



Support pédagogique Ansible - 6

Sommaire

- Ajout d'un répartiteur de charge
- Mise à jour et réentrance de script
- Ansible Tower
- Supplément
- Quelques projets à suivre

Ajout d'un répartiteur de charge

- Installer le package haproxy avec `apt`
- Configurer le serveur Haproxy
- Activer le service

Ajout d'un répartiteur de charge

Mise au point du rôle

roles/mediawiki/haproxy

```
mediawiki
├── common
│   └── defaults
│       └── main.yml
├── configuration
│   ├── meta
│   │   └── main.yml
│   └── tasks
│       ├── main.yml
│       └── main.yml.cp
├── haproxy
│   ├── defaults
│   │   └── main.yml
│   ├── handlers
│   │   └── main.yml
│   ├── tasks
│   │   └── main.yml
│   └── templates
│       └── mediawiki.cfg.j2
```

Ajout d'un répartiteur de charge

Sous Debian un seul fichier de configuration est nécessaire pour paramétrer haproxy (/etc/haproxy/haproxy.cfg)

Donc, nous allons préparer le fichier template mediawiki.cfg.j2 avec les configurations suivantes :

```
global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

defaults
    log      global
    mode     http
    option   httplog
    option   dontlognull
    timeout  connect 5000
    timeout  client 50000
    timeout  server 50000
```

Ajout d'un répartiteur de charge

On active l'interface Web

```
# Activation des statistiques et du portail Web

{% if activate_webstats|bool %}
listen webstats
    bind 0.0.0.0:8080
    stats enable
    stats hide-version
    stats scope wikilb
    stats scope mediawiki
    stats uri /
    stats realm Haproxy\ Statistics
    stats auth haproxy:{{secret}}
    stats refresh 10s
{% endif %}
```

Ajout d'un répartiteur de charge

Chiffrement du mot de passe de connexion à l'interface statistiques

```
$ ansible-vault encrypt_string 'sdq44sd0SKz!'
```

roles/mediawiki/haproxy/defaults/main.yml

```
# By default, we dont want to expose web statistics
activate_webstats: no
secret: !vault |
    $ANSIBLE_VAULT;1.1;AES256
    6634393338356130376239613136323164383966303330303534323530643638396161
    6535626334626330376638363261383236623562383830640a61383731326135653137
    6533346430363062376336303730336166656339633163383065386530616636366666
    6563656134336430340a37393134623964393062613534623763353935626565646135
    3332
```

Ajout d'un répartiteur de charge

Suite de la configuration avec le paramétrages du port d'écoute et du backend qui contiendra les serveurs Apache

```
listen wikilb
    bind 0.0.0.0:80 # port d'écoute du Proxy
    default_backend mediawiki # backend a utiliser

backend mediawiki # Options du backend
    mode http
    balance roundrobin # Load Balancing algorithm
    option httpchk GET /disabled
    http-check expect ! status 200
    option forwardfor

# Liste des serveurs Apache qui vont composer le cluster
{% for host in groups['apache'] %}
    server {{host}} {{host}}:80 weight 1 maxconn 512 check
{% endfor %}
```


Ajout d'un répartiteur de charge

Mise au point du rôle

tasks/main.yml

```
- name: "install haproxy package"
  apt:
    name: "haproxy"
    state: present

- name: "haproxy configuration"
  template:
    src: "mediawiki.conf.j2"
    dest: "/etc/haproxy/conf.d/mediawiki.conf"
    owner: "root"
    group: "root"
  notify: "restart haproxy"

- name: "enable and start haproxy service"
  service:
    name: "haproxy"
    state: started
    enabled: yes
```

Ajout d'un répartiteur de charge

Définition du Playbook pour l'installation de Haproxy

install-haproxy.yml

```
- name: "Install haproxy"
  hosts: haproxy
  gather_facts: no
  roles:
    - role: "mediawiki/haproxy"
```

Ajout d'un répartiteur de charge

```
$ ansible-playbook -i inventaire install-haproxy.yml -b -K \
    --ask-vault-pass -e activate_webstats=yes
```

```
PLAY [Install haproxy] *****
```

```
TASK [mediawiki/haproxy : install haproxy package] *****
changed: [haproxy]
```

```
TASK [mediawiki/haproxy : haproxy configuration] *****
changed: [haproxy]
```

```
RUNNING HANDLER [mediawiki/haproxy : restart haproxy] *****
changed: [haproxy]
```

