Solution - Enoda

By Gaurav Desai

Problem Statement

Imagine there is an existing cloud infrastructure for a private blockchain network. We want you to create a monitoring system for this environment to track the metrics that help us trace the blockchain status, our node's performance and the supporting infrastructure:

- Cloud environment - Google Cloud Platform
- Blockchain software is running in 3 different Kubernetes clusters each in a different region
- The type of blockchain node can be your choice (e.g., Hyperledger Fabric, Ethereum, etc.).
- The scope of this assessment does not include the configuration of Kubernetes clusters or any other infrastructure.

Bonus Challenge

This is an optional opportunity to demonstrate your technical expertise:

- How would you approach a chain reorganization event (fork)?
- What specific metrics and thresholds would you implement to monitor for and potentially mitigate the impact of such an event?

Also:

- We value your analytical and critical thinking skills more than your coding abilities.
- We want to understand how you would approach a challenge involving a distributed network. This includes your ability to research new concepts, make informed assumptions, and justify your technology decisions based on specific criteria.

Solution

Based on the architecture and volume of blockchain nodes, there are 2 ways to establish the observability and monitoring system in place using Prometheus and Grafana dashboard.

If there's less numbers of blockchain nodes as per the tech challenge then we will go for the centralised prometheus - grafana implementation where exporters from all different blockchain nodes in different regions would export the metrics to prometheus server and then we can establish the alerting mechanism and grafana ui part there.

Solution

For the other case which I assume that the business may consider in the future with growth - where we have to monitor great numbers of blockchain nodes across different regions and zones then we can consider deploying prometheus instances in every region. Collect the metrics to the central prometheus instance and then the alerting and Grafana UI implementation part get involved from there.

Solution

In regards to the technical challenge - my approach would be first to create 3 kubernetes clusters with blockchain nodes, which are already present. I would either use helm charts for prometheus with adding prometheus community repo by following command on cloudshell:

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

Or

Solution

Would rather prefer bitnami as it sometimes fails while trying to access the above url due to network restrictions (VPN) or cloud constraints.

With bitnami I can follow below steps:

$ helm repo add bitnami https://charts.bitnami.com/bitnami

Followed by

$ helm repo update

Solution

To ensure the helm charts repo would be updated till latest changes.

Later we can install prometheus and grafana with the same bitnami source.

```
garyd2502@cloudshell:~ (challenge-449613)$
garyd2502@cloudshell:~ (challenge-449613)$ helm install prometheus bitnami/prometheus --namespace monitoring
```

```
garyd2502@cloudshell:~ (challenge-449613)$
garyd2502@cloudshell:~ (challenge-449613)$
garyd2502@cloudshell:~ (challenge-449613)$ helm install grafana bitnami/grafana --namespace monitoring
```

Solution

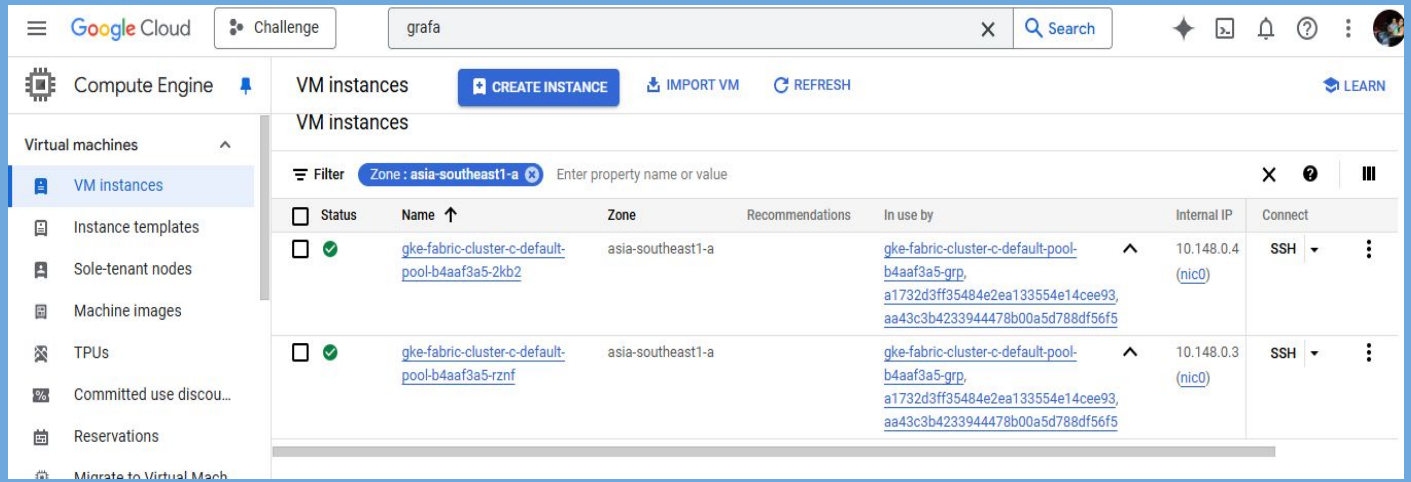
Once done we can verify the installations as below:

```
garyd2502@cloudshell:~ (challenge-449613)$
garyd2502@cloudshell:~ (challenge-449613)$ kubectl get pods -n monitoring
NAME                                READY   STATUS    RESTARTS   AGE
grafana-f8bb45c9b-xsk5g             1/1     Running   0          4h25m
prometheus-alertmanager-0           1/1     Running   0          4h26m
prometheus-server-b94d6d444-klsjx   1/1     Running   0          4h26m
garyd2502@cloudshell:~ (challenge-449613)$
```

