

Where's your money going?

The Beginners Guide To Measuring Kubernetes Costs



Richard "RichiH" Hartmann
Office of the CTO

<ul style="list-style-type: none"> <ul style="list-style-type: none"> Elastic Compute Cloud <ul style="list-style-type: none"> US East (N. Virginia) 	
Amazon Elastic Compute Cloud NatGateway	
\$0.045 per GB Data Processed by NAT Gateways	
\$0.045 per NAT Gateway Hour	
Amazon Elastic Compute Cloud running Linux/UNIX	
\$0.0052 per On Demand Linux t3.nano Instance Hour	
\$0.0116 per On Demand Linux t2.micro Instance Hour	
\$0.0464 per On Demand Linux t2.medium Instance Hour	
\$0.096 per On Demand Linux m5.large Instance Hour	
EBS	
\$0.00 for 480 Mbps per m5.large instance-hour (or partial hour)	
\$0.05 per GB-month of snapshot data stored - US East (Northern Virginia)	
\$0.10 per GB-month of General Purpose SSD (gp2) provisioned storage - US East (Northern Virginia)	
Elastic Load Balancing - Classic	
\$0.008 per GB Data Processed by the LoadBalancer	
\$0.025 per LoadBalancer-hour (or partial hour)	

1,263.028 GB
744 Hrs

1,488 Hrs
2,976 Hrs
4,923.783 Hrs
744 Hrs

744 Hrs
254.756 GB-Mo
665.162 GB-Mo

625.658 GB
1,488 Hrs

\$553.92

\$553.92

\$90.32

\$56.84
\$33.48

\$342.15

\$7.74
\$34.52
\$228.46
\$71.42

\$79.25

\$0.00
\$12.74
\$66.52

\$42.21

\$5.01
\$37.20

*Not an actual bill

Monthly costs by service

Last 6 Months

Monthly

Stack

Group by: Instance Type

Service

Linked Account

Region

Usage Type

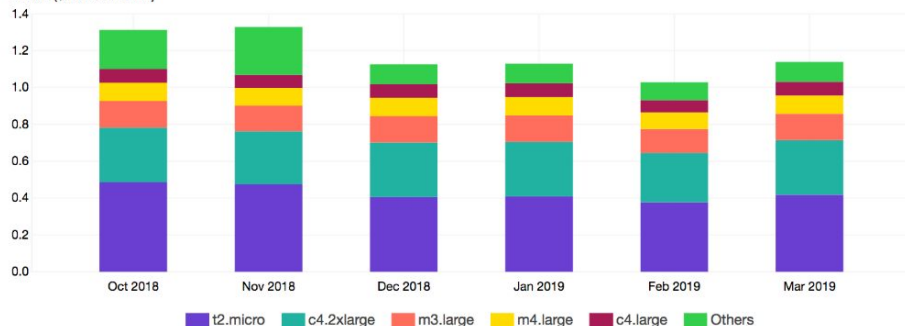
Tag

API Operation

Availability Zone

More

Costs (\$ in thousands)



Download CSV

Instance Type	Oct 1, 2018	Nov 1, 2018	Dec 1, 2018	Jan 1, 2019
Total cost (\$)	1,312.71	1,328.54	1,125.99	1,129.65
t2.micro (\$)	486.75	475.89	405.63	409.27
c4.2xlarge (\$)	296.11	286.56	296.11	296.11

FILTERS

CLEAR ALL

Service

Include only

EC2-Instances

1

Linked Account

Include all

Region

Include all

Instance Type

Include all

Usage Type

Include all

Usage Type Group

Include all

Tag

Include All

API Operation

Include all

Charge Type

Include all

More filters

ADVANCED OPTIONS

1

Show costs as

Unblended costs

Include costs related to

☐ Show only untagged resources

*Not an actual bill

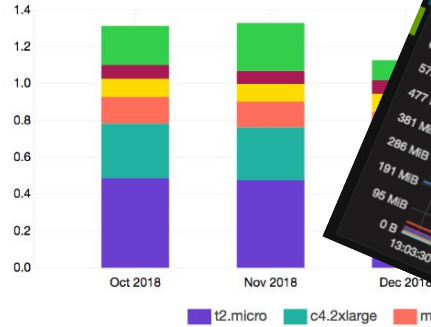
Monthly costs by service

Last 6 Months

Monthly

Group by: Instance Type x Service Linked Account Region

Costs (\$ in thousands)



Instance Type	Oct 1, 2018	Nov 1, 2018	Dec 1, 2018
Total cost (\$)	1,312.71	1,328.54	1,125.99
t2.micro (\$)	486.75	475.89	405.63
c4.2xlarge (\$)	296.11	286.56	296.11

