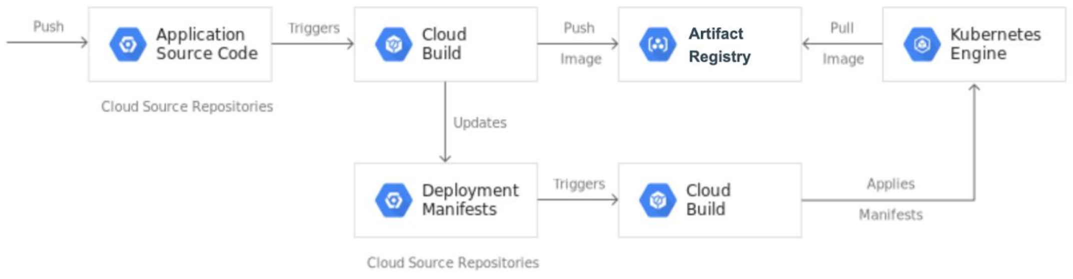
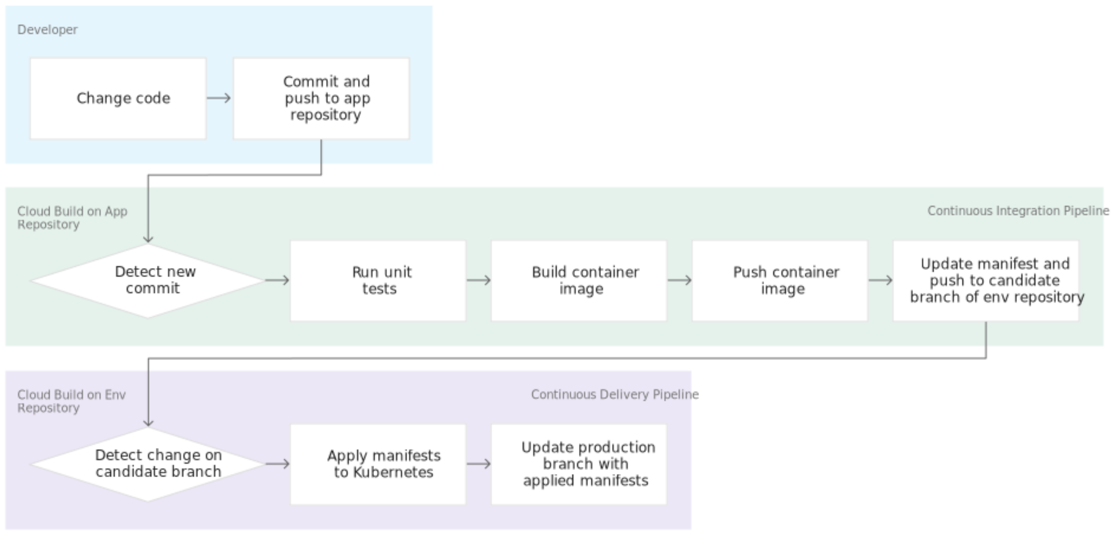
# Google Kubernetes Engine Pipeline using Cloud Build




Developer
Change code
Commit and push to app repository
Cloud Build on App Repository
Continuous Integration Pipeline
Detect new commit
Run unit tests
Build container image
Push container image
Update manifest and push to candidate branch of env repository
Cloud Build on Env Repository
Continuous Delivery Pipeline
Detect change on candidate branch
Apply manifests to Kubernetes
Update production branch with applied manifests

## Objectives

In this lab, you learn how to perform the following:

- Create Kubernetes Engine clusters.
- Create GitHub repositories.
- Trigger Cloud Build from GitHub repositories.
- Automate tests and publish deployable container image via Cloud Build.
- Manage resources deployed in a Kubernetes Engine cluster via Cloud Build.

## Task 1. Initialize your lab

In this task, you set up your environment:

- Import your project ID and project number as variables
- Enable the APIs for GKE, Cloud Build, Secret Manager, and Artifact Analysis
- Create an Artifact Registry Docker repository
- Create a GKE cluster to deploy the sample application of this lab

1. In Cloud Shell, run the following command to set your project ID and project number. Save them as PROJECT_ID and PROJECT_NUMBER variables:

   ```
   export PROJECT_ID=$(gcloud config get-value project) export
   PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --
   format='value(projectNumber)') export REGION= gcloud config set compute/region
   $REGION
   ```

2. Run the following command to enable the APIs for GKE, Cloud Build, Secret Manager and Artifact Analysis:

   ```
   gcloud services enable container.googleapis.com \
     cloudbuild.googleapis.com \
     secretmanager.googleapis.com \
     Containeranalysis.googleapis.com
   ```

3. Create an Artifact Registry Docker repository named my-repository in the <filled in at lab start> region to store your container images:

   ```
   gcloud artifacts repositories create my-repository \
    --repository-format=docker \
    --location=$REGION
   ```

4. Create a GKE cluster to deploy the sample application of this lab:

   ```
    gcloud container clusters create hello-cloudbuild --num-nodes 1 --region $REGION
   ```

