Cloud Batchについて

期待する役割

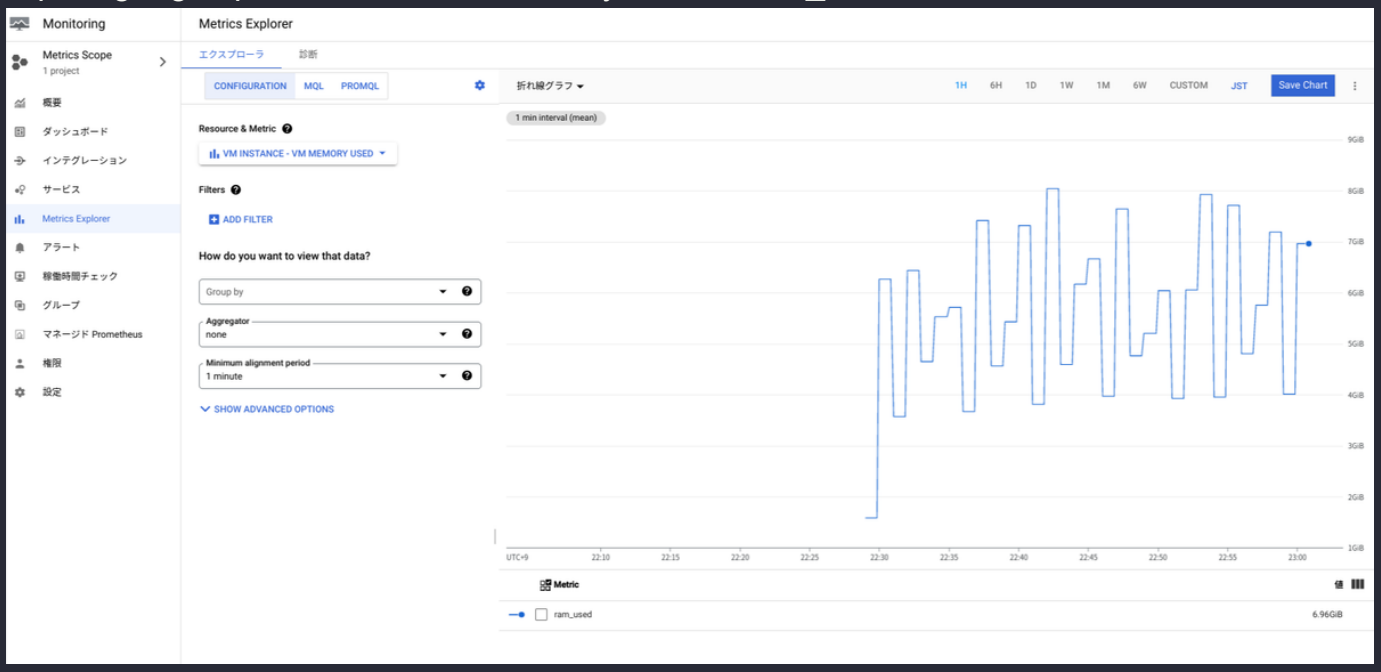
- コンテナ実行基盤としての役割
- jobの実行・ステータスの管理・リソースの確保 などを担ってもらう

使い方

基本的な使い方は以下の通り。

計算リソースについて

- ジョブごとに VM が作成され、ジョブが完了したら削除される。 並列化が効いている場合には、グループインスタンスによって複数の VM が起動する。
- CPU やメモリ等のリソース利用状況を細かく見たければ、Compute Engine のダッシュボードではなく、Cloud Monitoring を使う。 よく見るのは以下のような指標。結構わかりやすいので、良さそう。UI側の提供はリンクの生成のみで良さそう。
  - compute.googleapis.com/instance/cpu/utilization
  - compute.googleapis.com/instance/memory/balloon/ram_used



jobの作成に使用する`job.json`の定義は以下を参照。これを見ると、細かく設定ができる。例えば、タスクを並列するのか、ログをどこに吐くのかなど。

- *REST Resource: projects.locations.jobs | Batch | Google Cloud*

用意されているAPIは以下の通り。これを見る限り、jobの実行はこのAPIからはできなさそう!?

