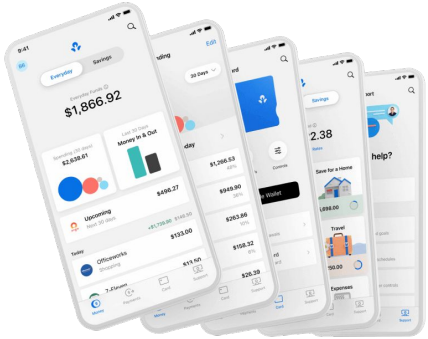
Lessons Learned Running GKE Clusters on Spot Instances

Olga Mirensky, Platform Engineer, ANZx

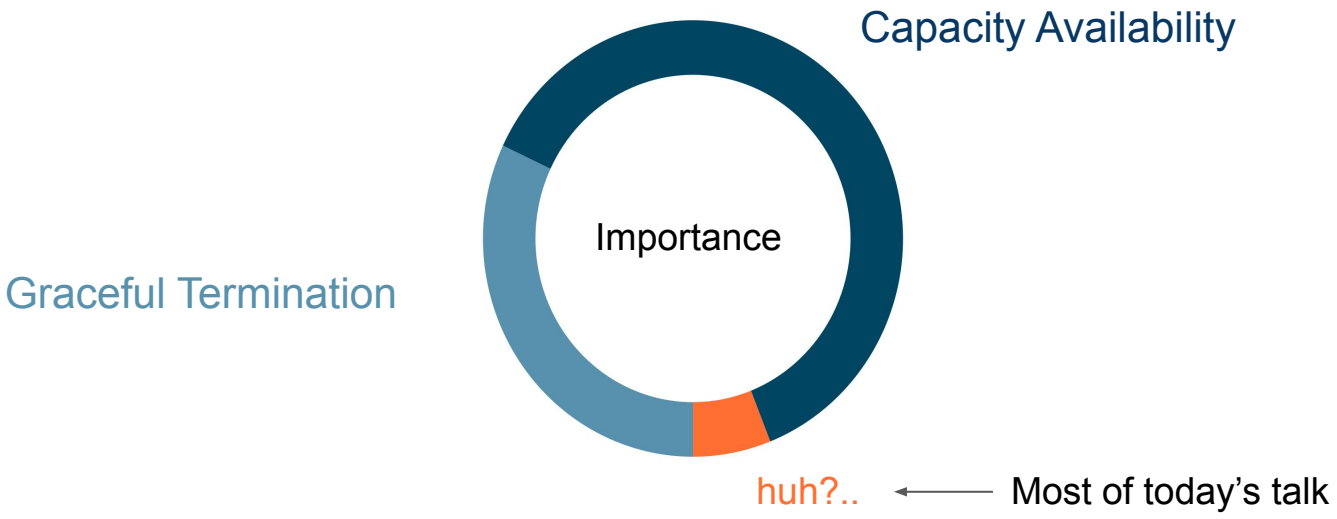


ANZ Plus

Spot Instances Quick Overview

	Google Cloud	aws	Azure
Discount	60 - 91%	Up to 90%	Up to 90%
Updates	Once a month	Can be frequent	Variable
Options	One size fits all	Price and/or capacity optimised	Set max price
Notice	30 sec	2 min	30 sec
On preemption	Stop/Hibernate	Stop/Hibernate/Terminate	Deallocate/Delete
Price Insights	API, Cost table	`aws ec2 describe-spot-price-history` Spot instance advisor	Portal price/eviction history, API

This Talk Scope



Disclaimer: This pie chart is a work of fiction. Any resemblance to actual stats is purely coincidental.

Spot Capacity Management in Kubernetes

- Fallback to on-demand automatically (and un-fallback)
  - Priority based expander for Cluster Auto Scaler
  - GKE Cluster Auto Scaler price-optimised by default
  - Weighted NodeAffinity
- Cluster Reserved Capacity
  - Cluster Auto Scaler config option
  - Headroom / balloon pods
- Managed Dataplane
  - Spot by NetApp, etc.
- Service quota limit for Spot CPU

