KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon | CloudNativeCon
Europe 2022

Autoscaling Kubernetes Deployments: A (Mostly) Practical Guide

Natalie Serrino, New Relic

About

- 💻 Software engineer / TLM at New Relic
- 💥 Contributor to Pixie
- 💜 Observability and performance
- 📊 Formerly worked in the data space

Natalie Serrino

Content

🤔 What is Kubernetes autoscaling (and why is it useful)?

⏲️ What knobs does Kubernetes autoscaling give us?

📈 Selecting an autoscaling metric for your application

🤓 A Turing-complete autoscaler (?!)

What is Kubernetes autoscaling?

Resource sizing in Kubernetes

How do you select the following values?

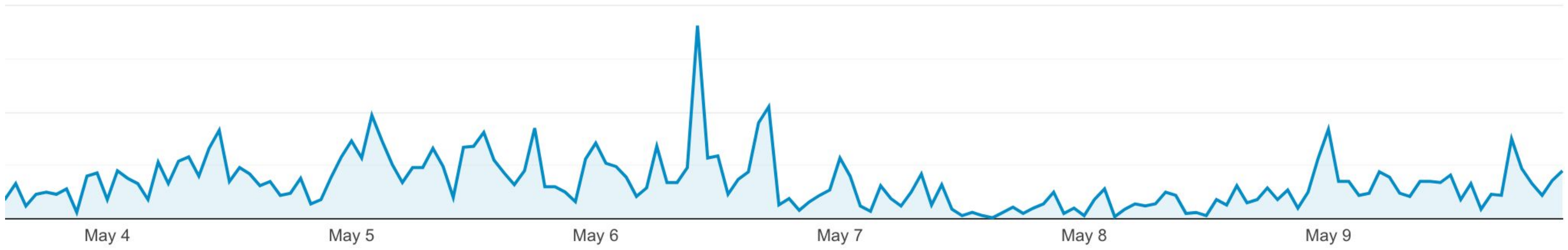
💰 # of nodes in your cluster
🚀 # of pods in a deployment
📶 Amount of resources to give a pod

Methods…

- Random guess
- Copypasta
- Proactive iteration
- Reactive iteration

Why is autoscaling useful?

- Ideal resource allocation depends on workload
- Workloads are often spikey and unpredictable



- Too few resources → bad user experience, latency, outages 😱
- Too many resources → wasteful, expensive 💸 💸 💸

The types of autoscaling in Kubernetes

Kubernetes autoscaler	**…adds/removes…**	**…to your…**	**…based on…**
`Cluster Autoscaler`	Nodes	Cluster	Resource utilization
`VerticalPodAutoscaler`	Resources like CPU/Memory	Existing replicas	Resource utilization
`HorizontalPodAutoscaler`	Replicas	Workload	Resource utilization or user-defined metrics

Cluster Autoscaler

- 💡 Set pod resource requests and limits
- 💡 Make sure resource requests reflect actual usage
- 💡 Specify PodDisruptionBudgets
- 💡 Compare limits with available quota from your cloud provider (if applicable)
- ⚠️ Kubernetes contributors tested for <=1000 nodes with 30 pods each

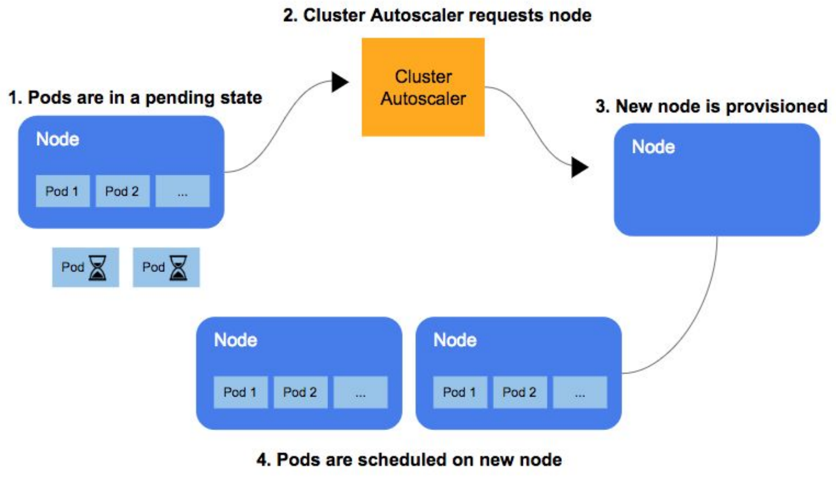

2. Cluster Autoscaler requests node
Cluster Autoscaler
1. Pods are in a pending state
Node
Pod 1
Pod 2
...
Pod
Pod
3. New node is provisioned
Node
Node
Pod 1
Pod 2
...
Node
Pod 1
Pod 2
...
4. Pods are scheduled on new node

Image Credit: Understanding Kubernetes Cluster Autoscaling by Ajay Tripathy

VerticalPodAutoscaler (VPA)

- 💡 Can still set resource caps with VPA
- 💡 Updates may result in container restarts or pod rescheduling
- 💡 Use in cluster autoscaler to avoid VPA recommending more than available resources
- ⚠️ Can't use with HorizontalPodAutoscaler on the same application for CPU/memory yet
- ⚠️ Has not yet been tested on large clusters

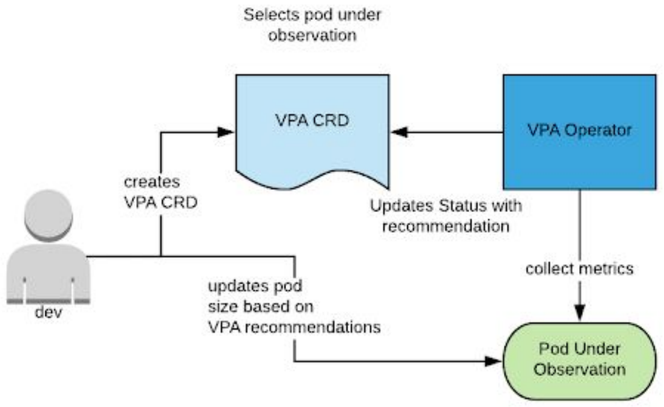
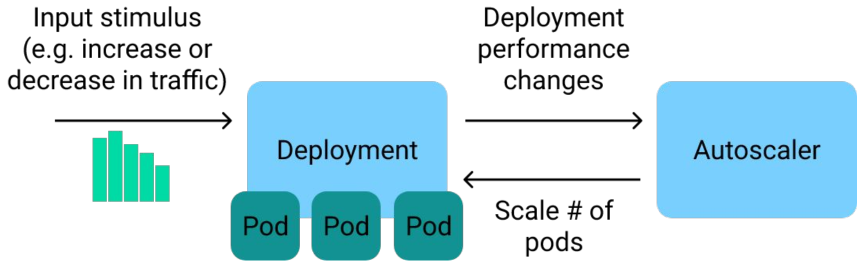

