Kubernetes' Metrics API

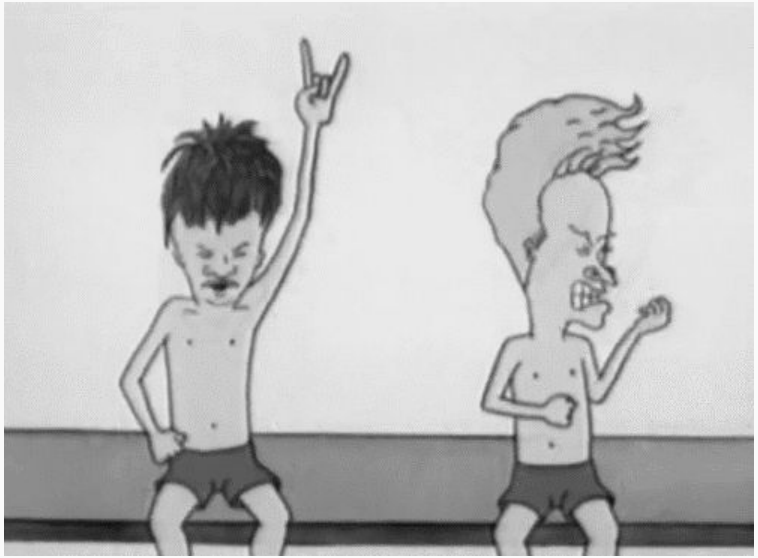
Past, Present and Future of the Prometheus Adapter

whoami - Sergiusz Urbaniak

Software Engineer at ~~CoreOS~~/RedHat

- Worked on
  - Mesos (Kubernetes scheduler integration)
  - rkt/rktnetes
  - minikube Linux
  - Kubernetes CRI, OCI
  - Tectonic Installer
- Now working on all things Prometheus in Kubernetes

whoami - Matthias Loibl



whoami - Matthias Loibl

Software Engineer at RedHat

Prev at *Loodse* & *JustWatch*

All about: *Go*, *Prometheus*, *Kubernetes*, *Drone* and much more

Creator of *gopass*

Organizer of the Berlin Prometheus MeetUp

Agenda

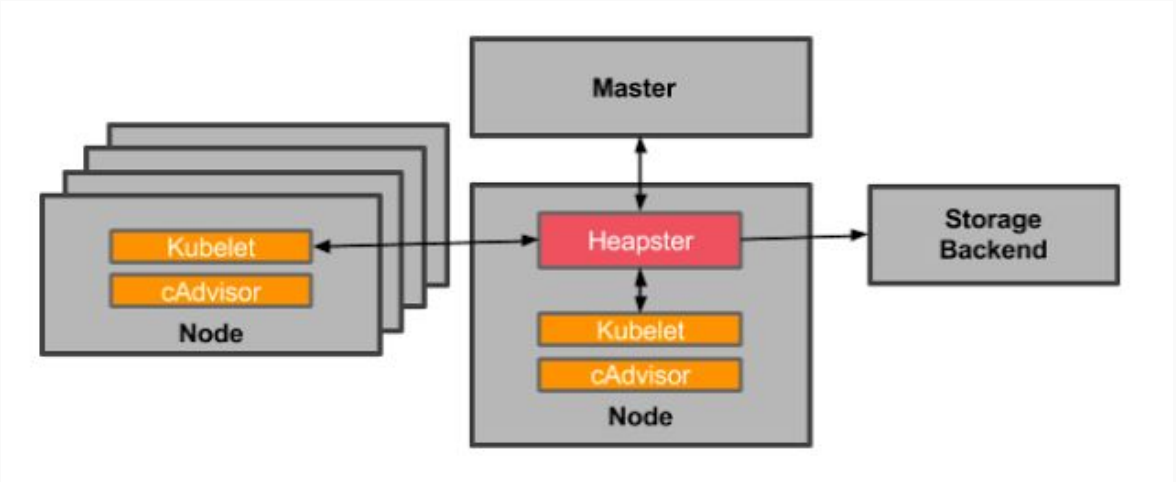
- History
- Current state
- Resource metrics
- Custom metrics
- External metrics
- HPA example
- Prometheus Adapter Future
- How to get involved

History: Metrics in Kubernetes ~v1.1

Scheduling Decision

Metrics	Node	Pod
CPU	Based on currently measured node capacity	Scheduled Resource Requests
Memory		

Architecture - ~ Kubernetes v1.1



https://kubernetes.io/blog/2015/05/resource-usage-monitoring-kubernetes/

Problems with Heapster

- Push based model
- Sink plugins results in a vendor dump
- Opinionated tooling
- No abstraction
- What if I want to use Prometheus instead?

https://brancz.com/2018/01/05/prometheus-vs-heapster-vs-kubernetes-metrics-apis/

Wait .... What about Prometheus ???

