Ansible is free software for deploying and managing systems, remote systems, servers, machines, etc.

First thing you need is an inventory. An inventory is a list of machines that you want to do things to. It can be as simple as a text file.

You can also create groups of machines in your inventory. You can put machines in multiple groups.

You can perform simple command line tools. For instance:

*$ ansible database -I inventory -m ping*

The database can be all, but for now I am just selecting a group. The -I option lets me know where the entry is. The -m option just lets me know that I am using the simply ping module.

Check out this next example:

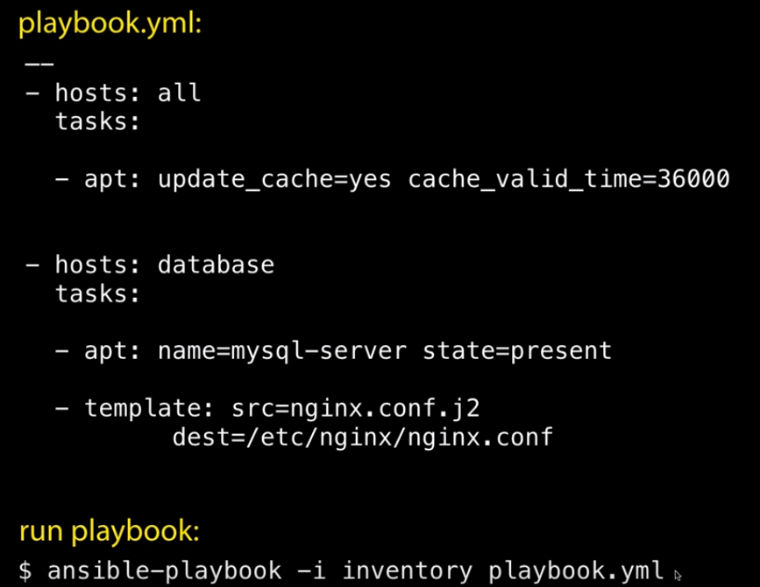
*$ ansible database -i inventory \ -m apt -a “name=msql-server state=present”*

It is slightly more complicated. Still using the same database group within the inventory but now it is using the apt module. The apt module just installs packages on linux servers. The -a is specifying arguments. We are installing MySQL Server and I want to make sure it is present.

There are many modules, dozens that perform various things. You very seldom have to write your own code, there is more than likely a module already written.

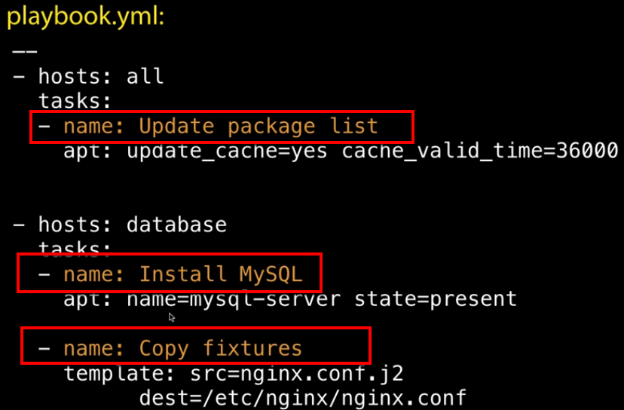
\*\*\*User Authentication\*\*\* - Assuming I will be able to SSH into my inventory. I will need to get up to speed on SSH in general.

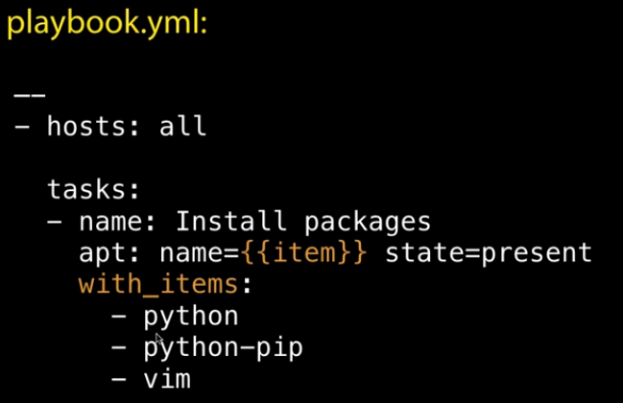
If I want to do something more complex than simple module commands, I will need playbook, which is associated with Yaml. Yaml is just a structured way of presenting some text. It is very comparable to a JSON file.

The playbook.yml file is having all the hosts check if there is an update. The valid time is just setting the time it checks, which is 10 hours.

