dagster

Shifting Left and Moving Forward

Modern Data Development with MotherDuck and Dagster

January 14, 2025

Table of contents

01 Introduction

02 Bluesky end-to-end project

03 Power of MotherDuck

04 Shifting left in data engineering

05 Discussion on modern data workflows

06 Q & A

dagster

Speakers

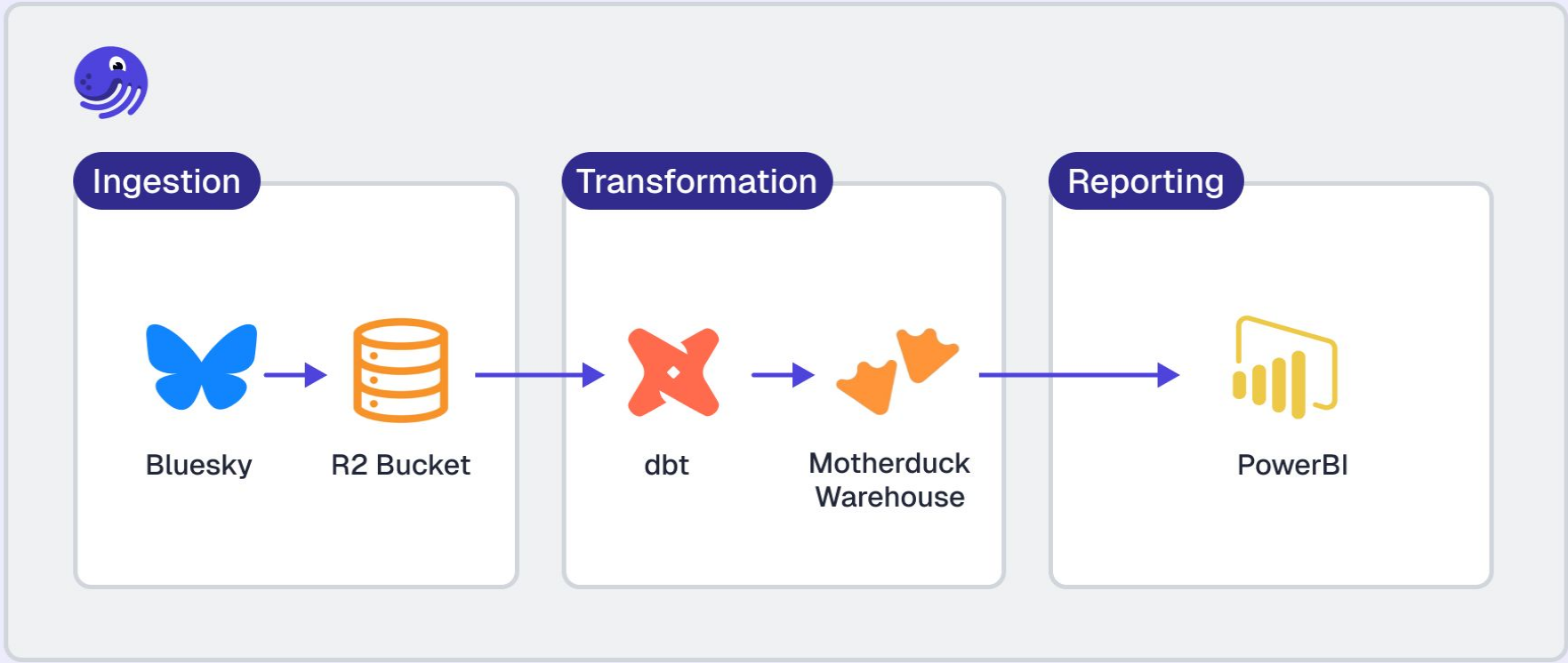
Colton Padden
Developer Advocate

Alex Noonan
Developer Advocate

Jacob Matson
Developer Advocate

dagster

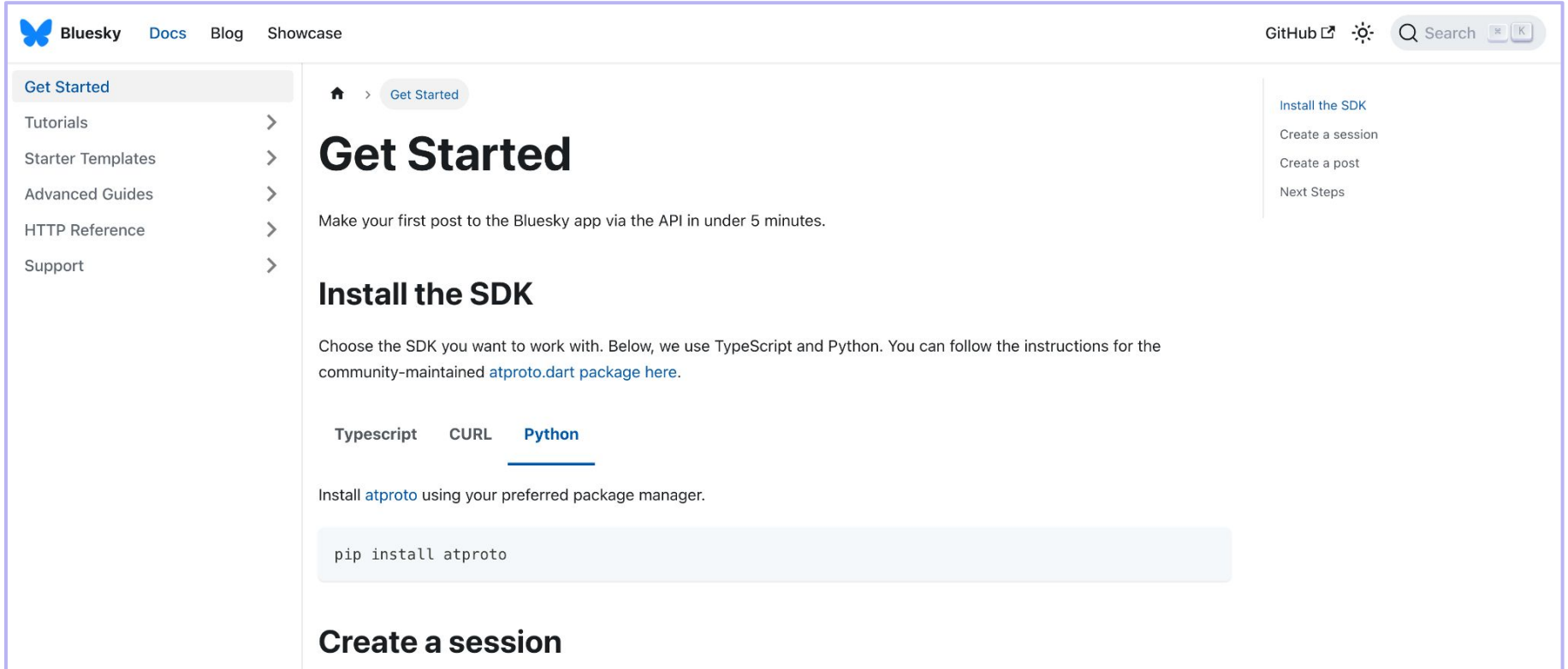
End to End Dagster Example



https://github.com/dagster-io/dagster/tree/master/examples/project_atproto_dashboard

