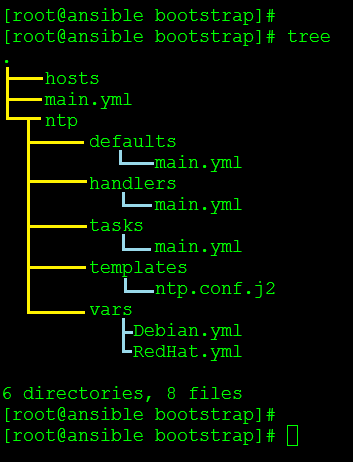
Ansible Notes:

1. Basic Playbook structure: Playbooks contain plays, plays contain tasks, tasks call modules. Tasks run sequentially. Handlers are run once, at the end of a play. Roles are basically self contained playbooks that can be called from a larger playbook.
   1. Module Format:

module: directive1=value directive2=value

examples: yum: name=httpd state=latest

template: src=/opt/files/httpd2.conf dest=/etc/httpd.conf



1. Main config file is /etc/ansible/ansible.cfg, but can be anywhere and referenced. If you place it in the same directory as the playbook, it will use that.
2. **ansible-doc -l** (l as in lima) to list all available modules
3. All servers kept in an **inventory** file. By default that file is /etc/ansible/hosts, but can be anywhere named anything. It's called by the **-i tag**.

In the inventory file, you can list individual servers or groups of servers. To list groups of servers use the [square] brackets. Sample inventory file below:

##By IP  
192.168.1.5  
192.168.1.6  
192.168.1.7

#By hostname  
web1.crotwell.net  
db1.crotwell.net

#By group   
[webservers]  
192.168.1.5  
web1.crotwell.net  
web2.crotwell.net

[databaseservers]  
db1.crotwell.net  
db2.crotwell.net ansible\_ssh\_pass=password ansible\_ssh\_user=root

**\*NOTE:** You may have to append the user and pass variables as shown above if you don’t have password-less ssh configured. To configure passwordless ssh run **ssh-keygen -t rsa** on the control server, then copy the **id\_rsa.pub** to the client server as the file **/root/.ssh/authorized\_keys**

1. The basic command to see if ansible is configured properly is:
   1. **ansible webservers –i /etc/ansible/hosts –m ping**
   2. Note that the –m tells the command to use a pre-defined module, in this case the ping module.
2. This is the command to use the yum module which will install the package you define on all servers you define:
   1. **ansible webservers –i /etc/ansible/hosts –m yum –a “name=mariadb state=present”**
   2. Note that mariadb can be replaced with any package name
   3. to add as a playbook entry:

- name: install the latest version of Apache

yum: name=httpd state=latest

1. This is the playbook command to add a yum repository:

**- name: Add repository**

**yum\_repository:**

**name: epel**

**description: EPEL YUM repo**

**baseurl:** [**http://download.fedoraproject.org/pub/epel/$releasever/$basearch/**](http://download.fedoraproject.org/pub/epel/$releasever/$basearch/)

1. Variables: You define your variables in the yaml file directly, or in main.yml in the vars directory under the role (/ansible/playbook/PBIS/roles/pbis/vars/main.yml)

A. How to use shell command output as variable.

**- shell: ifconfig | grep mtu | grep [ep][n0-9]p[0-9] | sed 's/:.\*$//g'**

**register: ifconfig\_output**

B. Encrypting Variables Using Vault:

1. If you have password or keys as variables, you don't want this in plaintext, so you need to use ansible-vault to create the file

**#ansible-vault create main.yml** (it will ask you for a password to use for the hash)

**#ansible-vault edit main.yml (to edit an already encrypted file)**

\*When running the playbook, you have to specify the option to use vault password:

**#ansible-playbook something.yml --ask-vault-pass**

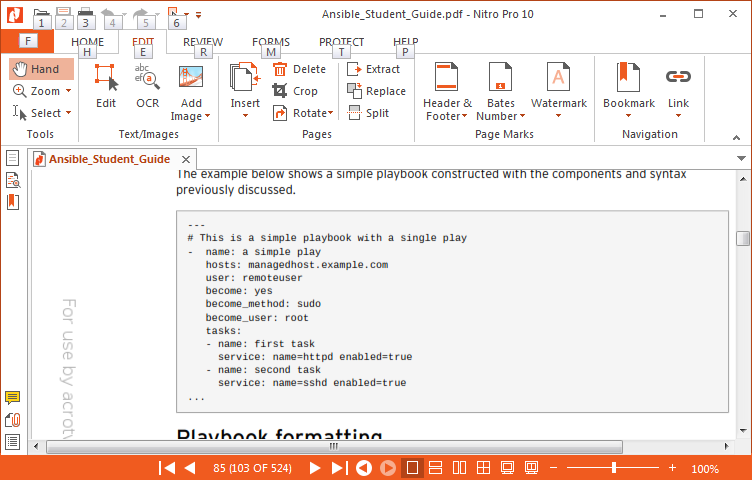
1. Meta Directory:

This keeps meta data, you have to have a main.yml, which can have no dependencies.

**---**

**-dependencies: []**

1. Inventory: In here you have a file that has your list of servers to apply
2. Two (2) spaces indent is format for yaml file. Sample below:



1. Use --step to walk through each task manually (ansible-playbook --step playbook.yml)
2. Each play starts with -name: (at beginning of file, no indent)
3. Edit ~/.vimrc file to set it for auto 2 space indent. So when you tab and return it goes to the correct location automatically.

----.vimrc contents---------

set expandtab

set tabstop=2

set ai

----------------------------

1. To use a list (ansible loop), you use the {{ item }} variable. Example to install packages and start them:

- yum:

name: "{{ item }}"

state: latest

with\_items:

- httpd

- mariadb

or

- service:

name: "{{ item }}"

state: started

with\_items:

- httpd

- mariadb

16. Handlers are dormant tasks that are called by the notify line

## Shell vs. Command Module

* They basically do the same thing, just one through a shell (ie. using script)
* With the Command module the command will be executed without being proceeded through a shell. As a consequence some variables like $HOME are not available. And also stream operations like  <, >, | and & will not work.
* The Shell module runs a command through a shell, by default /bin/sh. This can be changed with the option executable. Piping and redirection are here therefor available.
* The command module is more secure, because it will not be affected by the user’s environment.