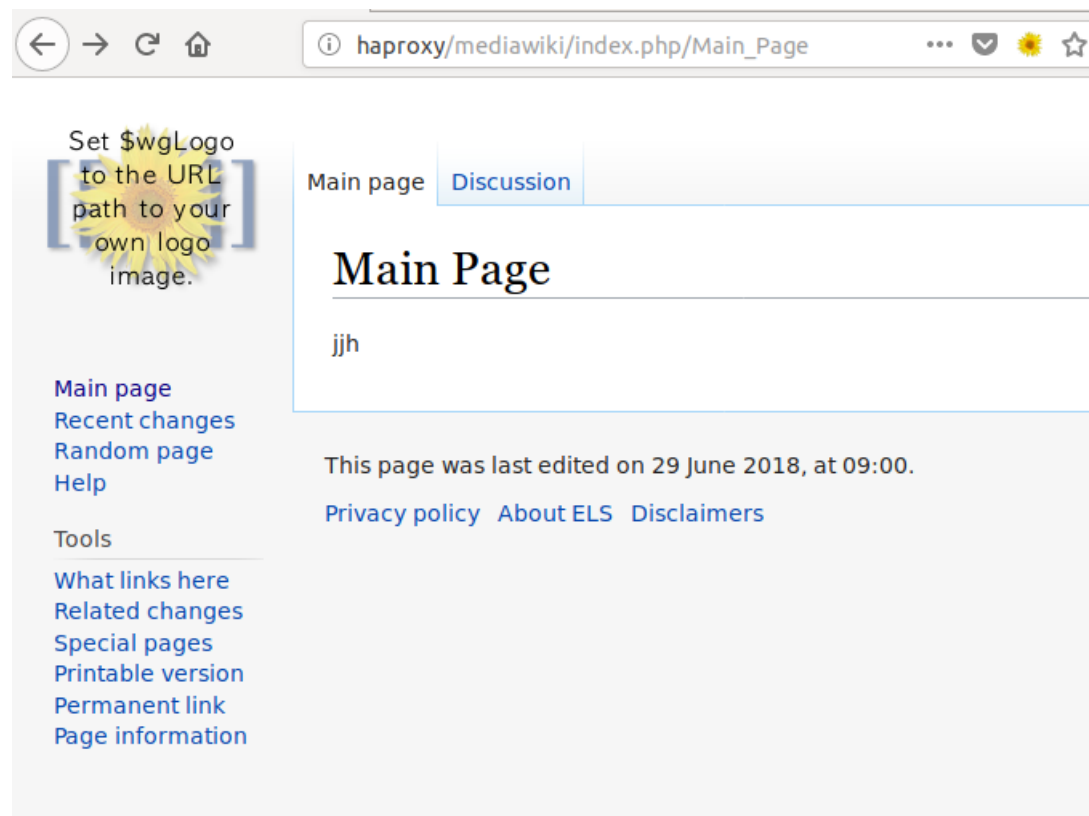
```
TASK [mediawiki/haproxy : enable and start haproxy service] *****
ok: [haproxy]
```

```
PLAY RECAP *****
```

```
haproxy                : ok=4    changed=3    unreachable=0    failed=0
```

