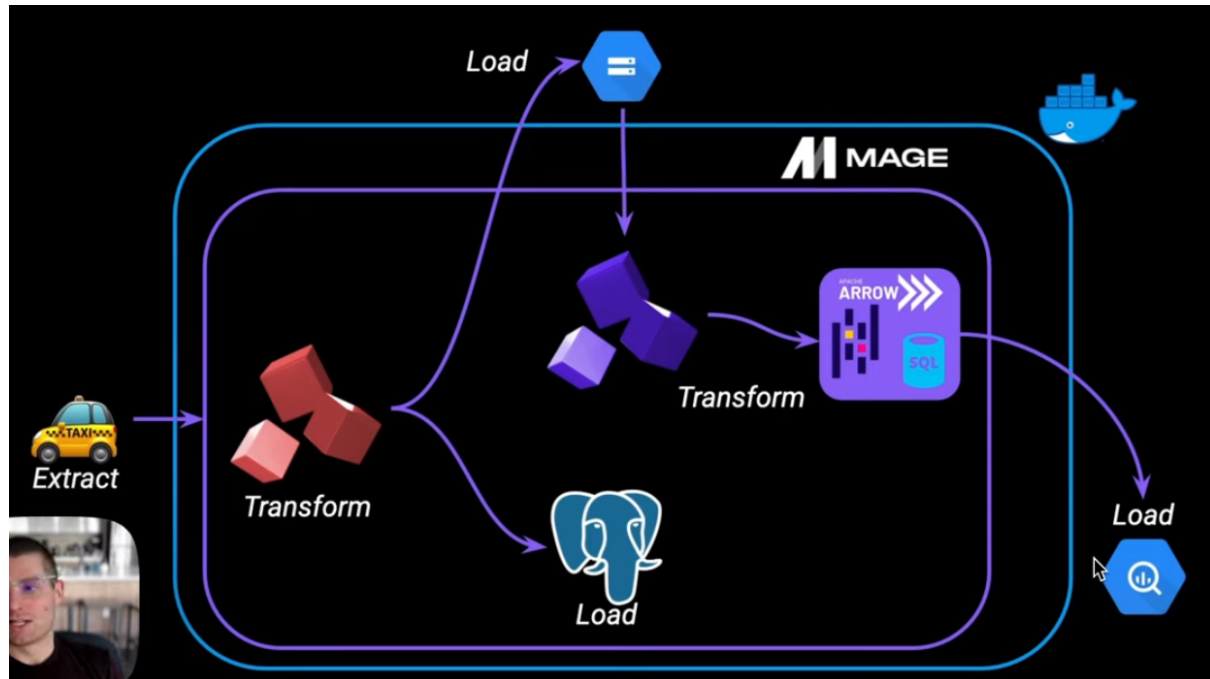


(32) DE Zoomcamp 2.2.1 - What is Orchestration? - YouTube

Final goal



What is Orchestration?

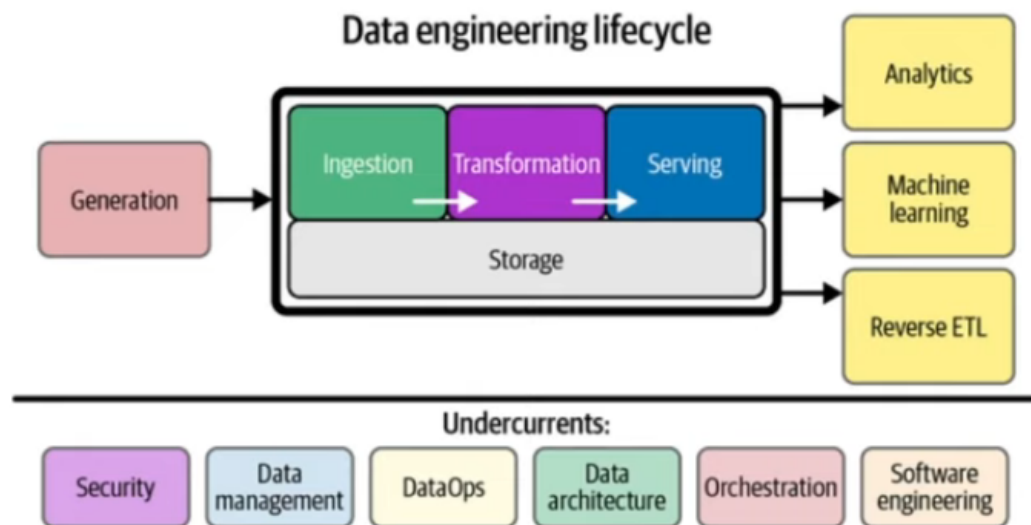
A large part of data engineering is **extracting, transforming, and loading** data between sources.

Orchestration is a process of dependency management, facilitated through **automation**.

The data orchestrator manages scheduling, triggering, monitoring, and resource allocation.

Step = task = block

Workflog = dag = directed acyclic graphs



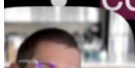
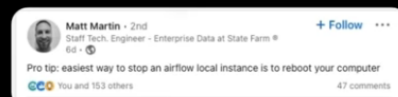
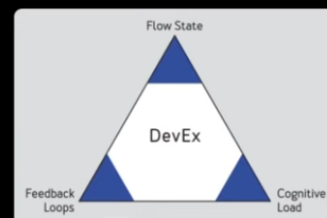
A good orchestrator handles...

- Workflow management
- Automation
- Error handling
- Recovery
- Monitoring, alerting
- Resource optimization
- Observability
- Debugging
- Compliance/Auditing

A good orchestrator prioritizes....

The developer experience

- **Flow state** 🌊
 - "I need to switch between 7 tools/services."
- **Feedback Loops** 🔄
 - "I spent 5 hours locally testing this DAG."
- **Cognitive Load** 🧠

