Maintaining SLAs

...when instances go down

SLAs and Error budgets

An error budget is the maximum amount of time that a technical system can fail without contractual consequences.

SLA Target	**Monthly allowed downtime - Error Budget**
99%	7 hours, 18 minutes
99.9%	43 minutes 50 seconds
99.99%	4 mins 23 seconds

“The tension between a fast pace of innovation and service reliability can be resolved by aiming to roll out new features as fast as possible without exhausting this error budget.”

- *Google SRE Handbook*

S3 availability SLA: https://aws.amazon.com/s3/sla/

Nodes are temporary

- Spot-instances get terminated in minutes to hours.
- There are 4 kubernetes releases every year. Every node needs to restart on a new machine image.
- On-demand instances can get unavailable/unreachable (around once a year)
- Autoscaling cluster nodes to keep costs low and maintain performance.

Nodes will keep going away...

Hence, we should design assuming this.

Goal

There should be no PagerDuty alerts where the root cause is an instance being terminated.

Services can be down “briefly”.

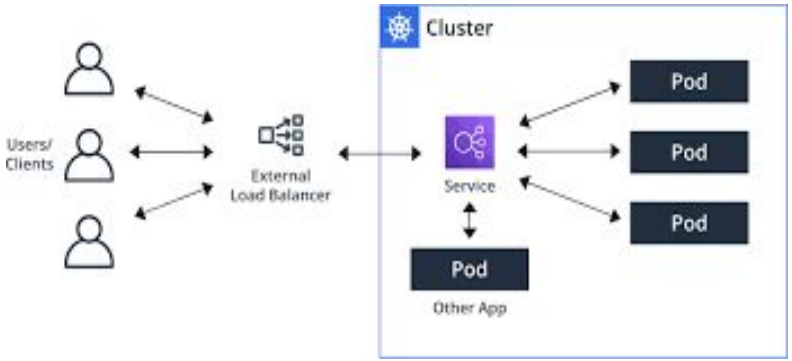
Issues should auto-resolve in minutes.

Kubernetes is opinionated

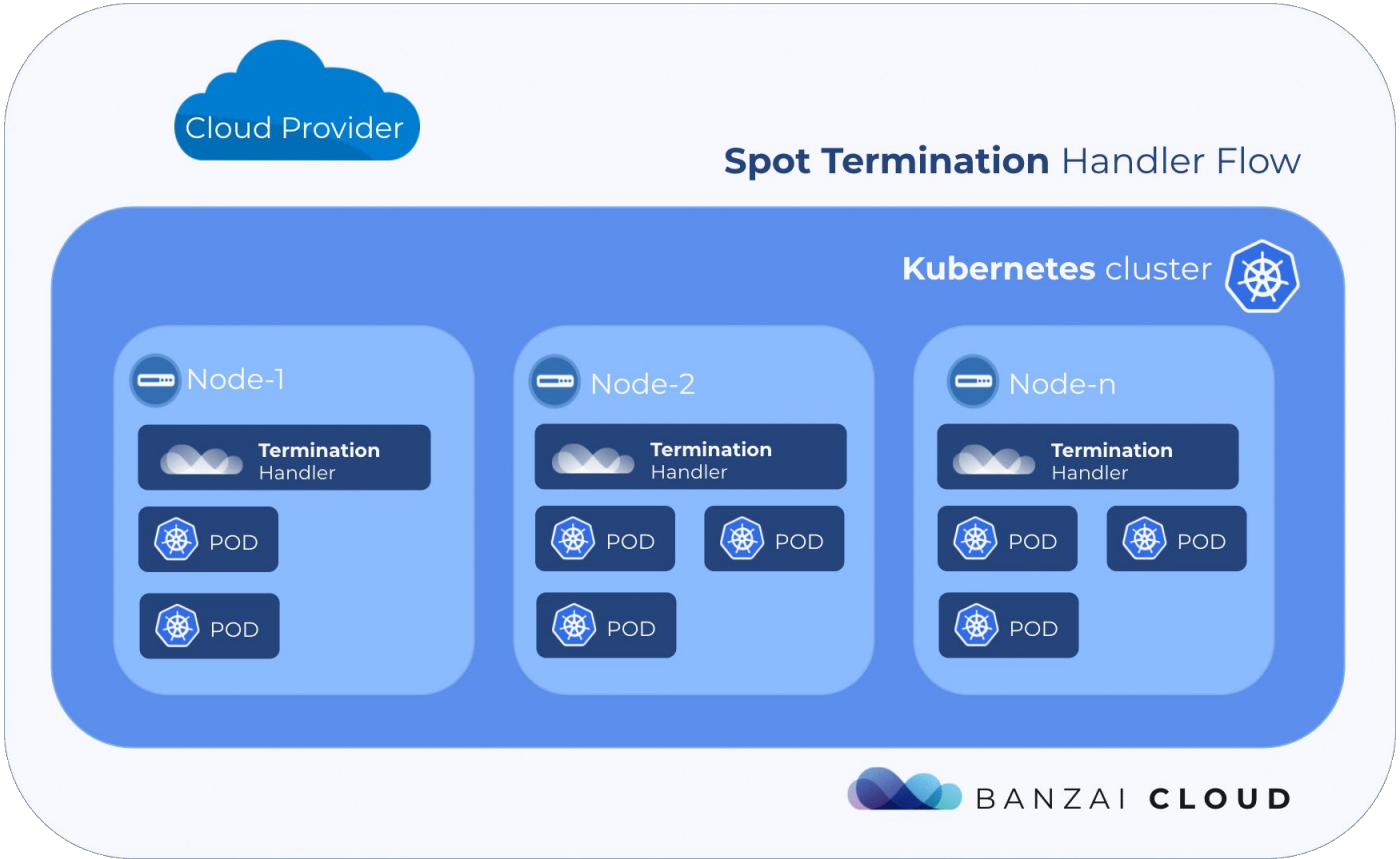
- Use existing mechanisms - labels, services, pod-lifecycle, etc.
- Built primarily for stateless services
  - Pods are considered expendable
- Some features for Stateful application support added
  - Databases (MySQL, Postgres), Queuing systems (RabbitMQ, Kafka) on Kubernetes is recent
  - Use complex Helm Charts or Kubernetes Operators to manage their deployments

