GOING SERVERLESS

with Google Cloud Run

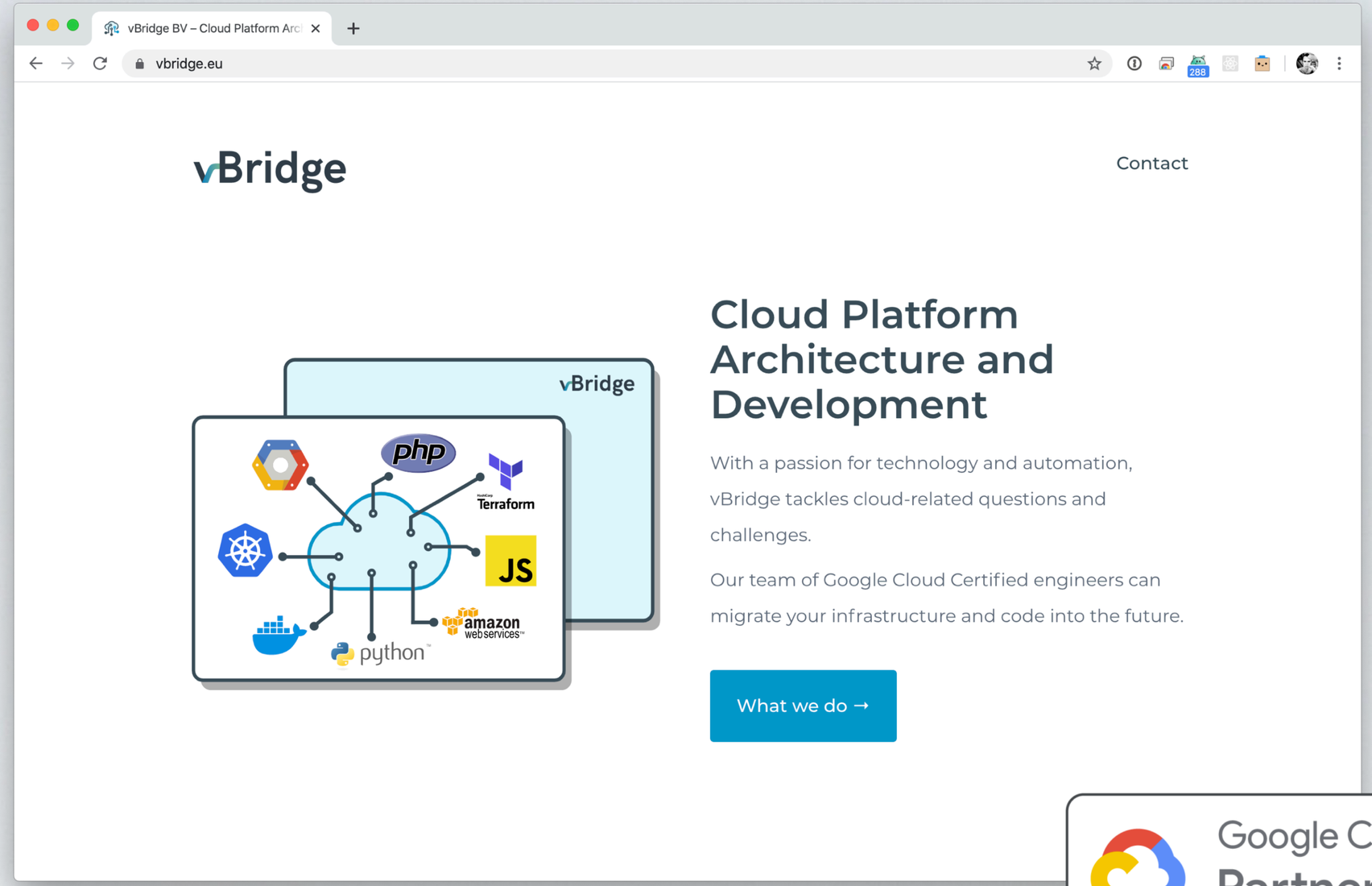
03/03/2020 @ PHPWVL

Yo!

Bram(us) Van Damme

https://www.bram.us/

@bramus
@bramusblog



Google Cloud Partner

https://www.vbridge.eu/

GOING SERVERLESS

with Google Cloud Run

SERVERLESS?

Serverless

"Serverless computing is a paradigm shift in application development that enables developers to focus on ***writing code without worrying about infrastructure****."*

https://cloud.google.com/serverless-options/

Benefits

*"It offers a variety of benefits over traditional computing, including **zero server management**, no up-front provisioning, **auto-scaling**, and paying only for the resources used."*

https://cloud.google.com/serverless-options/

Use Cases

*"These advantages make serverless **ideal for** use cases like **stateless HTTP applications**, web, mobile, IoT backends, batch and stream data processing, chatbots, and more."*

