Rapport labo 4 CLD

CLD Lab 04 : Développement sur Google App Engine

Auteurs : Amir Mouti et Harun Ouweis

Groupe : GrP

Date : 18.04.2024

Tâche 0 : Configuration initiale

Aucun délivrable.

Tâche 1 : Déploiement d'une simple application web

Délivrables Tâche 1

- **Commande Maven utilisée pour le build :** La commande utilisée pour construire l'application avant son déploiement sur Google App Engine est :
-

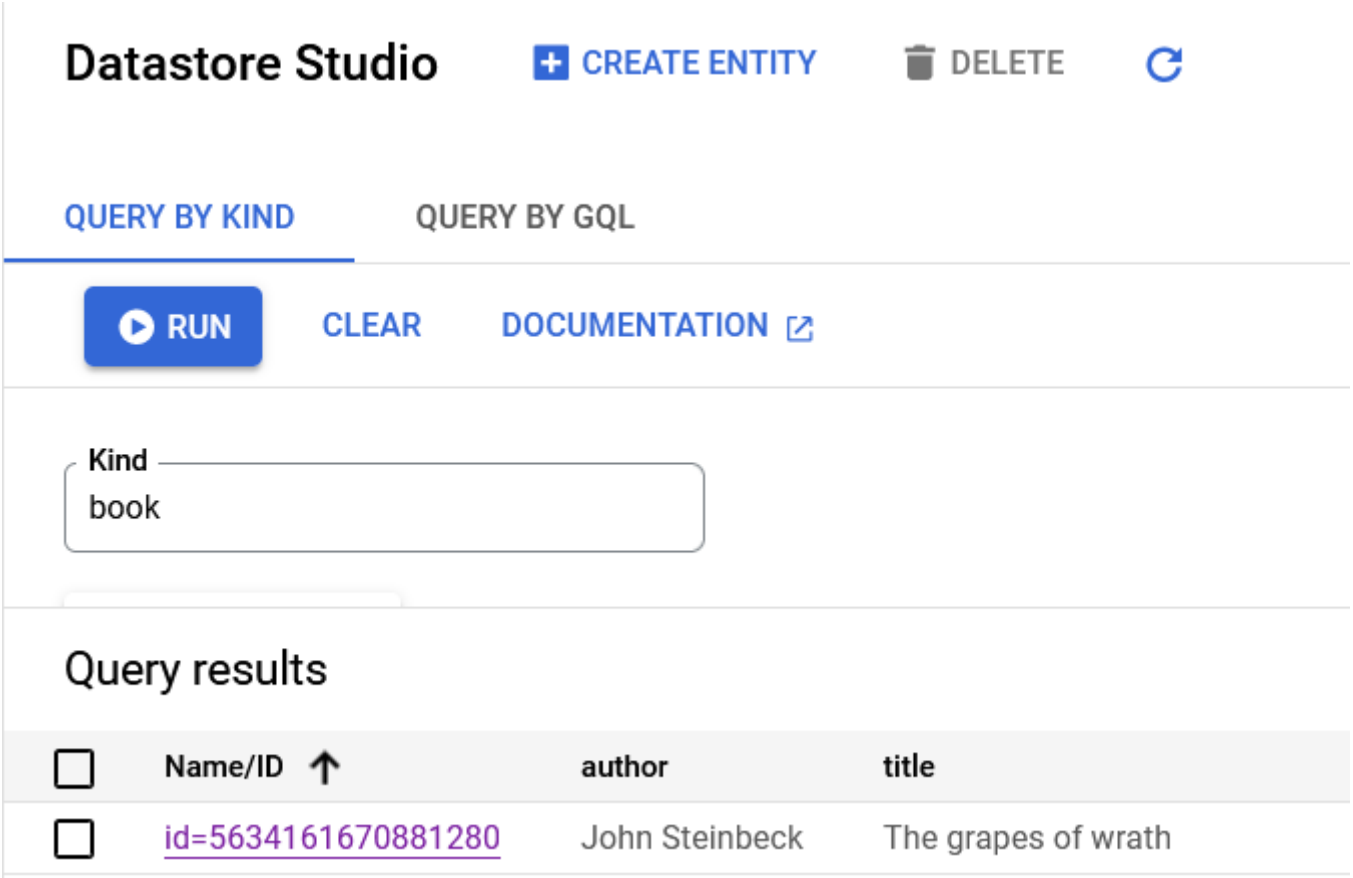
```
./mvnw clean package --batch-mode -DskipTests -Dhttp.keepAlive=false -f pom.xml --quiet
```

