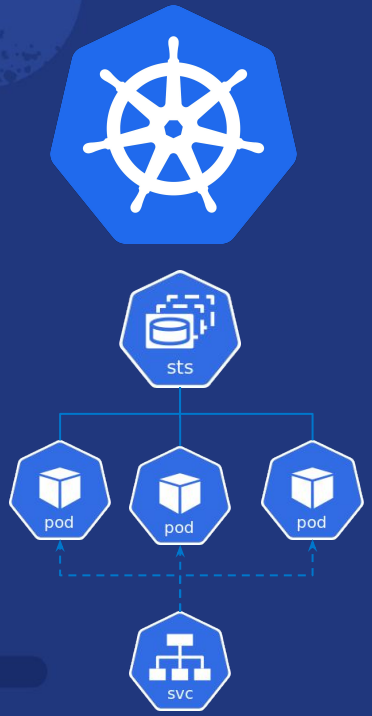
Autoscaling on K8S using Elasticsearch (along with other metrics servers)

Thursday, Mar 31, 2022

Michael Morello



ECK Operator

Prometheus

Agenda

1. The ECK operator
2. Scaling
3. (Horizontal Pod) Autoscaling

Operator ?

elastic

Deploying your applications and managing their lifecycle, following best practices.

- Create the required Kubernetes resources (Pods, Services, Secrets...)
- Manage applications lifecycle
  - Wait for data to be migrated during a scale down
- Plumbing
  - Setup secure connections between applications (Kibana, Beats, Enterprise Search)
- Upgrading your stack
  - Automatically apply best practices
  - Minimize downtime during a rolling upgrade

elastic

Elastic Cloud on Kubernetes

https://github.com/elastic/cloud-on-k8s

Deploy and Manage

Elasticsearch, Kibana, and APM Server, Beats, Agent and Enterprise Search

Native Kubernetes experience

Fully integrated with Kubernetes API, use kubectl to operate and control

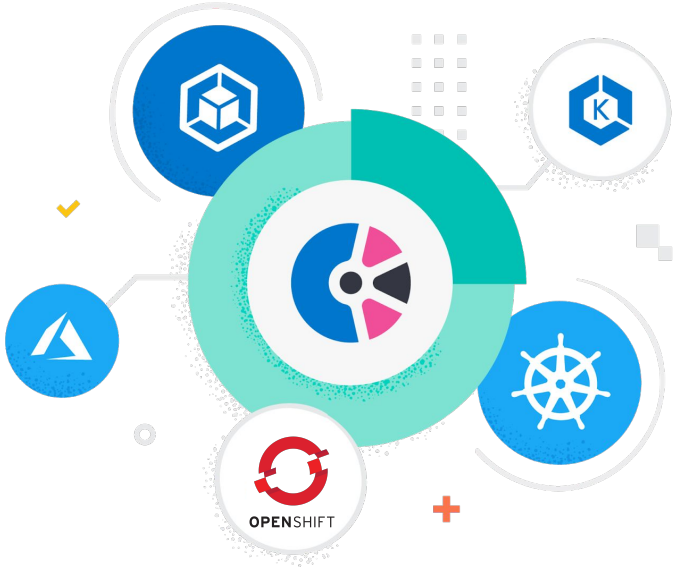
Supports multiple K8s distros

Azure Kubernetes Service (**AKS**)

Amazon Elastic Kubernetes Service (**EKS**)

Google Kubernetes Engine (**GKE**)