Ingestion

Bluesky SDK



Ingestion

Dagster Resource

- Created an ATProtoResource
- Wraps the `atproto` SDK
- Creates a shared session across resource instantiations

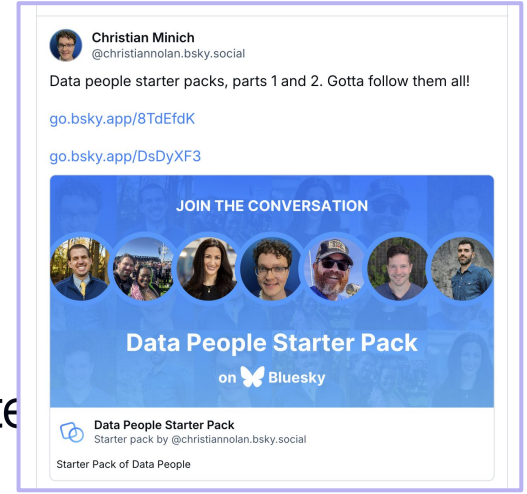
```python
class ATProtoResource(dg.ConfigurableResource):
    login: str
    password: str
    session_cache_path: str = "atproto-session.txt"

    def _login(self, client):
        """Create a re-usable session to be used across resource instances.

        We are rate limited to 30 requests / 5 minutes or 300 requests day.
        """
        if os.path.exists(self.session_cache_path):
            with open(self.session_cache_path) as f:
                session_string = f.read()
            client.login(session_string=session_string)
        else:
            client.login(login=self.login, password=self.password)
            session_string = client.export_session_string()
            with open(self.session_cache_path, "w") as f:
                f.write(session_string)

    def get_client(
        self,
    ) -> Client:
        client = Client()
        self._login(client)
        return client
```

dagster

Ingestion

Starter pack snapshots

- We create a snapshot of the “Data People Starter Pack”
- These members are used as dynamic partitions for user feed snapshots

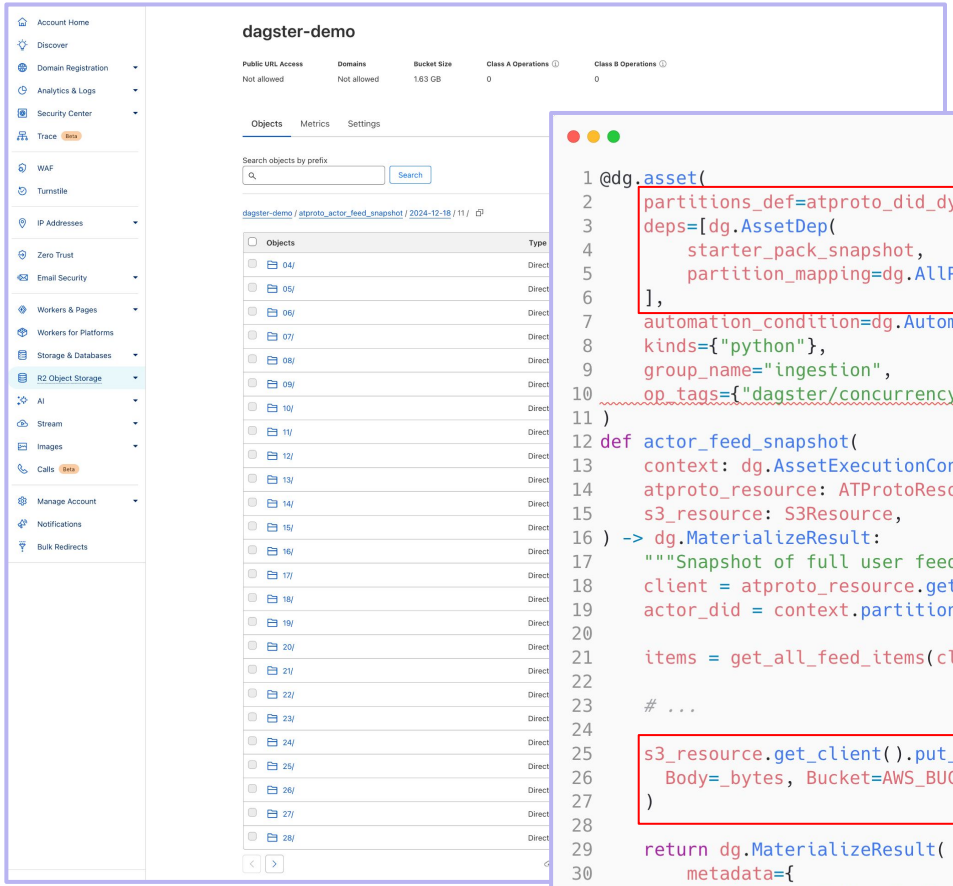



```python
@dg.asset(
    partitions_def=dg.StaticPartitionsDefinition(
        partition_keys=[
            "at://did:plc:lc5jzrr425fyah724df3z5ik/app.bsky.graph.starterpack/3l7cddlz5ja24",
        ]
    ),
    automation_condition=dg.AutomationCondition.on_cron("0 0 * * *"),
    kinds={"python"},
    group_name="ingestion",
)
def starter_pack_snapshot(
    context: dg.AssetExecutionContext,
    atproto_resource: ATProtoResource,
    s3_resource: S3Resource,
) -> dg.MaterializeResult:

    # ...

    list_items = get_all_starter_pack_members(atproto_client, starter_pack_uri)

    # ...

    context.instance.add_dynamic_partitions(
        partitions_def_name="atproto_did_dynamic_partition",
        partition_keys=[list_item_view.subject.did for list_item_view in list_items],
    )

    return dg.MaterializeResult(
        metadata={
            "len_members": len(list_items),
            "s3_object_key": object_key,
        }
    )
```

Ingestion

User feed snapshots

- Dynamic partitions
  - Partition mapping
- Automation conditions
- Files land in CloudFlare R2
  - S3 compatible cloud storage that's *way* easier to configure



dagster

Ingestion

Resiliency (Tenacity)

- Rate limiting
- Back-off retries w/ `Tenacity`
- Dagster concurrency limits

```yaml
# dagster.yaml

run_coordinator:
  module: dagster.core.run_coordinator
  class: QueuedRunCoordinator

concurrency:
  default_op_concurrency_limit: 1
```

```python
def get_all_feed_items(client: Client, actor: str) -> list["models.AppBskyFeedDefs.FeedViewPost"]:
    """Retrieves all author feed items for a given `actor`."""
    import math

    import tenacity

    @tenacity.retry(
        stop=tenacity.stop_after_attempt(5),
        wait=tenacity.wait_fixed(math.ceil(60 * 2.5)),
    )
    def _get_feed_with_retries(client: Client, actor: str, cursor: Optional[str]):
        return client.get_author_feed(actor=actor, cursor=cursor, limit=100)

    feed = []
    cursor = None
    while True:
        data = _get_feed_with_retries(client, actor, cursor)
        feed.extend(data.feed)
        cursor = data.cursor
        if not cursor:
            break

    return feed
```

dagster

Connecting dbt to MotherDuck

Reading right from r2

- S3 connection in dbt for r2
- One line to switch from local development and cloud production.