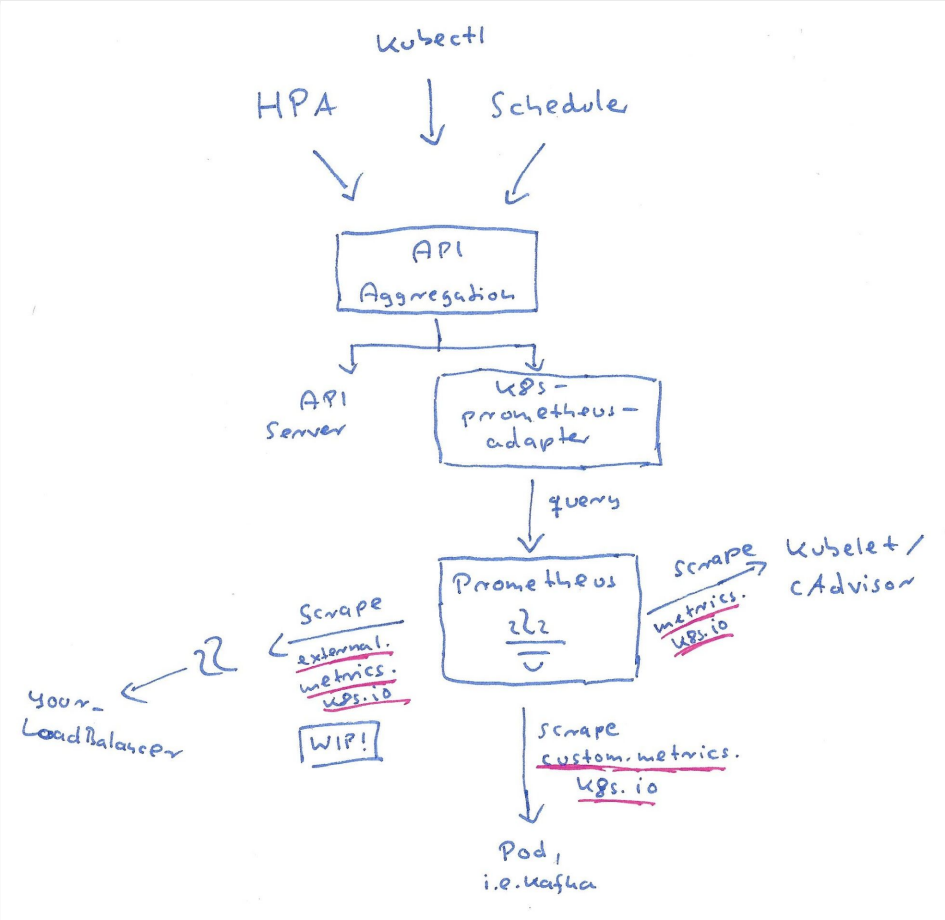
Goals

- Decouple scheduling and scaling decisions from Heapster
- Introduce an abstract API scheme
- Let different vendors implement that scheme
  - Prometheus
  - DataDog
  - Cloud Providers: Azure, AWS, GCP, etc

Meet the Metrics APIs

- Resource Metrics API
  - CPU & memory *per* node & pod
  - Opinionated
  - Concrete, pre-defined metrics
- Custom Metrics API
  - Relate to in-cluster objects, pods, etc.
  - Meta-model, abstract metrics
- External Metrics API
  - Relate to external non-cluster objects, i.e. Load Balancers, Infra components
  - Meta-model, abstract metrics

