Dasmeta task:

Create an EKS Cluster with Terraform Module
 - 2 spot nodes
 - node type as minimal as possible
Deploy an app, any app:
 - with Terraform using helm_release resource/module
Direct App logs and metrics (CPU / memory / ...) to Cloudwatch (use adot for metrics)

Deploy Prometheus and Grafana

Create a dashboard for the app
 - in Cloudwatch
 - in Grafana
Setup Alerting to Slack
Use ALB Controller
 - expose the App under public domain
Configure an external health check for the domain
Test the load,
Document the setup.

Creating the EKS Cluster with Terraform module.

Directory: **dasmeta-task**

To create the AWS EKS Cluster, I used this module:
https://github.com/terraform-aws-modules/terraform-aws-eks
And I chose the **EKS Managed Node Group** variant.
I added two attributes from list of available attributes:

- **authentication_mode** set to API_AND_CONFIG_MAP [to make my life easier]
- **iam_role_additional_policies** with **CloudWatchAgentServerPolicy** [inside of the eks managed node group block, this is needed later on for CloudWatch]

- **endpoint_public_access** set to true: This is not a best practice but for simplicity and time saving I put it like this.The suggested alternative here is a Bastion/VPN.

Kubernetes version is set to 1.31 since I learned the hard way that ADOT addon does not support 1.33.
This directory also creates the VPC Network and Subnets inside of it for simplicity, I went with this set up.
VPC:
Availability zones: eu-north-1a, eu-north-1b
Private subnets: 2
Public subnets: 2
EKS:
Instance type: t3.medium [this is far from smallest, but there is an issue with preemption [when the helm app was being installed there was an error that said no nodes available for preemption] which is fixed by choosing a stronger instance]
Capacity type: SPOT

This directory also places the state file into an S3 Bucket which was prebuilt (not with terraform to avoid the situation when a state file is placed on a bucket which is provisioned by that same state file). Steps for creating and attaching policies to that S3 Bucket can be found here:
/dasmeta-task/bucket-config/s3-bucket-for-state.md

There are also outputs defined which will be read by remote state resource later on [for helm, if i go with that solution].
The VPCs and EKS will [and a bunch of underlying resources(62)] will take about 12 minutes to create (on my laptop).
After that we can immediately add the EKS Cluster to our Local context [~/.kube/config] and monitor/check/port-forward from k9s.

```
aws eks update-kubeconfig --region eu-north-1 --name dasmeta
```

Deploy a simple app to K8s through Terraform resource helm_release.

One important thing I learned for this part
- remote state data resource usage and configuration

- helm_release resource

Directory: **helm-release-app** and **helm-v2**

helm-v2

This was an attempt to change the basic app I tried the first time, I tried to redo this step from last time, just to make sure I did not miss anything, but this time instead of just nginx I did an nginx with my custom web page and a few other improvements such as exposing metric to **/metrics** path, health to **/health** [done through AI].

helm-release-app

This was the first attempt to deploy an app through TF helm_release and I kind of forked/reused this example: https://developer.hashicorp.com/terraform/tutorials/kubernetes/helm-provider

Direct App Logs and Metrics to Cloudwatch, use ADOT.

Direct App logs and metrics (CPU / memory / ...) to Cloudwatch (use adot for metrics)

I know this is not a good thing to write like this when I am being interviewed for a job, but I have tried (eventually failed) so much on this ADOT thing that I can't really describe what I did.

I tried following the official documentation of ADOT for installation.

