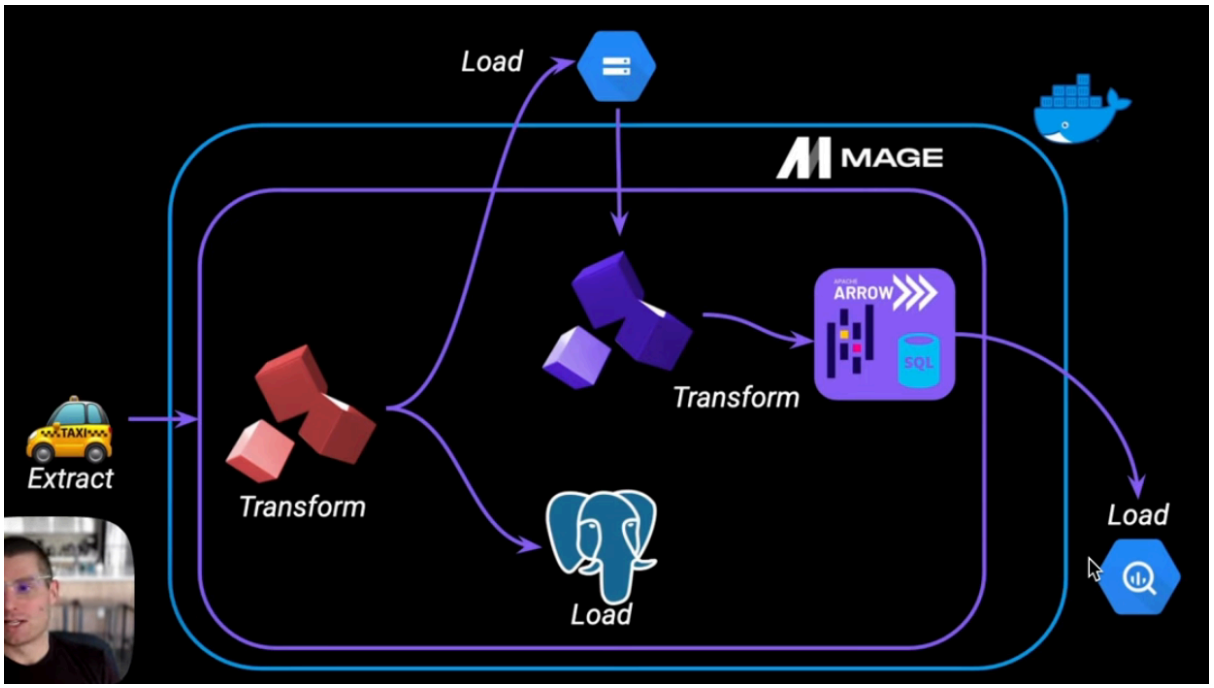
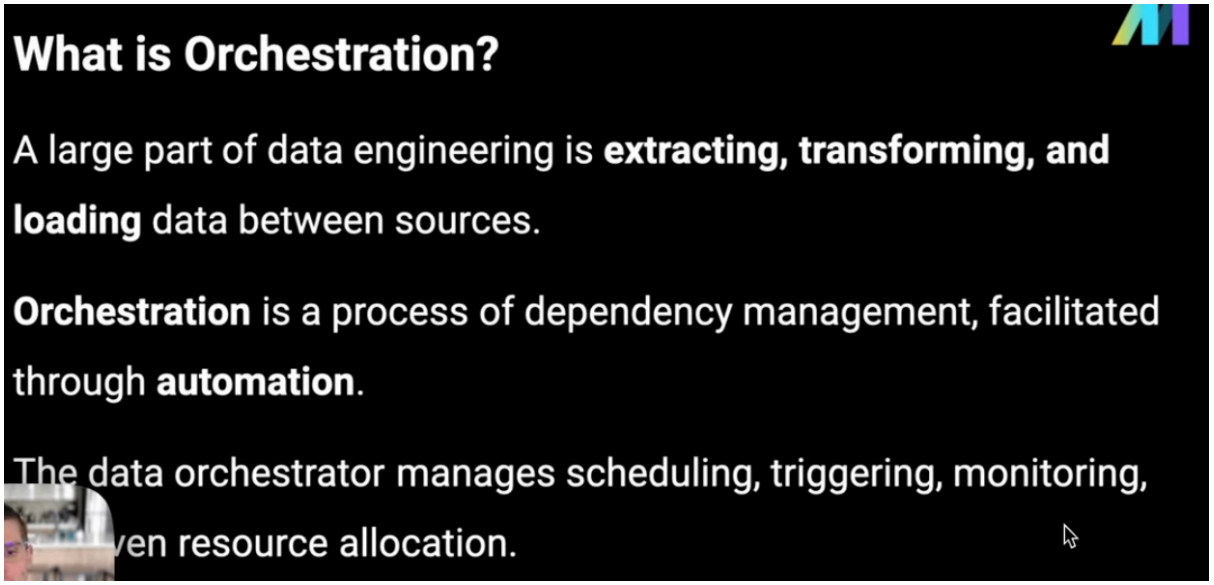
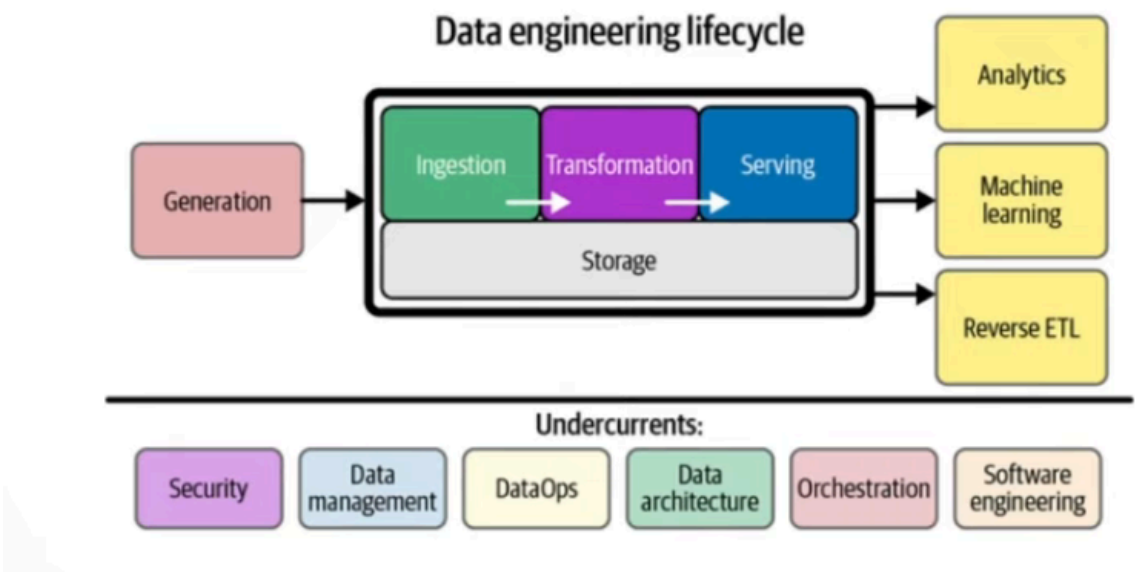
[(32) DE Zoomcamp 2.2.1 - What is Orchestration? - YouTube]

Final goal





Step = task = block
Workflog = dag = directed acyclic graphs



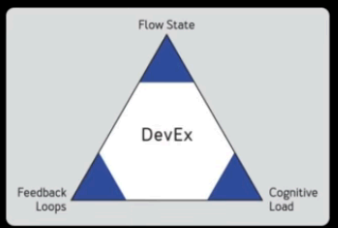
A good orchestrator handles...