Cette commande a été récupérée à partir du journal de build (build log) sur la console Cloud Build de Google Cloud. Elle effectue un nettoyage (`clean`), puis emballe l'application (`package`) en mode batch sans exécuter les tests unitaires (`-DskipTests`) et en désactivant le maintien des connexions HTTP actives (`-Dhttp.keepAlive=false`), le tout en minimisant les sorties dans le terminal (`--quiet`). La mention de l'avertissement `Warning: JAVA_HOME environment variable is not set.` indique que la variable d'environnement `JAVA_HOME` n'était pas configurée lors de l'exécution, mais cela n'a pas empêché la construction de l'application.

Tâche 2 : Ajout d'un contrôleur qui écrit dans Datastore

Délivrables Tâche 2

- **Capture d'écran de Datastore Studio :** Ci-joint une capture d'écran de Datastore Studio montrant l'entité créée avec notre contrôleur. L'entité est de type `book` avec deux propriétés : `author` et `title`. Dans cet exemple, l'entité créée possède les valeurs "John Steinbeck" pour l'auteur et "The grapes of wrath" pour le titre.



(
)

Cette entité a été écrite dans Datastore en utilisant le nouveau contrôleur que nous avons intégré à notre application web. Après avoir déployé l'application sur Google App Engine et accédé au point de terminaison, l'entité a été enregistrée avec succès et est visible dans Datastore Studio, prouvant ainsi le fonctionnement correct de notre contrôleur et la connectivité avec Google Cloud Datastore.

Tâche 3 : Développement d'un contrôleur pour écrire des entités arbitraires dans Datastore

Délivrables Tâche 3

- **Liste du code de l'application :** Nous avons ajouté un contrôleur à notre application Spring Boot qui gère les écritures d'entités arbitraires dans Google Cloud Datastore. Le contrôleur extrait les données des paramètres de la requête URL et crée une entité dans Datastore en utilisant ces informations. Une vérification est effectuée pour s'assurer que le paramètre `_kind` est présent; en l'absence de ce paramètre, un message d'erreur est retourné à l'utilisateur pour indiquer que le type de l'entité est manquant. Voici le code du contrôleur ajouté :

```
  @GetMapping("/dswrite")
  public String writeEntityToDatastore(@RequestParam Map<String, String>
queryParameters) {
  // Extract the kind name and key name from the query parameters
     String kindName = queryParameters.get("_kind");
  if (kindName == null || kindName.isEmpty()) {
   return "Error: '_kind' parameter is missing in the request.";
  }
```

```
      String keyName = queryParameters.get("_key");

  // Create or retrieve the KeyFactory for the specified kind
  KeyFactory keyFactory = datastore.newKeyFactory().setKind(kindName);

  // Create a key for the entity; if a key name is provided use it, otherwise
allocate an ID
  Key key = (keyName != null) ? keyFactory.newKey(keyName) :
datastore.allocateId(keyFactory.newKey());

  // Build the entity using the query parameters
  Entity.Builder entityBuilder = Entity.newBuilder(key);
  for (Map.Entry<String, String> entry : queryParameters.entrySet()) {
   if (!entry.getKey().equals("_kind") && !entry.getKey().equals("_key")) {
    entityBuilder.set(entry.getKey(), entry.getValue());
   }
  }

  Entity entity = entityBuilder.build();

  // Save the entity to Datastore
  datastore.put(entity);

  return "Entity with kind '" + kindName + "' and key id/name '" + (keyName !=
null ? keyName : key.getId()) + "' has been written to Datastore";
}
```

Tâche 4: test de la performance des écritures dans le datastore

Délivrables Tâche 4

controller helloworld

Chart settings

Requests by type

✓ 1 hour | 6 hours | 12 hours | 1 day | 2 days | 4 days | 7 days | 14 days | 30 days

