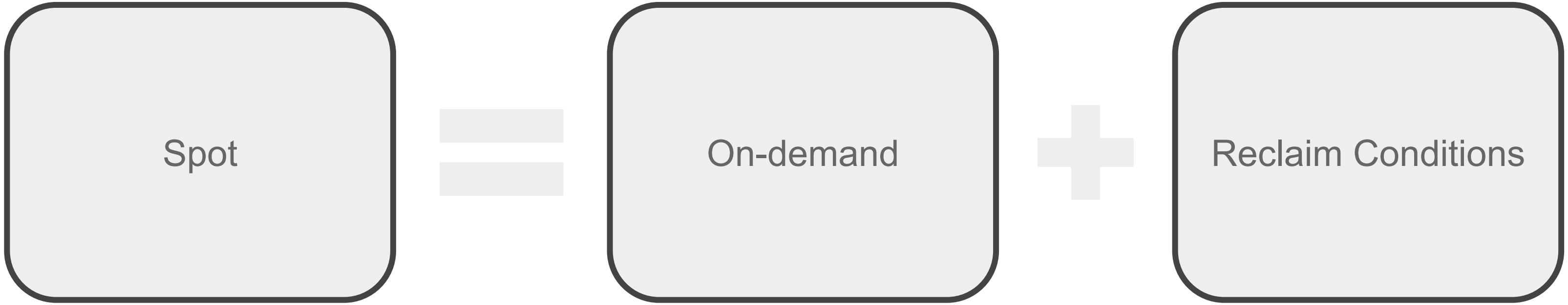
Spot Instances

Philipp Kühn @pkse as 1st killed soldier
Alby Hernández @achetronic as 2nd killed soldier

How we discovered they are not reliable

Spot Instances: on-demand vs. spot

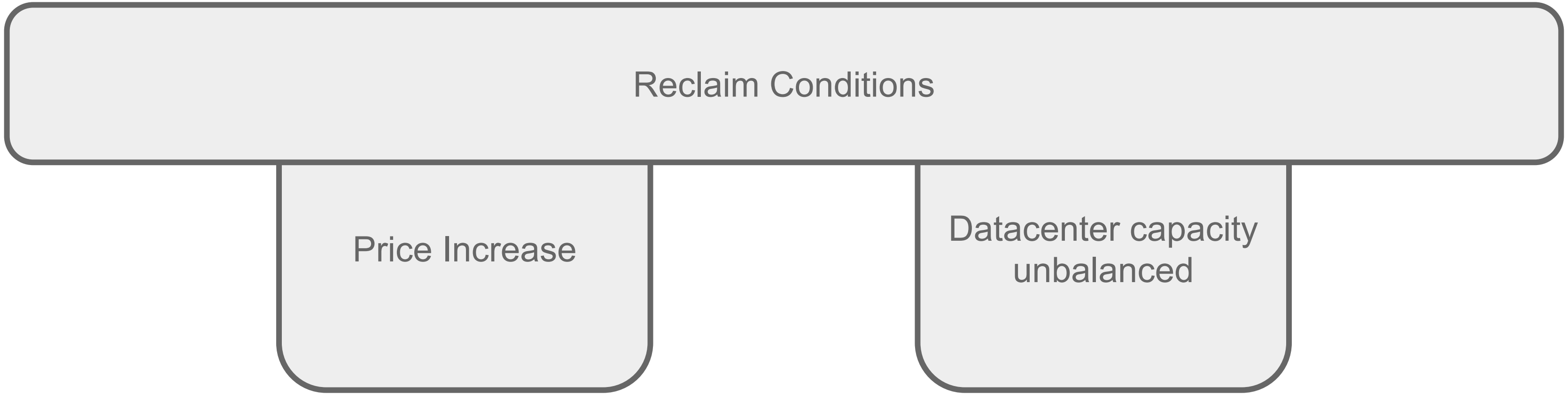


Spot = On-demand + Reclaim Conditions

Spot Instances: on-demand vs. spot

Reclaim Conditions = On some situations related to AWS, these instances can be destroyed

Spot Instances: reclaim conditions

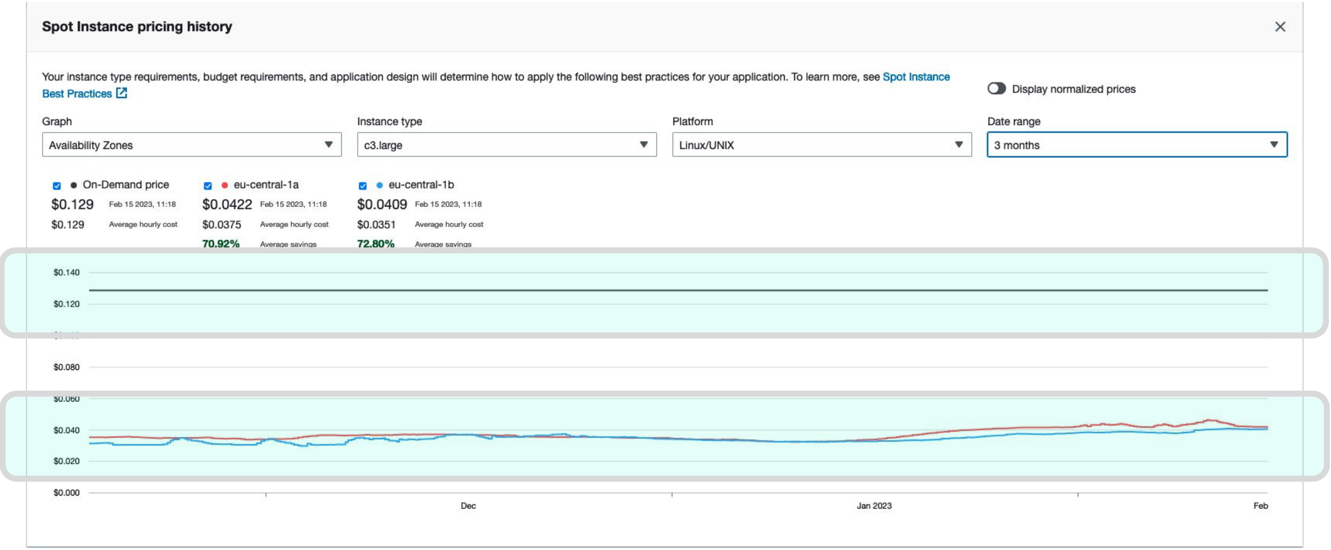


Reclaim Conditions

Price Increase

Datacenter capacity unbalanced

Spot Instances: Price increased



Spot Instance pricing history

Your instance type requirements, budget requirements, and application design will determine how to apply the following best practices for your application. To learn more, see Spot Instance Best Practices

○ Display normalized prices

Graph: Availability Zones

Instance type: c3.large

Platform: Linux/UNIX

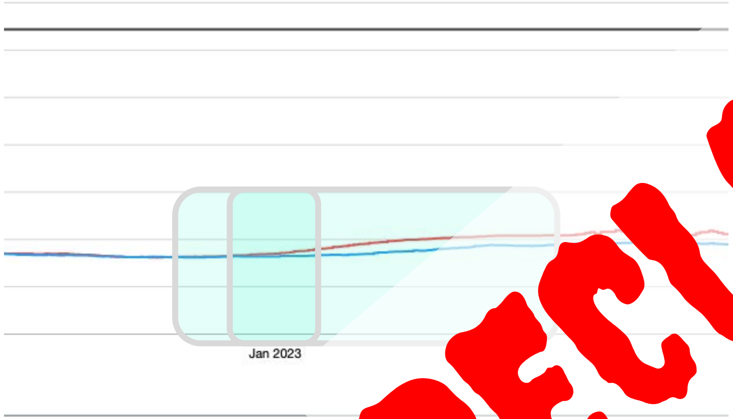
Date range: 3 months

☑ ● On-Demand price
$0.129 Feb 15 2023, 11:18
$0.129 Average hourly cost

☑ ● eu-central-1a
$0.0422 Feb 15 2023, 11:18
$0.0375 Average hourly cost
70.92% Average savings

☑ ● eu-central-1b
$0.0409 Feb 15 2023, 11:18
$0.0351 Average hourly cost
72.80% Average savings

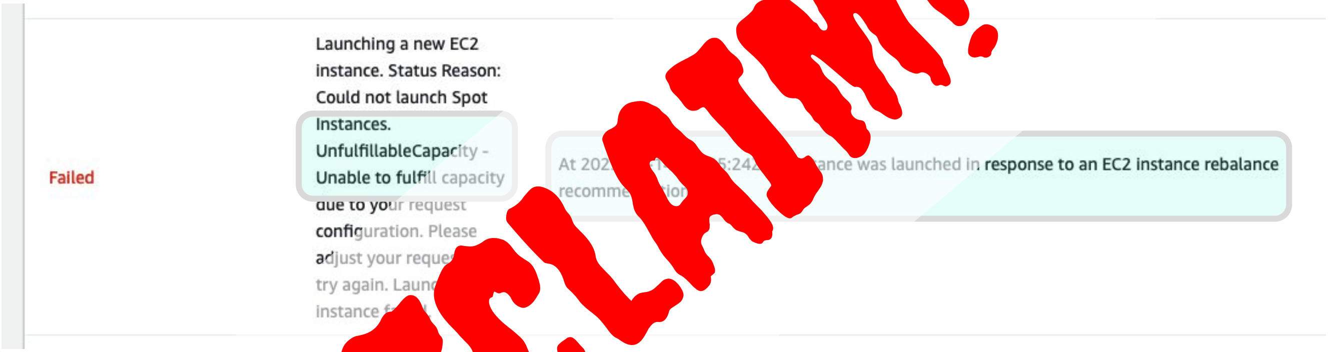
Spot Instances: reclaim conditions



RECLAIM!

Rented with old price?
Now we loose money

Spot Instances: unbalanced data center



Failed

Launching a new EC2 instance. Status Reason: Could not launch Spot Instances. UnfulfillableCapacity - Unable to fulfill capacity due to your request configuration. Please adjust your reque try again. Launc instance f l.

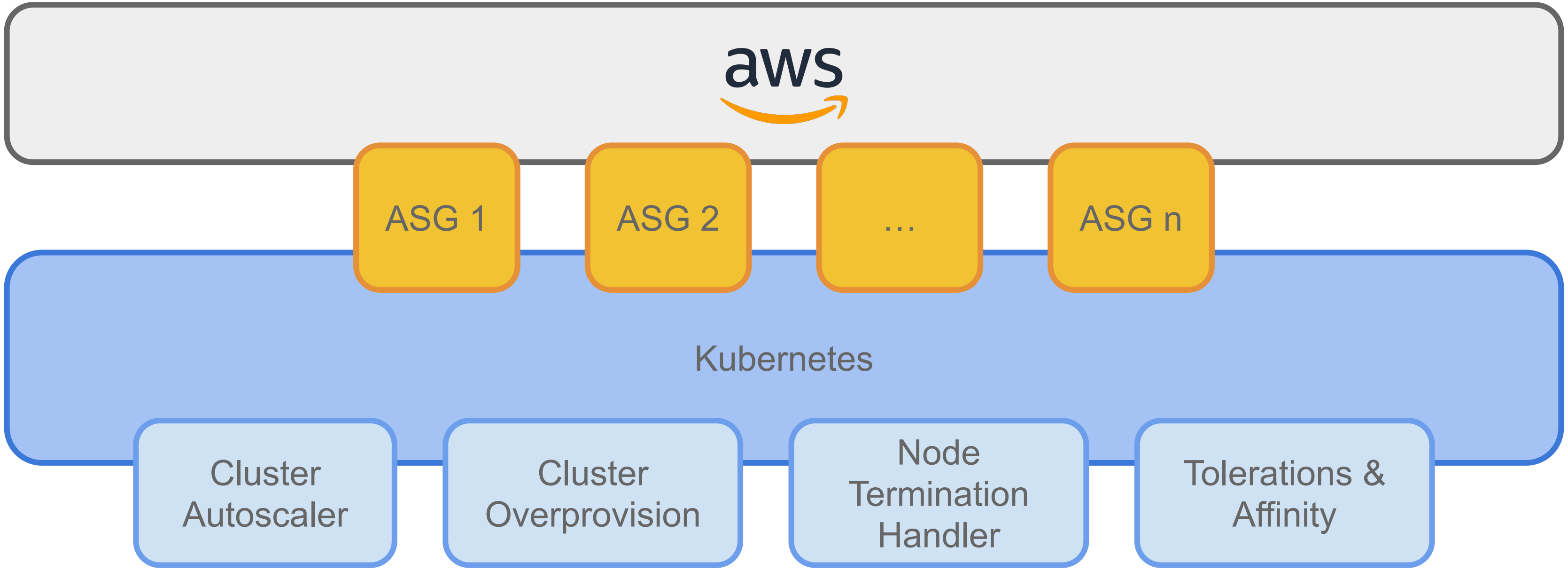
At 202 1 5:242 ance was launched in response to an EC2 instance rebalance recomme tion

RECLAIM!

