Prereqs
Install vagrant & virtualbox
sudo apt-get install vagrant
sudo apt-get install virtualbox

Create directories on you local machines
mkdir learn_proj
mkdir learn_proj/Vms
mkdir learn_proj/control_repo

Spin vagrant enviornment
cd learn_proj/Vms

Change IP address of the vm's from vagrant file
vi Vagrantfile
vagrant up

Assumptions
Control machine:- Linux
ansible doc reference for ansible server(control server)
All the commands will be executing from the control machine.
Control machine ssh access, can use ssh key exchange.
Super user access on control machine.

Ansible installation
Refer ansible doc
http://docs.ansible.com/ansible/intro_installation.html#latest-releases-via-apt-ubuntu

Verify installation
#ansible --help
ansible-playbook --help
ansible-galaxy --help

Inventory PT 1
Static inventory
Can specify extra detail like non-standard ssh port
Inventory can use groups of host
can use variable information

ansible --list-hosts all
sudo vi /etc/ansible/hosts
delete all the hosts

ansible --list-hosts all

Inventory PT 2
Creating our own inventory file
mkdir repo/ansible && cd repo/ansible
vi dev (inventory)
ansible -i dev --list-hosts all

Set inventory path in ansible.cfg
Benefit of tracking it in version control system
vi ansible.cfg

ansible --list-hosts all

Host selection

Can use pattern syntax that allow you to select subset from the inventory.

ansible --list-hosts all
ansible --list-hosts "*"
ansible --list-hosts loadbalancer
ansible --list-hosts webserver
ansible --list-hosts db01
ansible --list-hosts database:control
#ansible --list-hosts webserver[0]
ansible --list-hosts \!control

https://docs.ansible.com/ansible/intro_patterns.html

Tasks
https://docs.ansible.com/ansible/intro_getting_started.html#your-first-commands
https://docs.ansible.com/ansible/ping_module.html
https://docs.ansible.com/ansible/command_module.html

ansible -m ping all
ansible -m command -a "hostname" all
Default module is command
ansible -a "hostname" all

All tasks are gonna have return status

Return exit code non zero
ansible -a "/bin/false" all

http://docs.ansible.com/ansible/modules_by_category.html

Plays

mkdir -p ansible/playbooks
#vi hostname.yml
```
 ---
 - hosts: all
   tasks:
- command: hostname
```

Playbook Execution
ansible-playbook hostname.yml

#vi hostname.yml
```
---
- hosts: all
  tasks:
    - name : get server info
      command: hostname
```

ansible-playbook hostname.yml

Quiz

Four pillar of linux application | Principles to setup app on linux
1. Pakcages: From repositories(apt or yum) or any other resources
2. Services: init.d or system.d or your own start script
3. System configurations: Files, directories, users, permission, firewall rules etc
4. Config files for the app itself

Playbook intro
http://docs.ansible.com/ansible/modules_by_category.html

Pakcages
vi loadbalancer.yml

```
---
- hosts: loadbalancer
become: true
  tasks:
    - name: Install nginx
      apt: name=nginx state=present update_cache=yes
```

vi database.yml

```
---
- hosts: database
  become: true
  tasks:
    - name: Install mysql-server
      apt: name=mysql-server state=present update_cache=yes
```

Execute loadbalancer playbook
ansible-playbook loadbalancer.yml
Execute it again
ansible-playbook loadbalancer.yml

Execute Database playbook
ansible-playbook database.yml

Write webserver playbook, which covers loops and jinja

https://docs.ansible.com/ansible/playbooks_loops.html#standard-loops
http://jinja.pocoo.org/

vi webserver.yml

```
---
- hosts: webserver
  become: true
  tasks:
    - name: Install apache2
      apt: name={{item}} state=present update_cache=yes
      with_items:
        - apache2
        - libapache2-mod-wsgi
        - python-pip
        - python-virtualenv
```

Services modules

vi loadbalancer.yml

```
+   - name: Ensure nginx started
+     service: name=nginx state=started enabled=yes
```

Test the nginx service

wget -qO- http://lb01 | less

Easier to do with curl, so install curl on the control server with playbook

vi control.yml

```
---
- hosts: control
  become: true
  tasks:
    - name: install tools
      apt: name={{item}} state=present update_cache=yes
      with_items:
        - curl
```

ansible-playbook control.yml
curl lb01

Add service module to webserver and database playbooks

vi webserver.yml

```
+   - name: Ensure Apache started
+     service: name=apache2 state=started enabled=yes
```

vi database.yml

```
>>

   - name: Ensure Mysql started
     service: name=mysql state=started enabled=yes

# ansible-playbook webserver.yml

# ansible-playbook database.yml
```

