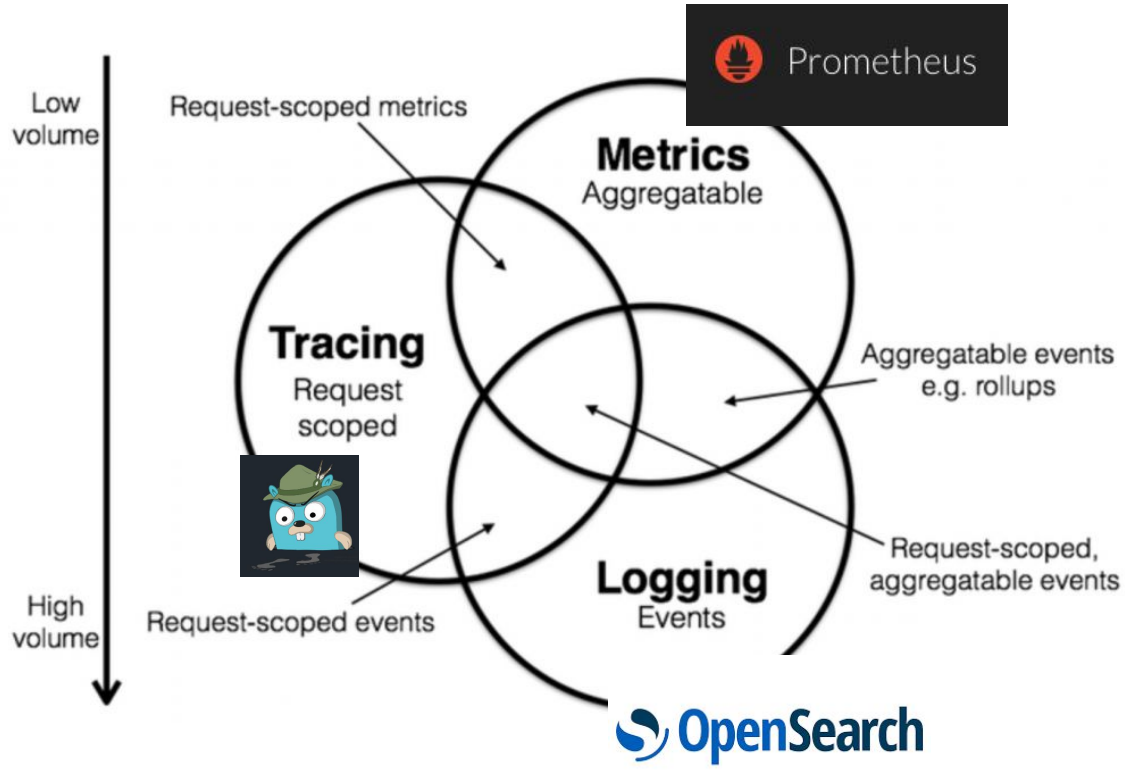
Observabillity (O11y)

phkoan, lichen

Observability - Three Pillars

- Metrics
  - Prometheus
  - Thanos
- Logging
  - OpenSearch
  - Vector
- Tracing
  - Jaeger



Observability - Metrics

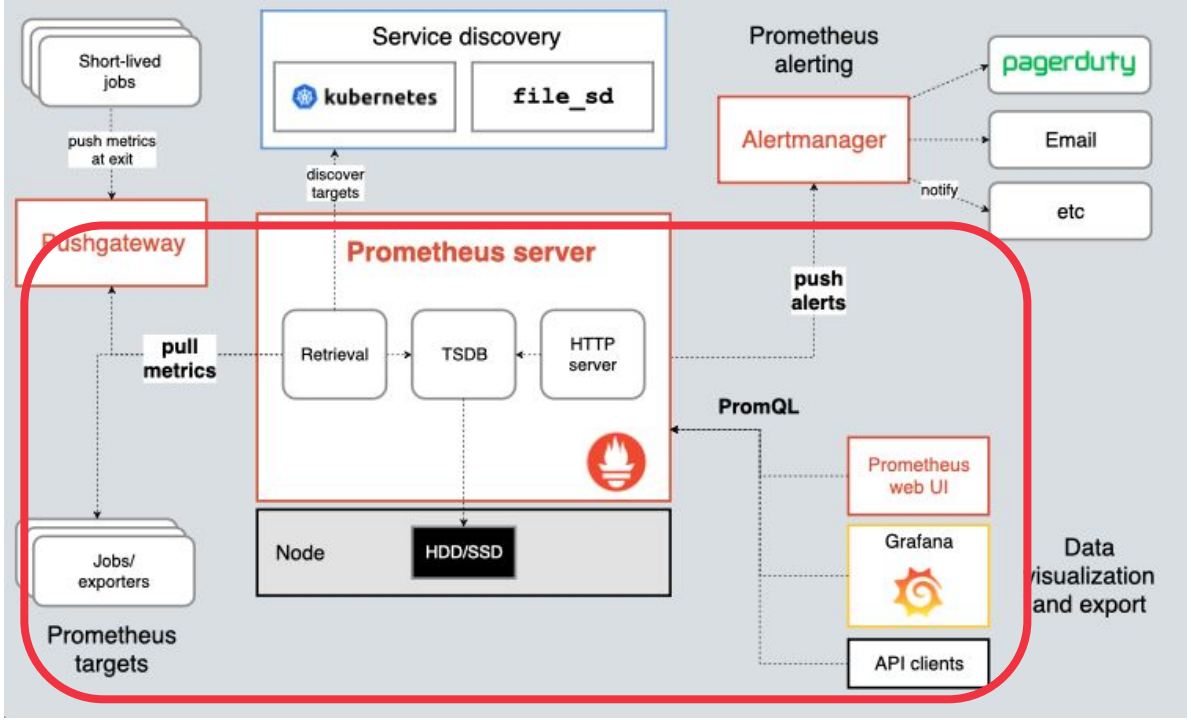
- Metrics are numerical measurements.
- The term *time series* refers to the recording of changes over time
- For example,
  - `node_memory_usage_bytes`
  - `rate(api_server_http_requests_total[5m])`