aws forgot to pre-balance data center’s VMs

The Pieces

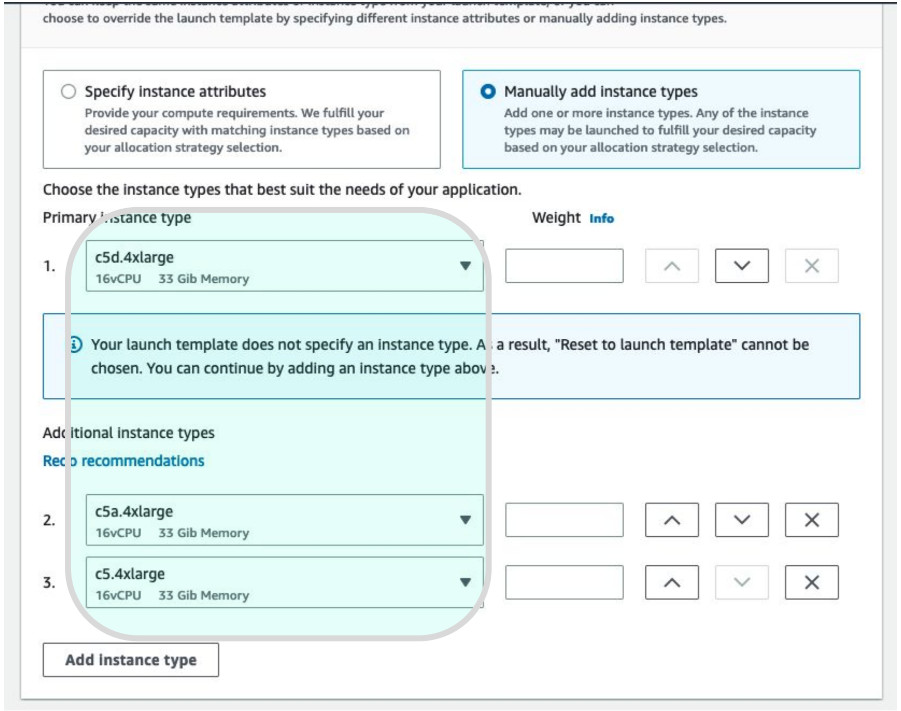
Overview


aws
ASG 1
ASG 2
...
ASG n
Kubernetes
Cluster Autoscaler
Cluster Overprovision
Node Termination Handler
Tolerations & Affinity

The Pieces

Auto Scaling Groups

Spot Auto Scaling Group: Instance types



choose to override the launch template by specifying different instance attributes or manually adding instance types.

○ Specify instance attributes
Provide your compute requirements. We fulfill your desired capacity with matching instance types based on your allocation strategy selection.

● Manually add instance types
Add one or more instance types. Any of the instance types may be launched to fulfill your desired capacity based on your allocation strategy selection.

Choose the instance types that best suit the needs of your application.

Primary instance type — Weight Info

1. c5d.4xlarge — 16vCPU 33 Gib Memory

Your launch template does not specify an instance type. As a result, "Reset to launch template" cannot be chosen. You can continue by adding an instance type above.

Additional instance types

Reco recommendations

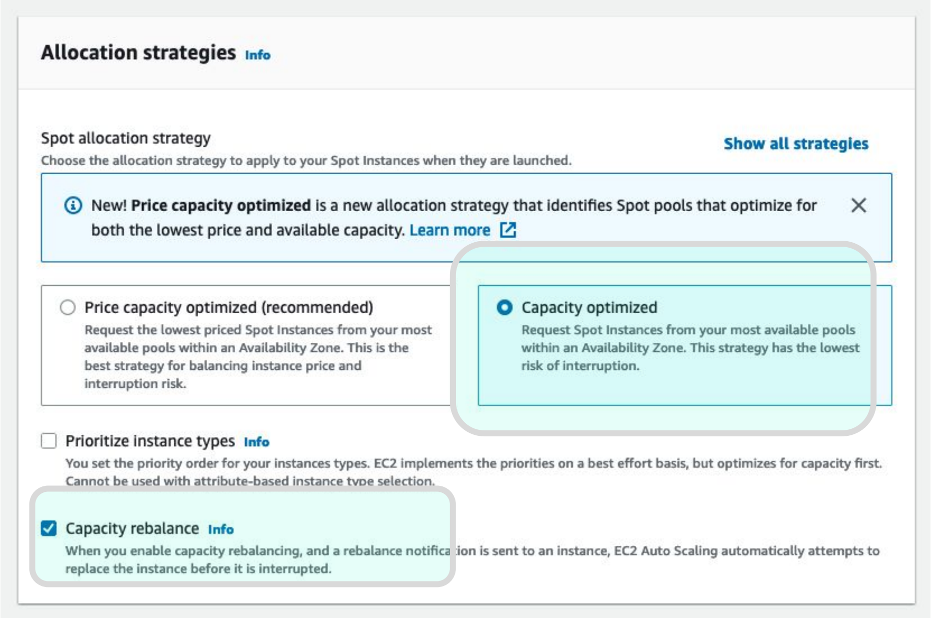
2. c5a.4xlarge — 16vCPU 33 Gib Memory

3. c5.4xlarge — 16vCPU 33 Gib Memory

Add instance type

- As many as possible
- Same CPU & Memory

Spot Auto Scaling Group: decrease the risk



Allocation strategies Info

Spot allocation strategy

Choose the allocation strategy to apply to your Spot Instances when they are launched.

Show all strategies

New! **Price capacity optimized** is a new allocation strategy that identifies Spot pools that optimize for both the lowest price and available capacity. Learn more

○ Price capacity optimized (recommended)

Request the lowest priced Spot Instances from your most available pools within an Availability Zone. This is the best strategy for balancing instance price and interruption risk.

● Capacity optimized

Request Spot Instances from your most available pools within an Availability Zone. This strategy has the lowest risk of interruption.

□ Prioritize instance types Info

You set the priority order for your instances types. EC2 implements the priorities on a best effort basis, but optimizes for capacity first. Cannot be used with attribute-based instance type selection.

☑ Capacity rebalance Info

When you enable capacity rebalancing, and a rebalance notification is sent to an instance, EC2 Auto Scaling automatically attempts to replace the instance before it is interrupted.

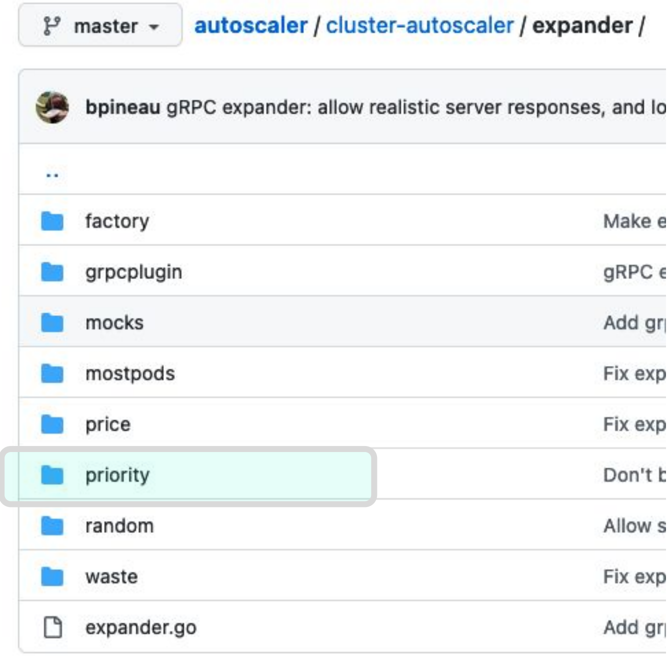
- Proper strategy
- Allow AWS create before destroy

Docplanner

The Pieces

Cluster Autoscaler

Cluster Autoscaler: Priority Expander



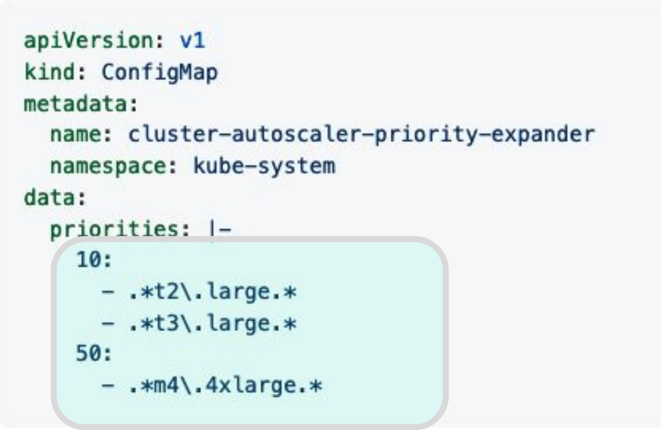
master ▾ autoscaler / cluster-autoscaler / expander /

bpineau gRPC expander: allow realistic server responses, and lo

..	
factory	Make e
grpcplugin	gRPC e
mocks	Add gr
mostpods	Fix exp
price	Fix exp
priority	Don't b
random	Allow s
waste	Fix exp
expander.go	Add gr

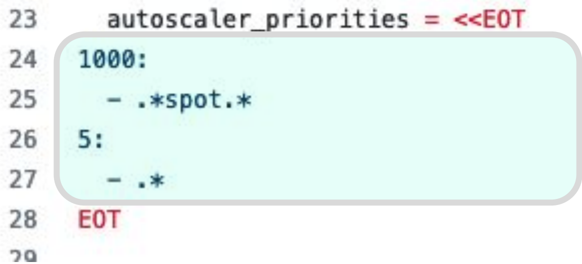
- Support Go regexp for matching ASGs
- Provide more control over ASGs we scale

Cluster Autoscaler: Priority Expander



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander
  namespace: kube-system
data:
  priorities: |-
    10:
      - .*t2\.large.*
      - .*t3\.large.*
    50:
      - .*m4\.4xlarge.*
```

Example on docs



