Reveal Your Deepest Kubernetes ~~Secrets~~ Metrics

KubeCon 2017 Prometheus Salon- 12/6/2017
Bob Cotton - FreshTracks.io

About Me

- Co-Founder - FreshTracks.io - A CA Accelerator Incubation
- bob@freshtracks.io
- @bob_cotton

FRESHTRACKS.io

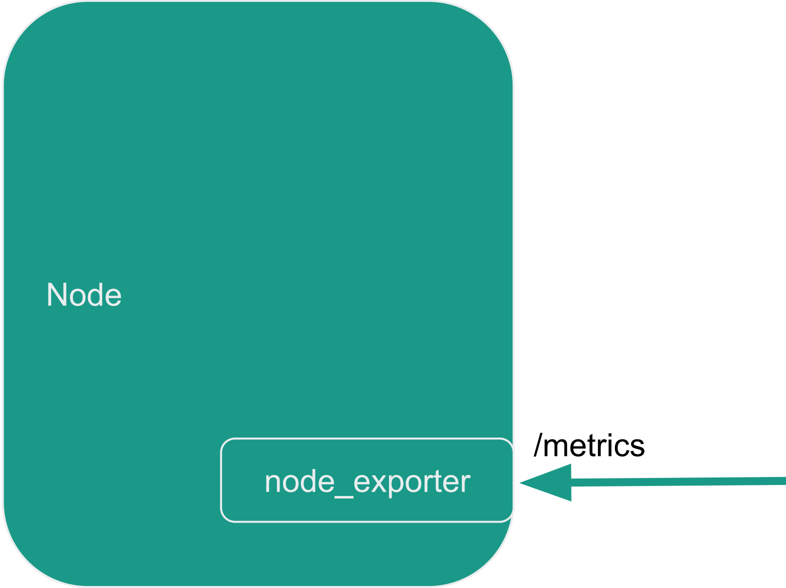
Agenda

- Sources of metrics
  - Node
  - kubelet and containers
  - Kubernetes API
  - etcd
  - Derived metrics (kube-state-metrics)
- The new K8s metrics server
- Horizontal pod auto-scaler
- Prometheus re-labeling and recording rules
- K8s cluster hierarchies and metrics aggregation

Sources of Metrics in Kubernetes

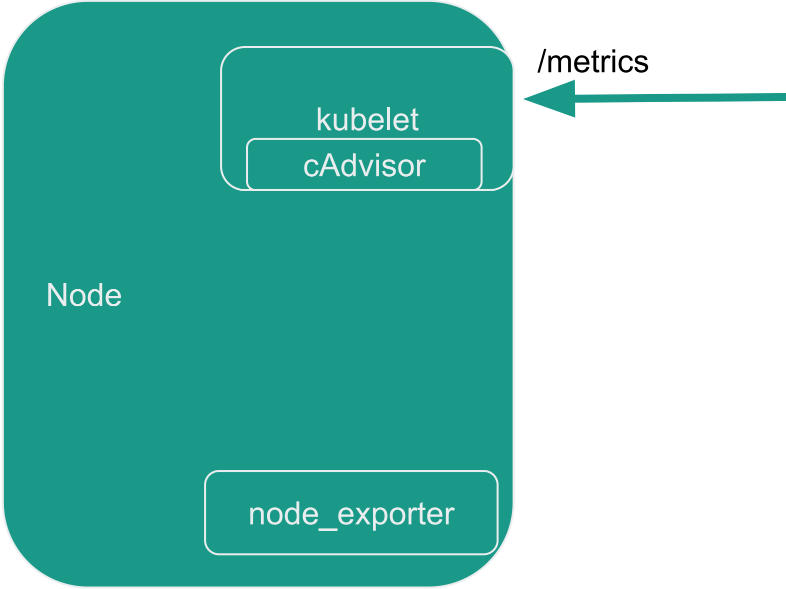
Host Metrics from the node_exporter

- Standard Host Metrics
  - Load Average
  - CPU
  - Memory
  - Disk
  - Network
  - Many others
- ~1000 Unique series in a typical node



