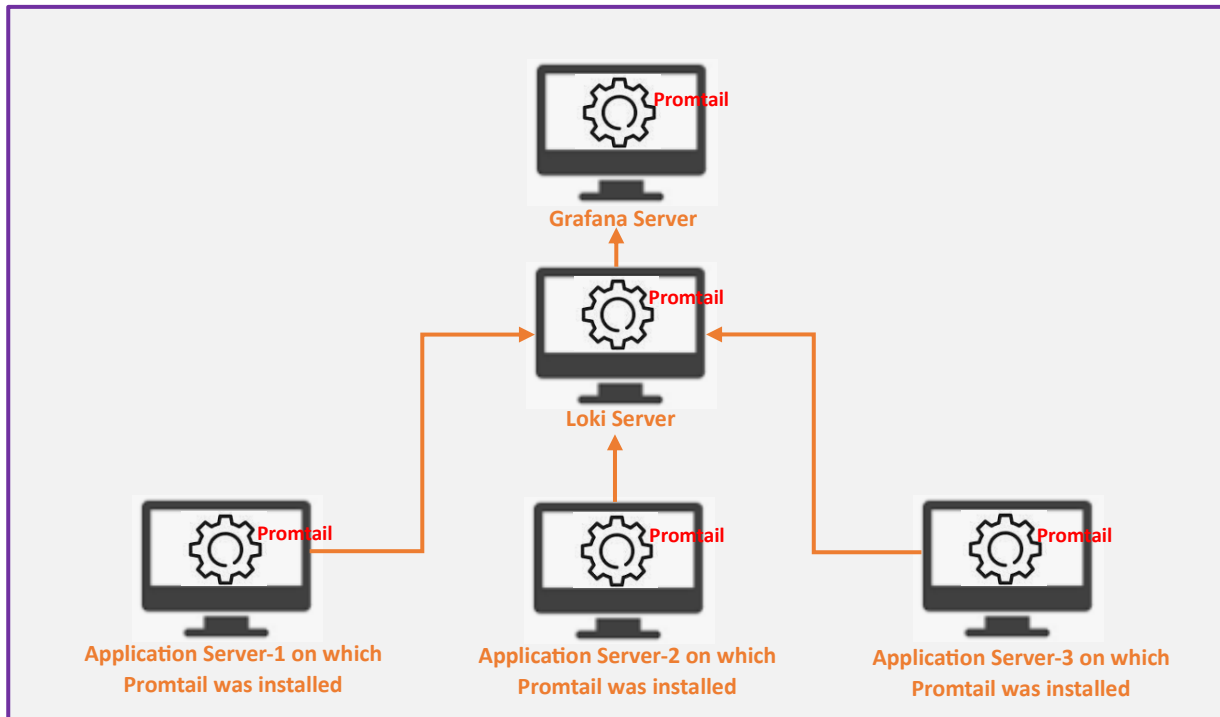# Log Aggregation using Grafana Loki and Promtail

High level architecture diagram of Grafana-Loki-Promtail to aggregate the logs from the Applications Servers and send them to the Loki server and finally to the Grafana is as shown in the diagram shown above.

Log Aggregation from Application Server

In this project Promtail acts as the agent which will extract the logs from the application servers, Loki Servers and Grafana Server and send them to the Loki Server and finally to the Grafana Server. Loki servers I had used as the Cluster of three servers (distributed servers) and there is an Application LoadBalancer and Target Group the three Loki Servers are the part of this Target Group. For extracting logs from the Servers, I have treated Application Server-1, Application Server-2, and Loki Server-1, Loki Server-2, and Loki Server-3 and Grafana Server simply a server.

To integrate Loki with Grafana

For integration of Loki to Grafana I did the entry for DNS Name of Application LoadBalancer of Loki to the Grafana. The health check port and path for Loki Application LoadBalancer is **3100** and **/ready** respectively. In the Target Group of Application Server, you will find all the three Loki Servers will be in Healthy State.