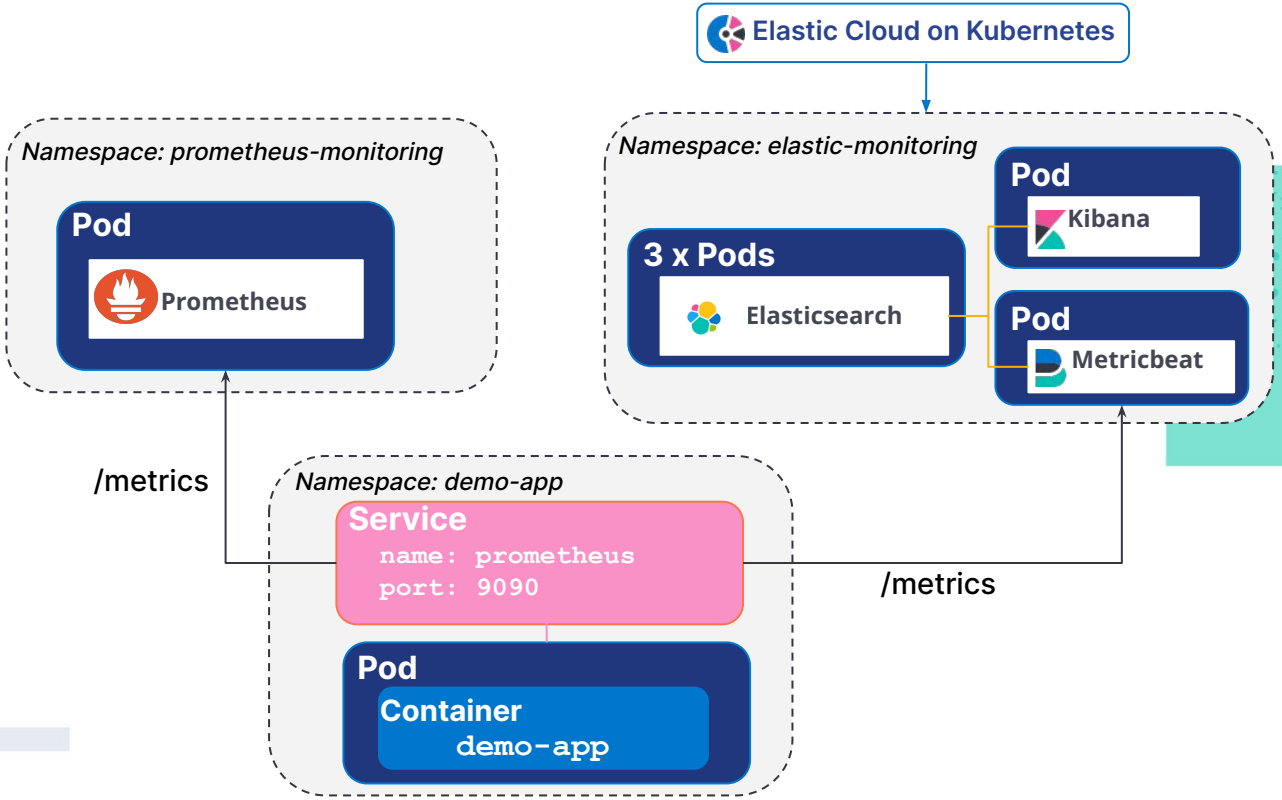
Vanilla Kubernetes, and Red Hat **OpenShift**



