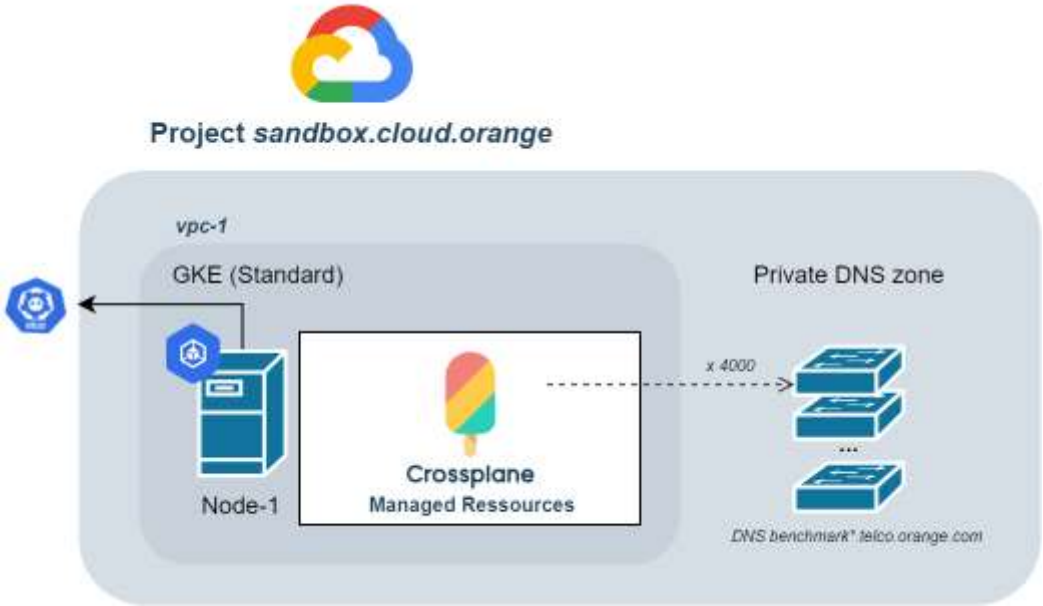
Benchmark architecture

Characteristics

- GKE cluster on *europe-west1-b*
- Node:
  - e2-highcpu-16 (16vCPU)
- Crossplane (v1.13 latest):
  - Debug enable on GCP DNS provider v0.37.0 (latest)

Points of attention

- Maximum 10000 Recordset by DNS zone
- Recordset in GCP are referenced by their name and not by UUID



Benchmark deployment

- Create 4000 Recordset in directory *out*
- *Kubectl apply –f out/*
- *Kubectl delete RecordSet.dns.gcp.upbound.io --all*

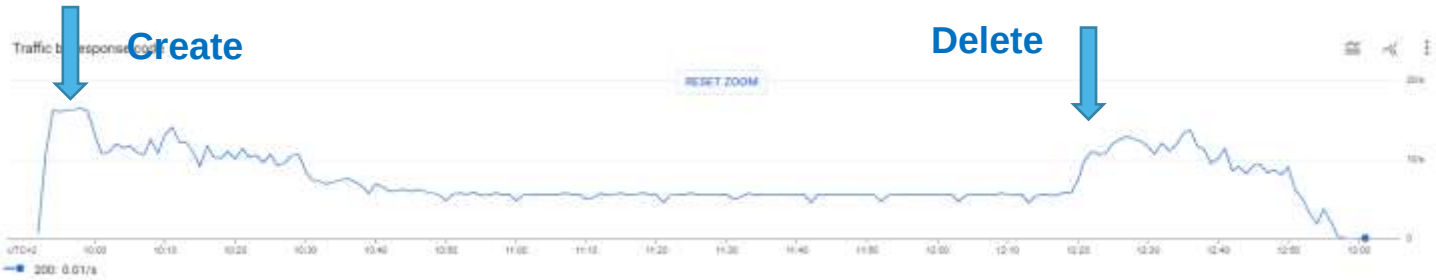
Result

- *~1 hours to have >95% recordset*
- Every resource is checked every 20 minutes
  - Delay of 20 minutes for reconciliation

```yaml
apiVersion: dns.gcp.upbound.io/v1beta1
kind: RecordSet
metadata:
  name: benchmarkeight__ID__
spec:
  forProvider:
    managedZoneSelector:
      matchLabels:
        managedzone: telco-zone-dns
    name: benchmarkeight__ID__.telco.orange.com.
    rrdatas:
      - 127.0.0.1
    ttl: 300
    type: A
```

