Kubernetes Advanced

kubernetes

Monitoring

kubernetes

Session Contents

- Use Kubectl
- Kubernetes Dashboard
- Log management
- Prometheus & Grafana
- Autoscaling

kubernetes

Use Kubectl

Kubectl for monitoring

- Many times kubectl can be the first (and only) tool available to access the cluster
- Cannot be used to have a proactive approach to monitoring but can be used to get detailed understanding about cluster issues
- With the needed permissions you can have complete understanding about cluster behavior

Kubectl describe

```
kubectl describe pod <pod> [-n <namespace>]
```

- Shows details about pod
  - Metadata
  - Network
- Lists all events occurred during pod lifecycle
- First place to go when pod don’t have “Running” status

kubernetes

Kubectl logs

```
kubectl logs <pod> [-n <namespace>]
```

- Shows pod stdout and stderr
- Flag –f blocks the console and show new lines

Kubectl port-forward

```
kubectl port-forward pod <pod> [-n <ns>] hostport:podPort
kubectl port-forward svc <svc> [-n <ns>] hostport:podPort
```

- Maps a port on machine with pod port
- Allow to make direct requests
- When using service, maps directly to only on container (no load balancing)

kubernetes

Kubectl top

```
kubectl top node <node>
kubectl top pod <pod> [-n <ns>]
```

- Display resource (CPU/memory) usage of the resources (nodes or pods)
- Due to the metrics service delay, they may be unavailable for a few minutes since pod creation
- Use native Kubernetes metrics server

Kubernetes Dashboard

Kubernetes Dashboard

- Web-based Kubernetes user interface to have a more user-friendly way to look into your cluster.
- Kubernetes Dashboard were created and is maintained by Kubernetes Community
- Initially, was the only Web-based tool to monitor your cluster
- Now, is not so used on production environment
  - Newer and better tools arrive on Kubernetes Landscape
  - Limitation on metrics since it uses only Kubernetes Vanilla metrics
- You can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster resources.

kubernetes

Kubernetes Dashboard

- You can use Dashboard to:
  - Monitor your cluster resources
  - Manage Kubernetes resources
  - Get an overview of applications running on your cluster
  - Troubleshoot your containerized application
  - Deploy containerized applications to a Kubernetes cluster
- Provide wizards to scale a Deployment, initiate a rolling update, restart a pod or deploy new applications using a deploy wizard.