https://cloud.google.com/serverless-options/

SERVERLESS ON GOOGLE CLOUD PLATFORM

Serverless on Google Cloud Platform

App Engine

Cloud Functions

Cloud Run

https://cloud.google.com/serverless-options/

Release Dates

2008

2017

2019

https://cloud.google.com/serverless-options/

Units of deploy

Serverless
HTTP Applications

Serverless
Functions & Events

Serverless
Containers

https://cloud.google.com/serverless-options/

Supported Languages

Java, PHP,
Node, Python,
and Go

Node,
Python,
and Go

Any language,
Any library,
Any binary

https://cloud.google.com/serverless-options/

Configuration

app.yaml

runtime + memory

Dockerfile

https://cloud.google.com/serverless-options/

SERVERLESS ON CLOUD PLATFORM X

Serverless on Google Cloud Platform

App Engine

Cloud Functions

Cloud Run

https://cloud.google.com/serverless-options/

Serverless on Microsoft Azure

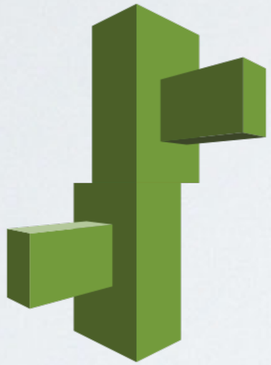
App Service

Functions

Container Instances

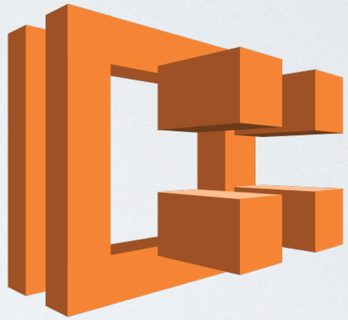
https://docs.microsoft.com/en-us/azure/

Serverless on Amazon AWS



Elastic Beanstalk

Lambda



Elastic Container Service

https://aws.amazon.com/

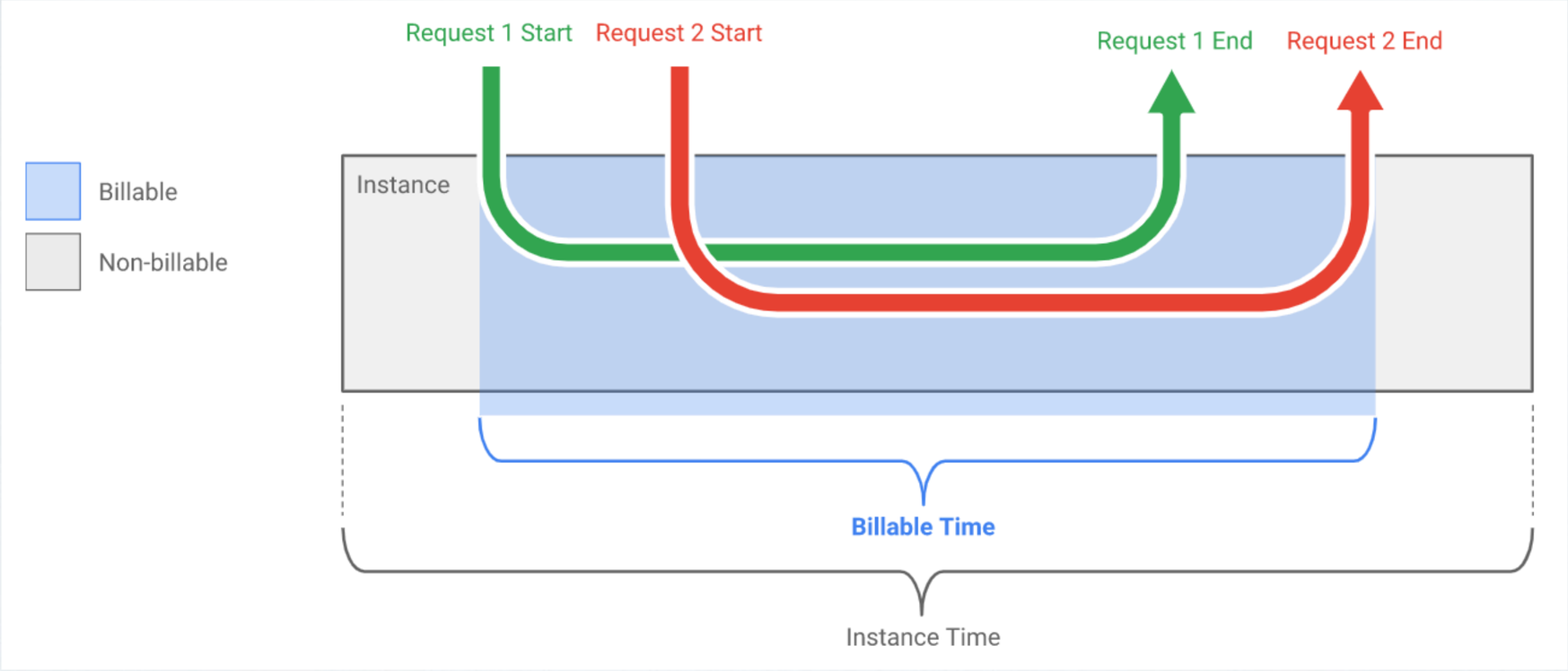
DEPLOYING TO GOOGLE CLOUD RUN

Google Cloud Run

- Runs containers
- Scales down to zero
- Built on Knative
  - Open API + Runtime Environment to run serverless workloads on a Kubernetes cluster
- Flavors
  - Cloud Run fully managed
  - Cloud Run for Anthos
- Pay-per-use
  - Billed to nearest 100ms

https://cloud.google.com/run

Cloud Run Pay-per-use



https://cloud.google.com/run/pricing

Prerequisites

- Google Cloud SDK
- GCP Account + Project
  - Link billing!
- Enabled Project Services
  - `cloudbuild.googleapis.com` *(optional)*
  - `containerregistry.googleapis.com`
  - `run.googleapis.com`

```
# Set correct Project ID
GC_PROJECT=cloudrun-example
GC_PROJECT_NUMBER=123456789
gcloud config set core/project $GC_PROJECT

# enable required services
gcloud services enable containerregistry.googleapis.com cloudbuild.googleapis.com run.googleapis.com
```

Simple PHP App + Dockerfile

```php
<?php

echo "Hello, world!";
```

```
FROM php:7-apache

# Cloud Run will inject a PORT envvar (default value: 8080), so we adjust Apache to listen on that
RUN sed -i 's/80/${PORT}/g' /etc/apache2/sites-available/000-default.conf /etc/apache2/ports.conf
RUN mv "$PHP_INI_DIR/php.ini-production" "$PHP_INI_DIR/php.ini"

COPY . /var/www/html
RUN chown -R www-data:www-data /var/www/html
```

For quick demo purposes only. This is not how you run PHP using Docker in production!

