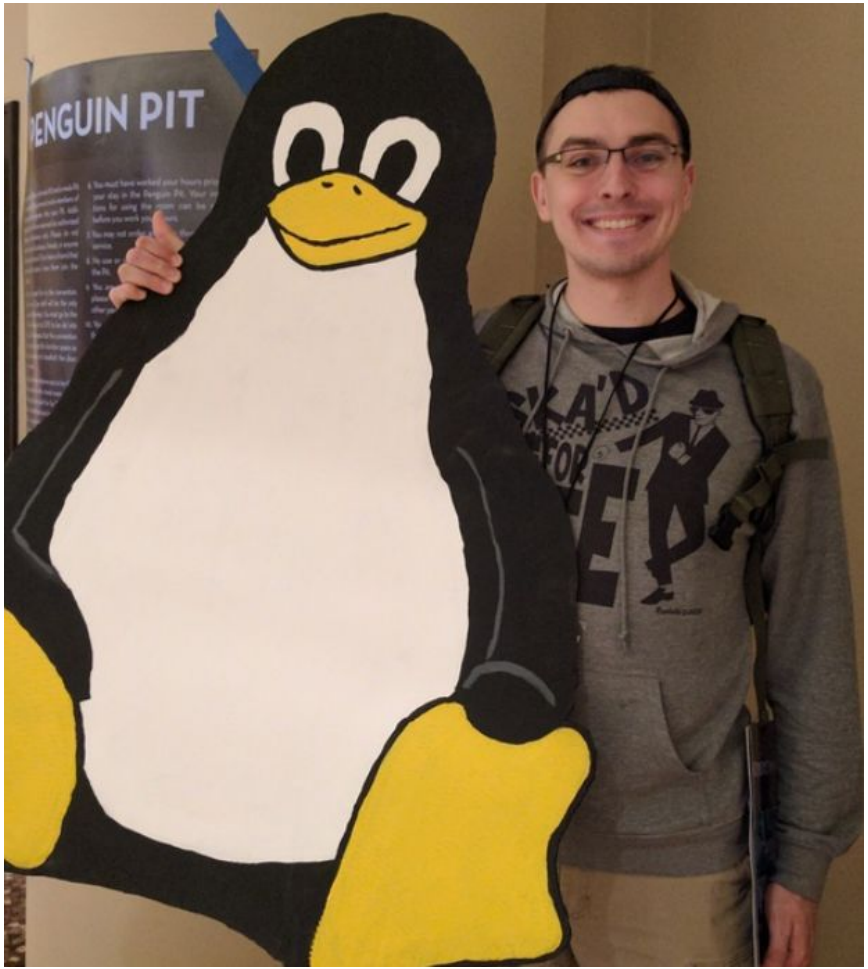
Operating CT Logs

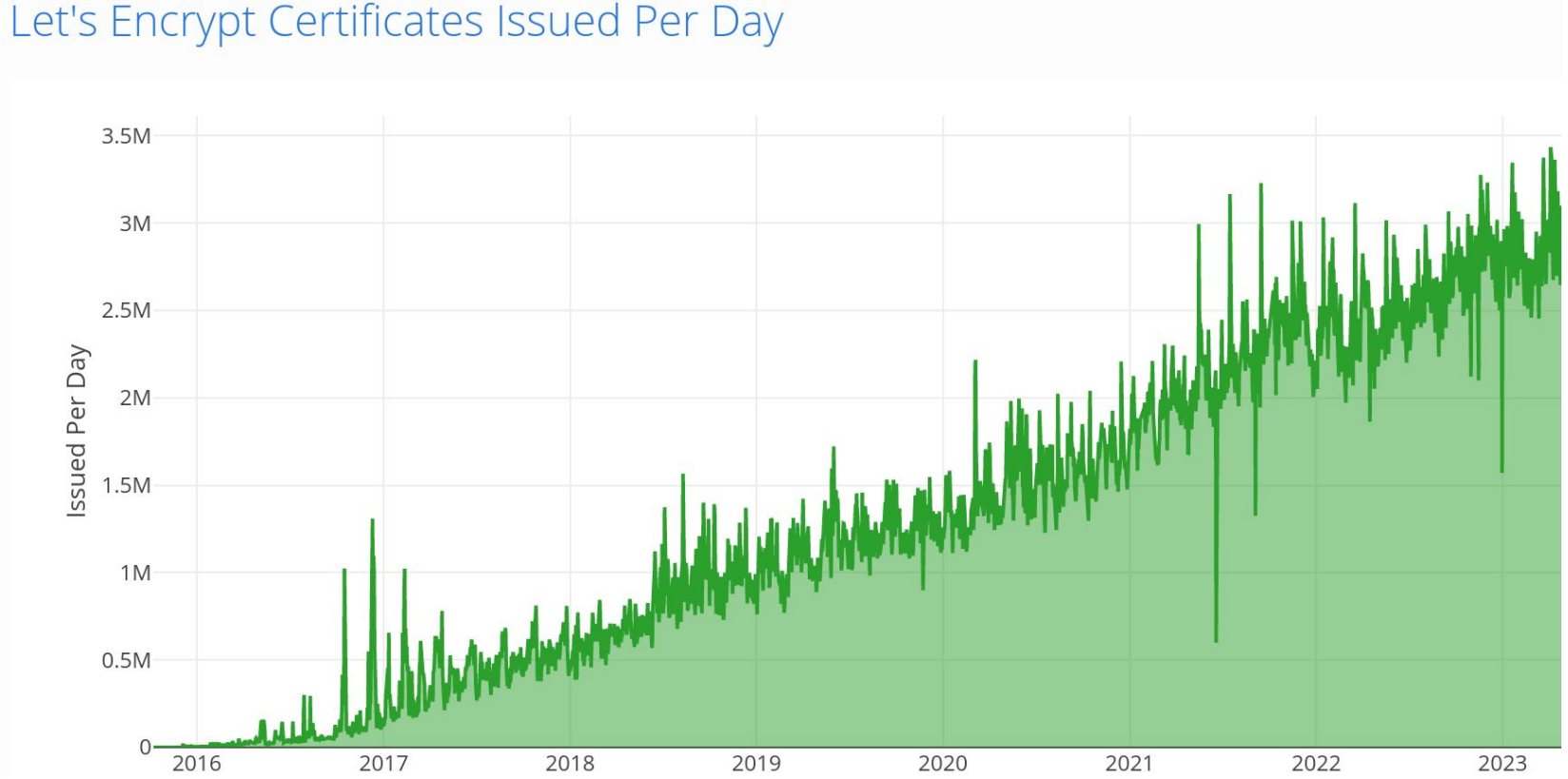
Wins and Woes

Who am I?

- phil@letsencrypt.org
- github.com/pgporada
- linkedin.com/in/philporada



What is Let's Encrypt?



Blogs we've written about CT

2019