- *APIs and reference | Batch | Google Cloud*

VMにexternal IPを付与しないために、`NetworkInterface` の `noExternalIpAddress` は true にする必要がある。この場合、Google Service と連携するために、Cloud NATなどの設定が必要になる（*Configure Private Google Access | VPC | Google CloudUse Public NAT with Compute Engine | Google Cloud* を参照）。

- *REST Resource: projects.locations.jobs | Batch | Google Cloud*

TODO

- CPUやメモリのリソース利用状況について、バッチが終了すればVMは削除されるが、その場合過去のものでも確認することはできるのか？
  - 理想: 実行後も確認できるようにしたい。

困ったこと

- `gcloud batch jobs submit` でjobを作成できるが、作成すると同時に必ず実行される。これを回避する引数はなさそう。
- Cloud Batchでは作成したbatchはその場で実行され、再実行はできない。再実行するためには、Cloud Workflowsと組み合わせる必要がある。
  - *Batch と Cloud Run Jobs ってどっち使えばいいの？ - Batch 編 -*
    - > Batch との連携の実態はワークフロー定義に Batch のジョブ定義を記述しておき、Worflows を実行する度に Batch ジョブを生成しているようなイメージです。
    - とあり、毎回Batchジョブを作る。作る単位 = 実行単位ということ。
- jobのstatusが、`scheduled` から変更されない。`e2-micro`を選択したため？
  - *Job creation and execution overview | Batch | Google Cloud*

実行結果

成功

成功した場合 (cloud batchの`LOGS`より, cloud loggingから見れない？)

Status	Tasks running	Tasks succeeded
Succeeded	0	1

DETAILS EVENTS LOGS

Severity: Default

Filter: Search all fields and values

SEVERITY	TIMESTAMP	SUMMARY
	2023-10-29 00:16:52.514 JST	us-central1-docker.pkg.dev/haru256-schedule-panel/batch/sample-batch:latest
i	2023-10-29 00:16:52.520 JST	Task action/STARTUP/0/0/group0 runnable 1 exited with status 0
i	2023-10-29 00:16:52.520 JST	Task action/STARTUP/0/0/group0 background runnables all exited on their own.
i	2023-10-29 00:16:52.520 JST	Task action/STARTUP/0/0/group0 succeeded
i	2023-10-29 00:16:52.520 JST	Scheduler reported task "action/STARTUP/0/0/group0"
i	2023-10-29 00:16:52.520 JST	report agent state: metadata:{parent:"projects/268399420571/locations/us-centr
i	2023-10-29 00:16:52.643 JST	Server response for instance 8447406148378729322: tasks:{task:"action/STARTUP/
i	2023-10-29 00:16:52.644 JST	Executing runnable container:{image_uri:"us-central1-docker.pkg.dev/haru256-sc
i	2023-10-29 00:16:55.854 JST	Hello World from main.py
i	2023-10-29 00:16:55.854 JST	1.26.1
i	2023-10-29 00:16:56.017 JST	Task task/sample-batch-4eeaf202-8fa2-41b6-b983-0-group0-0/0/0 runnable 0 exite
i	2023-10-29 00:16:56.017 JST	Task task/sample-batch-4eeaf202-8fa2-41b6-b983-0-group0-0/0/0 background runna
i	2023-10-29 00:16:56.017 JST	Task task/sample-batch-4eeaf202-8fa2-41b6-b983-0-group0-0/0/0 succeeded
i	2023-10-29 00:16:56.017 JST	Scheduler reported task "task/sample-batch-4eeaf202-8fa2-41b6-b983-0-group0-0/

失敗

docker image の起動失敗について (cloud loggingからしかわからない？)

2023-10-28 23:52:56.179 WARNING: The requested image's platform (linux/arm64/v8) does not mat