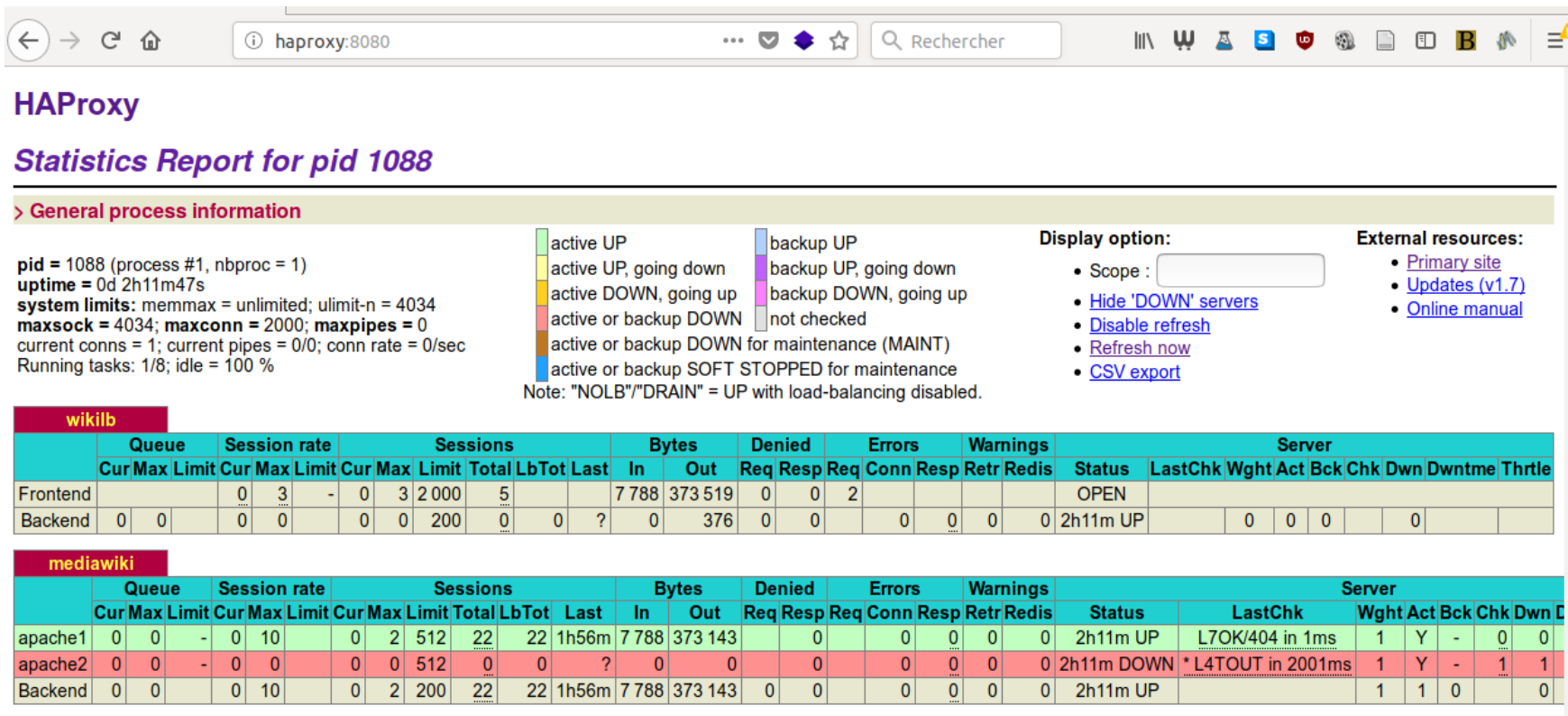
Ajout d'un répartiteur de charge

http://haproxy/mediawiki/index.php/Main_Page



Ajout d'un répartiteur de charge

<http://haproxy:8080>



HAProxy

Statistics Report for pid 1088

> General process information

pid = 1088 (process #1, nbproc = 1)
uptime = 0d 2h11m47s
system limits: memmax = unlimited; ulimit-n = 4034
maxsock = 4034; maxconn = 2000; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/8; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance

backup UP
backup UP, going down
backup DOWN, going up
not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope :
- [Hide 'DOWN' servers](#)
- [Disable refresh](#)
- [Refresh now](#)
- [CSV export](#)

External resources:

- [Primary site](#)
- [Updates \(v1.7\)](#)
- [Online manual](#)

wikilb

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	3	-	0	3	2 000	5			7 788	373 519	0	0	2					OPEN									
Backend	0	0		0	0		0	0	200	0	0	?	0	376	0	0		0	0	0	0	2h11m UP		0	0	0			0		

mediawiki

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server										
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme
apache1	0	0	-	0	10		0	2	512	22	22	1h56m	7 788	373 143		0		0	0	0	0	2h11m UP	L7OK/404 in 1ms	1	Y	-	0	0	
apache2	0	0	-	0	0		0	0	512	0	0	?	0	0		0		0	0	0	0	2h11m DOWN	* L4TOUT in 2001ms	1	Y	-	1	1	
Backend	0	0		0	10		0	2	200	22	22	1h56m	7 788	373 143	0	0		0	0	0	0	2h11m UP		1	1	0			0

Mise à jour et réentrance de script

La version de Mediawiki a changé, vous voulez la mettre à jour, mais elle implique une modification de la base de données.

- Modification des fichiers sources avec la nouvelle version
- Mise à jour de la base de données.

Modification de la version du fichier source dans

`mediawiki/configuration/defaults/main.yml`

```
# archive of mediawiki to download
mediawiki_archive_url: "https://releases.wikimedia.org/mediawiki/
1.29/mediawiki-1.31.tar.gz"
```