Demo | Kubernetes Dashboard

Log Management

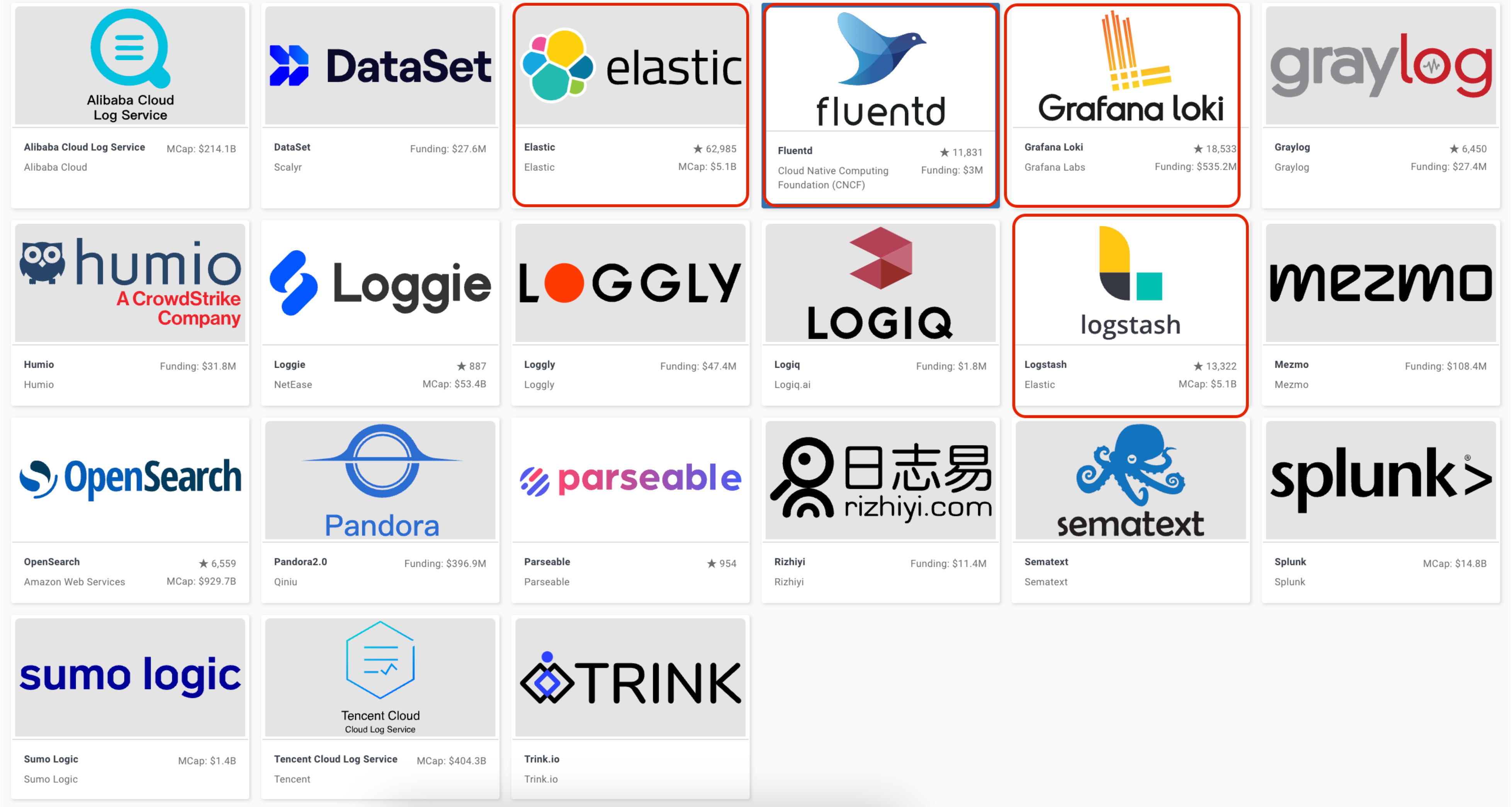
Motivation

- Containers runs on top of an ephemeral layer that is deleted each time a pod is deleted
- If you write your logs to a file in this layer you may lose them
- Even during execution can be hard to reach them
- How to have access to these logs and keep them for as long as needed?

Logs on Kubernetes

- As a best practice, everything that needs to be logged should be write to standard output or standard error of each container
- Log Management tools for Kubernetes uses a concept of creating a DaemonSet to have a pod in each node to access to those streams
- Then, after collecting the data, send them to a centralized server where the logs are kept for as long as needed
- Finally, the full solution have a visualization layer where logs can be queried and accessed from outside of the cluster

CNCF Logging



Alibaba Cloud Log Service MCap: $214.1B
Alibaba Cloud

DataSet Funding: $27.6M
Scalyr

Elastic ★ 62,985
Elastic MCap: $5.1B

Fluentd ★ 11,831
Cloud Native Computing Foundation (CNCF) Funding: $3M

Grafana Loki ★ 18,533
Grafana Labs Funding: $535.2M

Graylog ★ 6,450
Graylog Funding: $27.4M

Humio Funding: $31.8M
Humio

Loggie ★ 887
NetEase MCap: $53.4B

Loggly Funding: $47.4M
Loggly

Logiq Funding: $1.8M
Logiq.ai

Logstash ★ 13,322
Elastic MCap: $5.1B

Mezmo Funding: $108.4M
Mezmo

OpenSearch ★ 6,559
Amazon Web Services MCap: $929.7B

Pandora2.0 Funding: $396.9M
Qiniu

Parseable ★ 954
Parseable

Rizhiyi Funding: $11.4M
Rizhiyi

Sematext
Sematext

Splunk MCap: $14.8B
Splunk

Sumo Logic MCap: $1.4B
Sumo Logic

Tencent Cloud Log Service MCap: $404.3B
Tencent

Trink.io
Trink.io

ELK stack

- One common platform for logging is ELK: Elastic Search, Logstash and Kibana
- Logstash for log gathering and store
- Elastic Search to provide a query layer on top of logs
- Kibana a visualization platform

