



CI/CD for Kubeflow Pipelines on Vertex AI



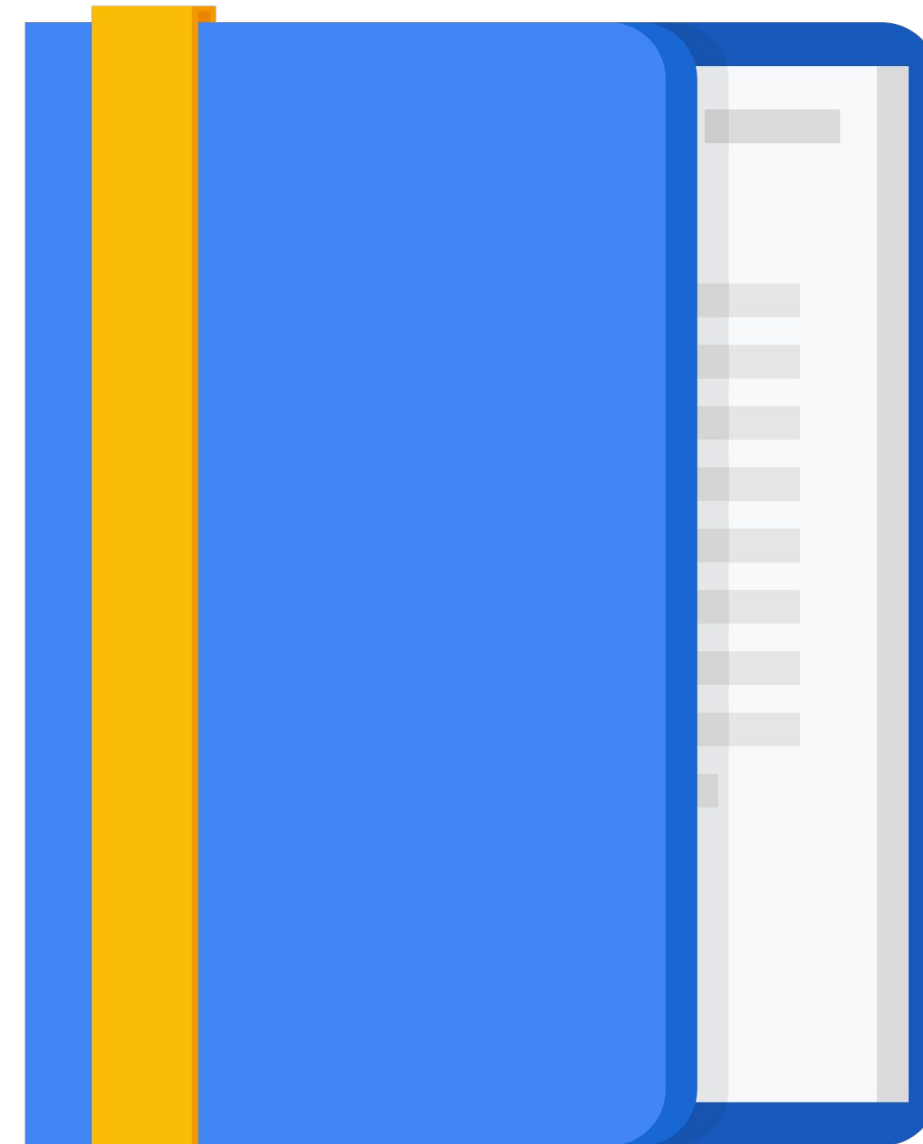
Agenda

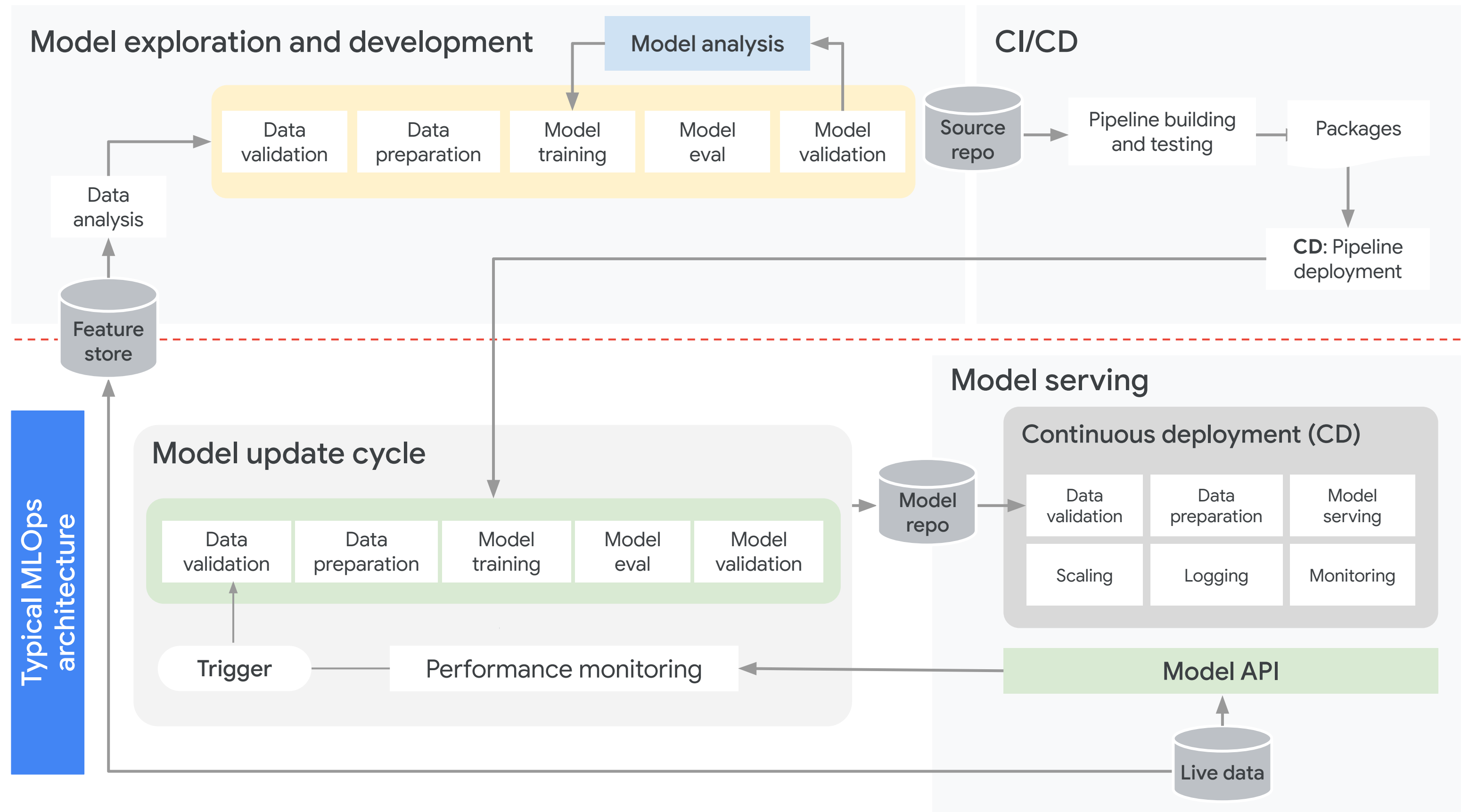
Concept Overview

Cloud Build Builders

Cloud Build Configuration

Cloud Build Triggers







Every container is a self-contained directory in repo


If any of these files is changed,
you need to rebuild and push
the Docker image.

[master](#) [mlops-on-gcp](#) / [immersion](#) / [kubeflow_pipelines](#) / [cicd](#) / [solutions](#) / [trainer_image_vertex](#) /


 **BenoitDherin** implement cloudbuild Vertex kfp pipeline compilation

..


 Dockerfile implement cloudbuild Vertex kfp pipeline compilation


 train.py implement cloudbuild Vertex kfp pipeline compilation

[master](#) [mlops-on-gcp](#) / [immersion](#) / [kubeflow_pipelines](#) / [cicd](#) / [solutions](#) / [kfp-cli_vertex](#) /

 **BenoitDherin** implement a cloudbuilder to run the pipeline

..

 Dockerfile implement a cloudbuilder to run the pipeline

 run_pipeline.py implement a cloudbuilder to run the pipeline

