dagster

Dagster Deep Dive

Rapidly developing data pipelines with dltHub and Dagster

Table of contents

dagster

Who we are

Colton Padden

- DevRel at Dagster
- Background in data and software engineering

Alex Noonan

- DevRel at Dagster
- 4 years as a Data engineer bringing the newest and best tech to SMBs

Who we are

Alena Astrakhantseva

- Data Engineer | DevRel at dltHub: 2+ years
- Celsus: 2+ years - ML Engineer in CV
- Scientist in Applied Math

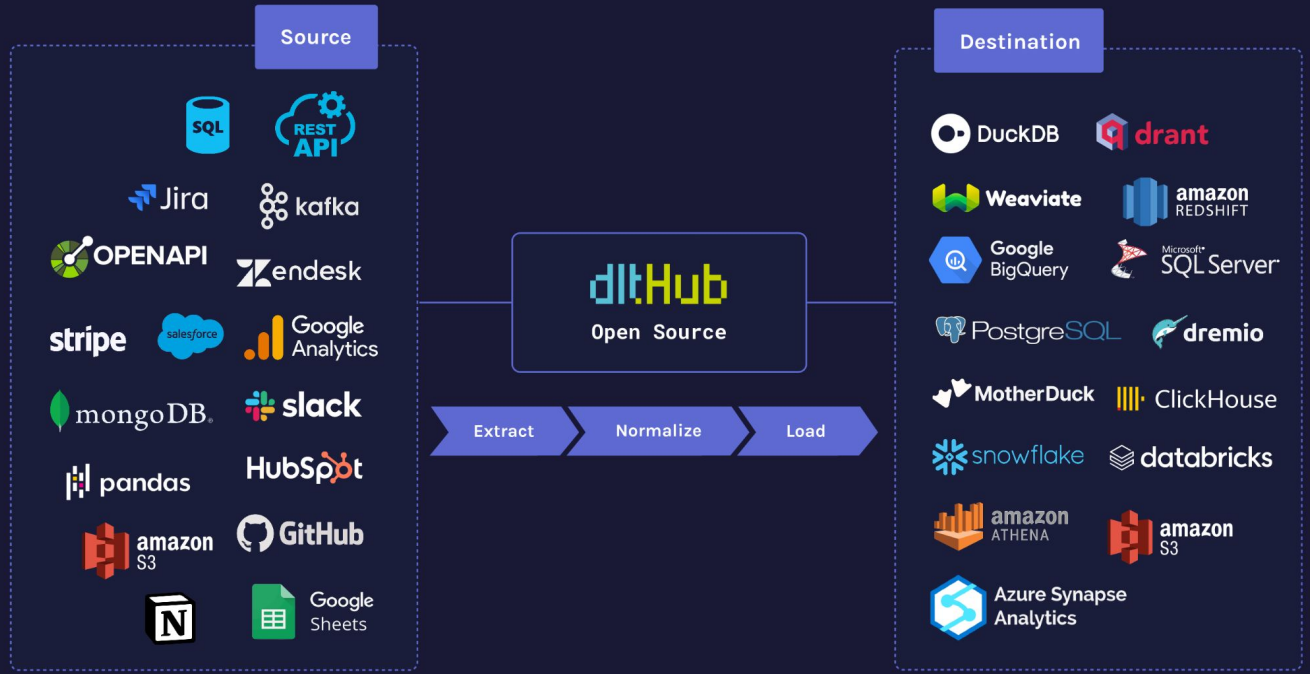
Aashish Nair

- Data Engineer at dltHub
- Optimal Solutions Group: 2+ years - Data Scientist

dltHub

What is dlt?

- dlt is an open-source Python library for data ingestion (ELT)
- automates schema creation, normalization, and data loading



dltHub

What is the solution?

