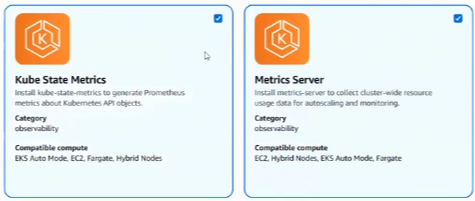
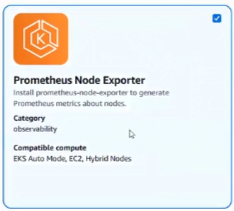
AddOns, Control Plane, Worker Node

......CSI Driver need IAM role mandatory so we need manual installation steps to work with it, so we wont be selecting it at GUI level



>> Create control plane in EKS=”demo-radical-cluster”

>> Start jump host

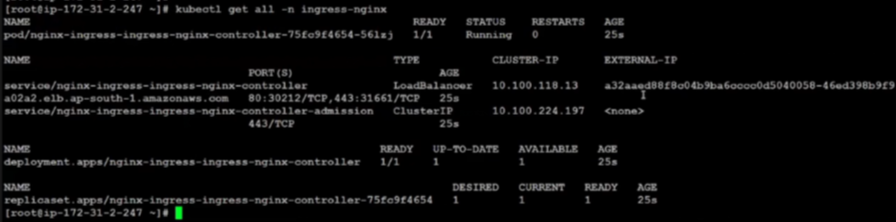
|  |
| --- |
| Nginx ingress controller ……Install using helm by default you will get classic or NLB but you will not get ALB in CHATGPT; helm2 vs helm3 in CHATGPT |

>> Once active, check control plane state, and go for node creation >> Goto Compute >> Node group name=”demo-radical-cluster-ng” >> Select worker role in IAM

>> Node group scaling config >> min=3, desired=3, max=5 >> Next >> Create

>> Login jump-host >> Install helm, update repo of [github.kubernetes.io](http://github.kubernetes.io), add repo & update repository >> Install nginx-ingress

* Kubectl get all -n -ingress-nginx



>> Goto AWS >> check LB provisioning >> Once active >> Add hosted zone and create LB alias to DNS

>> Create a test-ingress.yaml file in jump-host where the cluster is connected and deployed to the cluster with below 2 lines.

* Cat > test-ingress.yaml …Create one file that will store deployment, service, ingress
* Kubectl apply -f test-ingress.yaml

Verify application running of the hello-world in default namespace:

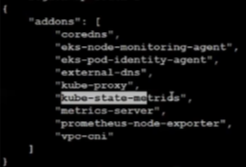
* Kubectl get ing; kubectl get svc; kubectl get deploy; kubectl get po;

Try pinging the Load balancer or use external ip [www.cloudcortex.in](http://www.cloudcortex.in) or hit curl http://<External-ip>

ADD-ONS from AWS EKS:

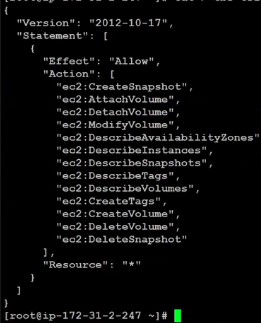
>> Check what all add-ons are installed in cluster

* Aws eks list-addons –cluster-name demo-radical-cluster –region ap-south-1



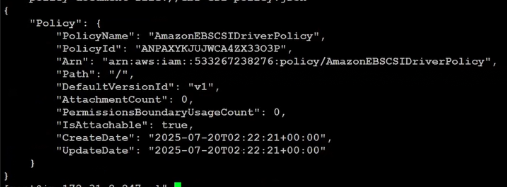
>> Install EBS CSI Driver on Kubernetes manually

* Cat > ebs-csi-policy.json ……Add policy document

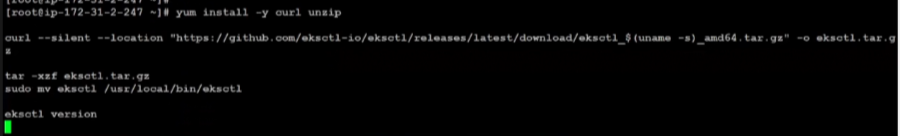


>> Create policy

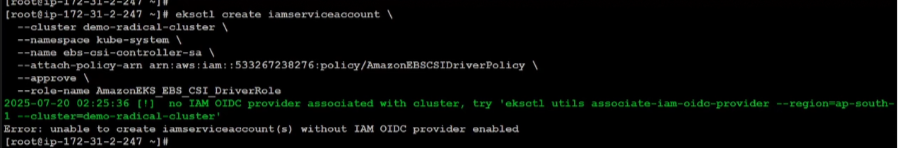
* Aws iam create-policy –policy-name AmazonEBSCSIDriverPolicy –policy-document file://ebs-csi-policy.json



