Autoscaling All Things Kubernetes with Prometheus

Michael Hausenblas & Frederic Branczyk, Red Hat
@mhausenblas @fredbrancz

OPENSHIFT by Red Hat

Autoscaling?

- On an abstract level:
  - Calculate resources to cover demand
  - Demand measured by metrics
  - Metrics must be collected, stored and queryable
- Ultimately to fulfill
  - Service Level Objectives (SLO) …
  - of Service Level Agreements (SLA) …
  - through Service Level Indicators (SLI)

Types of autoscaling (in Kubernetes)

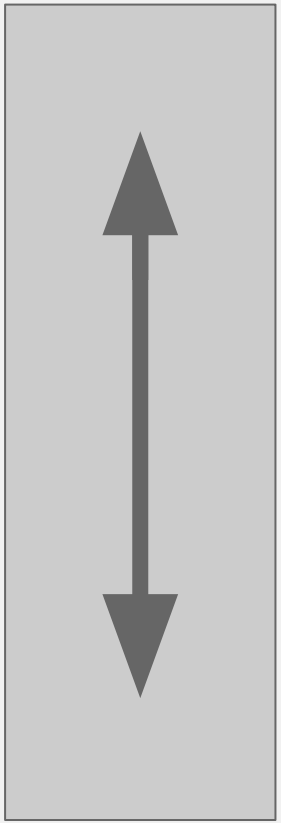
- Cluster-level
- App-level
  - Horizontal
  - Vertical

Horizontal autoscaling

- Horizontal pod autoscaler
- Resource: replicas
- “Increasing replicas when necessary”
- Requires application to be designed to scale horizontally

Vertical autoscaling

- Vertical pod autoscaler
- Resource: CPU/Memory
- “Increasing CPU/Memory when necessary”
- Less complicated to design for resource increase
- Harder to autoscale



History of autoscaling on Kubernetes

- Autoscaling used to heavily rely on Heapster
  - Heapster collects metrics and writes to time-series database
  - Metrics collection via cAdvisor (container + custom-metrics)
- We could autoscale!

… but *not* based on

Prometheus metrics :(

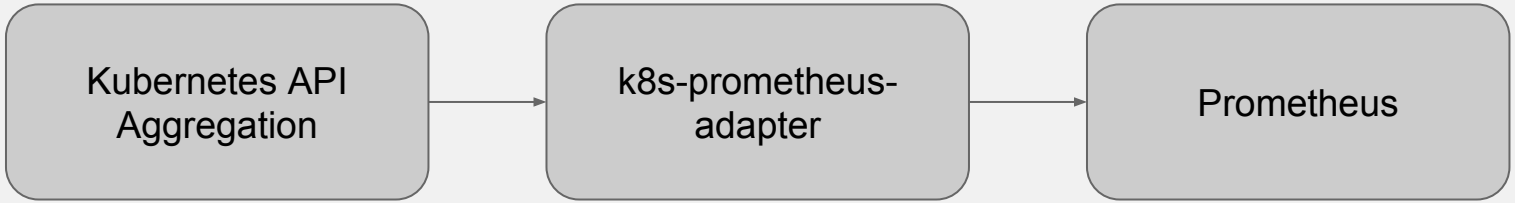
Enter:

Resource & Custom Metrics API

Resource & Custom Metrics APIs

- Well defined APIs:
  - Not an implementation, an API spec
  - Implemented and maintained by vendors
  - Returns single value

