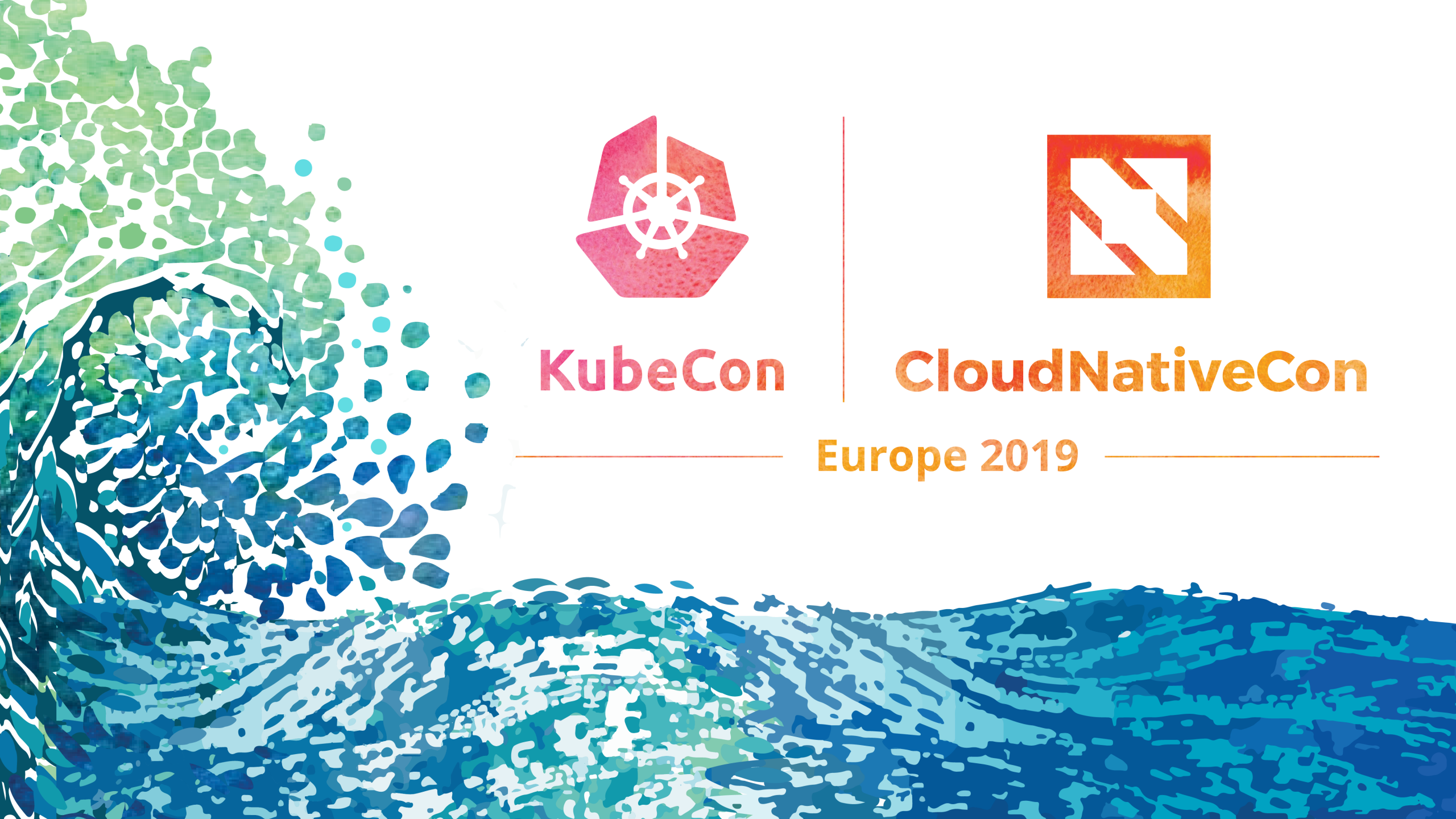

KubeCon
CloudNativeCon
Europe 2019

KubeCon | CloudNativeCon

Europe 2019

Restart-Free Vertical Scaling for Kubernetes Pods

Vinay Kulkarni

Peng Du

Huawei Cloud R&D

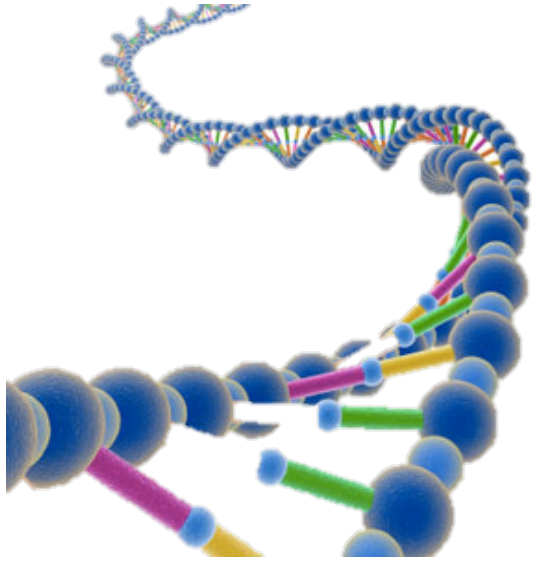
Agenda

- Our Customer Scenario
- Kubernetes Scaling Overview
- K8s Pod Scheduling Overview
- Vertical Scaling Design (Our solution)
- Policy Controls & Failure Handling
- Integration with Vertical Pod Autoscaler
- Handling Memory Spikes
- Demo
- Q&A

Motivation & Customer Scenario

Genome Analysis Use Case

- A long process .. huge files, lots of data crunching
