Migrating billions records from SQL to NoSQL using continuous migration technique with PySpark and DataProc

Piti Champeethong (A.K.A: Fyi == ฟี่)

PyCon Thailand Ambassador
MongoDB Thailand User Group Leader
Microsoft MVP (Python and DevOps)
https://github.com/ninefyi

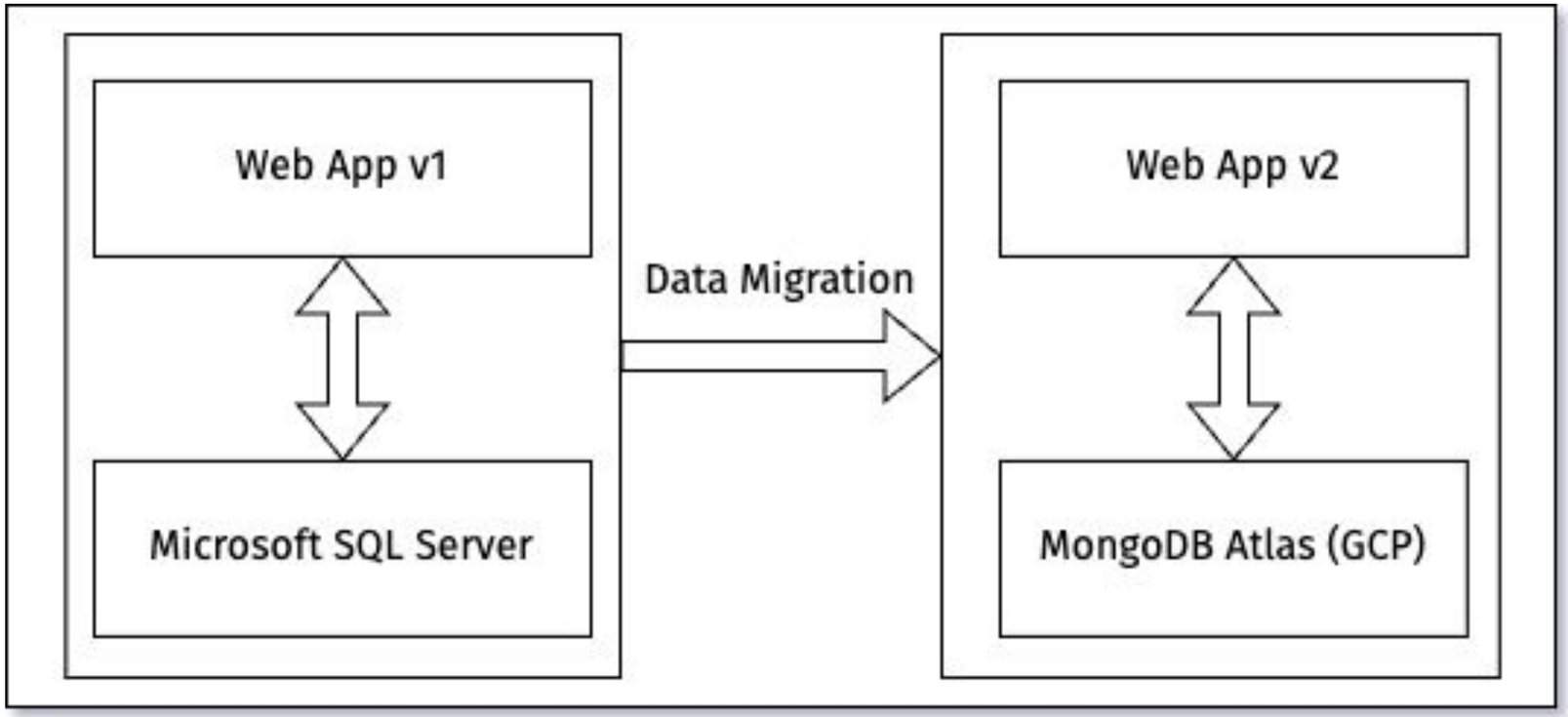
Agenda

- Business context and technical challenge of million rows data migration.
- Data Pipeline Architecture cover below technology
  - Microsoft SQL Server
  - Google Cloud Platform (GCP): DataProc
  - PySpark
  - MongoDB Atlas
- Migration approach for handling billion-row from MsSQL to MongoDB Atlas
- Q&A

