Logging Management with Grafana

Running Services

1. Kubernetes
   - What It Does:
     - Kubernetes manages the deployment, scaling, and operation of containerized applications.
   - Role in Logging:
     - It generates logs from various workloads (pods, services, etc.).
     - Provides APIs for log collection via kubectl logs or via log agents installed on nodes.
2. Grafana
   - What It Does:
     - Grafana is a visualization and monitoring tool. It creates dashboards and panels for metrics and logs.
   - Role in Logging:
     - Acts as a user interface to query and visualize logs stored in Loki.
     - Connects to Loki as a data source for log analysis.
3. Loki
   - What It Does:
     - Loki is a log aggregation system designed to work like Prometheus but for logs.
   - Role in Logging:
     - Stores log data in a structured format (optimized for fast retrieval).
     - Supports queries via Grafana, allowing users to search and filter logs using a PromQL-like syntax.
4. Promtail
   - What It Does:
     - Promtail is an agent that collects logs from local files and forwards them to Loki.
   - Role in Logging:
     - Deployed on Kubernetes nodes to read logs from pod log files or the system journal.
     - Adds Kubernetes metadata (like pod labels) to logs for better filtering and correlation in Grafana.
5. Fluentd
   - What It Does:
     - Fluentd is a log processor and forwarder. It supports complex data pipelines.
   - Role in Logging:
     - Collects logs from various sources (applications, Kubernetes nodes).
     - Can process, transform, and enrich logs (e.g., parsing JSON, adding custom metadata).
     - Forwards logs to Loki or another storage backend.
6. Fluent Bit

- What It Does:
    - Fluent Bit is a lightweight log forwarder (a more efficient version of Fluentd for resource-constrained environments).
- Role in Logging:
    - Deployed as a sidecar or daemonset in Kubernetes.
    - Collects logs from Kubernetes workloads or nodes.
    - Forwards logs to Fluentd (for further processing) or directly to Loki.

How They Work Together

1. Kubernetes generates logs from its nodes, pods, and system components.
2. Promtail, Fluent Bit, or Fluentd agents run on Kubernetes nodes:
    - They collect logs from different sources (e.g., container stdout, application logs).
    - These agents enrich logs with Kubernetes metadata (namespace, pod labels).
    - Logs are then forwarded to Loki for storage.
3. Loki stores the logs, indexing them for efficient querying.
4. Grafana queries Loki:
    - Grafana connects to Loki as a data source.
    - Users search logs through the Grafana interface using filters and visualizations.

Simplified Workflow

1. Logs generated: Kubernetes workloads (pods, containers) produce logs.
2. Log collection: Fluentd, Fluent Bit, or Promtail collect and process logs.
3. Log storage: Logs are sent to Loki for indexing and storage.
4. Log visualization: Grafana queries Loki and displays logs for analysis.

This setup enables scalable, efficient log collection and real-time monitoring.

Logging Storage

1. Loki Storage Backend

Loki stores logs in a backend system that is configured when Loki is deployed. It uses two main components for storage:

a. Index Store

- Stores metadata (labels, timestamps, etc.) to allow quick search and filtering of logs.
- By default, Loki minimizes the amount of indexing (compared to systems like Elasticsearch) to optimize for cost and speed.

b. Log Store (Chunks)

- Stores the actual log content in compressed chunks.
- The logs are stored in object storage, or local disk, depending on the configuration.

Common Storage Options for Loki

Loki supports multiple storage backends for scalability and reliability:

1. Local Disk

- Logs are stored on the server's local filesystem.
- Best suited for small-scale setups or testing environments.

Example configuration:

```
storage_config:
  boltdb_shipper:
    active_index_directory: /tmp/loki/boltdb-shipper-active
  filesystem:
    directory: /tmp/loki/chunks
```

2. Object Storage (Cloud or On-Premise)

- Amazon S3, Google Cloud Storage (GCS), Azure Blob Storage, MinIO, etc.

- Preferred for large-scale production setups due to high durability and scalability.

Example configuration for S3:

```
storage_config:
  aws:
    s3: s3://<bucket-name>
    region: us-west-2
```

3. Network Storage

- Shared network file systems like NFS.
- Useful for redundancy across multiple nodes.

4. DynamoDB (Indexing only)

- Loki can store its index in Amazon DynamoDB while storing chunks in S3.
- This is common in highly distributed systems.

2. Where Does Grafana Fit?

- Grafana itself doesn’t store logs.
- It queries Loki, which retrieves logs from its storage backend and serves them to Grafana for visualization.

How to Check or Configure Loki Storage

- Configuration files (typically loki-config.yaml) define the storage backend.
- Example for Kubernetes:
    - Loki configuration is usually passed as a ConfigMap.
    - Check the ConfigMap by running:

```
kubectl get configmap loki-config -n <namespace> -o yaml
```

MinIO as Storage

1. What is MinIO?

- MinIO is a high-performance, distributed object storage system that is compatible with the Amazon S3 API.
- It's often used in self-hosted environments to provide object storage for applications like Loki.

