Lab 1

Setup

During this setup you will perform the following steps:

- Setup ansible (either on Fedora or Ubuntu)
- Run machines for the labs using vagrant
- Copy your ssh-key to them using ansible

Before you start download the lab code ansible.tar.gz (/downloads/ansible.tar.gz) and extract this inside a directory named `~/course/ansible` (you will probably have to create this directory first)

```
$ mkdir -p ~/course/ansible && \
   tar -xf `xdg-user-dir DOWNLOAD`/ansible.tar.gz -C  ~/course/ansible
```

Install the prerequisites (most likely already done as this is part of the course setup requirements)

```
$ sudo apt-get install -y openssh-server  python
```

Then to install ansible on ubuntu we need to first add `software-properties-common`

```
$ sudo apt-get update \
  && sudo apt-get install software-properties-common
```

Then we will add the ansible repository

```
$ sudo apt-add-repository ppa:ansible/ansible
```

Now install Ansible:

```
$ sudo apt-get update \
    && sudo apt-get install ansible
```

On Fedora just run the following command

```
$ sudo dnf install ansible
```

At this point there is no official bash completion for ansible. However there are a few unofficial ones available.

Let's install one of these (see https://github.com/dysosmus/ansible-completion (https://github.com/dysosmus/ansible-completion))

```
$ sudo wget -P  /etc/bash_completion.d https://raw.githubusercontent.com/dysosm
us/ansible-completion/master/ansible-completion.bash
$ sudo wget -P  /etc/bash_completion.d https://raw.githubusercontent.com/dysosm
us/ansible-completion/master/ansible-playbook-completion.bash
```

To test run the following command:

```
$ ansible --version
ansible 2.7.8
    config file = /etc/ansible/ansible.cfg
    configured module search path = [u'/home/student/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
    ansible python module location = /usr/lib/python2.7/dist-packages/ansible
    executable location = /usr/bin/ansible
    python version = 2.7.15rc1 (default, Apr 15 2018, 21:51:34) [GCC 7.3.0]
```

Some labs require Docker. Therefore you will need to install Docker CE as well. We have setup an ansible playbook to install Docker CE and Docker Compose on your local computer.

Navigate to the `~/course/ansible/setup/docker` directory. Notice the `docker.yaml` . In order to run this perform the following steps:

- Install the required roles

  ```
  $  ansible-galaxy install -r requirements.yaml
  ```

- Run the playbook (add `-e "user=…"` if you are running this course as a different user than *student*):

  ```
  $ ansible-playbook docker.yaml --ask-become-pass
  ```

You have been added to the docker linux user group, First try the following:

```
$ newgrp docker
$ id
uid=1000(student) gid=998(docker) groups=998(docker),4(adm),24(cdrom),
27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare),999(vboxsf),1000(stud
ent)
```

The second command should list `docker` as one of your groups. If not then you need to logout and log back in to your machine (sometimes even a restart of the machine is required)

After that check that docker is installed and that you have access to the daemon:

```
$ docker ps
CONTAINER ID    IMAGE   COMMAND    CREATED   STATUS   PORTS     NAMES
```

Check! You have completed this step. Go to the next

If this course is held at your own premises then there might be things missing from the required software installation.

Check the following;

- Is **Virtualbox** installed (should be 6 or higher)

  ```
  $ vboxmanage --version
  6.0.4r128413
  ```

- Is **Vagrant** installed (should be 2.2.4 or higher)

```
$ vagrant --version
  Vagrant 2.2.4
```

If not then you might need to run the setup script available in the `~/course/ansible/setup/course-requirements/` directory:

But check with your instructor first

```
$ ansible-playbook install-software.yaml --ask-become-pass
```

To setup your lab environment we are using Vagrant (https://www.vagrantup.com/). It allows you to run a couple of machines that you will use during this class.