For CI/CD, use a GitHub trigger to rebuild ML artifacts

Connect the GitHub repository to your Google Cloud account, and then trigger a Cloud Build from a GitHub trigger. Do this for every container in your ML pipeline.

```
steps:
# Build the trainer image
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'gcr.io/$PROJECT_ID/trainer_image_covertypes:latest', '.']
  dir: trainer_image_vertex

# Compile the pipeline
- name: 'gcr.io/$PROJECT_ID/kfp-cli-vertex'
  args:
  - '-c'
  - |
    dsl-compile-v2 --py pipeline.py --output covertypes_kfp_pipeline.json
  env:
  - 'PIPELINE_ROOT=gs://$PROJECT_ID-vertex/pipeline'
  - 'PROJECT_ID=$PROJECT_ID'
  - 'REGION=$REGION'
  - 'SERVING_CONTAINER_IMAGE_URI=us-docker.pkg.dev/vertex-ai/prediction/sklearn-cpu.0-20:latest'
  - 'TRAINING_CONTAINER_IMAGE_URI=gcr.io/$PROJECT_ID/trainer_image_covertypes:latest'
  - 'TRAINING_FILE_PATH=gs://$PROJECT_ID-vertex/data/training/dataset.csv'
  - 'VALIDATION_FILE_PATH=gs://$PROJECT_ID-vertex/data/validation/dataset.csv'
  dir: pipeline_vertex
```