2. How Loki Uses MinIO

- Loki stores its log data (chunks) in MinIO as objects.
- Metadata (indexes) may also be stored in MinIO or another supported backend, depending on your configuration.

3. Loki-MinIO Configuration

The Loki setup with MinIO typically works as follows:

- MinIO Pod: The pod you see (loki-minio-0) runs a MinIO instance, which serves as the object storage.
- Loki Configuration: Loki is configured to use this MinIO instance as its storage backend.

The configuration might look like this in loki-config.yaml:

```
storage_config:
  aws:
    s3: http://loki-minio:9000
    bucketnames: loki
    access_key_id: <your-minio-access-key>
    secret_access_key: <your-minio-secret-key>
```

4. How Data Flows in This Setup

1. Log Collection:
    - Logs from Kubernetes pods are collected by agents like Promtail or Fluent Bit.
2. Log Ingestion:
    - Logs are ingested into Loki, which splits them into chunks and indexes.
3. Storage in MinIO:
    - Chunks (compressed log data) are stored as objects in MinIO.
    - Indexes are stored either in MinIO or another indexing backend like BoltDB or DynamoDB.
4. Querying via Grafana:
    - When you query logs in Grafana, it asks Loki.
    - Loki retrieves the relevant log chunks and metadata from MinIO to fulfill the query.

5. Verifying MinIO Storage

You can check the logs and storage directly in MinIO:

1. Access MinIO UI (if enabled):
    - MinIO typically runs a web interface on port 9000.
    - Access it via http://:9000 or through a Kubernetes service.
2. List Stored Chunks:
    - Once logged in, you’ll see the loki bucket.
    - Inside, you’ll find folders corresponding to different log streams.

6. Useful Commands

- Check Loki-MinIO Connection: Inspect Loki’s logs to see if it’s writing to MinIO successfully:

```
kubectl logs <loki-pod> -n <namespace>
```

- Check MinIO Logs:

```
kubectl logs loki-minio-0 -n <namespace>
```

Sample of Pod's Log

1. loki-minio-0

```
MinIO Object Storage Server
Copyright: 2015-2022 MinIO, Inc.
License: GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2022-09-17T00-09-45Z (go1.18.6 linux/amd64)

Status:         2 Online, 0 Offline.
API: http://10.42.10.212:9000  http://127.0.0.1:9000
Console: http://10.42.10.212:9001 http://127.0.0.1:9001

Documentation: https://docs.min.io

 You are running an older version of MinIO released 2 years ago
 Update: Run `mc admin update`
```

2. loki-write-0

Uploading

```
level=info ts=2024-11-09T22:17:38.46012347Z caller=index_set.go:86 msg="uploading table
loki_index_20023"
level=info ts=2024-11-09T22:17:38.46012908Z caller=index_set.go:107 msg="finished uploading table
loki_index_20023"
level=info ts=2024-11-09T22:17:38.460135521Z caller=index_set.go:185 msg="cleaning up unwanted indexes
from table loki_index_20023"
level=info ts=2024-11-09T22:17:38.460142591Z caller=index_set.go:86 msg="uploading table
loki_index_20034"
level=info ts=2024-11-09T22:17:38.460148231Z caller=index_set.go:107 msg="finished uploading table
loki_index_20034"
level=info ts=2024-11-09T22:17:38.460155091Z caller=index_set.go:185 msg="cleaning up unwanted indexes
from table loki_index_20034"
level=info ts=2024-11-09T22:17:38.473078798Z caller=table_manager.go:171 index-store=boltdb-shipper-
```

```
2022-01-11 msg="handing over indexes to shipper"
```

Flushing Stream

```
level=info ts=2024-11-09T22:17:47.39610939Z caller=flush.go:167 msg="flushing stream" user=fake fp=b766631cb1175681 immediate=false num_chunks=1 labels="{app=\"town-server73\", container=\"town-server\", filename=\"/var/log/pods/prd-ns_town-server73-64f86fc75b-m9xdz_cd0855bf-4e12-4c92-9e1b-2efdfa2da1f5/town-server/0.log\", job=\"prd-ns/town-server73\", namespace=\"prd-ns\", node_name=\"gamep-server17-new\", pod=\"town-server73-64f86fc75b-m9xdz\", stream=\"stdout\"}"
level=info ts=2024-11-09T22:17:47.396172471Z caller=flush.go:167 msg="flushing stream" user=fake fp=6212867a2fd2a163 immediate=false num_chunks=1 labels="{app=\"cattle-cluster-agent\", container=\"cluster-register\", filename=\"/var/log/pods/cattle-system_cattle-cluster-agent-5459b78d66-tsnqr_17c665be-8d3e-43e0-8091-9995ee89e88f/cluster-register/5528.log\", job=\"cattle-system/cattle-cluster-agent\", namespace=\"cattle-system\", node_name=\"gamelog-log02\", pod=\"cattle-cluster-agent-5459b78d66-tsnqr\", stream=\"stderr\"}"
level=info ts=2024-11-09T22:17:47.396315393Z caller=flush.go:167 msg="flushing stream" user=fake fp=91943891befc33f6 immediate=false num_chunks=1 labels="{app=\"town-server75\", container=\"town-server\", filename=\"/var/log/pods/prd-ns_town-server75-7fd49dcf6b-nq472_d5a894ed-1a5d-4816-8cc7-5e3eacc0f865/town-server/0.log\", job=\"prd-ns/town-server75\", namespace=\"prd-ns\", node_name=\"game-server18-new\", pod=\"town-server75-7fd49dcf6b-nq472\", stream=\"stdout\"}"
```