sources.yml

```yaml
bluesky:
  target: prod
  outputs:
    dev:
      type: duckdb
      schema: bluesky_dev
      path: "local.duckdb"
      threads: 1
      extensions:
        - httpfs
      settings:
        s3_region: "auto"
        s3_access_key_id: "{{ env_var('AWS_ACCESS_KEY_ID') }}"
        s3_secret_access_key: "{{ env_var('AWS_SECRET_ACCESS_KEY') }}"
        s3_endpoint: "{{ env_var('AWS_ENDPOINT_URL') | replace('https://', '') }}"
    prod:
      type: duckdb
      schema: bluesky
      path: "md:prod_bluesky"
      threads: 1
      extensions:
        - httpfs
      settings:
        s3_region: "auto"
        s3_access_key_id: "{{ env_var('AWS_ACCESS_KEY_ID') }}"
        s3_secret_access_key: "{{ env_var('AWS_SECRET_ACCESS_KEY') }}"
        s3_endpoint: "{{ env_var('AWS_ENDPOINT_URL') | replace('https://', '') }}"

```

dagster

Transformation

Reading right from r2

- Upstream Dagster asset mapping
- Blob select Json files

sources.yml

```yaml
version: 2

sources:
  - name: r2_bucket
    meta:
      external_location: "read_ndjson_objects('r2://dagster-demo/atproto_{name}/**/*.json', filename=true)"
    tables:
      - name: actor_feed_snapshot
        description: "external r2 bucket with json files of actor feeds"
      - name: starter_pack_snapshot
        description: "external r2 bucket with json files for feed snapshots"


```

stg_profiles.sql

```sql
select * from {{ source('r2_bucket', 'starter_pack_snapshot') }}
```

dagster

PowerBI

- Business intelligence in your asset graph!
- Clearly map upstream dependencies
- Superior dashboard debugging experience

dagster

End-to-End Lineage and Control

Ingestion & Transformation

Integrate with best in class tools and build custom logic to give stakeholders the data they need when they need it.

Testing & BI

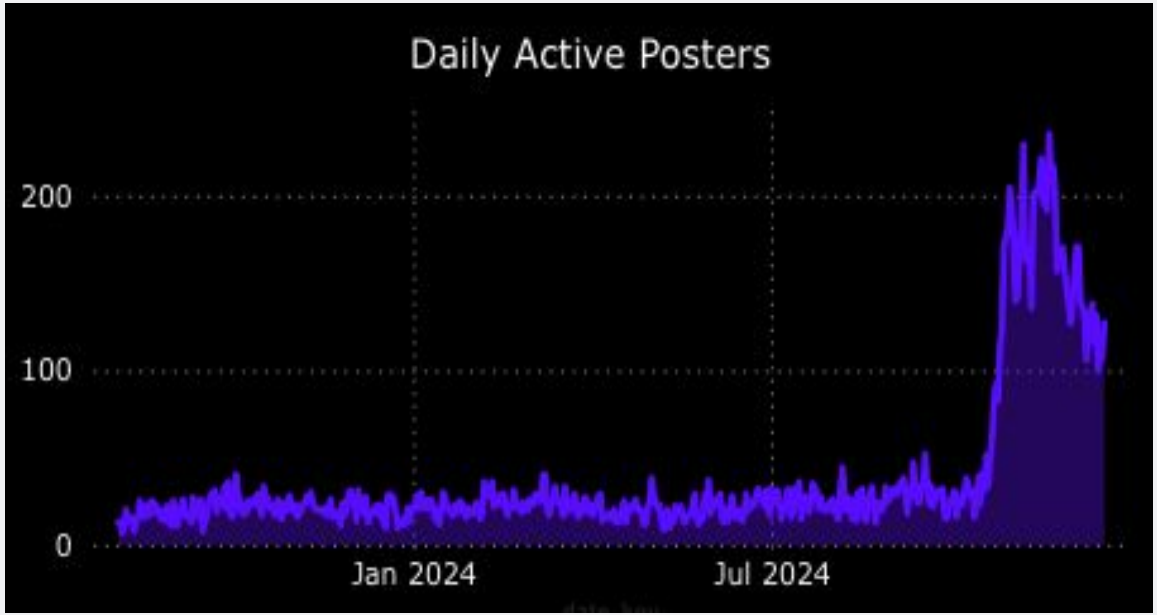
Catch issues before they show up in dashboards while give business intelligence users better insight to data assets and their lineage.

System of Record

The orchestrator touches all systems and processes in your data platforms. Making it the natural tool to build your system of record.

Flash in the Pan?

- Starterpacks
- Keeping the Dream Alive
- Outbound Links!



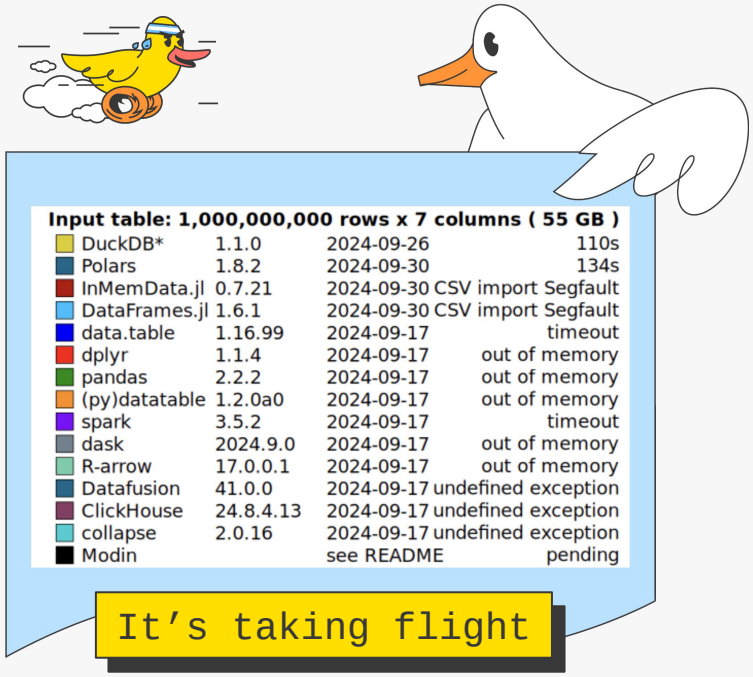
dagster

Fast Feedback Loops: Dagster 🤝 MotherDuck

dagster

What is DuckDB?

