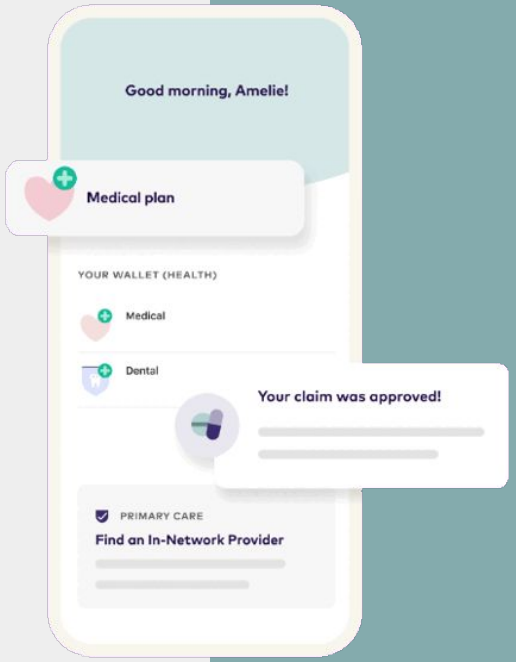
Mapping Data to FHIR

Alex Fragotsis
League Inc.
@alexfragotsis

BEAM SUMMIT
NYC 2023

Building the Leading Healthcare CX Platform Company

- Founded in 2014
- The platform used by industry leaders to **accelerate transformation**
- **Reaching over 14 million members across the healthcare ecosystem**



accenture

HIGHMARK HEALTH

Humana

SHOPPERS DRUG MART

Deloitte.

Google

Mapping Data to FHIR

(How I tricked my co-workers to use dataflow without even knowing it)