Selects pod under observation
VPA CRD
VPA Operator
creates VPA CRD
Updates Status with recommendation
collect metrics
dev
updates pod size based on VPA recommendations
Pod Under Observation

Image Credit: Red Hat, How Full is my Cluster, Raffaele Spazzoli

HorizontalPodAutoscaler (HPA)

- 💡 Lots of flexibility for metric selection
- 💡 Check your service client affinity policies to ensure even load distribution
- 💡 When scaling on CPU/memory, make sure to set resource requests
- ⚠️ Can’t use with VerticalPodAutoscaler on same application on CPU/memory yet


Input stimulus (e.g. increase or decrease in traffic)
Deployment
Pod
Pod
Pod
Deployment performance changes
Autoscaler
Scale # of pods

Demo: Horizontal pod autoscaling on CPU

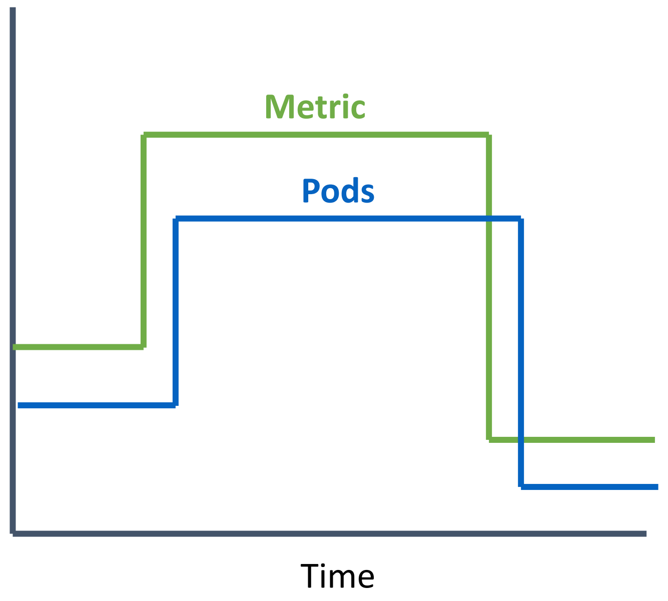
HPA autoscaling equation

```
outputReplicas = ceil(
    currentReplicas * ( currentMetricValue / desiredMetricValue )
)
```

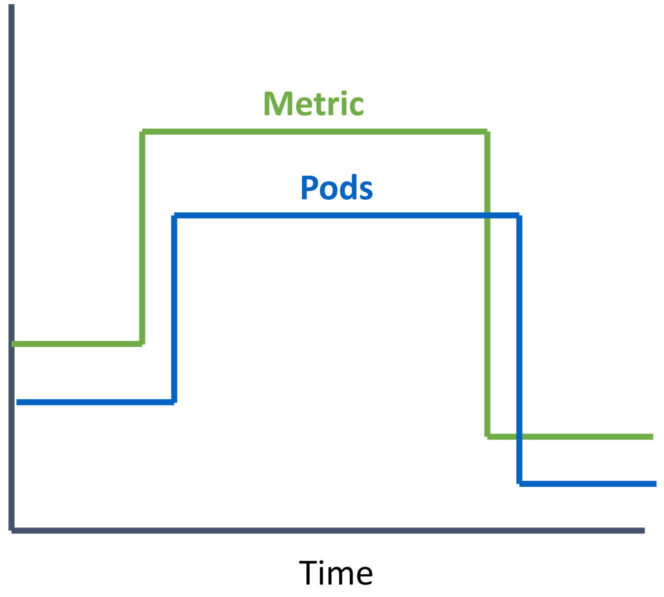
What knobs does Kubernetes autoscaling give us?

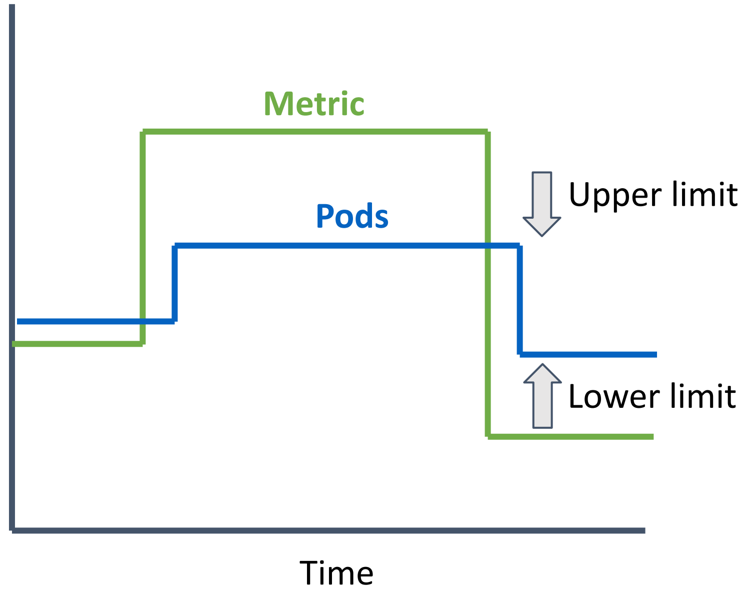
Scaling up and down

- How to set the minimum and maximum number of replicas?
- How often to look for changes in the metric?
- How quickly to add or remove pods?
- How many pods to add/remove in a single period?

