What is OpenSearch?

Search

Observability

Security Analytics

Visualization

Machine Learning

OpenSearch by the Numbers

More than 600 million	Project downloads
18	New releases since launch
More than 75	Project partners
2,400	Members of public Slack workspace
5,700	Members of user forum
300,000	Forum views per month
113	GitHub repositories
2,000	Non-AWS contributors to project code
43	Non-AWS repository maintainers
1,400,000	Monthly page views for opensearch.org

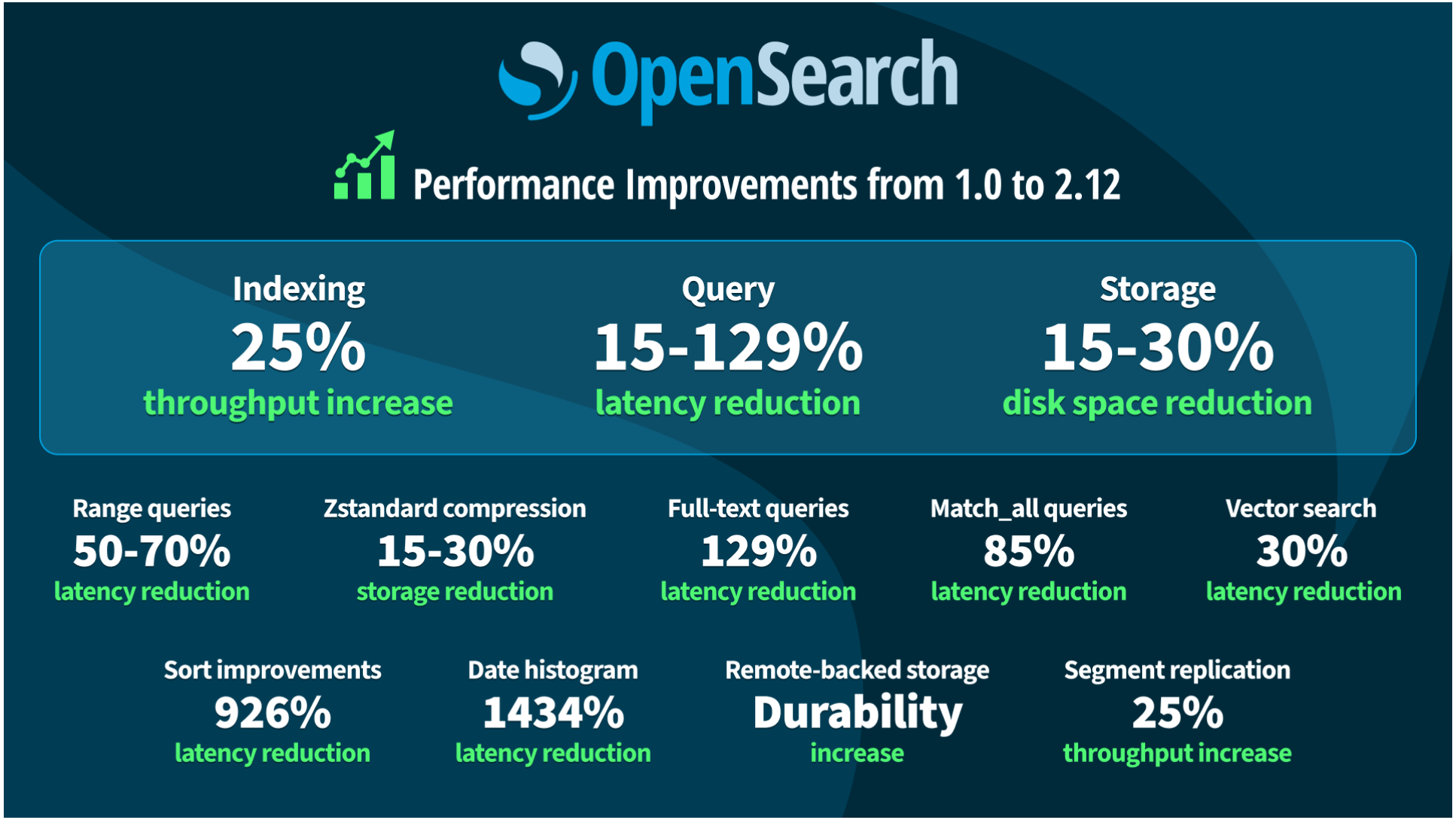
THE LINUX FOUNDATION

OpenSearch Project Vision

The OpenSearch Project provides a flexible, scalable, open-source way to build solutions for data-intensive applications in on-premises, cloud, or hybrid environments. The project is supported by a large and growing community of developers, users, and administrators who use OpenSearch to solve challenges in search, observability, and security.