*Not an actual bill



Monthly costs by service

Last 6 Months

Monthly

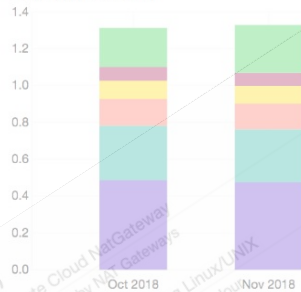
Group by: Instance Type

Service

Linked Account

Region

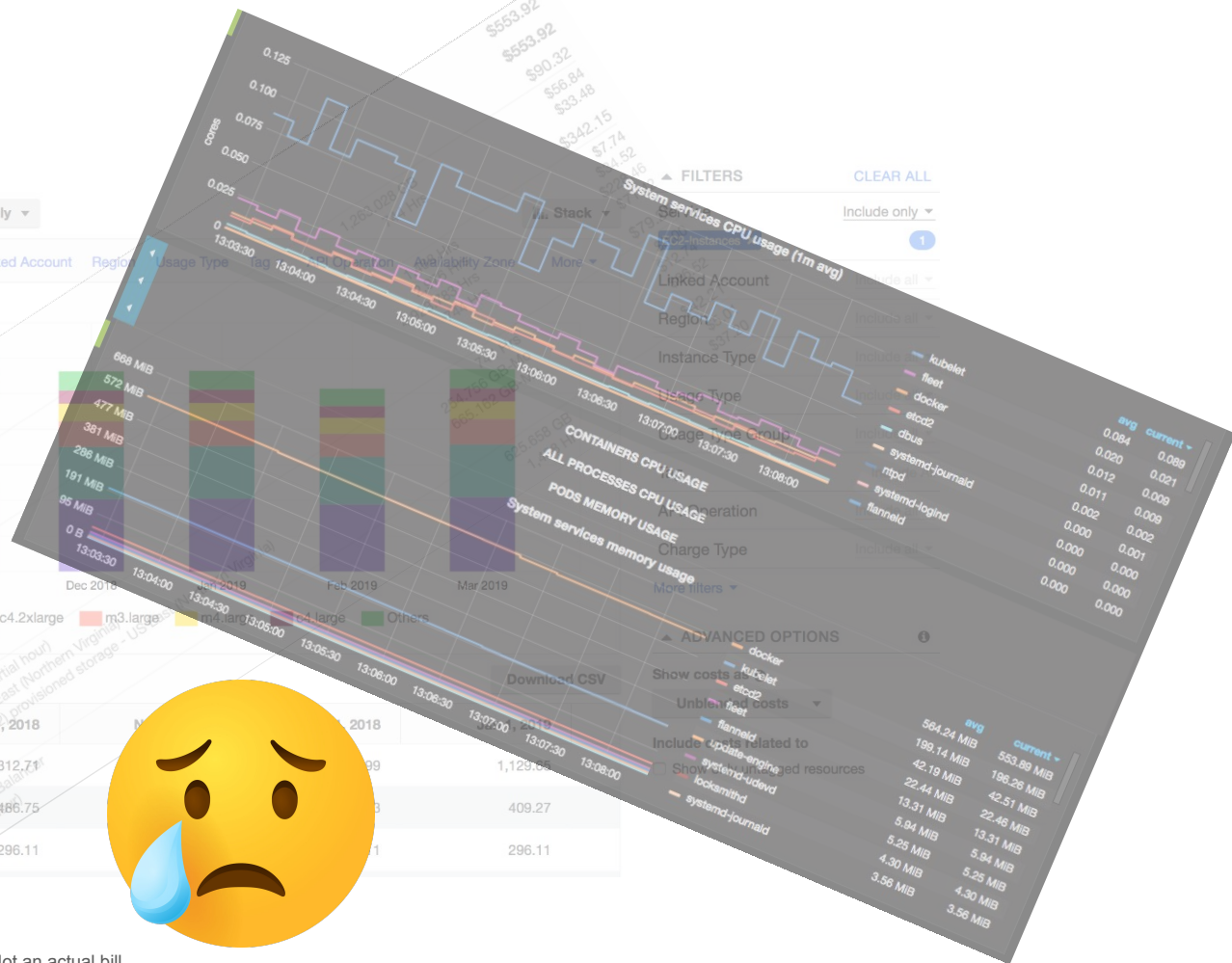
Costs (\$ in thousands)



Instance Type	Oct 1, 2018
Total cost (\$)	1,312.71
t2.micro (\$)	486.75
c4.2xlarge (\$)	296.11



*Not an actual bill



What you can expect

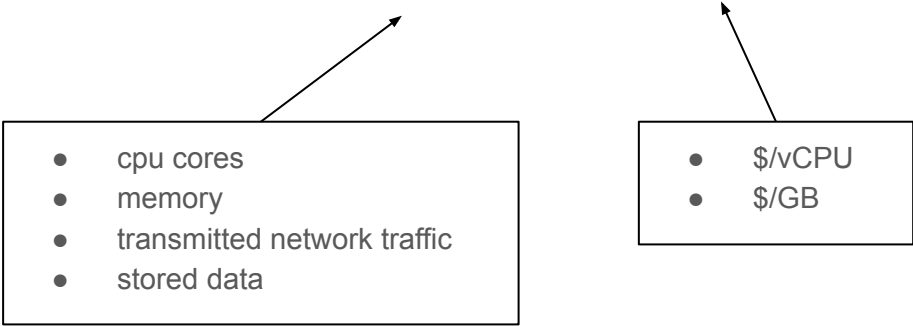
- Disconnect between billing statement and metrics
- How to attribute costs of workloads
- Lessons learned

A simple yet powerful formula ...

$$\text{Spend} = \text{Usage} \times \text{Rate}$$

A simple yet powerful formula ...

$$\text{Spend} = \text{Usage} \times \text{Rate}$$

- 
- cpu cores
 - memory
 - transmitted network traffic
 - stored data

- \$/vCPU
- \$/GB

A simple yet powerful formula ...