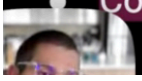
- Workflow management
- Automation
- Error handling
- Recovery
- Monitoring, alerting
- Resource optimization
- Observability
- Debugging
- Compliance/Auditing

A good orchestrator prioritizes....

The *developer experience*

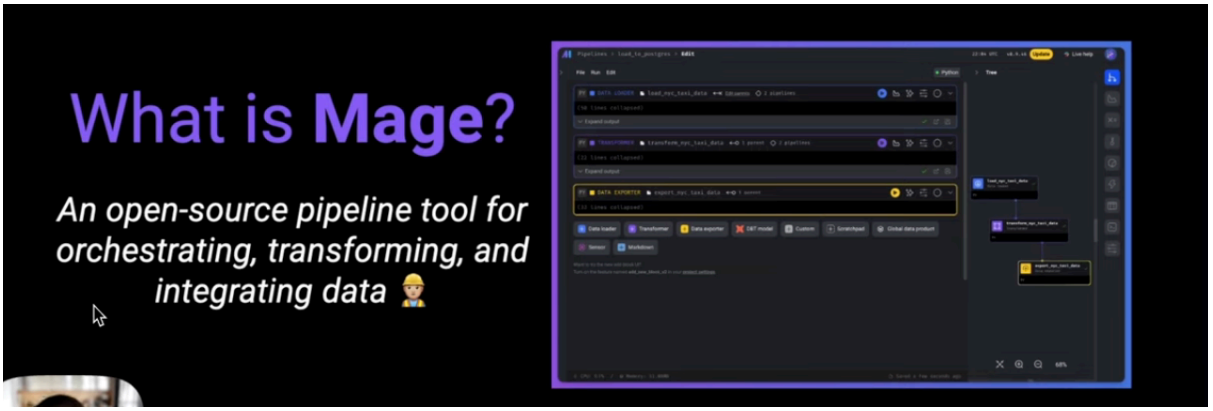
- Flow state 🌊
  - *"I need to switch between 7 tools/services."*
- Feedback Loops 🔁
  - *"I spent 5 hours locally testing this DAG."*
- Cognitive Load 🤯

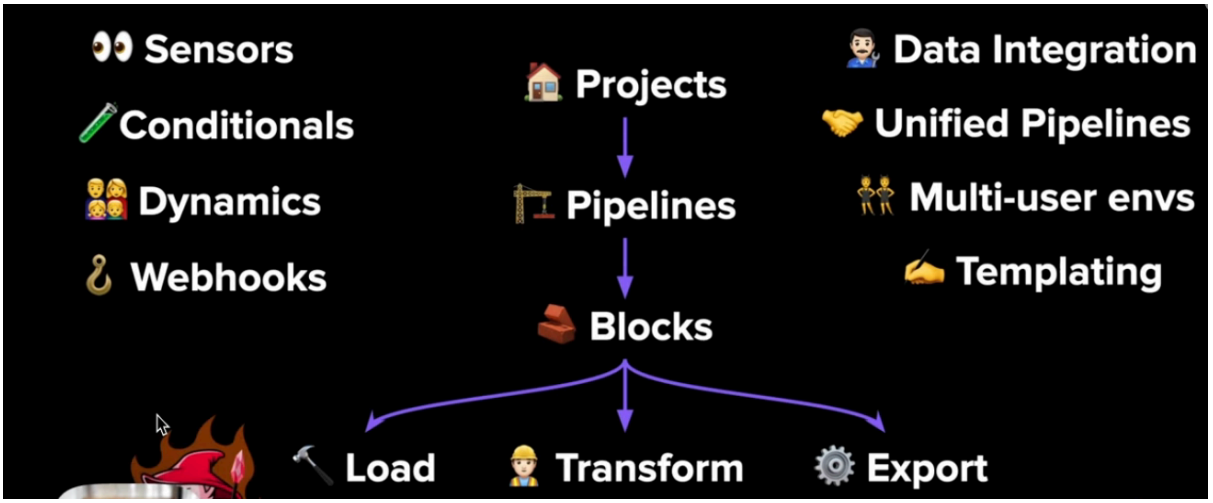


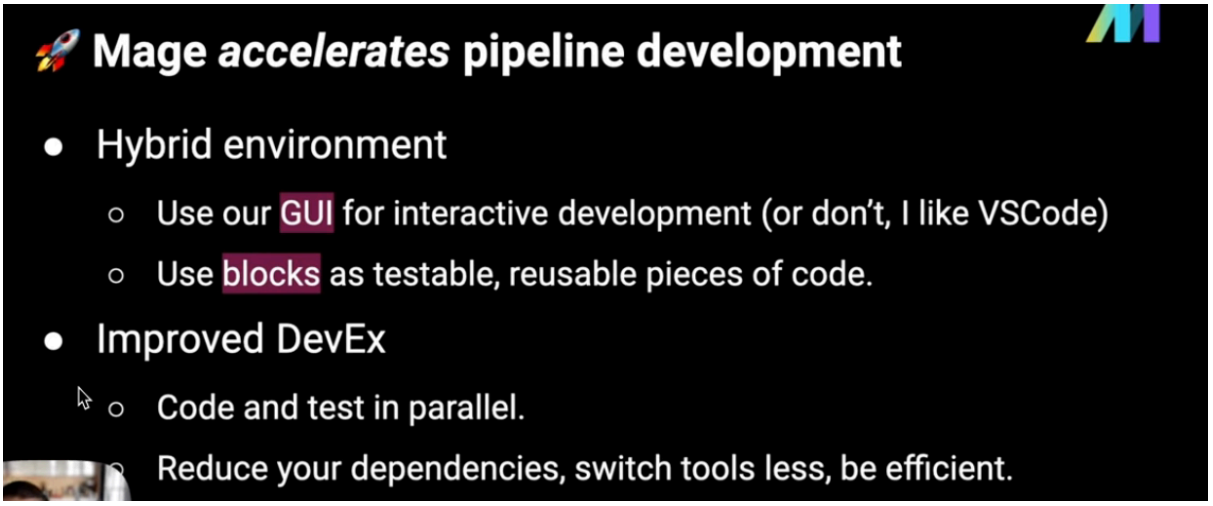


How much do you need to know to do your job?

