

# Build a Serverless Github Bot in GCP

Franklin Diaz

DE:AD:10:C5

Tuesday January 10, 2023

# INTRODUCTION



- Click here for Session Details
- Project source files are available:  
<https://github.com/devsecfranklin/workshop-codemash-2023>
- Pework available at this link.



Github: [devsecfranklin .xXx](#). E-mail: [devsecfranklin@duck.com](mailto:devsecfranklin@duck.com)

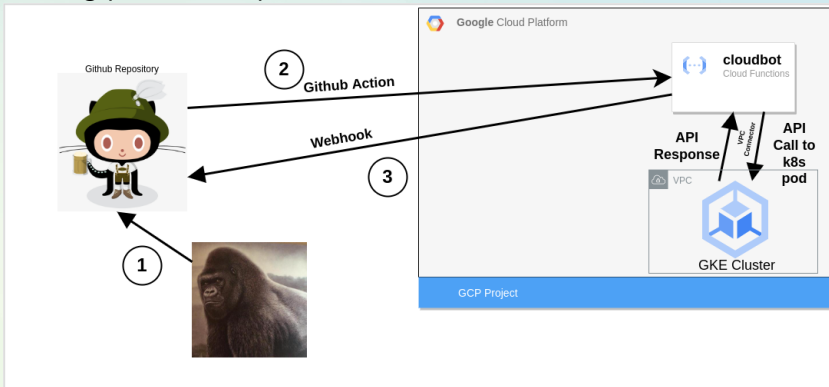


Twitter:  
[@thedeilsvoice](#)

Mastodon:  
[@devsecfranklin@defcon.social](#)

# Overview: Usage

The big picture for operation.



# Outline: What will we cover?

A high level overview of the learning path is as follows:

- Prework
  - Github project repository setup.
  - Set up the development environment.
  - Set up Google Cloud account.
- In Class setup slides.
- Review the Python source for the bot.
- Configure Terraform and deploy the bot.
- Test everything out on one of your repositories in Github.
- Explore possibilities for extending the functionality.



# PRE-WORK



# Setup: VSCode

VSCode (<https://code.visualstudio.com>)

- Windows 64 bit User Installer: VSCodeUserSetup-x64-1.73.1.exe
- Mac Universal: VSCode-darwin-universal.zip
- Linux (Debian, Ubuntu): code\_1.73.1-1667967334\_amd64.deb
- Linux (Red Hat, Fedora, SUSE): code-1.73.1-1667967421.el7.x86\_64.rpm

Click this link for details on using dev containers in VSCode





# Setup: git

GIT (<https://git-scm.com/downloads>)

- Windows 32 Bit: Git-2.38.1-64-bit.exe
- Windows 64 Bit: Git-2.38.1-32-bit.exe
- Mac: git-2.15.0-intel-universal-mavericks.dmg



# Setup: Docker Desktop

Docker Desktop (<https://www.docker.com/>)

- Windows: Docker Desktop Installer.exe
- MacOS (Intel Chip): Docker.dmg
- MacOS (M1 Chip): Docker.dmg
- Linux instructions can be found: [here](#)

Click [here](#) to see Docker setup steps from Microsoft



# Setup: Clone and Open the Project Repository

- Time to clone the repository.
- Click this link for the Github repository
- In VSCode, press F1 and enter the command “Dev Containers: Open Folder in Container”
  - You can also choose “Dev Containers: Open Workspace in Container”
  - Here is the Microsoft VSCode dev containers tutorial
- From the top menu select “Terminal – New Terminal”
- Now “cd /workspaces/workshop-codemash-2023/bin” and type “setup-dev-env.sh”



# Google Cloud: Account Setup

- Sign up for a free tier GCP account.
- Navigate to <https://cloud.google.com/> and make sure you have a usable project to work in.
- Here is some information about creating projects in GCP



# IN CLASS SETUP



# Google Cloud: Update Project Name and Login

- Update your project name in the file `"/workspaces/workshop-codemash-2023/.envrc"`
- Update your project name in the file `"/workspaces/workshop-codemash-2023/src/config.ini"`
- Type the command `"direnv allow ."` to reload the ENV variables.
- In the dev container, run the command `"gcloud auth login"` and follow the directions there.
- Verify you are connected to GCP with the command `"gcloud auth list"`



# Google Cloud: Create Service User

We create a service user in GCP with limited scope of permissions.