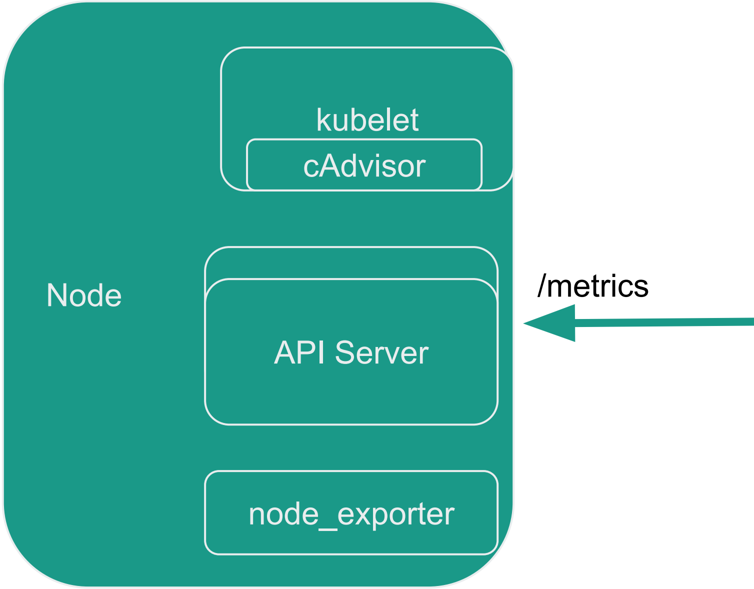
Container Metrics from cAdvisor

- cAdvisor is embedded into the kubelet, so we scrape the kubelet to get container metrics
- These are the so-called “core” metrics
- For each container on the node:
  - CPU Usage (user and system) and time throttled
  - Filesystem read/writes/limits
  - Memory usage and limits
  - Network transmit/receive/dropped



Kubernetes Metrics from the K8s API Server

- Metrics about the performance of the K8s API Server
  - Performance of controller work queues
  - Request Rates and Latencies
  - Etcd helper cache work queues and cache performance
  - General process status (File Descriptors/Memory/CPU Seconds)
  - Golang status (GC/Memory/Threads)



Etcd Metrics from etcd

- Etcd is “master of all truth” within a K8s cluster
  - Leader existence and leader change rate
  - Proposals committed/applied/pending/failed
  - Disk write performance
  - Network and gRPC counters

K8s Derived Metrics from kube-state-metrics

- Counts and meta-data about many K8s types
  - Counts of many “nouns”
  - Resource Limits
  - Container states
    - ready/restarts/running/terminated/waiting
  - _labels series just carries labels from Pods

- `cronjob`
- `daemonset`
- `deployment`
- `horizontalpodautoscaler`
- `job`
- `limitrange`
- `namespace`
- `node`
- `persistentvolumeclaim`
- `pod`
- `replicaset`
- `replicationcontroller`
- `resourcequota`
- `service`
- `statefulset`

Sources of Metrics in Kubernetes

- Node via the node_exporter
- Container metrics via the kubelet and cAdvisor
- Kubernetes API server
- etcd
- Derived metrics via kube-state-metrics

Scheduling and Autoscaling i.e. The Metrics Pipeline

The New “Metrics Server”