DuckDB is an open-source, in-process, SQL, OLAP query engine for insanely fast analytics



- **Efficient by Design:**
  Automatic multi-threading, morsel-driven parallelism, and WebAssembly (Wasm) support

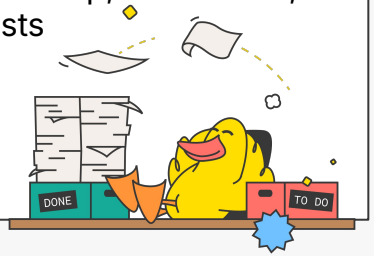
- **Unparalleled Ergonomics:**
  Advanced SQL features, cross-format queries, and geospatial support

- **Dynamic File Format Compatibility:**
  Query CSV, Parquet, JSON, Iceberg, Delta, dataframes, and more — no data movement needed

The MotherDuck Architecture

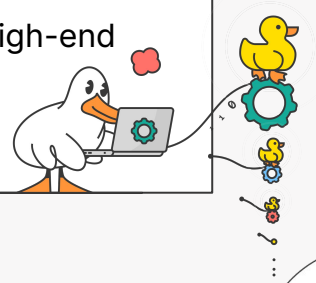
Effortless and Serverless

- Pay only for what you need
- Zero resource or system management
- Sub-100ms cold start response
- No warm up, wind down, or IO costs



Scalably Single Node

- Scale out with dedicated execution environments per user or tenant token
- No more resource contention or noisy neighbor problems
- Efficient network costs
- Optimized for high-end hardware

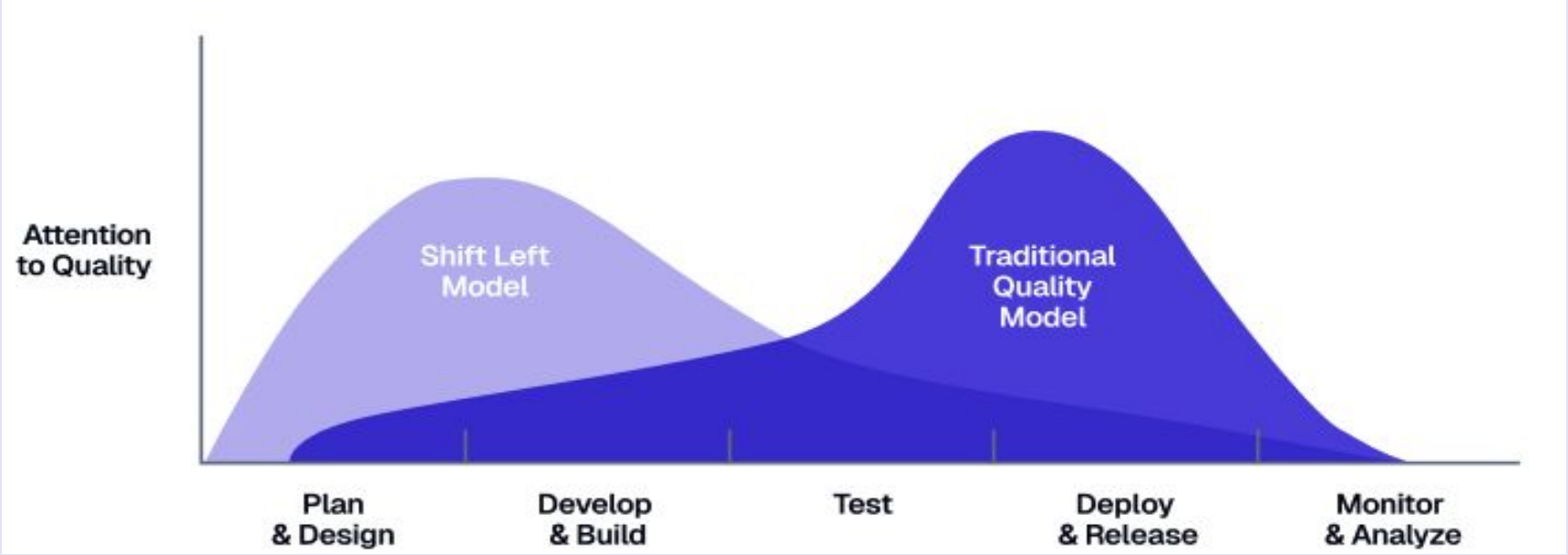


Dual Execution Query Engine

- Run sub-10ms ad-hoc queries locally
- A local environment that exactly matches the cloud
- Efficiency through minimized data movement
- Cache/buffer data locally
- Adaptive query planning executes in the best place *automagically*

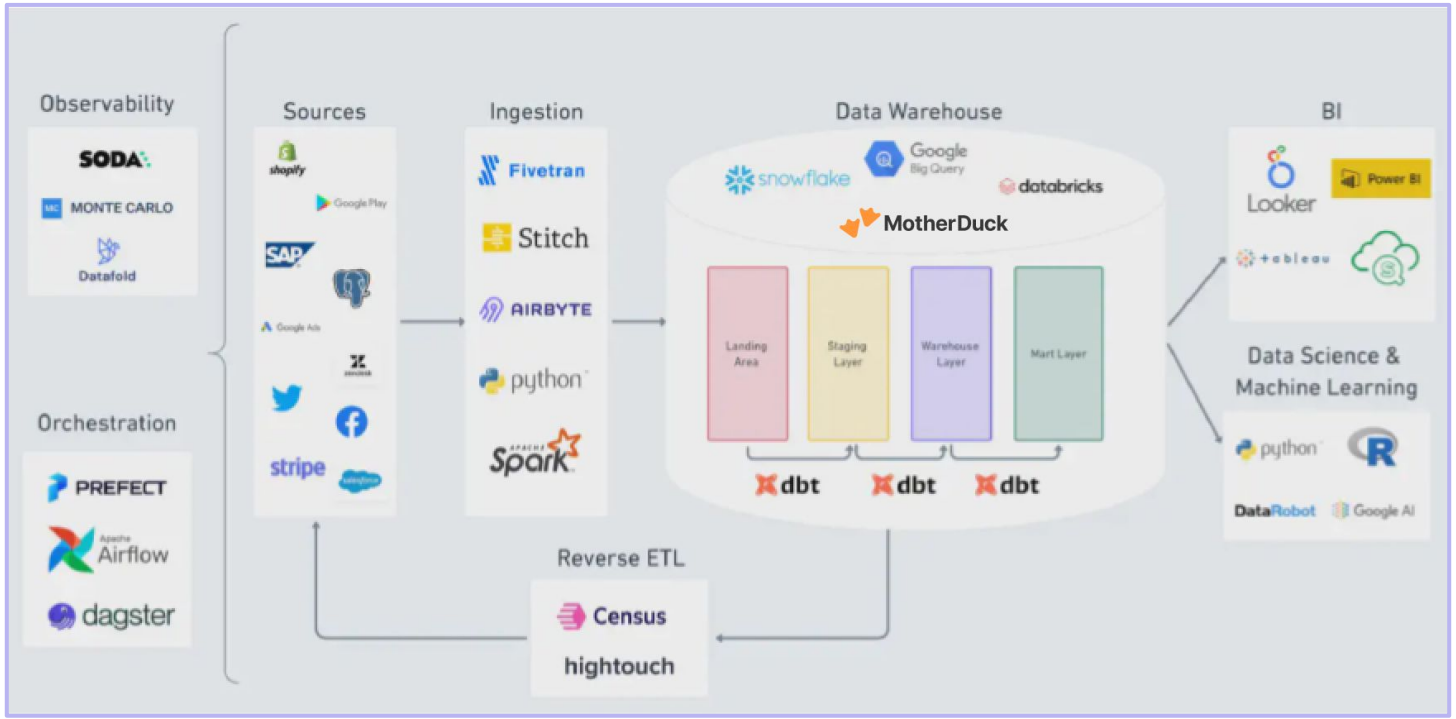
Shifting Left in Data Engineering

Shift Left?