# Google Cloud: Create Secret in Secrets Mgr

- The Cloud Function is expecting us to create a secret named “gh\_secret\_token”.
- Enable the Secret Manager service.
- Add the secret.

**Secret: "gh\_secret\_token"**  
projects/552552096122/secrets/gh\_secret\_token

OVERVIEW **VERSIONS** PERMISSIONS LOGS

Versions [+ NEW VERSION](#) ENABLE DISABLE DESTROY

<input type="checkbox"/>	Version	Status	Encryption	Created on ↓	Actions
<input type="checkbox"/>	1	✓ Enabled	Google-managed	1/8/23, 11:16 AM	⋮

No versions selected

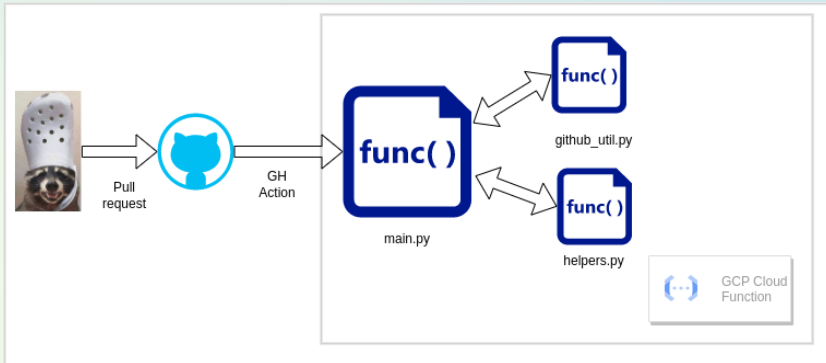


# PYTHON



# Overview: Python Functions

The big picture for the Python code files.



# The Python Application

- The main function is essentially a Flask app that waits for an incoming JSON messages.
- Let's take a closer look

```
if __name__ == "__main__":  
    app = Flask(__name__)  
    app.route("/")(lambda: main(request))  
    app.run()
```



# Python: Logging

- Logging is set to the “INFO” level.
- The log files show up in GCP under the cloud function.

```
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
> 2023-01-08 11:03:38.769 EST cloudbot-franklin qbt1a81zext Function execution started
> 2023-01-08 11:03:38.185 EST cloudbot-franklin qbt1a81zext INFO:root:Started cloudbot
> 2023-01-08 11:03:38.198 EST cloudbot-franklin qbt1a81zext INFO:root:Validate the user defined configuration.
> 2023-01-08 11:03:38.198 EST cloudbot-franklin qbt1a81zext INFO:root:User defined configuration is formatted properly.
> 2023-01-08 11:03:38.280 EST cloudbot-franklin qbt1a81zext INFO:root:Pull secret from Secret Manager for project_id gcp-gm-pao
> 2023-01-08 11:03:38.417 EST cloudbot-franklin qbt1a81zext INFO:root:Secret pulled successfully from GCP Secret Manager.
> 2023-01-08 11:03:38.424 EST cloudbot-franklin qbt1a81zext INFO:root:Instantiate GH object with label cloudbot-testing
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Received JSON message: Pull request number 46 by devsecfranklin on repository devsecfranklin/workshop
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Check JSON fields in GH msg.
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:PR Number found in commit: 48
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Username found in commit: devsecfranklin
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Github repo name found: devsecfranklin/workshop-codemash-2023
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:ref: refs/pull/48/merge
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Commit SHA found: 659583b6b7b4fb44b5ba64ef585c5c05a49
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Completed check JSON fields in GH msg.
> 2023-01-08 11:03:38.425 EST cloudbot-franklin qbt1a81zext INFO:root:Check PR label cloudbot-testing
> 2023-01-08 11:03:38.364 EST cloudbot-franklin qbt1a81zext INFO:root:Setting label cloudbot-testing on PR 48
> 2023-01-08 11:03:38.488 EST cloudbot-franklin qbt1a81zext INFO:root:1: <github.MainClass.Github object at 0x0c645e55108>
> 2023-01-08 11:03:37.847 EST cloudbot-franklin qbt1a81zext INFO:root:Found filename: README.md
> 2023-01-08 11:03:37.847 EST cloudbot-franklin qbt1a81zext INFO:root:Found filename: docs/images/config.ini.png
> 2023-01-08 11:03:37.847 EST cloudbot-franklin qbt1a81zext INFO:root:Found filename: docs/slides/workshop-codemash-2023.pdf
> 2023-01-08 11:03:37.847 EST cloudbot-franklin qbt1a81zext INFO:root:Found filename: docs/slides/workshop-codemash-2023.tex
> 2023-01-08 11:03:37.847 EST cloudbot-franklin qbt1a81zext INFO:root:Found filename: src/config.ini
> 2023-01-08 11:03:37.847 EST cloudbot-franklin qbt1a81zext INFO:root:Looking for string in comment: #@cloudt
> 2023-01-08 11:03:37.154 EST cloudbot-franklin qbt1a81zext INFO:root:Adding comment to the commit.
> 2023-01-08 11:03:37.154 EST cloudbot-franklin qbt1a81zext INFO:root:Cloudbot adding comment on repo devsecfranklin/workshop-codemash-2023 to PR 48
> 2023-01-08 11:03:38.126 EST cloudbot-franklin qbt1a81zext INFO:root:Finished adding comment to PR 48
> 2023-01-08 11:03:38.281 EST cloudbot-franklin qbt1a81zext INFO:root:<Response [200]>
> 2023-01-08 11:03:38.286 EST cloudbot-franklin qbt1a81zext Function execution took 7516 ms, finished with status code: 200
```