PORT Environment Variable: https://cloud.google.com/run/docs/reference/container-contract#port

Build + Run Container Locally

```
# Build image
docker build -t my-app .

# Run container
docker run -e PORT=8080 -p 8080:8080 -d --name my-container my-app

# Make request
curl localhost:8080

# Check logs
docker logs my-container

# Stop container
docker stop my-container
```

Build Locally + Deploy to Cloud Run

```
# Build Docker Image
docker build -t eu.gcr.io/$GC_PROJECT/my-app .

# Push Docker Image to Google Container Registry
docker push eu.gcr.io/$GC_PROJECT/my-app

# Deploy to Image from GCR onto Cloud Run
gcloud run deploy my-service \
    --image eu.gcr.io/$GC_PROJECT/my-app \
    --project $GC_PROJECT \
    --region "europe-west1" \
    --platform managed \
    --allow-unauthenticated
```

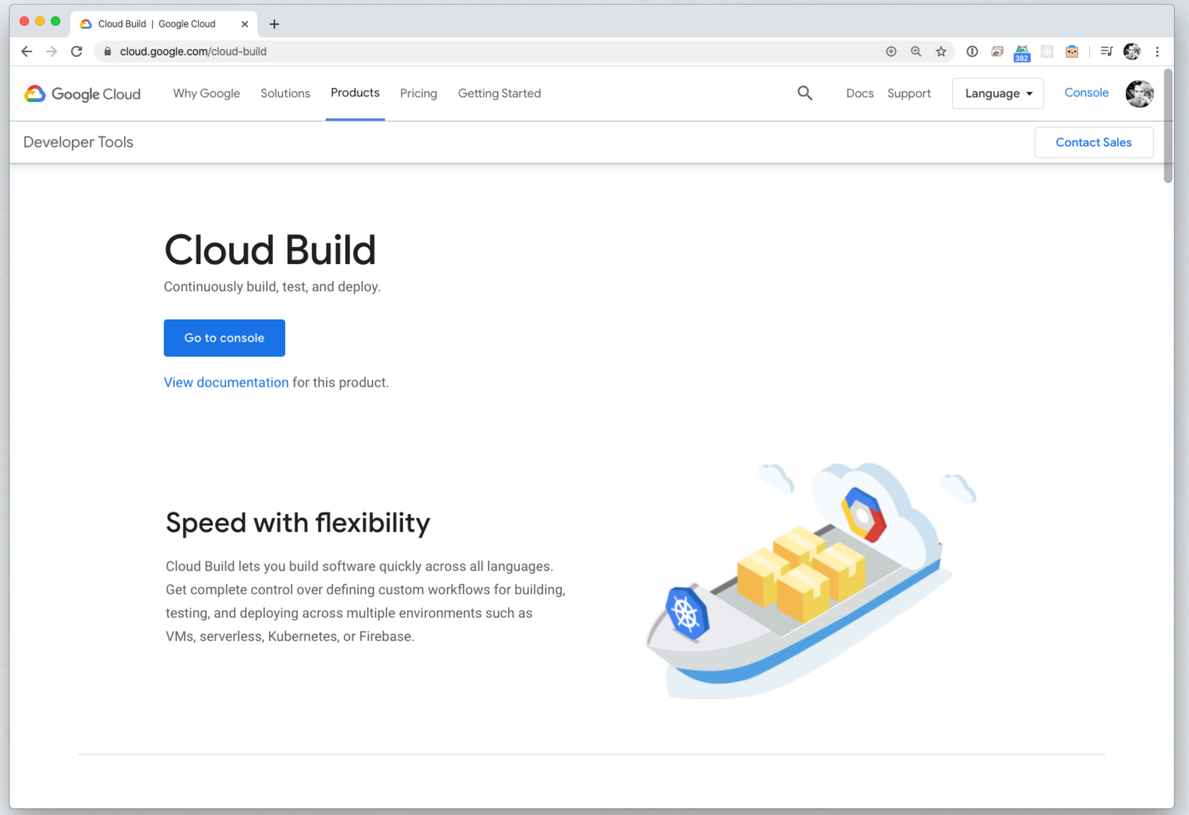
Tip: Set registry to eu.gcr.io as you're paying for egress out of the registry.

Build on Cloud Build + Deploy to Cloud Run

```
# Build (using Cloud Build)
# ???




# Deploy to Cloud Run
gcloud run deploy my-service \
    --image eu.gcr.io/$GC_PROJECT/my-app \
    --project $GC_PROJECT \
    --region "europe-west1" \
    --platform managed \
    --allow-unauthenticated
```

Cloud Build

- Build on Google's Infra
- Free 120 build-minutes/day
  - Additional build-minutes: $0.003 / minute
- 10 Concurrent Builds
- Native Docker support



https://cloud.google.com/cloud-build

Build on Cloud Build + Deploy to Cloud Run

```
# Build (using Cloud Build)
gcloud builds submit --tag eu.gcr.io/$GC_PROJECT/my-app




# Deploy to Cloud Run
gcloud run deploy my-service \
    --image eu.gcr.io/$GC_PROJECT/my-app \
    --project $GC_PROJECT \
    --region "europe-west1" \
    --platform managed \
    --allow-unauthenticated
```

