

# MASHR.



A data pipeline framework for GCP

# Overview

- What is Mashr
- What are data pipelines?
- Challenges of creating your own data pipelines
- Alternate Solutions for creating a data pipeline
- Discuss how Mashr solves the problem
- Design and Challenges of Mashr



# What is Mashr?

Mashr is an easy-to-use data pipeline orchestration and monitoring framework for small applications which allows you to take data from multiple sources and combine it so that you can use it.



# What is mashr

- CLI
- Node js
- Npm
- GCP
- Docker
- Embulk

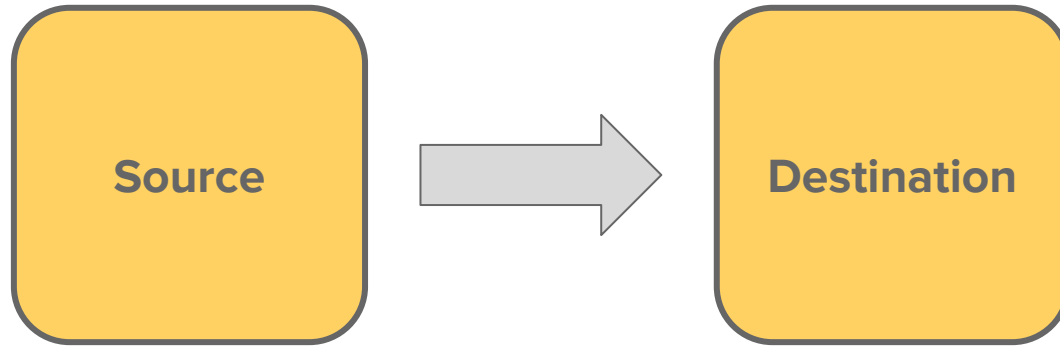


# Mashr Supports Google Cloud Platform

- GCP comes with a number of robust machine learning and analytics tools

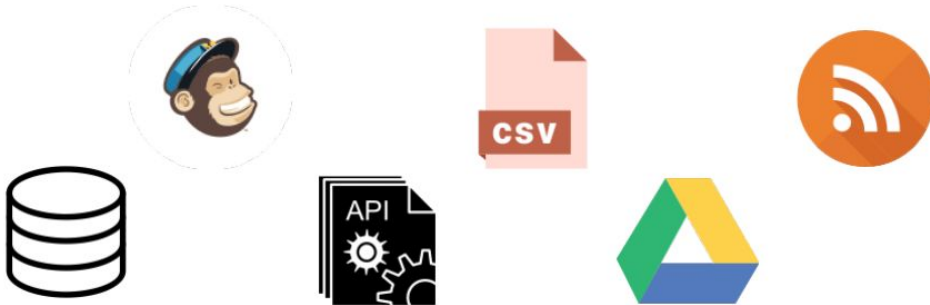


# Data pipelines

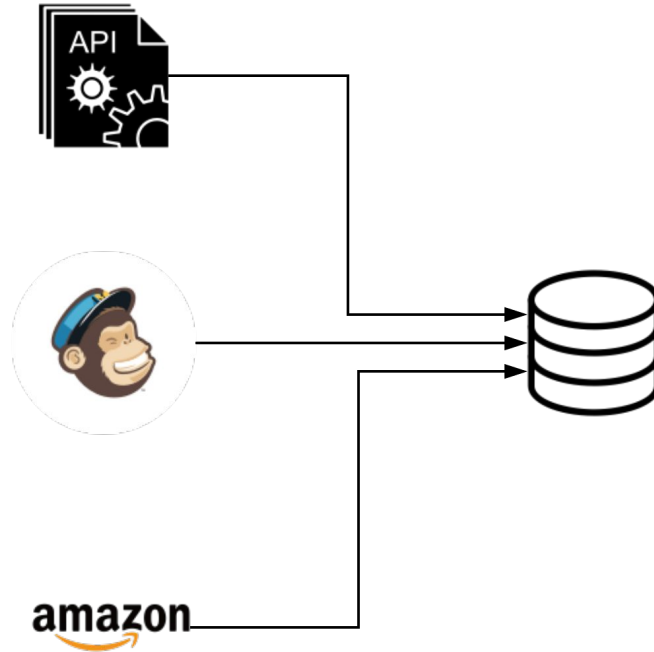


# Use Case

You are a developer for a small application and you have data spread out in a variety of sources - a psql database, salesforce, other applications with REST api's, etc. You want to take data from multiple sources and combine it so that you can use it.

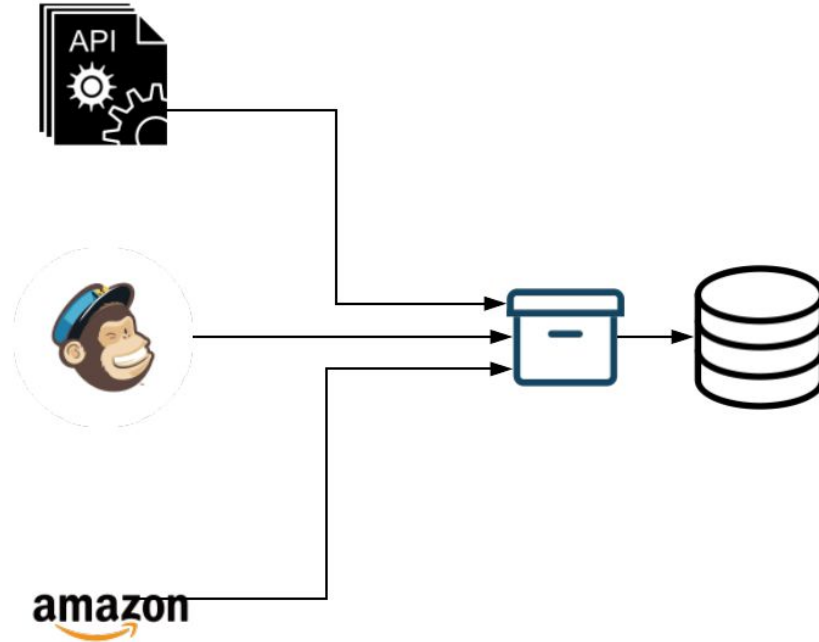


# Use Case

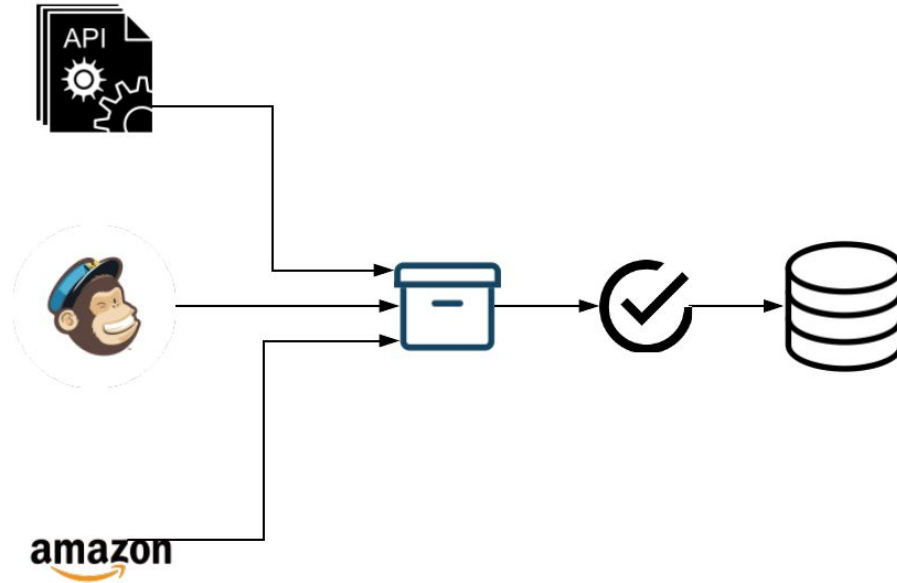




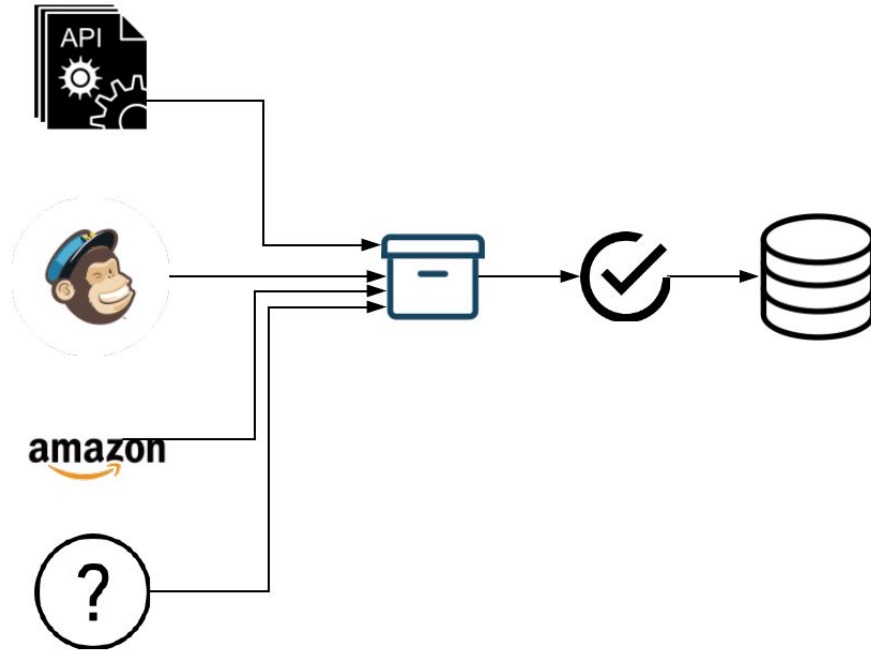
# Use Case



# Use Case



# Use Case



# **Principles of a good Data Pipeline**



# Principles of a good Data Pipeline

Extracting data from the source

- Extraction
- Formatting
- Scheduling



# Principles of a good Data Pipeline

## Validate

- Confirm whether data pulled from sources has the expected values
- Checking data integrity



# Principles of a good Data Pipeline

## Transform Data

- Removing extraneous or erroneous data (cleaning),
- Encoding free-form values
- Translating coded values
- Deriving a new calculated value
- Splitting a column into multiple columns



