FANDUEL GROUP

Using Trino as a Strangler Fig

Trevor Kennedy | Data Architect | An agile approach to database migrations

Introductions

Speaker

Trevor Kennedy

Staff Data Architect at FanDuel

Previous experience:

- Solution Architect
- Consultant
- Data Engineer
- Software Engineer

“I’ve performed numerous database migrations over the past 15+ years”

Company

FanDuel (NYSE: FLUT)

America’s #1 Sportsbook and the premier mobile sports betting operator with 12M+ registered users

- Sportsbook & Retail Sportsbook
- Casino
- Fantasy Sports
- FanDuel TV
- Racing
- Faceoff

Gambling Problem?

Call 1-800-GAMBLER or visit https://rg-help.com

Situation & Context

We're fighting a "monolithic" data swamp in AWS Redshift

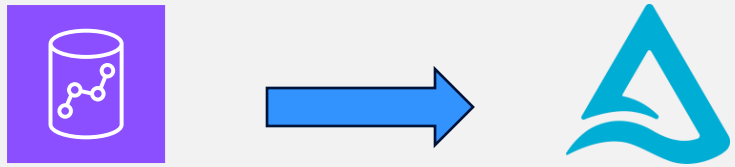
- 6500+ tables (haphazard and evolutionary growth)
- Hundreds of stakeholders (Analysts, ML Engineers, etc.)

We want to migrate and create a single source of truth in Delta Lake

- Decouple analytical workspaces
- Reduce costs and improve governance

Desire to minimize impact:

- Limit the number of edits to existing queries and notebooks
- Reduce federated in memory joins of Redshift and Delta Lake data
- Phased and flexible changes to data producing systems



Big Bang Modernizations

*“If you do a big-bang rewrite, the only thing you’re guaranteed of is a **big bang**.”*

— Martin Fowler

Downsides

- High risk
- Longer time to market
- Trick to roll-back
- Waterfall SDLC
- Not agile

*“It’s important to remember that when you start from scratch there is **absolutely no reason** to believe that you are going to do a better job than you did the first time”*

— Joel Spolsky (2000)

The Stranger Fig Pattern

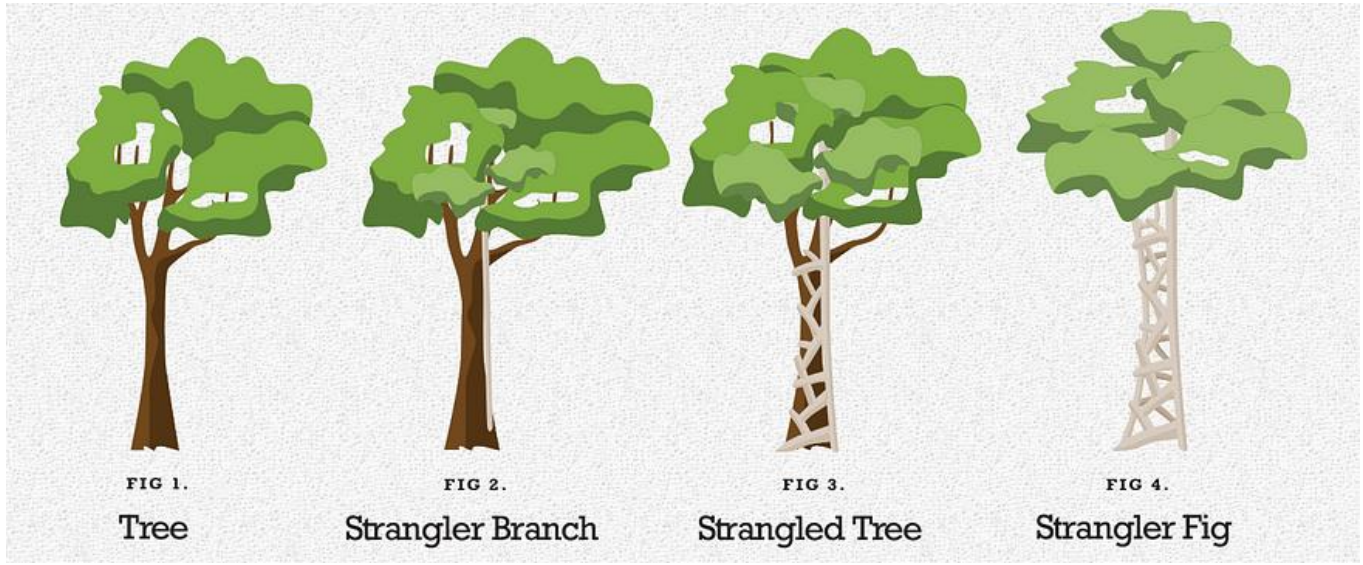
A strangler fig germinates in the nook of a tree as a shortcut to reach sunlight and eventually grows large enough to envelope and kill its host tree.