Observability - Metrics / Prometheus

- Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud.
- Prometheus collects and stores the metrics as time series data.
    - Metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

Observability - Metrics / Prometheus Arch

- Target
- TSDB
- PromQL
- Alert pushing



Observability - Metrics / Type

- Counter
  - A cumulative metric that represents a single monotonically increasing counter whose value can only increase or be reset to zero on restart.
  - E.g., number of requests served, tasks completed, or errors
- Gauge
  - A metric that represents a single numerical value that can arbitrarily go up and down.
  - E.g., temperatures, current memory usage, number of concurrent requests

Observability - Metrics / Type

- Histogram
  - Records observations by counting them in configurable buckets.
  - Prefilter
  - E.g., request durations or response sizes
- Summary
  - Similar to a histogram, a summary samples observations. While it also provides a total count of observations and a sum of all observed values, it calculates configurable quantiles over a sliding time window.
  - Postfilter
  - E.g., request durations and response sizes

Observability - Metrics / PromQL Data Types

- Instant vector
  - A set of time series containing a single sample for each time series, all sharing the same timestamp
  - `http_requests_total`
- Range vector
  - A set of time series containing a range of data points over time for each time series
  - `http_requests_total{}[5m]`
- Scalar - a simple numeric floating point value
- String - a simple string value

Observability - Metrics / PromQL Examples

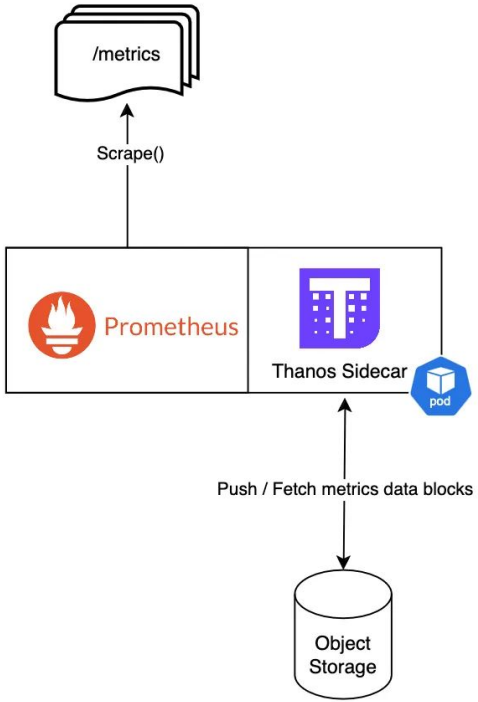
- Filters
  - `http_requests_total{job="apiserver", handler!="/api/comments"}`
  - `http_requests_total{job=~".*server", status!~"4.."}`
- Bulit-in functions & operators
  - `rate(http_requests_total[5m])`
  - `sum by (app, proc) (instance_memory_limit_bytes - instance_memory_usage_bytes) / 1024 / 1024`
  - `topk(3, sum by (app, proc) (rate(instance_cpu_time_ns[5m])))`
  - `count by (app) (instance_cpu_time_ns)`

Observability - Metrics / Thanos

- Prometheus does not store long term metrics
- Thanos: open source, highly available Prometheus setup with long term storage capabilities.
  - Running with Prometheus
- Multiple external storage support, such as S3

Observability - Metrics / Thanos Arch / Sidecar

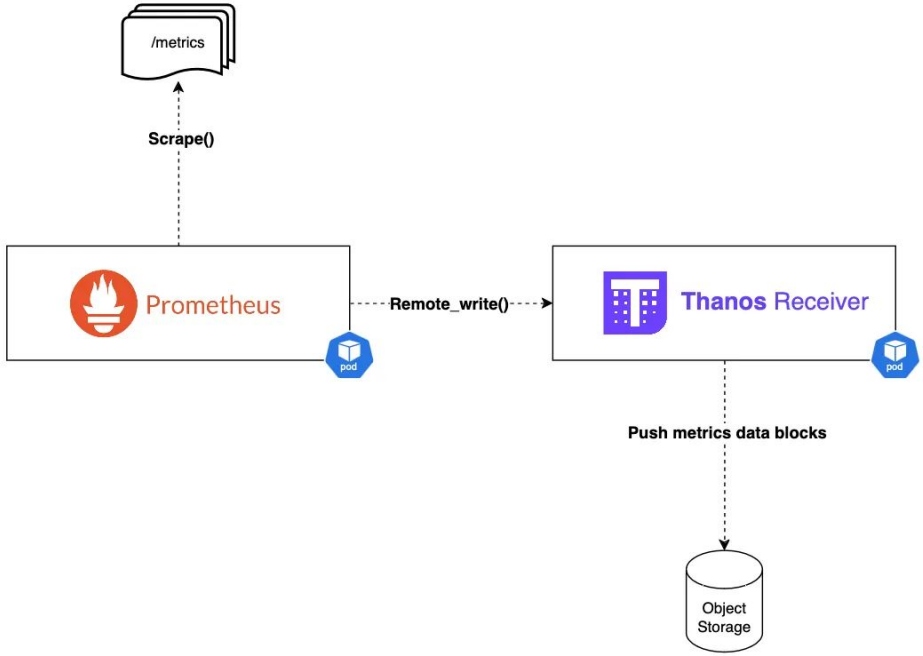
Prometheus works together with Thanos sidecar, which push and fetch metrics data from external storage.