Log Aggregation from EKS

To extract the Logs from the EKS cluster I installed Promtail on EKS cluster as a part of daemonset using the helm chart and hence the Promtail pod will be created on each node of EKS cluster and if in future any new worker node will be created to this EKS cluster then promtail pod will be created automatically which is the property of daemonset. The promtail pod was created as part of

daemonset using the helm chart and in the helm chart I had updated the url for Loki as shown in screenshot attached below.

```
enabled: true
# -- The log level of the Promtail server
# Must be reference in `config.file` to configure `server.log_level`
# See default config in `values.yaml`
logLevel: info
# -- The log format of the Promtail server
# Must be reference in `config.file` to configure `server.log_format`
# Valid formats: `logfmt, json`
# See default config in `values.yaml`
logFormat: logfmt
# -- The port of the Promtail server
# Must be reference in `config.file` to configure `server.http_listen_port`
# See default config in `values.yaml`
serverPort: 3101
# -- The config of clients of the Promtail server
# Must be reference in `config.file` to configure `clients`
# @default -- See `values.yaml`
clients:
  - url: http://10.10.4.165:3100/loki/api/v1/push
  - url: http://10.10.4.14:3100/loki/api/v1/push
  - url: http://10.10.4.195:3100/loki/api/v1/push

# -- Configures where Promtail will save it's positions file, to resume reading after restarts.
# Must be referenced in `config.file` to configure `positions`
positions:
  filename: /run/promtail/positions.yaml
# -- The config to enable tracing
enableTracing: false
# -- A section of reusable snippets that can be reference in `config.file`.
# Custom snippets may be added in order to reduce redundancy.
# This is especially helpful when multiple `kubernetes_sd_configs` are use which usually have large parts in common.
# @default -- See `values.yaml`
snippets:
  pipelineStages:
    - cri: {}
  common:
    - action: replace
      source_labels:
/url                                                                            426,7
```

The high-level architecture diagram for extracting the logs from EKS cluster is as shown below.
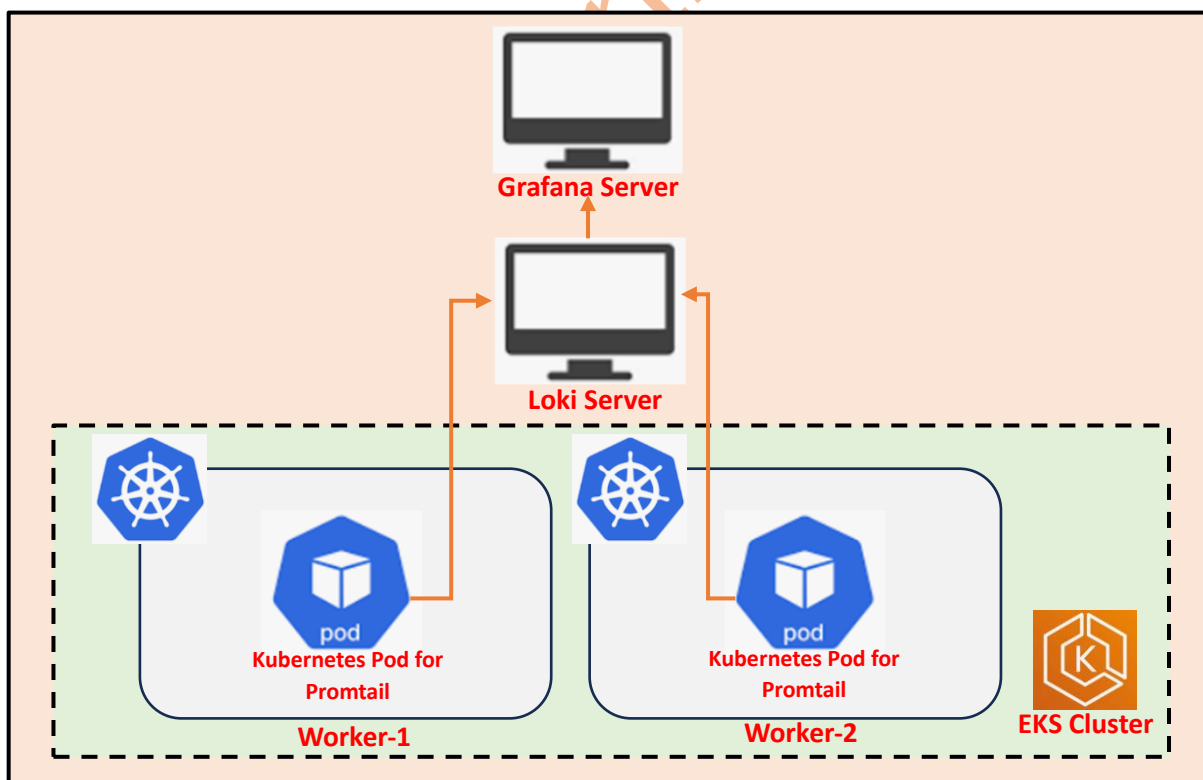
Ran the NodeJS Application on both the application servers as shown in the screenshot attached below.

```
[root@██████████ simple-nodejs-app]# npm install
[root@██████████ simple-nodejs-app]# nohup npm start >> app.log &
[root@██████████ simple-nodejs-app]# npm install
[root@██████████ simple-nodejs-app]# nohup npm start >> app.log &
```