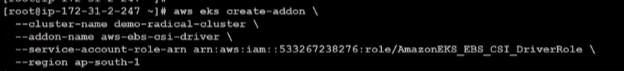
>> EKS ctl utility required here: ……ctl is pre-requisite to create ebs-csi role tagging to ebs-csi policy

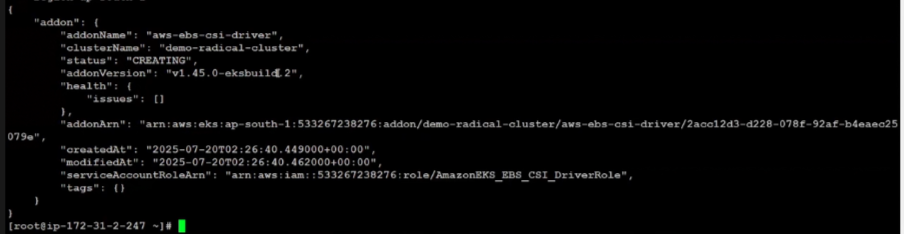


>> Create IAM role:



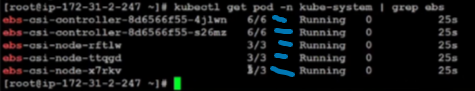
>> Create EBS CSI add-on:





>> EBS CSI installation verification in new POD:

* Kubectl get pod -n kube-system|grep ebs
* Aws eks list-addons –cluster-name demo-radical-cluster –region ap-south-1



Prometheus and Grafana installation steps via Helm

>> Create storage class file: ……Monitoring always needs more space

* Cat > ebs-sc-storageclass.yaml

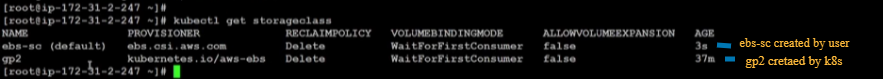


>> Apply to cluster:

* Kubectl apply -f ebs-sc-storageclass.yaml

>> Verify storage class:

* Kubectl get storageclass



>> Here we are managing storage for prometheus and grafana

>> Create namespace:

* Kubectl create ns monitoring

>> Add repo and update



>> Create 2 yaml files ……Helm uses them as pre-requisite to installation of Prometheus and Grafana as they store PersistentVolume information

* Cat > prometheus-values.yaml
* Cat > grafana-values.yaml



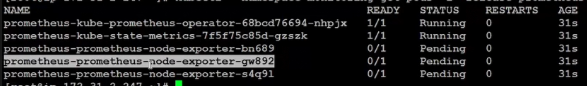
>> Install Prometheus

* Helm install prometheus prometheus-community/kube-prometheus-stack -n monitoring -f prometheus-values.yaml



>> Verify installation on k8s

* Kubectl –namespace monitoring get pods -l “release=prometheus”



Wait for state to be running, then execute Grafana installation

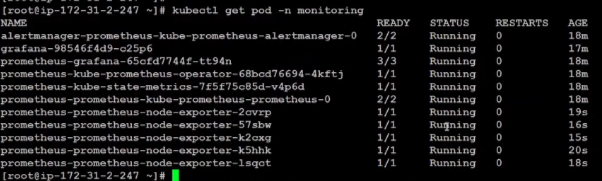
* Helm install grafana grafana/grafana -n monitoring -f grafana-values.yaml

>> There are chances that the WN size of all 3 nodes is insufficient even when we install both prometheus and grafana step by step. ……Try increasing WN to 5 which is the industry standard. We can use cluster auto-scaler and karpenter to automate that step not recommended at study level. Share POD, NODE and installation stats using below command to troubleshoot pending stuff using:

* Kubectl get pod -n monitoring
* Kubectl describe pod grafana-<nameid> -n monitoring
* Kubectl get pvc -n monitoring ……lecture error
* Kubectl top nodes
* Kubectl get node
* Kubectl describe node <ip>
* Kubectl get daemonset -n monitoring
* Kubectl get storageclass
* Kubectl get pods -n kube-system|grep ebs
* Kubectl delete pod grafana-<nameid> -n monitoring
* **Helm uninstall grafana -n monitoring …contingency plan**
* **Helm list -n monitoring …contingency plan**
* **Helm uninstall prometheus -n monitoring …contingency plan**