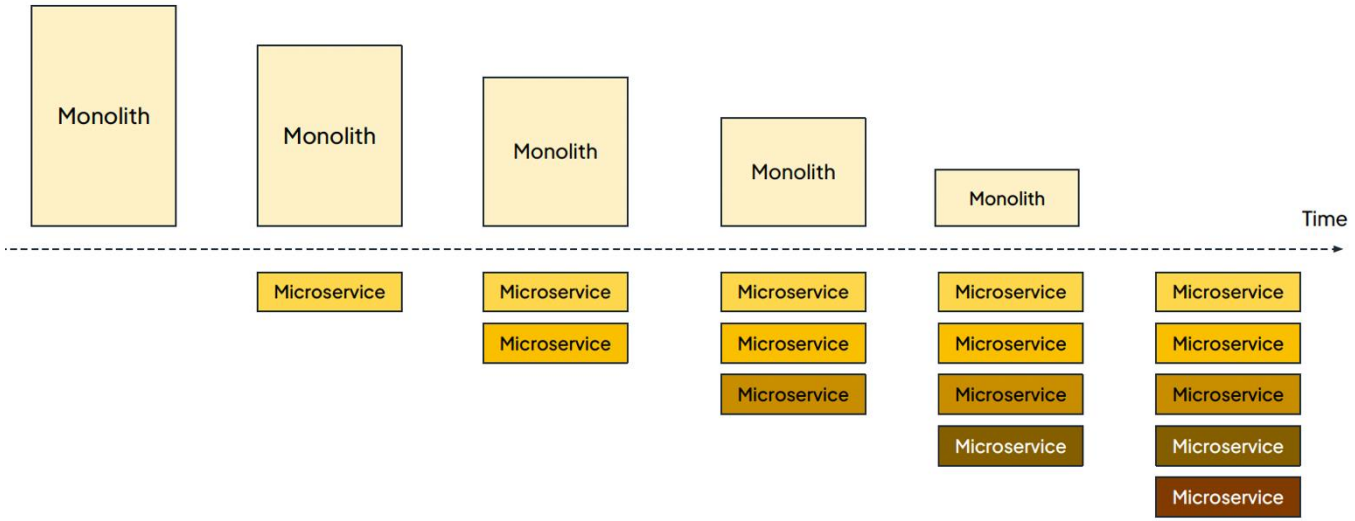
"Gradually create a new system around the edges of the old, letting it ground slowly over several years until the old system is strangled"

— Martin Fowler, Thoughtworks (2004)

Illustration generated by Microsoft Bing AI

Changing Gradually





Benefits

- Agility
- Frequent releases
- Lowers risk
- Immediate benefits
- Easier rollback

Phases

- Extract
- Co-exist
- Eliminate

AWS Prescriptive Guidance

Use the Strangler Fig Pattern when:

- You want to migrate your monolithic application gradually.
- A big bang migration approach is risky because of the size and complexity of the monolith.
- The business wants to add new features and cannot wait for the transformation to be complete.
- End users must be minimally impacted during the transformation.
- Large monoliths benefit the most from the strangler fig pattern.