Below screenshot shows the configuration for Loki and Promtail on the three Loki Servers as I used Loki distributed architecture.

```
[root@██████████ ~]# cat /opt/loki-local-config.yaml
auth_enabled: false

server:
  http_listen_port: 3100
  grpc_listen_port: 9096
  log_level: debug
  grpc_server_max_concurrent_streams: 1000

common:
  instance_addr: 10.10.4.165
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
      rules_directory: /tmp/loki/rules
  replication_factor: 3
  ring:
    kvstore:
      store: memberlist

memberlist:
  join_members:
    - 10.10.4.165:7946
    - 10.10.4.14:7946
    - 10.10.4.195:7946

query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100

schema_config:
  configs:
    - from: 2020-10-24
      store: tsdb
      object_store: filesystem
```

```yaml
        schema: v13
        index:
          prefix: index_
          period: 24h

pattern_ingester:
  enabled: true
  metric_aggregation:
    enabled: true
    loki_address: 10.10.4.165:3100

ruler:
  alertmanager_url: http://10.10.4.165:9093

frontend:
  encoding: protobuf
```

```
cat /opt/loki-local-config.yaml

auth_enabled: false


server:

  http_listen_port: 3100

  grpc_listen_port: 9096

  log_level: debug

  grpc_server_max_concurrent_streams: 1000


common:

  instance_addr: 10.10.4.165

  path_prefix: /tmp/loki

  storage:

    filesystem:

      chunks_directory: /tmp/loki/chunks

      rules_directory: /tmp/loki/rules

  replication_factor: 3

  ring:

    kvstore:

      store: memberlist


memberlist:

  join_members:

    - 10.10.4.165:7946

    - 10.10.4.14:7946

    - 10.10.4.195:7946


query_range:

  results_cache:

    cache:

      embedded_cache:
```

```yaml
      enabled: true

      max_size_mb: 100


schema_config:
  configs:
    - from: 2020-10-24
      store: tsdb
      object_store: filesystem
      schema: v13
      index:
        prefix: index_
        period: 24h


pattern_ingester:
  enabled: true
  metric_aggregation:
    enabled: true
    loki_address: 10.10.4.165:3100


ruler:
  alertmanager_url: http://10.10.4.165:9093


frontend:
  encoding: protobuf
```

```
[root@████████████ ~]# cat /opt/promtail-local-config.yaml
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://10.10.4.165:3100/loki/api/v1/push
  - url: http://10.10.4.14:3100/loki/api/v1/push
  - url: http://10.10.4.195:3100/loki/api/v1/push

scrape_configs:
- job_name: system
  static_configs:
  - targets:
      - localhost
    labels:
      job: varlogs
      __path__: /var/log/*log
      stream: stdout
```

```
cat /opt/promtail-local-config.yaml

server:

  http_listen_port: 9080

  grpc_listen_port: 0


positions:

  filename: /tmp/positions.yaml


clients:

  - url: http://10.10.4.165:3100/loki/api/v1/push

  - url: http://10.10.4.14:3100/loki/api/v1/push

  - url: http://10.10.4.195:3100/loki/api/v1/push


scrape_configs:

- job_name: system

  static_configs:

  - targets:

      - localhost

    labels:

      job: varlogs

      __path__: /var/log/*log

      stream: stdout
```

