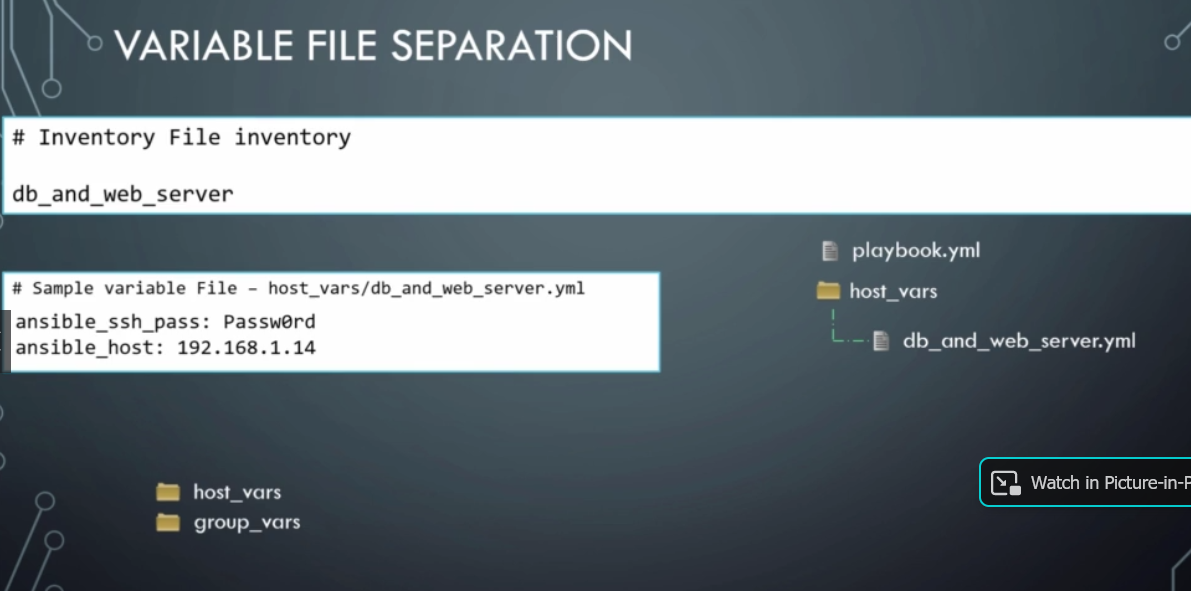
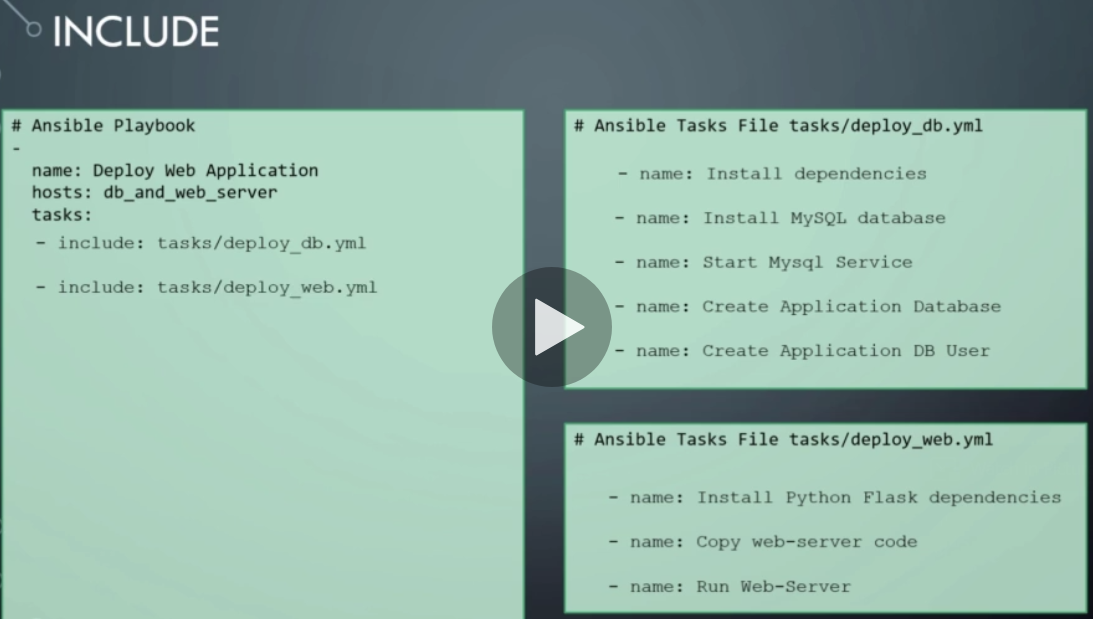
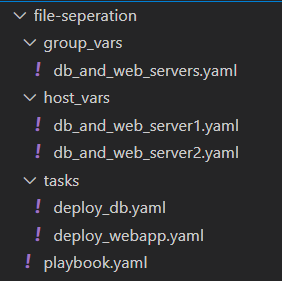
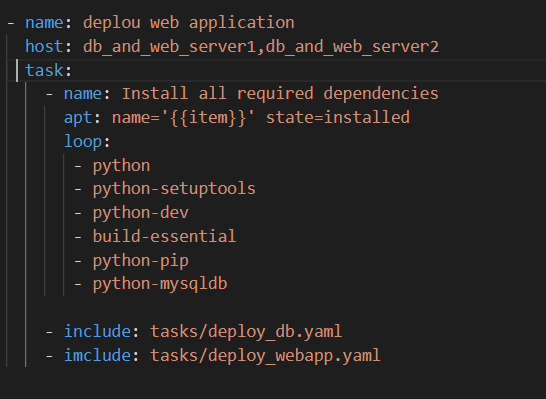
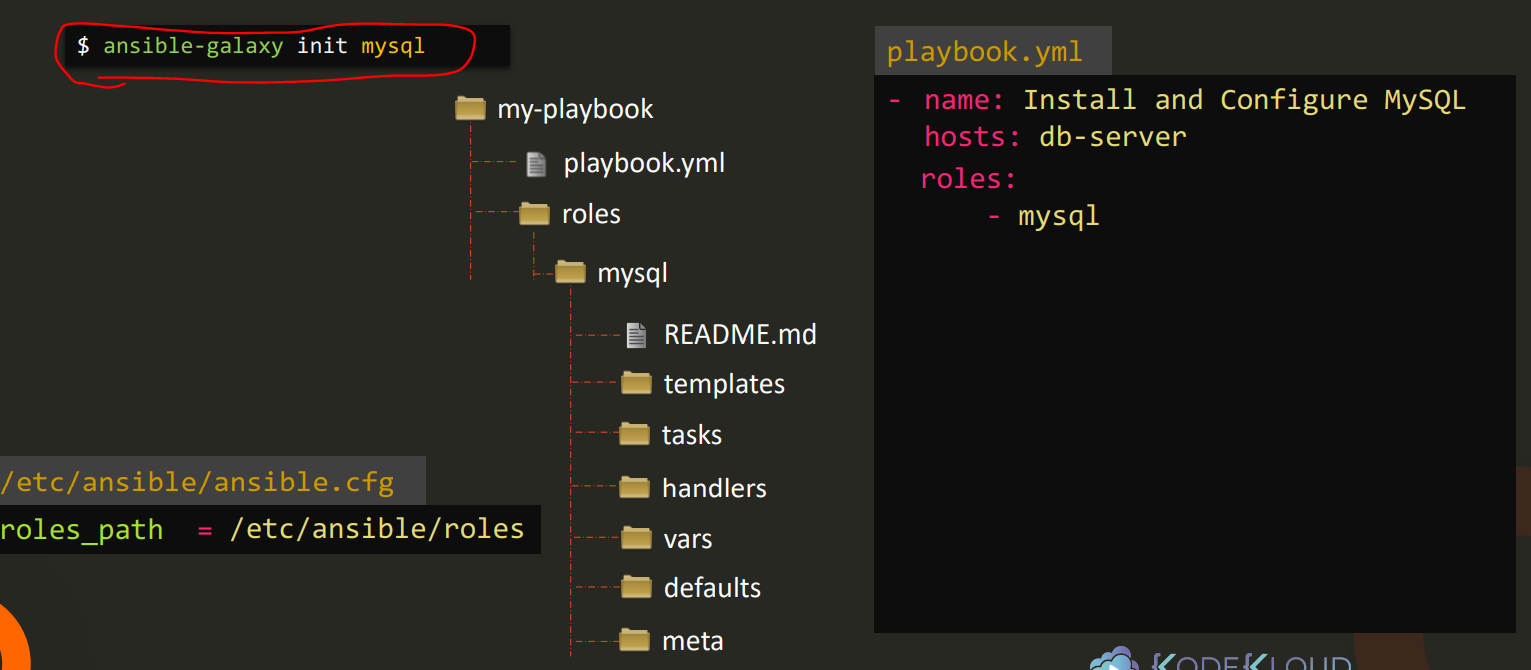
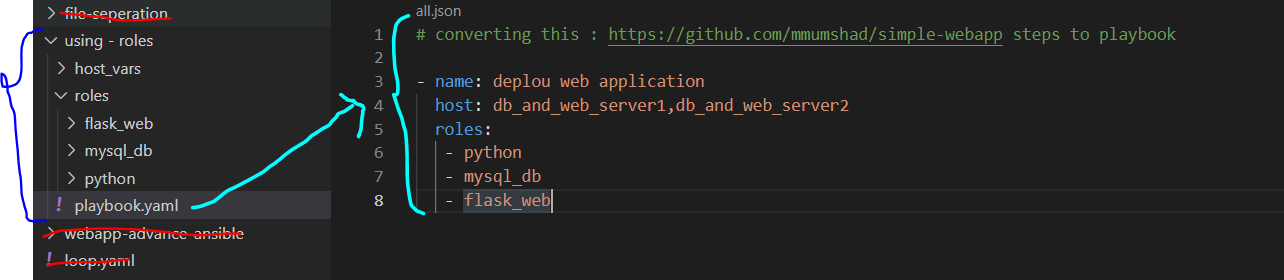
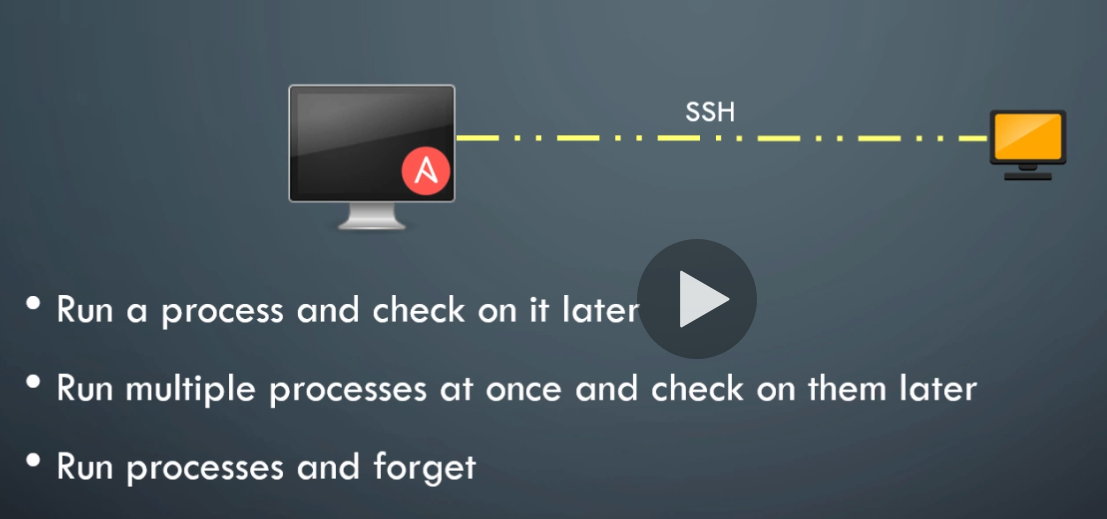
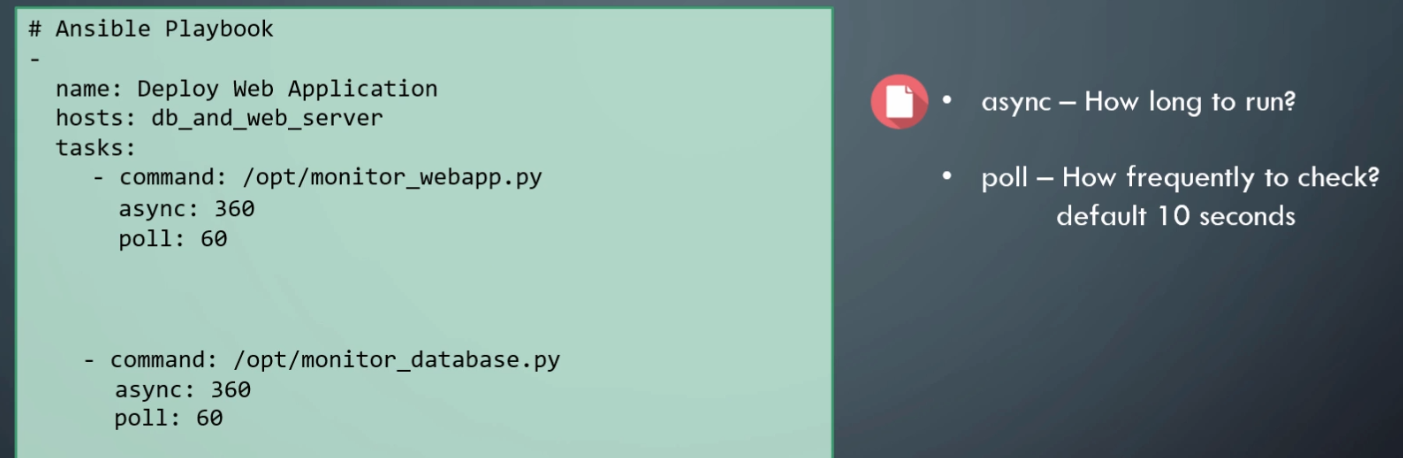
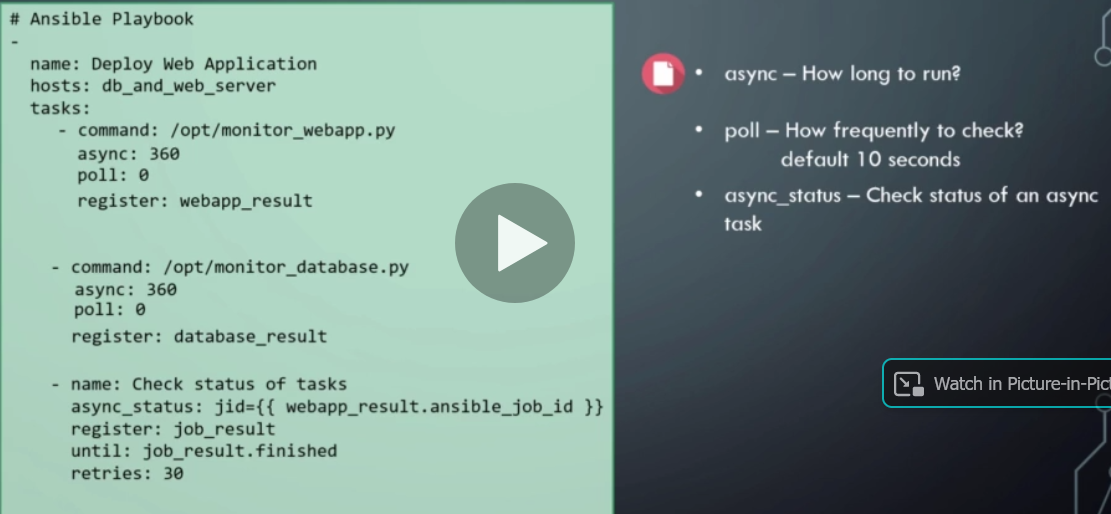
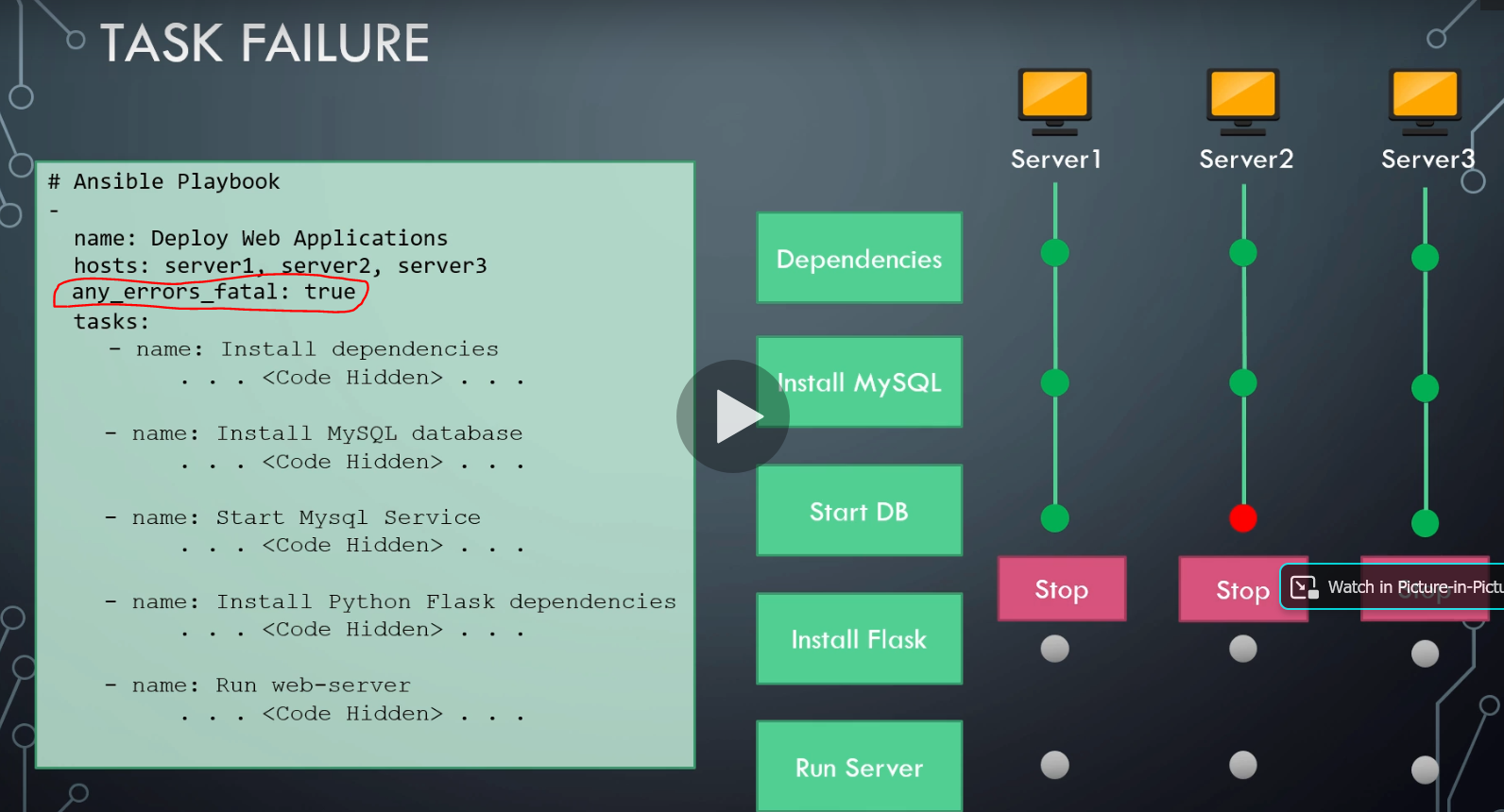
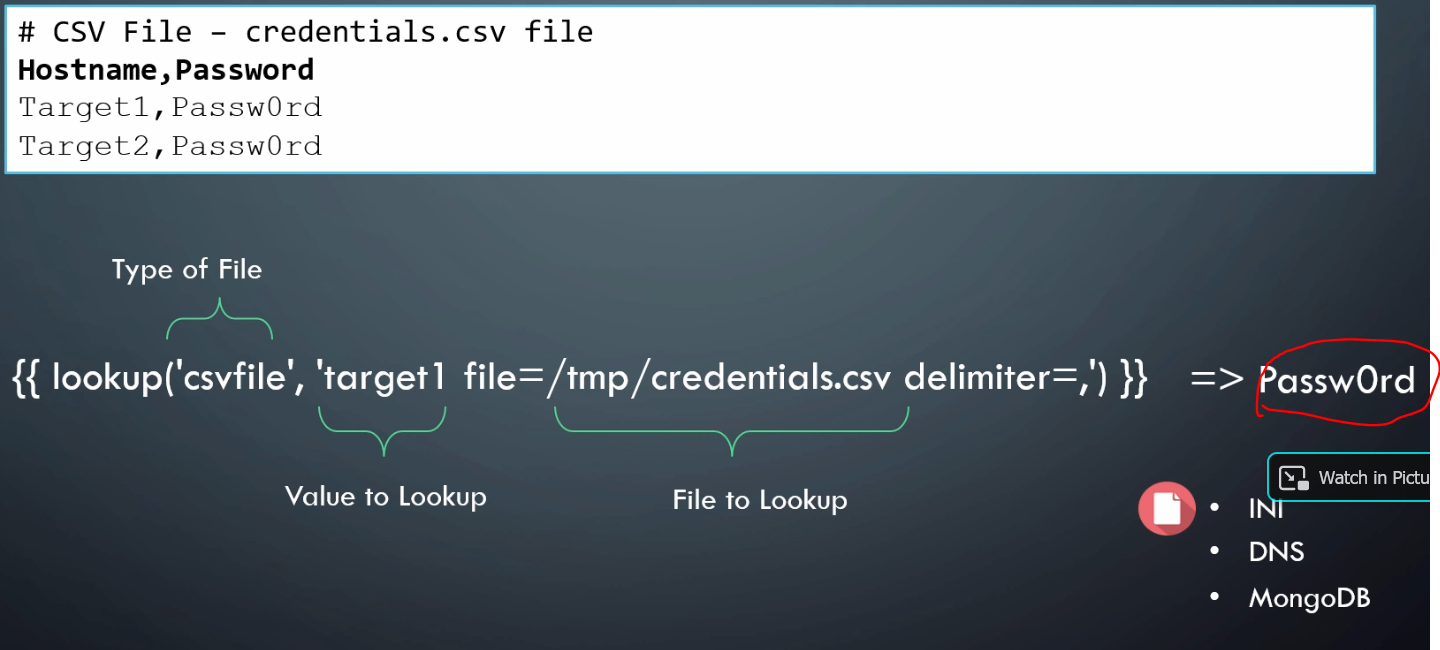
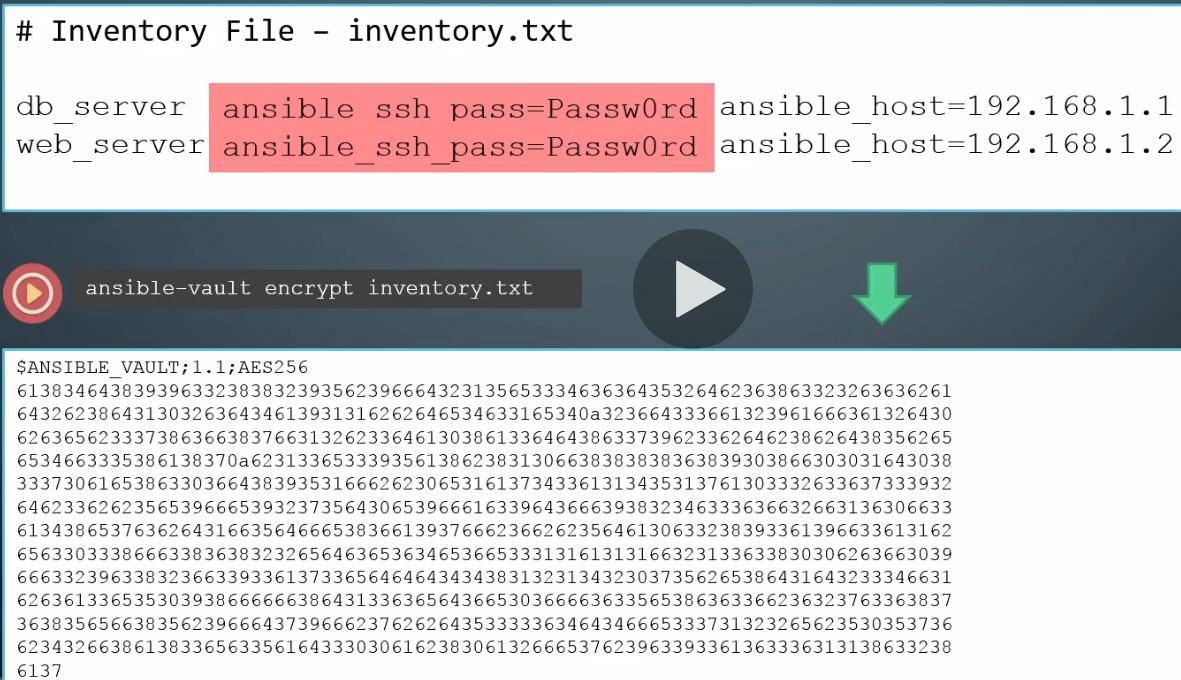
* **File separation :**
* define variable in separate file : Create a .yaml file with same name as ‘server name’ from inventory file and save all the variable