Observability - Metrics / Thanos Arch / Thanos Receiver

Prometheus remote-writes data in stream to Thanos

Receiver for storing.



Observability - Logging

Logging is the act of keeping a log of events that occur in a computer system, such as problems, errors or broad information on current operations.

- Syslog
- `/var/log/nginx/access.log`
- `console.log("log")`
- `kubectl log`

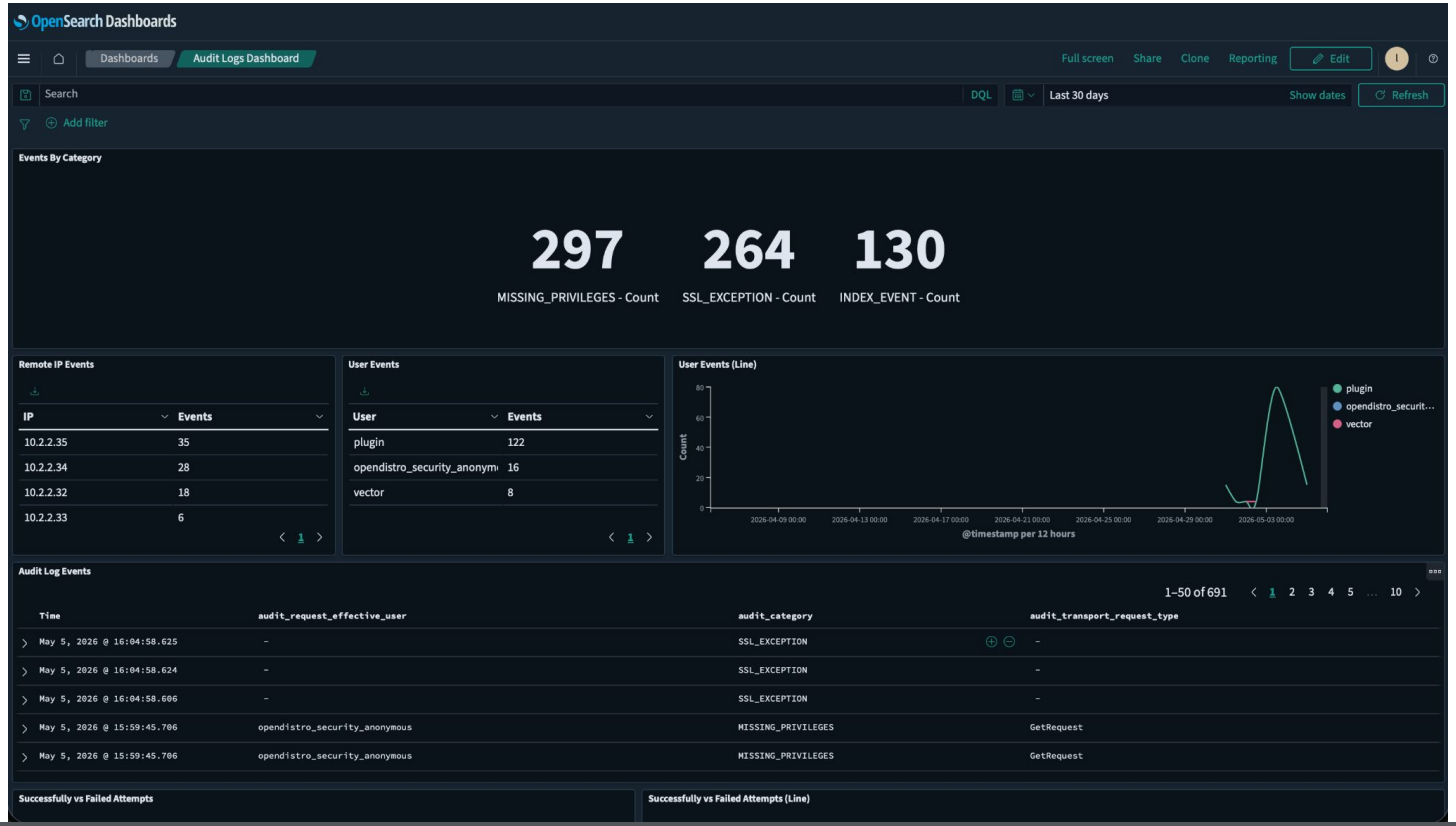
Observability - Logging / OpenSearch

- OpenSearch is a distributed search and analytics engine that supports various use cases.
- Fork from ElasticSearch

