Kubernetes Load Balancer Configuration – Beware when draining nodes

Marcel Juhnke 1. Februar 2019 Veröffentliche einen Kommentar

German version

Recently, we created a new node-pool for our Kubernetes cluster hosted by GKE, to migrate the workloads from an existing node-pool to the new one.

During the draining operation of the old nodes, we noticed that none of the services in the cluster were reachable anymore via our ingress-nginx.

While checking the Load Balancer service of our ingress-nginx and its nginx-ingress-controller Pod endpoints, we found the following:

- The Load Balancer service changed the GCP Load Balancer backend pool configuration to point only to the new nodes, but no longer to the old ones
- The nginx-ingress-controller Pods were still running on the old nodes. We had some issues with Pod disruption budgets which led to the Pods not being evicted from the old nodes

But why was this a problem and the services not reachable via the Ingress? kube-proxy usually transparently forwards non-local traffic to the correct node, regardless of which node receives the original traffic.

Kubernetes has a feature for certain public cloud load balancer configs that helps keeping the original client IP address of requests instead of replacing them with the source IP of whatever node received the traffic and forwards it. Setting `service.spec.externalTrafficPolicy` to `Local` in the Service Spec proxies requests only to Endpoints that are local to the node receiving the traffic.

This is set by default in the deployment manifest of ingress-nginx for GCP (and makes sense if you want to have the original client IPs in your webserver logs).

Next is that Kubernetes creates an HTTP health check in GCP for the load balancer that queries the `/healthz` endpoint on the load balancer's `healthCheckNodePort`. Due to the externalTrafficPolicy setting this node health check deliberately fails on all nodes that don't have active Service endpoints (ingress-nginx Pods in our case) running.

To top it off, the Kubernetes service controller only adds those nodes into the load balancer backend config that are Ready and Schedulable. As soon as a node is no longer schedulable (because it has been cordoned for the drain operation) it gets removed from the load balancer backend config. As I understand the linked code, this is to filter out master servers from being added to a load balancer endpoint, but it also affects normal worker nodes, as soon as they are cordoned.

All those circumstances together led to the health checks in the load balancer only pointing to the new nodes, which all failed now because there were no ingress-nginx Pods running there yet and no traffic being forwarded to the old nodes, resulting effectively in a total traffic outage for our Ingresses on this cluster.

The solution in our case was simple, we deleted the old nginx-ingress-controller Pods that were still hanging around on the old nodes so they got rescheduled on the new ready nodes.

Nevertheless, before doing any changes that might touch any traffic, look twice into the documentation.

References

Load Balancer deciding if default healthcheck port should be used or specific healthCheckNodePort

kube-proxy deciding if nodeHealthCheck passes or not, depending on if a local Service endpoint exists

Unsatisfied Pod disruption budget blocks node drain operation

Marcel Juhnke 1. Februar 2019 Allgemein kubernetes, load balancer

—Vorheriger Beitrag
Kubernetes Load Balancer Konfiguration – Vorsicht beim Drainen von Nodes

Schreib einen Kommentar

Du musst angemeldet sein, um einen Kommentar abzugeben.

Seiten

Willkommen

Suche ...

Suche

Neue Beiträge

Kubernetes Load Balancer Configuration – Beware when draining nodes

Kubernetes Load Balancer Konfiguration – Vorsicht beim Drainen von Nodes

GitHub Actions in Action

Archive

Februar 2019

Januar 2019

Kategorien

Allgemein

Tech