Collector Configuration

Installation

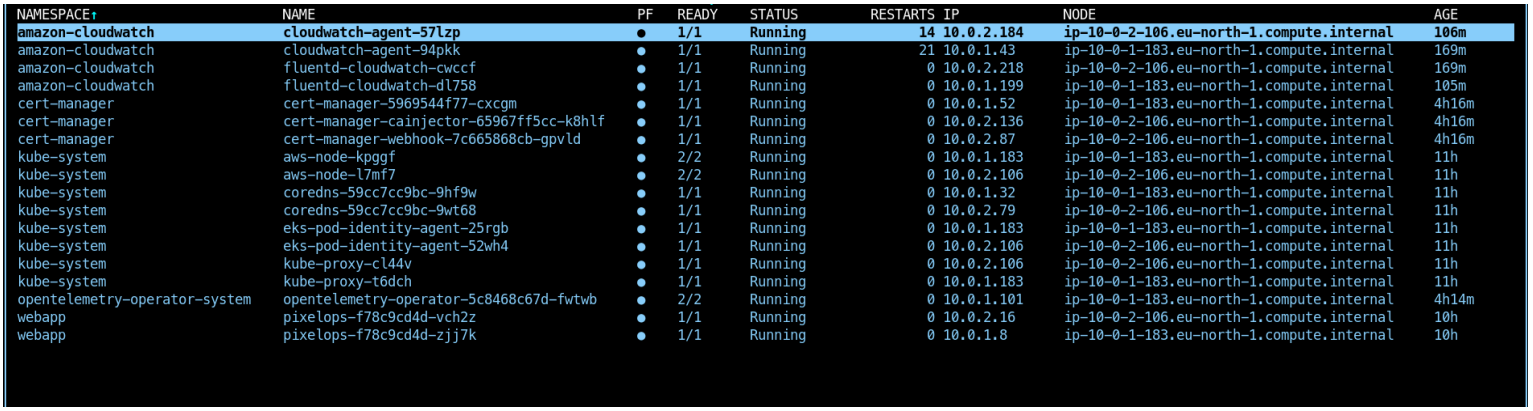
I tried forking**[/container-insights/]** the EKS Workshop part on this and proceeding from there, but never I was able to see my actual metrics in the Cloudwatch.

I tried following this guy to setup fluentd but that also failed.

https://www.youtube.com/watch?v=-8kFawk-EFs

I tried asking AI to correct and configure the yaml files that I applied.

I also tried turning it on from the Console (the ADOT Addon).



NAMESPACE	NAME	PF	READY	STATUS	RESTARTS	IP	NODE	AGE
amazon-cloudwatch	cloudwatch-agent-57lzp	●	1/1	Running	14	10.0.2.184	ip-10-0-2-106.eu-north-1.compute.internal	106m
amazon-cloudwatch	cloudwatch-agent-94pkk	●	1/1	Running	21	10.0.1.43	ip-10-0-1-183.eu-north-1.compute.internal	169m
amazon-cloudwatch	fluentd-cloudwatch-cwccf	●	1/1	Running	0	10.0.2.218	ip-10-0-2-106.eu-north-1.compute.internal	169m
amazon-cloudwatch	fluentd-cloudwatch-dl758	●	1/1	Running	0	10.0.1.199	ip-10-0-1-183.eu-north-1.compute.internal	105m
cert-manager	cert-manager-5969544f77-cxcgm	●	1/1	Running	0	10.0.1.52	ip-10-0-1-183.eu-north-1.compute.internal	4h16m
cert-manager	cert-manager-cainjector-65967ff5cc-k8hlf	●	1/1	Running	0	10.0.2.136	ip-10-0-2-106.eu-north-1.compute.internal	4h16m
cert-manager	cert-manager-webhook-7c665868cb-gpvld	●	1/1	Running	0	10.0.2.87	ip-10-0-2-106.eu-north-1.compute.internal	4h16m
kube-system	aws-node-kpggf	●	2/2	Running	0	10.0.1.183	ip-10-0-1-183.eu-north-1.compute.internal	11h
kube-system	aws-node-l7mf7	●	2/2	Running	0	10.0.2.106	ip-10-0-2-106.eu-north-1.compute.internal	11h
kube-system	coredns-59cc7cc9bc-9hf9w	●	1/1	Running	0	10.0.1.32	ip-10-0-1-183.eu-north-1.compute.internal	11h
kube-system	coredns-59cc7cc9bc-9wt68	●	1/1	Running	0	10.0.2.79	ip-10-0-2-106.eu-north-1.compute.internal	11h
kube-system	eks-pod-identity-agent-25rgb	●	1/1	Running	0	10.0.1.183	ip-10-0-1-183.eu-north-1.compute.internal	11h
kube-system	eks-pod-identity-agent-52wh4	●	1/1	Running	0	10.0.2.106	ip-10-0-2-106.eu-north-1.compute.internal	11h
kube-system	kube-proxy-cl44v	●	1/1	Running	0	10.0.2.106	ip-10-0-2-106.eu-north-1.compute.internal	11h
kube-system	kube-proxy-t6dch	●	1/1	Running	0	10.0.1.183	ip-10-0-1-183.eu-north-1.compute.internal	11h
opentelemetry-operator-system	opentelemetry-operator-5c8468c67d-fwtwb	●	2/2	Running	0	10.0.1.101	ip-10-0-1-183.eu-north-1.compute.internal	4h14m
webapp	pixelops-f78c9cd4d-vch2z	●	1/1	Running	0	10.0.2.16	ip-10-0-2-106.eu-north-1.compute.internal	10h
webapp	pixelops-f78c9cd4d-zjj7k	●	1/1	Running	0	10.0.1.8	ip-10-0-1-183.eu-north-1.compute.internal	10h