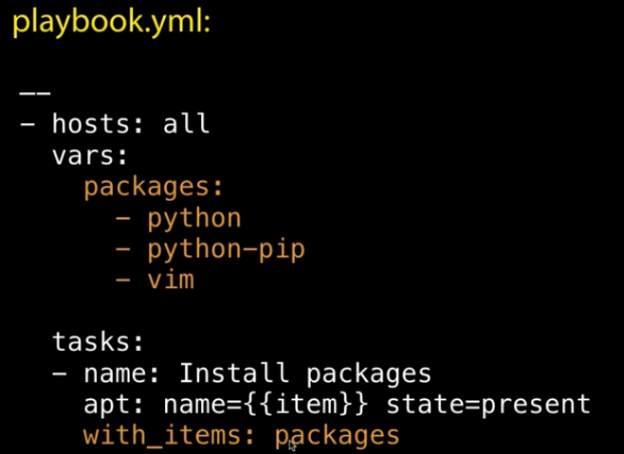
It also is having the database group install MySQL Server if they are not already installed.

Finally, we are copying a template from our source and placing it in a specific destination. The playbook is then ran from the inventory.

Another good habit that I need to follow is naming the tasks. This is good for documentation purposes as well as keeping it very clean/visual with exactly what is going on or intended with each command.

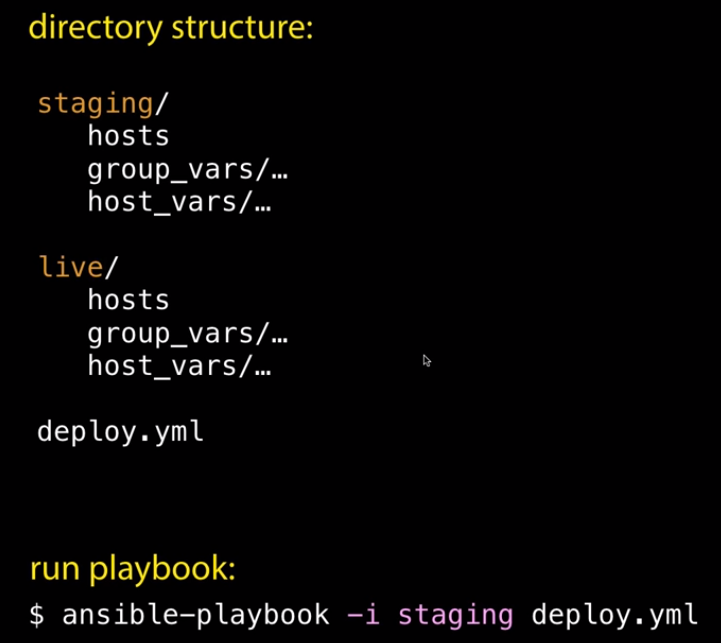


Here we have an example where we list out the variables. By doing so we eliminate the need to repeat the task for each item we want installed and can do so all at once.

We also can create variables. Once we write out what hosts this is for, we set up vars: and name *packages*.

Then we complete the tasks almost the same as above, except we name *packages* instead of itemizing the different things we want installed.

There are conventions as to how you should lay out your Ansible project.

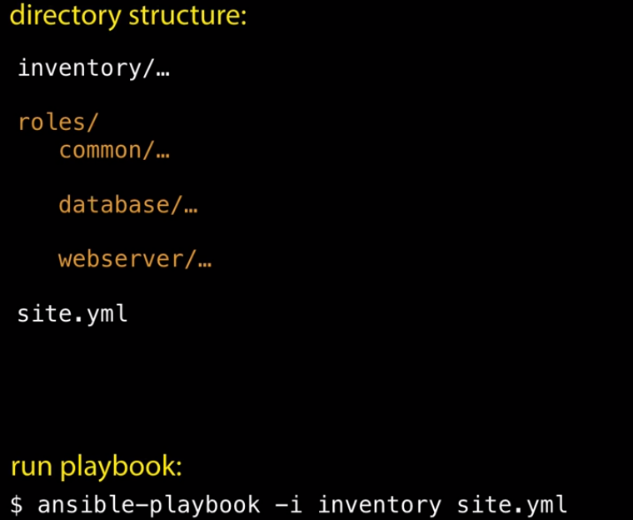


Ansible has something called Roles. This is how they can group things once the playbooks get large. It is a way to group things up by functionality.

For instance, if we want to group together a set of rules, configurations or apps that are only for our webservers, we can have roles for that. For webservers we have 2 roles here that we would like executed: common and webserver roles.

