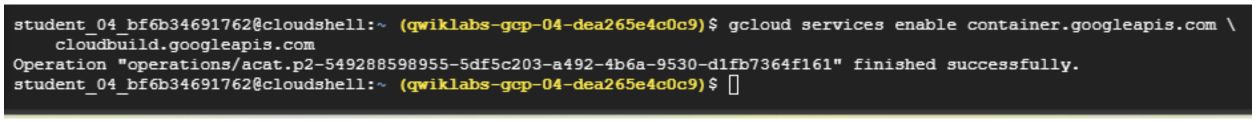
Implement DevOps Workflows in Google Cloud: Challenge Lab

Project Overview

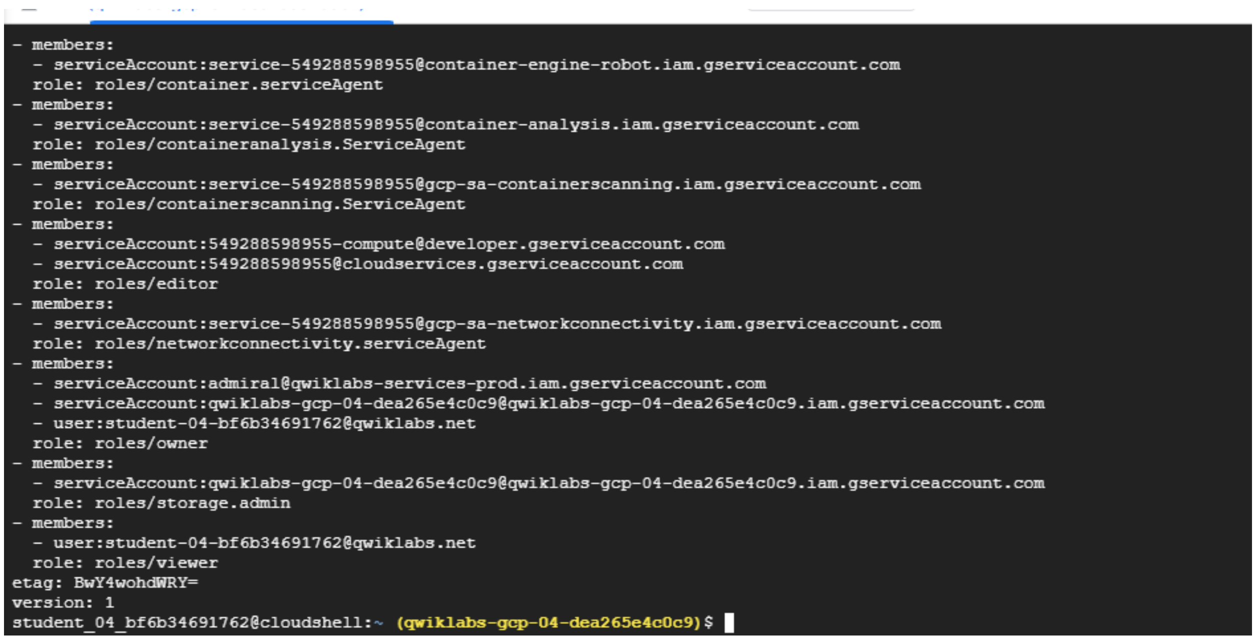
In this project, a fully automated **CI/CD pipeline** was created for a Go-based sample application using **Google Kubernetes Engine (GKE)**, **Cloud Build**, **Artifact Registry**, and **GitHub**. This pipeline deploys applications to separate **production** and **development** namespaces on GKE, with automated build, test, and deployment workflows triggered by GitHub branch updates.

