DNS Lookup Failures Debugging

During the Hacker News soft-launch (April 27, 2021), I encountered a peculiar error: https://console.cloud.google.com/errors/CN-N8I-Gu4qt0QE?time=P30D&project=ufincs2

```
GeneralError: getaddrinfo EAI_AGAIN backend-database
    at new GeneralError
(/backend/node_modules/@feathersjs/errors/lib/index.js:177:17)
    at exports.errorHandler (/backend/node_modules/feathers-
sequelize/lib/utils.js:29:20)
    at tryCatcher
(/backend/node_modules/bluebird/js/release/util.js:16:23)
    at Promise._settlePromiseFromHandler
(/backend/node_modules/bluebird/js/release/promise.js:547:31)
    at Promise._settlePromise
(/backend/node_modules/bluebird/js/release/promise.js:604:18)
    at Promise._settlePromise0
(/backend/node_modules/bluebird/js/release/promise.js:649:10)
    at Promise._settlePromises
(/backend/node_modules/bluebird/js/release/promise.js:725:18)
    at _drainQueueStep
(/backend/node_modules/bluebird/js/release/async.js:93:12)
    at _drainQueue
(/backend/node_modules/bluebird/js/release/async.js:86:9)
    at Async._drainQueues
(/backend/node_modules/bluebird/js/release/async.js:102:5)
    at Immediate.Async.drainQueues [as _onImmediate]
(/backend/node_modules/bluebird/js/release/async.js:15:14)
    at processImmediate (internal/timers.js:461:21)
```

According to https://stackoverflow.com/questions/40182121/whats-the-source-of-error-getaddrinfo-eai-again, this would appear to be a DNS lookup failure, particularly from the Backend service to the Database.

Since it was just a part of the Hacker News, I thought the heightened load might have had something to do with it. Additionally, I'd had never encountered this error before, so I thought little of it.

However, it recurred today (April 30, 2021). Same error message. Upon further investigation, I was also seeing the following errors from `ingress-nginx`:

```
2021/04/30 18:40:45 [error] 37#37: *37533 connect() failed (113: Host
is unreachable) while connecting to upstream, client: 173.33.167.29,
server: backend.ufincs.com, request: "GET /featureFlags HTTP/2.0",
upstream: "http://10.101.1.3:5000/featureFlags", host:
"backend.ufincs.com"
```

This would seem to indicate that the ingress was having trouble connecting to the Backend service.

Noteworthy is that this error occurs roughly 3 minutes prior to the `backend-database` DNS issue occurring.

Also noteworthy is the fact that one of the Backend pods would get restarted shortly after I was notified of the DNS error. In fact, one of each pod from *all three services* would get evicted:

```
28m         Normal   TaintManagerEviction   pod/backend-688556cb8d-
tcj65    Cancelling deletion of Pod master/backend-688556cb8d-tcj65
28m         Normal   Pulling                pod/backend-688556cb8d-
tcj65    Pulling image "busybox"
27m         Normal   Pulled                 pod/backend-688556cb8d-
tcj65    Successfully pulled image "busybox"
27m         Normal   Created                pod/backend-688556cb8d-
tcj65    Created container wait-for-service
27m         Normal   Started                pod/backend-688556cb8d-
tcj65    Started container wait-for-service
26m         Normal   Pulling                pod/backend-688556cb8d-
tcj65    Pulling image "gcr.io/ufincs2/ufincs-backend:6aa9fb4"
26m         Normal   Pulled                 pod/backend-688556cb8d-
tcj65    Successfully pulled image "gcr.io/ufincs2/ufincs-
backend:6aa9fb4"
26m         Normal   Created                pod/backend-688556cb8d-
tcj65    Created container backend
26m         Normal   Started                pod/backend-688556cb8d-
tcj65    Started container backend
27m         Normal   Sync                   ingress/backend-ingress
Scheduled for sync
28m         Normal   TaintManagerEviction   pod/frontend-5477b86f9-
p75gw    Cancelling deletion of Pod master/frontend-5477b86f9-p75gw
28m         Normal   Pulling                pod/frontend-5477b86f9-
p75gw    Pulling image "gcr.io/ufincs2/ufincs-frontend:6aa9fb4"
27m         Normal   Pulled                 pod/frontend-5477b86f9-
p75gw    Successfully pulled image "gcr.io/ufincs2/ufincs-
frontend:6aa9fb4"
27m         Normal   Created                pod/frontend-5477b86f9-
p75gw    Created container frontend
27m         Normal   Started                pod/frontend-5477b86f9-
p75gw    Started container frontend
27m         Normal   Sync                   ingress/frontend-ingress
Scheduled for sync
28m         Normal   TaintManagerEviction   pod/marketing-c9c775897-
btk9p   Cancelling deletion of Pod master/marketing-c9c775897-btk9p
28m         Normal   Pulling                pod/marketing-c9c775897-
btk9p   Pulling image "gcr.io/ufincs2/ufincs-marketing:6aa9fb4"
28m         Normal   Pulled                 pod/marketing-c9c775897-
btk9p   Successfully pulled image "gcr.io/ufincs2/ufincs-
```

```
marketing:6aa9fb4"
28m         Normal   Created                  pod/marketing-c9c775897-
btk9p   Created container marketing
28m         Normal   Started                  pod/marketing-c9c775897-
btk9p   Started container marketing
27m         Normal   Sync                     ingress/marketing-ingress
Scheduled for sync
```