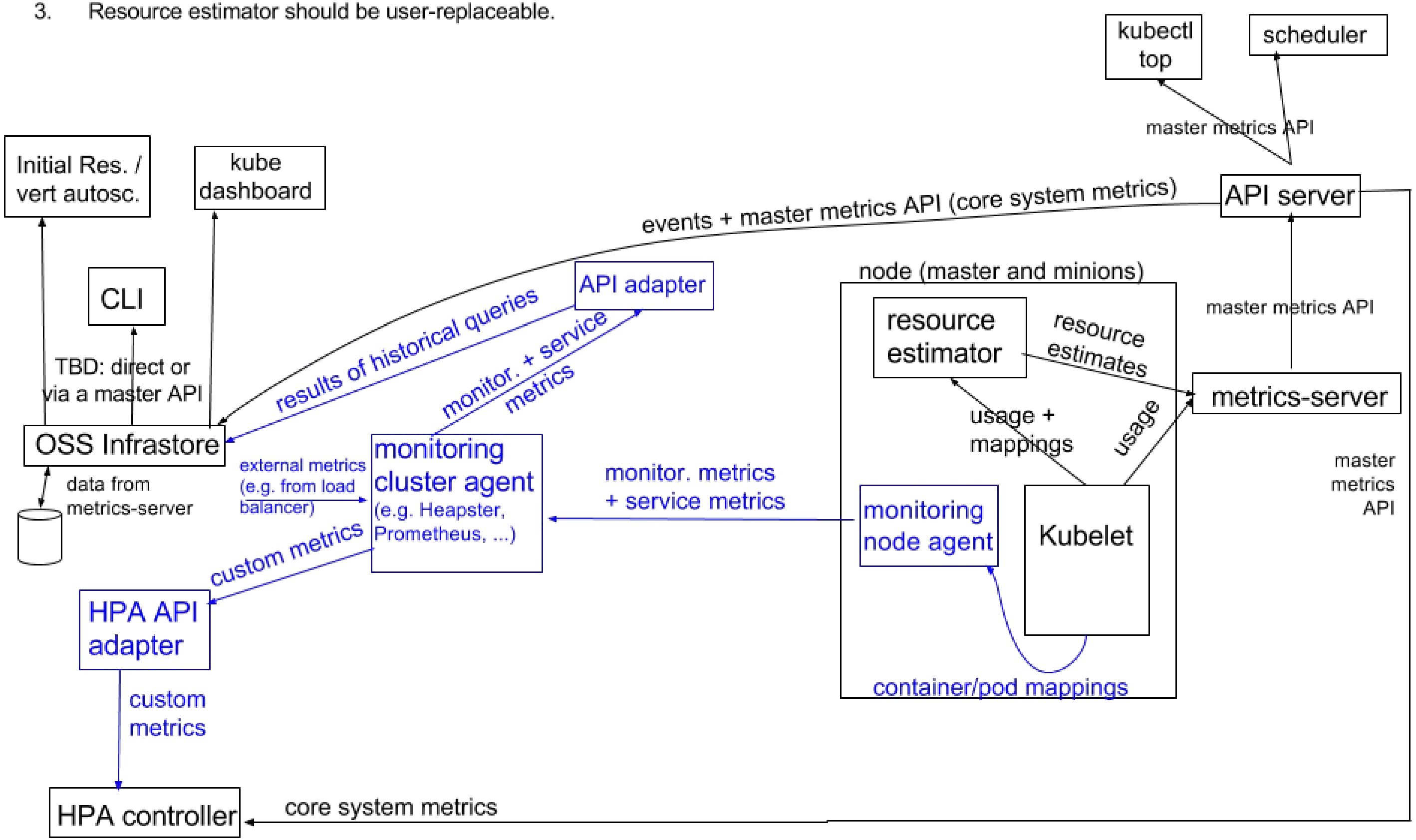
- Replaces Heapster
- Standard (versioned and auth) API aggregated into the K8s API Server
- In “beta” in K8s 1.8
- Used by the scheduler and (eventually) the Horizontal Pod Autoscaler
- A stripped-down version of Heapster
- Reports on “core” metrics (CPU/Memory/Network) gathered from cAdvisor
- For internal to K8s use only.
- Pluggable for custom metrics

Monitoring architecture proposal: OSS

(arrows show direction of metrics flow)

Notes

1. Arrows show direction of metrics flow.
2. Monitoring pipeline is in blue. It is user-supplied and optional.
3. Resource estimator should be user-replaceable.



Feeding the Horizontal Pod Autoscaler

- Before the metrics server the HPA utilized Heapster for it’s Core metrics
  - This will be the metrics-server going forward
- API Adapter will bridge to third party monitoring system
  - e.g. Prometheus

Labels, Re-Label and Recording Rules Oh My...

Metric Metadata

In the beginning:

<metric name> = <metric value>

```
    http_requests_total = 1.4
```

Increased complexity lead to workarounds

```
    region.az.instance_type.instance.hostname.http_requests_total = 5439
```