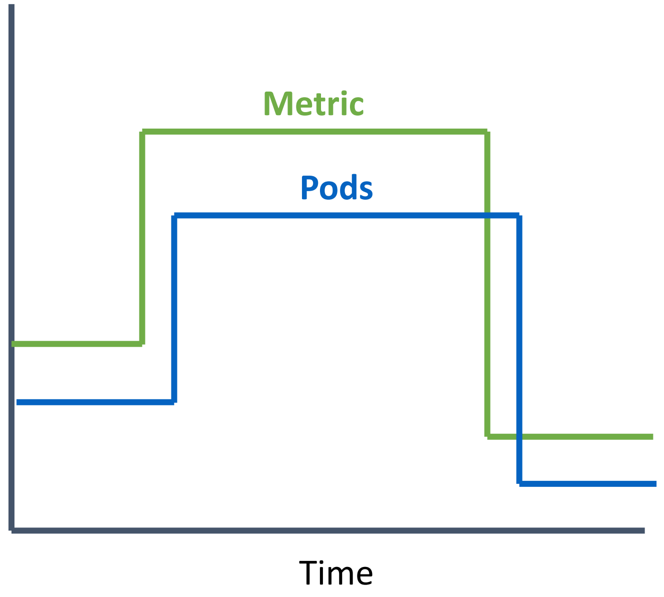


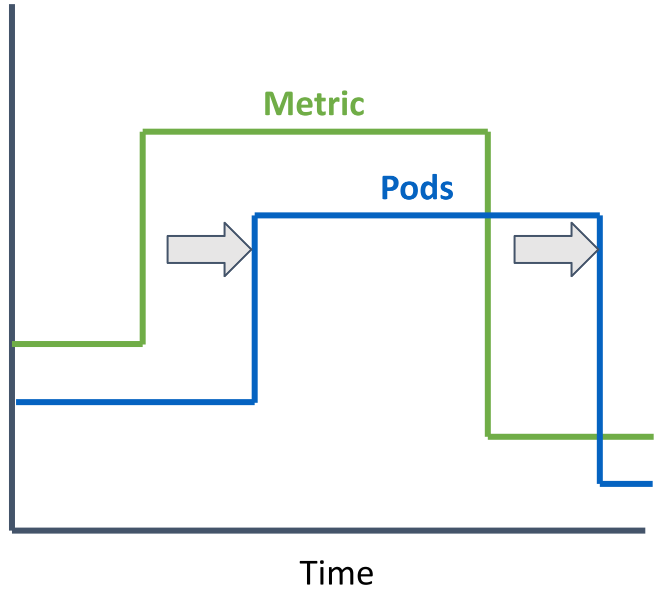
Capping min/max pods



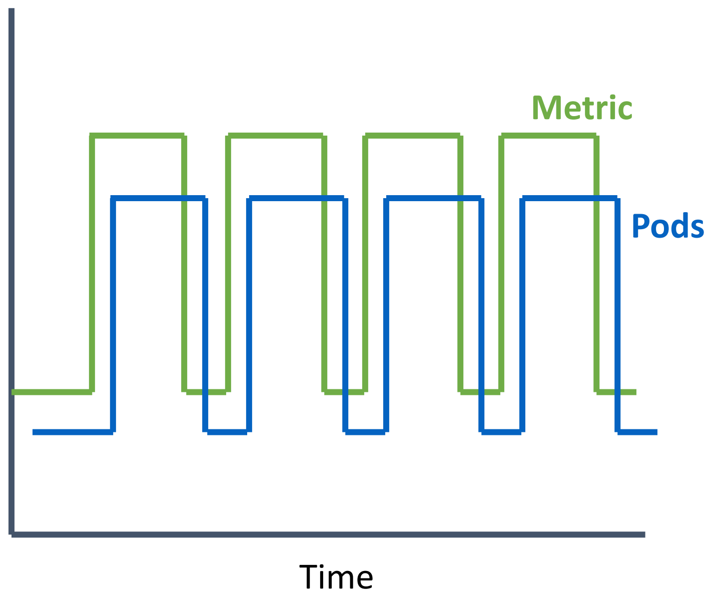


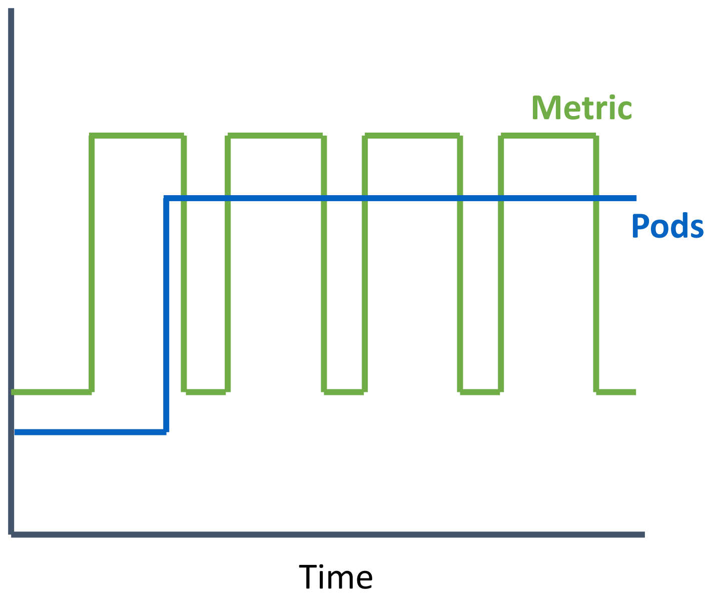
Stabilization period



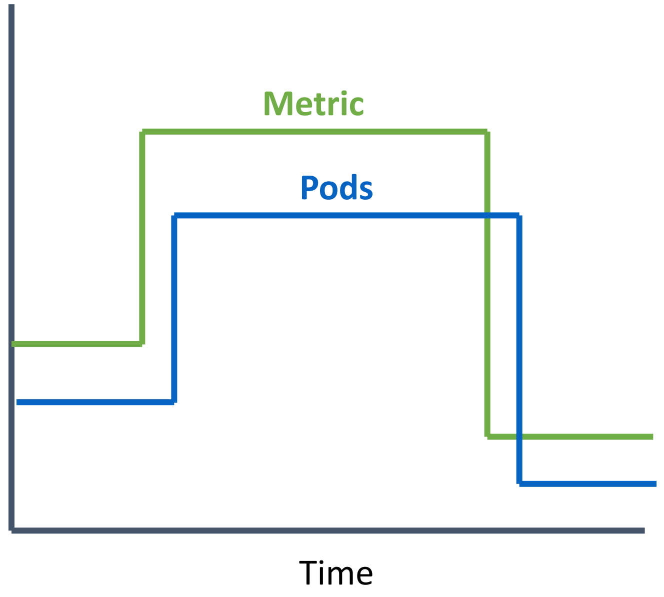


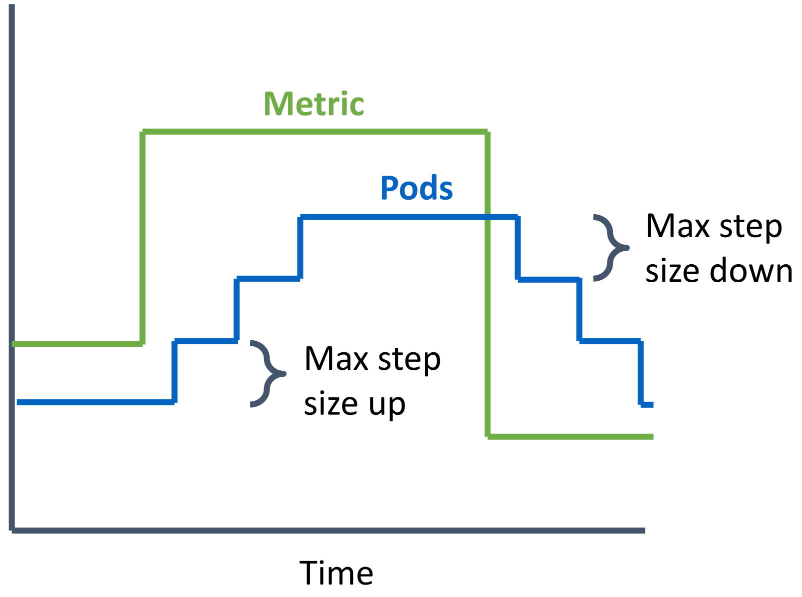
Stabilization period reduces pod churn



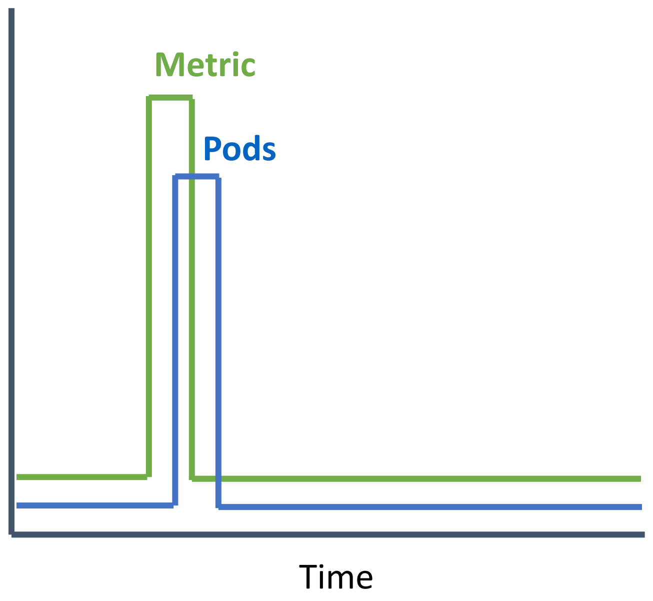


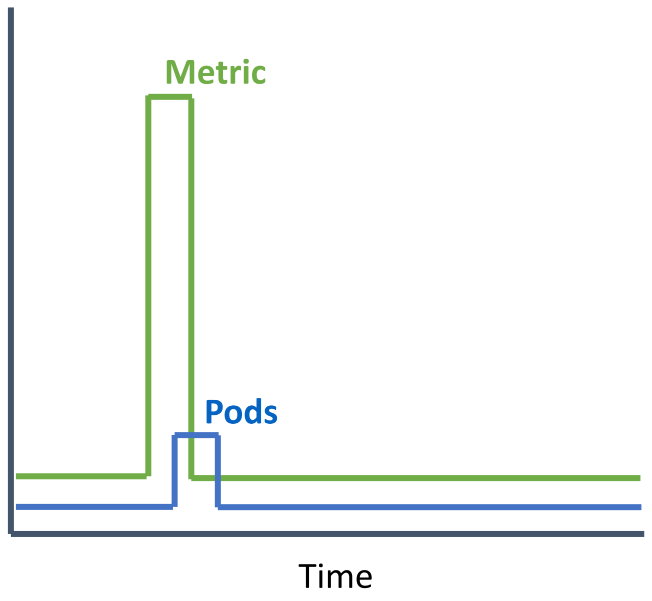
Capping max pods to add/remove per period





Capping max step size reduces pod churn





Selecting an autoscaling metric for your application

HPA Metric Types

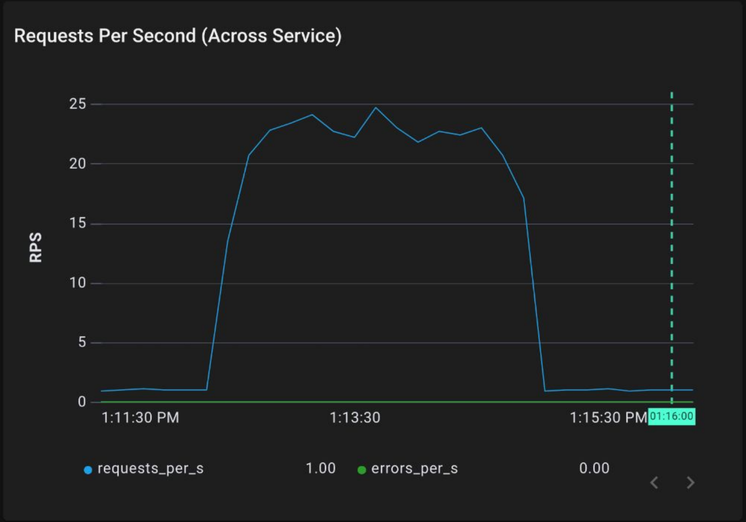