5. Run the following command to configure Git and GitHub in Cloud Shell:

   ```
   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}
```

**Task 2. Create the Git repositories in GitHub repositories**

1. In Cloud Shell, run the following commands to create the two Git repositories:

   ```
   gh repo create  hello-cloudbuild-app –private
   gh repo create  hello-cloudbuild-env –private
   ```

2. Download the sample code from Cloud Storage:

   ```
   cd ~
   mkdir hello-cloudbuild-app
   gcloud storage cp -r gs://spls/gsp1077/gke-gitops-tutorial-cloudbuild/* hello-cloudbuild-app
   ```

3. Configure the GitHub repository as a remote:

   ```
   cd ~/hello-cloudbuild-app
   export REGION="REGION"
   sed -i "s/us-central1/$REGION/g" cloudbuild.yaml
   sed -i "s/us-central1/$REGION/g" cloudbuild-delivery.yaml
   sed -i "s/us-central1/$REGION/g" cloudbuild-trigger-cd.yaml
   sed -i "s/us-central1/$REGION/g" kubernetes.yaml.tpl PROJECT_ID=$(gcloud config get-value project)
   ```

```
from flask import Flask
app = Flask('hello-cloudbuild')
@app.route('/')
def hello():
  return "Hello World!\n"
if __name__ == '__main__':
  app.run(host = '0.0.0.0', port = 8080)
```

**Task 3. Create a container image with Cloud Build**

In this task, with an existing Dockerfile, you use Cloud Build to create and store a container image.

The code you previously cloned contains the Docker file:

```
FROM python:3.7-slim
RUN pip install flask
WORKDIR /app
COPY app.py /app/app.py
ENTRYPOINT ["python"]
CMD ["/app/app.py"]
```

1. In Cloud Shell, create a Cloud Build build based on the latest commit with the following command:

   cd ~/hello-cloudbuild-app

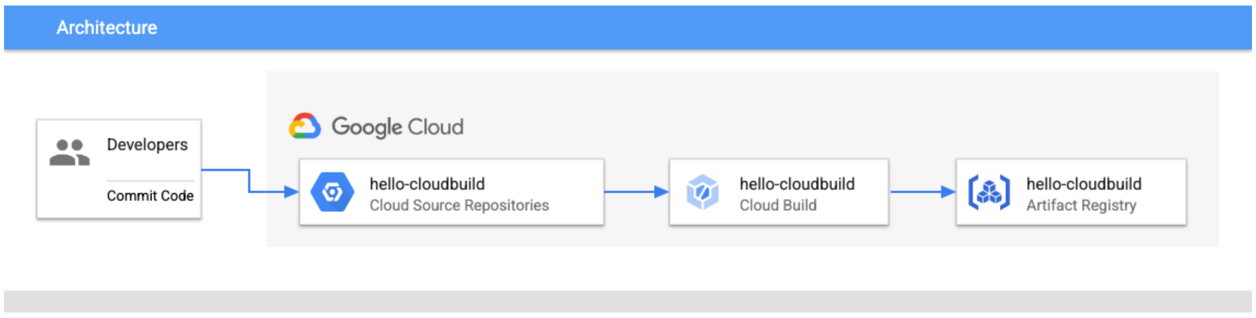
   COMMIT_ID="$(git rev-parse --short=7 HEAD)"

   gcloud builds submit --tag="${REGION}-docker.pkg.dev/${PROJECT_ID}/my-repository/hello-cloudbuild:${COMMIT_ID}" .

2. After the build finishes, in the Google title bar, enter **Artifact Registry** in the **Search** field, and then click **Artifact Registry** in the search results. Verify that your new container image is indeed available in Artifact Registry. Click **my-repository** to see the hello-cloudbuild image in the **Image** list.

**Task 4. Create the Continuous Integration (CI) pipeline**

In this task, you configure Cloud Build to automatically run a small unit test, build the container image, and then push it to Artifact Registry. Pushing a new commit to GitHub repositories triggers this pipeline automatically.