Google Cloud Platform mlops-course

Cloud Build

Dashboard

History

Triggers

Settings

Create trigger

Name *

Must be unique within the project

Description

Event

Repository event that invokes trigger

☒ Push to a branch

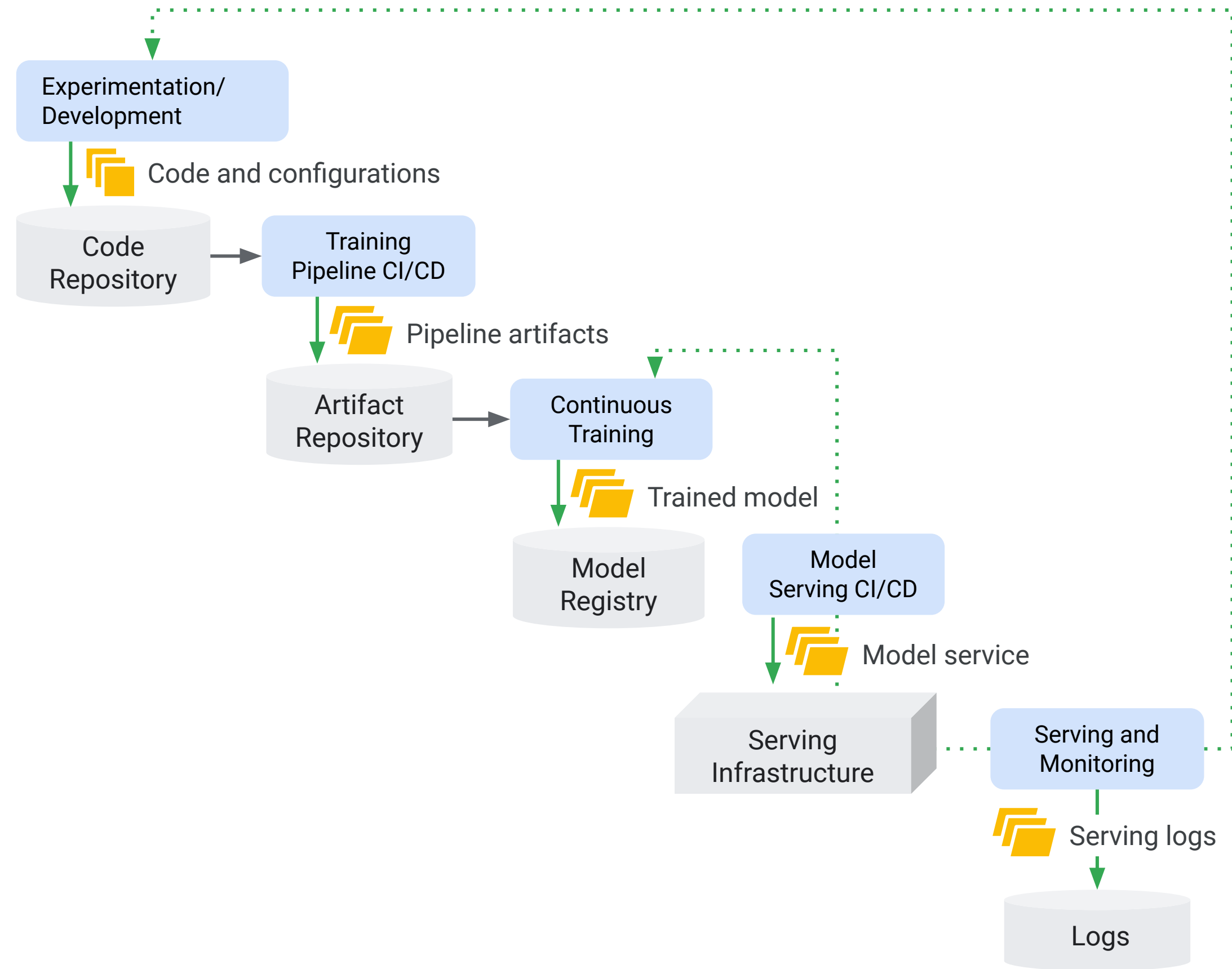
☐ Push new tag

☐ Pull request (GitHub App only)

Or when

☐ Manually run

Where we are
headed next



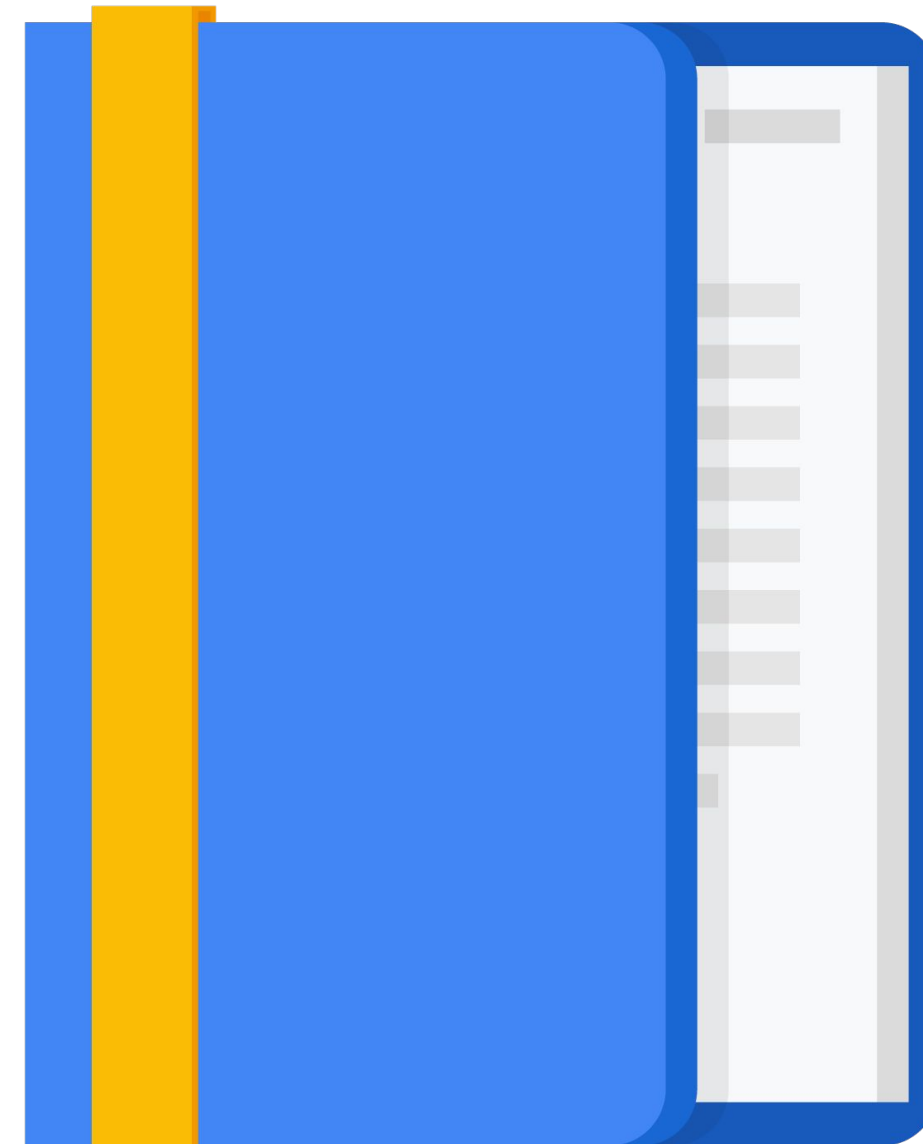
Agenda

Concept Overview

Cloud Build Builders

Cloud Build Configuration

Cloud Build Triggers



What are cloud builders?

Cloud configuration/provisioning actions that are packaged as Docker containers

Typical cloud builder actions:

- Building a Docker image from a Dockerfile
- Pushing a Docker image into a Google Cloud project registry
- Deploying a VM instance on Compute Engine
- Running a Kubeflow pipeline on Vertex Pipelines

What are cloud builders?

Standard

Already packaged config actions

Docker Registry:

`gcr.io/cloud-builders/`

Container Code:

[GoogleCloudPlatform/cloud-builders](https://cloud.google.com/cloud-build/docs/container-code)

Custom

config actions packaged by you

Docker Registry:

`gcr.io/<YOUR_PROJECT>/`




Standard Cloud Builders

Standard cloud builders

Builder	Name	Example
bazel	<code>gcr.io/cloud-builders/bazel</code>	bazel example
docker	<code>gcr.io/cloud-builders/docker</code>	docker example
git	<code>gcr.io/cloud-builders/git</code>	git example
gcloud	<code>gcr.io/cloud-builders/gcloud</code>	gcloud example
gke-deploy	<code>gcr.io/cloud-builders/gke-deploy</code>	gke-deploy example


