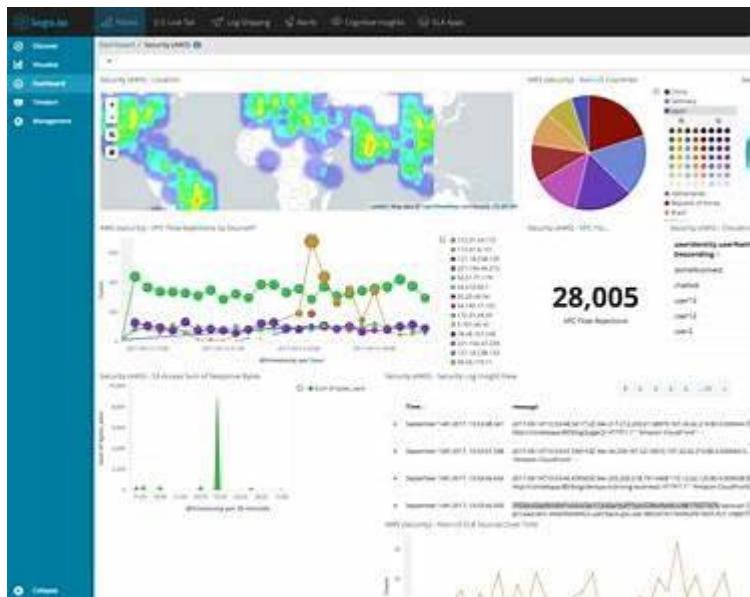
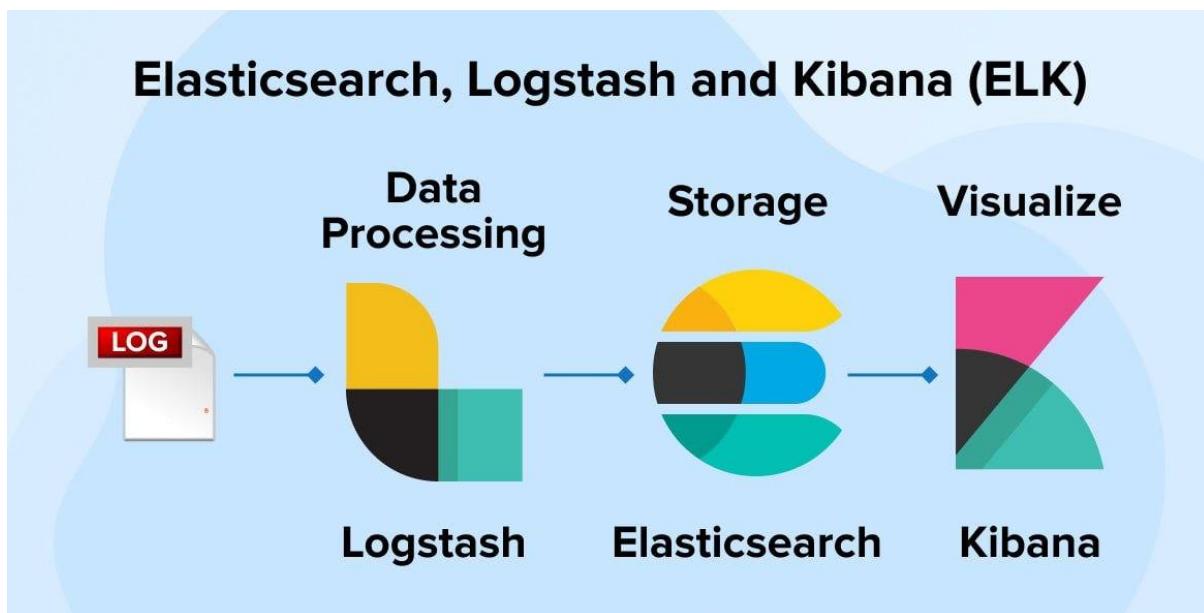


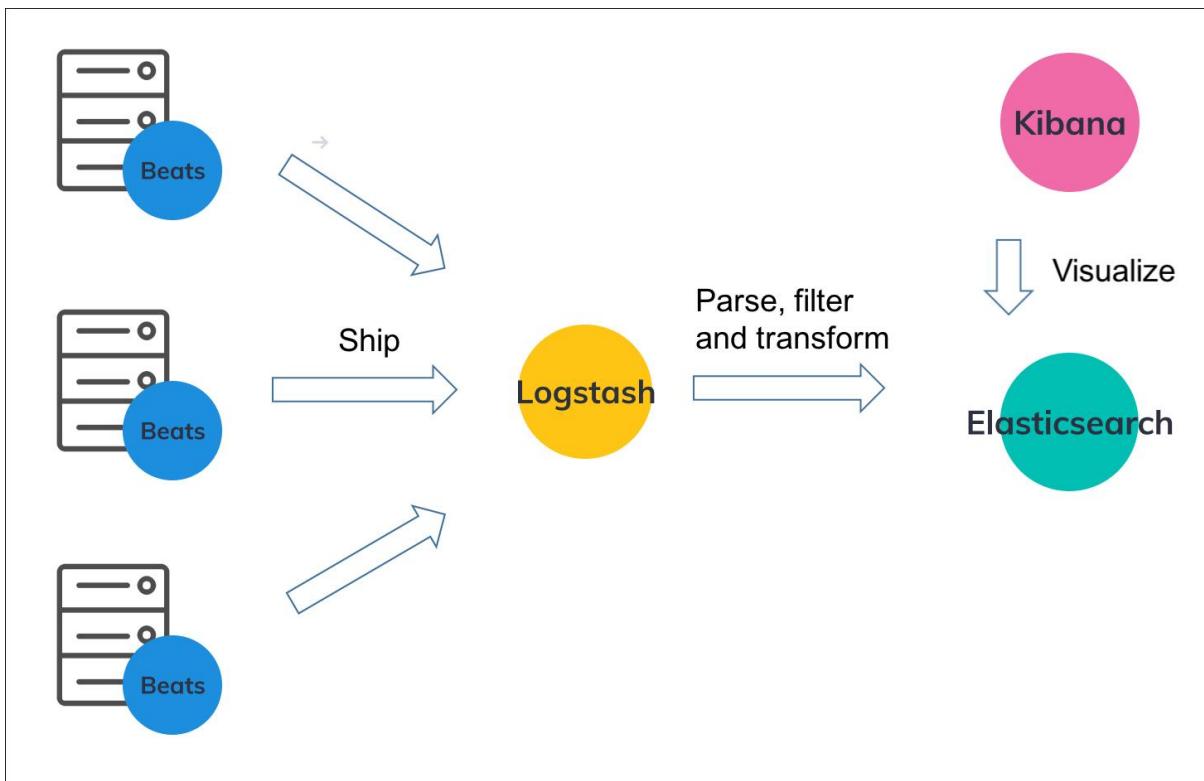
If you are **NOT** using **Kubernetes** and want to **view server logs in a clean, powerful UI**, here are the **best practical options** (from simple → enterprise-grade). I'll clearly tell you **what to choose and why** 👇

