## Lab 4-03: Integrating Google Cloud Function

### Introduction

Google Cloud Functions can help with that. Each time the admin adds a product, the write event could trigger a function to notify all its users about the new addition.

Breaking down monolithic applications into a series of microservices is a viable programming model. Such models can be achieved using Google Cloud Functions, as they are event-driven, serverless functions that can scale effortlessly.

### Problem

A national business wants its user base to stay updated on every record they enter into its product database as a way of advertisement. Whenever a new article about their product is added to the database, a new write must occur in the real-time database informing all the customers about the new article.

### Solution

In this lab, we will lay the groundwork for managing the cloud function in a source code in a Google Cloud Source Repository and then deploy and test it.

|  |
| --- |
| 1. Enable the APIs; One to handle the cloud source repository and another to handle cloud functions themselves. Go to **APIs & Services** from the main navigation menu and select **Library**.      1. Search for cloud source repository.      1. Click on Cloud Source Repository and enable it.      1. To enable cloud function API, go to the navigation menu. Go to **APIs and Services** and select **Library**.      1. Search for Cloud Functions API.      1. Enable this API as well.     Now, we are going to add a source repository.   1. From the main navigation, choose **Source Repositories** from the **CI/CD** section      1. Click **Get started**.      1. Then click **Create repository**.      1. Select **Create new repository** and click **Continue**.      1. Then, give a valid name and select your project. Click **Create**.      1. Select **Push code from a local Git repository**.      1. Then, go back to the cloud console by clicking the following button.      1. Click **Activate Cloud Shell**.     Now, we will clone the repo that we just created, then bring some files in from an external repository on Github and move those files into that repo. We will then push all of that up to the repo in Google Cloud Repositories.   1. Clone the repo that we just created using : gcloud source repos clone <repo name>.      1. Clone a repository from GitHub using: git clone <github url>      1. Next, move the files that are in the folder content-gcpro-developer into ips-demo-repo. Use the following command: cp content-gpro-developer/cloud-function-lab/\* ips-demo-repo.      1. To check your repo, navigate to your directory using cd <directory name> and then check files using command **ls**.      1. Now, we will push these files to our existing source repository. We need to configure two variables; User email and username. We will do this by using the following commands:   Git congif --global user.email “<name of the user>”  Git config --global user.name “<your name>”     1. Now, we will push the file to the source repository. To include all the files, we will use git add.      1. Now, commit using git commit -m “<your message>.”      1. Finally, push the code to the master branch by using git push origin master.      1. Return to the repository page and refresh it. You will see the files of your repository.     These files have been placed into the cloud source repository. Now, we will move to Cloud Functions.   1. Return to the cloud console.      1. Navigate to **Cloud Functions** from the main navigation menu and click **Create Functions.**        1. Pick a name for the cloud function and change the trigger type to HTTP.        1. Choose python 3.7 for runtime.      1. Click the **Deploy** button, and your function will appear in the list with a green checkmark.        1. In the General section, you will see the data as the function is invoked repeatedly.      1. In Trigger, you can copy the url and paste it into the browser to test it.      1. In the testing tab, we will test our function by giving it data in json format. |