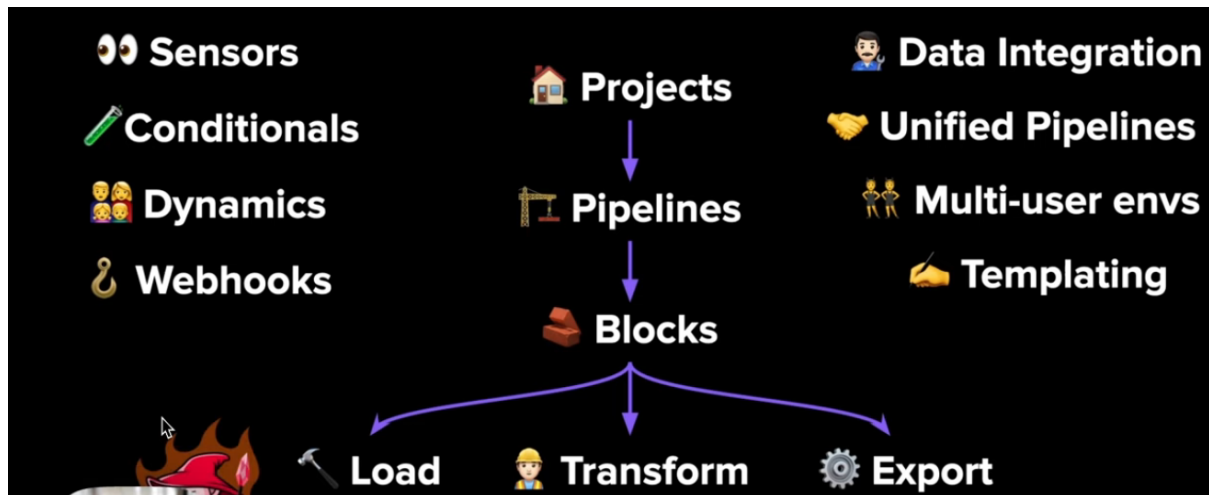
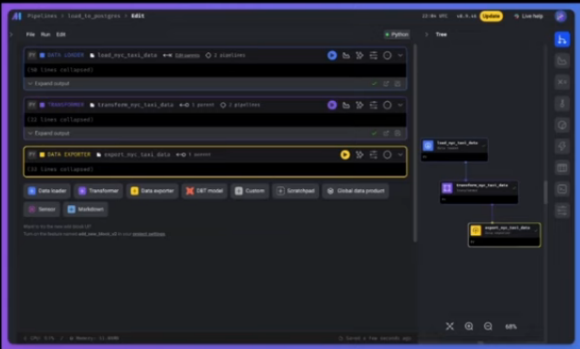


How much do you need to know to do your job?

(32) DE Zoomcamp 2.2.2 - What is Mage? - YouTube

What is Mage?

An open-source pipeline tool for orchestrating, transforming, and integrating data 🧑‍🔧



🚀 Mage accelerates pipeline development

- Hybrid environment
 - Use our **GUI** for interactive development (or don't, I like VSCode)
 - Use **blocks** as testable, reusable pieces of code.
- Improved DevEx
 - Code and test in parallel.
 - Reduce your dependencies, switch tools less, be efficient.

Engineering best-practices built-in

-  In-line testing and debugging
 - Familiar, notebook-style format
-  Fully-featured observability
 - Transformation *in one place*: dbt models, streaming, & more.
-  DRY principles
 - No more  DAGs with duplicate functions and weird imports
 - DEaaS (sorry, I had to 😅)

M MAGE

 Projects

 Pipelines

 Blocks





Projects

- A project forms the basis for all the work you can do in Mage—you can think of it like a GitHub repo.
- It contains the code for all of your pipelines, blocks, and other assets.
- A Mage instance has one or more projects



Pipelines

- A pipeline is a workflow that executes some data operation—maybe extracting, transforming, and loading data from an API. They're also called DAGs on other platforms
- In Mage, pipelines can contain *Blocks* (written in SQL, Python, or R) and charts.
- Each pipeline is represented by a YAML file in the “pipelines” folder of your project.



Blocks

- A block is a file that can be executed independently or within a pipeline.
- Together, blocks form Directed Acyclic Graphs (DAGs), which we call pipelines.
- A block won't start running in a pipeline until all its upstream dependencies are met.

Blocks continued

- Blocks are reusable, atomic pieces of code that perform certain actions.
- Changing one block will change it everywhere it's used, but **don't worry**, it's easy to detach blocks to separate instances if necessary.
- Blocks can be used to perform a variety of actions, from simple data transformations to complex machine learning models.

blocks should return dataframes

Anatomy of a Block

```
1 import io
2 import pandas as pd
3 import requests
4 if "data_loader" not in globals():
5     from mage_ai.data_preparation.decorators import data_loader
6 if "test" not in globals():
7     from mage_ai.data_preparation.decorators import test
8
9 @data_loader
10 def load_data_from_api(*args, **kwargs):
11     url = "https://github.com/DataTalksClub/nyc-tlc-data/releases/download/yellow/yellow_tripdata_2021-01.csv.gz"
12
13     taxi_dtypes = {
14         # native data parsing
15         'tpep_pickup_datetime': 'datetime',
16         'tpep_dropoff_datetime': 'datetime'
17     }
18     return pd.read_csv(
19         url,
20         dtype=taxi_dtypes,
21         compression='gzip'
22     )
23
24 @test
25 def test_output(output, *args) -> None:
26     """
27     Test code for testing the output of the block.
28     """
29     assert output is not None, "The output is undefined"
```

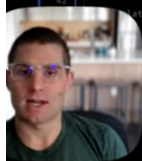
Imports

Decorator

Function*

Assertion

*returns df



tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID
1609461010000	1609461372000	1	2.1	1	N	142

(32) DE Zoomcamp 2.2.2 - Configure Mage - YouTube

[mage-ai/mage-zoomcamp](https://github.com/mage-ai/mage-zoomcamp): This repository will contain all of the resources for the Mage component of the Data Engineering Zoomcamp:
<https://github.com/DataTalksClub/data-engineering-zoomcamp/tree/main>

1. git clone <https://github.com/mage-ai/mage-zoomcamp.git> mage-zoomcamp
2. cd the folder
3. Create .env from dev.env
4. docker compose build
5. docker pull mageai/mageai:latest (optional)
6. docker compose up
7. <http://localhost:6789/>

[DE Zoomcamp 2.2.2 - A Simple Pipeline \(youtube.com\)](#)

If two blocks are connected DFs will be passed between them. The result of one is the input of the other