Navigate to the `~/course/ansible/machines` directory in a terminal. Then run `vagrant up` to start up your machines (We estimate this to take 15 to 20 minutes)

```
$ vagrant up
  Bringing machine 'machine-2' up with 'virtualbox' provider...
  Bringing machine 'machine-3' up with 'virtualbox' provider...
  Bringing machine 'machine-4' up with 'virtualbox' provider...
  Bringing machine 'machine-5' up with 'virtualbox' provider...
  Bringing machine 'machine-6' up with 'virtualbox' provider...
  Bringing machine 'machine-7' up with 'virtualbox' provider...
  …
```

This will take some time. When vagrant is finished setting up your machines, check their status:

```
$ vagrant status
  Current machine states:

  machine-2                 running (virtualbox)
  machine-3                 running (virtualbox)
  machine-4                 running (virtualbox)
  machine-5                 running (virtualbox)
  machine-6                 running (virtualbox)
  machine-7                 running (virtualbox)
```

Check the IP

So you feel a little more at home later on, check the IP addresses of your machines. We have installed the vagrant-shell-commander (https://github.com/fgimenez/vagrant-shell-commander) plugin to run commands against each machine in our multi-machine environment (you can check this with `vagrant plugin list`)

Use it to get the IP address of each machine (the command we are executed on each machine is explained on explainshell (https://explainshell.com/explain?cmd=ip+-4+addr+show+eth1+%7C+grep+inet+%7C+awk+%27%7Bprint+%5C%242%7D%27+%7C+cut+-d+%2F+-f+1))

```
$ vagrant sh -c "ip -4 addr show eth1 | grep inet | awk '{print \$2}' | cut -d
/ -f 1"
machine-2::
10.20.1.2
machine-3::
10.20.1.3
…
```

Notice we are in the `10.20.1.0/24` as a subnet.

The easiest way to work with ansible is using SSH keys. We'll need to make sure that we provide your local machine's public key to the nodes. We could of course use `scp` or `ssh-copy-id` , but we have ansible!

You might need to generate a private key first. Check if you have a private key:

```
$ ls $HOME/.ssh/id_rsa
```

If you don't generate one:

```
$ ssh-keygen
```

What better way to test if your ansible install works, than by using it? Obviously you still have everything to learn about Ansible. However one of the promises is that even people not understanding ansible, should be able to **read** it. So let's put that to the test.

Open terminal session to `~/course/ansible/setup` . First open the file `hosts` . This defines the machines you want to talk to (this is called an **inventory**)

```
10.20.1.2
10.20.1.3
10.20.1.4
…
```

Notice these are the ip addresses for your the machines you created with vagrant. **Important**: if these differ from your ip addresses, then tell your instructor. Now open the `authorise.yaml` . This file contains the instructions to perform (and is called a **playbook**). You might be able to *read* and *understand* what it will do, no?

Time to run the playbook.

```
$ ansible-playbook  authorise.yaml
```

The vagrant password is also *vagrant*

You should see in the output that your machines have been *changed*

When you ran the playbook, the `ansible.cfg` was used. You might want to have a quick peek at that file.

Test ssh access

Try to access oe of your machines without a password:

```
$ ssh vagrant@10.20.1.2
[vagrant@machine-2 ~]$
```

You should not be prompted for a password as your key has been copied to the remote machines using an ansible playbook.

An important note. Sometimes during the class you might want to reset your machines (e.g, `vagrand destroy --force` and then again `vagrant up`). When doing so their certificates will change and therefore you'll have to :

- remove the public keys from the known_hosts

  ```
  $ for i in {2..7}; do ssh-keygen -R 10.20.1.$i; done
  ```

- Rerun the playbook (from the `~/course/ansible/setup` directory)

```
$ ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook  -i hosts \
    authorise.yaml --ask-pass -u vagrant
```

Now that your machines have been created and that you've copied your ssk-key, make a snapshot of them so you can always return to a "clean slate" installation of these machines (run this inside the `~/course/ansible/machines`)

```
$ vagrant snapshot save initial
```

This allows you to restore to this saved state *when needed*. For example later you could run the following command to restore *all* machines

```
$ vagrant snapshot restore initial
```

If you just want to restore a *single* machine you would run the following command (this one resets machine-3):

```
$ vagrant snapshot restore machine-3 initial
```

If ever you wanted to reset your machines completely (*don't* do this now!) perform the following steps *inside* this `~/course/ansible/machines` directory:

- Remove your virtual machines

```
$ vagrant destroy --force
```

- Create them again

```
$ vagrant up
```

Remember that in this case you also need to remove the public keys from your local machine (see earlier step)