# Principles of a good Data Pipeline

## Stage / Archiving

- Audit reports & compliance
- Diagnose and debugging
- Failover





# Principles of a good Data Pipeline

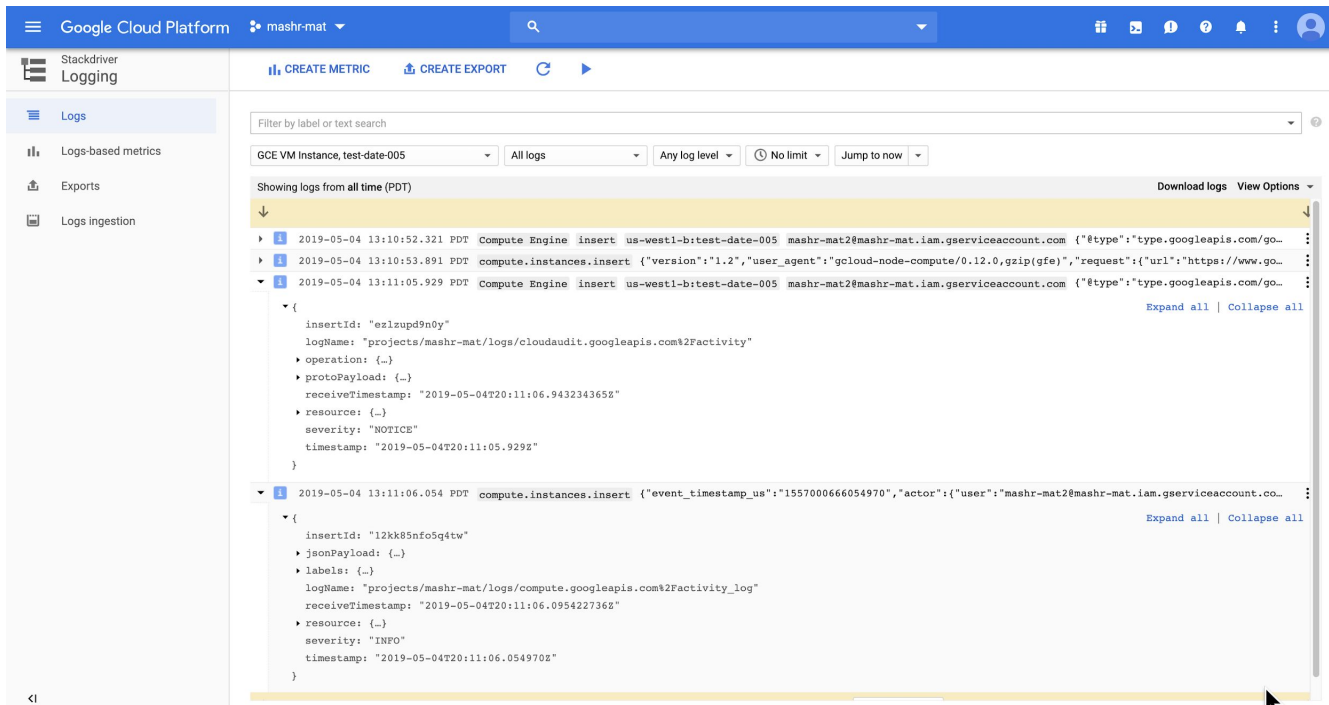
Publishing data to the target

- Decide method of loading data



# Principles of a good Data Pipeline

## Monitoring



The screenshot displays the Google Cloud Platform (GCP) Logging console. The interface includes a top navigation bar with the GCP logo, project name 'mashr-mat', and a search bar. On the left, a sidebar shows navigation options: Stackdriver Logging, Logs, Logs-based metrics, Exports, and Logs ingestion. The main content area is titled 'GCE VM Instance, test-date-005' and shows a list of log entries. The first entry is expanded, revealing a JSON payload with details about a 'compute.instances.insert' operation. The log entry includes a timestamp of '2019-05-04 13:11:05.929 PDT', a severity of 'NOTICE', and a log name of 'projects/mashr-mat/logs/cloudaudit.googleapis.com%2Factivity'.

Google Cloud Platform mashr-mat

Stackdriver Logging

Logs

Logs-based metrics

Exports

Logs ingestion

Filter by label or text search

GCE VM Instance, test-date-005 All logs Any log level No limit Jump to now

Showing logs from all time (PDT)

Download logs View Options

2019-05-04 13:10:52.321 PDT Compute Engine insert us-west1-b:test-date-005 mashr-mat@mashr-mat.iam.gserviceaccount.com {"@type":"type.googleapis.com/go-...

2019-05-04 13:10:53.891 PDT compute.instances.insert {"version":"1.2","user\_agent":"gcloud-node-compute/0.12.0,gzip(gfe)","request":{"url":"https://www.go-...

2019-05-04 13:11:05.929 PDT Compute Engine insert us-west1-b:test-date-005 mashr-mat@mashr-mat.iam.gserviceaccount.com {"@type":"type.googleapis.com/go-...

{

insertId: "ex1zupd9n0y"

logName: "projects/mashr-mat/logs/cloudaudit.googleapis.com%2Factivity"

operation: {...}

protoPayload: {...}

receiveTimestamp: "2019-05-04T20:11:06.943234365Z"

resource: {...}

severity: "NOTICE"

timestamp: "2019-05-04T20:11:05.929Z"

Expand all Collapse all

2019-05-04 13:11:06.054 PDT compute.instances.insert {"event\_timestamp\_us":"155700666054970","actor":{"user":"mashr-mat@mashr-mat.iam.gserviceaccount.co-...

{

insertId: "12kk85mfo5q4tw"

jsonPayload: {...}

labels: {...}

logName: "projects/mashr-mat/logs/compute.googleapis.com%2Factivity\_log"

receiveTimestamp: "2019-05-04T20:11:06.095422736Z"

resource: {...}

severity: "INFO"

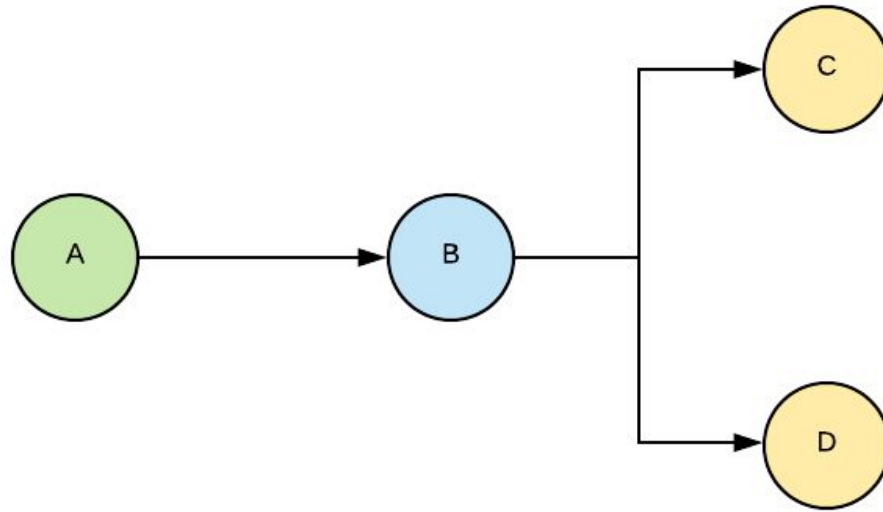
timestamp: "2019-05-04T20:11:06.054970Z"

Expand all Collapse all



# Principles of a good Data Pipeline

Orchestration and ordering of tasks



# Principles of a good Data Pipeline

## Performance

- Optimizations
- Bottlenecks



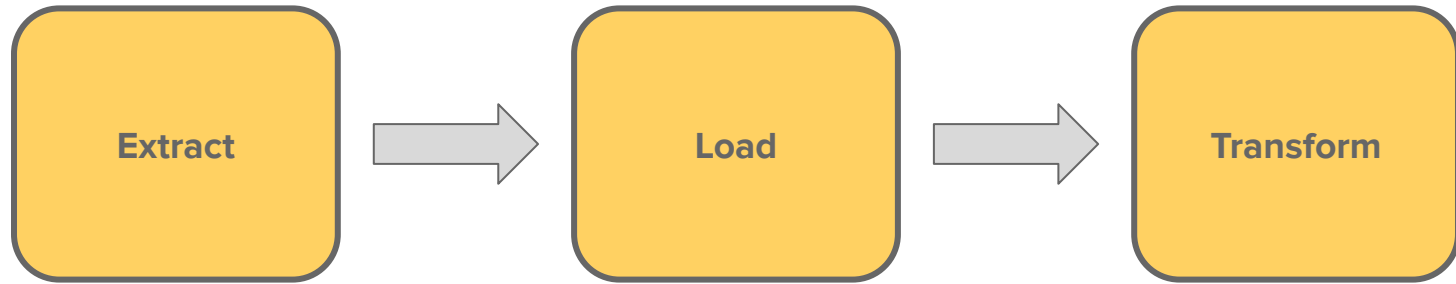
# ETL

Extract, Transform, Load

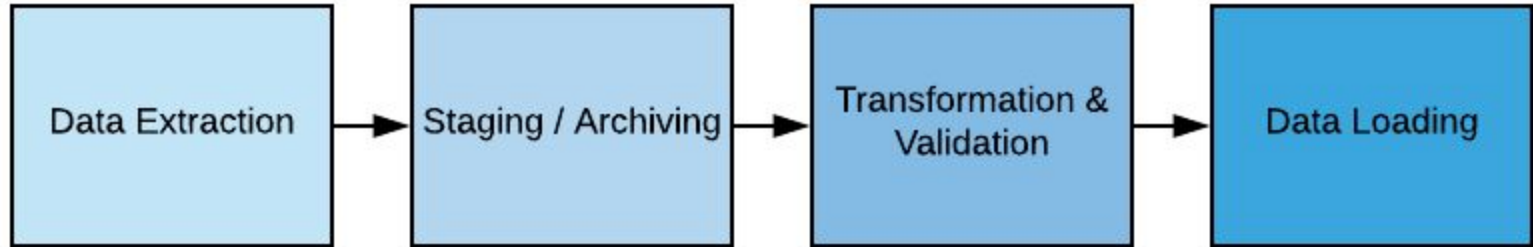


# ELT

Extract, Load, Transform



# Components of a data pipeline



# Principles of a good Data Pipeline

- Extraction
- Validation
- Transformation
- Staging / Archiving
- Publishing to target
- Monitoring
- Orchestration and Ordering
- Performance





# Mashr Supports Google Cloud Platform

- GCP comes with a number of robust machine learning and analytics tools



# Challenges of deploying your own data pipeline on GCP

- Overwhelming number of configuration options
- Client libraries may not be complete
- Many “tasks” are actually comprised of a series of smaller tasks and decision points



# Challenges of deploying your own data pipeline on GCP

Metric	Count
Number of API calls to GCP	23
Number of GCP services used	7



# Hosted Solutions



Informatica™



# Hosted Solutions

**Integrations** Destination Notifications

Set Up Account Add an Integration Select a Destination

### Start by adding an integration

Integrations are apps and databases that you can use to send data through your pipeline. Select your first integration below to get started. If you don't have this information, you can [skip this step](#) for now or [invite someone from your team](#) to help.