Fluentd

- Fluentd is an open source data collector for unified logging layer
- Is getting more space on logging inside a Kubernetes cluster
- Less resource consumption compared with ELK
- More integrated with components outside the cluster

Prometheus & Grafana

Motivation

- Kubernetes grant basic metrics about pods (memory, cpu)
- Those metrics are not sufficient when you want to have a better monitorization from your cluster and your applications
- Not only you need to gather new metrics, but you need a better visualization for them
- Dynamic and sharable dashboards are crucial for an efficient and proactive monitorization of any system and infrastructure

kubernetes

Prometheus

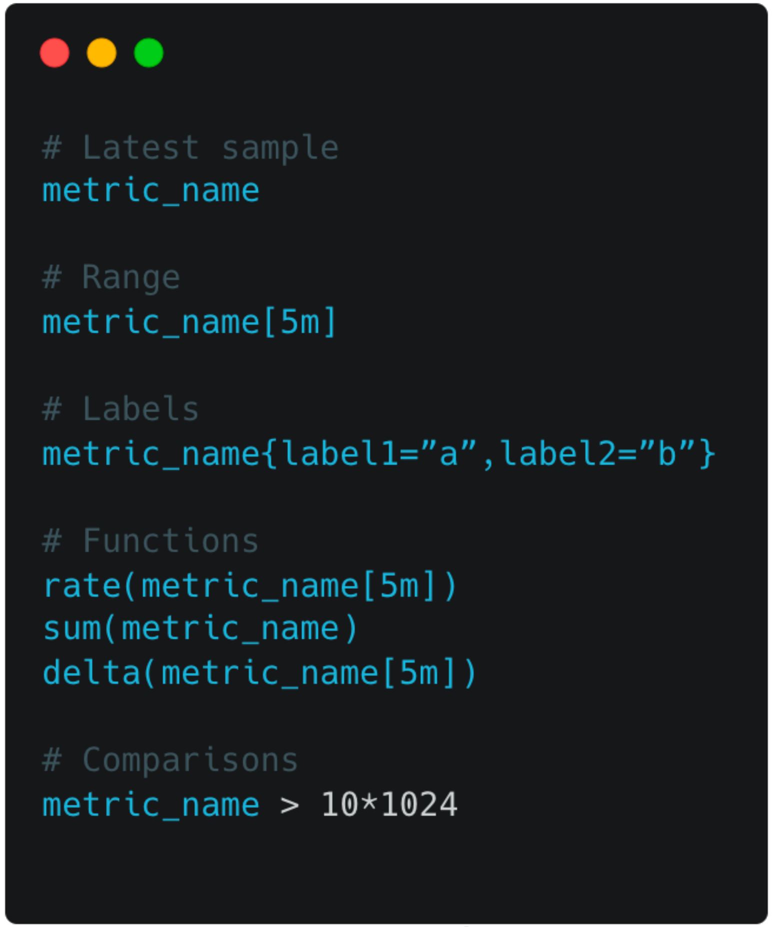
- Prometheus an open-source systems monitoring and alerting toolkit originally built at SoundCloud.
- It is now a standalone open source project and maintained independently of any company.
- Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.
- Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

Prometheus Features

- Prometheus's main features are:
  - a multi-dimensional data model with time series data identified by metric name and key/value pairs
  - PromQL, a flexible query language to leverage this dimensionality
  - no reliance on distributed storage; single server nodes are autonomous
  - time series collection happens via a pull model over HTTP
  - pushing time series is supported via an intermediary gateway
  - targets are discovered via service discovery or static configuration
  - multiple modes of graphing and dashboarding support