Operational playbook | Support playbook | Stack Restart
We will need a playbook to manage services of the entire stack or manage cluster services.

```
# vi stackrestart.yml

---
# Bring stack down
- hosts: loadbalancer
  become: true
  tasks:
    - service: name=nginx state=stopped

- hosts: webserver
  become: true
  tasks:
    - service: name=apache2 state=stopped

# Restart mysql
- hosts: database
  become: true
  tasks:
    - service: name=mysql state=restarted

# Bring stack up
- hosts: webserver
  become: true
  tasks:
    - service: name=apache2 state=started

- hosts: loadbalancer
  become: true
  tasks:
    - service: name=nginx state=started
```

apache2_module, handlers, notify
We need to setup the apache reciving python application. Python app gonna use mod-wsgi to serve the request. Make sure mod-wsgi is enabled with apache_module.
https://docs.ansible.com/ansible/apache2_module_module.html
https://docs.ansible.com/ansible/playbooks_intro.html#handlers-running-operations-on-change

```
# ansible-playbook webserver.yml

+    - name: Ensure mod_wsgi enables
```

```
+     apache2_module: state=present name=wsgi
+     notify: Restart apache2
+  handlers:
+    - name: Restart apache2
+      service: name=apache2 state=restarted

# ansible-playbook webserver.yml
```

Files: Copy

Copy the demo directory in the ansible directory which contains the python app written in flask.

```
# vi webserver.yml
>>
    - name: copy demo app source
      copy: src=demo/app/ dest=/var/www/demo mode=0755
      notify: Restart apache

    - name: copy apache virtual host config
      copy: src=demo/demo.conf dest=/etc/apache2/sites-available mode=0755
      notify: Restart apache

# ansible-playbook webserver.yml

# curl app01
```

Still shows the default apache site for python app working we need to configure PIP and Virtualenv

Application Modules

```
# vi webserver.yml

>>
    - name: setup python virtualenv
      pip: requirements=/var/www/demo/requirements.txt virtualenv=/var/www/demo/.venv
      notify: Restart apache2

# ansible-playbook webserver.yml
```

Files:File | Activating python site and deactivating apache default site

```
# vi webserver.yml
>>
    - name: de-activate default apache site
      file: path=/etc/apache2/sites-enabled/000-default.conf state=absent
      notify: Restart apache2

    - name: activate demo apache site
      file: src=/etc/apache2/sites-available/demo.conf dest=/etc/apache2/sites-enabled/demo.conf
state=link
      notify: Restart apache2
```

ansible-playbook webserver.yml
curl app01
curl app02
curl lb01, will show the default page from nginx, we need to configure it to point to the App servers

Templates | Configure lb01 to point to app01 & app02

curl lb01
mkdir templates
vi templates/nginx.conf.j2

```
upstream demo {
{% for server in groups.webserver %}
   server {{ server }};
{% endfor %}
}

server {
   listen 80;

   location / {
      proxy_pass http://demo;
   }
}
```

vi loadbalancer.yml

```
>>
   - name: configure nginx site
     template: src=templates/nginx.conf.j2 dest=/etc/nginx/sites-available/demo mode=0644
     notify: restart nginx

   - name: de-activate default nginx site
     file: path=/etc/nginx/sites-enabled/default state=absent
     notify: restart nginx

   - name: activate demo nginx site
     file: src=/etc/nginx/sites-available/demo dest=/etc/nginx/sites-enabled/demo state=link
     notify: restart nginx

  handlers:
   - name: restart nginx
     service: name=nginx state=restarted
```

ansible-playbook loadbalancer.yml

Test if its working.
curl lb01

Lineinfile | Make db server to listen to on all interface 0.0.0.0.
https://docs.ansible.com/ansible/lineinfile_module.html

Test the connection first
curl app01/db

- Check demo.wsgi which has db connection info.
- Check demo.py which establishes connection from app to db server
- Login to db server and see myql listening only on local interface
netstat -an
- Open /etc/mysql/my.cnf
With ansible lineinfile module replace "bind-address 127.0.0.1" to bind-address to "bind-address 0.0.0.0"

Login to control server
vi database.yml

>>

```
  - name: ensure mysql listening on all ports
    lineinfile: dest=/etc/mysql/my.cnf regexp=^bind-address
                line="bind-address = 0.0.0.0"

    notify: restart mysql

 handlers:
  - name: restart mysql
    service: name=mysql state=restarted
```

ansible-playbook database.yml

Verify
curl app01

Appication Modules: mydql_db, mysql_user | Install python-mysqldb in app and db server, create demo db and demo user.