$$\text{Spend} = \text{Usage} \times \text{Rate}$$

-
- cpu cores
 - memory
 - transmitted network traffic
 - stored data

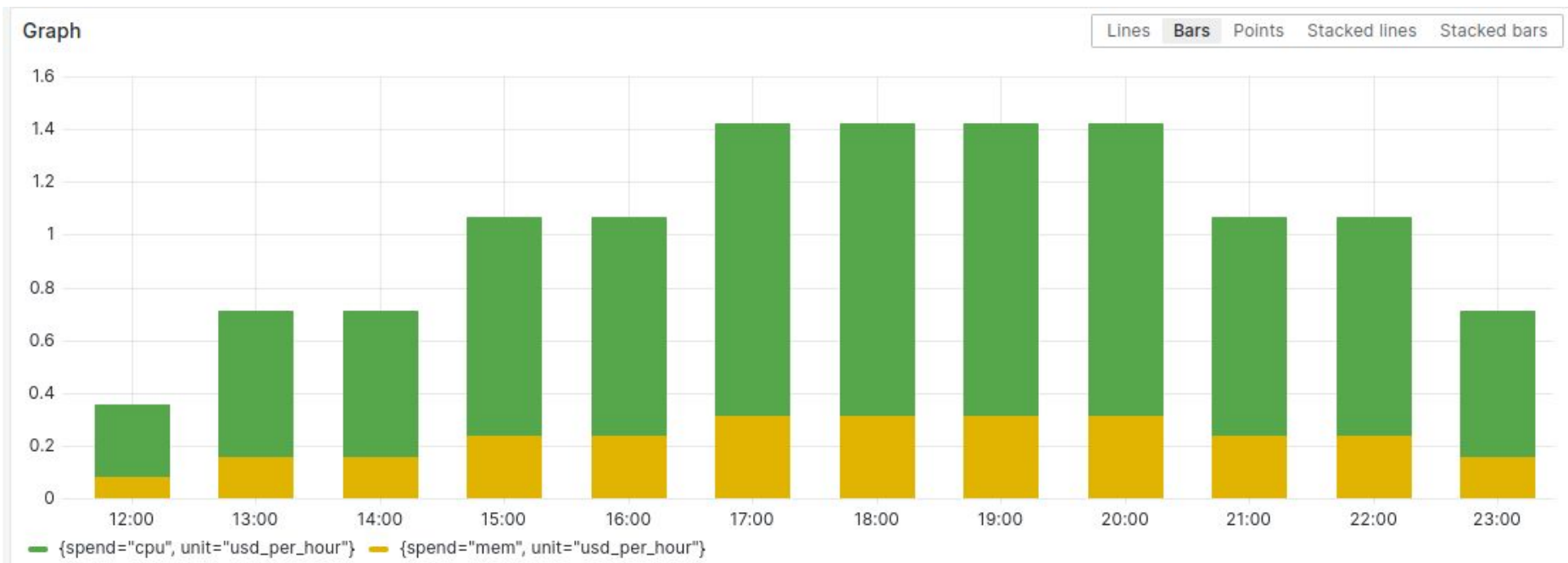
- \$/vCPU
- \$/GB

- Workload rightsizing
- Demand-based autoscaling
- **Cluster bin-packing**

- Discount coverage
(commitments, cpu arch,
spotVMs)

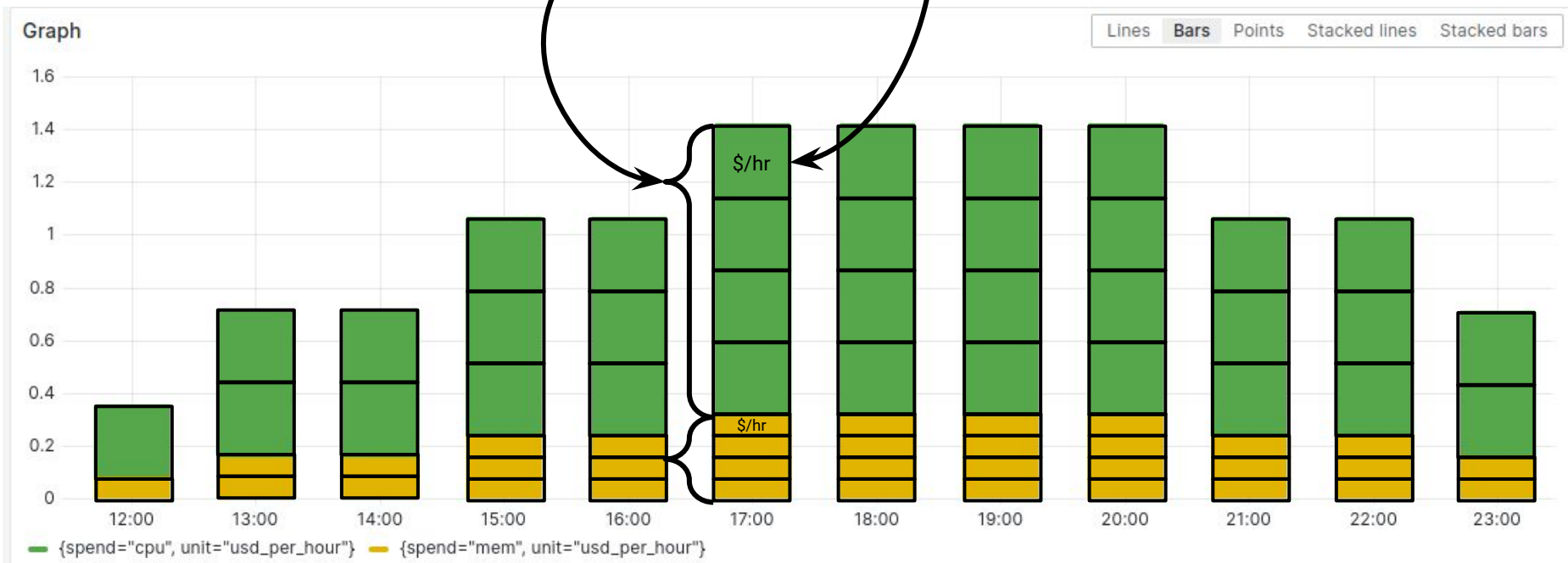
A simple yet powerful formula ...

