Ansible:

Setup of ansible:

1. Create aws instance and name them as controller and servers
2. On all the machines downgrade the python to 2.7
3. Connect instace and update apt repo

Sudo apt-get update

Sudo apt-get dist-upgrade

1. Install python

Suso apt-get install -y python2.7 python-pip

1. Establish the password less ssh connectivity b/w the controller and manage nodes
2. Connect to controller and generate ssh-keygen

Ssh-keygen

Ssh-copy-id Ubuntu@private-ip-nodes

1. Install ansible on the controller
2. Update apt repository

Sudo apt-get update

1. Install software properties

Sudo apt-get install software-properties-common

1. Add latest version of ansible to apt

Sudo apt-add-repository ppa:ansible/ansible

1. Sudo apt-get update
2. Sudo apt-get install -y ansible

Ansible reads file called inventory file where we have to store the ip address of all the manage nodes

1. Sudo vim /etc/ansible /hosts

Copy and paste all the p address of manage nodes

Ansible performs remote configuration of servers in the fallowing three ways

1. Using Adhoc commands
2. Using playbooks
3. Using roles

Ansible is created using python and it uses inbuilt python modules for remote configuration of servers. These modules are used in above three ways

Important modules in ansible:

1. command: this is used for executing linux command on the manage nodes
2. shell: this is used for executing linux command which involves re direction and piping
3. ping: used to check if the remote manage node is pingle or not
4. user: used in user management where we can create user, assigning home directories, default working shells etc
5. file: used for creating or deleting files and directories on the manage node
6. copy: this is used for copying files and directories from controller to the managed nodes
7. fetch: this is used for copying file from manage nodes to the controller
8. apt: this is used for package management on the manage nodes like installing s/w upgrading

this works on Ubuntu, Dbane etc flavors of linux

1. yum: this is also same as apt bit it works on centOS, redhat linux etc
2. service: this is used for starting stopping restarting services on the manage nodes
3. uri: this is used for checking of remote url is reachable or not from the manage nodes
4. git: usfull in performing git version controlling on the manage nodes
5. debug: this is used for displaying the o/p of a module
6. include: this is used for calling child playbook from a parent playbook
7. stat: used for capturing detailed info about files and folders present on the manage nodes

Adhoc commands:

=================================================================================

Syntax: ansible all/group\_nam/ip\_address -i path\_of\_inventory\_file -m module\_name -a ‘arguments’

Command module:

ansible command to memory info of all manage nodes

ansible all -i /etc/ansible /hosts -m command -a ‘free -m’

The default inventory file is /etc/ansible/hosts and it is not mandatory to give -i option when working on this file. The default modul is command and it is not mandatory use –m command when we are using command module

Ansible all -a ‘free -m’

Shell module:

1. Ansible command to store the o/p of ls –la into the file1 of all managed nodes

Ansible all –m shell –a ‘ls –la > file1

1. Ansible command for downloading and installing docker

Ansible all –m shell –a ‘curl-fsSL <https://getdocker.com> –o get-docker.sh’

Ansible all –m shell –a ’sh get-docker.sh’

User module:

1. Ansible command to create user in manage nodes

Ansible all -m user -a ‘name=srikala password=sri’ –b---------b is for giving root permissions in manage nodes

1. User module can also be used for assigning user id home directory, default working shell etc

Ansible all –m user –a ‘name =srik password=asri home=/home/ubuntu/srik uid=1234 comment=”normal user” shell=/bon/bash’ –b

File module:

This is used for creating files and directories on the managed nodes

1. Ansible command to create a file on all managed nodes

Ansible all –m file –a ‘name=/tmp/file1 state=touch’

Note: State = touch for creating file

State = directory creating directory

State=absent deleting file or directory

1. Ansible command to create a file on managed nodes and also control ownership group ownership permissions etc…

Ansible all –m file –a ‘name = file2 state =touch owner=srikala group=durga mode=770’ -b

Copy module:

This is used for copying files and directories form controller to managed nodes

1. Anisble command to copy /etc/pwd file from controller to all managed nodes

Ansible all –m copy =a ‘src=/etc/passwd dest=/tmp’ –b

1. Copy module cab also be used for changing the ownership group ownership and permissions of the files that are copied into the managed nodes

Ansible all –m copy –a ‘name = file2 state =touch owner=root group=durga mode=770’ -b

1. To change the content of the file present in the managed nodes

Ansible all –m copy –a ‘content = “hello/n” dest=/tmp/passwd’ –b

Fetch module:

1. Ansible command to copy /etc/passwd file from manage nodes to controller

Ansible all –m fetch =a ‘src=/etc/passwd dest=/tmp’ –b

Apt module-:

This is used for installing and uninstalling software’s on the managed nodes

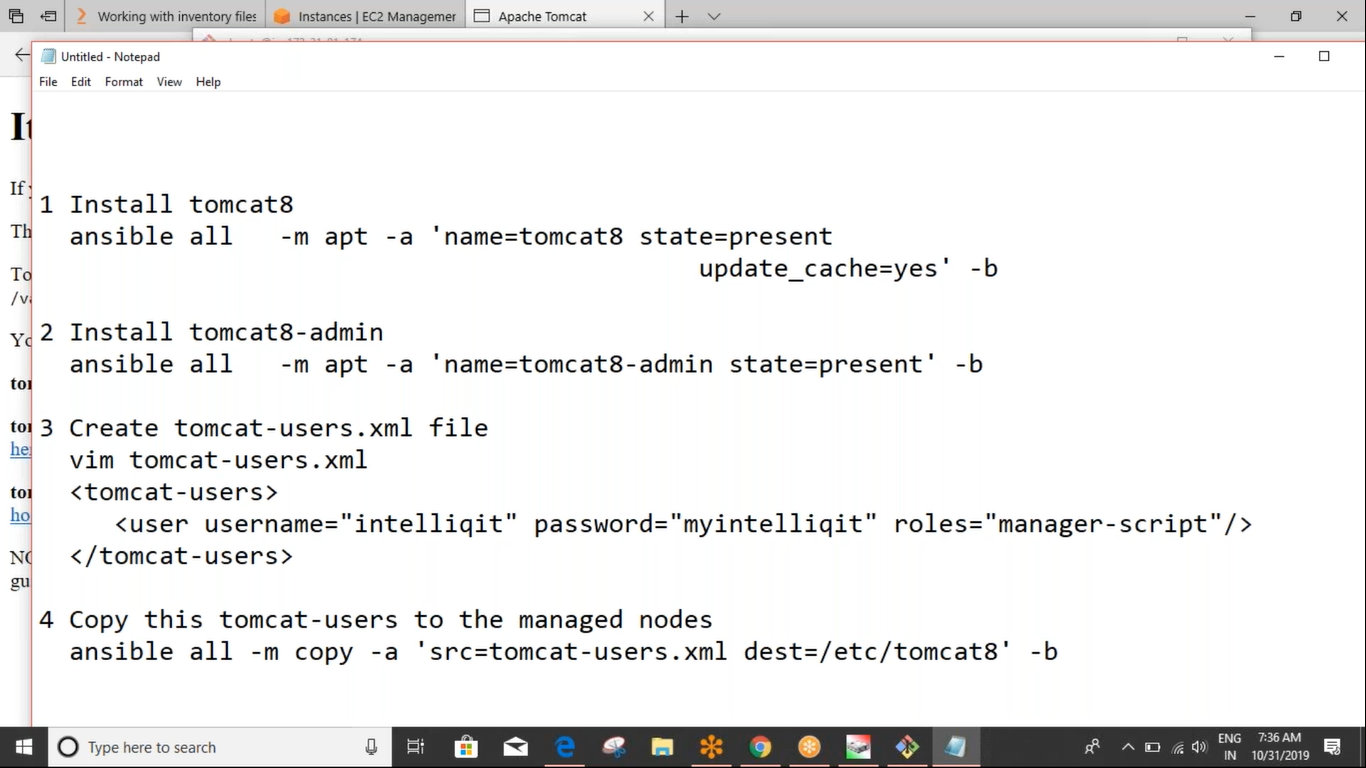
1. Ansible command to install the tomcat in managed nodes

Ansible all –m apt –a ‘name=tomcat8 state=present’ –b

Notes: State = present for installation

State = absent for uninstallation

State = latest is for upgrading to the latest version

1. to update the apt repository before installing s/ws we can use the option update\_cache=yes
2. Install tomcat8 and tomcat8-admin in all managed nodes and create tomcat-users .xml file on controller, copy that in to all managed nodes
3. 
4. Install apache install nodes and edit the default apache file location /var/www/html/index/html

Ansible all –m apt –a ‘name =apche2 state=present update\_cache=yes’ –b

Service module:

This is used for stating stopping and restarting services

1. Restart tomcat8

Ansible all –m service –a ‘name=tomcat8 state=restarted’ –b

Note : State = restarted foe restarting the service

State = started for staring the service

State= stopped for stopping the service

Git module:

This is for Git version controlling on managed nodes

1. Ansible command to clone remote git repository into all managed nodes

Ansible all –m git –a ‘repo=repourl dest=/tmp/folder1

Replace module:

1. Change the port number of tomcat from 8080 to 9090 and restart tomcat8

Ansible all –m replace –a ‘regexp=8080 replace=9090 path=/etc/tomcat8/server.xml’ –b

Uri module:

This is used to check if the remote url is reachable or not if it is reachable it will return the status as 200 else it will return the status as -1

1. Ansible command whether Google is reachable from all managed nodes

Ansible all –m uri –a ‘url=http://google.com status=200’

Sample yaml code:

---

intelliq:

  trainesr:

    devops: dd

    aws: ddd

  cooardinatea:

   devops: laskm

   aws: mm

Adhoc command can be used to work on only a single module and a single set of arguments. When we want to persorm complex configuration management activities adhoc command are difficult to manage so we can use playbooks. Play books are created using yaml syntax and they are designed to work on the single host or group of hosts or all the hosts. Play book is a combination of plays. Main advantage of using playbooks over adhoc commands is reusability

Ansible playbook creating on all managed nodes:

---

- name: creating files

  hosts: all

  tasks:

  - name: file creating

    file:

      name: /home/ubuntu/file1

      state: touch

...

