1. Create Ansible Role to launch 3 AWS EC2 Instances.
2. Create Ansible Role to configure Docker over those instances.
3. Create Role to configure K8S Master, K8S Worker Nodes on the above created EC2 Instances using kubeadm.

# Step-1 : Creating three Ansible Roles

* Create one workspace, let's say “aws-kubernetes-ansible”.
* Go inside this workspace & create one folder called “roles”. Now go inside this folder & run these below mentioned three commands.

Graphical user interface, text

Description automatically generated

# Step-2 :Creating AWS Key-pair & putting it in the Workspace

* Go to AWS => EC2 => Key-pair & there create one key pair — let's say “ansible.pem”. Then download the key in your VM workspace and put it under aws-kubernetes-ansible folder . This folder will be deleted from github and will not check in for security purpose
* Run the following command

chmod 400 ansible.pem

* When Ansible will login to AWS instances to setup K8s via SSH, then it needs the private key file

# Step-3 : Creating Ansible Vault to store the AWS Credentials

* Run the following command from the workspace folder

ansible-vault create cred.yml

* It will ask to provide one vault password & then it will open the VI editor on Linux, create two variables in this file & put your AWS access key & secret key as values. This file will be intentionally deleted from github

access\_key: ABCDEFGHIJK

secret\_key: abcdefghijk12345

* Run the following command as shown in the below screen shot to verify the installation so far

Graphical user interface, text

Description automatically generated

# Step-4 : Setting up Ansible Configuration File

* In Ansible we have two kinds of configuration file — Global & Local. We gonna create one local configuration file inside “aws-ansible” folder & whatever Ansible commands we want to run in future we will run on this folder. Because then only Ansible will be able to read this Local configuration file & can work accordingly.
* Create a file ansible.cfg [ look at the contents in github] inside the aws-kubernetes-ansible folder
* “private\_key\_file” which signifies to the aws key pair. When Ansible gonna login to AWS instances to setup K8s via SSH, then it needs the private key file. Also the default remote user of EC2 Instance is “ec2-user”

Text

Description automatically generated

**Text

Description automatically generated**

# Step-5 : Writing Code for ec2 Role

* **Go inside the folder “aws-kubernetes-ansible /roles/ec2/tasks/” & start editing the “main.yml” file.**In this file write the below mentioned code

Graphical user interface, application

Description automatically generated

* Next I used “ec2\_group” module to create Security Group on AWS. Although we can create one strong Security Group for our Instances, but to make things simple I allowed ingress & egress in all the ports. But in real scenario we never do this.
* Next I used “ec2” module to launch instance on AWS, & here all the parameters are known to us. Only I want to talk about two parameters — first is “register” which will store all the Metadata in a variable called “ec2” so that in future we can parse the required information from it. Second is “loop” which again using one variable which contains one list. Next using “item” keyword we are calling the list values one after another. This gonna run ec2 module 3 times with different instance tags, which finally will launch 3 instances.
* Next I used “add\_host” module which has the capability to create one dynamic inventory while running the playbook. Under this module I used “hostname” keyword to tell the values to store in the dynamic host group. Here I used that “ec2” variable & do the JSON parsing to find the public ip of 1st instance.
* We have created two host groups — ec2\_master & ec2\_slave & 1st instance belong to “ec2\_master” & 2nd, 3rd instance belong to “ec2\_slave” host group
* Finally “Wait\_for” module to hold the palybook for few seconds till all the node’s SSH service started
* Open the “aws-kubernetes-ansible/roles/ec2/vars/main.yml” file & store all the variables that we mentioned on the “task/main.yml” file along with their respective values. The one which I have used is shown below
* Graphical user interface, text, chat or text message

  Description automatically generated

# Step-6 : Writing Code for k8s\_master Role

* Similarly like previous time open the “aws-kubernetes-ansible/roles/k8s\_master/tasks/main.yml” file where the respective tasks for install the kubeadm command on master node and kubernete cluster set up is mentioned
* Here we need to install kubeadm program on our master node to setup K8s cluster. So, for that I'm adding the yum repository provided by K8s community. Here as I'm using AWS Linux 2 for all the instances so we don't need to configure repository for docker cli.
* Next using “package” module we are installing “Docker”, “Kubeadm” & “iproute-tc” package on our Master Instance.
* Next I used “service” module to start the docker & kubelet service. Here again I used the loop on the list called “service\_names” to run the same module twice.
* Next I used “command” module to run one kubeadm command which will pull all the Docker Images required to run Kubernetes Cluster. Here in Ansible we don't have any module to run “kubeadm” command, that's why I'm using “command” module.
* Next we need to change our Docker default cgroup to “systemd”, otherwise kubeadm won't be able to setup K8s cluster. To do that at first using “copy” module we are creating one file “/etc/docker/daemon.json” & putting some content in it. Next again using “service” module we are restarting docker to change the cgroup.
* Next using “command” module we are initializing the cluster & then using “shell” module we are setting up “kubectl” command on our Master Node.
* Next using “command” module I deployed Flannel on the Kubernetes Cluster so that it create the overlay network setup.
* Also the 2nd “command” module is used to get the token for the slave node to join the cluster. Using “register” I stored the output of 2nd “command” module in a variable called “token”. Now this token variable contain the command that we need to run on slave node, so that it joins the master node.
* Lastly I used “shell” module to clean the buffer cache on my master node, because while doing the setup it gonna create lots of temporary data on RAM.
* Open the “aws-ansible/roles/k8s\_master/vars/main.yml” file & store the variable “service\_name” with values in list

Text

Description automatically generated

# Step-7 : Writing Code for k8s\_Slave Role

* Similarly like previously open the “aws-kubernetes-ansible/roles/k8s\_slave/tasks/main.yml” file & put the tasks for configuring the slave.
* Till docker service restart task, this file is exactly same like the previous file of “k8s\_master” role. On slave node we don't need to initialize the cluster & also we don't need to setup kubectl. Rest things we need to do because in slave node also we need “kubeadm” command & Docker as container engine.
* Next I used “copy” module to create one configuration file called “/etc/sysctl.d/k8s.conf” which will allow the slave to enabled certain networking rules. Next to enable the rules we need to reload the “sysctl” & for that I used “command” module.
* If you can remember we used “token” variable on our previous role to store the token command for the slave to join the cluster. Now each role has their own separate namespaces to store the variables. So we need to go to the namespace of previous hostgroup that we created dynamically.
* For that we use “hostvars” keyword & inside it we call the “ec2\_master” hostgroup. Next in this host group we can have multiple hosts (nodes). To pick the 1st host we use “[0]” option. That means finally “hostvars[groups[‘ec2\_master’][0]]” option is calling the namespace of the master node.
* Next using “[‘token’][‘stdout’]” we just parsed the command that we can use in slave to join the master node.
* Here again we need to create the same “vars/main.yml” file that we created on “k8s\_master” role.

# Step-8 : Create the set up file

Now it's finally the time to create the “setup.yml” file which we gonna run to create this entire infrastructure on AWS. Remember one thing that we need to create this file inside the folder “aws-kubernetes-ansible”

Graphical user interface, text, application, chat or text message

Description automatically generated

* Here as you can see, we are running the first “ec2” role on our localhost because it gonna contact to AWS API from our localhost. Also using “vars\_files” I included the “cred.yml” file in this task so that “ec2” role can access it.

# Step-8 : Run the set up file

# Run the following command to create the instance and set up the Kubernetes cluster.It will ask for the the vault password which you have created in step 3

# ansible-playbook setup.yml --ask-vault-pass