Build on Cloud Build + Deploy to Cloud Run

```
# One-time: Register gcloud as a Docker credential helper
gcloud auth configure-docker
```

```
# Build (using Cloud Build)
gcloud builds submit --tag eu.gcr.io/$GC_PROJECT/my-app




# Deploy to Cloud Run
gcloud run deploy my-service \
    --image eu.gcr.io/$GC_PROJECT/my-app \
    --project $GC_PROJECT \
    --region "europe-west1" \
    --platform managed \
    --allow-unauthenticated
```

https://cloud.google.com/sdk/gcloud/reference/auth/configure-docker

Link your own domain

```
# Verify Domain
gcloud beta domains verify mydomain.tld

# Link it!
gcloud beta run domain-mappings create \
    --service my-service \
    --domain sub.mydomain.tld \
    --platform managed \
    --region europe-west1 \
    --project $GC_PROJECT
```

https://cloud.google.com/sdk/gcloud/reference/auth/configure-docker

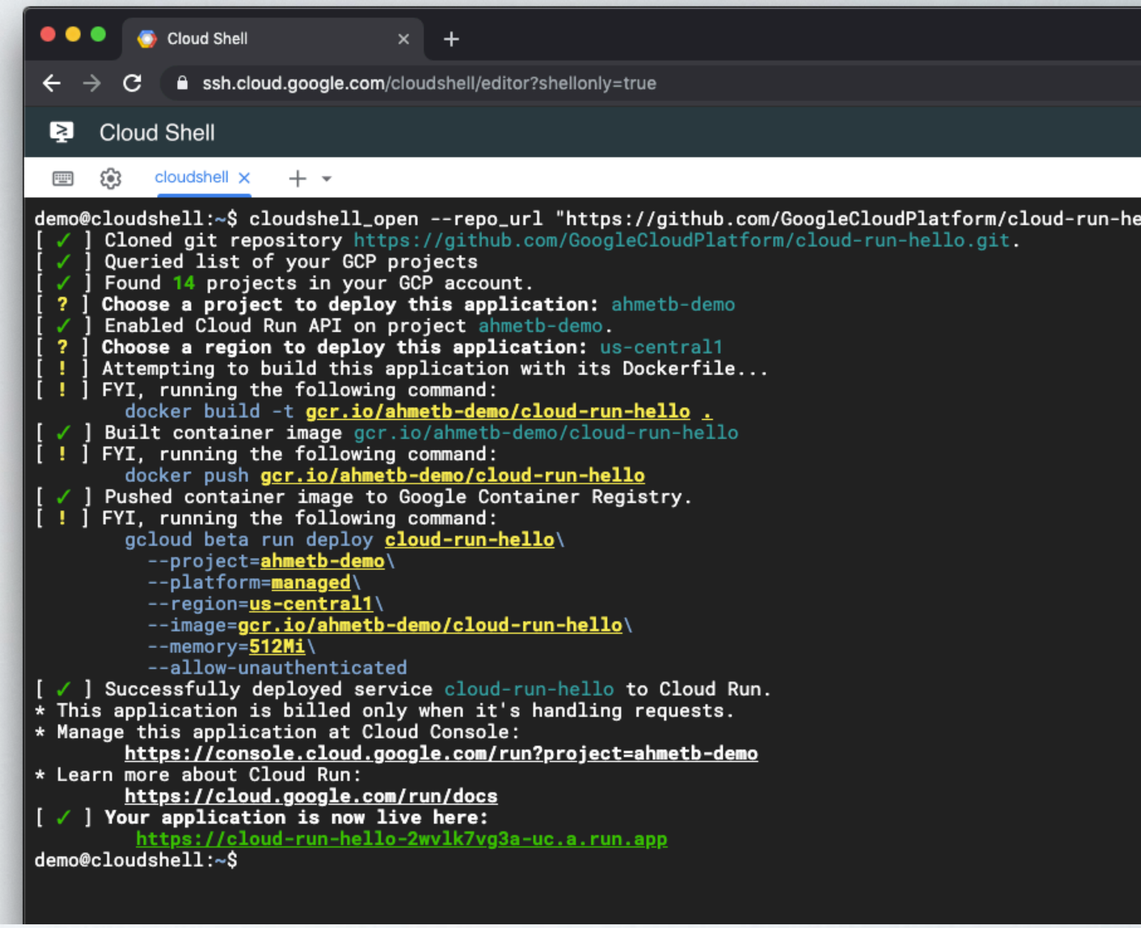
Clean up

```
# Clean up
gcloud run services delete my-service \
    --platform managed \
    --region "europe-west1"
```

DEPLOY
WITH A CLICK

Cloud Run Button

- Public Repos Only
- Link to https://deploy.cloud.run/
- Auto Clones, Builds, and Deploys
  - Cloud Shell



Run on Google Cloud

https://www.bram.us/2020/03/02/cloud-run-button-deploy-public-docker-images-to-google-cloud-run-with-a-single-click/