What is FHIR

Fast Healthcare Interoperability Resources

HL7® FHIR®

HTTP RESTFul

Single / Bundle

JSON / XML

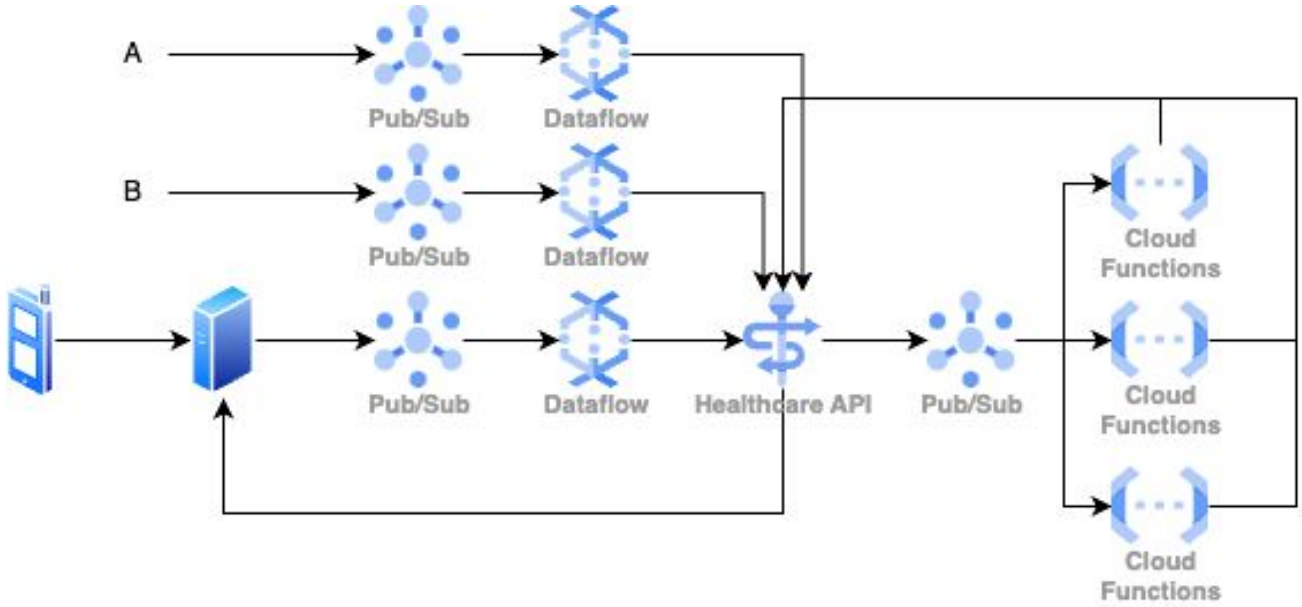
Interoperability

Modeled and Documented Resources

Patient

```
{
  "resourceType" : "Patient",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // An identifier for this patient
  "active" : <boolean>, // Whether this patient's record is in active use
  "name" : [{ HumanName }], // A name associated with the patient
  "telecom" : [{ ContactPoint }], // A contact detail for the individual
  "gender" : "<code>", // male | female | other | unknown
  "birthDate" : "<date>", // The date of birth for the individual
  // deceased[x]: Indicates if the individual is deceased or not. One of these 2:
  "deceasedBoolean" : <boolean>,
  "deceasedDateTime" : "<dateTime>",
  "address" : [{ Address }], // An address for the individual
  "maritalStatus" : { CodeableConcept }, // Marital (civil) status of a patient
  // multipleBirth[x]: Whether patient is part of a multiple birth. One of these 2:
  "multipleBirthBoolean" : <boolean>,
  "multipleBirthInteger" : <integer>,
  "photo" : [{ Attachment }], // Image of the patient
  "contact" : [{ // A contact party (e.g. guardian, partner, friend) for the patient
    "relationship" : [{ CodeableConcept }], // The kind of relationship
    "name" : { HumanName }, // I A name associated with the contact person
    "telecom" : [{ ContactPoint }], // I A contact detail for the person
    "address" : { Address }, // I Address for the contact person
    "gender" : "<code>", // male | female | other | unknown
    "organization" : { Reference(Organization) }, // I Organization that is associated with the contact
    "period" : { Period } // The period during which this contact person or organization is valid to be contacted re
lating to this patient
  }],
  "communication" : [{ // A language which may be used to communicate with the patient about his or her health
    "language" : { CodeableConcept }, // R!  The language which can be used to communicate with the patient about hi
s or her health
    "preferred" : <boolean> // Language preference indicator
  }],
  "generalPractitioner" : [{ Reference(Organization|Practitioner|
   PractitionerRole) }], // Patient's nominated primary care provider
  "managingOrganization" : { Reference(Organization) }, // Organization that is the custodian of the patient record
  "link" : [{ // Link to a Patient or RelatedPerson resource that concerns the same actual individual
    "other" : { Reference(Patient|RelatedPerson) }, // R!  The other patient or related person resource that the lin
k refers to
    "type" : "<code>" // R!  replaced-by | replaces | refer | seealso
  }]
}
```

The problem


A
Pub/Sub
Dataflow
B
Pub/Sub
Dataflow
Pub/Sub
Dataflow
Healthcare API
Pub/Sub
Cloud Functions
Cloud Functions
Cloud Functions

PubSub to CHAPI
(Cloud Healthcare API)