2023-10-28 23:52:56.603 exec /usr/local/bin/python: exec format error

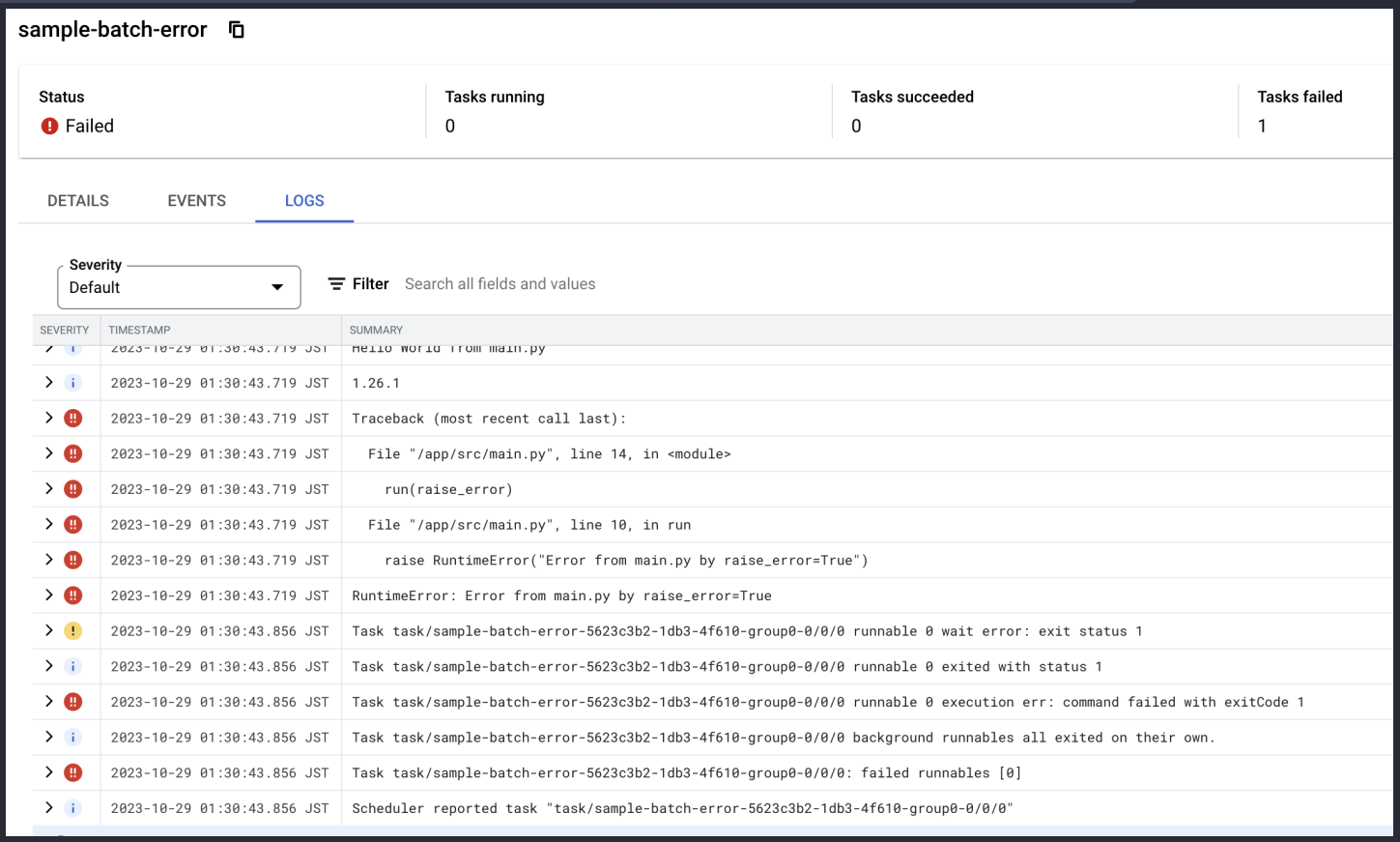
Copy | Similar entries | Expand nested fields | Hide log summary


```
{
  insertId: "xavf0bf2ji2xt"
  labels: {3}
  logName: "projects/haru256-schedule-panel/logs/batch_task_logs"
  receiveTimestamp: "2023-10-28T14:52:56.805499803Z"
  resource: {2}
  severity: "ERROR"
  textPayload: "exec /usr/local/bin/python: exec format error"
  timestamp: "2023-10-28T14:52:56.603187656Z"
}
```


2023-10-28 23:52:56.729 6574406614656261839 Task task/sample-batch-308b37b9-e256-4ef8-8

docker image 起動後の失敗（以下のようなURLを表示するだけでも良さそう。）

`https://console.cloud.google.com/batch/jobsDetail/regions/us-central1/jobs/sample-batch-error/logs?project=haru256-schedule-panel`