3. loki-backend-0

```
Defaulted container "loki-sc-rules" out of: loki-sc-rules, loki
{"time": "2024-09-05T03:44:49.989787+00:00", "msg": "Starting collector", "level": "INFO"}
{"time": "2024-09-05T03:44:49.990064+00:00", "msg": "No folder annotation was provided, defaulting to k8s-sidecar-target-directory", "level": "WARNING"}
{"time": "2024-09-05T03:44:49.990367+00:00", "msg": "Loading incluster config ...", "level": "INFO"}
{"time": "2024-09-05T03:44:49.991621+00:00", "msg": "Config for cluster api at 'https://10.43.0.1:443' loaded...", "level": "INFO"}
{"time": "2024-09-05T03:44:49.991801+00:00", "msg": "Unique filenames will not be enforced.", "level": "INFO"}
{"time": "2024-09-05T03:44:49.991954+00:00", "msg": "5xx response content will not be enabled.", "level": "INFO"}
```

4. loki-canary-xxxxxx

Confirmation Entry

```
confirmation query result: 1730337782384497148
confirmation query result: 1730337781386565219
confirmation query result: 1730337780384644663
missing websocket entry 1730337790384749414 was found 119.99978837 seconds after it was originally sent
missing websocket entry 1730337791385351900 was found 118.999185884 seconds after it was originally sent
missing websocket entry 1730337792384428548 was found 118.000109216 seconds after it was originally sent
```

Queriying

```
Querying loki for logs with query: http://loki-gateway.log-
ns.svc.cluster.local.:80/loki/api/v1/query_range?start=1730325543384460783&end=1730325563384460783&q
uery=%7Bstream%3D%22stdout%22%2Cpod%3D%22loki-canary-2cn9h%22%7D+&limit=1000
Querying loki for logs with query: http://loki-gateway.log-
ns.svc.cluster.local.:80/loki/api/v1/query_range?start=1730326444384455826&end=1730326464384455826&q
uery=%7Bstream%3D%22stdout%22%2Cpod%3D%22loki-canary-2cn9h%22%7D+&limit=1000
Querying loki for logs with query: http://loki-gateway.log-
ns.svc.cluster.local.:80/loki/api/v1/query_range?start=1730327344384760541&end=1730327364384760541&q
uery=%7Bstream%3D%22stdout%22%2Cpod%3D%22loki-canary-2cn9h%22%7D+&limit=1000
Connecting to loki at ws://loki-gateway.log-
ns.svc.cluster.local.:80/loki/api/v1/tail?query=%7Bstream%3D%22stdout%22%2Cpod%3D%22loki-canary-
2cn9h%22%7D+, querying for label 'pod' with value 'loki-canary-2cn9h'
error reading websocket, will retry in 10 seconds: websocket: close 1011 (internal server error): rpc error: code
= Code(400) desc = max concurrent tail requests limit exceeded, count > limit (11 > 10)
1730337921385246358
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
1730337922384964178
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
1730337923385028676
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

5. loki-gateway

```
10.42.107.139 - - [13/Nov/2024:18:40:21 +0000]  200 "GET
/loki/api/v1/query_range?start=1731520755239250219&end=1731520775239250219&query=%7Bstream%3D
%22stdout%22%2Cpod%3D%22loki-canary-7krp4%22%7D+&limit=1000 HTTP/1.1" 4562 "-" "loki-canary/2.9.4"
"-"
10.42.107.139 - - [13/Nov/2024:18:40:21 +0000]  200 "GET
/loki/api/v1/query_range?start=1731521656239239310&end=1731521676239239310&query=%7Bstream%3D
```

```
%22stdout%22%2Cpod%3D%22loki-canary-7krp4%22%7D+&limit=1000 HTTP/1.1" 4530 "-" "loki-canary/2.9.4"
"-"
10.42.107.139 - - [13/Nov/2024:18:40:21 +0000]  200 "GET
/loki/api/v1/query_range?start=1731522556239246429&end=1731522576239246429&query=%7Bstream%3D
%22stdout%22%2Cpod%3D%22loki-canary-7krp4%22%7D+&limit=1000 HTTP/1.1" 4402 "-" "loki-canary/2.9.4"
"-"
10.42.20.200 - - [13/Nov/2024:18:40:21 +0000]  204 "POST /loki/api/v1/push HTTP/1.1" 0 "-" "promtail/2.9.3" "-"
10.42.1.157 - - [13/Nov/2024:18:40:21 +0000]  204 "POST /loki/api/v1/push HTTP/1.1" 0 "-" "promtail/2.9.3" "-"
10.42.0.43 - - [13/Nov/2024:18:40:21 +0000]  204 "POST /loki/api/v1/push HTTP/1.1" 0 "-" "promtail/2.9.3" "-"
```

6. loki-logs-xxxxx