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



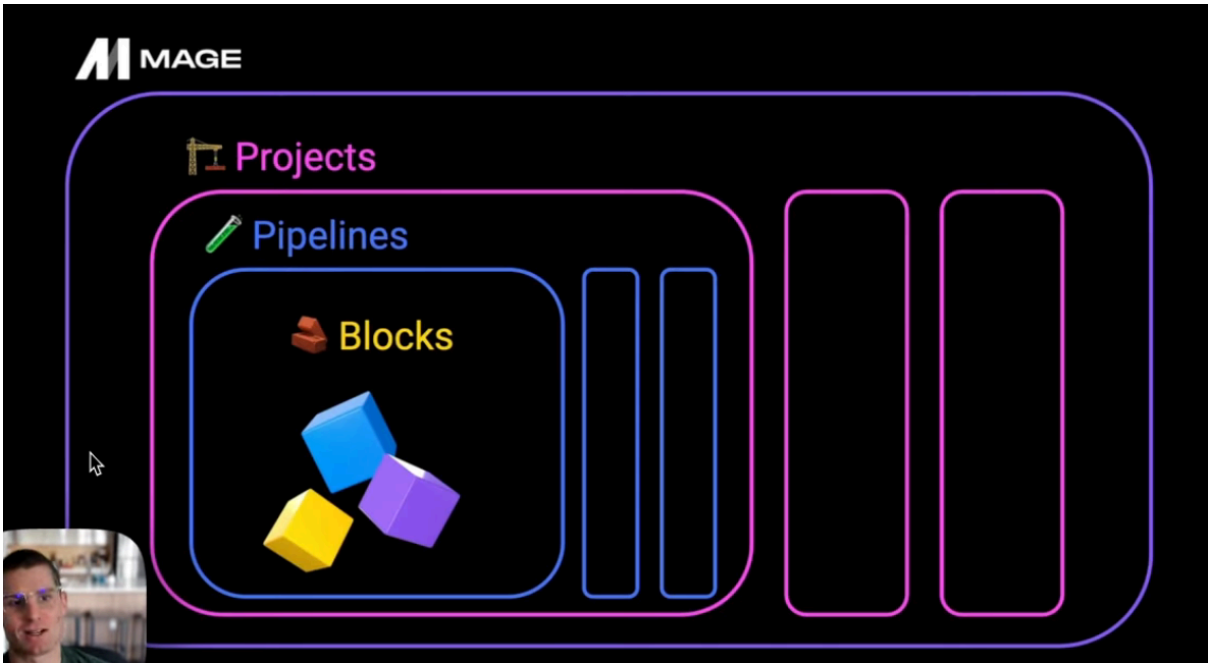




Engineering best-practices built-in

- 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)

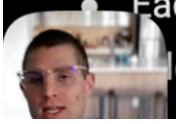


Projects

- 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

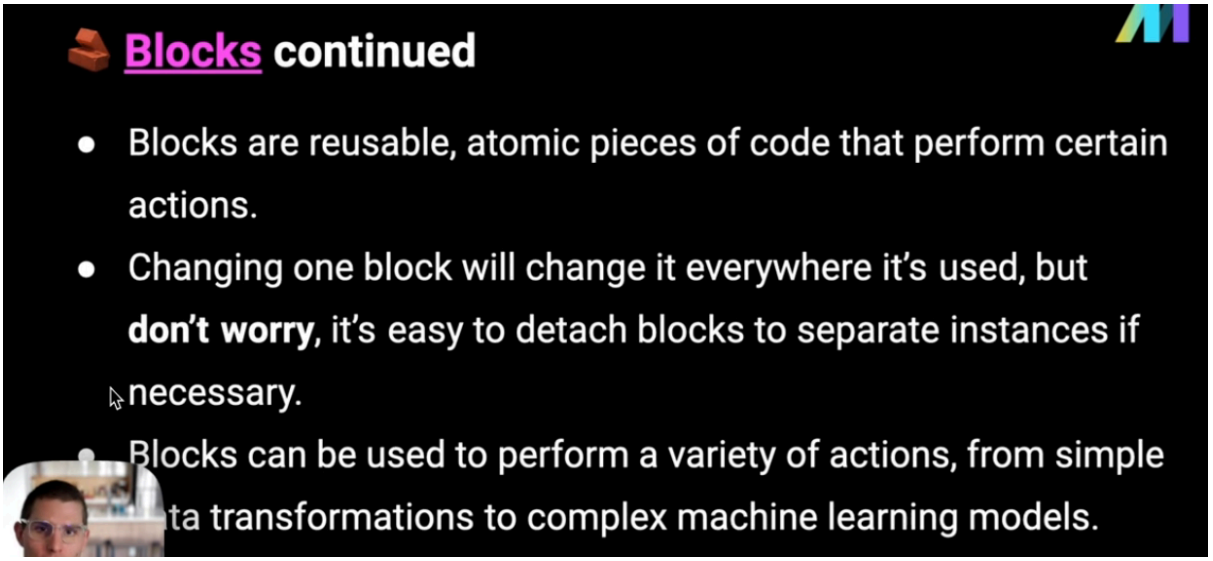
Pipelines

- 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.