The earlier mentioned **iam permissions attribute** was a requirement for Cloudwatch.

Deploy prometheus and grafana

We will use helm and kubectl to configure kube-prometheus-stack to deploy Prometheus and Grafana.
To install metrics-server:

```
kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

Add the repository to helm repo list:

```
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts
```

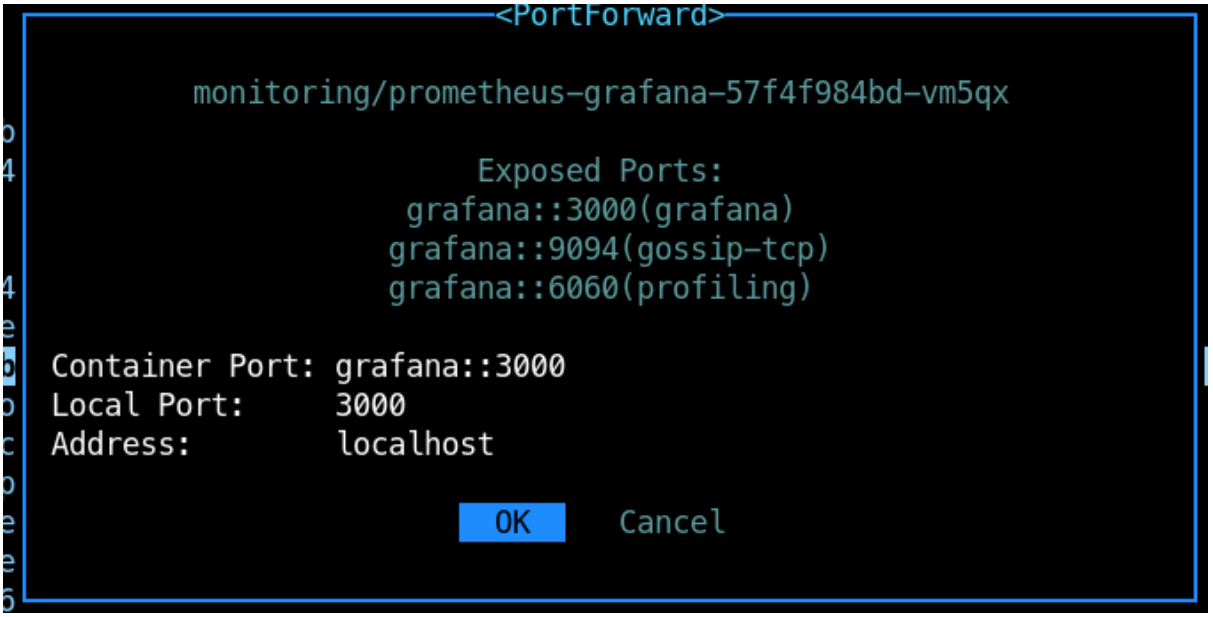
