# REAL-TIME MONITORING AND LOGGING ON AWS EC2 USING DOCKER, PROMETHEUS AND GRAFANA PLATFORM

PRIYANKA RAJAGOPAL
GitHub

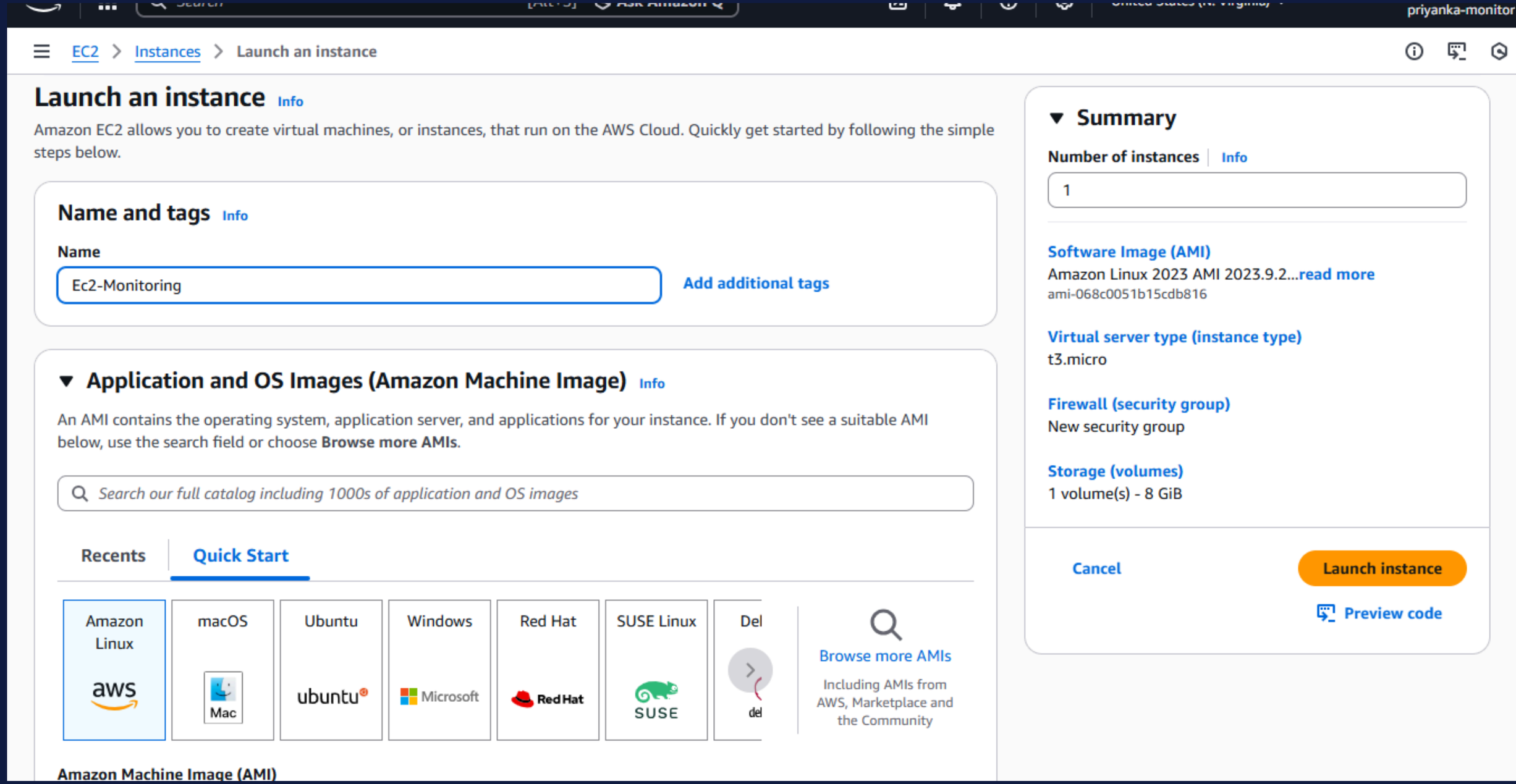
# PROJECT OVERVIEW

- The Project demonstrates a complete monitoring and logging solution for cloud-based applications using AWS EC2 and Docker containers.

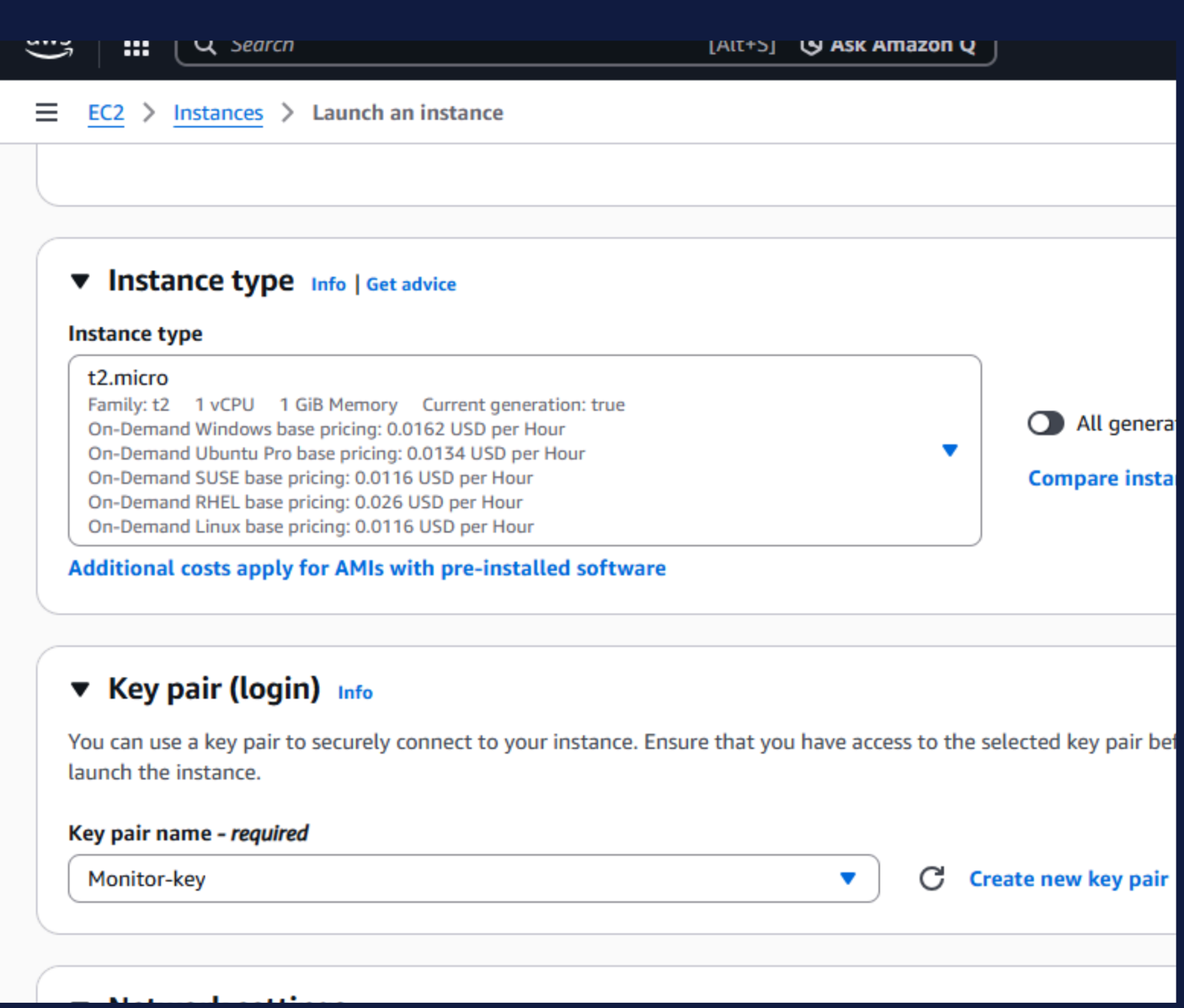
- It integrates Prometheus for metrics collection, Grafana for visualization, Node Exporter for server-level metrics, Loki and Promtail for log management, and Alertmanager for proactive notifications.