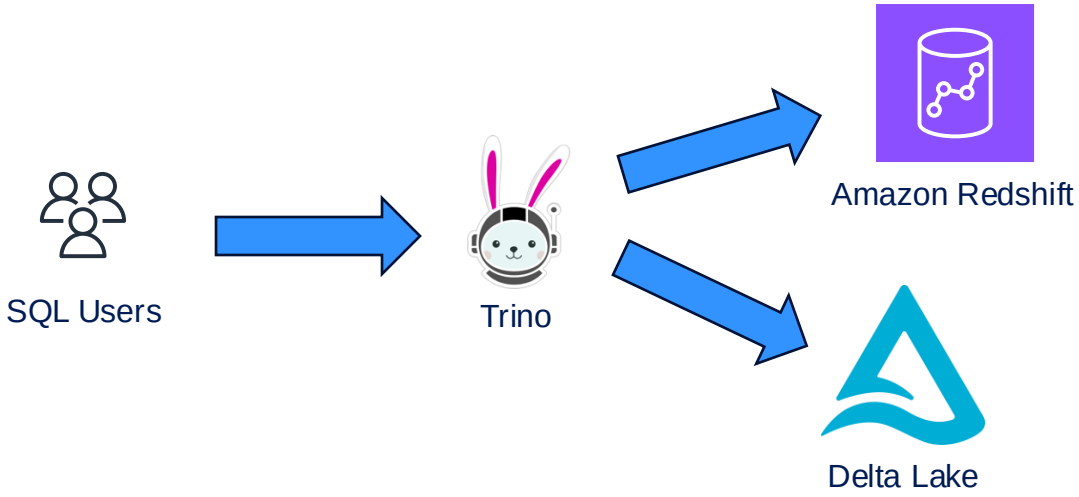
Best Practices

- Domain-driven design (DDD) is a mechanism for understanding the domain.
- Event storming is a technique for determining domain boundaries.

Migration Phases

1. Analyze Redshift query log to elicit domain boundaries
2. Conduct user interviews to verify business boundaries i.e. the group of tables required by each group of domain users
3. Identify common steel threads (e.g. customer_deposits) and create a migration plan based on the critical path method
4. Introduce a façade for each domain group (i.e. marketing)
5. Migrate a domain group from Redshift to Delta Lake
6. Repeat process for the next domain boundary

Trino as a SQL Façade



Leverage Trino to provide an ANSI SQL compliant abstraction layer for an (almost) seamless end user migration experience across heterogeneous data sources

Prepare Redshift

Amazon Redshift

`monolithic_schema` → `marketing`

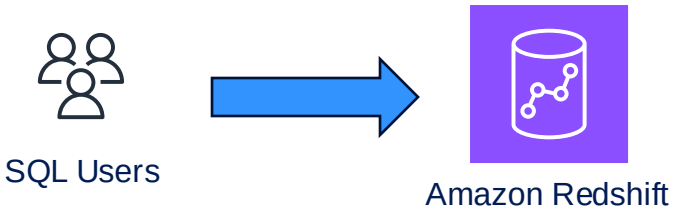
Begin disentangling the data swamp by creating domain specific curated hub for business domains.

```
CREATE SCHEMA marketing;


CREATE VIEW marketing.ads as select * from monolithic_schema.ads;
CREATE VIEW marketing.campaigns as select * from monolithic_schema.campaigns;
```

Introduce the Trino Facade

Before:

```
USE marketing;
SELECT * FROM ads;
```



After:

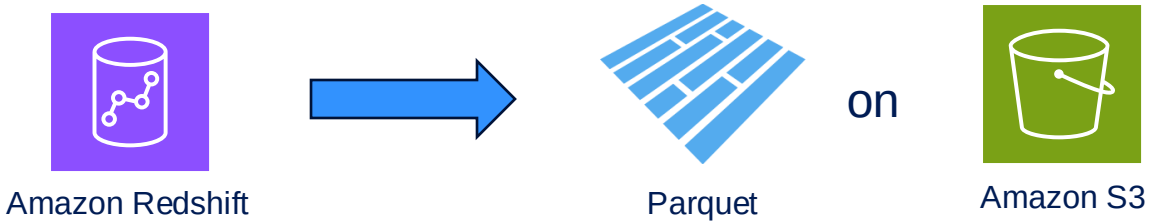
```
USE marketing_hub.marketing;
SELECT * FROM ads;
```

Connect Trino to Redshift by creating a catalog:

```
CREATE CATALOG marketing_hub USING redshift
WITH (
  "connection-url" = 'jdbc:redshift://example.net:5439/database'
);
```

Users now connect to Trino instead of connecting to directly to Redshift.

