DataLens Data Connectors Reference

Document ID: PRD-DL-020 **Last Updated:** 2024-02-15 **Owner:** DataLens Product Team **Classification:** Public

---

Overview

DataLens connects to 50+ data sources. This reference covers connector setup, configuration, and best practices.

Connector Categories

Databases

Connector	Status	Features
PostgreSQL	GA	Full SQL, SSL, SSH tunnel
MySQL	GA	Full SQL, SSL
Microsoft SQL Server	GA	Full SQL, Windows auth
Oracle	GA	Full SQL, wallet support
MongoDB	GA	Aggregation pipeline
Amazon Redshift	GA	Full SQL, IAM auth
Google BigQuery	GA	Full SQL, service account
Snowflake	GA	Full SQL, key pair auth
ClickHouse	GA	Full SQL
Elasticsearch	GA	Query DSL
Amazon DynamoDB	Beta	Scan and query
Cassandra	Beta	CQL

Cloud Services

Connector	Status	Features
AWS CloudWatch	GA	Metrics and logs
Google Cloud Monitoring	GA	Metrics
Azure Monitor	GA	Metrics and logs
Datadog	GA	Metrics
New Relic	GA	Metrics

Connector	Status	Features
Prometheus	GA	PromQL

Business Applications

Connector	Status	Features
Salesforce	GA	SOQL, reports
HubSpot	GA	API, reports
Google Analytics	GA	GA4 API
Stripe	GA	API
Zendesk	GA	API
Jira	GA	JQL
GitHub	GA	API

Files and APIs

Connector	Status	Features
REST API	GA	Custom endpoints
GraphQL	GA	Custom queries
CSV	GA	Upload or URL
JSON	GA	Upload or URL
Google Sheets	GA	OAuth
Excel	GA	Upload

Database Connectors

PostgreSQL

Connection Settings:

```yaml
type: postgresql
host: your-host.example.com
port: 5432
database: analytics
username: ${POSTGRES_USER}
password: ${POSTGRES_PASSWORD}
ssl_mode: require  # disable, require, verify-full
```

Advanced Options: - SSL certificates - SSH tunnel - Connection pooling - Read replicas

Example Query:

```sql
SELECT
    date_trunc('day', created_at) as day,
    COUNT(*) as orders
FROM orders
WHERE created_at >= NOW() - INTERVAL '30 days'
GROUP BY 1
```

MongoDB

Connection Settings:

```yaml
type: mongodb
connection_string: mongodb+srv://user:pass@cluster.mongodb.net/db
database: analytics
auth_source: admin
```

Example Aggregation:

```
[
    { $match: { status: "completed" } },
    { $group: {
        _id: { $dateToString: { format: "%Y-%m-%d", date: "$created_at" } },
        total: { $sum: "$amount" }
    }},
    { $sort: { _id: 1 } }
]
```

BigQuery

Connection Settings:

```yaml
type: bigquery
project: your-gcp-project
credentials: ${BIGQUERY_SERVICE_ACCOUNT_JSON}
default_dataset: analytics
```

Authentication: 1. Create service account in GCP 2. Grant BigQuery User role 3. Download JSON key 4. Add as secure credential in DataLens

Snowflake

Connection Settings:

```yaml
type: snowflake
account: your-account.us-east-1
warehouse: ANALYTICS_WH
database: ANALYTICS
schema: PUBLIC
username: ${SNOWFLAKE_USER}
password: ${SNOWFLAKE_PASSWORD}
role: ANALYST
```

Key Pair Authentication:

```yaml
type: snowflake
account: your-account.us-east-1
username: ${SNOWFLAKE_USER}
private_key: ${SNOWFLAKE_PRIVATE_KEY}
private_key_passphrase: ${SNOWFLAKE_KEY_PASSPHRASE}
```

Cloud Monitoring Connectors

AWS CloudWatch

Connection Settings:

```yaml
type: cloudwatch
region: us-west-2
access_key_id: ${AWS_ACCESS_KEY}
secret_access_key: ${AWS_SECRET_KEY}
# Or use IAM role
assume_role_arn: arn:aws:iam::123456789:role/DataLensRole
```

Example Metrics Query:

```
namespace: AWS/EC2
metric_name: CPUUtilization
dimensions:
  InstanceId: i-1234567890abcdef0
statistic: Average
period: 300
```

Prometheus

Connection Settings:

```yaml
type: prometheus
url: https://prometheus.example.com
# Optional authentication
username: ${PROMETHEUS_USER}
password: ${PROMETHEUS_PASSWORD}
```

Example PromQL:

```
sum(rate(http_requests_total{status=~"5.."}[5m]))
/
sum(rate(http_requests_total[5m]))
```