● Best Overall (Most Popular): ELK Stack



Elasticsearch, Logstash and Kibana (ELK)





◆ Components

- **Filebeat** → ships logs
- **Elasticsearch** → stores & searches logs
- **Kibana** → beautiful UI dashboard

✓ Why it's great

- ✓ Web UI with search, filters, charts
- ✓ Works perfectly on **VM / bare metal / EC2**
- ✓ Supports **system logs, app logs, Nginx, Apache, MySQL**
- ✓ Free (OpenSearch alternative available)

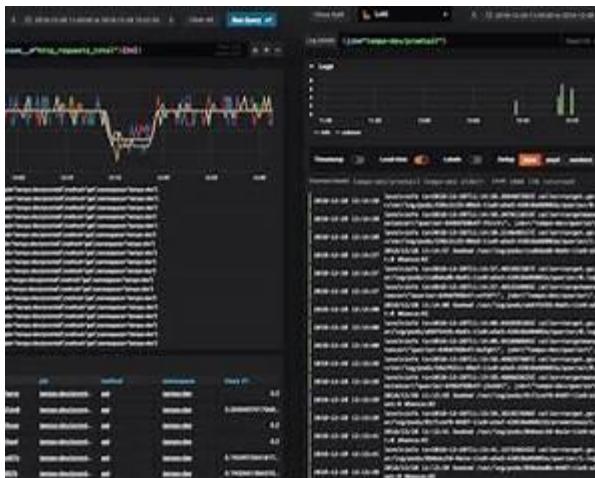
📌 Best for

Production servers, long-term log analysis, alerts

💻 UI Features

- Live log streaming
- Search by hostname, date, severity
- Graphs & dashboards

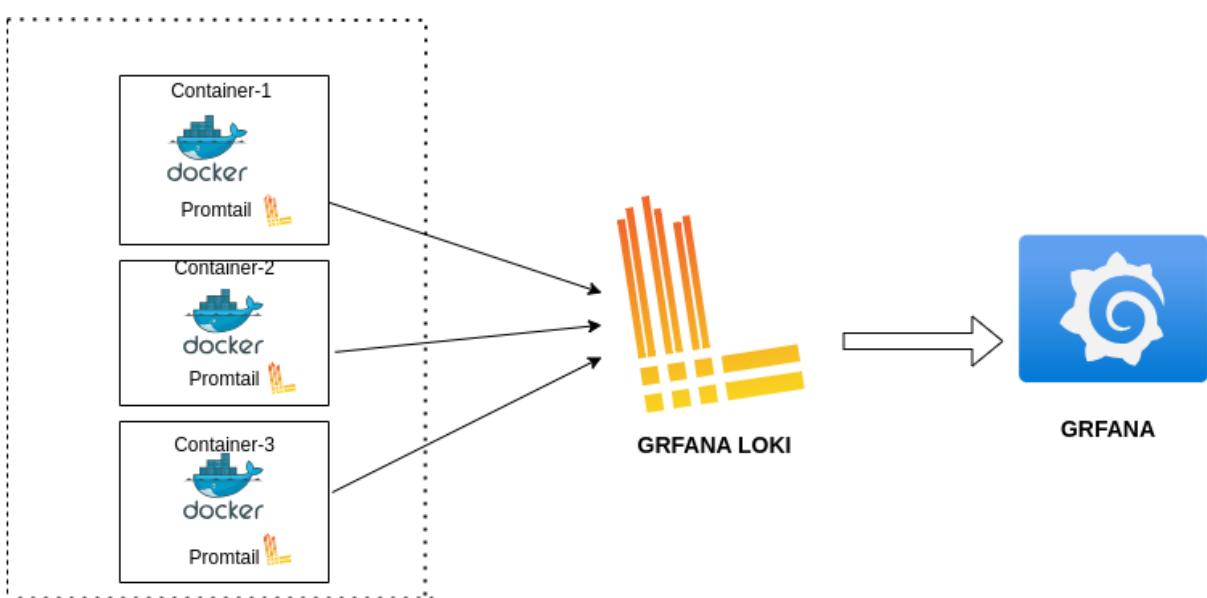
● Lightweight & Modern: Grafana + Loki



Logs Drilldown

A new queryless experience for Loki

The interface shows a search bar for 'Data source' set to 'Explore Logs' and 'service_name' set to 'mimir-ingester'. Below are two time-series charts: one for 'level' (info, error) and another for 'cluster' (eu-west-1, us-east-1, us-east-2, us-west-1). On the right, there are additional panels for 'Labels', 'Fields', and 'Patterns'.



◆ Components

- **Promtail** → collects logs
- **Loki** → stores logs
- **Grafana** → UI

✓ Why it's great

- ✓ Much **lighter** than ELK
- ✓ Very fast
- ✓ Same UI for **metrics + logs**
- ✓ Simple setup

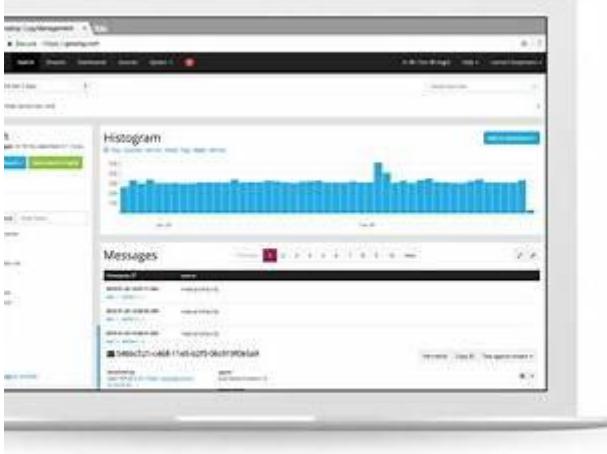
📌 Best for

DevOps, system monitoring, low resource servers

🌐 UI Features

- Live logs
- Label-based filtering
- Time correlation with CPU/RAM metrics

● Super Simple (Single Server): Graylog



The screenshot shows the Graylog web interface. At the top, there's a navigation bar with links for Dashboards, Sources, TypeDefs, and Help. Below the navigation is a toolbar with buttons for 'Update in background', 'Fullscreen', and a gear icon.

The main area displays two log entries:

- 1h MySQL errors**: A few seconds ago, 81 messages. A chart titled '24h MySQL errors' shows message counts from 18:00 to 06:00.
- 1h kernel messages**: A few seconds ago, 1,054 messages. A chart titled '24h kernel messages' shows message counts from 18:00 to 06:00.

