

For pull configuration, there is a client installed on each node server while for push there is no need for installing on node servers.

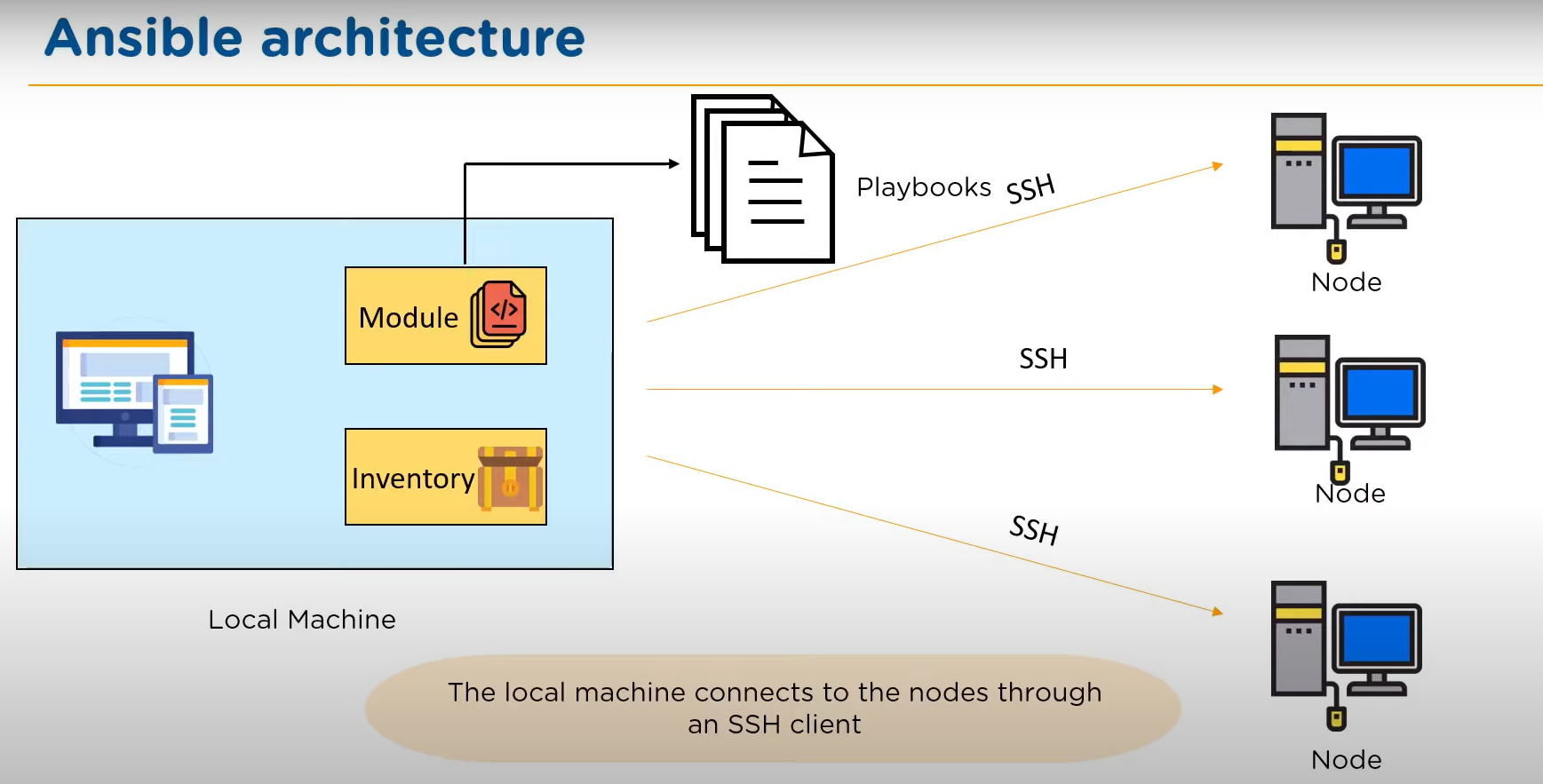
Ansible is a push configuration unlike chef and puppet.

—----------------------------

Architecture.

The local machine is where Ansible is installed, Nodes are the systems to be configured. They are controlled by the local machine, Module is a collection of configuration code files.

These configuration code files are called playbooks. Inventory is a document that groups the nodes under specific labels. Local machine connects to nodes through SSH.