...wrap standard config tools as Docker containers















- `bazel` : runs the [bazel](#) tool
- `curl` : runs the [curl](#) tool
- `docker` : runs the [docker](#) tool
- `dotnet` : run the [dotnet](#) tool
- `gcloud` : runs the [gcloud](#) tool
- `gcs-fetcher` : efficiently fetches objects from Google Cloud
- `git` : runs the [git](#) tool
- `gke-deploy` : deploys an application to a Kubernetes cluster,
- `go` : runs the [go](#) tool
- `gradle` : runs the [gradle](#) tool
- `gsutil` : runs the [gsutil](#) tool
- `javac` : runs the [javac](#) tool
- `kubectl` : runs the [kubectl](#) tool
- `mvn` : runs the [maven](#) tool
- `npm` : runs the [npm](#) tool
- `wget` : runs the [wget](#) tool
- `yarn` : runs the [yarn](#) tool


[GoogleCloudPlatform](#) / [cloud-builders](#)

[Code](#)
[Issues 14](#)
[Pull requests](#)
[Actions](#)
[Projects](#)
[Security](#)
[Insights](#)


Branch: master
Go to file
Add file
Clone


bendory committed 237d8e7 7 days ago
663 commits
1 branch
0 tags


	.github	Create issue template	17 months ago
	bazel	Active voice.	29 days ago
	curl	Fix argument forwarding in notice entrypoints. (#686)	28 days ago
	docker	Fix argument forwarding in notice entrypoints. (#686)	28 days ago
	dotnet	Fix argument forwarding in notice entrypoints. (#686)	28 days ago
	gcloud	Configure git in cloud-sdk to use ADC. (#694)	22 days ago
	gcs-fetcher	Use the Google-hosted cloud-sdk image.	last month
	git	Whitespace.	22 days ago
	gke-deploy	Add kubectl apply --server-dry-run flag to gke-deployer (#696)	15 days ago
	go	Fixed broken modules example. (#701)	7 days ago
	gradle	Fix argument forwarding in notice entrypoints. (#686)	28 days ago
	gsutil	Configure git in cloud-sdk to use ADC. (#694)	22 days ago
	javac	Active voice.	29 days ago
	kubectl	Configure git in cloud-sdk to use ADC. (#694)	22 days ago

<https://github.com/GoogleCloudPlatform/cloud-builders/tree/master/docker>


Example: Docker cloud builder


 [GoogleCloudPlatform](#) / [cloud-builders](#)


[Code](#) [Issues 14](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)


 **rafikk** committed f5f0767 28 days ago [...](#)


..

 [examples/helloworld](#)

 [Dockerfile-19.03.8](#)

 [README.md](#)

 [cloudbuild.yaml](#)

 [notice.sh](#)

GCB rebrand. ([#309](#))

Document use of alternative official supported images. ([#680](#))

Active voice.

Deprecated docker builder. ([#671](#))

Fix argument forwarding in notice entrypoints. ([#686](#))


```
FROM launcher.gcr.io/google/ubuntu16_04
```

```
ARG DOCKER_VERSION=5:19.03.8~3-0~ubuntu-xenial
```

```
RUN apt-get -y update && \  
    apt-get -y install \  
        apt-transport-https \  
        ca-certificates \  
        curl \  
        make \  
        software-properties-common && \  
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add - && \  
    apt-key fingerprint 0EBFCD88 && \  
    add-apt-repository \  
        "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  
        xenial \  
        edge" && \  
    apt-get -y update && \  
    apt-get -y install docker-ce=${DOCKER_VERSION} docker-ce-cli=${DOCKER_VERSION}
```

```
COPY notice.sh /usr/bin
```

```
ENTRYPOINT ["/usr/bin/notice.sh"]
```

When ran, /usr/bin/notice.sh is executed

Example: Docker cloud builder

notice.sh

```
#!/bin/sh
```

```
echo '
```

```
***** NOTICE *****
```

```
Alternative official `docker` images, including multiple versions across  
multiple platforms, are maintained by the Docker Team. For details, please  
visit https://hub.docker.com/\_/docker.
```

```
***** END OF NOTICE *****
```

```
'
```

```
/usr/bin/docker "$@"
```

When run, the container
passes its args to the
docker command.

