what about sending notifications to email and slack

**ChatGPT said:**

✅ Great next step — real FinOps workflows almost *always* include automatic **notifications** (email, Slack, etc.) when budgets are breached or anomalies are detected.

Here’s how you can do it **on GCP / BigQuery**, even with synthetic data:

**📦 Option 1: Budget Alerts via Google Cloud Budgets & Pub/Sub**

1. **Create a budget** in Cloud Billing.
2. **Set thresholds** (e.g., 50%, 75%, 100%).
3. **Attach a Pub/Sub topic** as the notification channel.
4. Use:
   * **Cloud Functions** or **Cloud Run** to subscribe to Pub/Sub.
   * Function parses the message → sends to:
     + **Email** (via SendGrid / Mailgun / SMTP)
     + **Slack** (via Incoming Webhook).

This approach is *production-grade*: actual GCP budgets → native alerts.

**🧩 Option 2: DIY alerts from scheduled BigQuery queries**

Since we’re using **synthetic data**, you can:

* Create scheduled queries like budgeting\_alerts.sql or cost\_threshold\_alerts.sql that **write tables with alert rows**.
* Use:
  + **Cloud Functions** that run daily, query these alert tables, and if any row has 'ALERT' status:
    - Send email (e.g., SendGrid)
    - Send Slack message

You can automate this with Terraform too.

**🛠 Terraform (conceptual):**

* google\_cloudfunctions\_function → deploy function
* google\_pubsub\_topic & google\_pubsub\_subscription → trigger function
* google\_bigquery\_data\_transfer\_config → scheduled query produces alert table

**✏ Example workflow:**

1. Scheduled query → writes daily\_threshold\_alerts table.
2. Cloud Function runs daily:
   * Queries table:

sql

CopyEdit

SELECT \* FROM `finops-practice.billing\_data.daily\_threshold\_alerts`

WHERE alert\_status != 'OK'

* + If results → sends Slack or email.

**✅ Summary:**

| **What** | **Best tool** |
| --- | --- |
| Actual budget alerts | Cloud Budgets + Pub/Sub + Cloud Function |
| Custom BigQuery alerts | Scheduled queries + Cloud Function / Cloud Run |
| Send Slack / Email | Cloud Function + SendGrid / SMTP / Slack Webhooks |
| Automate infra | Terraform |

⚡ **If you say:**

*“Yes, add Terraform & Python Cloud Function to send alerts to Slack + email”*  
I’ll generate:

* Terraform infra for Pub/Sub & Function
* Example Python function: query BigQuery + send alerts
* Update README & folder layout

Would you like me to do that? 🚀

**You said:**

Terraform infra for Pub/Sub & Function Example Python function: query BigQuery + send alerts

**ChatGPT said:**

Got it! Here's a concise Terraform setup for Pub/Sub and Cloud Function, plus a sample Python Cloud Function that queries BigQuery for alerts and sends notifications to Slack and email.

**1. Terraform Infra for Pub/Sub & Cloud Function**

hcl

CopyEdit

terraform {

required\_providers {

google = {

source = "hashicorp/google"

version = "~> 4.0"

}

}

}

provider "google" {

project = "finops-practice"

region = "us-central1"

}

resource "google\_pubsub\_topic" "alerts\_topic" {

name = "billing-alerts-topic"

}

resource "google\_pubsub\_subscription" "alerts\_subscription" {

name = "billing-alerts-subscription"

topic = google\_pubsub\_topic.alerts\_topic.name

}

resource "google\_storage\_bucket" "function\_bucket" {

name = "${var.project}-functions-bucket"

location = "US"

}

resource "google\_cloudfunctions\_function" "alert\_function" {

name = "billing-alert-function"

runtime = "python39"

entry\_point = "check\_billing\_alerts"

source\_archive\_bucket = google\_storage\_bucket.function\_bucket.name

source\_archive\_object = "billing\_alert\_function.zip"

trigger\_topic = google\_pubsub\_topic.alerts\_topic.name

available\_memory\_mb = 256

timeout = 60

environment\_variables = {

SLACK\_WEBHOOK\_URL = var.slack\_webhook\_url

EMAIL\_SENDER = var.email\_sender

EMAIL\_RECIPIENT = var.email\_recipient

PROJECT\_ID = var.project

DATASET\_ID = "billing\_data"

ALERT\_TABLE = "daily\_threshold\_alerts"

}

}

variable "project" {

default = "finops-practice"

}

variable "slack\_webhook\_url" {

description = "Slack Incoming Webhook URL"

type = string

}

variable "email\_sender" {

description = "Sender email address"

type = string

}

variable "email\_recipient" {

description = "Recipient email address"

type = string

}

**2. Python Cloud Function (main.py)**

python

CopyEdit

import os

import base64

import json

from google.cloud import bigquery

import requests

import smtplib

from email.mime.text import MIMEText

# Environment variables

SLACK\_WEBHOOK\_URL = os.environ.get("SLACK\_WEBHOOK\_URL")

EMAIL\_SENDER = os.environ.get("EMAIL\_SENDER")

EMAIL\_RECIPIENT = os.environ.get("EMAIL\_RECIPIENT")

PROJECT\_ID = os.environ.get("PROJECT\_ID")

DATASET\_ID = os.environ.get("DATASET\_ID")