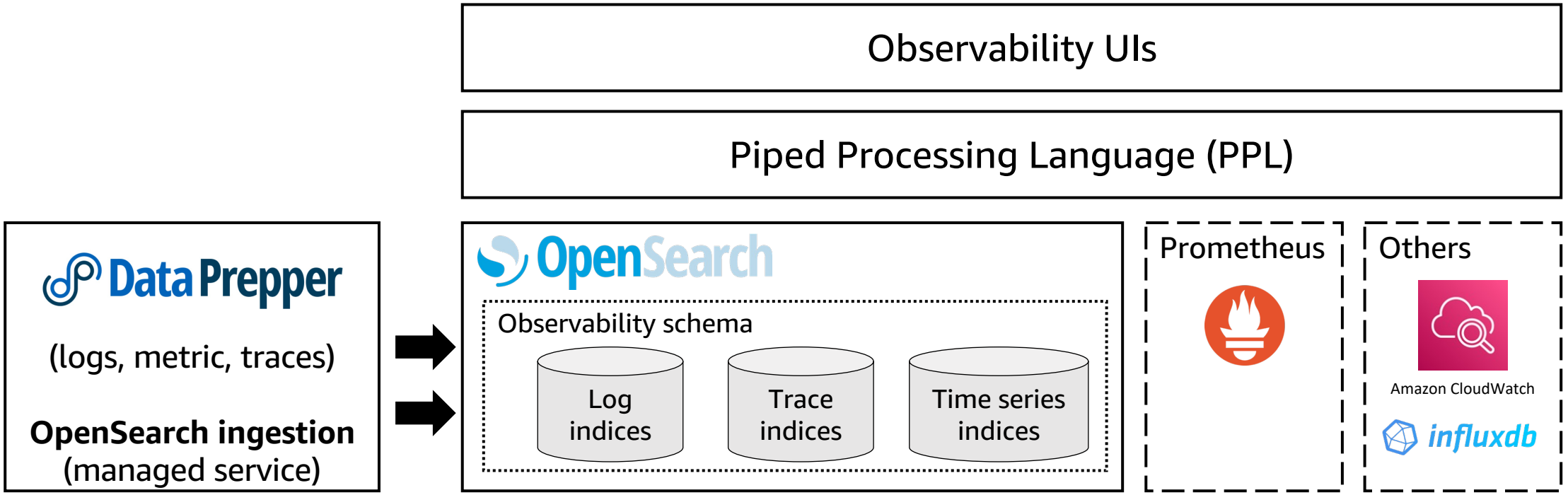
Open-source under the Apache 2.0 license, OpenSearch equips users to explore, enrich, and visualize complex information with built-in performance, developer-friendly tools, and powerful integrations for data processing, machine learning, and more.

Performance is critical



OpenSearch observability

Open source | Open standards | Customizable



Piped Processing language

A multi-data source query language to aid discoverability and visualization

- **Commands**
  - Syntax
  - ad command
  - dedup command
  - describe command
  - show datasources command
  - eval command
  - fields command
  - grok command
  - kmeans command
  - ml command
  - parse command
  - patterns command
  - rename command
  - search command
  - sort command
  - stats command
  - where command
  - head command
  - rare command
  - top command
  - metadata commands

```
source=accounts
| where age > 18
| fields firstname, lastname
```

- **Interfaces**
  - Endpoint
  - Protocol
- **Administration**
  - Plugin Settings
  - Security Settings
  - Monitoring
  - Datasource Settings
  - Prometheus Connector
  - Cross-Cluster Search

- **Functions**
  - Expressions
  - Math Functions
  - Date and Time Functions
  - String Functions
  - Condition Functions
  - Relevance Functions
  - Type Conversion Functions
  - System Functions
- **Optimization**
  - Optimization
- **Language Structure**
  - Identifiers
  - Data Types

```
os> source=accounts | stats count(age) by span(age, 10) as age_span
```

```
os> source=accounts | parse email '.+@(?<host>.+)' | fields email, host ;
```

```
os> source=apache | patterns message | fields message, patterns_field ;
```

https://github.com/opensearch-project/sql/blob/main/docs/user/ppl/index.rst

https://github.com/cncf/tag-observability/tree/main/working-groups/query-standardization/dsl-survey/ppl

