Building Pipelines for Workflow Orchestration Using Google Composer

INTRODUCING GOOGLE COMPOSER

Vitthal Srinivasan
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Workflow orchestration service on GCP

Define pipeline as DAG

Simple Python code

Great visualization support

Runs on Kubernetes

Host of useful operators

Prerequisites and Course Outline

Prerequisites: Basic Cloud Computing

Architecting Scalable Web Applications Using Google App Engine

Architecting Event-driven Serverless Solutions Using Google Cloud Functions

Course Outline

Introducing Google Composer

Creating, configuring, and accessing environments

Managing and monitoring workflows

Scenarios: SpikeySales.com

Hypothetical online retailer

- Flash sales of trending products
- Spikes in user traffic

SpikeySales on the GCP

- Cloud computing fits perfectly
- Pay-as-you-go
- No idle capacity during off-sale periods

Important Composer Concepts

Composer is a pipeline orchestration technology similar to Oozie or Azkaban

Using Composer

Write code for pipeline

Copy into GCS bucket

Airflow picks up and schedules

Pipeline parallelized and executed

Important Composer Concepts

Airflow	Composer	Environment
DAG	Operator	Trigger

Airflow

Apache Airflow (incubating)

Create workflows as DAGs

Airflow schedules and executes

Ensures dependencies satisfied

Airflow

Simple Python API

Scalable architecture

Powerful operators

Jinja templates

Airflow

Airflow workers

Web server to managed pipelines

Scheduler tracks and executes DAGs

Celery task queue to scale workers

Redis as message broker for Celery

Composer

GCP managed service for Airflow

Airflow workers on GKE cluster

Airflow metadata on Cloud SQL

Airflow web server on App Engine Flex

GCS bucket for DAGs

Environment

Self-contained Airflow deployment

Google-managed tenant project

Hosted entirely within a region

DAG

Directed-Acyclic-Graph

Defined in Airflow Python script

- Must execute instantaneously
- Merely DAG definition file

Contains operators and dependencies

Different tasks run on different workers

DAG

DAG defined in global namespace

Uses context manager

Contains dependencies

Copied to environment GCS bucket

Scheduled and executed periodically

Operator

Corresponds to one step in pipeline

Each instantiation of an operator is one task instance

Each task instance exists inside pipeline

Used with Python context manager

Operator

BashOperator

PythonOperator

BranchPythonOperator

SendGrid

BigQuery

Dataproc

Cloud Storage buckets

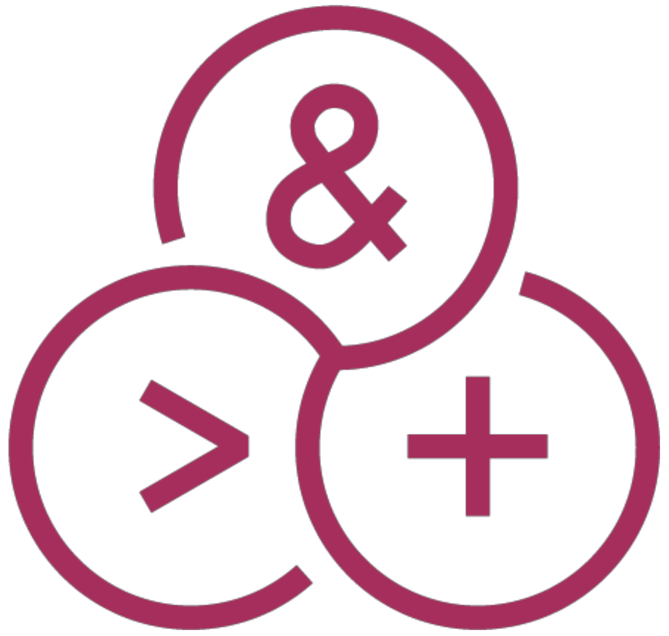
BashOperator

Cloud Composer runs the provided commands in a Bash script on a worker

Worker is a Debian-based Docker container and includes several packages

- gcloud
- bq
- gsutil
- kubectl

PythonOperator



Cloud Composer runs the Python code in a container that includes several packages