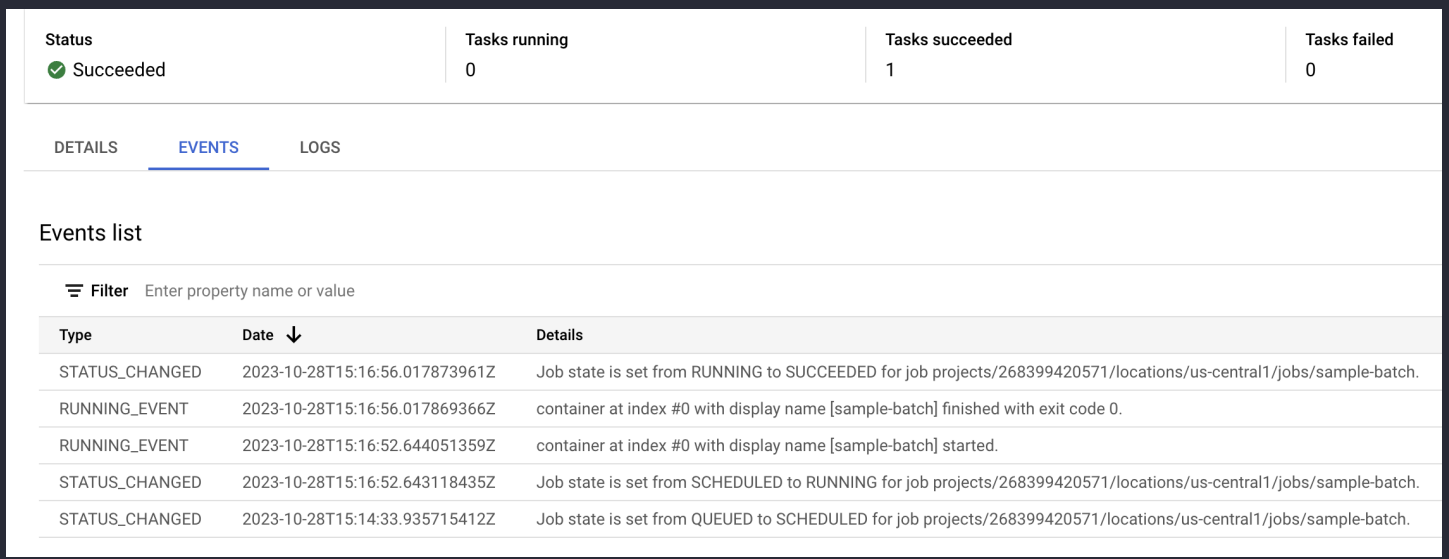


コメント

- VMにGCSをマウントすることができるので、ログをGCSに吐き出させれば、それを取得してうまく表示することもできるかもしれない。

JobのStatusについて

以下のような形でステータスの変化のログはわかる。



gcloud CLI からもstatusなどの変化ログが記載されている。以下のAPIからもjobの状態は確認できそう。

```
curl https://batch.googleapis.com/v1/projects/haru256-schedule-panel/locations/us-central1/jobs/sample-batch
```

```
haru256 in schedule-panel/apps/cloud-batch on main [!?] on haru256-schedule-panel
λ  gcloud batch jobs describe projects/haru256-schedule-panel/locations/us-central1/jobs/sample-batch
allocationPolicy:
  instances:
  - policy:
      machineType: e2-standard-2
  labels:
    batch-job-id: sample-batch
  location:
    allowedLocations:
    - regions/us-central1
    - zones/us-central1-a
    - zones/us-central1-b
    - zones/us-central1-c
    - zones/us-central1-f
  network:
    networkInterfaces:
    - network: projects/haru256-schedule-panel/global/networks/batch
      noExternalIpAddress: true
      subnetwork: projects/haru256-schedule-panel/regions/us-central1/subnetworks/batch
  serviceAccount:
    email: 268399420571-compute@developer.gserviceaccount.com
createTime: '2023-10-28T15:14:30.739752709Z'
logsPolicy:
  destination: CLOUD_LOGGING
name: projects/haru256-schedule-panel/locations/us-central1/jobs/sample-batch
status:
  runDuration: 3.374755526s
  state: SUCCEEDED
  statusEvents:
  - description: Job state is set from QUEUED to SCHEDULED for job projects/268399420571/locations/us-central1/jobs/sample-batch.
    eventTime: '2023-10-28T15:14:33.935715412Z'
    type: STATUS_CHANGED
  - description: 'container at index #0 with display name [sample-batch] started.'
    eventTime: '2023-10-28T15:16:52.644051359Z'
    type: RUNNING_EVENT
  - description: 'container at index #0 with display name [sample-batch] finished
      with exit code 0.'
    eventTime: '2023-10-28T15:16:56.017869366Z'
    type: RUNNING_EVENT
  - description: Job state is set from SCHEDULED to RUNNING for job projects/268399420571/locations/us-central1/jobs/sample-batch.
    eventTime: '2023-10-28T15:16:52.643118435Z'
    type: STATUS_CHANGED
  - description: Job state is set from RUNNING to SUCCEEDED for job projects/268399420571/locations/us-central1/jobs/sample-batch.
    eventTime: '2023-10-28T15:16:56.017873961Z'
    type: STATUS_CHANGED
  taskGroups:
    group0:
      counts:
        SUCCEEDED: '1'
      instances:
      - bootDisk:
          image: projects/batch-custom-image/global/images/batch-cos-stable-official-20231018-00-p00
          sizeGb: '30'
          type: pd-balanced
        machineType: e2-standard-2
        provisioningModel: STANDARD
        taskPack: '1'
taskGroups:
- name: projects/268399420571/locations/us-central1/jobs/sample-batch/taskGroups/group0
  parallelism: '1'
  schedulingPolicy: IN_ORDER
  taskCount: '1'
  taskSpec:
    computeResource:
      cpuMilli: '1000'
      memoryMib: '1000'
    runnables:
    - container:
```

