Ansible

Ansible is used in this project to provision the VMs in production as well as in the development stage. This doc will give a quick overview over the main concepts that a developer might interact with.

Provisioning with Playbooks

An Ansible Playbook is the entry point for provisioning one or multiple VMs. For example, the playbook for the production environment exists in `provisioning/production.yml`.

To provision in production you use the command

```
ansible-galaxy install -r ./requirements.yml -f

ansible-playbook -i provisioning/hosts --vault-password-
file=~/.klee_vault_password provisioning/production.yml -v
```

The first command downloads some standardised playbooks for execution onto your local machine. The second command specifies an inventory file, the location of the Ansible Vault password and the Playbook.

Conveniently, provisioning in Ansible is idempodent and efficient. Even if you provision with the same playbook multiple times, Ansible only actually executes a task if it would change the VM.

Inventory File

The inventory file allows grouping between the different VMs. It can look like this:

```
[master]
cloud-vm-41-210.doc.ic.ac.uk:22

[worker]
cloud-vm-40-70.doc.ic.ac.uk:22
cloud-vm-41-188.doc.ic.ac.uk:22

[testing]
cloud-vm-41-189.doc.ic.ac.uk:22

[all:children]
master
worker
testing

[all:vars]
ansible_ssh_user=dg3718
```

When provisioning, the rules can be applied to one or more groups in parallel. In this case when provisioning the group `worker` two VMs are provisioned at the same time.

The Playbook

The playbook is what defines which hosts (according to the inventory file's groups) are provisioned how.

```
- hosts: master
  remote_user: vagrant
  become: yes
  vars_files:
    - vars/secrets.yml
    - vars/master-prod.yml
    - vars/common.yml
  roles:
    - common
    - redis
    - db
    - web
```

This is a part of the `provisioning/production.yml` file. It specifies that the roles should only be executed on the `master` VMs, three variable files four roles should be used for provisioning.

Variable Files

`vars_files` hold the variables for the Ansible tasks and offer a single place where variables can be held. This follows the DRY (Don't Repeat Yourself) principle.

Variables can be substituted with the help of Jinja2 templating.

Roles

Roles hold the actual steps that are being used to provision the VMs. For example, in the `db` role the configuration files for the PostgreSQL database are created according to the variables in the vars_files. Then the custom PostgreSQL Docker image is being built and also run.

Roles offer a convenient way to group provisioning steps.

Vault for Secrets

One convenient feature of Ansible is the **Ansible Vault**. It can keep files password encrypted so these files can be pushed into version control safely.

One file with secret variables is the `provisioning/vars/secrets.yml` file. It holds secrets such as the PostgreSQL database passwords.

All you need to access this secrets is the password. You can place the password within a file called `.klee_vault_password` into you home directory on your local machine. With the command

```
ansible-vault view --vault-password-file=~/.klee_vault_password
provisioning/vars/secrets.yml
```

you can view the passwords or with

```
ansible-vault edit --vault-password-file=~/.klee_vault_password
provisioning/vars/secrets.yml
```

you can ammend them.