inside it for that particular server, then when you run the ansible playbook, it will go to respective inventory file and will also use the .yaml for vars since it has the same name as server name in inventory file. (this .yaml file has to be under ‘host\_vars’ folder)
* 
* **Include:** we can create a ‘task’ folder and move all the task under that folder and use it in play book whenever required , which is reusable and best practice   
  
* So final setup would look something like this  
     
   set-up Playbook.yaml
* **Roles:** it modularize the work, so it can be easily reusable and readable !!
* Running $ ansible-galaxy init mysql will create the layout similar to this  
  
* We don’t have to use the $ ansible-galaxy init mysql cmd, we can create the folder structure ourselves , but if use the $ ansible-galaxy init mysql cmd then it makes it easier.
* The final layout  
  
* **Asynchronous actions :** All of the below can be achieve with async-actions

* When you make poll = 0 , ansible will run two task in parallel  
  
* **Strategy :**
* Strategy defines how the playbook is executed in ansible
* If you do not specify anything ansible will use default strategy which is linear strategy
* **Linear strategy** : all the task will be performed at a same time on all the servers
* **Free strategy** : all servers will perform task at different speed (servers do not depens on other servers to finish the task to move forward)
* **Batch** : you can specify how many servers you would like to process simultaneously (linear strategy process all servers together, but here you can select different number say 3)
* **Error Handling:**
* If ansible is performing task on 3 different servers, lets say one of the server task-4 failed , so ansible will move forward with remaining two and will finish the tasks , but if we do not want this we can use any\_error\_fatal directive like this , which will stop execution on all servers even if only one fails.  
  