Install mysql module in apache server
vi webserver.yml

```
  - name: Install apache2
    apt: name={{item}} state=present update_cache=yes
    with_items:
      - apache2
      - libapache2-mod-wsgi
      - python-pip
      - python-virtualenv
+     - python-mysqldb
```

ansible-playbook webserver.yml

https://docs.ansible.com/ansible/mysql_db_module.html
https://docs.ansible.com/ansible/mysql_user_module.html

```
# vi database.yml

- hosts: database
  become: true
  tasks:
+   - name: install tools
+     apt: name={{item}} state=present update_cache=yes
+     with_items:
+       - python-mysqldb
+
    - name: install mysql-server
      apt: name=mysql-server state=present update_cache=yes


      lineinfile: dest=/etc/mysql/my.cnf regexp=^bind-address line="bind-address = 0.0.0.0"
      notify: restart mysql

+   - name: create demo database
+     mysql_db: name=demo state=present
+
+   - name: create demo user
+     mysql_user: name=demo password=demo priv=demo.*:ALL host='%' state=present
+
  handlers:
    - name: restart mysql
      service: name=mysql state=restarted
```

Execute db playbook:

```
# ansible-playbook database.yml
```

Test the connection

```
# curl app01/db
# curl app02/db
# curl lb01/db
# curl lb01/db
# curl lb01/db
# curl lb01/db
```

Support Playbook 2 – Stack Status: wait_for | Playbook to check the current status of all the services in stack. Also modifying stackrestart playbook with wait_for module.

Create stack_status.yml

```
# vi stack_status.yml
+---
+- hosts: loadbalancer
+ become: true
+ tasks:
+   - name: verify nginx service
+     command: service nginx status
```

```
+
+   - name: verify nginx is listening on 80
+     wait_for: port=80 timeout=1
+
+- hosts: webserver
+  become: true
+  tasks:
+   - name: verify apache2 service
+     command: service apache2 status
+
+   - name: verify apache2 is listening on 80
+     wait_for: port=80 timeout=1
+
+- hosts: database
+  become: true
+  tasks:
+   - name: verify mysql service
+     command: service mysql status
+
+   - name: verify mysql is listening on 3306
+     wait_for: port=3306 timeout=1
```

ansible-playbook stack_status.yml

Modify stackrestart.yml

vi stackrestart.yml

```
 become: true
  tasks:
   - service: name=nginx state=stopped
+   - wait_for: port=80 state=drained

- hosts: webserver
  become: true
  tasks:
   - service: name=apache2 state=stopped
+   - wait_for: port=80 state=stopped

 # Restart mysql
- hosts: database
  become: true
  tasks:
   - service: name=mysql state=restarted
+   - wait_for: port=3306 state=started

 # Bring stack up
- hosts: webserver
  become: true
  tasks:
   - service: name=apache2 state=started
+   - wait_for: port=80
```

```
- hosts: loadbalancer
  become: true
  tasks:
    - service: name=nginx state=started
+    - wait_for: port=80
```

Support Playbook 2 -- Stack Status: uri,register,fall,when |

Install python-httplib2 package on loadbalancer(nginx) which is requirment to do the test.
We will install python-httplib2 package on loadbalancer because we are going to do test from the loadbalancer to the app servers.

https://docs.ansible.com/ansible/uri_module.html

vi loadbalancer.yml

```
- hosts: loadbalancer
  become: true
  tasks:
+    - name: install tools
+      apt: name={{item}} state=present update_cache=yes
+      with_items:
+        - python-httplib2
+
     - name: install nginx
       apt: name=nginx state=present update_cache=yes
```

We will be also doing test from control server so we will install python-httplib2 on control server also.

vi control.yml

```
       apt: name={{item}} state=present update_cache=yes
       with_items:
         - curl
+        - python-httplib2
```