Below these are sections for stream management:

- All events** index set *Graylog Events*: Stream containing all events created by Graylog. 0 messages/second. No configured rules. Buttons: Manage Rules, Manage Outputs, Manage Alerts, **Pause Stream**, More Actions.
- All messages** index set *Default index set* [Default]: Stream containing all messages. 0 messages/second. The default stream contains all messages. Buttons: Manage Rules, Manage Outputs, Manage Alerts, **Pause Stream**, More Actions.
- All system events** index set *Graylog System Events*: Stream containing all system events created by Graylog. 0 messages/second. No configured rules. Buttons: Manage Rules, Manage Outputs, Manage Alerts, **Pause Stream**, More Actions.
- DHCP** index set *DHCP*: Stream containing DHCP Logs. 0 messages/second. Must match all of the 1 configured stream rule. [Hide stream rules](#). A note says: Field `SourceModule` must match exactly `DHCP_IN`.
- Event Logs** index set *Event Log*: Stream containing Event Logs. 0 messages/second. Must match all of the 1 configured stream rule. [Hide stream rules](#). A note says: Field `SourceModule` must match exactly `EventLog_IN`.

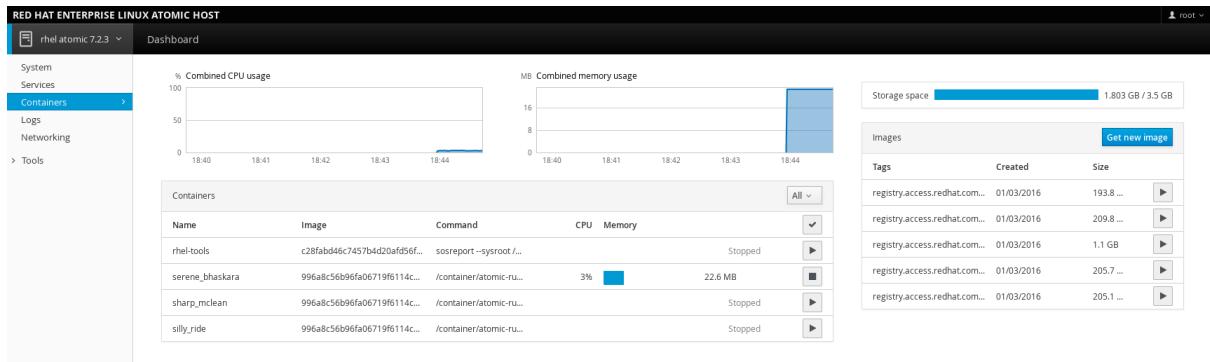
✓ Why it's great

- ✓ Easier than ELK
- ✓ Clean UI
- ✓ Built-in alerting
- ✓ Centralized logging

📌 Best for

Teams who want **ELK power without ELK complexity**

🟡 Quick & Local (Not Centralized): Cockpit



Why it's useful

- ✓ Built into Linux
- ✓ View journalctl logs in browser
- ✓ Zero setup

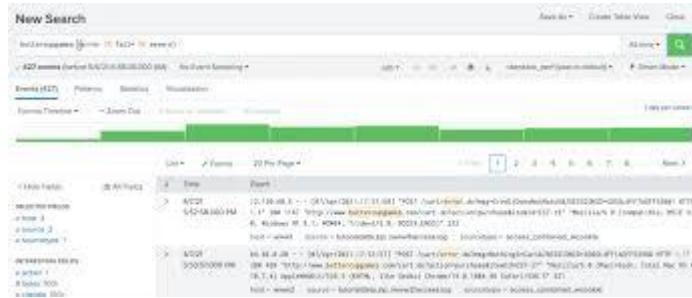
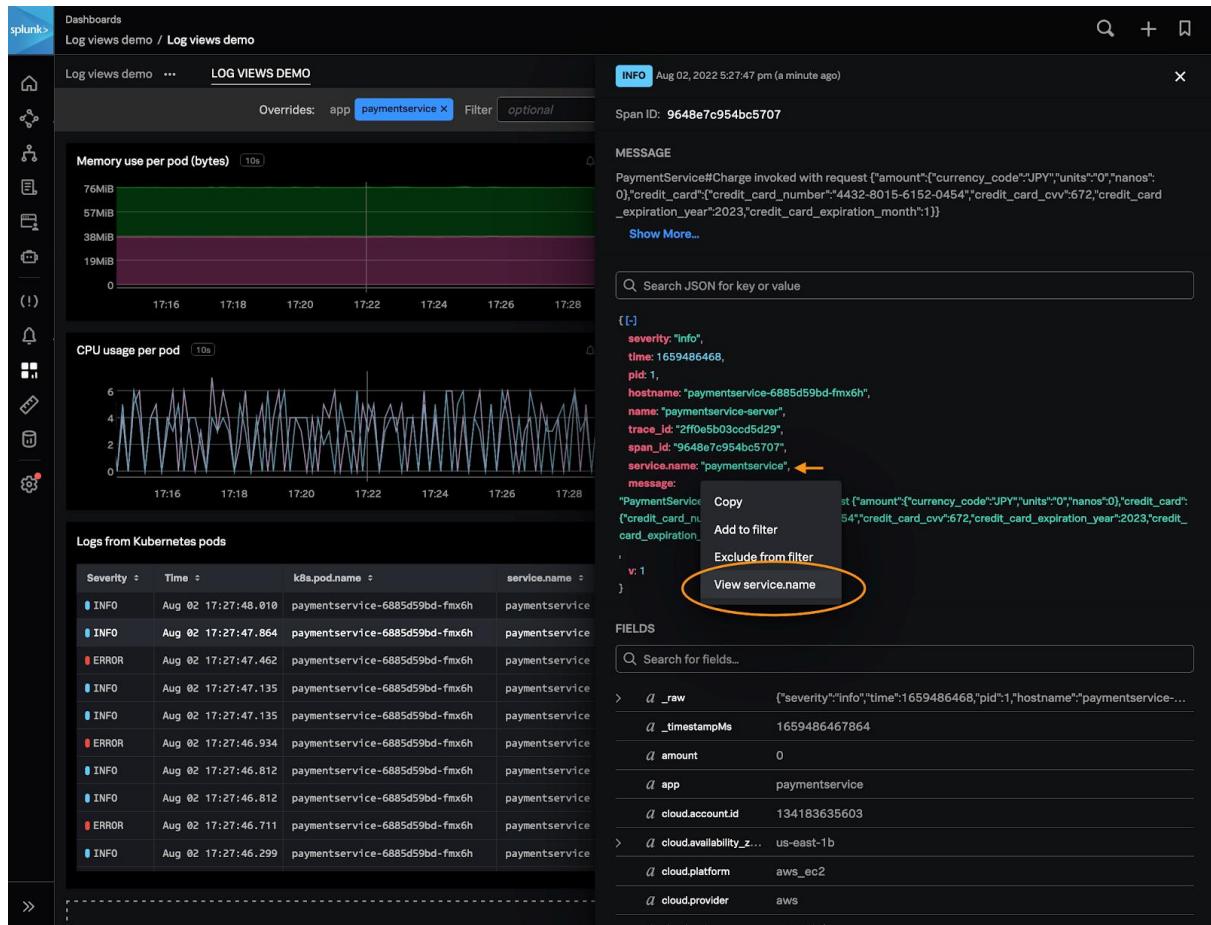
Limitations