$$\text{Spend} = \text{Usage} \times \text{Rate}$$



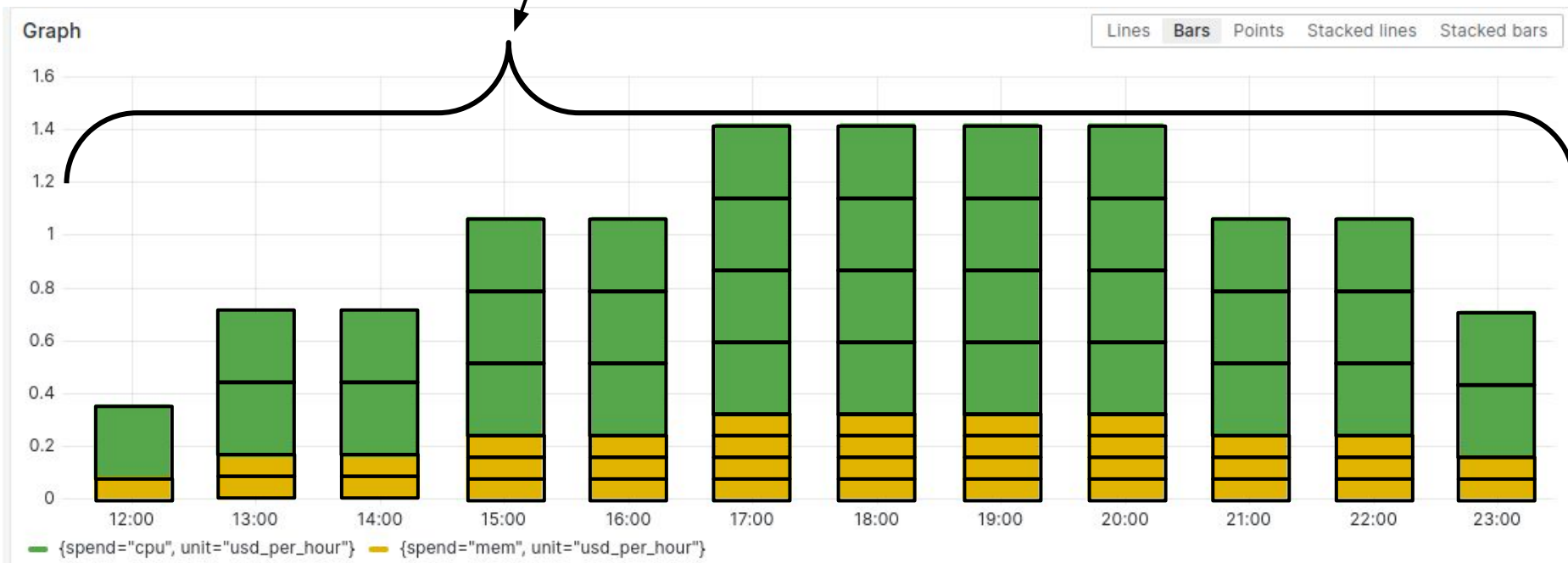
A simple yet powerful formula ...

$$\text{Spend} = \text{Usage} \times \text{Rate}$$



A simple yet powerful formula ...

$$\text{Spend} = \sum \text{Usage} \times \text{Rate} \Delta t$$



What drives k8s costs

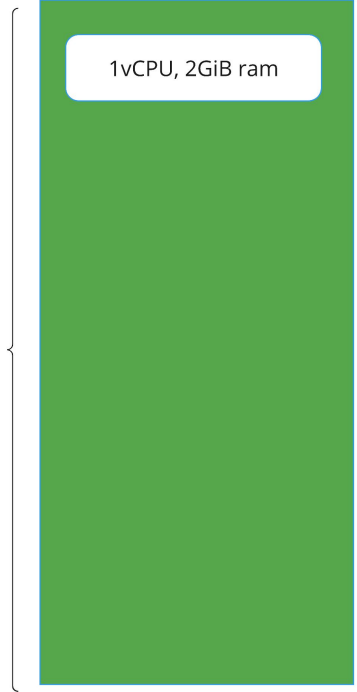
What drives k8s costs

Web service

1vCPU, 2GiB ram

What drives k8s costs

Node
8 vCPU
16GiB Ram

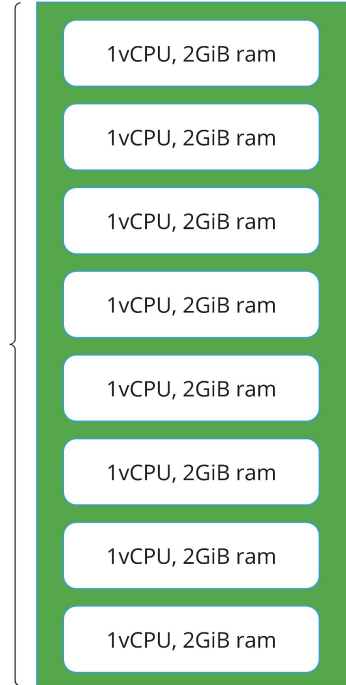


A diagram illustrating the relationship between a Kubernetes node and a pod. On the left, a vertical green rectangle represents the node. To its left, a bracket indicates the node's specifications: 8 vCPU and 16GiB Ram. Inside the top of the green rectangle is a smaller white rounded rectangle representing a pod, which contains the text '1vCPU, 2GiB ram'.