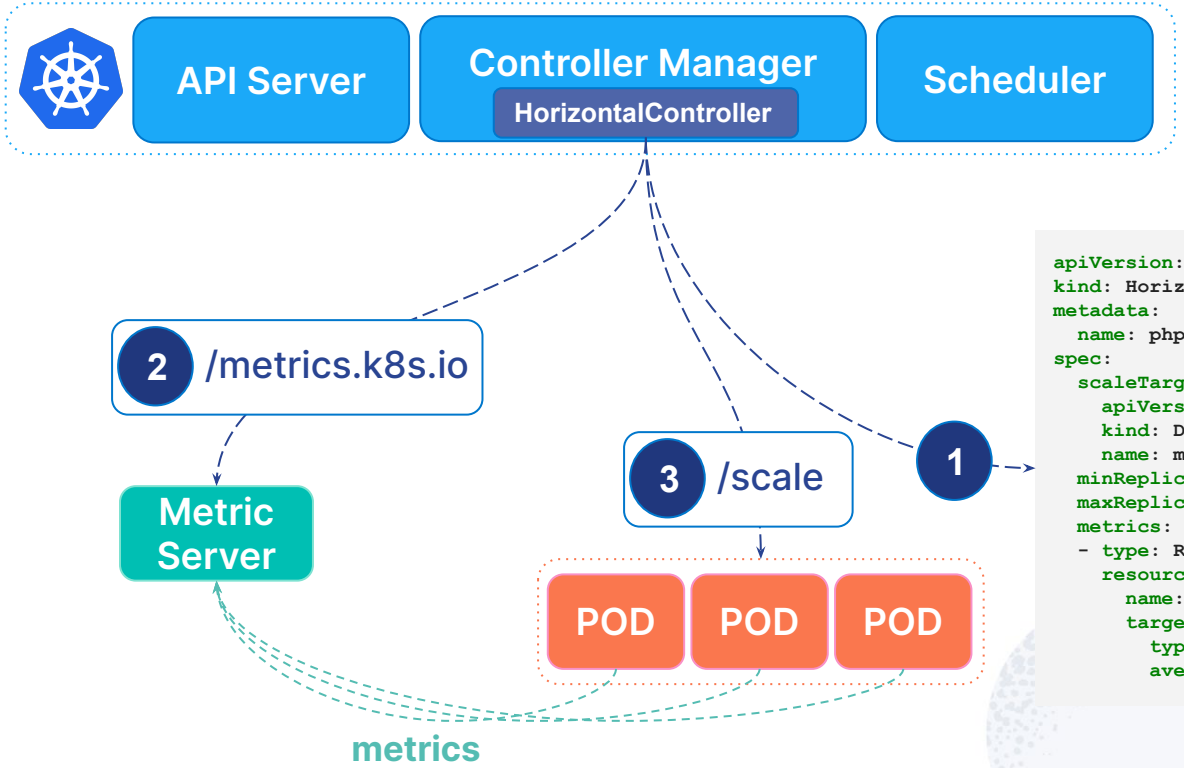
ECK demo

elastic

Demo environment



K8S Horizontal Autoscaling 101




```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

elastic

/scale ?

elastic

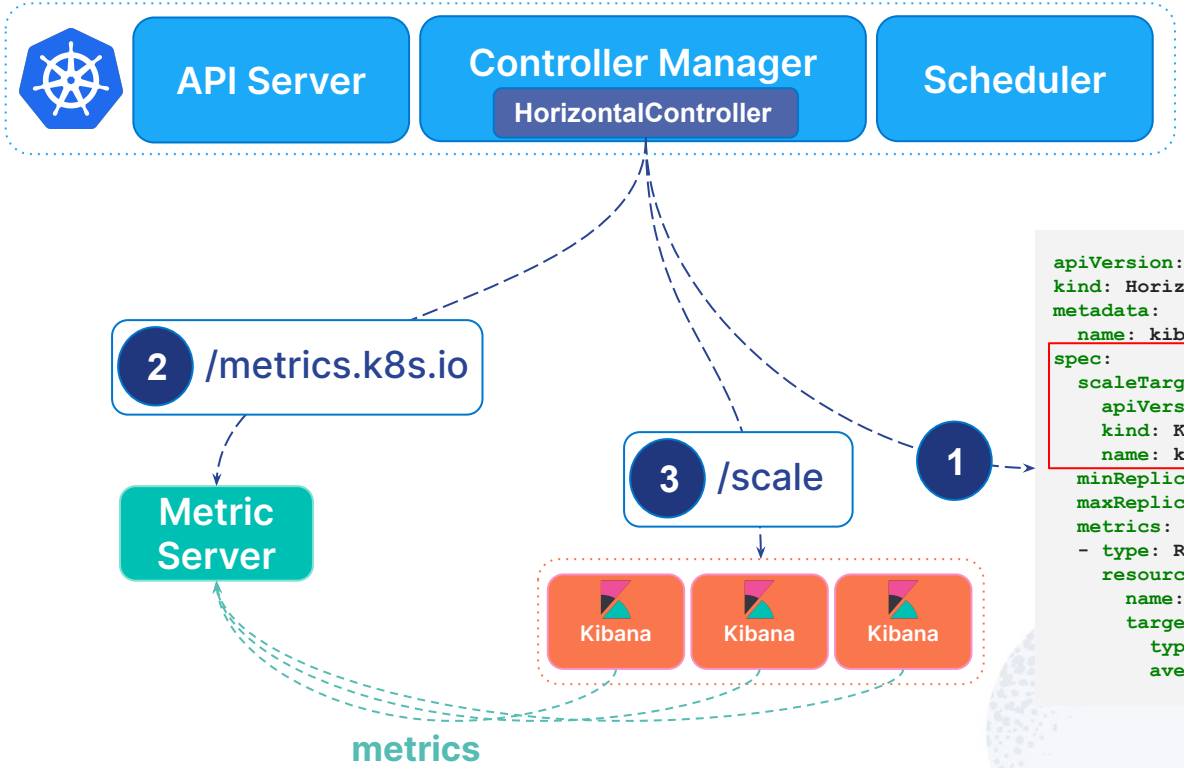
/scale

```
kubectl -v=8 scale --replicas 3 deployment.apps/<name>
```

```
request.go:1181] Request Body: {"spec":{"replicas":3}}

round_trippers.go:432] PATCH /apis/apps/v1/namespaces/demo/deployments/<name>/scale
```

elastic




```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: kibana-autoscaler
spec:
  scaleTargetRef:
    apiVersion: kibana.k8s.elastic.co/v1
    kind: Kibana
    name: kibana-example
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

elastic

What about Elasticsearch ?

elastic Products Customers Learn Company Pricing

Docs

Elastic Cloud on Kubernetes [2.1] » Orchestrating Elastic Stack applications » Run Elasticsearch on ECK » **Elasticsearch autoscaling**

« Pod PreStop hook JVM heap dumps »

Elasticsearch autoscaling

edit

NOTE Elasticsearch autoscaling requires a valid Enterprise license or Enterprise trial license. Check the license documentation for more details about managing licenses.

WARNING This functionality is in technical preview and may be changed or removed in a future release. Elastic will apply best effort to fix any issues, but features in technical preview are not subject to the support SLA of official GA features.

ECK can leverage the autoscaling API introduced in Elasticsearch 7.11 to adjust automatically the number of Pods and the allocated resources in a tier. Currently, autoscaling is supported for Elasticsearch data tiers and machine learning nodes.

Enable autoscaling

edit

To enable autoscaling on an Elasticsearch cluster, you need to define one or more autoscaling policies. Each autoscaling policy applies to one or more NodeSets which share the same set of roles specified in the `node.roles` setting in the Elasticsearch configuration.