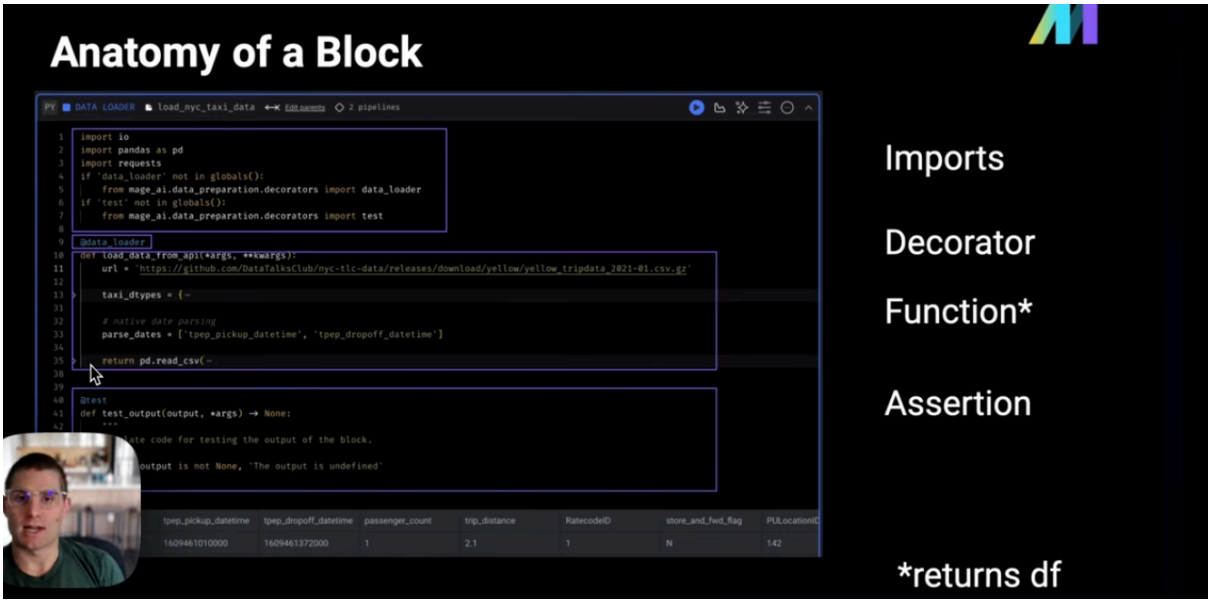


Blocks

- 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 should return dataframes



(32) DE Zoomcamp 2.2.2 - Configure Mage - YouTube

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

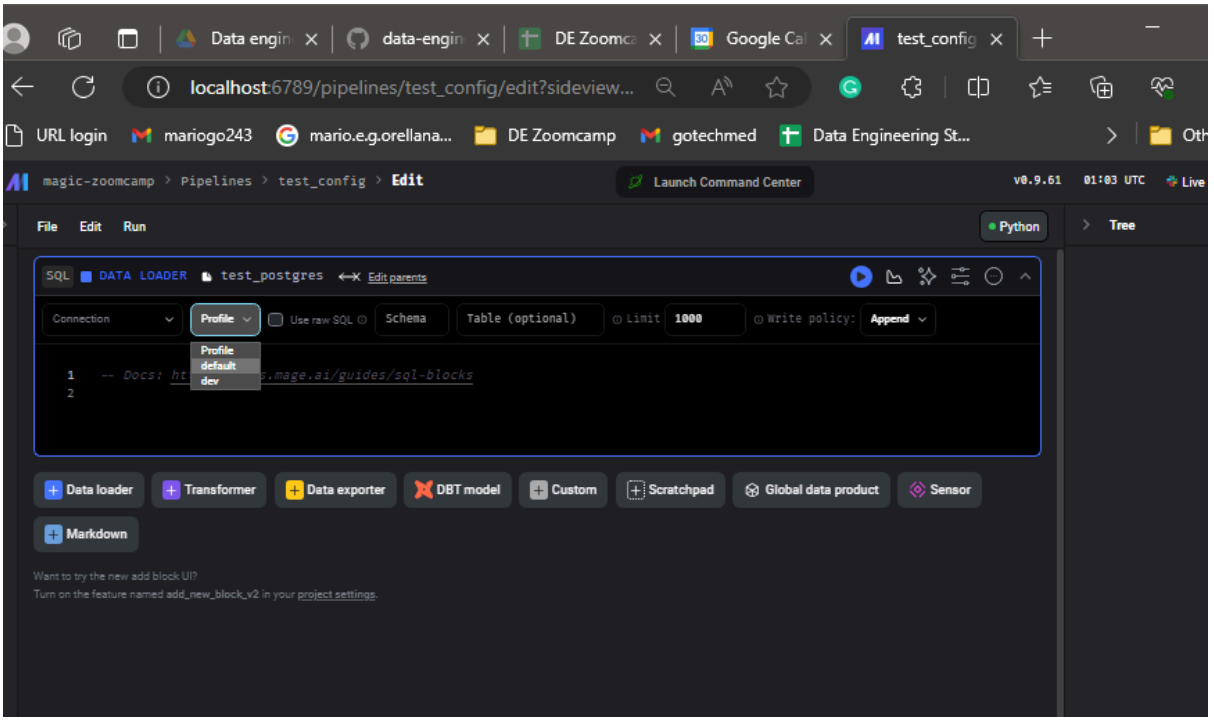
Pipelines | Mage

DE Zoomcamp 2.2.3 - Configuring Postgres (youtube.com)

1. Setgup io_config.yaml is where we manage the connection, we can define a new profile
2. Let's set up a new profila calle 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:
  # PostgresSQL
  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)

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

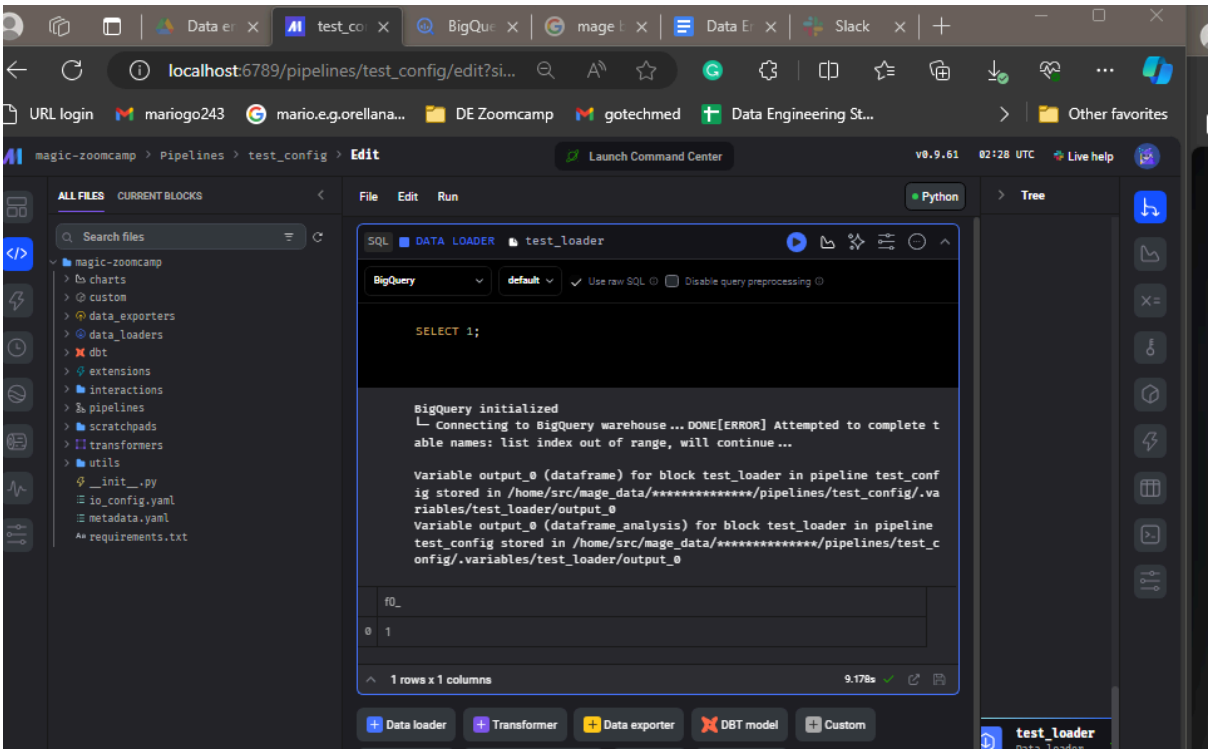
```
volumes:
  - .:/home/src/
```

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

```
GOOGLE_SERVICE_ACC_KEY:
  type: service_account
  project_id: project-id
  private_key_id: key-id
  private_key: "-----BEGIN PRIVATE KEY-----\nyour_private_key\n-----END_PRIVATE_KEY"
  client_email: your_service_account_email
  auth_uri: "https://accounts.google.com/o/oauth2/auth"
  token_uri: "https://accounts.google.com/o/oauth2/token"
  auth_provider_x509_cert_url: "https://www.googleapis.com/oauth2/v1/certs"
  client_x509_cert_url: "https://www.googleapis.com/robot/v1/metadata/x509/your_service_account_email"
```

    l. 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)