- Series of K8s Jobs with many concurrent containers
- Some Jobs run for ~4 hours
- Several different steps, E2E takes about 1 day
- So, Job restart == lost work
- Customer deploys Pods sized for peak use, thus wasting resources
- They estimate ‘Elastic Pod’ feature can yield 50-60% cost savings

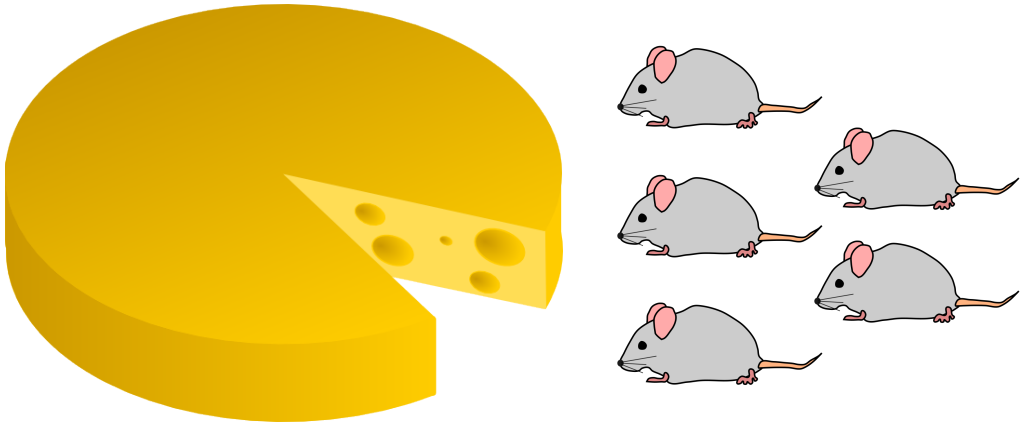


Long Desired Feature

- First In-Place Vertical Scaling feature request raised in 2015
- Statefulset , Deployment, Serverless use cases

Overview: Horizontal Scaling

- Increase or decrease Pod instances based on load
- Typically for stateless applications
- Triggers: CPU/Memory/custom metrics

