1. Creating EKS cluster with one master and two workers.

Install and configure Terraform and awscli(This one is optional), i.e. it is not a simple installation, because you have to set it up for multi-user environments. In order to do so, you have to consider S3 buckets(for different Environments) for keeping tfstate files of Terraform and you have to create tables in DynamoDB (For each S3 bucket which contains tfstate file of specific environment) to support lock mechanism. But since it is beyond the scope of your request, I dealt with tfstate files inside each directory and I consider that only one person from a machine is responsible for issuing Terraform commands. i.e. some mechanism like having Jenkins CI/CD machine which is the only node that can issue Terraform commands.

For Security reasons, it is better to install awscli via pip then issue aws configure to store your key and its secret in .aws/credentials file. After that you do not need to address them in Terraform, since Terraform will look into default place.

The best solution is to have different Environments in Terraform:

* cluster
* control-plane
* network
* And main directory for main.tf and variables and so forth

In order to access “control-plane” from one IP address, as you mentioned, I set it to my IP:

37.228.242.179

By the way, if you want some dynamic mechanism to automatically find your IP, I wrote “workstation-external-ip.tf” file for this purpose.

First for initializing and in order to download your provider’s module, you have to issue bellow command on main directory (Terraform Directory):

terraform init

And if you have S3 and DynamoDB as mentioned above, you have to issue bellow command:

terraform init -backend-config "key=<folder>/terraform.tfstate"

BTW, you have to specify your DynamoDB table and also your Terraform backend which is S3 in main.tf or any file in your Terraform plus your terraform.tfvars in each directory. But in our case, we do not deal with it.In our case , since I want to have a dynamic Terraform system, I did not ser terraform.tfvars, and consequently terraform will ask you about each variables that you defined in variable.tf file.

After applying terraform init, you need to issue:

terraform apply in main directory, and specify values for variables that you defined and terraform will ask you about them, then have a couple of coffee cups, and your cluster will be ready with one master and one worker.

See output.tf file contents in each directory to find your needed info, like pem files.

2. Deploy to K8S cluster:

You have to install kubectl in your machine in which you want to manage your K8S cluster.

Now take a look at output.tf file inside cluster directory, and copy kubeconfig

into your kubectl conf file which resides in $HOME/.kube/config file.

Or you can set it anywhere via KUBECONFIG environment variable.

Now you can see the number of nodes and pods and etc... via kubectl get <whatever-you-want>.

Now you can enable ingress (Nginx Ingress) in your cluster:

I personaly prefer to first download yml files in proper directory then issue kubectl apply -f, so I will create Kubernetes/Work folder then cd into it and get Nginx Ingress that you mentioned:

wget -c “https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/mandatory.yaml”

Then issue “kubectl apply -f mandatory.yaml”.

And create service with:

wget -c “<https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/provider/aws/service-nlb.yaml>”

Then : kubectl apply -f service-nlb.yaml

BTW, you can concat both aforementioned files into one and apply them.

Then create bellow Deploment, SVC and ingress that supports ModSecurity:

You have to just issue bellow command inside Juiceshop folder which resies in Work Folder:

kubectl apply -f \*

That is it, and you can see your system is running.

BTW, for sake of security, it is better to consider one PRIVATE Docker registry.

Anyway, for sake of tests before deploy, also it is better to have livenessProbe and readinessProbe as well.