

Cloud Schedular

|  |  |
| --- | --- |
| **Author** |  |
| **Approved By** |  |
| **Application Version** |  |
| **Date of Issue** |  |
| **Release Note Number** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Control Section** | | | | |
|  | | | | |
| **Revision History:** | | | | |
| **Document version** | **Pages** | **Action** | **Date of revision** | **By** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | | | | |

Table of Content

[1.0 Confidentiality / Legal notification 3](#_Toc331583772)

[2.0 Introduction 4](#_Toc331583773)

2.[1 APPLICATION ARCHITECTURE 3](#_Toc331583772)

[2.2 BEST PRACTISE 4](#_Toc331583773)

[2.3 sETUP THE CLOUD FUNCTIONS WITH CLOUD PUB/SUB 3](#_Toc331583772)

[2.4 SETUP THE CLOUD SCHEDULER JOBS TO CALL CLOUD PUB/SUB 4](#_Toc331583773)

[2.5 VERIFY THE JOBS WORK 3](#_Toc331583772)

1. Confidentiality / Legal notification

Confidential. Copyright © AI Enterprise., 2018. All rights reserved. This document, including the information contained

herein, is restricted, confidential and proprietary to AI Enterprise, and is to be used only by and disclosed only to those

within AI Enterprise with a need-to-know. DO NOT COPY OR FORWARD INTERNALLY OR RELEASE outside AI Enterprise

without authorization in writing by a Sr. Vice President, Principal, or Director-level manager or a direct designee

thereof who has responsibility for the information contained herein.

1. Introduction

This document describes about how to use Cloud Scheduler and Cloud Functions to automatically start and stop Compute Engine instances on a regular schedule using resource labels.

Objectives

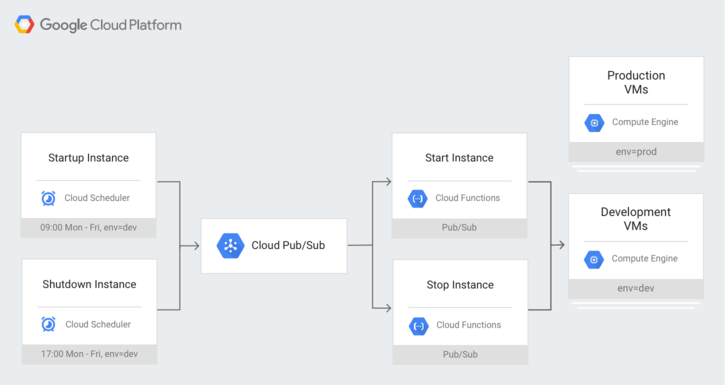
Write and deploy a set of functions with Cloud Functions that start and stop Compute Engine instances.

Create a set of jobs with Cloud Scheduler that schedule instances with an instance name to run 21:00-08:00, Monday-Friday to match typical business hours.

* 1. Application Architecture

This solution includes the following GCP components:

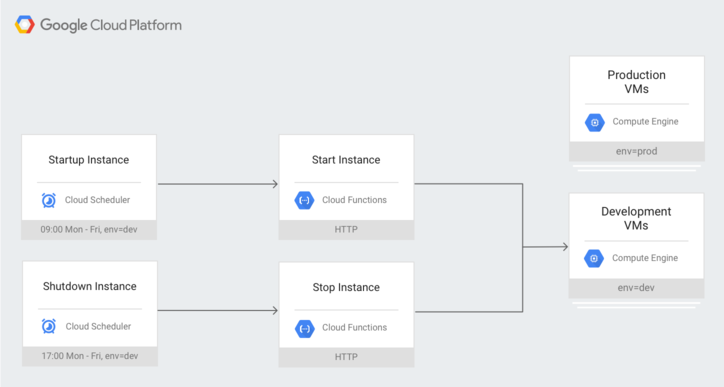
* Compute Engine Instance: A Compute Engine instance we want to run on a schedule.
* Cloud Functions functions: Functions to start and stop the instance we want to schedule.
* Cloud Pub/Sub messages: Messages sent and received for each start and stop event.
* Cloud Scheduler jobs: Jobs to make calls on a set schedule to start and stop the instance.