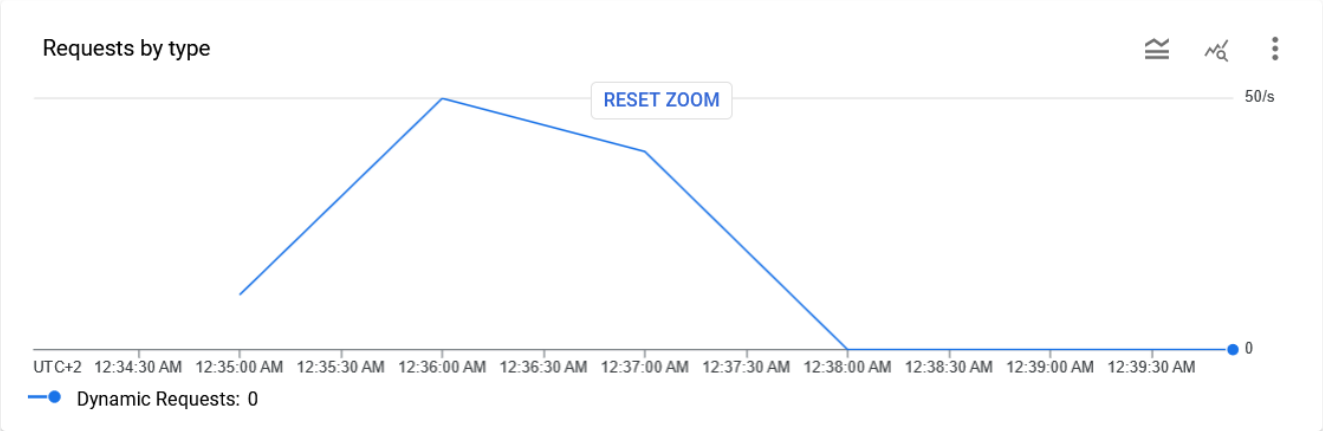


Chart settings

Instances

✓ 1 hour | 6 hours | 12 hours | 1 day | 2 days | 4 days | 7 days | 14 days | 30 days

