**Set up AWS Load Balancer and Ingress**

# Create a cluster and nodegroup with the help of terraform/eksctl

# Use aws eks update-kube-config command to connect to the cluster

Command - aws eks update kubeconfig –region <region> --name <cluster-name>

Optional: Check the connectivity through - kubectl get nodes command.

# Now we can go ahead with installing AWS Load Balancer controller

Follow this documentation to install AWS Load Balancer controller -

<https://docs.aws.amazon.com/eks/latest/userguide/aws-load-balancer-controller.html>

# Create a OIDC provider if not present already.

Command - eksctl utils associate-iam-oidc-provider --cluster <cluster-name> --approve

Refer the documentation to create a policy.

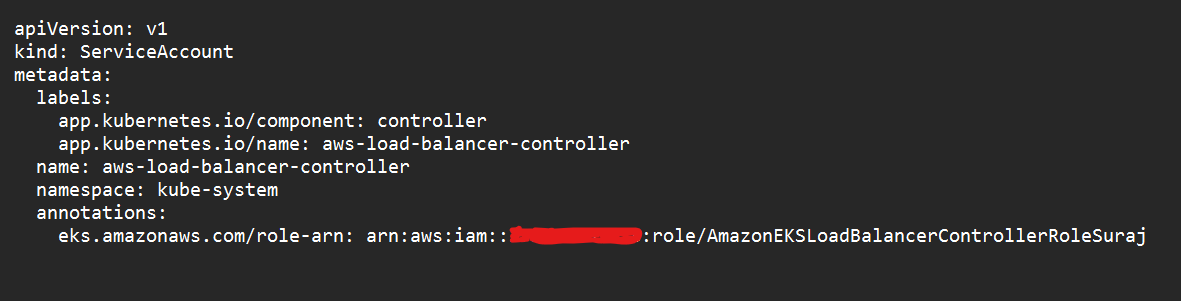
Now create role using the the above created policy.

AWS Console -> IAM -> Choose Trusted Entity type as Web Identity -> Provider -> choose OIDC

-> Audience (sts.amazonaws.com) -> Next -> Choose the above created policy -> Create

Now copy the role arn from the console and create service account using this YAML file.

---- YAML file screenshot ---



Install the aws load balancer controller using HELM

Command - helm upgrade --install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=Suraj-dev-ekscluster --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller

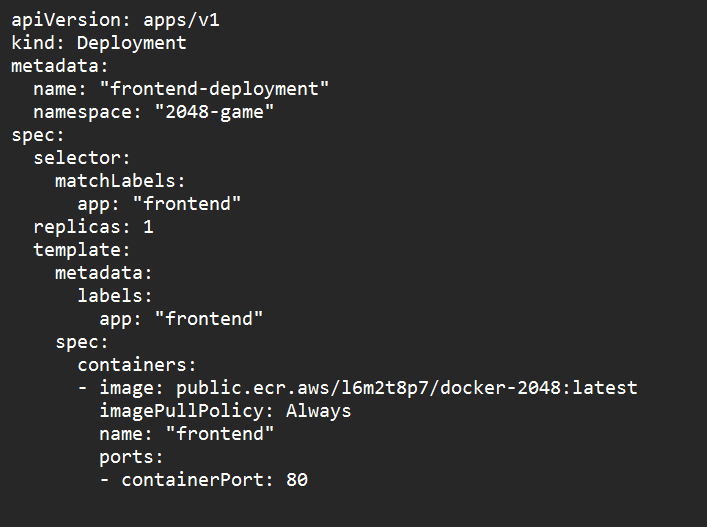
Check status by - kubectl get deploy -n kube-system

Now you can deploy your deployments and services

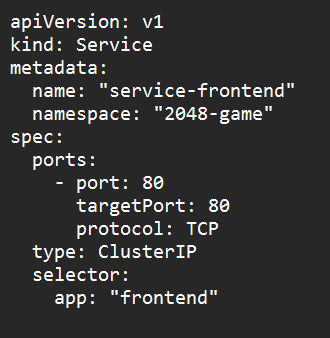
After the services and deployment are created, you can apply ingress resource which will create the load balancer in aws console.