1. To check whether play book is created syntactically correct or not

Ansible –playbook playbook.yaml --syntax-check

1. To run the playbook

Ansible-palybook playbook.yaml -b

Ansible playbook for installing git on all managed nodes:

---

- name: insatll git and clone remote repo

  hosts: all

  tasks:

    - name: install git

      apt:

        name: git

        state: present

    - name: clone remote repo

      git:

        repo: https://github.com/intelliqittrainings/maven.git

        dest: /home/ubuntu/mygit

ansible playbook for configuring tomcat8:

---

- name: configuring:

  hosts: all

  tasks:

    - name: tomcat8 install

      apt:

        name: tomcat8

        state: present

    - name: change port

      replace:

        regexp: 8080

        replace: 9090

        path: /etc/tomcat8/server.xml

    - name: restart tomcat8

      service:

        name: tomcat8

        state: restarted

    - name: check url respone of server1

      uri:

        url: http://ipaddress:9090

        status: 200

    - name: check url respone of server1

      uri:

        url: http://ipaddress:9090

        status: 200

Ansible playbook for edit index.html file:

---

- name: configaration

  hosts: all

  tasks:

    - name: installing apache2

      apt:

        name: apache2

        state: present

    - name: edit in index.html

      copy:

        content: "intelliq home page\n"

        dest: /var/www/html/index.html

        path:

    - name: restart the server

      service:

        name: apache2

        state: restarted

    - name: check the server

      uri:

        url: http://ipaddress

        status: 200

    - name: check the server

      uri:

        url: <http://ipaddr>

ess

        status: 200

Variables in Ansible :

Ansible supports three types of variables

1. Global scope variables
2. Host scope variables
3. Play scope variables

Global scope variables:

these variables are passed from the command prompt using

--extra-vars. They have the highest level of priority

---

- name: insatll/uninsatll s/w applications

  hosts: all

  tasks:

    - name: install/unintall

      apt:

        name: "{{a}}"

        state: "{{b}}"

        update\_cache: "{{c}}"

ansible-playbook playbbok.yaml --extras-vars “a=git b=present c=no” –b

ansible playbook for creating users

---

- name: creating users and files

  host: all

  tasks:

    - name: create user

      user:

        name: "{{a}}"

        password: "{{b}}"

        home: "{{c}}"

    - name: create files or directpries

      file:

        name: "{{d}}"

        state: "{{e}}"

ansible-playbook playbook.yaml --extras-vars “a= rano b=inet c=/home/Ubuntu/rano d=/home/Ubuntu/rani/dir1 e=directory”

Play scope variables:

These variables are defines within the playbook and they can effect only a single play. They have the last level of priority

---

- name: creating users and files

  host: all

  vars:

    - a: tomcat8

    - b: present

    - c: no

  tasks:

    - name: insatll tomcat

      apt:

        name: "{{a}}"

        state: "{{b}}"

        update\_cache: "{{c}}"

the above playbook works like a template whose default behavior is to install tomcat8 but we can bypass that and make it work on any other sets of data by passing global scope variables using --extrs -vars

Grouping in inventory file:

Group names can be provided by giving those names in square brackets

Vim /etc/ansible/hosts

[weserver]

Ipaddess

[appserver]

Ipaddress

[server:children]

Websever

Appserver

Host scope variable:

These variables are classifies into two types:

1. Variable to work on group of hosts
2. Variable to work on single host

Variable to work on group of hosts:  
-----------------------------------------------------------------------------------------------------------------------------------------

These variables should be created in a folder called group\_vars. This folder should be created in the same location where all the playbooks are present. In this folder create a file with a name same as group name from the inventory file and create the variables.

1. Change directory to the folder where playbooks present

Cd path\_of\_paybooks\_folder

1. Mkdir group\_vars

Cd gropu\_vars

1. Create file with name webserver

Vim webserver

---

A: usha

B: intellis

C: /home/usha

D: 2468

…

1. Create playbook to use the variables after going into playbook folder
2. Vim playbook.yaml

--

Name:user creation

Hosts: webserver

tasks:

name: crate auser

user:

name:{{a}}

password:{{b}}

home:{{c}}

uid:{{d}}

1. Run plaubook

Variable to work on single host:

These variables should be created in folder host\_vars and this folder should be created in same folder where playbooks are present. This this host\_vars folder we should create a file whose name is same as ipaddress of one machine from the inventory file and in this file we should create the variables. These variables will work only on that machine.

1. Change directory to the folder where playbooks present

Cd path\_of\_paybooks\_folder

1. Mkdir group\_vars

Cd gropu\_vars

1. Create file (name of file should be same as the ipaddress of machine)

Vim ipaddress

---

A: usha

B: intellis

C: /home/usha

D: 2468

…

1. Create playbook to use the variables after going into playbook folder
2. Vim playbook.yaml

--

Name: user creation

Hosts: ipaddress

tasks:

name: crate auser

user:

name:{{a}}

password:{{b}}

home:{{c}}

uid:{{d}}

1. Run plaubook

Looping in ansible:

We can make a specific module run multiple times using loops. Loops can be implemented using with\_items and with\_sequence

Ansinle playbook for installing multiple software packages using with\_items

---

- name: insatlling s/w application

  hosts: all

  tasks:

    - name: install s/w

      apt:

        name: "{{item}}"

        state: present

        upadate\_cache: no

      with\_items:

        - tree

        - openjdk-8-jdk

        - git

        - firewall

Ansible playbook for installing and uninstalling various s/w packages with or without updating apt repositories

---

- name: insatlling s/w application

  hosts: all

  tasks:

    - name: install s/w

      apt:

        name: "{{item.a}}"

        state: "{{item.b}}"

        upadate\_cache: "{{item.c}}"

      with\_items:

        - {a: git, b: present, c: no}

        - {a: openjdk-8-jdk, b: absent, c: no}

        - {a: git, b: present, c: no}

Ansible for installing tomcat 8 with\_items

---

- name: configuring:

  hosts: all

  tasks:

    - name: tomcat8 install

      apt:

        name: tomcat8

        state: present

    - name: change port

      replace:

        regexp: 8080

        replace: 9090

        path: /etc/tomcat8/server.xml

    - name: restart tomcat8

      service:

        name: tomcat8

        state: restarted

    - name: check url respone of server1

      uri:

        url: "{{item}}}"

        status: 200

      with\_items:

       - http://ipdreess

       - http://ipdreess

       - http://ipdreess

Always with\_items run on one module only

Creating files and directories using with\_items

---

- name: creating user and files

  hosts: all

  tasks:

    - name: creating multiple users

      user:

        name: "{{item.a}}"

        password: "{{item.b}}"

        home: "{{item.c}}"

      with\_items:

        - {a: ravi, b: ravi, c: /home/ubuntu/ravi}

        - {a: anu, b: anu, c: /home/ubuntu/anu}

    - name: creating files

      file:

        name: "{{item.a}}"

        state: "{{item.b}}"

      with\_items:

        - {a: /home/ubuntu/ravi/file1, b: touch}

        - {a: /home/ubuntu/anu/dir1, b: directory}

Aansible playbook for creating setup where dev env is configured for Jenkins, QA and Prod configured for tomcat8

---

- name: create CI/CD setup of jenkins

  hosts: ipadress

  tasks:

    - name: jenkins installation on dev server

      apt:

       name: "{{item}}"

       state: present

       update\_chache: no

     with\_items:

      - openjdk-8-jdk

      - git

      - maven

    - name: dowmload jenkins.war

      get\_url:

        url: http://mirrors.jenkins.io/war-stable/latest/jenkins.war

        dest: /tmp

- name: config aq and prod

  hosts: webserver

  tasks:

    - name: instal tomcat8

      apt:

      name: "{{item}}"

      state: present

    with\_items:

      - tomcat8

      - tomcat8-admin

    - name: copy tomcat file

      copy:

        src: tomcat-users.xml

        dest: /etc/tomcat8

Handlers:

Handlers are modules only if some other is executed successfully and it has made some changes. Handles are executed only after the entire tasks are executed. Handlers are executed in the order they are mentioned in the handlers section and not in the order that they are called in the tasks section. Even if a handler is called multiple times in the tasks section it will be executed only once

---

- name: implemeting handdlers

  host: all

  tasks:

    - name: install apache2

      apt:

        name: apache2

        state: present

      notify: check url responec

    - name: edit index.html

      copy:

        content: "Hello\n"

        dest: /var/ww/html/index.html

      notify: restart apche2

  handlers:

    - name: restart apache2

      service:

        name: apche2

        state: restarted

    - name: check url response

      uri:

        url: "{{item}}"

        status: 200

      with\_items:

        - <http://ipdrress>

- http://ipdrress

If a module fails and then still we want to call the module

Error handling or exceptional handling:

Whenever a specific module in ansible fails the program execution stops over there, if we want to continue the playbook even after encountering an error we can use error handling. The module which might generate error should be given in block section, if it fails the control comes into the rescue section, always section is executed every time irrespective of whether the code passes or fails

---

- name: error handling:

  hosts: all

  tasks:

    - block:

        - name: install tomcat7

          apt:

            name: tomcat7

            state: present

      rescue:

        - nmae: install tomcat 8

          apt:

            name: tomcat8

            state: present

      always:

        - name: restart

          service:

            name: tomcat8

            state: restarted

Ansible playbook for implementing all the stages of CI/CD:  
------------------------------------------------------------------------------------------------------------------------------------------

---

- name: insatall s/w for CI

  hosts: servers

  tasks:

    - name: install s/w

      apt:

        name: "{{item}}"

        state: present

      with\_items:

        - openjdk-8-jdk

        - git

        - maven

        - tomcat8

        - tomcat8-admin

- name: download the code from git and create artifact

  hosts: Devserver

  tasks:

    - name: continuos download

      git:

        repo: https://github.com/intelliqittrainings/maven.git

        dest: /home/ubuntu/project-code

    - name: continous build

      shell: cd /home/ubuntu/project- code; mvn package

    - name: copy artifact from dev to controller

      fetch:

        src: /home/ubuntu/project-code/webapp/target/webapp.war

        dest: /tmp

- name: deploy code into qa and run test

  hosts: qaserver

  tasks:

    - name: copy articat into qa serevr

      copy:

        src: /tmp/ipaddress/home/ubuntu/project-code/webapp/target/webapp.war

        dest: /var/lib/tomcat8/webapp/testwebapp.war

    - name: dowmload test programs

      git:

        repo: https://github.com/selenium-saikrishna/FunctionalTesting.git

        dest: /home/ubuntu/test-code

    - name: excute the code

      shell: java -jar /home/ubuntu/test-code/testing.jar

- name: deploy arrticat into prod server

  hosts: prodserver

    - name: contionus delivery

      copy:

        src: /tmp/ipaddress/home/ubuntu/project-code/webapp/target/webapp.war

        dest: /var/lib/tomcat8/webapp/prodwebapp.war

When conditions:

If we want to execute a module only when a specific condition is true then we can use when statement

---

- name: implementing conditional statments

  hosts: all

  vars:

    - a: 10

  tasks:

    -name: install tree

    apt:

      name: tree

      state: present

    when: a==10

Register and debug variable:

 ---

 - name: create user and display the o/p

  hosts: all

  tasks:

    - name: create user

      user:

        name: anu

        password: anu

        home: /home/anu

        uid: 8999

      register: a

    - name: dipsly o/p

      debug:

        var: a

Ansible playbook to check if a folder is present and if it is not present create file

---

- name: check if a drectoty present if note create afile

  hosts: all

  tasks:

    - name: check if dir f1

      stat:

        path: /home/ubuntu/f1

      register: a

    - name: display o/p of stat module

      debug:

        var: a

    - name: create file

      file:

        name: /home/unbuntu/f1

        state: touch

      when: a.stat.exists == false

Tags:

Tags are used for providing more modular control over the flow of execution of the playbook. This is done by assigning aliases to the modules.by using this only tagged modules we can excute like

1. --tags==tagged-----------------for only tagged one execution
2. --tags==untagged-------------only for untagged execution
3. --tags==user\_creation----------only specific module executed
4. ---
5. - name: tagging in ansible
6. hosts: all
7. tasks:
8. - name: install tree
9. apt:
10. name: tree
11. state: present
12. tags: tree\_installation
13. - name: user creation
14. user:
15. name: hari
16. password: intelliq
17. tags: user\_creation
18. - name: copy files
19. copy:
20. scr: /etc/passwd
21. dest: /tmp

Include module:

This module is used for calling child playbooks from a parent playbook. Only parent playbooks are executed not the child playbooks. Once creating all the modules we can include only which is wanted that is the advantage

Child playbook

---

- name: creationg child playbook

  file:

    name: /home/ubuntu/newdir

    state: directory

Parent playbook

---

- name: create directory using child playbook

  hosts: all

  tasks:

  - name: call child

    include: playbook.yaml

Create child playbooks for conf apche 2 and acll them from a parent playbook

---

- name: install apche2

  apt:

    naem: apche2

    state: present

---

- name: edit index file

  copy:

    content: "intelli/n"

    dest: /var/www/html/index/html

---

- name: restart apche2

  service:

    name: apche2

    state: restarted

---

- name: check url respone

  uri:

    url: "{{item}}"

    status: 200

  with\_items:

    - ipaddress

---

- name: configaring apche2 using child palybooks

  hosts: all

  tasks:

    - name: call child

      include: "{{item}}"

      with\_items:

      - playbooks.yaml

      - playbooks.yaml

      - playbooks.yaml

      - playbooks.yaml

Ansible vault:

This is a feature of ansible which is used for creating playbooks in a secure format. Thse playbooks can be accessed using password

1. To create valut playbook

Ansible-valut create playbook.yaml

1. To see content of valut playbook

Ansible-valut view playbook.yaml

1. To edit the content

Ansible-valut edit playbook.yaml

1. To convert ordinary playbook into valut playbook

Ansible-valut encrypt playbook.yaml

1. To convert vault playbook into normal

Ansible-valut decrypt playbook.yaml

Roles:

Roles is predefined folder structure which is used for configuring a s/w application. A role contains everything that is necessary for configuring the s/w application. Since it is pre-defined it is easy for everyone to understand in which folder what components are present. Roles support greater reusability and maintainability. Ansible roles are present in a site called ansible galaxy. This is cloud site of ansible, we can download roles from this site and customize them based on our requirements

To create a role:

Ansible-galaxy init role\_name --offline

Folder structure of roles:

Readme.md: This is the simple text file which contains info about the role in plain English

Defaults: This is folder is used for storing info about the application that is being configured. It also contains variables of less priority

Files: All the static configuration files that are necessary for configuring s/w application are stored in this folder

Handlers: These are modules which are executed only if some other module is executed successfully and it has made some changes. All such handles should be placed in this folder

Meta: Data about the data is called the meta data. This folder is used for storing info like the author of the role, date and time when it was created, the version it supports etc.

Tasks: The actual confi that should be performed on the remote server is stored in this folder

Templates: This is used for storing dynamic confi files which are used by the application

Tests: Ansible modules which are related to checking configuration activities on the remote servers are stored in this folder

Vars: This is used for storing variables and these variables are higher priority than the variables present in the vars folder

1. Create new role for apache2

Ansible-galaxy init apache2 --offline

1. Check the tree structure of role that we created

Tree apche2

1. Goto task folder and create the task for configuration

Cd apache2/tasks

Vim main.yaml

* Include: install apche2.yaml
* Include: config.yaml
* Include: vim check\_url\_reponce.yaml

Vim install.yaml

---

- name: install apche2

  apt:

    naem: apche2

    state: present

vim config.yaml

---

- name: copy index file

  copy:

    src: index.html

    dest: /var/www/html/index/html

notify:

restart apache2

vim check\_url\_reponce.yaml

---

- name: check url respone

  uri:

    url: "{{item}}"

    status: 200

  with\_items:

* ipaddress

    -

1. go to files folder to create index.html file

cd ..

cd files

sudo vim index.html

1. go to handlers folder

cd ..

cd handlers

sudo vim main.yaml

# handler for apche2

---

- name: restart apche2

  service:

    name: apche2

    state: restarted

cr 1eate the parent playbook to call the roles

cd ..

cd ..