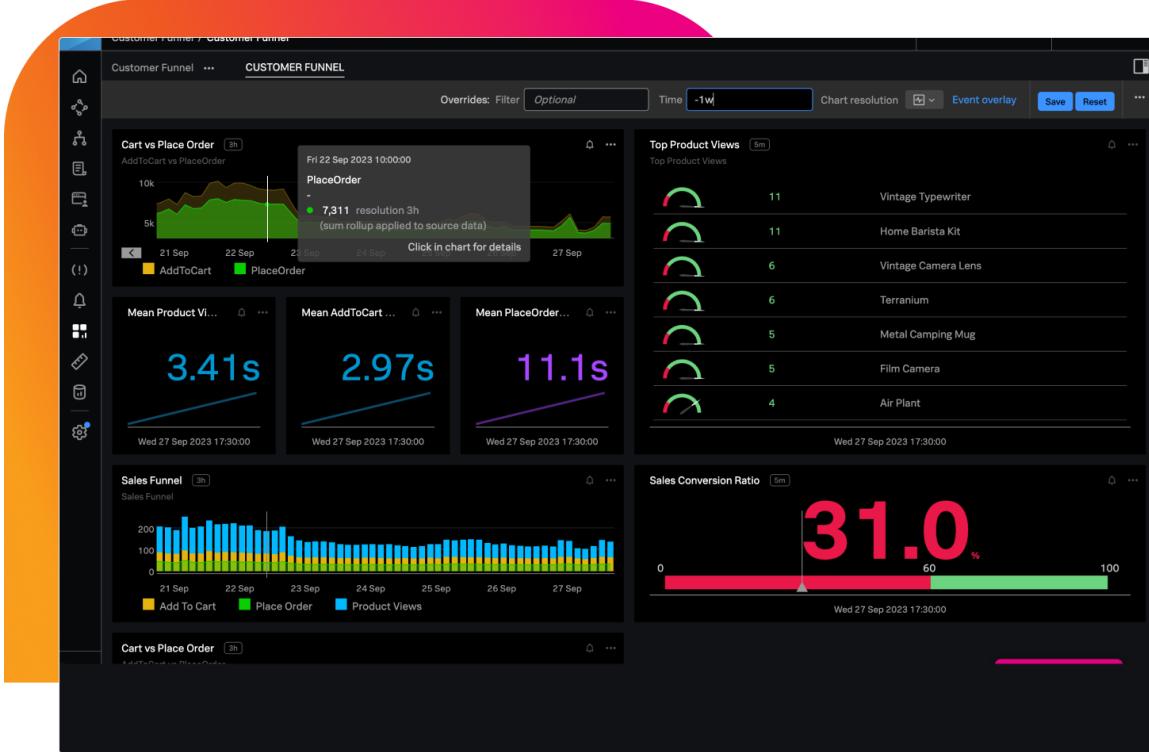
- Single server only
- No long-term log analysis

Best for

Quick troubleshooting on one machine

Commercial (Very Powerful): Splunk





✓ Why it's great

- ✓ Best UI in industry
- ✓ AI-powered searches
- ✓ Enterprise-grade alerts

✗ Downside

- 💰 Expensive (free tier limited)

🏆 My STRONG Recommendation (Based on Your Use)

Since you're:

- On **Ubuntu / VM / EC2**
- Doing **system monitoring**
- Want **clean UI**

👉 Choose this combo:

✓ Grafana + Loki (Best balance)

or

✓ ELK Stack (If you want enterprise-level power)

Want a Ready-to-Install Setup?

I can give you:

- **One-command install**
- **Filebeat / Promtail config**
- **Prebuilt dashboards**
- **Alert rules**
- **S3 log archival (since you already use AWS)**

 Tell me:

- 1** Single server or multiple servers?
- 2** Ubuntu version?
- 3** Logs type (system / app / nginx / mysql)?

- One-command install Filebeat / Promtail config Prebuilt dashboards Alert rules
 S3 log archival (since you already use AWS) **1** Single server or multiple servers? for both
2 Ubuntu version? 22.04 **3** Logs type (system / app / nginx / mysql)? for all

Perfect 

Below is a **production-ready, copy-paste solution** for **Ubuntu 22.04**, covering:

- One-command install**
- Promtail (logs) + Filebeat (optional)**
- Prebuilt dashboards**
- Alert rules (CPU / RAM / log errors)**
- S3 log archival**
- Works for BOTH single & multiple servers**

I'll give you **Grafana + Loki** (best UI + lightweight) and **S3 backup**.

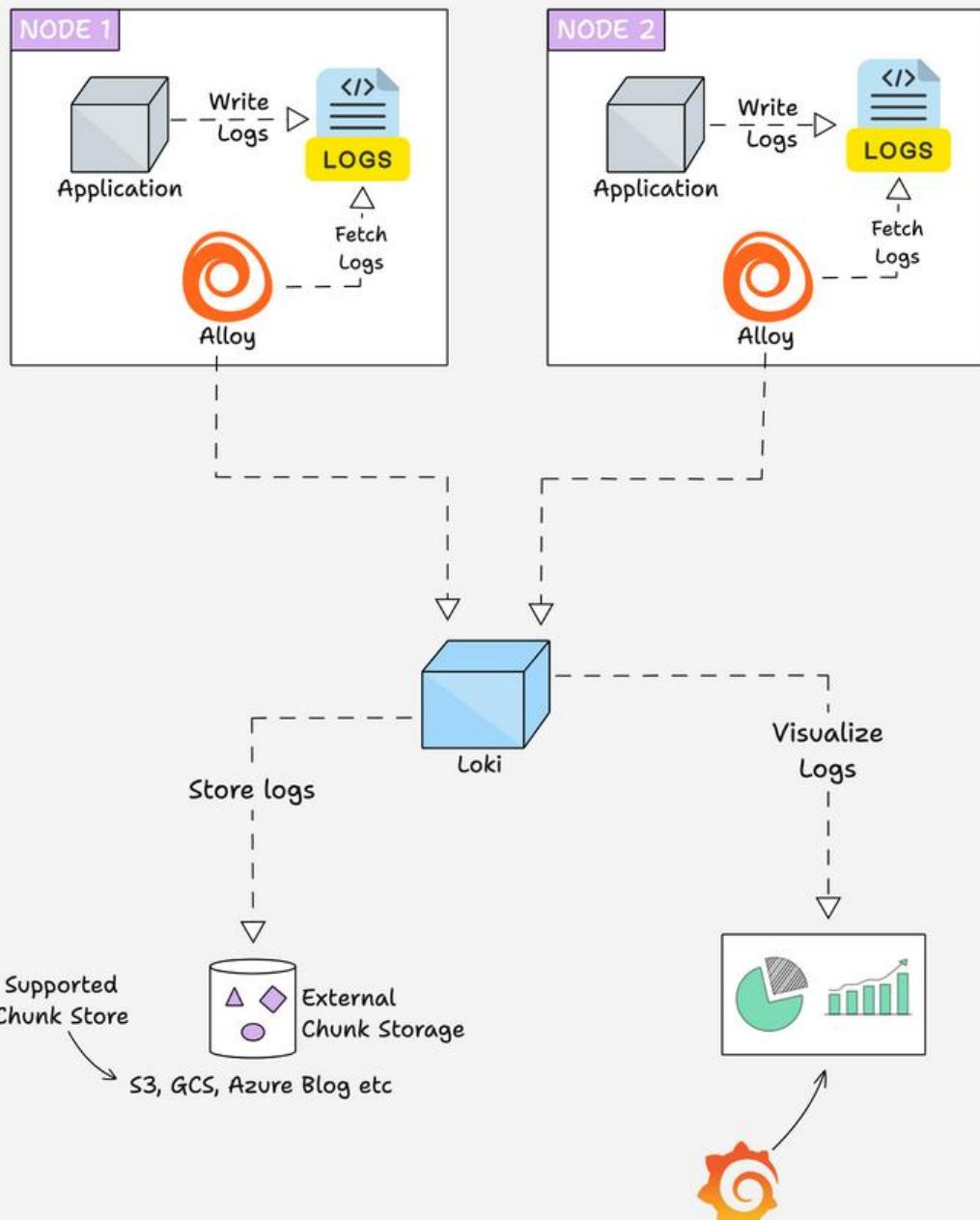
Recommended Stack (No Kubernetes)