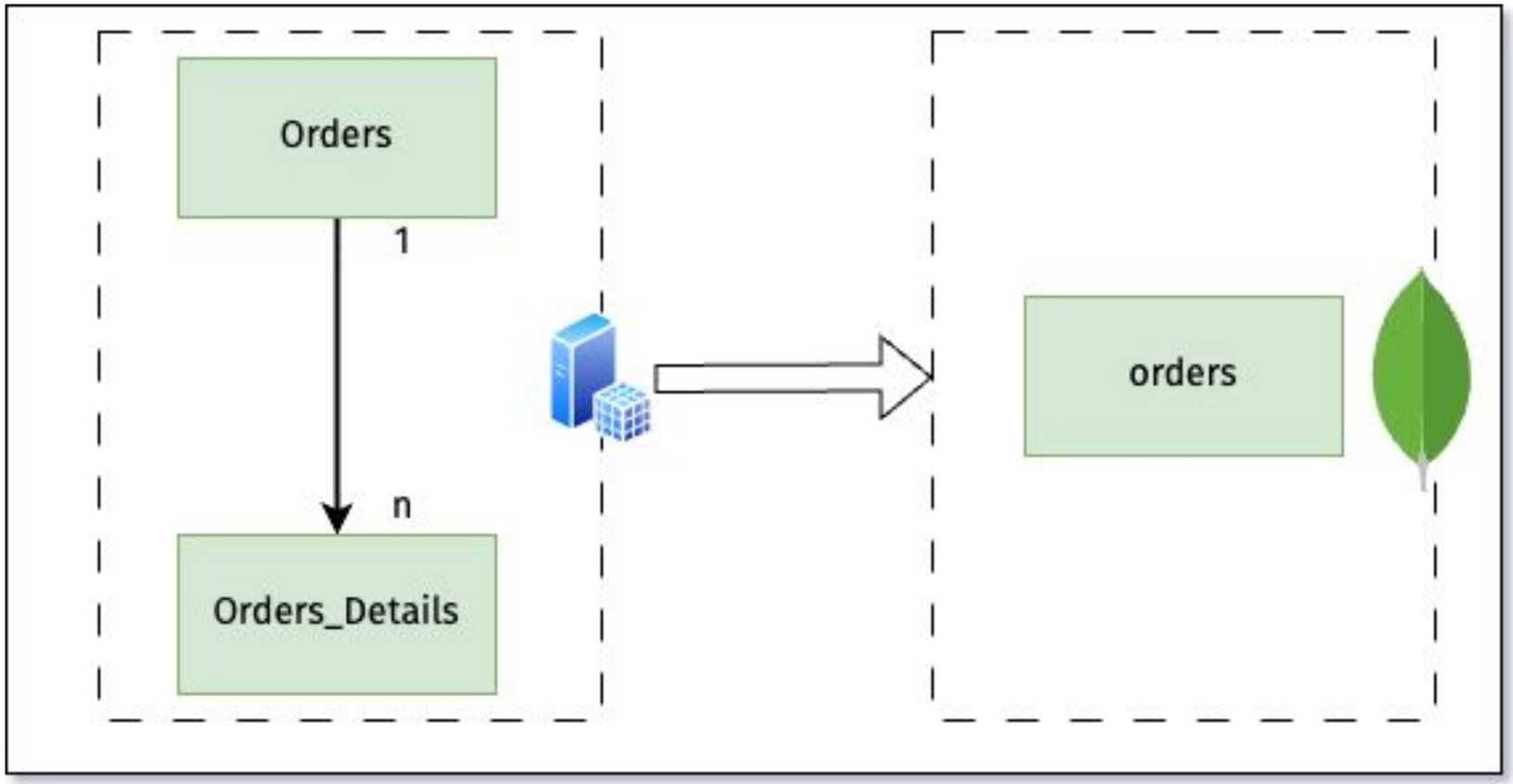
Business and Technical Challenges

- MongoDB Atlas (GCP)
- Massive data (~1B rows)
- Denormalization
- Data Migration with Minimum Downtime