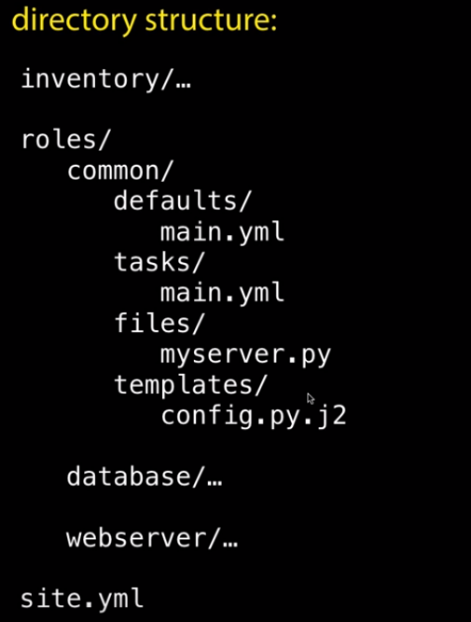
Then we can also have roles for our databases. In our example it shows database should have the common and mysql roles.

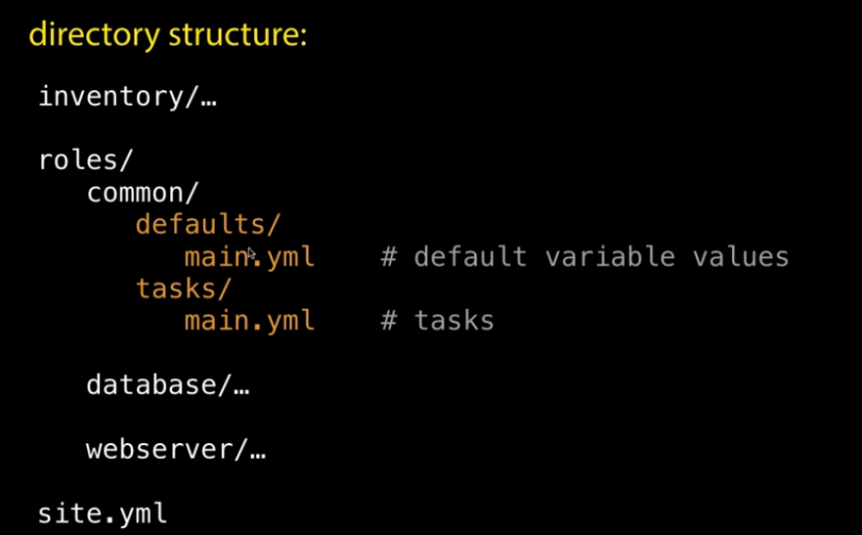
We can see that both the webservers and the database should have the common roles installed but they need to have their own respective roles as well.

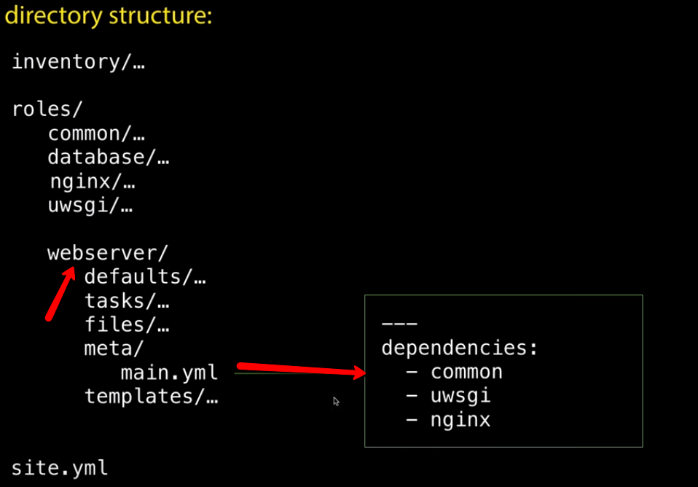
I need to create a roles directory in my inventory and specify the roles that I will be using with subdirectories.

Within the roles/sub-directories you will want to have tasks with a main.yml. This is going to be the tasks that we have it perform. i.e. Install this package, check for updates.

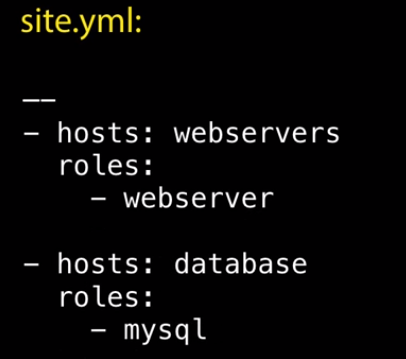
Other things might be files, templates we want to copy across. If we design our roles well, we can re-use them in other projects.





You can specify dependencies within roles. When you create roles, you can create a dependency that states, this role depends on these other roles being installed too.





If we include the dependencies within the roles, we can have it much cleaner and still perform the same thing.

This keeps everything neatly packaged up.