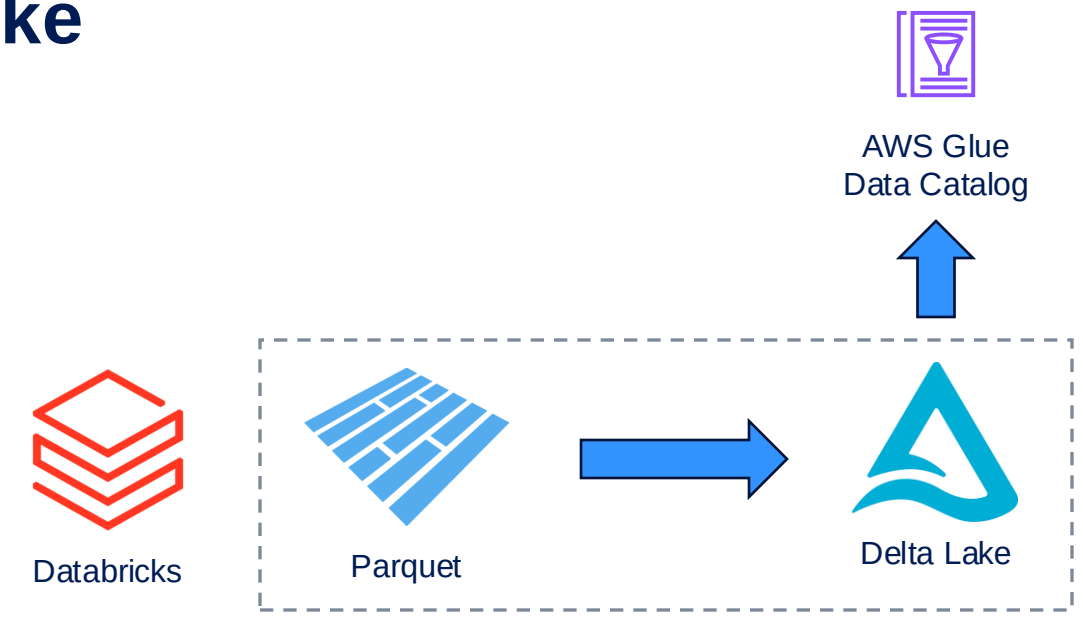
Export From Redshift



Export table contents from Redshift to S3 as Parquet files:

```
UNLOAD ('select * from marketing.ads') // Redshift SQL
TO 's3://my-bucket/marketing/ads'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRole'
FORMAT AS PARQUET;
```

Convert to Delta Lake



Enable the Glue Data Catalog in Databricks:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
.config("spark.databricks.hive.metastore.glueCatalog.enabled", "true") \