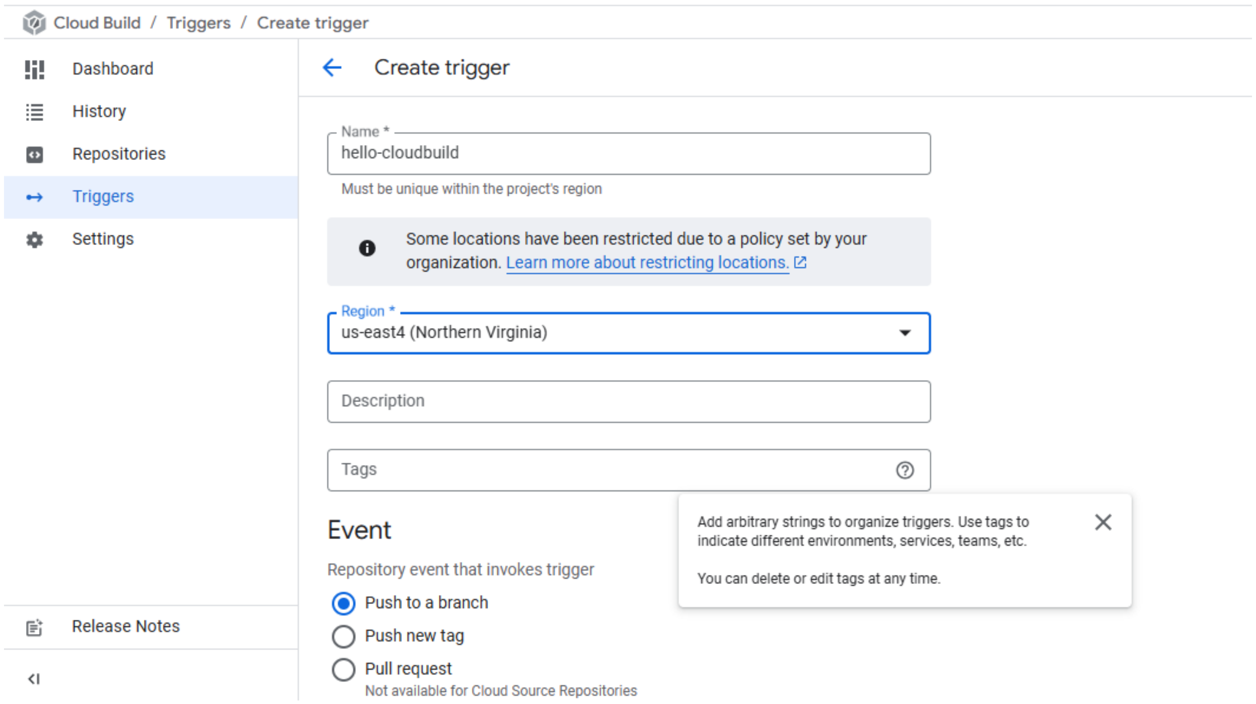
The cloudbuild.yaml file, already included in the code, is the pipeline's configuration.

1. In the console title bar, enter **Cloud Build triggers** in the **Search** field, and then click **Triggers, Cloud Build** in the search results.
2. Click **Create Trigger**.
3. For **Name**, type hello-cloudbuild. Set **Region** to <filled in at lab start>.
4. Set **Event** to **Push to a branch**.
5. Under **Source**, for **Repository**, click **Connect new repository**.

   a. Select **GitHub (Cloud Build GitHub App)**. Click **Continue**.

   b. Authenticate to your source repository with your username and password.

   c. If you get the pop up "The GitHub App is not installed on any of your repositories", follow these steps.



Name *
hello-cloudbuild
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 *
us-east4 (Northern Virginia)

Description

Tags

Add arbitrary strings to organize triggers. Use tags to indicate different environments, services, teams, etc.

You can delete or edit tags at any time.

Event

Repository event that invokes trigger

[✓ Push to a branch]

Push new tag

Pull request

Not available for Cloud Source Repositories

1. d. Select **${GITHUB_USERNAME}/hello-cloudbuild-app** for **Repository**. Click **OK**.

   e. Accept **I understand that GitHub content for the selected repositories...**.