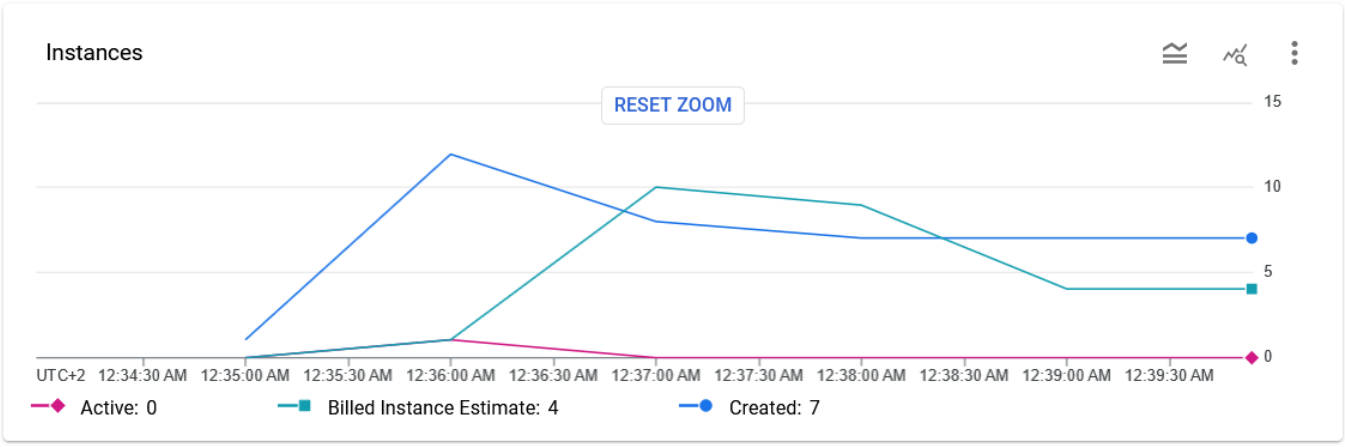
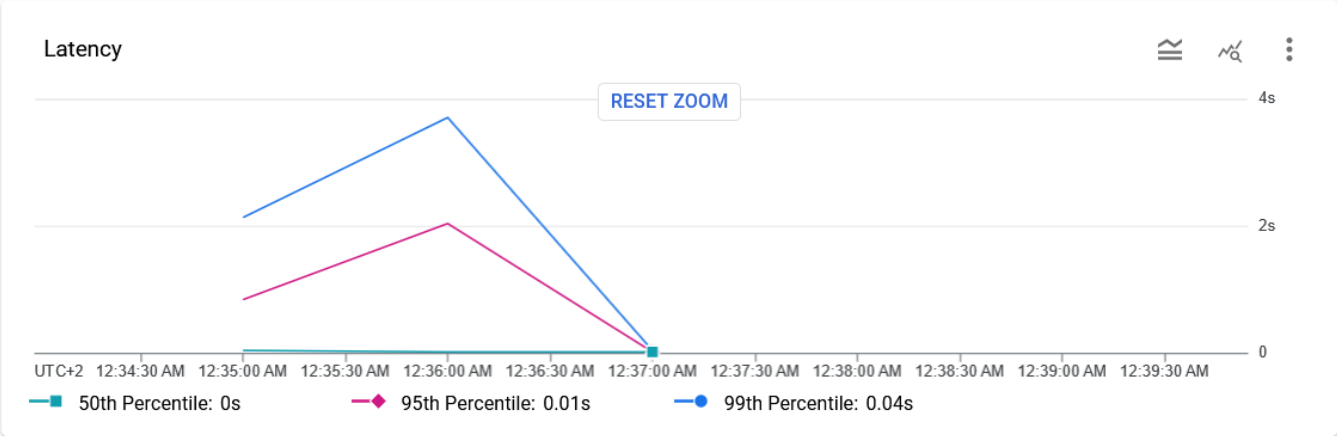
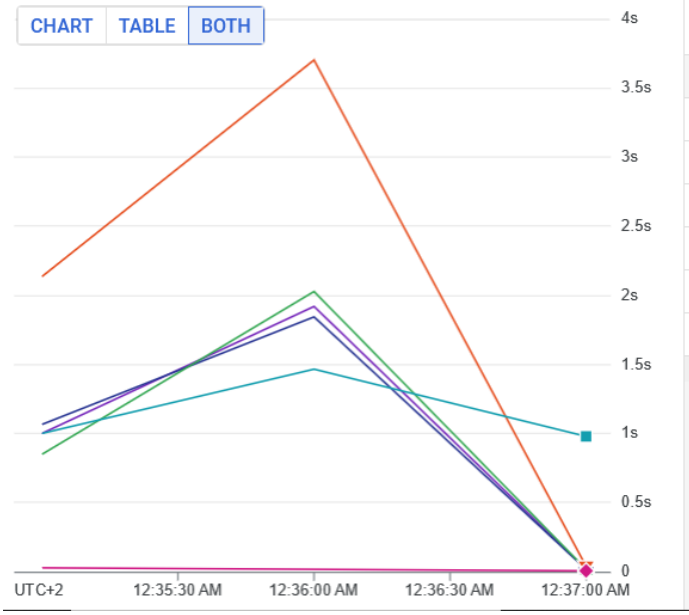


Chart settings

Latency

1 hour | 6 hours | 12 hours | 1 day | 2 days | 4 days | 7 days | 14 days | 30 days