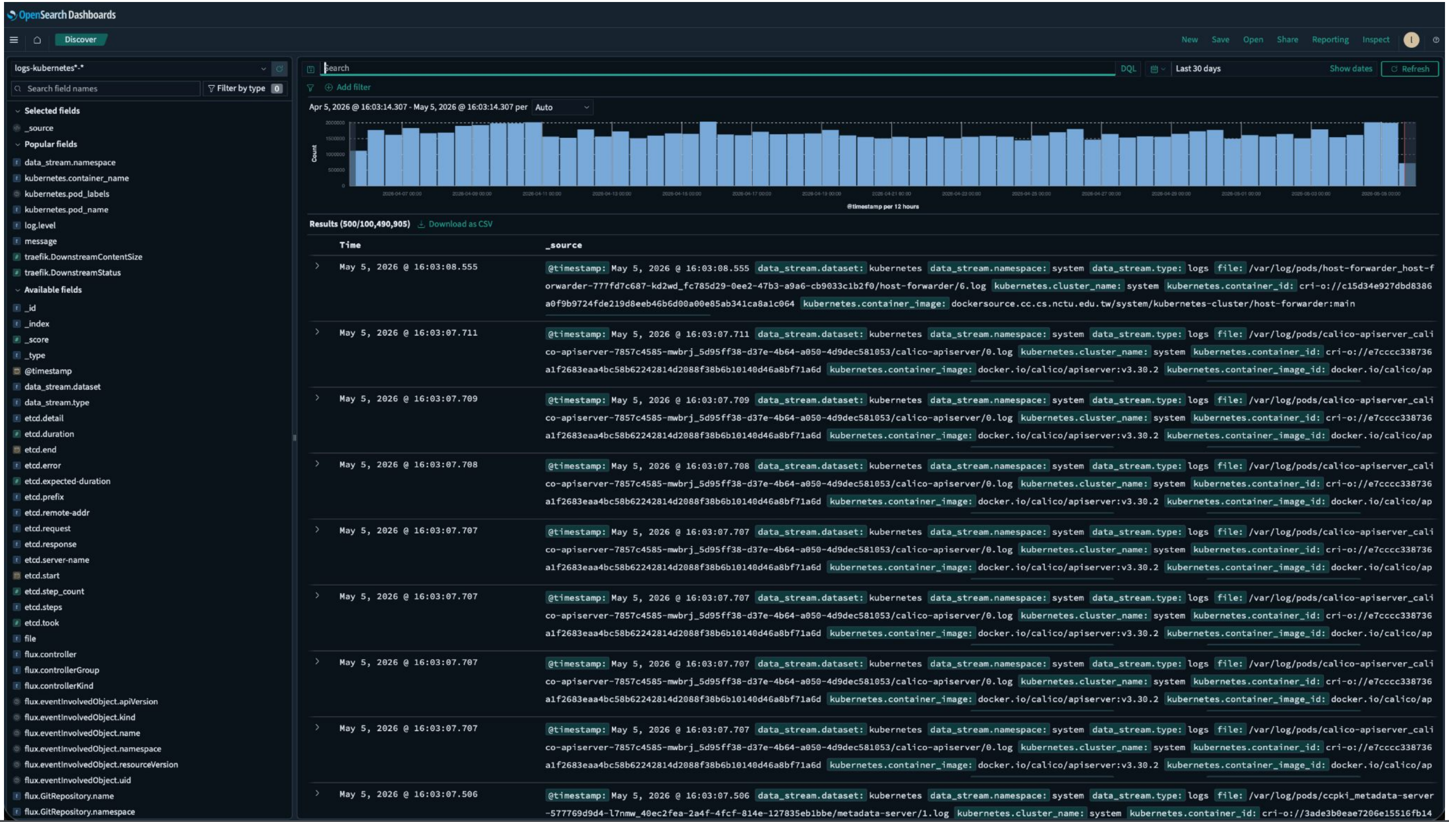
Observability - Logging / OpenSearch Arch

- Document
  - A document is a unit that stores information (text or structured data).
  - In OpenSearch, documents are stored in JSON format.
- Index
  - An index is a collection of documents.

Observability - Logging / Opensearch Dashboard



Observability - Logging / Opensearch Search log

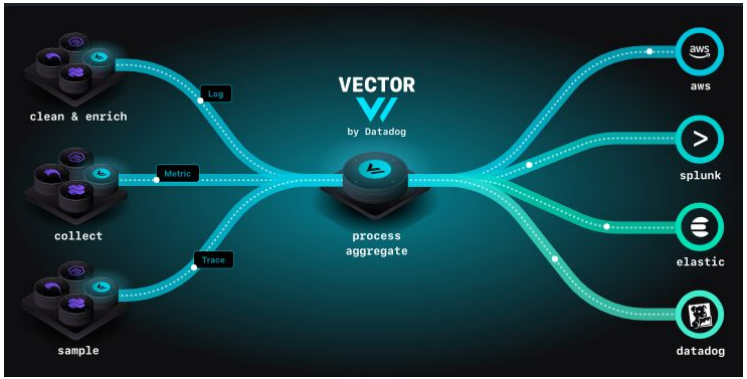


Observability - Logging / Dashboards Query Language (DQL)

- Field-specific search
  - `title: wind`
- Existence of a field
  - `description:*`
- Date range
  - `date >= "1939-01-01" and date <= "2013-12-31"`
- Boolean
  - `not media_type: article`
  - `media_type: film AND page_views: 100 AND title: (wind OR rises)`

Observability - Logging / Vector

- Vector is a high-performance observability **data pipeline** that puts you in control of your observability data.
- It supports various kinds of inputs and outputs.



Observability - Logging / Vector / Components

- Vector composes of three components
  - `source`: the log input
  - `transform`: the transformation code for logs
    - Vector Remap Language (VRL) is a DSL for log processing.
  - `sink`: the log output
- The connectivity among components are defined with YAML.