```
  autoscaler_priorities = <<EOT
1000:
  - .*spot.*
5:
  - .*
EOT
```

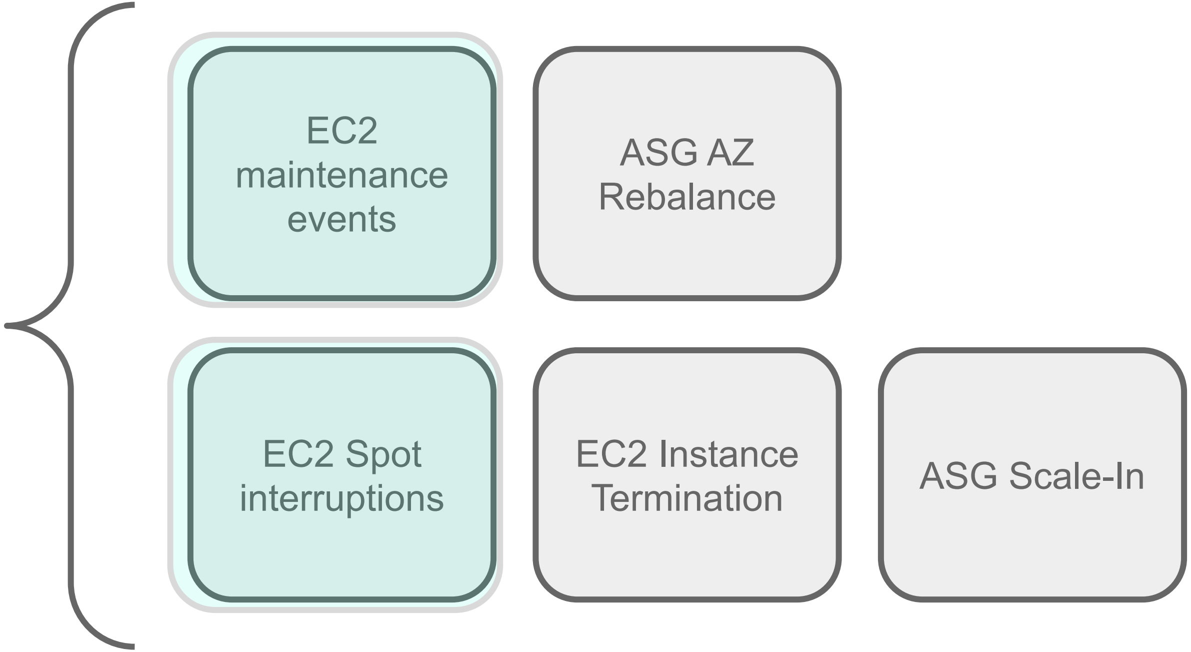
Our configuration

The Pieces

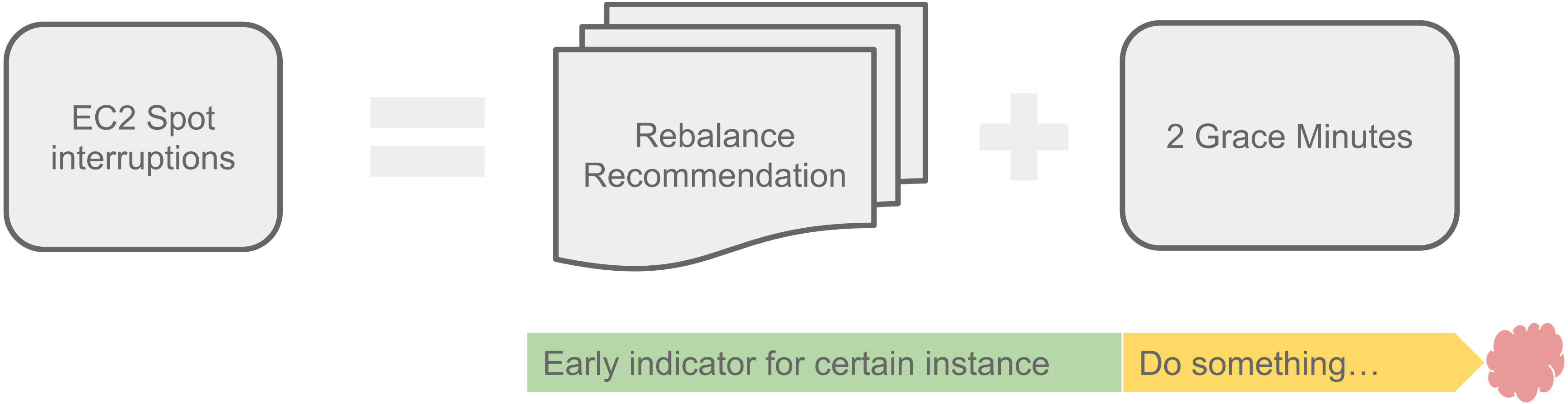
Node Termination Handler

Node Termination Handler: What?

Able to Cordon & drain on



Node Termination Handler: Why required for spots?



EC2 Spot interruptions

Rebalance Recommendation

2 Grace Minutes

Early indicator for certain instance

Do something…

Node Termination Handler: Why required for spots?

EC2 maintenance events

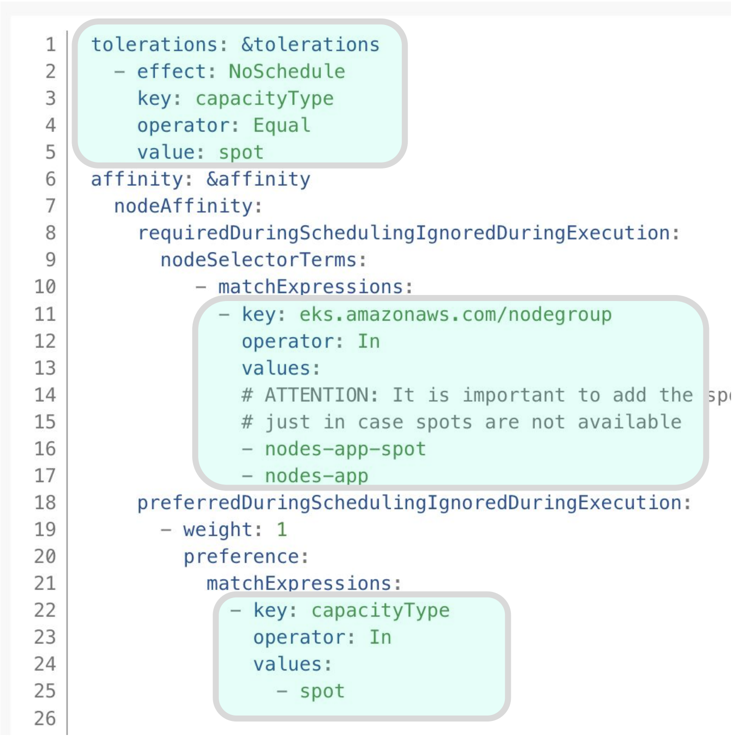
=

No more emails about rebooting some instances

The Pieces

Tolerations & Affinity

Tolerations & Affinity: Let Kubernetes be kubernetes



```yaml
tolerations: &tolerations
  - effect: NoSchedule
    key: capacityType
    operator: Equal
    value: spot
affinity: &affinity
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/nodegroup
            operator: In
            values:
            # ATTENTION: It is important to add the sp
            # just in case spots are not available
            - nodes-app-spot
            - nodes-app
    preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
           - key: capacityType
             operator: In
             values:
               - spot

```

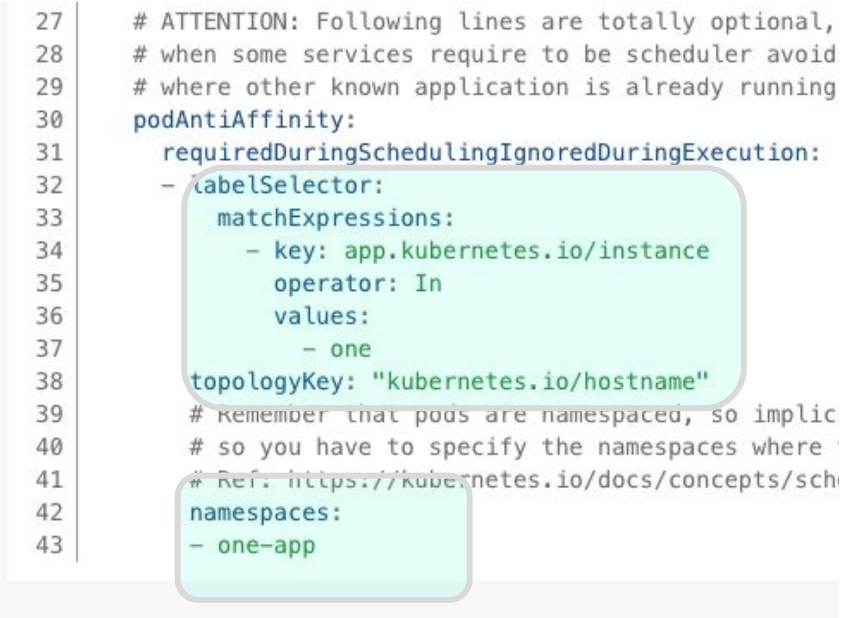
Hey, Kubernetes!
I can handle spot tainted nodes

Hey, Kubernetes!
I require nodes with these labels

Hey, Kubernetes!
I prefer spot

Docplanner

Tolerations & Affinity: Let Kubernetes be kubernetes



```yaml
# ATTENTION: Following lines are totally optional,
# when some services require to be scheduler avoid
# where other known application is already running
podAntiAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
  - labelSelector:
      matchExpressions:
        - key: app.kubernetes.io/instance
          operator: In
          values:
            - one
    topologyKey: "kubernetes.io/hostname"
    # Remember that pods are namespaced, so implic
    # so you have to specify the namespaces where
    # Ref: https://kubernetes.io/docs/concepts/sch
    namespaces:
    - one-app
```

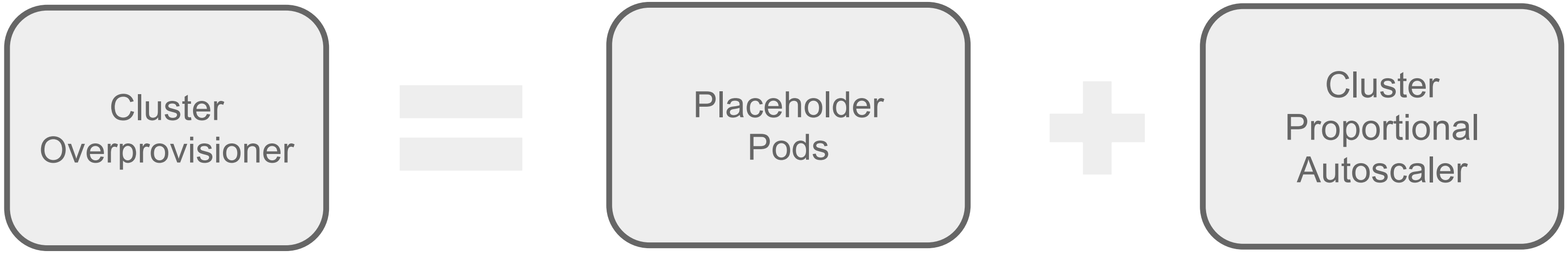
Hey, Kubernetes!
I hate Saas, I wanna be allocated in other place

Remember labels belong to pods, so they are namespaced too

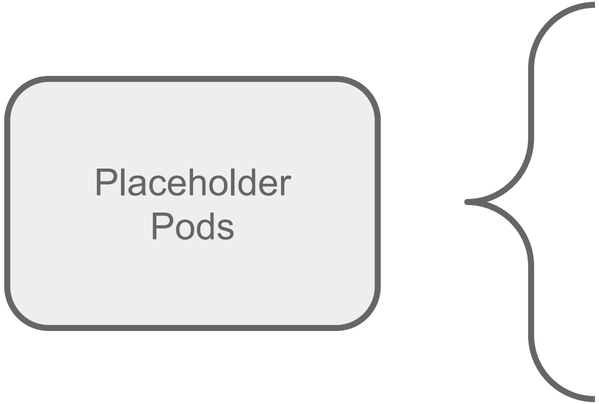
The Pieces

Cluster Overprovisioner

C. Overprovisioner: Components



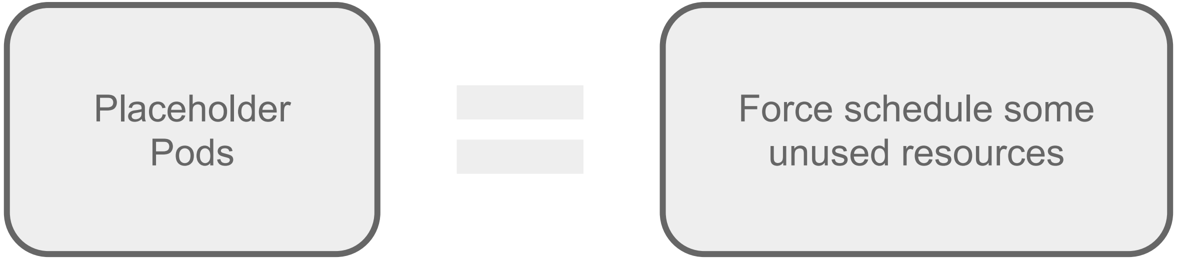
C. Overprovisioner: Placeholder



Placeholder Pods

- Use pause Docker image
- Requests resources
- Lowest PriorityClass

C. Overprovisioner: Placeholder

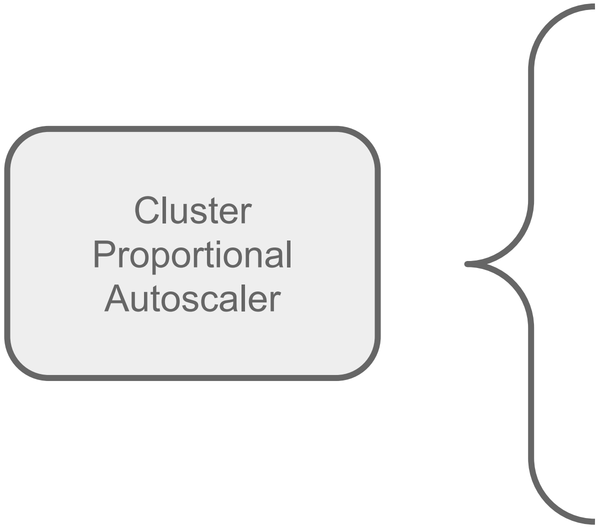


Placeholder Pods

=

Force schedule some unused resources

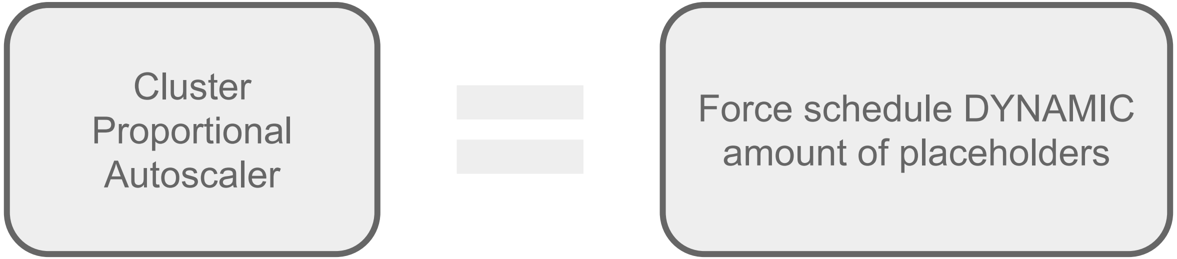
C. Overprovisioner: Cluster Proportional Autoscaler



Cluster Proportional Autoscaler

- Watches a deployment
- Count number of certain labeled nodes
- Scale paused pod replicas

C. Overprovisioner: Cluster Proportional Autoscaler

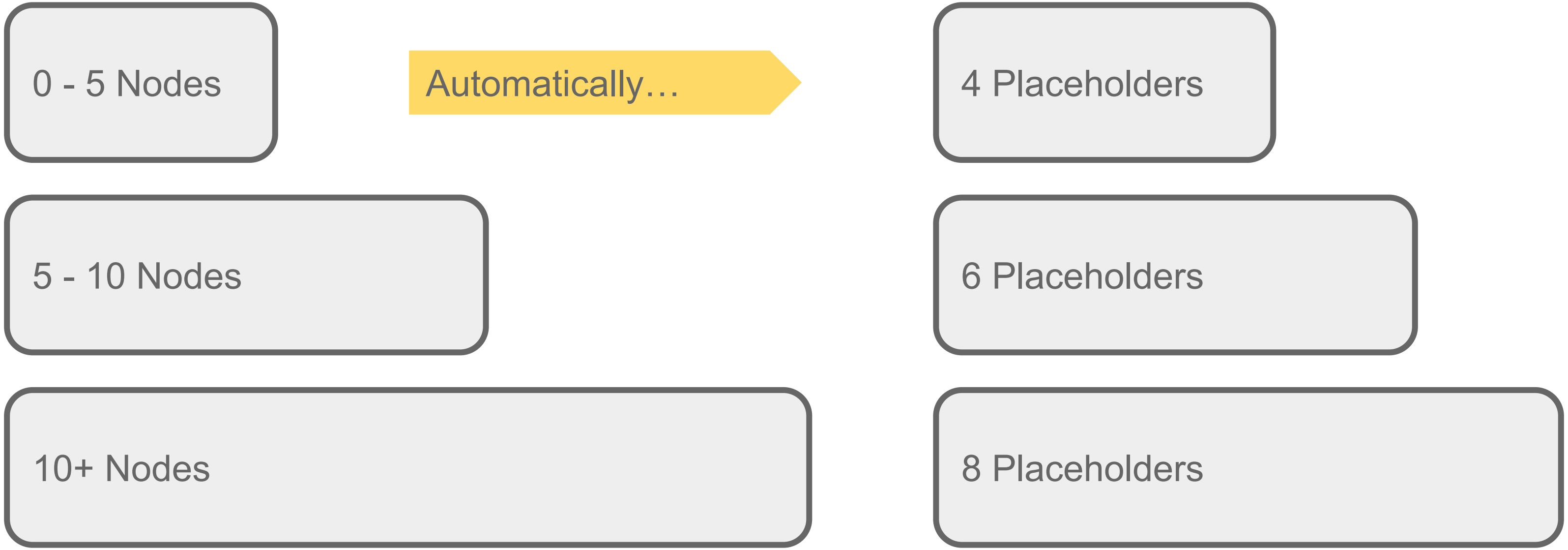


Cluster Proportional Autoscaler

=

Force schedule DYNAMIC amount of placeholders

C. Overprovisioner: CPA. Our config



0 - 5 Nodes

Automatically…

4 Placeholders

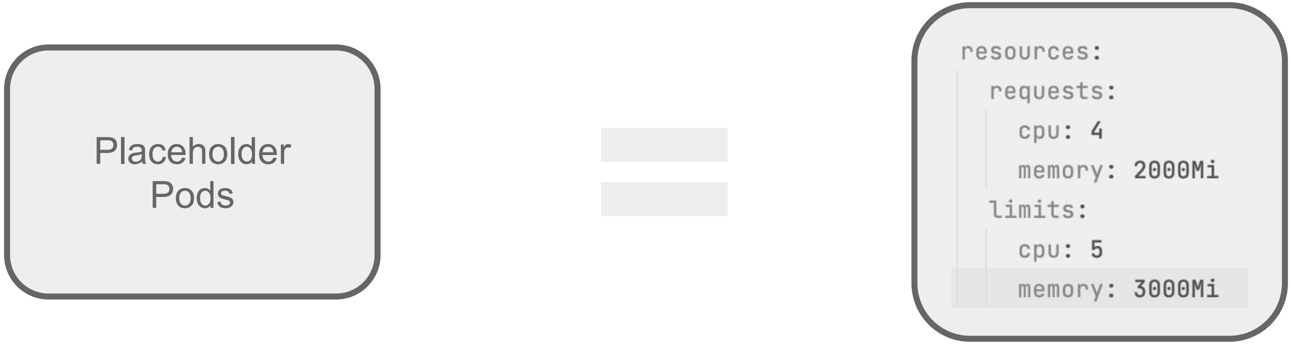
5 - 10 Nodes

6 Placeholders

10+ Nodes

8 Placeholders

C. Overprovisioner: CPA. Our config



Placeholder Pods