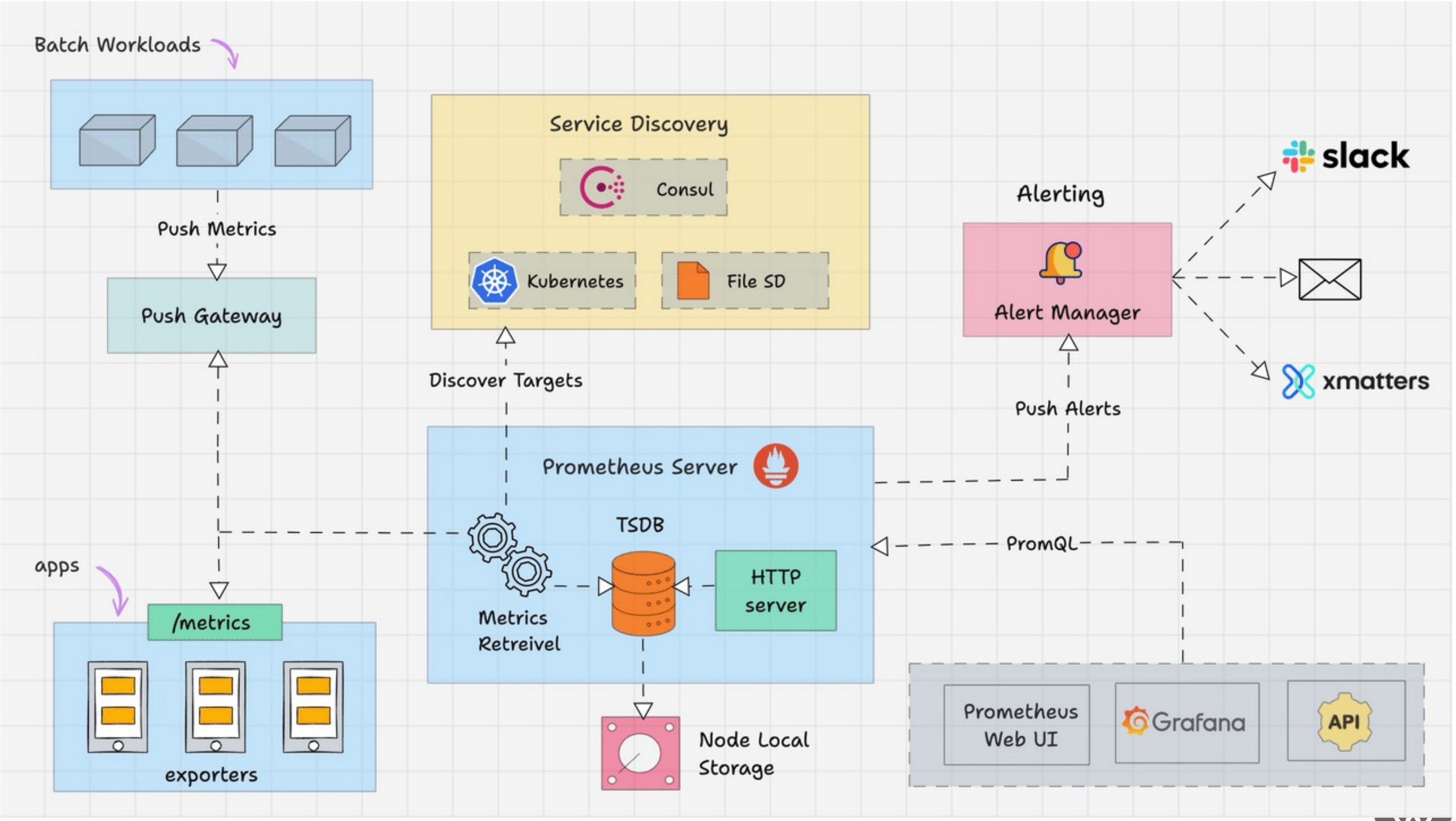
Prometheus PromQL

- Prometheus provides a functional query language called PromQL (Prometheus Query Language) that lets the user select and aggregate time series data in real time.
- The result of an expression can either be shown as a graph, viewed as tabular data in Prometheus's expression browser, or consumed by external systems via the HTTP API



```
# Latest sample
metric_name

# Range
metric_name[5m]

# Labels
metric_name{label1=”a”,label2=”b”}

# Functions
rate(metric_name[5m])
sum(metric_name)
delta(metric_name[5m])

# Comparisons
metric_name > 10*1024
```

Prometheus Architecture



kubernetes

Prometheus Components

Prometheus server which scrapes and stores time series data

Client libraries for instrumenting application code

Push gateway for supporting short-lived jobs

Alertmanager to handle alerts

Visualization tools are external but integration with Grafana is natural

Grafana

- Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored.
- Allow you to unify your data from several sources and make interactive dashboards
- Have a great linkage with Prometheus using PromQL to create dashboards
- Dashboards are described JSON what made really easy to share between the community

Demo | Prometheus & Grafana

Autoscaling

Motivation

- Kubernetes can handle several replicas of the same pods
  - ReplicaSets handle replication
  - Services handle load balancing between them
- However, if the demand of a service starts to grow, the number of replicas deployed may be not sufficient to handle requests
- Number of replicas can be changed manually but it’s not scalable
- Kubernetes have a HorizontalPodAutoscaller (HPA) object to handle scalability of a Deployment automatically

Horizontal Pod Autoscaler

- Horizontal scaling means that the response to increased load is to deploy more Pods
- HPA defines a minimum and maximum number of replicas
- If the load increases, and the number of Pods is below the configured maximum, the HPA instructs the Deployment to scale up
- If the load decreases, and the number of Pods is above the configured minimum, the HPA instructs the workload resource to scale down
- HPA uses a control loop with a definedinterval (default is 15 seconds) to check if some change is needed