Observability - Logging / Vector / Components

```
# sources/input_files.yaml
type: socket
mode: unix_datagram
path: /var/lib/log-socket/log
decoding:
  codec: syslog
```

```
{
  "message": "May 10 23:40:12 web01 nginx: GET /index.html 200",
  "file": "/var/log/nginx/access.log",
  "host": "web01"
}
```

```
# transforms/parse_files.yaml
type: remap
inputs: ["input_files"]
source: |
  # if message is syslog format, parse it
  data, error = parse_syslog(.message)
  if error == null {
    . = data
  } else {
    basename = split!(.file, "/")[-1]
    .appname = split!(basename, ".")[0]
  }
```

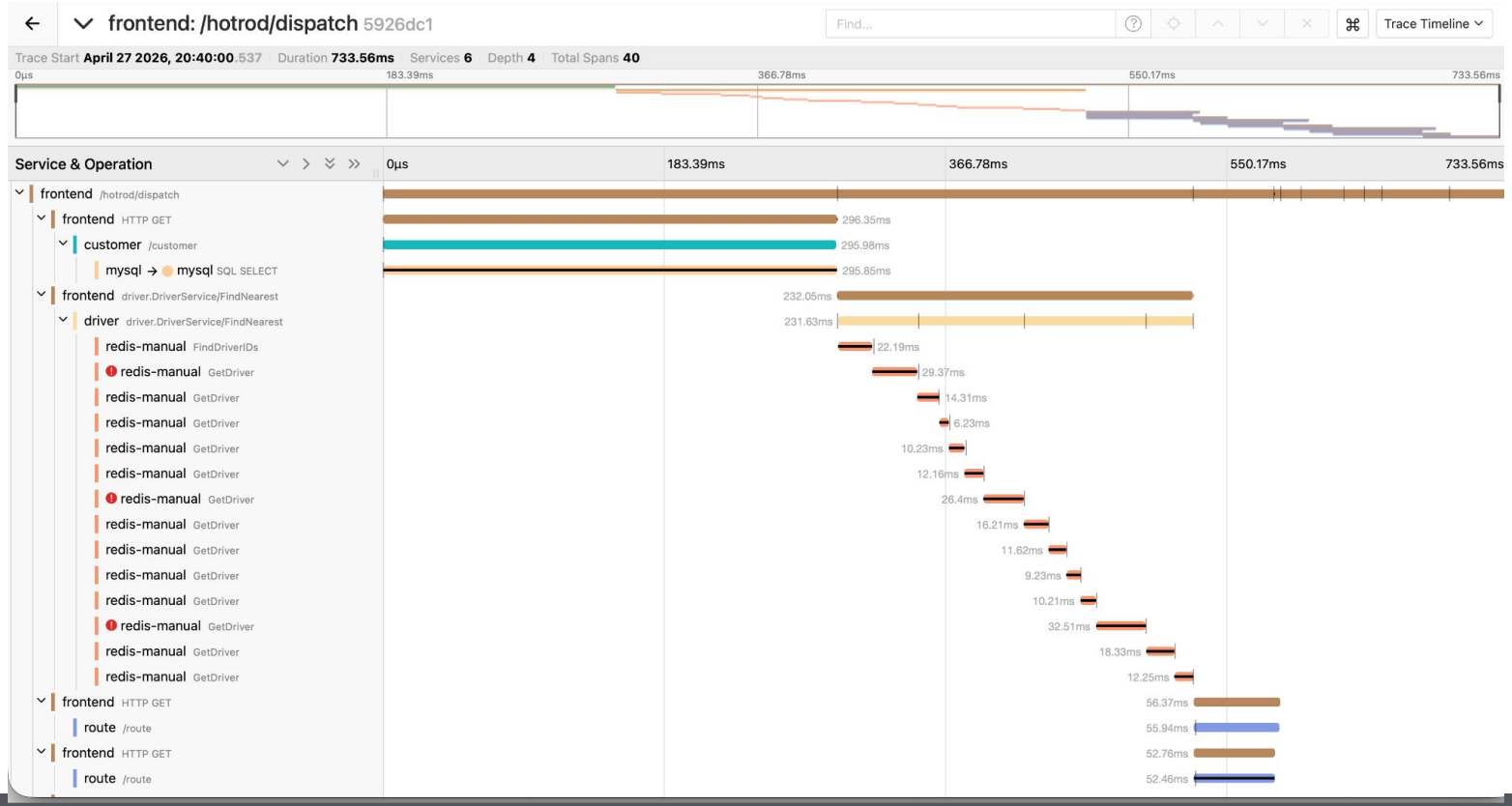
```
# sinks/output-files.yaml
type: "file"
inputs: ["parse_files"]
path: /data/log/archive.log
encoding:
  codec: "text"
```

```
{
  "timestamp": "2026-05-10T23:40:12Z",
  "hostname": "web01",
  "severity": "notice",
  "facility": "auth",
  "appname": "nginx",
  "message": "GET /index.html 200"
}
```

Observability - Tracing/Jaeger

- Jaeger
  - A distributed tracing platform released as open source by Uber Technologies in 2016
  - Donated to CNCF where it is a graduated project.
- With tracing you can:
  - Monitor and troubleshoot distributed workflows
  - Identify performance bottlenecks
  - Track down root causes
  - Analyze service dependencies