- For us, most importantly: Allowing Prometheus as a metric source



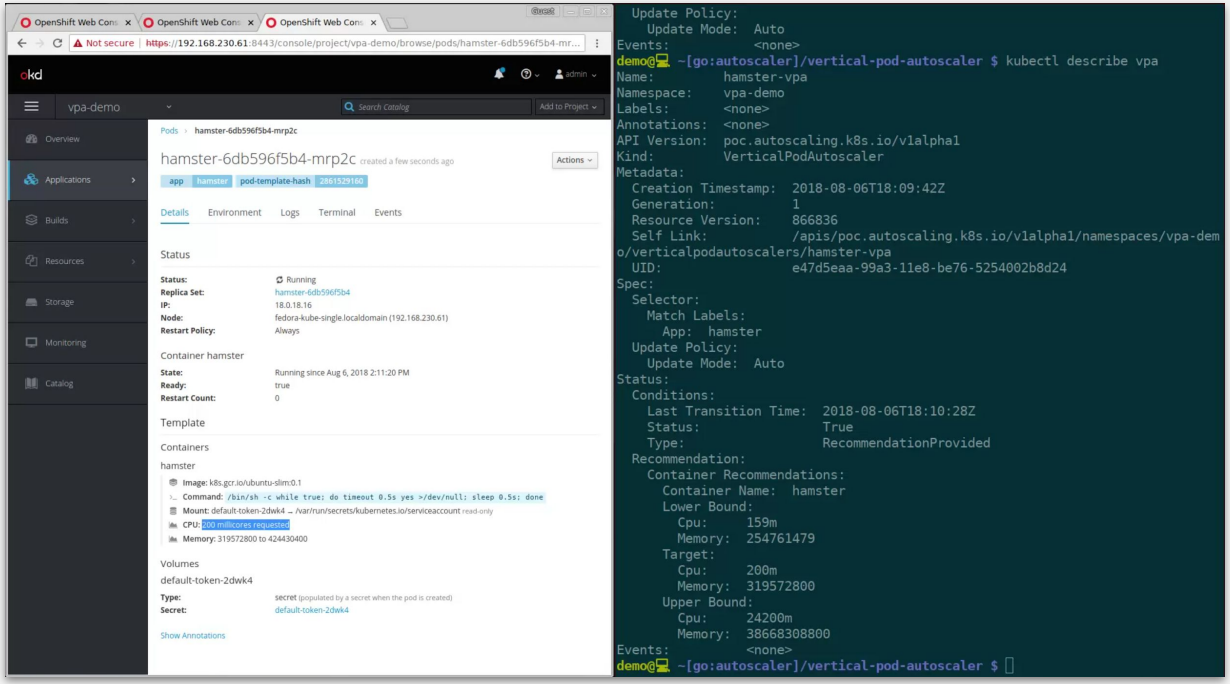
But only Horizontal Autoscaling

So what about vertical autoscaling?

Enter:

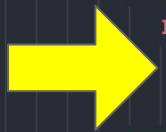
Vertical Pod Autoscaling

VPA demo


OpenShift Web Cons
OpenShift Web Cons
OpenShift Web Cons
Not secure | https://192.168.230.61:8443/console/project/vpa-demo/browse/pods/hamster-6db596f5b4-mr...
okd
admin
vpa-demo
Search Catalog
Add to Project
Overview
Applications
Builds
Resources
Storage
Monitoring
Catalog
Pods > hamster-6db596f5b4-mrp2c
hamster-6db596f5b4-mrp2c created a few seconds ago
Actions
app hamster pod-template-hash 2861529160
Details Environment Logs Terminal Events
Status
Status: Running
Replica Set: hamster-6db596f5b4
IP: 18.0.18.16
Node: fedora-kube-single.localdomain (192.168.230.61)
Restart Policy: Always
Container hamster
State: Running since Aug 6, 2018 2:11:20 PM
Ready: true
Restart Count: 0
Template
Containers
hamster
Image: k8s.gcr.io/ubuntu-slim:0.1
Command: /bin/sh -c while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
Mount: default-token-2dwk4 … /var/run/secrets/kubernetes.io/serviceaccount read-only
CPU: 200 millicores requested
Memory: 319572800 to 424430400
Volumes
default-token-2dwk4
Type: secret (populated by a secret when the pod is created)
Secret: default-token-2dwk4
Show Annotations
Update Policy:
Update Mode: Auto
Events: <none>
demo@ ~[go:autoscaler]/vertical-pod-autoscaler $ kubectl describe vpa
Name: hamster-vpa
Namespace: vpa-demo
Labels: <none>
Annotations: <none>
API Version: poc.autoscaling.k8s.io/v1alpha1
Kind: VerticalPodAutoscaler
Metadata:
Creation Timestamp: 2018-08-06T18:09:42Z
Generation: 1
Resource Version: 866836
Self Link: /apis/poc.autoscaling.k8s.io/v1alpha1/namespaces/vpa-demo/verticalpodautoscalers/hamster-vpa
UID: e47d5eaa-99a3-11e8-be76-5254002b8d24
Spec:
Selector:
Match Labels:
App: hamster
Update Policy:
Update Mode: Auto
Status:
Conditions:
Last Transition Time: 2018-08-06T18:10:28Z
Status: True
Type: RecommendationProvided
Recommendation:
Container Recommendations:
Container Name: hamster
Lower Bound:
Cpu: 159m
Memory: 254761479
Target:
Cpu: 200m
Memory: 319572800
Upper Bound:
Cpu: 24200m
Memory: 38668308800
Events: <none>
demo@ ~[go:autoscaler]/vertical-pod-autoscaler $

Background & terminology

```
15  apiVersion: extensions/v1beta1
16  kind: Deployment
17  metadata:
18    name: hamster
19    namespace: default
20  spec:
21    replicas: 2
22    template:
23      metadata:
24        labels:
25          app: hamster
26      spec:
27        containers:
28        - name: hamster
29          image: k8s.gcr.io/ubuntu-slim:0.1
30          resources:
31            requests:
32              cpu: 100m
33              memory: 50Mi
34          command: ["/bin/sh"]
35          args: ["-c", "while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done"]
```



Background & terminology

- Scheduling
  - nodes offer resources
  - pods consume resources
  - scheduler matches needs of pods based on **requests**
- Types of resources (compressible/incompressible)
- Quality-of-Service (QoS)
  - Guaranteed: limit == request
  - Burstable: limit > request > 0
  - Best-Effort: $\nexists$ (limit, request)

Motivation

Unfortunately, Kubernetes has not yet implemented dynamic resource management, which is why we have to set resource limits for our containers. I imagine that at some point Kubernetes will start implementing a less manual way to manage resources, but this is all we have for now.

Ben Visser, 12/2016
Kubernetes—Understanding Resources

Kubernetes doesn't have dynamic resource allocation, which means that requests and limits have to be determined and set by the user. When these numbers are not known precisely for a service, a good approach is to start it with overestimated resources requests and no limit, then let it run under normal production load for a certain time.

Antoine Cotten, 05/2016
1 year, lessons learned from a 0 to Kubernetes transition