kubernetes

Horizontal Pod Autoscaler

- To make the decision about scaling, HPA uses metrics about pods resources (CPU, Memory) utilization
- Metric target can be set as a percentage or raw value (preferable)
- Percentage value: controller calculates the utilization value as a percentage of the equivalent resource request
- Raw value: metric values are used directly

kubernetes

Horizontal Pod Autoscaler

- HPA uses a mean of the utilization or the raw value across all targeted Pods, and produces a ratio used to scale the number of desired replicas.
- Algorithm uses the following formula

  desiredReplicas = ceil[currentReplicas * (currentMetricValue / desiredMetricValue)]

- If metrics cannot be gathered from one pod, is considered as using 0% for scale up and 100% for scale down

kubernetes

HPA Metrics

- HPA can use 3 types of metrics to make the decision to scale up/down: per-pod resource metrics, custom metrics and external metrics
- Per-pod resource metrics: metrics gathered by native Metrics Server, like CPU, Memory and, GPU (near future)
- Custom metrics: metrics gathered by metrics scrappers plugin installed on the cluster, like Prometheus. For instance, you can autoscale your pods based on number of requests.
- External metrics: metrics that can be gathered from external resources using an additional plugin like Prometheus. For instance, you can autoscale your pods based on queued messages on a message queue.

kubernetes

HPA with Multiple Metrics

- You may specify more than one metric to be analyzed by HPA
- HPA makes the calculation for each metric and then select the biggest replica number from those calculations
- HPA can use multiple metrics from different sources

kubernetes

Other types of autoscaling

- Vertical Pod Autoscaler: adjusts the resource requests and limits of a container
- Cluster Autoscaler: adjusts the number of nodes of a cluster
- Both tools are defined and maintained by Kubernetes community but not available on a vanilla Kubernetes cluster
- VPA is fully implemented by Kubernetes community code
- Cluster Autoscaler needs to have specific implementation from nodes provider (and really hard to implement in on-prem cluster with bare metal ☺)

Questions?

kubernetes