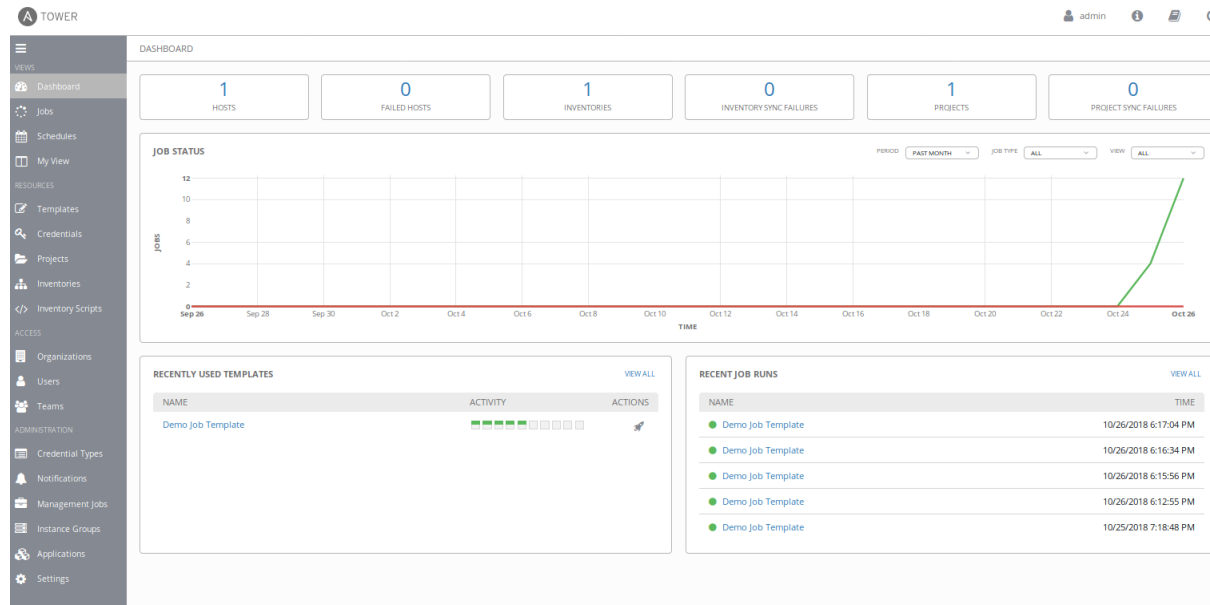
Mise à jour et réentrance de script

Dans le `/mediawiki/configuration/tasks/main.yml` une section `update db` est ajoutée pour mettre à jour la base de données en utilisant le script `update.php` disponible dans les fichiers d'administration de Mediawiki.

Le changement d'état est conditionné par la présence ou non du texte `'...done.'` dans la sortie de la commande `update.php`.

```
- name: "mediawiki db update"
  become: yes
  become_user: "apache"
  command:
    php update.php --quick
  args:
    chdir: "{{mediawiki_maintenance_directory}}"
    # Update need to be launched only once
  run_once: yes
  register: _
  changed_when: "'...done.'" in _.stdout.lower()
```

Ansible Tower



Ansible **Tower** c'est :

- une **API REST**
- un **Web service**
- une **console Web**

Tower est conçu pour rendre Ansible plus simple à utiliser et à centraliser les tâches d'automatisation.

Ansible Tower

- Cette solution commerciale a été lancée en **2016**
- 3 service-level agreements (SLAs)
 - Self-Support (pas de support et de SLA, expire > 10 nodes)
 - Standard (support et SLA: 8 × 5)
 - Premium (support et SLA: 24 × 7)

<https://www.ansible.com/products/tower/editions>

- Ansible Tower **Version 3.3.0**; September 12, 2018
 - Fin de support **2020**
- Ansible Tower **Life Cycle** : 4 versions en 2 ans

Ansible Tower

- En **2017** Red Hat a créer le **project AWX**
 - AWX est conçu pour être un projet **fréquemment mise à jour** et en **évolution constante** en intégrant les **nouveaux développements**
 - Ansible **Tower** est produit en prenant certaines versions de **AWX**
 - AWX project **FAQ** : <https://www.ansible.com/products/awx-project/faq>
 - AWX on **GitHub** : <https://github.com/ansible/awx>
- Projet **semaphore** - User Interface : <https://github.com/ansible-semaphore/semaphore>

Access Control

Lorsque l'organisation IT est complexe et que le nombre de personnels est important, Tower peut faciliter la gestion des habilitations et contrôler l'exécution des Playbook.

Les équipes n'ont pas accès direct aux hôtes. Cela réduit la complexité et augmente la sécurité.