Task 1: Create the Lab Resources

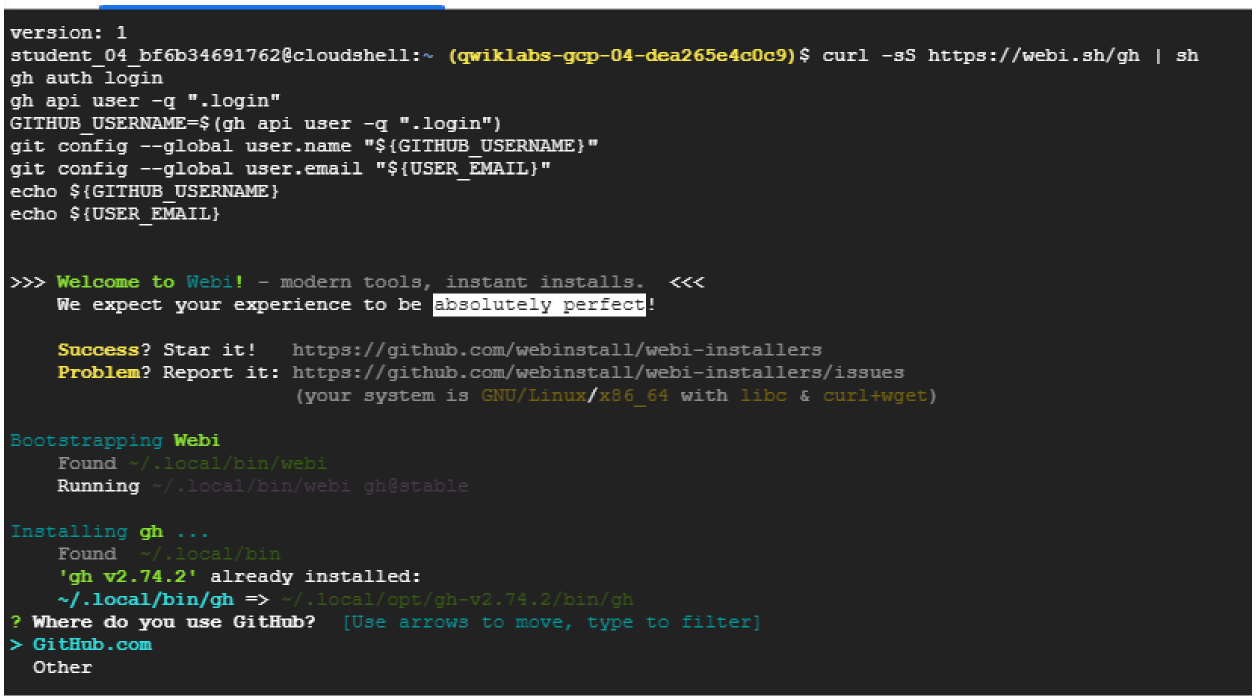
- **Enabled APIs:**
  - GKE (container.googleapis.com)
  - Cloud Build (cloudbuild.googleapis.com)
- **IAM Configuration:**
  - Added Kubernetes Developer role to Cloud Build service account.
- **Git and GitHub Configuration in Cloud Shell:**
  - Installed and authenticated GitHub CLI (gh).
  - Configured Git user details from GitHub account.
- **Artifact Registry:**
  - Created my-repository in the specified REGION for Docker images.
- **GKE Cluster:**
  - Created **Standard GKE cluster** named hello-cluster with:
    - Zone: ZONE
    - Release channel: Regular
    - Kubernetes version: 1.29+
    - Autoscaler: Enabled (2 min, 6 max nodes, 3 initial nodes)
  - Created prod and dev namespaces in the cluster.



```
student_04_bf6b34691762@cloudshell:~ (qwiklabs-gcp-04-dea265e4c0c9)$ gcloud services enable container.googleapis.com \
    cloudbuild.googleapis.com
Operation "operations/acat.p2-549288598955-5df5c203-a492-4b6a-9530-d1fb7364f161" finished successfully.
student_04_bf6b34691762@cloudshell:~ (qwiklabs-gcp-04-dea265e4c0c9)$
```



```
- members:
  - serviceAccount:service-549288598955@container-engine-robot.iam.gserviceaccount.com
  role: roles/container.serviceAgent
- members:
  - serviceAccount:service-549288598955@container-analysis.iam.gserviceaccount.com
  role: roles/containeranalysis.ServiceAgent
- members:
  - serviceAccount:service-549288598955@gcp-sa-containerscanning.iam.gserviceaccount.com
  role: roles/containerscanning.ServiceAgent
- members:
  - serviceAccount:549288598955-compute@developer.gserviceaccount.com
  - serviceAccount:549288598955@cloudservices.gserviceaccount.com
  role: roles/editor
- members:
  - serviceAccount:service-549288598955@gcp-sa-networkconnectivity.iam.gserviceaccount.com
  role: roles/networkconnectivity.serviceAgent
- members:
  - serviceAccount:admiral@qwiklabs-services-prod.iam.gserviceaccount.com
  - serviceAccount:qwiklabs-gcp-04-dea265e4c0c9@qwiklabs-gcp-04-dea265e4c0c9.iam.gserviceaccount.com
  - user:student-04-bf6b34691762@qwiklabs.net
  role: roles/owner
- members:
  - serviceAccount:qwiklabs-gcp-04-dea265e4c0c9@qwiklabs-gcp-04-dea265e4c0c9.iam.gserviceaccount.com
  role: roles/storage.admin
- members:
  - user:student-04-bf6b34691762@qwiklabs.net
  role: roles/viewer
etag: BwY4wohdWRY=
version: 1
student_04_bf6b34691762@cloudshell:~ (qwiklabs-gcp-04-dea265e4c0c9)$
```



```
version: 1
student_04_bf6b34691762@cloudshell:~ (qwiklabs-gcp-04-dea265e4c0c9)$ curl -sS https://webi.sh/gh | sh
gh auth login
gh api user -q ".login"
GITHUB_USERNAME=$(gh api user -q ".login")
git config --global user.name "${GITHUB_USERNAME}"
git config --global user.email "${USER_EMAIL}"
echo ${GITHUB_USERNAME}
echo ${USER_EMAIL}


>>> Welcome to Webi! - modern tools, instant installs.  <<<
    We expect your experience to be absolutely perfect!

    Success? Star it!   https://github.com/webinstall/webi-installers
    Problem? Report it: https://github.com/webinstall/webi-installers/issues
                        (your system is GNU/Linux/x86_64 with libc & curl+wget)

Bootstrapping Webi
    Found ~/.local/bin/webi
    Running ~/.local/bin/webi gh@stable

Installing gh ...
    Found ~/.local/bin
    'gh v2.74.2' already installed:
    ~/.local/bin/gh => ~/.local/opt/gh-v2.74.2/bin/gh
? Where do you use GitHub?  [Use arrows to move, type to filter]
> GitHub.com
  Other
```