Easy install and set up.

```
>> pip install dlt
```

Easy to use, learning curve is shallow, declarative interface.

```python
import dlt

pipeline = dlt.pipeline(
    pipeline_name="my_pipeline",
    destination="duckdb",
    dataset_name="my_dataset"
)

pipeline.run(my_data, table_name="users")
```

It's Pythonic, you don't have to learn new frameworks or programming languages.

dltHub

The cherry on top

- Run **dlt** anywhere where **Python** runs

- **dlt** is **Open Source**

- Extensive documentation

- Community and Support 4500+ users in Slack

- **Low** costs

- **LLM compatible:** vibe-code your pipelines

Spoiler: On dltHub, you will find 1000+ AI-friendly scaffolding and instructions on how to generate a dlt pipeline using Cursor.

dltHub

The *dlt Dagster* Integration

Blog > Stop Reinventing Orchestration: Embedded ELT in the Orchestrator

Stop Reinventing Orchestration: Embedded ELT in the Orchestrator

Solve data ingestion issues with Dagster's Embedded ELT feature, a lightweight embedded library.

Pedram Navid

https://dagster.io/blog/dagster-embedded-elt

dagster

Integration Goals *(tl;dr)*

- Don’t roll your own ingestion framework
  - Use the tools you love!
- Compose these purpose-built tools from your orchestrator
- Let Dagster handle extraction of assets / metadata / lineage

RFC: Community Input for the Dagster Embedded ELT #17300

PedramNavid started this conversation in **General**

PedramNavid on Oct 18, 2023 Maintainer

Hello!

With the great reception we've seen of our initial launch of `dagster-embedded-elt`, we'd love to get feedback from the community about what they think about our philosophy and approach. We've captured these thoughts in more detail in our blog post, but briefly, we believe that smaller embedded libraries work really well when paired with a powerful orches

We've shipped our initial version using Sling and the [docs] cover the API and code examples.

By not having to reinvent an orchestrator, these libraries can be focused on what makes ingesti fill in the orchestration gaps, such as state management, scheduling, logging, and so on.

Are there particular integrations that you would like us to focus on next? What do you think abo there are other parts of the data lifecycle that would benefit from this as well?

Appreciate it!

↑ 11

dduong1603 on Nov 2, 2023

I think dlt would be a great next integration 😄

↑ 15

Write a reply

dagster

https://github.com/dagster-io/dagster/discussions/17300

colton
@coltonpadden

Now that @dagster 1.7 has landed, I'm excited to share our official integration with @dltHub. With this integration it becomes more simple than ever to leverage the ever growing dlt ecosystem for ingesting data.

4:14 PM · Apr 5, 2024 · **6,501** Views

```
pip install  dagster-dlt
```

(formerly dagster-embedded-elt)

https://docs.dagster.io/integrations/libraries/dlt

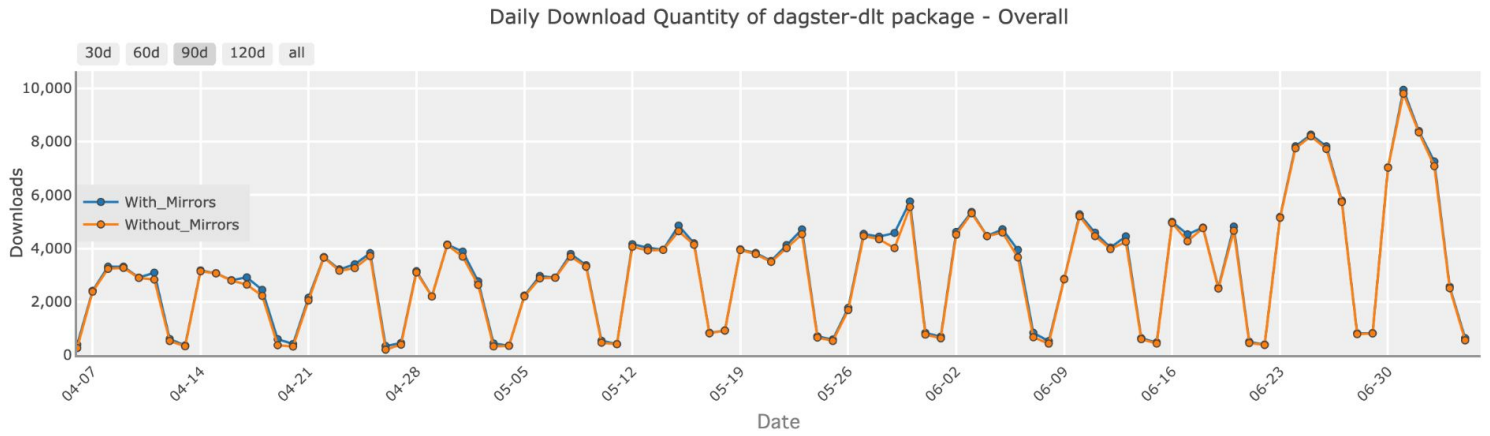
dlt code

```python
@dlt.source
def github_reactions(
    repos: Mapping[str, Sequence[str]],
    access_token: str = dlt.secrets.value,
    items_per_page: int = 100,
    max_items: Optional[int] = None,
) -> Sequence[DltResource]:
    """Get reactions associated with issues, pull requests and comments in the repo `name` with owner `owner`."""
    return (
        dlt.resource(
            get_reactions_data(
                "issues",
                repos,
                access_token,
                items_per_page,
                max_items,
            ),
            name="issues",
            write_disposition="merge",
            primary_key=["repository_name
        ),
```