Solution

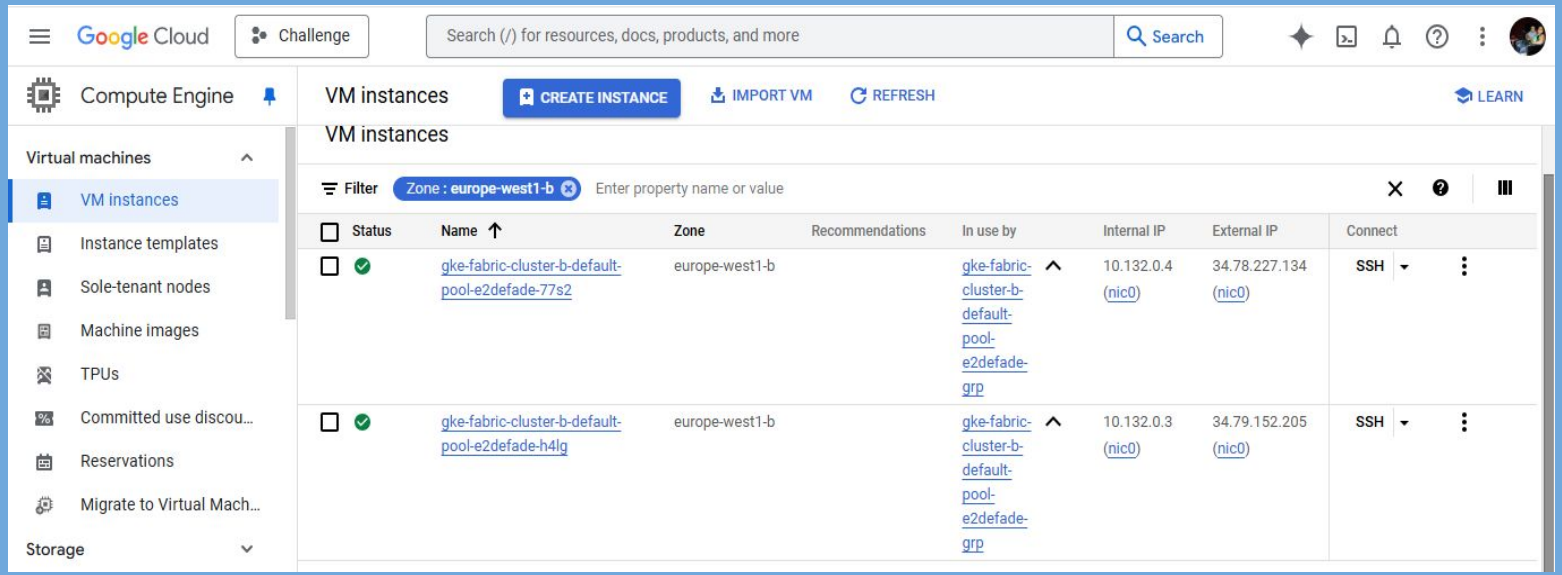
So far the infrastructure will look as follows:

1. Asia southeast region nodes:



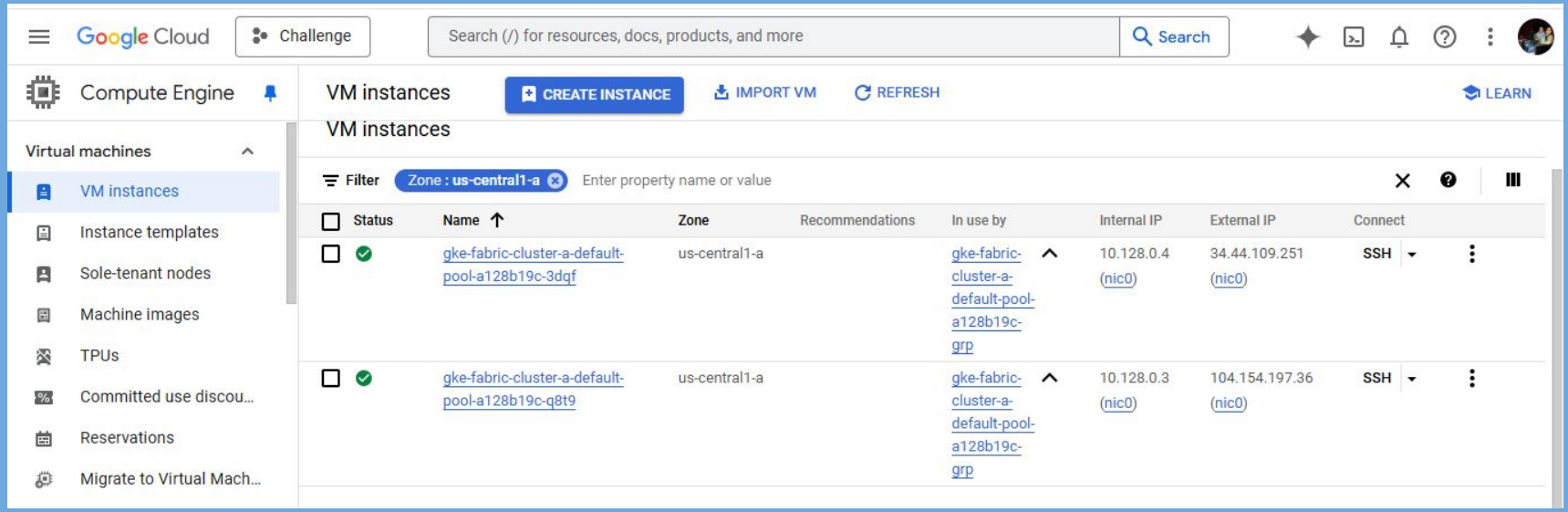
Solution

2. Europe west region nodes:



Solution

3. US central region:



Solution

Observability and monitoring:

As we are using grafana and prometheus for observability and monitoring, it's vital to consider firewall rules accordingly.

```
garyd2502@cloudshell:~ (challenge-449613)$ kubectl get svc grafana -n monitoring
NAME      TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
grafana   LoadBalancer   34.118.237.248   <pending>     3000:32635/TCP   6h41m
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute firewall-rules create allow-grafana \
  --allow=tcp:3000 \
  --target-tags=grafana \
  --description="Allow Grafana traffic"
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/challenge-449613/global/firewalls/allow-grafana].
Creating firewall...done.
NAME: allow-grafana
NETWORK: default
DIRECTION: INGRESS
PRIORITY: 1000
ALLOW: tcp:3000
DENY:
DISABLED: False
garyd2502@cloudshell:~ (challenge-449613)$
```