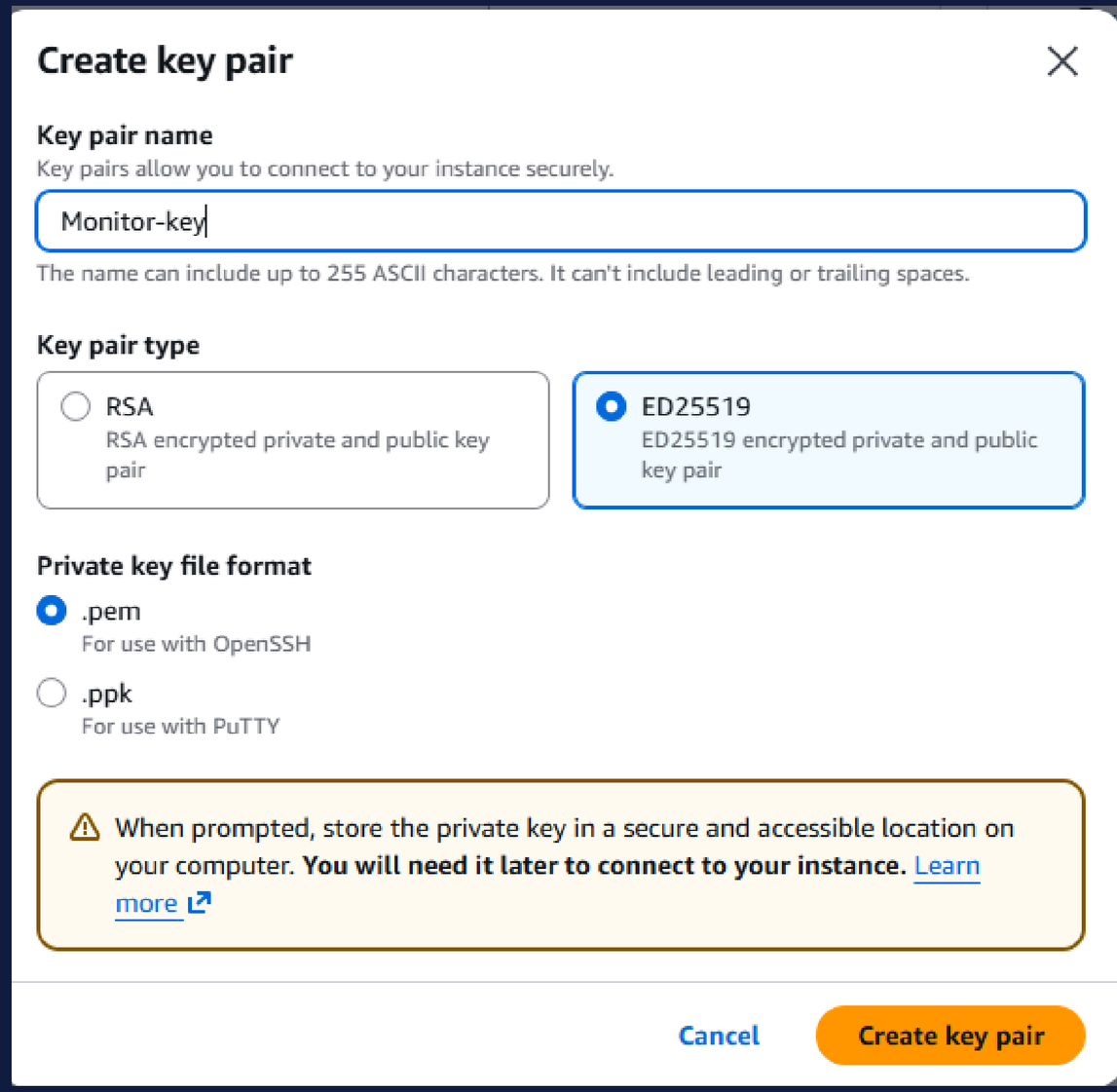
- The goal is to provide real-time insights into system performance, track application logs, and set up automated alerts for potential issues, ensuring reliability and operational efficiency.


AWS EC2
Amazon Linux
Docker Containers
Prometheus
Metrics Collection
& Monitoring
Grafana
Loki
Log Aggregation
Node Exporter
System Metrics
Promtail
Log Collection
Alertmanager
Alerting & Notifications

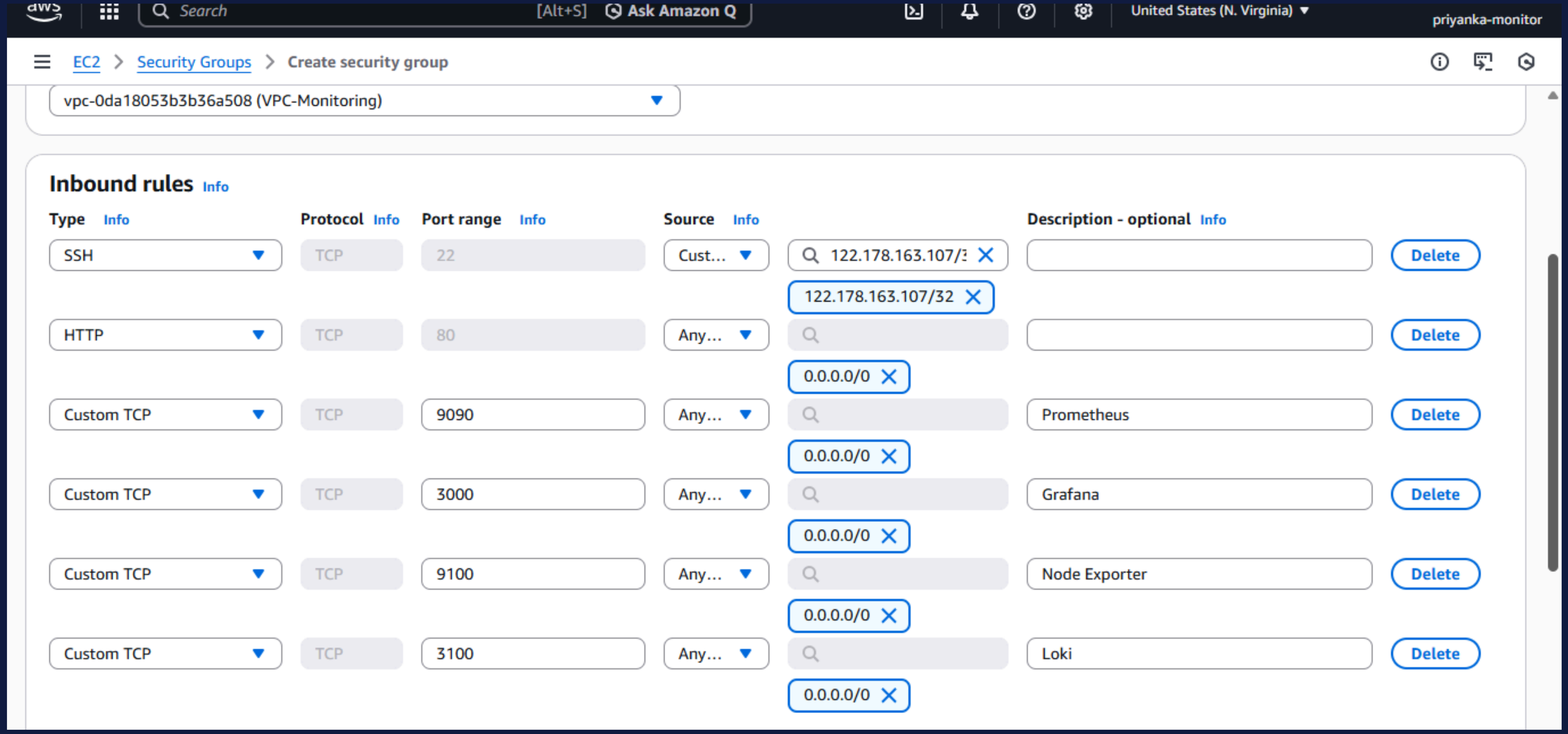



Launched an instance with “Amazon Linux”.