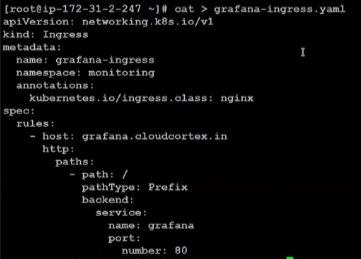
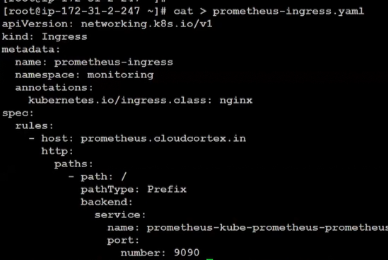
|  |
| --- |
| >>OIDC error is critical. If you face this error install OIDC and bounce ebs-csi-controller   * Eksctl utils associate-iam-oidc-provider –region ap-south-1 –cluster demo-radical-cluster –approve      * Kubectl rollout restart deployment ebs-csi-controller -n kube-system   >> Verify the pending.   * Kubectl get pvc -n monitoring * Kubectl get storageclass ebs-sc -o yaml ……Describes if pvc is healthy. Delete and install prometheus and grafana if still pending |

|  |
| --- |
| >> Lecture error was fixed by 2 separate steps for grafana and prometheus.  >> grafana error was fixed by adding trust-policy:   * Cat > trust-policy.json ……IAM role trust policy was pointing to OIDC provider which was replaced * Aws iam update-assume-role-policy –role-name AmazonEKS\_EBS\_CSI\_DriverRole –policy-document file://trust-policy.json * Aws iam get-role –role-name AmazonEKS\_EBS\_CSI\_DriverRole –query ‘Role.AssumeRolePolicyDocument’ ……check role status for OIDC * Kubectl rollout restart deployment ebs-csi-controller -n kube-system ……moves grafana to running 1/1   >> prometheus error for 5 POD was fixed by updating the daemonset port from 9100 to 9200   * Kubectl get daemonset prometheus-prometheus-node-exporter -n monitoring -o yaml|grep A20 affinity      * Kubectl rollout restart deployment ebs-csi-controller -n kube-system …Updates status to crashbackloop and error |



Once everything in POD comes up for ns monitoring.

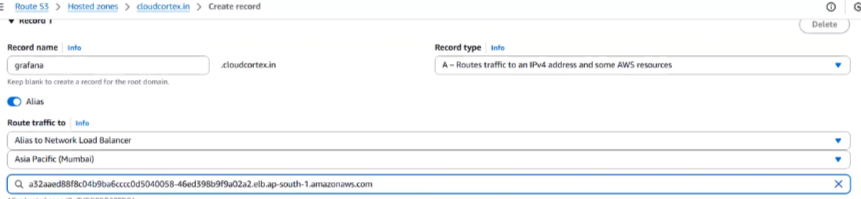
* Cat > prometheus-ingress.yaml
* Cat > grafana-ingress.yaml



* Kubectl get ing -n monitoring



>> Point in Route53 for grafana and prometheus as alias for accessing with cloudcortex on browser



>> <http://prometheus.cloudcortex.in> , <http://grafana.cloudcortex.in>

>> As Grafana will try to reach prometheus, update prometheus URL in grafana



>> Save & Test

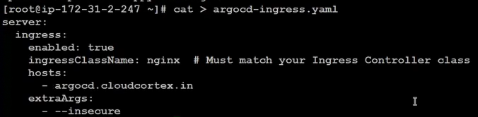
|  |
| --- |
| URL looks like above. In URL, 1st blue line is the servicename of port 9090, 2nd blue line is namespace, 3rd namespace is fixed for local service cluster |

>> Then add dashboard from [grafana.com](http://grafana.com) as mentioned on [grafana.cloudcortex.in](http://grafana.cloudcortex.in) and explore dashboards(K8S dashboard normally used)

>> Search prometheus query in CHATGPT

ArgoCD

* Kubectl create ns argocd
* Helm repo add argo <https://argoproj.github.io/argo-helm>
* Helm repo update
* Cat > argocd-ingress.yaml



* Helm install argocd argocd/argo-cd -n argocd -f argocd-ingress.yaml

>> Update Route53 to use argocd, update it in kubeconfig as per the attachment