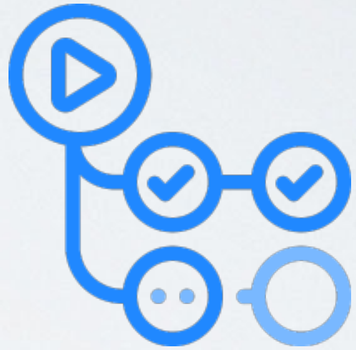
CONTINUOUS DEPLOYMENT

-

BUILD PIPELINE

Continuous Deployment

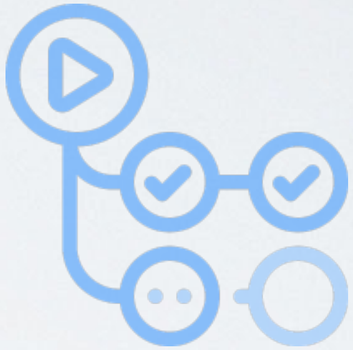
Cloud Build



GitHub Actions

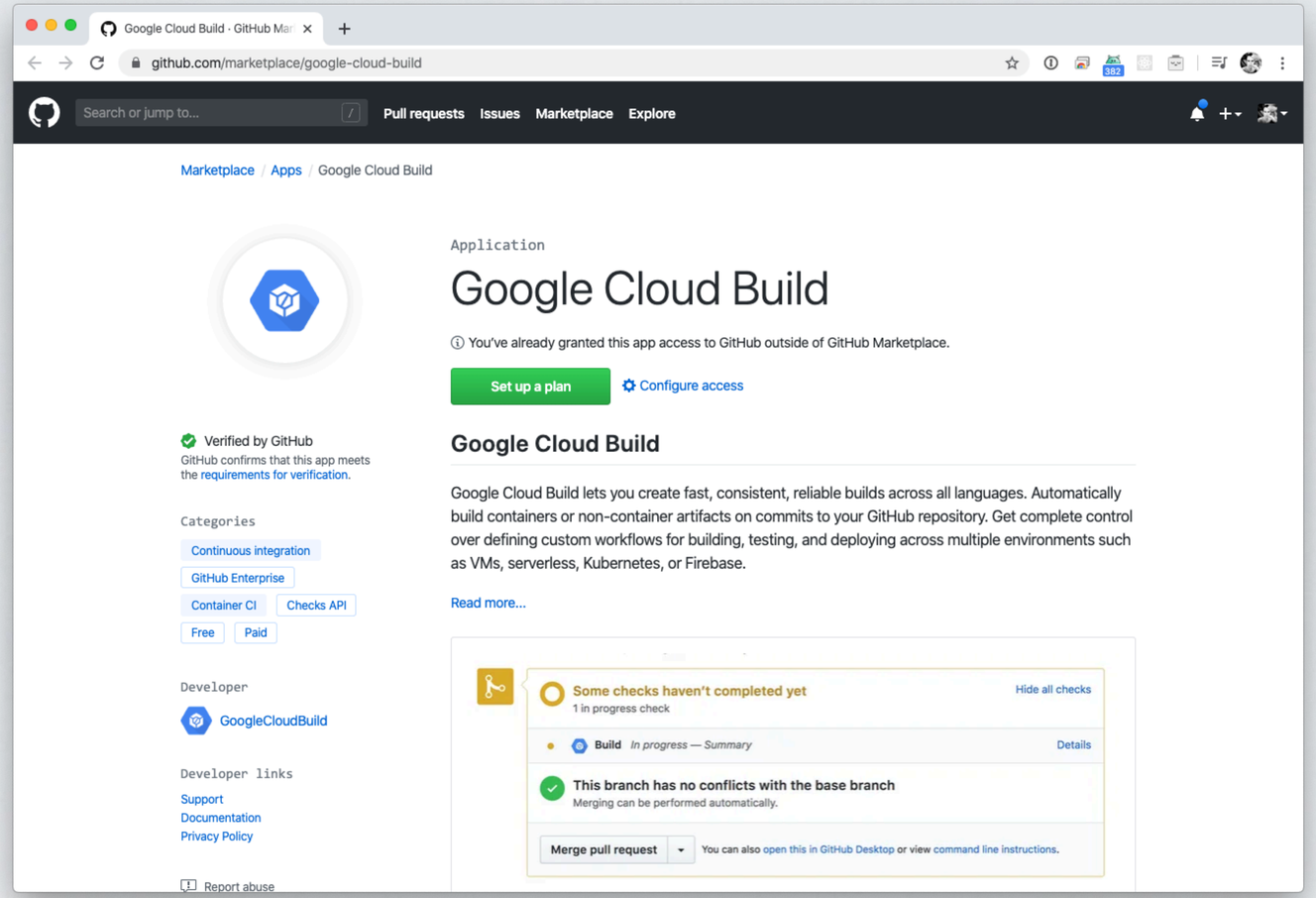
Continuous Deployment

Cloud Build



GitHub Actions

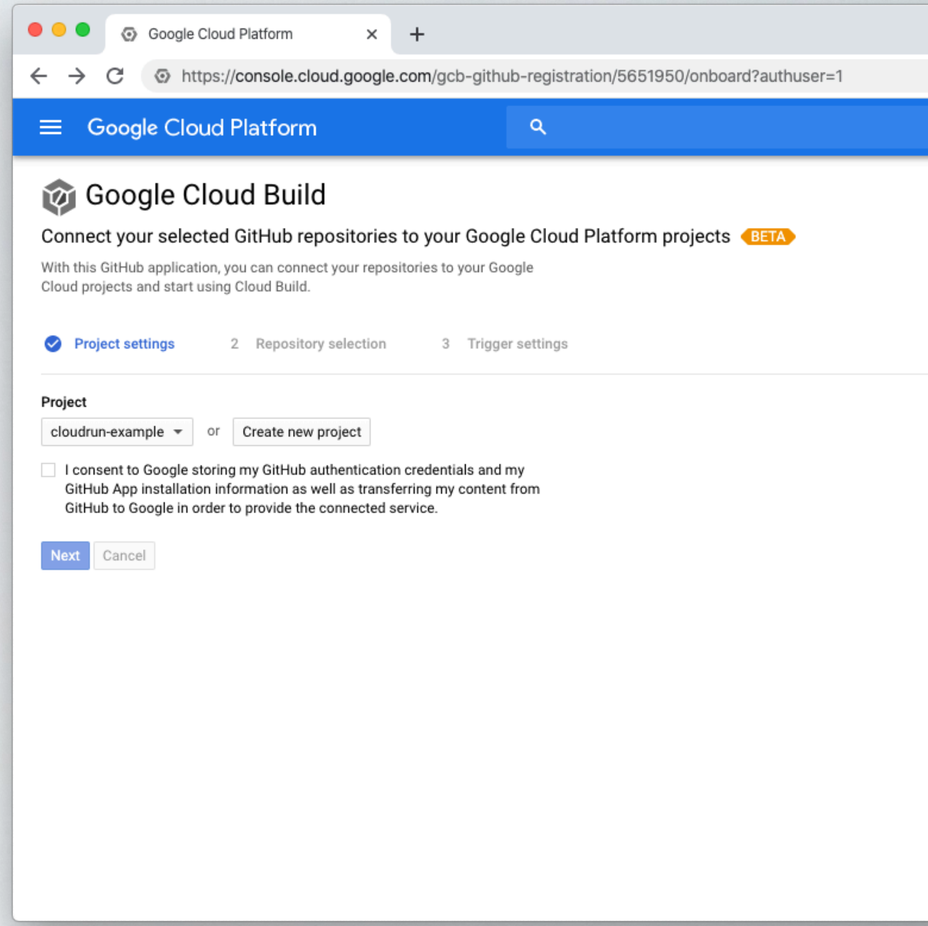
GitHub + Cloud Build



https://cloud.google.com/cloud-build/docs/run-builds-on-github
https://github.com/marketplace/google-cloud-build

GitHub + Cloud Build

- Manually link GitHub Repo to GCP Project
- Configure build with `cloudbuild.yaml`
  - If you don't it'll only build, not deploy
- Set up extra permissions for Cloud Build Service Account so that it may deploy to Cloud Run



https://cloud.google.com/run/docs/reference/iam/roles#additional-configuration

cloudbuild.yaml