```
ts=2024-11-15T06:26:54.325615779Z caller=filetarget.go:342 level=error component=logs logs_config=log-
ns/loki msg="failed to tail file, stat failed" error="stat /var/log/pods/log-ns_loki-canary-q2xv5_f36d6fff-d494-
42c5-bb27-e0d715fc6a19/loki-canary/0.log: no such file or directory" filename=/var/log/pods/log-ns_loki-canary-
q2xv5_f36d6fff-d494-42c5-bb27-e0d715fc6a19/loki-canary/0.log
ts=2024-11-15T06:26:54.325695409Z caller=filetarget.go:342 level=error component=logs logs_config=log-
ns/loki msg="failed to tail file, stat failed" error="stat /var/log/pods/log-ns_loki-canary-q2xv5_f36d6fff-d494-
42c5-bb27-e0d715fc6a19/loki-canary/1.log: no such file or directory" filename=/var/log/pods/log-ns_loki-canary-
q2xv5_f36d6fff-d494-42c5-bb27-e0d715fc6a19/loki-canary/1.log
```

7. loki-read

```
ts_results_hit=0 cache_stats_results_download_time=0s cache_result_req=0 cache_result_hit=0
cache_result_download_time=0s
level=info ts=2024-11-15T03:13:17.060839201Z caller=metrics.go:159 component=querier org_id=fake
latency=fast query="count_over_time({stream=\"stdout\", pod=\"loki-canary-hbdnn\"}[15m] offset 8h15m0s)"
query_hash=2449038259 query_type=metric range_type=instant length=0s start_delta=651.118217ms
end_delta=651.118467ms step=0s duration=3.186376ms status=200 limit=1000 returned_lines=0
throughput=0B total_bytes=0B total_bytes_structured_metadata=0B lines_per_second=0 total_lines=0
post_filter_lines=0 total_entries=0 store_chunks_download_time=0s queue_time=16.164835ms splits=0
shards=0 cache_chunk_req=0 cache_chunk_hit=0 cache_chunk_bytes_stored=0 cache_chunk_bytes_fetched=0
cache_chunk_download_time=0s cache_index_req=0 cache_index_hit=0 cache_index_download_time=0s
cache_stats_results_req=0 cache_stats_results_hit=0 cache_stats_results_download_time=0s
cache_result_req=0 cache_result_hit=0 cache_result_download_time=0s
level=info ts=2024-11-15T03:13:17.060892781Z caller=metrics.go:159 component=querier org_id=fake
latency=fast query="count_over_time({stream=\"stdout\", pod=\"loki-canary-hbdnn\"}[15m] offset 8h15m0s)"
```

```
query_hash=2449038259 query_type=metric range_type=instant length=0s start_delta=651.175718ms
end_delta=651.175908ms step=0s duration=3.038683ms status=200 limit=1000 returned_lines=0
throughput=0B total_bytes=0B total_bytes_structured_metadata=0B lines_per_second=0 total_lines=0
post_filter_lines=0 total_entries=0 store_chunks_download_time=0s queue_time=15.818169ms splits=0
shards=0 cache_chunk_req=0 cache_chunk_hit=0 cache_chunk_bytes_stored=0 cache_chunk_bytes_fetched=0
cache_chunk_download_time=0s cache_index_req=0 cache_index_hit=0 cache_index_download_time=0s
cache_stats_results_req=0 cache_stats_results_hit=0 cache_stats_results_download_time=0s
cache_result_req=0 cache_result_hit=0 cache_result_download_time=0s
level=info ts=2024-11-15T03:13:17.063239803Z caller=engine.go:232 component=querier org_id=fake
msg="executing query" type=instant query="count_over_time({stream=\"stdout\", pod=\"loki-canary-
hbdnn\"}[15m] offset 8h15m0s)" query_hash=2449038259
level=info ts=2024-11-15T03:13:17.063264453Z caller=engine.go:232 component=querier org_id=fake
msg="executing query" type=instant query="count_over_time({stream=\"stdout\", pod=\"loki-canary-
hbdnn\"}[15m] offset 8h15m0s)" query_hash=2449038259
level=info ts=2024-11-15T03:13:17.063264073Z caller=engine.go:232 component=querier org_id=fake
msg="executing query" type=instant query="count_over_time({stream=\"stdout\", pod=\"loki-canary-
hbdnn\"}[15m] offset 8h15m0s)" query_hash=2449038259
```

8. grafana