After the Configuration of distributed Loki and Promtail on all the three servers start Loki and Promtail as a service as shown in the screenshot attached below.

```
[root@            ~]# systemctl start loki.service
[root@            ~]# systemctl status loki.service

[root@            ~]# systemctl enable loki.service

[root@            ~]# systemctl start promtail.service
[root@            ~]# systemctl status promtail.service

[root@            ~]# systemctl enable promtail.service
```

NodeJS Applications are running on two servers app-server-1 and app-server-2 as shown in the screenshot attached below.

```
[root@_____ ~]# git clone https://github.com/singhritesh85/simple-nodejs-app.git

[root@_____ simple-nodejs-app]# npm install

[root@_____ simple-nodejs-app]# nohup npm start >> app.log &

[root@_____ ~]# cat /opt/promtail-local-config.yaml
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://10.10.4.165:3100/loki/api/v1/push
  - url: http://10.10.4.14:3100/loki/api/v1/push
  - url: http://10.10.4.195:3100/loki/api/v1/push

scrape_configs:
- job_name: system
  static_configs:
  - targets:
      - localhost
    labels:
      job: varlogs
      __path__: /var/log/*log
      stream: stdout
- job_name: dexter
  static_configs:
  - targets:
      - localhost
    labels:
      job: dexter-application-logs
      __path__: /root/simple-nodejs-app/*log
      stream: stdout
```

```
cat /opt/promtail-local-config.yaml

server:

 http_listen_port: 9080

 grpc_listen_port: 0


positions:

 filename: /tmp/positions.yaml


clients:

 - url: http://10.10.4.165:3100/loki/api/v1/push

 - url: http://10.10.4.14:3100/loki/api/v1/push

 - url: http://10.10.4.195:3100/loki/api/v1/push


scrape_configs:

- job_name: system

 static_configs:

 - targets:

   - localhost

  labels:

   job: varlogs

   __path__: /var/log/*log

   stream: stdout

- job_name: dexter

 static_configs:

 - targets:

   - localhost

  labels:

   job: dexter-application-logs

   __path__: /root/simple-nodejs-app/*log

   stream: stdout
```

Start Promtail as a service on these two application servers as shown in the screenshot attached below.

```
[root@            simple-nodejs-app]# systemctl start promtail.service
[root@            simple-nodejs-app]# systemctl status promtail.service

[root@            simple-nodejs-app]# systemctl enable promtail.service
```

Here for EKS Cluster I am aggregating logs from the EKS cluster by installing the promtail on EKS using helm chart as shown in the screenshot attached below.

```
[root@            ~]# git clone https://github.com/singhritesh85/helm-chart-promtail.git

[root@            ~]# helm upgrade --values helm-chart-promtail/values.yaml --install promtail ./helm-chart-promtail -n promtail
Release "promtail" does not exist. Installing it now.
NAME: promtail
LAST DEPLOYED:
NAMESPACE: promtail
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
**********************************************************************
 Welcome to Grafana Promtail
 Chart version: 6.16.6
 Promtail version: 3.0.0
**********************************************************************

Verify the application is working by running these commands:
* kubectl --namespace promtail port-forward daemonset/promtail 3101
* curl http://127.0.0.1:3101/metrics
[root@            ~]# kubectl get all -n promtail
NAME                  READY   STATUS    RESTARTS   AGE
pod/promtail-hc64g    1/1     Running   0          12s
pod/promtail-t66jd    1/1     Running   0          11s

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/promtail   2         2         2       2            2           <none>          12s
```

The promtail pods will be created on each node of the EKS cluster and will scrape the logs and send to the Loki which was created as a distributed cluster as explain earlier.

```
[root@            ~]# kubectl get nodes
NAME                                  STATUS   ROLES    AGE     VERSION
ip-10-   -218.us-east-2.compute.internal   Ready    <none>   3h54m   v1.27.9-eks-5e0fdde
ip-10-   -208.us-east-2.compute.internal   Ready    <none>   3h54m   v1.27.9-eks-5e0fdde
[root@            ~]# kubectl get pods -n promtail -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP           NODE                                  NOMINATED NODE   READINESS GATES
promtail-h  g   1/1     Running   0          17m   10.   .242   ip-10-   -218.us-east-2.compute.internal   <none>           <none>
promtail-t  d   1/1     Running   0          17m   10.   .238   ip-10-   -208.us-east-2.compute.internal   <none>           <none>
```