Kubernetes Services

- Maintains a list of “Endpoints” (usually, Pod-Port)
- Exposes a cluster visible DNS name
- Requests to a service are round-robin load balanced among endpoints
- Various Kubernetes control loops keep updating this list of Endpoints



Node termination - Pod migration



When a node is terminated...

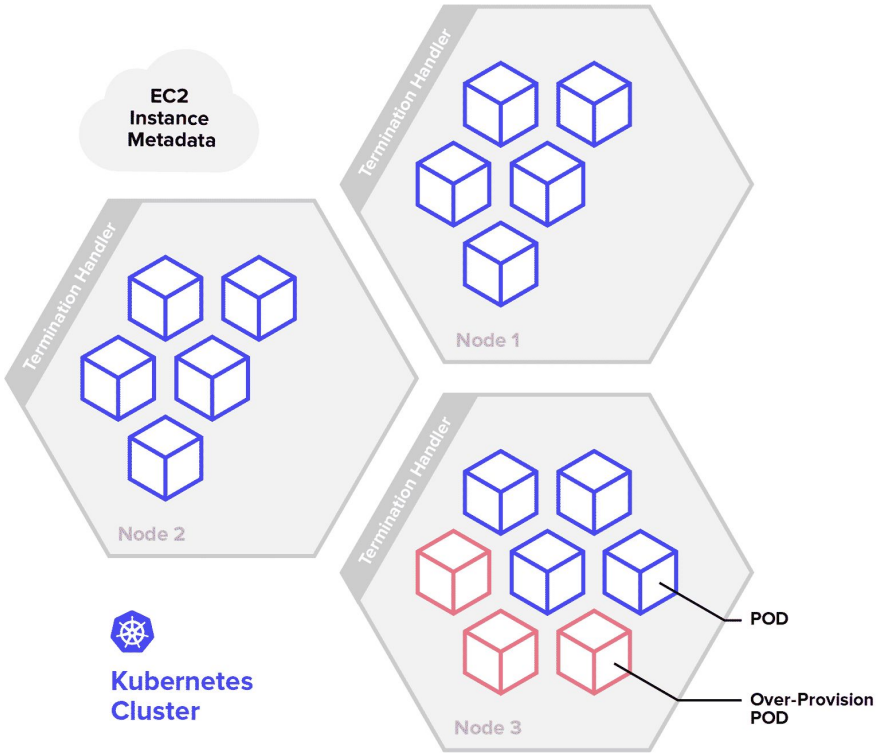
- Kubelet detects node system shutdown
  - GracefulNodeShutdown feature (beta since Kubernetes 1.21)
  - Uses systemd "*Inhibitor Locks*" to delay shutdown by a given duration

- Node is "cordoned"
  - No new pods are scheduled on it
- All pods get SIGTERM
  - "critical" pods - terminated after "regular" pods
- Pod's **preStop** hook is run if defined
  - Set **terminationGracePeriodSeconds** in the pod (to change grace period from default 30s)
- Pod Endpoint is removal from Service and ReplicaSet

- When the grace-period expires, all pods get SIGKILL
- API server deletes Pod's API object

- Kubelet may not detect a shutdown and Non-Graceful Node Shutdown
- *"...even with graceful node shutdown, you should still design your deployments to be resilient to node failures."*

Compromise uptime and costs



https://medium.com/riskified-technology/run-kubernetes-on-aws-ec2-spot-instances-with-zero-downtime-f7327a95dea

How does the cluster-autoscaler work

This is different from the HorizontalPodAutoscaler resource - which scales number of replicas.

Cluster-autoscaler has a 30 second control-loop:

- ScaleUp: If pods are Pending, start a new node(s) in appropriate nodegroup
- ScaleDown: If resource utilization (CPU/Mem) is less than set value, “drain” the node and delete it.
- Respects nodeSelector, taints and tolerations, podAntiAffinity etc

Coda

- Stateless pods - add slide
- Let nodes contain many pods, some room left in nodes
- Measure and improve SLA metric

More...

Kubernetes Deployments

- RollOutStrategy
- PodDisruptionBudget
- PodAntiAffinity