sudo vim apache\_role.yaml

---

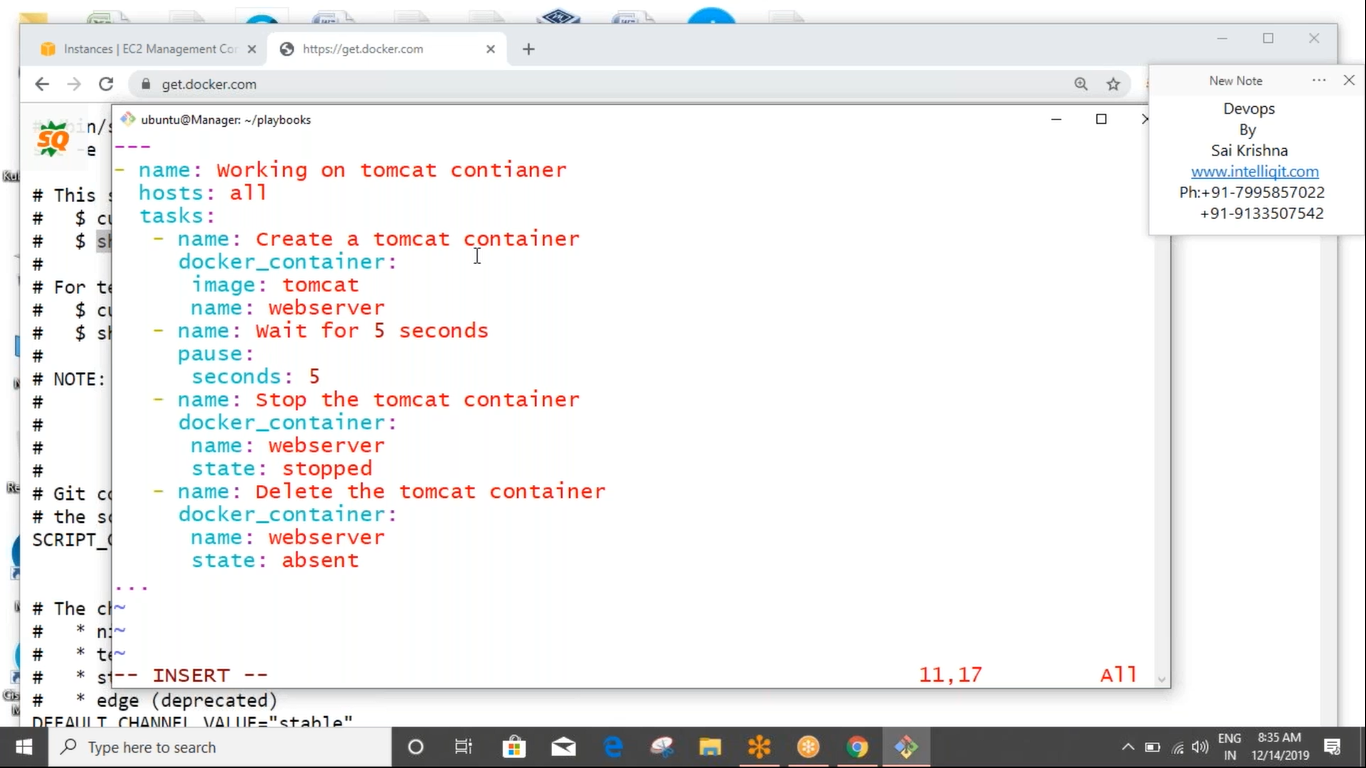
- name: implementing roles for apche2

Hosta: all

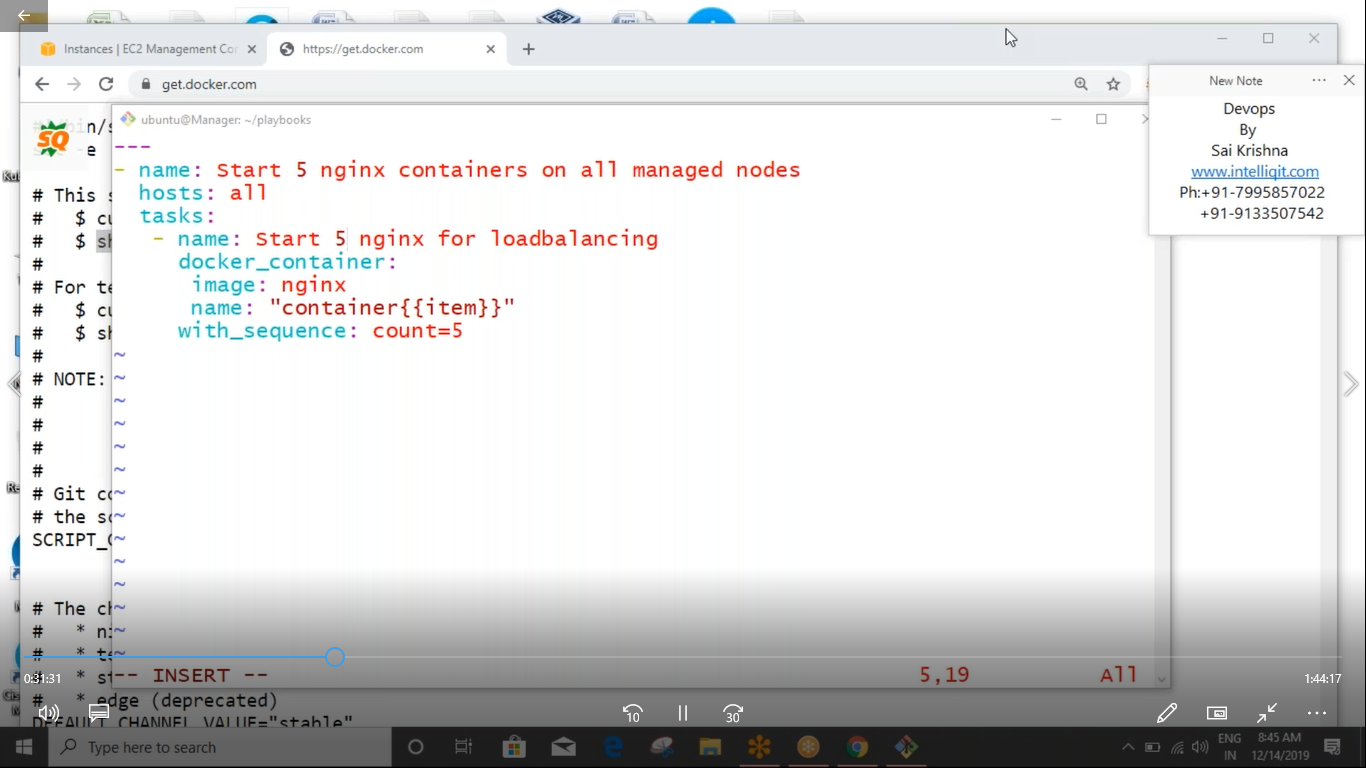
Roles: `

-apache2