Metric Metadata - Metrics 2.0

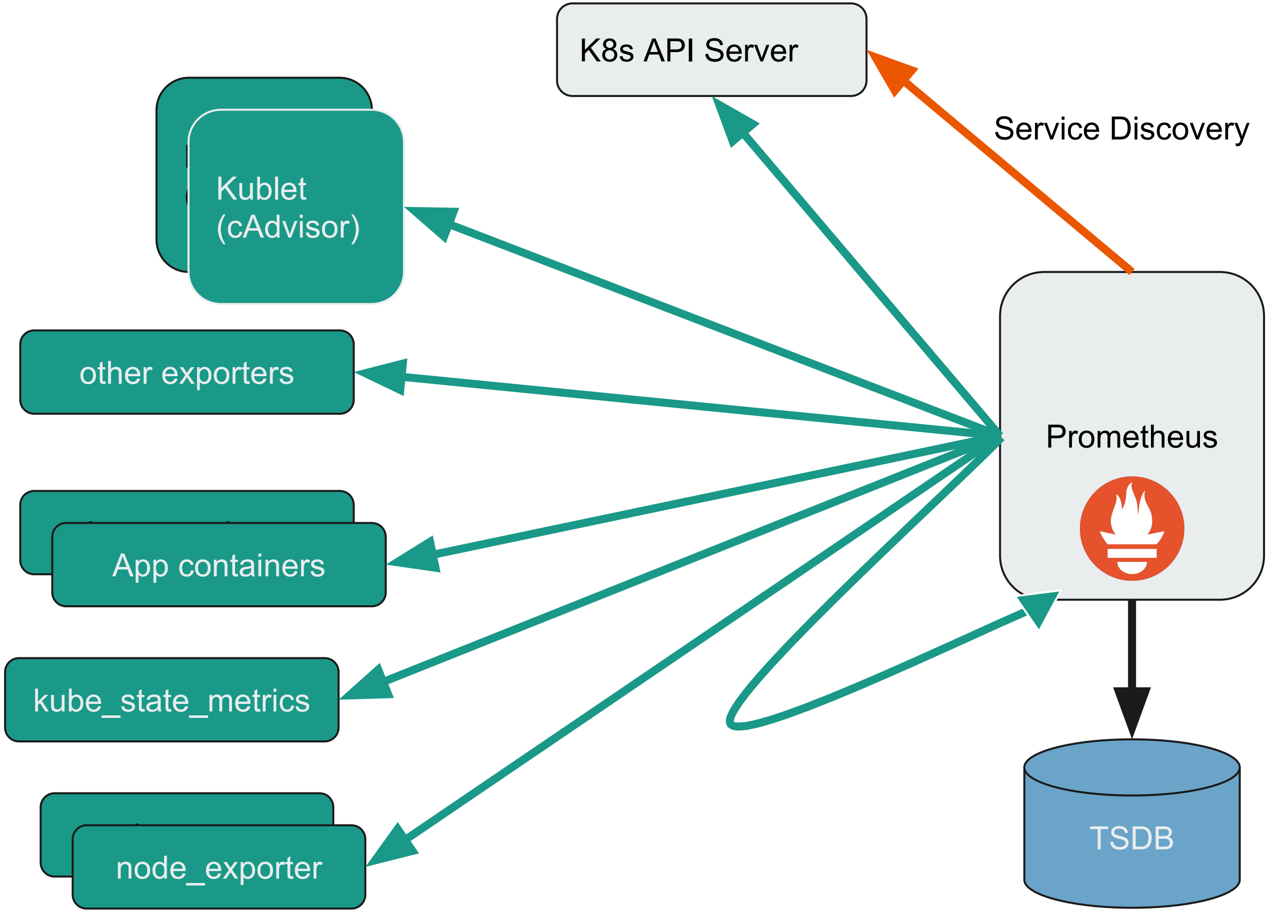
```
us-east.us-east-1.m2_xlarge.i-3582k8.host1.http_requests_total = 5439
```