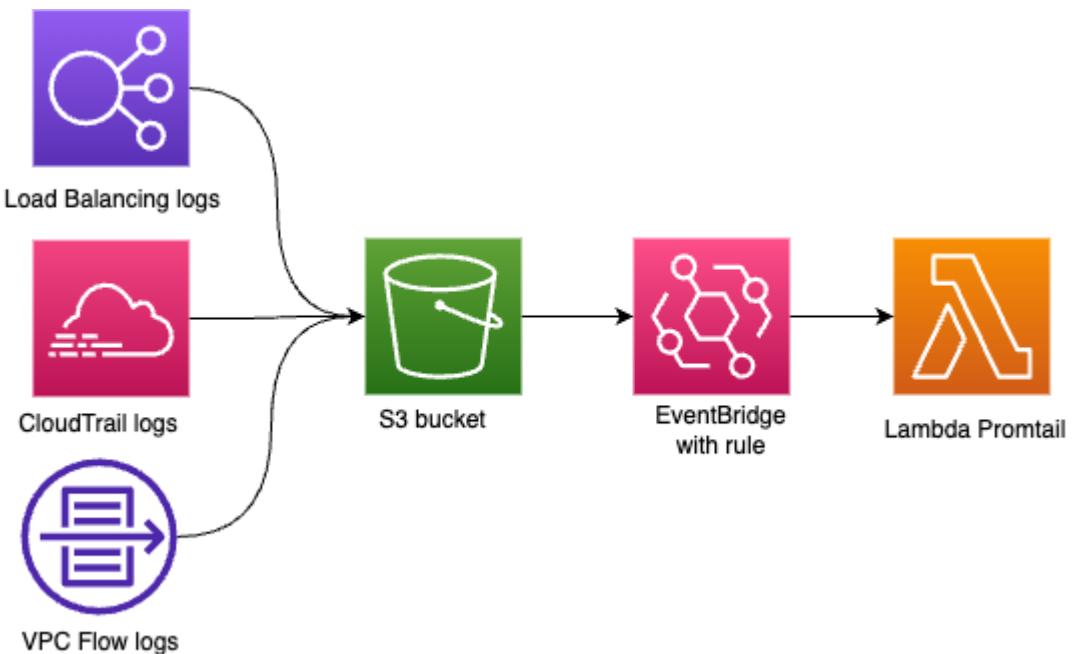
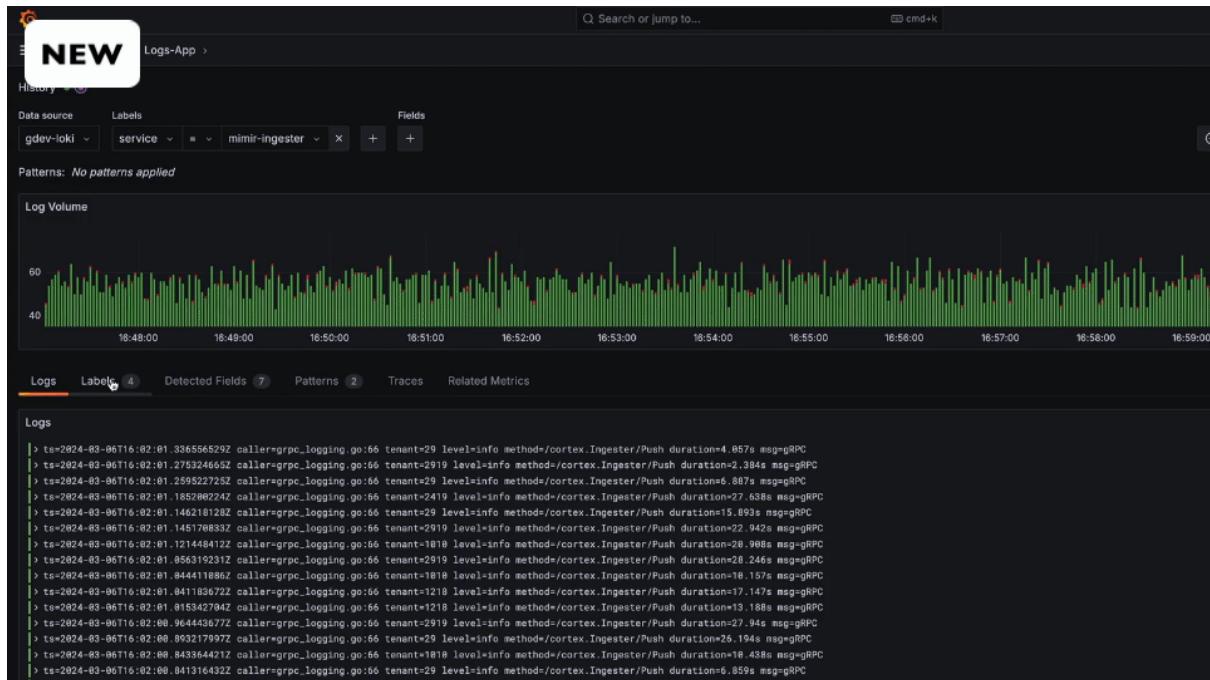


LOKI ARCHITECTURE



devopscube.com





◆ Components

Component Purpose

Grafana UI (logs + dashboards)

Loki Log storage

Promtail Log collector

Node Exporter CPU / RAM metrics

Component	Purpose
-----------	---------

AWS S3	Long-term log backup
--------	----------------------

STEP 1 — ONE-COMMAND INSTALL (Ubuntu 22.04)

Run **ONCE per server:**

```
curl -fsSL https://raw.githubusercontent.com/grafana/loki/main/tools/install.sh | sudo bash
```

Then install **Promtail + Grafana + Node Exporter:**

```
sudo apt update
```

```
sudo apt install -y grafana promtail prometheus-node-exporter awscli
```