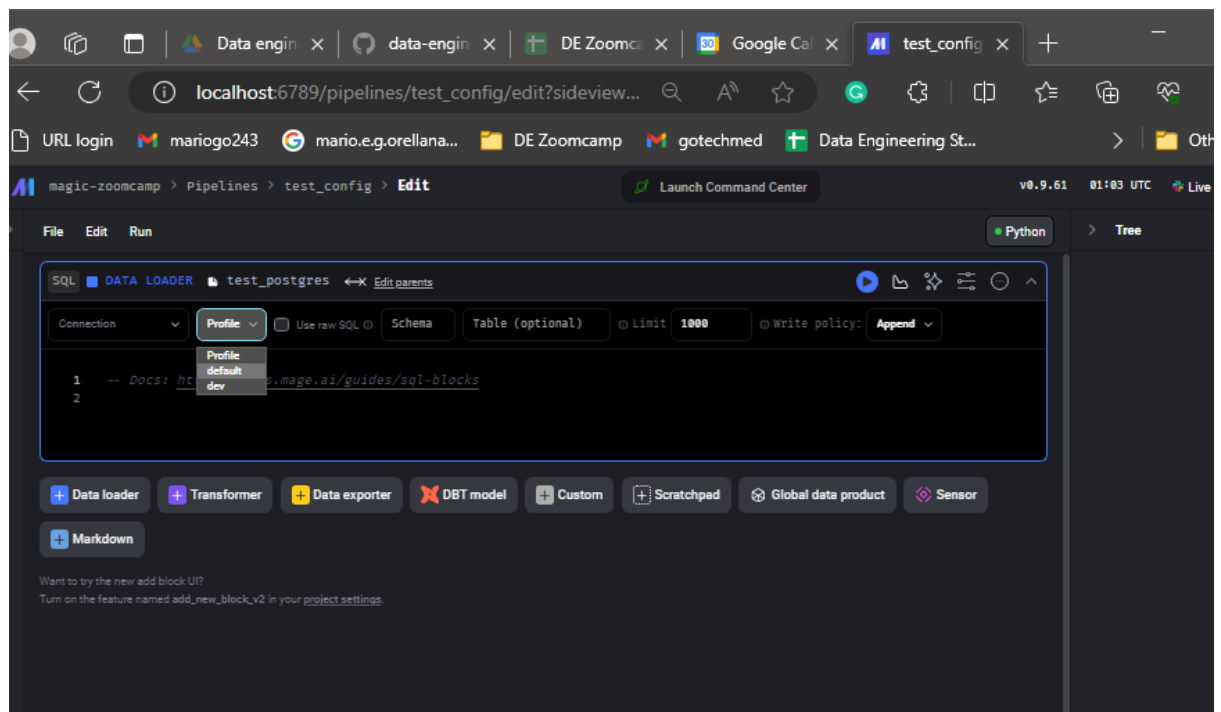
[Pipelines | Mage](#)

DE Zoomcamp 2.2.3 - Configuring Postgres (youtube.com)

1. Setup io_config.yaml is where we manage the connection, we can define a new profile
2. Let's set up a new profile called dev, let's set up environment variables, the way to do this in mage is with jinja notation "{{ }}"

```
POSTGRES_DBNAME: "{{ env_var('POSTGRES_DBNAME') }}"
dev:
  # PostgreSQL
  POSTGRES_CONNECT_TIMEOUT: 10
  POSTGRES_DBNAME: "{{ env_var('POSTGRES_DBNAME') }}"
  POSTGRES_SCHEMA: "{{ env_var('POSTGRES_SCHEMA') }}"
  POSTGRES_USER: "{{ env_var('POSTGRES_USER') }}"
  POSTGRES_PASSWORD: "{{ env_var('POSTGRES_PASSWORD') }}"
  POSTGRES_HOST: "{{ env_var('POSTGRES_HOST') }}"
  POSTGRES_PORT: "{{ env_var('POSTGRES_PORT') }}"
```

3. Create new pipeline and change name in edit pipeline settings
4. Add a new sql data loader
5. define profile



6. RUN SELECT 1; to test the connection

DE Zoomcamp 2.2.3 - ETL: API to Postgres (youtube.com)

[data-engineering-zoomcamp/Module 2/my_mage_files_at_master · MarioOrellana58/data-engineering-zoomcamp \(github.com\)](https://github.com/MarioOrellana58/data-engineering-zoomcamp)

1. Create a new pipeline
2. Add a new data loader as python api
3. Add a new transformer python generic
4. Add a new data exporter python postgres
5. Add a new data loader for sql postgres to inspect the data, select postgres connection and mark raw sql checkbox

DE Zoomcamp 2.2.4 - Configuring GCP (youtube.com)

1. create a new google cloud storage bucket
2. Go to IAM > service accounts
3. Go to the new account and create a new key, paste the json into mage directory
4. Make sure that docker-compose.yml has this, this will make that all the files of my localhost directory will be mapped into that directory inside docker

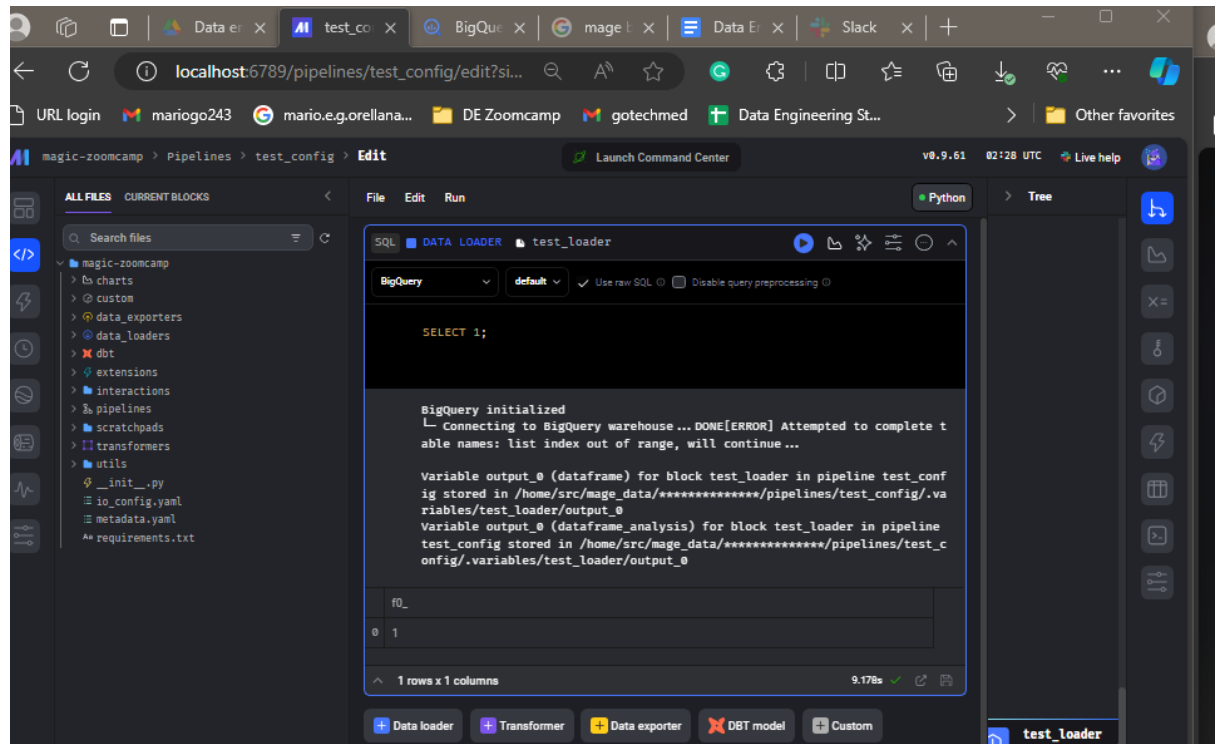
```
volumes:  
  - ./home/src/
```

5. Open mage, go to files then to io_config.yaml
 - a. Here you will find two ways to authenticate with Google, one is to paste all the payload from the key into the file, delete this one

```
0.  GOOGLE_SERVICE_ACC_KEY:  
1.      type: service_account  
2.      project_id: project-id  
3.      private_key_id: key-id  
4.      private_key: "-----BEGIN PRIVATE  
5. KEY-----\nyour_private_key\n-----END_PRIVATE_KEY"  
6.      client_email: your_service_account_email  
7.      auth_uri: "https://accounts.google.com/o/oauth2/auth"  
8.      token_uri: "https://accounts.google.com/o/oauth2/token"  
9.      auth_provider_x509_cert_url:  
10. "https://www.googleapis.com/oauth2/v1/certs"  
11.      client_x509_cert_url:  
12. "https://www.googleapis.com/robot/v1/metadata/x509/your_serv  
ice_account_email"
```