Task 2: Create GitHub Repository

- Created a GitHub repository named sample-app.
- Cloned the repository into Cloud Shell.

- Copied sample Go application code from Cloud Storage.
- Updated cloudbuild-dev.yaml and cloudbuild.yaml with:
  - <your-region> → REGION
  - <your-zone> → ZONE
- Created and pushed:
  - master branch with initial commit.
  - dev branch with the same initial commit.

Task 3: Create Cloud Build Triggers

Created **two Cloud Build Triggers:**

1️⃣ **sample-app-prod-deploy**

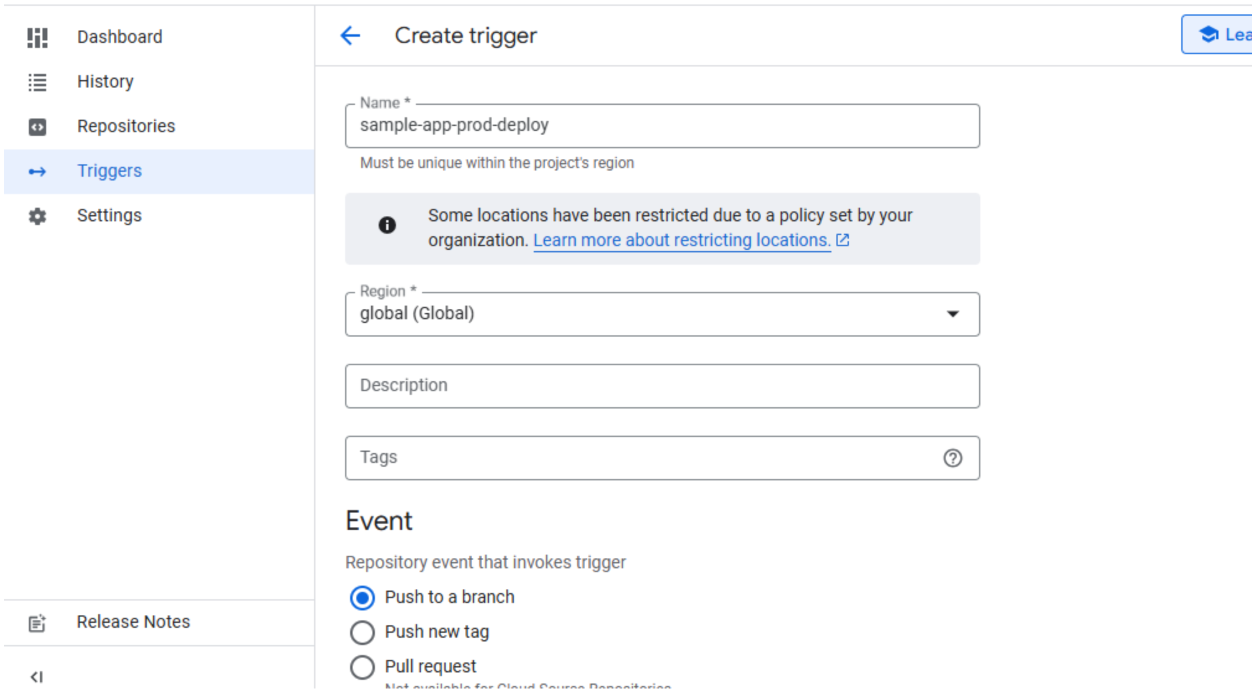
- Event: Push to master
- Source: GitHub repository sample-app
- Config: cloudbuild.yaml

