Task : Créer VM (e2-micro) avec un label lié à l'environnement (Développement ou Production) / Objectif : Eteindre cette VM le soir pour éviter qu'elle consomme toute la nuit en utilisant Cloud Functions sur GCP.

1st version :

"Outils utilisés :

- **Cloud Scheduler** : Création de tâches pour appeler les fonctions de démarrage et d'arrêt de l'instance en fonction d'un calendrier défini.
- **Cloud Functions** : Création de fonctions permettant de démarrer et d'arrêter l'instance pour laquelle nous souhaitons définir un calendrier d'exécution.
- **Pub/Sub** : Envoyer et recevoir des messages pour chaque évènement de démarrage ou d'arrêt.
- **Compute Engine** : Créer une instance (Machine Virtuelle) qui sera démarrée ou arrêtée suivant l'exécution du calendrier

Afin de bénéficier d'une meilleure latence, tous les composants utilisés seront déployés dans la même région.

Réalisation :

Créer un VPC

- Nom du VPC : vpc-buildingpack-finops
- Nom du Subnet : test-buildingpack-finops

Configurer l'instance "Compute Engine"

- Accès à la page "Instances de VM" dans la console GCP
- Sélectionner "Créer une instance"
- Définir le nom
  - Name : dev-finops-test (e2-micro)
- Ajouter un libellé ou une étiquette
  - Key : env
  - Value : dev
- Choisir une région
  - Région : us-west1-b

Configurer les fonctions "Cloud Functions" avec "Pub/Sub"

Créer fonction "Start"

- Accès à la page Cloud Functions dans la console de GCP
- Sélectionner "Créer une fonction"
- Définir le nom
  - Name : startInstancePubSub
- Choisir une région
  - Région : us-west1-b
- Sélectionner le type de déclencheur
  - Déclencheur : Cloud Pub/Sub
- Créer un topic
  - Nom du Sujet / Topic : start-instance-event
- Sélectionner l'environnement d'exécution
  - Node.js 10
  - Point d'entrée : startInstancePubSub
- Sélectionner index.js et ajouter le code suivant

```
const compute = require('@google-cloud/compute');
const instancesClient = new compute.InstancesClient();
const operationsClient = new compute.ZoneOperationsClient();

async function waitForOperation(projectId, operation) {
 while (operation.status !== 'DONE') {
  [operation] = await operationsClient.wait({
   operation: operation.name,
   project: projectId,
   zone: operation.zone.split('/').pop(),
  });
 }
}

/**
 * Starts Compute Engine instances.
 *
 * Expects a PubSub message with JSON-formatted event data containing the
 * following attributes:
 *  zone - the GCP zone the instances are located in.
 *  label - the label of instances to start.
 *
 * @param {!object} event Cloud Function PubSub message event.
 * @param {!object} callback Cloud Function PubSub callback indicating
 *  completion.
 */
exports.startInstancePubSub = async (event, context, callback) => {
 try {
  const project = await instancesClient.getProjectId();
  const payload = _validatePayload(event);
  const options = {
   filter: `labels.${payload.label}`,
   project,
   zone: payload.zone,
  };

  const [instances] = await instancesClient.list(options);

  await Promise.all(
   instances.map(async instance => {
    const [response] = await instancesClient.start({
     project,
     zone: payload.zone,
     instance: instance.name,
    });

    return waitForOperation(project, response.latestResponse);
   })
  );

  // Operation complete. Instance successfully started.
```

```
    const message = 'Successfully started instance(s)';
    console.log(message);
    callback(null, message);
  } catch (err) {
    console.log(err);
    callback(err);
  }
};

/**
 * Validates that a request payload contains the expected fields.
 *
 * @param {!object} payload the request payload to validate.
 * @return {!object} the payload object.
 */
const _validatePayload = event => {
  let payload;
  try {
    payload = JSON.parse(Buffer.from(event.data, 'base64').toString());
  } catch (err) {
    throw new Error('Invalid Pub/Sub message: ' + err);
  }
  if (!payload.zone) {
    throw new Error("Attribute 'zone' missing from payload");
  } else if (!payload.label) {
    throw new Error("Attribute 'label' missing from payload");
  }
  return payload;
};
```

- Sélectionner package.json et ajouter le code suivant

```
{
  "name": "cloud-functions-schedule-instance",
  "version": "0.1.0",
  "private": true,
  "license": "Apache-2.0",
  "author": "Google Inc.",
  "repository": {
    "type": "git",
    "url": "https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git"
  },
  "engines": {
    "node": ">=12.0.0"
  },
  "scripts": {
    "test": "mocha test/*.test.js --timeout=20000"
  },
  "devDependencies": {
    "mocha": "^9.0.0",
    "proxyquire": "^2.0.0",
    "sinon": "^14.0.0"
```

```
  },
  "dependencies": {
    "@google-cloud/compute": "^3.1.0"
  }
}
```

- Déployer le code

Créer fonction "Stop"