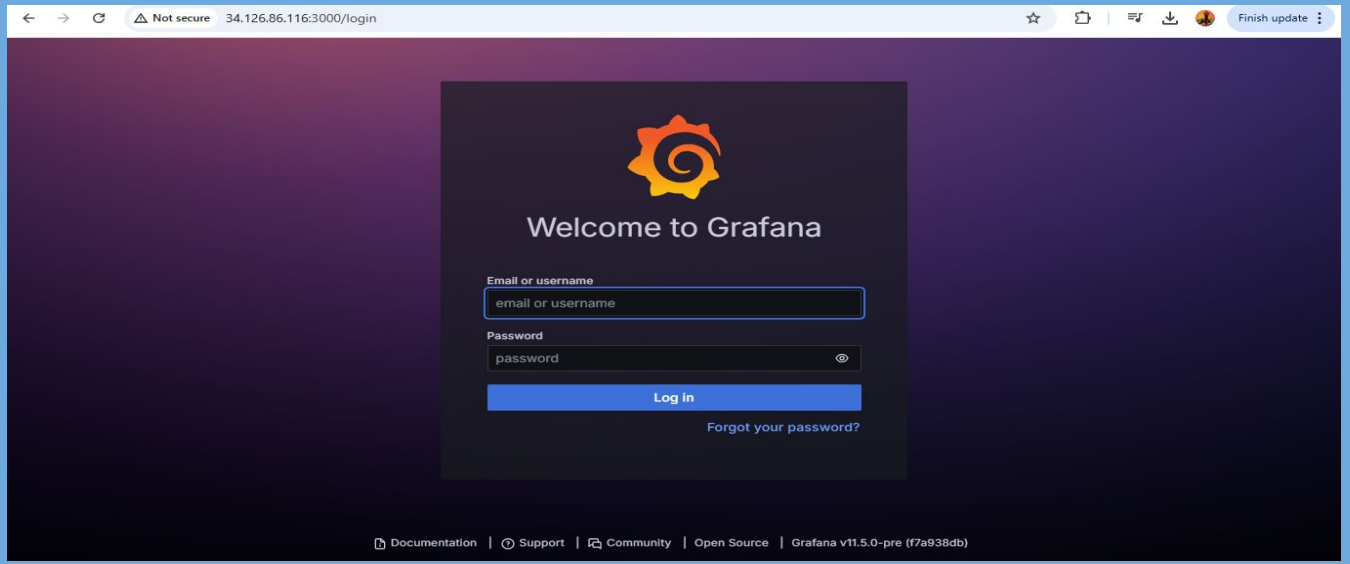
Solution

Adding the firewall tags to each cluster nodes to ensure the proper export of grafana UI.

```
DISABLED: False
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute instances add-tags gke-fabric-cluster-a-default-pool-a128b19c-3dqf --tags=grafana
Did you mean zone [europe-west4-b] for instance: [gke-fabric-cluster-a-default-pool-a128b19c-3dqf] (Y/n)?  n

No zone specified. Using zone [us-central1-a] for instance: [gke-fabric-cluster-a-default-pool-a128b19c-3dqf].
Updated [https://www.googleapis.com/compute/v1/projects/challenge-449613/zones/us-central1-a/instances/gke-fabric-cluster-a-default-pool-a128b19c-3dqf].
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute instances add-tags gke-fabric-cluster-a-default-pool-a128b19c-q8t9 --tags=grafana
Did you mean zone [europe-west4-b] for instance: [gke-fabric-cluster-a-default-pool-a128b19c-q8t9] (Y/n)?  n

No zone specified. Using zone [us-central1-a] for instance: [gke-fabric-cluster-a-default-pool-a128b19c-q8t9].
Updated [https://www.googleapis.com/compute/v1/projects/challenge-449613/zones/us-central1-a/instances/gke-fabric-cluster-a-default-pool-a128b19c-q8t9].
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute instances add-tags gke-fabric-cluster-b-default-pool-e2defade-77s2 --tags=grafana
Did you mean zone [europe-west4-b] for instance: [gke-fabric-cluster-b-default-pool-e2defade-77s2] (Y/n)?  n

No zone specified. Using zone [europe-west1-b] for instance: [gke-fabric-cluster-b-default-pool-e2defade-77s2].
Updated [https://www.googleapis.com/compute/v1/projects/challenge-449613/zones/europe-west1-b/instances/gke-fabric-cluster-b-default-pool-e2defade-77s2].
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute instances add-tags gke-fabric-cluster-b-default-pool-e2defade-h4lg --tags=grafana
Did you mean zone [europe-west4-b] for instance: [gke-fabric-cluster-b-default-pool-e2defade-h4lg] (Y/n)?  n

No zone specified. Using zone [europe-west1-b] for instance: [gke-fabric-cluster-b-default-pool-e2defade-h4lg].
Updated [https://www.googleapis.com/compute/v1/projects/challenge-449613/zones/europe-west1-b/instances/gke-fabric-cluster-b-default-pool-e2defade-h4lg].
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute instances add-tags gke-fabric-cluster-c-default-pool-b4aaf3a5-2kb2 --tags=grafana
Did you mean zone [europe-west4-b] for instance: [gke-fabric-cluster-c-default-pool-b4aaf3a5-2kb2] (Y/n)?  n

No zone specified. Using zone [asia-southeast1-a] for instance: [gke-fabric-cluster-c-default-pool-b4aaf3a5-2kb2].
Updated [https://www.googleapis.com/compute/v1/projects/challenge-449613/zones/asia-southeast1-a/instances/gke-fabric-cluster-c-default-pool-b4aaf3a5-2kb2].
garyd2502@cloudshell:~ (challenge-449613)$ gcloud compute instances add-tags gke-fabric-cluster-c-default-pool-b4aaf3a5-rznf --tags=grafana
Did you mean zone [europe-west4-b] for instance: [gke-fabric-cluster-c-default-pool-b4aaf3a5-rznf] (Y/n)?  n

No zone specified. Using zone [asia-southeast1-a] for instance: [gke-fabric-cluster-c-default-pool-b4aaf3a5-rznf].
Updated [https://www.googleapis.com/compute/v1/projects/challenge-449613/zones/asia-southeast1-a/instances/gke-fabric-cluster-c-default-pool-b4aaf3a5-rznf].
garyd2502@cloudshell:~ (challenge-449613)$
```