Use these sample codes if required

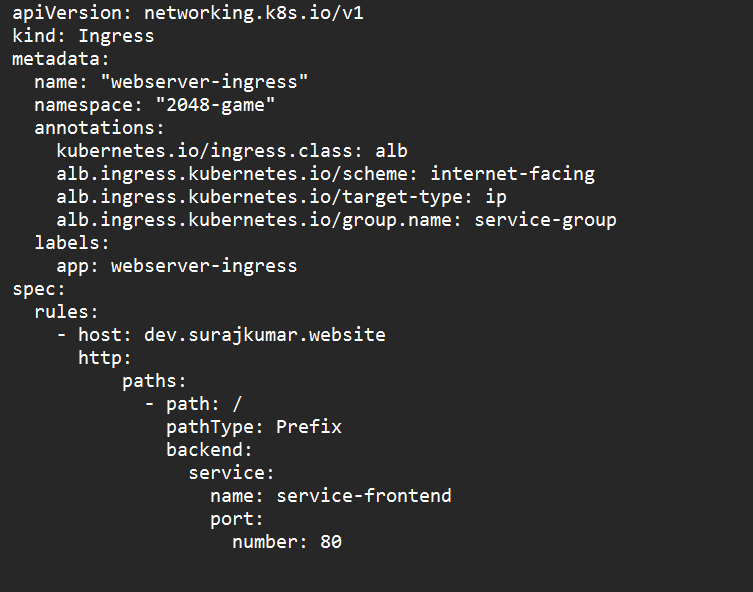
Deployment file using the 2048 game public image



Service file



Ingress file for 2048-game namespace



You can use any host of your choice

Use the group.name annotation to launch a single alb and to connect multiple services across namespaces if required.

If you need all the deployments in a single namespace you can omit the above annotation.

**Launching prometheus and Grafana.**

* In order to launch Prometheus, you need an ebs-csi driver. In order to add this, use the add-ons option in the console.
* However, this will not work as your node does not have the necessary permissions yet.
* In order to provide the permission to nodes, modify the IAM role and add AmazonEBSCSIDriverPolicy to the node IAM role.

Now follow the below commands to install prometheus in the EKS Cluster.

Command:

* kubectl get --raw /metrics
* kubectl create namespace prometheus
* helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>
* helm upgrade -i prometheus prometheus-community/prometheus --namespace prometheus --set alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2"

Check if installation is successful:

* kubectl get pods -n prometheus

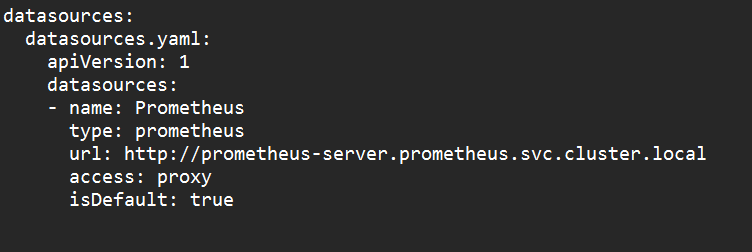
You can port-forward this using below command to your localhost

* kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090

**Now install Grafana**

Create a namespace for grafana

* kubectl create namespace grafana
* helm install grafana grafana/grafana --namespace grafana --set persistence.storageClassName="gp2" --set persistence.enabled=true --set adminPassword='surajteleglobal' --values <grafana.yaml>



* Use the ingress resource to access grafana publicly and set up using previous file in documentation.

You can add a dashboard for the node by - Dashboard -> Import -> 1860 -> Load

Continue using Grafana and set up monitoring using the application use case.