2️⃣ **sample-app-dev-deploy**

- Event: Push to dev
- Source: GitHub repository sample-app
- Config: cloudbuild-dev.yaml

These triggers:

- Build Docker images.
- Push to Artifact Registry.
- Deploy to the respective GKE namespace (prod or dev).



Create trigger

Name *
sample-app-prod-deploy
Must be unique within the project's region

Some locations have been restricted due to a policy set by your organization. Learn more about restricting locations.

Region *
global (Global)

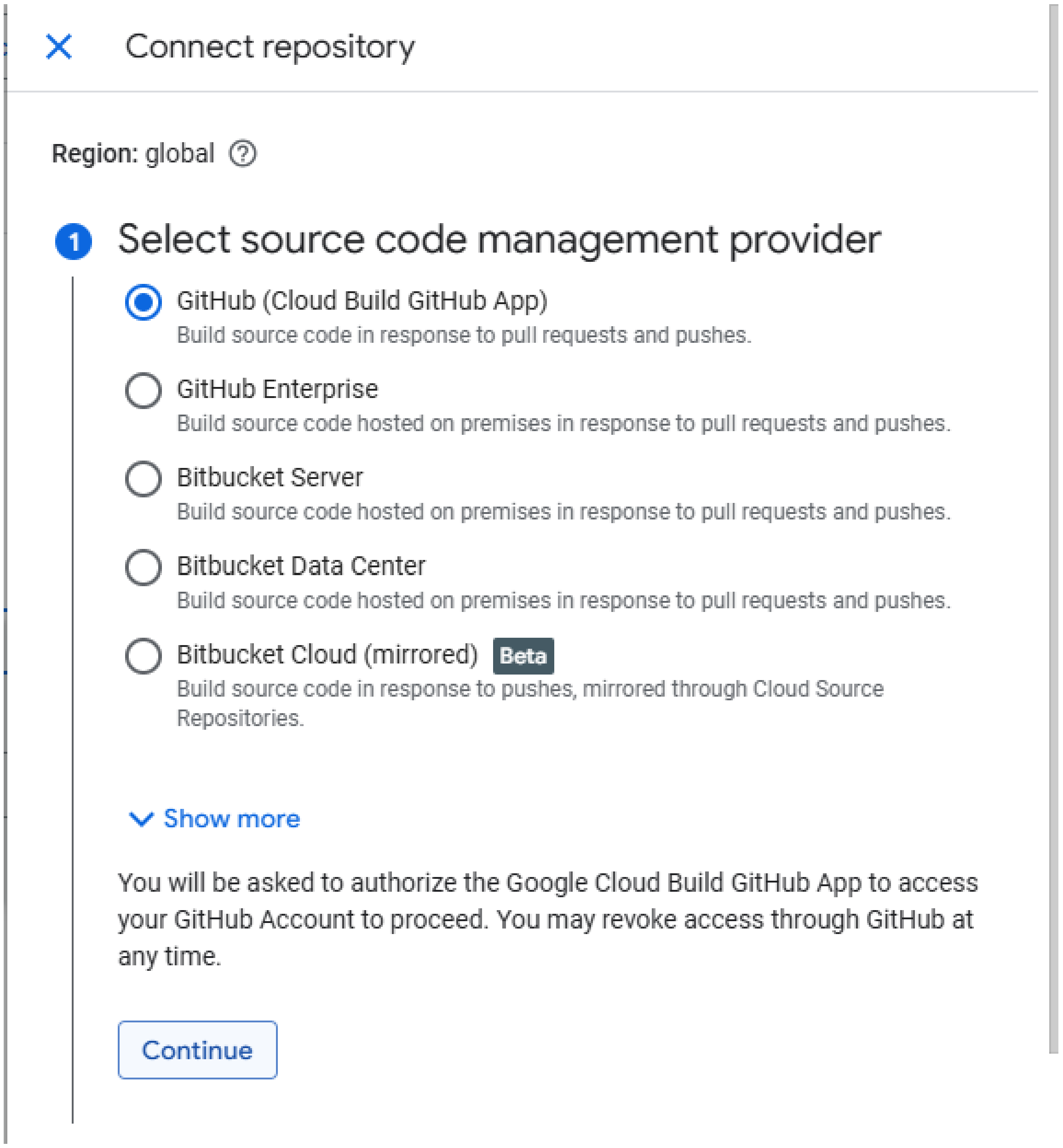
Description

Tags

Event

Repository event that invokes trigger