Before installing promtail pods using helm chart make sure you provided the correct information regarding Loki distributed cluster as shown in the screenshot attached below.

```
[root@                    ~]# vim helm-chart-promtail/values.yaml
```
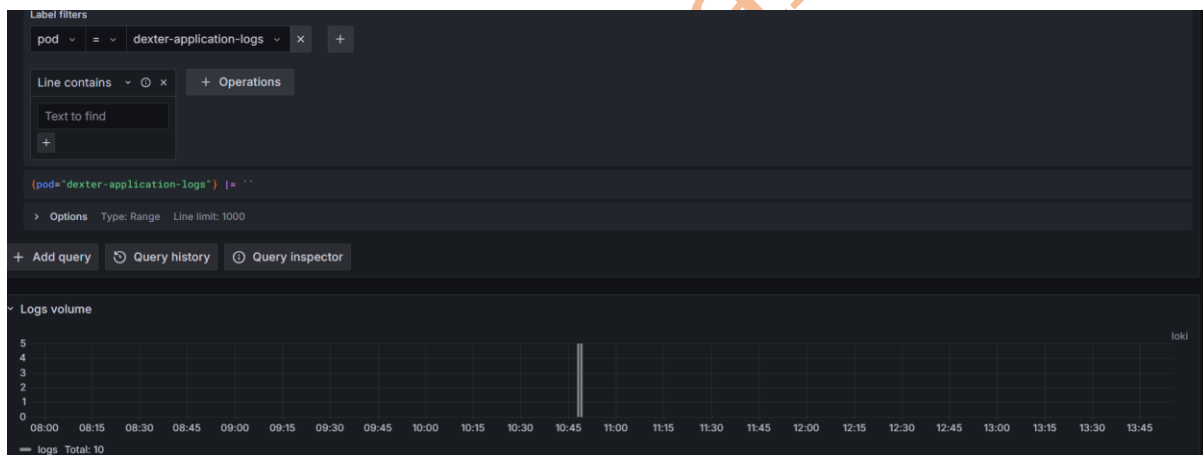
```
  enabled: true
  # -- The log level of the Promtail server
  # Must be reference in `config.file` to configure `server.log_level`
  # See default config in `values.yaml`
  logLevel: info
  # -- The log format of the Promtail server
  # Must be reference in `config.file` to configure `server.log_format`
  # Valid formats: `logfmt, json`
  # See default config in `values.yaml`
  logFormat: logfmt
  # -- The port of the Promtail server
  # Must be reference in `config.file` to configure `server.http_listen_port`
  # See default config in `values.yaml`
  serverPort: 3101
  # -- The config of clients of the Promtail server
  # Must be reference in `config.file` to configure `clients`
  # @default -- See `values.yaml`
  clients:
    - url: http://10.10.4.165:3100/loki/api/v1/push
    - url: http://10.10.4.14:3100/loki/api/v1/push
    - url: http://10.10.4.195:3100/loki/api/v1/push

  # -- Configures where Promtail will save it's positions file, to resume reading after restarts.
  # Must be referenced in `config.file` to configure `positions`
  positions:
    filename: /run/promtail/positions.yaml
  # -- The config to enable tracing
  enableTracing: false
  # -- A section of reusable snippets that can be reference in `config.file`.
  # Custom snippets may be added in order to reduce redundancy.
  # This is especially helpful when multiple `kubernetes_sd_configs` are use which usually have large parts in common.
  # @default -- See `values.yaml`
  snippets:
    pipelineStages:
      - cri: {}
    common:
      - action: replace
        source_labels:
/url                                                                                                    426,7
```
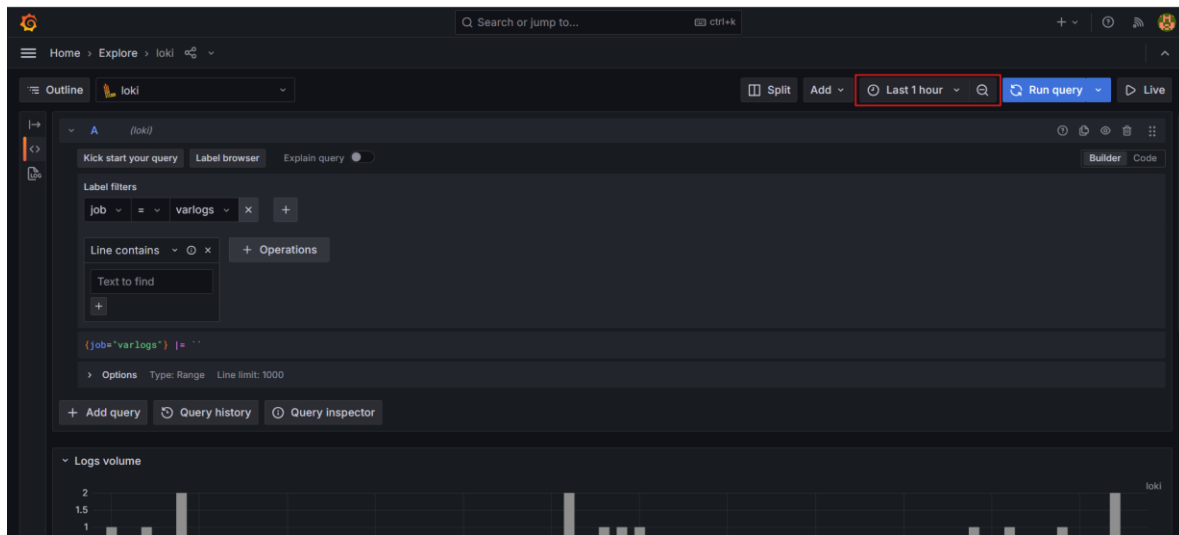
Finally, you can filter the logs as per the requirement.