- I. Use GOOGLE_SERVICE_ACC_KEY_FILEPATH instead
 - i. Go to terminal, run ls -la to check the files
 - ii. Copy the key name
 - iii. Update io config
6. Save the file
 7. Use the test_config pipeline, change the connection to bigquery, then default profile, running it threw indexerror, here's the solution [Data Engineering Zoomcamp FAQ -](#)

[Documentos de Google](#)



8. Go to example_pipeline go to the last step, execute with the previous steps
9. Upload titanic_clean.csv to gcp bucket
10. go to test config pipeline, delete the bigquery block
11. Create a new data loader, python, google cloud storage
12. Update bucket name and object key (filename)
13. Run it

DE Zoomcamp 2.2.4 - ETL: API to GCS (youtube.com)

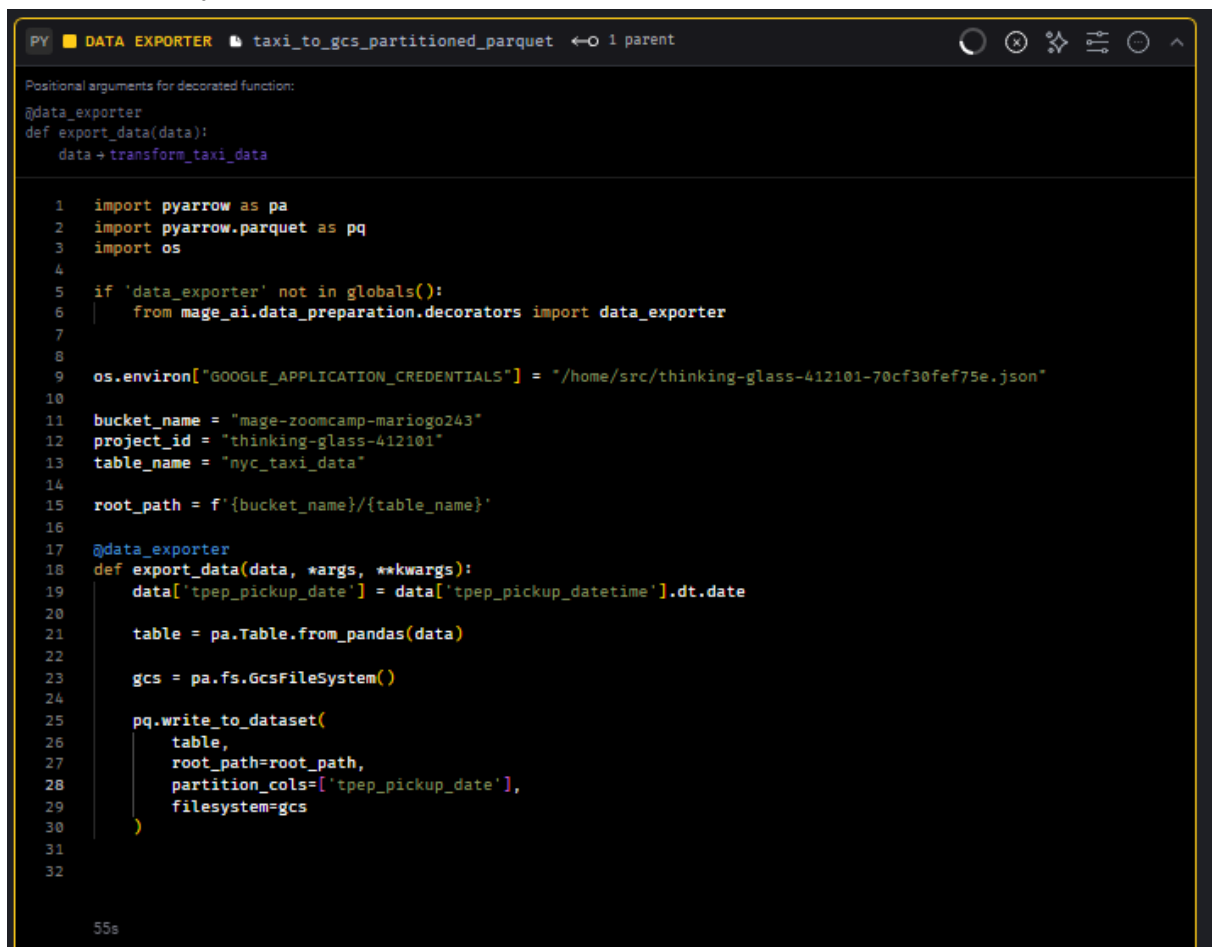
[data-engineering-zoomcamp/Module 2/my_mage_files/3. Load Data GCS Partitioned.py at master · MarioOrellana58/data-engineering-zoomcamp \(github.com\)](#)

Write to a single parquet file

1. New standard pipeline
2. Expand data_loaders folder in the left, then drag and drop load_api_data
3. Expand transformers, drag and drop transform taxi data
4. In the blocks view connect the two blocks dragging the lower dot of data loader to the top dot of transformer
5. Create a new data exporter python gcs
6. Run all the blocks