  .enableHiveSupport() \

  .getOrCreate()
```

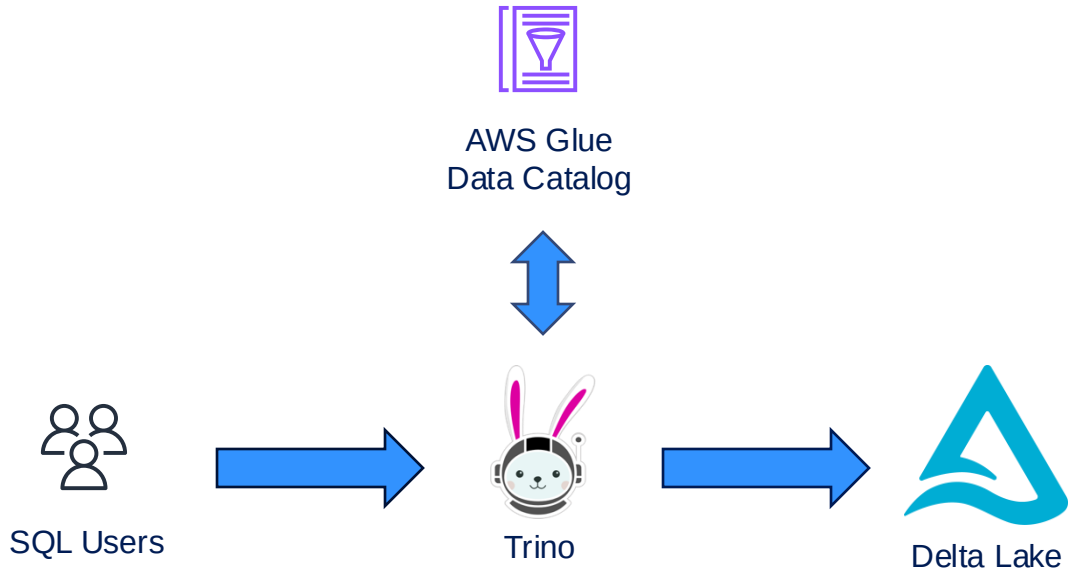
Read in Parquet file, write it out as Delta Table and register it in the Data Catalog:

```
df = spark.read.parquet("s3://my-bucket/marketing/ads")

df.write \

    .format("delta") \

    .option("path","s3://my-bucket/marketing/ads) \

    .mode("overwrite") \

    .saveAsTable(”maketing.ads")
```

Exported table is now in Delta Lake format and registered in the Glue Data Catalog

Redirect Users to Delta Lake

```
USE marketing_hub.marketing;
SELECT * FROM ads;
```

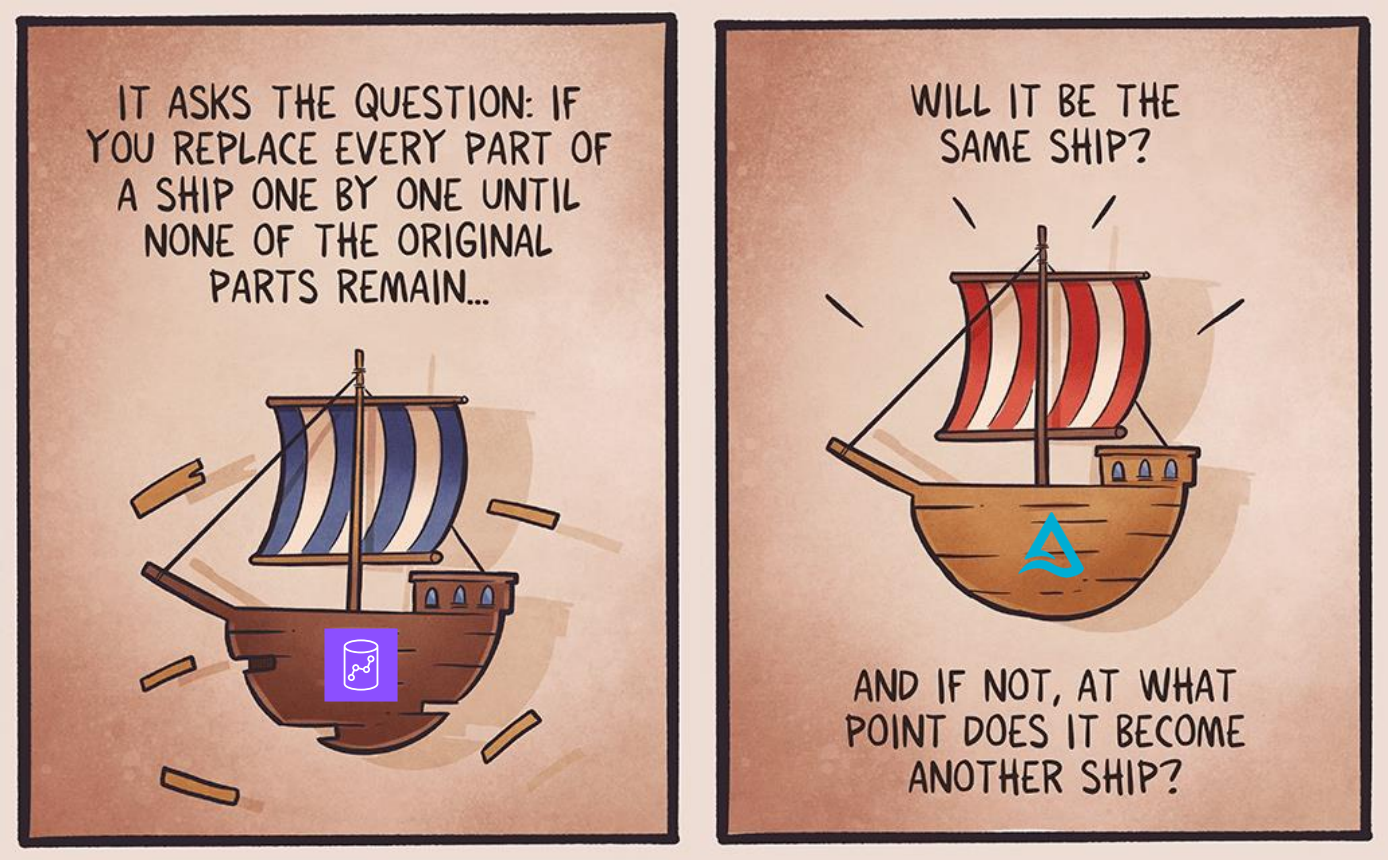


Update the catalog entry to point to Delta Lake instead of Redshift (drop and create):