```
resources:
  requests:
    cpu: 4
    memory: 2000Mi
  limits:
    cpu: 5
    memory: 3000Mi
```

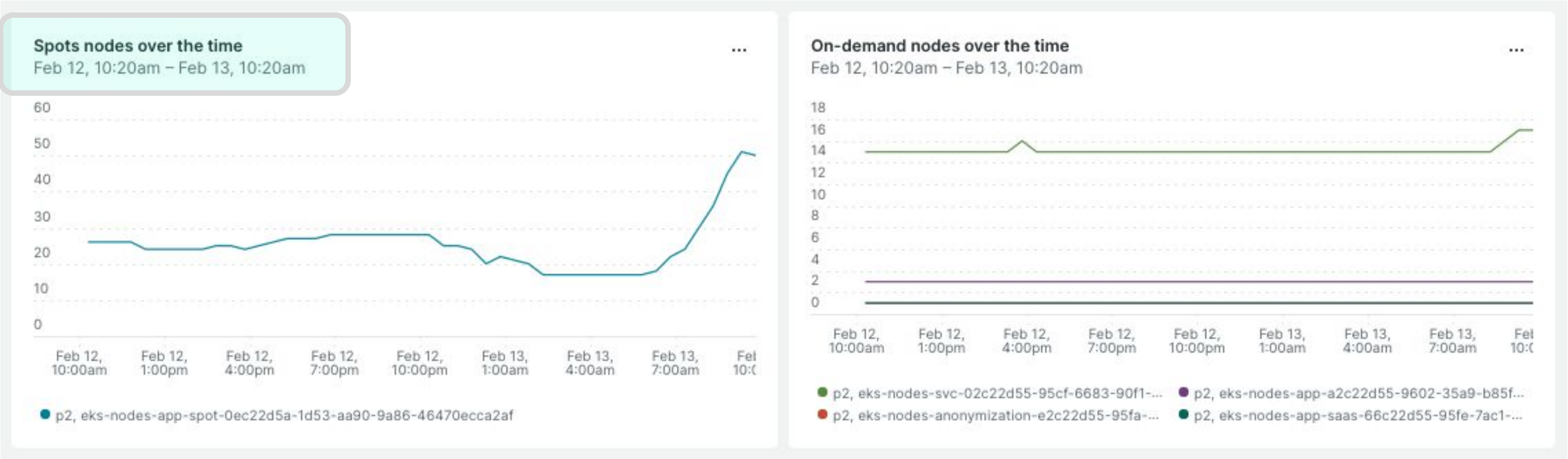
More than a Monolith

Initial Results

Round 1

Docplanner

Results: Everything started to work



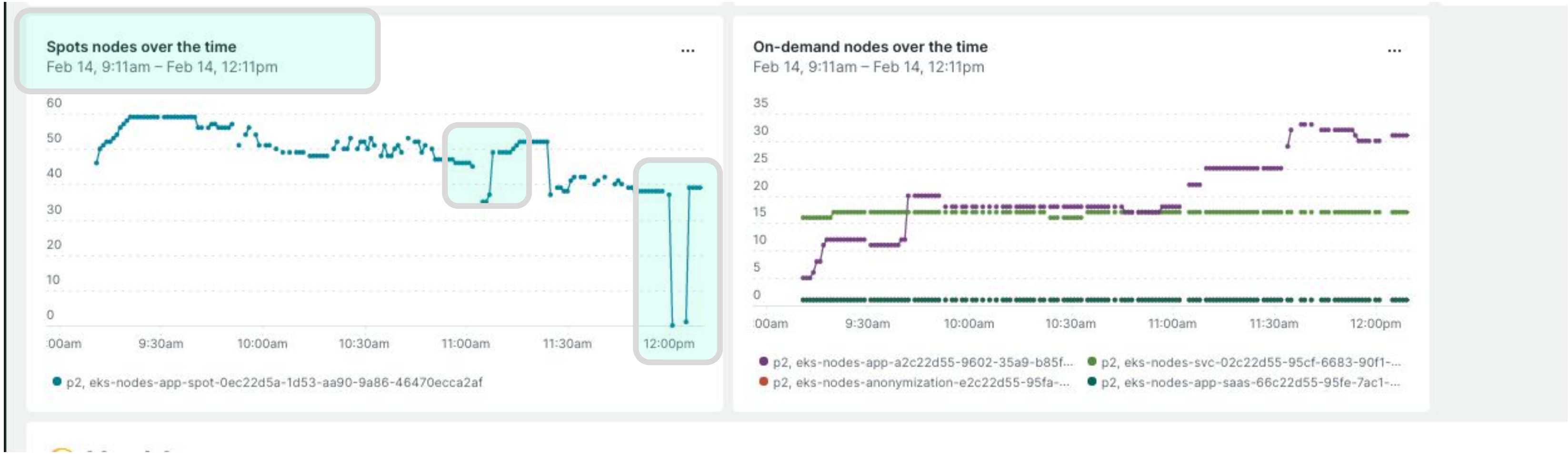
Spots nodes over the time

Feb 12, 10:20am – Feb 13, 10:20am

On-demand nodes over the time

Feb 12, 10:20am – Feb 13, 10:20am

Results: Everything broke



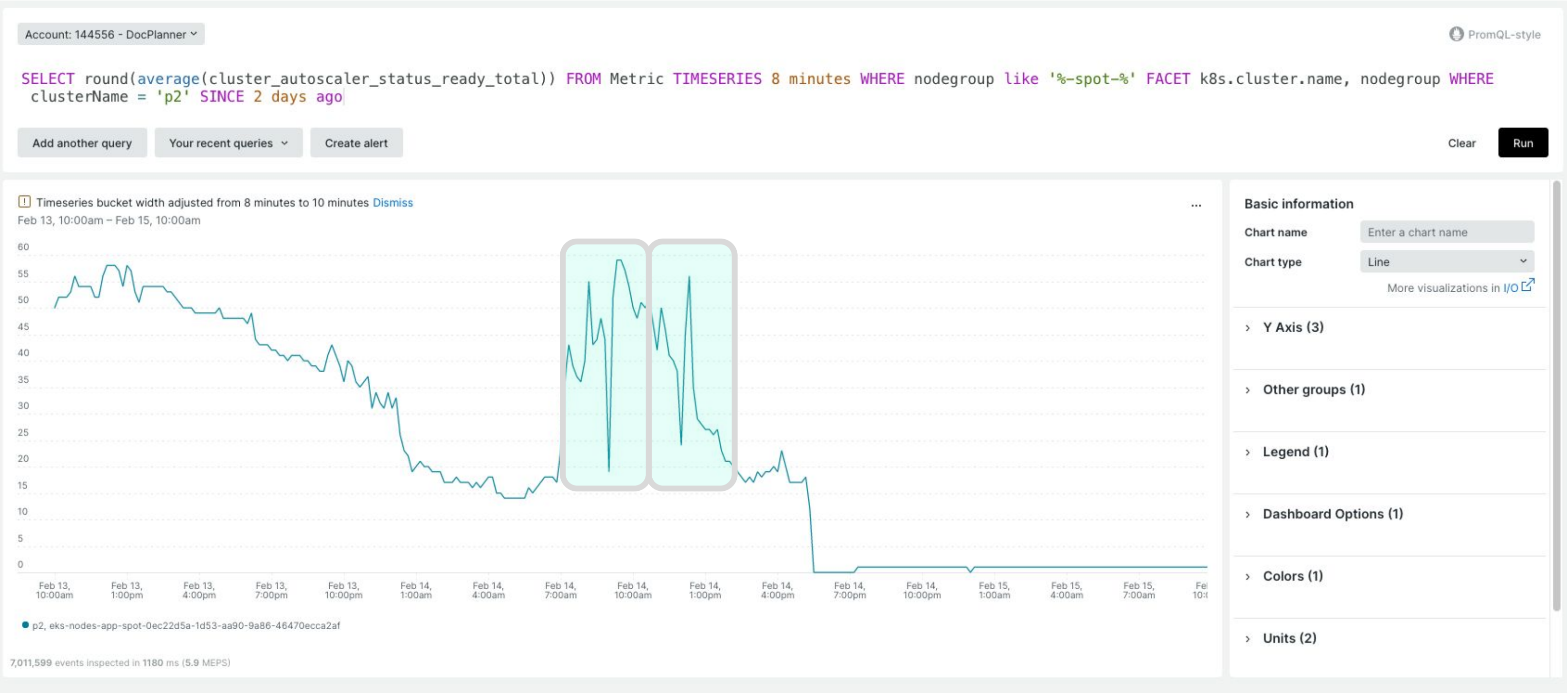
Spots nodes over the time
Feb 14, 9:11am – Feb 14, 12:11pm

...
60
50
40
30
20
10
0
00am
9:30am
10:00am
10:30am
11:00am
11:30am
12:00pm
● p2, eks-nodes-app-spot-0ec22d5a-1d53-aa90-9a86-46470ecca2af

On-demand nodes over the time
Feb 14, 9:11am – Feb 14, 12:11pm

...
35
30
25
20
15
10
5
0
00am
9:30am
10:00am
10:30am
11:00am
11:30am
12:00pm
● p2, eks-nodes-app-a2c22d55-9602-35a9-b85f...
● p2, eks-nodes-svc-02c22d55-95cf-6683-90f1-...
● p2, eks-nodes-anonymization-e2c22d55-95fa-...
● p2, eks-nodes-app-saas-66c22d55-95fe-7ac1-...

Results: Everything broke



Account: 144556 - DocPlanner

PromQL-style