Install the stack with specific configurations for persistent volumes

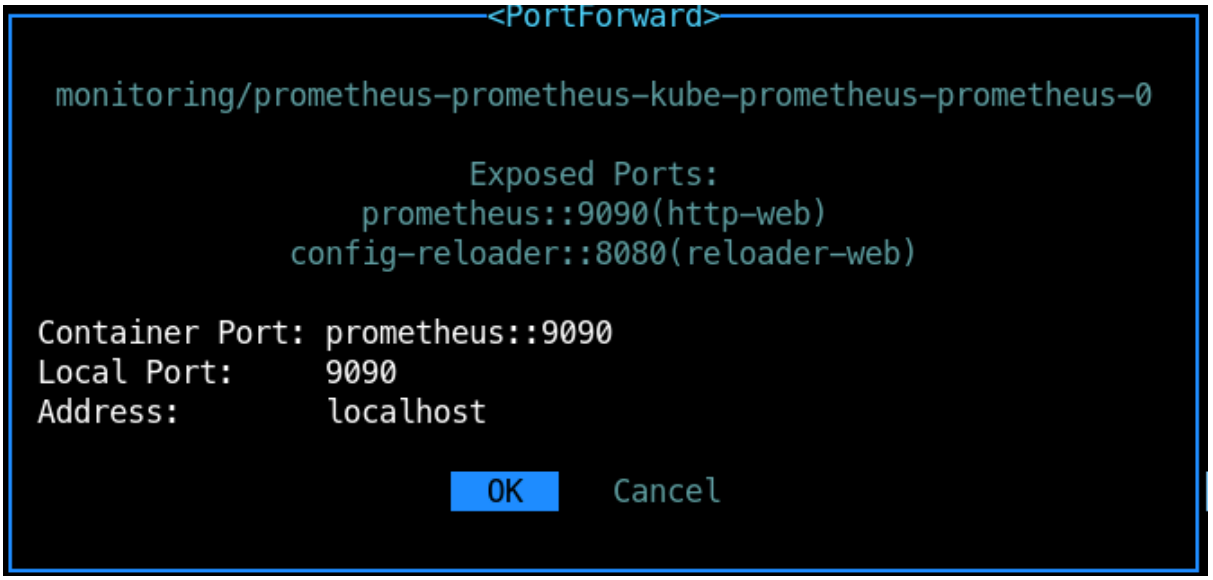
```
helm install prometheus prometheus-community/kube-prometheus-stack \\n
--namespace monitoring --create-namespace \\n  --set
alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume
.storageClass="gp2"
```

To retrieve the Grafana password (login is admin, pass is prom-operator)

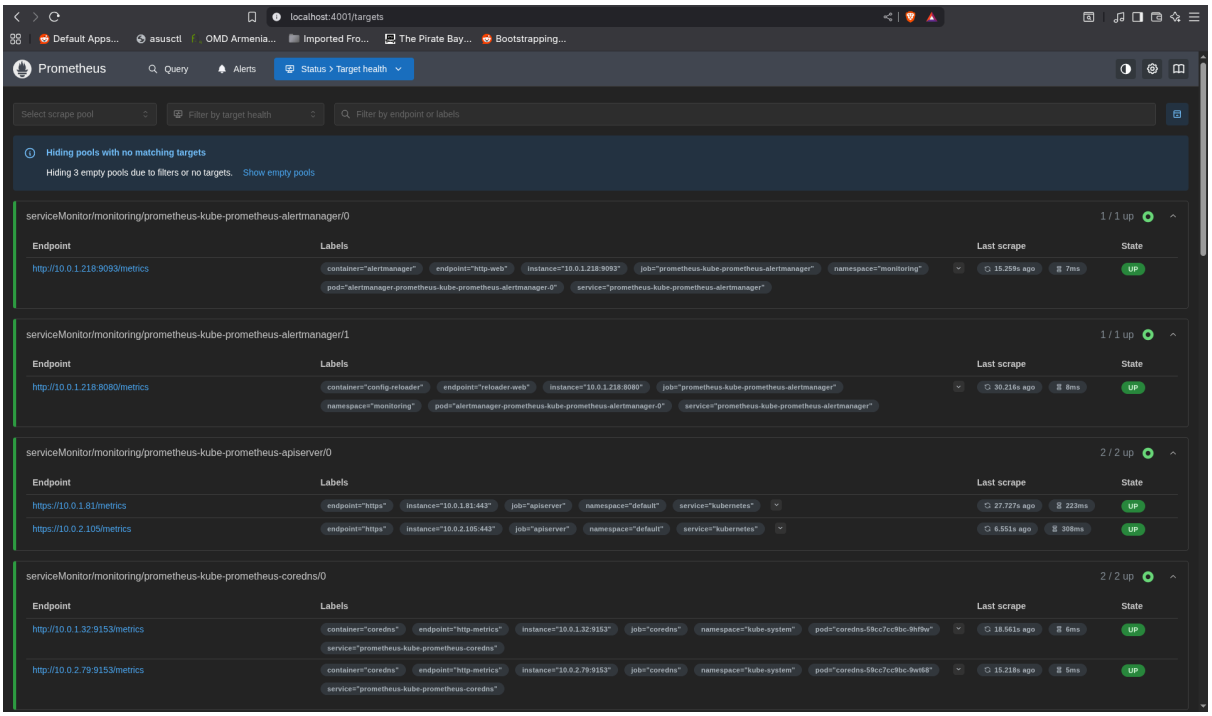
```
kubectl --namespace monitoring get secrets prometheus-grafana -o
