R.Ex. 

Réseau SARI – 2 juillet 2018

G.Mella - OSUG

Sommaire

- Présentation rapide d'Ansible
- Présentation des cas d'utilisation
- Retours d'expériences

Ansible – présentation

- Projet récent (2012) racheté par RedHat
  - Aujourd'hui `pip install ansible` vous installe la version 2.6
  - Stable et largement utilisé / documenté
- Classé Infra-as-code, DevOps
- Solution de déploiement sur (une ou) plusieurs machines
  - Pas d'agent requis
  - Exige côté client :
    - python
    - ssh (**+ sudo**)
  - Description/codage en YAML

Ansible – concepts basiques

- Inventory
  - Liste des machines et les groupe (.ini / yaml / ou dynamique)
    (ex. webserver, dbservers, backends, frontends…)
    par défaut `all` et `ungrouped`
  - Définit des variables => configuration
    (système & applicatif)
- Playbooks
- Roles
- Tasks / modules

Ansible – concepts basiques

- Inventory
- Playbooks
  - Instructions haut niveau (valent documentation)
  - Mettent en musique des rôles/tâches/modules sur un ensemble de machines de l'inventory indiqué.

    ```
    ansible-playbook -i production site.yml
    ```
- Roles
- Tasks / modules

Ansible – concepts basiques

- Inventory
- Playbooks
- Roles
  - Regroupement de traitement
  - Utilisation des variables, boucles, conditions, templates, debug, gestion d'erreurs
  - Possibilité *d'étendre* des rôles : `dependancies`
- Tasks / modules