Tomcat installing with docker container using ansible playbook

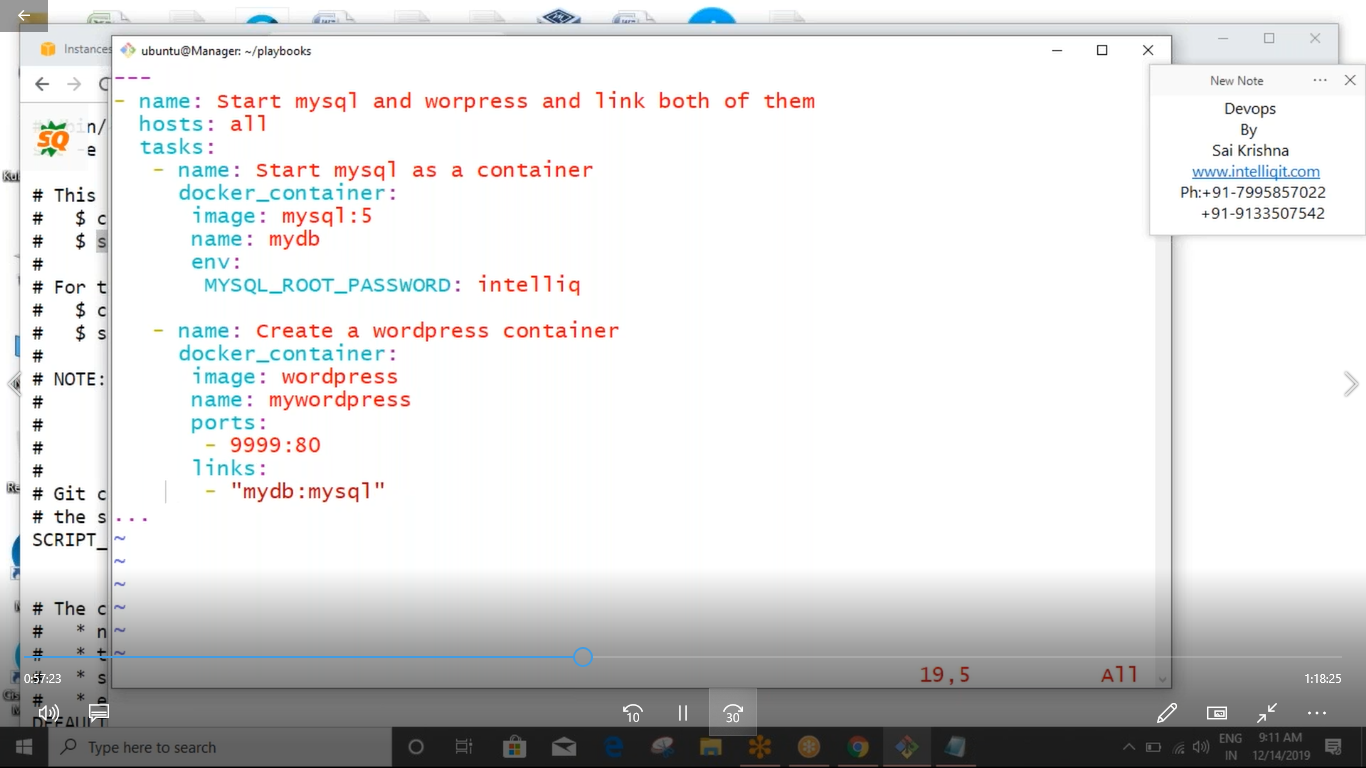


Nginx with 5 conatienrs

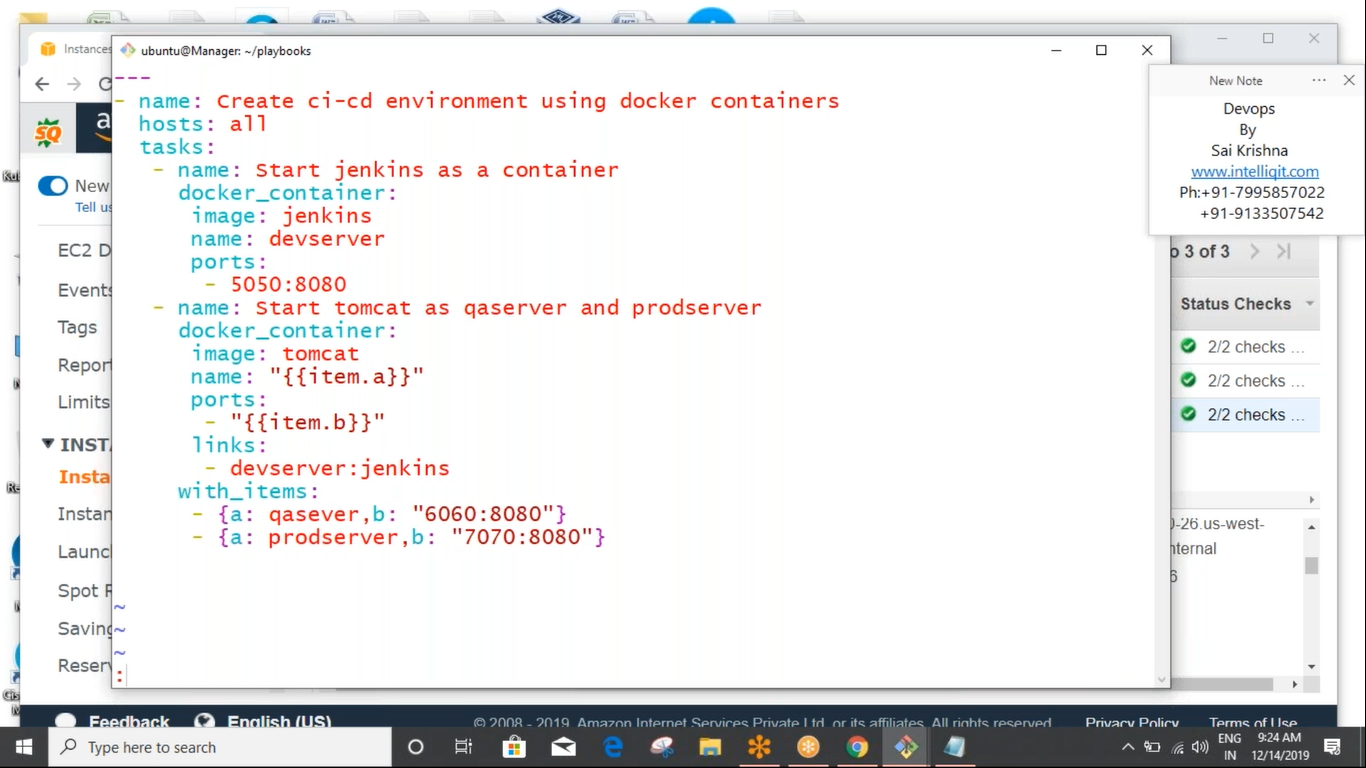


Ansible for starting Ubuntu and attaching volume to it