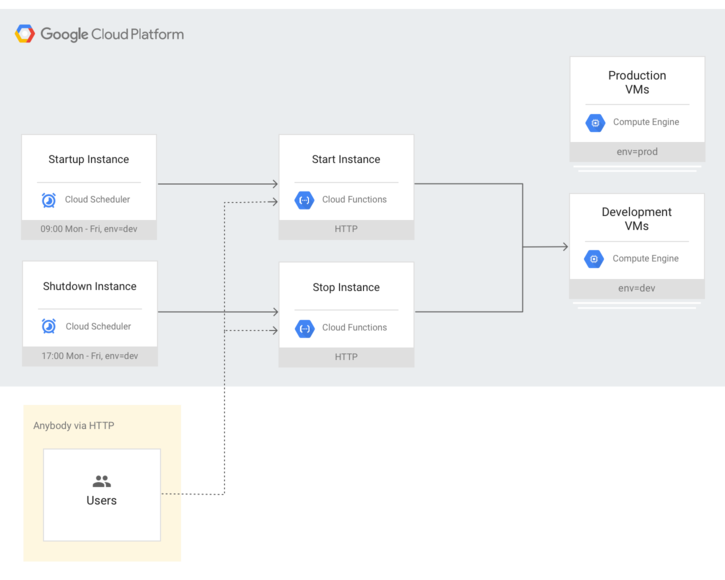
Note: Fortunately, for this application, no component needs to be run in the same region as any other component. So even if you deploy your Cloud Scheduler jobs to a certain region, you are free to locate your Compute Engine instance anywhere else. With that said, it is also true that locating your components in the same region will result in better latency performance.

* 1. Best Practice: Why not HTTP instead of Cloud Pub/Sub?

You may want to simplify this architecture by using Cloud Functions HTTP Triggers instead of Cloud Pub/Sub Triggers.



Important: HTTP functions in Cloud Functions have no authentication and are not secure. HTTP functions are unprotected and will respond to any HTTP request, which means anybody on the internet could start and stop your Compute Engine instances.



* 1. Set up the Cloud Functions with Cloud Pub/Sub

**Create and deploy the functions**

**Create the start function.**

1. Go to the **Cloud Functions** page in the GCP Console.   
   [GO TO THE CLOUD FUNCTIONS PAGE](https://console.cloud.google.com/functions).
2. Click **Create Function**.
3. Set the **Name** to startInstancePubSub.
4. Leave **Memory allocated** at its default value.
5. For **Trigger**, select Cloud Pub/Sub.
6. For **Topic**, select Create new topic....
7. A **New pub/sub topic** dialog box should appear.
   1. Under **Name**, enter start-instance-event.
   2. Click **Create** to finish the dialog box.
8. For **Runtime**, select Node.js 8.
9. Above the code text block, select the index.js tab.
10. Replace the starter code with the following code:

|  |
| --- |
| const Buffer = require('safe-buffer').Buffer; const Compute = require('@google-cloud/compute'); const compute = new Compute();  /\*\*  \* Starts a Compute Engine instance.  \*  \* 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 = (event, callback) => {   try {     const pubsubMessage = event.data;     const payload = \_validatePayload(       JSON.parse(Buffer.from(pubsubMessage.data, 'base64').toString())     );     const options = {filter: `labels.${payload.label}`};     compute.getVMs(options).then(vms => {       vms[0].forEach(instance => {         if (payload.zone === instance.zone.id) {           compute             .zone(payload.zone)             .vm(instance.name)             .start()             .then(data => {               // Operation pending.               const operation = data[0];               return operation.promise();             })             .then(() => {               // Operation complete. Instance successfully started.               const message = 'Successfully started instance ' + instance.name;               console.log(message);               callback(null, message);             })             .catch(err => {               console.log(err);               callback(err);             });         }       });     });   } 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.  \*/ function \_validatePayload(payload) {   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; } |