Define autoscaling policies

edit

Autoscaling policies can be defined in the `elasticsearch.alpha.elastic.co/autoscaling-spec` annotation of the Elasticsearch resource. Each autoscaling policy must have the following fields:

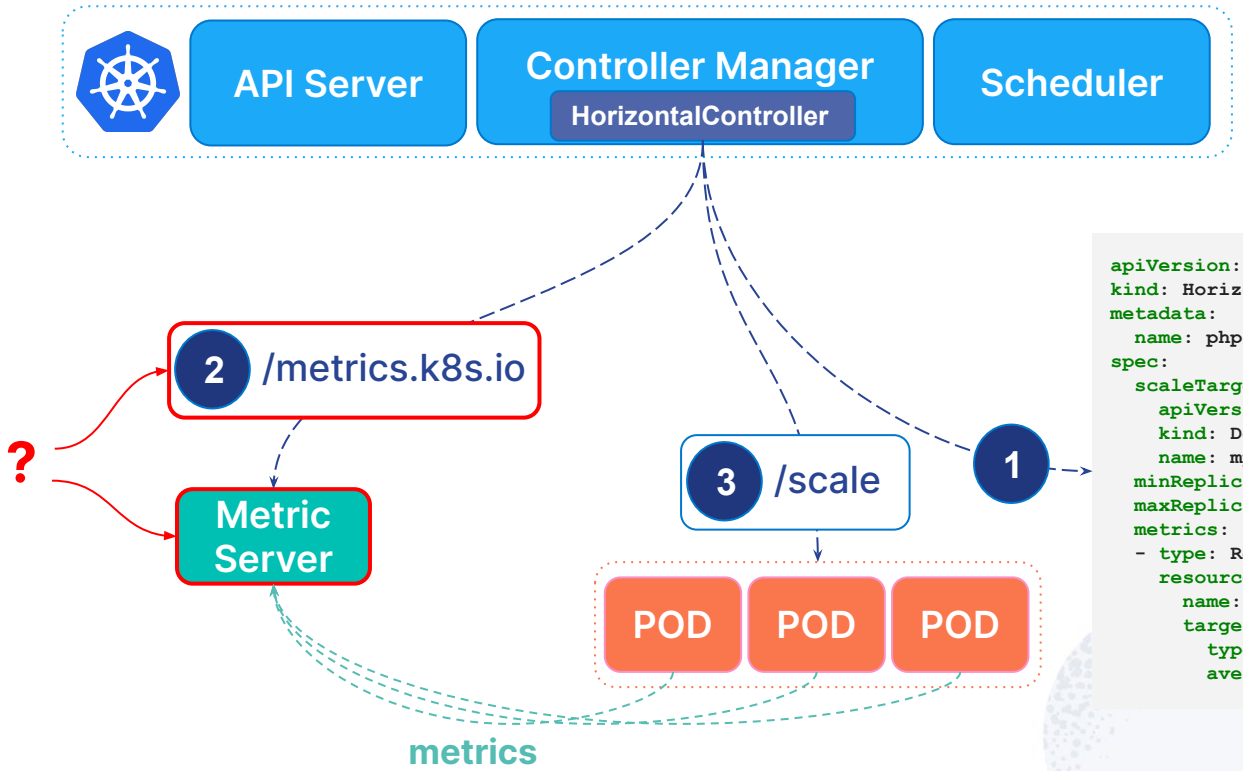
- `name` is a unique name used to identify the autoscaling policy.
- `roles` contains a set of node roles, unique across all the autoscaling policies, used to identify the NodeSets to which this policy applies. At least one NodeSet with the exact same set of roles must exist in the Elasticsearch resource specification.
- `resources` helps define the minimum and maximum compute resources usage:
  - `nodeCount` defines the minimum and maximum nodes allowed in the tier.
  - `cpu` and `memory` enforce minimum and maximum compute resources usage for the Elasticsearch container.
  - `storage` enforces minimum and maximum storage request per PersistentVolumeClaim.