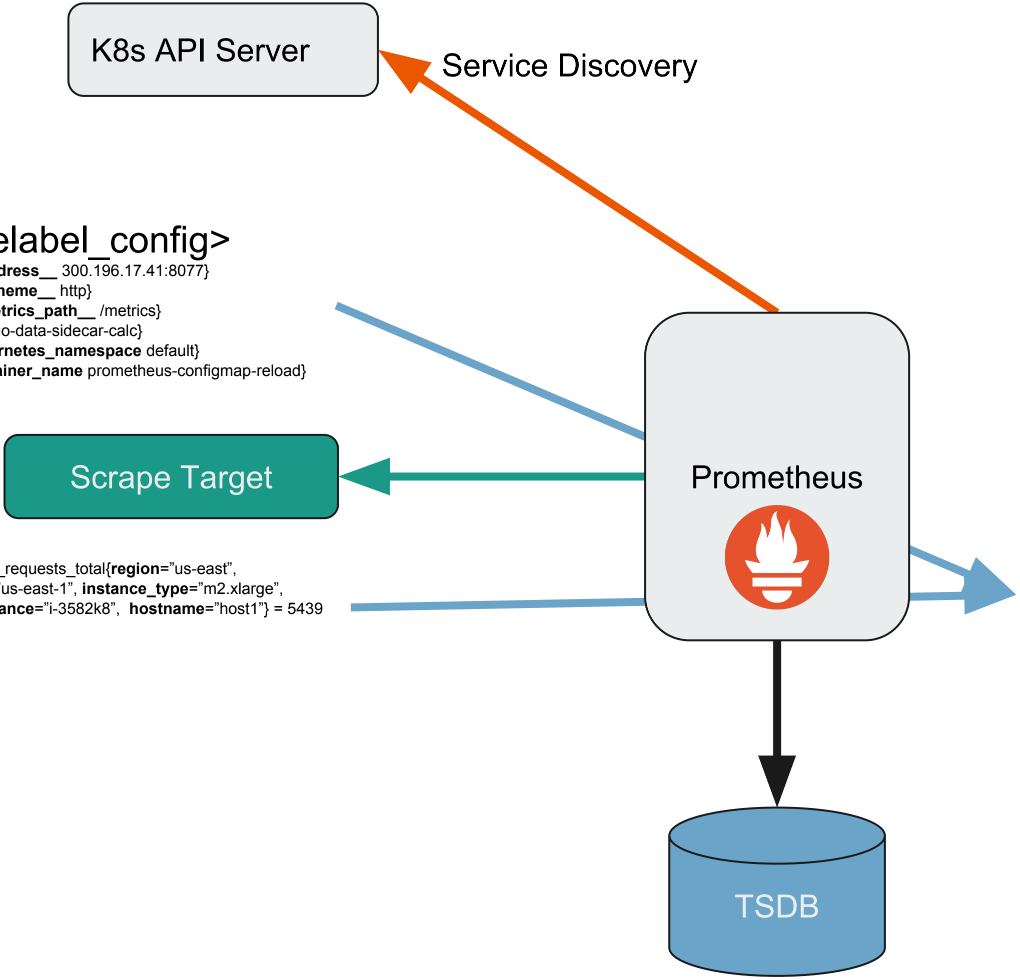
```
http_requests_total{region=” us-east”,
        az=”us-east-1”,
        instance_type=” m2.xlarge”,
        instance=” i-3582k8”,
        hostname=” host1”} = 5439
```

Kubernetes Labels

- Kubernetes gives us labels on all the things
- Our scrape targets live in the context of the K8s labels
- We want to enhance the scraped metric labels with K8s labels

- This is why we need relabel rules in Prometheus


K8s API Server
Service Discovery
Kublet
(cAdvisor)
other exporters
Prometheus
App containers
kube_state_metrics
node_exporter
TSDB



0="{**__address__** 300.196.17.41}"
1="{**__meta_kubernetes_namespace** default}"
2="{**__meta_kubernetes_pod_annotation_freshtracks_io_data_sidecar** true}"
3="{**__meta_kubernetes_pod_annotation_freshtracks_io_path** /metrics2}"
4="{**__meta_kubernetes_pod_annotation_kubernetes_io_created_by** "kind":"SerializedReference"?}"
5="{**__meta_kubernetes_pod_annotation_kubernetes_io_limit_ranger** LimitRanger plugin set: cpu request for container prometheus-configmap-reload; cpu request for container data-sidecar}"
6="{**__meta_kubernetes_pod_annotation_prometheus_io_port** 8077}"
7="{**__meta_kubernetes_pod_annotation_prometheus_io_scrape** false}"
8="{**__meta_kubernetes_pod_container_name** prometheus-configmap-reload}"
9="{**__meta_kubernetes_pod_host_ip** 172.20.42.119}"
10="{**__meta_kubernetes_pod_ip** 100.96.17.41}"
11="{**__meta_kubernetes_pod_label_freshtracks_io_cluster** bowl.freshtracks.io}"
12="{**__meta_kubernetes_pod_label_pod_template_hash** 1636686694}"
13="{**__meta_kubernetes_pod_label_run** data-sidecar}"
14="{**__meta_kubernetes_pod_name** data-sidecar-1636686694-83crm}"
15="{**__meta_kubernetes_pod_node_name** ip-xx-xxx-xx-xxx.us-west-2.compute.internal}"
16="{**__meta_kubernetes_pod_ready** false}"
17="{**__metrics_path__** /metrics}"
18="{**__scheme__** http}"
19="{**job** ftio-data-sidecar-calc}"