Datadog

Connection Settings:

```yaml
type: datadog
api_key: ${DATADOG_API_KEY}
app_key: ${DATADOG_APP_KEY}
site: datadoghq.com  # or datadoghq.eu
```

Example Query:

```
avg:system.cpu.user{host:web-*} by {host}
```

Business Application Connectors

Salesforce

Connection Settings:

```yaml
type: salesforce
auth_type: oauth  # or username_password
instance_url: https://your-org.salesforce.com
client_id: ${SF_CLIENT_ID}
client_secret: ${SF_CLIENT_SECRET}
```

Example SOQL:

```
SELECT
    Account.Name,
    SUM(Amount) TotalValue,
    COUNT(Id) DealCount
FROM Opportunity
WHERE CloseDate = THIS_QUARTER
    AND StageName = 'Closed Won'
GROUP BY Account.Name
ORDER BY TotalValue DESC
```

Google Analytics (GA4)

Connection Settings:

```
type: google_analytics
property_id: "123456789"
credentials: ${GA_SERVICE_ACCOUNT_JSON}
```

Example Report:

```
dimensions:
  - date
  - country
metrics:
  - activeUsers
  - sessions
  - screenPageViews
date_range:
  start: 30daysAgo
  end: today
```

Stripe

Connection Settings:

```
type: stripe
api_key: ${STRIPE_SECRET_KEY}
```

Available Data: - Payments - Subscriptions - Customers - Invoices - Refunds - Disputes

---

API Connectors

REST API

Connection Settings:

```yaml
type: rest_api
base_url: https://api.example.com
authentication:
  type: bearer  # none, basic, bearer, api_key
  token: ${API_TOKEN}
headers:
  X-Custom-Header: value
```

Query Configuration:

```yaml
endpoint: /v1/metrics
method: GET
parameters:
  start_date: ${__from:date}
  end_date: ${__to:date}
response_path: $.data.items
pagination:
  type: cursor
  cursor_param: cursor
  cursor_path: $.next_cursor
```

GraphQL

Connection Settings:

```yaml
type: graphql
endpoint: https://api.example.com/graphql
authentication:
  type: bearer
  token: ${GRAPHQL_TOKEN}
```

Example Query:

```
query GetMetrics($startDate: Date!, $endDate: Date!) {
    metrics(startDate: $startDate, endDate: $endDate) {
        date
        activeUsers
```

```
        revenue
        orders
    }
}
```

Connection Management

Adding a Connection

1. Navigate to **Settings → Data Sources**
2. Click **Add Data Source**
3. Select connector type
4. Enter connection details
5. Test connection
6. Save

Secure Credentials

Store sensitive values as secrets:

```
# Reference secrets in connection
password: ${SECRET_NAME}
api_key: ${ANOTHER_SECRET}
```

Add secrets in **Settings → Secrets**.

Connection Pooling

For high-traffic dashboards:

```
pool:
  min_connections: 5
  max_connections: 20
  idle_timeout: 300
```

Read Replicas

Route queries to read replicas:

```yaml
read_replica:
  host: replica.example.com
  port: 5432
  # Other settings inherited from primary
```

Performance Tips

Query Optimization

1. **Add time filters** - Always filter by dashboard time range
2. **Limit results** - Use LIMIT for large tables
3. **Index usage** - Filter on indexed columns
4. **Aggregations** - Push aggregations to database

Caching

Configure query caching:

```yaml
cache:
  enabled: true
  ttl: 300  # seconds
  max_size: 1000  # entries
```

Query Timeouts

Set appropriate timeouts:

```yaml
timeout:
  connect: 10  # seconds
  query: 120   # seconds
```

Troubleshooting

Connection Failed

Error	Solution
Connection refused	Check host/port, firewall rules
Authentication failed	Verify credentials
SSL error	Check SSL configuration
Timeout	Increase timeout, check network

Query Errors

Error	Solution
Permission denied	Check user permissions
Table not found	Verify schema/database
Syntax error	Check query syntax for dialect
Timeout	Optimize query, add indexes

Data Issues

Issue	Solution
Missing data	Check time zone settings
Wrong values	Verify data types and formatting
Stale data	Check cache settings

Best Practices

1. **Use read replicas** for analytics queries
2. **Create dedicated users** with minimal permissions
3. **Store credentials** as secrets, never in queries
4. **Test queries** before using in dashboards
5. **Monitor query performance** and optimize
6. **Set appropriate timeouts** to avoid hanging queries

Related Documents: Getting Started (PRD-DL-001), Query Language Reference (PRD-DL-010), Dashboard Creation Guide (PRD-DL-005)