PPL is not ‘just’ OpenSearch

- OpenSeach PPL is here
  - https://github.com/opensearch-project/sql/tree/main/ppl
- Spark PPL is here
  - https://github.com/opensearch-project/opensearch-spark/issues/30
- Metrics / PromQL support with PPL is here
  - https://github.com/opensearch-project/sql/issues/561
- PPL based visualizations are here
  - https://github.com/opensearch-project/dashboards-observability
- Language spec is here
  - https://github.com/opensearch-project/piped-processing-language

OpenTelemetry Ingestion

OpenSearch Exporter

- OpenSearch Exporter contrib allows directly submitting telemetry signals into OpenSearch indices that conform with OpenTelemetry protocol and semantic conventions.
  - https://github.com/open-telemetry/opentelemetry-collector-contrib/tree/main/exporter/opensearchexporter

OpenSearch Data-Prepper

- OpenSearch data-prepper pipeline connects directly to Otel collector pipeline and allows additional data processing & transformation such as creation of a services map, additional filtering & sampling to reduce data contention.
  - https://opensearch.org/docs/latest/data-prepper/pipelines/configuration/processors/otel-trace-raw/

OpenTelemetry Analytics

- Trace Analytics
  - Dedicated UX experience based on ingested spans & traces to monitor and detect specific anomalies and potential issues.
    - https://opensearch.org/docs/latest/observing-your-data/trace/ta-dashboards/
- Services Analytics
  - Dedicated UX experience based on services RED indicators (Requests, Errors, Duration) to determine application health
    - https://opensearch.org/docs/latest/observing-your-data/trace/ta-dashboards/#trace-analytics-with-otel-protocol-analytics
- Metrics Analytics
  - A federated UX allowing to investigate collected metrics locally and also query remote Prometheus metrics data-source
    - https://opensearch.org/docs/latest/observing-your-data/prometheusmetrics/
- Logs Analytics
  - Discover and investigate applicative logs including correlation between different signals using common dimensionality

OpenTelemetry Schema & Catalog

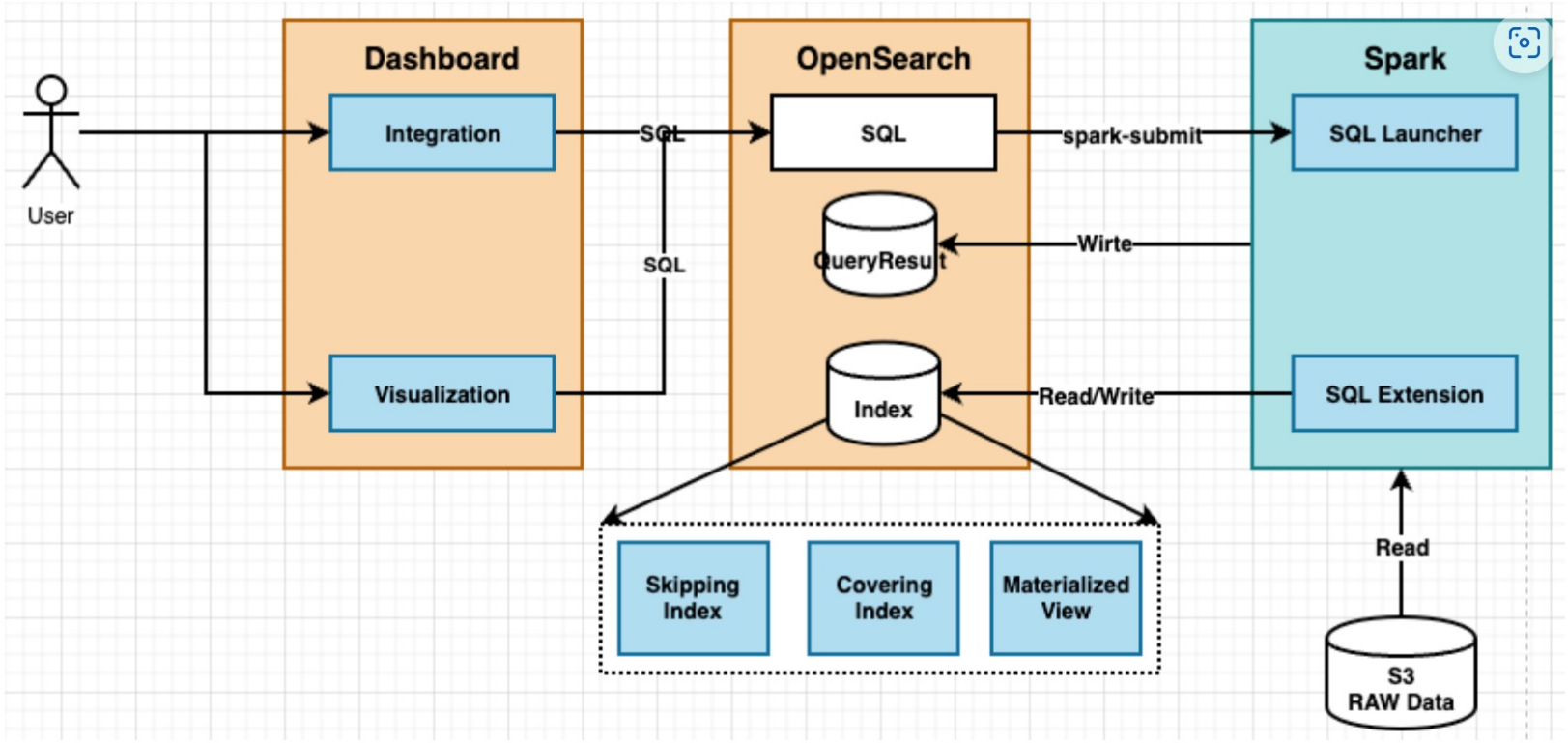
- Confirm with OpenTelemetry Protocol by using a concrete mapping template which follows Otel's semantic convention
  - https://github.com/opensearch-project/opensearch-catalog/tree/main/schema/observability
- Created a unique catalog repository containing opinionated pre-build observability assets that simplify user on-boarding into the observability domain.
  - https://github.com/opensearch-project/opensearch-catalog
- Dedicated visualizations for specific Observability components for better understanding and quickly forming custom Otel based dashboards
  - https://github.com/opensearch-project/opensearch-catalog/tree/main/visualizations