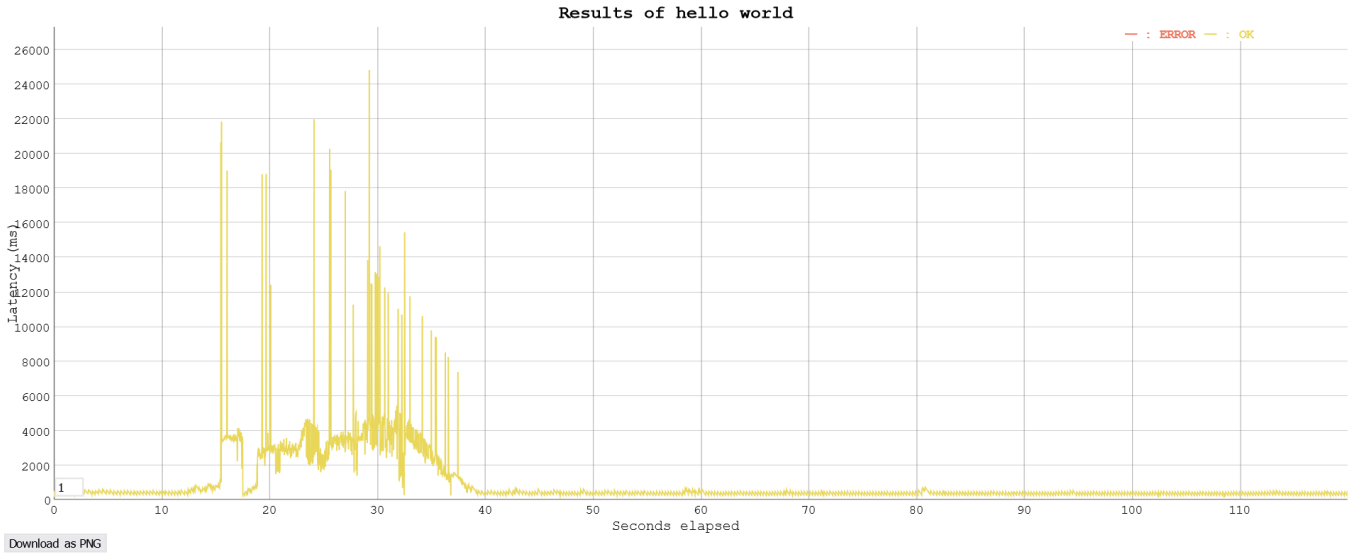




Filter Enter property name or value

Name	All Values	Value
5 Minute Moving Average	-	979.82 ms
5th Percentile	0s	4.457 ms
95th Percentile	3.7s	41.714 ms
Mean	0.98s	19.83 ms
Median	0.04s	13.318 ms
Standard Deviation	1.34s	19.463 ms

```
amir@LAPTOP-K4DBKGM7:/mnt/c/dev/CLD/lab04$ echo "GET https://lab04-420711.uc.r.appspot.com/" | ./vegeta attack -duration
=120s -rate=50 | tee results_hello_50.bin | ./vegeta report
Requests      [total, rate, throughput]         6000, 50.01, 49.81
Duration      [total, attack, wait]             2m0s, 2m0s, 456.592ms
Latencies     [min, mean, 50, 90, 95, 99, max]  180.21µs, 937.271ms, 414.734ms, 3.01s, 3.637s, 4.78s, 24.846s
Bytes In      [total, mean]                     77987, 13.00
Bytes Out     [total, mean]                     0, 0.00
Success       [ratio]                           99.98%
Status Codes  [code:count]                      0:1  200:5999
Error Set:
Get "https://lab04-420711.uc.r.appspot.com/": dial tcp 0.0.0.0:0->[2a00:1450:400a:803::2014]:443: connect: network is un
reachable
```



controller dswrite

Chart settings

Requests by type

✓ 1 hour | 6 hours | 12 hours | 1 day | 2 days | 4 days | 7 days | 14 days | 30 days