```
logger=ngalert.sender.router rule_uid=fdsom9xvzc7i8f org_id=1 t=2024-09-15T07:30:12.692705191Z
level=info msg="Sending alerts to local notifier" count=1
logger=cleanup t=2024-09-15T07:35:22.30089577Z level=info msg="Completed cleanup jobs"
duration=42.170638ms
logger=plugins.update.checker t=2024-09-15T07:35:22.64731653Z level=info msg="Update check succeeded"
duration=178.444195ms
logger=context userId=0 orgId=0 uname= t=2024-09-15T08:23:53.918612841Z level=info msg="Request
Completed" method=GET path=/dana-na/nc/nc_gina_ver.txt status=302 remote_addr=34.171.200.78
time_ms=0 duration=76.821µs size=29 referer= handler=notfound status_source=server
logger=context userId=0 orgId=0 uname= t=2024-09-15T08:23:53.922143144Z level=info msg="Request
Completed" method=GET path=/tmui/login.jsp status=302 remote_addr=34.171.200.78 time_ms=0
duration=90.711µs size=29 referer= handler=notfound status_source=server
logger=context userId=0 orgId=0 uname= t=2024-09-15T08:23:53.92303671Z level=info msg="Request
Completed" method=GET path=/application.wadl status=302 remote_addr=34.171.200.78 time_ms=0
duration=84.821µs size=29 referer= handler=notfound status_source=server
```

9. loki-grafana-agent-operator-xxxxx

```
level=info ts=2024-07-15T09:26:16.665620737Z controller=node controllerGroup= controllerKind=Node
Node=/game-server13-new namespace= name=game-server13-new reconcileID=cd04fbd3-e1b3-40b2-bd12-
3151ef3bb842 msg=Reconciling
level=info ts=2024-07-15T09:26:16.665650397Z controller=node controllerGroup= controllerKind=Node
Node=/game-server13-new namespace= name=game-server13-new reconcileID=cd04fbd3-e1b3-40b2-bd12-
3151ef3bb842 msg="reconciling node"
level=info ts=2024-07-15T09:26:16.676399643Z controller=node controllerGroup= controllerKind=Node
Node=/game-server13-new namespace= name=game-server13-new reconcileID=cd04fbd3-e1b3-40b2-bd12-
3151ef3bb842 msg="Reconcile successful"
```

10. promtail-xxxxx

```
level=info ts=2024-11-14T05:16:49.938657804Z caller=filetarget.go:313 msg="watching new directory"
directory=/var/log/pods/prd-ns_tokenize-server02-66c8dd64b6-vsdp6_2160bff1-98ad-4a61-8fad-
ce6a786cffb4/tokenize-server
level=info ts=2024-11-14T05:16:49.938674284Z caller=filetarget.go:313 msg="watching new directory"
directory=/var/log/pods/prd-ns_tokenize-server02-66c8dd64b6-nkzdj_7dc38fae-1145-445e-b63e-
802cc6fe7c79/tokenize-server
level=info ts=2024-11-14T05:16:49.938766216Z caller=tailer.go:145 component=tailer msg="tail routine:
started" path=/var/log/pods/prd-ns_tokenize-server02-66c8dd64b6-vsdp6_2160bff1-98ad-4a61-8fad-
ce6a786cffb4/tokenize-server/0.log
ts=2024-11-14T05:16:49.938800927Z caller=log.go:168 level=info msg="Seeked /var/log/pods/prd-
ns_tokenize-server02-66c8dd64b6-vsdp6_2160bff1-98ad-4a61-8fad-ce6a786cffb4/tokenize-server/0.log -
&{Offset:0 Whence:0}"
level=info ts=2024-11-14T05:16:49.938803937Z caller=tailer.go:145 component=tailer msg="tail routine:
started" path=/var/log/pods/prd-ns_tokenize-server02-66c8dd64b6-nkzdj_7dc38fae-1145-445e-b63e-
802cc6fe7c79/tokenize-server/0.log
ts=2024-11-14T05:16:49.938815617Z caller=log.go:168 level=info msg="Seeked /var/log/pods/prd-
ns_tokenize-server02-66c8dd64b6-nkzdj_7dc38fae-1145-445e-b63e-802cc6fe7c79/tokenize-server/0.log -
&{Offset:0 Whence:0}"
ts=2024-11-15T03:57:24.36579026Z caller=log.go:168 level=info msg="Re-opening moved/deleted file
/var/log/pods/prd-ns_tokenize-server02-66c8dd64b6-vsdp6_2160bff1-98ad-4a61-8fad-ce6a786cffb4/tokenize-
server/0.log ..."
ts=2024-11-15T03:57:24.365882122Z caller=log.go:168 level=info msg="Successfully reopened
/var/log/pods/prd-ns_tokenize-server02-66c8dd64b6-vsdp6_2160bff1-98ad-4a61-8fad-ce6a786cffb4/tokenize-
server/0.log"
```

Logging Data Backup Simulation with Docker

1. Prerequisites

- Install Docker and Docker Compose:
    - Install Docker
    - Install Docker Compose

2. Create a Docker Compose File

Set up a docker-compose.yaml file to simulate the environment.

