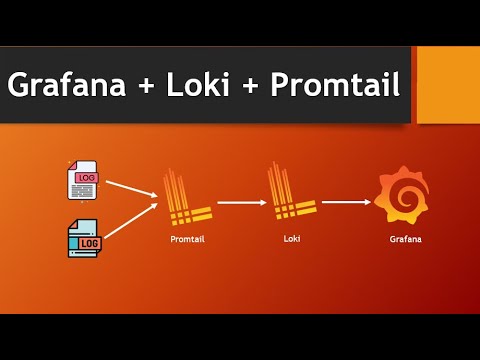
**CAPTURING LOGS USING LOKI STACK**



DESIGN DIAGRAM FOR LOKI BASED STACK

**Loki Based Stack Components**:

**Promtail:**

Promtail is an agent which ships the contents of local logs to a private Grafana Loki instance .

**Loki:**

It is the main server responsible for storing logs and processing queries.

**Grafana**:

This front-end dashboard allows you to query and visualize your data.

**Approach:**

The best approach is to create an application and deploy it as a microservice which accepts logs from all the frontend and backend microservices and then it sends those logs to promtail. Nodejs fits this scenario very well cause we will have to deal heavy writes here.

Why to create a centralized application to collect logs?

1. By isolating it as a separate service, we can scale it very well on demand
2. Better Security.

In case of collecting frontend logs, we can configure our frontend applications to send logs to promtail directly but we need to configure them with promtail ip address or public url and port. It is better not to share those details publicly.

1. Reusability.

We can use this centralized microservice to collect logs from multiple projects across our organization.

**Costs Involved:**

Loki based architecture is very cost effective cause all the log content and index can be saved directly into Amazon s3, GCP cloud storage, Azure Stroage buckets.

Only costs involved in this infrastructure are compute resources, storage and bandwidth pricing.

It only saves labels into index so even for 10TB log data, the index will be just like around 300Mb. We can easily load this index data into memory and can query the data easily. It leads to very less usage of compute resources.

Note: All Components in Loki based Stack are completely open source. There are no other costs like licensing or any special fees.

**Efforts Required:**

**For SRE Team:**

SRE Team will have heavy work regarding initial setup and also future maintenance and scaling of different components of Loki based Stack.

**Note**: It is easy maintain Loki based stack when compared to other stacks.

**For Dev Team:**

Need to put efforts to create a centralized logging app. Apart from that, remaining work is pretty much same as we connect to any third party logging service.

**Pros And Cons**:

**Pros:**

1. Completely Opensource solution
2. Centralized Logging Capabilities
3. Less resource utilization like compute and storage utilization
4. Easy to maintain when compared to other stacks
5. Very Cost Effective

**Cons:**

1. High cost of ownership
2. Management Requirements(SRE Team)
3. Need to learn logql to query efficiently
4. Slow query performance.