OpenTelemetry Demo

- Forked the Otel-Demo Astronomy webstore showcasing OpenSearch ability to ingest, store and analyze Telemetry signals
  - https://github.com/opensearch-project/opentelemetry-demo
- Attach Data-Prepper ingestion pipelines for services analysis, traces, logs and metrics
- Connecting to external metrics store Prometheus for a federated and consolidated analytics experience
  - https://opensearch.org/docs/latest/observing-your-data/prometheusmetrics/
- Pre-Build Otel-Demo integration dashboard flow based on services RED indicators helping users investigate potential issues.
  - https://opensearch.org/docs/latest/integrations/

Scaling OpenSearch with Apache Spark

- Spark Accelerator framework
  - Enables secondary indices to remote data stores
  - https://github.com/opensearch-project/opensearch-spark



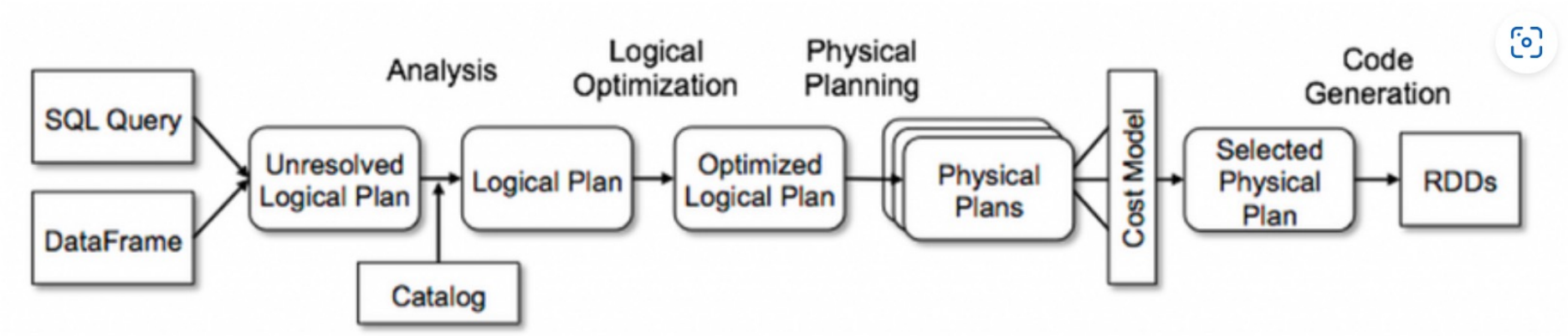
https://github.com/opensearch-project/opensearch-spark/issues/4

Adding PPL to Spark

Using Catalyst Logical Plan Grammar

Another Option for translation would be using the Catalyst Grammar for directly translating the Logical plan steps
Here is an example of such translation outcome:

Our goal would be translating the PPL into the Unresolved logical plan so that the Analysis phase would behave in the similar manner to the SQL originated query.



