Cloud Computing Applications and Services
Provisioning and Deployment

2023

Ansible

We have seen in previous guides how to provision virtual machines and use them to serve an application. We have also noticed that manually configuring an application can be a laborious and error-prone task. In this guide, our goal is to understand how to provision systems and deploy services in an automatic and reproducible fashion allowing us to reduce errors and increase the scale of our deployments. To achieve this, we will be using a software tool called Ansible (`http://ansible.com`).

By the end of this guide you should have been able to install Ansible, write an Ansible Playbook (Ansible's instructions and configurations), and use it to deploy Swap. Please refer to the slides "*3 - System Provisioning*" from the theoretical class to remind some Ansible-related topics.

Additional 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`

Tasks

VMs Setup

1. Create 3 VMs (*server1, server2, provisionVM*). In *ProvisionVM*, we will run Ansible for automating the process of deployment and provisioning. The Swap application will be deployed in *server1* and *server2* VMs as done in *Guide 1*.
   Note 1: You may use the Vagrantfile provided along with this guide to create these VMs.
   Note 2: Explore the `scp` command to copy your playbook files to the *ProvisionVM*.

Inventory

1. Create an Ansible Inventory file (*e.g.*, *hosts.inv*) with a group named *app* that includes *server1*, and a group named *db* that includes *server2*.
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 (e.g., *vagrant*).
   Note 2: `<INVENTORY_FILE>` stands for the name of our inventory file (e.g., *hosts.inv*).
   Note 3: In the inventory file use the following group variable to avoid the key host checking:

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

3. Use the Ansible ping module to gather information from all VMs:

   ```
   ansible -i <INVENTORY_FILE> -u <USERNAME> all -m ping
   ```

Playbook

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

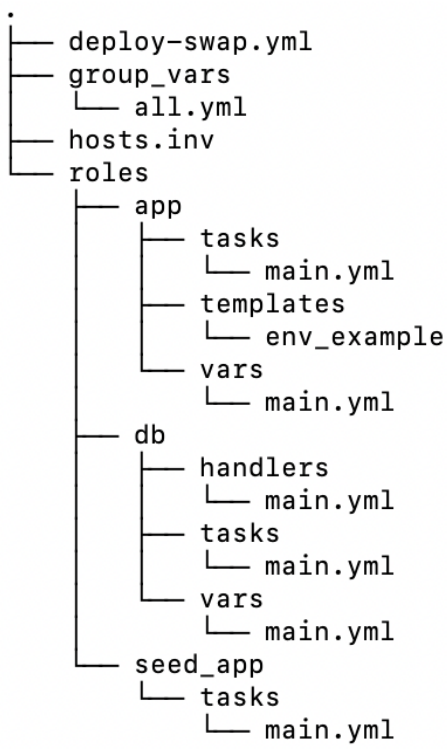

Figure 1: Ansible's structure example for this guide. It has 3 main roles: one for deploying MySQL (*db*), another for deploying Swap (*app*), and another for seeding Swap with sample data (*seed_app*).

Let us do this step by step:

1. Start by exploring the *deploy-mysql.yml* playbook provided along with this guide. What is the goal of this specific playbook?
2. Test this simple Playbook by running the following command:

   ```
   ansible-playbook -i <INVENTORY_FILE> -u <USERNAME> deploy-mysql.yml
   ```

3. Now, write a *deploy-swap.yml* playbook that includes the instructions from the *deploy-mysql.yml* playbook, but following the structure depicted in Figure 1. The playbook should invoke the role *"db"* which includes the tasks, variables and handlers necessary for deploying the MySQL service.

   Hints:

   (a) Explore the different variables defined at the *vars* section. Which variables may be common to *all* groups? Are there specific variables that only make sense to be defined for the *db* group?

   (b) Separate the required steps into handlers and tasks. For example, the *restart mysql* step should be defined as a handler.

4. Test your new playbook:

   ```
   ansible-playbook -i <INVENTORY_FILE> -u <USERNAME> deploy-swap.yml
   ```

5. Complete the playbook by adding a new role that installs the Swap application for the *app* group.

   Hints:

   (a) Check again the *Guide 1* 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 &
   ```

   Note: The output and errors generated by this command are being redirected to the *app_out.log* file, which allows for posteriorly accessing these in case of any error.

6. Test your new Playbook and check if both tasks are working. After the execution of all the tasks you should be able to access the Swap application through your browser.
7. Finally, create a third role for 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.

Additional hints:

1. Note that these roles have variables in common (*e.g.*, *database name, user, ...*). Avoid duplicate variable definitions by specifying them at a shared configuration (*e.g.*, *group_vars* folder at Figure 1).
2. Explore the Ansible Tags, adding tags in your playbook for different main tasks (*e.g.*, database and application deployment). Then, use the flags `--tags <TAGS_NAME>` and `--skip-tags <TAGS_NAME>` to run only one of the main tasks, or exclude a main task, when executing your playbook.
3. As a practical example, 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. Explore Ansible Vault to protect passwords and avoid having these in plaintext (*e.g.*, MySQL user's password).
2. Update your Playbook to also configure the Redis and Email services.

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.