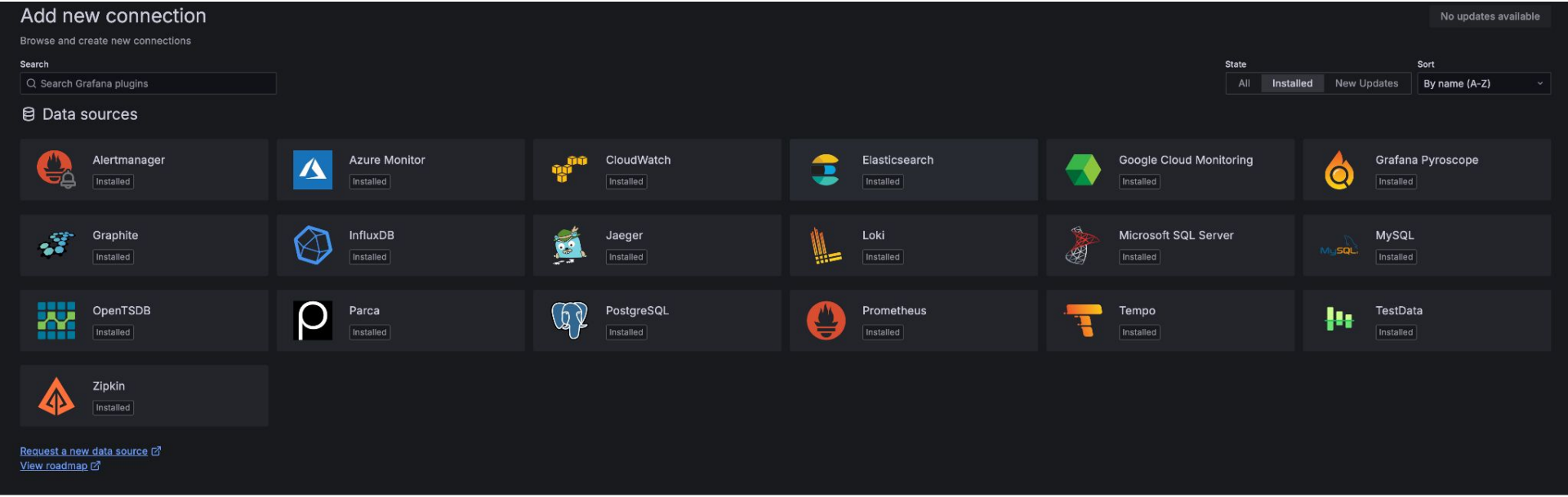
Observability - Tracing/Jaeger UI



Observability - Grafana

- Grafana is a open-source platform for monitoring and observability that allows you to query, visualize, alert on and understand your metrics no matter where they are stored.
- Create, explore, and share dashboards with your team and foster a data-driven culture.

Observability - Grafana / Source



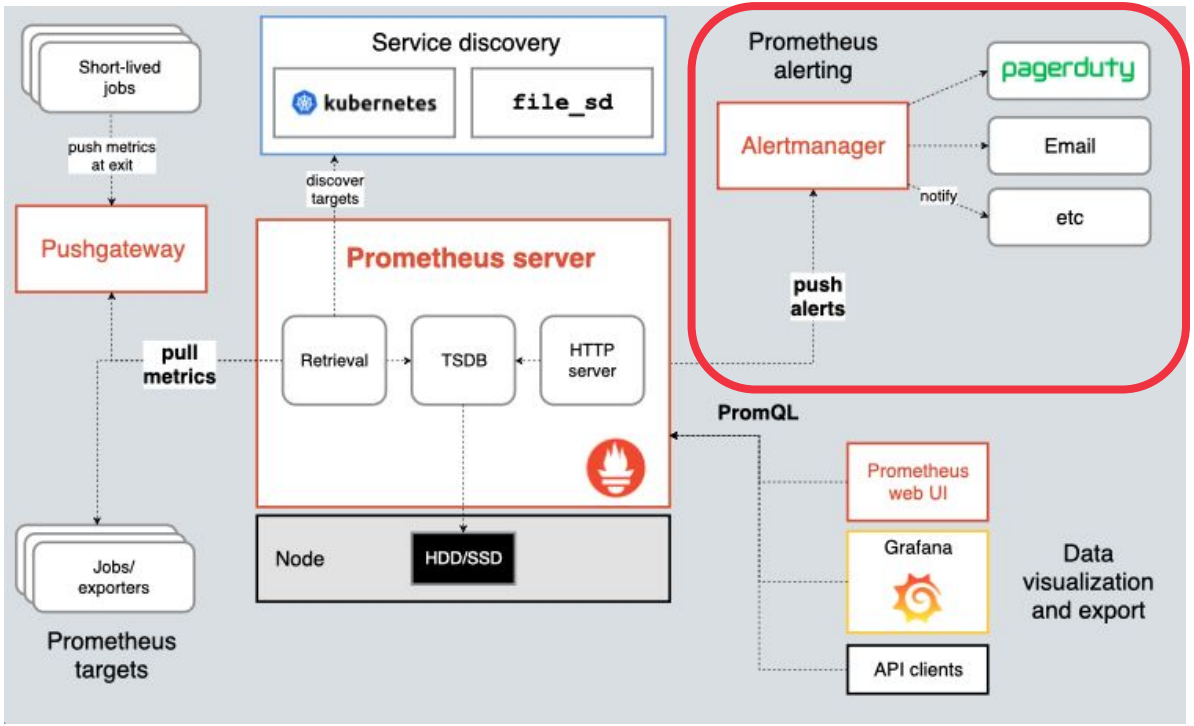
Observability - Grafana / Dashboard

- Import from JSON / [Grafana Dashboards]
- Manual

Observability - Alertmanager

- Alertmanager handles alerts sent by client applications such as the Prometheus server.
- It takes care of deduplicating, grouping, and routing them to the correct receiver integrations such as Email, Slack, etc.
- It also takes care of silencing and inhibition of alerts.
- It uses YAML to define the rules.