11. Above the code text block, select the package.json tab.

12. Replace the starter code with the following code:

{  
  "name": "cloud-functions-schedule-instance",  
  "version": "0.0.2",  
  "private": true,  
  "license": "Apache-2.0",  
  "author": "Google Inc.",  
  "repository": {  
    "type": "git",  
    "url": "https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git"  
  },  
  "engines": {  
    "node": ">=8.13.0"  
  },  
  "scripts": {  
    "test": "mocha test/\*.test.js --timeout=20000"  
  },  
  "devDependencies": {  
    "@google-cloud/nodejs-repo-tools": "^3.3.0",  
    "mocha": "^6.0.0",  
    "proxyquire": "^2.0.0",  
    "sinon": "^7.0.0"  
  },  
  "dependencies": {  
    "@google-cloud/compute": "^0.12.0",  
    "safe-buffer": "^5.1.2"  
  }  
}

13. For Function to execute, enter startInstancePubSub.

14. Click Create.

**Create the stop function.**

1. You should be on the Cloud Functions page in the GCP Console.
2. Click Create Function.
3. Set the Name to stopInstancePubSub.
4. Leave Memory allocated at its default value.
5. For Trigger, select Cloud Pub/Sub.
6. For Topic, select Create new topic....
7. A New pub/sub topic dialog box should appear.
8. Under Name, enter stop-instance-event.
9. Click Create to finish the dialog box.
10. For Runtime, select Node.js 8.
11. Above the code text block, select the index.js tab.
12. Replace the starter code with the following code:

const Buffer = require('safe-buffer').Buffer;  
const Compute = require('@google-cloud/compute');  
const compute = new Compute();  
  
/\*\*  
 \* Stops a Compute Engine instance.  
 \*  
 \* Expects a PubSub message with JSON-formatted event data containing the  
 \* following attributes:  
 \*  zone - the GCP zone the instances are located in.  
 \*  instance - the name of a single instance.  
 \*  label - the label of instances to start.  
 \*  
 \* Exactly one of instance or label must be specified.  
 \*  
 \* @param {!object} event Cloud Function PubSub message event.  
 \* @param {!object} callback Cloud Function PubSub callback indicating completion.  
 \*/  
exports.stopInstancePubSub = (event, callback) => {  
  try {  
    const pubsubMessage = event.data;  
    const payload = \_validatePayload(  
      JSON.parse(Buffer.from(pubsubMessage.data, 'base64').toString())  
    );  
    const options = {filter: `labels.${payload.label}`};  
    compute.getVMs(options).then(vms => {  
      vms[0].forEach(instance => {  
        if (payload.zone === instance.zone.id) {  
          compute  
            .zone(payload.zone)  
            .vm(instance.name)  
            .stop()  
            .then(data => {  
              // Operation pending.  
              const operation = data[0];  
              return operation.promise();  
            })  
            .then(() => {  
              // Operation complete. Instance successfully stopped.  
              const message = 'Successfully stopped instance ' + instance.name;  
              console.log(message);  
              callback(null, message);  
            })  
            .catch(err => {  
              console.log(err);  
              callback(err);  
            });  
        }  
      });  
    });  
  } 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.  
 \*/  
function \_validatePayload(payload) {  
  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;  
}

1. Above the code text block, select the package.json tab.
2. Replace the starter code with the following code:

{  
  "name": "cloud-functions-schedule-instance",  
  "version": "0.0.2",  
  "private": true,  
  "license": "Apache-2.0",  
  "author": "Google Inc.",  
  "repository": {  
    "type": "git",  
    "url": "https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git"  
  },  
  "engines": {  
    "node": ">=8.13.0"  
  },  
  "scripts": {  
    "test": "mocha test/\*.test.js --timeout=20000"  
  },  
  "devDependencies": {  
    "@google-cloud/nodejs-repo-tools": "^3.3.0",  
    "mocha": "^6.0.0",  
    "proxyquire": "^2.0.0",  
    "sinon": "^7.0.0"  
  },  
  "dependencies": {  
    "@google-cloud/compute": "^0.12.0",  
    "safe-buffer": "^5.1.2"  
  }  
}