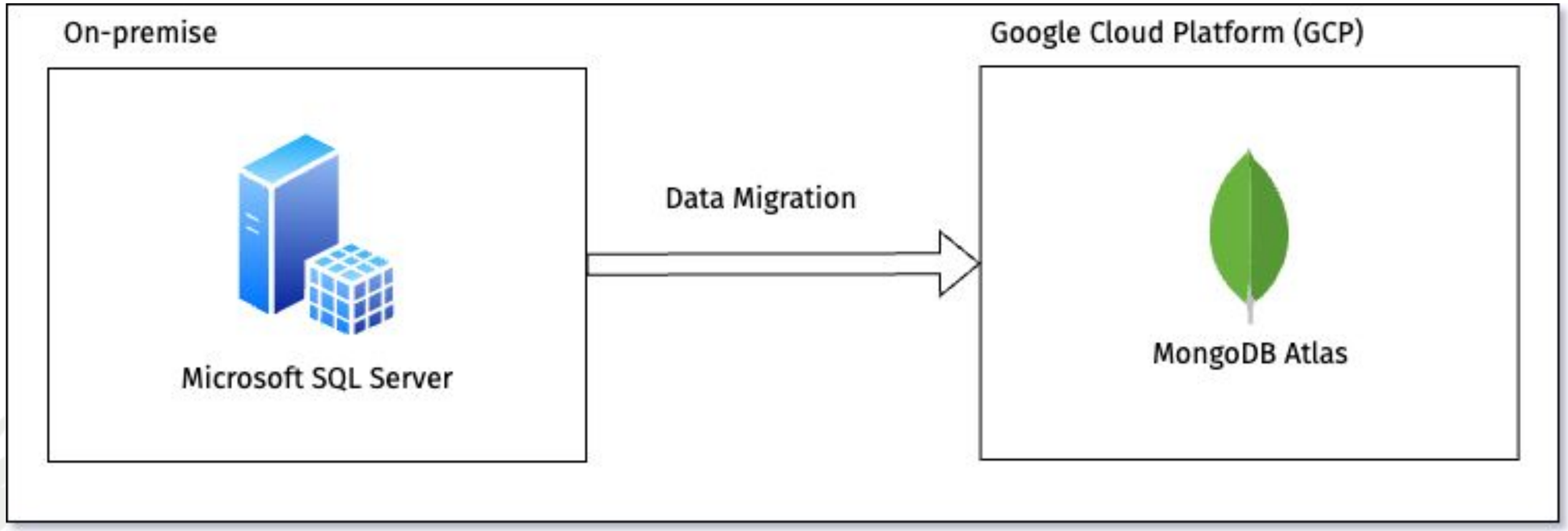
Challenges - Minimum Downtime



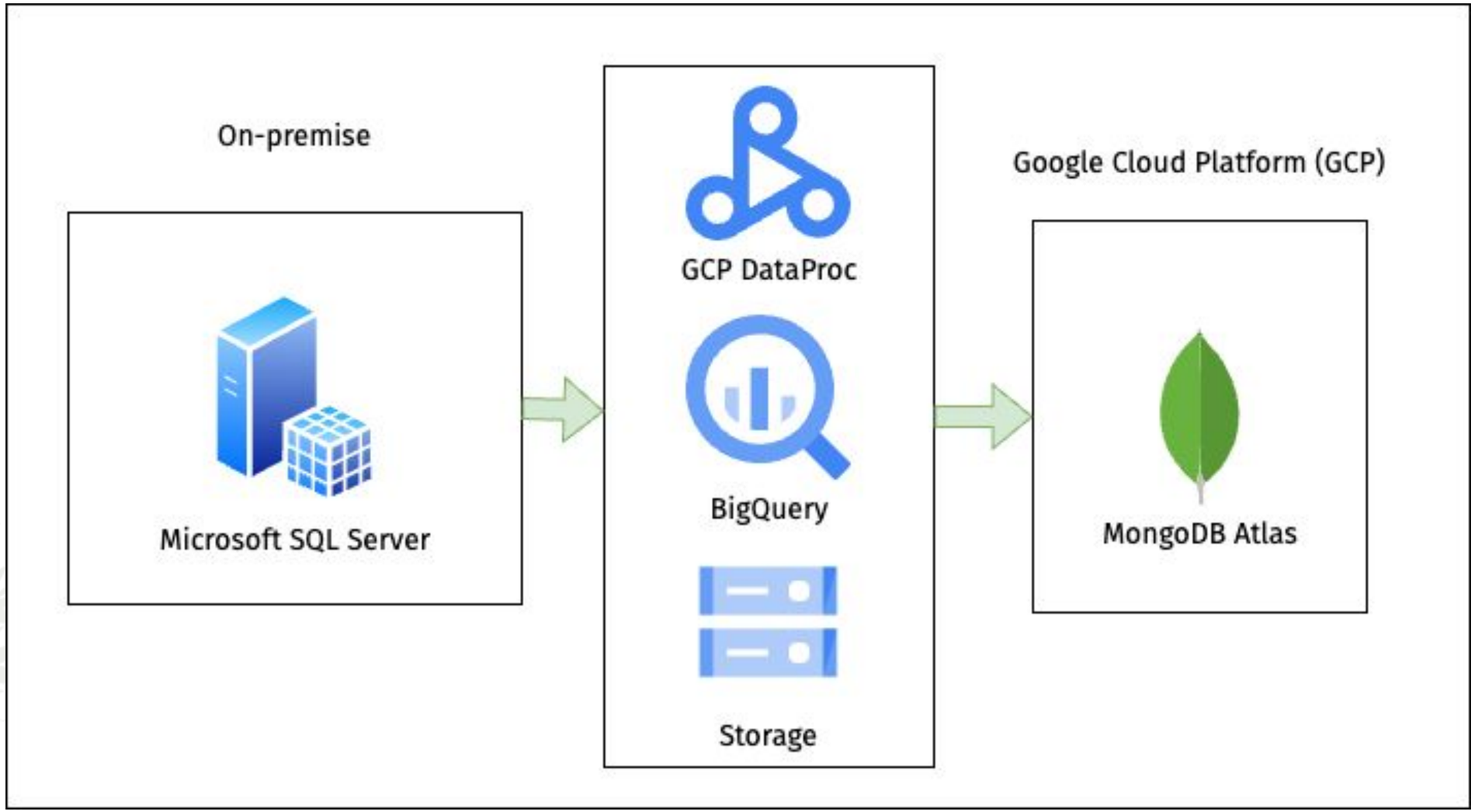
Challenges - Denormalization



Data Pipeline Architecture