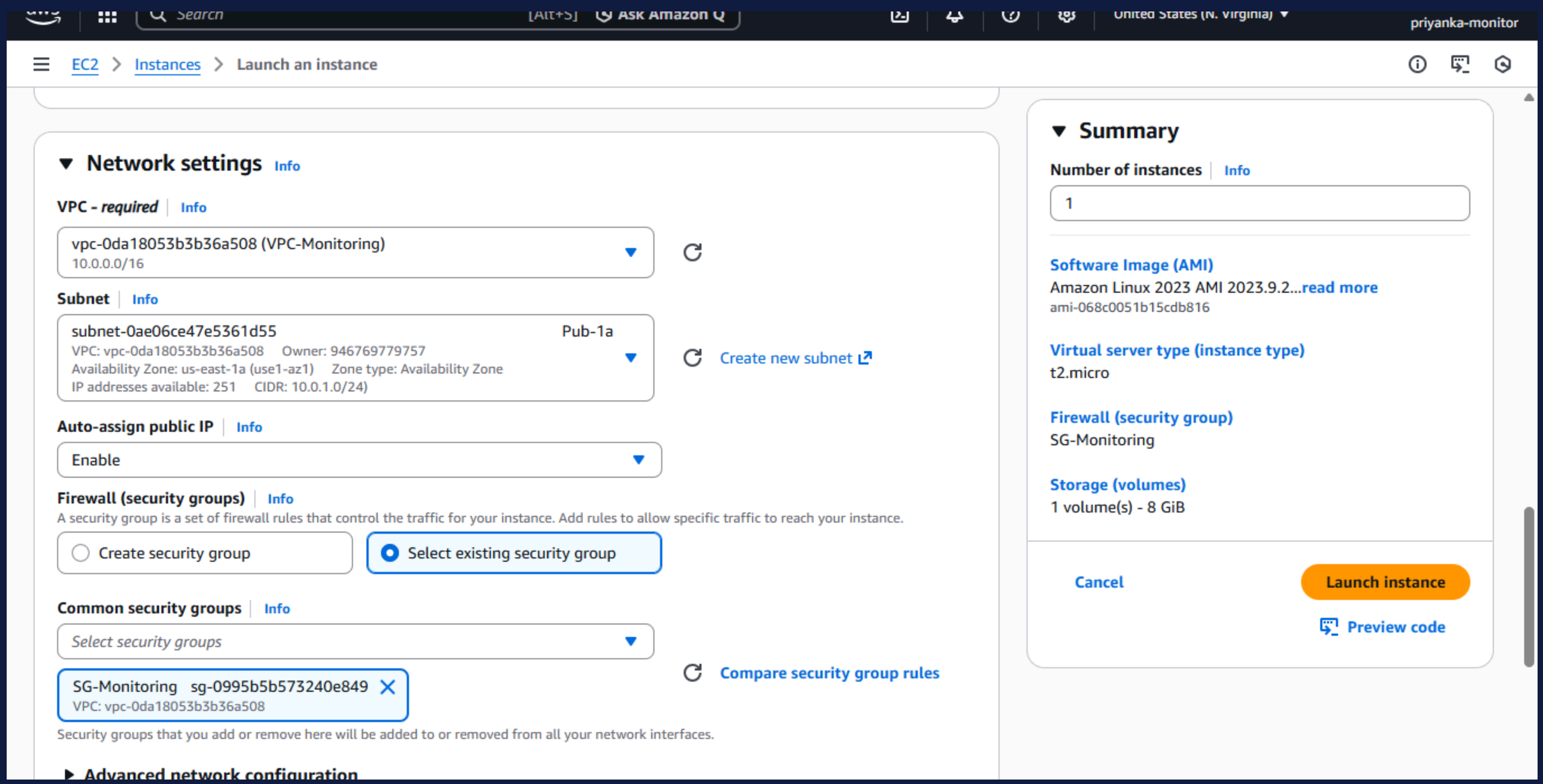


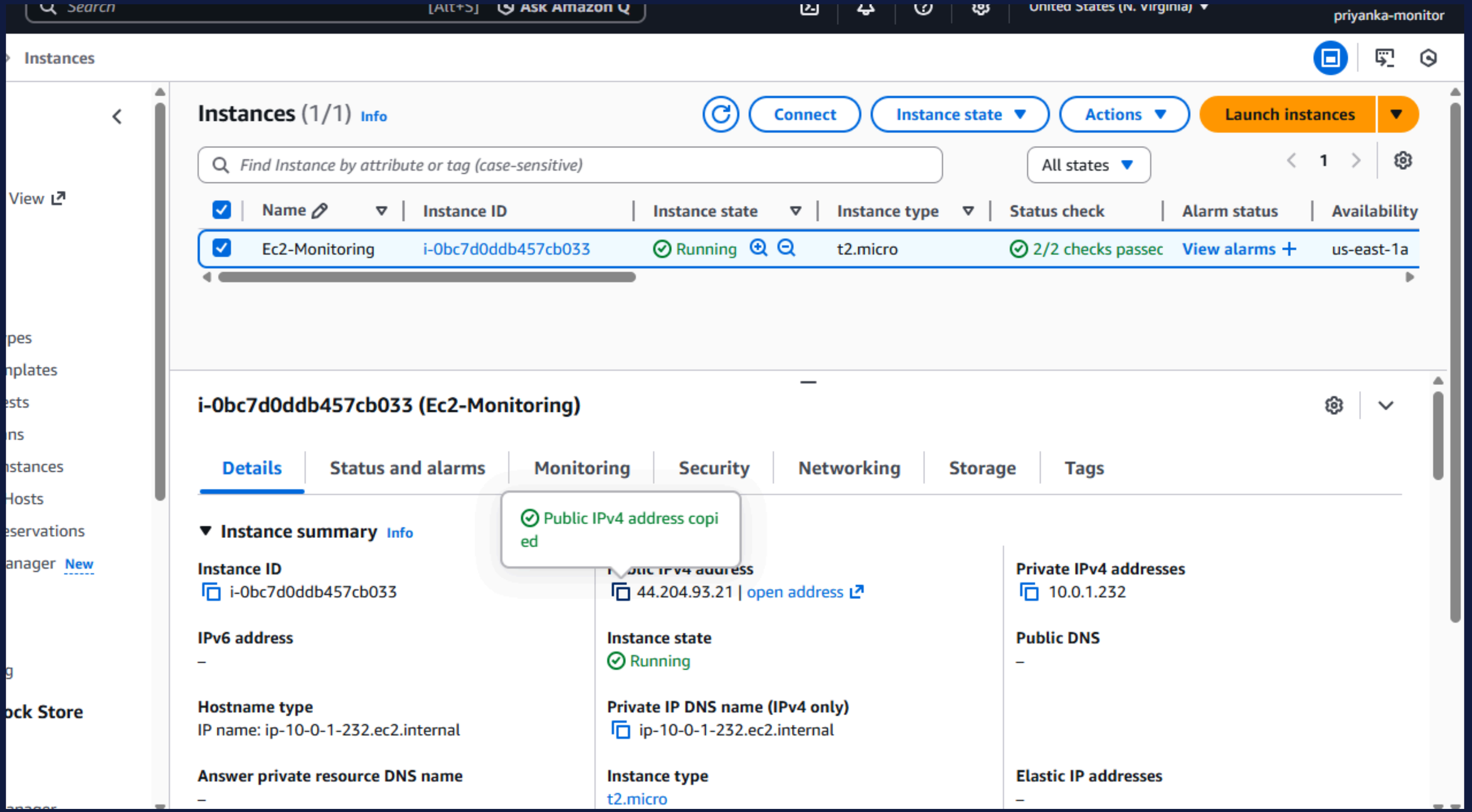

Used t2 micro instance type with the respective key pair.




Security group -Inbound rules are created for Prometheus,Grafana,Node exporter and Loki .Outbound rules are given as All traffic.




Attached the security group with instance creating and “Auto assign public IP” is enabled.

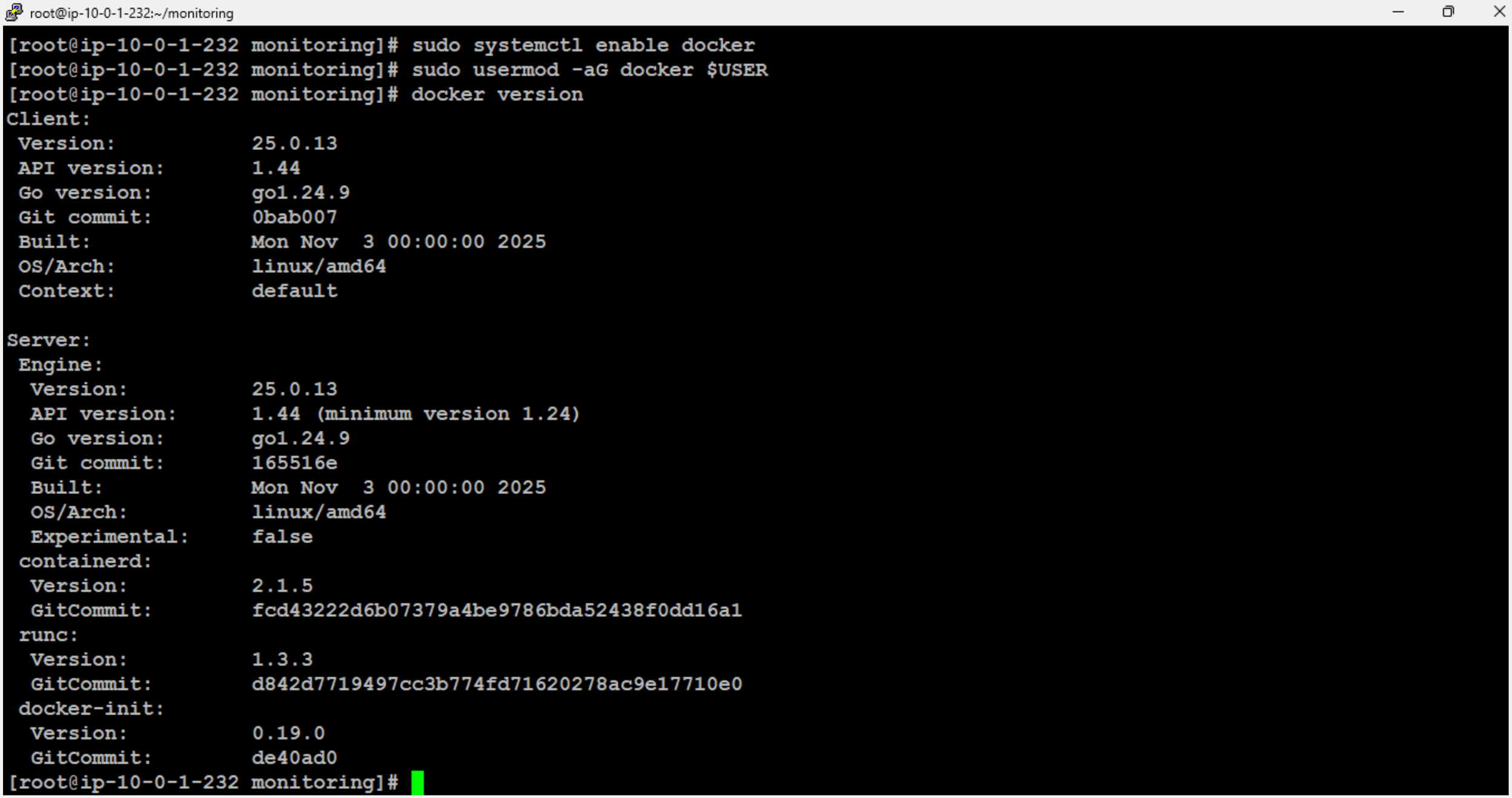



Launched instance with the Putty Software using the Public IP address.

root@ip-10-0-1-232:~

```
[root@ip-10-0-1-232 ~]# sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository                   223 kB/s |  29 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-1-232 ~]# sudo yum install docker -y
Last metadata expiration check: 0:05:56 ago on Fri Dec 26 03:19:13 2025.
Dependencies resolved.
================================================================================================
 Package                  Architecture      Version                        Repository       Size
================================================================================================
Installing:
 docker                   x86_64            25.0.13-1.amzn2023.0.2         amazonlinux      46 M
Installing dependencies:
 container-selinux        noarch            4:2.242.0-1.amzn2023           amazonlinux      58 k
 containerd               x86_64            2.1.5-1.amzn2023.0.1           amazonlinux      23 M
 iptables-libs            x86_64            1.8.8-3.amzn2023.0.2           amazonlinux     401 k
 iptables-nft             x86_64            1.8.8-3.amzn2023.0.2           amazonlinux     183 k
 libcgroup                x86_64            3.0-1.amzn2023.0.1             amazonlinux      75 k
 libnetfilter_conntrack   x86_64            1.0.8-2.amzn2023.0.2           amazonlinux      58 k
 libnfnetlink             x86_64            1.0.1-19.amzn2023.0.2          amazonlinux      30 k
 libnftnl                 x86_64            1.2.2-2.amzn2023.0.2           amazonlinux      84 k
 pigz                     x86_64            2.5-1.amzn2023.0.3             amazonlinux      83 k
 runc                     x86_64            1.3.3-2.amzn2023.0.1           amazonlinux     3.9 M

Transaction Summary
================================================================================================
Install  11 Packages

Total download size: 74 M
Installed size: 280 M
```