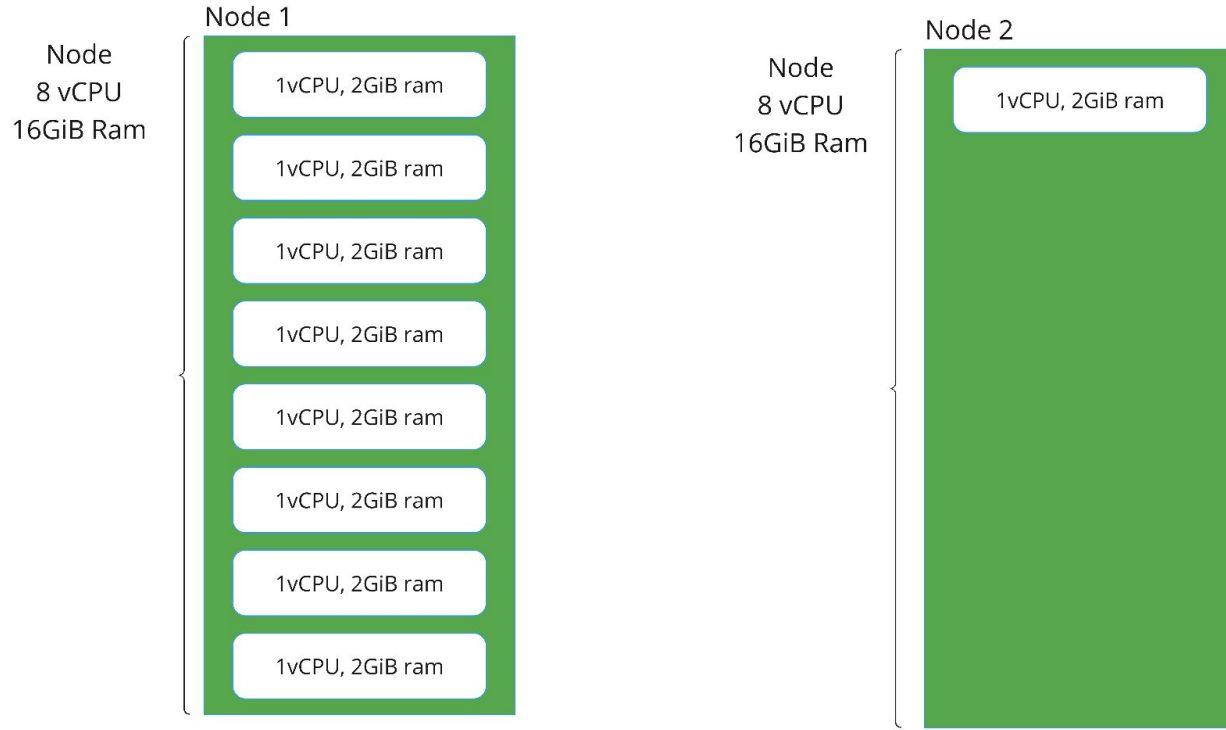
1vCPU, 2GiB ram

What drives k8s costs

Node
8 vCPU
16GiB Ram



What drives k8s costs



How to measure usage

- cpu|memory of nodes
- cpu|memory requests of workloads

How to measure usage



cpu|memory of nodes

`kube_node_status_capacity{cluster, resource, node}`



cpu|memory requests of workloads

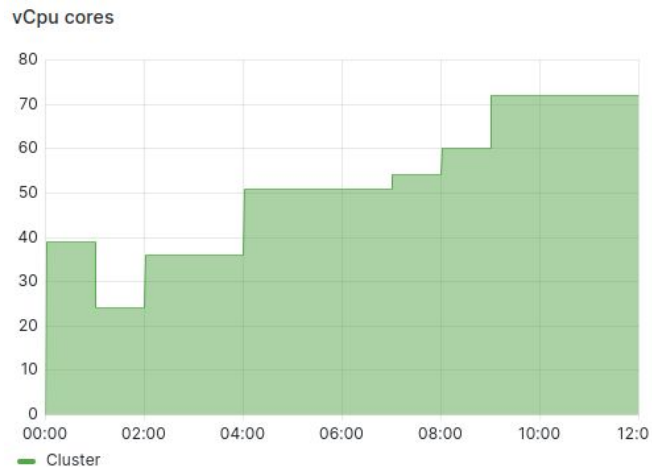
`kube_pod_container_resource_requests{cluster, resource, node, namespace}`

How to measure your nodes

```
sum (  
  usage  
  *  
  rate  
)
```

How to measure your nodes

```
sum (  
  kube_node_status_capacity{resource="cpu"}  
  *  
  rate  
)
```



How to measure your nodes

N2 machine types

South Carolina (us-east1) ▼

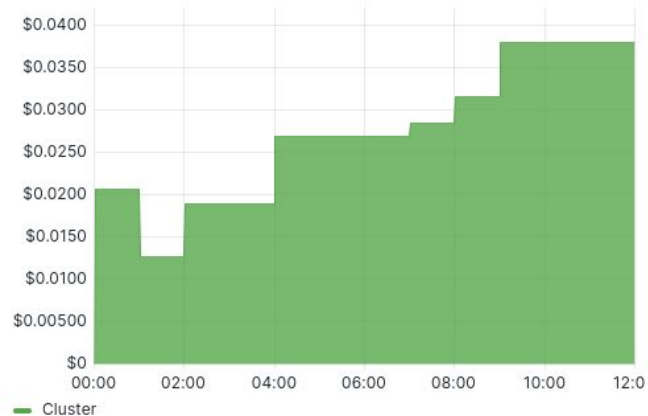
Hourly ☒ Monthly

Item	On-demand price (USD)	Spot price* (USD)
Predefined vCPUs	\$0.031611 / vCPU hour	\$0.00836 / vCPU hour
Predefined Memory	\$0.004237 / GB hour	\$0.00112 / GB hour

How to measure your nodes

```
sum (  
  kube_node_status_capacity{resource="cpu"}  
  *  
  (0.031611 / 60)  
)
```