Category	**Type**	**Description**	**Examples**
`Resource Metrics`	Built-in	Resource utilization metrics for pods and nodes only	Currently limited to CPU and memory
`Custom Metrics`	User-defined	Custom metrics about Kubernetes resources	Latency, throughput, queue depth
`External Metrics`	User-defined	Custom metrics NOT about Kubernetes resources	# of customers using website

Possible bottlenecks in an application

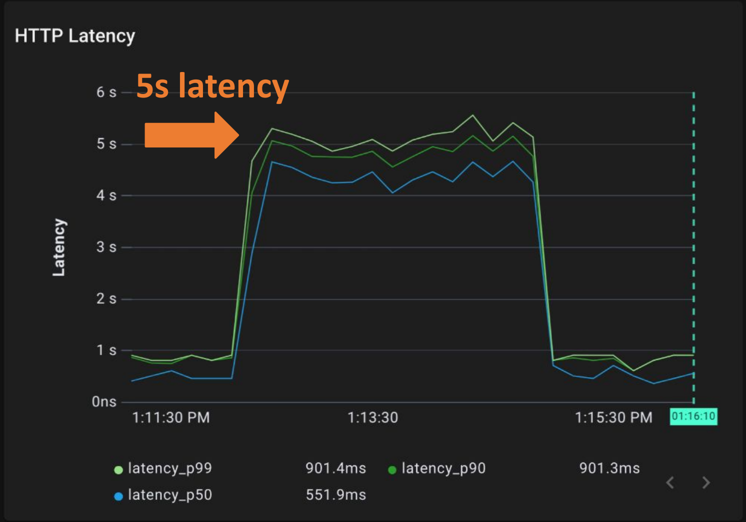
- CPU
- Memory
- Network
- # of worker threads
- # of outbound connections
- Downstream dependencies
- Queue depth
- …Many more

The best metric to scale on depends on your workload!