**Installed dependencies and docker.**




**Started docker and Version of docker is confirmed.**

```
[root@ip-10-0-1-232 monitoring]# docker run -d --name node-exporter --network monitoring-net -p 9100:9100 prom/node-export
er
e3ddee30179a374488ff55df27b5a97aa1453e4972f2b1effa60ed5771ed0d75
[root@ip-10-0-1-232 monitoring]#
```

```
[root@ip-10-0-1-232 monitoring]# curl localhost:9100/metrics
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This value is set by the GOGC
 environment variable, and the runtime/debug.SetGCPercent function. Sourced from /gc/gogc:percent.
# TYPE go_gc_gogc_percent gauge
go_gc_gogc_percent 100
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is set b
y the GOMEMLIMIT environment variable, and the runtime/debug.SetMemoryLimit function. Sourced from /gc/gomemlimit:bytes.
# TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 9.223372036854776e+18
# HELP go_goroutines Number of goroutines that currently exist.
```

**Started Node Exporter using Docker and validated metric exposure via the /metrics endpoint.**

root@ip-10-0-1-232:~/monitoring

```
  GNU nano 8.3                                prometheus/prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: node
    static_configs:
      - targets: ["localhost:9100"]
```

root@ip-10-0-1-232:~/monitoring

```
[root@ip-10-0-1-232 monitoring]# docker run -d \
--name prometheus \
-p 9090:9090 \
-v $(pwd)/prometheus:/etc/prometheus \
prom/prometheus
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9d85dc8d0609: Already exists
d0f7326b7716: Already exists
c5e95088860d: Pull complete
7408017f0d80: Pull complete
48a0f696008d: Pull complete
2430d5ddfd0e: Pull complete
b8d9ef68618b: Pull complete
b351d2c02e9d: Pull complete
08a0c52f4aa7: Pull complete
fd581fe2de26: Pull complete
Digest: sha256:2b6f734e372c1b4717008f7d0a0152316aedd4d13ae17ef1e3268dbfaf68041b
Status: Downloaded newer image for prom/prometheus:latest
daf4b3e7ac6ce1fa82e7ac4bb4f42da384e7075bd7b2a6adcf70041f9444bf65
```

**Configured Prometheus using a custom prometheus.yml file and started the service using Docker.**

root@ip-10-0-1-232:~/alertmanager

```
  GNU nano 8.3                    /root/monitoring/alertmanager/alertmanager.yml                    Modified
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname']
  receiver: 'dummy'

receivers:
  - name: 'dummy'
```

root@ip-10-0-1-232:~/monitoring

```
[root@ip-10-0-1-232 monitoring]# nano alertmanager/alertmanager.yml
[root@ip-10-0-1-232 monitoring]# docker run -d \
--name alertmanager \
-p 9093:9093 \
-v $(pwd)/alertmanager:/etc/alertmanager \
prom/alertmanager
Unable to find image 'prom/alertmanager:latest' locally
latest: Pulling from prom/alertmanager
9d85dc8d0609: Pull complete
d0f7326b7716: Pull complete
1e6a3a57898c: Pull complete
315eb988c8ec: Pull complete
e102c75aa13f: Pull complete
389784bee712: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:abb750ac7b63116761c16dd481ae92496fbe04721686c0920f0fa4d0728cd4a6
Status: Downloaded newer image for prom/alertmanager:latest
2bf6b5e20b845ee19541e4f076f6d9a12d27a4fa8561265db9b193c90b1e5125
[root@ip-10-0-1-232 monitoring]#
```

**Configured Alertmanager with alert routing rules and started it using Docker.**

root@ip-10-0-1-232:~/monitoring

```
[root@ip-10-0-1-232 monitoring]# docker run -d \
--name grafana \
-p 3000:3000 \
grafana/grafana

Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
014e56e61396: Pull complete
dafcd853cdea: Pull complete
fe0fb9ad3b64: Pull complete
f50da64eedfd: Pull complete
4a965f51fa32: Pull complete
835a6d0e4902: Pull complete
a2a0d06388d6: Pull complete
a7d3fdc2e5af: Pull complete
c8585378023b: Pull complete
d2772eb63bbd: Pull complete
Digest: sha256:2175aaa91c96733d86d31cf270d5310b278654b03f5718c59de12a865380a31f
Status: Downloaded newer image for grafana/grafana:latest
468bcde70523234276266d5baeb8a354301aa471649e93e3f3d2ad0a93b7906
[root@ip-10-0-1-232 monitoring]#
```

**Started Grafana using Docker**

```
 GNU nano 8.3                    /root/monitoring/loki/loki-config.yml                    Modified
auth_enabled: false

server:
  http_listen_port: 3100

ingester:
  lifecycler:
    address: 127.0.0.1
    ring:
      kvstore:
        store: inmemory
      replication_factor: 1
  chunk_idle_period: 5m
  max_chunk_age: 1h
  chunk_target_size: 1048576

schema_config:
  configs:
    - from: 2020-10-24
      store: boltdb
      object_store: filesystem
      schema: v11
      index:
        prefix: index_
        period: 168h

storage_config:
```

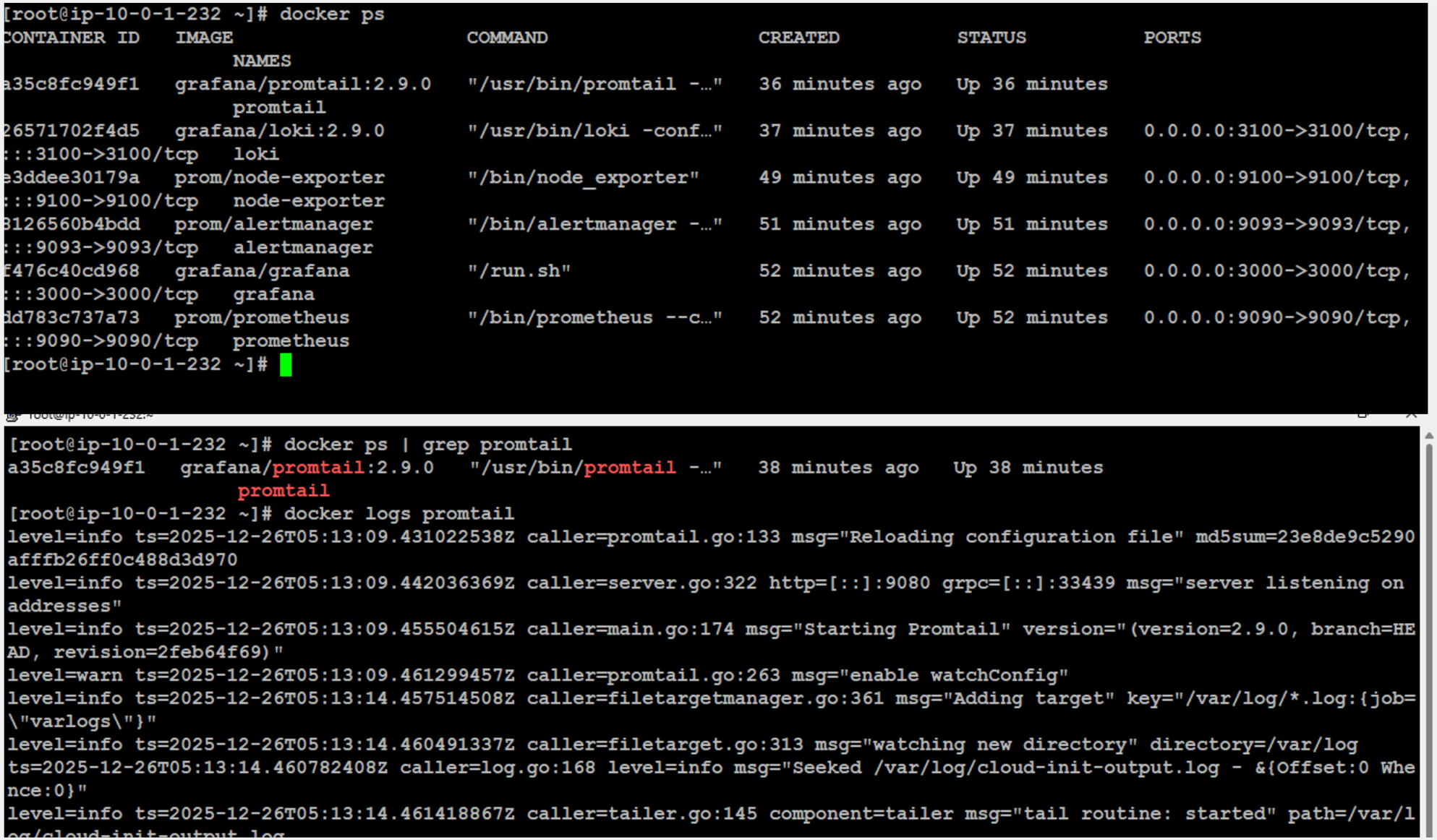
```
[root@ip-10-0-1-232 monitoring]# docker run -d \
--name loki \
-p 3100:3100 \
-v $(pwd)/loki/loki-config.yml:/etc/loki/loki-config.yml \
-v $(pwd)/loki/wal:/wal \
-v $(pwd)/loki/chunks:/loki/chunks \
-v $(pwd)/loki/index:/loki/index \
grafana/loki:2.9.0 \
-config.file=/etc/loki/loki-config.yml
046a03b5f418a02494268dbfdb4ed446417ee576260e2cb339a865c91631cb65
[root@ip-10-0-1-232 monitoring]#
```