If needed we can search any string as shown in the screenshot attached below.
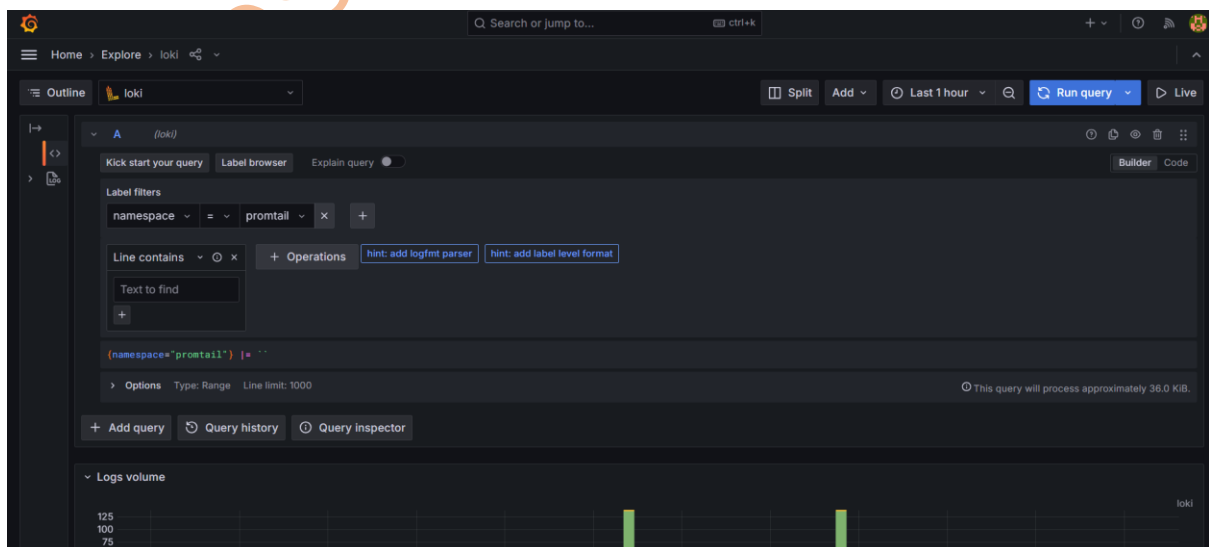