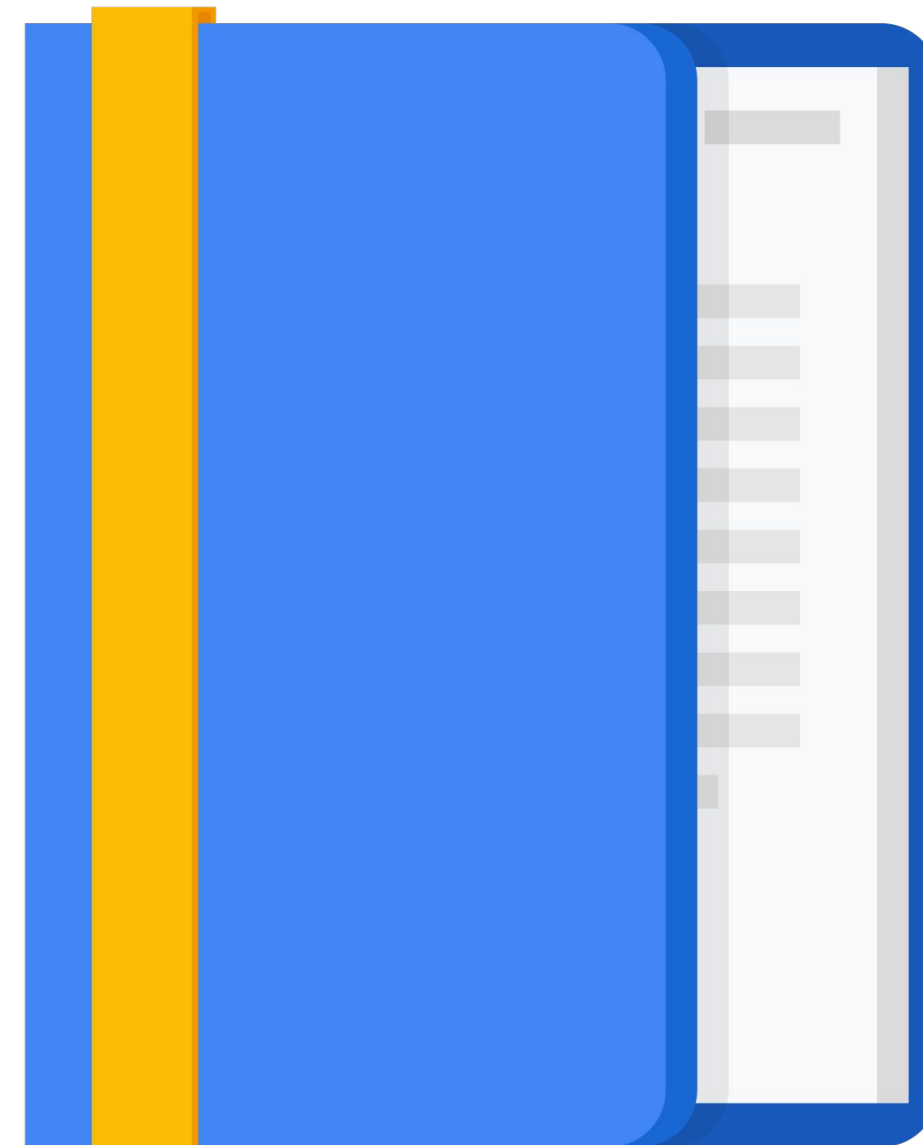
Agenda

Concept Overview

Cloud Build Builders

Cloud Build Configuration

Cloud Build Triggers



Cloud Build configuration file

cloudbuild.yaml

Describe the cloud builders to be run step by step

steps:

- name: 'gcr.io/cloud-builders/docker' ←
args: ['build', '-t', 'gcr.io/\$PROJECT_ID/trainer_image_covertypes_vertex:latest', '.']
dir: trainer_image_vertex

Running Cloud Build

```
gcloud builds submit . --config cloudbuild.yaml --substitutions $SUBSTITUTIONS
```

Command

Cloud Build config file
describing the cloud
builders to be run

Variable substitutions to be
made in the cloudbuild.yaml
config file

A simple Cloud Build step

steps:

- name: 'gcr.io/cloud-builders/docker'
args: ['build', '-t', 'gcr.io/\$PROJECT_ID/trainer_image_covertime_vertex:latest', '.']
dir: trainer_image_vertex

The cloud builder container URI to be run

The CWD in the Docker container from which the entrypoint is executed

The arguments to be passed to the container entrypoint

Persistence dir:

steps:

- **name:**
args:
dir: <rel path>

A cloud builder container has its current working directory:

- Set by default to /workspace in the container.
- Shared between steps.

This can be modified by specifying a **dir path** that:

- Will resolve to /workspace/<path>.
- Will still be shared between steps **if the path is relative.**

Caution: If the path is absolute, the CWD will not persist between steps.

Substitutions: \$_VARIABLE_NAME

steps:

```
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'gcr.io/$PROJECT_ID/$_TRAINER_IMAGE_NAME:$TAG_NAME', '.']
  dir: $_PIPELINE_FOLDER/$_IMAGE_NAME
```

```
gcloud builds submit . --config cloudbuild.yaml \
  --substitutions '_PIPELINE_FOLDER=.,_IMAGE_NAME=trainer_base'
```

Custom cloud builder

```
# Upload the pipeline
- name: 'gcr.io/$PROJECT_ID/kfp-cli-vertex'
  args:
    - '-c'
    - |
        dsl-compile-v2 --py pipeline.py --output covertime_kfp_pipeline.json
  env:
    - 'PIPELINE_ROOT=gs://$PROJECT_ID-vertex/pipeline'
    - 'REGION=$_REGION'
    - etc.

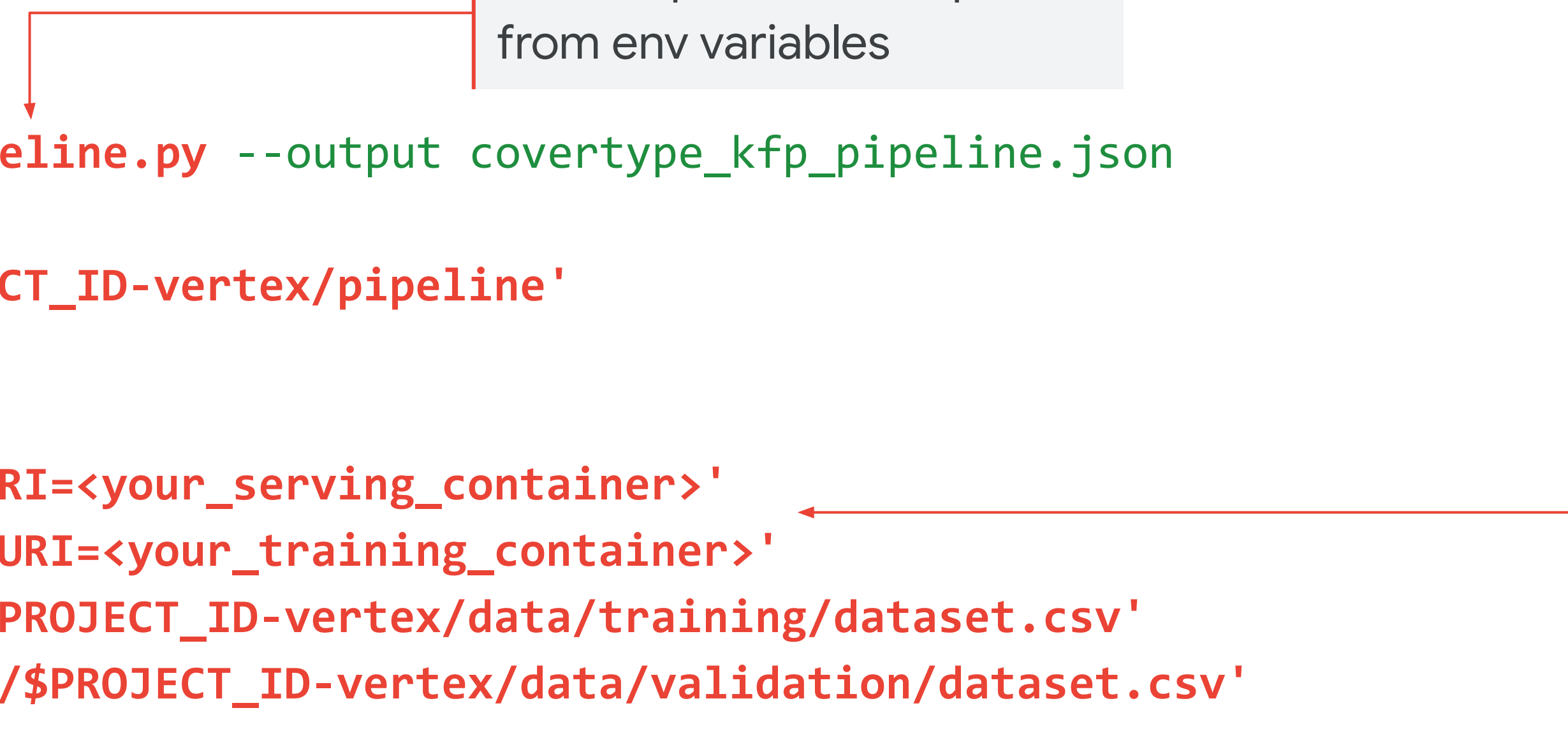
dir: pipeline_vertex
```