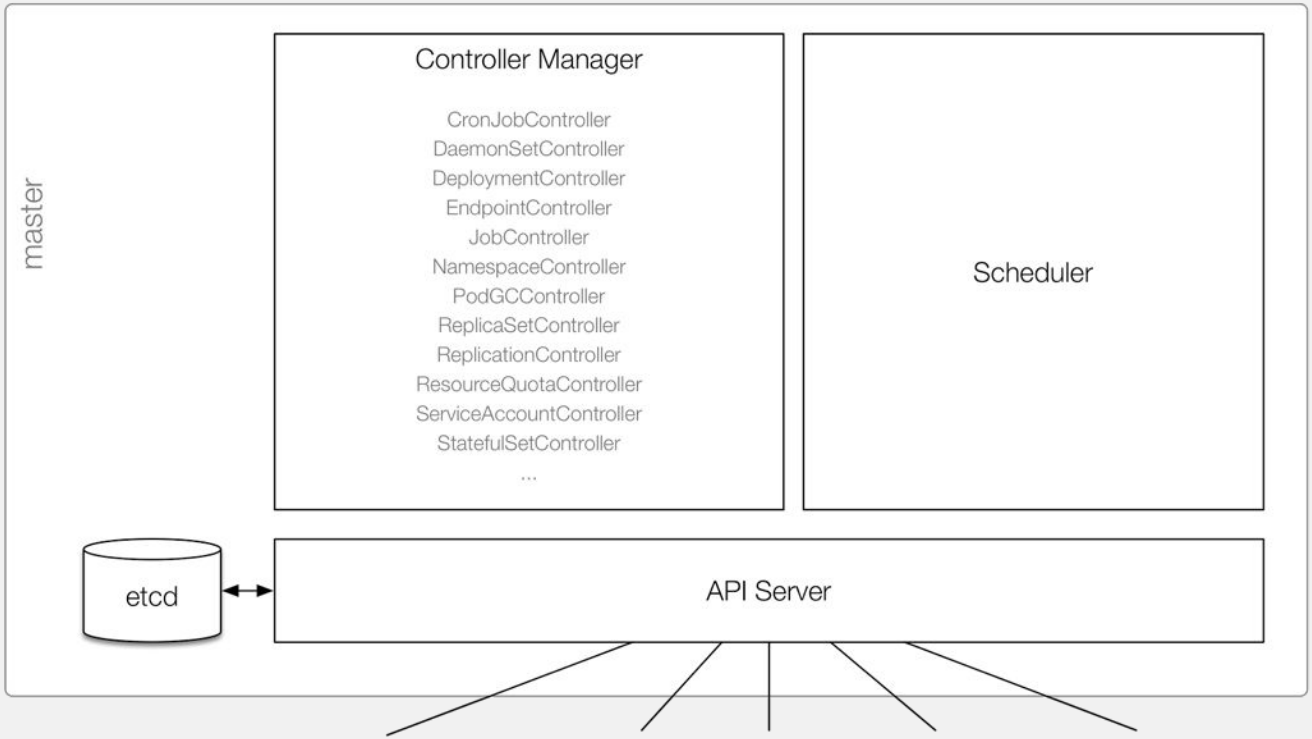
Goals

- Automating configuration of resource requirements
  - manually setting requests is brittle & hard so people don't do it
  - no requests set ➔ QoS is *best effort* :(
- Improving utilization
  - can better bin pack
  - impact on other functionality such as out of resource handling or an (aspirational) optimizing scheduler

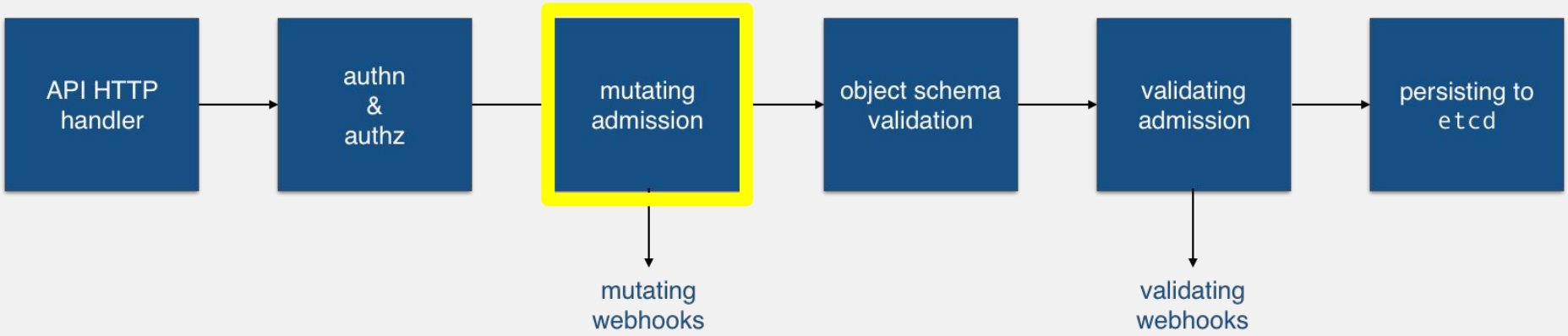
Use Cases

- For stateful apps, for example Wordpress or single-node databases
- Can help on-boarding of "legacy" apps, that is, non-horizontally scalable ones