https://www.elastic.co/guide/en/cloud-on-k8s/current/k8s-autoscaling.html

elastic

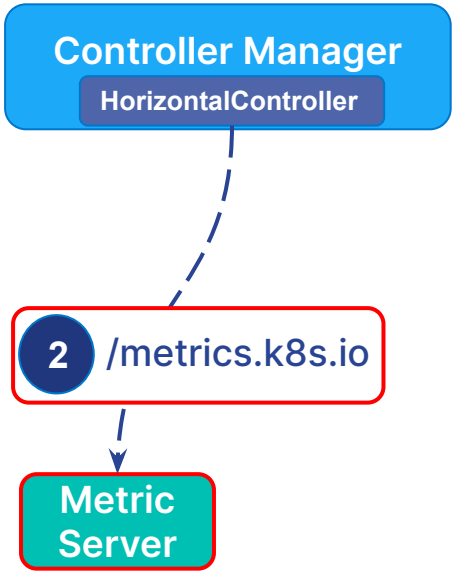
Metrics

elastic




```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

elastic

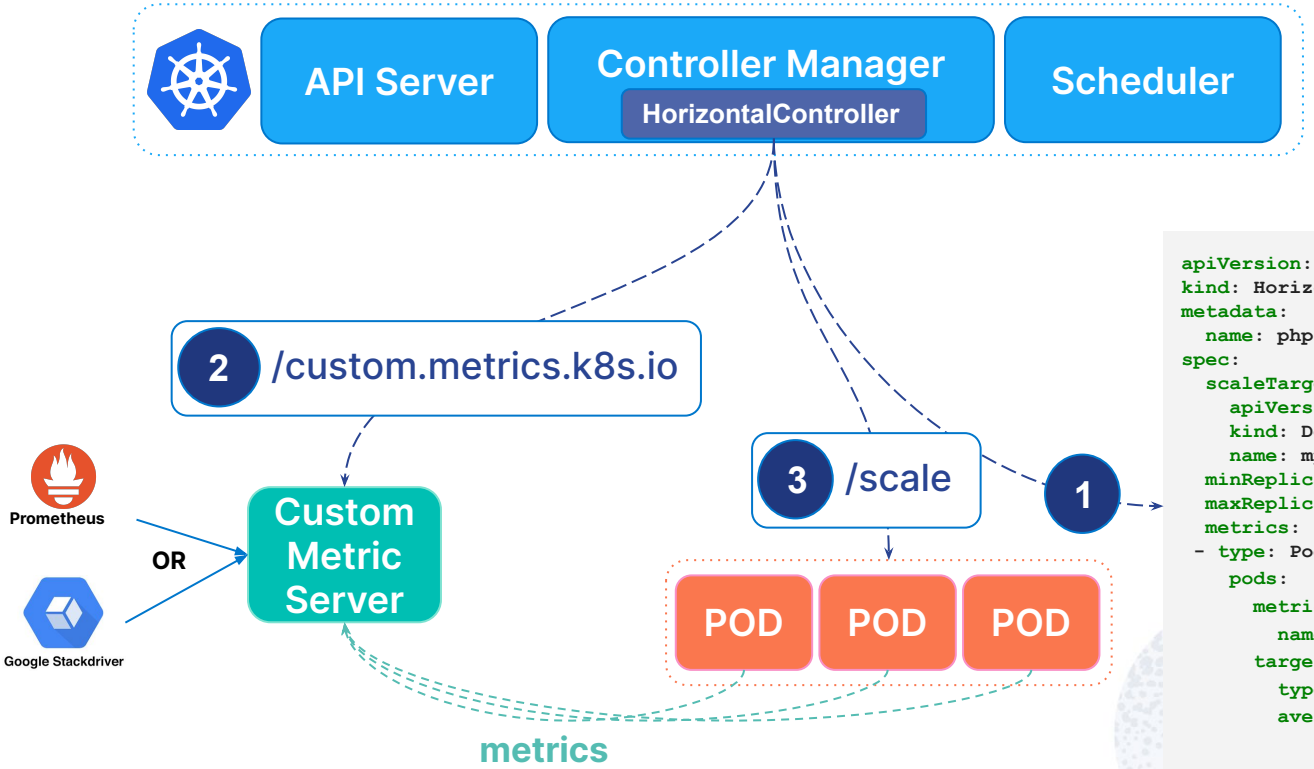