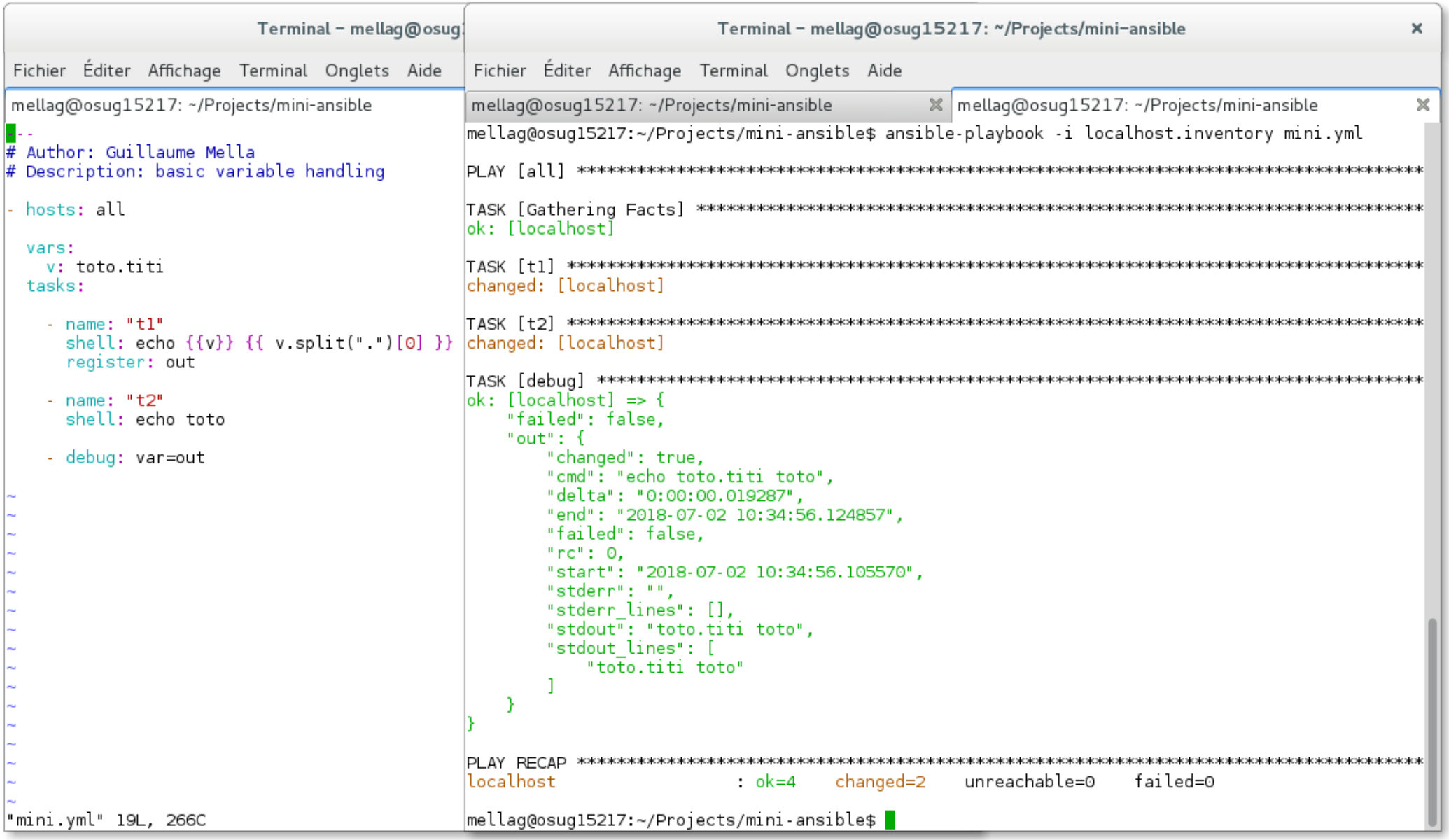
Ansible – concepts basiques

- Inventory
- Playbooks
- Roles
- Tasks / modules
  - Opérations de base
  - Bibliothèque standard très riche, variée et spécialisée !

Ansible – organisation du code

```
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 systems
   hostname2.yml
library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)
site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier
roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case
    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Ansible – utilisation



2 cas d'utilisation

- Construction d'une image virtuelle pour des écoles d'été
- Gestion d'une infrastructure de serveurs pour un service d'observation en astronomie

plusieurs autres potentiels à l'OSUG

Construction VMs école d'été

- Utilisation de Vagrant
  - provisionning ansible
  - VM virtualbox

```
$ vagrant up
$ vagrant provision
$ vagrant package
```

- Configuration d'un poste de travail avec les données et logiciels utiles aux T.P.
- Permet de repartir sur une base ajustable toute automatisée

  => mise au point simplifiée

  => gain de temps car gestion en configuration
- A améliorer : généralisation / découplage des rôles des applications scientifiques

Infrastructure serveurs JMMC

- Objectif 2016 : passer d'un serveur monolithique à une infrastructure serveur plus modulaire
- Avec Raphaël Jacquot (avant le gitlab gricad *)
- Etat courant :
  - de nouveaux services instanciés, d'anciens services portés et redirection vers l'ancien serveur pour le reste.
  - Frontal HAProxy + serveurs backends (VM dédiées + containers docker) + contrôleur

    « Mixte orchestration pour containers pas prêts/adaptés aux micro-services en production »
  - Preprod + prod + legacy server (~15 VM)

(*) https://gricad-gitlab.univ-grenoble-alpes.fr/OSUG/JMMC/jmmc-srv-infra

Retours – courbe d'apprentissage

- Plus compliqué que le shell
  - Mais le potentiel est énorme

    (make, shell, ssh, librairies en un tout)
  - Beaucoup de doc.
  - Debug / pas à pas /dry-run (ca ne vaut pas certains outils de dev…)
  - Il faut s'y mettre et « se forcer »
    - Ne plus modifier la conf sur les serveurs à la main !!
- Une fois ok, le concept d'**idempotence** est très appréciable : lancée une ou plusieurs fois sans intervention le résultat est là et toujours le même :)

Retours – utilisé avec succès

- Packages :
  - apt_repository,
  - apt, pip
- Containers :
  - docker_container, docker_image
- git, svn
- file,stat, get_url, unarchive, sync
- shell, expect

… et les boucles, manipulations de variables, branchements, blocks ...

Retours – détails 'importants'

- Installation de packages couplée aux distribs (apt/yum)
- Gestion des mots de passe / données cryptés
  - Fichiers secrets.dist à renseigner sur place
  - Prompt interactif possible
  - Ansible vault – évite le stockage en clair (pas testé)
- Petits réglages `ansible.cfg` :
  - `stdout_callback=debug` sous `[default]`

« sinon rendez YAML plus verbeux qu'XML avec ansible ;-) »

L'écosystème d'ansible

- jsaispastrop en fait :(
- Ansible Tower
  - centralise la gestion / traçabilité/automatisation

    (gestion de l'inventaire graphique , panneaux de bord, programmation temporelle, notifications…)
- Ansible Galaxy
  - Partage de rôles
    - très nombreuses contributions
    - System, Dev, Network, Cloud, Database, Monitoring, Packaging, Playbook Bundles, Security, Web

Conclusion

- Champs d'action très large **DevOps**
  - Cloud, Clustering, Commands, Crypto, Database, Files, Identity, Inventory, Messaging, Monitoring, Net Tools, Network, Notification, Packaging, Remote Management, Source Control, Storage, System, Utilities, Web Infrastructure, Windows
- Conventions/cadre permettant une réutilisation sur un même socle (concepts basiques)
  - Permet d'aller plus loin vite grâce au développement communautaire
- Gestion de code source en // indispensable
- Les modules marchent bien (parfois dans un sens surprenant), le plus dur reste la créations de rôles sophistiqués*

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

Qu?est?ons