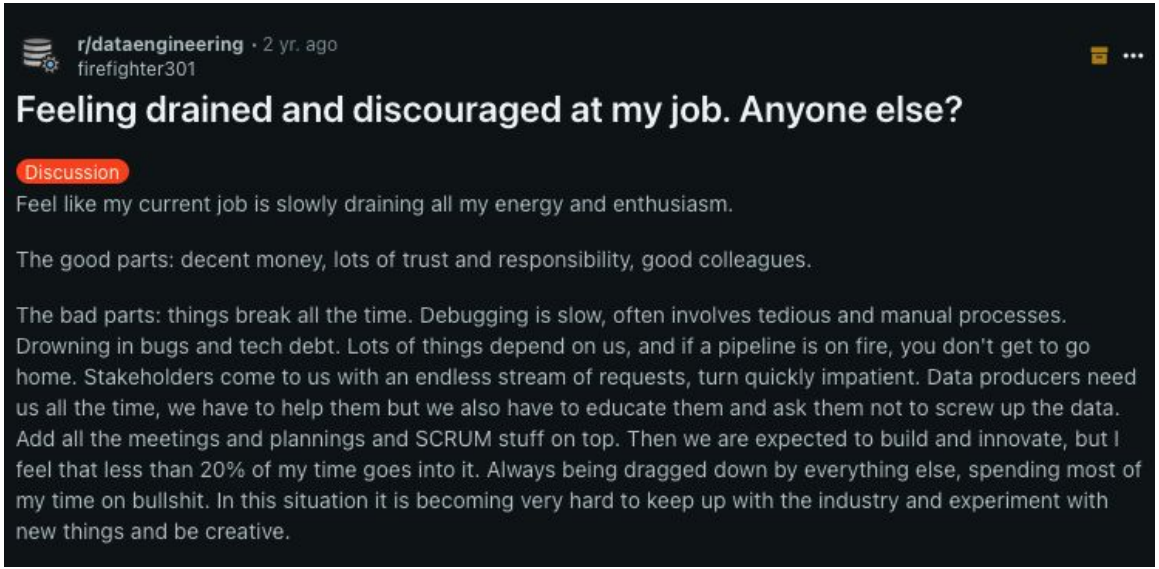


dagster

Platform complexity



Drowning in Reactive Work



r/dataengineering · 2 yr. ago
firefighter301

Feeling drained and discouraged at my job. Anyone else?

Discussion

Feel like my current job is slowly draining all my energy and enthusiasm.

The good parts: decent money, lots of trust and responsibility, good colleagues.

The bad parts: things break all the time. Debugging is slow, often involves tedious and manual processes. Drowning in bugs and tech debt. Lots of things depend on us, and if a pipeline is on fire, you don't get to go home. Stakeholders come to us with an endless stream of requests, turn quickly impatient. Data producers need us all the time, we have to help them but we also have to educate them and ask them not to screw up the data. Add all the meetings and plannings and SCRUM stuff on top. Then we are expected to build and innovate, but I feel that less than 20% of my time goes into it. Always being dragged down by everything else, spending most of my time on bullshit. In this situation it is becoming very hard to keep up with the industry and experiment with new things and be creative.

dagster

The Path Forward

Environment Configuration

Local Development

Making your local as close to prod as possible. No more push and pray

CI/CD

Catching things before they go to prod. Branch deployments using live data in a staging environment.

Access and Permissions

Credentials for key systems and tools. Governance without introducing excess friction.

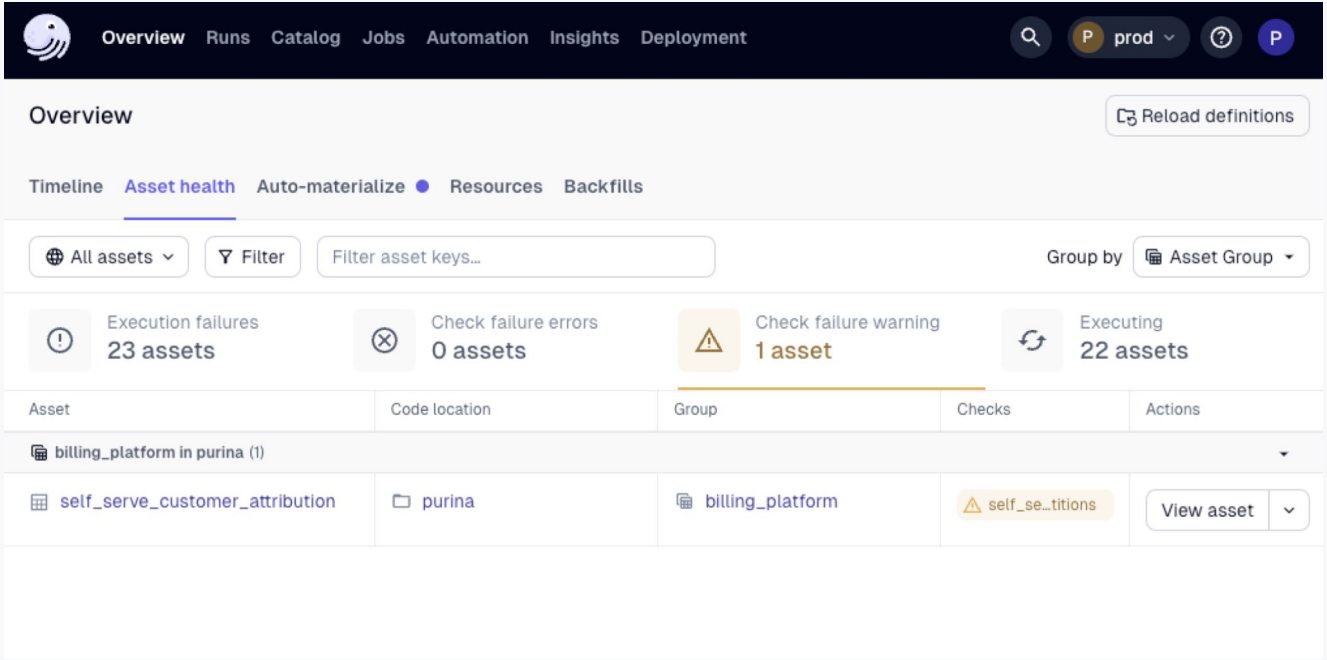
Shift Left with Dagster

dagster

Asset Checks

Dagster OSS

- Catch data quality issues and schema changes early using Python-based logic.
- Freshness checks provide a way to identify data assets that are overdue for an update.
- dbt tests come through as asset checks