```
SELECT round(average(cluster_autoscaler_status_ready_total)) FROM Metric TIMESERIES 8 minutes WHERE nodegroup like '%-spot-%' FACET k8s.cluster.name, nodegroup WHERE
 clusterName = 'p2' SINCE 2 days ago
```

Timeseries bucket width adjusted from 8 minutes to 10 minutes Dismiss

Feb 13, 10:00am – Feb 15, 10:00am

● p2, eks-nodes-app-spot-0ec22d5a-1d53-aa90-9a86-46470ecca2af

7,011,599 events inspected in 1180 ms (5.9 MEPS)

Basic information

Chart name: Enter a chart name

Chart type: Line

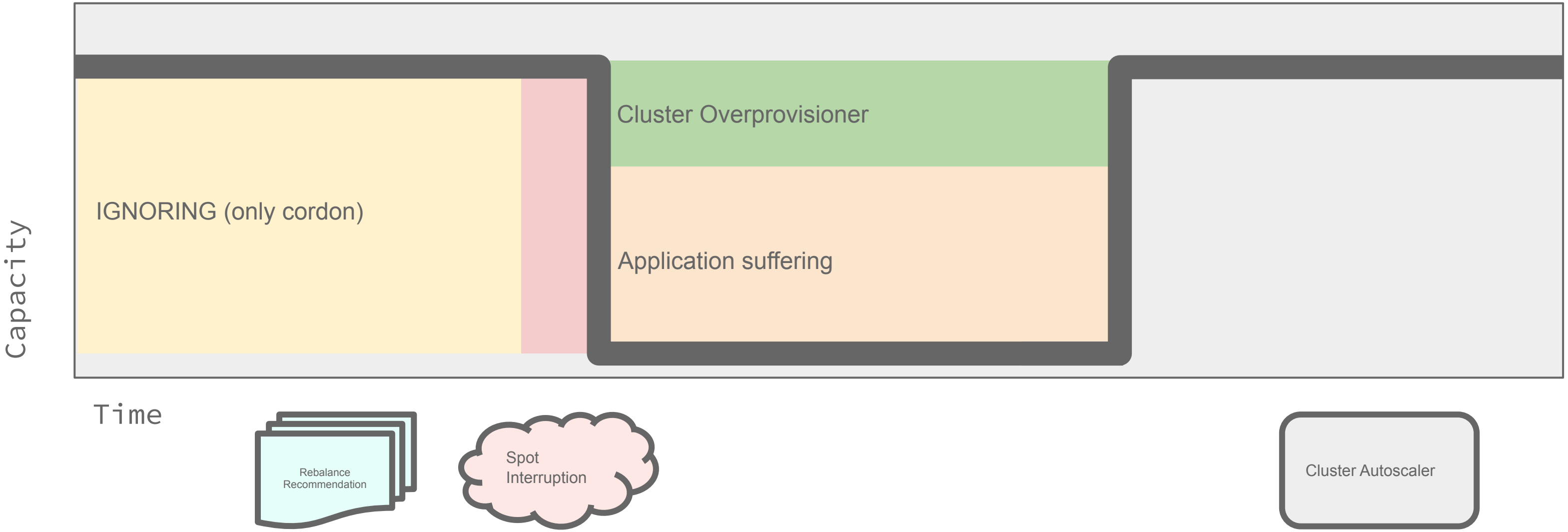
Break (10 minutes)

Go for a coffee *cough cough w.c*

Potential solution

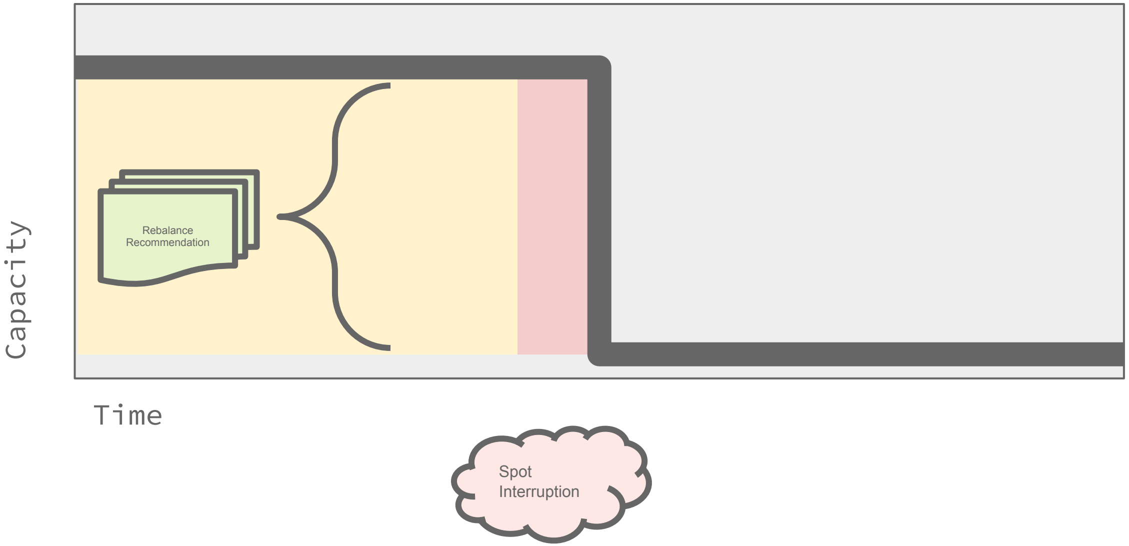
Round 2

Possible solutions: The problem



Docplanner

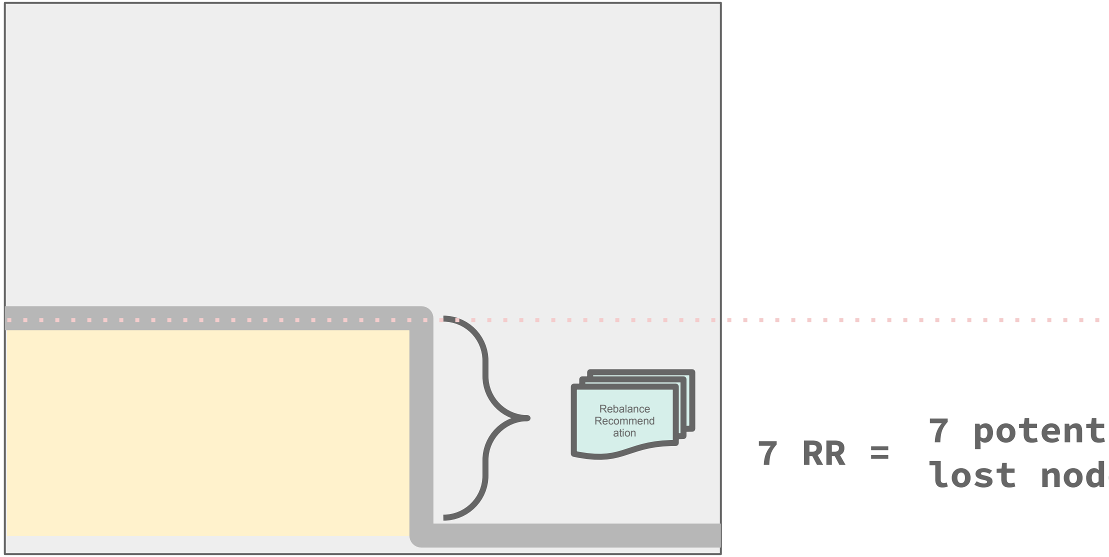
Possible solutions: The problem



Docplanner

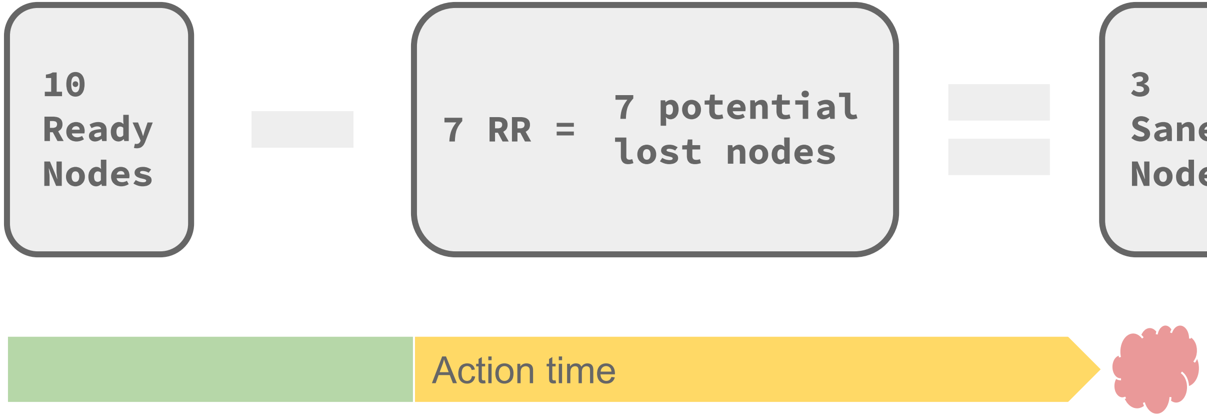
Possible solutions: Dig Deeper