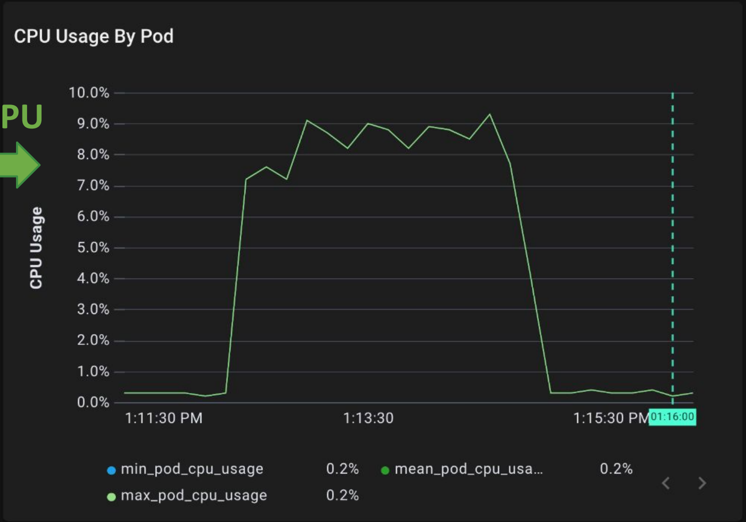
Before #1

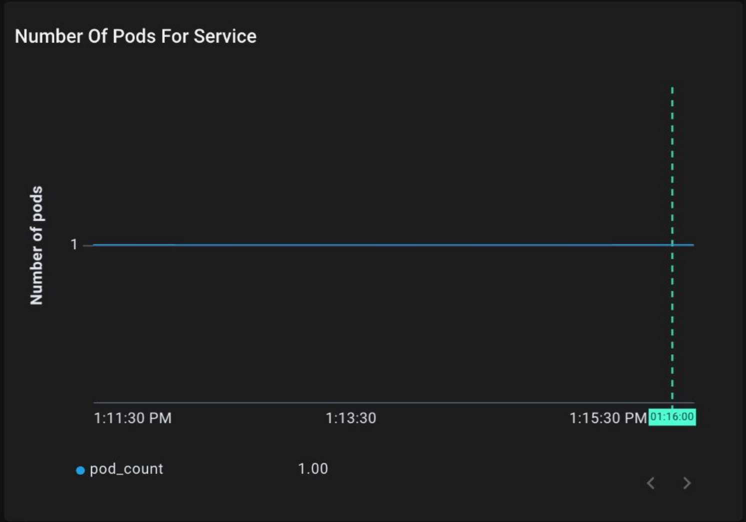


5s latency



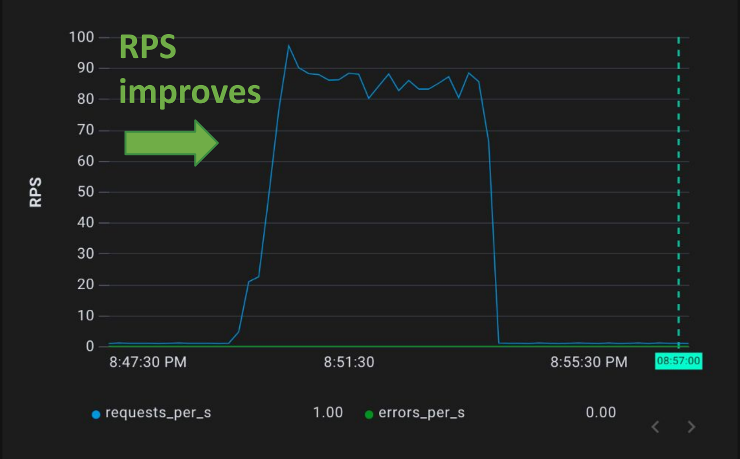
Low CPU



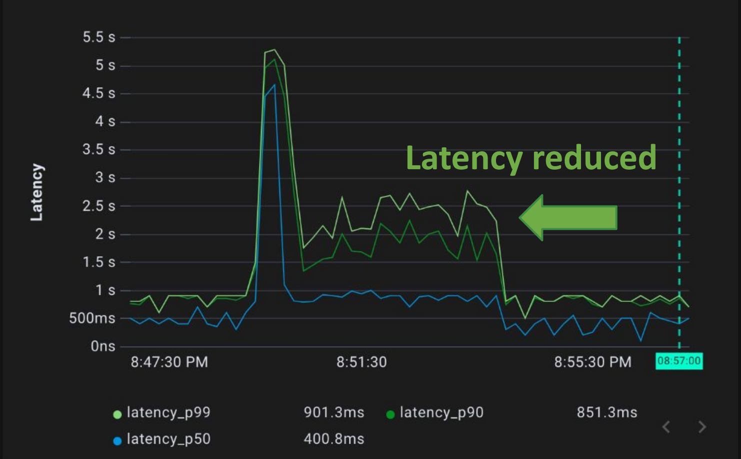


After #1

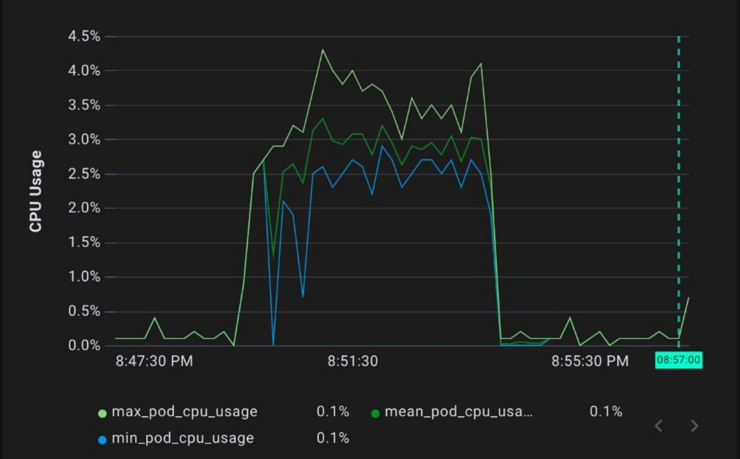
Requests Per Second (Across Service)