- google-cloud-bigquery
- google-cloud-dataflow
- google-cloud-storage
- pandas
- Tensorflow

GCP Operators

Cloud Composer automatically configures an Airflow connection to the environment's project

- BigQuery operators query and process data in BigQuery
- Cloud Dataflow operators run Apache Beam jobs in Cloud Dataflow

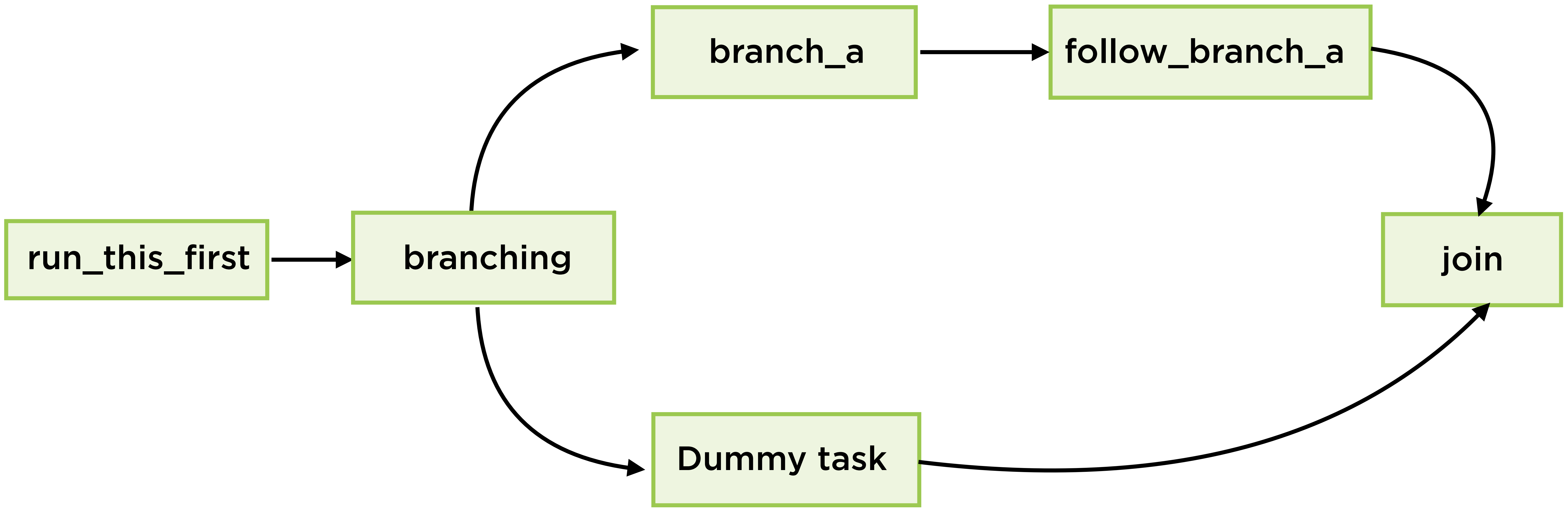
EmailOperator

Use the EmailOperator to send email from a DAG. To send email from a Cloud Composer environment, you must configure your environment to use SendGrid.

Branching

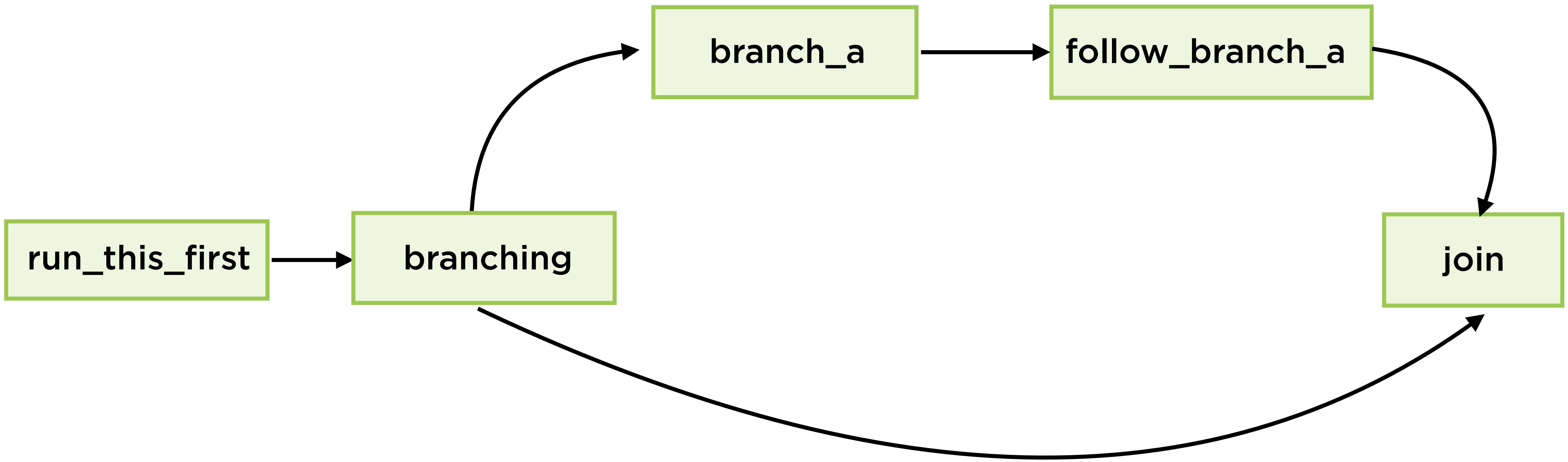
BranchPythonOperator expects a python_callable that returns a task_id; The task_id returned is followed, and all of the other paths are skipped.

Branching Needs Dummy Operators



Never leave an empty path after a branch - use a dummy task instead

Branching Done Wrong



Here the join task is skipped

Trigger

Airflow scheduler monitors DAGs

Triggers task instances whose dependencies have been met

Runs each DAG one schedule_interval after start date

Trigger

Several trigger rules

- all_success
- all_failed
- one_failed
- one_success
- dummy

Composer is a pipeline orchestration technology similar to Oozie or Azkaban

Dataflow vs. Composer

Dataflow	Composer
Data processing pipelines	General purpose pipelines
Focus on windowing and streaming	Focus on scripting and Python
Complex to implement	Simple to implement
Cumbersome to trigger	Trivial to trigger
Visualization UI exists but not central	Fundamentally UI-based
Few specialized operators	Many helpful specialized operators

Dataflow vs. Composer

Dataflow	Composer
Serverless, no clusters provisioned	Runs on Kubernetes cluster
No access to compute nodes	Compute nodes in GKE cluster
Apache Beam API	Apache Airflow API

Using Dataflow

Write code for pipeline

Submit job for execution

Dataflow assigns workers to execute

Pipeline parallelized and executed

Using Composer

Write code for pipeline

Copy into GCS bucket

Airflow picks up and schedules

Pipeline parallelized and executed

Demo

Enable APIs and create Composer environment

Demo

Write and run DAG on Composer environment

Pricing

Google Composer Pricing

Based on the size of Cloud Composer environment

Per-minute billing

In addition to charges for

- Google Kubernetes Engine
- Cloud Storage

Indicative Pricing

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

Item	Price (USD)
Web core hours	$0.074/vCPU hour
Database core hours	$0.125/vCPU hour
Web and database storage	$0.273 per GB/month
Network egress	$0.156/GB

Environment Sizing

Settings	Default	Adjustable
Storage (GB)	20	No
Database vCPUs	2	No
Web server vCPUs	2	No
Worker machine type	n1-standard-1	Yes
Worker nodes	3	Yes
Worker storage (GB per worker)	100	Yes

Google Composer Pricing Example

Create a Cloud Composer environment

Assume

- Default specs
- Environment used for 25% of time
- Equivalent to 182 hours/month
- 6.5 GB of egress

Google Composer Pricing Example

Resource	Total cost
Database core hours	$45.66
Web core hours	$26.91
Web and database storage	$1.37
Network egress	$1.44
Total Cloud Composer cost	$76.06

Summary

Workflow orchestration service on GCP

Simple Python code

Easy to visualize

Runs on Kubernetes

Host of useful operators