ALERT\_TABLE = os.environ.get("ALERT\_TABLE")

client = bigquery.Client(project=PROJECT\_ID)

def check\_billing\_alerts(event, context):

"""Triggered by Pub/Sub. Queries BigQuery alerts and sends notifications."""

query = f"""

SELECT \*

FROM `{PROJECT\_ID}.{DATASET\_ID}.{ALERT\_TABLE}`

WHERE alert\_status != 'OK'

"""

query\_job = client.query(query)

results = query\_job.result()

alerts = [dict(row.items()) for row in results]

if not alerts:

print("No alerts found. Exiting.")

return

message = "🚨 \*Billing Alerts Detected\* 🚨\n\n"

for alert in alerts:

message += f"- Project: {alert.get('project\_id')}\n"

message += f" Alert: {alert.get('alert\_message')}\n"

message += f" Cost: {alert.get('cost')}\n\n"

# Send Slack notification

slack\_payload = {"text": message}

slack\_response = requests.post(SLACK\_WEBHOOK\_URL, json=slack\_payload)

if slack\_response.status\_code != 200:

print(f"Slack notification failed: {slack\_response.text}")

# Send Email notification

send\_email(message)

def send\_email(body):

msg = MIMEText(body)

msg["Subject"] = "Billing Alerts Notification"

msg["From"] = EMAIL\_SENDER

msg["To"] = EMAIL\_RECIPIENT

# Simple SMTP (adjust as needed for your SMTP server)

try:

with smtplib.SMTP("smtp.gmail.com", 587) as server:

server.starttls()

# Use app password or OAuth for Gmail

server.login(EMAIL\_SENDER, os.environ.get("EMAIL\_PASSWORD"))

server.send\_message(msg)

print("Email sent successfully.")

except Exception as e:

print(f"Failed to send email: {e}")

**3. Packaging & Deploying Cloud Function**

1. Create a folder for the function with main.py and requirements.txt (include google-cloud-bigquery and requests).
2. Zip the folder contents as billing\_alert\_function.zip.
3. Upload to the GCS bucket created by Terraform (function\_bucket).
4. Terraform apply will deploy the function and hook it to the Pub/Sub topic.

**4. How it works:**

* Any alert-producing service (could be BigQuery scheduled query, Cloud Scheduler, or manual) publishes to Pub/Sub topic billing-alerts-topic.
* The Cloud Function listens on this topic, queries BigQuery for any alert rows, then sends Slack + email notifications accordingly.

If you'd like, I can help you with:

* **requirements.txt** for the Python function
* A **sample BigQuery alert table schema** for daily\_threshold\_alerts
* A sample **Terraform variable file** with secrets management tips
* **Automated publishing** to Pub/Sub via scheduled query or Cloud Scheduler

Ready for next steps?

**You said:**

where do i place this in my vc code structure

**ChatGPT said:**

Great question! Here’s **how to place this in your VS Code project** so your FinOps lab stays clean, modular, and ready to scale.

Imagine your folder structure (based on what you’ve built so far):

cpp

CopyEdit

finops-practice/

├─ data/

│ └─ synthetic\_gcp\_billing\_export.csv

├─ queries/

│ ├─ cost\_visibility.sql

│ ├─ anomaly\_detection.sql

│ ├─ trend\_forecast.sql

│ └─ ... (others)

├─ scripts/

│ ├─ generate\_synthetic\_gcp\_billing.py

│ ├─ upload\_to\_bigquery.sh

│ └─ upload\_to\_bigquery.ps1

├─ terraform/

│ ├─ main.tf

│ ├─ variables.tf (optional, recommended)

│ └─ outputs.tf (optional)

└─ functions/

└─ billing\_alert\_function/

├─ main.py

└─ requirements.txt

README.md

✅ **Details:**

**1️⃣ functions/billing\_alert\_function/**

Create a folder:

bash

CopyEdit

finops-practice/functions/billing\_alert\_function/

Put inside:

* main.py (the Python code I shared)
* requirements.txt (will include: google-cloud-bigquery, requests, smtplib is standard)

This keeps your Cloud Function isolated and version controlled.

**2️⃣ terraform/**

In finops-practice/terraform/:

* Place the **Terraform infra** code I shared (add to your existing main.tf)
* Keep variables in variables.tf if you want cleaner code:

hcl

CopyEdit

variable "slack\_webhook\_url" {}

variable "email\_sender" {}

variable "email\_recipient" {}

variable "project" {

default = "finops-practice"

}

If you already have main.tf for BigQuery scheduled queries, you can **extend** it by adding the Pub/Sub + Cloud Function resources **at the end**.

**3️⃣ queries/**

Your SQL files (alerting, anomaly detection, forecasting, cost by department, etc.) stay here:

bash

CopyEdit

finops-practice/queries/

**4️⃣ Zip and deploy**

When deploying:

* Zip only the contents of functions/billing\_alert\_function/ (i.e., main.py + requirements.txt), not the folder itself.
* Name the archive billing\_alert\_function.zip and upload to the GCS bucket defined in Terraform.

**✍️Update your README.md to document:**

* Purpose of functions/billing\_alert\_function/
* Terraform modules you have
* How to deploy the Cloud Function
* How alerts are triggered

If you'd like 👉  
✅ I can generate:

* requirements.txt
* variables.tf
* Updated README.md section explaining this new alerting infra

Would you like me to?