Dagster code

```python
@dlt_assets(
    dlt_source=github_reactions(
        "dagster-io", "dagster", max_items=250
    ),
    dlt_pipeline=pipeline(
        pipeline_name="github_issues",
        dataset_name="github",
        destination="snowflake",
        progress="log",
    ),
    name="github",
    group_name="github",
)
def dagster_github_assets(context: AssetExecutionContext, dlt: DagsterDltResource):
    yield from dlt.run(context=context)
```

Downloads last day: 812
Downloads last week: 36,155
Downloads last month: 104,924



https://pypistats.org/packages/dagster-dlt

How we use dlt internally at Dagster

dlt usage at Dagster - Buildkite Example

https://dlthub.com/docs/dlt-ecosystem/verified-sources/rest_api/basic

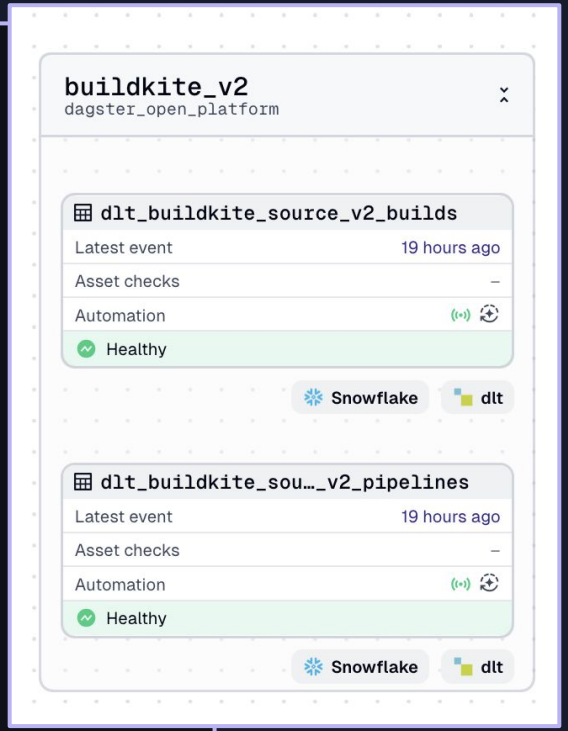
```python
@dlt.source
def buildkite_source_v2(org_slug: str, buildkite_api_token=dlt.secrets.value):
    headers = {"Authorization": f"Bearer {buildkite_api_token}"}

    config: RESTAPIConfig = {
        "client": {
            "base_url": f"https://api.buildkite.com/v2/organizations/{org_slug}",
            "headers": headers,
        },
        "resource_defaults": {
            "primary_key": "id",
            "write_disposition": "merge",
            "endpoint": {
                "params": {
                    "per_page": 100,
                },
            },
        }
```

dagste

dlt usage at Dagster - Buildkite Example

```python
@dlt_assets(
    dlt_source=pipelines(
        org_slug="dagster",
        pipeline_slugs=["internal", "dagster"],
    ),
    dlt_pipeline=pipeline(
        pipeline_name="buildkite_pipelines_internal",
        dataset_name="buildkite",
        destination="snowflake",
        progress="log",
    ),
    name="buildkite",
    group_name="buildkite",
    dagster_dlt_translator=BuildkiteDltTranslator(),
)
def buildkite_assets(context: AssetExecutionContext, dlt: DagsterDltResource):
    yield from dlt.run(context=context)
```

