DataLens Data Sources Guide

Document ID: PRD-DL-015 **Last Updated:** 2024-02-15 **Owner:** DataLens Product Team **Classification:** Public

---

Overview

DataLens connects to a variety of data sources to power your analytics and visualizations. This guide covers supported data sources, configuration, and best practices.

Supported Data Sources

Databases

Database	Version	Features
PostgreSQL	10+	Full support, SSL
MySQL	5.7+	Full support, SSL
Microsoft SQL Server	2016+	Full support
Oracle	12c+	Full support
MongoDB	4.0+	Read support
Amazon Redshift	Latest	Full support
Google BigQuery	Latest	Full support
Snowflake	Latest	Full support
ClickHouse	20+	Full support

Cloud Services

Service	Features
AWS CloudWatch	Metrics, logs
Azure Monitor	Metrics, logs
Google Cloud Monitoring	Metrics, logs
Datadog	Metrics
New Relic	Metrics
Prometheus	Metrics

Files & APIs

Source	Features
CSV/Excel	File upload
JSON	File upload, API
REST API	Custom endpoints
GraphQL	Query support
Google Sheets	Live connection

NovaTech Products

Product	Data Available
CloudForge	Infrastructure metrics, costs
DevPipeline	Build metrics, test results
SecureVault	Audit logs, access patterns

Adding Data Sources

Via UI

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

Via API

```
from datalens import DataSourceAPI

client = DataSourceAPI()

# Add PostgreSQL data source
datasource = client.create(
    name="Production Database",
    type="postgresql",
    host="postgres.prod.internal",
    port=5432,
    database="analytics",
    user="datalens_reader",
    password="${POSTGRES_PASSWORD}",
    ssl_mode="require"
)
```

Via Configuration File

```yaml
# datasources.yaml
datasources:
  - name: production-db
    type: postgresql
    host: postgres.prod.internal
    port: 5432
    database: analytics
    user: datalens_reader
    password: ${POSTGRES_PASSWORD}
    ssl_mode: require
    max_connections: 10
```

Database Connections

PostgreSQL

```yaml
datasource:
  name: postgres-prod
  type: postgresql
  host: postgres.prod.internal
  port: 5432
  database: analytics
  user: datalens
  password: ${POSTGRES_PASSWORD}
  ssl_mode: require
  options:
    max_connections: 10
    connection_timeout: 30
    statement_timeout: 300000  # 5 minutes
```

MySQL

```yaml
datasource:
  name: mysql-prod
  type: mysql
  host: mysql.prod.internal
  port: 3306
  database: analytics
  user: datalens
  password: ${MYSQL_PASSWORD}
  ssl: true
  options:
    max_connections: 10
```

BigQuery

```yaml
datasource:
  name: bigquery-analytics
  type: bigquery
  project: novatech-analytics
  credentials: ${BIGQUERY_CREDENTIALS_JSON}
  options:
    max_bytes_billed: 10737418240  # 10GB
    location: US
```

Snowflake

```yaml
datasource:
  name: snowflake-warehouse
  type: snowflake
  account: novatech.us-west-2
  warehouse: analytics_wh
  database: analytics
  schema: public
  user: datalens
  password: ${SNOWFLAKE_PASSWORD}
  options:
    role: analytics_reader
```

Redshift

```yaml
datasource:
  name: redshift-cluster
  type: redshift
  host: cluster.xxx.us-west-2.redshift.amazonaws.com
  port: 5439
  database: analytics
  user: datalens
  password: ${REDSHIFT_PASSWORD}
  ssl: true
```

Cloud Monitoring Sources

AWS CloudWatch

```
datasource:
  name: aws-cloudwatch
  type: cloudwatch
  region: us-west-2
  auth:
    type: iam_role
    role_arn: arn:aws:iam::123456789012:role/DataLensRole
  default_namespace: AWS/EC2
```

Prometheus

```
datasource:
  name: prometheus-prod
  type: prometheus
  url: http://prometheus.monitoring:9090
  auth:
    type: basic
    user: datalens
    password: ${PROMETHEUS_PASSWORD}
  options:
    timeout: 30
    query_timeout: 120
```

Datadog

```
datasource:
  name: datadog
  type: datadog
  api_key: ${DATADOG_API_KEY}
  app_key: ${DATADOG_APP_KEY}
  site: datadoghq.com
```

File & API Sources

CSV Upload

```python
# Upload CSV file
datasource = client.upload_file(
    name="Sales Data Q1",
    file_path="sales_q1.csv",
    options={
        "delimiter": ",",
        "header": True,
        "date_columns": ["order_date"],
        "date_format": "%Y-%m-%d"
    }
)
```