No different from the standard cloud builders (but the registry is your own project registry)

Passing environment variables

```
- name: 'gcr.io/$PROJECT_ID/kfp-cli-vertex'
args:
  - '-c'
  - 'dsl-compile-v2 --py pipeline.py --output coverted_kfp_pipeline.json'
env:
  - 'PIPELINE_ROOT=gs://$PROJECT_ID-vertex/pipeline'
  - 'PROJECT_ID=$PROJECT_ID'
  - 'REGION=$_REGION'
  - 'SERVING_CONTAINER_IMAGE_URI=<your_serving_container>'
  - 'TRAINING_CONTAINER_IMAGE_URI=<your_training_container>'
  - 'TRAINING_FILE_PATH=gs://$PROJECT_ID-vertex/data/training/dataset.csv'
  - 'VALIDATION_FILE_PATH=gs://$PROJECT_ID-vertex/data/validation/dataset.csv'

dir: pipeline_vertex
```



This script takes its input from env variables

That's how the env variables are passed to the script

Pushing images to Container Registry

Build the image locally on the build node

steps:

- name: 'gcr.io/cloud-builders/docker'
args: ['build', '-t', 'gcr.io/\$PROJECT_ID/trainer_image_covertypes_vertex:latest', '.']
dir: trainer_image_vertex

Images: ['gcr.io/\$PROJECT_ID/trainer_image_covertypes_vertex:latest']

Push the image to the registry

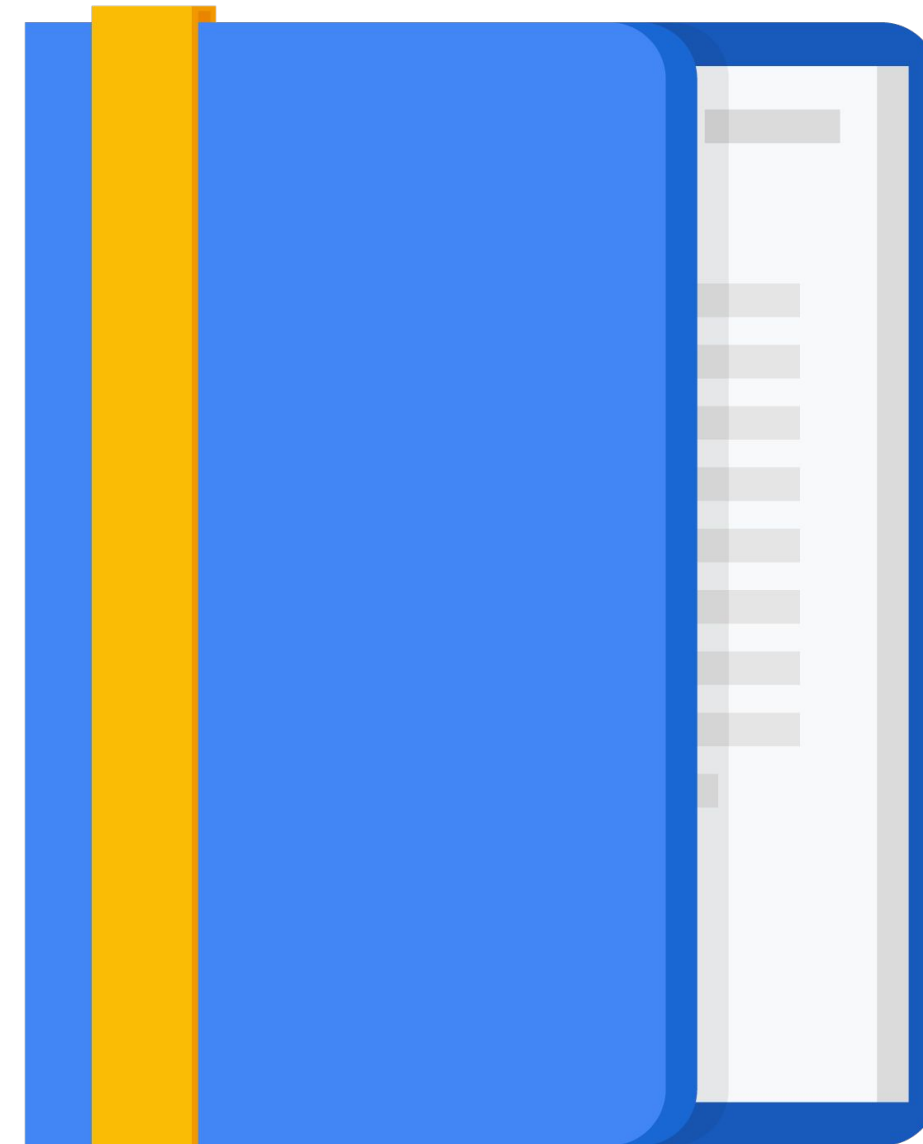
Agenda

Concept Overview

Cloud Build Builders

Cloud Build Configuration

Cloud Build Triggers



Manually executing a Cloud Build

1. Clone the Github repo with your cloudbuild.yaml and ML code on a build node.
 - The build node can be any node with the gcloud sdk properly authenticated:
 - JupyterLab VM
 - Cloud Shell
(if the containers to build are small)
 - Your laptop
 - A dedicated build VM on Cloud Compute
2. Run `gcloud builds submit` on the cloudbuild.yaml with the proper substitutions.