Architecture



Implementations

API	metrics-server	k8s-prometheus-adapter	azure-metrics-adapter	custom-metrics-stackdriver-adapter	k8s-cloudwatch-adapter	zalando/kube-metrics-adapter	keda
metrics.k8s.io	X	X					
custom.metrics.k8s.io		X	X	X	X	X	X
external.metrics.k8s.io		X	X	X	X		X

Resource Metrics

Discovering Metrics APIs

```
$ kubectl get apiservices v1beta1.metrics.k8s.io

NAME                     SERVICE                           AVAILABLE   AGE
v1beta1.metrics.k8s.io   monitoring/prometheus-adapter     True        4h
```

Discovering Metrics APIs

```
$ kubectl api-resources | grep metrics.k8s.io

NAME   SHORTNAMES   APIGROUP           NAMESPACED   KIND
nodes               metrics.k8s.io     false        NodeMetrics
pods                metrics.k8s.io     true         PodMetrics
```

Resource Metrics API

/apis/metrics.k8s.io

```
$ kubectl get pods.metrics.k8s.io grafana-5c5d49b4c4-m25pc -o yaml

kind: PodMetrics
apiVersion: metrics.k8s.io/v1beta1
metadata:
  creationTimestamp: 2018-11-13T12:44:37Z
  name: grafana-5c5d49b4c4-m25pc
  namespace: monitoring
containers:
- name: grafana
  usage:
    cpu: 10m
    memory: 44444Ki
- name: ""
  usage:
    cpu: 14m
    memory: 45780Ki
timestamp: 2018-11-13T12:44:37Z
window: 1m0s
```

Example

```
$ kubectl top pod

NAME                                  CPU(cores)   MEMORY(bytes)
alertmanager-main-0                   15m          27Mi
alertmanager-main-1                   9m           26Mi
alertmanager-main-2                   11m          26Mi
grafana-5c5d49b4c4-m25pc              29m          76Mi
kube-state-metrics-668bd94f74-xpfjf   9m           107Mi
node-exporter-rfmjb                   12m          55Mi
prometheus-adapter-6fbffbd98c-k6gc4   4m           26Mi
prometheus-k8s-0                      41m          307Mi
prometheus-k8s-1                      64m          315Mi
prometheus-operator-88fcf6d95-4phlm   34m          54Mi
```

Conversion Kubernetes <-> Prometheus

```
$ kubectl get configmap adapter-config -o jsonpath='{.data.config\.yaml}'

    resourceRules:
      cpu:
        containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>}[1m])) by
(<<.GroupBy>>)
```

Conversion Kubernetes <-> Prometheus

```
$ kubectl get configmap adapter-config -o jsonpath='{.data.config\.yaml}'

    resourceRules:
      cpu:
        containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>}[1m])) by
(<<.GroupBy>>)
        nodeQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>, id='/'}[1m]))
by (<<.GroupBy>>)
```

Conversion Kubernetes <-> Prometheus

```
$ kubectl get configmap adapter-config -o jsonpath='{.data.config\.yaml}'

    resourceRules:
      cpu:
        containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>}[1m])) by
(<<.GroupBy>>)
        nodeQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>, id='/'}[1m]))
by (<<.GroupBy>>)
        resources:
          overrides:
            node:
              resource: node
            namespace:
              resource: namespace
            pod_name:
              resource: pod
```

Conversion Kubernetes <-> Prometheus

```
$ kubectl get configmap adapter-config -o jsonpath='{.data.config\.yaml}'

    resourceRules:
      cpu:
        containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>}[1m])) by
(<<.GroupBy>>)
        nodeQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>, id='/'}[1m]))
by (<<.GroupBy>>)
        resources:
          overrides:
            node:
              resource: node
            namespace:
              resource: namespace
            pod_name:
              resource: pod
        containerLabel: container_name
```

Association

```
$ curl -s 'http://localhost:9090/api/v1/series?match[]=container_cpu_usage_seconds_total' \
    | jq .

{
  "status": "success",
  "data": [
    {
      "__name__": "container_cpu_usage_seconds_total",
      "container_name": "POD",                              ===> Associates with container
      "cpu": "total",
      "endpoint": "https-metrics",
      "id": "/kubepods/besteffort/pod446c9ff5-2781-...",
      "image": "k8s.gcr.io/pause:3.1",
      "instance": "192.168.122.2:10250",
      "job": "kubelet",
      "name": "k8s_POD_kube-proxy-kh87t...",
      "namespace": "kube-system",                           ===> Associates with namespace[.core]
      "node": "minikube",                                   ===> Associates with node[.core]
      "pod_name": "kube-proxy-kh87t",                       ===> Associates with pod[.core]
      "service": "kubelet"
    },
```