From macbook:

created machine on cloud

nano hosts

[prod]

3.73.126.105 ansible\_ssh\_private\_key\_file=~/Documents/personal/ansible-p1/Personal-Jenkins-key.pem ansible\_ssh\_user=ubuntu

[demo]

127.0.0.1

ansible prod -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts -m ping

//Answer ping pong 🙂

ansible prod,demo -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts -m command -a "date"

//Answer with date from demo group and prod group only  
  
ansible all -m command -a “date” //all groups  
  
What if i want to parse ip not in the list,that is not there

ansible all -i 192.168.122.11, -m command -a “date” //do not forget comma after ip

//By this way IP is added to hosts file

In Ansible, an ad-hoc command is a one-time, on-the-fly command that you can run on remote hosts without the need for writing a separate Ansible playbook. Ad-hoc commands are useful for performing quick tasks or checks on remote servers. They are especially handy for tasks that don't require complex automation or for situations where you need to execute a command immediately without creating a playbook.

Here's the basic syntax of an Ansible ad-hoc command:

bash

ansible [target] -m [module] -a "[module arguments]"

ansible prod,demo -i /Users/abdelrahmanmagdy/Documents/personal/ansible-docker/hosts -m command -a "date"

While you can use the command module in Ansible for many tasks, there are several reasons why you might want to consider using specialized modules for specific operations instead of relying solely on the command module:

1)**Idempotence**:

2)Cross platform:

3)**Abstraction and Readability**:

4)Error Handling:

ansible prod -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts -m file -a "path=~/filee state=touch mode=777"

ansible prod -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts -m file -a "path=~/filee state=absent"

//The second command to delete the file

===

Playbook:  
Files on personal github  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts 1stplay.yml

Playbook Organize:

yaml file can contain many plays, we only did one play in all above examples,

If you started from hosts again then it is another play :)

So each play is a set of tasks with host. And each playbook can be multiple plays.

Playbook → Plays → Tasks  
  
what if i have 50 plays each having 10 tasks and there are many things that they are repeated ?

It is not logic to put all of this in a single playbook file.

Solution:

1) Use multi playbook files –we will need to know how they will call each other

2)Are all tasks sequential or conditional ? – solution: subtasks as “triggers” in programming

3) Play that have common tasks as close firewall , switch network manager to normal network, selinux set it to default. So the full play is constant.

A)About multi playbook files.

Project x have 1st playbook have 2 plays each having 3 tasks

2nd playbook have 1 play with 5 tasks

3rd playbook with 3 plays each have 1 tasl

All the above playbooks are yaml. To organize them you will need main.yml file

inside main.yml

import playbook1

import playbook2

import playbook3

When you run you will need to run the main.yml …..

ansible-playbook main.yml

B)About subtasks:

How will they talk to each others..

Here comes handlers

Handlers: We will talk in task level, you finish your task normally and at the end of task you type handlers: h1

after section of tasks

You come at the end of the play and type h1: makes tasks….

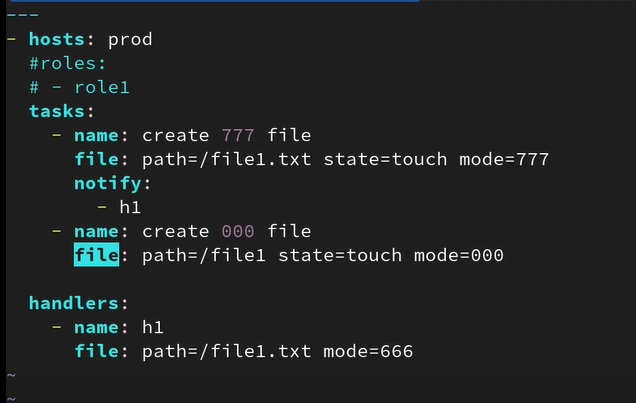
h2: makes tasks …..

h3: ….

So whenever you call them above they are run from the end of the play

Similar to functions in coding

vim 1stplay.yml

  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts 1stplay.yml

Important note: If task create 777 file failed the call for h1 will not be executed, while create 000 file will be executed even if create 777 file failed. roles commented as it will be discussed later.