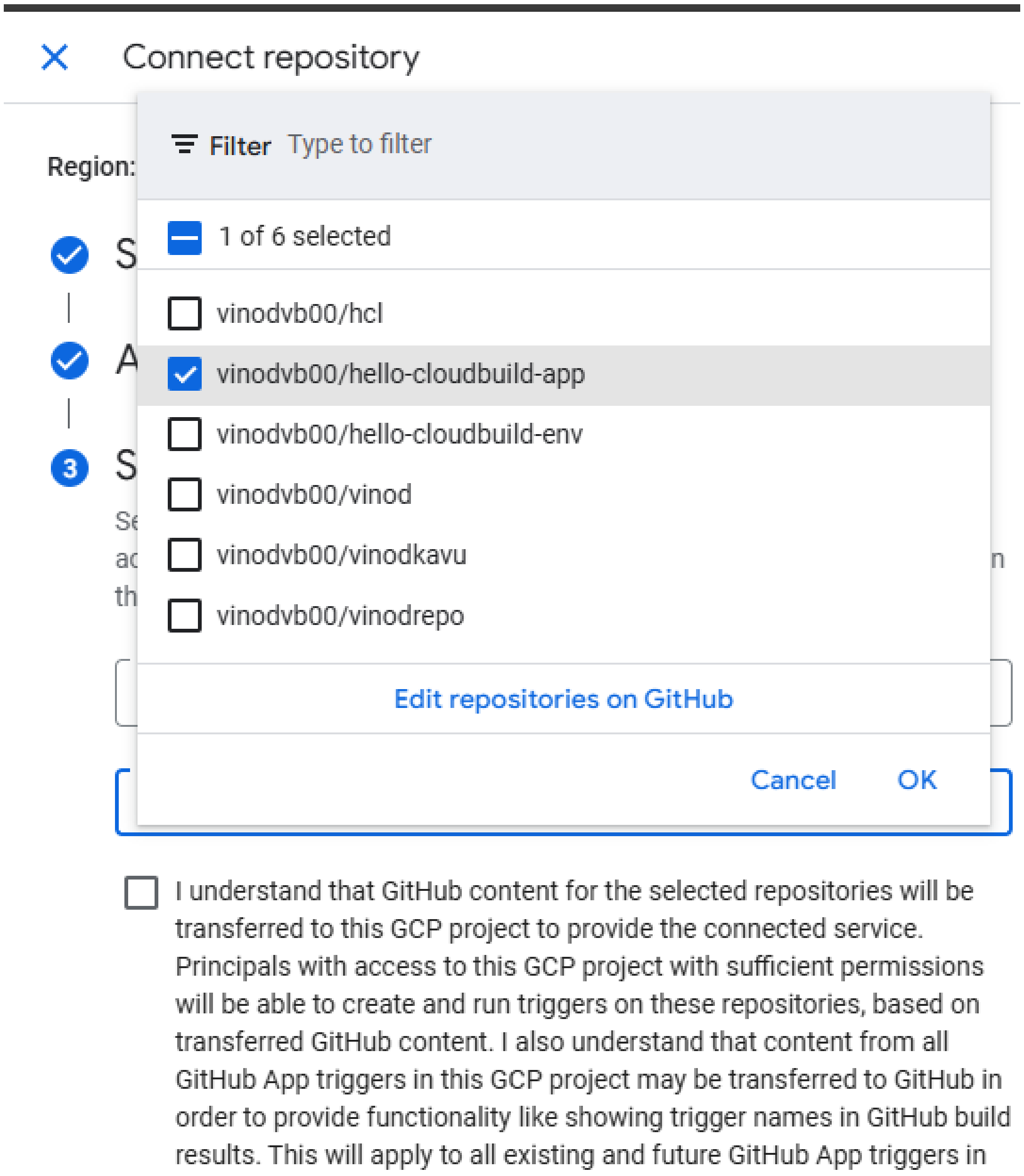
   f. Click **Connect**.

2. If the Cloud Build GitHub App is already installed in your account, you get the option to **Edit Repositories** on GitHub.

   a. Under **Repository access** choose **Only select repositories**. Click the **Select repositories** menu and select the repository **$${GITHUB_USERNAME}/hello-cloudbuild-app** and **$${GITHUB_USERNAME}/hello-cloudbuild-env**.

   b. Click **Save**.

3. On the Trigger page, from the **Repository list**, click **${GITHUB_USERNAME}/hello-cloudbuild-app**.
4. For **Branch** type .* (any branch).
5. In the **Configuration** section, set **Type** to **Cloud Build configuration file**.
6. In the **Location** field, type cloudbuild.yaml after the /.
7. Set **Service account** to the **Compute Engine default service account**.
8. Click **Create**.



Connect repository

Region:

Filter Type to filter

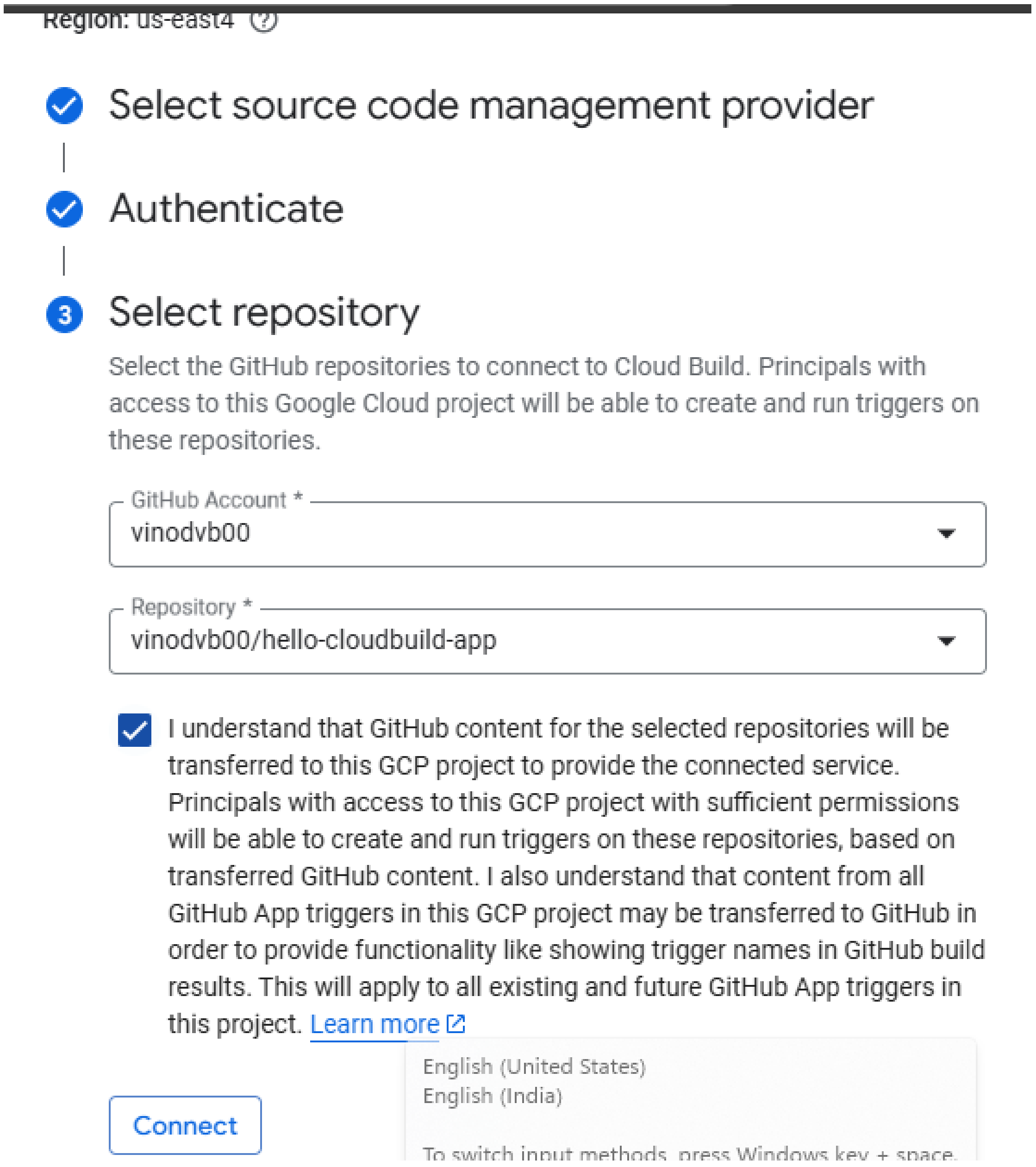
1 of 6 selected

- [ ] vinodvb00/hcl
- [x] vinodvb00/hello-cloudbuild-app
- [ ] vinodvb00/hello-cloudbuild-env
- [ ] vinodvb00/vinod
- [ ] vinodvb00/vinodkavu
- [ ] vinodvb00/vinodrepo

Edit repositories on GitHub

Cancel OK

- [ ] I understand that GitHub content for the selected repositories will be transferred to this GCP project to provide the connected service. Principals with access to this GCP project with sufficient permissions will be able to create and run triggers on these repositories, based on transferred GitHub content. I also understand that content from all GitHub App triggers in this GCP project may be transferred to GitHub in order to provide functionality like showing trigger names in GitHub build results. This will apply to all existing and future GitHub App triggers in



Region: us-east4

Select source code management provider

Authenticate

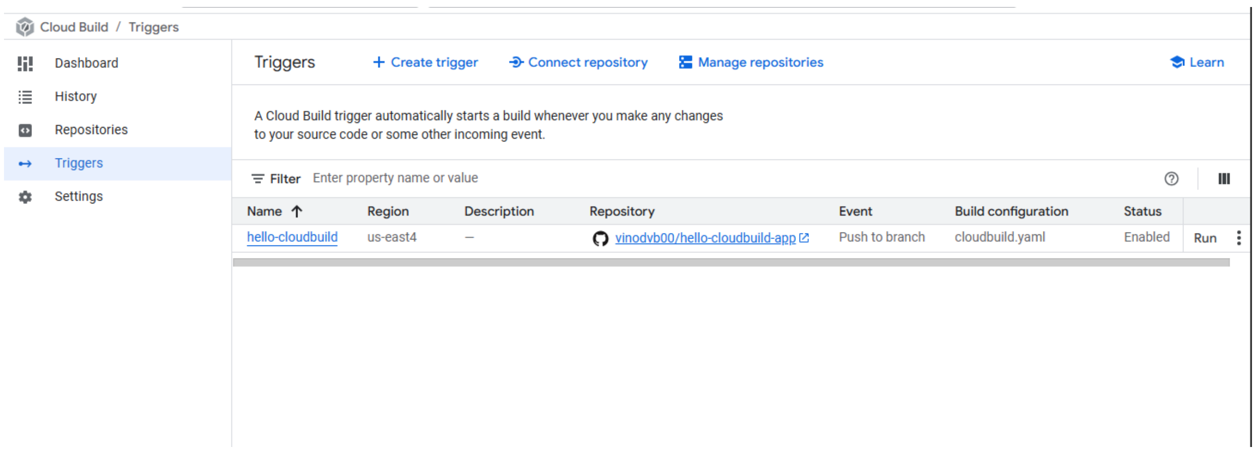
Select repository

Select the GitHub repositories to connect to Cloud Build. Principals with access to this Google Cloud project will be able to create and run triggers on these repositories.

