MANAGING AND KEEPING TRACK OF ALEPH CHANGES
Statsbiblioteket – State and University Library

• University Library - Aarhus University

• Archive - Legal deposit of published material (together with the Royal Danish Library)

• Service provider – ILL star point for the public libraries of Denmark
Organization

- 44 people in the IT division; preservation(16), web(13), operation(15)

- About 250 in total
Aleph

• Aleph pickup location about 20
• About 300km in distance between the larges locations.
• Transit workflow between all pickup locations
• (Only) psychical materials in Aleph
# Some Database Numbers

<table>
<thead>
<tr>
<th>Table</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOL01.z00</td>
<td>3,939,111</td>
<td>Bibliographic</td>
</tr>
<tr>
<td>SOL50.z30</td>
<td>5,023,886</td>
<td>Items</td>
</tr>
<tr>
<td>SOL60.z00</td>
<td>3,241,339</td>
<td>Holdings</td>
</tr>
<tr>
<td>SOL50.Z36+z36h</td>
<td>1,079,604</td>
<td>Loans (2013)</td>
</tr>
<tr>
<td>(z36_loan_date 2013)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
General Setup
Git in Aleph Context

• Live with Aleph 6th of January 2010
• First 5 month with CVS as version control (client/server architecture)
• Initial Git commit 5th of May 2010
• 2 repos per Aleph instance a21_x u21_x
• Most commits are linked to an external issue tracker – first Bugzilla, now Jira
Aleph Setup

- Development (sandbox)
- Stage (quality assurance)
- Snapshot of Yesterday (problem solving)
- Production

Data (pull via RMAN)
Configuration + App. (pull via git)
Git Setup Demo
Getting Started with Git

- Install Git if RHEL then; yum install git
- cd ~/u21_1
- git init
- Create a .gitignore file to ignore some files (things like print files, scratch, logs etc.)
- git add .gitignore xxx01 xxx50 etc.
- git commit -m "Initial commit"
- git clone ~/u21_1 ~/u21_x
Change Request

- Request received from an external partner or in-house.
- Request is prioritized by a person or group depending on the complexity.
- Add request to issue tracker; perhaps split into smaller pieces.
- Configuration changes are added to git in the stage environment.
- User staff do QA; if OK then pull from prod.
Change Request Example

• Reorganize library of public health sub library UIDRA issue aleph-1118
• New sub library XIDRA
• New departments SBMAG, XIDRA
• UIDRA closed until 22/9
• Create UIDRASKR counter
• Create pickup location UIDRA
Git Pull – Change Request Demo

[jve@jve-linux ~]$ ssh aleph@klio
Git Rebase Demo

Author: Appdrift <appdrift@statsbiblioteket.dk>
Date: Sat Sep 13 17:25:46 2014 +0200

jve: issue-111y close SB while at IGELU

commit bb8556275dab35c52624f4a0836ac4d5a0122698
Author: Appdrift <appdrift@statsbiblioteket.dk>
Date: Sat Sep 13 17:23:27 2014 +0200

jve: issue-111x - new igelu sub library

commit 61353e585e80aac6cb64cfa3d66fda1f5ec8ba7b
Author: Appdrift <appdrift@statsbiblioteket.dk>
Date: Sat Sep 13 17:06:28 2014 +0200

jve: aleph-1118 reorganize library of public health

commit bb9d977e7696c847287132a4756cc52d51f2d813
Author: Appdrift <appdrift@statsbiblioteket.dk>
Date: Thu Jan 24 13:45:34 2013 +0100

jve: change to slot 1 config
Broken pipe
aleph@klio(a21_1) SOL50>
Best Practice

• Only rebase if changes have not yet been pulled by others
• Changes in the same file when doing interactive rebase (conflicts)
• Keep the staging period as short as possible
• Get team consensus on the use of Git
• Can be a bit slow with big repositories
Key Benefits

• Great tool for problem solving because changes over time are visible with git log
• Synchronization between systems (environments as similar as possible)
• Discover and handle changes by others (users or Ex Libris support)
• Easy to setup new aleph instance
Problem solving demo
Questions ?
Time for a Break 😊

Thank You!