(●) Push to a branch
( ) Push new tag
( ) Pull request



Connect repository

Region: global

1 Select source code management provider

[✓ GitHub (Cloud Build GitHub App)]
Build source code in response to pull requests and pushes.

GitHub Enterprise
Build source code hosted on premises in response to pull requests and pushes.

Bitbucket Server
Build source code hosted on premises in response to pull requests and pushes.

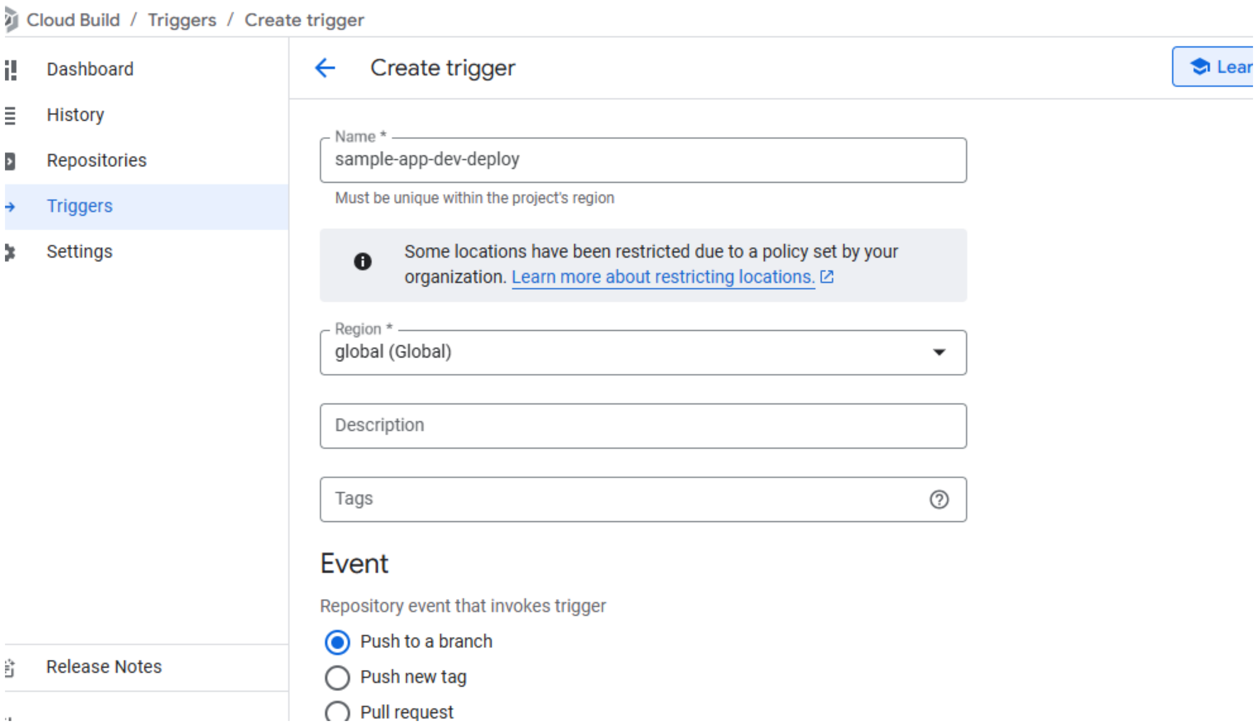
Bitbucket Data Center
Build source code hosted on premises in response to pull requests and pushes.