**Loki was configured with storage and ingestion settings and then started it using docker**

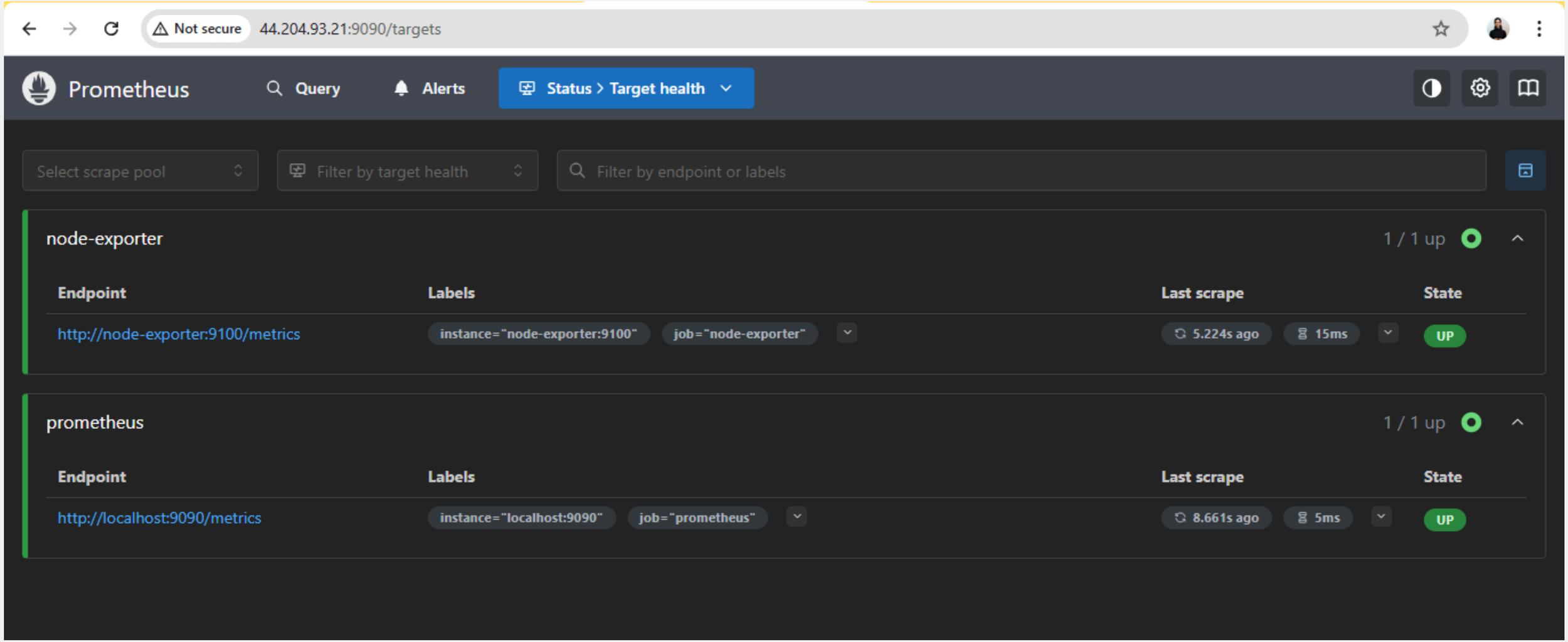
```
  GNU nano 8.3                          /root/promtail/promtail-config.yml                          Modified
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

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

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