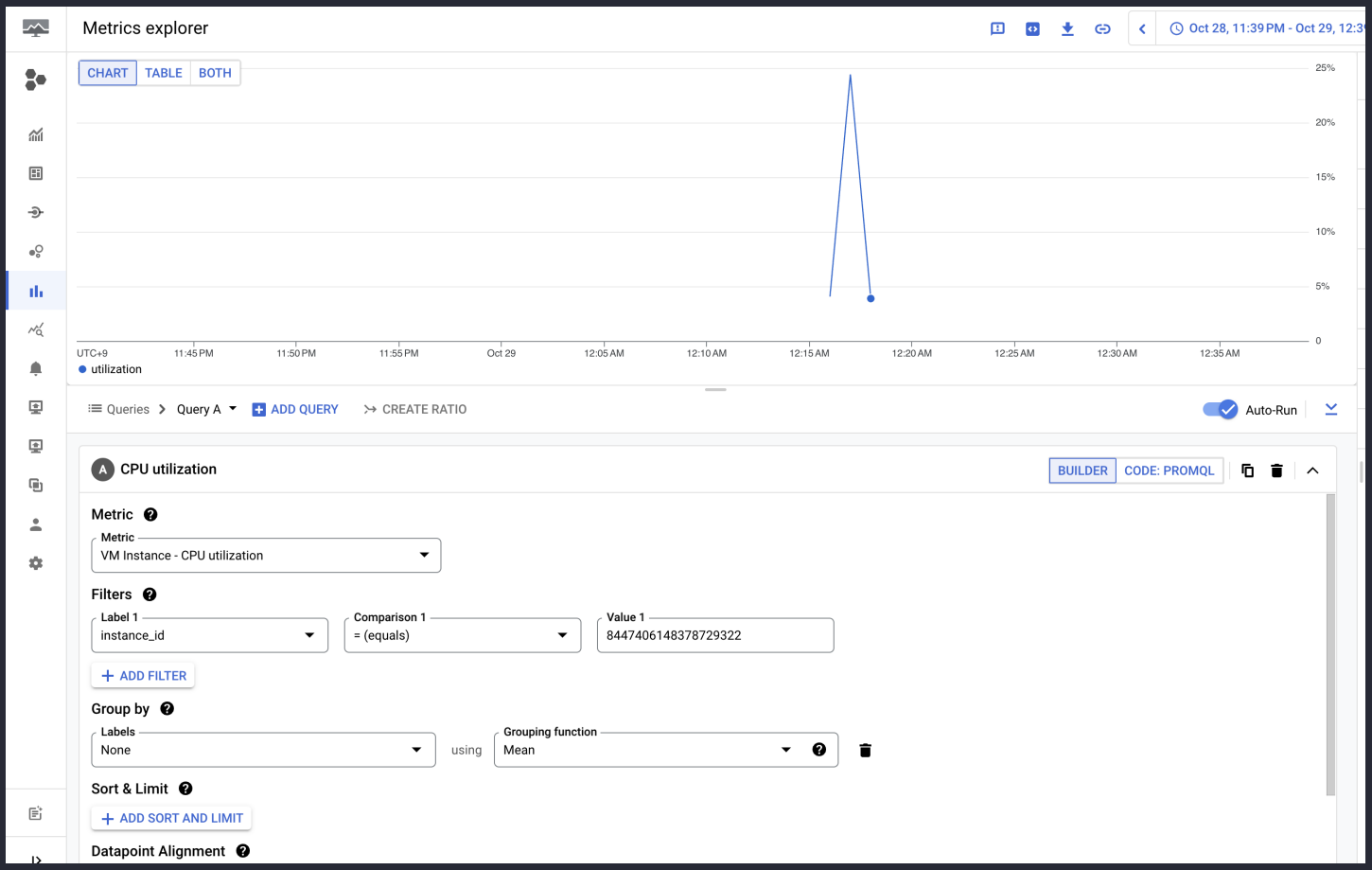
There was a

Source: Python

リソースのモニタリング

こんな感じで Cloud Monitoringを使えば見れる。Cloud Monitoringの画面はリンクで共有可能なので、UIではそのリンクを表示するだけで良さそう。ただし、VMインスタンスを特定する

ための識別子(`instance_id` or `instance_name`)が明示的にわからないため、リンクを生成できない。`instance_name`は`sample-batch-4eeaf202-8fa2-41b6-b983-0-group0-0-5tvc`のように、`{job名}-{job UID}-group{index}-{?}`になっており、ランダム性がある。`label`をつけることはできそうなので、`label`で絞り込むことはできるかも？



料金

制限

Batch で重い CSV を ETL する - G-gen Tech Blog の「# 類似プロダクトとの比較」が詳しい。これを見る限り、バッチとしての制限は特にないように思える。
ただし、実行トリガーは以下のとおりで、単なる定期実行でもCloud Schedulerと組み合わせる必要がある。また、DAGのような高機能サービスは提供されていない（Cloud Workflowsと組み合わせろということ？）。

- gcloud コマンド
- HTTP トリガー
- Cloud Scheduler
- *Cloud Workflows*

参考資料

- *あらゆる規模でバッチジョブをスケジュールできる新しいマネージド サービス、Batch の*

ご紹介 | Google Cloud 公式ブログ

- Get started with Batch | Google Cloud