Bitbucket Cloud (mirrored) Beta
Build source code in response to pushes, mirrored through Cloud Source Repositories.

Show more

You will be asked to authorize the Google Cloud Build GitHub App to access your GitHub Account to proceed. You may revoke access through GitHub at any time.

Continue



Create trigger

Name *
sample-app-dev-deploy
Must be unique within the project's region

Some locations have been restricted due to a policy set by your organization. Learn more about restricting locations.

Region *
global (Global)

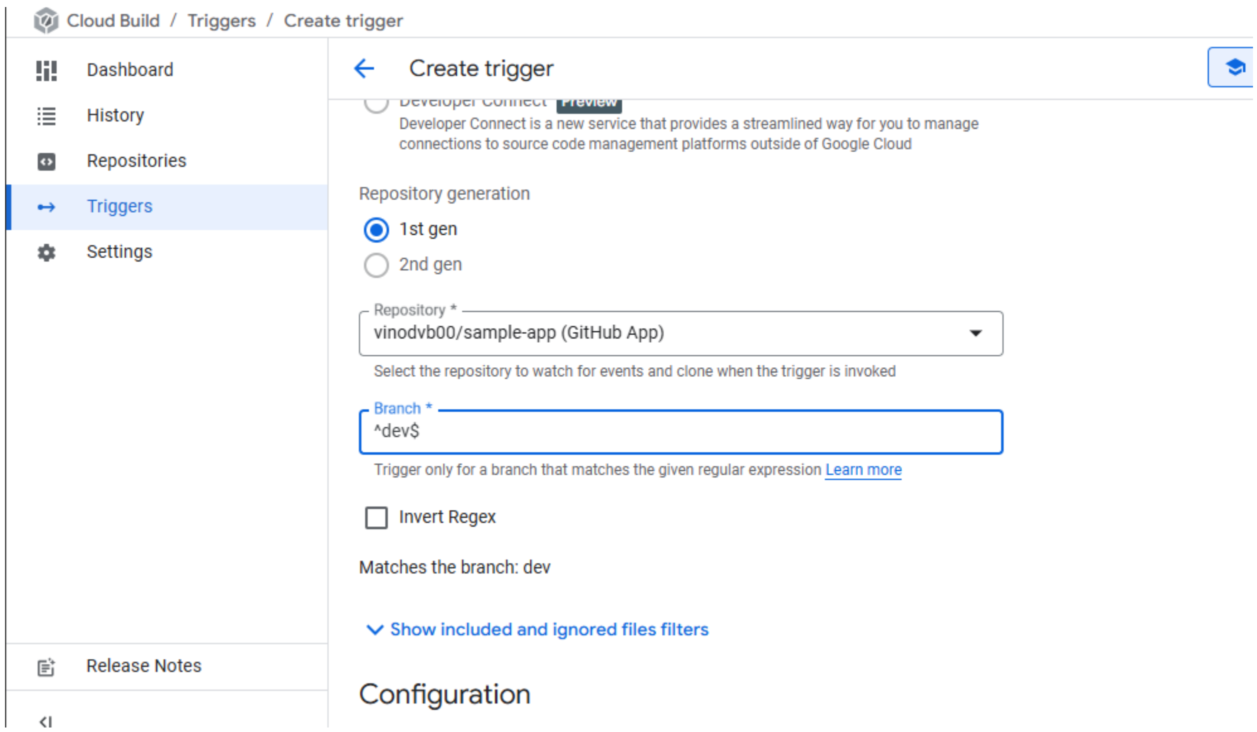
Description

Tags

Event

Repository event that invokes trigger

[✓ Push to a branch]
Push new tag
Pull request



Create trigger

Developer Connect
Developer Connect is a new service that provides a streamlined way for you to manage connections to source code management platforms outside of Google Cloud

Repository generation
[✓ 1st gen]
2nd gen

Repository *
vinodvb00/sample-app (GitHub App)
Select the repository to watch for events and clone when the trigger is invoked

