UToronto Hub Login failing with 500 errors

Status: Draft

Overview

Users were unable to log in or start servers due to the disk containing the JupyterHub sqlite db being full, due to a super large log file. Clearing this out restored service.

What Happened

The UToronto JupyterHub was storing hub logs on the same disk as the hub database. The disk capacity was 10G, and this eventually completely filled up. DB operations could no longer succeed, causing hub logins and server starts to fail. Users who were already logged in could continue to use the hub.

UToronto is the *only* hub maintained by 2i2c that keeps hub logs on the same disk as the hub database. Investigating why, we come across https://github.com/2i2c-org/infrastructure/pull/886/commits/9c30396a46202de639f12f5b924cd9fe03a8f086. The commit message acknowledges that this is unideal, but is just replicating the prior config we had for UToronto. Most likely this is driven by the fact that we don't rely on Azure for logging at all, and the desire to keep some deep logs for analytics purposes.

Resolution

Upon acknowledging the issue, the hub logs made it fairly clear what the problem was. The logs were gzipped (cut size by 10x) to restore service. The hub database disk was also resized (from 10G to 60G) to buy some more time, as the log writing currently still persists.

Where we got lucky

A user noticed the issue and notified the appropriate channel (2i2c support email) that we monitor.

What Went Well?

1. Dynamic image resizing in Kubernetes 'just worked', and our disk was resized as required without needing a lot of fiddling! https://kubernetes.io/blog/2018/07/12/resizing-persistent-volumes-using-kubernetes/

What Didn't Go So Well?

1. We had identified this as a potential problem during migration in https://github.com/2i2c-org/infrastructure/pull/886/commits/9c30396a46202de639f12f5b924cd9fe03a8f086, but didn't have a solid process for following through. Our processes have significantly improved since, and hopefully will handle these better.
2. There were no alerts for this, and we had to wait for a human to notify us.

OWNER OF REVIEW PROCESS

Yuvi Panda

IMPACT TIME

Oct 31 at 18:27 to Oct 31 at 21:00

DURATION

2h 33m

*All times listed in this report are in Pacific Time (US & Canada).

Action Items

1. Stop keeping logs on the same disk as the hub database https://github.com/2i2c-org/infrastructure/pull/1847
2. Add a graph charting free space in the JupyterHub db disk to Grafana https://github.com/jupyterhub/grafana-dashboards/pull/49
3. Alert based on elevated error rates (in-progress, Q4 goal for us) https://github.com/2i2c-org/infrastructure/issues/1804

Timeline

Oct 31, 2022

6:27 PM JupyterHub DB Disk fills up, and new users can not log in or start servers

Determined by looking at Grafana logs for last successful user login

7:13 PM Email received at support@2i2c.org that users unable to log in

https://2i2c.freshdesk.com/a/tickets/249

8:33 PM Issue acknowledged

8:47 PM Hub Logs showed that the db disk was full

Logs of the hub pod contained:

```
[E 2022-11-01 03:39:39.905 JupyterHub base:97] Rolling back session due to database error
--- Logging error ---
Traceback (most recent call last):
File "/usr/local/lib/python3.9/site-packages/sqlalchemy/engine/base.py", line 1900, in _execute_context
self.dialect.do_execute(
File "/usr/local/lib/python3.9/site-packages/sqlalchemy/engine/default.py", line 736, in do_execute
cursor.execute(statement, parameters)
sqlite3.OperationalError: database or disk is full
```

8:47 PM Looking at config, we discover that we store the hub logs in same disk as hub db

https://github.com/2i2c-org/infrastructure/blob/1f916f8851fa636af065d427ba70267c55c8c1de/config/clusters/utoronto/common.values.yaml#L89 is the relevant config

8:50 PM Existing log file is gzipped to reduce size

Can not be gzipped in place (because disk is full), so instead it is gzipped into /tmp/jupyterhub.log.

Shell on hub pod is obtained with `kubectl -n prod exec -it hub-7f4f9d4648-849w5 -c hub -- /bin/bash`.

`gzip < jupyterhub.log > /tmp/jupyterhub.log.gz` is then run to gzip the file. This is done rather than deletion to preserve any analytics we *might* want to perform, at the cost of a

few more minutes of downtime

8:55 PM gzipped file is copied locally so pod can be deleted and restarted

Deleting the uncompressed file on disk doesn't actually free up space, as the JupyterHub process is still holding an open file handle to that file! So the hub process needs to be restarted for the space to be reclaimed. However, restarting the pod will lose our compressed log file as well, so it is copied locally first.

`k -n prod cp -c hub hub-7f4f9d4648-849w5:/tmp/jupyterhub.log.gz jupyterhub.log.gz` copies the file locally

8:57 PM DB Disk resized to 60G from 10G

To buy us more time to figure out what to do with logs, the hub db disk is resized to 60G. Since kubernetes dynamic resizing requires a pod restart, doing this now (while the log copying was still ongoing) allowed us to get both these done with one restart.

`kubectl -n prod edit pvc hub-db-dir` is used to edit the PVC object, and the storage attribute is set to 60G

8:59 PM The hub pod is restarted and service is restored

It comes back up with a larger disk, and everything is alright in the world again

10:01 PM Compressed jupyterhub.log.gz file is copied back to the hub disk

`kubectl -n prod cp -c hub jupyterhub.log.gz hub-7f4f9d4648-mcrrb:/srv/jupyterhub /jupyterhub.log.gz` copies the local gzipped file back to the hub db dir