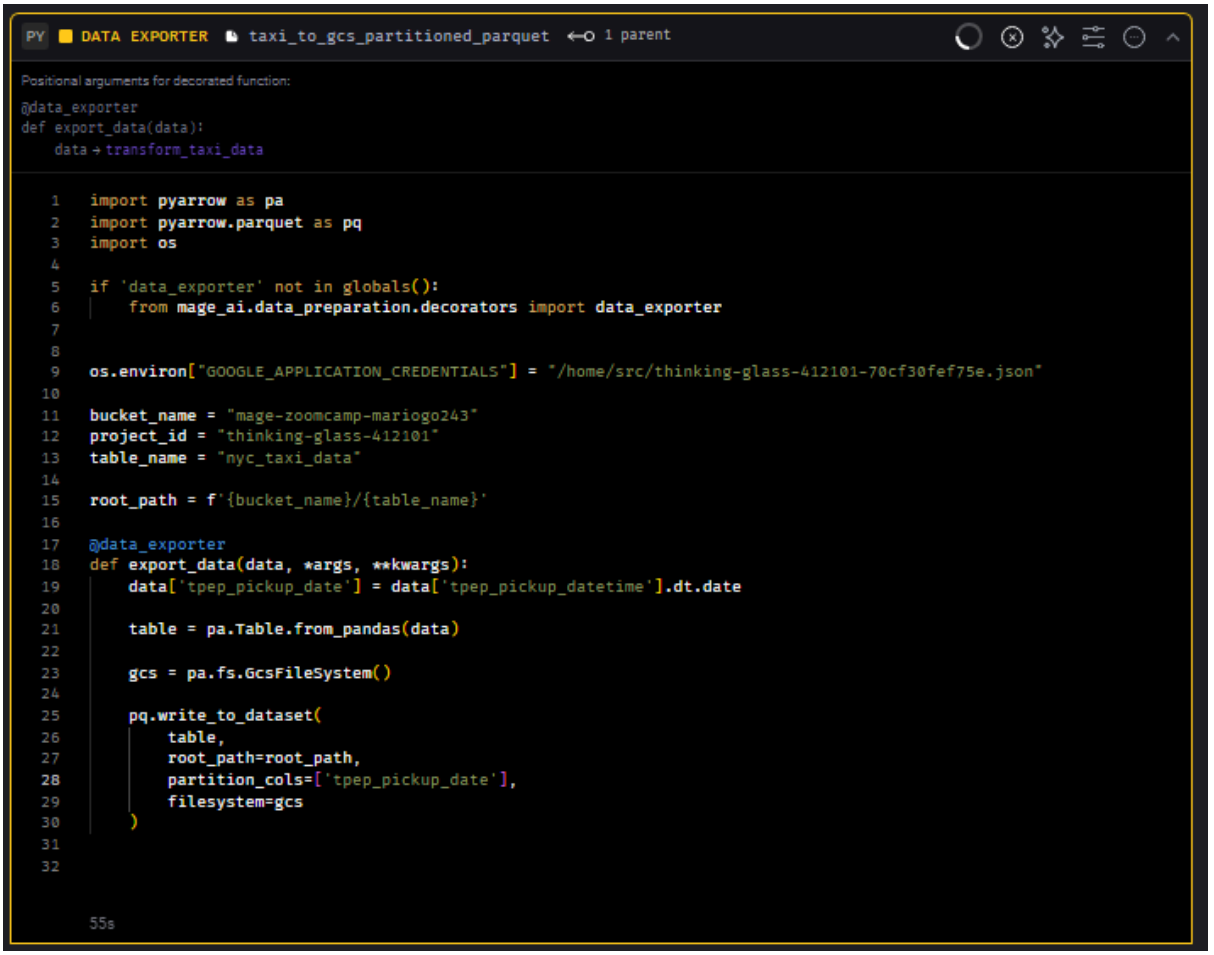
data-engineering-zoomcamp/Module 2/my_mage_files/3. Load_Data_GCS_Partitioned.py at master · MarioOrellana58/data-engineering-zoomcamp (github.com)

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

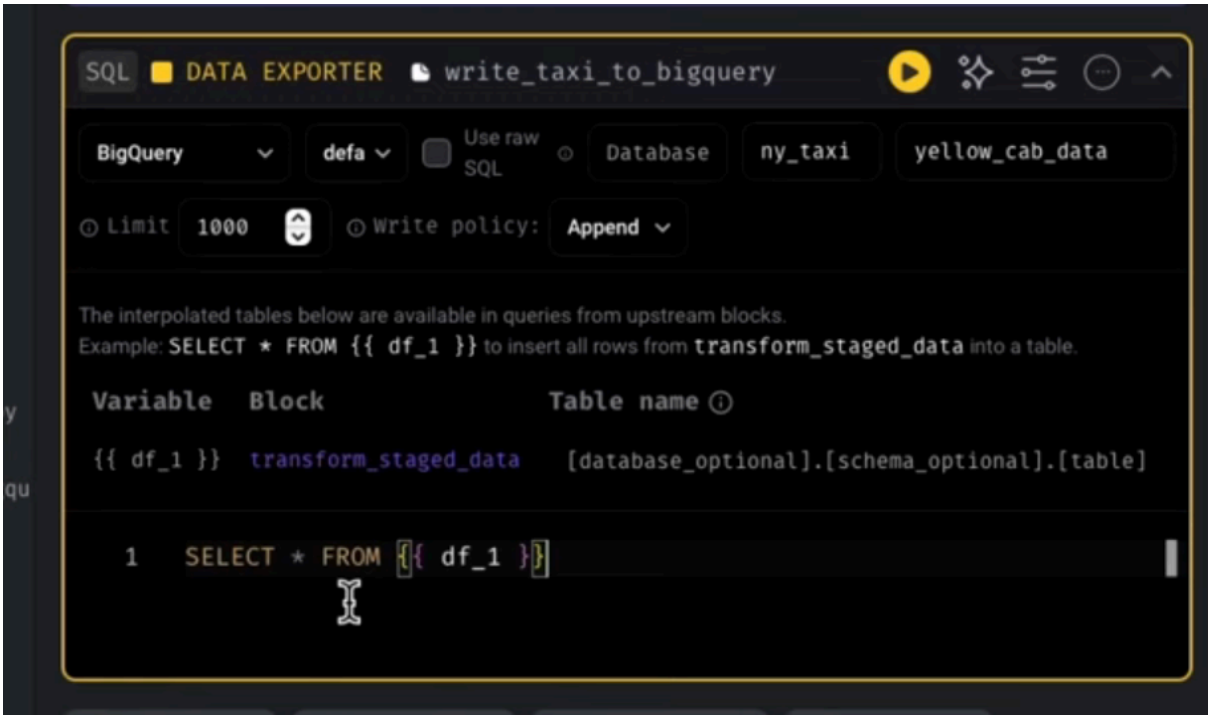
1. Add a new data exporter python generic
2. Create a new date column
3. Partition data by date




```python
import pyarrow as pa
import pyarrow.parquet as pq
import os

if 'data_exporter' not in globals():
    from mage_ai.data_preparation.decorators import data_exporter


os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "/home/src/thinking-glass-412101-70cf30fef75e.json"

bucket_name = "mage-zoomcamp-mariogo243"
project_id = "thinking-glass-412101"
table_name = "nyc_taxi_data"

root_path = f'{bucket_name}/{table_name}'

@data_exporter
def export_data(data, *args, **kwargs):
    data['tpep_pickup_date'] = data['tpep_pickup_datetime'].dt.date

    table = pa.Table.from_pandas(data)

    gcs = pa.fs.GcsFileSystem()

    pq.write_to_dataset(
        table,
        root_path=root_path,
        partition_cols=['tpep_pickup_date'],
        filesystem=gcs
    )
```