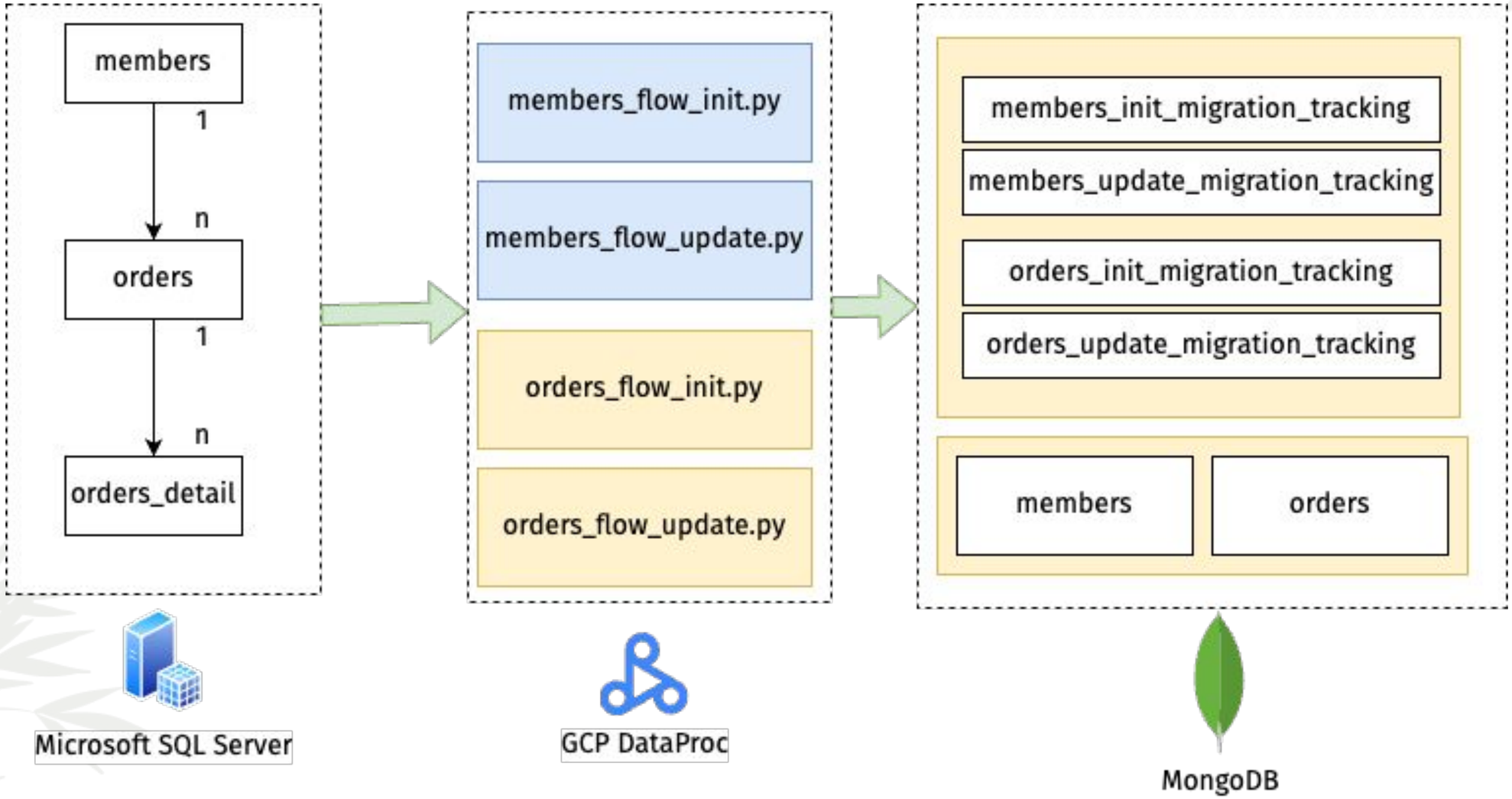
Data Pipeline Architecture



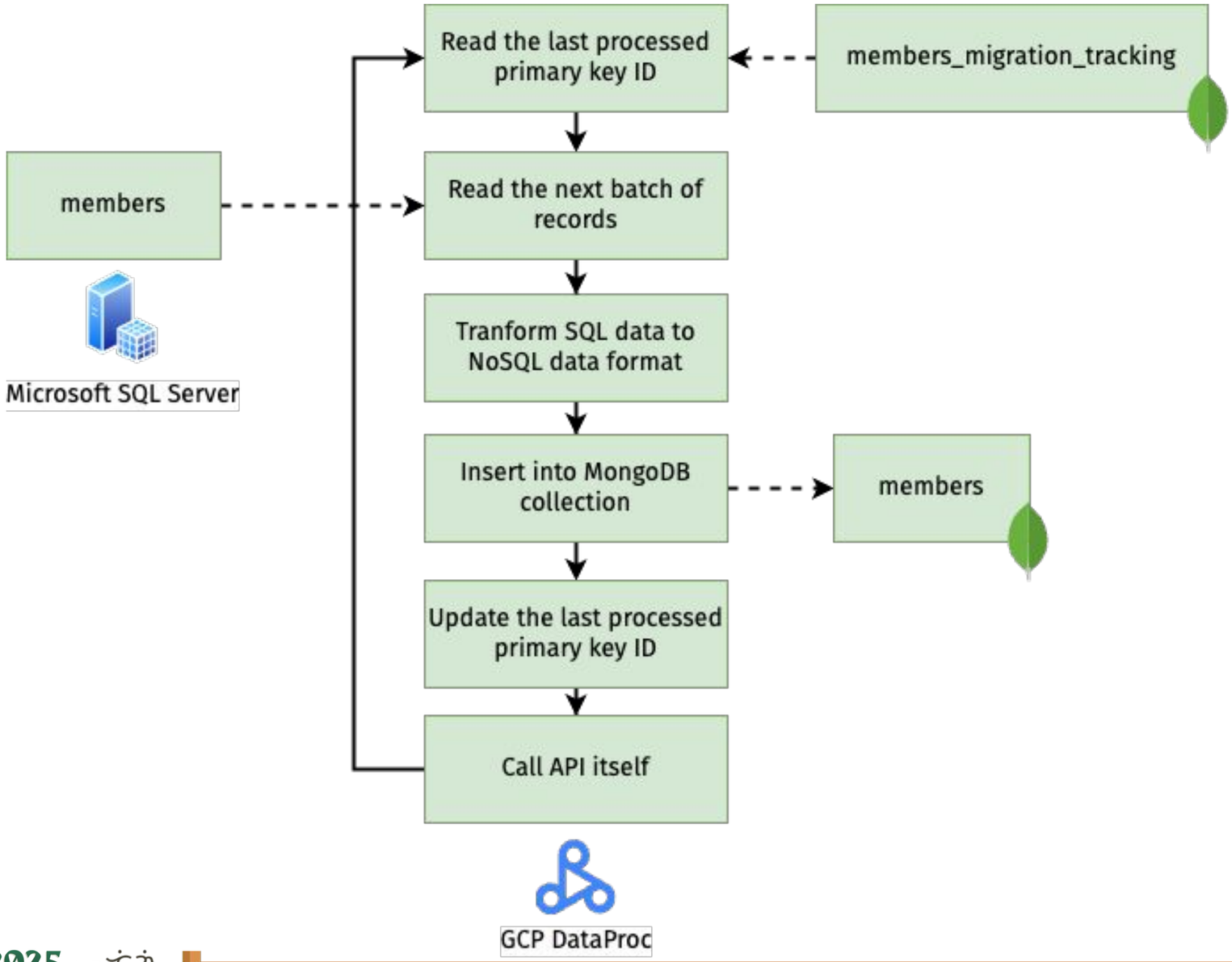
Data Pipeline Architecture

- GCP Cloud Storage (Python files and libraries)
- GCP DataProc (Continuous running)
- Microsoft SQL Server (Source)
- MongoDB Atlas GCP (Destination)

Deep dive to continuous technique



Deep dive to continuous technique



Microsoft SQL Server

orders_id	orders_no	orders_date	orders_total
1	A20250211	20250211	1000

orders_detail_id	orders_id	product_id	unit_price	amount	total
1	1	1	100	2	200
2	1	2	100	4	400
3	1	3	200	2	400