jsonpath="{.data.admin-password}" | base64 -d ; echo
```

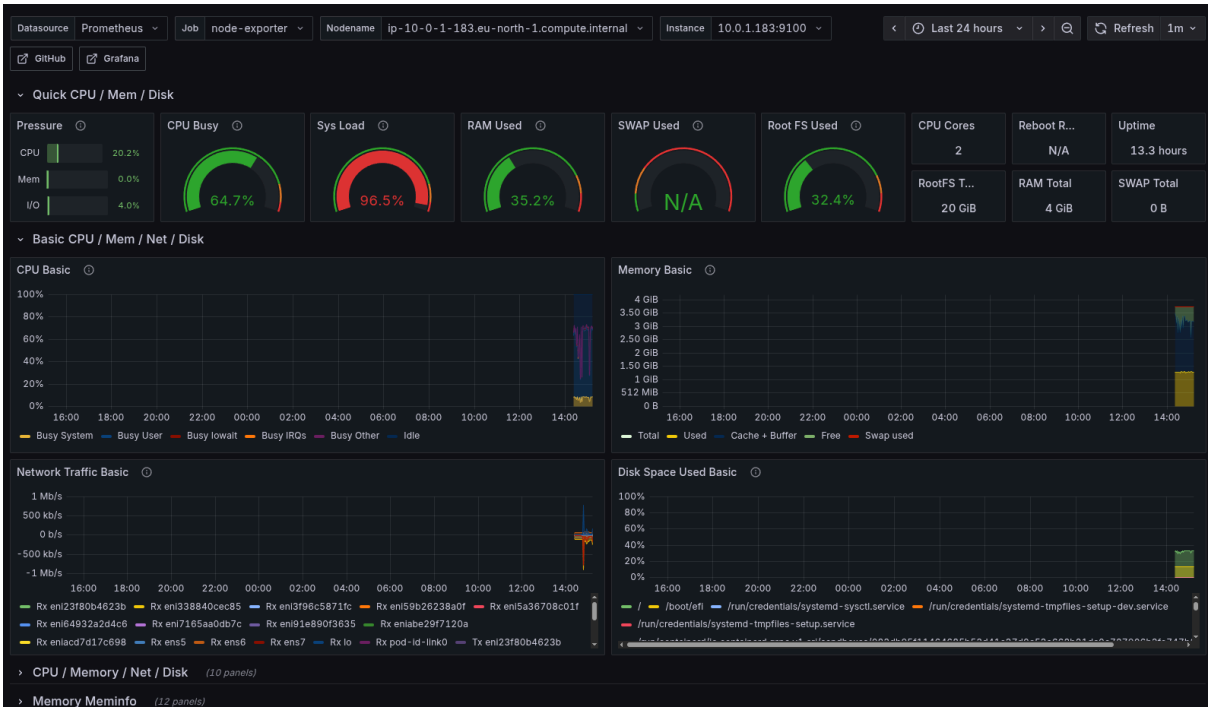
After this we can port forward Prometheus and Grafana to check the results on our machine





Then we access
Grafana at localhost:3000 (then login and check the data sources and the dashboards)
Prometheus at localhost:9090 (then check paths **/metrics** and **/health**)





I was not able to start the ALB Stuff because of poor time management and a significant gap of knowledge and skill.