Data Eng + Product Team

Understand the data

Map

Write a “mapper“ dataflow job in Python

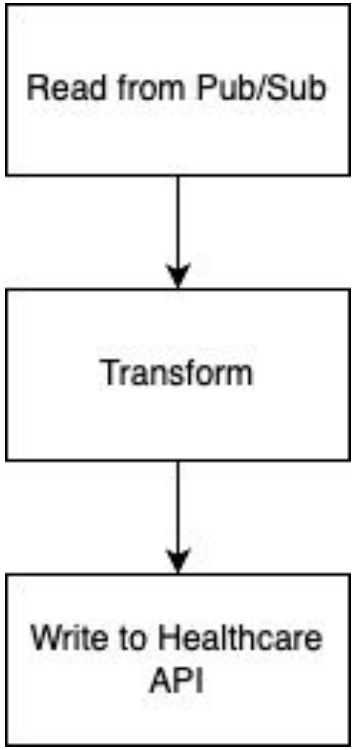
Deploy

Build CI/CD for test and prod

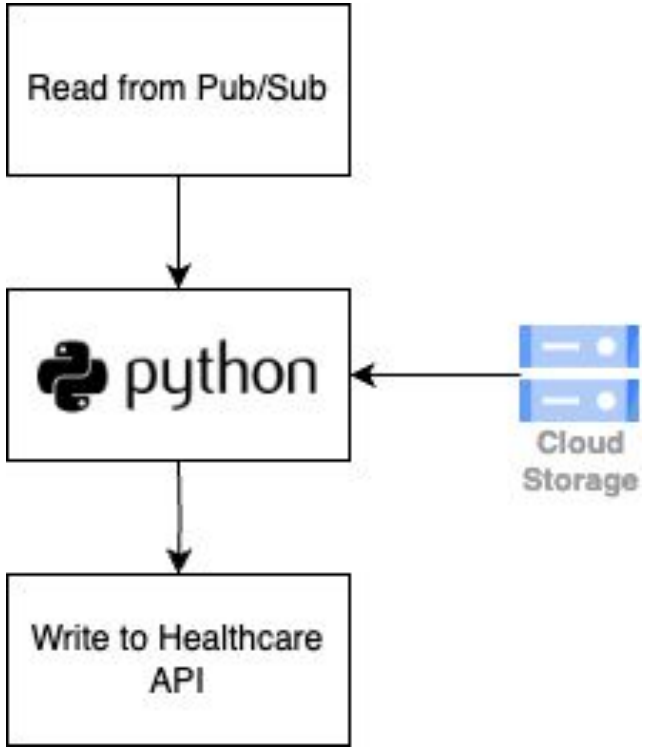
The solution

PubSub to CHAPI

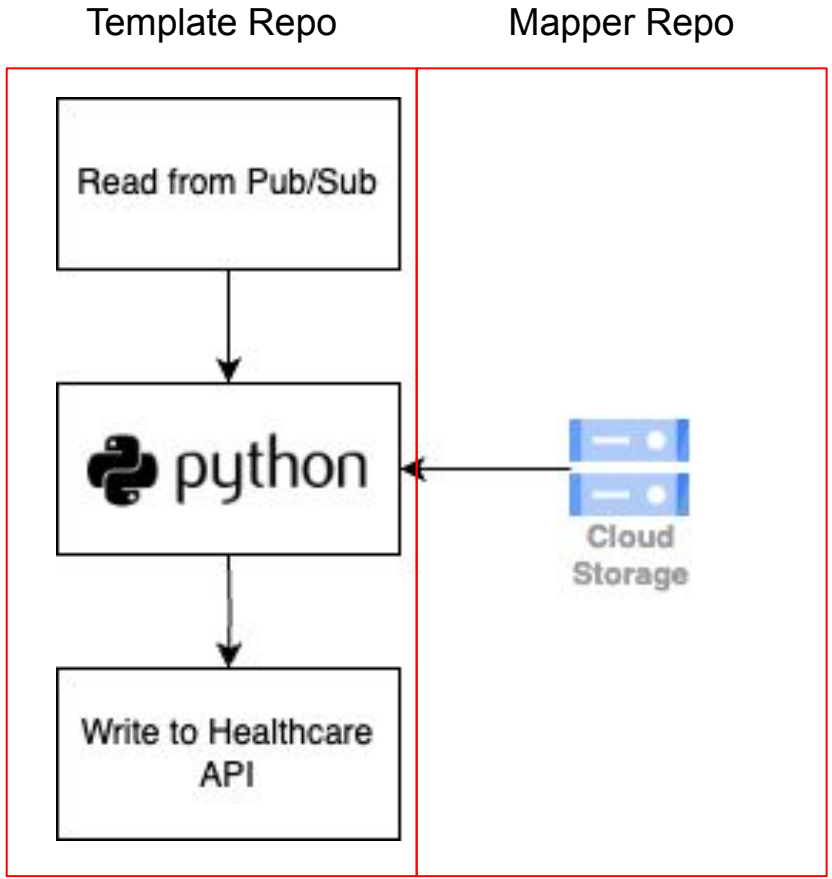
(Cloud Healthcare API)



PubSub to CHAPI



PubSub to CHAPI



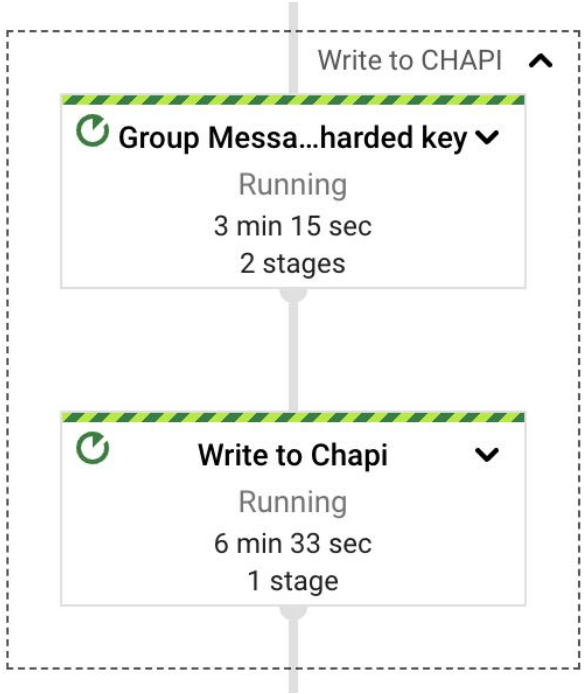
Loading Python UDF

```
def import_code(self, code, name):
    # create blank module
    module = types.ModuleType(name)
    # populate the module with code
    exec(code, module.__dict__)
    return module

def check_inputs(self, code):...

def setup(self):
    if not self.udf_module:

        matches = re.match("gs://(.*?)/(.*.py)", self.input_python_udf_gcs_path)
        if not matches:
            raise Exception("Invalid Python UDF path")

        bucket_name, object_name = matches.groups()
        logging.info(f"Downloading {bucket_name} , {object_name}")
        # Download udf from gcs
        client = Client(project=self.project)
        bucket = client.get_bucket(bucket_name)
        blob = bucket.get_blob(object_name)
        self.code = blob.download_as_string().decode("utf-8")

        forbidden_lib = self.check_inputs(self.code)
        if forbidden_lib:
            raise Exception(f"UDF uses forbidden import {forbidden_lib}")

        self.udf_module = self.import_code(self.code, 'udf_main')
```