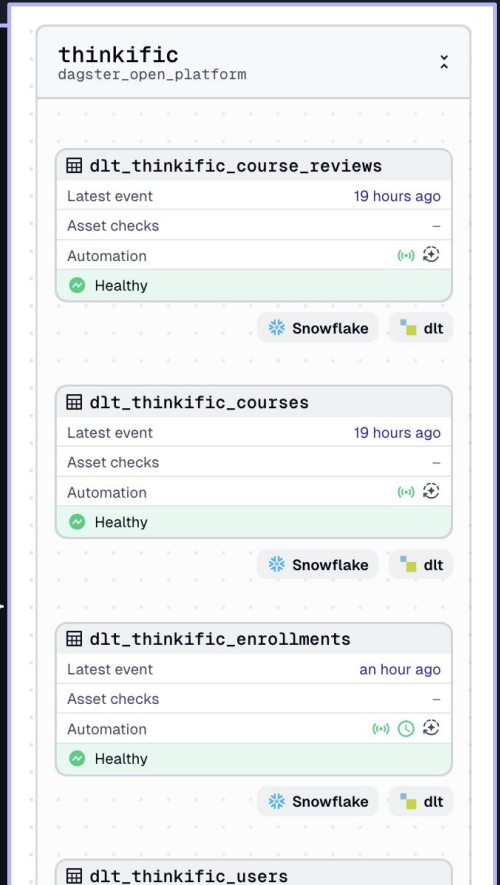


dlt usage at Dagster - Thinkific Example

```python
@dlt.resource(primary_key="id", write_disposition="merge")
def courses():
    response = requests.get(
        url=THINKIFIC_BASE_URL + "courses",
        headers=thinkific_headers,
    )
    response.raise_for_status()
    yield response.json().get("items")

@dlt.transformer(primary_key="id", write_disposition="merge", data_from=courses)
def course_reviews(courses):
    for course in courses:
        yield from _paginate(
            THINKIFIC_BASE_URL + "course_reviews", params={"course_id": course["id"]}
        )

@dlt.resource(primary_key="id", write_disposition="merge")
def enrollments():
    # Enhancement - update to do incremental loads, see:
    # https://dlthub.com/docs/examples/incremental_loading/
```



See all usage in Dagster Open Platform!

https://github.com/dagster-io/dagster-open-platform

Code Demo

https://github.com/dlt-hub/dlthub-education/tree/main/workshops/deep_dive_dagster

1. Build a dlt pipeline for a REST API
    a. Using basic dlt features
2. Expose the pipeline as a Dagster Asset
    a. Using dlt_assets decorator
3. Parallelize the pipeline execution
    a. Run multiple pipelines concurrently

dltHub

Integration Next Steps

- Working with the dlt team on integration improvements
- Components support *(available now)*

```yaml
type: dagster_open_platform.defs.dlt.custom_component.CustomDltLoadCollectionComponent

attributes:
  loads:
    - source: .loads.thinkific_source
      pipeline: .loads.thinkific_pipeline
      translation:
        automation_condition: "{{ daily_not_in_progress }}"
        group_name: "thinkific"
        key: "dlt_{{ resource.source_name }}_{{ resource.name }}"
```

dagster

Next Steps

Experiment

- Try dlt and the `dagster-dlt` integration!

Community

- Join the dlt Slack and Dagster Slack
- GitHub Discussions

Learn

- Check out dltHub's education content!
- Dagster documentation

dagster

Q / A

Thank you!