```
k get --raw /apis/metrics.k8s.io/v1beta1/namespaces/<ns>/pods/<pod>
```


```
{
 "kind": "PodMetrics",
 "apiVersion": "metrics.k8s.io/v1beta1",
 "metadata": {
   "name": "pod",
   "namespace": "ns",
 },
 "window": "30s",
 "containers": [
   {
     "name": "container",
     "usage": {
       "cpu": "14514455n",
       "memory": "533396Ki"
     }
   }
 ]
}
```

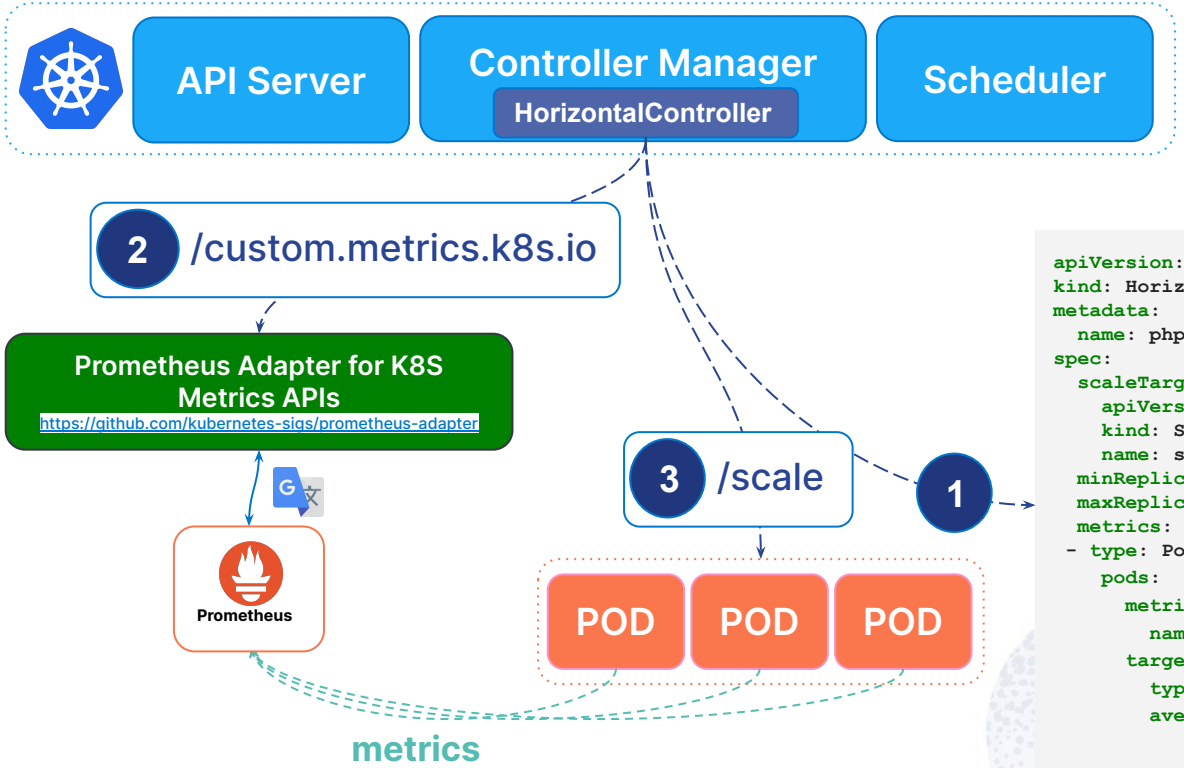
(Custom) Metrics

elastic



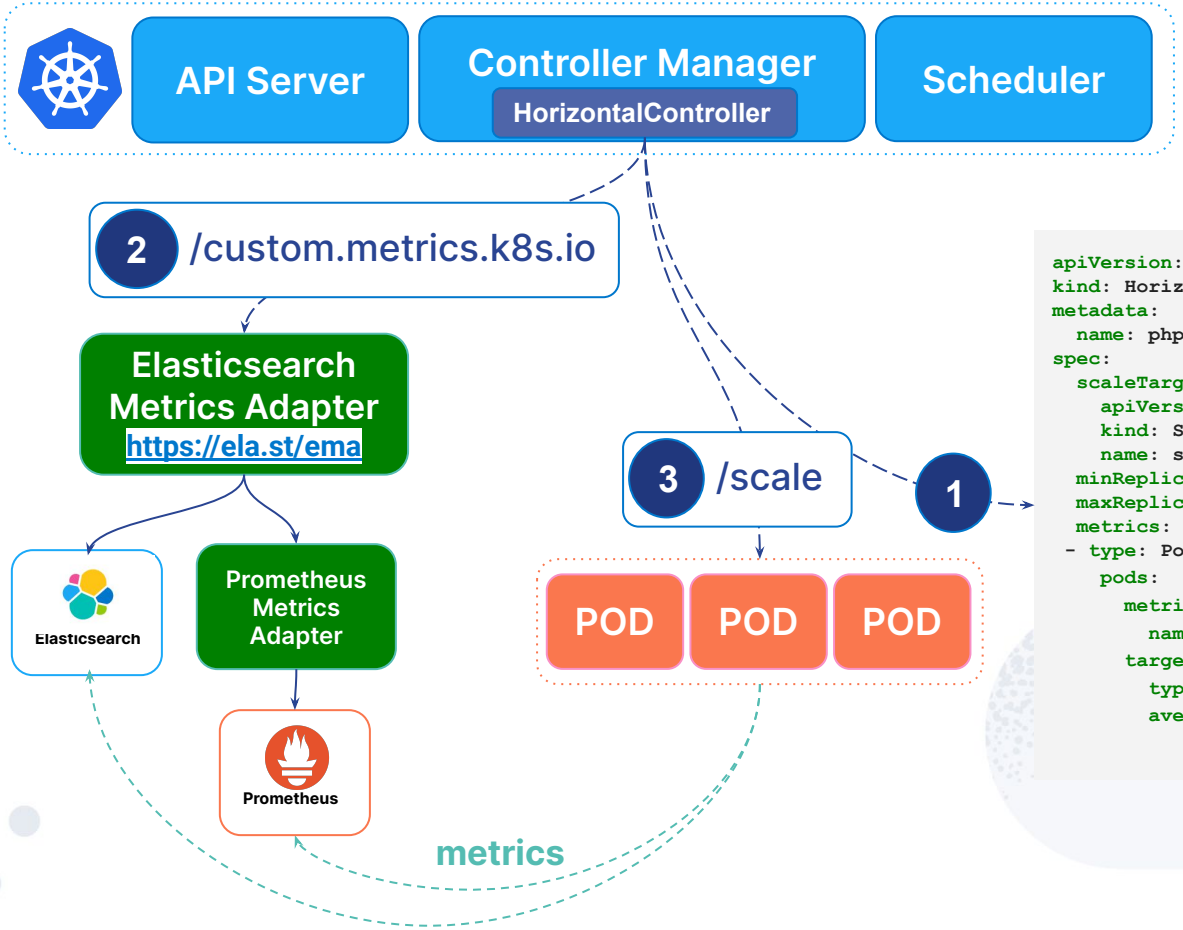

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
 - type: Pods
    pods:
      metric:
        name: packets-per-second
      target:
        type: AverageValue
        averageValue: 1k
```