GitHub Account *: vinodvb00

Repository *: vinodvb00/hello-cloudbuild-app

[✓] I understand that GitHub content for the selected repositories will be transferred to this GCP project to provide the connected service. Principals with access to this GCP project with sufficient permissions will be able to create and run triggers on these repositories, based on transferred GitHub content. I also understand that content from all GitHub App triggers in this GCP project may be transferred to GitHub in order to provide functionality like showing trigger names in GitHub build results. This will apply to all existing and future GitHub App triggers in this project. Learn more



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

| Name | Region | Description | Repository | Event | Build configuration | Status | |
|---|---|---|---|---|---|---|---|
| hello-cloudbuild | us-east4 | — | vinodvb00/hello-cloudbuild-app | Push to branch | cloudbuild.yaml | Enabled | Run |

**Task 5. Accessing GitHub from a build via SSH keys**

---

In this step use the Secret Manager with Cloud Build to access private GitHub repositories.

Create a SSH key

1. In Cloud Shell, change to the home directory.

   cd ~

   2.Create a new directory named workingdir and navigate to it:

   mkdir workingdir
   cd workingdir

1. Create a new GitHub SSH key, replace [*your-github-email*] with your personal GitHub email address:

   ssh-keygen -t rsa -b 4096 -N '' -f id_github -C [your-github-email]

4. In the Cloud Shell action bar, click **More** (⋮) and then **Download > Toggle file browser** and select the dropdown and workingdir folder to download the id_github file on your local machine.

Store the private SSH key in Secret Manager

1. In the console title bar, enter **Secret Manager**, and then click **Secret Manager** in the search results.
2. Click **Create Secret**.
3. Set **Name** to **ssh_key_secret**.
4. Set **Secret value** to **Upload** and upload your id_github file.
5. Leave other settings at their defaults.
6. Click **Create secret**.

Add the public SSH key to your private repository's deploy keys

1. Login to your personal GitHub account
2. In the top right corner, click your profile photo, then click **Your profile**.
3. On your profile page, click **Repositories**, then click the hello-cloudbuild-env repository.
4. From your repository, click **Settings**.

5. In the left pane, click **Deploy Keys**, then click **Add deploy key**.
6. Provide the title **SSH_KEY**, paste your public SSH key from workingdir/id_github.pub from Cloud Shell.
7. Select **Allow write access** so this key has write access to the repository. A deploy key with write access lets a deployment push to the repository.
8. Click **Add key**.

## Task 6. Create the test environment and CD pipeline

You can also use Cloud Build for the continuous delivery pipeline. The pipeline runs each time a commit is pushed to the candidate branch of the hello-cloudbuild-env repository. The pipeline applies the new version of the manifest to the Kubernetes cluster and, if successful, copies the manifest over to the production branch. This process has the following properties:

- The candidate branch is a history of the deployment attempts.
- The production branch is a history of the successful deployments.
- You have a view of successful and failed deployments in Cloud Build.
- You can rollback to any previous deployment by re-executing the corresponding build in Cloud Build. A rollback also updates the production branch to truthfully reflect the history of deployments.

Next, you modify the continuous integration pipeline to update the candidate branch of the hello-cloudbuild-env repository, triggering the continuous delivery pipeline.

Create the trigger for the continuous delivery pipeline

1. In the console title bar, enter **Cloud Build Triggers**, and then click **Triggers, Cloud Build**.
2. Click **Create Trigger**.
3. Set **Name** to **hello-cloudbuild-deploy**. Set **Region** to <filled in at lab start>.
4. Under **Event**, select **Push to a branch**.
5. Under **Source**, for **Repository** click **Connect new repository**.

   a. Select **GitHub (Cloud Build GitHub App)**. Click **Continue**.

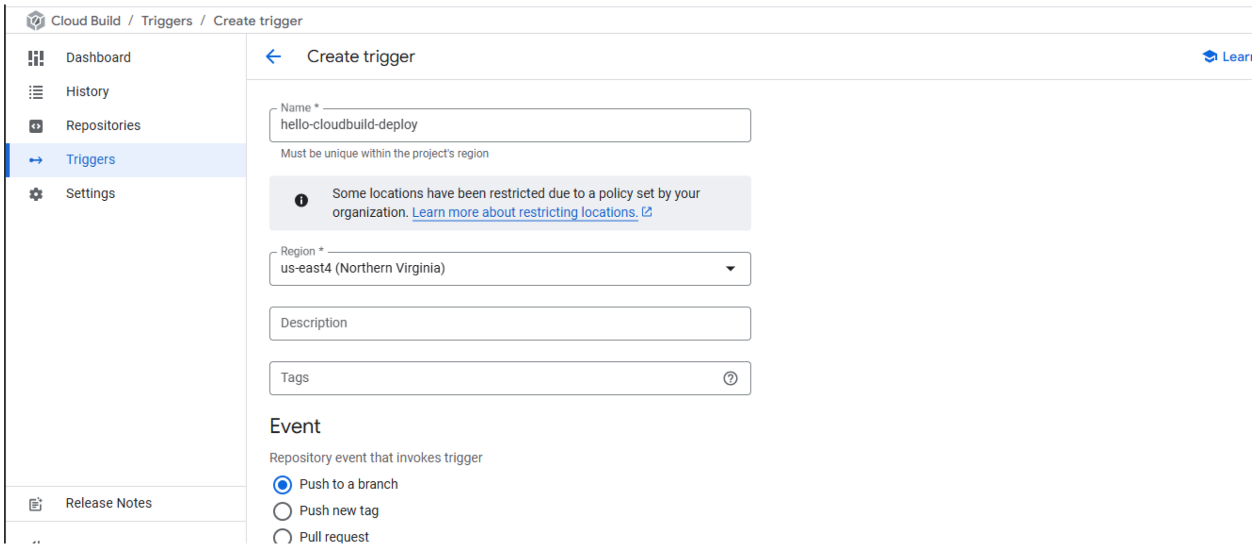
   b. Authenticate to your source repository with your GitHub username and password.

