Cloud Computing Applications and Services

Provisioning and Deployment

October 9, 2022

The main goal of this guide is to understand how to provision systems and deploy services in an automatic and reproducible fashion.

For the exercises described next, Ansible may be installed in your computer or on a separate VM. The latter option is mandatory for **Windows** environments.

- Ansible - `http://ansible.com`

Ansible documentation is available at:

- Ansible - `http://docs.ansible.com/ansible/`
- Ansible Getting Started - `https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html#intro-getting-started`
- Example of ansible playbook - `https://github.com/ansible/ansible-examples/tree/master/wordpress-nginx`

Steps

VMs Setup

1. Create two VMs (*e.g.*, VM1 and VM2).

   Optional: Create an independent VM for deployment and provisioning. You may use the Vagrantfile provided along with this guide to create this VM.

Inventory

1. Create an Ansible Inventory file (*e.g.*, hosts.inv) with a group named *app* that includes VM1, and a group named *db* that includes VM2.
2. Use the Ansible ping module to check the app group's VM.

```
ansible -i "inventory_file" -u "username" app -m ping
```

Note 1: *username* identifies the user executing tasks at the groups' hosts.
Note 2: at the inventory file one can specify the following group variable to avoid the key host checking process:

```
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

3. Use the Ansible ping module to gather information from all VMs.
   e.g., ansible -i "inventory_file" -u "username" all -m ping

Playbook

Goal: create an Ansible Playbook, named *deploy-swap.yml*, that will be used for installing the Swap application and all its dependencies. In terms of structure, the solution can be based on the one shown at Figure 1.

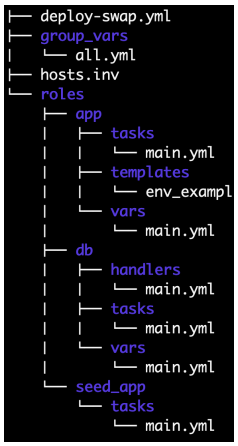

Figure 1: Ansible's structure example for this guide. It has 3 main roles: one for deploying MySQL (db), another for deploying Swap (app), and another for seeding Swap with sample data (seed_app).

1. Start by creating and testing an Ansible Role for installing the MySQL database at VM2 (*i.e.*, db group).

   Hints:

   (a) Check again the *Case-study application: Swap* practical guide for instructions on how to deploy MySQL.
   (b) Explore the *apt* module for installing the *mysql-server* and *python3-mysqldb* packages. The latter package is required for using the *mysql_user* module mentioned below.

(c) Explore the *lineinfile* module to update MySQL's bind address configuration.

(d) Explore Ansible Handlers and the *service* module to restart the MySQL database server after updating the previous configuration.

(e) Explore the *mysql_db* module to create Swap's database.

(f) Explore the *mysql_user* module to create a MySQL user.

2. Test the first version of the Playbook with:

```
ansible-playbook -i "inventory_file" -u "username" deploy-swap.yml
```

Note 1: the *-K* flag can be used when Ansible needs to request sudo password from users.
Note 2: after running the Playbook, login (ssh) into VM2 and check if the deployment steps were executed correctly. For example, by double checking the changed configuration files and ensuring that MySQL is running with the command *systemctl status mysql*.

3. Create a second role for installing the Swap application at VM1 (*i.e.*, app group).

   Hints:

   (a) Check again the *Case-study application: Swap* practical guide for instructions on how to deploy Swap.

   (b) Explore the *apt-repository* and *apt* modules to add external apt repositories and to install Swap's package dependencies (*i.e.,* PHP, nodejs, composer, npm).

   (c) Explore the *git* module to clone Swap's git repository.

   (d) Explore Ansible Templates, and the *template* module, to update Swap's configurations (*i.e.*, .env file).

   (e) Explore the *shell* module to run generic shell commands.

   (f) The Swap server can be executed in background with the following shell command

   ```
   nohup php artisan serve --host=0.0.0.0 > app_out.log 2>&1 &
   ```

4. Test the updated version of the Playbook. After executing all tasks you should be able to access the Swap application through your browser.

Improvements and Considerations

1. Note that the two roles have variables in common (*e.g., database name, user, ...*). Avoid duplicate variable definitions by spefifying them at a shared configuration (*e.g., group_vars* folder at Figure 1).

2. Explore using ansible tags in your playbook for different main tasks (*e.g.*, database and application deployment).
   Then use the flags

   ```
   --tags and --skip-tags
   ```

   to run only one of the main tasks, or exclude a main task, when executing your playbook.

3. Create a new role, named *app_seed*, with the sole purpose of seeding Swap's database with sample data (see Guide 1). Swap's database should only be seeded with sample data if the flag *(-e seed_data=true)* is specified when running the Playbook.

4. As a practical example, after solving Steps 2 and 3, and assuming that MySQL deployment was associated with the tag *dbinstall* and the Swap deployment was associated with the tag *appinstall*, one should be able to run:

   ```
   ansible-playbook (...) --skip-tags dbinstall -e seed_data=true
   ```

   In this example, the database deployment should not be executed, while the Swap application should be installed and seeded with sample data. As another example:

   ```
   ansible-playbook (...) --tags appinstall
   ```

   In this case, again, only the Swap application's deployment should be executed and the application should not be seeded with sample data.

Extra

1. Update your Playbook to also configure the Redis and Email services.

2. Explore Ansible Vault to protect passwords and avoid having these in plaintext (*e.g.*, MySQL user's password).

Learning outcomes

Experiment systems provisioning, deployment and configuration management workflows with Ansible Develop playbooks that hold reproducible provisioning and deployment recipes. Understand the importance of task automation and self documentation.