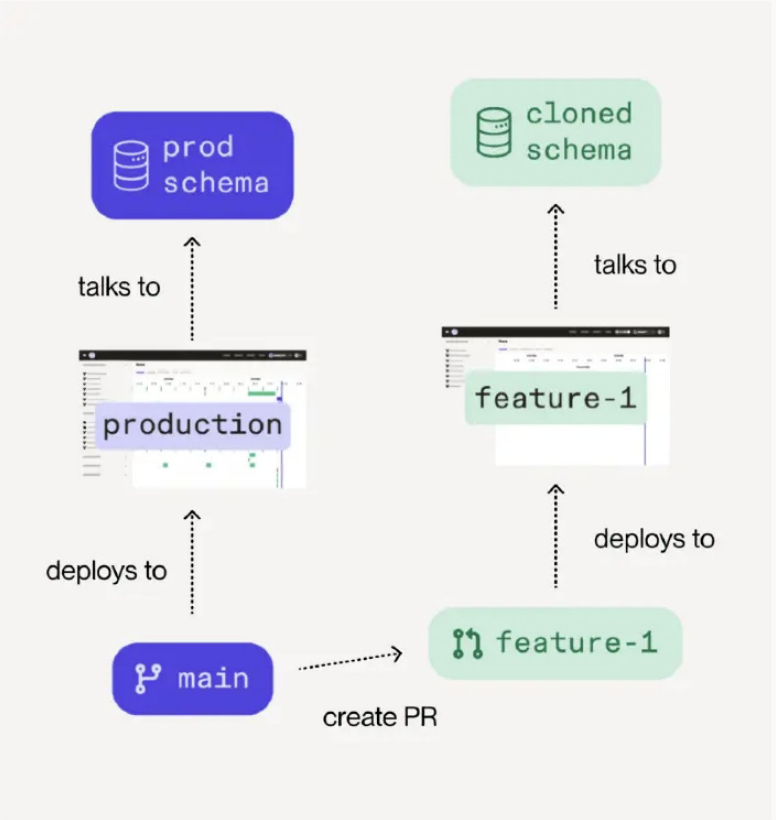
dagster

Branch Deployments

Dagster+

Don’t Test in Production

Every git branch push can automatically create a unique Dagster deployment for a real-time staging preview



dagster

Modern Data Workflows

The brave new world of AI-assisted development

AI Chat Apps

There's still a ton of alpha in using ChatGPT & Claudes chat interface.