- **Utilisateur** (normal, auditeur ou administrateur)
- **Team** : organisation logique avec des utilisateurs, des projets, des informations d'identification et des autorisations associés

The screenshot displays the Ansible Tower web interface for user management. The left sidebar contains navigation links for Schedules, My View, Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration (Credential Types, Notifications, Management Jobs, Instance Groups, Applications, Settings). The main content area shows the 'alexandre' user details. The 'DETAILS' tab is selected, showing a form with the following fields: FIRST NAME (Alexandre), LAST NAME (Domont), EMAIL (a@a.com), USERNAME (alexandre), PASSWORD (with a 'SHOW' toggle), and CONFIRM PASSWORD (with a 'SHOW' toggle). The USER TYPE is set to 'Normal User'. There are 'CANCEL' and 'SAVE' buttons at the bottom right of the form. Below the form, there is a 'USERS' section with a search bar and a table listing users. The table has columns for 'USERNAME', 'FIRST NAME', 'LAST NAME', and 'ACTIONS'. Two users are listed: 'admin' and 'alexandre'. The 'alexandre' user is highlighted with a blue bar. The 'ACTIONS' column for 'alexandre' shows a pencil icon for editing and a trash icon for deleting. The bottom right corner of the table indicates 'ITEMS 1 - 2'.

Projects

Un **projet** est un espace qui contient des **Playbooks** et des **rôles** liés logiquement.

The screenshot shows the Ansible Tower web interface. On the left is a sidebar with navigation links: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION (Credential Types, Notifications, Management Jobs, Instance Groups). The main content area is titled 'PROJECTS / Demo Project'. It features a form for configuring the 'Demo Project' with tabs for DETAILS, PERMISSIONS, NOTIFICATIONS, JOB TEMPLATES, and SCHEDULES. The form includes fields for NAME (Demo Project), DESCRIPTION, ORGANIZATION (Default), SCM TYPE (Git), SCM URL (https://github.com/ansible/ansible-tower-samples), SCM BRANCH/TAG/COMMIT, SCM CREDENTIAL, SCM UPDATE OPTIONS (Clean, Delete on Update, Update Revision on Launch), and CACHE TIMEOUT (SECONDS) (0). At the bottom right of the form are CANCEL and SAVE buttons. Below the form is a 'PROJECTS' section with a search bar and a table listing projects. The table has columns for NAME, TYPE, REVISION, LAST UPDATED, and ACTIONS. One project is listed: Demo Project, Git, 347e44f, 10/26/2018 6:16:58 PM.

NAME	TYPE	REVISION	LAST UPDATED	ACTIONS
Demo Project	Git	347e44f	10/26/2018 6:16:58 PM	

Le projet peut être lié à un source-control management (SCM) type **Git**. Le code source du projet évolue donc avec le rythme des modifications du code.

Inventory Management

Dans ces inventaires, il est possible d'ajouter des variables par défaut et ajouter manuellement des groupes et des hôtes.

Il est possible de permettre d'interroger les hôtes de manière dynamique à partir d'une source (par exemple, d'un VMware vCenter), et mettre ces hôtes dans un groupe.

The screenshot displays the Ansible Tower web interface. On the left is a sidebar menu with categories: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION (Credential Types, Notifications, Management Jobs, Instance Groups). The main content area is titled 'INVENTORIES / Demo Inventory'. It features a form for configuring the 'Demo Inventory' with tabs for DETAILS, PERMISSIONS, GROUPS, HOSTS, SOURCES, and COMPLETED JOBS. The form includes fields for NAME (set to 'Demo Inventory'), DESCRIPTION, ORGANIZATION (set to 'Default'), INSIGHTS CREDENTIAL, and INSTANCE GROUPS. There is a section for VARIABLES with a 'VARIABLE JSON' button and a text area containing '1 ---'. At the bottom right of the form are 'CANCEL' and 'SAVE' buttons. Below the form is a table view with tabs for INVENTORIES and HOSTS. The table has columns for NAME, TYPE, ORGANIZATION, and ACTIONS. It shows one entry: 'Demo Inventory' of type 'Inventory' under the 'Default' organization.

NAME	TYPE	ORGANIZATION	ACTIONS
Demo Inventory	Inventory	Default	[Icons]