Template RecordSet K8s

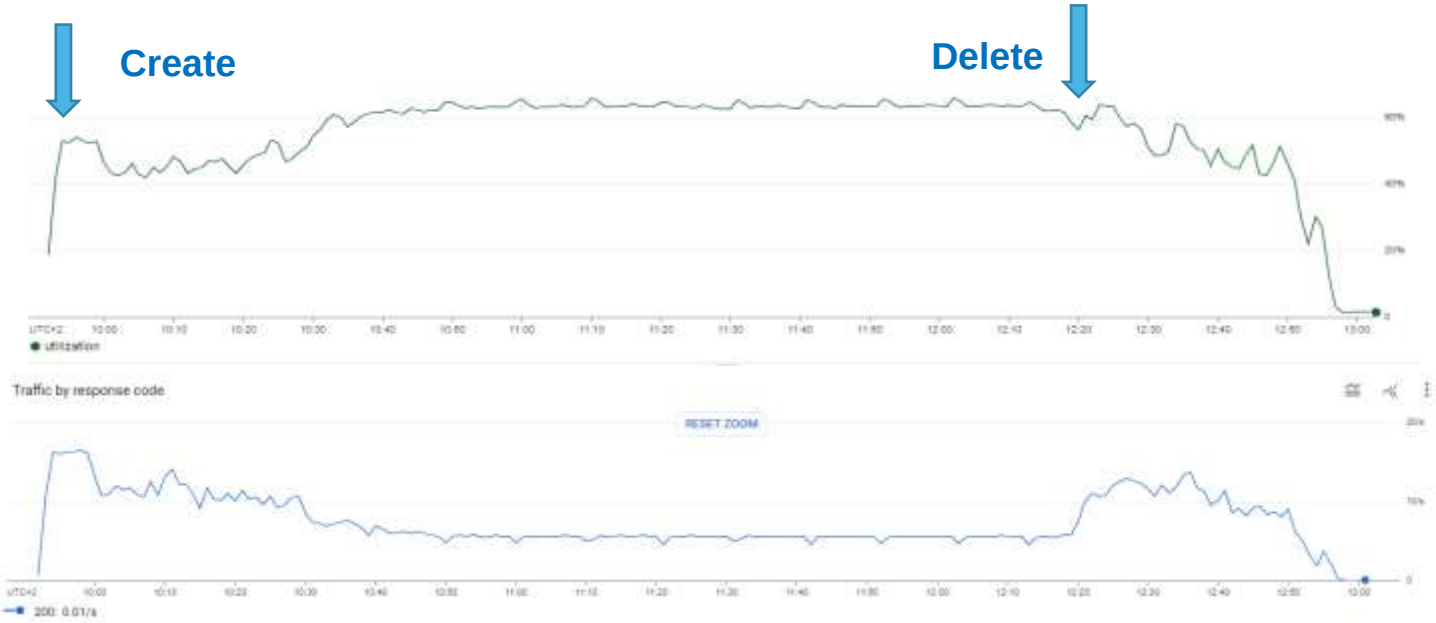
gitlab.tech.orange/smart-demo/crossplane-deployment-gcp-and-fe



Benchmark deployment

CPU Usage

- Permanent usage of 10 core (?)
- 1 Terraform process is called per API call



Benchmark deployment

Increase reconciliation

- Increased poll, sync and max-reconcile-rate by 10
- Decrease reconcile creation, speedup GCP patch
  - Poll set from 10m to 1m
  - Sync set from 1h to 10m
  - Max-reconcile-rate from 10 to 100
    - GCP resource are patched quicker
    - Deployment rate drop to 100 Recordsets/hour
  - > Maybe limited by CPU time available used at 100%
- Max reconcile rate is lock at 10