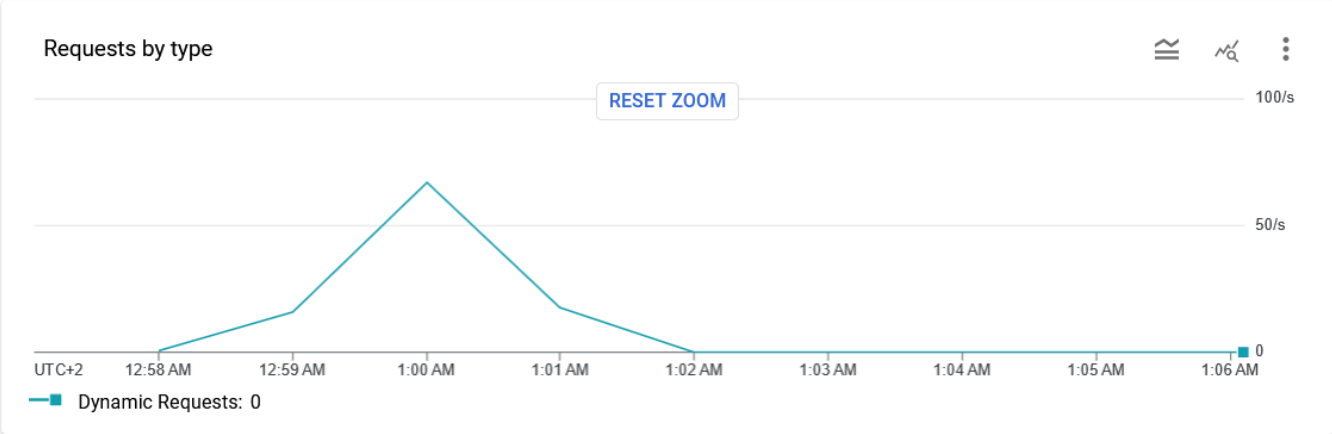


Chart settings

Instances

✓ 1 hour | 6 hours | 12 hours | 1 day | 2 days | 4 days | 7 days | 14 days | 30 days