Conversion Kubernetes <-> Prometheus

```
$ kubectl get configmap adapter-config -o jsonpath='{.data.config\.yaml}'

    resourceRules:
      memory:
        containerQuery: sum(container_memory_working_set_bytes{<<.LabelMatchers>>}) by
(<<.GroupBy>>)
        nodeQuery: sum(container_memory_working_set_bytes{<<.LabelMatchers>>,id='/'}) by
(<<.GroupBy>>)
...
```

Conversion Kubernetes <-> Prometheus

```
$ kubectl get configmap adapter-config -o jsonpath='{.data.config\.yaml}'

    resourceRules:
      cpu:
        containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>}[1m])) by
(<<.GroupBy>>)
        nodeQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>, id='/'}[1m]))
by (<<.GroupBy>>)
...
      memory:
...
      window: 1m
```

Custom Metrics

Custom Metrics API

/apis/custom-metrics/v1beta1

Scale based on different Kubernetes types:

- Pods
- Services
- Jobs
- ...

Custom Metrics API Service

```
$ kubectl get apiservices v1beta1.custom.metrics.k8s.io

NAME                              SERVICE                                      AVAILABLE   AGE
v1beta1.custom.metrics.k8s.io     custom-metrics/custom-metrics-apiserver      True        3h
```

Discovering registered custom metrics

```
$ kubectl get --raw '/apis/custom.metrics.k8s.io/v1beta1' | jq .

{
  "kind": "APIResourceList",
  "apiVersion": "v1",
  "groupVersion": "custom.metrics.k8s.io/v1beta1",
  "resources": [
...
    {
      "name": "pods/http_requests_per_second",
      "singularName": "",
      "namespaced": true,
      "kind": "MetricValueList",
      "verbs": [
        "get"
      ]
    }
  ]
}
```

Conversion Kubernetes <-> Prometheus

- Discovery of metrics in Prometheus
- Association with Kubernetes resources
- Naming of metrics
- Querying of metric values in Prometheus

Discovery

```
$ kubectl -n custom-metrics get configmaps adapter-config -o yaml

apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    rules:

      # 1. Discovery: match all http_requests_total metrics, having a (k8s) namespace set
    - seriesQuery: 'http_requests_total{namespace!=""}'
```

Discovery

```
$ curl -s 'http://localhost:9090/api/v1/series?match[]=http_requests_total\{namespace!=""\}' \
    | jq .

{
  "status": "success",
  "data": [
    {
      "__name__": "http_requests_total",
      "endpoint": "http",
      "instance": "172.17.0.14:8080",
      "job": "podinfo",
      "namespace": "default",
      "pod": "podinfo-67c9fd95d-fqk4g",
      "service": "podinfo",
      "status": "200"
    },
...
```

Association

```
$ kubectl -n custom-metrics get configmaps  adapter-config -o yaml

apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    rules:

      # 1. Discovery: match all http_requests_total metrics, having a (k8s) namespace set
    - seriesQuery: 'http_requests_total{namespace!=""}'

      # 2. Association: Match k8s resources to discovered metrics
      resources:
        template: <<.Resource>>
```

Association

```
$ curl -s 'http://localhost:9090/api/v1/series?match[]=http_requests_total\{namespace!=""\}' \
    | jq .

{
  "status": "success",
  "data": [
    {
      "__name__": "http_requests_total",
      "endpoint": "http",
      "instance": "172.17.0.14:8080",
      "job": "podinfo",                          ===> Associates with jobs.batch
      "namespace": "default",                    ===> Associates with namespace[.core]
      "pod": "podinfo-67c9fd95d-fqk4g",          ===> Associates with pod[.core]
      "service": "podinfo",                      ===> Associates with service[.core]
      "status": "200"
    },
...
```