Interlude: API server



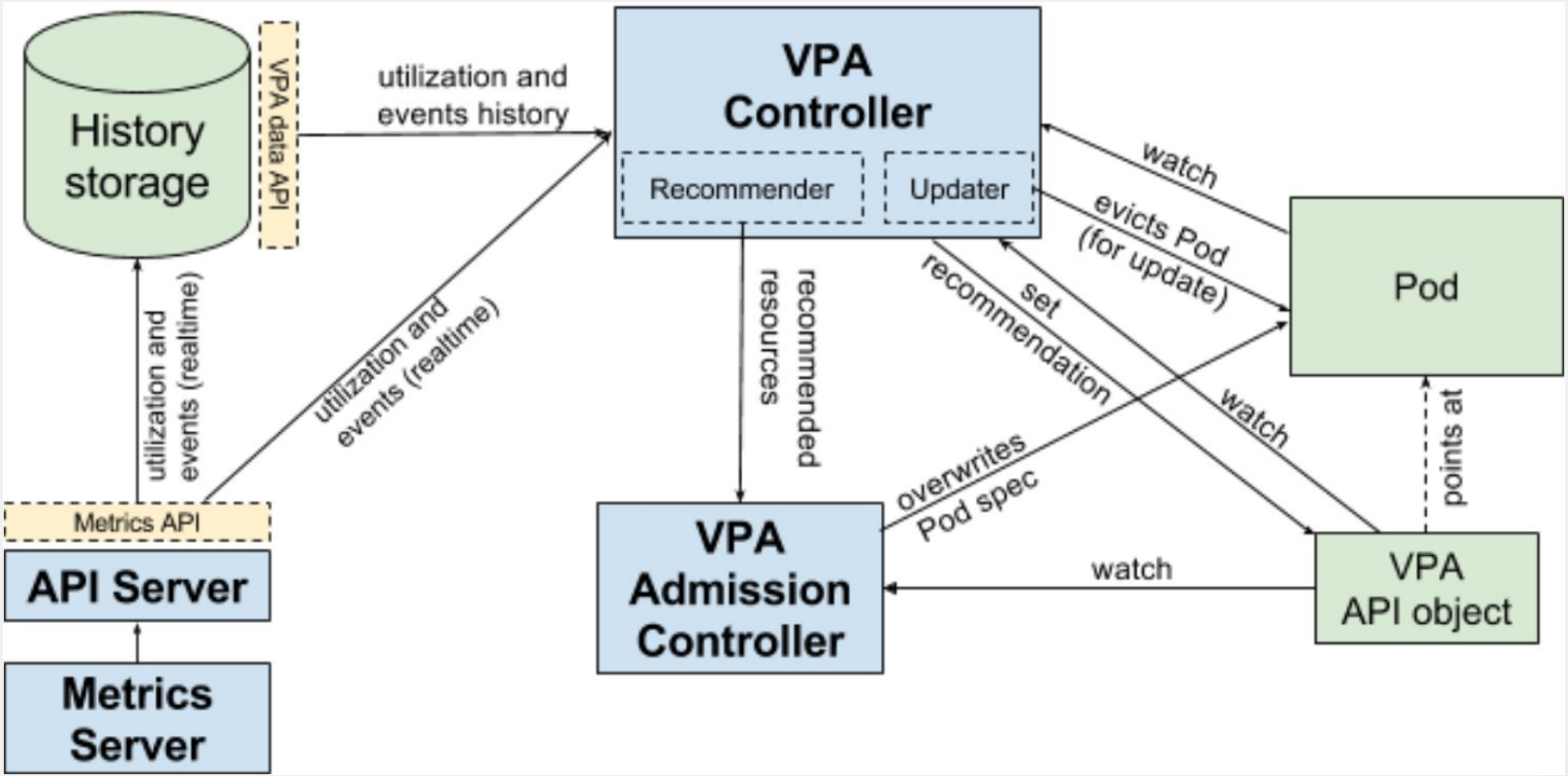
Interlude: API server



Basic idea

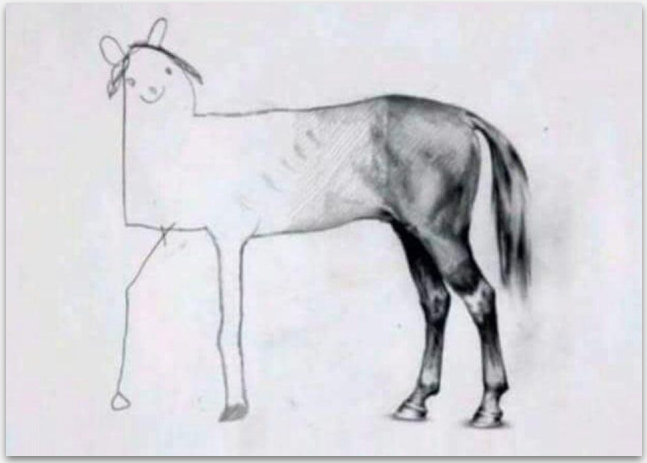
- observe resource consumption of all pods
- build up historic profile (*recommender*)
- apply to pods on an opt-in basis via labels (*updater*)

VPA architecture



Limitations

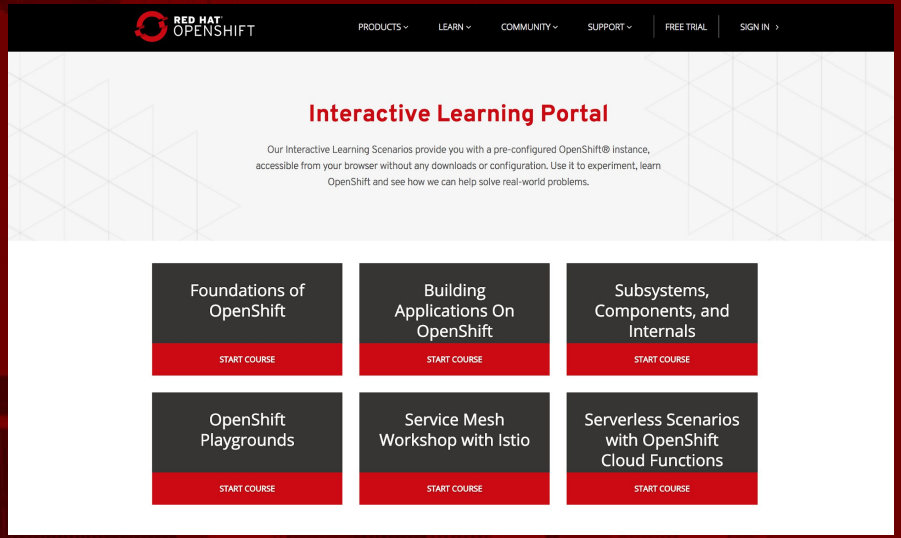
- pre-alpha, so need testing and tease out edge-cases
- in-place updates (requires support from container runtime)
- usage spikes—how to deal with it best?



Resources & what's next?

- VPA issue 10782
- VPA design
- Test, provide feedback
- SIG Autoscaling—come and join us on #sig-autoscaling or weekly online meetings on Monday
- SIG Instrumentation and SIG Autoscaling work towards a historical metrics API—get involved there!

learn.openshift.com



plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos