

BEAM  
SUMMIT

# CI CD for Dataflow with Flex Template




Mazlum TOSUN



Head of Data and  
Cloud GroupBees



- ❖ Google Cloud Evangelist, Data Architect, functional programming, Devops, Serverless...
- ❖  Fan

<https://www.youtube.com/@GCPLearning-ce9bq> 

<https://github.com/tosun-si> 

<https://twitter.com/MazlumTosun3> 

<https://www.linkedin.com/in/mazlum-tosun-900b1812/> 

<https://medium.com/@mazlum.tosun> 

<https://stackoverflow.com/users/9261558/mazlum-tosun> 



## Flex Templates

- ❑ Standardisation of Dataflow Template deployment based on a Docker image
- ❑ Standardisation whatever the language and the sdk
- ❑ All the dependencies can be pre installed in the container
- ❑ Template spec from GCS

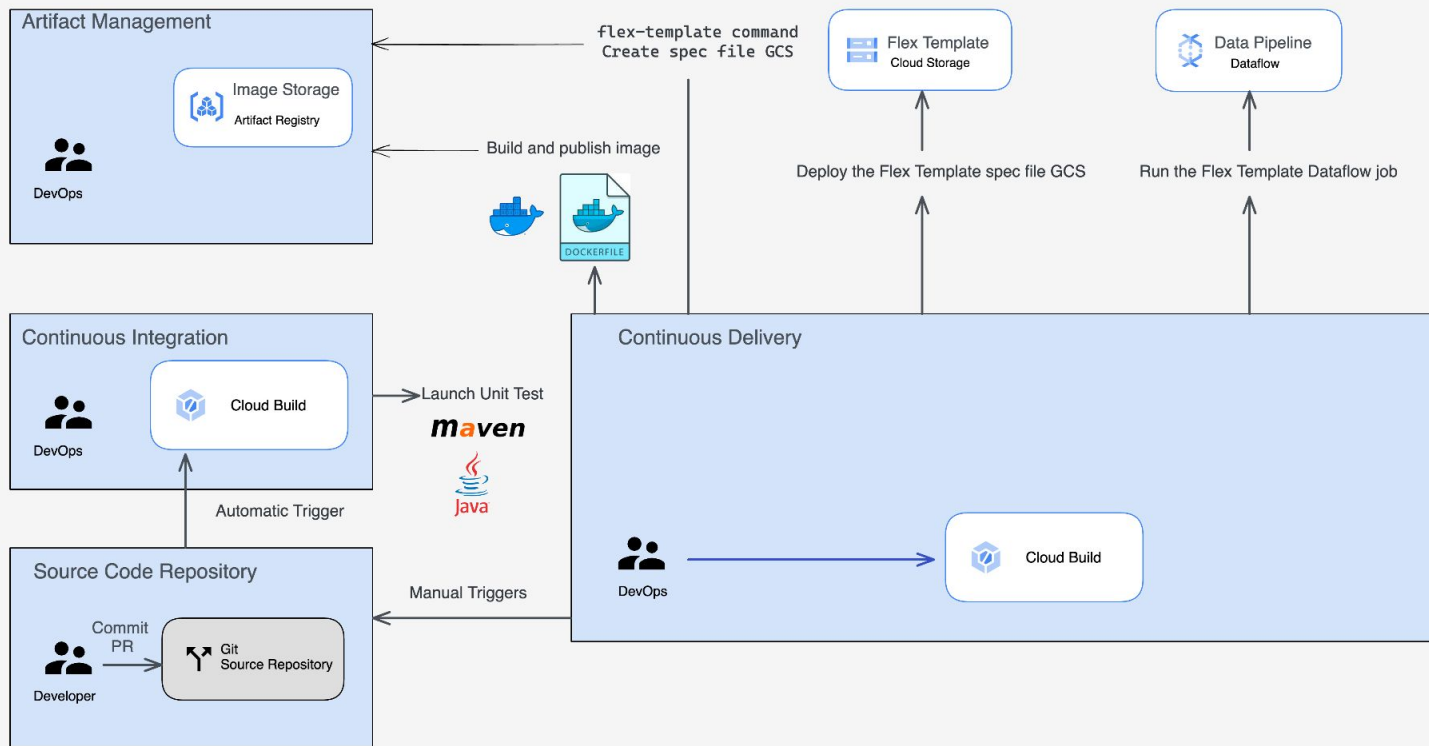


## Advantages of Flex Templates over classic template

- ❑ Docker image offers more flexibility
- ❑ Classic templates requires the ValueProvider interface for input parameters
- ❑ Flex templates don't require the ValueProvider
- ❑ Classic templates have a static job graph
- ❑ Flex templates can dynamically construct the job graph
- ❑ Example, select a different I/O connector based on input parameters

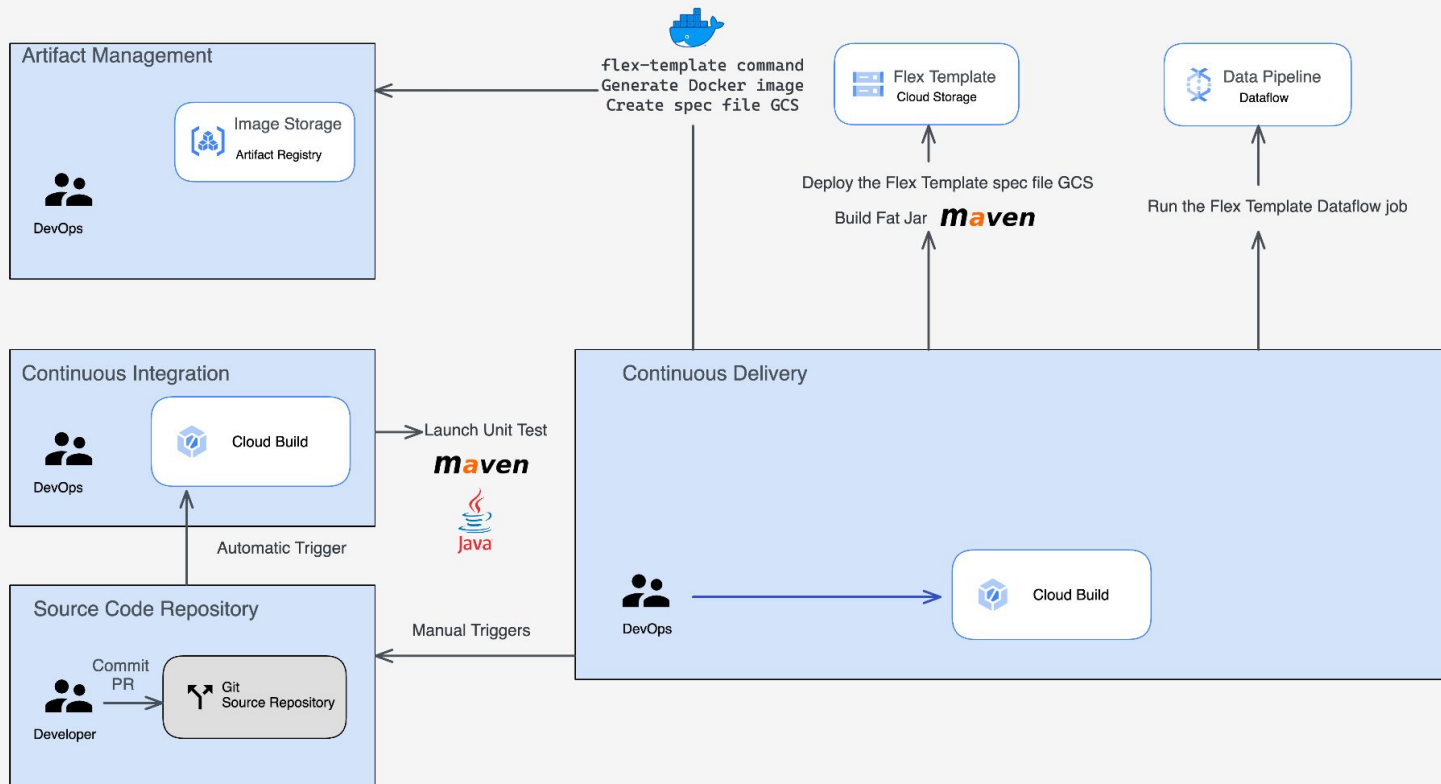


## Dataflow CI CD with Flex Template and Cloud Build (with Dockerfile)





## Dataflow CI CD with Flex Template and Cloud Build (without Dockerfile)





## Cloud Build



- ❑ Run unit tests with Maven
- ❑ Build image with a Dockerfile and all the dependencies in the container
- ❑ Create spec file in Cloud Storage





## Cloud Build



- ❑ Run unit tests with Maven
- ❑ Build the fat jar
- ❑ Generate the image and spec file in GCS with flex-template command



## Cloud Build



- ❑ Run unit tests with PyTest
- ❑ Build image with a Dockerfile and all the dependencies
- ❑ Create spec file in Cloud Storage



## Gitlab CI



- ❑ Run unit tests with Maven
- ❑ Build image with a Dockerfile and all the dependencies using Kaniko
- ❑ Create spec file in a Cloud Storage



## Gitlab CI



- ❑ Run unit tests with PyTest
- ❑ Build image with a Dockerfile and all the dependencies using Kaniko
- ❑ Create spec in Cloud Storage bucket



## Dagger



- ❑ Pipeline As Code with Go SDK
- ❑ Build image with a Dockerfile and all the dependencies using Kaniko
- ❑ Create spec in Cloud Storage bucket



## Cloud Build



- ❑ Serverless
- ❑ No need a token key for authentication
- ❑ Not an interactive CI and no orchestration for the pipeline
- ❑ Natively can't share YAML templates



## Gitlab CI



CI⚡CD

- ❑ Graphic and interactive CI CD pipeline
- ❑ Manual and automatic jobs
- ❑ Dependencies between jobs / can share YAML templates
- ❑ Can require a token key if installed outside of Google Cloud
- ❑ If installed on GCP, need having a VM or a GKE cluster
- ❑ Can use Workload Identity Federation if installed on GKE



## Dagger



- ❑ Pipeline As Code
- ❑ Different SDKs : Go, Node, Python
- ❑ More easy to add some code logics than YAML (example : error handling)
- ❑ Depends only on Docker
- ❑ Need to handle the authentication with ADC or a token key
- ❑ No orchestration for the pipeline (manual and automatic steps)





<https://github.com/tosun-si/dataflow-java-ci-cd>

<https://github.com/tosun-si/dataflow-python-ci-cd>

<https://medium.com/google-cloud/ci-cd-for-dataflow-java-with-flex-templates-and-cloud-build-e3c584b8e564>



Thank you :)

QUESTIONS?