Lab 2

Core Ansible Concepts

During this lab you'll explore some basic concepts of Ansible such as

- Inventories
- Ansible configuration
- Run ad-hoc commands

In this first task you'll be playing with inventories. You'll be trying out different techniques for defining inventories. You'll be exploring:

- Grouping
- Ranges
- Behavioural parameters
- Group parameters
- Groups of Groups
- group variables (in external files)
- Using yaml to define these

Initially you use the ini format, but if there's time, try them using yaml.

Navigate to the `/~/course/ansible/inventories` directories and let's get started.

Groups

Open the file `inv-groups` .

In this file define two groups `[…]` :

- `front` for `10.20.1.3`
- `rest` for `10.20.1.4` and `10.20.1.5`

Use the ansible CLI to test if you've defined it correctly `--list-hosts`

```
$ ansible rest -i inv-groups --list-hosts
hosts (2):
    10.20.1.4
    10.20.1.5
```

Also use the `ansible-inventory` to show a graph:

```
$ ansible-inventory -i inv-groups --graph
```

Range

Open `inv-range` and it it define `10.20.1.3` , `10.20.1.4` and `10.20.1.5` using a range `[…:…]`

Test if it works by listing the hosts of all

```
$  ansible all -i inv-range --list-hosts
    hosts (3):
      10.20.1.3
      10.20.1.5
      10.20.1.4
```

Behavioural parameters

For this one, use `inv-behavioural-params` .

- Define three host with an alias: `machine3` for `10.20.1.3` , `machine4` for `10.20.1.4` and `machine5` for `10.20.1.5`) `ansible_host`
- In addition set the default ssh user name to `vagrant` `ansible_user` for each.
- You don't have to define groups, but feel free to do so

To test run a simple `ip addr show` as-hoc command for just `machine3` and *without* specifying the ssh user name with the `-u` option:

```
$ ansible machine3 -i inv-behavioural-params  -a "/usr/sbin/ip -c addr show"
```

Group params

Open `inv_group-params` .

- As you've done before, define two groups
  - `front` for `10.20.1.3`
  - `rest` for `10.20.1.4` and `10.20.1.5`
- Set the ssh user for the hosts in the `rest` group `[rest:vars]`

Confirm you can access the hosts in the `rest` group without specifying the ssh user:

```
$ ansible rest -i inv-group-params  -a "/usr/sbin/ip -c addr show"
```

And notice it does *not* work for the hosts in the `front` group:

```
$ ansible front -i inv-group-params  -a "/usr/sbin/ip -c addr show"
```

We wanted you to see that the group variable really only applies to the hosts in the group. Let's make sure we can do the same for `front` group by specifying the ssh user. After that test that it works now as well for the hosts in the `front` group.

Let's add an additional group variable named `network_name` to both groups:

- for `front` set it to `eth1`
- for `rest` set it to `eth0`

Now use the variable as an argument to the `ip` command you've used above `{{…}}`

```
$ ansible all -i inv-group-params  -a "/usr/sbin/ip -c addr show {{network_name
}}"
```

Notice how is shows the ip information for different devices depending on the group to which the host belongs.

As a final step: At this moment you have duplicated the ssh user in both groups. How can you improve this and

only specify the ssh user once use the all groups `[all:vars]`

Groups of groups

Open `inv-groups-of-groups` .

- As you've done before, define two groups
  - `front` for `10.20.1.3`
  - `rest` for `10.20.1.4` and `10.20.1.5`
- Then make a group named `vagrant` that groups `front` and `rest` `[vagrant:children]`
- Set the ssh user as a group variable for the `vagrant` group

First check if your three hosts are part of the group:

```
$ ansible vagrant  -i inv-groups-of-groups --list-hosts
```

And check if the ssh user variable is working and while you are at, use a different module, for example ipinfoio_facts (https://docs.ansible.com/ansible/latest/modules/ipinfoio_facts_module.html#ipinfoio-facts-module) which shows ip geolocation facts for your hosts:

```
$ ansible vagrant  -i inv-groups-of-groups -m ipinfoio_facts
```