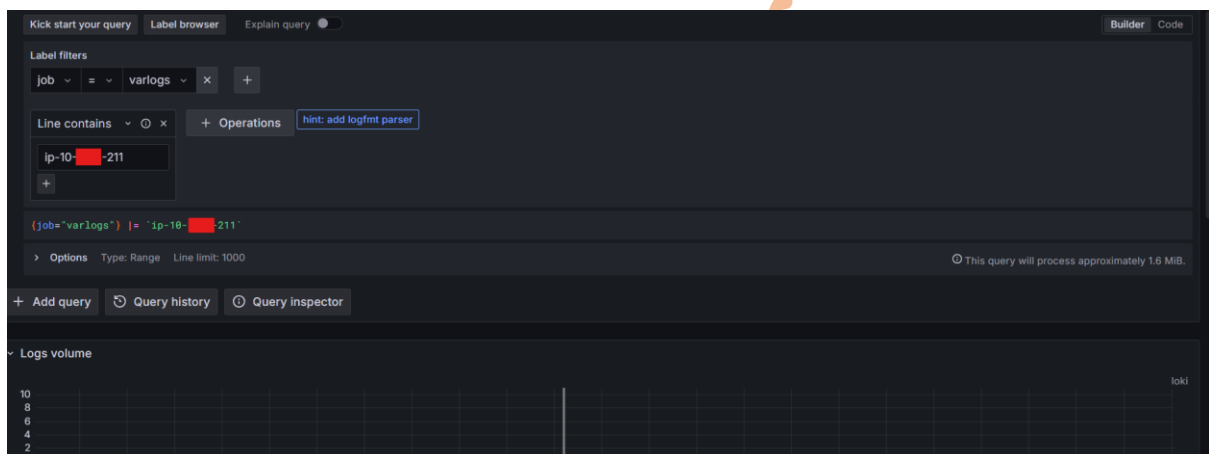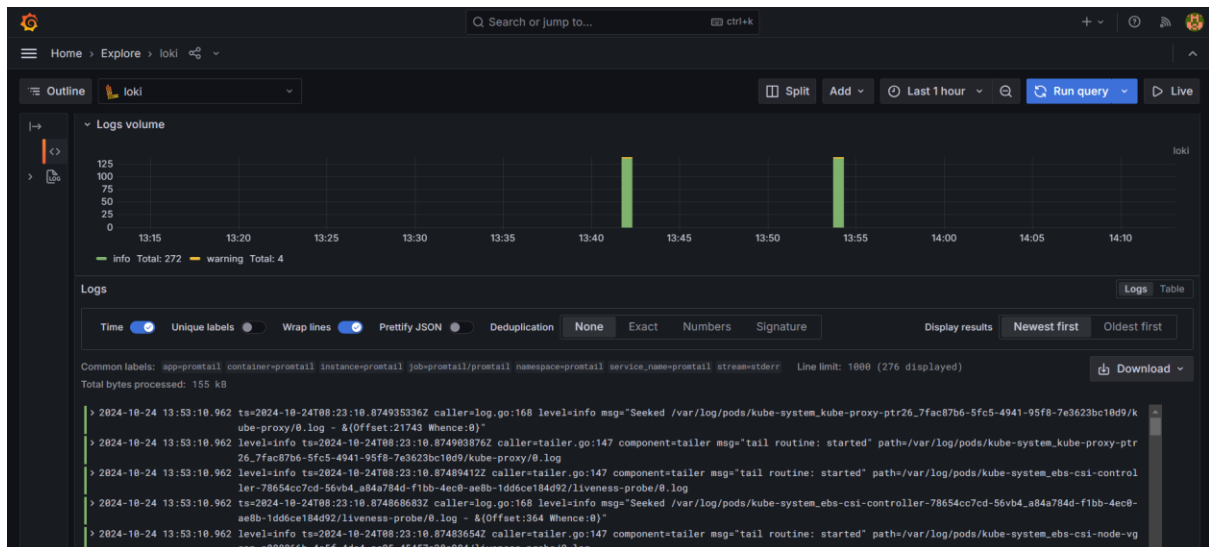
The entry for Route53 is as shown in the screenshot attached below.