Edit content of stack_status.yml to test the connection from control and loadbalancer(backend). Connection will be tested by checking the content returned by hitting http://lb01, http://app01, http://app02/, http://lb01/db, http://app01db, http://app02/db

vi stack_status.yml

```
    - name: verify mysql is listening on 3306
      wait_for: port=3306 timeout=1
+
```

```
+- hosts: control
+  tasks:
+    - name: verify end-to-end index response
+      uri: url=http://{{item}} return_content=yes
+      with_items: groups.loadbalancer
+      register: lb_index
+
+    - fail: msg="index failed to return content"
+      when: "'Hello, from sunny' not in item.content"
+      with_items: "{{lb_index.results}}"
+
+    - name: verify end-to-end db response
+      uri: url=http://{{item}}/db return_content=yes
+      with_items: groups.loadbalancer
+      register: lb_db
+
+    - fail: msg="db failed to return content"
+      when: "'Database Connected from' not in item.content"
+      with_items: "{{lb_db.results}}"
+
+- hosts: loadbalancer
+  tasks:
+    - name: verify backend index response
+      uri: url=http://{{item}} return_content=yes
+      with_items: groups.webserver
+      register: app_index
+
+    - fail: msg="index failed to return content"
+      when: "'Hello, from sunny {{item.item}}!' not in item.content"
+      with_items: "{{app_index.results}}"
+
+    - name: verify backend db response
+      uri: url=http://{{item}}/db return_content=yes
+      with_items: groups.webserver
+      register: app_db
+
+    - fail: msg="db failed to return content"
+      when: "'Database Connected from {{item.item}}!' not in item.content"
+      with_items: "{{app_db.results}}"
```

ansible-plabook stack_status.yml

Roles:

We have three tier in our app
How much of code in reusable?
Benefits
Code reuse
Encapsulation: we dont keep everything in one place, will help with scaling our app configs

mkdir repo/ansible-roles
cp -r repo/ansible/* repo/ansible-roles/

cd repo/ansible-roles

Create skeleton of roles with ansible-galaxy

```
# ansible-galaxy init control
# ansible-galaxy init mysql
# ansible-galaxy init nginx
# ansible-galaxy init apache2
# ansible-galaxy init demo_app
```

Need to move content of all the playbooks to their respective roles

1. control.yml:
   Move lists of tasks to control role's tasks/main.yml
   Replace tasks section from the playbook with roles content.
   Execute control.yml to test.
   # ansible-playbook control.yml

```
---
- hosts: control
  become: true
  roles:
    - control
```

2. database.yml:
   Move lists of tasks to mysql role's tasks/main.yml.
   Move mysql start task after my.cnf file change section.
   Move lists of handlers to mysql role's handlers/main.yml.
   Replace tasks & handlers section from the playbook with roles content.

```
---
- hosts: database
  become: true
  roles:
    - mysql
```

3. loadbalancer.yml
   Move lists of tasks to nginx role's tasks/main.yml.
   Move lists of handlers to nginx role's handlers/main.yml.
   Move nginx.conf.j2 template to nginx role's templates directory
   Open tasks/main.yml of the nginx role, change templates src path
   from src= templates/nginx.conf.j2 to src=nginx.conf.j2
   Replace tasks & handlers section from the playbook with roles content.

```
---
- hosts: loadbalancer
  become: true
  roles:
- nginx
```

4. webserver.yml
Move lists of apache tasks to apache role's tasks/main.yml.
Move apache2 start task to the end of the.
Move lists of apache handlers to apache role's handlers/main.yml.
Move lists of app tasks to demo_app role's tasks/main.yml.
Move lists of app handlers to demo_app role's handlers/main.yml.
Copy demo app directory to demo_app role's files/

Replace tasks & handlers section from the playbook with roles content.

```
---
- hosts: webserver
  become: true
  roles:
   - apache2
   - demo_app
```

Site.yml: include | include all the playbook into site.yml
Create site.yml at the same place where all the playbooks are located
vi site.yml

```
---
- include: control.yml
- include: database.yml
- include: webserver.yml
- include: loadbalancer.yml
```

ansible-playboook site.yml

Variables: facts | Will use fact variable to make mysql listen only on its own ip address and not on 0.0.0.0

Get the list of all the fact variable with setup module
ansible -m setup database

Edit database role's tasks
vi roles/mysql/tasks/main.yml

```
 - name: ensure mysql listening on all ports
-  lineinfile: dest=/etc/mysql/my.cnf regexp=^bind-address line="bind-address = 0.0.0.0"
+  lineinfile: dest=/etc/mysql/my.cnf regexp=^bind-address
+              line="bind-address = {{ ansible_eth0.ipv4.address }}"
   notify: restart mysql

 - name: ensure mysql started
```

Execute stack_status.yml
ansible-playbook stack_status.yml

Update stack_status.yml
vi playbooks/stack_status.yml

```
    - name: verify mysql is listening on 3306
-     wait_for: port=3306 timeout=1
+     wait_for: host={{ ansible_eth1.ipv4.address }} port=3306 timeout=1
```