#### Add an Integration

Search Directory










Every integration you add comes with **seven days** of free data replication. [Learn more.](#)

**Filter by Type**

- All
- Beta
- Coming Soon
- Enterprise
- Free
- Paid

**Filter by Support**

- All
- Certified by Stitch
- Community Supported

 <b>AdRoll</b> Advertising Free	 <b>AdWords</b> Advertising Free	 <b>AfterShip</b> Business Free
 <b>Amazon Aurora</b> Databases Free	 <b>AppsFlyer</b> Analytics Free	 <b>Autopilot</b> Business Free
 <b>Bing Ads</b> Advertising Free	 <b>Braintree</b> Business Free	 <b>Branch</b> Business Free



# Hosted Solutions

## Pros:

- Infrastructure is abstracted from user
- Manages failover and backup
- UI
- Time

## Cons:

- Costs money
- Privacy
- Control and customization

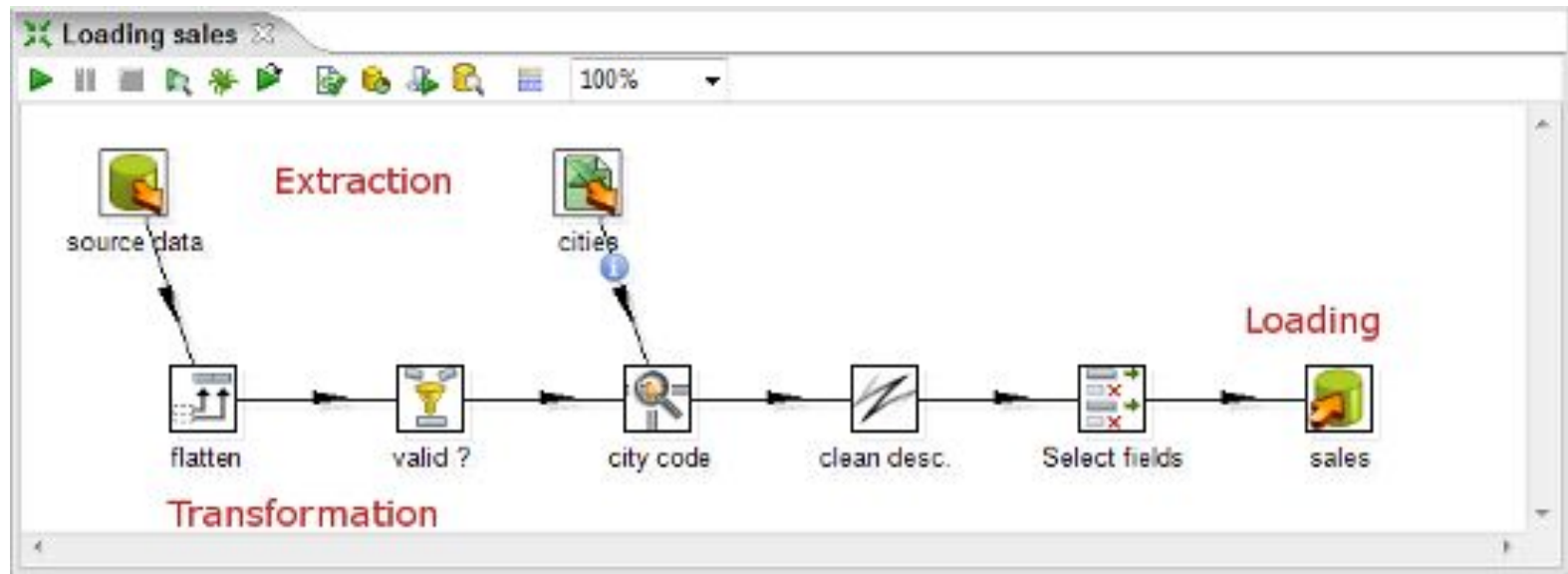


# **Self-Hosted Solutions**



# Self-Hosted Solutions

Pentaho Data Integration (PDI):





# Self-Hosted Solutions

## Pros:

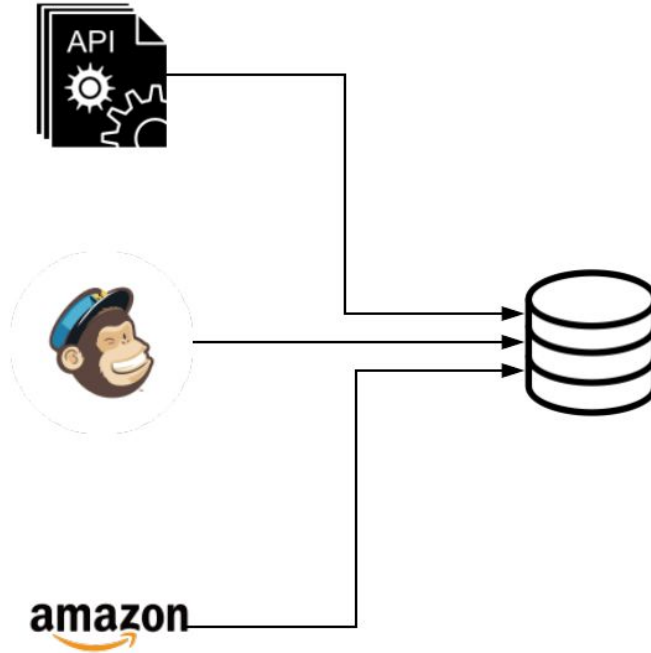
- Customizable
- Cheap
- Data is yours

## Cons:

- Not free
- Time
- Failover and backup



# Use Case



# Mashr





Mashr is an easy to use data pipeline framework that orchestrates moving data from external sources like salesforce, psql databases or REST apis into a single destination database where it can be used.



# Mashr



# Mashr

## Pros:

- Data is yours
- Customizable
- Abstracted...mostly
- Manages archiving and failover






## Cons:

- Still need to manage servers
- Not real-time
- Requires GCP account



# Mashr

## GCP Services

Icon	Name	Description
	GCE Instance	Virtual Machine Instance
	GCS	Data Storage Service
	Cloud Function	Function-as-a-Service
	BigQuery	Data Warehouse
	Stackdriver Logging	Logging Service

# Mashr

## Embulk

- We chose Embulk as the data extractor piece for the data pipeline
- Embulk is an open source plugin-based bulk data loader.

```
in:
  type: http
  url: https://jsonplaceholder.typicode.com/posts/42
  method: get
  parser:
    type: json
    schema:
      - { name: "userId", type: string }
      - { name: "id", type: long }
      - { name: "title", type: string }
      - { name: "body", type: string }
out:
  type: 'gcs'
  bucket: 'mashrintname'
  path_prefix: {{ env.DATE }}
  file_ext: '.json'
  auth_method: 'compute_engine'
  formatter:
    type: 'jsonl'
```





# Mashr

## Embulk

- embulk run config.yml
- embulk run config.yml -c diff.yml

```
in:
  type: postgresql
  host: 55.55.55.555
  user: postgres
  password: "password"
  database: postgres
  table: users
  incremental: true
  incremental_columns: [created_at, id]
```



# Mashr

## Embulk

### Pros:

- Modular / plugin-based
- Well-maintained

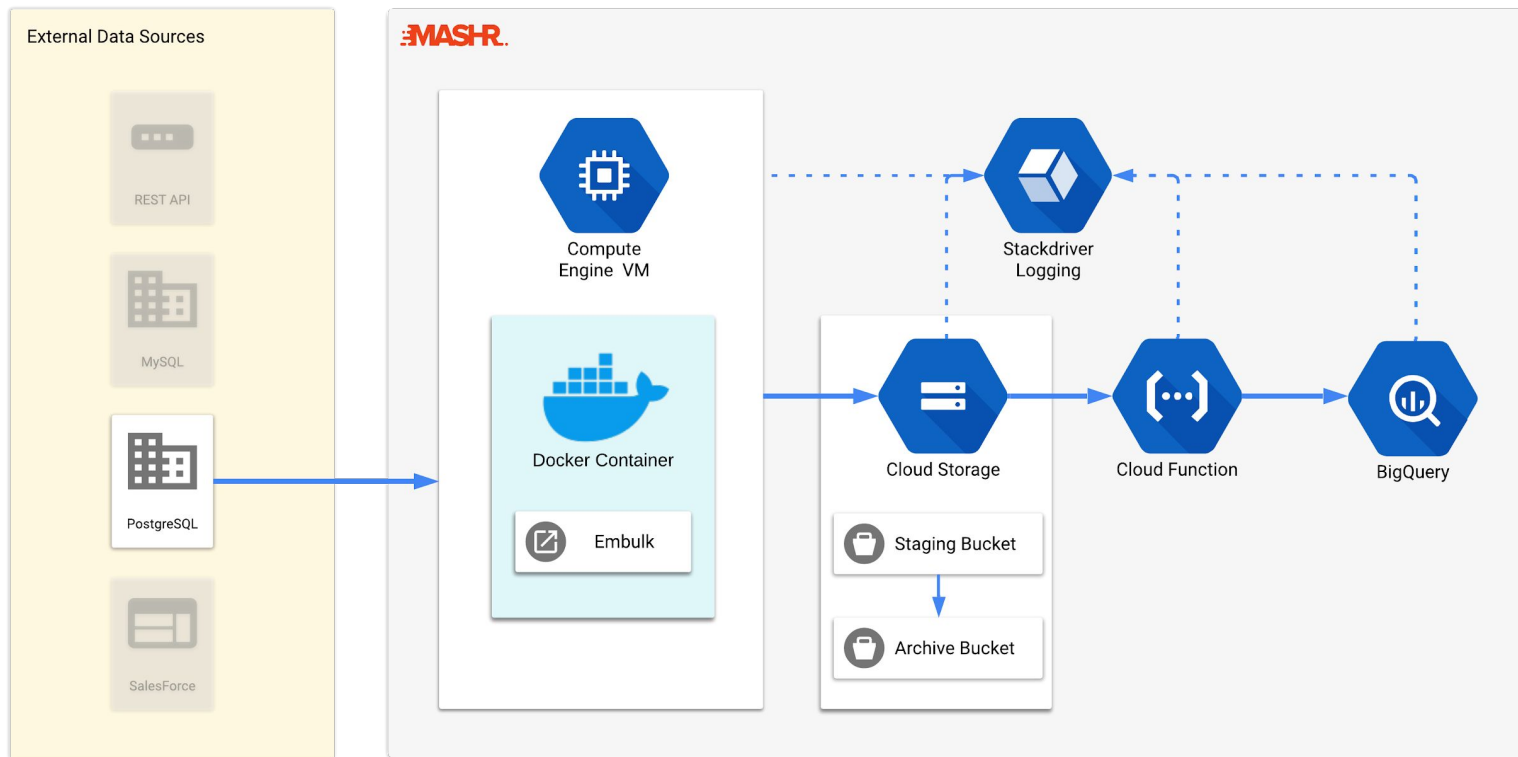
### Cons:

- Requires hosting

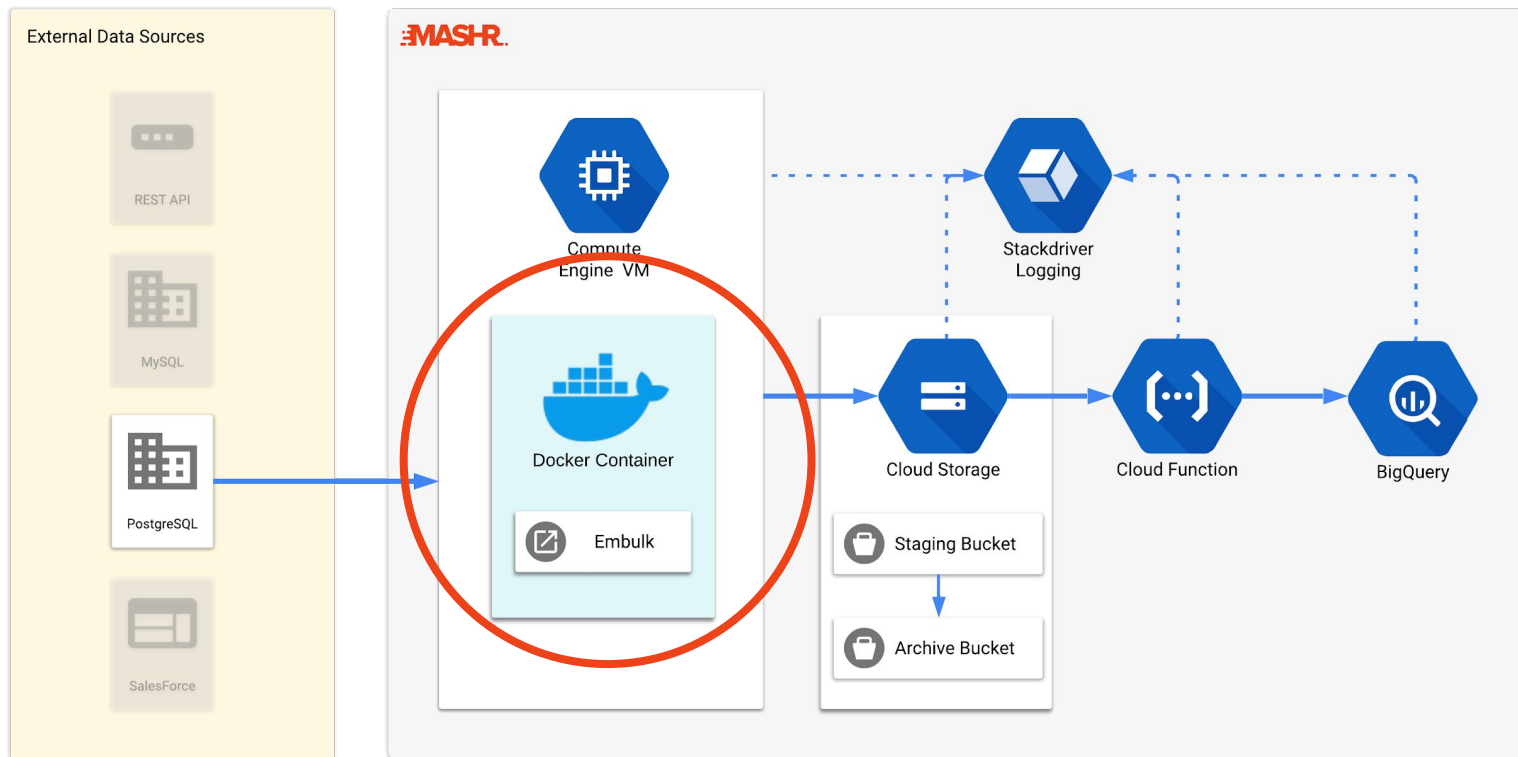
```
in:
  type: http
  url: https://jsonplaceholder.typicode.com/posts/42
  method: get
  parser:
    type: json
    schema:
      - { name: "userId", type: string }
      - { name: "id", type: long }
      - { name: "title", type: string }
      - { name: "body", type: string }
out:
  type: 'gcs'
  bucket: 'mashrintname'
  path_prefix: {{ env.DATE }}
  file_ext: '.json'
  auth_method: 'compute_engine'
  formatter:
    type: 'jsonl'
```



