Lab 5

Dynamic Inventories

This tasks involves python development, so whether or not you will create the script yourself depends on your experience with this. The directory for this task is `~/course/ansible/dynamic-inventories`, the solution can be found in its `solutions` sub-folder.

The dynamic inventory you will be creating will query all running virtualbox machines and eventually provide the same kind of host mapping as you've done for `/~/course/ansible/inventories/inv-behavioural-params` (so being able to use the machine names where the inventory maps to ip addresses)

If you are not familiar with developing python scripts, then you might want to use the solution instead and jump straight to "Using your dynamic inventory" below.

In order to create a dynamic inventory you will need write a python script which returns a json format resembling the one below:

```
{
  "running": {
    "hosts": [
      …
    ]
  },
  …
}
```

The name of our group will be `running` (we could also have created a group with virtual machines not running). The `_meta` element is used by ansible for facts and variables.

Open the `~course/dynamic-inventories/dynamic.py`. Notice we've already given you some boilerplate code (the imports and a function)

You will be using the VirtualBox SDK. The following two lines obtain the `VirtualBoxManager` and the `IVirtualBox` singleton

```
vbox_mgr = vboxapi.VirtualBoxManager(None, None)
vbox = vbox_mgr.getVirtualBox()
```

Then to get all the machines use the `vbox_mgr.getArray`, passing `vbox` instance and the type we want, which is "machines":

```
vbox_mgr.getArray(vbox, 'machines')
```

That's going to return all machines while we are only interested in the *running* ones. So we need to apply a filter. Each `machine` in the array has a `state` attribute, which we can compare to an enumeration value `vbox_mgr.constants.MachineState_Running`. Go ahead and filter the array and return a `list`

```
machines = list(filter(lambda m: m.state == vbox_mgr.constants.MachineState_Run
ning, vbox_mgr.getArray(vbox, 'machines')))
```

We only need the name and ip address of each machine. Let's therefore map the list of machines, to a list name/ip tuples. To get the IP we will need to get the value of a guest property. If would run the following in a terminal, you'll get the guest properties for a machine (in this case `machine-2`):

```
$ VBoxManage guestproperty enumerate machine-2
…
```

You should find `/VirtualBox/GuestInfo/Net/1/V4/IP` having the IP value we are looking for, you can check this on the command line as well:

```
$ VBoxManage guestproperty get machine-2  "/VirtualBox/GuestInfo/Net/1/V4/IP"
```

To obtain the `/VirtualBox/GuestInfo/Net/1/V4/IP` guest property using python, use `getGuestPropertyValue` on the `machine` . The name is obtained using the `name` property of each `machine` . Use this to map your list to a list of name/ip tuples.

```
NET_V_IP = "/VirtualBox/GuestInfo/Net/1/V4/IP"
data = list(map(lambda m: (m.name, m.getGuestPropertyValue(NET_V_IP)), machines
))
```

Next we can create the json. You might have noticed we have imported the json module (https://docs.python.org/2/library/json.html). We can use the dumps (https://docs.python.org/2/library/json.html#json.dumps) method which converts to json using this conversion table (https://docs.python.org/2/library/json.html#py-to-json-table).

So all you'll need to do is create a python dictionary with the structure expected by ansible for the `--list` argument. For the `hosts` you'll need to create a child element `running` with an array of our machine names:

```
if len(sys.argv) == 2 and sys.argv[1] == '--list':
    print json.dumps(
        {
            "running":
                {
                    "hosts": map(lambda m: m[0], data)
                }
        })
```

We need to also specify the `ansible_host` host variable for each host. What is the most efficient way of doing this? `_meta` :

```
map(lambda m: {m[0]: {"ansible_host": m[1]}}, data)
```

However we need to turn this list of dictionaries to a flattened dictionary. This is where the `list_2_dict` function comes into place. Use this to transform your list of dictionaries to a flat dictionary.

```
 "_meta":
    {
        "hostvars": list_2_dict(map(lambda m: {m[0]: {"ansible_host": m[1]}}, d
ata))
    }
```

And to finalise the script, reply with an emtpty json for the `--host …` argument:

```
elif len(sys.argv) == 3 and sys.argv[1] == '--host':
    print(json.dumps({}))
else:
    print("was expecting --list or --host <name>")
```

Let's first run our python script without ansible. Make sure you have some machines running (we are using jq (https://stedolan.github.io/jq/) here to format the output. It should be installed already as it is part of te setup requirements for this course.)

```
$ ./dynamic.py --list | jq
…
```

And try the `--host` argument

```
$ ./dynamic.py --host machine-2 | jq
…
```

The use it then together with ansible. Let's first list all the hosts:

```
$ ansible all -i dynamic.py   --list-hosts
```

And then try to ping each:

```
$ ansible -i dynamic.py all  -m ping -u vagrant
```

That concludes this exercise. You will learn another way to create an inventory for all running VirtualBox virtual machines using a bundled plugin instead.