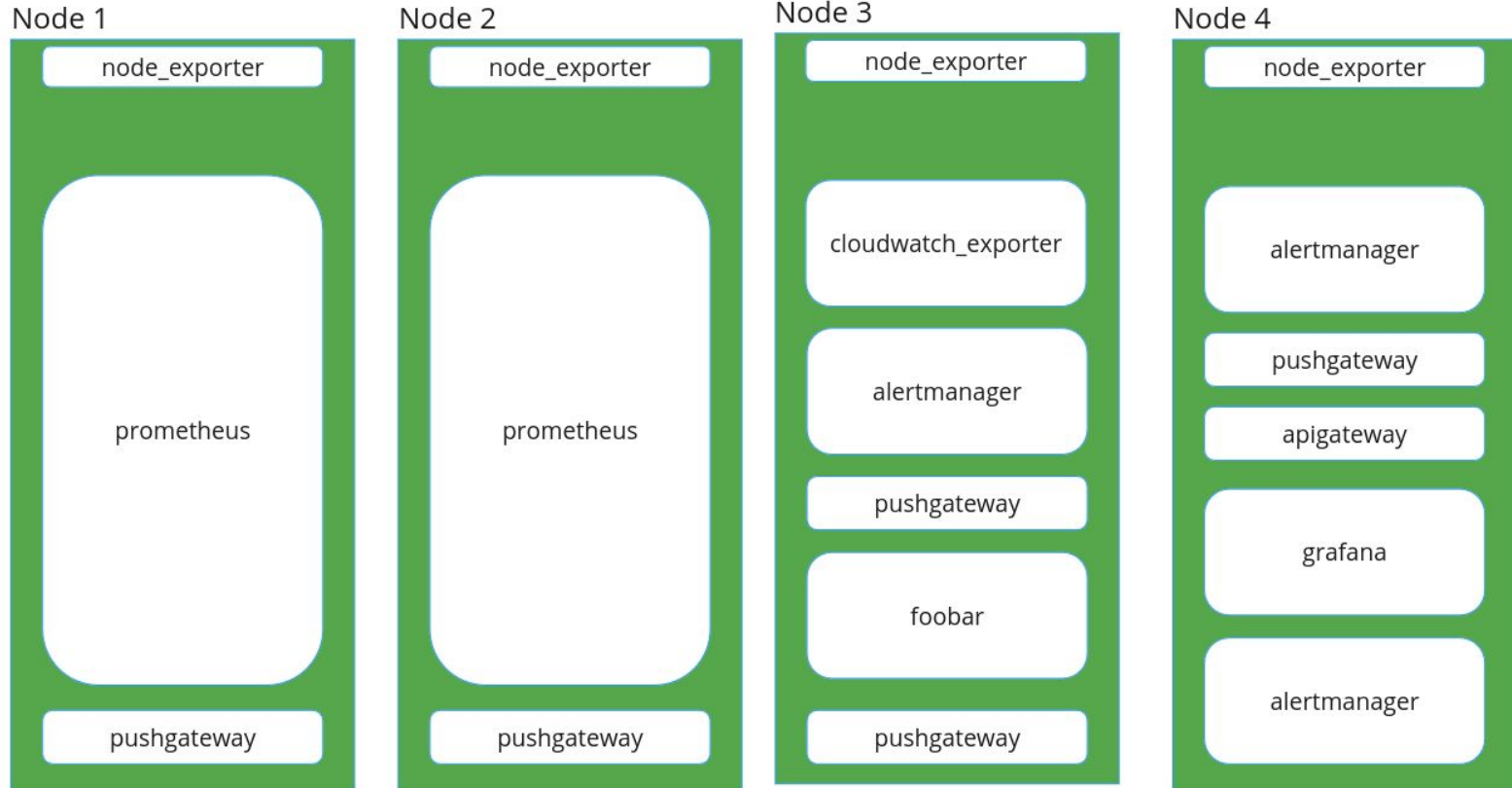
Cost in \$ per minute



How to measure your nodes

```
- record: cluster:cost_per_minute:sum
  expr: |
    sum by (cluster) (
      kube_node_status_capacity{resource="cpu"}
      *
      (0.031611 / 60)
    )
  labels:
    resource: "cpu"
```


What drives k8s costs (or who)



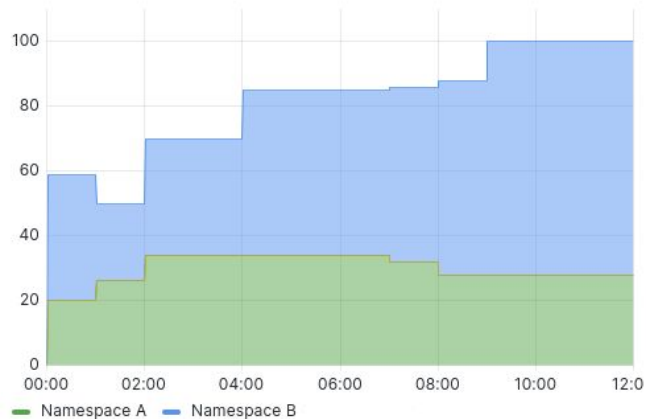
How to measure your workloads impact

```
sum by (namespace) (  
    requests  
    *  
    rate  
)
```

How to measure your workloads impact

```
sum by (namespace) (  
  kube_pod_container_resource_requests{resource="cpu"}  
  *  
  rate  
)
```

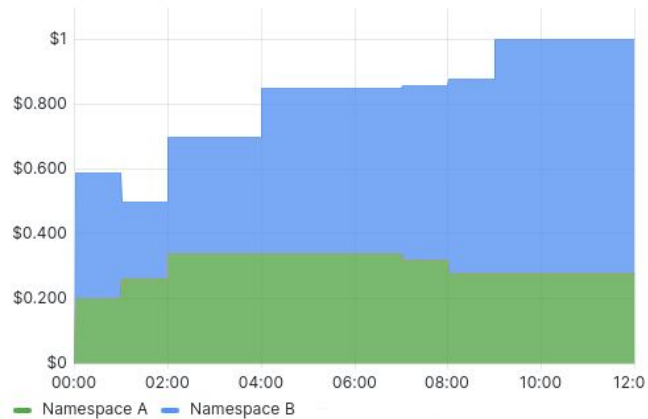
vCpu cores by namespace



How to measure your workloads impact

```
sum by (namespace) (  
  kube_pod_container_resource_requests{resource="cpu"}  
  *  
  (0.031611 / 60)  
)
```