```
version: "3.9"

services:
  loki:
    image: grafana/loki:latest
    ports:
      - "3100:3100"
    command: -config.file=/etc/loki/local-config.yaml
    volumes:
      - ./loki-config.yaml:/etc/loki/local-config.yaml:ro
    depends_on:
      - minio

  minio:
    image: minio/minio:latest
    ports:
      - "9000:9000"
    environment:
      MINIO_ROOT_USER: enterprise-logs
      MINIO_ROOT_PASSWORD: supersecret
    command: server /data
    volumes:
      - minio-data:/data

  grafana:
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=admin
    depends_on:
      - loki

volumes:
  minio-data:
```

3. Create the Loki Configuration

Create a loki-config.yaml file in the same directory as your docker-compose.yaml:

```
auth_enabled: false

server:
  http_listen_port: 3100

ingester:
  wal:
    enabled: false
  chunk_idle_period: 5m
  max_chunk_age: 1h
  chunk_target_size: 1048576
  lifecycler:
    ring:
      kvstore:
        store: inmemory
    replication_factor: 1

schema_config:
  configs:
    - from: 2022-01-01
      store: boltdb-shipper
      object_store: s3
      schema: v12
      index:
        prefix: loki_index_
        period: 24h

storage_config:
  boltdb_shipper:
    active_index_directory: /data/loki/boltdb-shipper-active
    shared_store: s3
    cache_location: /data/loki/boltdb-shipper-cache
  aws:
    s3: http://minio:9000
    bucketnames: chunks
    access_key_id: enterprise-logs
    secret_access_key: supersecret
    s3forcepathstyle: true

limits_config:
  retention_period: 744h
```

4. Start the Environment

Run the following command in the directory containing your files:

```
docker-compose up -d
```

This will spin up:

- MinIO on http://localhost:9000 (Access Key: enterprise-logs, Secret Key: supersecret)
- Loki on http://localhost:3100
- Grafana on http://localhost:3000 (User: admin, Password: admin)

5. Add Buckets to MinIO

Using AWS

Configure AWS:

```
aws configure
```

Or

```
aws configure set aws_access_key_id enterprise-logs --profile minio
aws configure set aws_secret_access_key supersecret --profile minio
```

Create buckets:

```
aws --endpoint-url http://localhost:9000 s3 mb s3://chunks --region us-east-1 --profile minio
aws --endpoint-url http://localhost:9000 s3 mb s3://rules --region us-east-1 --profile minio
```

Show bucket list:

```
aws --endpoint-url http://localhost:9000 s3 ls
```

The output should like this:

```
2024-11-15 18:52:53 chunks
2024-11-15 18:54:49 rules
```

6. Configure Grafana to Use Loki

1. Open Grafana: http://localhost:3000.
2. Log in with the default credentials (admin / admin).
3. Add Loki as a data source:

- Go to Configuration > Data Sources > Add data source.
- Select Loki.
- Set the URL to http://loki:3100.
- Click Save & Test.

7. Send Test Logs

To simulate log ingestion:

- Install and run Promtail or send HTTP POST requests to Loki’s /loki/api/v1/push.

Example Promtail Docker Compose service:

```
promtail:
  image: grafana/promtail:latest
  command: -config.file=/etc/promtail/config.yml
  volumes:
    - ./promtail-config.yaml:/etc/promtail/config.yml:ro
  depends_on:
    - loki
```

Promtail config.yaml:

```
server:
  http_listen_port: 9080
clients:
  - url: http://loki:3100/loki/api/v1/push
scrape_configs:
  - job_name: system
    static_configs:
      - targets:
          - localhost
        labels:
          job: system
          host: localhost
          __path__: /var/log/*.log
```

To run only the Promtail service, you can use the following command:

```
docker-compose up -d promtail
```

If you want to stop and remove the container in one command, you can do:

```
docker-compose down promtail
```

8. Check Logs in Grafana

1. Open Grafana.
2. Go to Explore.
3. Choose Loki as the data source and query logs using the built-in query editor.

Cleanup

To stop and remove the environment:

```
docker-compose down -v
```

Backup Loki-Minio Data into AWS S3

Configure AWS Credential:

```
aws configure
```

Show list of configuration

```
aws configure list
```

Check Bucket List on Loki-Minio

```
aws --endpoint-url http://<server-ip>:9000 s3 ls
```

Show Bucket Contents Recursively

```
aws --endpoint-url http://localhost:9000 s3 ls s3://chunks --recursive
```

Synchronize

```
aws s3 sync s3://<minio-endpoint>/loki-data s3://my-loki-backup --endpoint-url http://<minio-endpoint>
```

Fluent-Bit Synchronization Database

Settings

```
[INPUT]
    Name              tail
    Path             /var/external_logs/*.log
    Tag              local.*
    Refresh_Interval  5
    DB               /fluent-bit/logs/fluent-bit.db
    DB.Sync           Normal
    Parser            json
    Rotate_Wait       30
```

Database DDL

```
-- in_tail_files definition
CREATE TABLE in_tail_files (
  id INTEGER PRIMARY KEY, name TEXT NOT NULL,
  offset INTEGER, inode INTEGER, created INTEGER,
  rotated INTEGER DEFAULT 0
);
```

Fluent-Bit Generate Log When Log is Written

Docker Compose