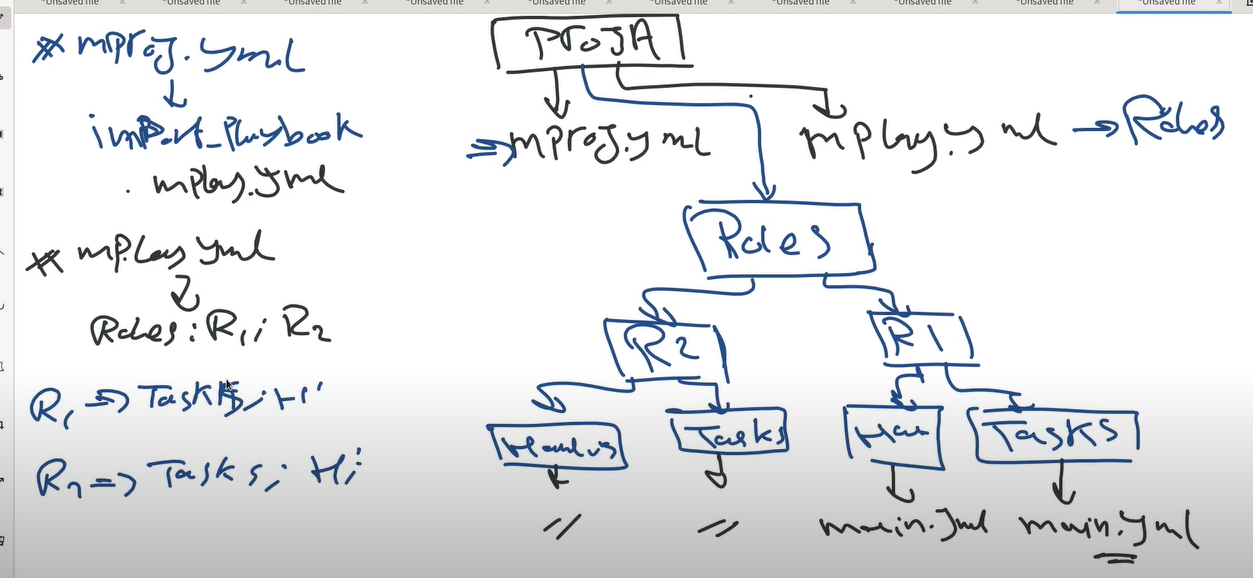
From mainjob folder, i did run  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p1/hosts main.yaml

Added extraplay and run the same above command again.

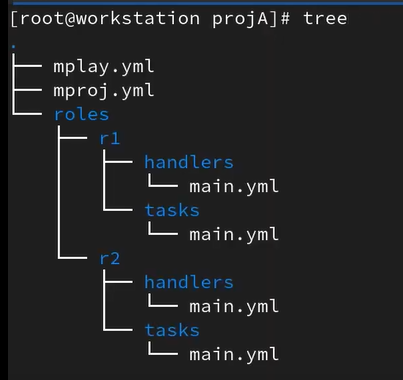
Check also code in extraplay2 and run it..

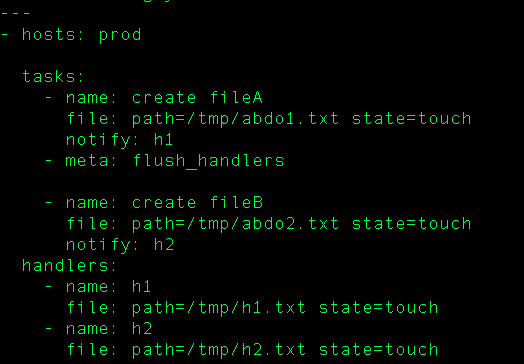
Section 2: In p2 directory

Roles:



Under R1 and R2 is tasks and handlers, mproj will import mplay.yml and mplay.yml will have the roles : r1, r2

  
If we will do it in the old way in one file the whole project it will be as below:

  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p2/hosts oldway.yml

But instead we will divide it as in ansible-p2 roles.

Then run:

ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p2/hosts mproj.yml

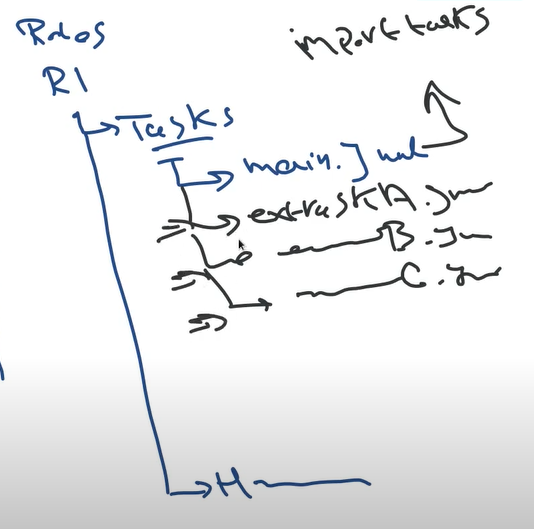
Note: In main.yml inside handlers or tasks we start directly from beginning of file, we also do not write handlers or tasks keyword.