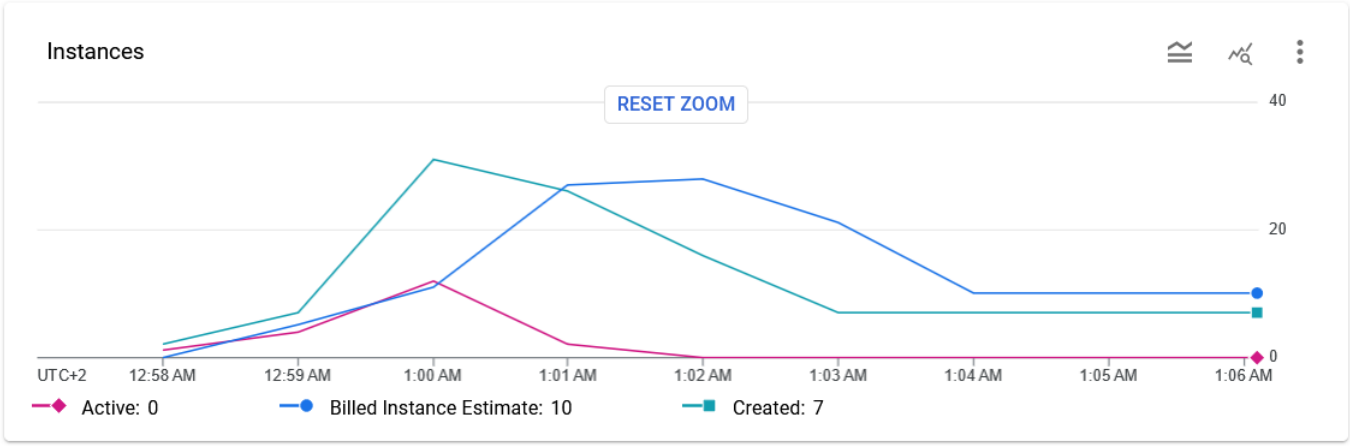
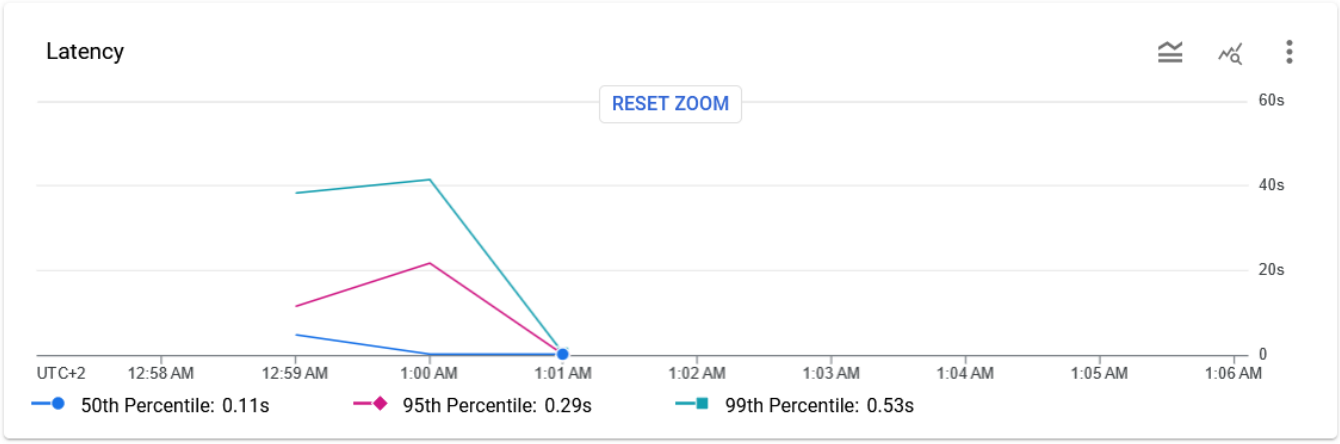
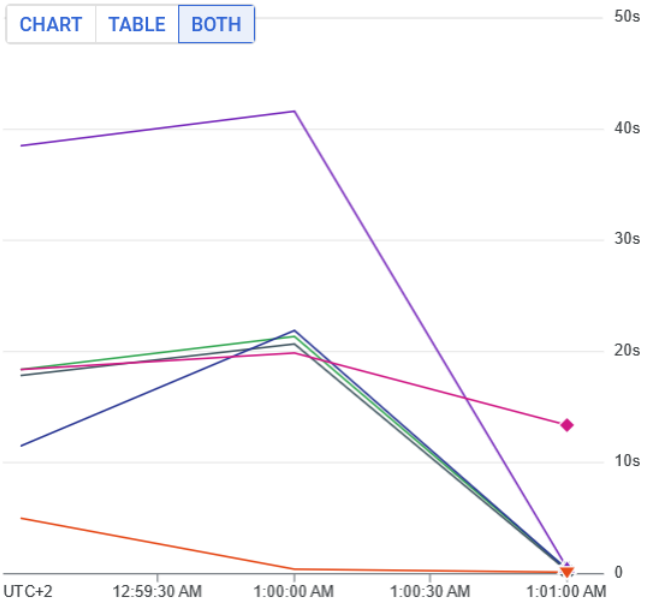