[(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



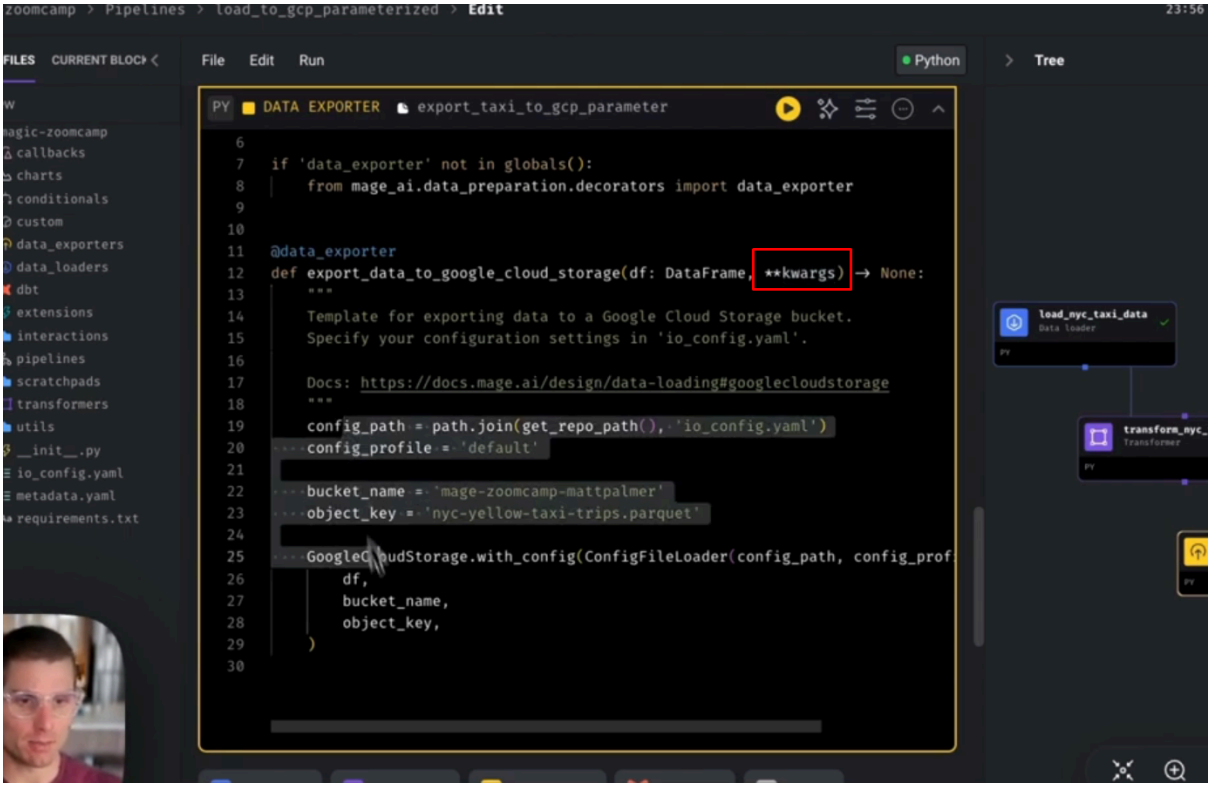

```
    dfs = []
    for object_key in object_keys:
        df =
pq.read_table(f'gs://{bucket_name}/{object_key}').to_pandas()
        dfs.append(df)

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

[DE Zoomcamp 2.2.6 - Parameterized Execution (youtube.com)]

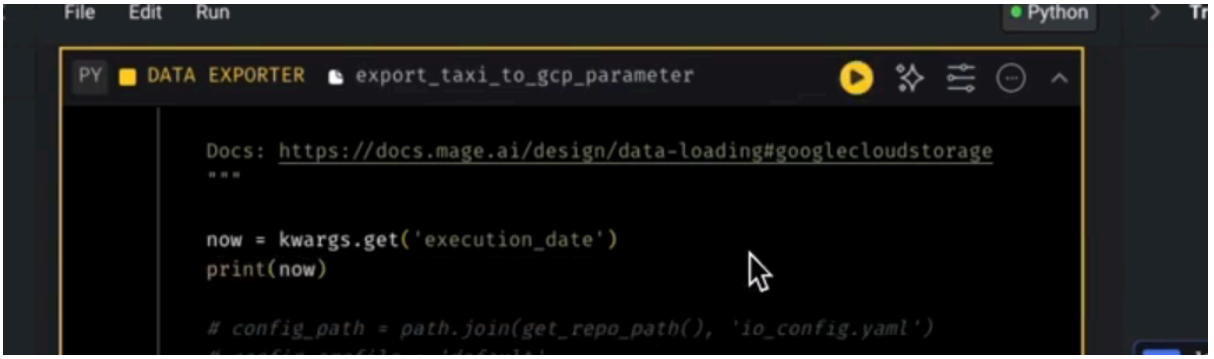
Mage has runtime variables, global variables, many variables… Look into the documentation.