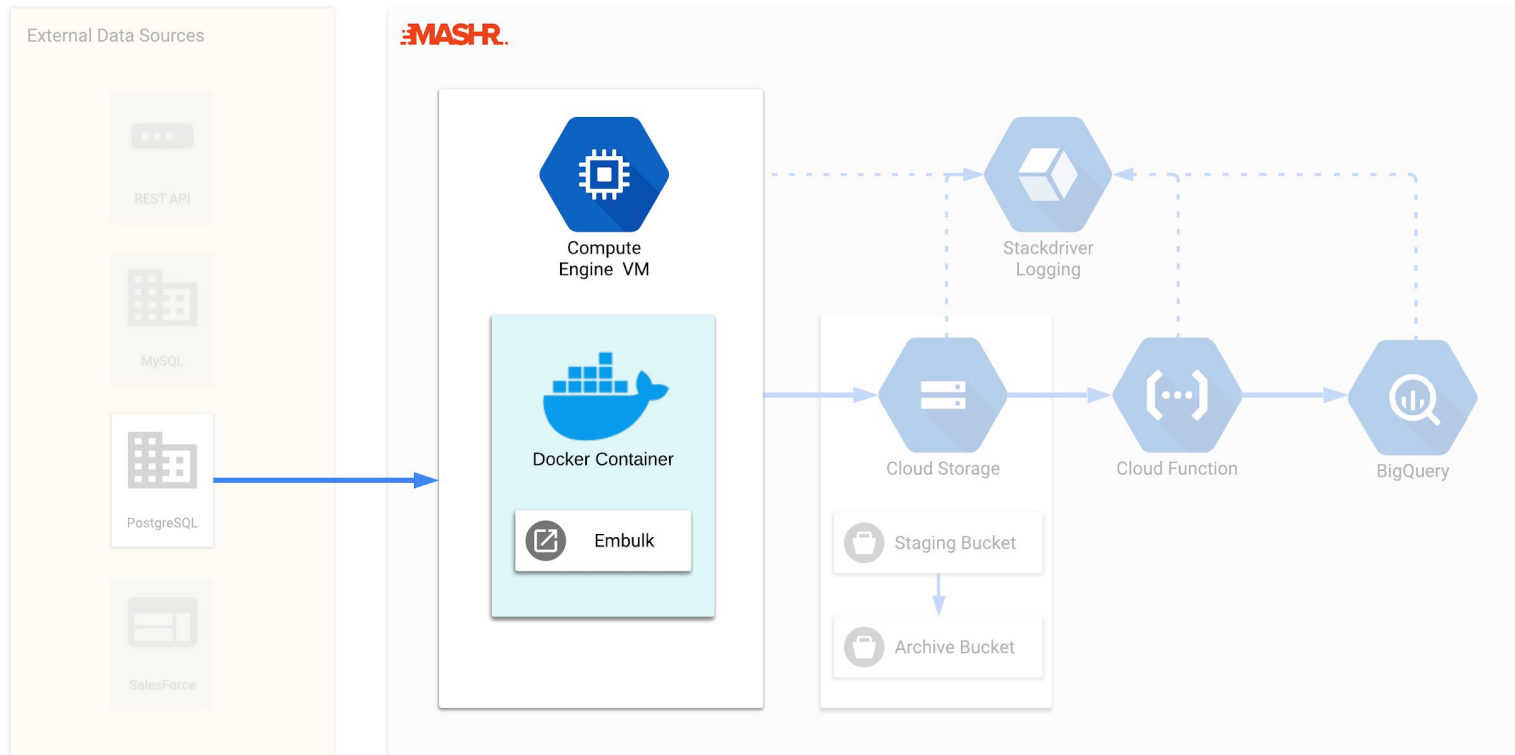
# Mashr



# Mashr



# Mashr



# Docker

```
FROM openjdk:8-jre-stretch

RUN apt-get -y update && apt-get -y upgrade

RUN apt-get install -y vim git zip unzip less wget

ENV EMBULK_VERSION 0.9.17

RUN curl -o /usr/local/bin/embulk \
  --create-dirs -L "http://dl.embulk.org/embulk-${EMBULK_VERSION}.jar" && \
  chmod +x /usr/local/bin/embulk

WORKDIR /root
RUN mkdir mashr

RUN apt-get install cron -y
```

```
FROM jacobleecd/mashr:latest
WORKDIR /root
COPY mashr/ mashr/

RUN chmod +x mashr/install_gems.sh
RUN mashr/install_gems.sh
RUN embulk gem install embulk-output-gcs
RUN embulk gem install embulk-formatter-jsonl

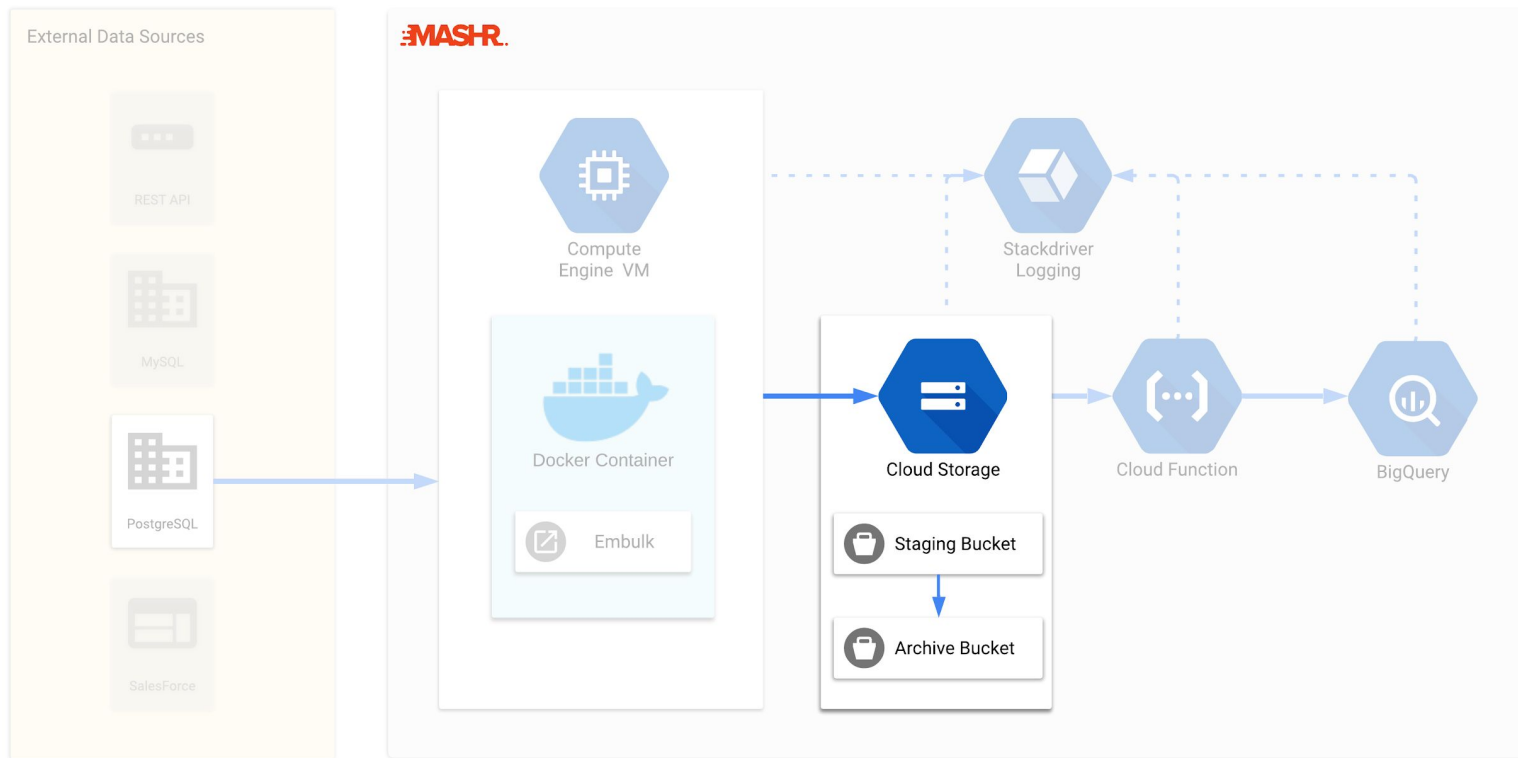
# Copy and make executable embulkScript.sh
COPY embulkScript.sh /etc/cron.d/embulkScript.sh
RUN chmod +x /etc/cron.d/embulkScript.sh

# Copy and make executable crontab, mashr-cron
COPY crontab /etc/cron.d/mashr-cron
RUN chmod 0644 /etc/cron.d/mashr-cron

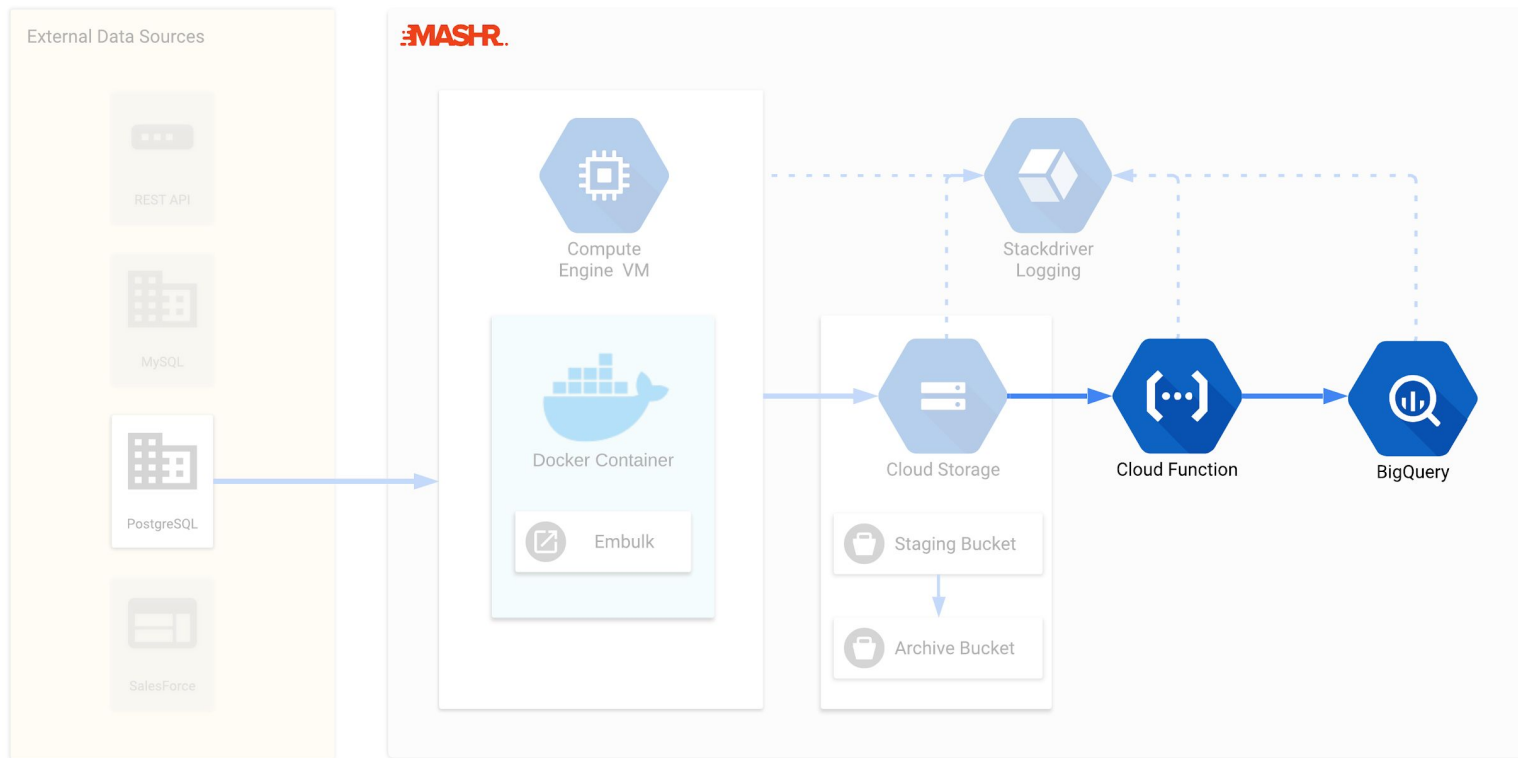
# activate crontab, mashr-cron
RUN crontab /etc/cron.d/mashr-cron

RUN touch /var/log/cron.log
CMD service cron start && tail -f /var/log/cron.log
```