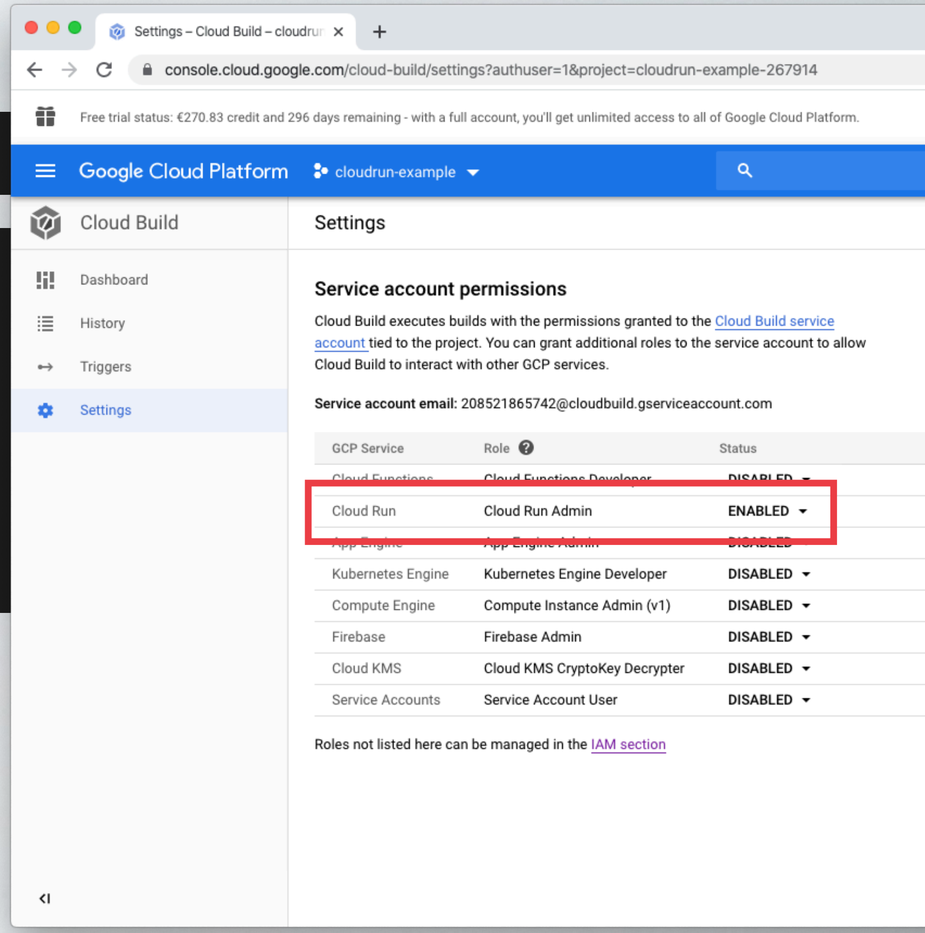
```yaml
steps:
# Build the container image
- name: 'gcr.io/cloud-builders/docker'
  args: [ 'build', '-t', 'eu.gcr.io/$PROJECT_ID/my-app', '.' ]

# Push the container image to Container Registry
- name: 'gcr.io/cloud-builders/docker'
  args: ['push', 'eu.gcr.io/$PROJECT_ID/my-app']

# Deploy container image to Cloud Run
- name: 'gcr.io/cloud-builders/gcloud'
  args: ['run', 'deploy', 'my-service', '--project', '$PROJECT_ID', '--image', 'eu.gcr.io/$PROJECT_ID/my-app', '--region',
'europe-west1', '--platform', 'managed', '--allow-unauthenticated']
```

https://cloud.google.com/cloud-build/docs/deploying-builds/deploy-cloud-run#building_and_deploying_a_container

Permissions

```
ERROR: (gcloud.run.deploy) PERMISSION_DENIED: The caller does not have permission
```

