Tampereen yliopisto
Tampere University

Kari Systä

COMP.SE.140 – exercise number 6: Ansible

Versions

1.0	10.10.2022	Initial version, feedback requested.
1.1	16.10.2022	Translated the Finnish text to English
1.2	25.20.2022	Clarifications from the discussion session

Background

Assume that you have a huge number that you need to maintain: update software, take backups, etc. For that should you login to each of these to conduct the maintenance tasks? An alternative way is to use some automation tools. One, but not only, automation tool for this kind of purpose is Ansible.

Ideally, this exercise should use "full" virtual machines as targets, but since our infra does not provide enough those, we use docker containers instead. Preparation of Docker-containers adds complexity but at the same time provides some learnings about containers and networking.

This exercise is not in critical path in the course, so skipping will not lead to difficulties in forthcoming exercises or project. So, this is for getting bonus-like points for the grading.

This exercise will require self-learning and independent problem solving.

The tasks

There are 5 phases that you should do

1) Read Ansible tutorials to understand how it works. A good starting point is: https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html
2) Prepare a docker image that can be used as a target. See details in below.
3) Install Ansible in your computer.
4) Make simple playbook
5) Return result to Plus

Preparation of the Docker Image

The target machine needs to have some certain components, and this your image should be built to include (including example lines for the Dockerfile)

1) SSH for secure communication:

```
RUN apt-get install -y openssh-server
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
```

Kari Systä

2) A user that is used to communicate with the host:
```
RUN useradd -m -s /bin/bash -G sudo -p $(openssl passwd -1 eee) ssluser
```

3) Software components
   Python for running the commands sent by Ansible
```
RUN apt-get install -y python3
```

   sudo for gaining “root” permissions
```
RUN apt-get install -y sudo
```

   net-tools for easy way to enquire the API address
```
RUN apt-get install -y net-tools
```

4) Entrypoint to start the SSH service and wait until terminated.
```
ENTRYPOINT service ssh start && <something that waits, e.g. a small httpserver>
```

An example Dockerfile and some other material on *Intro to Ansible* lecture.

An important note on SSH. Ansible can use two SSH-based methods for accessing the host:

1) Password-based method that is less secure. Although we show how to do it, this method is not recommended. Thus, you get step-by-step instructions on how use this, but there will be a small reduction in points.

2) Key-based method that is more secure and recommended. Student who wants the full points should use this. See the getting started section of Ansible documentation.

The task

After creating the Docker image should test with the following steps

1) Start the image with “run”
2) Check what is the IP address of the host (e.g. docker exec ifconfig)
3) Do the ssh-login to the host (ssh ssluser@ipaddress)
4) Test that python works

Create an Ansible playbook that has two tasks (plays)

1) Ensure that the image has the latest version of git version management system
2) Queries the uptime (linux command uptime) of target host

Test the playbook as follows

1) Start one container from the image, get its IP-address.
   (in case of password-based authentication you need a manual login after start)
2) Ensure that the IP address is in /etc/ansible/hosts (or some other Ansible configuration file you decide to use).
3) Run the playbook
4) Copy the output (O1) – including output of “uptime”
5) Run the playbook again
6) Copy that output, too (O2) – including output of “uptime”
7) Start a second contained from the image, get its IP-address.
8) Ensure that this IP address is in /etc/ansible/hosts (or…) , too.
9) Run the playbook

Kari Systä

10) Copy the output (O3) – including output of “uptime”
11) Run the playbook again
12) Copy that output, too (O4) – including output of “uptime”

Submission

Git link of the code. Use branch “ansible”. The teacher uses

```
git clone -b ansible <the url>
```

Make also sure that the teacher is instructed (e.g. in the beginning of the README.md) how to run. Note, that Ansible is about automation, so make sure that manual work is minimized. Complicated steps are considered partial functionality in evaluation.

The report should have a “report.pdf” or README.rd with the following contents.

- All the copied output (O1,O2,O3,O4)
- Analyze: is the output of uptime-command as expected. If not, why?
- Comments on what was easy and what was difficult.

Hints

Note, if you are using the password-based SSH-connection, the password can be given in the Ansible command as follows:

```
ansible-playbook -b e.yaml --extra-vars "ansible_user=ssluser ansible_password=eee ansible_sudo_pass=eee"
```

Also, unless the hosts are not presented to each other ansible can output:

```
fatal: [172.17.0.4]: FAILED! => {"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this.  Please add this host's fingerprint to your known_hosts file to manage this host."}
```

The first ssh-login to host fixes this.

If you get a warning “[WARNING]: Updating cache and auto-installing missing dependency: python3-apt” it is advisable to install that python3-apt in your Dockerfile. At least for me the auto-install was not done as sudo and thus failed.

~~This looks well written (but I have not read all): https://code-maven.com/ansible-playbook-remove-file~~