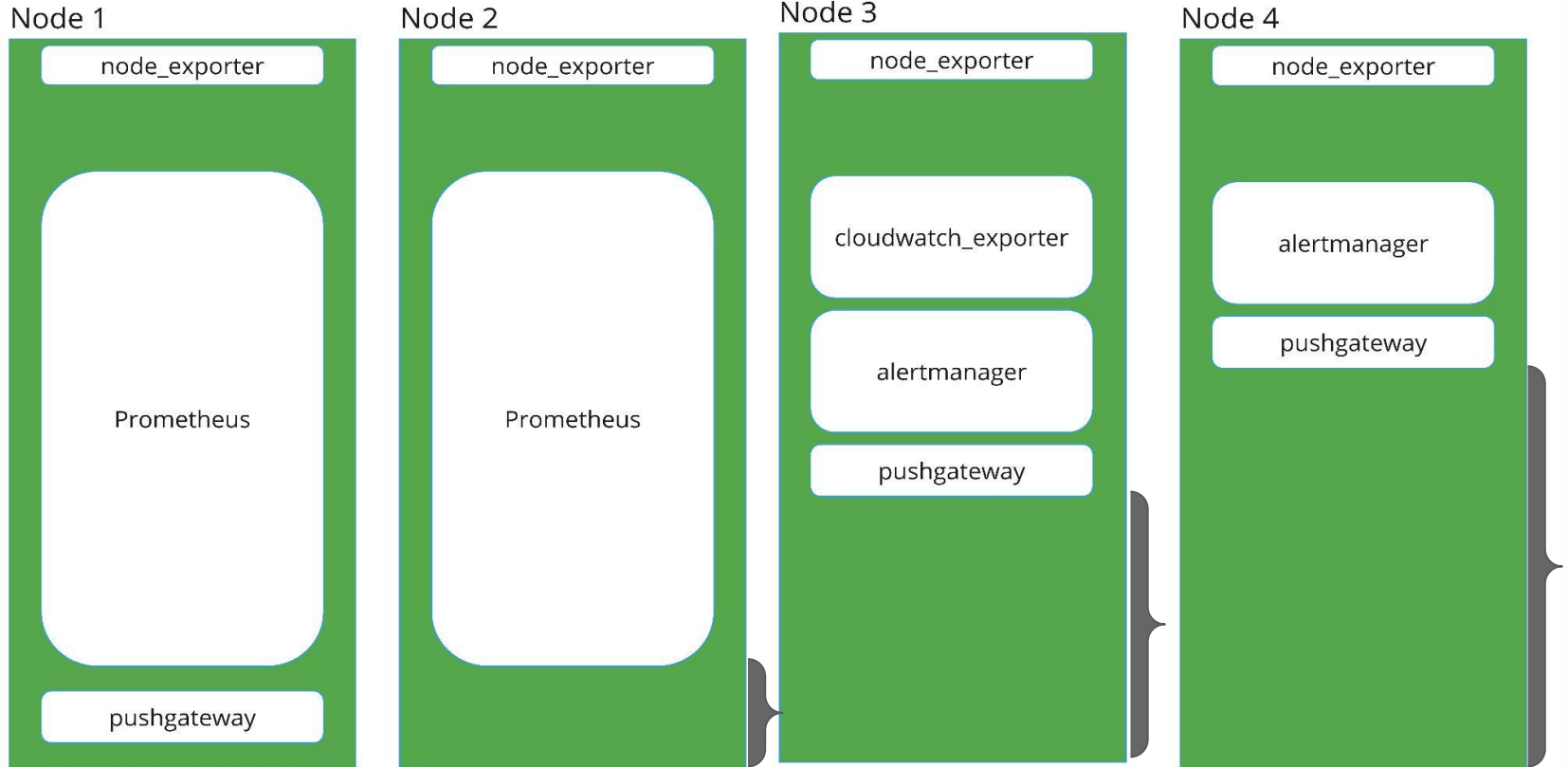
Cost in \$ per minute by namespace



How to measure your workloads impact

```
- record: cluster_namespace:cost_per_minute:sum
  expr: |
    sum by (cluster, namespace) (
      kube_pod_container_resource_requests{resource="cpu"}
      *
      (0.031611 / 60)
    )
  labels:
    resource: "cpu"
```

What drives k8s costs (realistic)



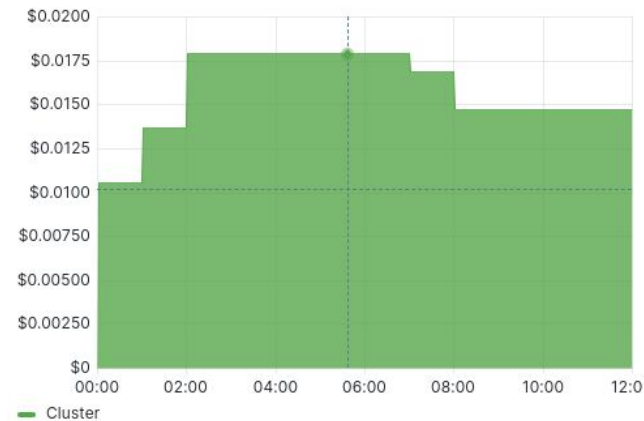
How to measure idle resources

```
sum (  
  (  
    capacity  
    -  
    requests  
  )  
  *  
  rate  
)
```

How to measure idle resources

```
sum (
  (
    sum by (node) (
      kube_node_status_capacity{resource="cpu"}
    )
    -
    sum by (node) (
      kube_pod_container_resource_requests{resource="cpu"}
    )
  )
  *
  (0.031611 / 60)
)
```

Idle cost in \$ per minute



How to measure idle resources

```
- record: cluster_namespace:cost_per_minute:sum
  expr: |
    sum by (cluster) (
      (
        sum by (cluster, node) (
          kube_node_status_capacity{resource="cpu"}
        )
        -
        sum by (cluster, node) (
          kube_pod_container_resource_requests{resource="cpu"}
        )
      )
      *
      (0.031611 / 60)
    )
  labels:
    resource: "cpu"
    namespace: "__idle__"
```

How to draw an Owl.