Update stack_restart.yml
vi playbooks/stack_restart.yml

```
  become: true
  tasks:
   - service: name=mysql state=restarted
-   - wait_for: port=3306 state=started
+   - wait_for: host={{ ansible_eth1.ipv4.address }} port=3306 state=started
```

Variables:defaults | Using variable in mysql_db & mysql_user module
Lowest priorities is for default vars, refer the doc.
https://docs.ansible.com/ansible/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

roles/mysql/defaults/main.yml

```
 ---
-# defaults file for mysql
+db_name: myapp
+db_user_name: dbuser
+db_user_pass: dbpass
+db_user_host: localhost
```

roles/mysql/tasks/main.yml

```
 - name: ensure mysql started
   service: name=mysql state=started enabled=yes

-- name: create demo database
-  mysql_db: name=demo state=present
+- name: create database
+  mysql_db: name={{ db_name }} state=present

-- name: create demo user
-  mysql_user: name=demo password=demo priv=demo.*:ALL host='%' state=present
+- name: create user
+  mysql_user: name={{ db_user_name }} password={{ db_user_pass }}
priv={{ db_name }}.*:ALL
+             host='{{ db_user_host }}' state=present
```

Variables: Vars | Multiple ways/places to define vars, refer the doc.
https://docs.ansible.com/ansible/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Already defines vars in mysql role's defaults/main.yml which has the last precedence and would be used if vars are nowhere defined.
We will define vars in database.yml playbook with a hash.

```
# vi database.yml
- hosts: database
  become: true
  roles:
-    - mysql
+    - { role: mysql, db_name: demo, db_user_name: demo, db_user_pass: demo, db_user_host:
'%' }
```

Variables: with_dict | Use of with_dict module in nginx.yml playbook.

Will provide lists of variables in nginx role's defaults/main.yml. As specified below for reference. Name of the list is "sites".

```
---
# defaults file for nginx
sites:
   myapp:
      frontend: 80
      backend: 80
```

Will refer to those variables with the module with_dict in nginx.yml playbook. As specified below for reference.

```
with_dict: sites
```

with_dict will supply variable name in hash/dictionary format, variable names are {{ item.key }} & {{ item.value.frontend }} & {{ item.value.backend }}

```
(item={'value': {u'frontend': 80, u'backend': 80}, 'key': u'myapp'})
```

Update nginx role's defaults/main.yml

```
+sites:
+  myapp:
+    frontend: 80
+    backend: 80
```

Update nginx role's tasks/main.yml

```
 - name: install nginx
   apt: name=nginx state=present update_cache=yes

-- name: configure nginx site
-  template: src=nginx.conf.j2 dest=/etc/nginx/sites-available/demo mode=0644
+- name: configure nginx sites
+  template: src=nginx.conf.j2 dest=/etc/nginx/sites-available/{{ item.key }} mode=0644
+  with_dict: sites
   notify: restart nginx

 - name: de-activate default nginx site
   file: path=/etc/nginx/sites-enabled/default state=absent
   notify: restart nginx

-- name: activate demo nginx site
```

```
-  file: src=/etc/nginx/sites-available/demo dest=/etc/nginx/sites-enabled/demo state=link
+- name: activate nginx sites
+  file: src=/etc/nginx/sites-available/{{ item.key }} dest=/etc/nginx/sites-enabled/{{ item.key }}
state=link
+  with_dict: sites
   notify: restart nginx
```

Update nginx role's templates/nginx.conf.j2

```
-upstream demo {
+upstream {{ item.key }} {
 {% for server in groups.webserver %}
-    server {{ server }};
+    server {{ server }}:{{ item.value.backend }};
 {% endfor %}
 }

 server {
-    listen 80;
+    listen {{ item.value.frontend }};

     location / {
-        proxy_pass http://demo;
+        proxy_pass http://{{ item.key }};
     }
 }
```

Selective Removal: shell, register, with_items, when |

From nginx role's tasks/main.yml we will find all the nginx enabled sites (soft link in /etc/nginx/sites-enabled) and unlink those which we do not want now (demo -> /etc/nginx/sites-available/myapp)

```
# vi roles/nginx/tasks/main.yml
-- name: de-activate default nginx site
-  file: path=/etc/nginx/sites-enabled/default state=absent
+- name: get active sites
+  shell: ls -1 /etc/nginx/sites-enabled
+  register: active
+
+- name: de-activate sites
+  file: path=/etc/nginx/sites-enabled/{{ item }} state=absent
+  with_items: active.stdout_lines
+  when: item not in sites
   notify: restart nginx

 - name: activate nginx sites
```

