Introduction to Ansible.

1. **Overview**

1.1 Ansible is an orchestration and configuration tool, created by RedHat™. The most important fuctions and features are Open Source.

Ansible services needs in deploying, configuring and managing your compute environment. You can easily manage groups of systems from one control-server based on e.g networks, service types, application types, etc.

It uses YAML[1] in the form of playbooks to manage your environment. Playbooks are equivalent to scripts that execute one or more tasks when invoked. However, you can also run single commands if you like.

1.2 Modules

Ansible uses tiny programs called modules that are pushed to clients. These modules will execute tasks on the destination clients. There are thousands of modules.

These modules support many operating environments. Think of Linux, Juniper, vmware, windows, NetApp, etc.

1.3 Keys

Ansible does support password authentication, but it is preferable to use public-key authentication. So what you in fact need, is automatic SSH-access to your clients. In that way, you can either run single (Ad-Hoc) commands or playbooks to automate management without passwords.

2. Ad-Hoc commands.

Ad-Hoc commands create a one time ansible session. Usually you will use Ad-Hoc commands to run single executions of a module to a number of hosts only once. So, you do not feel the need to save the command because you only want to run it once.

This is usually done when you want to test things or make quick changes that do not need repetition.

Some simple examples:

To ping all clients you can use the ping module.

```
ansible -m ping all
192.168.4.131 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.4.132 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

To run a command on a single client you can pass the command as an argument.

```
$ ansible 192.168.4.131 -a "uname -n"
192.168.4.131 | SUCCESS | rc=0 >>
linux1
```

To install the ksh rpm on all CentOS clients, you can use the *yum* module with the arguments *name* and *state*.

```
ansible all -m yum -a "name=ksh state=present"
```

3. Playbooks

To automate tasks by grouping them together, you can run a series of tasks on any selection of clients at any time.

A playbook is an ASCII file in YAML syntax. The playbook can be executed using the *ansible-playbook* command.

A very simple example:

```
---
- hosts: linux
  tasks:
     - name: install apache
       yum:
        name: httpd
        state: present
```

In fact. The above playbook will have the same result as the following Ad-Hoc line:

```
$ ansible -m yum -a "pkg=httpd state=present" linux
```

In both cases, all hosts that are addressed by the collection “linux”, will have one task: install httpd.

4. Setup

We will take the following steps.

1. install ansible
2. setup connections to clients
3. configure ansible hosts and config

4.1 To install ansible you run the following command

$ yum install ansible -y

This will setup /etc/ansible

4.2 Setup connection to clients.

The control system needs a key-pair:

$ ssh-keygen -t rsa

Setup automatic login to clients.

(we have two clients: 192.168.4.133 and 192.168.4.134)

$ ssh-copy-id root@192.168.4.133

4.4 Configure basic ansible usage.

In /etc/ansible you will find two files and a directory.
The file we will change is hosts. This file contains the
clients that we will manage.

We open the hosts file. It contains only comments so we remove that and only enter the following:

```
[linux]
192.168.4.133
192.168.4.134
```

There are multiple ways of configuring this hosts file. The above example is only one of them.
You can address both clients as one by using 'linux' or you can address them separately by ip address.

Now as an example, you can add a package to one or more hosts in your environment with an ad-hoc command.

```
ansible -m yum -a "pkg=httpd state=present" linux
ansible -m yum -a "pkg=httpd state=absent" 192.168.4.133
```

In the above examples you use the module yum (-m),
and the argument (-a) is that you first add httpd to both nodes and then remove it from the first node.

Some examples of ad-hoc commands:

```
ansible linux -m copy -a "src=/etc/hosts dest=/tmp/hosts"
```
(copy /etc/hosts to /tmp/hosts on all linux machines)

```
ansible linux -m file -a "dest=/tmp/hosts mode=600 owner=linuser group=linuser"
```
(change the permissions, uid and gid of the file hosts in tmp on all hosts)

List the hostname of every linux machine:

```
ansible -m shell -a "hostname" all
```

5. Playbooks examples.

Playbooks are stored in the following directory:

roles/<rolename>/tasks/

So for example:

/etc/ansible/roles/*basic*/tasks/

A playbook can contain a number of tasks.

In the following example we create a playbook that will install 2 packages and it sets selinux to permissive mode
and stops and disable firewalld.

cat /etc/ansible/roles/basic/tasks/main.yml
If you have only got one taskfile in here, it should be called **main.yml**.