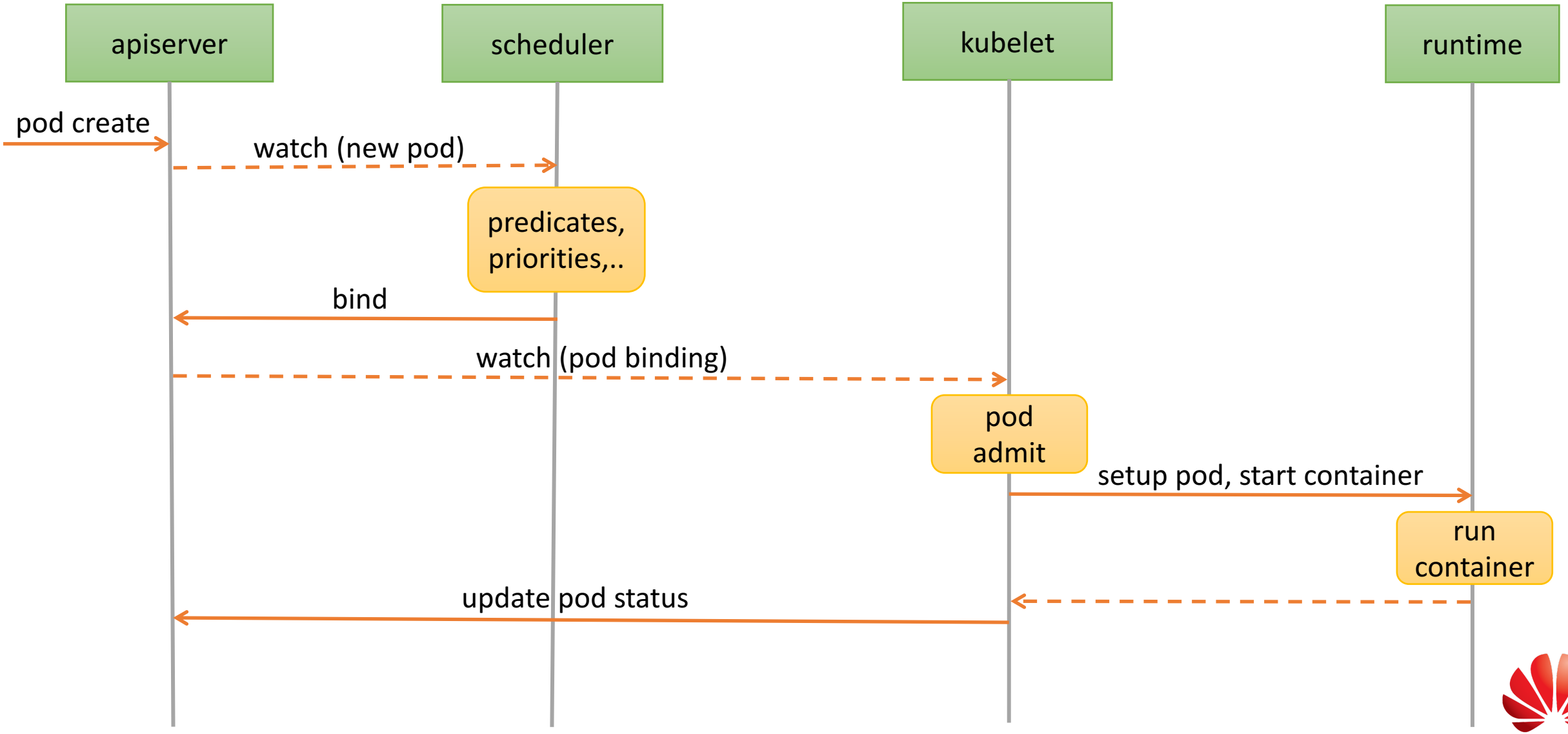


Overview: Vertical Scaling

- Increase or decrease Pod resources (CPU/Memory)
- Typically for stateful applications
- Triggers: CPU/Memory usage metrics

- Vertical Pod Autoscaler (VPA) project automates this
  - But requires Pod restart (currently)

