Continuous Integration for XML and RDF Data

Sandro Cirulli
Language Technologist
Oxford University Press (OUP)

6 June 2015
Table of contents

1. Context

2. Continuous Integration with Jenkins

3. Automatic Deployment with Docker

4. Future Work
Oxford University Press

Context

- Oxford University Press (OUP) is a world-renowned dictionary publisher

- OUP launched the Oxford Global Languages (OGL) initiative to **digitize under-represented languages**

- Language data is converted into **XML and RDF**
Where we started from

Challenges

- OUP dictionary data was originally developed for print products
- OUP acquired dictionaries from other publishers in various formats
- Data conversions were performed by freelancers using various programming languages, tools, and development environments
- No testing, no code reuse
Our aim

- Produce lean, machine-interpretable XML and RDF
- Leverage **Semantic Web technologies** for linking and inference
- Convert tens of language resources in a **scalable, maintainable, and cost-effective** manner
Continuous Integration
What it is

- Continuous Integration (CI) is a software development practice where a development team commits their work frequently and each commit is integrated by an automated build tool detecting integration errors.

- CI requires a build server to monitor changes in the code, run tests, build, and notify developers.

- We use Jenkins as it is the most popular open-source CI server.
Continuous Integration
Workflow and components

1. GET
2. PUT
3a. PUT
3b. PUT
4. GET

- Jenkins CI
- eXist-db XML Repository
- JIRA Bug Tracker
- SVN Code Repository
- Graph DB RDF Triple Store
Continuous Integration

Nightly Builds

- Nightly builds are automated builds scheduled on a nightly basis

- We currently builds XML and RDF for 7 datasets

- Nightly builds currently take on average 5 hours on a multi-core Linux machine with 132 GB RAM

- Builds are parallelized using 8 cores
Continuous Integration
Unit Testing

- XSpec for XSLT code
- RDFUnit for RDF data
- XProcspec for XProc pipeline

Test results are converted into JUnit reports via XSLT

Unit tests are run shortly after a developer commits the code
## Continuous Integration

### Monitor View

<table>
<thead>
<tr>
<th>API Docker Build</th>
<th>Data Conversion - British English</th>
<th>Data Conversion - English-Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>#96</td>
<td>6 hours ago</td>
<td>7 minutes ago</td>
</tr>
<tr>
<td>Data Conversion - English-isiZulu</td>
<td>8 hours ago</td>
<td></td>
</tr>
<tr>
<td>#57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Conversion - Spanish-English</td>
<td>8 hours ago</td>
<td></td>
</tr>
<tr>
<td>#7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Conversion - Hindi</td>
<td>16 hours ago</td>
<td></td>
</tr>
<tr>
<td>#58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linked Data Platform Unit Tests</td>
<td>6 hours ago</td>
<td></td>
</tr>
<tr>
<td>#43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical Conversion</td>
<td>10 hours ago</td>
<td></td>
</tr>
<tr>
<td>#548</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical To RDF Conversion</td>
<td>5 hours ago</td>
<td></td>
</tr>
<tr>
<td>#479</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical Conversion Nightly Builds</td>
<td>8 hours ago</td>
<td></td>
</tr>
<tr>
<td>#69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Conversion - Slovenian</td>
<td>9 hours ago</td>
<td></td>
</tr>
<tr>
<td>#56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linked Data Platform Docker Build</td>
<td>6 hours ago</td>
<td></td>
</tr>
<tr>
<td>#45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical Conversion Validation</td>
<td>1 day ago</td>
<td></td>
</tr>
<tr>
<td>#90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical To RDF Validation</td>
<td>1 day ago</td>
<td></td>
</tr>
<tr>
<td>#84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Conversion - Polish</td>
<td>12 hours ago</td>
<td></td>
</tr>
<tr>
<td>#29</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*GraphDB Full Text Indexing: 3 days ago*  
*HTTP Unit Testing: 4 hours ago*  

*Flushed by Mertxell Gonzales: There are 5 unit tests failing*
Continuous Integration

Benefits of CI

- **Code reuse**: on average, 70-80% of the code could be reused for new XML/RDF conversions

- **Code quality**: regression bugs are avoided

- **Bug fixes**: bugs are spotted quickly and fixed more rapidly

- **Automation**: no manual steps, faster and less error-prone build process

- **Integration**: reduced risks, time, and costs for integration with other systems
Continuous Integration

Jenkins Demo
Automatic Deployment with Docker

Docker

- Docker is an open source platform for deploying distributed applications running inside containers

- Docker provides development and operational teams with a shared, consistent environment for development, testing, and release

- Docker avoids the classic 'but it worked on my machine' issue

- Docker allows applications and their dependencies to be moved portably across development and production environments
Automatic Deployment with Docker

Dockerfile

FROM platform_base
MAINTAINER Sandro Cirulli <sandro.cirulli@oup.com>

# eXist-DB version
ENV EXISTDB_VERSION 2.2

# install exist
WORKDIR /tmp
RUN curl -LO http://downloads.sourceforge.net/exist/Stable/${EXISTDB_VERSION}/eXist-db-setup-${EXISTDB_VERSION}.jar
ADD exist-setup.cmd /tmp/exist-setup.cmd

# run command line configuration
RUN expect -f exist-setup.cmd
Automatic Deployment with Docker

Dockerfile (cont.)

```
RUN rm eXist-db-setup-\${EXISTDB_VERSION}.jar exist-
set-up.cmd

# set persistent volume
VOLUME /data/existdb
WORKDIR /opt/exist

# change default port to 8008
RUN sed -i 's/default="8080"/default="8008"/g' tools/
    jetty/etc/jetty.xml

EXPOSE 8008 8443

ENV EXISTDB_HOME /opt/exist

CMD bin/startup.sh
```
Future Work

- **Scalability**: cloud instances to run compute-intensive processes, distribute builds across slave machines

- **Availability**: Circuit Breaker Design Pattern

- **Code coverage**: lack of code coverage tools for XSLT (XSpec and Cakupan are the best we could find)

- **Deployment orchestration**: docker-compose to orchestrate Docker containers
Acknowledgements

The work described here was carried out by a developers team at OUP:

- Khalil Ahmed
- Nick Cross
- Matt Kohl
- and myself
Thank you for your attention!
Any questions?

Slides available at:
www.sandrocirulli.net/xml-london-2015

Contact me at:
sandro.cirulli@oup.com