Enable services:

```
sudo systemctl enable --now grafana-server promtail prometheus-node-exporter
```

STEP 2 — PROMTAIL CONFIG (SYSTEM + APP + NGINX + MYSQL)

Edit config:

```
sudo nano /etc/promtail/config.yml
```

PRODUCTION CONFIG

server:

```
http_listen_port: 9080
```

```
grpc_listen_port: 0
```

positions:

```
filename: /var/log/positions.yaml
```

clients:

```
- url: http://localhost:3100/loki/api/v1/push
```

scrape_configs:

```
- job_name: system
```

static_configs:

```
- targets:
```

- localhost
 - labels:
 - job: system
 - host: \${HOSTNAME}
 - __path__: /var/log/syslog
- job_name: auth
 - static_configs:
 - targets:
 - localhost
 - labels:
 - job: auth
 - __path__: /var/log/auth.log
- job_name: nginx
 - static_configs:
 - targets:
 - localhost
 - labels:
 - job: nginx
 - __path__: /var/log/nginx/*.log
- job_name: mysql
 - static_configs:
 - targets:
 - localhost
 - labels:
 - job: mysql
 - __path__: /var/log/mysql/*.log
- job_name: app

`static_configs:`

- targets:

- localhost

`labels:`

`job: app`

`__path__: /var/www/*storage/logs/*.log`

`Restart:`

`sudo systemctl restart promtail`

🌐 STEP 3 — GRAFANA UI ACCESS

`http://<SERVER_IP>:3000`

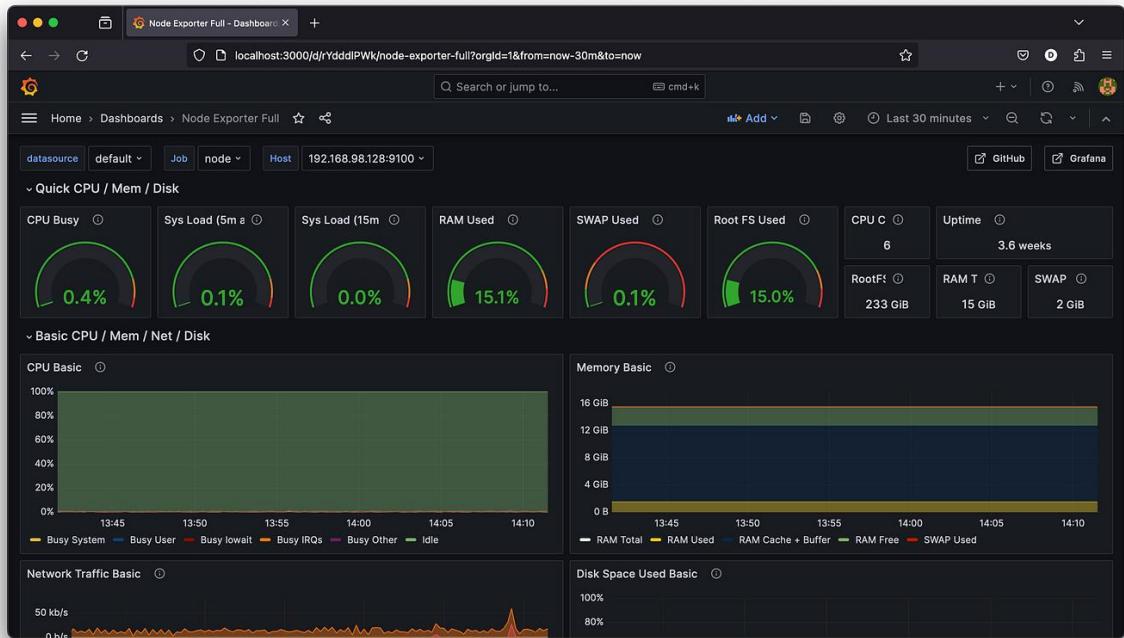
🔑 Default login:

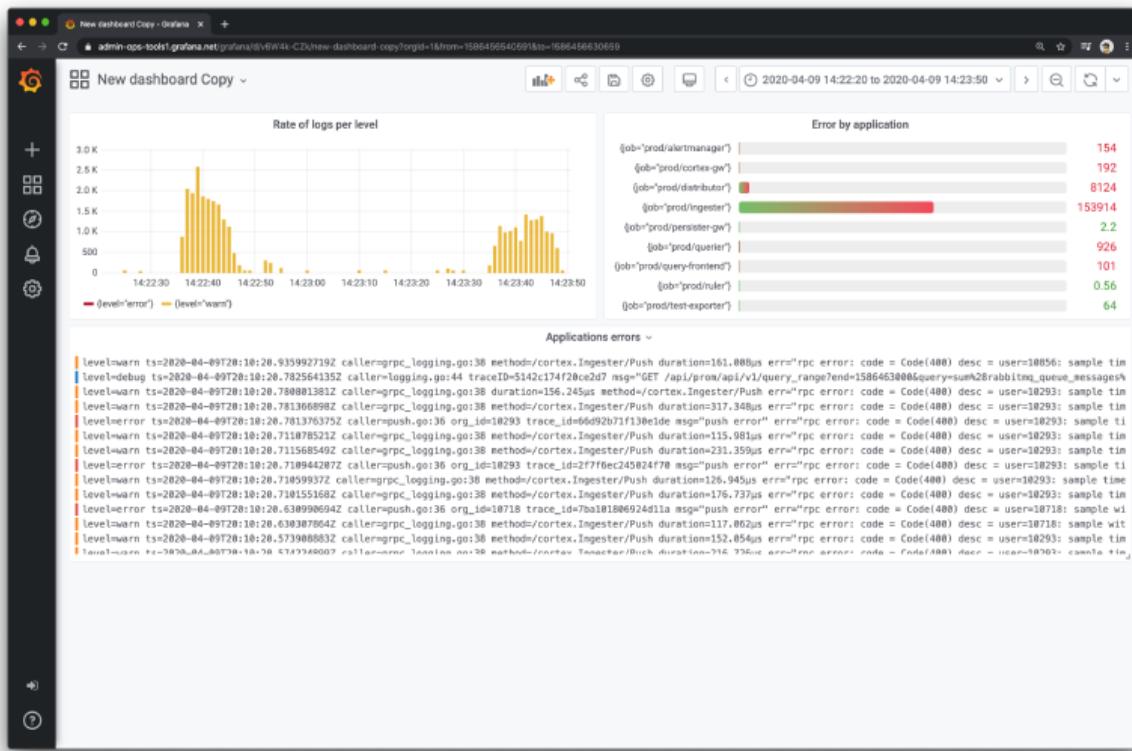
admin / admin

Add Loki datasource:

- URL: `http://localhost:3100`
 - Save & Test ✓
-

📊 STEP 4 — PREBUILT DASHBOARDS (IMPORT)





Import these **official dashboards**:

Purpose	Dashboard ID
---------	--------------

Node Metrics (CPU/RAM/Disk) **1860**

Loki Logs Overview **15141**

Linux Logs **13639**

📌 Grafana → Dashboards → Import → Paste ID

⚠️ STEP 5 — ALERT RULES (CPU / RAM / LOG ERRORS)

🔥 CPU Alert

- alert: HighCPUUsage

```
expr: 100 - (avg by(instance)(rate(node_cpu_seconds_total{mode="idle"}[2m])) * 100) > 80
```

for: 2m

MemoryWarning Alert

- alert: HighMemoryUsage

```
expr: (node_memory_Active_bytes / node_memory_MemTotal_bytes) * 100 > 85
```

for: 2m

ERROR LOG ALERT

```
count_over_time({job="app"} |= "ERROR" [1m]) > 5
```

→ Alerts can send to [Email / Slack / Webhook](#)

STEP 6 — S3 LOG ARCHIVAL (AUTOMATED)

Create script:

```
sudo nano /usr/local/bin/log-s3-backup.sh
```

```
#!/bin/bash
```

```
DATE=$(date +%F)
```

```
HOST=$(hostname)
```

```
ARCHIVE="/tmp/logs-$HOST-$DATE.tar.gz"
```

```
tar -czf $ARCHIVE /var/log /var/www/*storage/logs
```

```
aws s3 cp $ARCHIVE s3://YOUR_BUCKET/logs/$HOST/
```

Permissions:

```
sudo chmod +x /usr/local/bin/log-s3-backup.sh
```

Cron (Daily backup)

```
crontab -e
```

```
0 2 * * * /usr/local/bin/log-s3-backup.sh
```

MULTIPLE SERVER SETUP

On **all servers**:

- Install Promtail + Node Exporter
- Point Promtail to **central Loki IP**

Change:

clients:

```
- url: http://LOKI_SERVER_IP:3100/loki/api/v1/push
```

Grafana stays **only on one server**.

FINAL RESULT (WHAT YOU GET)

- ✓ Live logs (system, app, nginx, mysql)
 - ✓ Powerful search & filters
 - ✓ CPU/RAM correlated with logs
 - ✓ Alerts on spikes & errors
 - ✓ S3 archived logs
 - ✓ Works without Kubernetes
-

🔥 Want Me To?

I can give you:

- 🔒 IAM policy for S3
- 🏙 Docker version
- 📧 Email alert config
- 📊 Custom dashboard JSON
- 🔎 Regex-based log parsing

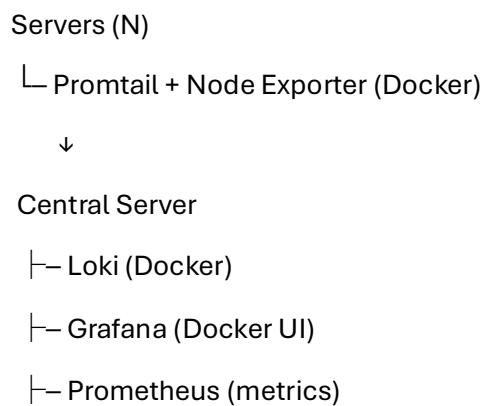
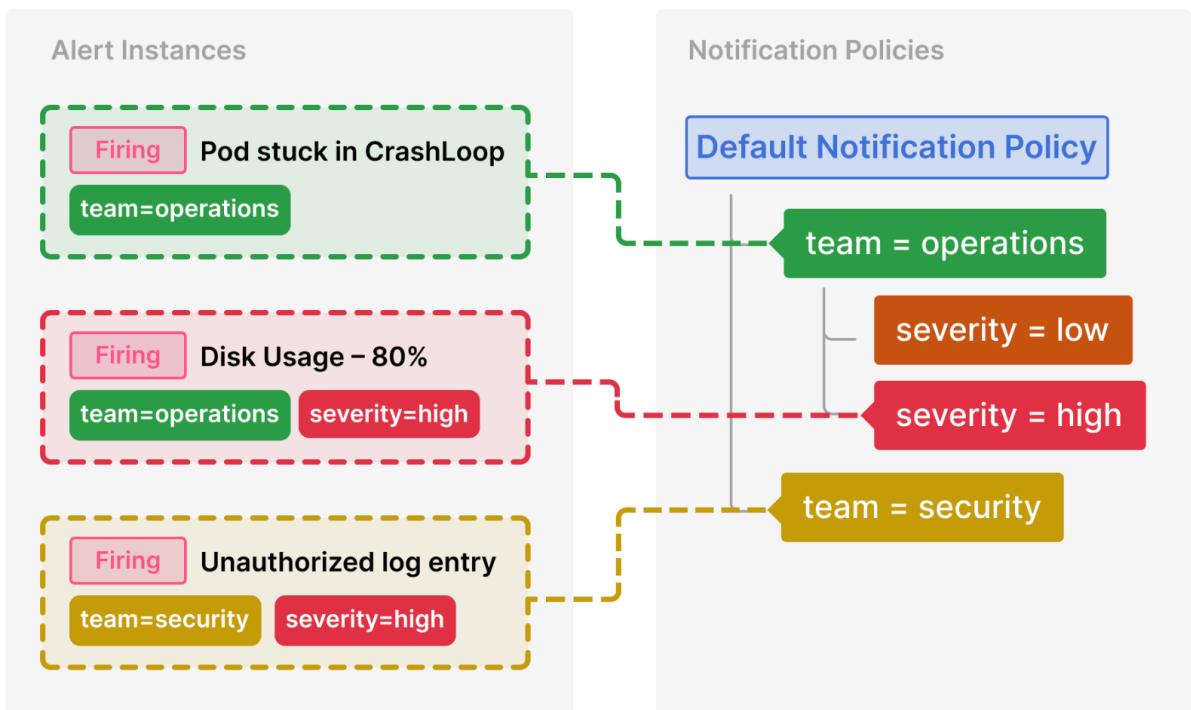
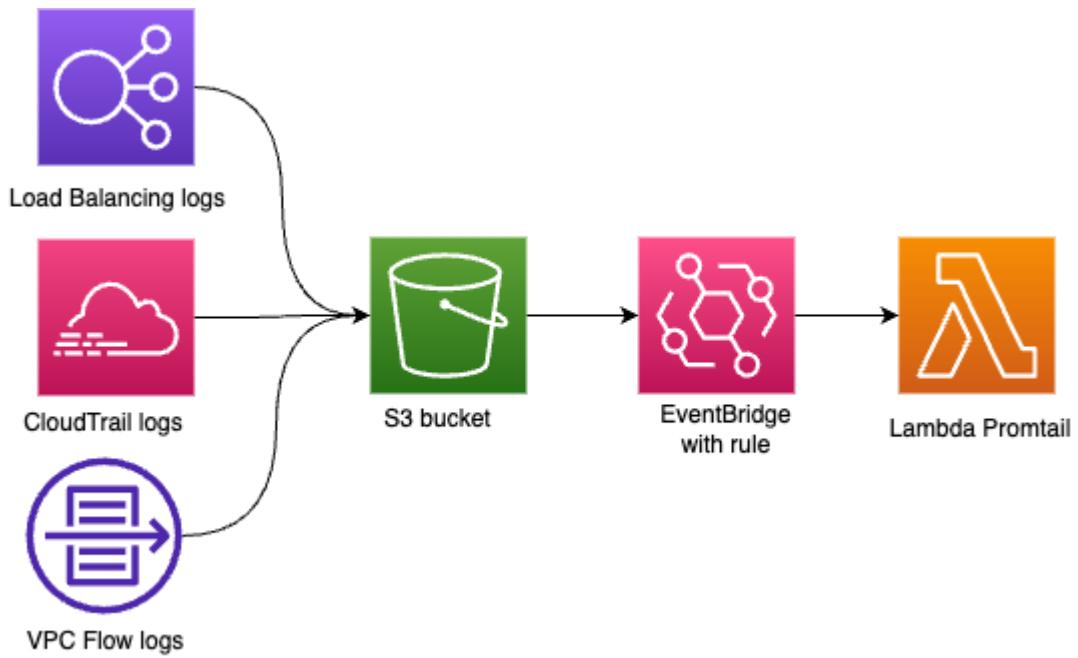
Just tell me 👍