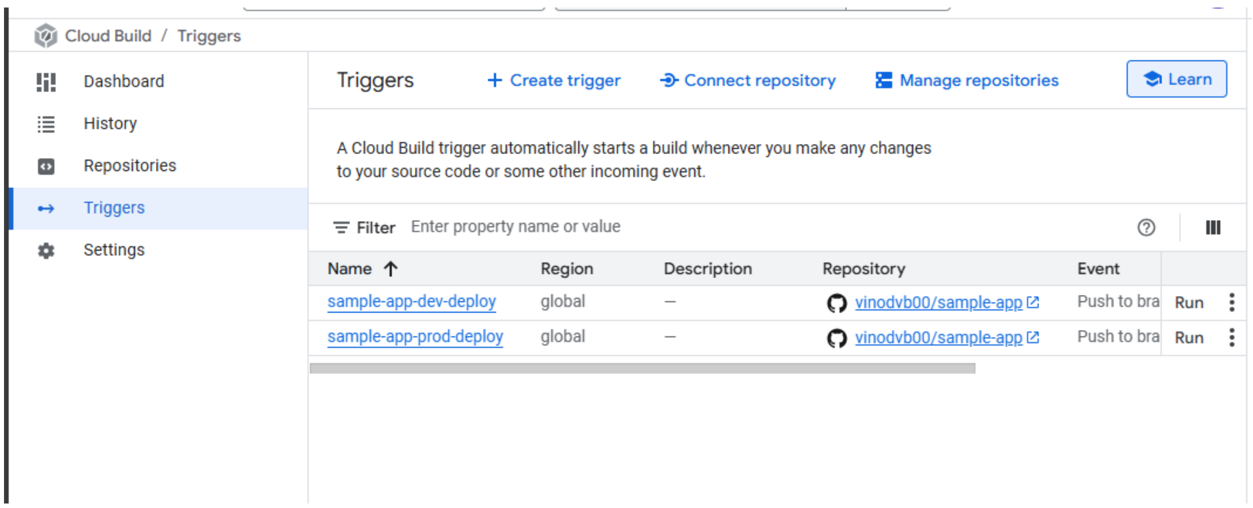
Branch *
^dev$
Trigger only for a branch that matches the given regular expression Learn more

☐ Invert Regex

Matches the branch: dev

Show included and ignored files filters

Configuration


Cloud Build / Triggers

Dashboard
History
Repositories
Triggers
Settings

Triggers + Create trigger Connect repository Manage repositories Learn

A Cloud Build trigger automatically starts a build whenever you make any changes to your source code or some other incoming event.

Filter Enter property name or value

Name ↑	Region	Description	Repository	Event		
sample-app-dev-deploy	global	–	vinodvb00/sample-app	Push to bra	Run	⋮
sample-app-prod-deploy	global	–	vinodvb00/sample-app	Push to bra	Run	⋮

Task 4: Deploy First Versions of the Application

Development Deployment:

- Updated cloudbuild-dev.yaml and dev/deployment.yaml with:
  - Version: v1.0
  - Correct container image name with PROJECT_ID.
- Committed and pushed to dev branch.
- Verified build and deployment.
- Exposed with LoadBalancer dev-deployment-service on port 8080.
- Verified endpoint:

Production Deployment:

- Updated cloudbuild.yaml and prod/deployment.yaml with:
  - Version: v1.0
  - Correct container image name with PROJECT_ID.
- Committed and pushed to master branch.
- Verified build and deployment.
- Exposed with LoadBalancer prod-deployment-service on port 8080.

Task 5: Deploy Second Versions of the Application

Added /red endpoint:

- Updated main.go with:
    - redHandler to serve a red square PNG.
    - Registered /red in main().

Development Deployment:

- Updated Docker image version to v2.0 in cloudbuild-dev.yaml and dev/deployment.yaml.
- Committed and pushed to dev branch.
- Verified build and deployment using:

Production Deployment:

- Updated Docker image version to v2.0 in cloudbuild.yaml and prod/deployment.yaml.
- Committed and pushed to master branch.
- Verified build and deployment using

Task 6: Roll Back the Production Deployment

- Used Cloud Build history to redeploy the v1.0 version.
- Verified the /red endpoint returned a 404 as expected, confirming the rollback.

Key Learnings and Takeaways

✅ Hands-on practice in setting up CI/CD pipelines on Google Cloud using Cloud Build.
✅ Automated builds and deployments using GitHub branch triggers.
✅ Managed Artifact Registry and GKE clusters with namespaces for environment separation.
✅ Learned version management and rollback mechanisms in GKE using Cloud Build history.
✅ Reinforced DevOps practices and Kubernetes deployment pipelines for microservices.