<relabel_config>

{**__address__** 300.196.17.41:8077}
{**__scheme__** http}
{**__metrics_path__** /metrics}
{**job** ftio-data-sidecar-calc}
{**kubernetes_namespace** default}
{**container_name** prometheus-configmap-reload}

http_requests_total{**region**="us-east",
az="us-east-1", **instance_type**="m2.xlarge",
instance="i-3582k8", **hostname**="host1"} = 5439

http_requests_total{**region**="us-east",
az="us-east-1",
instance_type="m2.xlarge",
instance="i-3582k8",
hostname="host1",
instance="300.196.17.41:8077",
job="ftio-data-sidecar-calc",
kubernetes_namespace="default",
container_name="prometheus-configmap-reload",
} = 5439

<metric_relabel_config>

Recording Rules

Create a new series, derived from one or more existing series

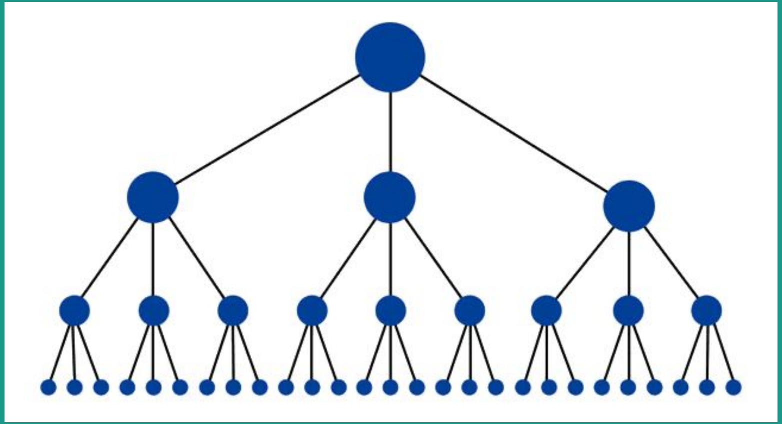
```
# The name of the time series to output to. Must be a valid metric name.
record: <string>

# The PromQL expression to evaluate. Every evaluation cycle this is
# evaluated at the current time, and the result recorded as a new set of
# time series with the metric name as given by 'record'.
expr: <string>

# Labels to add or overwrite before storing the result.
labels:
  [ <labelname>: <labelvalue> ]
```

Recording Rules

Create a new series, derived from one or more existing series

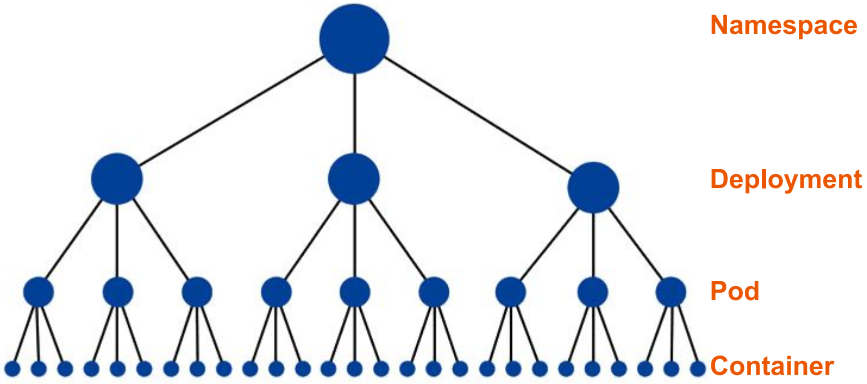
```
record: pod_name:cpu_usage_seconds:rate5m
expr: sum(rate(container_cpu_usage_seconds_total{pod_name=~"^(?:.+)$"}[5m]))
  BY (pod_name)
labels:
  ft_target: "true"
```