🔒 IAM policy for S3 🏙 Docker version 📧 Email alert config 📊 Custom dashboard JSON 🔎
Regex-based log parsing Step by Step PleSE

Perfect 👍

Below is a **FULL STEP-BY-STEP production guide** for **Ubuntu 22.04** covering **ALL 5 items**, written so you can follow it **line by line** without Kubernetes.

🧱 ARCHITECTURE (What we're building)



└ AWS S3 (log archive)

STEP 1 — IAM POLICY FOR S3 (SECURE)

1 Create IAM Policy (JSON)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::YOUR_BUCKET_NAME",  
                "arn:aws:s3:::YOUR_BUCKET_NAME/*"  
            ]  
        }  
    ]  
}
```

2 Attach Policy

- Attach to **EC2 IAM Role**
- OR IAM User (if not EC2)

3 Test

aws s3 ls s3://YOUR_BUCKET_NAME

-  If it lists → policy is correct
-

STEP 2 — DOCKER INSTALL (ONE TIME)

sudo apt update

```
sudo apt install -y docker.io docker-compose-plugin  
sudo systemctl enable --now docker
```

Verify:

```
docker --version
```

STEP 3 — DOCKER COMPOSE (GRAFANA + LOKI + PROMETHEUS)

1 Create directory

```
mkdir -p ~/observability && cd ~/observability
```

2 docker-compose.yml

```
version: "3.8"
```

```
services:
```

```
loki:
```

```
  image: grafana/loki:2.9.4  
  command: -config.file=/etc/loki/local-config.yaml  
  ports:  
    - "3100:3100"
```

```
prometheus:
```

```
  image: prom/prometheus  
  ports:  
    - "9090:9090"  
  volumes:  
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
```

```
grafana:
```

```
  image: grafana/grafana  
  ports:  
    - "3000:3000"  
  environment:  
    - GF_SECURITY_ADMIN_PASSWORD=admin
```

```
depends_on:
```

- loki
- prometheus

3 prometheus.yml

```
global:
```

```
scrape_interval: 15s
```

```
scrape_configs:
```

```
- job_name: node
```

```
static_configs:
```

- targets: ["HOST_IP:9100"]

4 Start stack

```
docker compose up -d
```

Access:

- Grafana → http://SERVER_IP:3000
 - Loki → http://SERVER_IP:3100
-

📦 STEP 4 — PROMTAIL (LOG COLLECTOR) — ALL LOGS

1 Install Promtail

```
sudo apt install -y promtail
```

2 /etc/promtail/config.yml

```
server:
```

```
http_listen_port: 9080
```

```
clients:
```

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

```
scrape_configs:
```

```
- job_name: system
```

```
static_configs:
```

```
- labels:
```

```
job: syslog
host: ${HOSTNAME}
__path__: /var/log/syslog

- job_name: nginx
  static_configs:
    - labels:
        job: nginx
      __path__: /var/log/nginx/*.log

- job_name: mysql
  static_configs:
    - labels:
        job: mysql
      __path__: /var/log/mysql/*.log

- job_name: app
  static_configs:
    - labels:
        job: app
      __path__: /var/www/*/storage/logs/*.log
```

Restart:

```
sudo systemctl restart promtail
```

STEP 5 — REGEX-BASED LOG PARSING (IMPORTANT)

Example: NGINX access log parsing

```
pipeline_stages:
  - regex:
      expression: '^(?P<ip>[^ ]+).*"(?P<method>GET|POST) (?P<uri>[^ ]+).*(?P<status>\d{3})'
    - labels:
        method:
```

status:

Query in Grafana

```
{job="nginx",status="500"}
```

STEP 6 — EMAIL ALERT CONFIG (GRAFANA)

1 Enable SMTP

Grafana → Configuration → Alerting → Contact Points

2 SMTP Settings

```
[smtp]
enabled = true
host = smtp.gmail.com:587
user = yourmail@gmail.com
password = APP_PASSWORD
from_address = yourmail@gmail.com
```

Restart Grafana:

```
docker restart observability-grafana-1
```

STEP 7 — ALERT RULES

CPU Alert

```
100 - avg(rate(node_cpu_seconds_total{mode="idle"}[2m])) * 100 > 80
```

RAM Alert

```
(node_memory_Active_bytes / node_memory_MemTotal_bytes) * 100 > 85
```

ERROR LOG Alert

```
count_over_time({job="app"} |= "ERROR" [1m]) > 5
```

STEP 8 — CUSTOM DASHBOARD JSON (IMPORT)

Grafana → Dashboards → Import → Paste JSON

```
{
  "title": "Server Logs & Health",
  "panels": [
    {
```

```
"type": "logs",
"title": "Application Errors",
"targets": [
{
  "expr": "{job=\"app\"} |= \"ERROR\""
}
]
}
```

STEP 9 — S3 LOG ARCHIVAL (DAILY)

Script

```
sudo nano /usr/local/bin/log-backup.sh
#!/bin/bash
DATE=$(date +%F)
HOST=$(hostname)
tar -czf /tmp/$HOST-$DATE.tar.gz /var/log
aws s3 cp /tmp/$HOST-$DATE.tar.gz s3://YOUR_BUCKET/logs/$HOST/
chmod +x /usr/local/bin/log-backup.sh
```

Cron

```
0 2 * * * /usr/local/bin/log-backup.sh
```

FINAL RESULT

- ✓ Beautiful Grafana UI
 - ✓ Live logs (system/app/nginx/mysql)
 - ✓ Regex parsing
 - ✓ Email alerts
 - ✓ Docker based
 - ✓ S3 archived logs
 - ✓ Scales to multiple servers
-

🔥 NEXT (OPTIONAL)

I can now:

- 🔒 Harden Loki (TLS + auth)
- 📊 Create advanced dashboards
- 💡 Add anomaly detection
- 📦 Convert everything to Ansible

Just tell me 👍