K8s Scheduling Overview

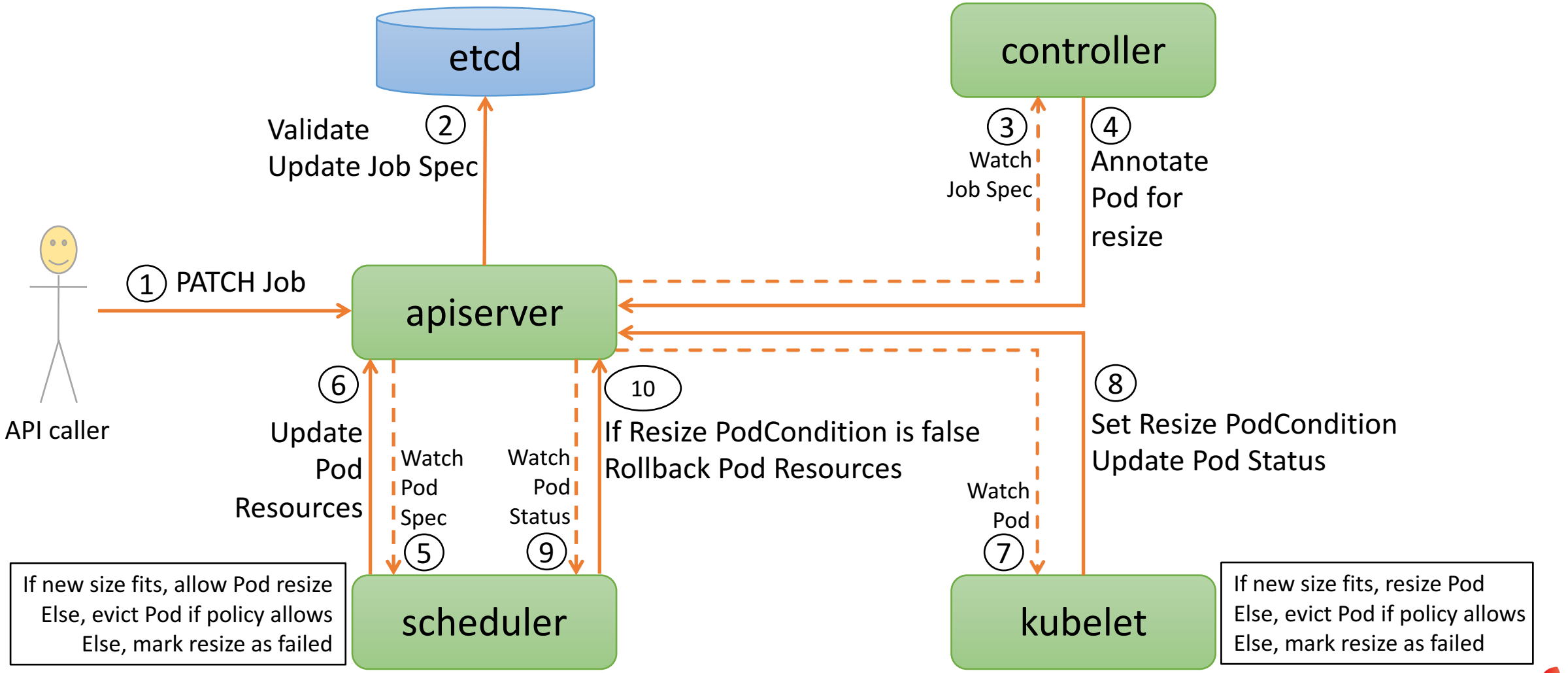


Vertical Scaling Design Choices

- Option 1
  - Directly update Pod resources
  - Scheduler updates cache, Kubelet applies resize in parallel
  - Problem: Race condition between Scheduler & Kubelet
- Option 2
  - Annotate Pod with desired resources
  - Scheduler reads annotation, updates Pod resources if node has capacity in its view
  - Kubelet ‘admits’ new resource values, if node has capacity

```
root@master:~/VS# cat 1job2do.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: 1job2do
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
      - name: stress
        image: skiibum/ubuntu-stress:18.10
        command: ["tail", "-f", "/dev/null"]
        resources:
          limits:
            cpu: "1"
            memory: "2Gi"
          requests:
            cpu: "1"
            memory: "2Gi"
```

Vertical Scaling Design



Policies & Failure Handling

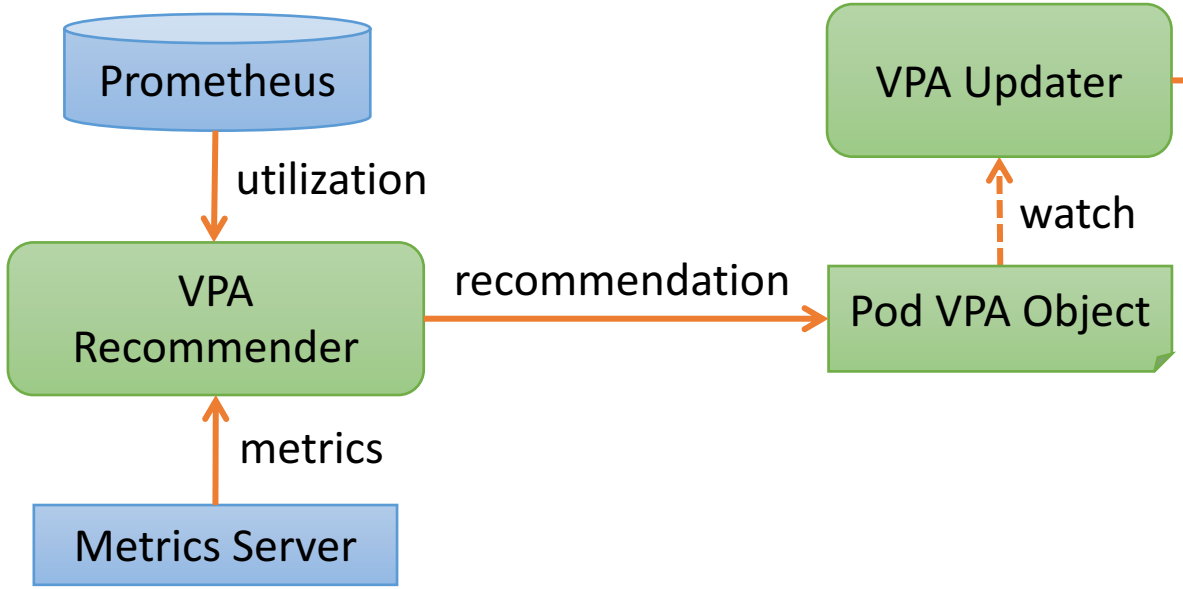
- Resource Resize Policies
  - InPlacePreferred
    - Respect PodDisruptionBudget if rescheduling
  - InPlaceOnly
    - For apps that don't tolerate restart
  - Restart
    - For Java apps or similar

```
root@master:~/VS# cat 1job2do.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: 1job2do
spec:
  template:
    metadata:
      annotations:
        oomKillDisable: "false"
        scheduler.alpha.kubernetes.io/resize-resources-policy: "InPlaceOnly"
    spec:
      restartPolicy: OnFailure
      containers:
      - name: stress
        image: skiibum/ubuntu-stress:18.10
        command: ["tail", "-f", "/dev/null"]
        resources:
          limits:
            cpu: "1"
            memory: "2Gi"
          requests:
            cpu: "1"
            memory: "2Gi"
```