Chart settings

Latency

✓ 1 hour | 6 hours | 12 hours | 1 day | 2 days | 4 days | 7 days | 14 days | 30 days

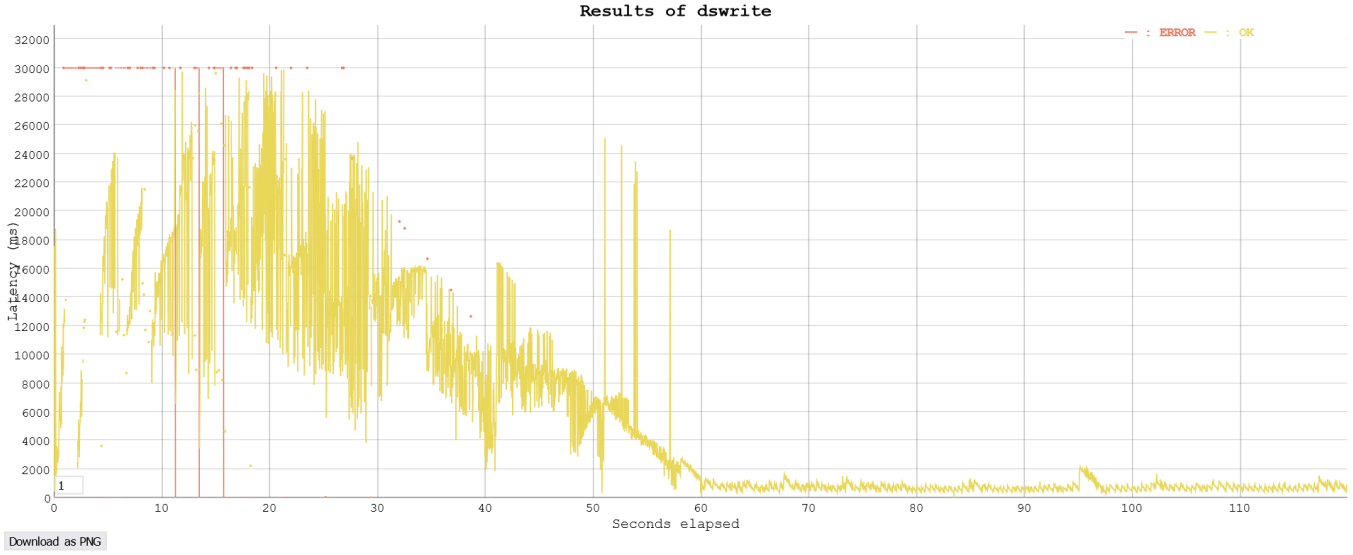




Filter Enter property name or value

Name	All Values	Value
5 Minute Moving Average	-	13.263 s
5th Percentile	0.11s	111.953 ms
95th Percentile	41.5s	534.723 ms
Mean	13.26s	312.068 ms
Median	4.94s	289.527 ms
Standard Deviation	16.77s	212.284 ms

```
amir@LAPTOP-K4DBKGM7:/mnt/c/dev/CLD/lab04$ echo "GET https://lab04-420711.uc.r.appspot.com/dswrite?_kind=akjdshkhdkjsah&
content=yes" | ./vegeta attack -duration=120s -rate=50 | tee results_dswrite.bin | ./vegeta report
Requests      [total, rate, throughput]         6000, 50.01, 46.95
Duration      [total, attack, wait]             2m1s, 2m0s, 817.59ms
Latencies     [min, mean, 50, 90, 95, 99, max]  143.571µs, 7.191s, 1.537s, 21.229s, 29.28s, 30.001s, 30.001s
Bytes In      [total, mean]                     2491946, 415.32
Bytes Out     [total, mean]                     0, 0.00
Success       [ratio]                           94.53%
Status Codes  [code:count]                      0:322  200:5672  500:6
Error Set:
Get "https://lab04-420711.uc.r.appspot.com/dswrite?_kind=akjdshkhdkjsah&content=yes": dial tcp 0.0.0.0:0->[2a00:1450:400
a:808::2014]:443: connect: network is unreachable
Get "https://lab04-420711.uc.r.appspot.com/dswrite?_kind=akjdshkhdkjsah&content=yes": net/http: request canceled (Client
.Timeout exceeded while awaiting headers)
Get "https://lab04-420711.uc.r.appspot.com/dswrite?_kind=akjdshkhdkjsah&content=yes": context deadline exceeded
Get "https://lab04-420711.uc.r.appspot.com/dswrite?_kind=akjdshkhdkjsah&content=yes": context deadline exceeded (Client.
Timeout exceeded while awaiting headers)
500 Internal Server Error
```



What average response times do you observe in the test tool for each controller?:

Environ 0.9 secondes pour le controlleur hello world et environ 7 secondes pour le dswrite

Compare the response times shown by the test tool and the App Engine console. Explain the difference:

For the hello world controller, the stats in the google app engine and the vegeta report are very similar.

Pour le controlleur helloworld, les statistiques sur la console google app et dans le rapport de vegeta ont l'air d'être plus ou moins les mêmes.

Par contre, pour dswrite, on observe que la moyenne des latences est beaucoup plus grande sur la console google app que sur dans le rapport de vegeta (13.26 secondes sur la console et 7.19 secondes chez vegeta). On voit aussi que la console a des données avec des latences plus haute que vegeta. Dans le rapport de vegeta, la latence max est 30 secondes alors que sur google app le 95e percentile est à 41.5 secondes.

Cette différence est probablement car vegeta traite les requètes qui prennent trop de temps comme des erreurs et elles n'apparaissent donc pas dans son rapport alors que la console google app les garde car du côté serveur la requète a été traitée. On voit que dans le rapport vegeta, 5.5% des requêtes ont donné des

erreurs et dans les erreurs que vegeta a rencontré on voit des erreurs "client timeout exceeded" ce qui supporte cette explication.

Let’s suppose you become suspicious that the algorithm for the automatic scaling of instances is not working correctly. Imagine a way in which the algorithm could be broken. Which measures shown in the console would you use to detect this failure?

- Si l'algorithme crée trop d'instances: on peut détecter cette erreur en monitorant l'utilisation du CPU et le nombre d'instances. Si l'algorithme garde beaucoup d'instances en vie mais qu'elles n'utilisent que peu leur CPU, elles ne sont pas nécessaires.
- L'algorithme pourrait créer trop peu d'instances : nous pouvons détecter cela en regardant la latence, les taux d'erreur et le nombre d'instances. S'il y a beaucoup d'erreurs timeout et que le nombre d'instances n'augmente pas il est possible que l'algorithme ne réagit pas correctement à une augmentation de trafic
- Réaction lente aux changements de trafic : nous pouvons détecter cela en observant le nombre de requêtes par seconde et le nombre d'instances (si les instances augmentent trop lentement).