10
Ready
Nodes



7 RR = 7 potential
lost nodes

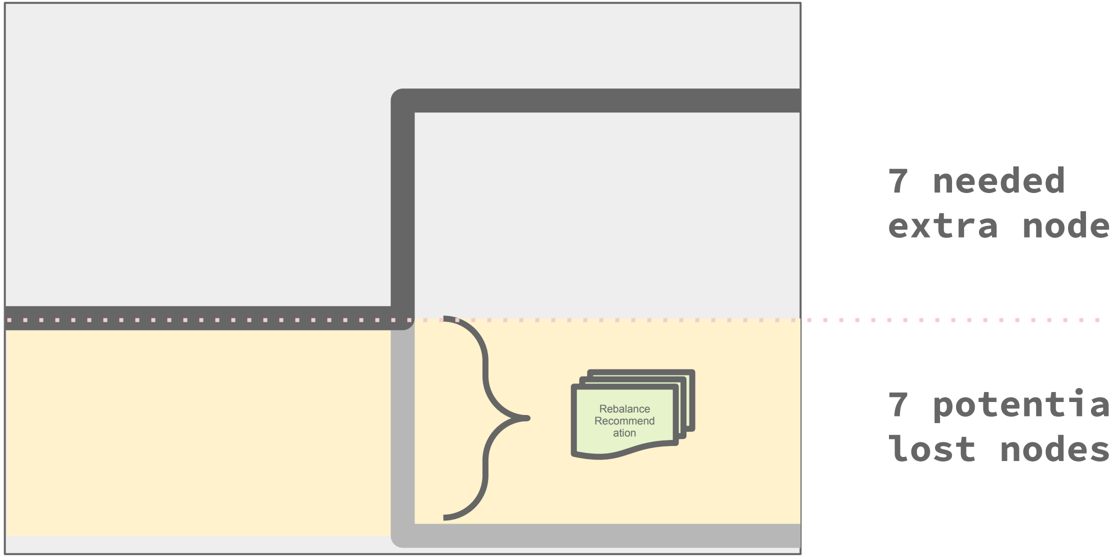
Possible solutions: Dig Deeper



10 Ready Nodes − 7 RR = 7 potential lost nodes = 3 Sane Nodes

Possible solutions: Dig Deeper

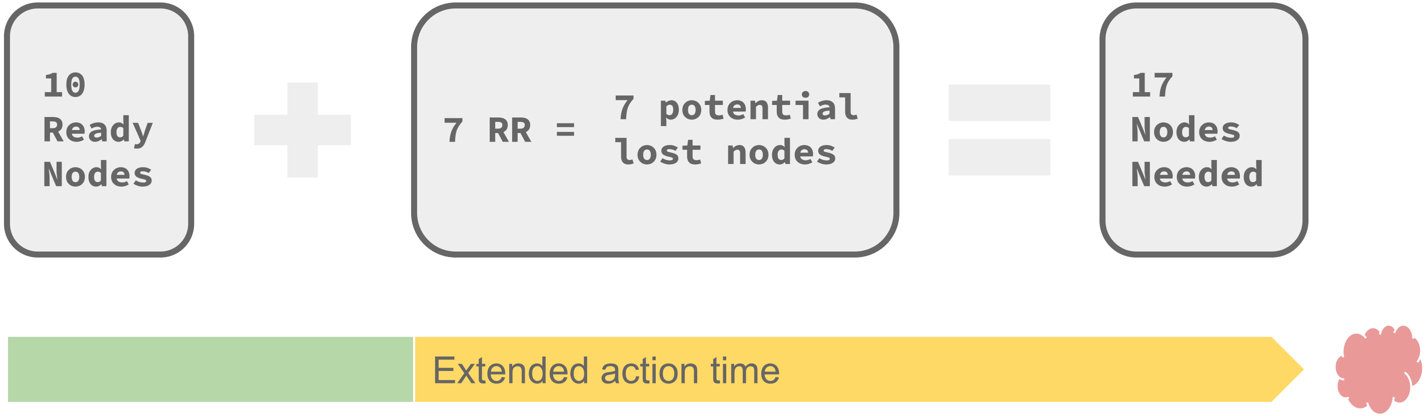
10 Ready Nodes



7 needed extra nodes

7 potential lost nodes

Possible solutions: Solution



10 Ready Nodes

+

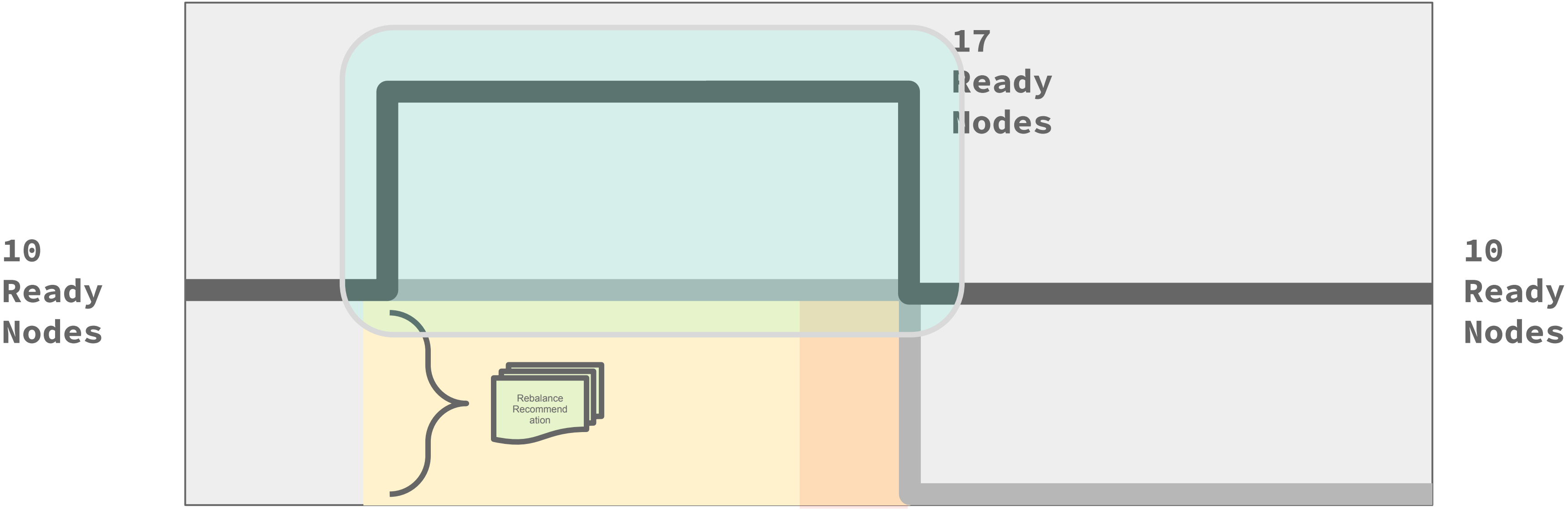
7 RR = 7 potential lost nodes

=

17 Nodes Needed

Extended action time

Possible solutions: Solution

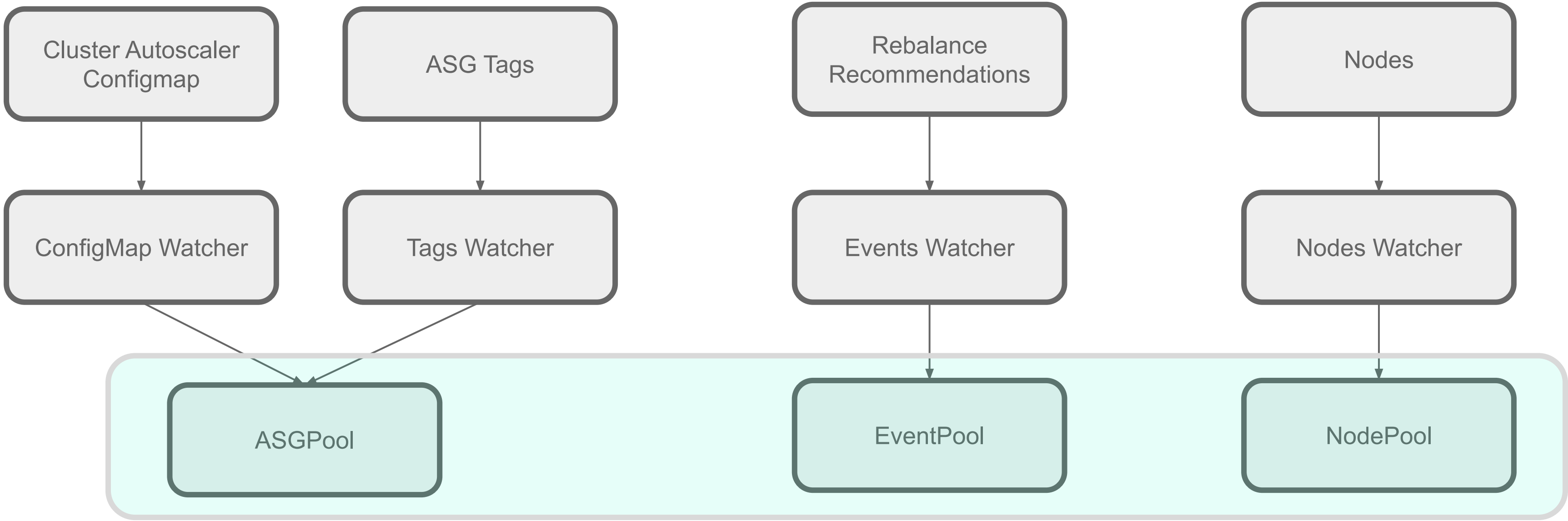


Possible solutions: How to calculate a boost

Resources Calculations

Rebalance Recommendation

Possible solutions: Calculations. Not that easy



Cluster Autoscaler Configmap

ASG Tags

Rebalance Recommendations

Nodes

ConfigMap Watcher

Tags Watcher

Events Watcher

Nodes Watcher