Variables continued | will push demo.wsgi as a template from demo_app role
Move demo.wsgi to demo_app role's templates directory

```
# mv roles/demo_app/files/demo/app/demo.wsgi roles/demo_app/templates/demo.wsgi.j2
```

Update the demo.wsgi.j2 with db variables names.

```
activate_this = '/var/www/demo/.venv/bin/activate_this.py'
execfile(activate_this, dict(__file__=activate_this))

import os
+os.environ['DATABASE_URI'] = 'mysql://{{ db_user }}:
{{ db_pass }}@{{ groups.database[0] }}/{{ db_name }}'

import sys
sys.path.insert(0, '/var/www/demo')

from demo import app as application
```

Refer the template demo.wsgi.j2 in demo_app role's tasks/main.yml

```
# vi roles/demo_app/tasks/main.yml

  copy: src=demo/app/ dest=/var/www/demo mode=0755
  notify: restart apache2

+- name: copy demo.wsgi
+  template: src=demo.wsgi.j2 dest=/var/www/demo/demo.wsgi mode=0755
+  notify: restart apache2
+
 - name: copy apache virtual host config
   copy: src=demo/demo.conf dest=/etc/apache2/sites-available mode=0755
   notify: restart apache2
```

Put the variables name in webserver.yml playbook where we refer the nginx role

```
# vi webserver.yml
 become: true
  roles:
    - apache2
-    - demo_app
+    - { role: demo_app, db_user: demo, db_pass: demo, db_name: demo }

# ansible-playbook webserver.yml
```

Variables: vars_files, group_vars | Use group_vars/all file to specify variables for database.
Database variables are referred in two places
demo.wsgi.j2 template from demo_app role &
tasks/main.yml from mysql role.

We will use group_vars/all file to specify below variables
db_name: demo
db_user: demo
db_pass: demo

```
# mkdir group_vars
# vi group_vars/all

+---
+db_name: demo
+db_user: demo
+db_pass: demo
```

Remove db variables from websever.yml playbook

```
# vi webserver.yml

  become: true
  roles:
    - apache2
-    - { role: demo_app, db_user: demo, db_pass: demo, db_name: demo }
+    - demo_app
```

Refer db variables from group_vars/all file inside database.yml playbook also called as variable routing.
Variable routing can de defined as one variable pointing to another variable.
E:g db_user_name: "{{ db_user }}"

```
# vi database.yml

 - hosts: database
   become: true
   roles:
-    - { role: mysql, db_name: demo, db_user_name: demo, db_user_pass: demo, db_user_host: '%' }
+    - role: mysql
+      db_user_name: "{{ db_user }}"
+      db_user_pass: "{{ db_pass }}"
+      db_user_host: '%'
```

Variables: vautl | encrypting db_pass variables value with vault
https://docs.ansible.com/ansible/playbooks_vault.html

Instead of having all file for variables, we will create all directory and move our variables in all directory.

```
# cd group_vars
# mv all vars
# mkdir all
# mv vars all
# export EDITOR=vim
```

Change directory to all directory and create vault. Vault file should be present in all directory for our scenario.

cd all
ansible-vault create vault
Give a vault password and put below mentioned content.

+ ---
+ vault_db_pass : admin123

Refer to vault_db_pass in all/vars file

vi vars
+---
+db_name: demo
+db_user: demo
+db_pass: "{{ vault_db_pass }}"

Execute database.yml playbook, you should get some error like below

ansible-playbook database.yml

ERROR! Decryption failed
ERROR! A vault password must be specified

Either you can give --ask-vault-pass option and execute the playbook which will ask you the vaultpassword or you can use a file where you specify vault password.

"vaultpass" is our vault password.

echo "vaultpass" > ~/.vault_pass.txt
chmod 0600 ~/.vault_pass.txt
vi ansible.cfg
 [defaults]
 inventory = /dev
+vault_password_file = ~/.vault_pass.txt

Execute database playbook to test it
ansible-playbook database.yml

Login to db01 instance and login to mysql database to verify.
ssh db01
mysql -h localhost -u demo -p

Advanced Execution

Advanced Execution: gather_facts | By disabling gather_facts we can save the execution time.