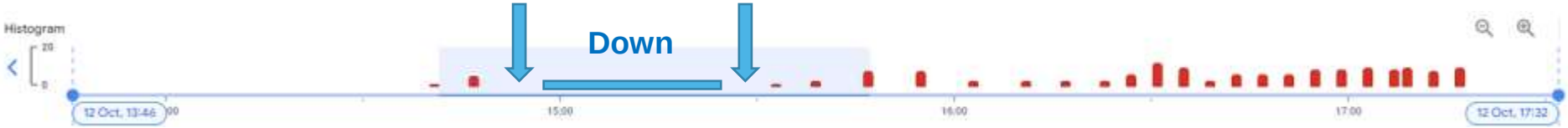
Sep 25, 2021 https://github.com/crossplane/crossplane/issues/2595#issuecomment-927030438
Maybe limited by Kubernetes:

```
45  func LimitRESTConfig(cfg *rest.Config, rps int) *rest.Config {
46          // The Kubernetes controller manager and controller-runtime controller
47          // managers use 20qps with 30 burst. We default to 10 reconciles per
48          // second so our defaults are designed to accommodate that.
49          out := rest.CopyConfig(cfg)
```

Link to LimitRESTConfig

Study n°1: Service interruption during ressource creation (12/10 at 2PM)

- Start deploying the 4000 RecordSet
- Shutdown gracefully the single node



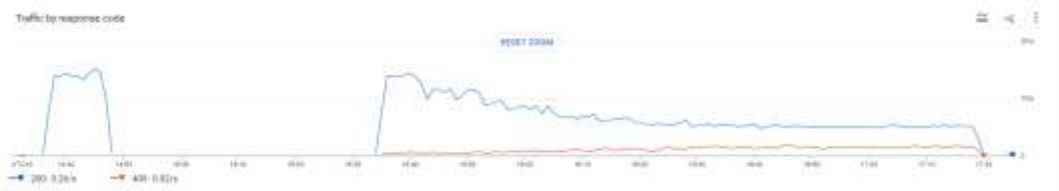
Logs of benchmarkeight339

Result

- After provider restart resource already created
- Try in loop to create the ressource

Issue

- Need to enable debug mode to have deployment information
- No fault recovery management



Study n°2: Ressource not deleted (11/10 at 4PM)

- Wait for deploy of 4000 RecordSet
- Shutdown gracefully the single node

Result

- RecordSet deleted from K8s but not from GCP
- No performance issue, CPU is 60%

Issue

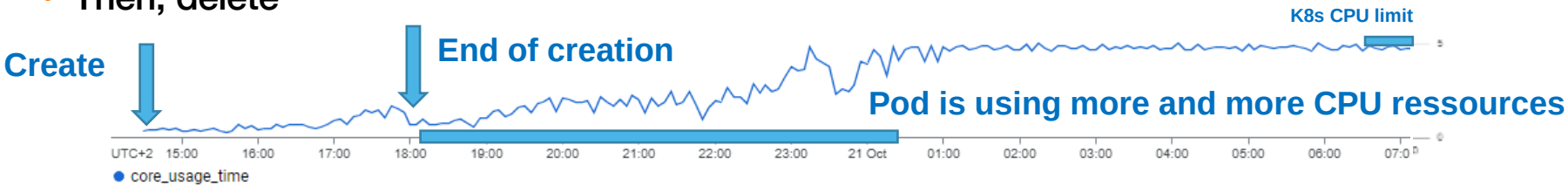
- Some resource become unmanaged by Crossplane
- No error from the log, happen randomly

Logs of benchmarksix10

Logs of benchmarksix11

Study n°3: Ressource not deleted (11/10 at 4PM)

- Wait for deploy of 2000 Bucket (1 rq/s)
- Then, delete



Logs of bucketbenchone1929 ***Logs of bucketbenchone1928***

Result

- Bucket deleted from K8s but not from GCP
- CPU is increasing gradually after the end of creation (here the CPU limit is 5)
- CPU behavior is less intense with Bucket than with RecordSet

Issue

- Delete request is not receive by GCP
- No error from the log, happen randomly