Naming

```
$ kubectl -n custom-metrics get configmaps  adapter-config -o yaml

apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    rules:

      # 1. Discovery: match all http_requests_total metrics, having a (k8s) namespace set
    - seriesQuery: 'http_requests_total{namespace!=""}'

      # 2. Association: Match k8s resources to discovered metrics
      resources:
        template: <<.Resource>>

      # 3. Naming: Convert `http_requests_total` to `http_requests_total_per_second`
      name:
        matches: "^(.*)_total"
        as: "${1}_per_second"
```

Naming

```
$ curl -s 'http://localhost:9090/api/v1/series?match[]=http_requests_total\{namespace!=""\}' \
    | jq .

{
  "status": "success",
  "data": [
    {
      "__name__": "http_requests_total",        ===> Renames to http_requests_total_seconds
      "endpoint": "http",
      "instance": "172.17.0.14:8080",
      "job": "podinfo",
      "namespace": "default",
      "pod": "podinfo-67c9fd95d-fqk4g",
      "service": "podinfo",
      "status": "200"
    },
...
```

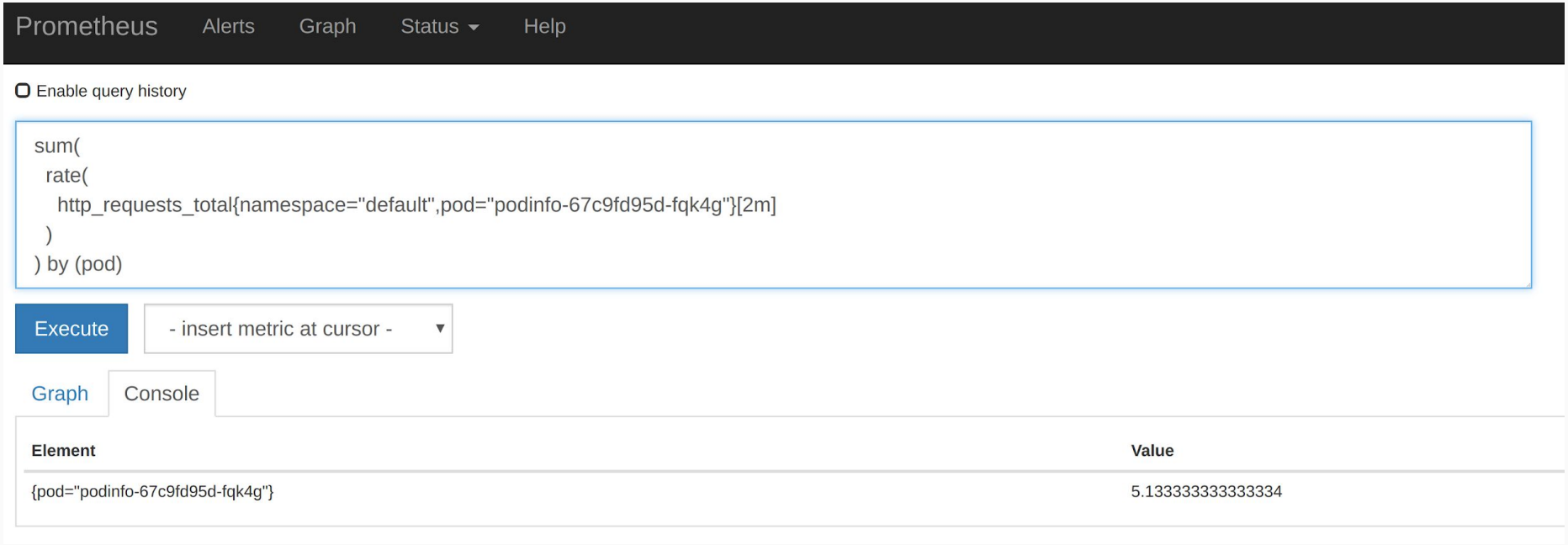
Querying

```
$ kubectl -n custom-metrics get configmaps  adapter-config -o yaml

apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    rules:

      # 1. Discovery: match all http_requests_total metrics, having a (k8s) namespace set
    - seriesQuery: 'http_requests_total{namespace!=""}'

      # 2. Association: Match k8s resources to discovered metrics
      resources:
        template: <<.Resource>>

      # 3. Naming: Convert `http_requests_total` to `http_requests_total_per_second`
      name:
        matches: "^(.*)_total"
        as: "${1}_per_second"

      # 4. Querying: Execute a per-second rate query
      metricsQuery: 'sum(rate(<<.Series>>{<<.LabelMatchers>>}[2m])) by (<<.GroupBy>>)'
```