# Python: config.ini

- The configparser module is used to make customization easier.
- The Cloud Function is expecting us to create a secret named “gh\_secret\_token”.

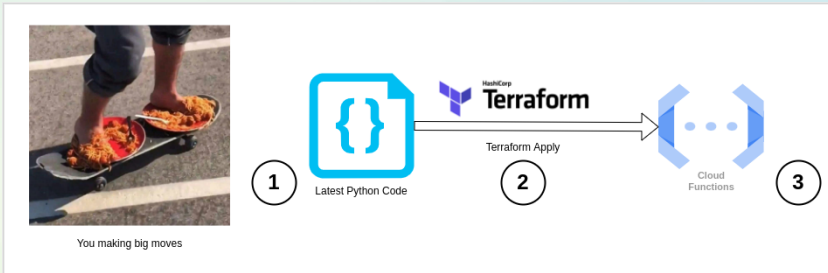
```
workshop-codemash-2023 > src > config.ini
1  [required_options]
2
3  # GCP Project ID
4  project_id = bot-stuff
5  # Create this secret in GCP Secret Mgr, holds GH token.
6  secret_name = gh_secret_token
7  # this is the label that gets set when a PR is opened.
8  label_name = cloudbot-testing
9
```

# TERRAFORM



# Overview: Terraform aka Deployment

The big picture for deployment.



# The Terraform Installer

We use Terraform to automate the Cloud Function installation.





# Deploying with Terraform

Let's do a Terraform deployment of the Python code to GCP Cloud Function.



# Github: Configure Webhook

Configure the webhook in the settings of each repo we want to add our bot to.

Webhooks / Manage webhook

Settings Recent Deliveries

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL \*

Content type

Secret

If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — [Change Secret](#)

# Github: Configure Webhook (cont.)

We will do a custom response, only to this single event.

## ☒ Pull requests

Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.

## ☐ Pushes

Git push to a repository.

# EXTRA



## Extra: Connect it to your GKE cluster

- Assuming you already have a GKE cluster, add a VPC connector so the cloud function can talk to the VPC the cluster is in.
- There is YAML in “yaml/cloudbot” that can be used to add a service to an existing GKE cluster.



- Execute the “bootstrap.sh” script from the top level of the repository.
- That should generate the “configure” script and the Makefiles listed in “configure.ac”
- Type “make python” at the top level to build all the python deps. Now you can do “. \_build/bin/activate” to get into Python venv.
- You can type “make docs” to build the PDF files from LaTeX.
- The docker directory has separate Makefile, type “make build” and “make push” from that directory.



# Extra: Dockerfile and docker-compose.yml

We use Docker to build the container we are working in.



# Extra: Github CI Pipeline

Information about what actions we are using and why.





# Future: Scan the PR comments for commands

The Cloud Function could monitor the PR for certain strings, using these to trigger actions.