```
...

  fluent-bit:
    build:
      context: ./fluent-bit
    volumes:
      - ./fluent-bit-logs:/var/log
      - ./fluent-bit-src-logs:/var/source_logs
      - ./fluent-bit-dst-logs:/var/destination_logs
      - ./fluent-bit-db:/fluent-bit/logs
    depends_on:
      - loki
      - fluentd
      - minio
```

Configuration File

```
[SERVICE]
    Flush        1
    Log_Level    info
    Daemon       off

[INPUT]
    Name              tail
    Path             /var/source_logs/*.log
    Tag              local.*
    Refresh_Interval  5
    DB               /fluent-bit/logs/fluent-bit.db
    DB.Sync           Normal
    Parser            json
    Rotate_Wait       30

[OUTPUT]
    Name file
    Match local.*
    Path /var/destination_logs
    Format plain

[OUTPUT]
    Name              loki
    Match             local.*
    Host              loki
    Port              3100
    uri               http://loki:3100/loki/api/v1/push
    Label_Keys        $job, $instance

[OUTPUT]
    name  stdout
    match *
```

Simulation

Build the container with command:

```
docker-compose up -d fluent-bit
```

Write log to a log file with command:

```
echo "$(date) Test log entry" >> test.log
```

Error Debugging

Kubernetes Command Not Found

```
2024-11-22 06:00:02 - Script execution started.
/data/scripts/log_cleaner.sh: line 60: kubectl: command not found
2024-11-22 06:00:02 - No pods found matching the patterns in namespace prd-ns.
```

Solution:

```
which kubectl
```

Replace `kubectl` command with absolute path of `kubectl` from the command. It will be like this.

```
/usr/local/bin/kubectl
```

Connection Refused

```
E1125 06:00:02.151541  190966 memcache.go:265] couldn't get current server API group list: Get
"http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused
The connection to the server localhost:8080 was refused - did you specify the right host or port?
2024-11-25 06:00:02 - No pods found matching the patterns in namespace prd-ns.
```

Solution:

Describe Kubernetes IP and port on global environment variables. It will be like this.

```
export KUBECONFIG=~/.kube/config
export KUBERNETES_SERVICE_HOST=192.168.1.10
export KUBERNETES_SERVICE_PORT=443
```

Promtail Restore Log Data Into Loki

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0
clients:
  - url: http://loki:3100/loki/api/v1/push
    batchsize: 512000
    batchwait: 1s
positions:
  filename: /tmp/positions.yaml
scrape_configs:
  - job_name: system
    static_configs:
      - targets:
          - localhost
        labels:
          job: varlogs
          __path__: /var/backups/**/*.log
    pipeline_stages:
      - json:
          expressions:
            log: log
            stream: stream
            time: time
      - regex:
          source: filename
          expression: '.*/(?P<app_name>[^_]+)_.+\.log$'
      - labels:
          stream: stream
          app: app_name
      - timestamp:
          source: time
          format: RFC3339Nano
  - output:
      format: json
      source: log
```

Restore Log Data from AWS S3 into Loki

Docker Compose

```
version: '3.8'
services:
  s3-downloader:
    build: .
    volumes:
      - ./logs:/logs
      - ./.s3cfg:/root/.s3cfg
    entrypoint: ["/usr/local/bin/download_logs.sh"]
```

Dockerfile

```
# Use an official lightweight image with bash
FROM alpine:3.17

# Install necessary packages (bash, s3cmd, etc.)
RUN apk add --no-cache bash py3-pip && pip install s3cmd

# Copy the Bash script into the container
COPY download_logs.sh /usr/local/bin/download_logs.sh

# Set executable permissions on the script
RUN chmod +x /usr/local/bin/download_logs.sh

# Set the default command to run the Bash script
# CMD ["/usr/local/bin/download_logs.sh"]
```