* **Jinja2 templating :**
* We can use jinja template with different filters to get desire outputs
* Go to ansible documentation to get list of prebuild filters.
* **Lookup:**
* Lets say that the username and pass that we use in inventory file are stored in separate .csv file
* We can use lookup plugin to use this info in inventory file.  
  
* **Vault:**
* Storing credentials in plain text format is not a best practice
* Simply run this cmd to encrypt the inventory.txt file $ ansible-vault encrypt inventory.txt
* 
* $ ansible-playbook playbook.yaml -i inventory.txt -ask-vault-pass : But then run this cmd to run the playbook (since the inventory.txt is encrypted now)  
  $ ansible-playbook playbook.yaml -i inventory.txt -vault-password-file ~./vault\_pass.txt : if you run this cmd, it will look for vault\_pass.txt file for the password
* **Dynamic Inventory:**
* Up till now we show that we were storing server info in plain inventory.txt file (static inventory file)
* But In real time application, we would like to pull this info from CMDB or Cloud API , so we can use python scripted dynamic inventory file to do so.
* There are numerous dynamic inventory list examples are provided on ansible’s github site.  
  <https://github.com/ansible/ansible/tree/devel/contrib/inventory>
* **Custom modules:**
* We can make our own custom modules and use it inside ansible playbook
* **Plugins:**
* There are plenty pre-build plugins that ansible offers (like: action, connection, lookup, filter, strategy etc)
* We can create our own custom plugin as well.
* Lets say we want to create a filter to print out an average number.
  + First create average.py   
    
  + Running $ ANSIBLE\_FILTER)PLUGIN=/filter\_plugins will let ansible know where to find the avg filter.
* **Call back plugin** : it prints the info of every task while ansible executes them
* You can switch output from simple text to json format using below cmd.  
  