- Resize Failure Handling
  - Resource resize can fail for a few reasons
    - Multiple Schedulers race condition
    - PodDisruptionBudget violation
  - On failure, Scheduler rolls back Pod resource update
  - Controller retries resource resize
    - Retry InPlaceOnly when other Pods depart
    - Retry when PodDisruptionBudget allows Pod eviction

Vertical Pod Autoscaler Integration

- VPA reads Pod resource usage from Metrics Server
- It reads utilization history from a time-series database
- VPA Recommender writes recommendation to VPA Object
- VPA Updater watches VPA Object, triggers resource resize




```
root@master:~/VS# cat 1job2do.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: 1job2do
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
      - name: stress
        image: skiibum/ubuntu-stress:18.10
        command: ["tail", "-f", "/dev/null"]
        resources:
          limits:
            cpu: "1"
            memory: "2Gi"
          requests:
            cpu: "1"
            memory: "2Gi"
```

```
root@master:~# k describe VerticalPodAutoscaler 1job2do-vpa
Name:     1job2do-vpa
Kind:     VerticalPodAutoscaler
...
 Recommendation:
  Container Recommendations:
   Container Name:  stress
   Lower Bound:
    Memory:  64Mi
   Target:
    Memory:  129Mi
   Upper Bound:
    Memory:  256Mi
```

Memory Usage Spikes

- Even with Restart-Free Vertical Scaling, spikes in memory usage can cause OOM Kill app terminations
  - Metrics sampling interval
  - VPA response time
- oomKillDisable annotation controls OOM Killer for Pod
- Pod apps are paused on reaching limit, until VPA can react
- Suitable for long-running Jobs where OOM Kill means significant loss of work

```
root@master:~/VS# cat 1job2do.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: 1job2do
spec:
  template:
    metadata:
      annotations:
        oomKillDisable: "true"
    spec:
      restartPolicy: OnFailure
      containers:
      - name: stress
        image: skiibum/ubuntu-stress:18.10
        command: ["tail", "-f", "/dev/null"]
        resources:
          limits:
            cpu: "1"
            memory: "2Gi"
          requests:
            cpu: "1"
            memory: "2Gi"
```

Demo


```
root@master:~/VS# kubectl patch job 1job2do --patch '{"spec":{"template":{"spec":{"containers":
[{"name":"stress", "resources":{"requests":{"memory":"3Gi"}, "limits":{"memory":"3Gi"}}}]}}}}'
job.batch/1job2do patched
root@master:~/VS#
```

update

```
root@master:~/VS# cat 1job2do.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: 1job2do
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
      - name: stress
        image: skiibum/ubuntu-stress:18.10
        command: ["tail", "-f", "/dev/null"]
        resources:
          limits:
            cpu: "1"
            memory: "2Gi"
          requests:
            cpu: "1"
            memory: "2Gi"
```

Vertical Scaling Success Story

- JD.com, one of China's largest online retailers, is using our work in pre-prod
  - Resize Deployment Pods for more optimal cluster resource utilization
  - Resize Pod resources down in order to schedule pending Pods

Resources + Q&A

- Design doc
  - https://docs.google.com/document/d/18K-bl1EVsmJ04xeRq9o_vfY2GDgek6B6wmLjXw-kos4/
- Implementation
  - https://github.com/Huawei-PaaS/kubernetes/tree/vertical-scaling
  - https://github.com/Huawei-PaaS/kubernetes/pull/37
- Latest In-Place Vertical Scaling Kubernetes Enhancement Proposal (KEP)
  - https://github.com/kubernetes/enhancements/pull/686#