Introducing Oak, a Free and Open Certificate Transparency Log

https://letsencrypt.org/2019/05/15/introducing-oak-ct-log.html

How Let's Encrypt Runs CT Logs

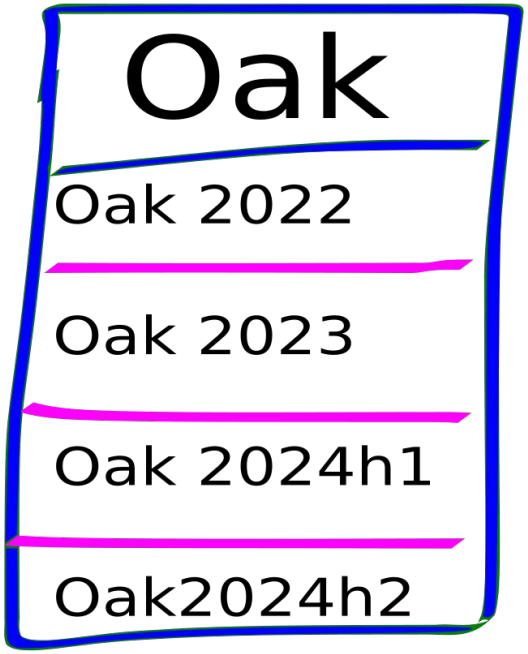
https://letsencrypt.org/2019/11/20/how-le-runs-ct-logs.html