Time the execution with time command for site.yml, stack_status and stack_restart.yml
time ansible-plabook site.yml
time ansible-plabook stack_status.yml
time ansible-plabook stack_restart.yml

Disable gather_facts for all the plays except dbserver because we use fact varible for dbservers play. ansible_eth1.ipv4.address

control.yml

```
---
- hosts: control
  become: true
+  gather_facts: false
  roles:
    - control
```

loadbalancer.yml

```
---
- hosts: loadbalancer
  become: true
+  gather_facts: false
  roles:
    - nginx
```

hostname.yml

```
---
  - hosts: all
+    gather_facts: false
     tasks:
       - name: get server hostname
         command: hostname
```

stack_restart.yml

```
# Bring stack down
- hosts: loadbalancer
  become: true
+  gather_facts: false
  tasks:
    - service: name=nginx state=stopped
    - wait_for: port=80 state=drained

- hosts: webserver
  become: true
+  gather_facts: false
  tasks:
    - service: name=apache2 state=stopped
    - wait_for: port=80 state=stopped

# Bring stack up
- hosts: webserver
  become: true
+  gather_facts: false
  tasks:
    - service: name=apache2 state=started
    - wait_for: port=80
```

```
- hosts: loadbalancer
  become: true
+ gather_facts: false
  tasks:
    - service: name=nginx state=started
    - wait_for: port=80
```

stack_status.yml

```
---
- hosts: loadbalancer
  become: true
+ gather_facts: false
  tasks:
    - name: verify nginx service
      command: service nginx status


- hosts: webserver
  become: true
+ gather_facts: false
  tasks:
    - name: verify apache2 service
      command: service apache2 status

      wait_for: host={{ ansible_eth0.ipv4.address }} port=3306 timeout=1

- hosts: control
+ gather_facts: false
  tasks:
    - name: verify end-to-end index response
      uri: url=http://{{item}} return_content=yes

      with_items: "{{lb_db.results}}"

- hosts: loadbalancer
+ gather_facts: false
  tasks:
    - name: verify backend index response
      uri: url=http://{{item}} return_content=yes
```

webserver.yml

```
---
- hosts: webserver
  become: true
+ gather_facts: false
  roles:
    - apache2
```

Time the execution again with time command to verify.

Extracting Repetitive Tasks: cache_valid_time |

Open site.yml and add below mentioned play

```
 ---
+- hosts: all
+  become: true
+  gather_facts: false
+  tasks:
+    - name: update apt cache
+      apt: update_cache=yes cache_valid_time=86400
+
 - include: control.yml
 - include: database.yml
 - include: webserver.yml
```

Remove update_cache=yes parameter from all the tasks/main.yml of all the roles.

```
# vi roles/mysql/tasks/main.yml
E:g
-  apt: name=mysql-server state=present update_cache=yes
+  apt: name=mysql-server state=present
```

Limiting Execution by Hosts : limit

```
# ansible-playbook site.yml --limit app01
```

Limiting Execution by Tasks : tags

We can also select particular task or tasks by tagging them and then using tag name while executing playbooks or site.yml.

```
# vi roles/control/tasks/main.yml
   with_items:
     - apache2
     - libapache2-mod-wsgi
+  tags: [ 'packages' ]
```

List the available tags.

```
#  ansible-playbook site.yml --list-tags
```

Execute it to verify

```
# ansible-playbook site.yml --tags "packages"
```

We can also skip the specific tags and execute the rest.

```
#  ansible-playbook site.yml --skip-tags "packages"
```