Kubernetes
Hierarchy and Aggregation

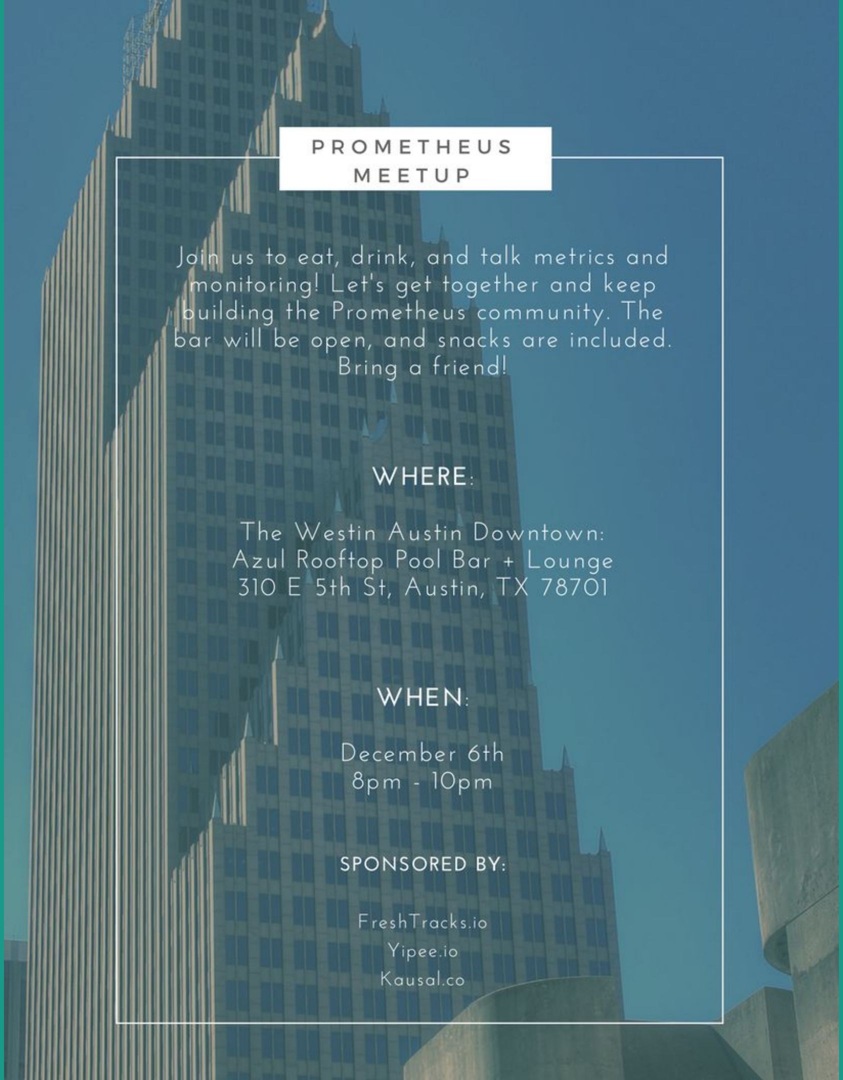


Core Metrics Aggregation

- K8s clusters form a hierarchy
- We can aggregate the “core” metrics to any level
- This allows for some interesting monitoring opportunities
  - Using Prometheus “recording rules” aggregate the core metrics at every level
  - Insights into all levels of your Kubernetes cluster
- This also applies to any custom application metric



Demo



PROMETHEUS MEETUP

Join us to eat, drink, and talk metrics and monitoring! Let's get together and keep building the Prometheus community. The bar will be open, and snacks are included. Bring a friend!

WHERE:

The Westin Austin Downtown:
Azul Rooftop Pool Bar + Lounge
310 E 5th St, Austin, TX 78701

WHEN:

December 6th
8pm - 10pm

SPONSORED BY:

FreshTracks.io
Yipee.io
Kausal.co

FreshTracks.io is Hiring!

FRESHTRACKS.io

https://searchjobs.ca.com/go/Accelerator/3279000/

Questions?

Resources

- Prometheus.io
- Core Metrics in Kubelet
- Kubernetes monitoring architecture
- What is the new metrics-server?