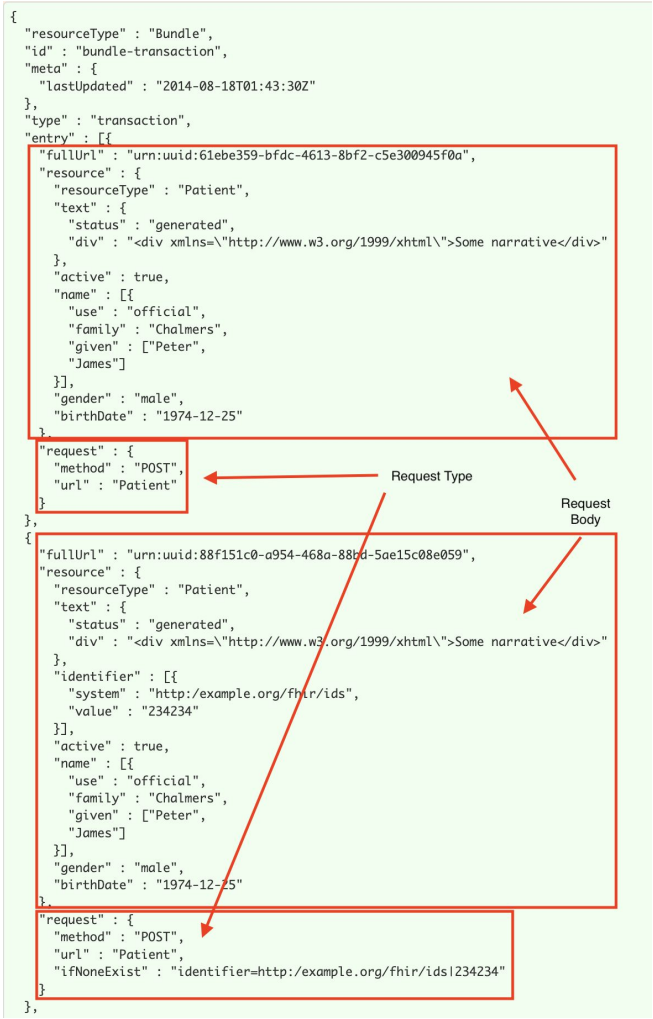
Processing messages

```
def process(self, element):
    try:
        result = self.udf_module.udf_main(element, self.settings)
        # In order to support functions that use "return" and "yield"
        if isinstance(result, Generator):
            yield from result
        elif result:  # Check if the function returned anything
            yield result
    except Exception as e:
        logging.info(f"Exception = {e}")
        yield pvalue.TaggedOutput(self.TAG_EXCEPTION, ...)
```