Variable files

As you know it is better to place group and host variables in separate files. You will be using the `inv_group` inventory you've created earlier and *without* changing it add additional variables:

- create the required directory structure `group_vars/` subdirectory
- set a variable named `conf_dir` to `/var` for the `front` group yaml inside `group_vars/front`
- set the same variable to `/etc` for the `rest` group
- the ssh user for both groups

Run a command to list the contents of the `conf_dir` on each hosts

```
$ ansible all -i inv-groups -a "ls -d {{conf_dir}}"
```

The output should resemble:

```
10.20.1.4 | SUCCESS | rc=0 >>
/etc

10.20.1.3 | SUCCESS | rc=0 >>
/var

10.20.1.5 | SUCCESS | rc=0 >>
/etc
```

Using yaml

If time allows, try all your configuration using yaml (but you might just want to continue with the next steps first and then when completed all come back to try and do these in yaml)

During this step you will save fact values on the remote machine itself. This is known as *local facts*

- SSH into machine-3:

```
$ ssh vagrant@10.20.1.3
```

- Then create the local facts directory `sudo mkdir -p /etc/ansible/facts.d`

Add a an ini file with a correct naming scheme `.fact` suffix. Then some content to this file, for example an email address:

```
[support]
email=support@cardiff-electric.com
```

Check the facts of this machine using the `ansible` CLI (try to filter and only show local facts `filter=ansible_local`)

```
$ ansible all -i "10.20.1.3," -m setup -a "filter=ansible_local" -u vagrant
```

If time allows, try making an executable that produces prints JSON to the STDOUT.

Open `~/course/ansible/config` in a terminal window.

Notice we have provided you with a simple inventory file.

It is your task to run the following command without specifying any additional parameters and without changing the `host` file. If you would try it now, it would fail, so you need to perform some steps to make this work. Have a look underneath the code sample.

```
$ ansible all -a whoami
machine3 | CHANGED | rc=0 >>
root

machine4 | CHANGED | rc=0 >>
root

machine5 | CHANGED | rc=0 >>
root
```

You may find some hints below. You also might want to use `ansible-config list` to search for the parameters and know under which ini-section you'll have to place the configuration parameters

- Create the configuration file:
  - in a correct directory for such a file (e.g, the current directory)
  - using the correct name for this file `ansible.cfg`
- Set the default location for the inventory to the `hosts` in this directory `inventory` (don't forget the correct header `[defaults]`)
- Set the default remote user name to `vagrant` `remote_user` (under the same header)
- Make sure the user's privilege is escalated to `root` `become` (don't forget the correct header `[privilege_escalation]`)

In addition you'll be setting `-C -o ControlMaster=auto -o ControlPersist=30m` ssh connection arguments (see explainshell.com (https://explainshell.com/explain?cmd=ssh+-C+-o+ControlMaster%3Dauto+-o+ControlPersist%3D30m) for an explanation of these arguments)

- Use the correct key to set ssh arguments `ssh_args`
- Use the correct header `ssh_connection`
- Enable compression of all data using the `-C` flag
- Use a `ControlMaster` value of `auto` to reuse existing connections `-o ControlMaster=auto`
- Set `ControlPersist` to 30 minutes (`30m`) in order to keep the master connection open between subsequent ssh calls `-o ControlPersist=30m`

(If you want more information on these settings, have a look at https://developer.rackspace.com/blog/speeding-up-

ssh-session-creation/ (https://developer.rackspace.com/blog/speeding-up-ssh-session-creation/))

To see the SSH command used by ansible, increase the verbosity level to "more"

```
$ ansible all –vvv –a whoami
```

Overriding configuration

Let's say we want to run an ad-hoc command, but not as root (as you have configured right now). How can we run an ad-hoc ansible command, overriding the the privilege escalation currently configured in the config file, *without* changing the configuration file? use the env: `ANSIBLE_BECOME=false ansible all –a whoami`

```
$ ANSIBLE_BECOME=false ansible all –a whoami
```