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

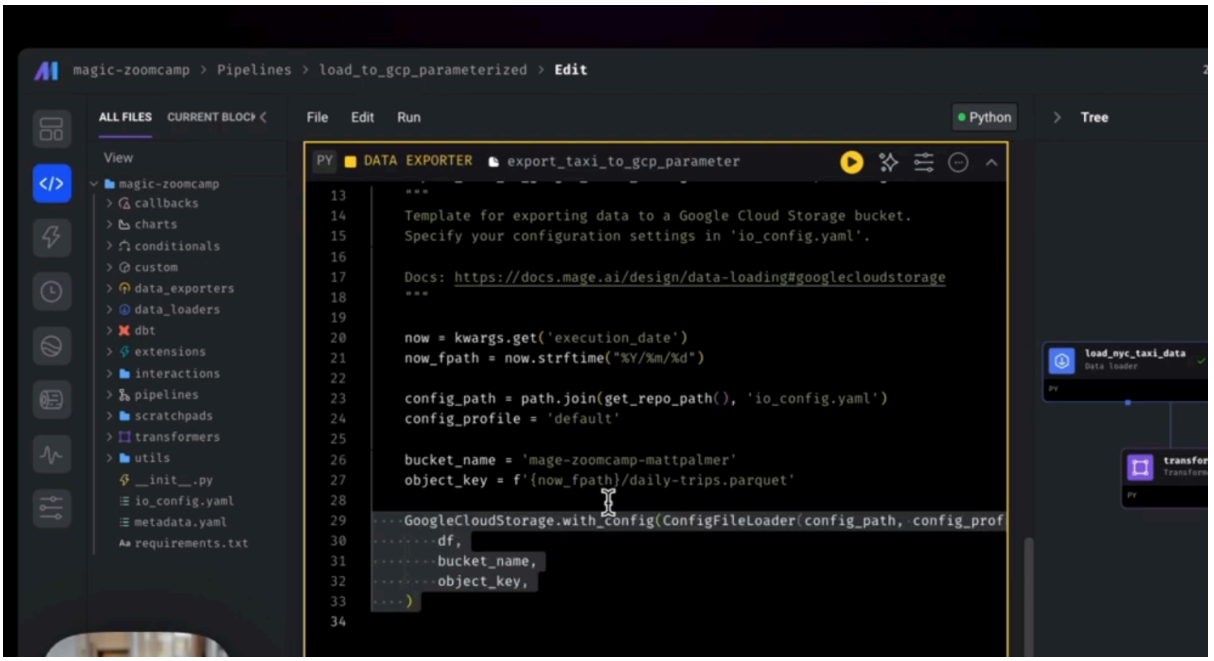
1. Clone the previous pipeline
2. If you're going to edit a block create a new one and copy the code, if you edit an existing block it will affect all the pipelines where it's in use
3. This saves the runtime variables



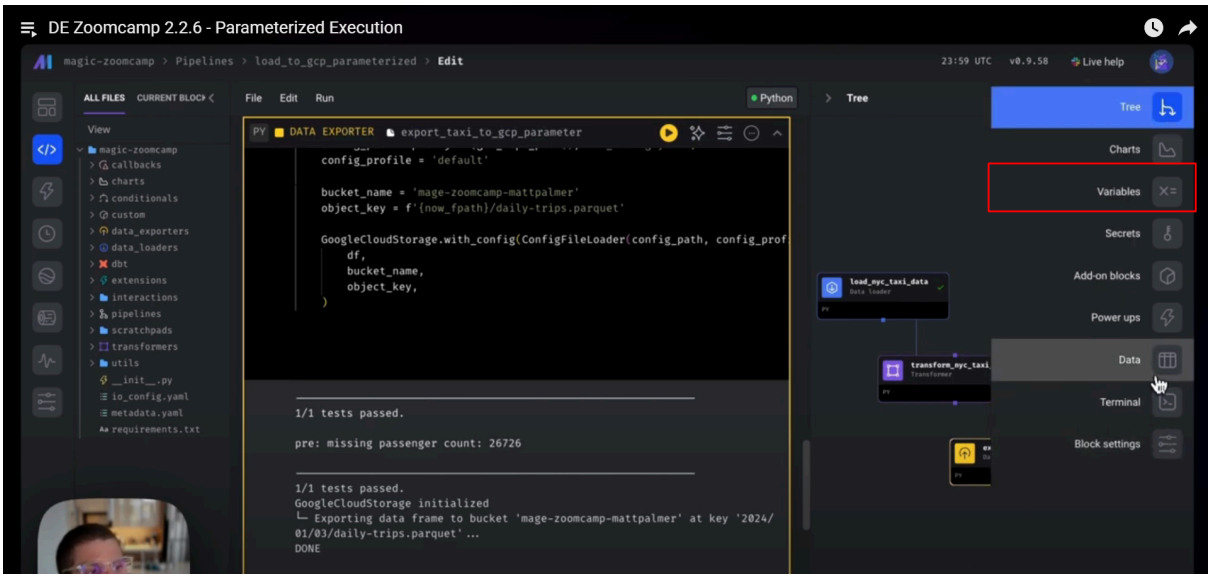
4. This is how to get it

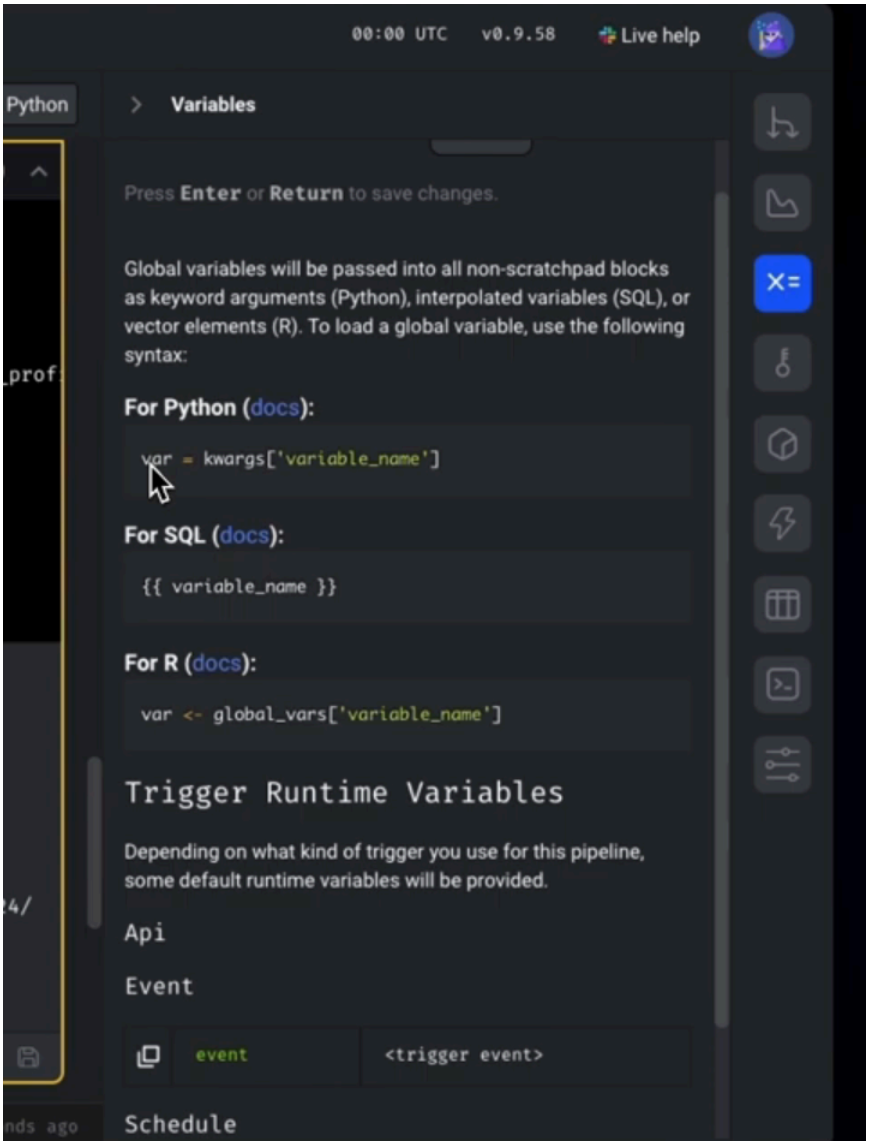