"A fun and creative guide for beginners"

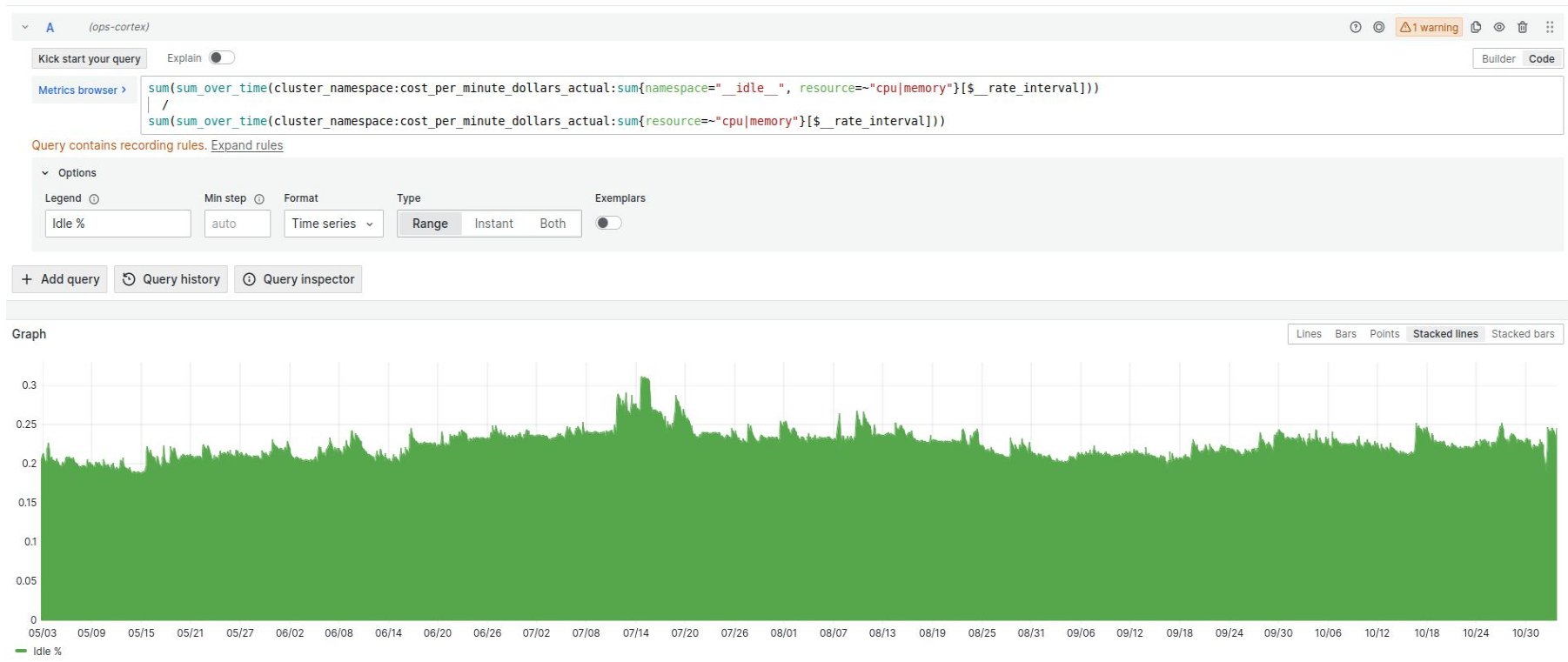


Fig 1. Draw two circles



Fig 2. Draw the rest of the damn Owl

How to measure idle resources



Lessons learned with Measuring Costs

- This approach only works for homogeneous clusters
- Takes only compute resources into account
- Not all CSPs will give you the breakdown on compute resources costs
- Namespaces is not a 1:1 mapping to teams

Lessons learned: Bin Packing



YOU DO HAVE

**CPU AND MEM
REQUESTS, RIGHT?**

Lessons learned: Bin Packing



In Google, using `OPTIMIZE_UTILIZATION` for `GKE`

Lessons learned: Bin Packing



In Google, using `OPTIMIZE_UTILIZATION` for `GKE`



Kubernetes `descheduler` helps improving node utilization

Lessons learned: Bin Packing

- In Google, using `OPTIMIZE_UTILIZATION` for `GKE`
- Kubernetes `descheduler` helps improving node utilization
- In AWS, using `Karpenter` as cluster autoscaler for `EKS`

Lessons learned: Bin Packing

- In Google, using `OPTIMIZE_UTILIZATION` for `GKE`
- Kubernetes `descheduler` helps improving node utilization
- In AWS, using `Karpenter*` as cluster autoscaler for `EKS`
- [Google's State of Kubernetes Cost Optimization](#)

<https://bit.ly/grafana-karpenter>



Thank you

- Erik Sommer
- JuanJo Ciarlante
- Mark Poko
- Paula Julve



Join us at
[https://slack.grafana.com/
#cost-observability](https://slack.grafana.com/#cost-observability)