Write to CHAPI

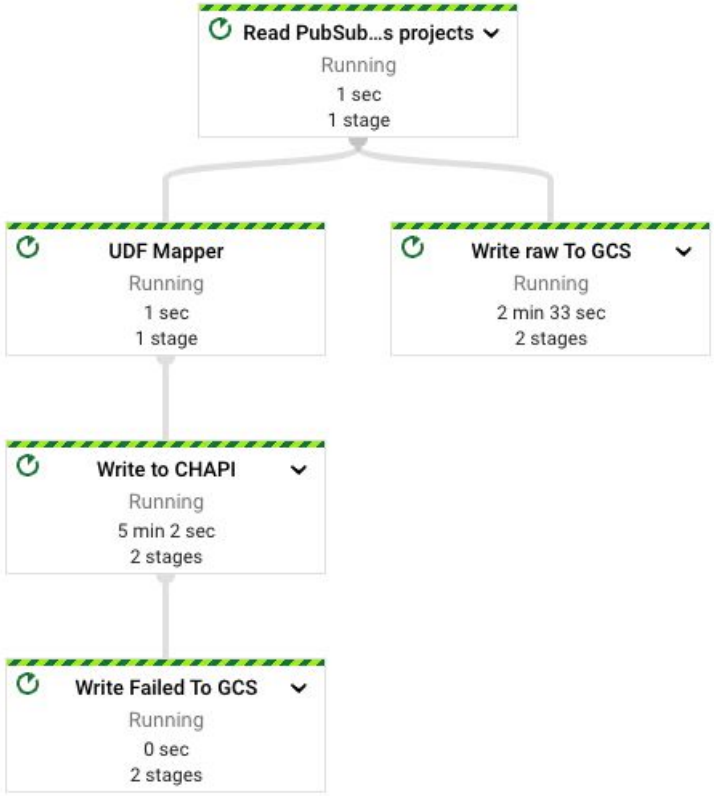


Write Bundles

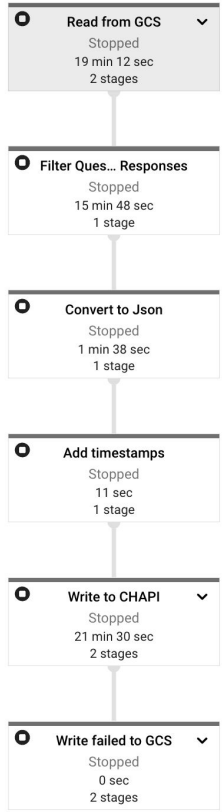


Batch Mode

Real-Time



Batch



Beam Summit 2022 - Unified Streaming and Batch Pipelines at LinkedIn using Beam

Batch Mode

```python
# 1. Select Input
if options.input_query:
    # Read from BQ for backfilling data
    # SELECT * FROM `some_dataset.fhir_resources.Patient` LIMIT 1000
    messages = (
        pipeline
        | 'Query BQ Table' >> beam.io.ReadFromBigQuery(query=options.input_query, use_standard_sql=True, use_json_exports=True)
        | "convert to PubsubObject" >> beam.Map(lambda elem: PubsubMessage(json.dumps(elem).encode('utf-8'), {}))
        | 'Add Timestamps' >> beam.Map(lambda x: beam.window.TimestampedValue(x, time.time()))
    )
elif options.input_gcs_filepath:
    messages = (
            pipeline
            | 'Read From GCS' >> beam.io.ReadFromText(options.input_gcs_filepath)
            | "convert to PubsubObject" >> beam.Map(lambda elem: PubsubMessage(json.dumps(elem).encode('utf-8'), {}))
            | 'Add Timestamps' >> beam.Map(lambda x: beam.window.TimestampedValue(x, time.time()))
    )
else:
    # Read from Pubsub
    messages = (
        pipeline
        | f"Read PubSub Messages {options.input_subscription}" >> beam.io.ReadFromPubSub(subscription=options.input_subscription, with_attributes=True)
    )
```

Beam Summit 2022 - Unified Streaming and Batch Pipelines at LinkedIn using Beam

Python UDF

```python
# udf_main is the main entry-point of every mapper function and needs to exist
def udf_main(message, settings):
    # message is of type PubSubMessage
    # get main body (data) and attributes like this
    data = json.loads(message.data.decode("utf-8"))
    attributes = dict(message.attributes)

    logging.info(f"Transforming {data} with {attributes} ")

    version = settings["version"]
    project = settings["project_id"]
    region = settings["location"]
    dataset = settings["dataset"]
    fhirstore = "core"

    league_user_id = data["league_details"]["league_uid"]
    patient_id = hashlib.sha1(league_user_id.encode("UTF-8")).hexdigest()

    patient_reference_url = f"https://healthcare.googleapis.com/{version}/projects/{project

    yield get_patient(data, patient_id, league_user_id, patient_reference_url)

    if "services" in data:
        yield from get_services(data, league_user_id, patient_reference_url)

    if "hsa_eligibility" in data:
        yield get_hsa_eligibility(data, league_user_id, patient_reference_url)

    if "active_benefit_plans" in data:
        yield from get_active_benefit_plans(data, league_user_id, patient_reference_url)
```

FHIR Path

```
{
    "resourceType": "'Observation'",
    "id": "sha1('user-seg/'+Observation.identifier.value.first()+'/.../'+Observation.effectiveDateTime.toString())",
    "identifier":
    [
        {
            "system": "'some-fhir-system'",
            "value": "Observation.identifier.value.first()"
        }
    ],
    "effectiveDateTime": "Observation.effectiveDateTime",
    "valueString": "Observation.component.where(code.coding.code = 'workplace_location').valueString",
}
```

https://build.fhir.org/fhirpath.html

FHIR Path

Python

```python
if "services" in data:
    yield from get_services(data, league_user_id, patient_reference_url)

if "hsa_eligibility" in data:
    yield get_hsa_eligibility(data, league_user_id, patient_reference_url)

if "active_benefit_plans" in data:
    yield from get_active_benefit_plans(data, league_user_id, patient_reference_url)
```

FHIR Path

```
"meta":
{
    "id": "template_a",
     ...
    "optional_mapping_query": "services.exists()"
},
```

https://build.fhir.org/fhirpath.html

FHIR Path

Multiple Mappings

```
[
    {
        mapping 1
    },
    {
        mapping 2
    },
    {
        mapping 3
    },
]
```

Inheritance

```
"meta":
{
    "id": "template_b",
    "parent": "template_a"
},
```