Some points on above structure:

–Names of directories are predefined: tasks and handlers ,etc…must be as they are; you can not change their names. You can check names of directories from ansible documentation.

--Why are they all called main.yml ? Because ansible searches for main as it is the main that he searches for :) It can be main.yml or main.yaml or main ,, if not with these names he will not find it.

– There is also import tasks if you want to use more than main.yml , you will make extrataskA.yml , extrataskB.yml ,extrataskC.yml then use import tasks in main.yml



Section 3:

Variables: Vars

Used if you have one value used more than once. as constant in programming.  
Or more than value that will be stored in one variable. as array in programming for example.

It can be used in single file or in directory roles, etc as before.. So we made under each role a directory for vars  
  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p3/hosts singlefile.yml

//In directories: we changed in main.yml under tasks and main.yml under vars

ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p3/hosts mproj.yml  
  
Scope of vars , Each role uses its own variable x for example, if it does not have x it will use global x, global x is the first one declared in the tree

ansible-playbook mproj.yml -e “fileb\_path = /root/fileB” //run ,-e means extra vars

No errors :)

ansible-playbook mproj.yml -e “file\_path = /root/fileA” //run ,remember we already did this var before in r1 :)  
//It will override on the r1 as it is stronger

Loops:

2 ways : first way using “with\_item” keyword and item as var,second way using “loop” keyword with item as var  
  
Way 1:   
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts loops1.yml  
  
Way 2:  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts loops2.yml  
  
//For custom variable name instead of item, also about gather facts  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts loops3.yml  
  
Conditionals:  
handlers were related to tasks , Here we are talking about conditions related to variables.

ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts conditionals1.yml -e “x=no”  
//you will find task skipped  
  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts conditionals1.yml -e “x=no”  
//task changed

Combining more than one condition → (both of them must happen)  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts conditionals2.yml  
  
  
Debug:  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts debug.yml  
  
  
Register:

Used to define variable from inside a task, Task done and it has an output, I want to take that output and declare it in a variable.  
ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts register1.yml  
  
  
,,So when can be used with vars or vars from facts or vars from registers.

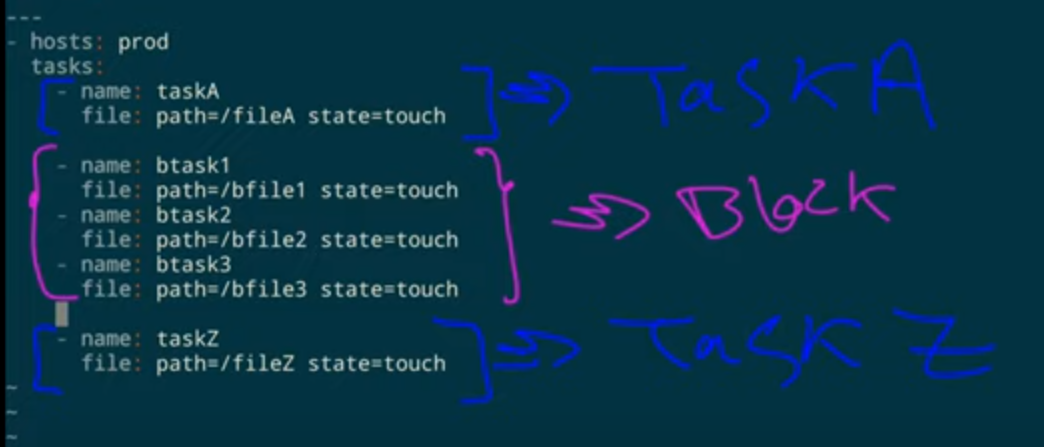
Lets talk more about when + vars from registers:

Create task a which creates an empty file, then check that empty file if it is empty or have a content, if there is content then delete it. If it is empty put inside it content.

ansible-playbook -i /Users/abdelrahmanmagdy/Documents/personal/ansible-p4/hosts register2.yml

– Blocks:

You take a group of tasks and group them together.  
Before:

  
After using block as below🙂