```
- name: "Installing httpd"
  yum:
    name: httpd
    state: latest

- name: "Installing epel-release"
  yum:
    name: epel-release
    state: latest

- selinux:
    policy: targeted
    state: permissive

- name: disable firewalld
  service: name=firewalld state=stopped
  enabled=no
```

Then, in your ansible directory you create the playbook that will execute this task. This file contains the hosts on which to execute and the task you want to execute.
Cat /ansible/pb.yml

```
- hosts: linux
  roles:
  - basic
```

To execute the role, run the following command:

ansible-playbook pb.yml

Adding a user via a playbook entry:
(to generate an encrypted password you can use
mkpasswd <passwd>)

```
- name: Create a login user
  user:
      name: peter
      password: 'H/MUXAJHPNvgI'
      groups: # Empty by default
      state: present
      shell: /bin/bash       # Defaults to
                            /bin/bash
      system: no             # Defaults to no
      createhome: yes        # Defaults to yes
      home: /home/peter  # Defaults to /home/<username>
```

If you want to add more task files in the tasks directory,
you can add an include line in main.yml

```
- include_tasks: sub.yml tags=sub
```

So you might consider to create multiple yml files in the tasks directory. And then you would only add *include* entries to the main.yml file, and comment out lines for tasks you do not want to perform.

```
- include_tasks: sub.yml tags=sub
#- include-tasks: rpm.yml tags=rpms
- include_tasks: users.yml tags=users
- include_tasks: files.yml tags=filecopy
```

Ansible exercises.

Team up with some other students. This will give you both multiple vms to work with.

1. To see all available modules and how to configure them, check the ansible documtation.

https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

2. Install ansible on your vm.

```
[root@linux1 ~]# yum install ansible
```

3. Go to the ansible configuration directory

```
[root@linux1 ~]# cd /etc/ansible
```

4. Check the directory content.

```
[root@linux1 ansible]# ls
ansible.cfg  hosts  roles
```

5. You first edit the hosts file and add some ip addresses to it.

```
[linux]
192.168.4.132
192.168.4.133
```

6. If necessary generate a key-pair and copy this to the linux machines.

```
[root@linux1 /]# ssh-keygen -t rsa
[root@linux1 /]# ssh-copy-id root@192.168.4.132
[root@linux1 /]# ssh-copy-id root@192.168.4.133
```

7. Test using the ping module

```
[root@linux1 ansible]# ansible -m ping all
192.168.4.132 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.4.133 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

8. Ad-Hoc commands.

Install apache on all linux hosts.

```
[root@linux1 /]# cd /etc/ansible/
[root@linux1 ansible]# ansible -m yum -a "pkg=httpd state=present" linux
```

Remove apache from all linux hosts.

```
[root@linux1 ansible]# ansible -m yum -a "pkg=httpd state=absent" linux
```

Create some random files and copy them to the /tmp directory of all linux hosts.

```
[root@linux1 ansible]# mkdir randomfiles
[root@linux1 ansible]# echo random1 > randomfiles/rf1
[root@linux1 ansible]# echo random2 > randomfiles/rf2

[root@linux1 ansible]# ansible -m copy -a "src=/etc/ansible/randomfiles
dest=/tmp" linux
```

Reboot all linux machines.

```
[root@linux1 ansible]# ansible -a "/sbin/reboot" linux
```

Stop the cron daemon on all linux machines.

```
[root@linux1 ansible]# ansible -m service -a "name=crond \
                       state=stopped" linux
[root@linux1 ansible]# ansible linux -a "systemctl status crond"
```

Create a user on the linux vms.

First generate an md5 password.

```
[root@linux1 ansible]# openssl passwd -1 mypass
$1$ZG.Ihns4$s9zXi0e2oKbXask6HWCsR0
```

Then create the user with the encrypted password.

```
[root@linux1 ansible]# ansible all -m user -a 'name=user1
password=$1$ZG.Ihns4$s9zXi0e2oKbXask6HWCsR0'
[root@linux1 ansible]# ssh user1@l2
user1@l2's password: mypass
```

Remove the user.

```
[root@linux1 ansible]# ansible all -m user -a 'name=user1 state=absent'
```

9. Playbooks.

First create a valid directory structure. You will need a *role* and a *tasks* directory. My role is *basic*, so I create the following directory:

```
[root@linux1 ansible]# mkdir -p roles/basic/tasks
```

Then you create your main playbook file in /etc/ansible.

```
[root@linux1 ansible]# cat pb.yml
---
- hosts: linux
  roles:
  - basic
```

In tasks directory create a the following yml files:

```
[root@linux1 ansible]# cat roles/basic/tasks/rpms.yml
- name: "Installing httpd"
  yum:
    name: httpd
    state: latest

- name: "Installing epel-release"
  yum:
    name: epel-release
    state: latest

[root@linux1 ansible]# cat roles/basic/tasks/files.yml
- name: copy files
  copy: src=randomfiles dest=/tmp/ owner=root group=root mode=0664
```

Finally create the main.yml file in the tasks directory.

```
[root@linux1 ansible]# cat roles/basic/tasks/main.yml
- include_tasks: files.yml tags=files
- include_tasks: rpms.yml tags=rpms
```

Test your playbook.

```
[root@linux1 ansible]# ansible-playbook pb.yml
```

End of exercise.