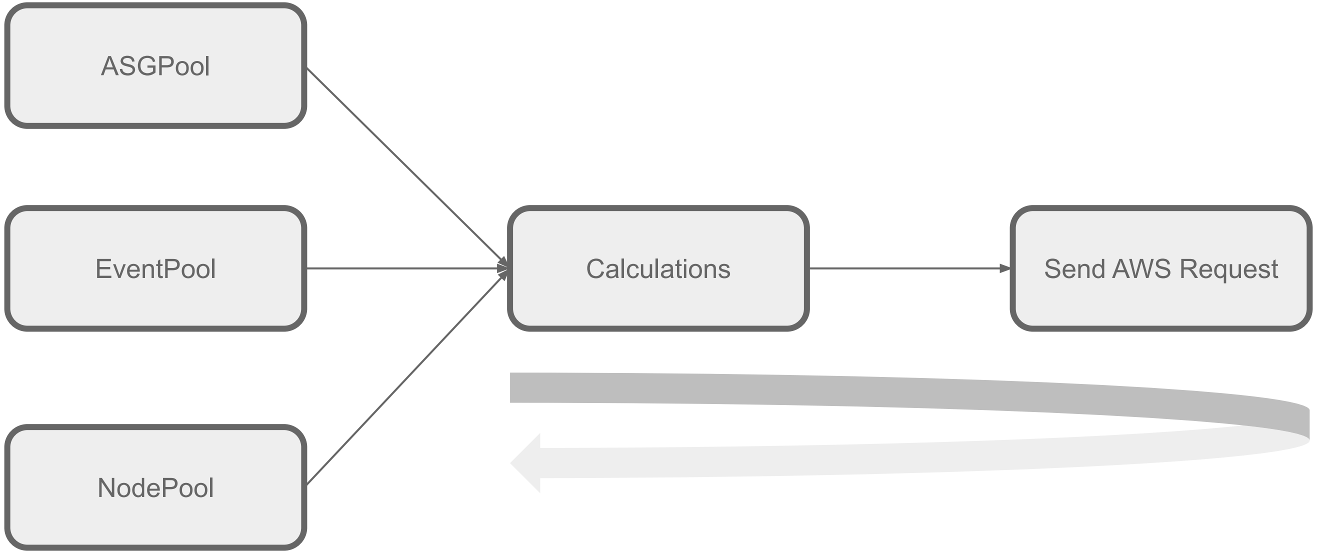
ASGPool

EventPool

NodePool

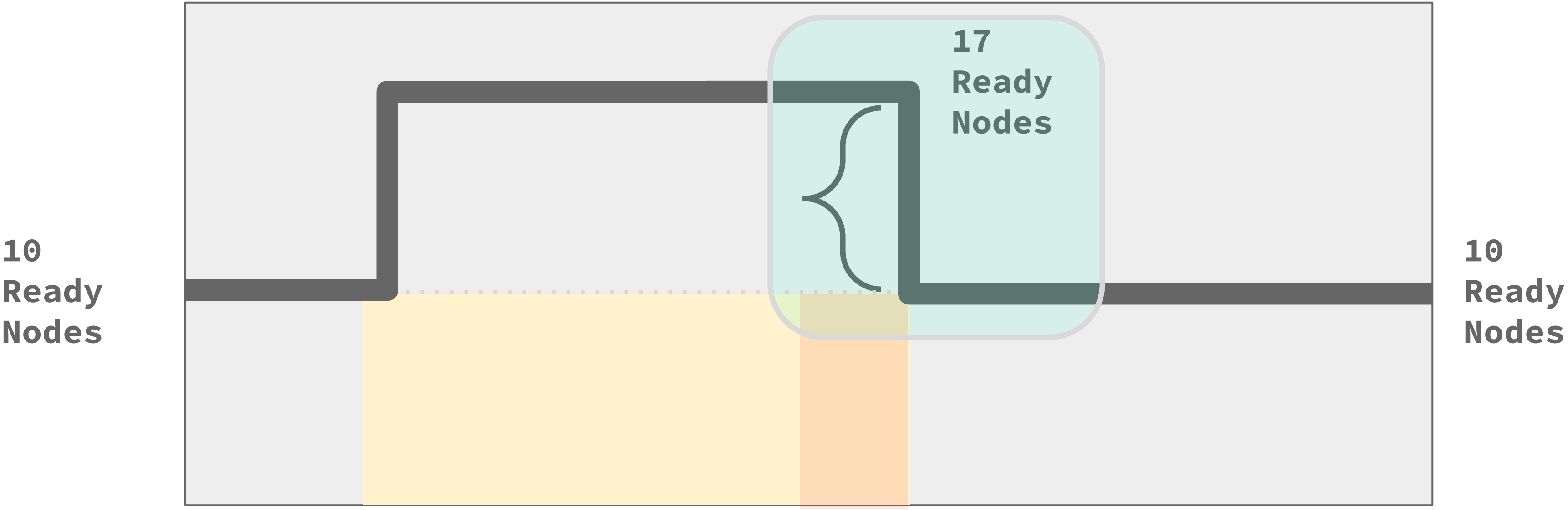
Docplanner

Possible solutions: Calculations overview

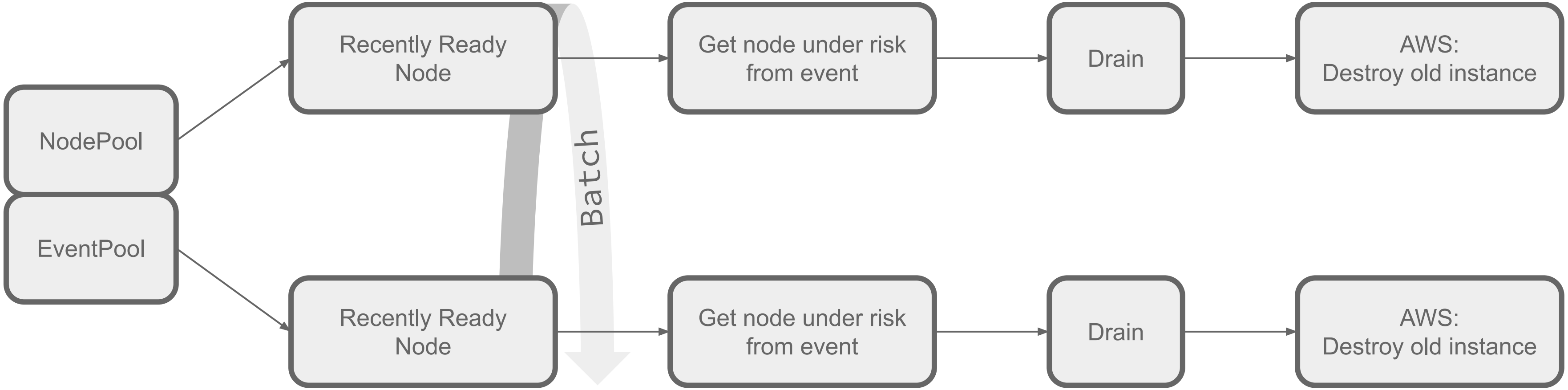


Docplanner

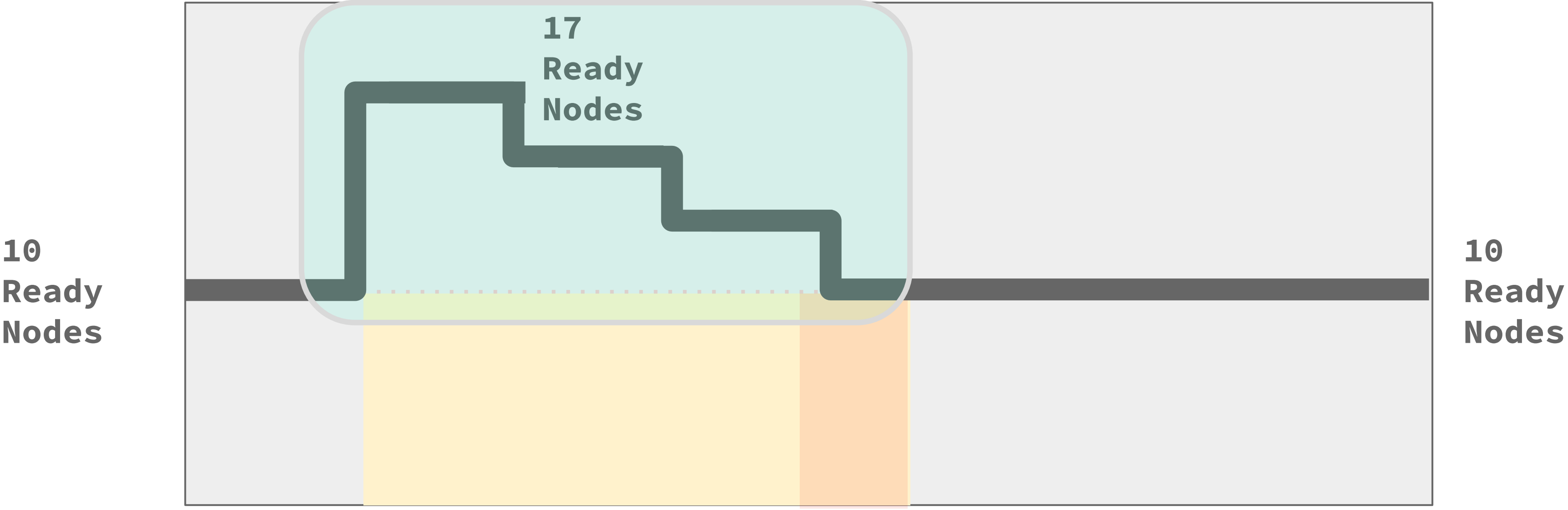
Possible solutions: Drain problem



Possible solutions: Batch Drain Solution


NodePool
EventPool
Recently Ready Node
Get node under risk from event
Drain
AWS: Destroy old instance
Batch
Recently Ready Node
Get node under risk from event
Drain
AWS: Destroy old instance

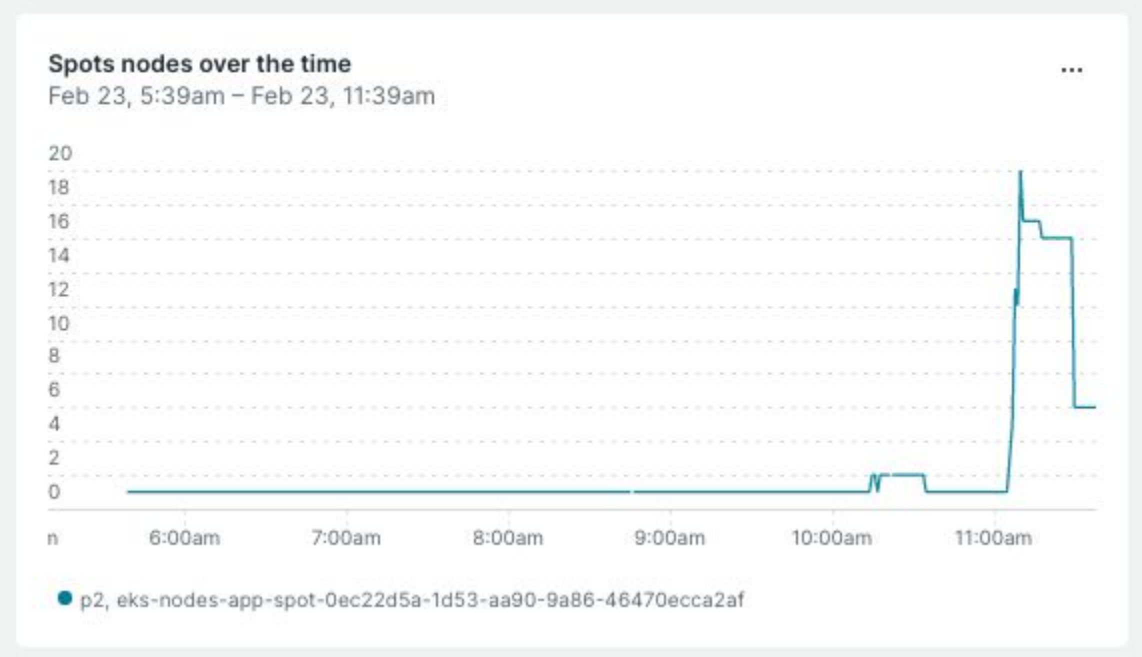
Possible solutions: Drain Result



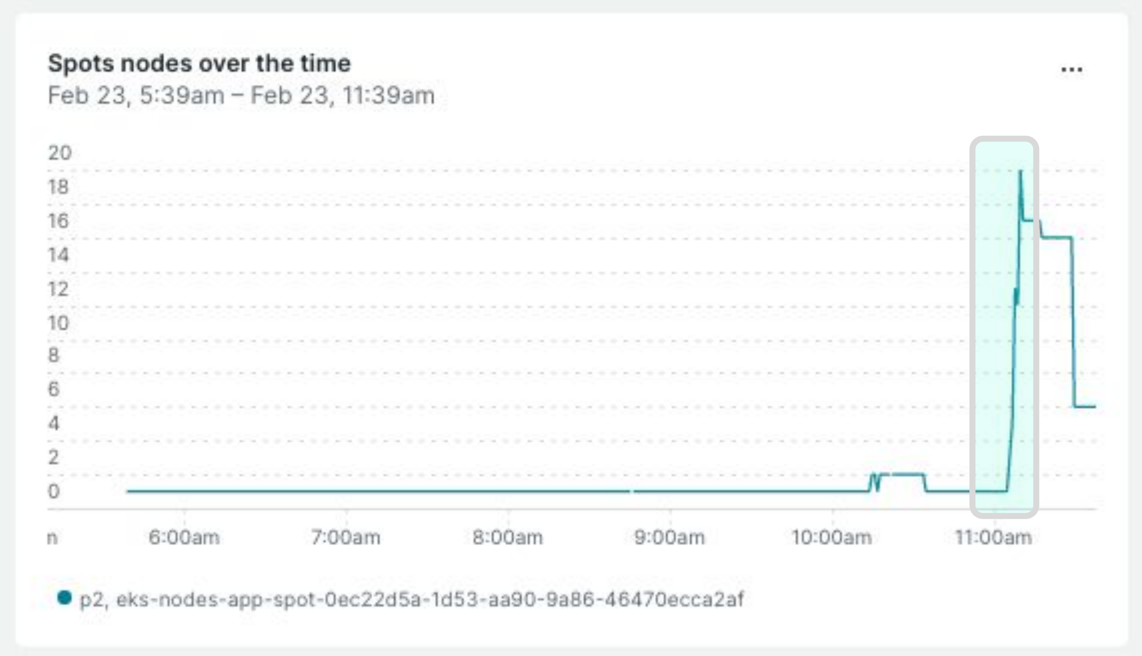
Will it work?

Case 1: Stable situation

Will Work?: real case

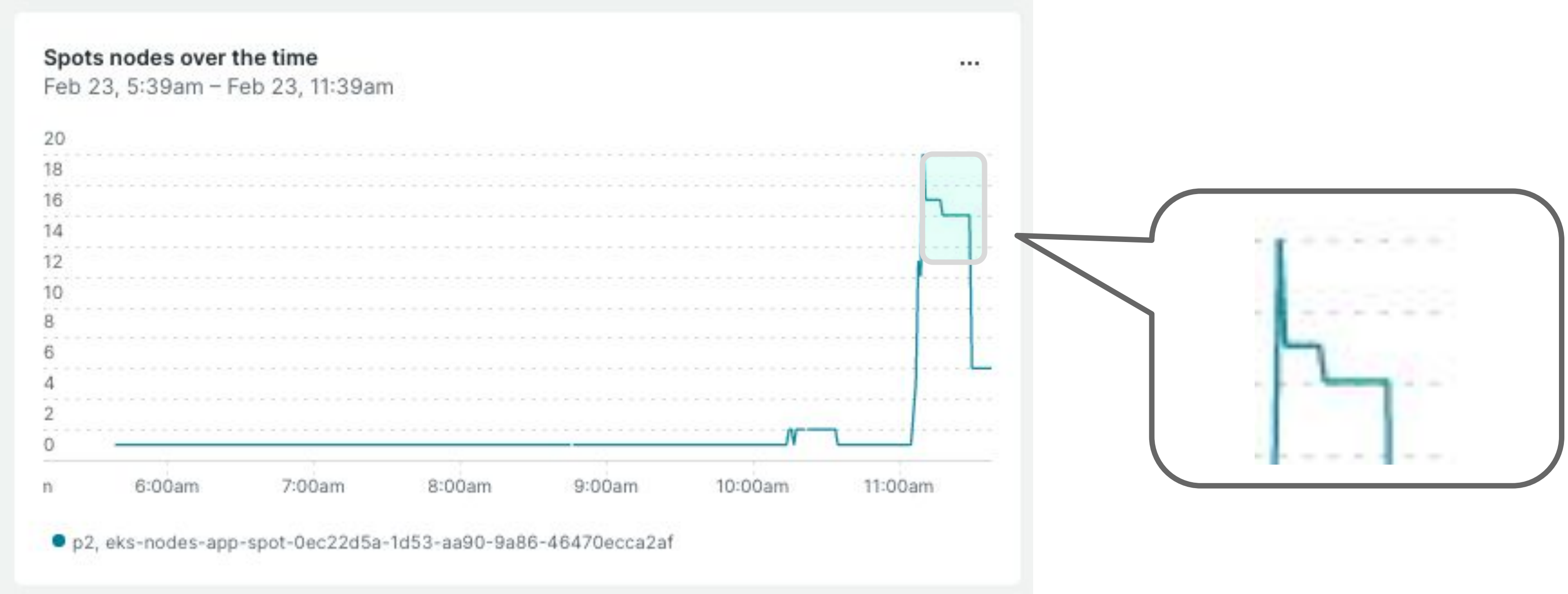


Will Work?: 1. Initial Spike of capacity



Docplanner

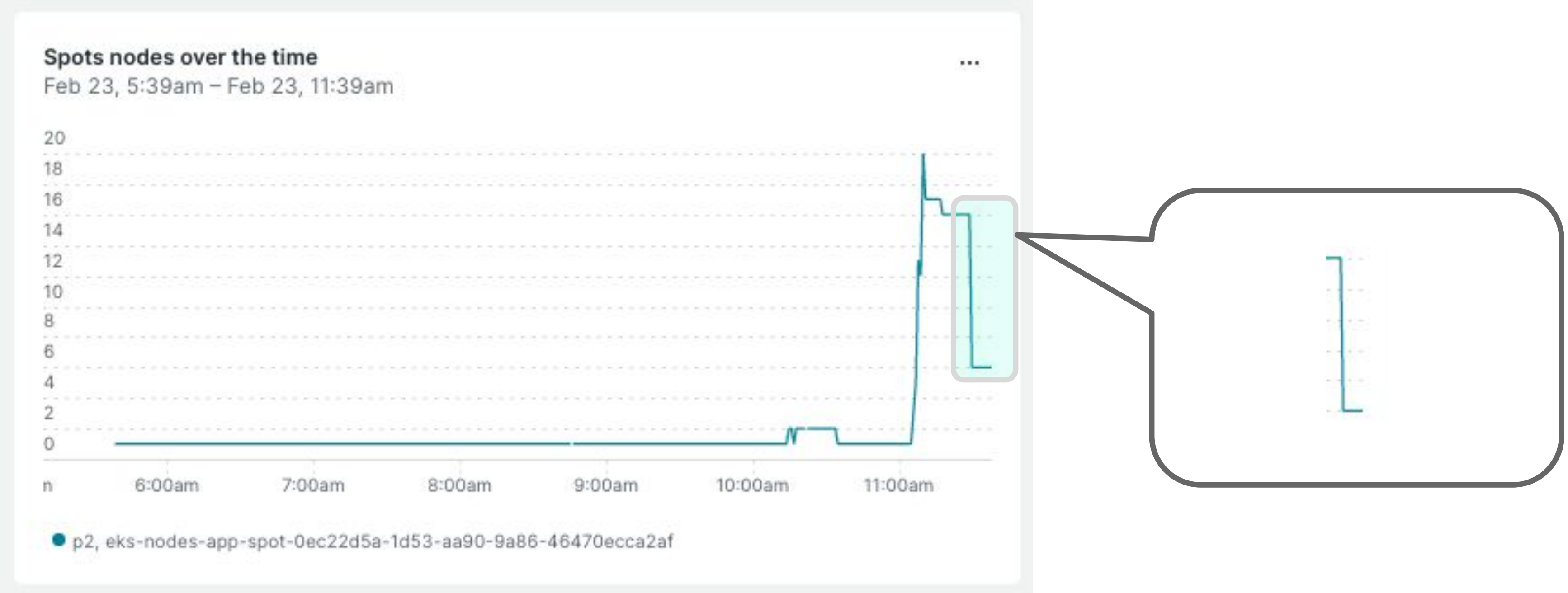
Will Work?: 2. Drain ladder