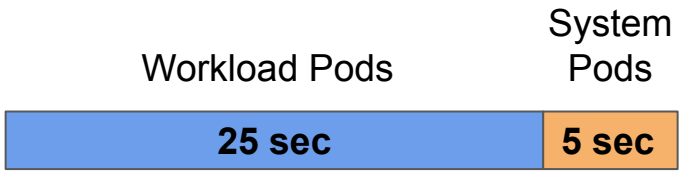
priority-expander-cm.yaml

Raw

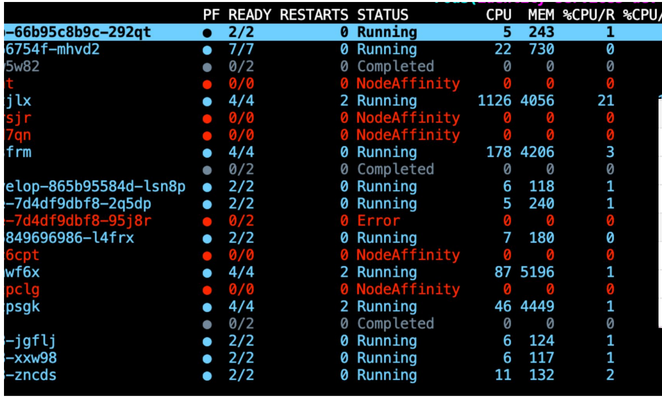
```yaml
# based on https://github.com/kubernetes/autoscaler
# higher number - higher priority (not in %)
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander
  namespace: kube-system
data:
  priorities: |-
    10:
      - .*on-demand.*
    50:
      - .*spot.*
```

Graceful Shutdown

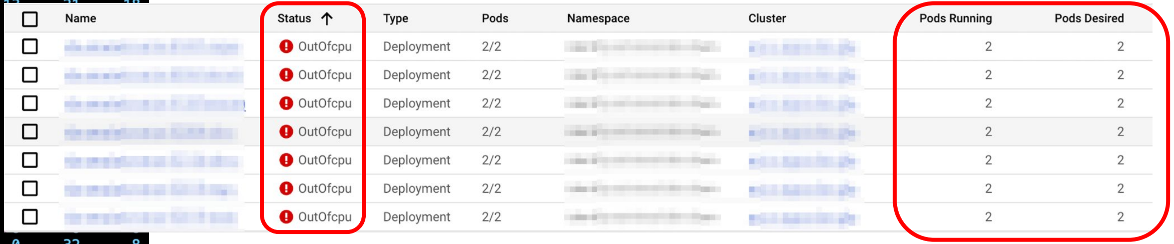
- Fault tolerant applications
- Graceful shutdown on SIGTERM
  - In-flight requests handled
  - New requests not routed and not accepted
  - External connections are closed (DB)
  - App specific requirements
- Node Graceful Shutdown feature in k8s
  - Enabled by default since 1.21
  - Node NotReady
  - SIGTERM propagation: workload vs system pods



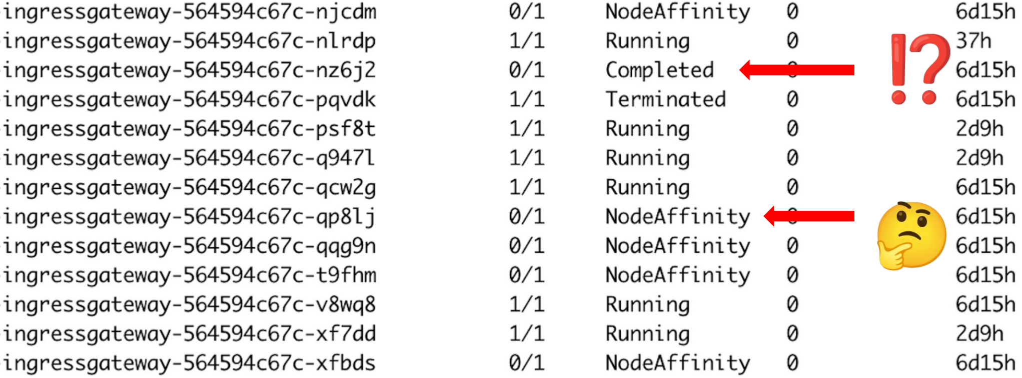
We broke everything (but not really)



```
                              PF READY RESTARTS STATUS          CPU  MEM %CPU/R %CPU/L %MEM/R %MEM/L
-66b95c8b9c-292qt             ●  2/2          0 Running           5  243      1      0     31      7
6754f-mhvd2                   ●  7/7          0 Running          22  730      0      0     16      5
5w82                          ●  0/2          0 Completed         0    0      0      0      0      0
t                             ●  0/0          0 NodeAffinity      0    0      0      0      0      0
jlx                           ●  4/4          2 Running        1126 4056     21
sjr                           ●  0/0          0 NodeAffinity      0    0      0
7qn                           ●  0/0          0 NodeAffinity      0    0      0
frm                           ●  4/4          0 Running         178 4206      3
                              ●  0/2          0 Completed         0    0      0
elop-865b95584d-lsn8p         ●  2/2          0 Running           6  118      1
-7d4df9dbf8-2q5dp             ●  2/2          0 Running           5  240      1
-7d4df9dbf8-95j8r             ●  0/2          0 Error             0    0      0
849696986-l4frx               ●  2/2          0 Running           7  180      0
6cpt                          ●  0/0          0 NodeAffinity      0    0      0
wf6x                          ●  4/4          2 Running          87 5196      1
pclg                          ●  0/0          0 NodeAffinity      0    0      0
psgk                          ●  4/4          2 Running          46 4449      1
                              ●  0/2          0 Completed         0    0      0
-jgflj                        ●  2/2          0 Running           6  124      1      0     32      8
-xxw98                        ●  2/2          0 Running           6  117      1      0     30      7
-zncds                        ●  2/2          0 Running          11  132      2      0     34      8
```



☐	Name	Status ↑	Type	Pods	Namespace	Cluster	Pods Running	Pods Desired
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2
☐	[redacted]	❗ OutOfcpu	Deployment	2/2	[redacted]	[redacted]	2	2



```
ingressgateway-564594c67c-njcdm       0/1     NodeAffinity   0           6d15h
ingressgateway-564594c67c-nlrdp       1/1     Running        0           37h
ingressgateway-564594c67c-nz6j2       0/1     Completed      0           6d15h
ingressgateway-564594c67c-pqvdk       1/1     Terminated     0           6d15h
ingressgateway-564594c67c-psf8t       1/1     Running        0           2d9h
ingressgateway-564594c67c-q947l       1/1     Running        0           2d9h
ingressgateway-564594c67c-qcw2g       1/1     Running        0           6d15h
ingressgateway-564594c67c-qp8lj       0/1     NodeAffinity   0           6d15h
ingressgateway-564594c67c-qqg9n       0/1     NodeAffinity   0           6d15h
ingressgateway-564594c67c-t9fhm       0/1     NodeAffinity   0           6d15h
ingressgateway-564594c67c-v8wq8       1/1     Running        0           6d15h
ingressgateway-564594c67c-xf7dd       1/1     Running        0           2d9h
ingressgateway-564594c67c-xfbds       0/1     NodeAffinity   0           6d15h
```

OutOfpods, Error, NotReady,

ContainerStatusUknown,

NodeShutdown, Terminated,

Init:ContainerStatusUnknown

and more!! 😵‍💫

message: Pod Predicate NodeAffinity failed

```
Warning  FailedMount  2m21s (x3870 over 5d10h) kubelet  MountVolume.SetUp failed for volume “xxxx”
                          : object “<namespace>“/”<name>” not registered
```

```
Warning  FailedMount   45m (x185 over 6h45m)  kubelet  MountVolume.SetUp failed for volume
.                        kube-api-access-12345" : object "my-ns"/"kube-root-ca.crt" not registered
Warning  NodeAffinity  41m                    kubelet  Predicate NodeAffinity failed
Warning  FailedMount   26s (x28 over 41m)     kubelet  MountVolume.SetUp failed for volume
.                        kube-api-access-12345" : object "my-ns"/"kube-root-ca.crt" not registered
```

Automatic Reclaiming

```
1 protoPayload.request.@type=(
2 "type.googleapis.com/compute.instances.repair.recreateInstance" OR
3 "type.googleapis.com/compute.instances.preempted" OR
4 "type.googleapis.com/compute.instances.delete")
5                                                  -676b2d4d-gnjb"
```

Log fields Histogram

Histogram

4
0

May 17, 9:30 AM 10:00 AM 11:00 AM

Query results 8 log entries

SEVERITY	TIMESTAMP ↑ AEST	SUMMARY		
	2023-05-17 09:37:53.995 AEST	compute.instances.preempted	compute.googleapis.com	compute.instances.preempted …pps-r
	2023-05-17 09:37:55.866 AEST	compute.instances.repair.recreateInstance	compute.googleapis.com	…e.instances.repair.re
	2023-05-17 09:37:57.635 AEST	v1.compute.instances.delete	GCE Managed Instance Group for GKE	compute.googleapis.com
	2023-05-17 09:38:05.249 AEST	v1.compute.instances.delete	GCE Managed Instance Group for GKE	compute.googleapis.com
	2023-05-17 10:08:33.560 AEST	compute.instances.preempted	compute.googleapis.com	compute.instances.preempted …pps-r
	2023-05-17 10:08:35.550 AEST	compute.instances.repair.recreateInstance	compute.googleapis.com	…e.instances.repair.re
	2023-05-17 10:08:37.338 AEST	v1.compute.instances.delete	GCE Managed Instance Group for GKE	compute.googleapis.com
	2023-05-17 10:08:44.780 AEST	v1.compute.instances.delete	GCE Managed Instance Group for GKE	compute.googleapis.com

