016
RobertoSolis.php	26 Jan, 2016
RobertoSolis.mysql	26 Jan, 2016
openapphack.java	26 Jan, 2016
openapphack.shellinabox	26 Jan, 2016
panickervinod.shellinabox	26 Jan, 2016
HanXH0C.check-mik-agent	26 Jan, 2016
alexcoppe.swift	26 Jan, 2016
abelboldu.midonet-cluster	26 Jan, 2016

[View More](#)

[View All Roles](#)

- Rôle communautaire
- Classé par catégorie
- Activement maintenu
- Chaque rôle est lié à un repo github
- Ansible-galaxy

TP-8: Déployez wordpress

- Créez un cluster (1 ansible et 1 client) et récupérez votre code provenant du TP-7 depuis github
- Créez un playbook wordpress.yml afin de déployez wordpress (sous forme de conteneur docker) sur le client à l'aide du rôle suivant :
<https://github.com/diranetafen/ansible-role-containerized-wordpress.git>
- Lancez le playbook et vérifiez que marche comme sur des roulettes et que vous avez bien wordpress (vous pouvez exposer wordpress sur le port externe de votre choix, 80 ou 8080 ou autre, tant qu'il n'est pas déjà utilisé sur votre machine client par un autre conteneur)

Plan

- Présentation du formateur
- Introduction au DevOps et Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop, condition
- Sécurité
- Rôle ansible
- Mini-projet

Mini-projet: Créez votre propre rôle webapp

- Vous avez reçu la demande d'une autre équipe qui souhaiterait utiliser votre playbook webapp, mais sous forme de rôle car sous cette forme ils pourront mieux variabiliser et adapter à leur situation
- Leur objectif est que votre rôle possède un playbook tests afin de leur permettre de tester rapidement votre rôle et ainsi l'intégrer à leur process de déploiement, exactement comme le rôle wordpress
- La dernière raison pour laquelle vous devez faire ce rôle est que votre entreprise souhaite mettre en place une galaxy privée pour stocker tous les rôles fait dans l'entreprise <https://github.com/diranetafen/ansible-role-containerized-wordpress#example-playbook>
- A la fin de votre travail, poussez votre rôle sur github et envoyez nous le lien de votre repo à easytrainingfr@gmail.com et nous vous dirons si votre solution respecte les bonnes pratiques et si votre rôle est utilisable par une entreprise

Merci et à bientôt sur
eazytraining