13. For Function to execute, enter stopInstancePubSub.

14. Click Create.

* 1. Set up the Cloud Scheduler jobs to call Cloud Pub/Sub

### Create the jobs

#### Create the start job.

1. Go to the **Cloud Scheduler** page in the GCP Console.   
   [GO TO THE CLOUD SCHEDULER PAGE](https://console.cloud.google.com/cloudscheduler).
2. Click **Create Job**.
3. Set the **Name** to startup-workday-instances.
4. For **Frequency**, enter 0 8 \* \* MON-FRI.
5. For **Timezone**, select as india.
6. For **Target**, select Pub/Sub.
7. For **Topic**, enter start-instance-event.
8. For **Payload**, enter the following:

[{"zone":"us-central1-a","instance":"aie-consul-server"},{"zone":"us-central1-a","instance":"gpu-ds-01"},{"zone":"us-central1-c","instance":"aie-artifact-srv"},{"zone":"us-central1-c","instance":"aie-sc-tx-svr"},{"zone":"us-central1-c","instance":"deploy-server"},{"zone":"us-central1-c","instance":"dns-server"},{"zone":"us-central1-c","instance":"jenkins-master-0"},{"zone":"us-central1-c","instance":"jenkins-windows-slave01"},{"zone":"us-central1-f","instance":"aie-sonarqube"},{"zone":"us-central1-c","instance":"aie-sitecore"},{"zone":"us-central1-c","instance":"aie-sc-tx-svr"}]

1. Click Create.

#### Create the stop job.

1. You should be on the **Cloud Functions** page in the GCP Console.
2. Click **Create Job**.
3. Set the **Name** to shutdown-workday-instances.
4. For **Frequency**, enter 0 21 \* \* MON-FRI.
5. For **Timezone**, select as India
6. For **Target**, select Pub/Sub.
7. For **Topic**, enter stop-instance-event.
8. For **Payload**, enter the following:

[{"zone":"us-central1-a","instance":"aie-consul-server"},{"zone":"us-central1-a","instance":"gpu-ds-01"},{"zone":"us-central1-c","instance":"aie-artifact-srv"},{"zone":"us-central1-c","instance":"aie-sc-tx-svr"},{"zone":"us-central1-c","instance":"deploy-server"},{"zone":"us-central1-c","instance":"dns-server"},{"zone":"us-central1-c","instance":"jenkins-master-0"},{"zone":"us-central1-c","instance":"jenkins-windows-slave01"},{"zone":"us-central1-f","instance":"aie-sonarqube"},{"zone":"us-central1-c","instance":"aie-sitecore"},{"zone":"us-central1-c","instance":"aie-sc-tx-svr"}]

1. Click **Create**.

Note: Your Cloud Scheduler jobs are automatically deployed to the region you selected for your App Engine app when Setting Up Your Environment at the beginning of this tutorial.

* 1. Verify the jobs work

### CONSOLE

#### Stop the instance

1. Go to the **Cloud Scheduler** page in the GCP Console.   
   [GO TO THE CLOUD SCHEDULER PAGE](https://console.cloud.google.com/cloudscheduler).
2. For the job named shutdown-dev-instances, click the **Run now** button on the far right side of the page.
3. Go to the **VM instances** page in the GCP Console.   
   [GO TO THE VM INSTANCES PAGE](https://console.cloud.google.com/compute/instances).
4. Verify that the instance named dev-instance has a grey square next to its name, indicating that it has stopped. It may take up to 30 seconds for it to finish shutting down.

#### Start the instance

1. Go to the **Cloud Scheduler** page in the GCP Console.   
   [GO TO THE CLOUD SCHEDULER PAGE](https://console.cloud.google.com/cloudscheduler).
2. For the job named startup-dev-instances, click the **Run now** button on the far right side of the page.
3. Go to the **VM instances** page in the GCP Console.   
   [GO TO THE VM INSTANCES PAGE](https://console.cloud.google.com/compute/instances).
4. Verify that the instance named dev-instance has a green checkmark next to its name, indicating that it is running. It may take up to 30 seconds for it to finish starting up.