Observability - Alertmanager / Source



Observability - Alertmanager / Route

- `route`
  - The matching rules
  - Root route has the default attributes that all sub-routes will inherit.
- `receiver`
  - The medium to receive alerts, such as Email and Slack
- `group_by`
  - Alerts with the same values will be sent together.

```yaml
# The root route with all parameters, which are inherited by the child
# routes if they are not overwritten.
route:
  receiver: 'default-receiver'
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 4h
  group_by: [cluster, alertname]
  # All alerts that do not match the following child routes
  # will remain at the root node and be dispatched to 'default-receiver'.
  routes:
  # All alerts with service=mysql or service=cassandra
  # are dispatched to the database pager.
  - receiver: 'database-pager'
    group_wait: 10s
    matchers:
    - service=~"mysql|cassandra"
  # All alerts with the team=frontend label match this sub-route.
  # They are grouped by product and environment rather than cluster
  # and alertname.
  - receiver: 'frontend-pager'
    group_by: [product, environment]
    matchers:
    - team="frontend"
```

Observability - Alertmanager / Inhibit Rule

- Inhibition allows muting a set of alerts based on the presence of another set of alerts.
- This allows establishing dependencies between systems or services such that only the most relevant of a set of interconnected alerts are sent out during an outage.

Observability - Jsonnet

- A configuration language for app and tool developers to generate config
- A living JSON

example1.jsonnet

```
// Edit me!
{
  person1: {
    name: "Alice",
    welcome: "Hello " + self.name + "!",
  },
  person2: self.person1 { name: "Bob" },
}
```

output.json

```
{
  "person1": {
    "name": "Alice",
    "welcome": "Hello Alice!"
  },
  "person2": {
    "name": "Bob",
    "welcome": "Hello Bob!"
  }
}
```

example2.jsonnet

```
// A function that returns an object.
local Person(name='Alice') = {
  name: name,
  welcome: 'Hello ' + name + '!',
};
{
  person1: Person(),
  person2: Person('Bob'),
}
```

output.json

```
{
  "person1": {
    "name": "Alice",
    "welcome": "Hello Alice!"
  },
  "person2": {
    "name": "Bob",
    "welcome": "Hello Bob!"
  }
}
```

Observability - Kube-Prometheus

- https://github.com/prometheus-operator/kube-prometheus
- Repository collects Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with Prometheus using the Prometheus Operator.
- Official deployment method

Appendix: Observability - Kube-Prometheus / Prometheus

- Prometheus Operator
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/prometheusOperator-deployment.yaml
- Prometheus
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/prometheus-prometheus.yaml
- Prometheus rule
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/alertmanager-prometheusRule.yaml

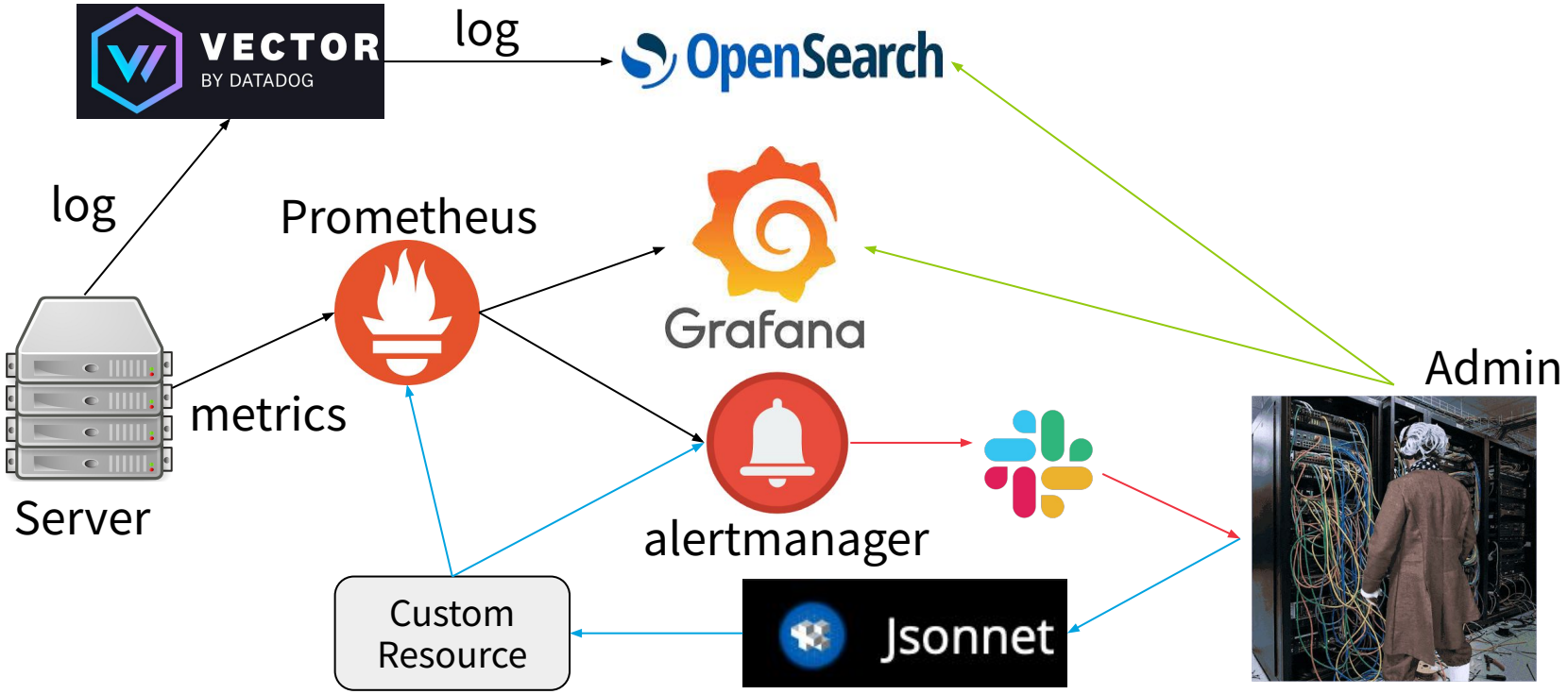
Appendix: Observability - Kube-Prometheus / Grafana