# Mashr

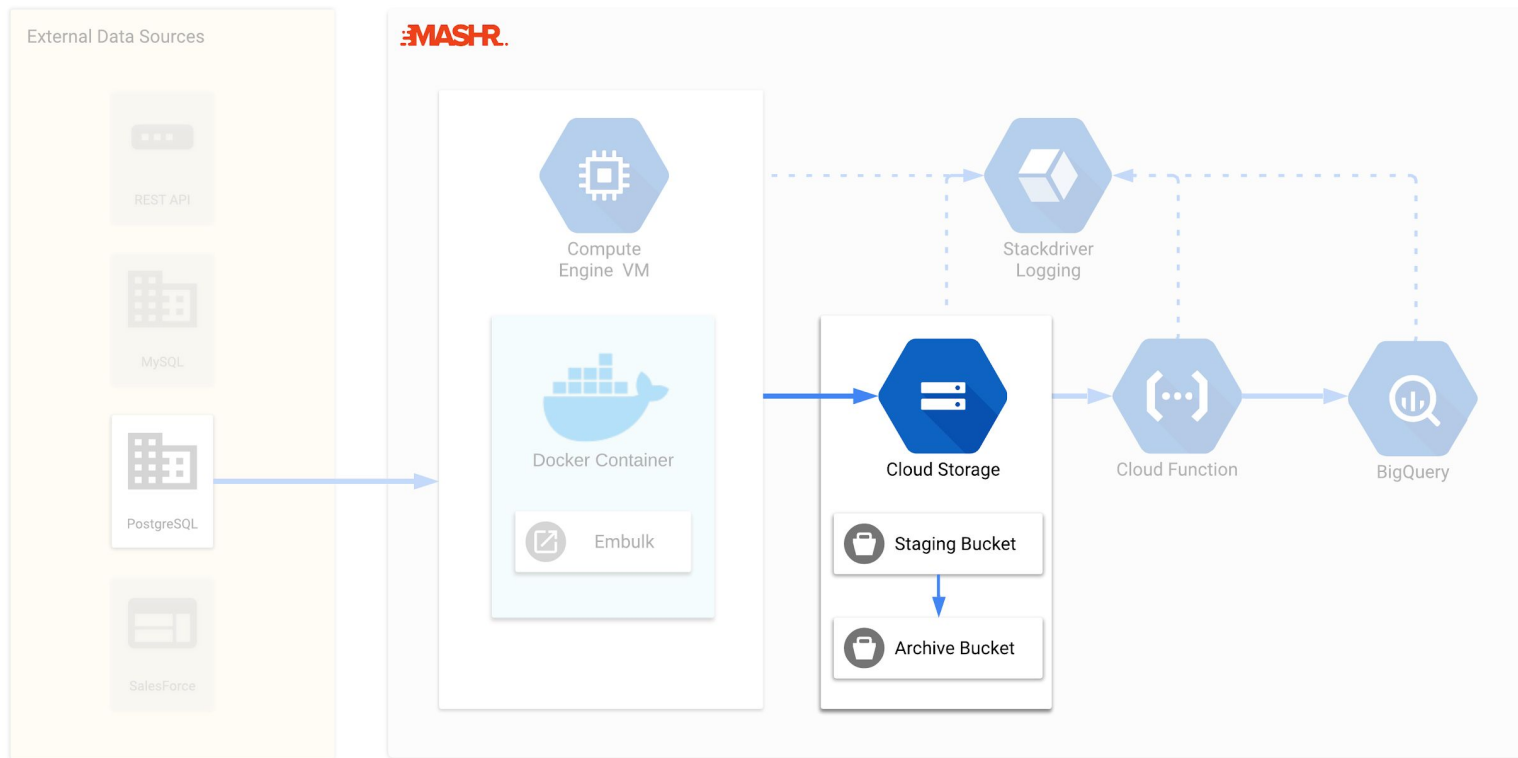


# Mashr

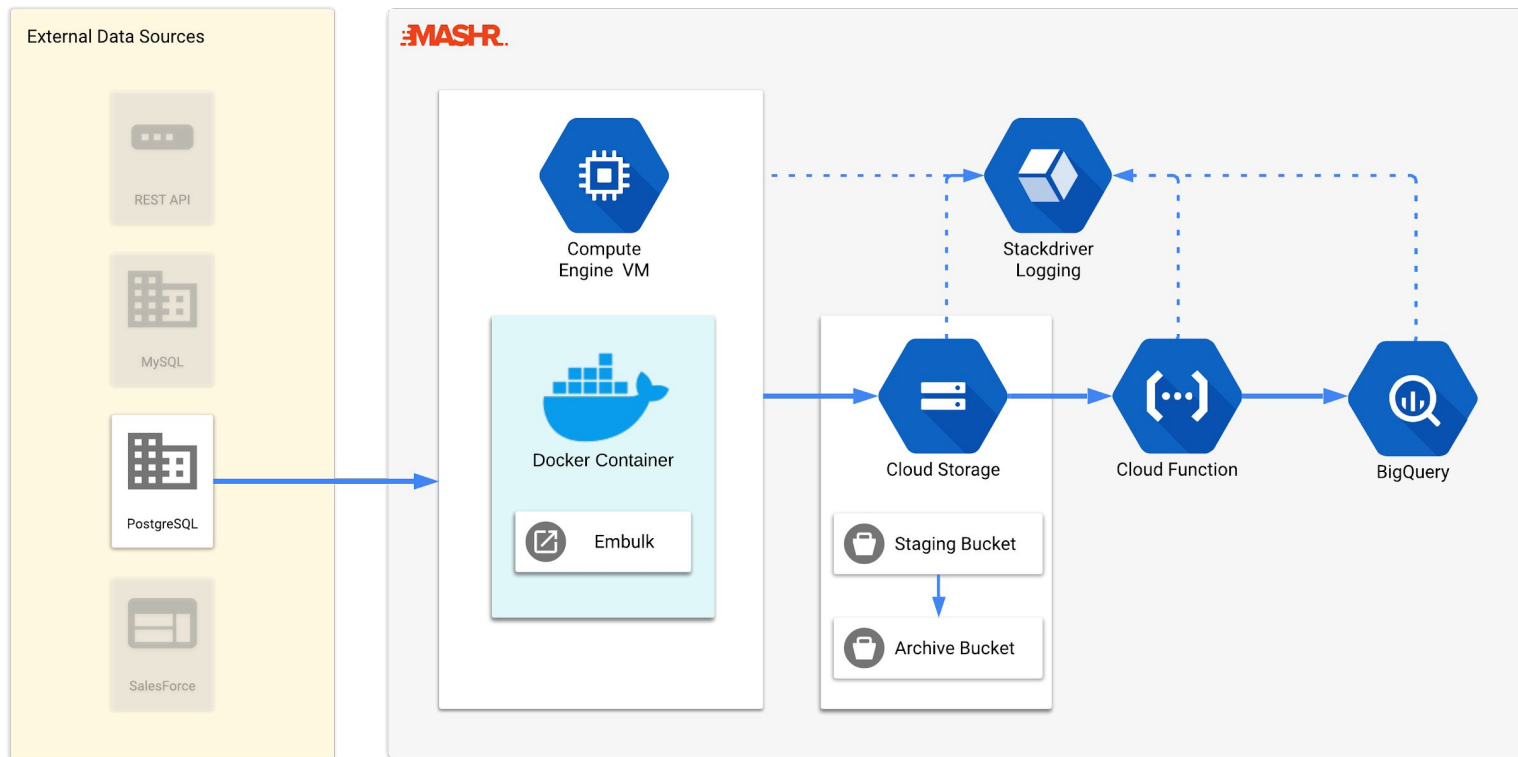




# Mashr



# Mashr



# Mashr commands

- `init` - creates a yaml configuration file in the users working directory
- `deploy` - launches all of the GCP resources to create the data pipeline
- `destroy` - destroys all of the GCP resources of a specific data pipeline
- `list` - lists your current data pipelines
- `help` - help text for Mashr



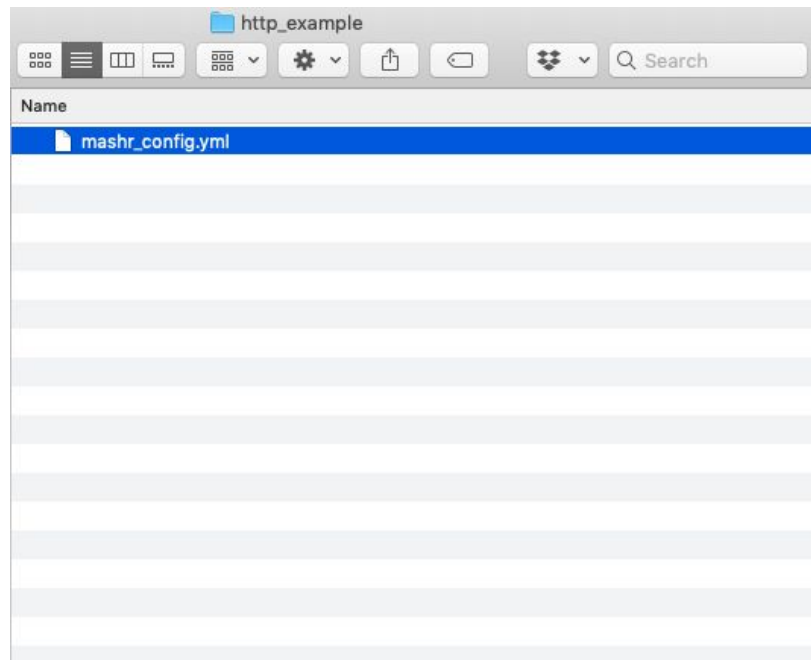
# mashr init



# mashr init

```
mashr_config.yml

mashr:
  service_account_email: ''
  json_keyfile: ''
  table_id: ''
  dataset_id: ''
  project_id: ''
  integration_name: ''
  embulk_run_command: 'embulk run embulk_config.yml'
  embulk_gems:
    - embulk-input-http
  embulk:
    in:
      type: http
      url: https://jsonplaceholder.typicode.com/posts/42
      method: get
      parser:
        type: json
        schema:
          - { name: "userId", type: string }
          - { name: "id", type: long }
          - { name: "title", type: string }
          - { name: "body", type: string }
```



# mashr init

```
mashr_config.yml

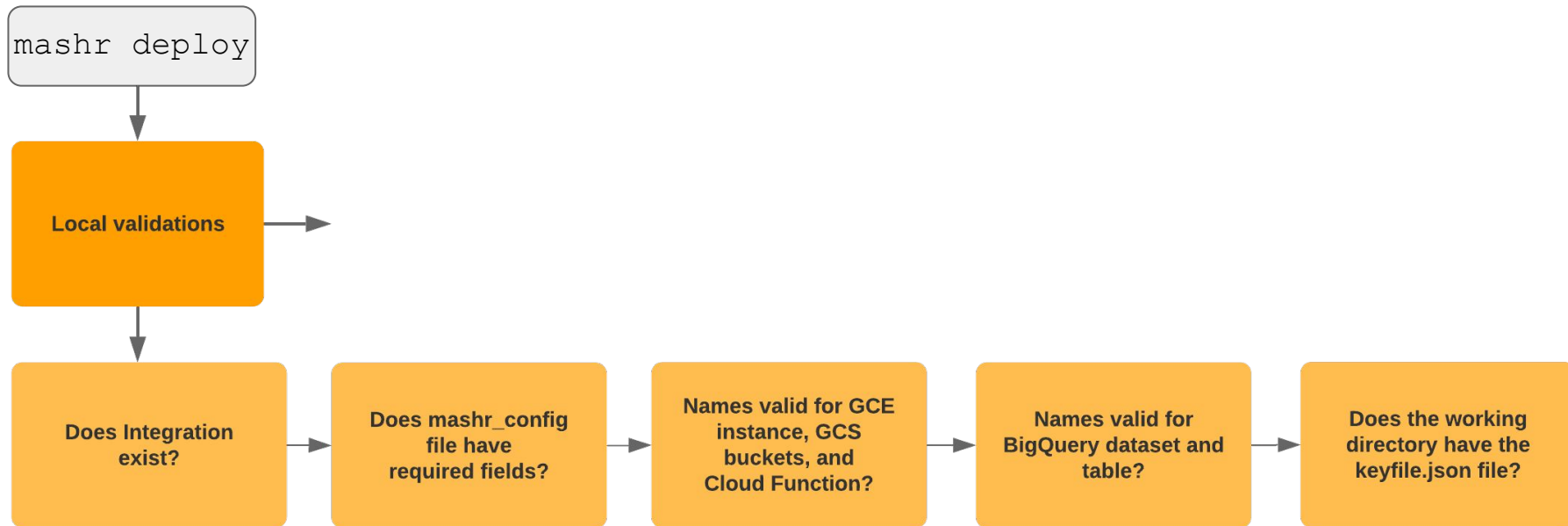
mashr:
  service_account_email: 'example@example.iam.gserviceaccount.com'
  json_keyfile: 'keyfile.json'
  table_id: 'myTableId'
  dataset_id: 'myDatasetId'
  project_id: 'exampleProject'
  integration_name: 'exampleIntegrationName'
  embulk_run_command: 'embulk run embulk_config.yml'
  embulk_gems:
    - embulk-input-http
  embulk:
    in:
      type: http
      url: https://jsonplaceholder.typicode.com/posts/42
      method: get
      parser:
        type: json
        schema:
          - { name: "userId", type: string }
          - { name: "id", type: long }
          - { name: "title", type: string }
          - { name: "body", type: string }
```



