Continuous RDF and OWL Vocabulary Quality Checks with

LintedData

Jan Martin Keil, Niklas Berndt
German Aerospace Center (DLR), Institute of Data Science
Mälzerstraße 3-5, 07745 Jena, Germany
jan_martin.keil@dlr.de

Poster
https://elib.dlr.de/223803/

Repository
https://gitlab.com/dlr-dw/linteddata/

Goals

- Run RDF and OWL quality checks automatically
- Easy setup and automation

Features

- 60+ checks implemented
- Docker image and executable JAR available
- JUnit XML and Markdown result export

What is LintedData?

LintedData is a linter for RDF. It is a CLI tool, intended for the use in CI/CD pipelines, that checks for common mistakes and compliance to best practices in RDF graphs. The scope of LintedData is domain-independent constrains on RDF graphs, aiming to complement technologies like SHACL or ShEx for domain-specific constraints.

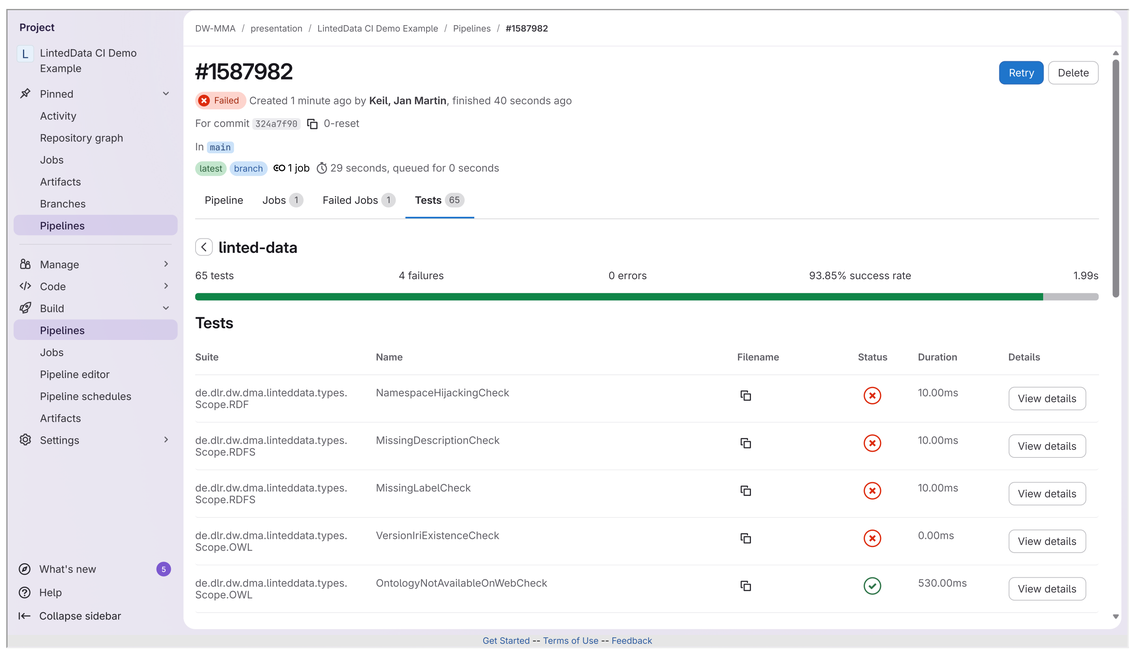
Use in GitLab CI Pipelines

```yaml
linted-data:
  stage: test
  image: dlrdw/linteddata:latest
  script:
    - LintedData <inputFile> --junit linted-data.xml
  artifacts:
    reports:
      junit: linted-data.xml
```

Use in GitHub CI Pipelines

```yaml
jobs:
  linted-data:
    runs-on: ubuntu-latest
    container: dlrdw/linteddata:latest
    steps:
      - name: Checkout Project
        uses: actions/checkout@v6
      - name: Check with LintedData
        run: LintedData <inputFile> --markdown linted-data.md
      - name: Add LintedData results to job summary
        if: ${{ !cancelled() }}
        run: head -qc1023k linted-data.md >> $GITHUB_STEP_SUMMARY
```

Results View in GitLab



What does LintedData check?

- Ontology Pitfalls Catalogue coverage
- OBO Foundry ontology principles support
- Checks for 4 scopes
  - **RDF**, for example
    - `DifferentNamingConventionsCheck`
    - `LeadingOrTrailingSpaceCheck`
  - **RDFS**, for example
    - `MissingLabelCheck`
    - `MultipleRangesCheck`
  - **OWL**, for example
    - `CycleInClassHierarchyCheck`
    - `VersionIriExistenceCheck`
  - **BFO**, for example
    - `ClassNotInBfoHierarchyCheck`
    - `NotReusedRelationsOntologyPropertyCheck`

Configuration

LintedData is intended to only execute meaningful checks automatically, but can be configured using a simple YAML configuration file:

```yaml
checks:
  RDF: false # deactivated scope
  RDFS: true # activated scope
  OWL: # activated scope
    SelfInversePropertyCheck: false # deactivated check
    OntologyIriOboConformanceCheck: true # activated check
    MissingLicenseForOntologyCheck: # activated check
      parameters: # configuration of a check
        permittedLicenses:
          - https://creativecommons.org/licenses/by/3.0/
          - https://creativecommons.org/licenses/by/4.0/
```

The configuration YAML is provided as CLI parameter:

```
LintedData <inputFile> --config <configFile>
```

How to extend LintedData?

LintedData is easy to extend with additional checks:

- Write Java Check Class providing descriptions
- Register Java Check Class
- Write SPARQL Check Query
- Write Java Unit Test Class

DLR