Manually executing a Cloud Build

```
[3]: SUBSTITUTIONS= f' _REGION={REGION} '  
SUBSTITUTIONS
```

```
[3]: '_REGION=us-central1'
```

```
[ ]: !gcloud builds submit . --config cloudbuild_vertex.yaml --substitutions {SUBSTITUTIONS}
```

CI/CD: Automated Cloud Build triggers

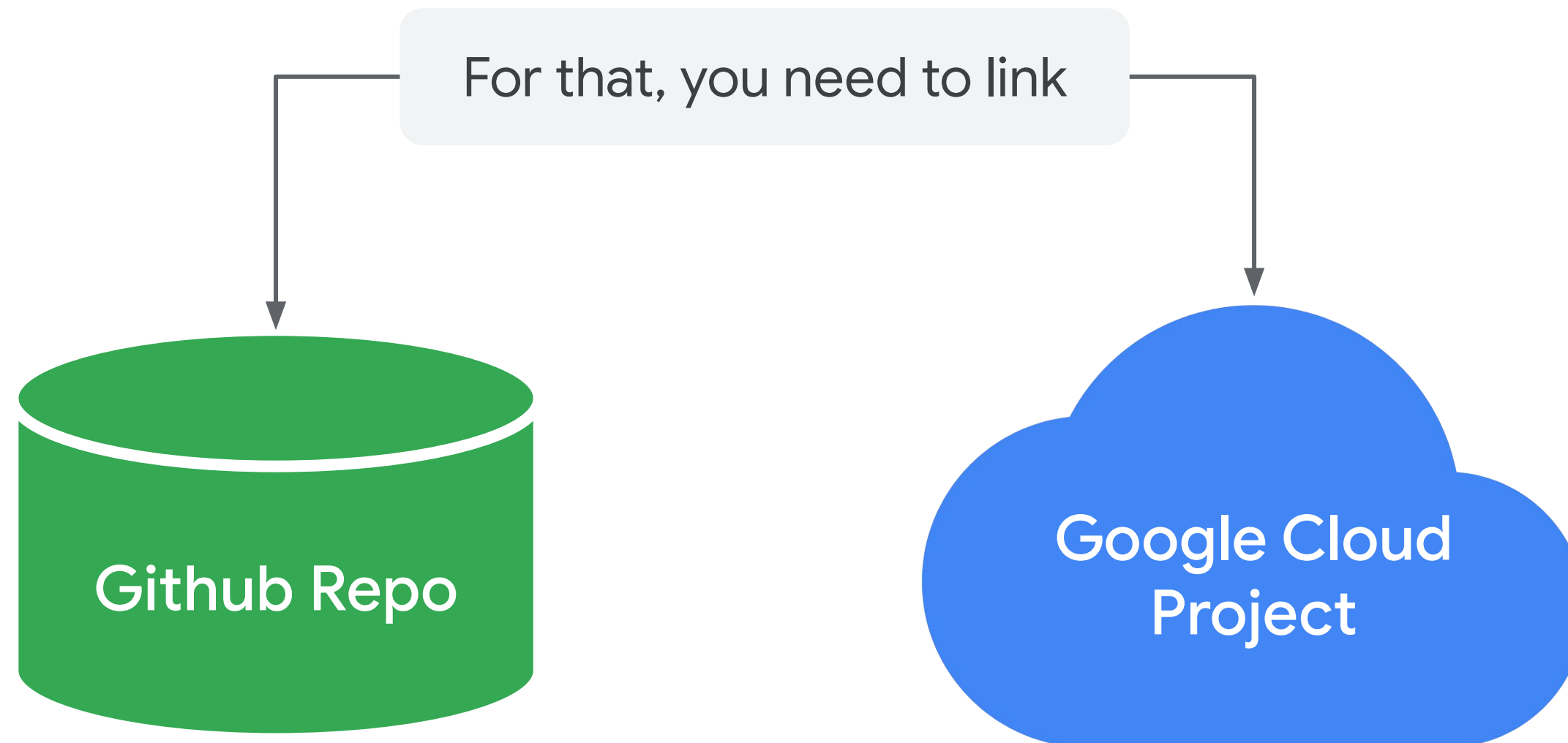
When the ML code changes in Github via:

- Push on a branch
- New tag
- Pull request
- etc.



...you want a Cloud Build to be triggered automatically, so that the training artifacts are updated from the new code:

- Training containers
- Kubeflow pipelines
- etc.


CI/CD: Automated Cloud Build triggers



Set up your github repo to work with Cloud Build




[Marketplace](#) / [Apps](#) / Google Cloud Build




Application

Google Cloud Build

 You've already granted this app access to GitHub outside of GitHub Marketplace.

[Set up a plan](#) [Configure access](#)

 **Verified by GitHub**
GitHub confirms that this app meets the [requirements for verification](#).

Categories

[Continuous integration](#)
[GitHub Enterprise](#)

Google Cloud Build

Google Cloud Build lets you create fast, consistent, reliable builds across all languages. Automatically build containers or non-container artifacts on commits to your GitHub repository. Get complete control over defining custom workflows for building, testing, and deploying across multiple environments such as VMs, serverless, Kubernetes, or Firebase.

Allow your repo to be accessed by Cloud Build

Repository access

☐ **All repositories**

This applies to all current *and* future repositories.

☒ **Only select repositories**


 Select repositories ▼


Select at least one repository.

Save

Cancel

Add your repo to Cloud Build

 Google Cloud Platform

 **Google Cloud Build**

Connect your selected GitHub repositories to your Google Cloud Platform projects

With this GitHub application, you can connect your repositories to your Google Cloud projects and start using Cloud Build.

1 Project settings

2 Repository selection

3 Trigger settings

Project

mlops-course ▼ or


Create new project


☒ I understand that GitHub content for the selected repositories will be transferred to this GCP project to provide the connected service. Members of 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](#)

Next

Cancel

Add your repo to Cloud Build

 Google Cloud Platform


 **Google Cloud Build**

Connect your selected GitHub repositories to your Google Cloud Platform projects

With this GitHub application, you can connect your repositories to your Google Cloud projects and start using Cloud Build.