c. Select **${GITHUB_USERNAME}/hello-cloudbuild-env** repository. Click **OK**.

d. Select **I understand that GitHub content for the selected repositories..**

e. Click **Connect**.

6. Under **Repository** select **${GITHUB_USERNAME}/hello-cloudbuild-env**.
7. Under **Source**, select ^candidate$ as your **Branch**.
8. Under **Build configuration**, select **Cloud Build configuration file**.
9. In the **Cloud Build configuration file location** field, type cloudbuild.yaml after the /.
10. Set **Service account** to the Compute Engine default service account.
11. Click **Create**.



Name *
hello-cloudbuild-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 *
us-east4 (Northern Virginia)

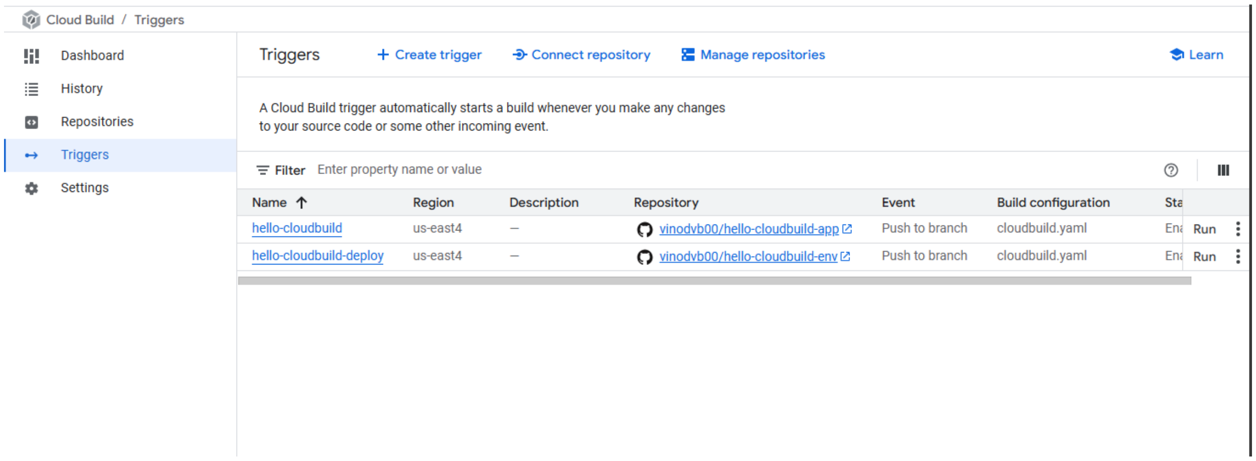
Description

Tags

Event

Repository event that invokes trigger

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



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

| Name ↑ | Region | Description | Repository | Event | Build configuration | Sta | | |
|---|---|---|---|---|---|---|---|---|
| hello-cloudbuild | us-east4 | – | vinodvb00/hello-cloudbuild-app | Push to branch | cloudbuild.yaml | Ena | Run | ⋮ |
| hello-cloudbuild-deploy | us-east4 | – | vinodvb00/hello-cloudbuild-env | Push to branch | cloudbuild.yaml | Ena | Run | ⋮ |

1. In your hello-cloudbuild-app directory, create a file named known_hosts.github, add the public SSH key to this file and provide the necessary permission to the file:

   ```
   cd ~/hello-cloudbuild-app

   ssh-keyscan -t rsa github.com > known_hosts.github

   chmod +x known_hosts.github

   git add .

   git commit -m "Adding known_host file."

   git push google master
   ```

## Task 7. Review Cloud Build pipeline

In this task, you review the Cloud Build pipeline in the console.

1. In the console, still in the Cloud Build page, click **Dashboard** in the left pane.
2. Click the **hello-cloudbuild-app** trigger to follow its execution and examine its logs. The last step of this pipeline pushes the new manifest to the hello-cloudbuild-env repository, which triggers the continuous delivery pipeline.
3. Return to the main **Dashboard**.
4. You should see a build either running or recently finished for the hello-cloudbuild-env repository.