Les Job Template

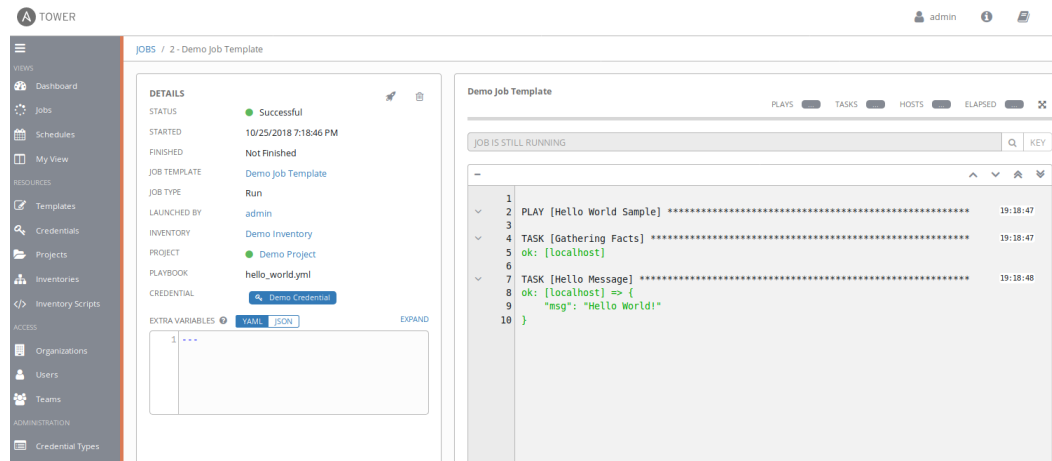
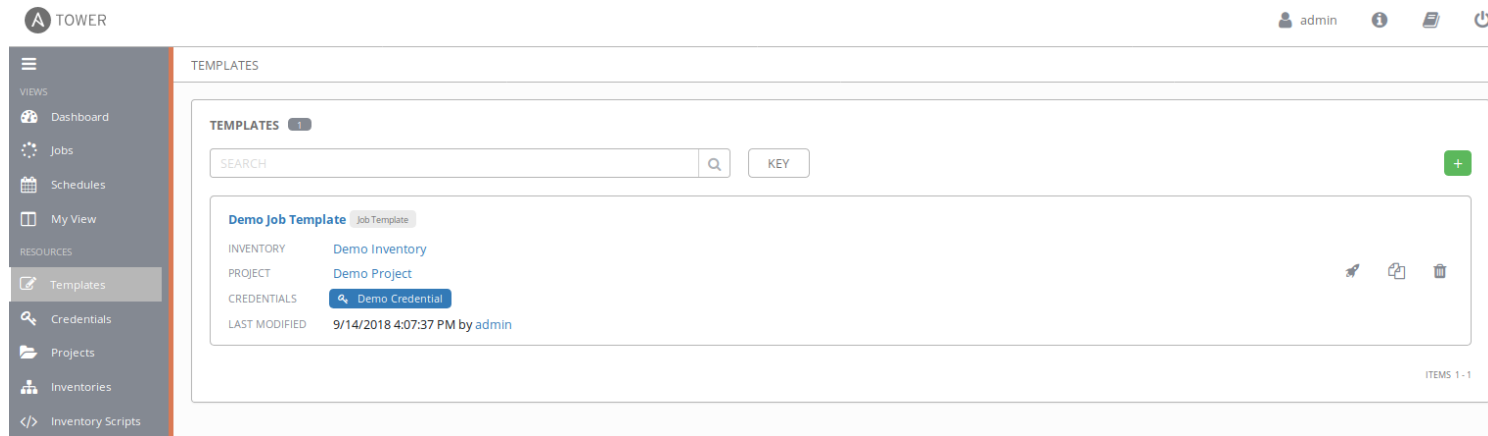
Les **Job Templates** permettent de relier des **projets** à des **inventaires**. Ils définissent comment les utilisateurs sont autorisés à exécuter un Playbook vers des cibles spécifiques à partir d'un inventaire sélectionné.

The screenshot shows the 'Demo Job Template' configuration page in Ansible Tower. The interface includes a sidebar with navigation links for Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration (Credential Types, Notifications, Management Jobs, Instance Groups). The main content area is titled 'TEMPLATES / Demo Job Template' and features tabs for DETAILS, PERMISSIONS, NOTIFICATIONS, COMPLETED JOBS, and SCHEDULES. The 'DETAILS' tab is active, showing various configuration fields:

- NAME:** Demo Job Template
- DESCRIPTION:** (empty)
- JOB TYPE:** Run (with a 'PROMPT ON LAUNCH' checkbox)
- INVENTORY:** Demo Inventory (with a 'PROMPT ON LAUNCH' checkbox)
- PROJECT:** Demo Project
- PLAYBOOK:** hello_world.yml
- CREDENTIAL:** Demo Credential (with a 'PROMPT ON LAUNCH' checkbox)
- FORKS:** DEFAULT
- LIMIT:** (empty, with a 'PROMPT ON LAUNCH' checkbox)
- VERBOSITY:** 0 (Normal) (with a 'PROMPT ON LAUNCH' checkbox)
- JOB TAGS:** (empty, with a 'PROMPT ON LAUNCH' checkbox)
- SKIP TAGS:** (empty, with a 'PROMPT ON LAUNCH' checkbox)
- LABELS:** (empty)
- INSTANCE GROUPS:** (empty)
- SHOW CHANGES:** OFF (with a 'PROMPT ON LAUNCH' checkbox)