Tag all the tasks as per our 4 principles of app deployment on Linux systems

```
['packages']
['service']
['system']
['configure']
```

```
# vi roles/apache2/tasks/main.yml

  with_items:
    - apache2
    - libapache2-mod-wsgi
+  tags: [ 'packages' ]

- name: ensure mod_wsgi enabled
  apache2_module: state=present name=wsgi
  notify: restart apache2
+  tags: [ 'system' ]

- name: de-activate default apache site
  file: path=/etc/apache2/sites-enabled/000-default.conf state=absent
  notify: restart apache2
+  tags: [ 'system' ]

- name: ensure apache2 started
  service: name=apache2 state=started enabled=yes
+  tags: [ 'service' ]

# vi roles/control/tasks/main.yml

  with_items:
    - curl
    - python-httplib2
+  tags: [ 'packages' ]

# vi roles/demo_app/tasks/main.yml

    - python-pip
    - python-virtualenv
    - python-mysqldb
+  tags: [ 'packages' ]

- name: copy demo app source
  copy: src=demo/app/ dest=/var/www/demo mode=0755
  notify: restart apache2
+  tags: [ 'configure' ]

- name: copy demo.wsgi
  template: src=demo.wsgi.j2 dest=/var/www/demo/demo.wsgi mode=0755
  notify: restart apache2
+  tags: [ 'configure' ]

- name: copy apache virtual host config
  copy: src=demo/demo.conf dest=/etc/apache2/sites-available mode=0755
  notify: restart apache2
+  tags: [ 'configure' ]

- name: setup python virtualenv
```

```
  pip: requirements=/var/www/demo/requirements.txt virtualenv=/var/www/demo/.venv
  notify: restart apache2
+  tags: [ 'system' ]

- name: activate demo apache site
  file: src=/etc/apache2/sites-available/demo.conf dest=/etc/apache2/sites-enabled/demo.conf
state=link
  notify: restart apache2
+  tags: [ 'configure' ]

# vi roles/mysql/tasks/main.yml

  apt: name={{item}} state=present
  with_items:
    - python-mysqldb
+  tags: [ 'packages' ]

- name: install mysql-server
  apt: name=mysql-server state=present
+  tags: [ 'packages' ]

- name: ensure mysql listening on all ports
  lineinfile: dest=/etc/mysql/my.cnf regexp=^bind-address
            line="bind-address = {{ ansible_eth0.ipv4.address }}"
  notify: restart mysql
+  tags: [ 'configure' ]

- name: ensure mysql started
  service: name=mysql state=started enabled=yes
+  tags: [ 'service' ]

- name: create database
  mysql_db: name={{ db_name }} state=present
+  tags: [ 'configure' ]

- name: create user
  mysql_user: name={{ db_user_name }} password={{ db_user_pass }}
priv={{ db_name }}.*:ALL
            host='{{ db_user_host }}' state=present
+  tags: [ 'configure' ]

# vi roles/nginx/tasks/main.yml

  apt: name={{item}} state=present
  with_items:
    - python-httplib2
+  tags: [ 'packages' ]

- name: install nginx
  apt: name=nginx state=present
+  tags: [ 'packages' ]
```

```
- name: configure nginx sites
  template: src=nginx.conf.j2 dest=/etc/nginx/sites-available/{{ item.key }} mode=0644
  with_dict: sites
  notify: restart nginx
+  tags: [ 'configure' ]

- name: get active sites
  shell: ls -1 /etc/nginx/sites-enabled
  register: active
+  tags: [ 'configure' ]

- name: de-activate sites
  file: path=/etc/nginx/sites-enabled/{{ item }} state=absent
  with_items: active.stdout_lines
  when: item not in sites
  notify: restart nginx
+  tags: [ 'configure' ]

- name: activate nginx sites
  file: src=/etc/nginx/sites-available/{{ item.key }} dest=/etc/nginx/sites-enabled/{{ item.key }}
state=link
  with_dict: sites
  notify: restart nginx
+  tags: [ 'configure' ]

- name: ensure nginx started
  service: name=nginx state=started enabled=yes
+  tags: [ 'service' ]

# vi site.yml

  tasks:
    - name: update apt cache
      apt: update_cache=yes cache_valid_time=86400
+      tags: [ 'packages' ]

- include: control.yml
- include: database.yml.
```

Save execution time by skipping packages tags
time ansible-playbook site.yml --skip-tags "packages"

Troubleshooting, Testing & Validation

Ordering problem

Make sure to have mysql service start task after you changed the mysql configuration.
If there is some typo or bad config in my.cnf, notify mysql restart is going to stop mysql and while starting it will fail. If you run the playbook again first it will hit mysql start task which will fail to start the mysql service and your play is not going to reach to the task where it pushes new my.cnf file to fix the issue. Ordering is very important in such scenario.

```
- name: Install mysql-server
  apt: name=mysql-server state=present


- name: ensure mysql listening on all ports
  lineinfile: dest=/etc/mysql/my.cnf regexp=^bind-address
            line="bind-address = {{ansible_eth1.ipv4.address}}"

  notify: restart mysql

- name: Ensure Mysql started
  service: name=mysql state=started enabled=yes
```

Jumping to Specific tasks: list-tasks, step, start-at-task
https://docs.ansible.com/ansible/playbooks_startnstep.html

ansible-playbook site.yml --step

Select y for yes, n for no and c to continue with the execution without asking the prompt.

List all the tasks from the playbook and select the task from where you want to start the execution.
ansible-playbook site.yml --list-tasks

ansible-playbook site.yml --start-at-task "copy demo app source"

Retrying failed hosts

ansible-playbook site.yml --limit @/home/ansible/site.retry

Syntax-Check & Dry-Run: syntax-check, check

ansible-playbook --syntax-check site.yml
ansible-playbook --check site.yml