REST API

```yaml
datasource:
  name: external-api
  type: rest_api
  base_url: https://api.example.com
  auth:
    type: bearer
    token: ${API_TOKEN}
  endpoints:
    - name: users
      path: /v1/users
      method: GET
      pagination:
        type: offset
        limit_param: limit
        offset_param: offset
    - name: orders
      path: /v1/orders
      method: GET
      params:
        status: completed
```

Google Sheets

```
datasource:
  name: marketing-tracker
  type: google_sheets
  spreadsheet_id: 1abc123xyz
  credentials: ${GOOGLE_CREDENTIALS_JSON}
  sheets:
    - name: Campaign Data
      range: A1:Z1000
      header_row: 1
```

Connection Security

SSL/TLS

Always use encrypted connections:

```
datasource:
  ssl_mode: require  # or verify-full
  ssl_cert: /path/to/client-cert.pem
  ssl_key: /path/to/client-key.pem
  ssl_ca: /path/to/ca-cert.pem
```

SSH Tunneling

For databases behind firewalls:

```
datasource:
  name: private-db
  type: postgresql
  host: private-postgres
  port: 5432
  ssh_tunnel:
    enabled: true
    host: bastion.novatech.com
    port: 22
    user: datalens
    private_key: ${SSH_PRIVATE_KEY}
```

Secret Management

Store credentials securely:

```yaml
# Reference secrets from SecureVault
datasource:
  password: vault:secret/data/datalens/postgres#password

# Or use environment variables
datasource:
  password: ${POSTGRES_PASSWORD}
```

Query Performance

Connection Pooling

```yaml
datasource:
  options:
    min_connections: 2
    max_connections: 10
    connection_timeout: 30
    idle_timeout: 300
```

Query Limits

```yaml
datasource:
  options:
    max_rows: 100000
    statement_timeout: 300000  # milliseconds
    max_concurrent_queries: 5
```

Caching

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

Data Source Health

Health Checks

DataLens automatically monitors data source health: - Connection availability - Query latency - Error rates

Alerts

```
datasource:
  alerts:
    - type: connection_failed
      notify: [ops@novatech.com]
    - type: high_latency
      threshold: 5000  # ms
      notify: [ops@novatech.com]
```

Status Dashboard

View data source health at **Settings** → **Data Sources** → **Health**

Permissions

Read-Only Access

Create dedicated read-only database users:

```
-- PostgreSQL
CREATE USER datalens WITH PASSWORD 'secure_password';
GRANT CONNECT ON DATABASE analytics TO datalens;
GRANT USAGE ON SCHEMA public TO datalens;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO datalens;
ALTER DEFAULT PRIVILEGES IN SCHEMA public
  GRANT SELECT ON TABLES TO datalens;
```

User Permissions

Control who can use data sources:

```yaml
datasource:
  permissions:
    - team: analytics
      access: full
    - team: engineering
      access: read
    - user: admin@novatech.com
      access: admin
```

Best Practices

Performance

1. **Use read replicas** for heavy analytics queries
2. **Create appropriate indexes** for common queries
3. **Limit concurrent queries** to prevent overload
4. **Enable caching** for frequently-run queries
5. **Set query timeouts** to prevent runaway queries

Security

1. **Use dedicated read-only accounts**
2. **Enable SSL/TLS** for all connections
3. **Store credentials in secret management**
4. **Use SSH tunneling** for private networks
5. **Regularly rotate credentials**

Organization

1. **Name data sources clearly** (include environment)
2. **Add descriptions** for discovery
3. **Tag data sources** by team/purpose
4. **Document schemas** for users
5. **Review unused sources** quarterly

Troubleshooting

Connection Failures

1. Verify network connectivity

2. Check firewall rules
3. Verify credentials
4. Check SSL certificates
5. Review database logs

Slow Queries

1. Check query complexity
2. Review index usage
3. Check database load
4. Increase statement timeout
5. Consider caching

Permission Denied

1. Verify user permissions
2. Check schema access
3. Review table permissions
4. Check DataLens user mapping

API Reference

Data Source Endpoints

Endpoint	Method	Description
`/api/datasources`	GET	List data sources
`/api/datasources`	POST	Create data source
`/api/datasources/{id}`	GET	Get data source
`/api/datasources/{id}`	PUT	Update data source
`/api/datasources/{id}`	DELETE	Delete data source
`/api/datasources/{id}/test`	POST	Test connection
`/api/datasources/{id}/schema`	GET	Get schema

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