Spots nodes over the time

Feb 23, 5:39am – Feb 23, 11:39am

Will Work?: 3. Cluster Autoscaler adjustments



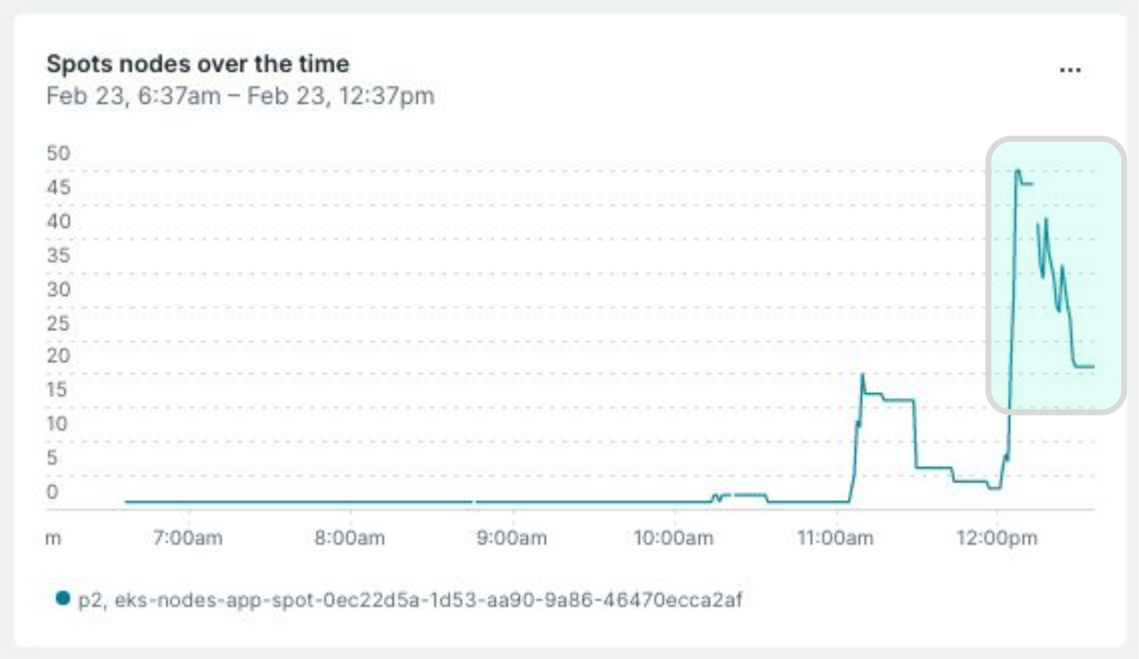
Spots nodes over the time

Feb 23, 5:39am – Feb 23, 11:39am

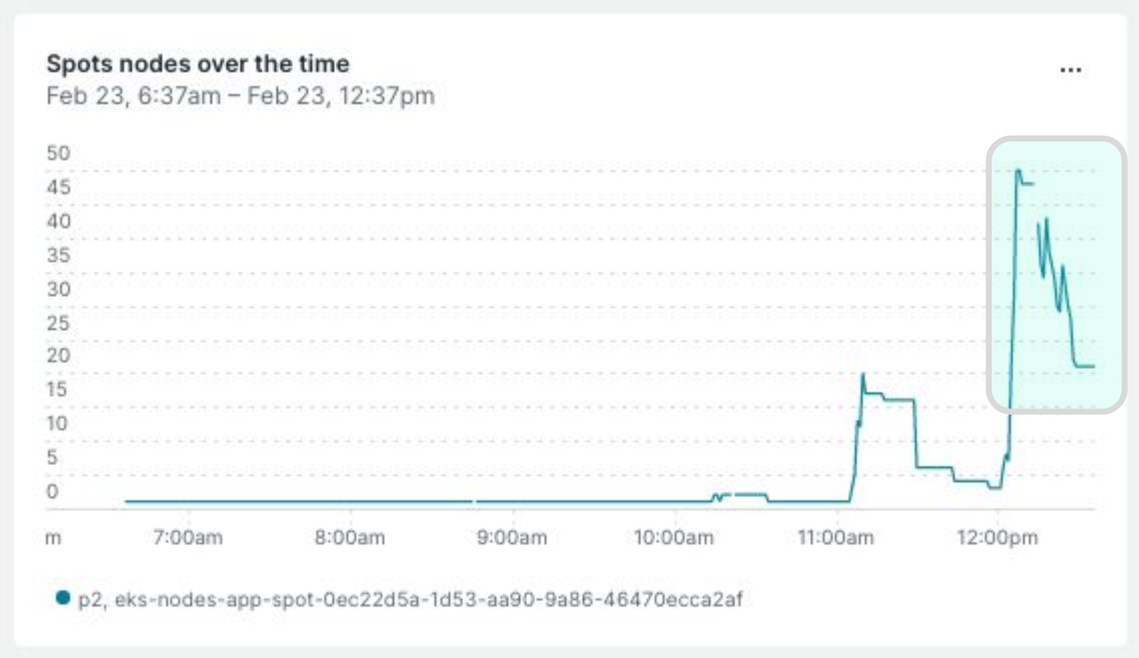
Will it work?

Case 2: AWS being greedy

Will Work?: Another Real case

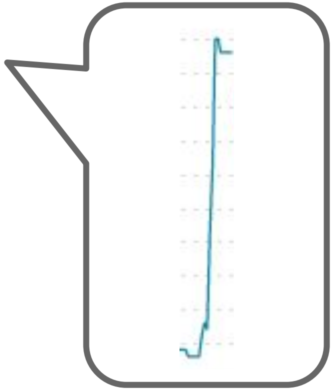


Will Work?: 1. Initial Spike of capacity

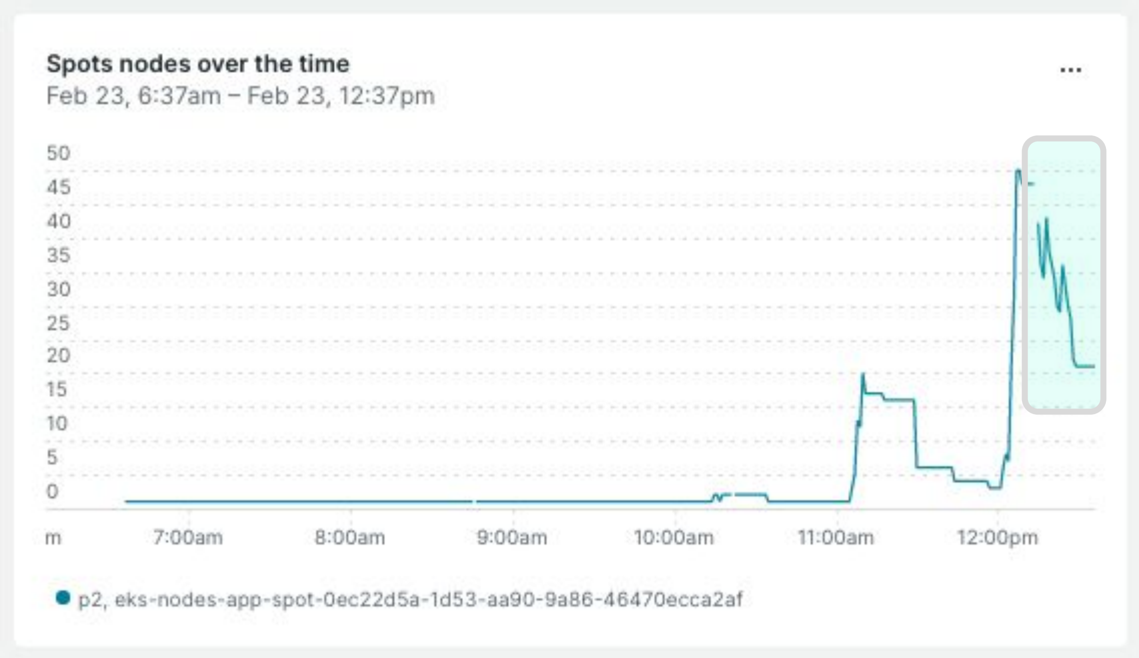


Spots nodes over the time

Feb 23, 6:37am – Feb 23, 12:37pm

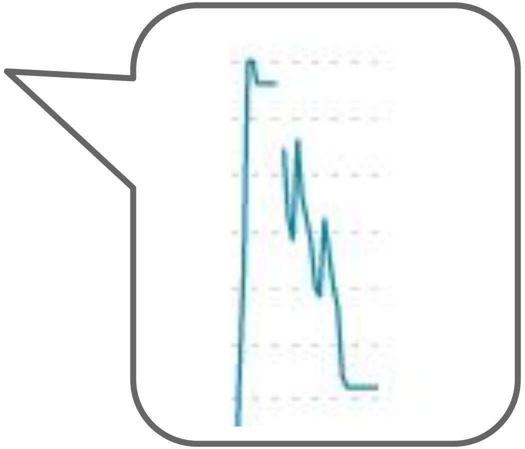


Will Work?: 2. Drain… ladder?

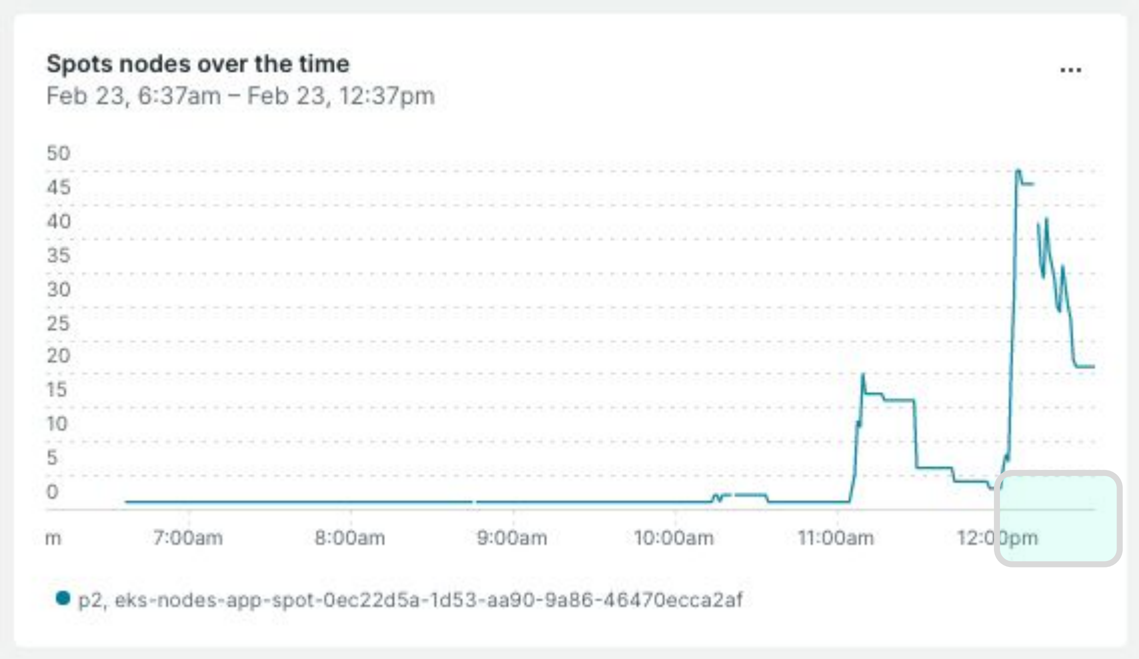


Spots nodes over the time

Feb 23, 6:37am – Feb 23, 12:37pm



Will Work?: 2. Drain… ladder?



- **Bad config:**
  Nodes are considered new since 10 minutes ago

- **Real case:**
  30 minutes ago a lot of nodes were joined