elastic



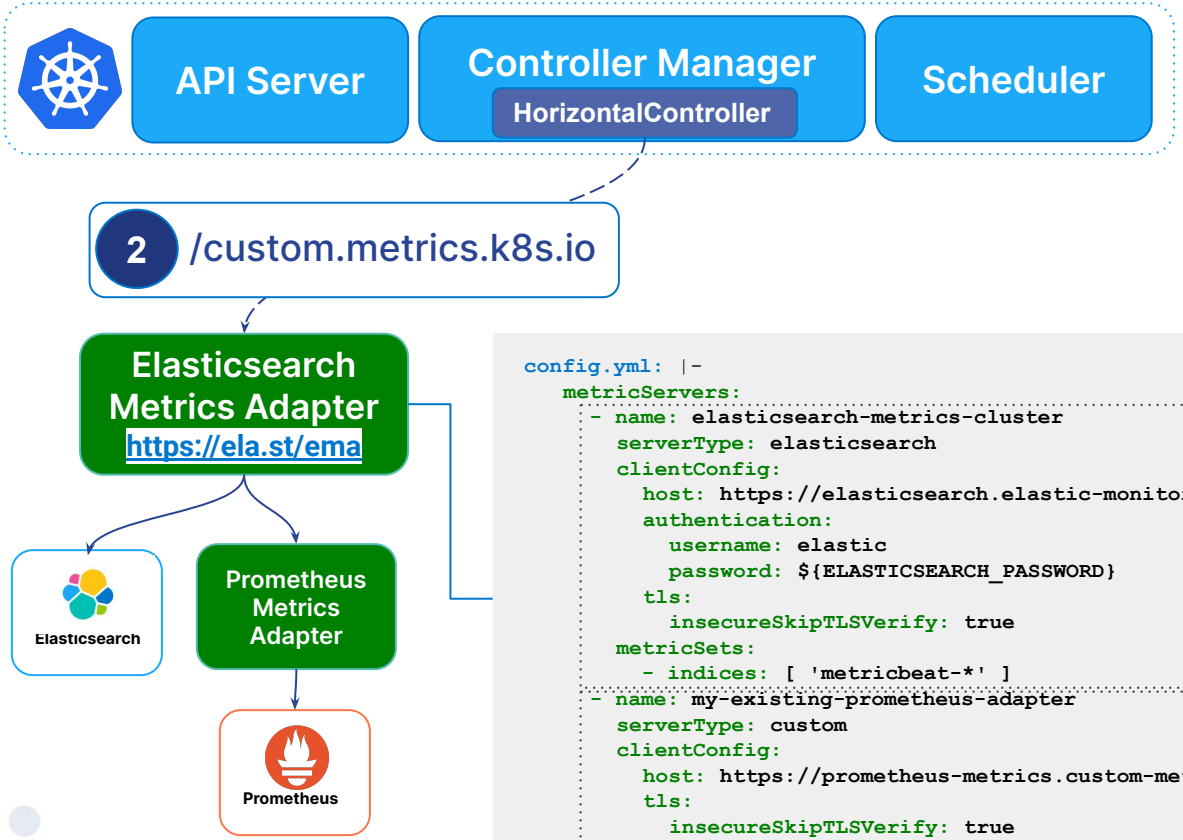

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: StatefulSet
    name: simple-go-service
  minReplicas: 1
  maxReplicas: 10
  metrics:
 - type: Pods
    pods:
      metric:
        name: follow_me
      target:
        type: AverageValue
        averageValue: "1"
```

elastic



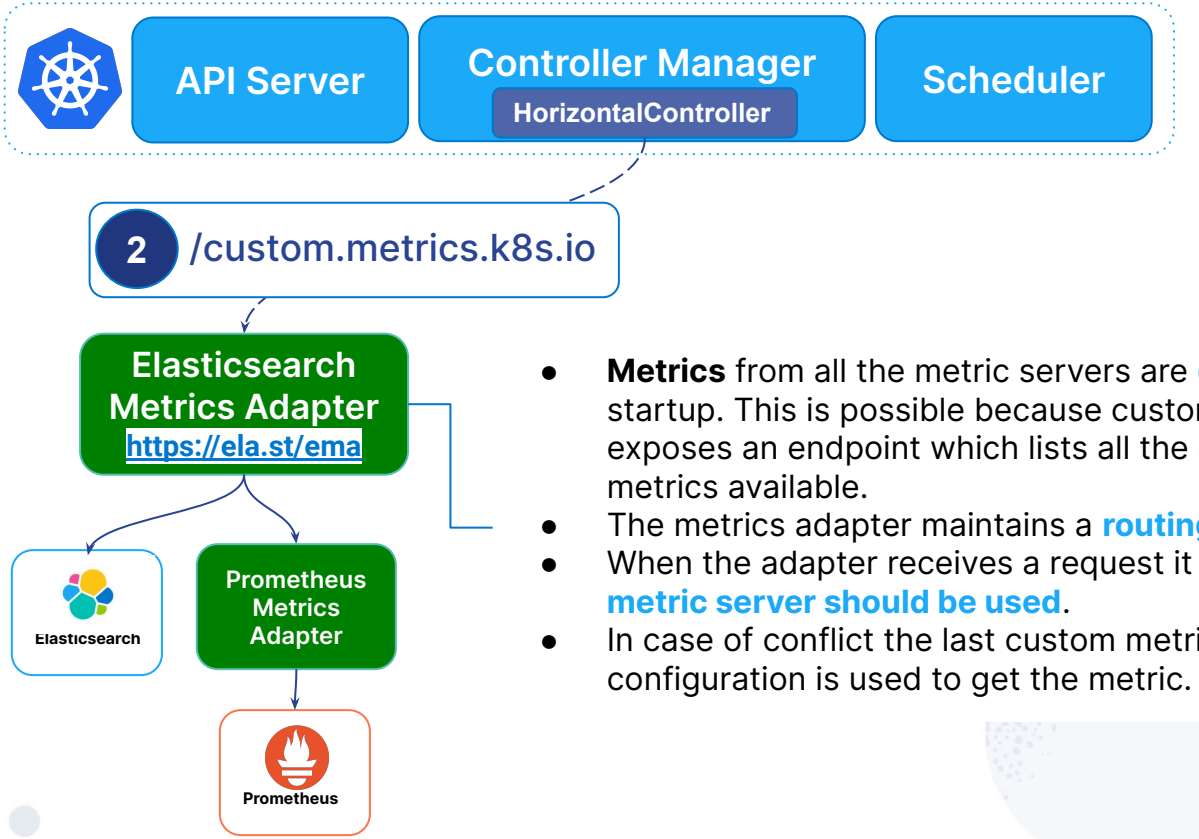

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: StatefulSet
    name: simple-go-service
  minReplicas: 1
  maxReplicas: 10
  metrics:
 - type: Pods
    pods:
      metric:
        name: follow_me
      target:
        type: AverageValue
        averageValue: "1"
```