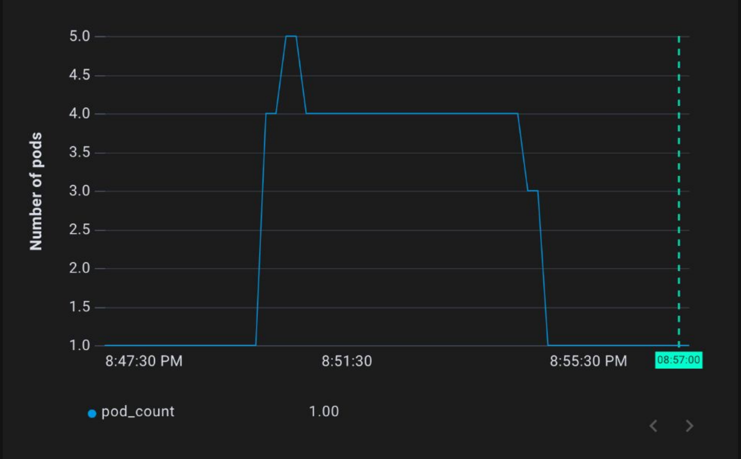
HTTP Latency



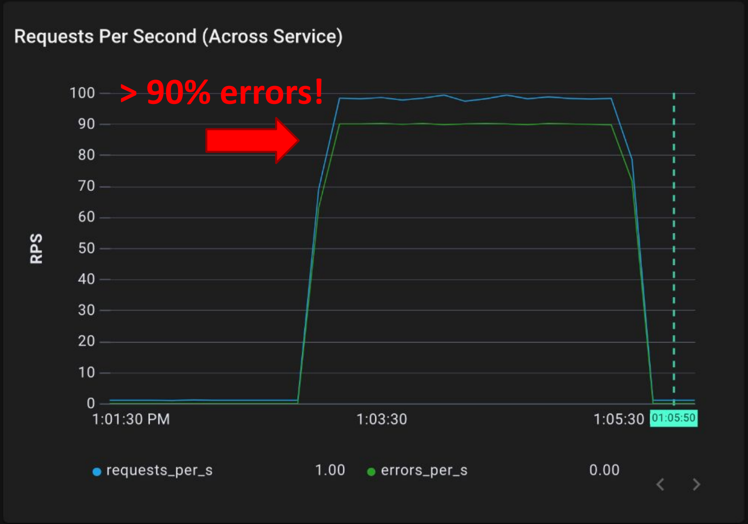
CPU Usage By Pod

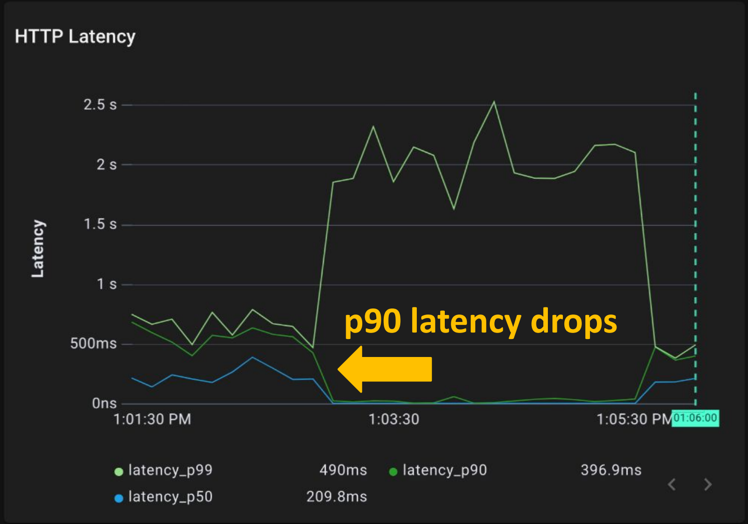


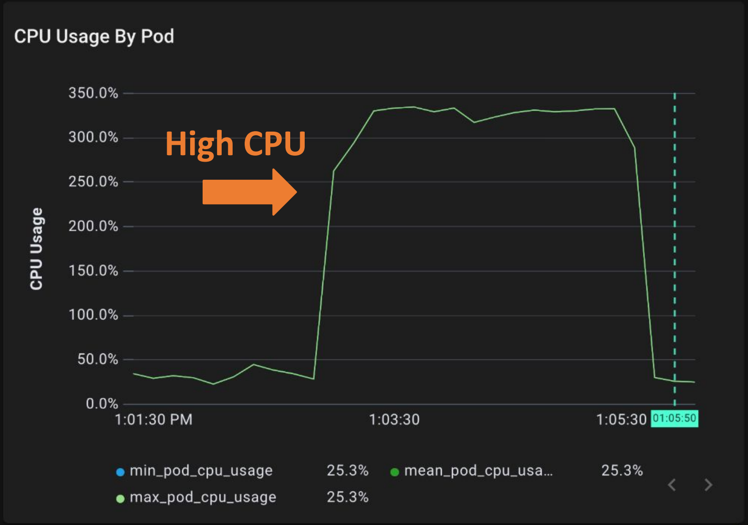
Number Of Pods For Service

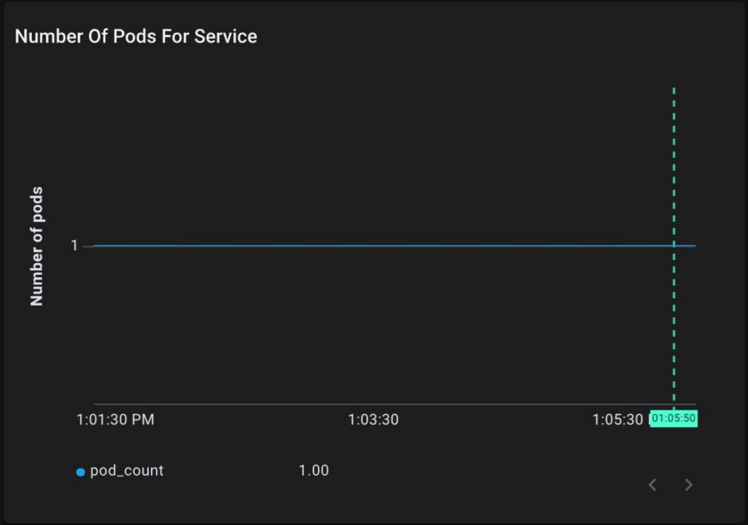


Before #2

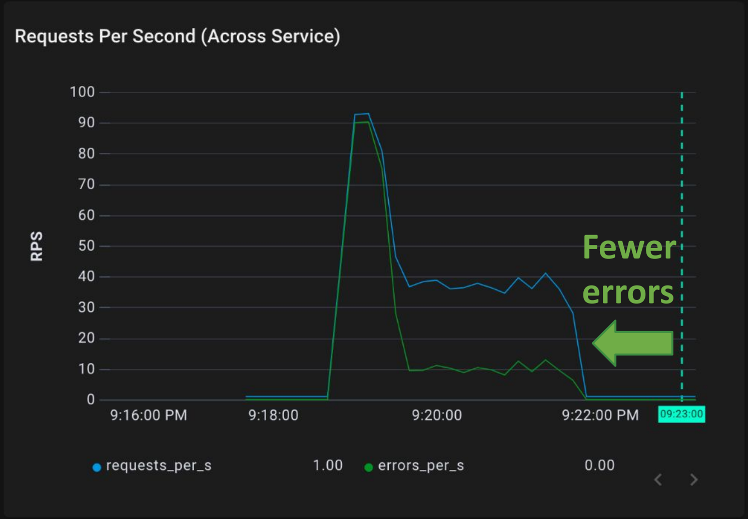

Requests Per Second (Across Service)
> 90% errors!
RPS
100
90
80
70
60
50
40
30
20
10
0
1:01:30 PM
1:03:30
1:05:30
01:05:50
● requests_per_s 1.00
● errors_per_s 0.00

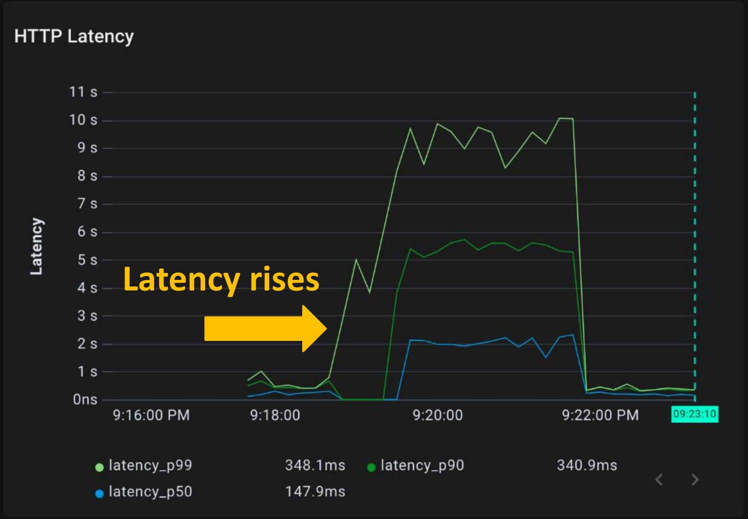

HTTP Latency
Latency
2.5 s
2 s
1.5 s
1 s
500ms
0ns
p90 latency drops
1:01:30 PM
1:03:30
1:05:30 PM
01:06:00
● latency_p99 490ms
● latency_p90 396.9ms
● latency_p50 209.8ms

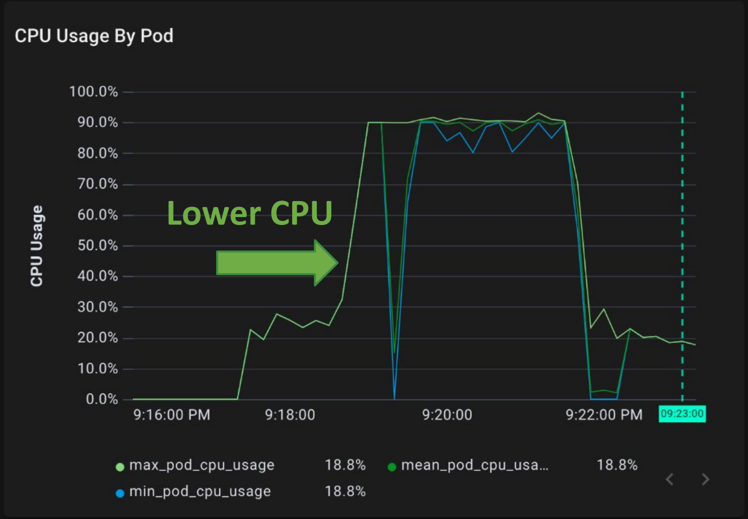

CPU Usage By Pod
CPU Usage
350.0%
300.0%
250.0%
200.0%
150.0%
100.0%
50.0%
0.0%
High CPU
1:01:30 PM
1:03:30
1:05:30
01:05:50
● min_pod_cpu_usage 25.3%
● mean_pod_cpu_usa... 25.3%
● max_pod_cpu_usage 25.3%

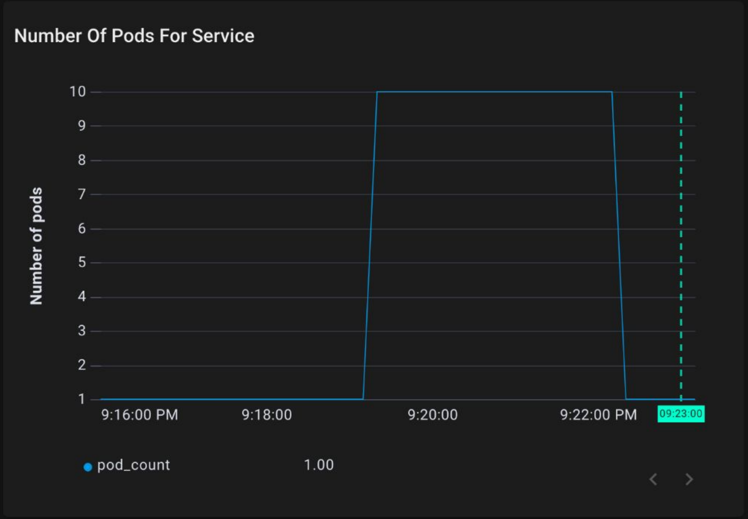

Number Of Pods For Service
Number of pods
1
1:01:30 PM
1:03:30
1:05:30
01:05:50
● pod_count 1.00

After #2


Requests Per Second (Across Service)
RPS
100
90
80
70
60
50
40
30
20
10
0
9:16:00 PM
9:18:00
9:20:00
9:22:00 PM
09:23:00
Fewer errors
● requests_per_s 1.00
● errors_per_s 0.00


HTTP Latency
Latency
11 s
10 s
9 s
8 s
7 s
6 s
5 s
4 s
3 s
2 s
1 s
0ns
9:16:00 PM
9:18:00
9:20:00
9:22:00 PM
09:23:10
Latency rises
● latency_p99 348.1ms
● latency_p90 340.9ms
● latency_p50 147.9ms

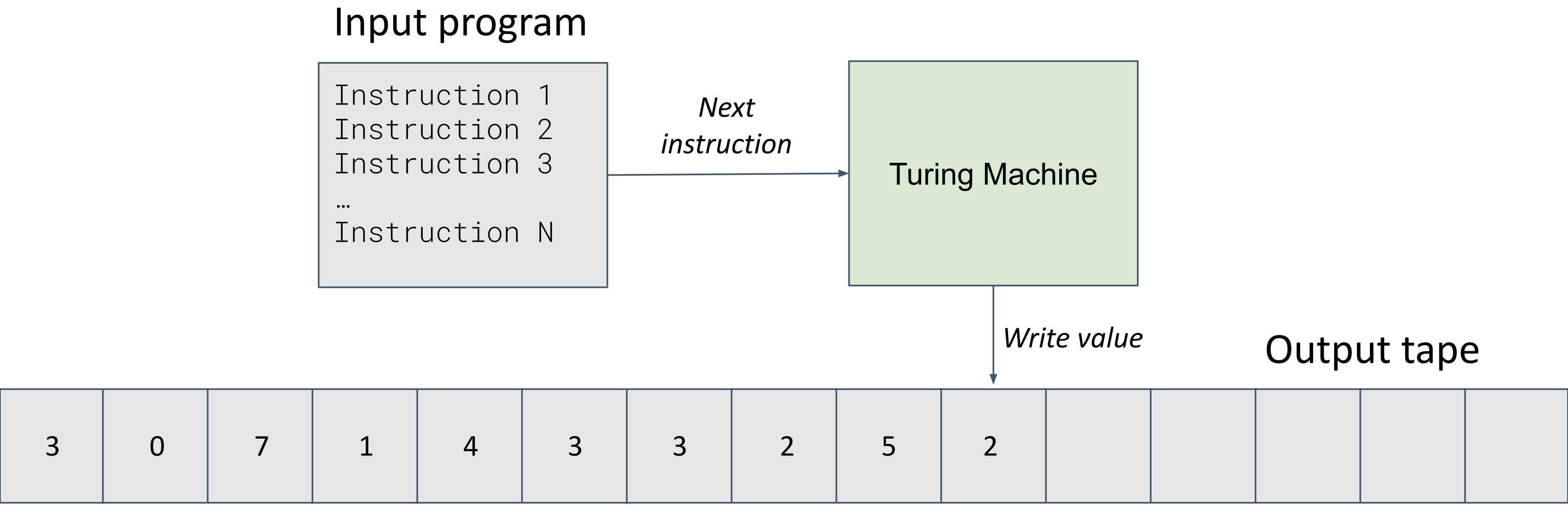

CPU Usage By Pod
CPU Usage
100.0%
90.0%
80.0%
70.0%
60.0%
50.0%
40.0%
30.0%
20.0%
10.0%
0.0%
9:16:00 PM
9:18:00
9:20:00
9:22:00 PM
09:23:00
Lower CPU
● max_pod_cpu_usage 18.8%
● mean_pod_cpu_usa... 18.8%
● min_pod_cpu_usage 18.8%


Number Of Pods For Service
Number of pods
10
9
8
7
6
5
4
3
2
1
9:16:00 PM
9:18:00
9:20:00
9:22:00 PM
09:23:00
● pod_count 1.00

A Turing-complete autoscaler?

Turing machine

Turing machines are capable of any computation, given enough time and tape.



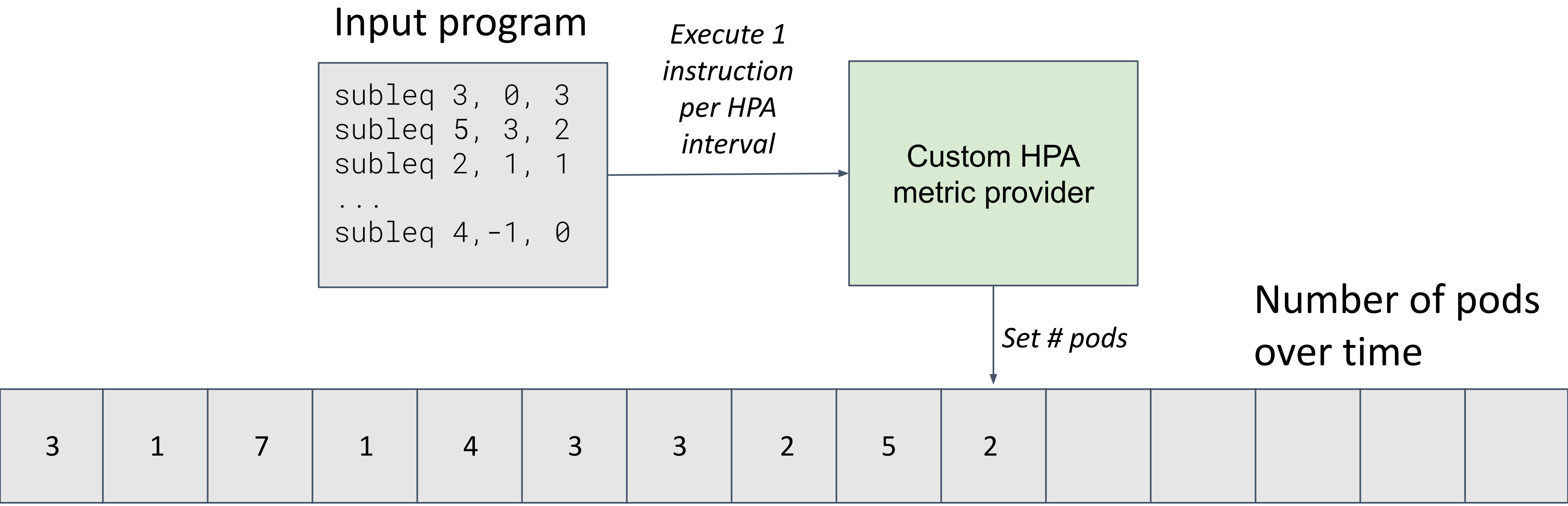
Subleq

“One-instruction set computer”, sufficient for Turing completeness.

```
Instruction subleq a, b, c
    Mem[b] = Mem[b] - Mem[a]
    if (Mem[b] ≤ 0)
        goto c
```

Subleq pseudocode (credit: Wikipedia)

HPA subleq Turing machine


Input program
Execute 1 instruction per HPA interval
Custom HPA metric provider
Set # pods
Number of pods over time
3
1
7
1
4
3
3
2
5
2

```
subleq 3, 0, 3
subleq 5, 3, 2
subleq 2, 1, 1
...
subleq 4,-1, 0
```

Input program: deployment name

```
9x-1x3x10x-1x6x0x0x-1x72x105x0
```

Split on "x"

```
9,-1,3,10,-1,6,0,0,-1,72,105,0
```

3 input args per subleq

```
subleq  9, -1, 3
subleq 10, -1, 6
subleq  0,  0,-1
subleq 72,105, 0
```

Setting a certain # of output pods

```
outputReplicas = ceil(
    currentReplicas * ( currentMetricValue / targetMetricValue )
)
```

Backwards calulating "current metric value"

```
currentMetricValue = (
    targetMetricValue * ( outputReplicas / currentReplicas )
)
```

Demo: Turing-complete autoscaler

kubernetes

SIG instrumentation

github.com/kubernetes-sigs/custom-metrics-apiserver

PIXIE

CNCF Sandbox Project

github.com/pixie-io/pixie

blog.px.dev

HEY

Load generator

github.com/rakyll/hey

Thanks!