```
[root@ip-10-0-1-232 monitoring]# docker run -d \
--name promtail \
-v /var/log:/var/log \
-v $(pwd)/promtail:/etc/promtail \
grafana/promtail:2.9.0 \
-config.file=/etc/promtail/promtail-config.yml
Unable to find image 'grafana/promtail:2.9.0' locally
2.9.0: Pulling from grafana/promtail
7d97e254a046: Pull complete
39932bcaa1cc: Pull complete
4059d897dde4: Pull complete
325d59b59788: Pull complete
9e9c5b842176: Pull complete
01befd6b403e: Pull complete
Digest: sha256:c2c423196c75a2c9c26f6fe0ba7200c3167334b14975747f5dcff678bd1a32e9
Status: Downloaded newer image for grafana/promtail:2.9.0
75d39e73bd4de8fb5b20e047cad1f3cb2ad7a04e4f847f431dac7f2130704618
[root@ip-10-0-1-232 monitoring]#
```

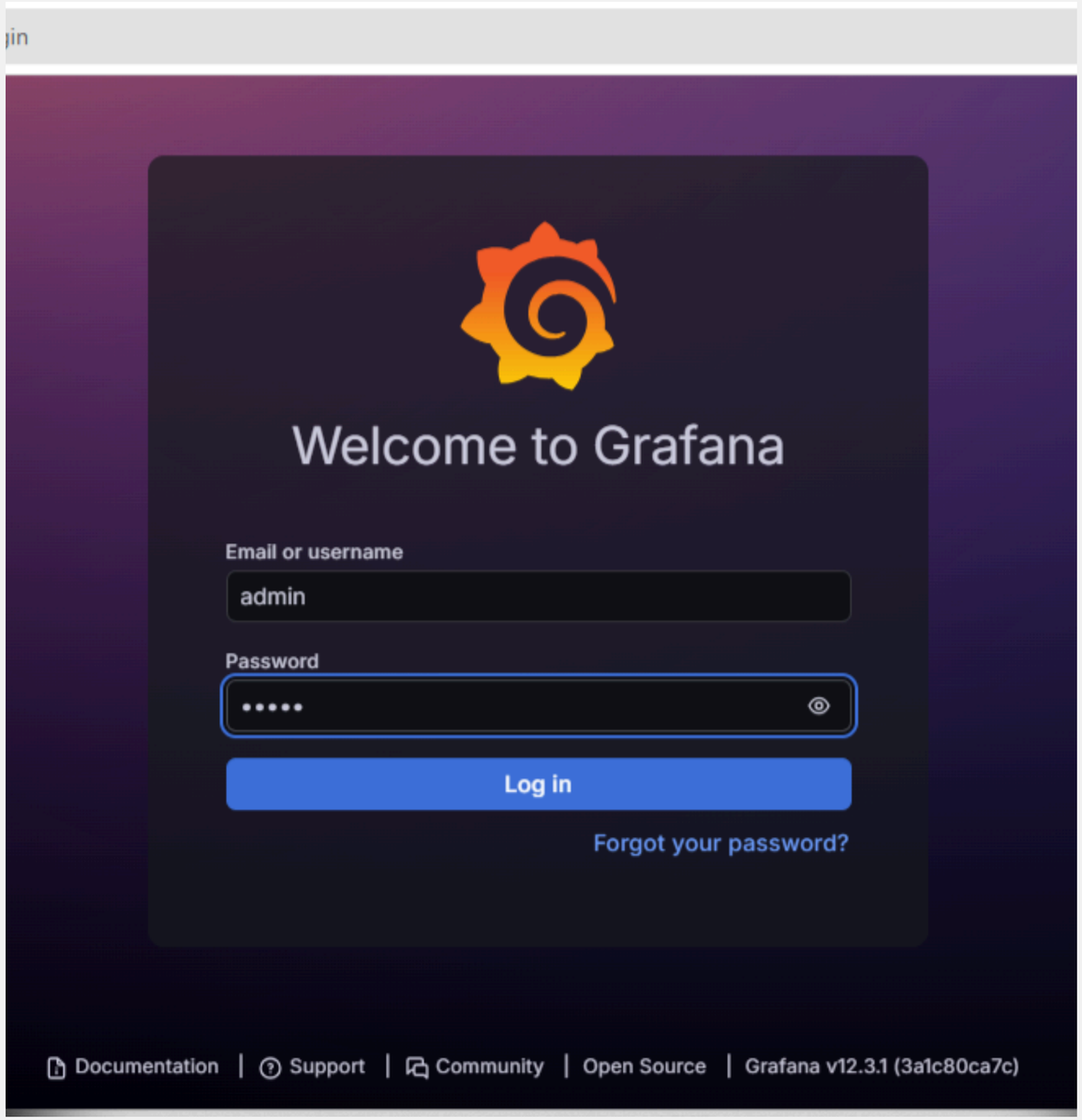
**Configured Promtail to ship logs to Loki and started it using Docker.**

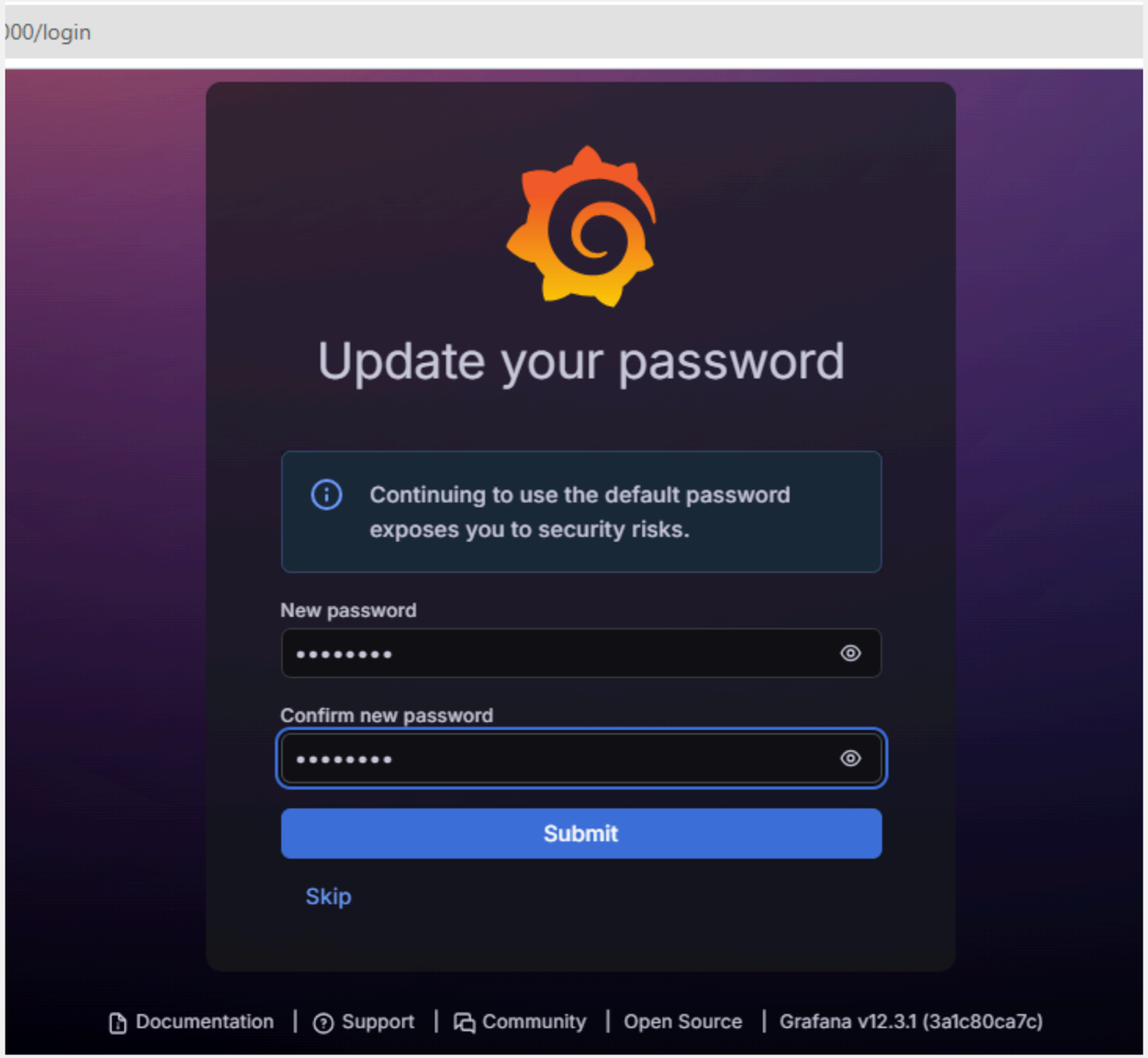