☒ **Project settings** 2 **Repository selection** 3 Trigger settings

Select repositories to connect with your project



 Filter repositories


☒ **Select all repositories**


☒ BenoitDherin/mlops-on-gcp


Connect repository Cancel


Set up triggers


 Google Cloud Platform 

 Cloud Build

 Dashboard

 History


 Triggers

 Settings

Dashboard

Your dashboard shows the results of the latest builds for each build trigger. To populate your dashboard, you will need to set up build triggers.

SET UP BUILD TRIGGERS

 Filter triggers

Cloud triggers

Google Cloud Platform

mlops-course

Search

Cloud Build

Dashboard

History

Triggers

Settings

Create trigger

Name *

Must be unique within the project

Description

Event

Repository event that invokes trigger

☒ Push to a branch

☐ Push new tag

☐ Pull request (GitHub App only)

Or when

☐ Manually run

Source

Repository *

Select the repository with the build configuration file

Branch *

^master\$

Use a regular expression to match to a specific branch [Learn more](#)

Specify the location of the cloudbuild.yaml file

Field	Value
Name	[YOUR TRIGGER NAME]
Description	[YOUR TRIGGER DESCRIPTION]
Event	Tag
Source	[YOUR FORK]
Tag (regex)	.*
Build Configuration	Cloud Build configuration file (yaml or json)
Cloud Build configuration file location	./immersion/kubeflow_pipelines/cicd/solutions/cloudbuild_vertex.yaml

Set up the substitution variable values

Variable *

_REGION

Value

us-central1

+ ADD VARIABLE

A Cloud Build trigger is now listed

☰

Google Cloud Platform

mlops-course ▾

🔍 Search products and resources ▾

🗨

?

🔔

⋮

🏠 Cloud Build

📊 Dashboard

📜 History

➡ Triggers

⚙ Settings

Triggers

➡ CONNECT REPOSITORY

+ CREATE TRIGGER

Integrate your working repositories in order to start creating build triggers. Build triggers automatically build containers based on source code or tag changes in a repository.

Repositories

ACTIVE INACTIVE ?

☰ Filter repositories

🔗 BenoitDherin/mlops-on-gcp ↗

Cloud Build GitHub App ⋮

Name	Description	Event	Filter	Build configuration	Status
mlops-tag-trigger	—	Push to tag	.*	workshops/kfp-caip-sklearn/lab-03-kfp-cicd/cloudbuild.yaml	Enabled ▾ Run trigger

Pushing a new tag will now trigger a Cloud Build

```
git tag [TAG NAME]  
git push origin --tags
```

The build can be monitored

Google Cloud Platform

mlops-course

Search products and resources

Cloud Build

Dashboard

History


Triggers

Settings

Dashboard

Filter triggers

Running: BenoitDherin/mlops-on-gcp - mlops-tag-trigger

Latest Build	Duration	Trigger description	Source	Commit
6/25/20, 12:50 PM	00:00:14	-	 BenoitDherin/mlops-on-gcp	3cbdf6d

Build History

[View all](#)






Average Duration

-

Pass - Fail %

-

The build can be monitored

 Cloud Build	← Build details REBUILD COPY URL										
 Dashboard	<div><div>✓ Successful: 8c7488fb</div><div>Started on Aug 17, 2021, 5:08:48 PM</div></div>										
 History											
 Triggers											
 Settings											
	<table><tr><th>Steps</th><th>Duration</th></tr><tr><td>✓ Build Summary 3 Steps</td><td>00:24:53</td></tr><tr><td>✓ 0: gcr.io/cloud-builders/docker build -t gcr.io/qwiklabs-gcp-04-14242c0aa6a7/train...</td><td>00:02:05</td></tr><tr><td>✓ 1: gcr.io/qwiklabs-gcp-04-14242c0aa6a7/kfp-cli-ver... -c dsl-compile-v2 --py pipeline.py --output covertime...</td><td>00:00:09</td></tr><tr><td>✓ 2: gcr.io/qwiklabs-gcp-04-14242c0aa6a7/kfp-cli-ver... -c python kfp-cli_vertex/run_pipeline.py --project_id...</td><td>00:22:20</td></tr></table>	Steps	Duration	✓ Build Summary 3 Steps	00:24:53	✓ 0: gcr.io/cloud-builders/docker build -t gcr.io/qwiklabs-gcp-04-14242c0aa6a7/train...	00:02:05	✓ 1: gcr.io/qwiklabs-gcp-04-14242c0aa6a7/kfp-cli-ver... -c dsl-compile-v2 --py pipeline.py --output covertime...	00:00:09	✓ 2: gcr.io/qwiklabs-gcp-04-14242c0aa6a7/kfp-cli-ver... -c python kfp-cli_vertex/run_pipeline.py --project_id...	00:22:20
Steps	Duration										
✓ Build Summary 3 Steps	00:24:53										
✓ 0: gcr.io/cloud-builders/docker build -t gcr.io/qwiklabs-gcp-04-14242c0aa6a7/train...	00:02:05										
✓ 1: gcr.io/qwiklabs-gcp-04-14242c0aa6a7/kfp-cli-ver... -c dsl-compile-v2 --py pipeline.py --output covertime...	00:00:09										
✓ 2: gcr.io/qwiklabs-gcp-04-14242c0aa6a7/kfp-cli-ver... -c python kfp-cli_vertex/run_pipeline.py --project_id...	00:22:20										

After the build, new artifacts are available

<div>Container Registry</div> <div>Images</div> <div>Settings</div>	Repositories		
	qwiklabs-gcp-04-14242c0aa6a7		
	Filter Enter property name or value		
	Name ↑	Hostname ?	Visibility ?
	gcf	us.gcr.io	Private
	kfp-cli-vertex	gcr.io	Private
	taxifare_training_container	gcr.io	Private
	trainer_image	gcr.io	Private
	trainer_image_covertime_vertex	gcr.io	Private

Lab

CI/CD for a KFP Pipeline

In this lab you will walk through authoring a Cloud Build CI/CD workflow that automatically builds and deploys a KFP pipeline.

notebooks/[kubeflow_pipelines/cicd/labs/kfp_cicd_vertex.ipynb](#)

cloud.google.com