https://build.fhir.org/fhirpath.html

Architecture

Requirements

Set up CI/CD fast

Set up CI/CD for new jobs fast

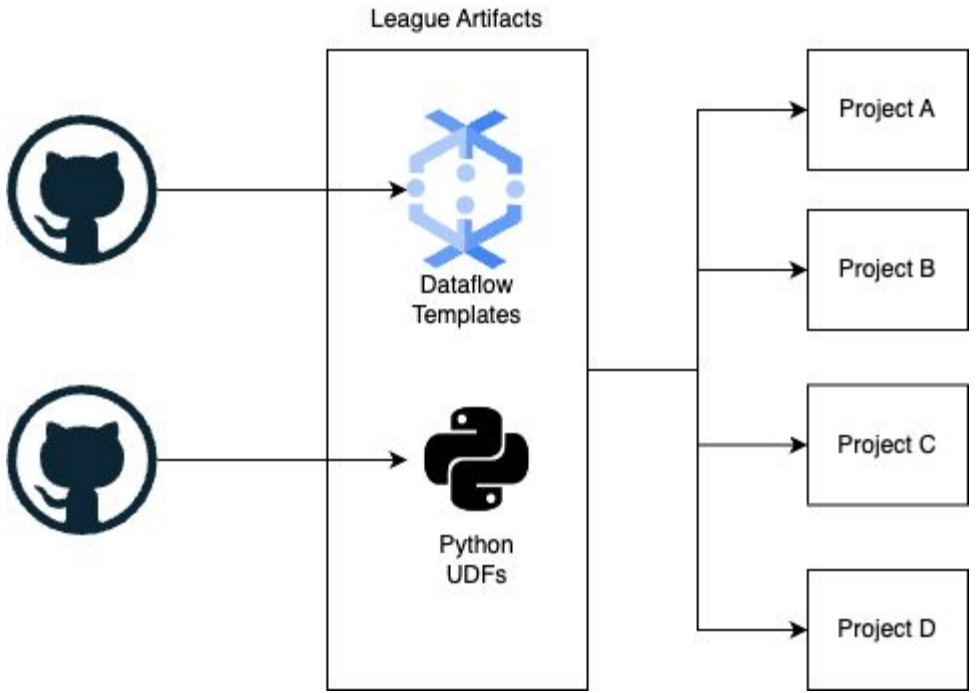
Different Environments

Deploy different version in different environments test / prod

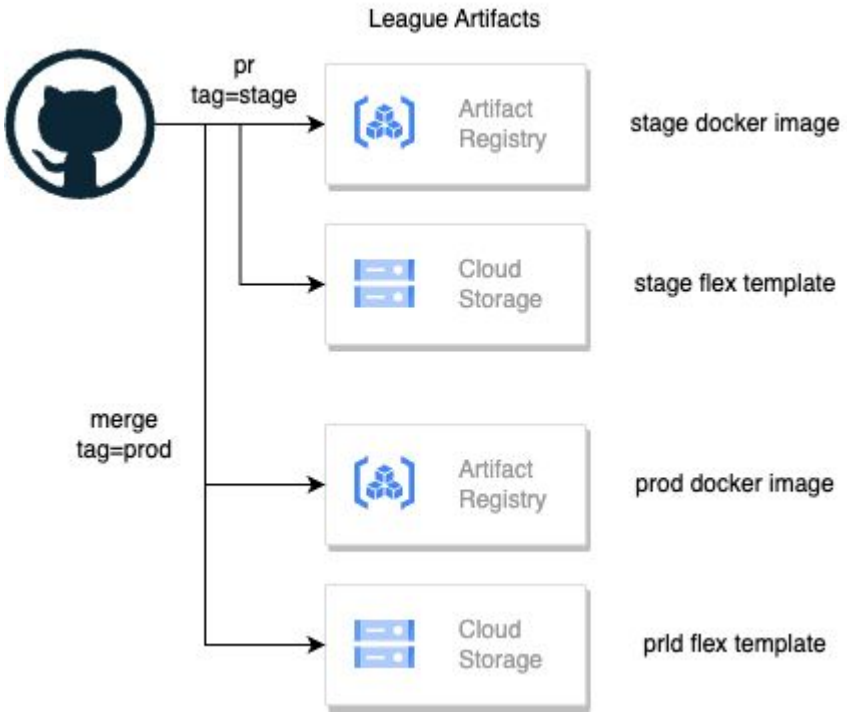
Different Projects

Automated deploy in different projects with different configs

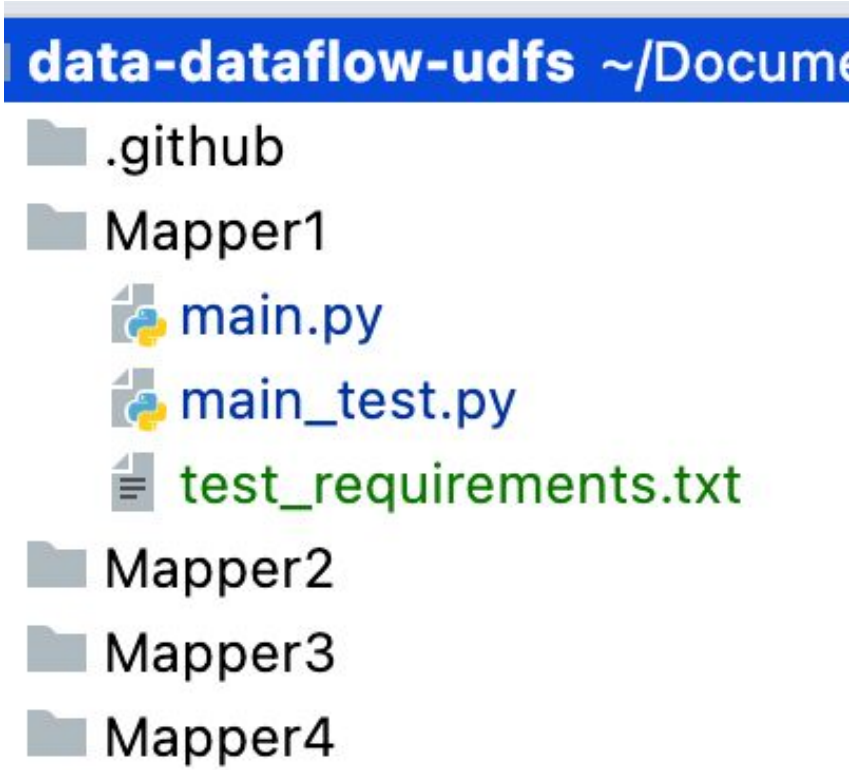
Requirements



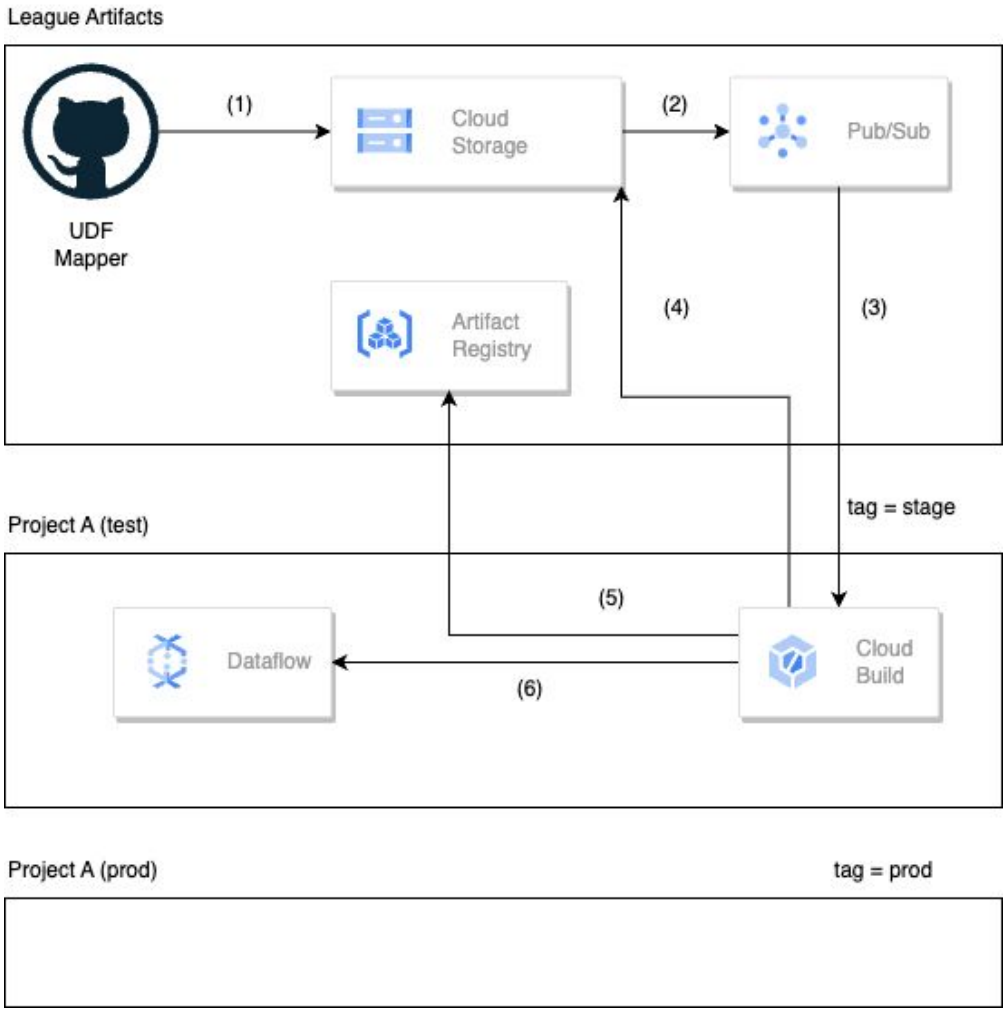
Template Generation



UDF Deployment



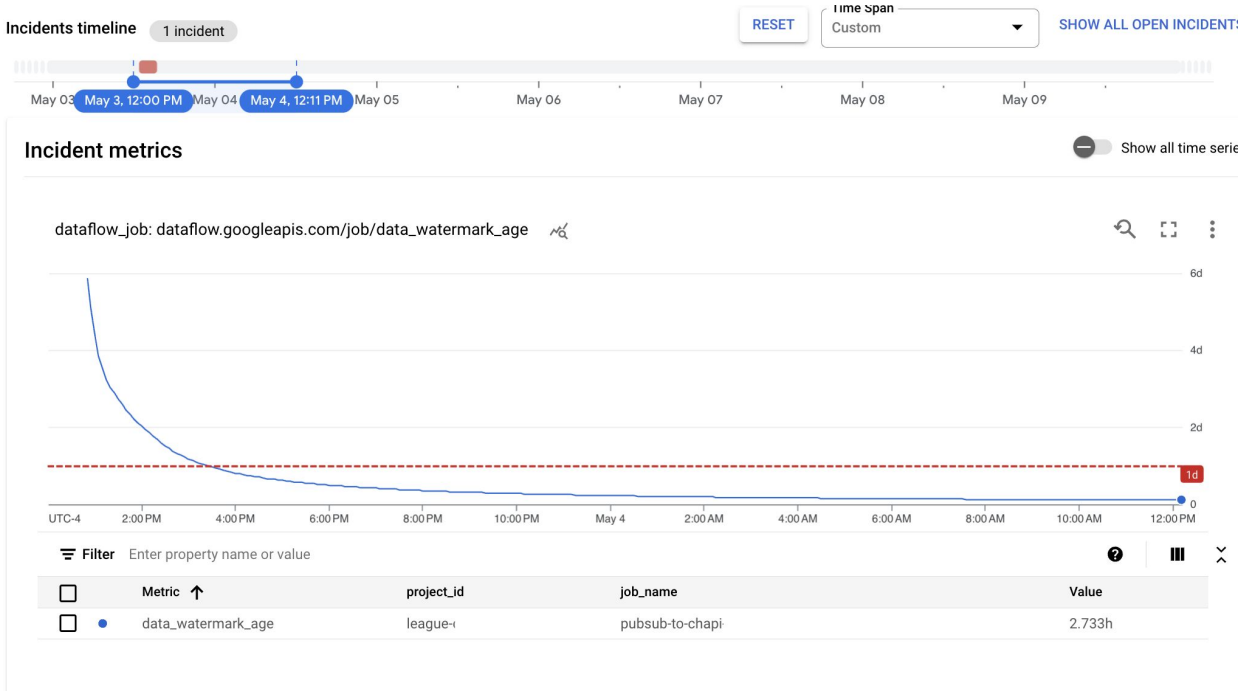
UDF Deployment



Terraform

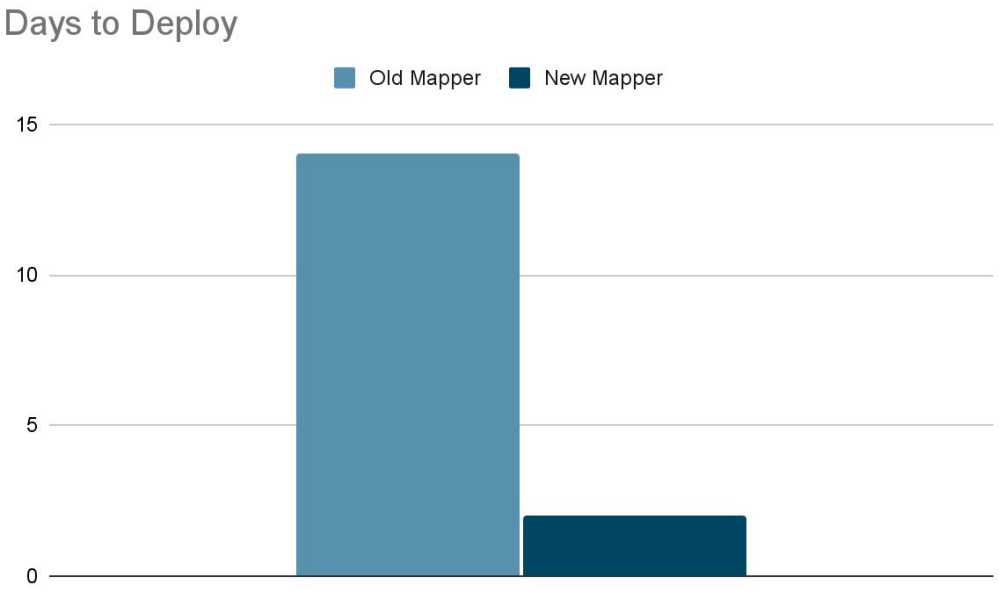
```
module "alex_test" {
  source = "../reference-datalake/modules/dataflow/jobs/udf-pubsub-to-chapi"