Write to a partitioned parquet file

1. Add a new data exporter python generic
2. Create a new date column
3. Partition data by date



```
PY DATA EXPORTER taxi_to_gcs_partitioned_parquet ← 1 parent

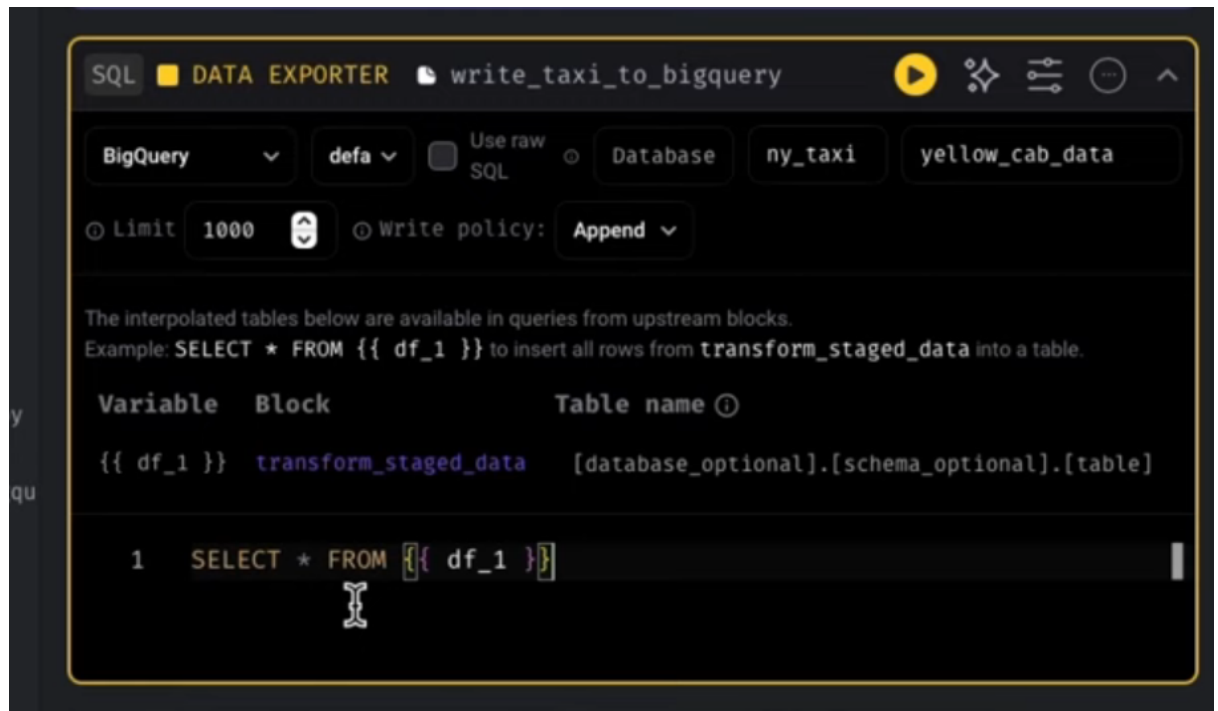
Positional arguments for decorated function:
@data_exporter
def export_data(data):
    data = transform_taxi_data

1  import pyarrow as pa
2  import pyarrow.parquet as pq
3  import os
4
5  if 'data_exporter' not in globals():
6      from mage_ai.data_preparation.decorators import data_exporter
7
8
9  os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "/home/src/thinking-glass-412101-70cf30fef75e.json"
10
11  bucket_name = "mage-zoomcamp-mariogo243"
12  project_id = "thinking-glass-412101"
13  table_name = "nyc_taxi_data"
14
15  root_path = f'{bucket_name}/{table_name}'
16
17  @data_exporter
18  def export_data(data, *args, **kwargs):
19      data['tpep_pickup_date'] = data['tpep_pickup_datetime'].dt.date
20
21      table = pa.Table.from_pandas(data)
22
23      gcs = pa.fs.GcsFileSystem()
24
25      pq.write_to_dataset(
26          table,
27          root_path=root_path,
28          partition_cols=['tpep_pickup_date'],
29          filesystem=gcs
30      )
31
32

55s
```

(35) DE Zoomcamp 2.2.5 - ETL: GCS to BigQuery - YouTube

1. New pipeline
2. Drag the loader python gcs
3. Transform the DF
4. Export, if this sql export fails try python export



```
dfs = []
for object_key in object_keys:
    df =
pq.read_table(f'gs://{bucket_name}/{object_key}').to_pandas()
    dfs.append(df)

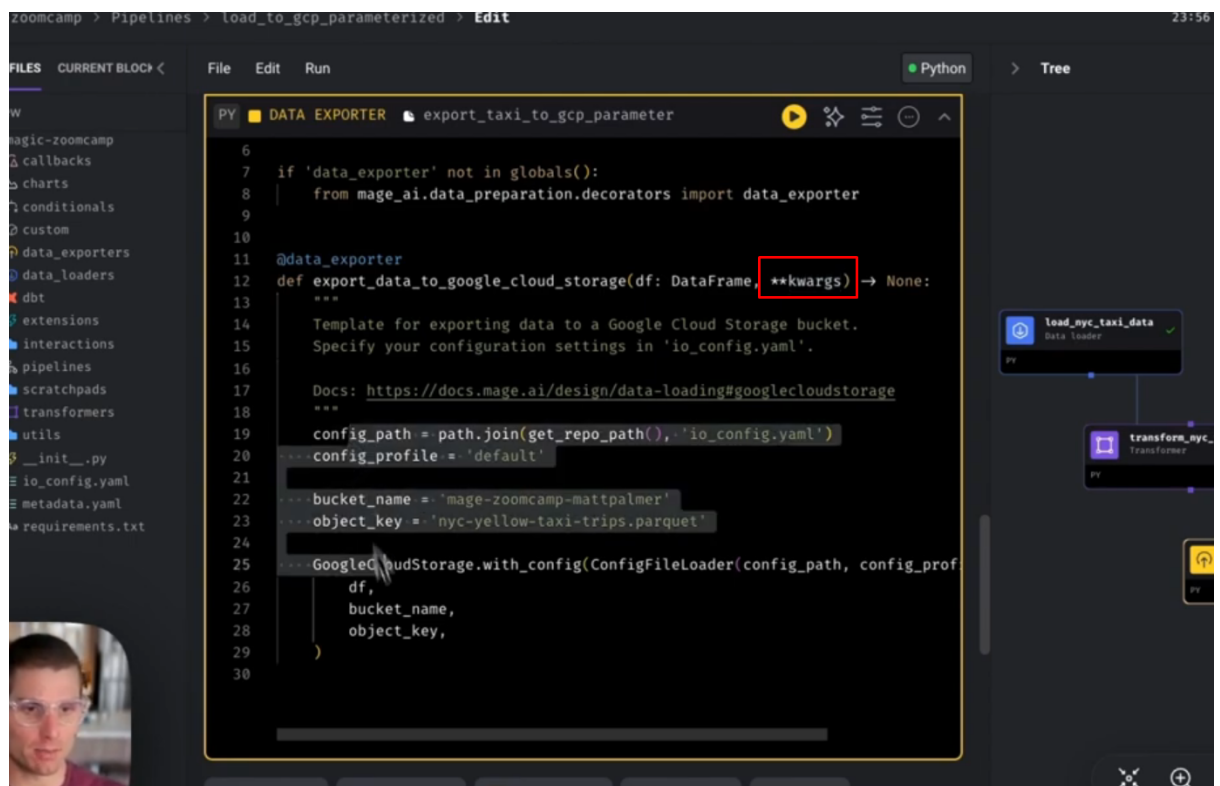
result_df = pd.concat(dfs, ignore_index=True)
```

DE Zoomcamp 2.2.6 - Parameterized Execution (youtube.com)

Mage has runtime variables, global variables, many variables... Look into the documentation.