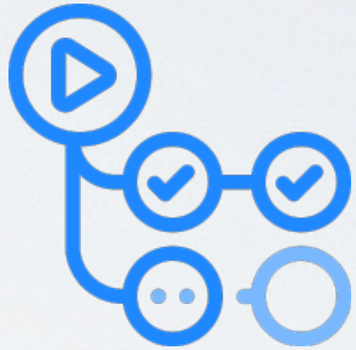
```
# Add Cloud Build SA as a "Cloud Run Admin"
gcloud projects add-iam-policy-binding $GC_PROJECT \
  --member "serviceAccount:$GC_PROJECT_NUMBER@cloudbuild.gserviceaccount.com" \
  --role roles/run.admin

# Allow Cloud Build SA to run as Default Compute SA
gcloud iam service-accounts add-iam-policy-binding \
  $GC_PROJECT_NUMBER-compute@developer.gserviceaccount.com \
  --member="serviceAccount:$GC_PROJECT_NUMBER@cloudbuild.gserviceaccount.com" \
  --role="roles/iam.serviceAccountUser"
```



https://www.bram.us/2020/02/13/google-cloud-build-google-cloud-run-fixing-error-gcloud-run-deploy-permission_denied-the-caller-does-not-have-permission/

Continuous Deployment

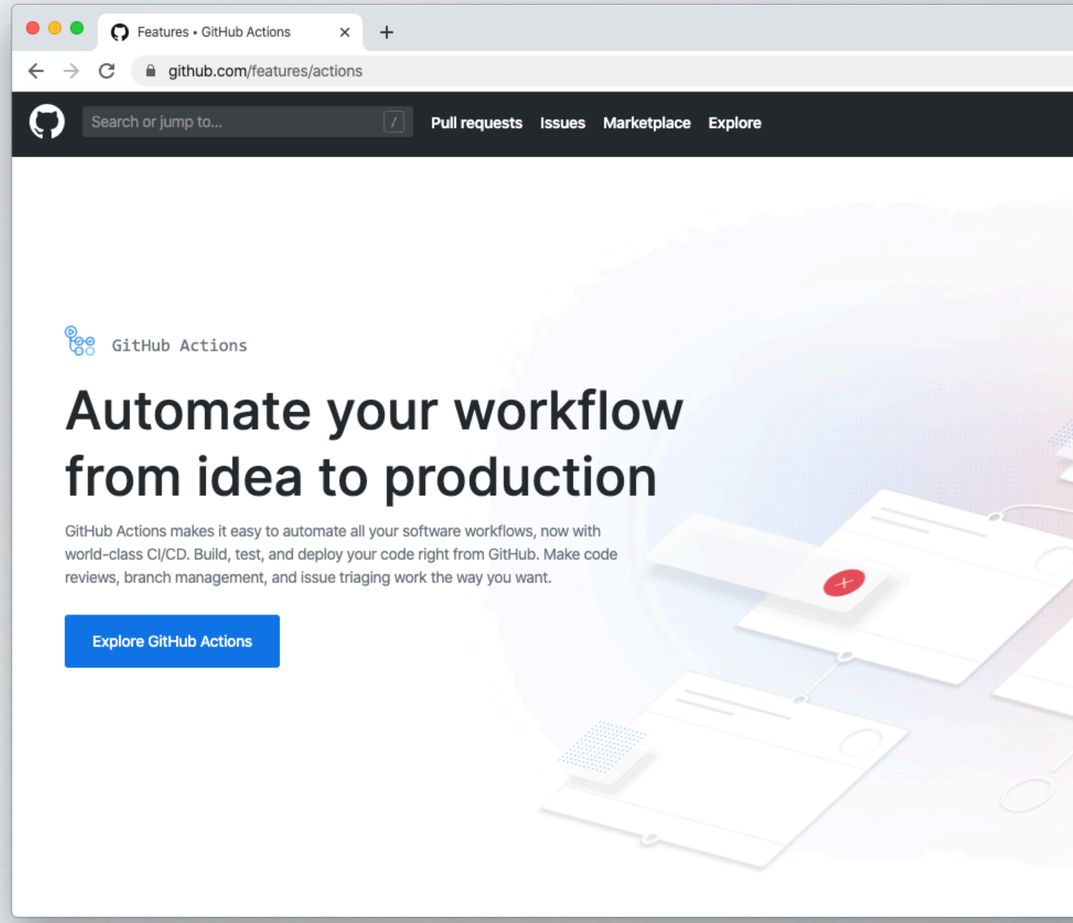
Cloud Build



GitHub Actions

Building with GitHub Actions

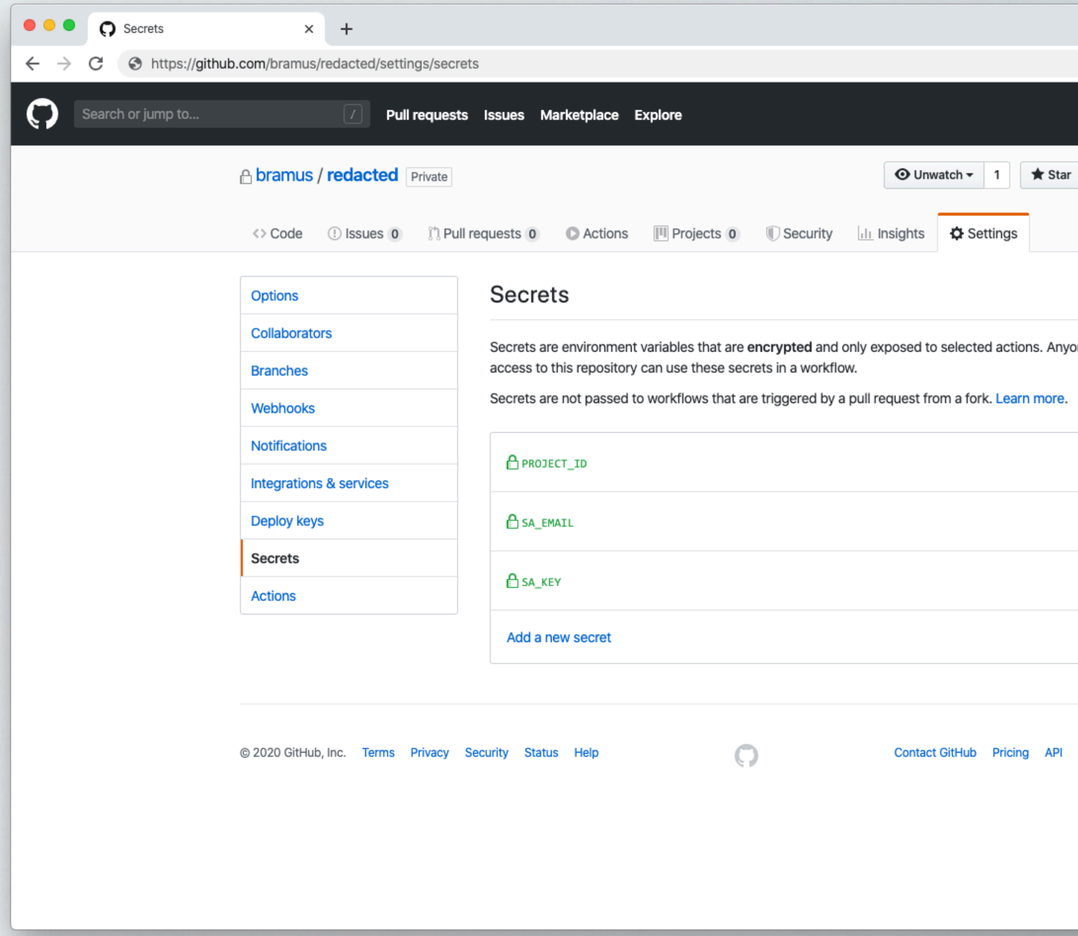
- Build on GitHub's Infra
- Free 2K build-minutes/month
  - Additional build-minutes: $0.008 / minute
- 20 Concurrent Builds
- Pro Account:
  - 3K minutes + 40 concurrent builds



https://help.github.com/en/actions/getting-started-with-github-actions/about-github-actions#usage-limits

Building with GitHub Actions

- Use Separate Service Account
  - Service Account User
  - Cloud Run Admin
  - Storage Admin
  - Storage Object Admin
- Generate key for SA



github-workflow.yaml (1/2)

```yaml
jobs:
  cloud-build-and-run:

    name: Setup, Build, Publish, and Deploy
    runs-on: ubuntu-latest

    steps:

    - name: Checkout Source Code
      uses: actions/checkout@v2

    - name: Setup gcloud CLI
      uses: GoogleCloudPlatform/github-actions/setup-gcloud@master
      with:
        version: '272.0.0'
        service_account_email: ${{ secrets.SA_EMAIL }}
        service_account_key: ${{ secrets.SA_KEY }}

    - name: Register gcloud as a Docker credential helper
      run: |
        gcloud auth configure-docker

…
```

github-workflow.yaml (2/2)

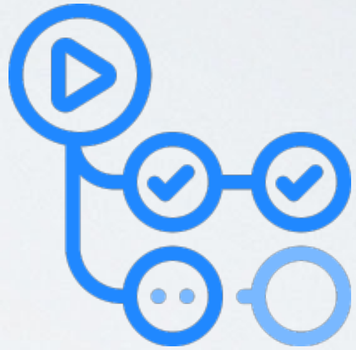
```
…

    - name: Build Docker Image