Solution

So that grafana UI can be accessible on local host (your device/laptop) and read the metrics for all present nodes across different regions of GCP.



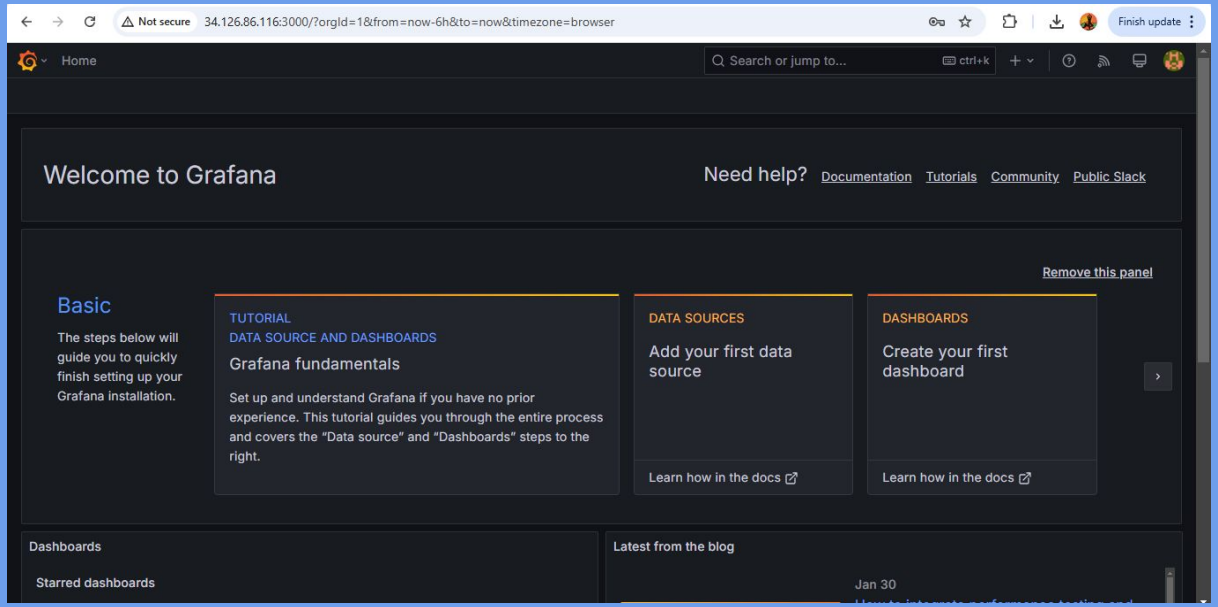
Solution

Generation of first login of admin account of Grafana

```
garyd2502@cloudshell:~ (challenge-449613)$ kubectl describe secret grafana-admin -n monitoring
Name:         grafana-admin
Namespace:    monitoring
Labels:       app.kubernetes.io/component=grafana
              app.kubernetes.io/instance=grafana
              app.kubernetes.io/managed-by=Helm
              app.kubernetes.io/name=grafana
              app.kubernetes.io/version=11.5.0
              helm.sh/chart=grafana-11.4.5
Annotations:  meta.helm.sh/release-name: grafana
              meta.helm.sh/release-namespace: monitoring

Type:  Opaque

Data
====
GF_SECURITY_ADMIN_PASSWORD:  10 bytes
garyd2502@cloudshell:~ (challenge-449613)$ kubectl get secret grafana-admin -n monitoring -o jsonpath="{.data.GF_SECURITY_ADMIN_PASSWORD}" | base64 --decode
D8MSLGwgycgaryd2502@cloudshell:~ (challenge-449613)$
```