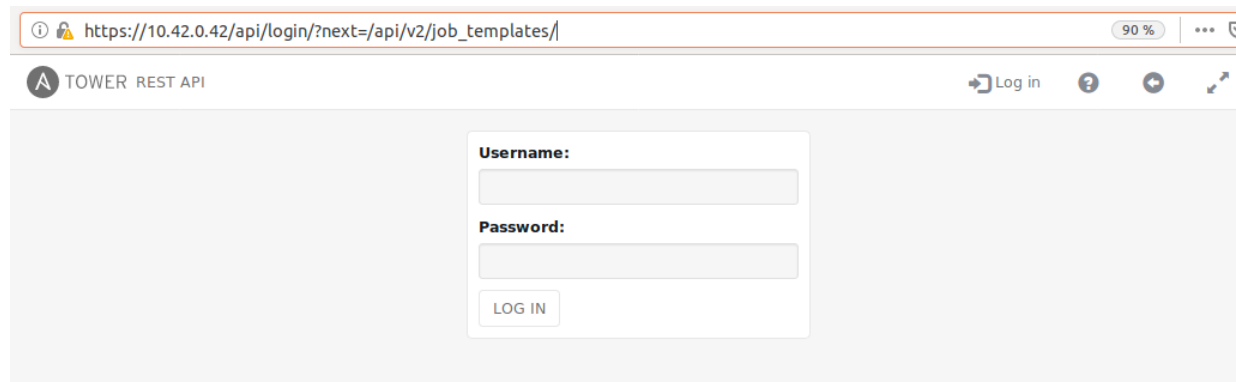
Below these fields are 'OPTIONS' (Enable Privilege Escalation, Allow Provisioning Callbacks, Enable Concurrent Jobs, Use Fact Cache) and 'EXTRA VARIABLES' (YAML/JSON tabs, with a 'PROMPT ON LAUNCH' checkbox). The 'EXTRA VARIABLES' section shows a single variable '1' with a value '...'.

Lancer un Job



Le source **Git** doit être synchronisé au moins une fois avant de lancer le Job

API et Ansible Tower CLI



- Une **API RESRful** est disponible sur le Tower à l'adresse `https://[10.42.0.42]/api` , elle permet potentiellement de s'intégrer à des pipelines de déploiement continus.
- Pour piloter l'API il y a un CLI : `ansible-tower-cli`

La documentation de l'API est disponible ici :

<https://docs.ansible.com/ansible-tower/latest/html/towerapi/index.html>

API et Ansible Tower CLI

Installation du CLI : `ansible-tower-cli`

```
$ pip install ansible-tower-cli
```

Exporte le chemin d'installation du client : `tower-cli`

```
$ export PATH=$PATH:$HOME/.local/bin
```

Création d'un utilisateur

Avant de créer un utilisateur, il faut ajouter à la configuration du CLI les paramètres de connexion à l'API

```
$ tower-cli config host 10.42.0.42  
Configuration updated successfully.
```

```
$ tower-cli config username admin  
Configuration updated successfully.
```

```
$ tower-cli config password GbVEHabyEW9H  
Configuration updated successfully.
```

Pour la démonstration, on désactive la vérification SSL, qui est activée par défaut.

```
$ tower-cli config verify_ssl false  
Configuration updated successfully.
```

Création d'un utilisateur

Vérification de la configuration

```
$ tower-cli config

# User options (set with `tower-cli config`; stored in ~/.tower_cli.cfg).
host: 10.42.0.42
username: admin
password: GbVEHabyEW9H
verify_ssl: False

# Defaults.
use_token: False
verbose: False
certificate:
format: human
color: True
description_on: False
oauth_token:
```

Création d'un utilisateur

```
$ tower-cli user create --username alex --password 'hsgtts7QG'  
--email 'a@b.com' --first-name alex --last-name alex
```

```
changed: true  
id: 3  
type: user  
url: /api/v2/users/3/  
related:  
  named_url: /api/v2/users/alex/  
  admin_of_organizations: /api/v2/users/3/admin_of_organizations/  
  authorized_tokens: /api/v2/users/3/authorized_tokens/  
[...]  
created: '2018-10-26T15:57:49.299116Z'  
username: alex  
first_name: alex  
last_name: alex  
email: a@b.com  
is_superuser: false  
is_system_auditor: false  
ldap_dn: ''  
external_account: null  
auth: []
```

Création d'un utilisateur

Liste des utilisateurs

```
$ tower-cli user list --format human
```

```
== =====  
id username      email           first_name last_name is_superuser is_system_audit  
== =====  
1 admin      admin@example.com           true           fal  
2 alexandre  a@a.com        Alexandre  Domont       false          fal  
3 alex       a@b.com        alex       alex         false          fal  
== =====
```