The following PPL query:

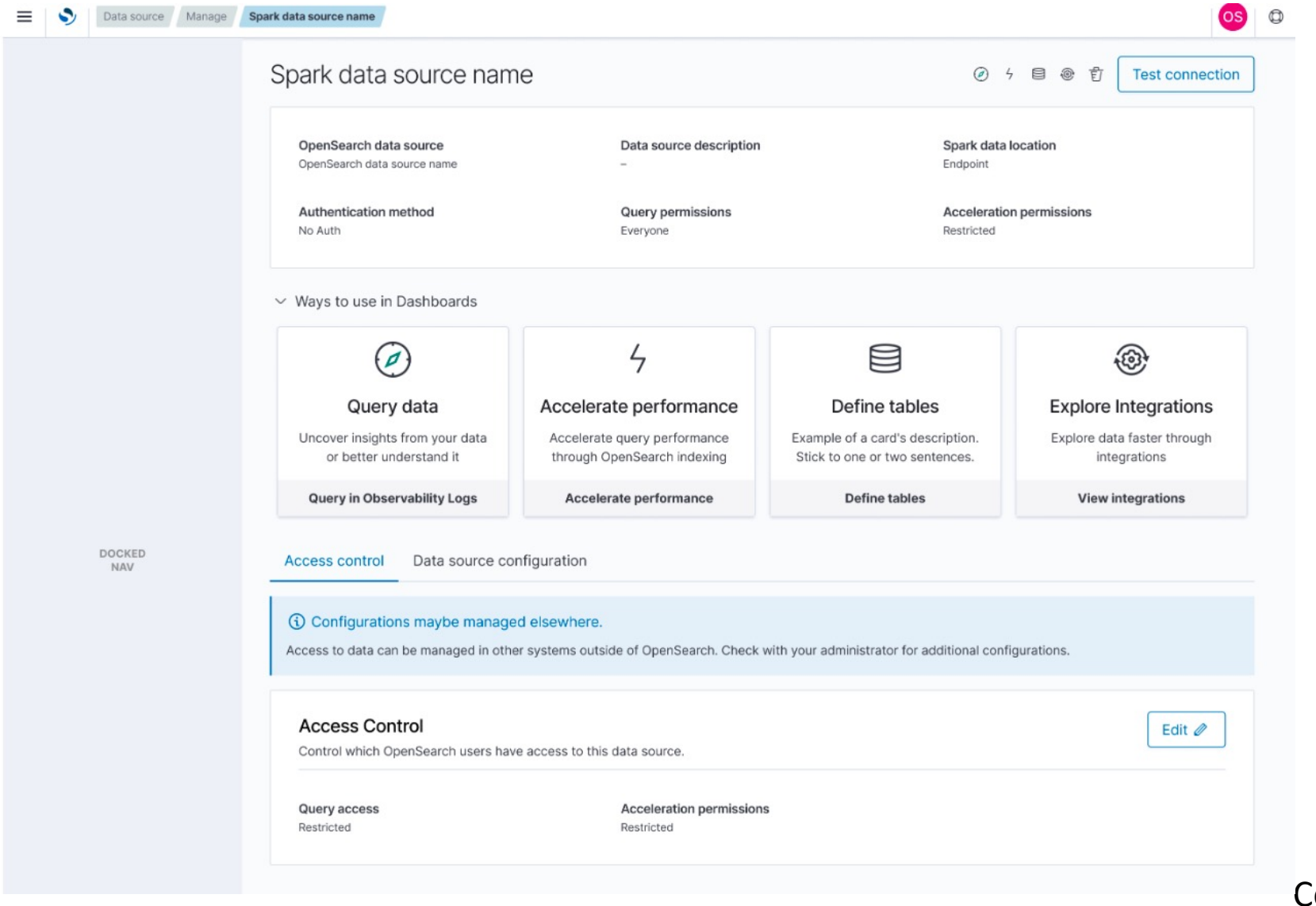
```
search source=t'| where a=1
```

Translates into the PPL logical plan:

```
Relation(tableName=t, alias=null), Compare(operator==, left=Field(field=a, fieldArgs=[]), right=1)
```

https://github.com/opensearch-project/opensearch-spark/issues/30

Spark Data Source in Dashboards



Coming to OpenSearch 2.11