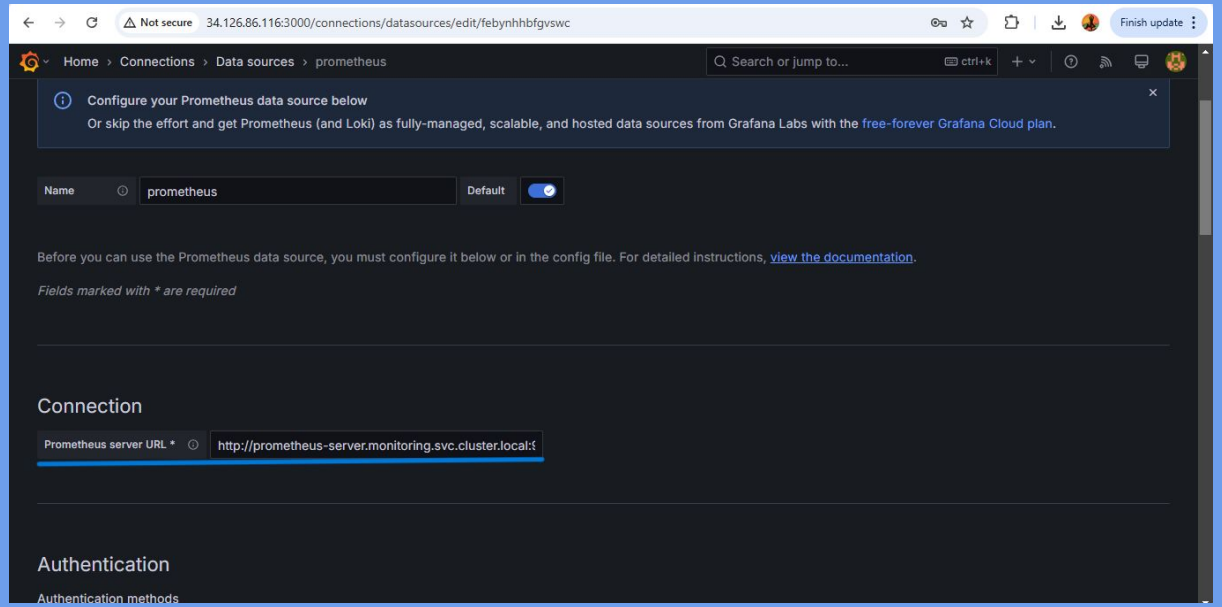
Solution

Login in to Grafana

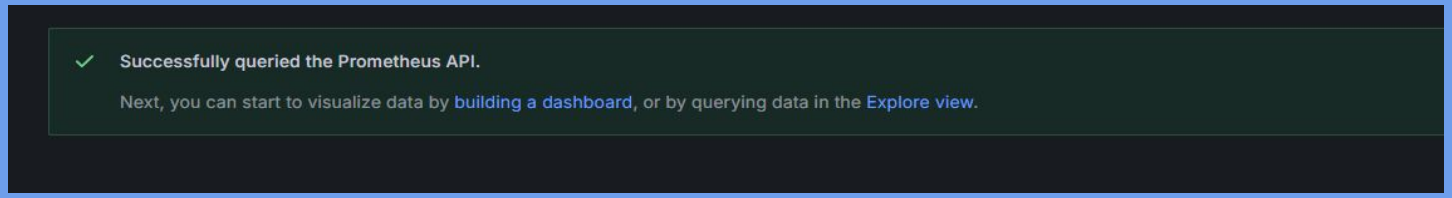


Solution

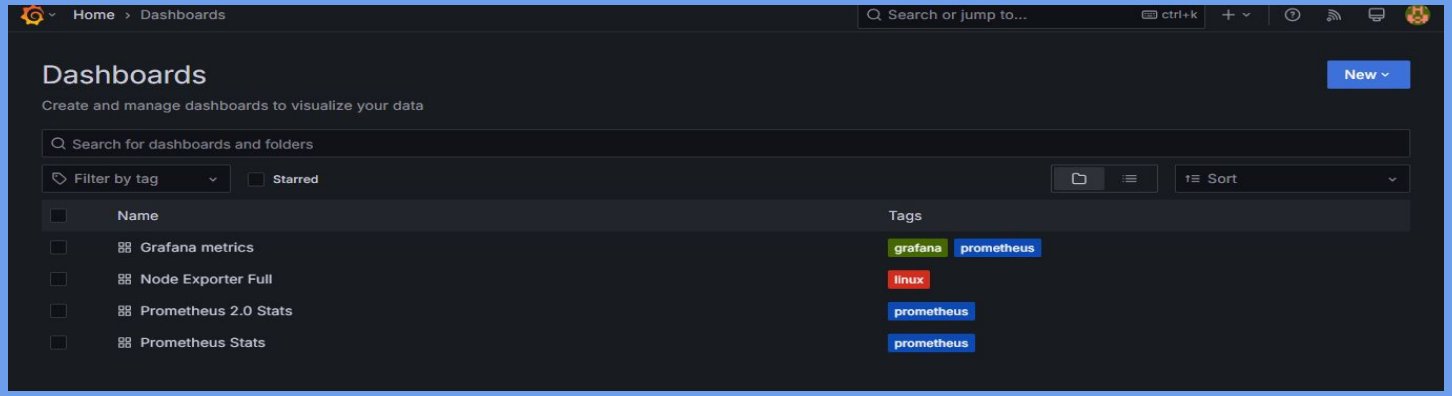
Setting up prometheus as datasource for Grafana UI