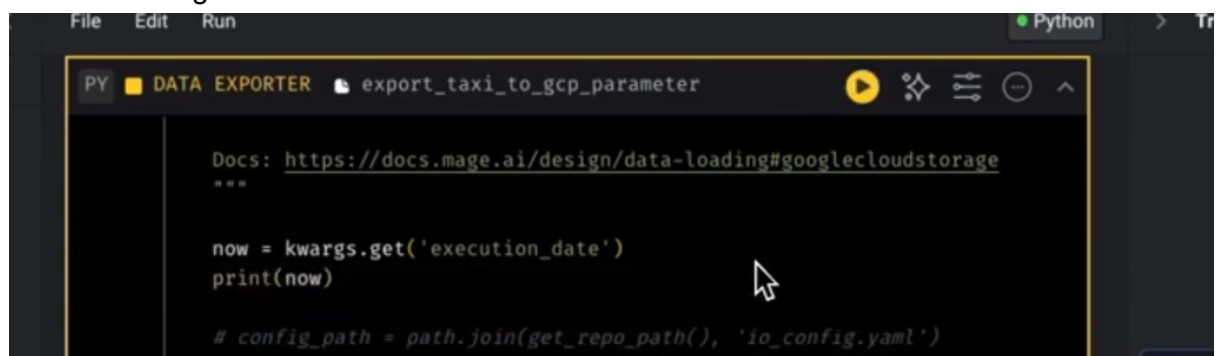
This video show how to create a different file for each day

1. Clone the previous pipeline
2. If you're going to edit a block create a new one and copy the code, if you edit an existing block it will affect all the pipelines where it's in use
3. This saves the runtime variables



```
6
7 if 'data_exporter' not in globals():
8     from mage_ai.data_preparation.decorators import data_exporter
9
10
11 @data_exporter
12 def export_data_to_google_cloud_storage(df: DataFrame, **kwargs) → None:
13     """
14     Template for exporting data to a Google Cloud Storage bucket.
15     Specify your configuration settings in 'io_config.yaml'.
16
17     Docs: https://docs.mage.ai/design/data-loading#googlecloudstorage
18     """
19     config_path = path.join(get_repo_path(), 'io_config.yaml')
20     config_profile = 'default'
21
22     bucket_name = 'mage-zoomcamp-mattpalmer'
23     object_key = 'nyc-yellow-taxi-trips.parquet'
24
25     GoogleCloudStorage.with_config(ConfigFileLoader(config_path, config_profile),
26                                   df,
27                                   bucket_name,
28                                   object_key,
29                                   )
30
```

4. This is how to get it

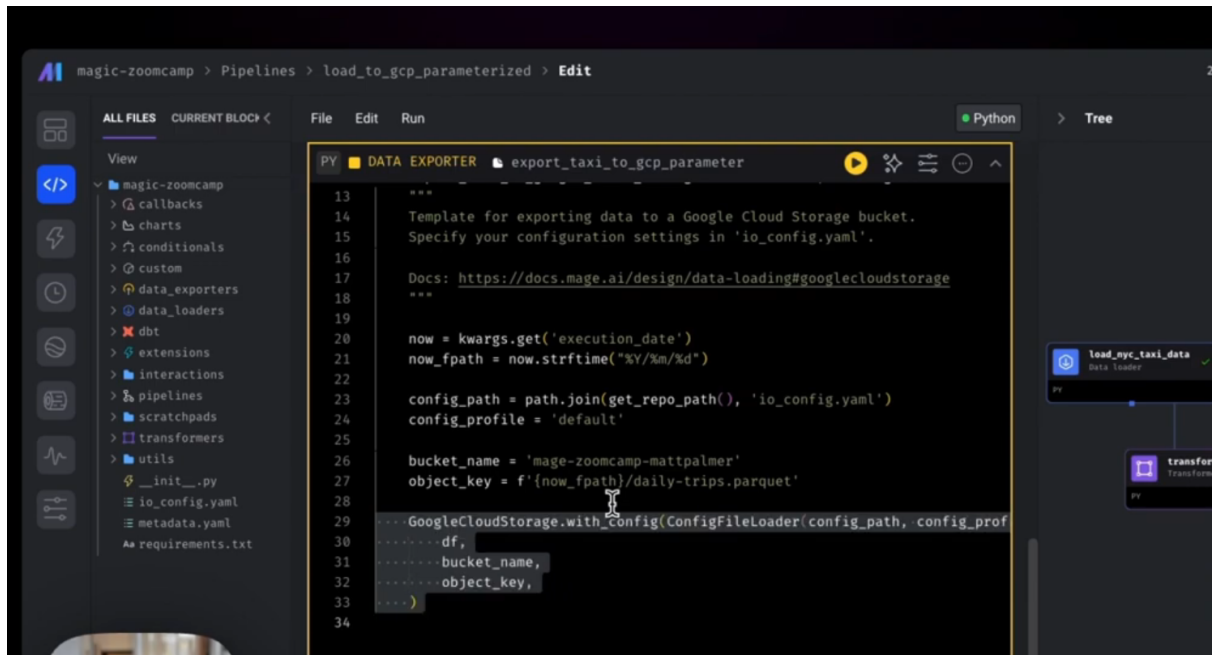


```
Docs: https://docs.mage.ai/design/data-loading#googlecloudstorage
"""

now = kwargs.get('execution_date')
print(now)

# config_path = path.join(get_repo_path(), 'io_config.yaml')
```