  project             = data.google_project.default.project_id
  project_number      = data.google_project.default.number
  metaregion          = var.metaregion
  env_type            = var.env_type
  udf_name            = "alex-test-df-job"

  builder_email       = module.dataflow_infra.builder_email
  runner_email        = module.dataflow_infra.runner_email

  pubsub_topic        = var.fhir_data_topic
  subscription_filter = "attributes.tenant_id = \"${var.tenant_id}\""

  initial_num_workers = 1
  max_num_workers     = 1

  chapi_base_url      = module.chapi.core_url
  error_bucket_url    = module.dataflow_infra.error_bucket_url

  enable_streaming_engine = true
  enable_dataflow_prime = true
  enable_write_raw_to_gcs = false
}
```

Alerting


Incidents timeline
1 incident
RESET
Time Span
Custom
SHOW ALL OPEN INCIDENTS
May 03
May 3, 12:00 PM
May 04
May 4, 12:11 PM
May 05
May 06
May 07
May 08
May 09
Incident metrics
Show all time serie
dataflow_job: dataflow.googleapis.com/job/data_watermark_age
6d
4d
2d
1d
0
UTC-4
2:00 PM
4:00 PM
6:00 PM
8:00 PM
10:00 PM
May 4
2:00 AM
4:00 AM
6:00 AM
8:00 AM
10:00 AM
12:00 PM
Filter
Enter property name or value
Metric
project_id
job_name
Value
data_watermark_age
league-
pubsub-to-chapi
2.733h

Results

Time to Deploy



Alex Fragotsis

QUESTIONS?

Alexfragotsis

@alexfragotsis