      run: |
        docker build \
          -t eu.gcr.io/"$PROJECT_ID"/"$IMAGE" \
          .

    - name: Publish Docker Image to Google Container Registry
      run: |
        docker push eu.gcr.io/$PROJECT_ID/$IMAGE

    - name: Deploy the Docker image to Cloud Run
      run: |
        gcloud run deploy "$SERVICE-$APP_ENV" \
          --image eu.gcr.io/"$PROJECT_ID"/"$IMAGE" \
          --set-env-vars "APP_ENV=$APP_ENV" \
          --project "$PROJECT_ID" \
          --region "$REGION" \
          --platform managed \
          --allow-unauthenticated
```

github-workflow.yaml (bonus)

```yaml
    steps:

    - name: Define APP_ENV for branch "Master" (= PRODUCTION)
      if: endsWith(github.ref, '/master')
      run: |
        echo "::set-env name=APP_ENV::production"

    - name: Define APP_ENV for branch "Develop" (= STAGING)
      if: endsWith(github.ref, '/develop')
      run: |
        echo "::set-env name=APP_ENV::staging"
```

Continuous Deployment

Cloud Build



GitHub Actions

vBridge

GOING SERVERLESS

with Google Cloud Run

@bramus – https://www.bram.us/

Thanks!

Bram(us) Van Damme

https://www.bram.us/

@bramus
@bramusblog

References

- https://cloud.google.com/run
- https://cloud.google.com/appengine/
- https://cloud.google.com/functions
- https://cloud.google.com/cloud-build/
- https://alexolivier.me/posts/deploy-container-stateless-cheap-google-cloud-run-serverless
- https://medium.com/swlh/say-hello-to-serverless-containers-with-cloud-run-4c32d90330fc
- https://cloudnative.ly/using-google-cloud-run-c2969d5950be
- https://github.com/armakuni/google-cloud-run-simplest-example/blob/master/Dockerfile
- https://dev.to/carlosazaustre/how-to-deploy-a-webapp-to-google-cloud-run-with-cloud-build-4eel
- https://sixtwothree.org/posts/continuous-deployment-from-github-to-google-cloud-run-using-build-triggers
- https://medium.com/google-cloud/knative-and-cloud-run-portability-in-action-1a655c914318
- https://medium.com/google-cloud/knative-to-cloud-run-f0ed1617e256