- Batch で重い CSV を ETL する - G-gen Tech Blog
- GCP Batchを使ってみた - GMOインターネットグループ グループ研究開発本部

検討すべきこと

- cloud batchが参照するimageはどこに置くのか？
  - 特定の場所に集める
    - メリット: 参照する際のSAの権限管理が容易
    - デメリット: 1 org x 1 repository x 1 pathでユニークな命名規則を導入しなければならない
  - 分散させる
    - メリット: そのプロジェクトないで命名規則を設けてもらうため、自分たちは命名規則の管理をしなくて良い
    - デメリット: 参照するプロジェクトが増えるたびにCloud BatchのSAの権限付与が必要となる
- DAGをどのように実現するのか？
  - Cloud Workflowsで実現可能？
    - 参考: Cloud Workflowsで簡易的なデータパイプラインを構築してみる - G-gen Tech Blog
- UIは必要？
  - Cloud Batchを見る限り、必要な情報はGoogle Consoleを確認すれば良さそう。特にUIとして不足していることはない。
    - ただし、Cloud Batchを登録するなどのCLI 操作の負担は一部担うことができそう。
      →こちらもGoogle Consoleからできそう。
      → DAGなどの高機能なサービスの提供がひつようなので、その操作はUI・バックエンドで担う。
  - Cloud Batchで動かすDockerコンテナを作成することができるITリテラシーがあれば、Google Consoleだけで十分そう。つまり、Dockerコンテナを作ることができるのであれば、UIの提供は不要ではと考える。
    - Dockerコンテナを作らせる = UIも不必要
    - Dockerコンテナも作る = UIも必要