I'm having trouble figuring out what this `TaintManagerEviction` is, or why it happened. Ostensibly, such an eviction would only happen if the pod's taint was no longer being respected, but that can't be the case considering we don't *use* any taints.

I was going to say that this doesn't seem related to node preemption, because `kubectl get nodes` lists each node as being greater than 24h old (already quite strange), but indeed, the compute instance UI lists one of the instances as having been created at 2:41.

So our current timeline is:

- 2:40 ingress-nginx can't resolve connection to Backend service/pod
- 2:41 A new node is created (presumably, the node has been preempted by now)
- 2:42 `TaintManagerEviction` happens to the pod
- 2:43 I am notified that there's an error with reaching the Database

Now, what's most concerning is that the first error (the DNS one) gets shown directly to the user if they are trying to log in (because we don't have any extra database error handling yet -- although we are literally in the middle of UFC-361 to fix that).

So once we have UFC-361 in place, we should be fine.

I had thought this might have been some more subtle DNS problem, perhaps like Solving DNS lookup failures in Kubernetes, but once I learned that a node had been preempted, it made more sense.

Note, however, that because this is a DNS issue, I'm making one extra change: scaling `kube-dns` back up so that it runs one replica on each node (at least, in our case).

- Refer to Reducing add-on resource usage in smaller clusters for how to do this.

So, the actions that should/have been taken as part of fixing/mitigating this error:

- Finish UFC-361 so that any database errors are merely counted as 'network' errors.
- Make `kube-dns` HA by enabling its autoscaling.
- Add a log metric/chart explicitly for node preemptions.

Should, after performing all those mitigations, this error reoccur, the following might help:

- Enable the NodeLocal DNSCache as outlined in the above article.

   - Here's the GKE article for how to enable it.

And if those don't work, then... investigate some more.

Related

Another error that I think is related is Google Cloud Platform:

```
Error: A request error occurred: getaddrinfo EAI_AGAIN hooks.slack.com
    at Object.requestErrorWithOriginal
(/backend/node_modules/@slack/webhook/dist/errors.js:26:33)
    at IncomingWebhook.send
(/backend/node_modules/@slack/webhook/dist/IncomingWebhook.js:55:32)
    at runMicrotasks (<anonymous>)
    at processTicksAndRejections
(internal/process/task_queues.js:93:5)
```

It seems to have a DNS lookup timeout when trying to submit an internal notification through Slack. I encountered this once during the HN soft-launch, and again today [May 2, 2021], just randomly.

I will note that a node preemption happened around the same time, so that's likely the cause. However, this probably means that we need to look deeper into why node preemptions cause DNS lookup failures -- particularly ones when the DNS lookup is to something outside the cluster.

Maybe we will just have to run the NodeLocal DNSCache...