MongoDB Collection

```
{
  "_id": "dlkafejaeoirefje0rfe",
  "orders_id": 1,
  "order_details":[
      { "orders_detail_id": 1, "product_id": 1, "unit_price":100, "amount":1 ,"total": 200} ,
      { "orders_detail_id": 1, "product_id": 1, "unit_price":100, "amount":1 ,"total": 200} ,
      { "orders_detail_id": 1, "product_id": 1, "unit_price":100, "amount":1 ,"total": 200}
  ]
}
```

Deep dive to continuous technique

```
query = f"""
(
    SELECT TOP {Config.BATCH_SIZE} {', '.join(columns)}
    FROM dbo.MEMBER
    WITH(NOLOCK)
    WHERE MemberID > '{last_processed_id}'
    ORDER BY MemberID ASC
) as subquery
"""
```

```python
def contious_migration():
    # Initialize a Dataproc instance
    client = dataproc.JobControllerClient(client_options={
        'api_endpoint': '{}-dataproc.googleapis.com:443'.format('asia-southeast1')
    })
    # Define your cluster details
    project_id = 'dataproc-demo-project'
    region = 'asia-southeast1'
    cluster_name = 'dataproc-demo-cluster'
    # Prepare your pyspark job details
    job_payload = {
        'placement': {
            'cluster_name': cluster_name
        },
        'pyspark_job': {
            'main_python_file_uri': 'gs://pyapac2025_bucket/member_flow_init.py',
            'jar_file_uris': ['gs://pyapac2025_bucket/jars/mongo-spark-connector_2.12-10.2.1-all.jar',
                              'gs://pyapac2025_bucket/jars/mssql-jdbc-12.8.0.jre11.jar',
                              'gs://pyapac2025_bucket/jars/spark-mssql-connector_2.12-1.2.0.jar'],
        }
    }
    # Submit the job
    job_response = client.submit_job(
        project_id=project_id, region=region, job=job_payload)
    # Output a response
    print('Submitted job ID {}'.format(job_response.reference.job_id))
```

```python
def main():
    mongo_handler = MongoDBHandler()
    spark_handler = SparkHandler()

    try:
        last_processed_id = mongo_handler.get_last_processed_id()
        print(f"Last processed MemberID: {last_processed_id}")

        df_base_member = spark_handler.load_member_data(last_processed_id)

        if not df_base_member.rdd.isEmpty():
            transformed_df = spark_handler.transform_data(df_base_member)
            spark_handler.write_to_mongodb(transformed_df)

            last_processed_id = df_base_member.select("MemberID").rdd.max()[0]
            mongo_handler.update_last_processed_id(last_processed_id)
            print(f"Migration completed successfully {last_processed_id}")

            sleep(10)
            submit_dataproc_job()

    except Exception as e:
        print(f"Error during migration: {e}")
    finally:
        mongo_handler.close()
        spark_handler.stop()

if __name__ == "__main__":
    main()
```

Deep dive to continuous technique

```
gcloud dataproc jobs submit pyspark dataproc-demo-project \
--cluster=dataproc-demo-cluster \
--region=asia-southeast1 \
--jars
gs://pyapac2025_bucket/jars/mongo-spark-connector_2.12-10.2.1-all.jar,
gs://pyapac2025_bucket/jars/mssql-jdbc-12.8.0.jre11.jar,
gs://pyapac2025_bucket/jars/spark-mssql-connector_2.12-1.2.0.jar \
gs://pybucket/member_flow_init.py
```


Limitation

- No support for running two applications at the same time (App v1 and App v2)
- Support only PRIMARY KEY as an auto number
- Need Created/Modified date field on source

References

- https://cloud.google.com/dataproc/docs/tutorials/python-library-example
- https://www.mongodb.com/docs/spark-connector/v10.2/python/api/

Workshop Day
17 October 2025
@Microsoft Thailand

Conference Days
18 October 2025
@Avani Sukhumvit Bangkok

https://th.pycon.org

THANK YOU!