**What is Prometheus?**

[Prometheus](https://github.com/prometheus) is an open-source systems monitoring and alerting toolkit originally built at [SoundCloud](https://soundcloud.com/). Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user [community](https://prometheus.io/community). It is now a standalone open source project and maintained independently of any company. To emphasize this, and to clarify the project's governance structure, Prometheus joined the [Cloud Native Computing Foundation](https://cncf.io/) in 2016 as the second hosted project, after [Kubernetes](http://kubernetes.io/).

Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

For more elaborate overviews of Prometheus, see the resources linked from the [media](https://prometheus.io/docs/introduction/media/) section.

**Features**

Prometheus's main features are:

* a multi-dimensional [data model](https://prometheus.io/docs/concepts/data_model/) with time series data identified by metric name and key/value pairs
* PromQL, a [flexible query language](https://prometheus.io/docs/prometheus/latest/querying/basics/) to leverage this dimensionality
* no reliance on distributed storage; single server nodes are autonomous
* time series collection happens via a pull model over HTTP
* [pushing time series](https://prometheus.io/docs/instrumenting/pushing/) is supported via an intermediary gateway
* targets are discovered via service discovery or static configuration
* multiple modes of graphing and dashboarding support

**What are metrics?**

Metrics are numerical measurements in layperson terms. The term time series refers to the recording of changes over time. What users want to measure differs from application to application. For a web server, it could be request times; for a database, it could be the number of active connections or active queries, and so on.

Metrics play an important role in understanding why your application is working in a certain way. Let's assume you are running a web application and discover that it is slow. To learn what is happening with your application, you will need some information. For example, when the number of requests is high, the application may become slow. If you have the request count metric, you can determine the cause and increase the number of servers to handle the load.

**Grafana**

[Grafana open source software](https://grafana.com/oss/) enables you to query, visualize, alert on, and explore your metrics, logs, and traces wherever they are stored. Grafana OSS provides you with tools to turn your time-series database (TSDB) data into insightful graphs and visualizations. The Grafana OSS plugin framework also enables you to connect other data sources like NoSQL/SQL databases, ticketing tools like Jira or ServiceNow, and CI/CD tooling like GitLab.

**Create a GKE cluster using cloud**

**Helm installation**

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

./get\_helm.sh

* check the ‘helm version’

**Clone the application repo**

git clone <https://github.com/GoogleCloudPlatform/bank-of-anthos.git>

cd bank-of-anthos/

**Deploy Prometheus, AlertManager and Grafana**

helm repo add bitnami <https://charts.bitnami.com/bitnami>

helm install tutorial bitnami/kube-prometheus \

--version 8.2.2 \

--values extras/prometheus/oss/values.yaml \

--wait

After executing this command some information will be visible , for future purpose we need to copy it at somewhere

Here we need to check pods

Next execute below command

kubectl port-forward --namespace default svc/tutorial-kube-prometheus-prometheus 9090:9090

Here we need to change the port number

helm repo add grafana <https://grafana.github.io/helm-charts>

helm repo update

helm install my-release grafana/Grafana

Here we will get admin and password details for accessing Grafana (Scroll up it will be visible and copy n paste it for decoding for password, copy and save the password)

Check the pods for Grafana

Check the services for both Prometheus n Grafana

Here edit each service as LoadBalancer(kubectl edit scv/svcname)

Next access the prometheus by using svc Eip(port no:9090)

And also Grafana (port no:80)

Create dash board

**THE END**