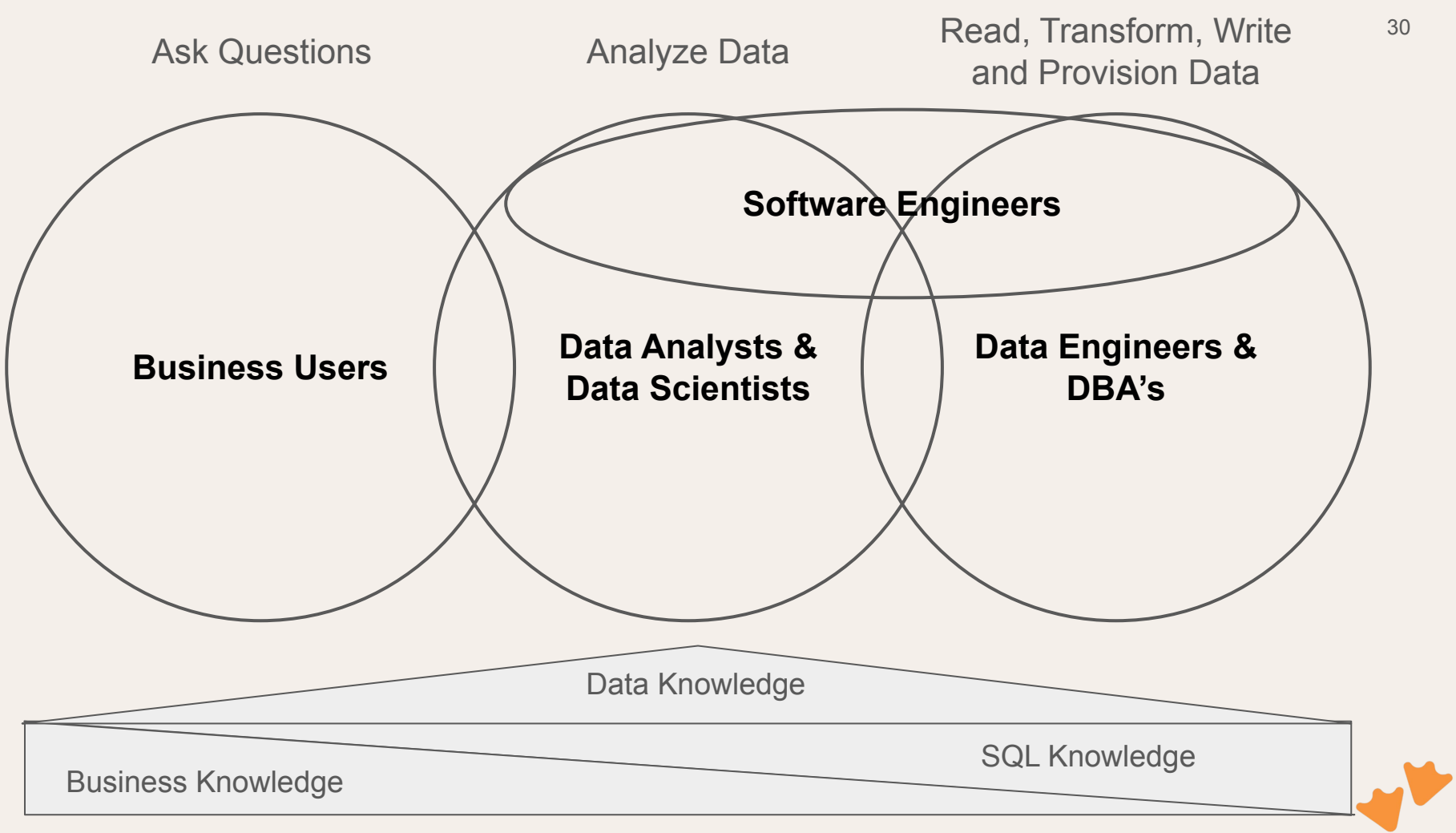
RAGs

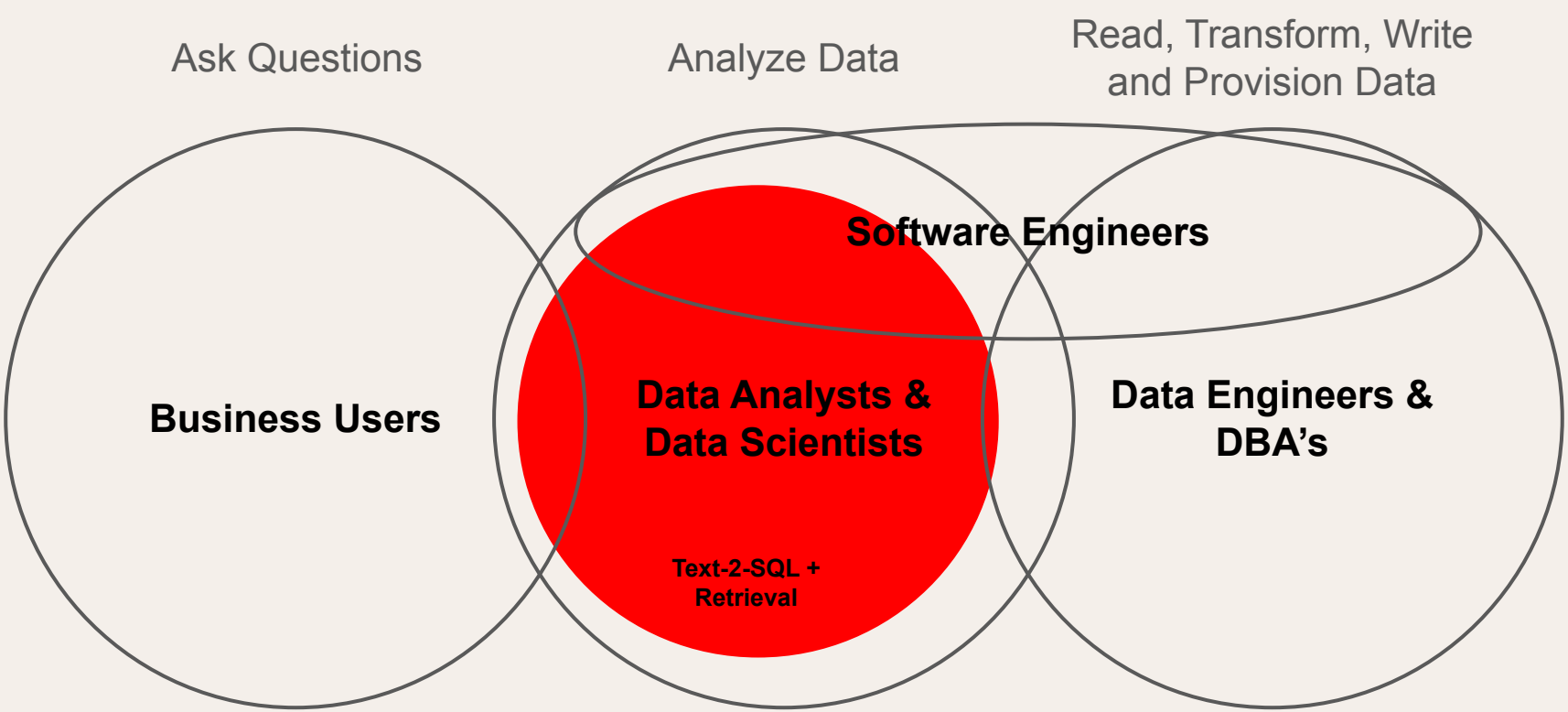
Use embedded search & relevant context to focus models around your specific use case.

AI based IDEs

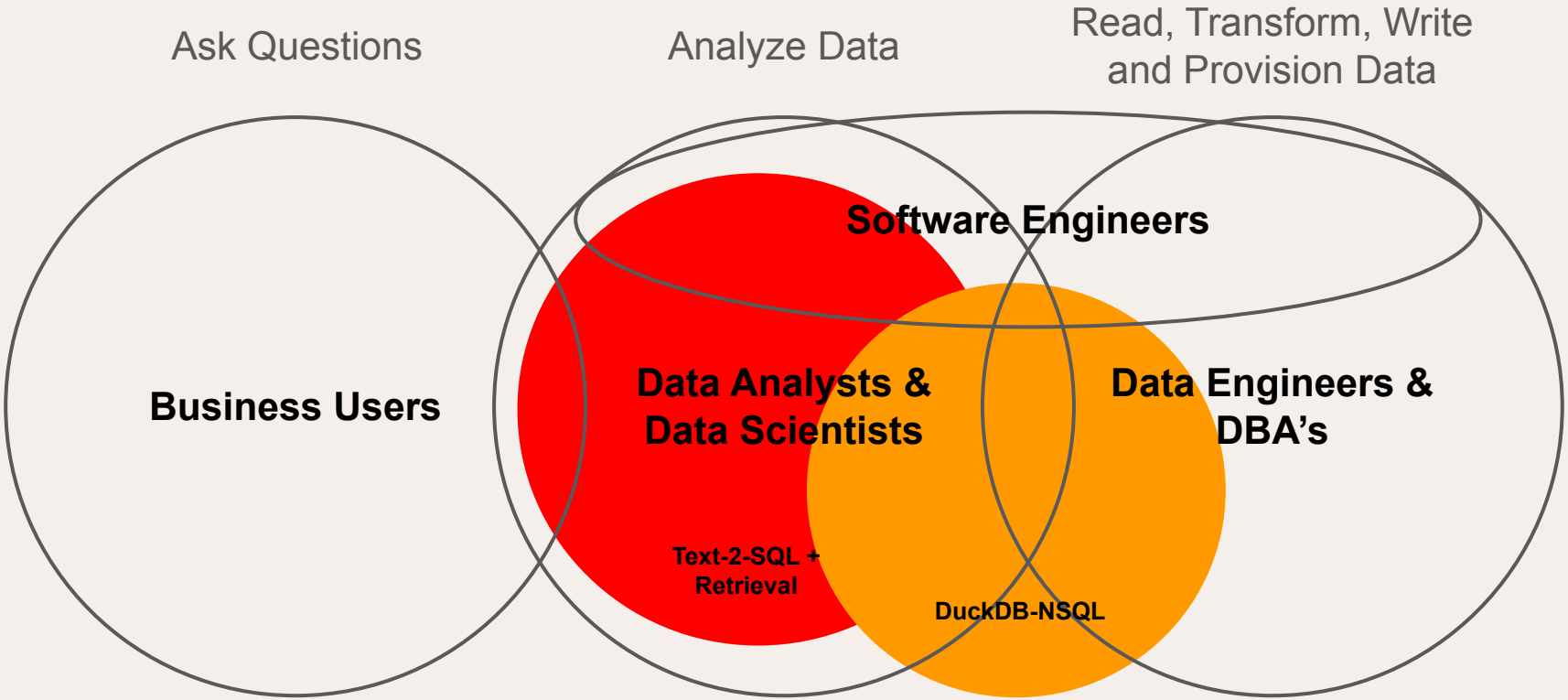
Cursor & windsurf integrate large language models to provide real-time code completion, automated refactoring, and intelligent code generation while you type.

dagster


Ask Questions
Analyze Data
Read, Transform, Write and Provision Data
Software Engineers
Business Users
Data Analysts & Data Scientists
Data Engineers & DBA's
Data Knowledge
SQL Knowledge
Business Knowledge



- Drafts for Analytical Queries
- Reduce Repetitive Work
- Requires Data & SQL Knowledge for Verification