- Sélectionner "Créer une fonction"
- Définir le nom
    - Name : stopInstancePubSub
- Choisir une région
    - Région : us-west1-b
- Sélectionner le type de déclencheur
    - Déclencheur : Cloud Pub/Sub
- Créer un topic
    - Nom du Sujet / Topic : stop-instance-event
- Sélectionner l'environnement d'exécution
    - Node.js 10
    - Point d'entrée : stopInstancePubSub
- Sélectionner index.js et ajouter le code suivant

```
const compute = require('@google-cloud/compute');
const instancesClient = new compute.InstancesClient();
const operationsClient = new compute.ZoneOperationsClient();

async function waitForOperation(projectId, operation) {
  while (operation.status !== 'DONE') {
    [operation] = await operationsClient.wait({
      operation: operation.name,
      project: projectId,
      zone: operation.zone.split('/').pop(),
    });
  }
}

/**
 * Stops Compute Engine instances.
 *
 * Expects a PubSub message with JSON-formatted event data containing the
 * following attributes:
 *  zone - the GCP zone the instances are located in.
 *  label - the label of instances to stop.
 *
 * @param {!object} event Cloud Function PubSub message event.
 * @param {!object} callback Cloud Function PubSub callback indicating completion.
 */
exports.stopInstancePubSub = async (event, context, callback) => {
  try {
    const project = await instancesClient.getProjectId();
    const payload = _validatePayload(event);
```

```
    const options = {
      filter: `labels.${payload.label}`,
      project,
      zone: payload.zone,
    };

    const [instances] = await instancesClient.list(options);

    await Promise.all(
      instances.map(async instance => {
        const [response] = await instancesClient.stop({
          project,
          zone: payload.zone,
          instance: instance.name,
        });

        return waitForOperation(project, response.latestResponse);
      })
    );

    // Operation complete. Instance successfully stopped.
    const message = 'Successfully stopped instance(s)';
    console.log(message);
    callback(null, message);
  } catch (err) {
    console.log(err);
    callback(err);
  }
};

/**
 * Validates that a request payload contains the expected fields.
 *
 * @param {!object} payload the request payload to validate.
 * @return {!object} the payload object.
 */
const _validatePayload = event => {
  let payload;
  try {
    payload = JSON.parse(Buffer.from(event.data, 'base64').toString());
  } catch (err) {
    throw new Error('Invalid Pub/Sub message: ' + err);
  }
  if (!payload.zone) {
    throw new Error("Attribute 'zone' missing from payload");
  } else if (!payload.label) {
    throw new Error("Attribute 'label' missing from payload");
  }
  return payload;
};
```

- Sélectionner package.json et ajouter le code suivant

```
{
  "name": "cloud-functions-schedule-instance",
  "version": "0.1.0",
  "private": true,
  "license": "Apache-2.0",
  "author": "Google Inc.",
  "repository": {
    "type": "git",
    "url": "https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git"
  },
  "engines": {
    "node": ">=12.0.0"
  },
  "scripts": {
    "test": "mocha test/*.test.js --timeout=20000"
  },
  "devDependencies": {
    "mocha": "^9.0.0",
    "proxyquire": "^2.0.0",
    "sinon": "^14.0.0"
  },
  "dependencies": {
    "@google-cloud/compute": "^3.1.0"
  }
}
```

- Déployer le code

Vérifier que les fonctions sont opérationnelles

Vérifier fonction "Stop"

- Accès à la page Cloud Functions dans la console de GCP
- Sélectionner la fonction stopInstancePubSub
- Cliquer sur l'onglet Test
    - Evènement déclencheur

```
{"data":"eyJ6b25lIjoidXMtd2VzdDEtYiIsICJsYWJlbCI6ImVudj1kZXYifQo="}
```

- Chaîne encodée en base64 pour {"zone":"us-west1-b", "label":"env=dev"}
- Tester la fonction

Vérifier fonction "Start"

- Accès à la page Instances de VM dans la console GCP
- Vérifier qu'une coche grise est présente à côté de l'instance (30 sec …)
- Accès à la page Cloud Functions dans la console de GCP
- Sélectionner la fonction startInstancePubSub
- Cliquer sur l'onglet Test
    - Evènement déclencheur

```
{"data":"eyJ6b25lIjoidXMtd2VzdDEtYiIsICJsYWJlbCI6ImVudj1kZXYifQo="}
```

- Chaîne encodée en base64 pour {"zone":"us-west1-b", "label":"env=dev"}
- Tester la fonction
- Accès à la page Instances de VM dans la console GCP
- Vérifier qu'une coche verte est présente à côté de l'instance (30 sec …)

Configurer les tâches "Cloud Scheduler" pour qu'elles appellent "Pub/Sub"

Créer la tâche de démarrage

- Accès à la page Cloud Scheduler dans la console GCP
- Sélectionner Créer une tâche
    - Région : us-west1-b
    - Name : startup-dev-instances
    - Fréquence : 0 9 * * 1-5 (Opération effectuée tous les jours à 9h du lundi au vendredi)
    - Fuseau horaire : France
    - Type de cible : Pub/Sub
- Sélectionner start-instance-event
    - Message

```
{"zone":"us-west1-b","label":"env=dev"}
```

- Cliquer sur créer

Créer la tâche d'arrêt

- Accès à la page Cloud Scheduler dans la console GCP
- Sélectionner Créer une tâche
    - Région : us-west1-b
    - Name : shutdown-dev-instances
    - Fréquence : 0 17 * * 1-5 (Opération est effectuée tous les jours à 17h du lundi au vendredi)
    - Fuseau horaire : France
    - Type de cible : Pub/Sub
- Sélectionner stop-instance-event
    - Message

```
{"zone":"us-west1-b","label":"env=dev"}
```

- Cliquer sur créer

Vérifier que les tâches sont opérationnelles

Arrêter l'instance

- Accès à la page Cloud Scheduler de la console GCP
- Sélectionner shutdown-dev-instances
- Cliquer sur exécuter maintenant
- Accès à la page Instances de VM de la console GCP
- Vérifier qu'un carré gris est présent à côté de dev-finops-test

Démarrer l'instance

- Accès à la page Cloud Scheduler de la console GCP
- Sélectionner startup-dev-instances
- Cliquer sur exécuter maintenant
- Accès à la page Instances de VM de la console GCP
- Vérifier qu'un carré vert est présent à côté de dev-finops-test