Launch a Job

Liste les Job présents sur le Tower

```
$ tower-cli job_template list --format human
```

```
== =====  
id      name      inventory project  playbook  
== =====  
5 Demo Job Template      1      4 hello_world.yml  
== =====
```

Il est possible de filtrer les Job avec l'option `--query`

```
$ tower-cli job_template list --query name__icontains Demo --format human
```

```
== =====  
id      name      inventory project  playbook  
== =====  
5 Demo Job Template      1      4 hello_world.yml  
== =====
```

Launch a Job

Le Job est lancé avec l'option suivante :

```
$ tower-cli job launch --job-template 5 --format human
```

Resource changed.

```
== =====  
id job_template          created              status elapsed  
== =====  
5           5 2018-10-26T16:12:44.698572Z pending 0.0  
== =====
```

Pour lancer et afficher l'exécution (monitor) en même temps, il faut utiliser la commande `jq`

```
$ tower-cli job monitor $(tower-cli job launch --job-template 5  
--format json | jq '.id')
```

Launch a Job

```
-----Starting Standard Out Stream-----

PLAY [Hello World Sample] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Hello Message] *****
ok: [localhost] => {
    "msg": "Hello World!"
}

PLAY RECAP *****
localhost          : ok=2    changed=0    unreachable=0    failed=0

-----End of Standard Out Stream-----
changed: true
id: '14'
type: job
url: /api/v2/jobs/14/
related:
  created_by: /api/v2/users/1/
  labels: /api/v2/jobs/14/labels/
  inventory: /api/v2/inventories/1/
  project: /api/v2/projects/4/
```


Supplément

Include

Pour structurer les actions dans plusieurs fichiers de configuration, il est possible de faire appel à `include` .

```
acme_corp/  
├─ configure.yml  
├─ provision.yml  
└─ site.yml
```

```
$ cat site.yml
```

```
---
```

```
- include: provision.yml  
- include: configure.yml
```

pre_tasks et post_tasks

Toujours pour apporter de la lisibilité et de la cohérence, des tâches peuvent être exécutées avant et après un rôle.

```
- name: "MediaWiki apache configuration"
hosts: apache
[...]
pre_tasks:
  - name: "disable this Apache instance"
    file:
      path: "/var/www/html/disabled"
      state: touch
  - name: "wait haproxy to ignore this host"
    pause:
      seconds: 2
roles:
  - role: "mediawiki/configuration"
post_tasks:
  - name: "enable this Apache instance"
    file:
      path: "/var/www/html/disabled"
      state: absent
```

hostvars

Ansible prévoit aussi des variables 'magiques' comme `hostvars` qui référence toutes les variables de toutes les machines.

```
ansible -m debug -a 'var=hostvars.localhost' localhost
```

```
ansible -m debug -a 'var=hostvars.localhost.ansible_version' localhost
```

```
localhost | SUCCESS => {
  "hostvars.localhost.ansible_version": {
    "full": "2.5.1",
    "major": 2,
    "minor": 5,
    "revision": 1,
    "string": "2.5.1"
  }
}
```

Liste des taches d'un playbook

```
$ ansible-playbook -i hosts vim-mouse-filget.yml --list-tasks
```

Local Actions

```
tasks:
- name: take out of load balancer pool
  local_action: >
    command /usr/bin/take_out_of_pool {{ inventory_hostname }}

- name: update application
  yum: name=acme-web-stack state=latest

- name: add back to load balancer pool
  local_action: >
    command /usr/bin/take_out_of_pool {{ inventory_hostname }}
```

Les taches peuvent être lancées étape par étape

```
$ ansible-playbook provision.yml -i hosts --step
```

```
> Perform task: TASK: setup (y/n/c): n  
> Perform task: TASK: First task (y/n/c): n  
> Perform task: TASK: Second task (y/n/c): y
```

Connaître les variables à déclarer dans l'inventaire

https://docs.ansible.com/ansible/2.5/user_guide/intro_inventory.html#list-of-behavioral-inventory-parameters

Log Ansible dans Syslog du serveur

Quelques projets à suivre

Mitogen: une partie du projet est une extension pour Ansible qui améliore les performances, entre autre en utilisant un interpréteur Python persistant sur les nœuds managés (install, upgrade, and manage OpenShift clusters - Kubernetes utilise mitogen)

<https://mitogen.readthedocs.io/en/latest/ansible.html>

Molecule: indispensable, ce projet facilite le test de playbooks et de rôles.

<https://github.com/ansible/molecule>

ARA, permet d'enregistrer et centraliser les journaux d'exécution via un plugin Ansible de type callback et de les rendre accessibles à l'aide d'une interface Web;

<https://github.com/openstack/ara>