   You can click on the build to follow its execution and examine its logs.

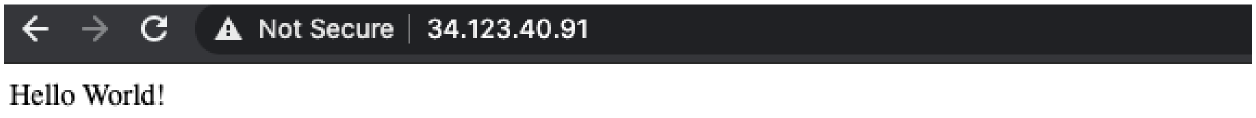
## Task 8. Test the complete pipeline

You've now configured the complete CI/CD pipeline. In this task you perform an end to end test.

1. In the console, in the **Navigation menu** (☰), click **Kubernetes Engine > Gateways, Services & Ingress > Services**.

   There should be a single service called **hello-cloudbuild** in the list. It has been created by the continuous delivery build that just ran.

2. Click on the endpoint for the **hello-cloudbuild** service. You should see "Hello World!". If there is no endpoint, or if you see a load balancer error, you may have to wait a few minutes for the load balancer to completely initialize. If needed, click **Refresh** to update the page.



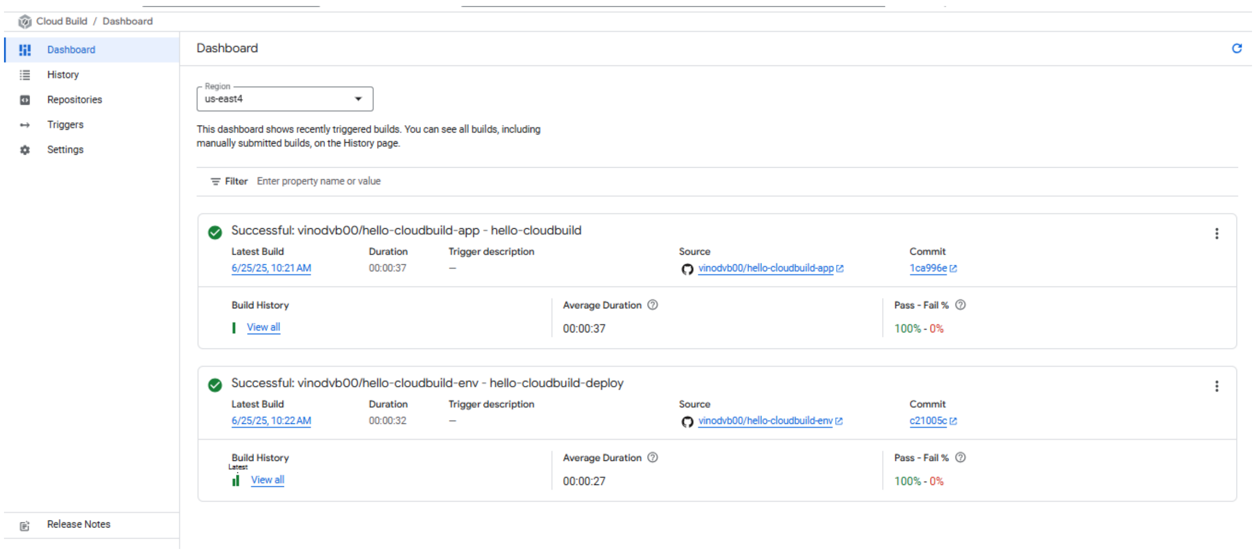
Not Secure | 34.123.40.91

Hello World!

## Task 9. Test the rollback

In this task, you rollback to the version of the application that said "Hello World!".

1. In the console title bar, type **Cloud Build Dashboard** in the **Search** field, and then click Cloud Build in the search results. Be sure **Dashboard** is selected in the left pane.
2. Click the **View all** link under **Build History** for the hello-cloudbuild-env repository.
3. Click on the second most recent build available.



This dashboard shows recently triggered builds. You can see all builds, including manually submitted builds, on the History page.

Successful: vinodvb00/hello-cloudbuild-app - hello-cloudbuild

| Latest Build | Duration | Trigger description | Source | Commit |
|---|---|---|---|---|
| 6/25/25, 10:21 AM | 00:00:37 | – | vinodvb00/hello-cloudbuild-app | 1ca996e |

| Build History | Average Duration | Pass - Fail % |
|---|---|---|
| View all | 00:00:37 | 100% - 0% |

Successful: vinodvb00/hello-cloudbuild-env - hello-cloudbuild-deploy

| Latest Build | Duration | Trigger description | Source | Commit |
|---|---|---|---|---|
| 6/25/25, 10:22 AM | 00:00:32 | – | vinodvb00/hello-cloudbuild-env | c21005c |

| Build History | Average Duration | Pass - Fail % |
|---|---|---|
| Latest View all | 00:00:27 | 100% - 0% |