# mashr deploy

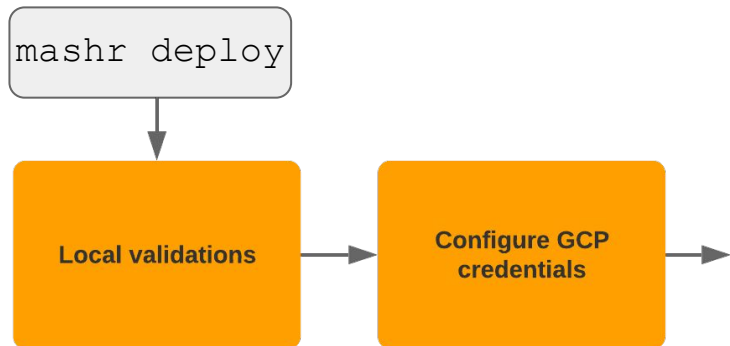


# mashr deploy

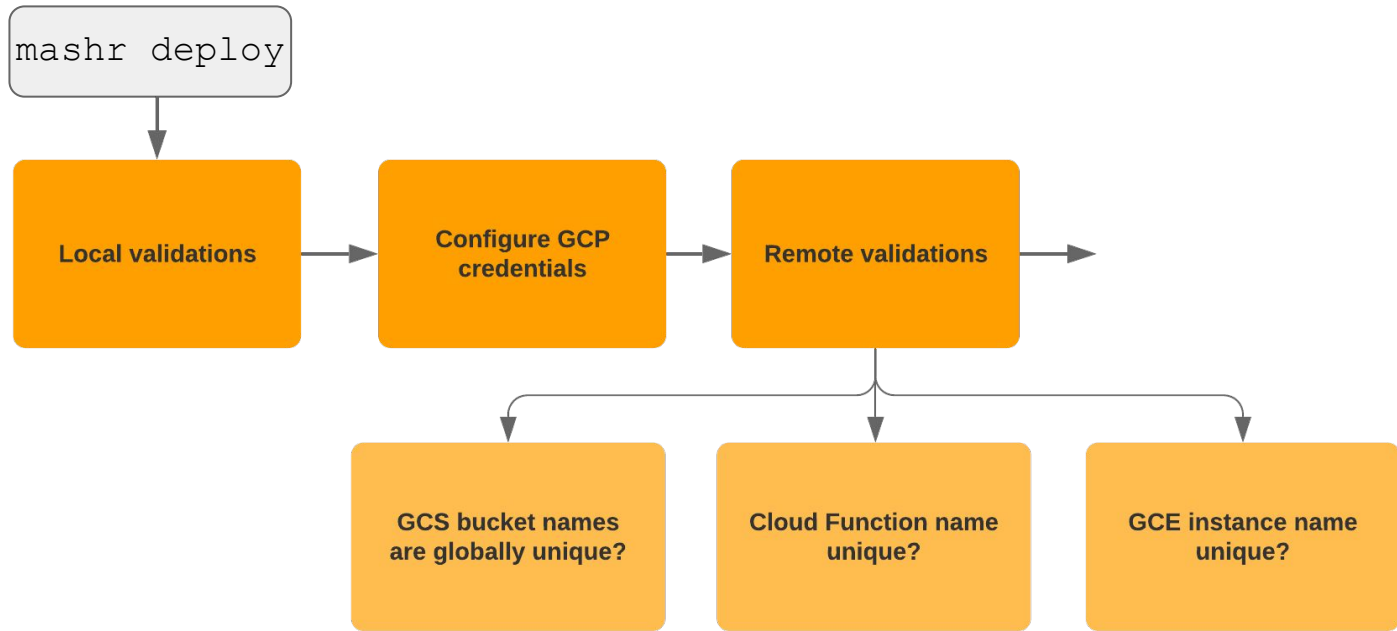




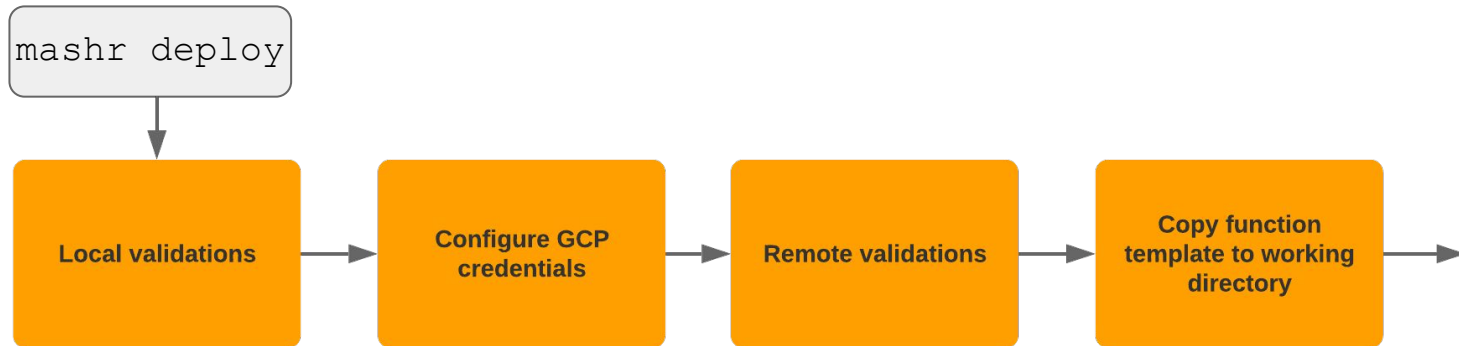
# mashr deploy



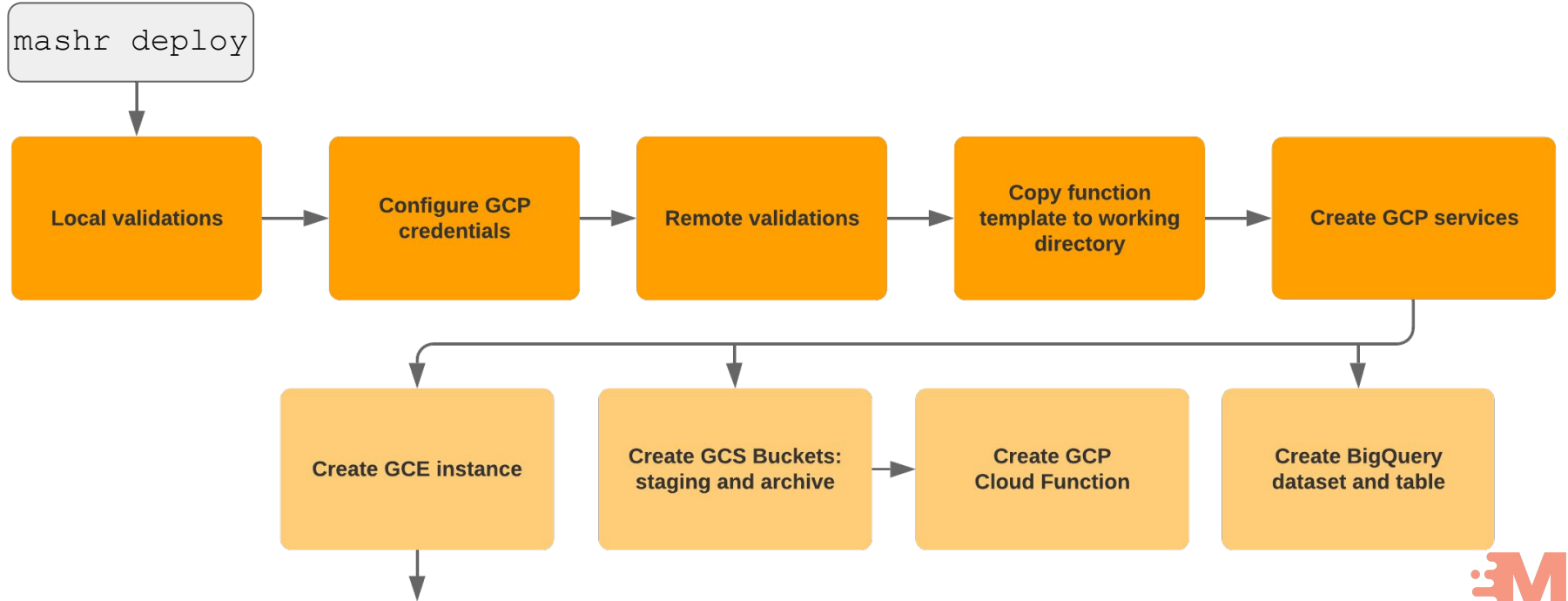
# mashr deploy



# mashr deploy

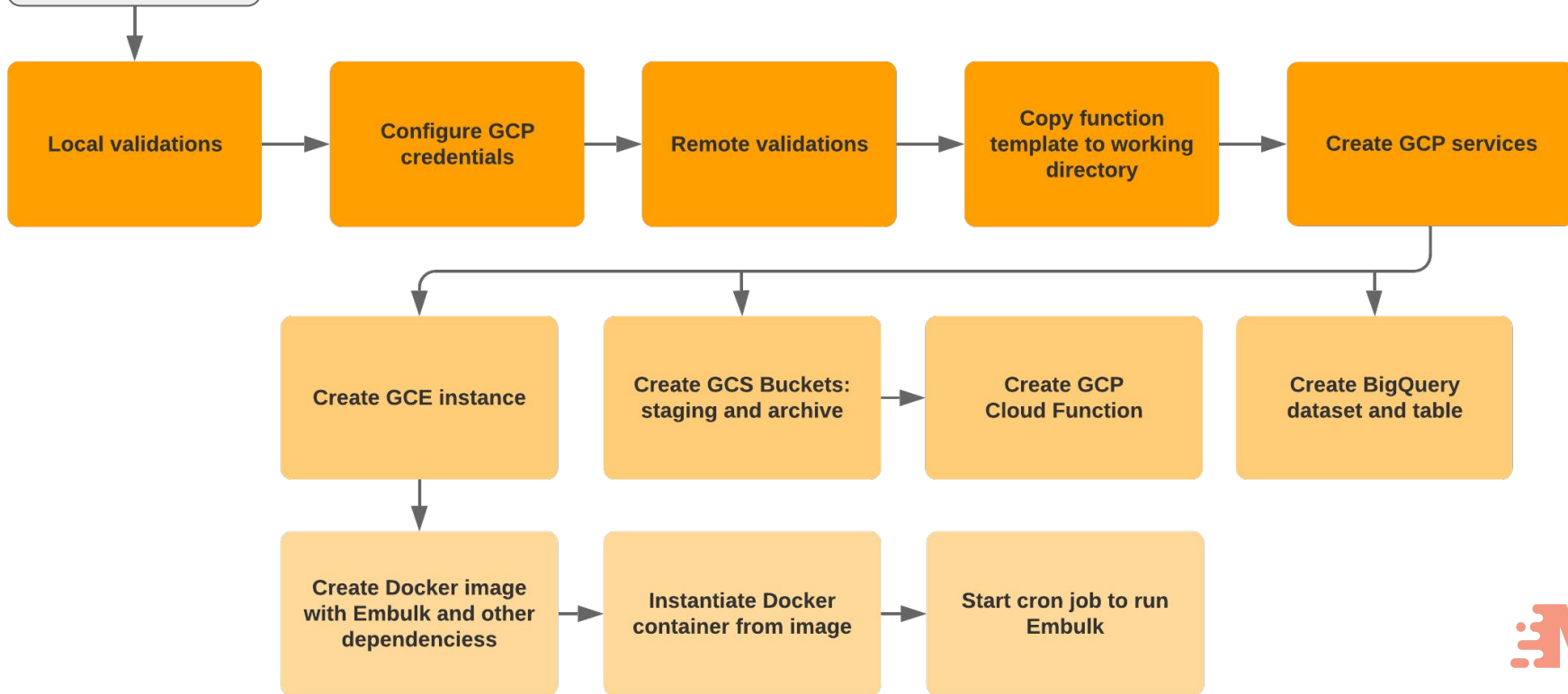


# mashr deploy

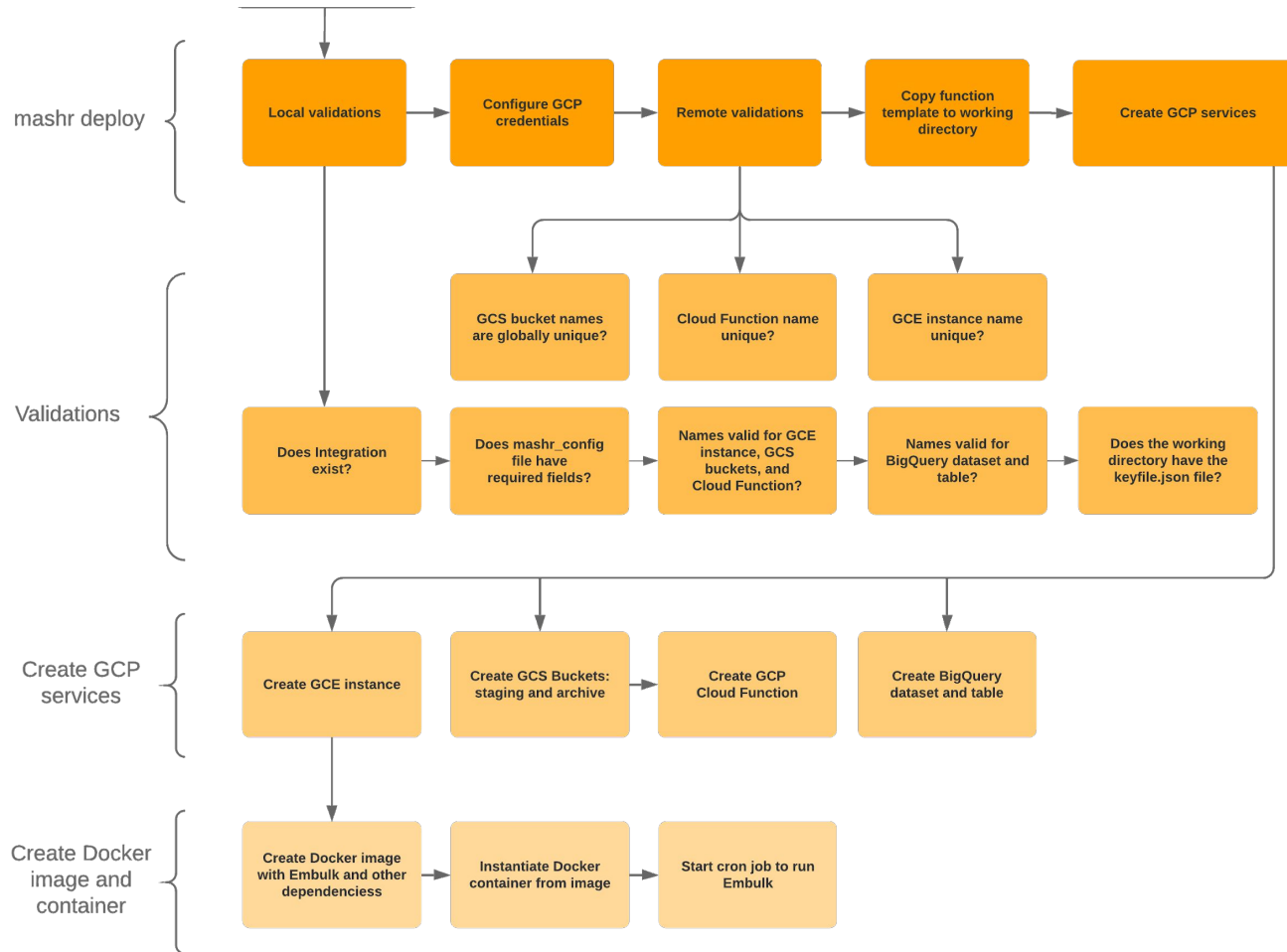


# mashr deploy

mashr deploy



## Mashr Deploy Command



# **mashr destroy**



# Mashr





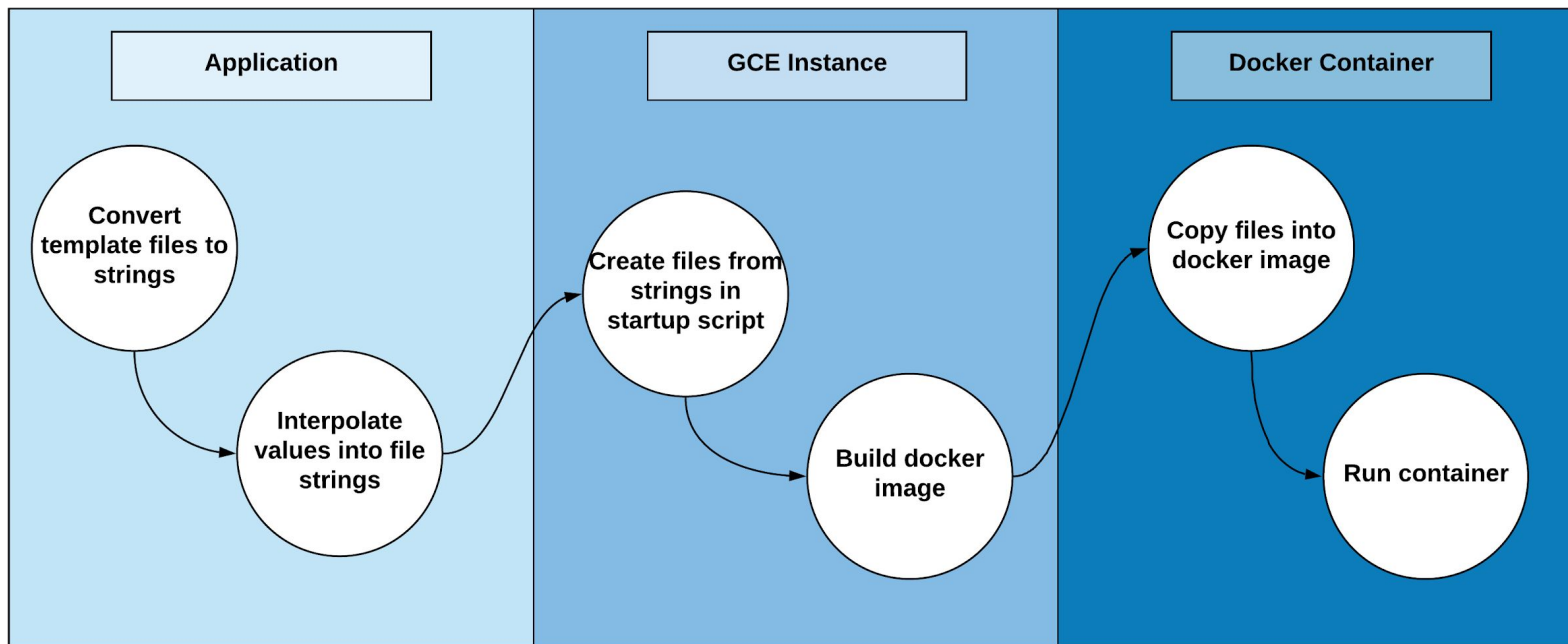
# Principles of a good Data Pipeline

- Extraction
- Validation
- Transformation
- Staging / Archiving
- Publishing to target
- Monitoring
- Orchestration and Ordering
- Performance



# Challenges

## Docker Containers



# Challenges

## Docker Cron Jobs

Where do we schedule embulk jobs to run at a regular interval?

Either:

- A)** A cron job on the virtual machine that the container runs on
- B)** Cloud Scheduler, GCP's built in cron service
- C)** A cron job that runs in a separate container
- D)** A cron job running inside the docker container itself



# Challenges

## Docker Cron Jobs

Either:

- ~~A) A cron job on the virtual machine that the container runs on~~
- ~~B) Cloud Scheduler, GCP's built-in cron service~~
- ~~C) A cron job that runs in a separate container~~
- D) A cron job running inside the docker container itself



```
CMD service cron start && tail -f /var/log/cron.log
```



# Challenges

## Docker Logging

How to get the output of cron jobs to Stackdriver

```
const createEmbulkScript = (runCommand) => {
  runCommand = runCommand.replace(
    'embulk_config.yml', '/root/mashr/embulk_config.yml.liquid');
  // sends logs of cron job to /proc/1/fd/1, where docker listens
  const script =
`#!/bin/bash
export DATE=$(date +"%Y-%m-%dT%H-%M-%S-%3N")
${runCommand} >> /proc/1/fd/1 2>&1
`,

  return script;
};
```



# Future Work

- Enable cross platform support - AWS!
- Enable other target destinations
- Enable users authentication with OAuth instead of keyfiles.
- Automatic schema pre-check and creation to ensure that your input values will upload to BigQuery successfully
- Add a 'redeploy' command that allows users to overwrite existing integrations
- Explore the option of using Kubernetes to protect against GCE failover



# Thanks!

Jacob Coker-Dukowitz

<http://jacobcd.io>

Mashr Github

<https://github.com/mashr-framework/mashr>