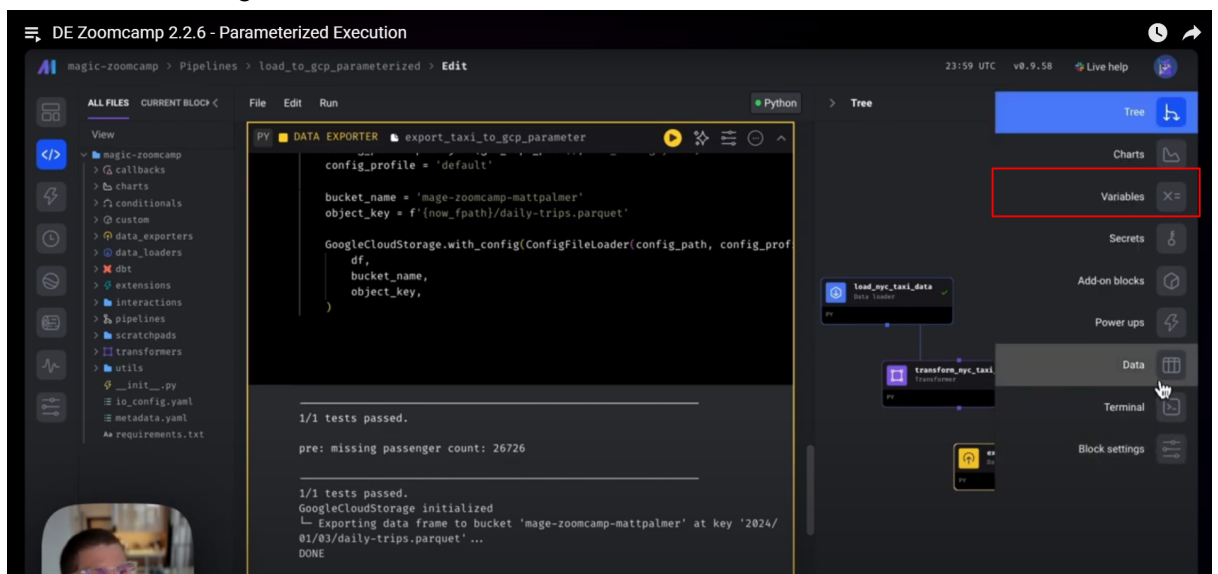
5. Create a file path for each file

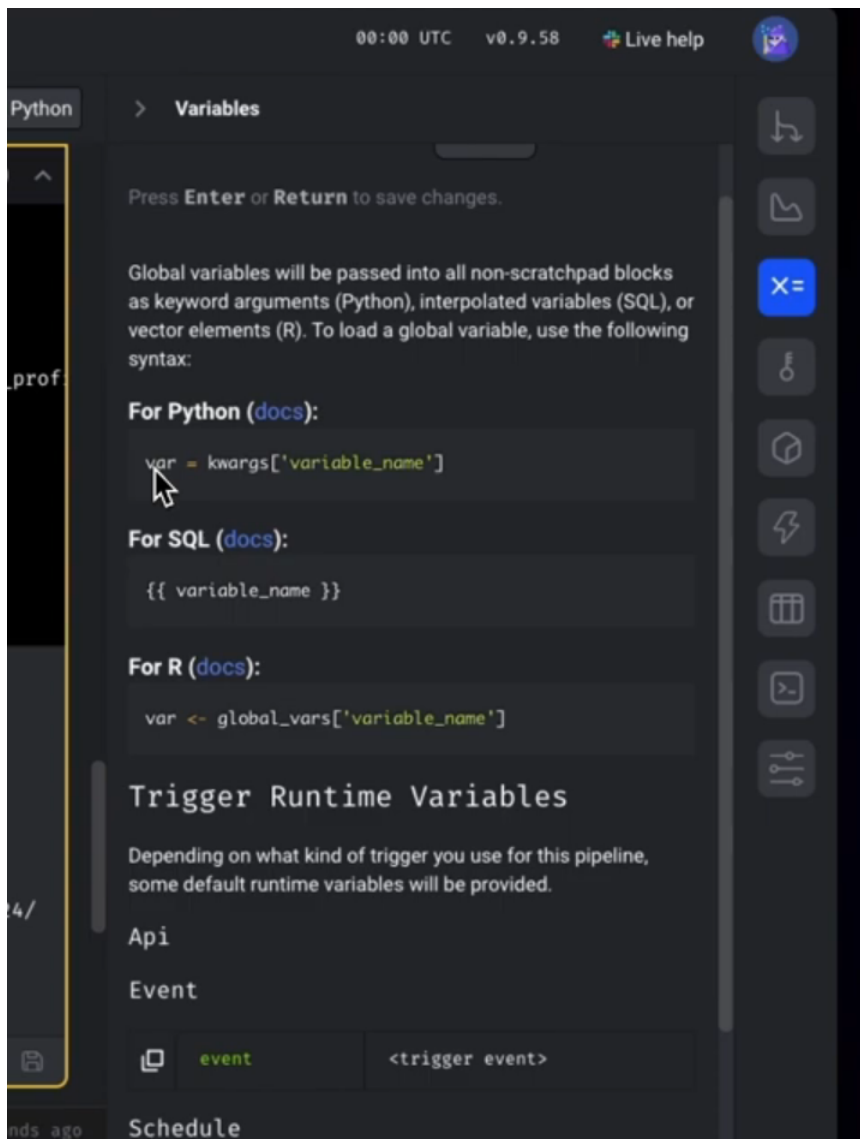


The screenshot shows the Databricks IDE interface. The left sidebar displays the file explorer with a tree view of the project structure. The main editor window shows a Python script named 'export_taxi_to_gcp_parameter' within a 'DATA EXPORTER' block. The script is a template for exporting data to a Google Cloud Storage bucket. It includes comments, a docstring with a link to a design document, and code for setting up the export path, bucket name, and object key. The script uses the 'GoogleCloudStorage.with_config' method to upload a DataFrame to the specified bucket.

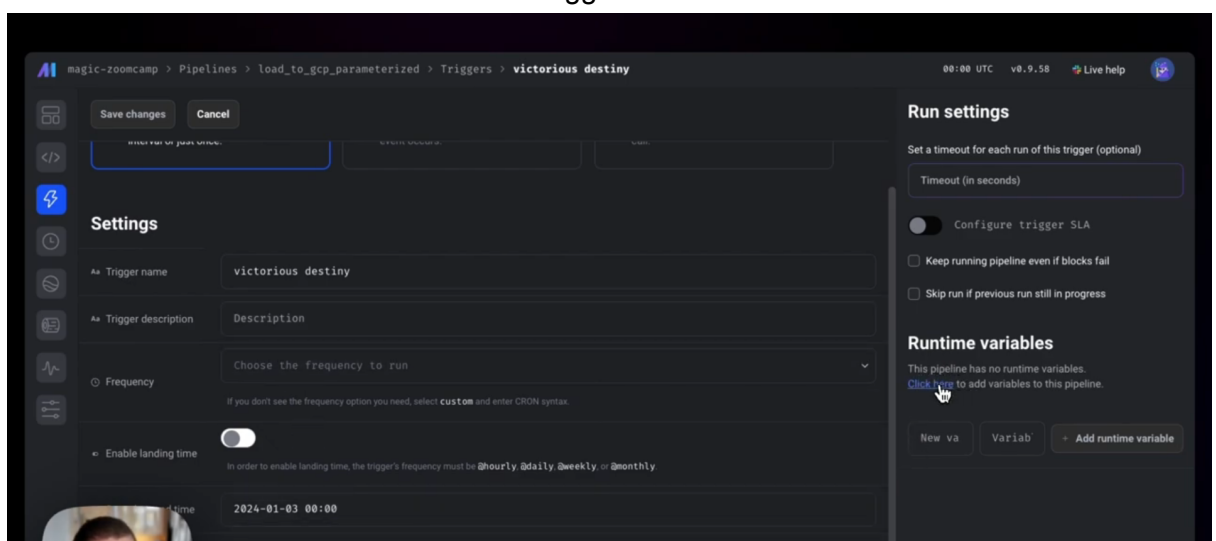
```
13  """
14  Template for exporting data to a Google Cloud Storage bucket.
15  Specify your configuration settings in 'io_config.yaml'.
16
17  Docs: https://docs.mage.ai/design/data-loading#googlecloudstorage
18  """
19
20  now = kwargs.get('execution_date')
21  now_fpath = now.strftime("%Y/%m/%d")
22
23  config_path = path.join(get_repo_path(), 'io_config.yaml')
24  config_profile = 'default'
25
26  bucket_name = 'mage-zoomcamp-mattpalmer'
27  object_key = f'{now_fpath}/daily-trips.parquet'
28
29  GoogleCloudStorage.with_config(ConfigFileLoader(config_path, config_prof
30  ..... df,
31  ..... bucket_name,
32  ..... object_key,
33  ..... )
34
```