To view older entries: Extend time by: 1 hour Edit time

SEVERITY TIMESTAMP ↑ AEST SUMMARY EDIT

method identity

The same node can be backed by different VMs over its lifetime

recreateInstance

NodeAffinity

- NodeAffinity pods traced back to a node with reclaimed VM and still in cluster
- Related to https://issuetracker.google.com/issues/185362914
  - Kubelet restart edge case
  - Still an issue with GKE preemptible VMs at the time.
  - Users still report this issue (1.24.10-gke.2300)

"Note that this issue has little to no impact on workloads. As long as the pod is backed by controller (deployment/statefulset, etc) a new pod is immediately created and rescheduled."

```
$ kubectl get pod $name -o yaml
…
status:
  message: Pod Predicate NodeAffinity failed
  phase: Failed
  reason: NodeAffinity
```

Little to no impact on workloads…

```
% k get pods --field-selector status.phase=Failed -o custom-columns=CREATED_AT:.metadata.creationTimestamp,NAME:.metadata.name,NODE:.spec.nodeName | sort
2023-02-28T02:50:24Z   prometheus-kube-state-metrics-6756f8f968-hxw7f   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:14Z   prometheus-kube-state-metrics-6756f8f968-s4nfz   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:18Z   prometheus-kube-state-metrics-6756f8f968-9b9fl   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:19Z   prometheus-kube-state-metrics-6756f8f968-f9vhw   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:20Z   prometheus-kube-state-metrics-6756f8f968-lrrjh   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:21Z   prometheus-kube-state-metrics-6756f8f968-fntvj   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:25Z   prometheus-kube-state-metrics-6756f8f968-2v8pt   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:26Z   prometheus-kube-state-metrics-6756f8f968-d2gm6   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:27Z   prometheus-kube-state-metrics-6756f8f968-l59nw   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:28Z   prometheus-kube-state-metrics-6756f8f968-7kv2d   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:28Z   prometheus-kube-state-metrics-6756f8f968-x9qjg   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:30Z   prometheus-kube-state-metrics-6756f8f968-chhzg   gke-[redacted]-03cbca2c-2pt6
2023-02-28T03:06:35Z   prometheus-kube-state-metrics-6756f8f968-qnjtg   gke-[redacted]-03cbca2c-2pt6
```

- 12 pods in 21 seconds on the same node
- At least 21 seconds deployment did not have desired capacity
- It is not a problem <u>now</u>, but <u>something happened</u> in the past

No Panic!

Does my Deployment (StatefulSet / DaemonSet) have desired number of replicas Running and Ready?

But there is a better way…

Not so “little impact”

- Platform should be easy to consume
- Software Engineers are not experts in Dead Pods
- Engineers raise “issues” and support requests again and again, wastes time
- Spot instances became the first suspect when anything goes wrong even when
  - Technically there is no issue
  - Or issues are not caused by Spot preemptions

Solutions

- k8s Garbage Collector
  - In GKE threshold is 1000 objects
- https://github.com/kubernetes-sigs/descheduler
  - Safely evicts (not deletes) pods
  - Rebalance Availability Zones
  - Spread pods of the same deployment across nodes
  - Remove ‘Failed’ pods immediately, and more
- Data
  - Platform Critical User Journeys (CUJ) and SLOs

Takeaways

Implementing well-known SRE k8s practices are crucial on Spot:

- Replication
- Spread across zones and nodes (TSC[1], pod AntiAffinity)
- Graceful shutdown
- Probes
- Tier applications by priority
- PDBs. Don’t protect from Spot preemptions, but improve overall availability

[1] new features: https://kubernetes.io/blog/2023/04/17/fine-grained-pod-topology-spread-features-beta/

What doesn’t kill you makes you stronger

Thank you

CAS expanders: https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/expander

Open Source CAS developed by AWS: https://karpenter.sh/

GKE on-demand fallback:
https://cloud.google.com/blog/topics/developers-practitioners/running-gke-application-spot-nodes-demand-nodes-fallback

TopologySpreadConstraints new features: https://kubernetes.io/blog/2023/04/17/fine-grained-pod-topology-spread-features-beta/

Resources

CAS expanders: https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/expander

Open Source CAS developed by AWS: https://karpenter.sh/

GKE on-demand fallback:
https://cloud.google.com/blog/topics/developers-practitioners/running-gke-application-spot-nodes-demand-nodes-fallback

TopologySpreadConstraints new features: https://kubernetes.io/blog/2023/04/17/fine-grained-pod-topology-spread-features-beta/