elastic




```yaml
config.yml: |-
  metricServers:
    - name: elasticsearch-metrics-cluster
      serverType: elasticsearch
      clientConfig:
        host: https://elasticsearch.elastic-monitoring.svc:9200
        authentication:
          username: elastic
          password: ${ELASTICSEARCH_PASSWORD}
        tls:
          insecureSkipTLSVerify: true
      metricSets:
        - indices: [ 'metricbeat-*' ]
    - name: my-existing-prometheus-adapter
      serverType: custom
      clientConfig:
        host: https://prometheus-metrics.custom-metrics.svc
        tls:
          insecureSkipTLSVerify: true
```

elastic



- **Metrics** from all the metric servers are **discovered** during startup. This is possible because custom metric servers exposes an endpoint which lists all the custom and external metrics available.
- The metrics adapter maintains a **routing table** in memory.
- When the adapter receives a request it looks up **which metric server should be used**.
- In case of conflict the last custom metrics server in the configuration is used to get the metric.

Elasticsearch Metrics Adapter for the K8S Metrics APIs

https://ela.st/ema

https://github.com/elastic/elasticsearch-k8s-metrics-adapter

elastic / elasticsearch-k8s-metrics-adapter Public

Unwatch 8

Code | Issues 6 | Pull requests | Discussions | Security | Insights | Settings

main | 1 branch | 1 tag | Go to file | Add file | Code

barkbay Fix ECK version requirement in README.md (#21) 796e82d on 31 Jan 2 commits

config	Initial commit	3 months ago
deploy	Initial commit	3 months ago
generated/openapi	Initial commit	3 months ago
hack	Initial commit	3 months ago
pkg	Initial commit	3 months ago
vendor	Initial commit	3 months ago
.gitignore	Initial commit	3 months ago
Dockerfile	Initial commit	3 months ago
LICENSE.txt	Initial commit	3 months ago
Makefile	Initial commit	3 months ago
NOTICE.txt	Initial commit	3 months ago
README.md	Fix ECK version requirement in README.md (#21)	2 months ago
VERSION	Initial commit	3 months ago
go.mod	Initial commit	3 months ago
go.sum	Initial commit	3 months ago
main.go	Initial commit	3 months ago

README.md

Elasticsearch Adapter for the Kubernetes Metrics APIs

The Elasticsearch adapter for the K8S metrics APIs is an implementation of the Kubernetes resource metrics and custom metrics APIs.

It can be used to automatically scale applications, using the Horizontal Pod Autoscaler querying metrics collected by Metricbeat or Agent, and stored in Elasticsearch.

Use of the adapter

This adapter is currently considered as experimental, and is thus not yet fully supported in production by Elastic. We'd love to get your feedback for any issues or contributions, in the form of issues in this repository.

About

An implementation of the Kubernetes Custom Metrics API for Elasticsearch

Readme
Apache-2.0 License
3 stars
8 watching
1 fork

Releases

1 tags
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

barkbay Michael Morello
naemono Michael Montgomery
Kushmaro Omer Kushmaro

Languages

Go 94.8% · Makefile 2.4% · Shell 1.4% · Dockerfile 1.4%

elastic

Thank You

https://github.com/barkbay/2022-03-31-elasticsearch-k8s-metrics-adapter-demo