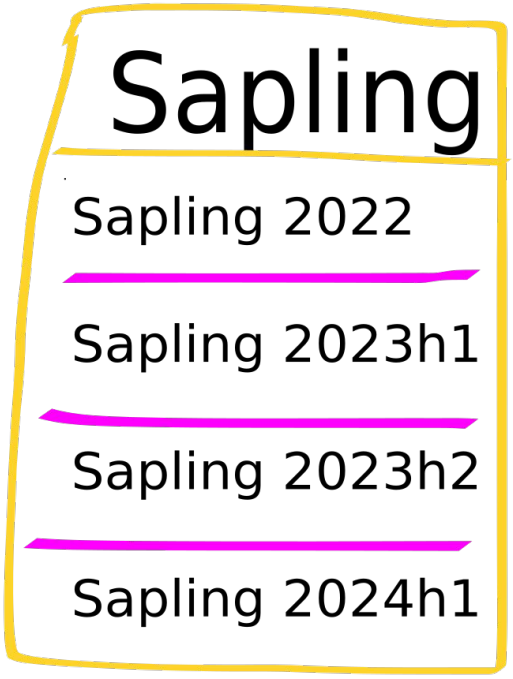
2022

Nurturing Continued Growth of Our Oak CT Log

https://letsencrypt.org/2022/05/19/nurturing-ct-log-growth.html

Logs? Shards? Oh my!





How does a CT log benefit a CA?

crt.sh Certificate Search

Criteria	ID = '8092441842'

8092441842

Precertificate

Log entries for this certificate:

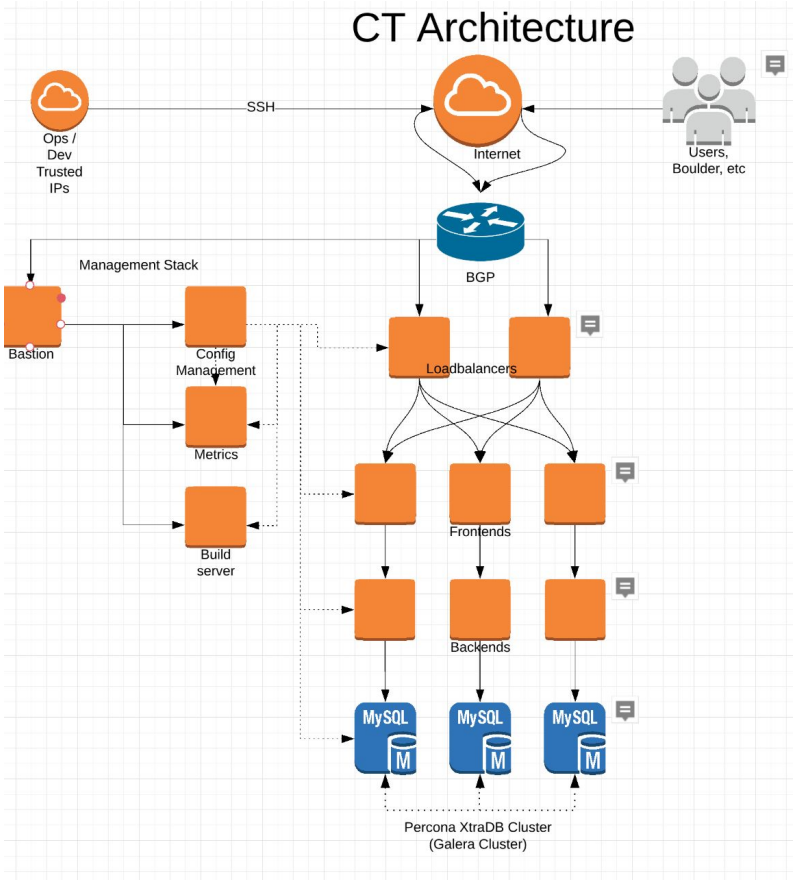
Timestamp	Entry #	Log Operator	Log URL
2022-12-01 01:55:09 UTC	389039413	Google	https://ct.googleapis.com/logs/argon2023
2022-12-01 01:55:09 UTC	222434106	Let's Encrypt	https://oak.ct.letsencrypt.org/2023
2022-12-01 01:55:09 UTC	2578	Let's Encrypt	https://oak.ct.letsencrypt.org/2024h1
2022-12-01 01:55:09 UTC	437567201	Google	https://ct.googleapis.com/logs/xenon2023

Mechanism	Provider	Status	Revocation Date	Last Observed in CRL	Last Checked (Error)
OCSP	The CA	Good	n/a	n/a	2023-04-27 19:52:23 UTC
CRL	The CA	Not Revoked	n/a	n/a	2023-04-27 17:21:44 UTC
CRLSet/Blocklist	Google	Not Revoked	n/a	n/a	n/a
disallowedcert.stl	Microsoft	Not Revoked	n/a	n/a	n/a
OneCRL	Mozilla	Not Revoked	n/a	n/a	n/a

SHA-256 670770F5932718312ED3F88679A8C9F76778D45368524FCCD81666519A206694 **SHA-1** CB1EAF2773A345D(

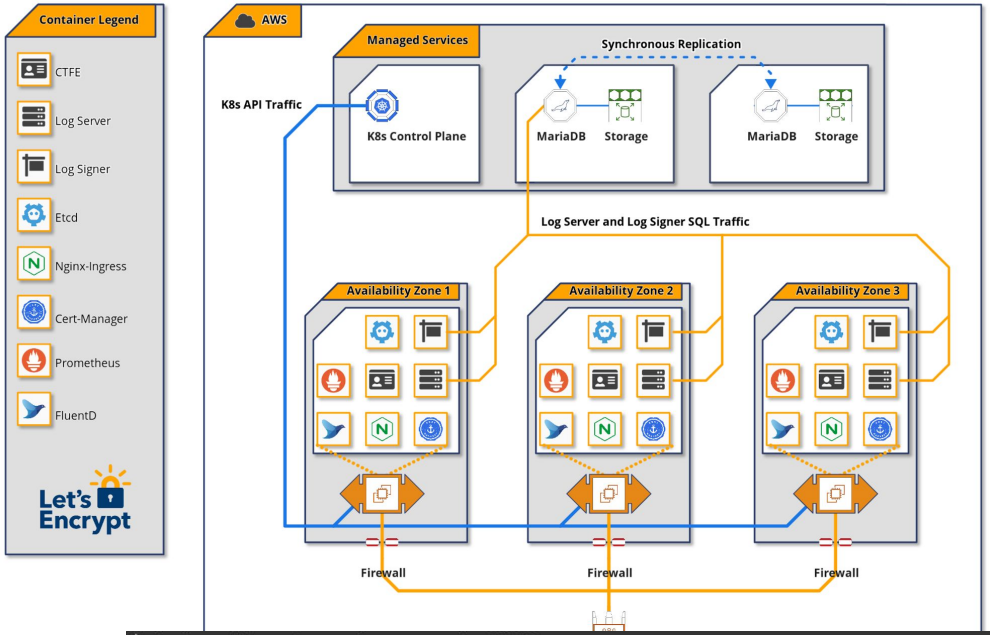
```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            02:da:37:36:34:23:55:cd:c3:21:36:95:29:05:0f:eb
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: (CA ID: 9324)
            commonName                = Amazon
            organizationalUnitName    = Server CA 1B
            organizationName          = Amazon
```

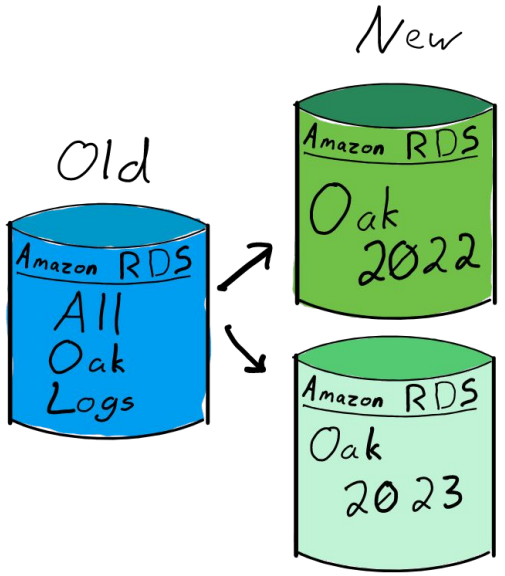
Initial Failed Architectures




```
galera:
  # Path to Galera library
  wsrep_provider: /usr/lib64/galera3/libgalera_smm.so

  # Cluster connection URL contains IPs of nodes
  # If no IP is found, this implies that a new cluster needs to be created,
  # in order to do that you need to bootstrap this node
  wsrep_cluster_address: gcomm://10.88.162.21,10.88.162.23,10.88.162.29

  # In order for Galera to work correctly binlog format should be ROW
  binlog_format: ROW
  default_storage_engine: InnoDB
  wsrep_slave_threads: 8
  wsrep_log_conflicts: NO_VAL

  # This changes how InnoDB autoincrement locks are managed and is a requirement for Galera
  innodb_autoinc_lock_mode: 2

  wsrep_node_address: {{ salt['grains.get']('ipv4')[0] }}
  wsrep_cluster_name: birch_ct
  wsrep_on: ON
  pxc_strict_mode: ENFORCING
  wsrep_sst_method: xtrabackup-v2

  # Encrypted as sstuser:GENERATEDPASSWORD
  wsrep_sst_auth: |
    -----BEGIN PGP MESSAGE-----
    -----END PGP MESSAGE-----
  skip_external_locking: NO_VAL
  wsrep_retry_autocommit: 5
  wsrep_sst_donor: birch-db-99
```

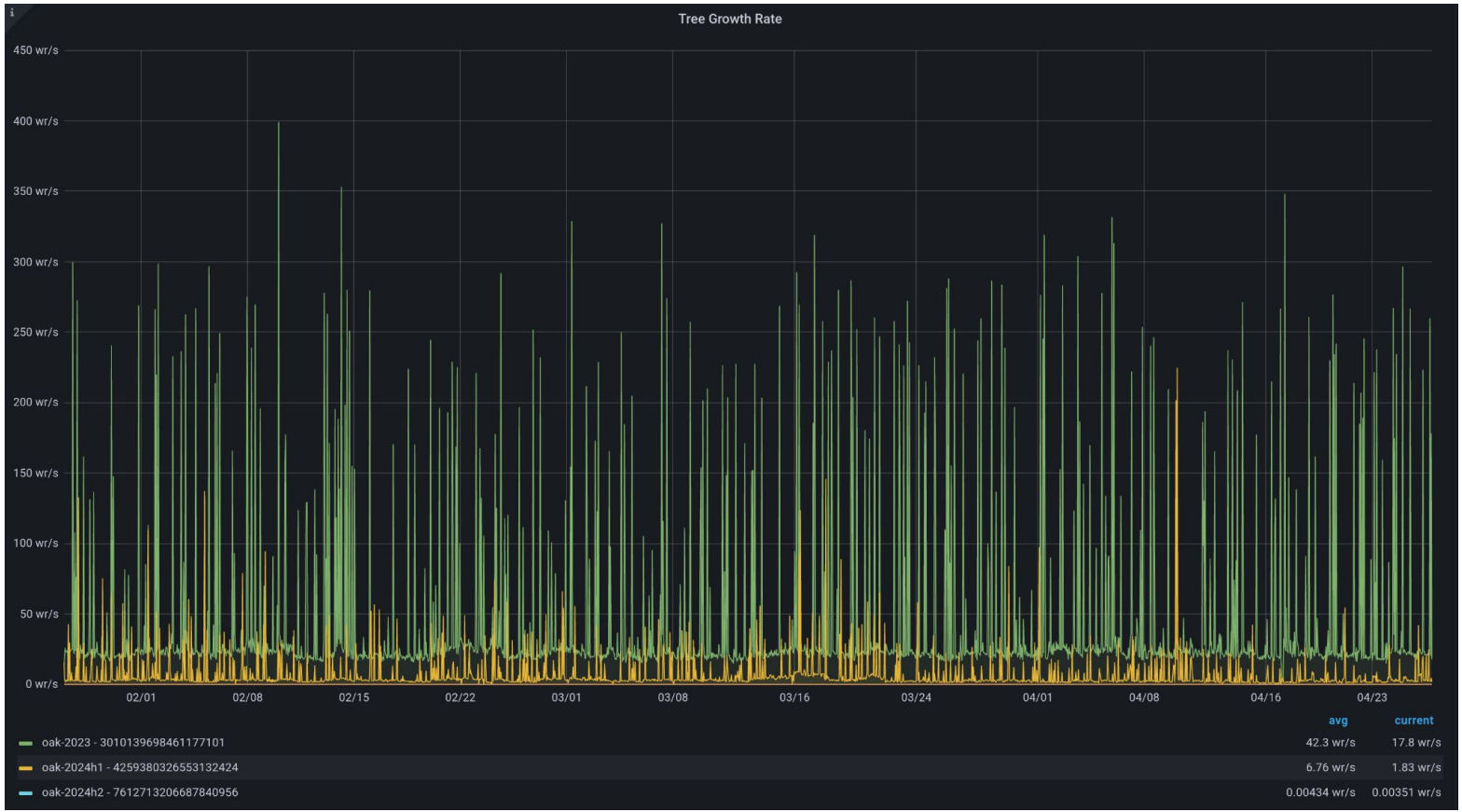
Current Functioning Architecture



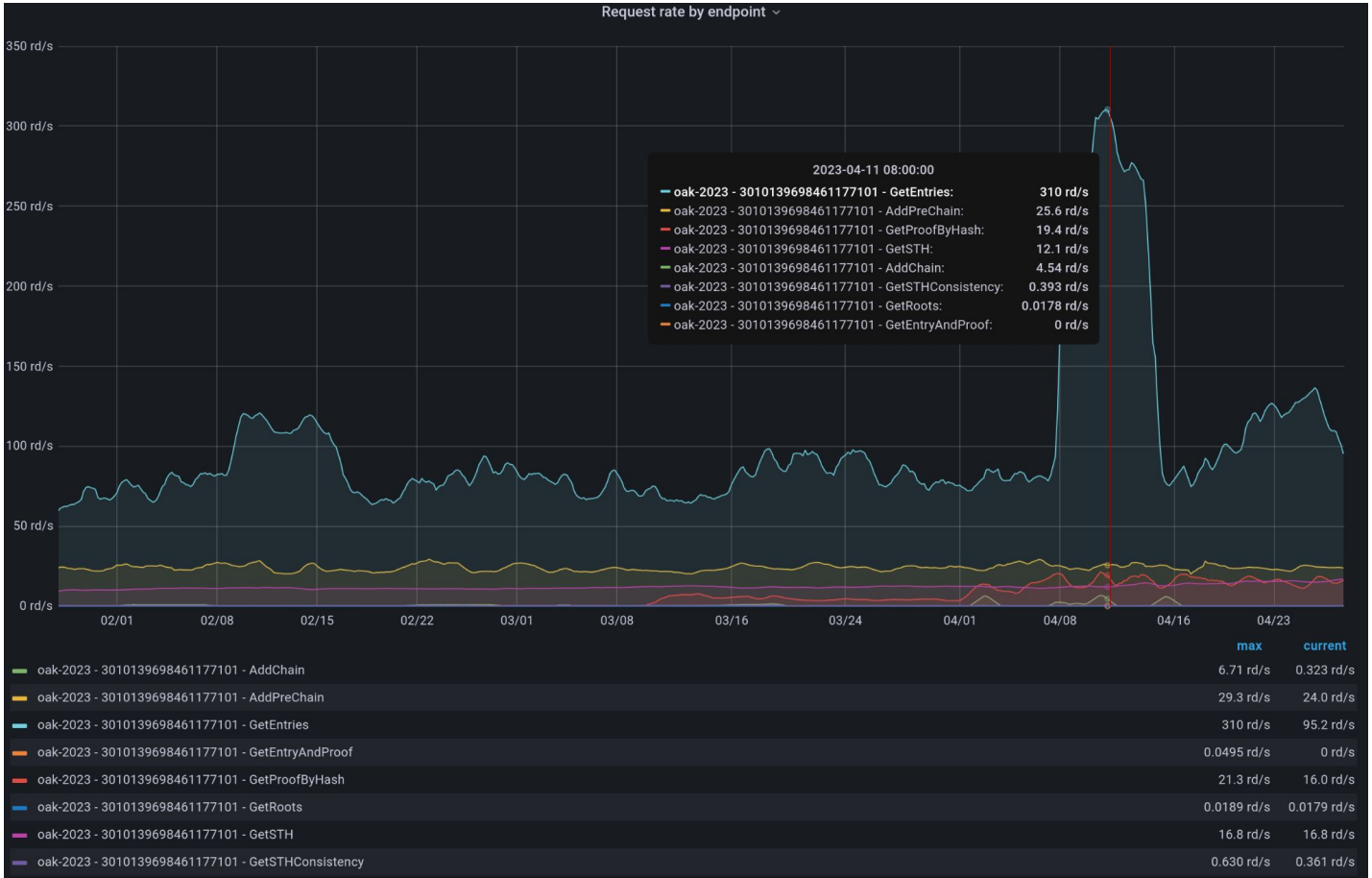



```
$ kubectl get pods -n oak-2023
NAME                                                              READY   STATUS    RESTARTS        AGE
oak-2023-etcd-operator-etcd-operator-etcd-operator-77968f6j66vs   1/1     Running   0               42m
prometheus-mysqld-exporter-deployment-65c58775f7-sp2kn            1/1     Running   2 (147d ago)    351d
trillian-ctfe-deployment-6fc9cff9d6-8f2nq                         1/1     Running   0               15md
trillian-ctfe-deployment-6fc9cff9d6-f4rjw                         1/1     Running   0               42m
trillian-etcd-cluster-jrfm7b4nqp                                  1/1     Running   0               351d
trillian-etcd-cluster-n8fjkfpht8                                  1/1     Running   0               41m
trillian-etcd-cluster-pld2rzj8q8                                  1/1     Running   0               90d
trillian-logserver-deployment-785d5c444d-b5kd8                    1/1     Running   0               21d
trillian-logserver-deployment-785d5c444d-qpj9v                    1/1     Running   0               21d
trillian-logsigner-deployment-d68cc6bf7-fnmvl                     1/1     Running   0               21d
```

Tree Growth Rate



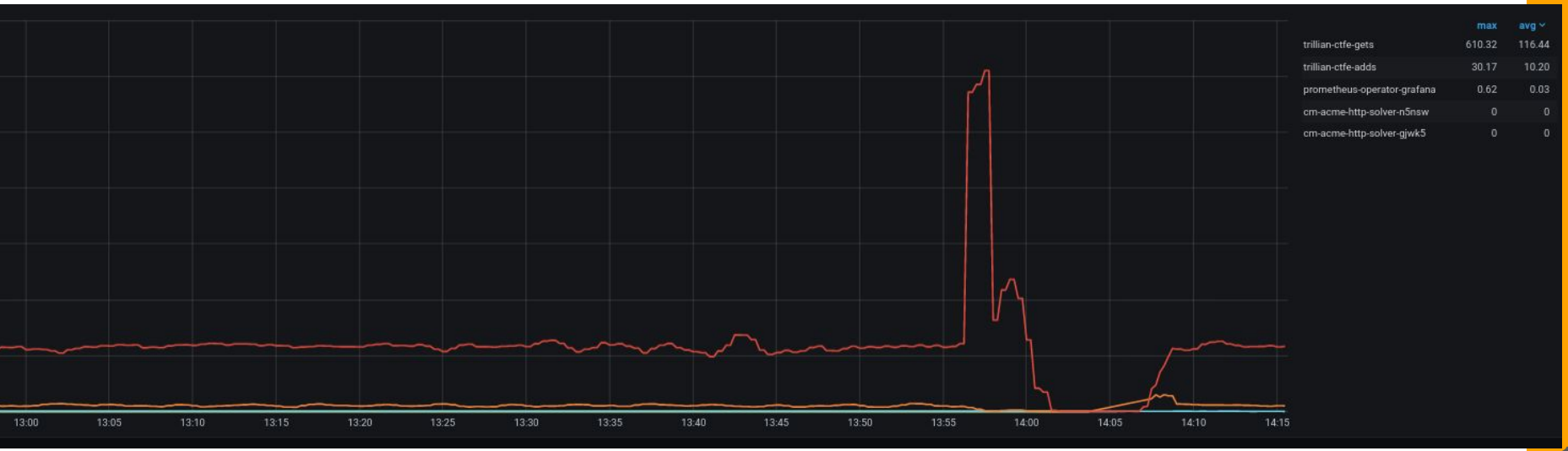
Request rate by endpoint



A Fun Incident

```
"log":"W0808 19:22:35.000186       1 tree_storage.go:81] Failed to set strict mode on mysql db: commands
out of sync. Did you run multiple statements at once?",
```

Error reproduction: github.com/koshatul/go-mysql-sync-issue



Cost of running our CT logs

Human

- 1 - 2x SREs spending ~3 months worth of time over the course of a year

Compute

- Compute nodes are basically commodity hardware.
- Storage and RAM for compute nodes is also negligible. Just enough to run applications.

Database

- This will be your pain point. More RAM allows for a bigger InnoDB buffer pool.
- Faster storage will give better log performance to a point.

Lessons Learned and Takeaways

- Have a testing log so you don't prematurely ruin your production log.
- Logs are ephemeral. When your log fails, root cause why and build a new better log with the lessons learned.
- Don’t be afraid to ask for help. The Trillian team on the GTrillian slack channel and github issues have been very kind and patient.
- Separation of concerns: run each binary in a different container, VM, or physical host. You’re after reliability.
- The log_signers (sequencers) perform an etcd election to determine which cluster member will communicate with the database. Make an alert if more than 1 cluster member has mastership for a particular shard because it will indicate a split brain scenario and cause an incident. We've been there.
- Build a rate limiting story to protect your log.
- We don't run database backups for CT logs.
- More automation is better than less. That's why we put our logs in Kubernetes.