5. Create a file path for each file

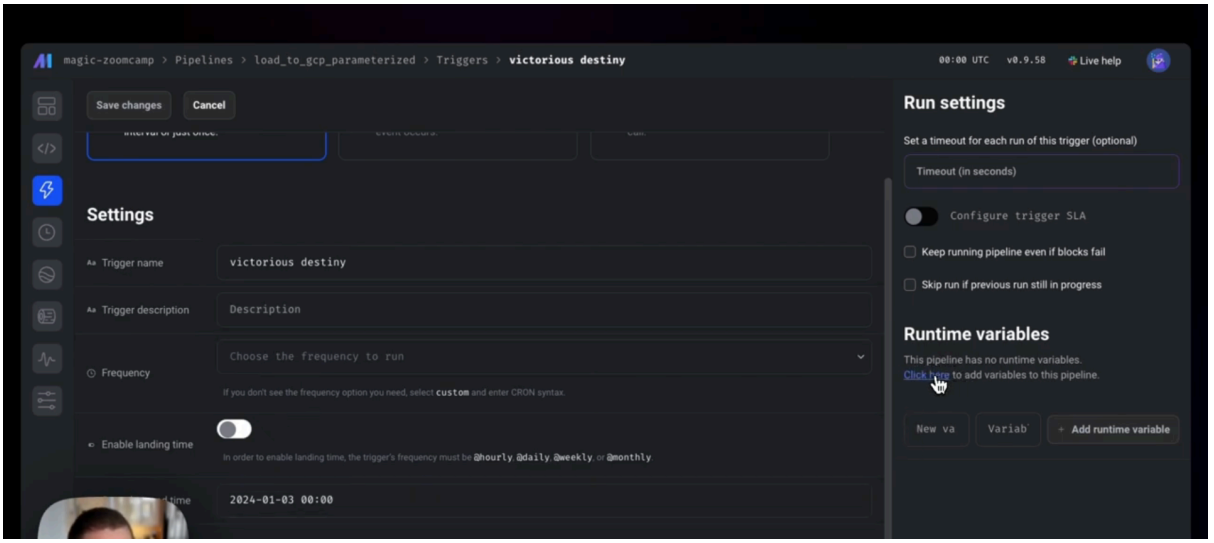


6. Here's how to set global variables



7. 

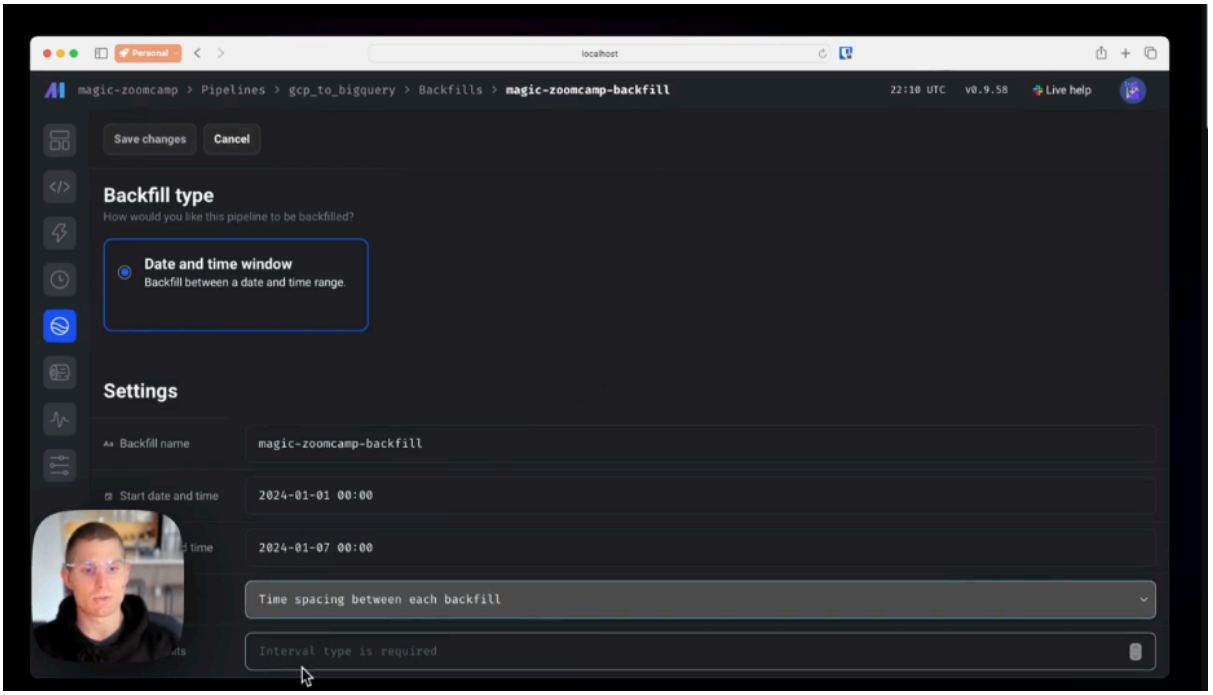
8. We can also add runtime variables in the trigger



[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)



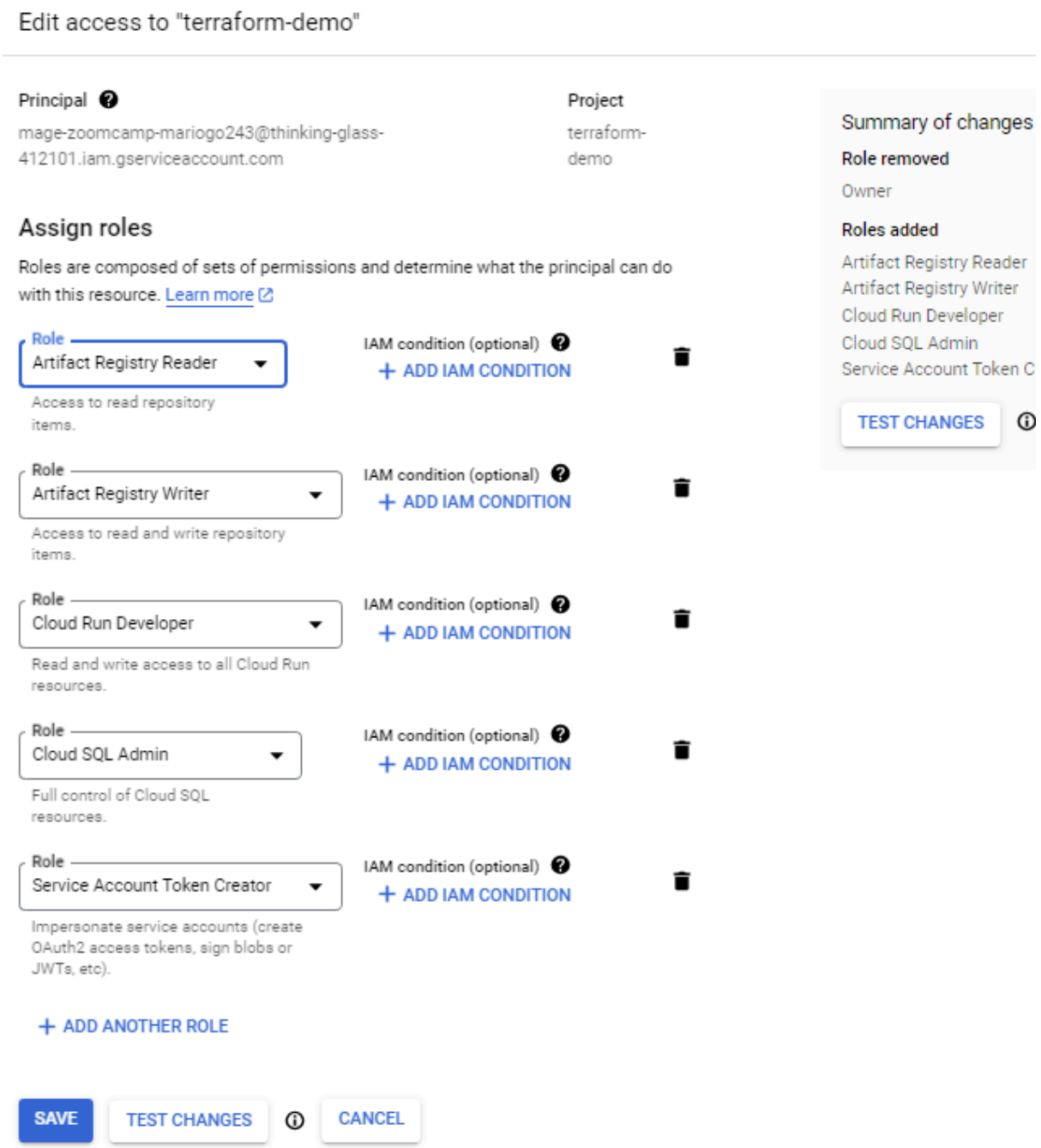
Deploy MAGE to Google Cloud

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



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