```
DROP CATALOG marketing_hub; // In Trino
CREATE CATALOG marketing_hub USING delta_lake
WITH (
  "hive.metastore.uri" = 'thrift://example.net:9083',
  "hive.metastore" = 'glue';
);
```

Now using the same Trino query as before, analysts will be hitting the Delta Lake instead of Redshift

Conclusion



Did we just solve Theseus's Paradox?

We nave replaced the planks one at at time resulting in the same ship (but different) so the sailors can keep on sailing

References

1. https://trino.io/docs/current/overview.html
2. https://trino.io/docs/current/language.html
3. https://trino.io/docs/current/connector/redshift.html
4. https://trino.io/docs/current/connector/delta-lake.html
5. https://docs.databricks.com/en/archive/external-metastores/aws-glue-metastore.html
6. https://martinfowler.com/bliki/StranglerFigApplication.html
7. https://www.thoughtworks.com/en-us/insights/articles/embracing-strangler-fig-pattern-legacy-modernization-part-one
8. https://samnewman.io/books/monolith-to-microservices/
9. https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/cloud-design-patterns/cloud-design-patterns.pdf
10. https://ddd-practitioners.com/2023/08/02/from-big-bang-to-iterative-evolution-embracing-the-strangler-fig-pattern
11. https://www.slideshare.net/slideshow/dissecting-our-legacy-the-strangler-fig-pattern-with-debezium-apache-kafka-mongodb-gunnar-morling-red-hat/250324151?embed_session_id=c93c0216-4979-4f55-b444-a4f1dd91269d#1
12. https://wso2.com/wso2_resources/wso2con2024-slides/not-just-microservices-rightsize-your-services.pdf
13. https://d1.awsstatic.com/events/reinvent/2021/Modernize_faster_using_the_Strangler_Fig_pattern_on_AWS_ENT401.pdf

Image Credits

1. https://scitechdaily.com/from-big-bang-to-big-picture-a-comprehensive-new-view-of-all-objects-in-the-universe/
2. https://medium.com/@sylvain.tiset/the-strangler-fig-pattern-is-what-you-need-to-migrate-monolithic-application-with-legacy-code-to-ec24cf7168eb
3. https://www.pastille.no/comics/ship-of-theseus