- Grafana
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/grafana-deployment.yaml
- Datasource
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/grafana-dashboardDatasources.yaml
- Dashboard
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/grafana-dashboardSources.yaml

Appendix: Observability - Kube-Prometheus / Alertmanager

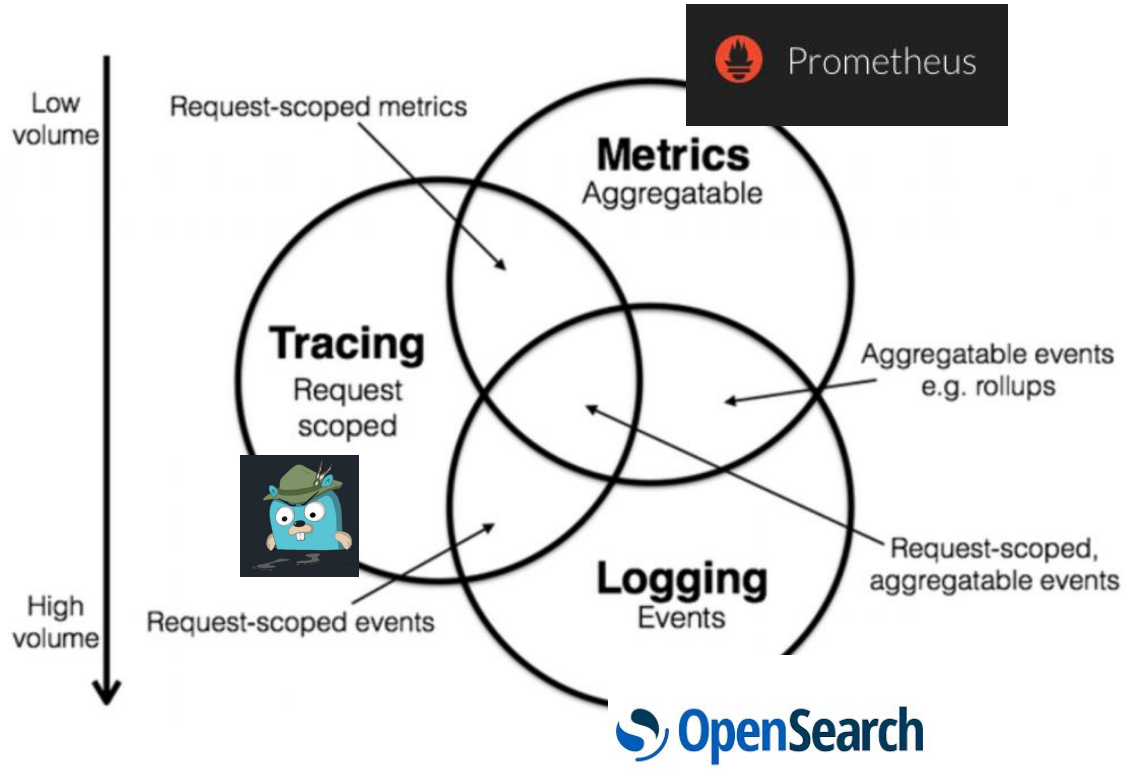
- Alertmanager
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/alertmanager-alertmanager.yaml
- Secret (alertmanager.yaml)
  - https://github.com/prometheus-operator/kube-prometheus/blob/main/manifests/alertmanager-secret.yaml

Observability - Overview



Observability - Three Pillars

- Metrics
  - Prometheus
  - Thanos
- Logging
  - OpenSearch
  - Vector
- Tracing
  - Jaeger



Reference

- https://docs.docker.com/
- https://opencontainers.org/
- https://d7y.io/
- https://prometheus.io/
- https://thanos.io/
- https://opensearch.org/
- https://vector.dev/
- https://www.jaegertracing.io/
- https://grafana.com/
- https://jsonnet.org/
- https://cncf.io/
- https://microsoft.github.io/
- https://awsmorocco.com/thanos-deep-dive-addressing-prometheus-limitations-at-scale-84a6b02b01bc?gi=c4519d136b4c