Querying

```
$ kubectl get --raw \
    '/apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pod/podinfo-67c9fd95d-fqk4g/http_requests_per_second' \
    | jq .

{
  "kind": "MetricValueList",
  "apiVersion": "custom.metrics.k8s.io/v1beta1",
  "metadata": {
    "selfLink":
"/apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pod/podinfo-67c9fd95d-fqk4g/http_requests_per_second"
  },
  "items": [
    {
      "describedObject": {
        "kind": "Pod",
        "namespace": "default",
        "name": "podinfo-67c9fd95d-fqk4g",
        "apiVersion": "/v1"
      },
      "metricName": "http_requests_per_second",
      "timestamp": "2019-02-02T11:52:35Z",
       "value": "5133m"
    }
  ]
}
```

Configurable queries



External Metrics API Service

```
$ kubectl get apiservices v1beta1.external.metrics.k8s.io

NAME                                SERVICE                                          AVAILABLE   AGE
v1beta1.external.metrics.k8s.io   custom-metrics/custom-metrics-apiserver   True        3h
```

Querying

```
$ kubectl -n custom-metrics get configmaps  adapter-config -o yaml

apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    externalRules:

      # 1. Discovery: match all http_requests_total metrics, having a (k8s) namespace set
    - seriesQuery: 'http_requests_total{namespace!=""}'

      # 2. Association: Match k8s resources to discovered metrics
      resources:
        template: <<.Resource>>

      # 3. Naming (optionally)

      # 4. Querying: Execute a per-second rate query
      metricsQuery: 'http_requests_total{<<.LabelMatchers>>}'
```

Querying

```
$ kubectl get --raw '/apis/external.metrics.k8s.io/v1beta1/namespaces/kube-system/http_requests_total' | jq .
{
  "kind": "ExternalMetricValueList",
  "apiVersion": "external.metrics.k8s.io/v1beta1",
  "metadata": {
    "selfLink": "/apis/external.metrics.k8s.io/v1beta1/namespaces/kube-system/http_requests_total"
  },
  "items": [
    {
      "metricName": "http_requests_total",
      "metricLabels": {
        "__name__": "http_requests_total",
        "code": "200",
        "endpoint": "https-metrics",
        "handler": "prometheus",
        "instance": "10.9.0.2:10250",
        "job": "kubelet",
        "method": "get",
        "namespace": "kube-system",
        "node": "kind-control-plane",
        "service": "kubelet"
      },
      "timestamp": "2019-05-17T13:17:34Z",
      "value": "944"
    }
  ]
}
```

Prometheus Adapter - Future

- Merge external metrics support
- Move to kubernetes organization
- Tackle complex configuration
- Tackle scalability issues due to discovery overhead
- Flatten library dependencies
- Planned refactoring potentially based on CRDs

Sample HPA configuration

```
kind: HorizontalPodAutoscaler
apiVersion: autoscaling/v2beta1
metadata:
  name: podinfo
  namespace: default

spec:
  minReplicas: 1
  maxReplicas: 10
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: podinfo

  metrics:

  # use a "Pods" metric, which takes the average of the
  # given metric across all pods controlled by the autoscaling target
  - type: Pods
    pods:

      metricName: http_requests_per_second

      # target 1000 milli-requests per second = 1 req/second,
      targetAverageValue: 1000m
```

User - QuickStart

- kube-prometheus ships with Prometheus Adapter and resource metrics enabled.
- Upstream Prometheus adapter includes custom metrics sample deployment.

In both cases:

```
$ kubectl apply -f manifests/
```

How to get involved?

Developer - Kubernetes SIGs

Community activity is organized into Special Interest Groups (SIGs)

The topics in this talk are related to

SIG-instrumentation mailing list and bi-weekly meetings

SIG-autoscaling mailing list and bi-weekly meetings

https://github.com/kubernetes/community/blob/master/sig-list.md

Thank you!
Questions?