AWS S3 Credential

```
[default]
access_key = xyz
secret_key = ***
```

Log Data Extractor

```
#!/bin/bash
set -e

# Configure s3cmd if not already configured
if [ ! -f "$HOME/.s3cfg" ]; then
  echo "s3cmd configuration not found."
  exit 1
fi

# S3 Bucket and path
LOG_DATE="20241118"
NAMESPACE="app1-ns"
BUCKET_NAME="log-data-bucket"
REMOTE_LOGS_PATH="fluentd_log/$LOG_DATE/$NAMESPACE"
LOCAL_DESTINATION="/logs"

# Array of patterns to include
PATTERNS=(
    "metadata-server-*.tar.gz"
    "admin-server*.tar.gz"
)

# Create local destination folder if it doesn't exist
mkdir -p "$LOCAL_DESTINATION"

# Build the include and exclude rules dynamically
INCLUDE_EXCLUDE=""
for PATTERN in "${PATTERNS[@]}"; do
```

```
  INCLUDE_EXCLUDE+="--include=\"$PATTERN\" "
done
INCLUDE_EXCLUDE+="--exclude=\"*\""
echo $INCLUDE_EXCLUDE

# Logic to download logs
echo "Downloading log files from s3://$BUCKET_NAME/$REMOTE_LOGS_PATH to $LOCAL_DESTINATION"
# eval s3cmd sync -v $INCLUDE_EXCLUDE s3://$BUCKET_NAME/$REMOTE_LOGS_PATH "$LOCAL_DESTINATION"

echo "Log files downloaded successfully to $LOCAL_DESTINATION"

# Extract each .tar.gz file into its own folder
echo "Extracting downloaded files..."
for FILE in "$LOCAL_DESTINATION/$NAMESPACE"/*.tar.gz; do
  if [ -f "$FILE" ]; then
    # Create a folder named after the file (without .tar.gz)
    FOLDER_NAME="${FILE%.tar.gz}"
    mkdir -p "$FOLDER_NAME"

     # Get list of files in the tar.gz archive
    ARCHIVE_CONTENTS=$(tar -tzf "$FILE" | grep -v '/$')
    SUCCESS=true

    # Check each file in the archive exists in the extraction folder
    for ARCHIVE_FILE in $ARCHIVE_CONTENTS; do
      FULL_PATH="$LOCAL_DESTINATION/$NAMESPACE/$ARCHIVE_FILE"
      if [ ! -f $FULL_PATH ]; then
        echo "Error: File $ARCHIVE_FILE is missing in $FOLDER_NAME"
        SUCCESS=false
        break
      fi
    done

    if $SUCCESS; then
      echo "All files successfully extracted for $FILE."
    else
      # Extract the tar.gz file into the folder
      tar -xzvf "$FILE" -C "$LOCAL_DESTINATION/$NAMESPACE"

      echo "Extracted $FILE into $FOLDER_NAME"
    fi
  fi
done

echo "All files downloaded and extracted successfully!"
```

Export Log Index Data from MinIO

Download `mc`:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc
sudo mv mc /usr/local/bin/
```

Check Log Index Data

```
./mc alias set minio http://minio:9000 enterprise-logs supersecret
./mc ls minio
./mc find minio/chunks --older-than 120d --print "{time} {base}"
```

Export Log Index Data

```
./mc find minio/chunks/fake --older-than 1d --exec "./mc cp {} /home/opsuser/ahmad/minio/backup/{}"
./mc find minio/chunks/index --older-than 1d --exec "./mc cp {} /home/opsuser/ahmad/minio/backup/{}"
```

Zip Log Index Data

```
tar -czvf backup-$(date +%Y%m%d).tar.gz ./backup/minio/chunks
```

Download Zip File from Server

```
scp -i ~/.ssh/cert.crt root@127.0.0.1:/home/root/minio/backup-20241121.tar.gz /Users/home/Documents/
```

Data Mover from Splitted Disk

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: data-mover
  namespace: loki
spec:
  serviceName: "data-mover-service"
  replicas: 1
  selector:
    matchLabels:
      app: data-mover
  template:
    metadata:
      labels:
        app: data-mover
      namespace: loki
    spec:
      containers:
        - name: data-mover-container
          image: ubuntu:latest
          command: [ "sleep", "infinity" ]
          volumeMounts:
            - mountPath: /export-0
              name: export-0
            - mountPath: /export-1
              name: export-1
      volumes:
        - name: export-0
          persistentVolumeClaim:
            claimName: export-0-loki-minio-0
        - name: export-1
          persistentVolumeClaim:
```

```
          claimName: export-1-loki-minio-0
```

Persistent Volume Claim

```
kubectl get pvc -n loki export-0-loki-minio-0 -o yaml > export-0-loki-minio-0.yaml
kubectl apply -f export-0-loki-minio-0.yaml
kubectl get pvc -n loki
kubectl cp /home/root/backup/chunks/ loki-minio-test-0:/export-test-0/chunks/ -n loki
kubectl cp backup/chunks loki/data-mover-0:/export-0/chunks/
```

Install s3cmd on Amazon Linux

To install `s3cmd` on Amazon Linux, you can follow these steps:

If you're encountering an error where `s3cmd` cannot be found, it might be due to a few reasons. Here are some steps to troubleshoot and resolve the issue:

1. **Update Package List**: Ensure your package list is up to date before attempting to install `s3cmd`.
   ```
   sudo yum update -y
   ```
2. **Install s3cmd from Source**: As a fallback, you can download and install `s3cmd` from its source on GitHub:
   ```
   sudo yum install -y python3-pip
   sudo pip3 install s3cmd
   ```
3. **Manual Installation**: Alternatively, you can download `s3cmd` manually and install it:
   ```
   wget https://github.com/s3tools/s3cmd/archive/refs/tags/v2.2.0.tar.gz
   tar -xvf v2.2.0.tar.gz
   cd s3cmd-2.2.0
   sudo python3 setup.py install
   ```

This should install `s3cmd` on your Amazon Linux instance and set it up for use with your AWS S3 buckets.