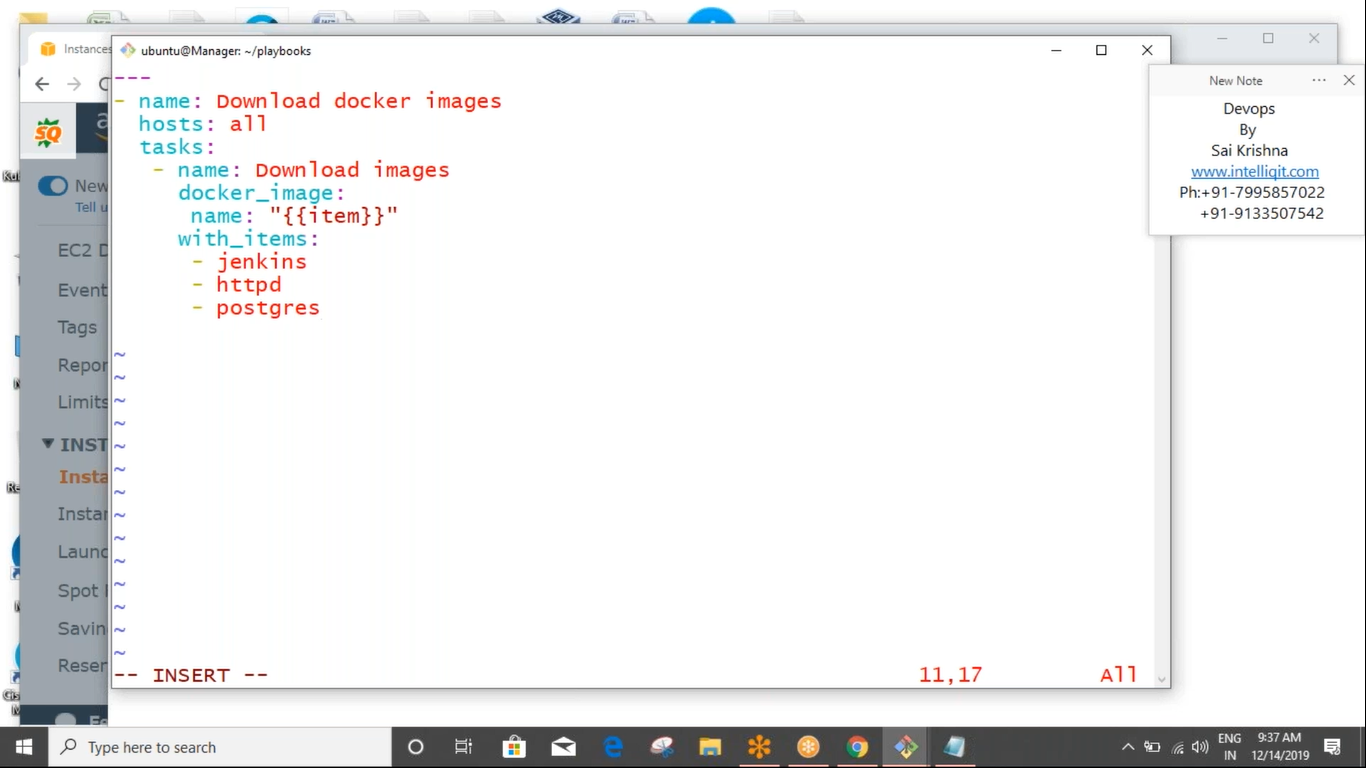
Docker compose where my sql is linked with wordpress



Jenkins with two tomcat servers



Download Docker image using ansibel playbook



Push customized docker image