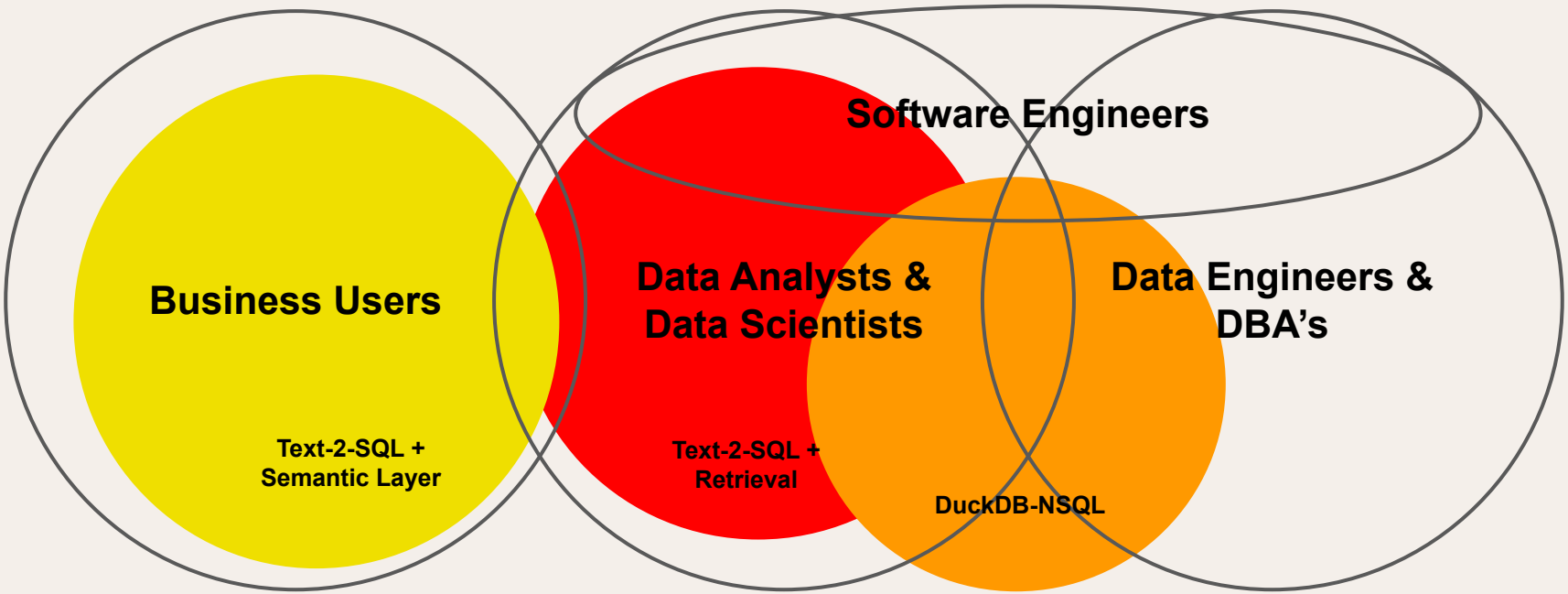


- Simple DuckDB SQL snippets for any type of statements
- Saves round trip to docs

Ask Questions

Analyze Data

Read, Transform, Write and Provision Data



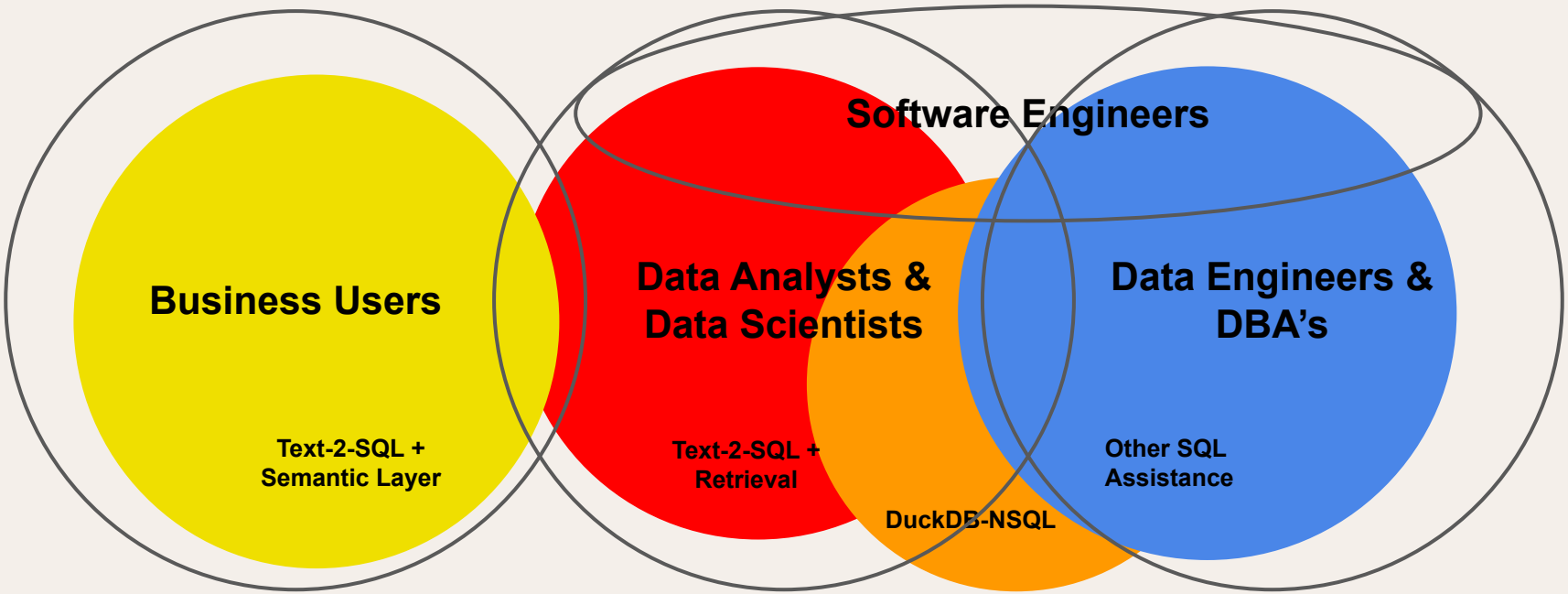
- Built-in Guardrails & Semantic Correctness
- Requires making tribal data knowledge explicit (lot of work!)

- Drafts for Analytical Queries
- Reduce Repetitive Work
- Requires Data & SQL Knowledge for Verification

Ask Questions

Analyze Data

Read, Transform, Write and Provision Data



- Built-in Guardrails & Semantic Correctness
- Requires making tribal data knowledge explicit (lot of work!)

- Drafts for Analytical Queries
- Reduce Repetitive Work
- Requires Data & SQL Knowledge for Verification

- IDE-Integrated
- Support for DDL / DML / ETL-Tasks
- Focus on Performance & Reliability

Upcoming webinar

Building Breakthrough AI Applications with Not Diamond and Dagster

dagster

Q&A

dagster

Thank you!