6. Here's how to set global variables





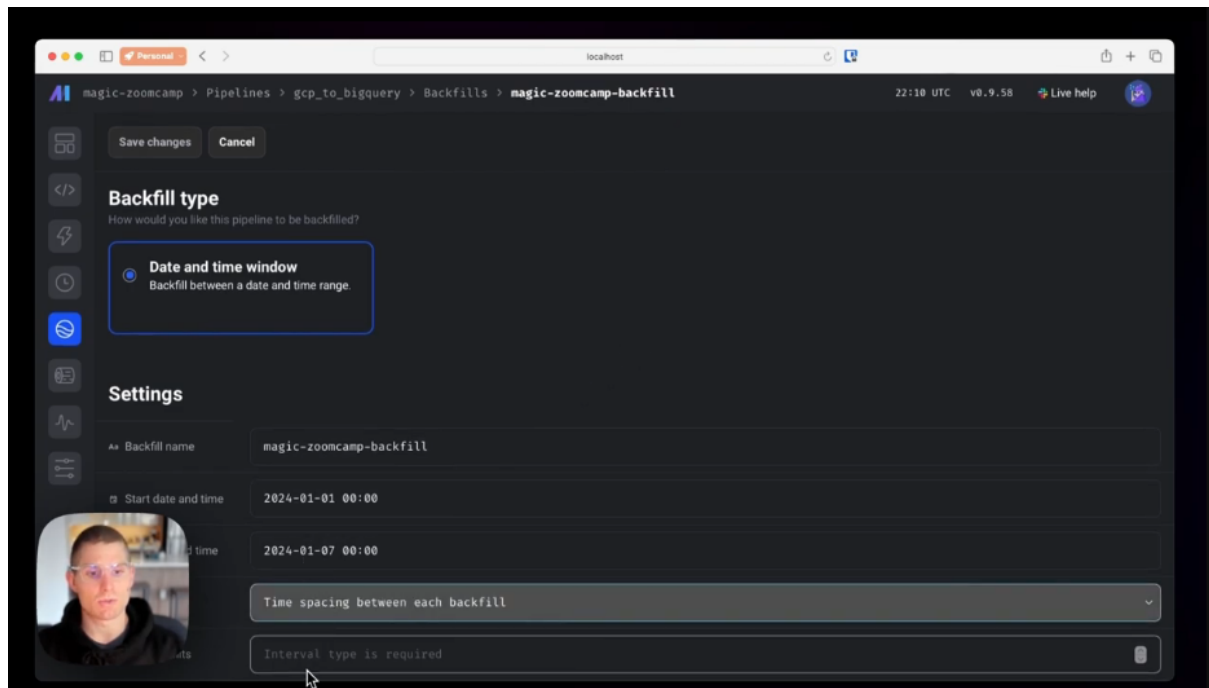
- 7.
8. We can also add runtime variables in the trigger



[DE Zoomcamp 2.2.6 - Backfills](#) [\(youtube.com\)](#)

Backfill is to restore missing data for example

1. Click any pipeline
2. Go to backfills tab
3. Select the data frame of the backfill (last day is inclusive)



Deploy MAGE to Google Cloud

Videos:

-  DE Zoomcamp 2.2.7 - Deployment Prerequisites
-  DE Zoomcamp 2.2.7 - Google Cloud Permissions
-  DE Zoomcamp 2.2.7 - Deploying to Google Cloud Part 1
-  DE Zoomcamp 2.2.7 - Deploying to Google Cloud Part 2

1. install terraform
2. install gcloud cli [Install the gcloud CLI | Google Cloud CLI Documentation](#)

3. Change gcloud permissions

Edit access to "terraform-demo"

Principal ⓘ
mage-zoomcamp-mariogo243@thinking-glass-412101.iam.gserviceaccount.com

Project
terraform-demo

Summary of changes
Role removed
Owner
Roles added
Artifact Registry Reader
Artifact Registry Writer
Cloud Run Developer
Cloud SQL Admin
Service Account Token C

Assign roles
Roles are composed of sets of permissions and determine what the principal can do with this resource. [Learn more](#) ⓘ

Role
Artifact Registry Reader
Access to read repository items.

IAM condition (optional) ⓘ
[+ ADD IAM CONDITION](#)

✕

Role
Artifact Registry Writer
Access to read and write repository items.

IAM condition (optional) ⓘ
[+ ADD IAM CONDITION](#)

✕

Role
Cloud Run Developer
Read and write access to all Cloud Run resources.

IAM condition (optional) ⓘ
[+ ADD IAM CONDITION](#)

✕

Role
Cloud SQL Admin
Full control of Cloud SQL resources.

IAM condition (optional) ⓘ
[+ ADD IAM CONDITION](#)

✕

Role
Service Account Token Creator
Impersonate service accounts (create OAuth2 access tokens, sign blobs or JWTs, etc).

IAM condition (optional) ⓘ
[+ ADD IAM CONDITION](#)

✕

[+ ADD ANOTHER ROLE](#)

SAVE

TEST CHANGES ⓘ

CANCEL

4. Git clone [mage-ai/mage-ai-terraform-templates: Terraform templates for deploying mage-ai to AWS, GCP and Azure \(github.com\)](#)
5. cd into gcp folder
6. gcloud auth application-default login
7. Enable google cloud firestore api
8. terraform init, terraform plan and terraform apply
9. Go to google cloud run, allow all Ips and access the new service URL

d