**Verified container status using docker ps and confirmed logs were successfully shipped to Loki.**

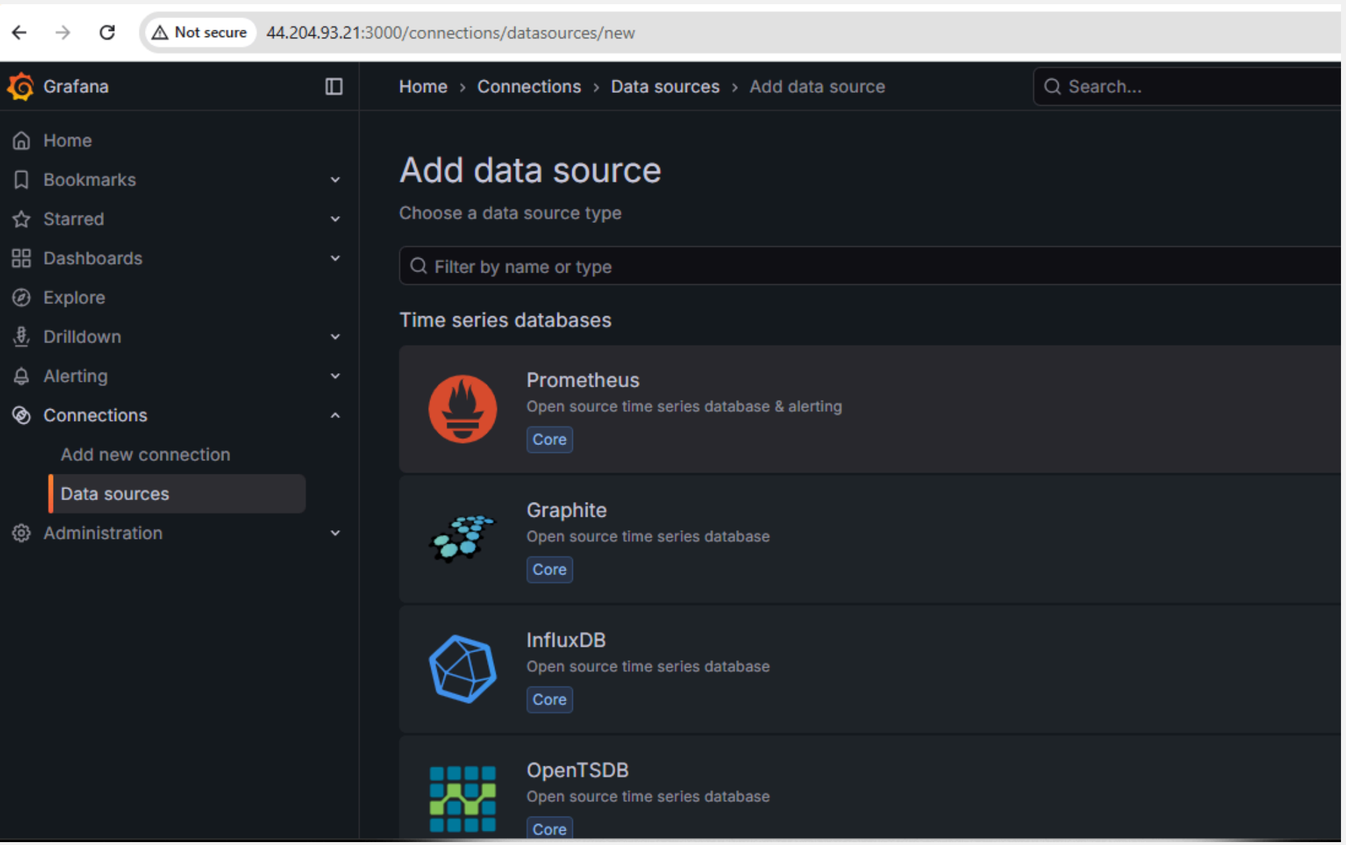


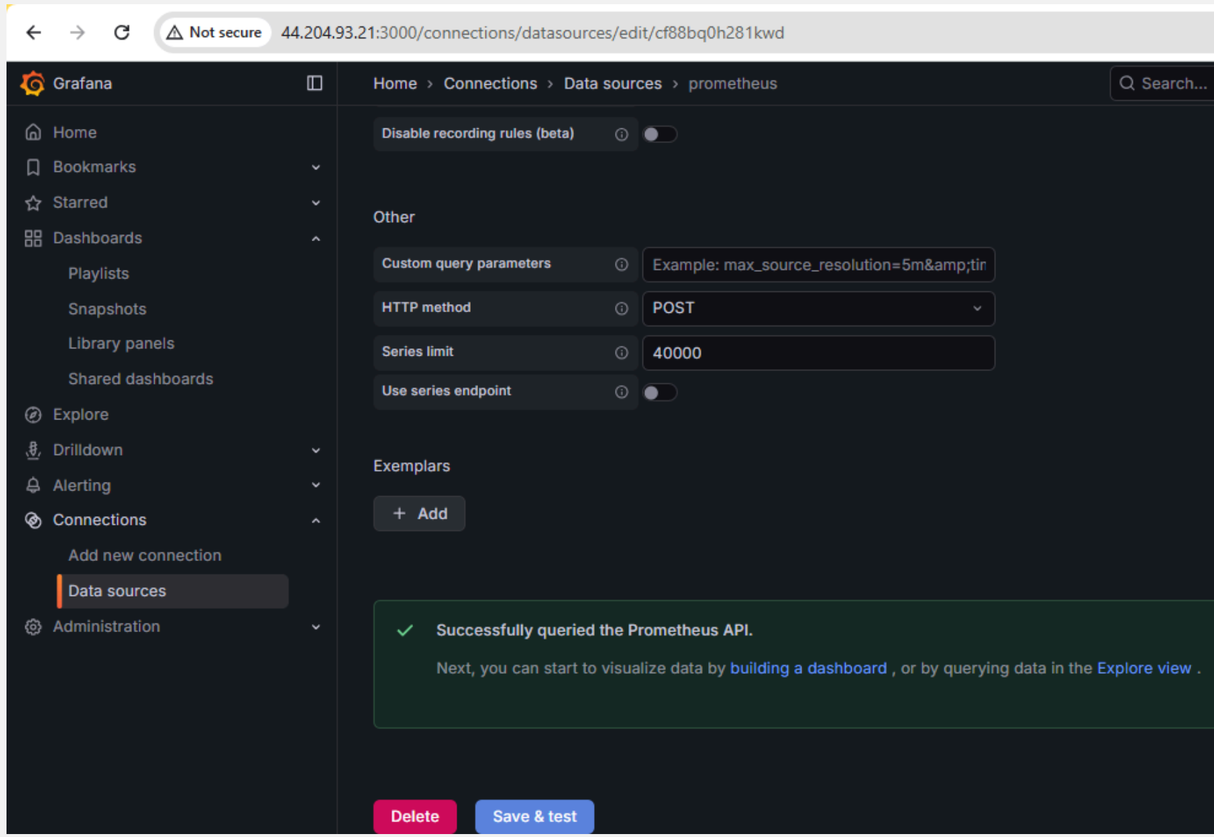

**Verified target status of Prometheus and confirmed Node Exporter was successfully scraped**



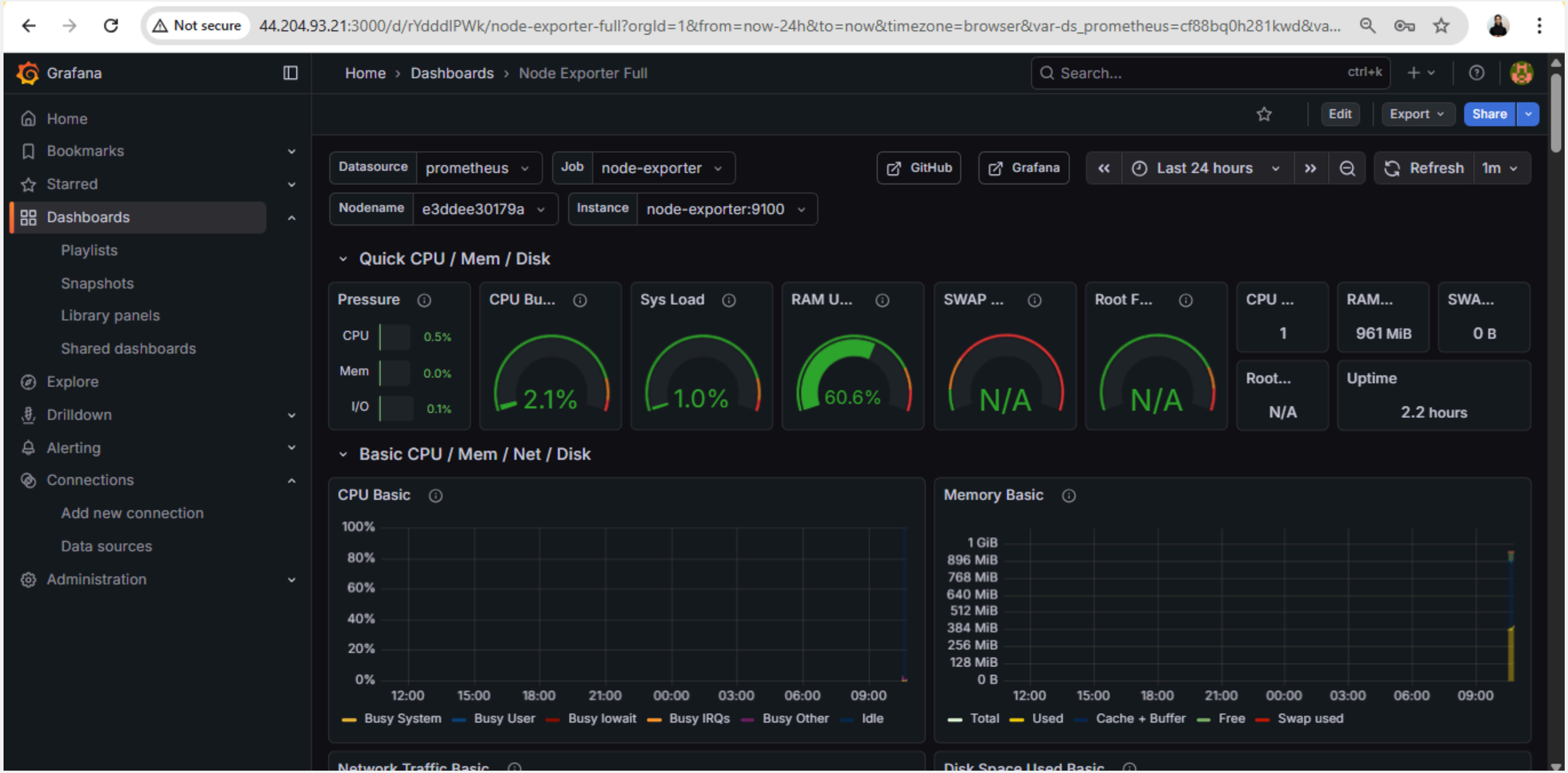




**Grafana service is running and accessible via web interface.”**

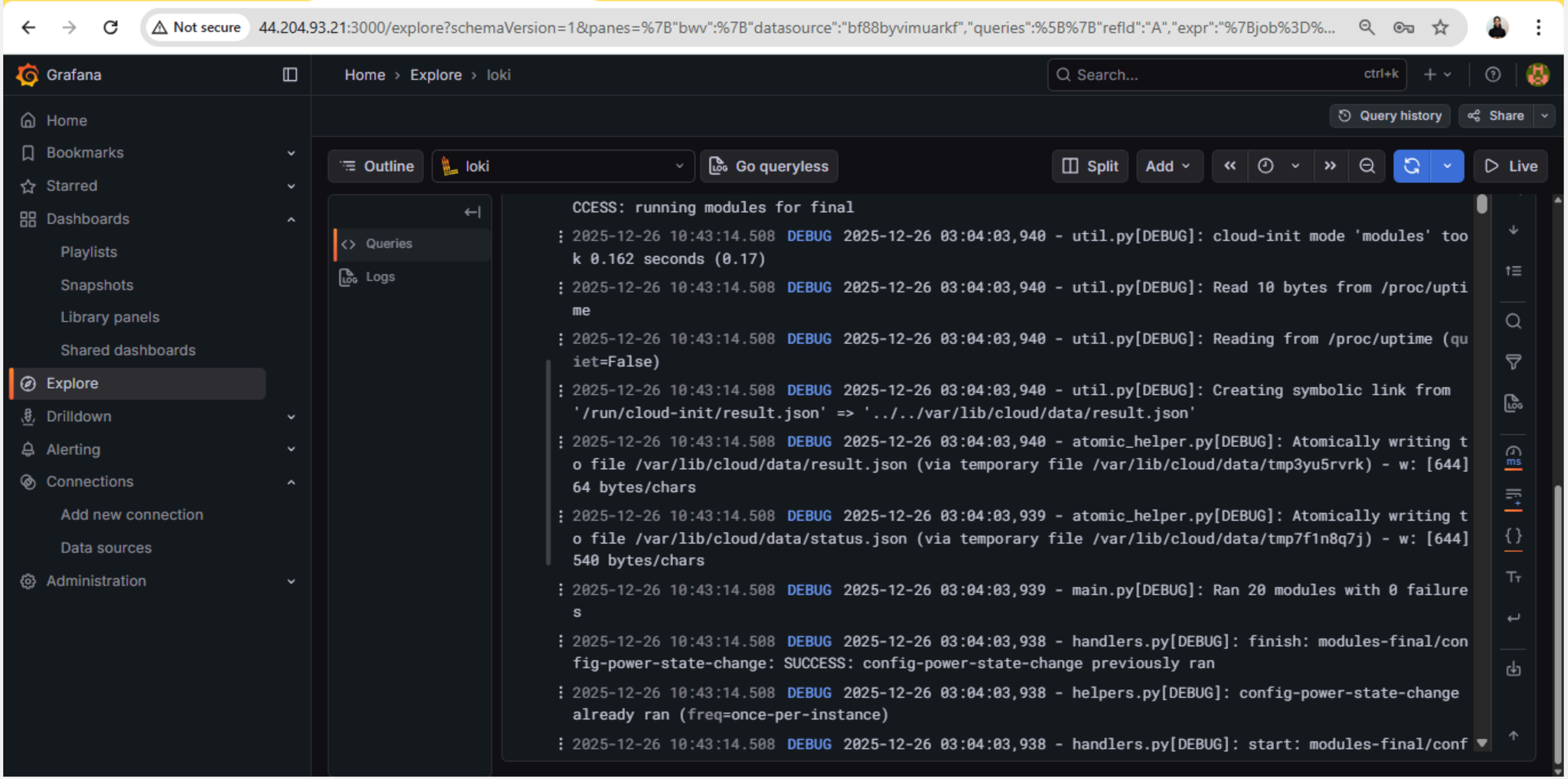


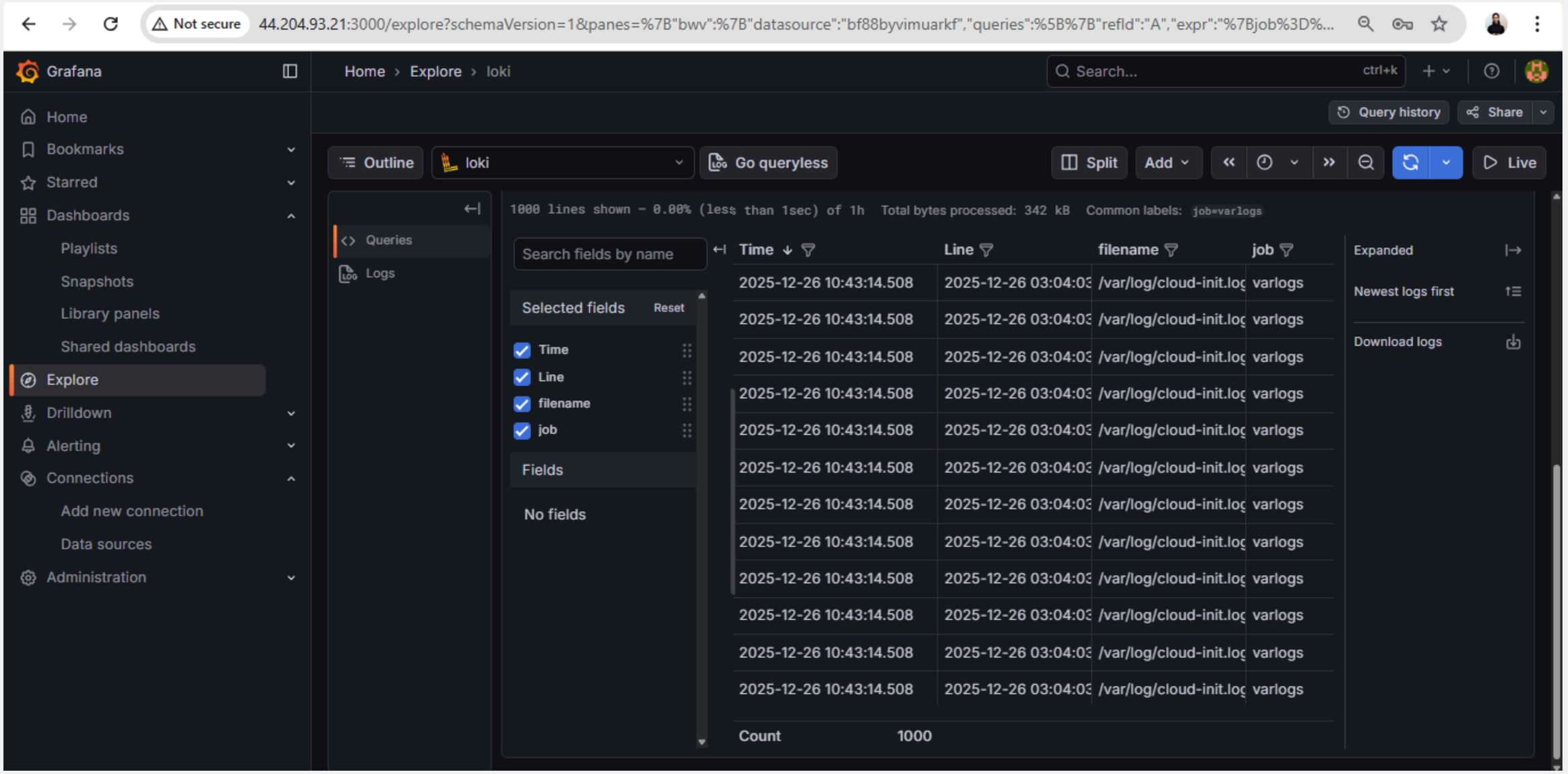

**Configured Prometheus (port 9090) and Loki (port 3100) as datasources in Grafana for metrics and log visualization.**



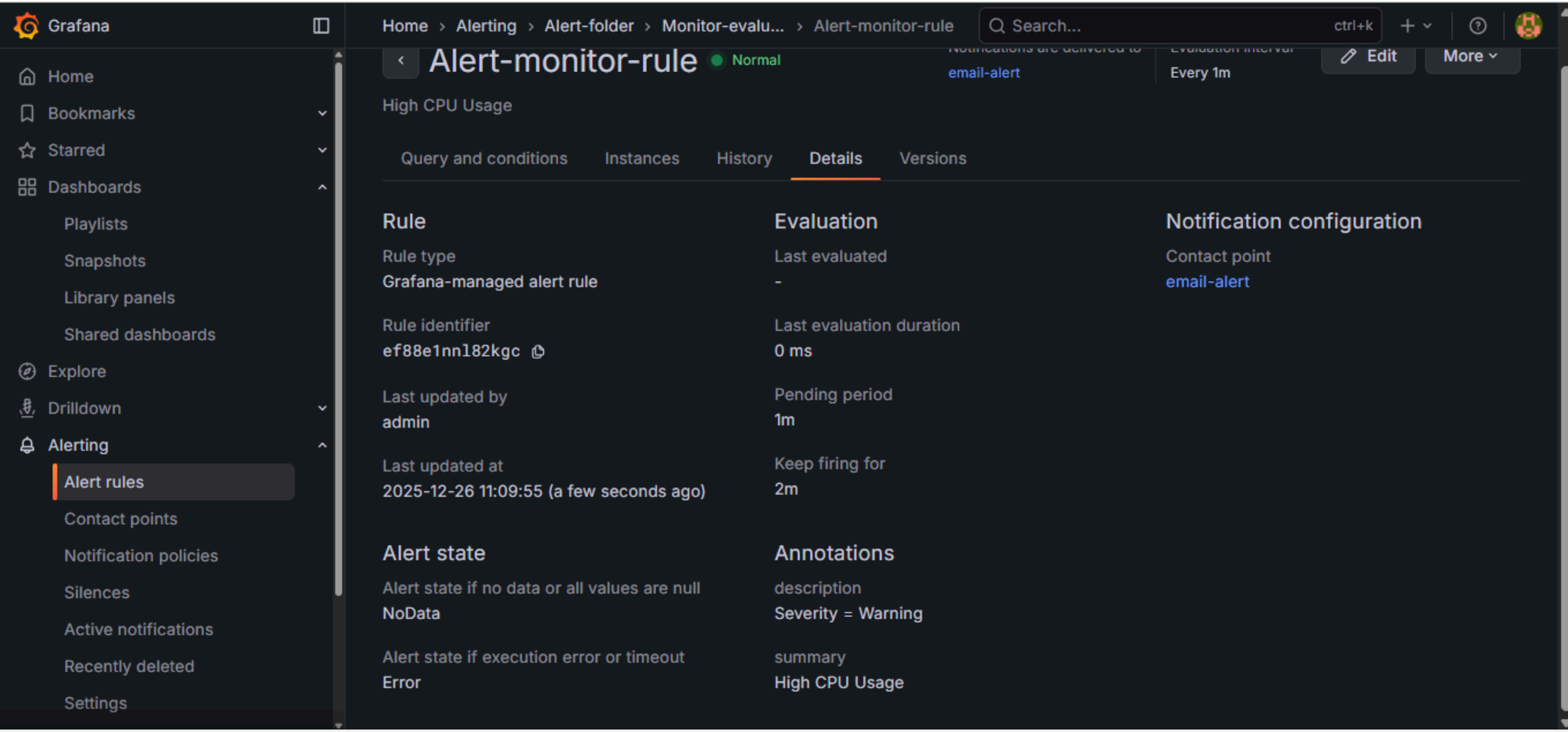

**Verified metrics visualization on Prometheus dashboard**



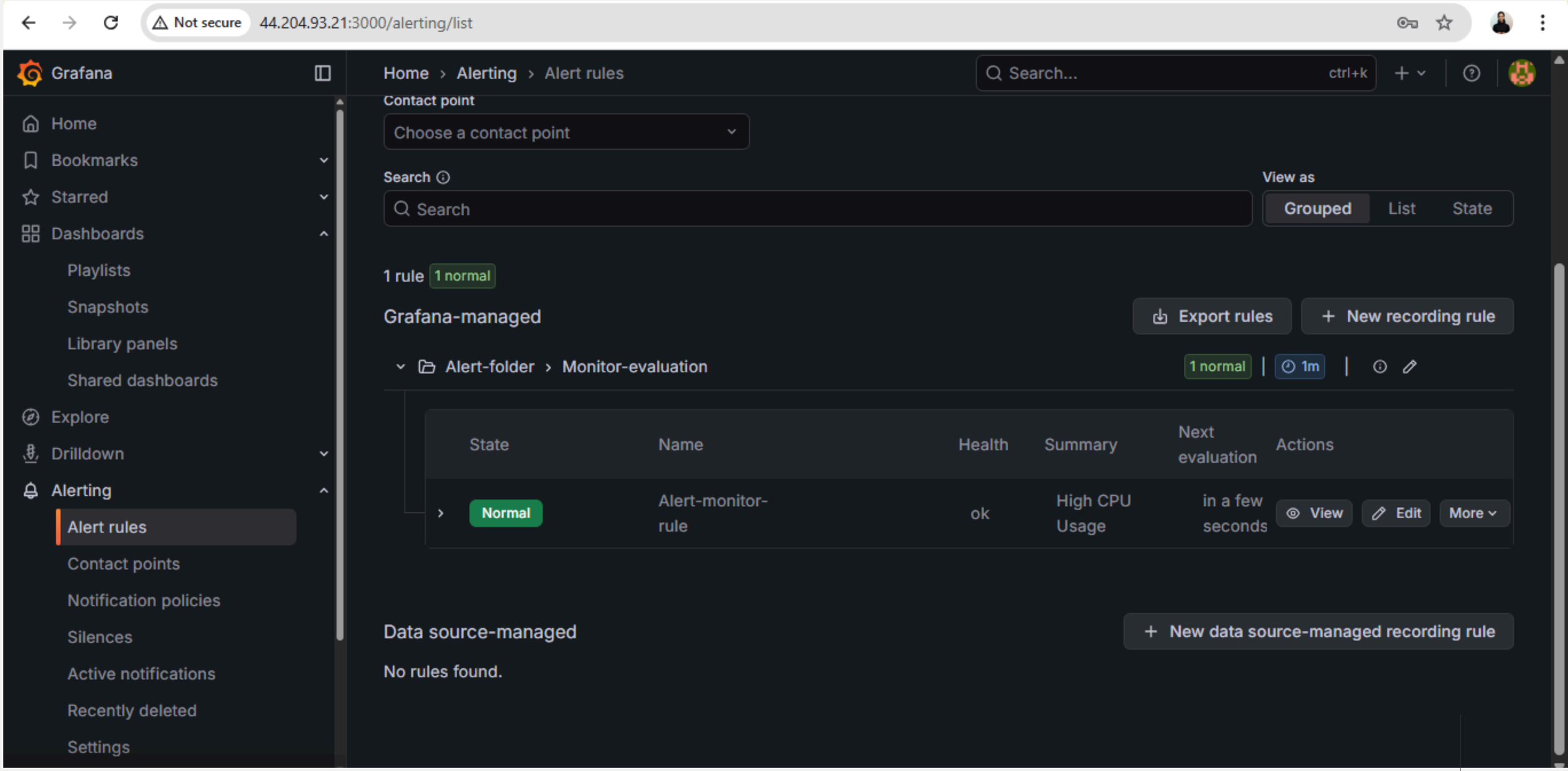

**Logs collected by Promtail are displayed in Grafana through Loki for monitoring and analysis.**




**Logs are organized and displayed in table format in Grafana via Loki for easy analysis**




**Alertmanager rule created, showing all configured settings and routing details.**




**Alert rule status displayed as normal, indicating no active alerts**

# PROJECT SUMMARY

- This project successfully showcases an end-to-end monitoring and alerting framework using modern cloud and container technologies.

- By combining metrics collection, visualization, and alerting, it enables real-time system monitoring, faster troubleshooting, and improved infrastructure reliability.

- The setup demonstrates cloud-native tools like Prometheus, Grafana, Loki, and Alertmanager can work together seamlessly, providing a scalable and efficient solution for monitoring applications.

- Detailed documentation is available in GitHub

# Thank you

PRIYANKA RAJAGOPAL
priyankaraj0919@gmail.com