Solution

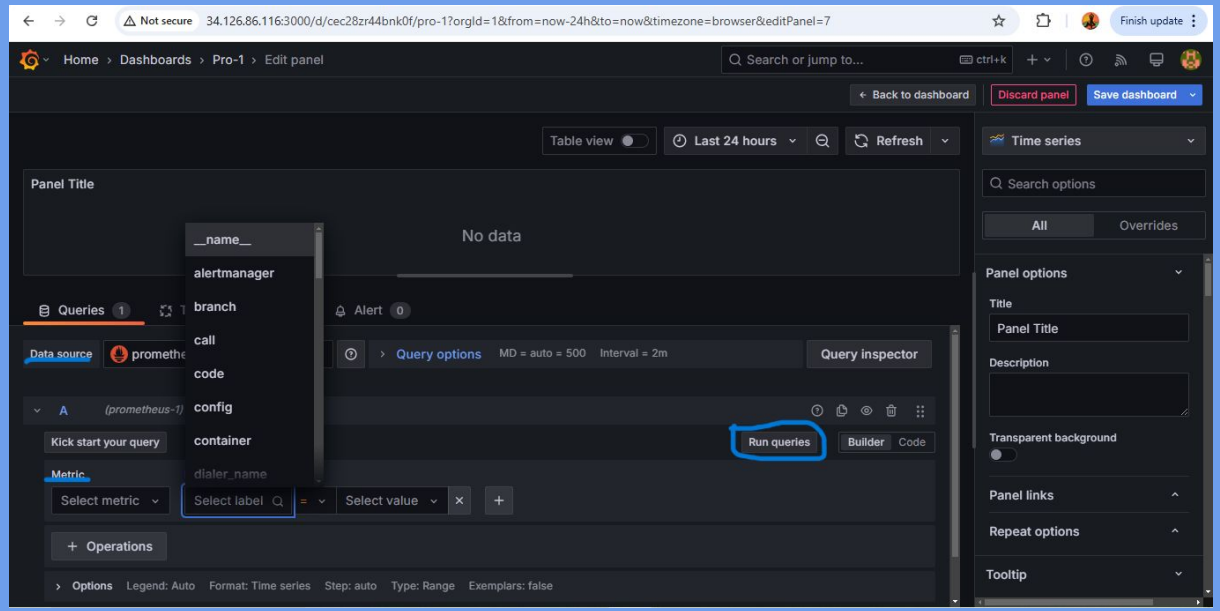


Importing the prometheus dashboards



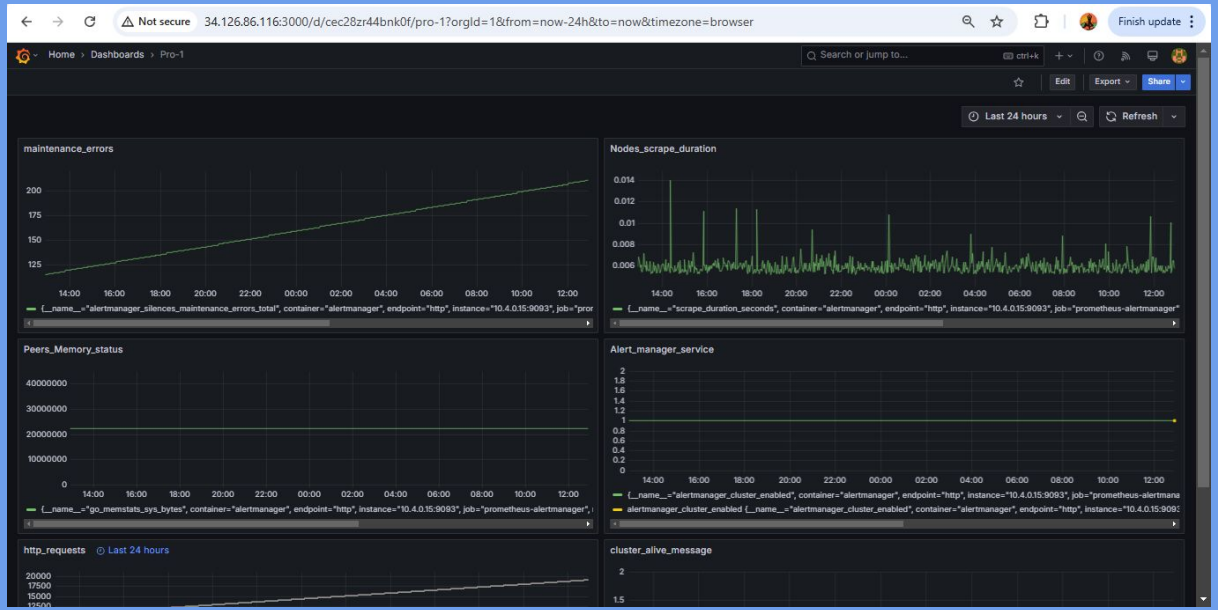
Solution

Configuring the alerts with the help of the correct data source. In this case we will use Prometheus.



Solution

A dashboard with few metrics monitoring panels would look like below:



Bonus Challenge Solution

Block reorganisation event or fork happens for many reasons being - Network latency , Malicious attacks and Accidently simultaneous block generation.

These factors contribute to raising conflicts by creating unwanted blocks and it makes it harder to identify the original ones. This way the blockchains diverge and the trade gets impacted due to this situation.

To get control over these situations, we can tighten the tracing block hashes and establish appropriate monitoring and alerting with use of prometheus, grafana alerts manager and hyperledger fabric events.

Bonus Challenge Solution

- Approaching this situation can be tricky but some known methods can be a great start. Simplest of all is to identify the nature of the fork - whether it is a short fork or a long fork.
- A short fork term is used when no more than 2 blocks get impacted. Whereas it's called a long fork when 3 or more blocks get impacted.
- Short fork issues can be resolved with the other nodes referencing and analysis of the logs from other nodes and identifying the correct chain.
- Long fork issues can be caused due to problems in the network or a malicious attack on blockchain. To resolve this issue one needs to identify if there was an issue with the network or some malicious activity took place. This requires strong monitoring to be able to counter such issues.

Bonus Challenge Solution

- It would be ideal to keep more focused tracking of events where a threshold can be introduced to first identify the nature of the fork.
- There can be the some metrics monitored at prometheus such as blockheights, fork events (as discussed above), changes in committed block hash.
- The further extensions to these checks the Grafana dashboard panels can be configured.
- The alerting can be setup to email alerts or organisational alerting channels like slack.
- I would highly commend to track the orphan blocks as they can be part of malicious attack, if not they can increase network latencies and computational power wastage.

Thank you!