Making the Move to Open Journal Systems 3: Recommendations for a (mostly) painless upgrade

From June 2017 to August 2018, Scholars Portal, a consortial service of the Ontario Council of University Libraries, upgraded 10 different multi-journal instances of the Open Journal Systems (OJS) 3 software, building expertise on the upgrade process along the way. The final and the largest instance to be upgraded was the University of Toronto Libraries, which hosts over 50 journals. In this article, we will discuss the upgrade planning and process, problems encountered along the way, and some best practices in supporting journal teams through the upgrade on a multi-journal instance. We will also include checklists and technical troubleshooting tips to help institutions make their upgrade as smooth and worry-free as possible. Finally, we will go over post-upgrade support strategies and next steps in making the most out of your transition to OJS 3.

This article will primarily be useful for institutions hosting instances of OJS 2, but those that have already upgraded, or are considering hosting the software, may find the outlined approach to support and testing helpful.

by Mariya Maistrovskaya and Kaitlin Newson

Introduction

While software upgrades are typically not a glamorous or exciting aspect of library technology, they are essential to providing secure and usable applications to our end users. Software upgrades that do not impact the user interface or functionality may go unnoticed by users, but for upgrades that affect user experience or system functionality, special care needs to be taken in communicating these changes to users throughout the process.

In the summer of 2018, the University of Toronto Libraries (UTL) and Scholars Portal (SP) undertook a collaborative project to move from Open Journal Systems version 2 to 3. Open Journal Systems (OJS) is an open-source scholarly journal publishing platform developed by the Public Knowledge Project (PKP) that provides functionality to facilitate peer-review, editorial, and online publishing processes. It also includes various plugins to enhance its functionality, including DOI minting, subscription management, and a variety of themes. The transition from OJS 2 to 3 involved significant changes to the software, with many improvements to the user interface, navigation, and functionality.

Background

The University of Toronto Libraries system is the largest academic library in Canada and supports over 14,000 faculty members and 90,000 students across 3 campuses [1]. In 2005, the UTL’s Information Technology Services launched a journal hosting service on a locally hosted instance of OJS to support open access publishing among faculty, students, and staff. In 2014, the OJS instance moved to Scholars Portal’s shared hosting service, with the UTL staff providing user support, troubleshooting, and training to the journal teams.

The UTL is one of 21 university libraries that make up the Ontario Council of University Libraries (OCUL), an academic library consortium which serves almost 500,000 students [2]. Scholars Portal is the technology service arm of OCUL and provides a wide range of services to Ontario’s academic community. Both OCUL and the UTL support PKP development activities, with OCUL as a Development Partner and UTL a financial supporter. Both co-authors are also involved in the PKP Documentation Interest Group.

Since 2012 Scholars Portal has offered an OJS hosting service, which is used by 10 universities within the OCUL consortium. Across all OJS instances hosted at Scholars Portal, there are approximately 125 actively publishing journals, containing over 30,000 published articles. The UTL instance is by far the largest, with 60 journals, including 37 actively publishing, and over 17,000 published articles across all journals as of December 1, 2018.

Scholars Portal handles all updates to OJS for each instance, manages servers and hosting, installs new plugins, and troubleshoots technical issues. They also offer custom journal design services. The OJS hosting service at Scholars Portal is run by two librarians, along with support from the Systems Support team.

Upgrade decision

OJS 3.0 was released in 2016 after almost 10 years of development, improvement, and thorough UX testing. It came as a welcome change from the version 2 that many editors found difficult to navigate and that limited the ability to customize individual journals on a multi-journal instance without affecting the default templates.

At the consortial level, Scholars Portal upgraded 10 instances from OJS 2 to 3 from June 2017 to August 2018. Each institution set their own upgrade date based on a number of factors, including publishing schedules and upgrade interest from journal editors. Our first upgrade came from an instance that was keen to make the move, and they were upgraded to version 3.0. As is common with early adoption of new software, earlier upgrades encountered bugs more frequently and required more staff time to support, research, and resolve these issues. As we made our way through each upgrade, they gradually became easier, with fewer bugs found or users encountering known issues that could be resolved quickly.

Despite having a few keen journal teams interested in the upgrade, the UTL decided to wait until version 3.1 was released, which added key missing features from the first release, such as the subscription module used by our moving paywall journals. At the time of the OJS 3 upgrade in the summer of 2018, the UTL instance had 57 faculty and student journals, of which 36 were actively publishing. Considering the number of journals the upgrade would impact, we had to be careful and thorough in our preparation and communication throughout the process.

Upgrade preparation & testing

Below we outline the steps in pre-upgrade planning, preparation, testing, and outreach that are summarized in checklist points at the end of each section. The full checklist is available in Appendix A.

Assessing your upgrade readiness

Despite the excitement around the new release, two things to watch out for are:

1. whether it includes the critical functionality your users depend on, and
2. whether the release is sufficiently tested and bug-free for roll out.
Checklist item #1: Work with your journals to identify their essential features and make sure they are available in the new version

Cleaning up your content

At the time of the OJS 3.1 release, the UTL instance had 24 extra journals that had moved elsewhere, ceased publishing, or never launched after the initial setup. The UTL staff took this opportunity to reconnect with journal teams and archive the journals that ceased publishing indefinitely. Our archiving strategy included:

- depositing articles into Scholars Portal Journals for continued open access, with proper URL redirects set up in the web server configuration;
- capturing journals’ websites via the Internet Archive’s Wayback Machine; and
- revising our Service Agreement to include an archiving provision for inactive journals going forward.

The upgrade from OJS 2 to 3 is also an opportunity to clean up any users that may be spam users, which is a common issue in OJS, particularly for instances that existed before PKP implemented reCAPTCHA on account registration. There are several ways to identify these users, but one option is through the email domains used when the user signed up. Using the following SQL query, you or your system administrator can identify the domains in order to identify possible spam users:

```
SELECT substring_index(email, '@', -1) domain, COUNT(*) email_count
FROM users
GROUP BY substring_index(email, '@', -1)
ORDER BY email_count DESC;
```

With this list of usernames, you can then use OJS’s built-in user merging tool (in your OJS directory at tools/mergeUsers.php) to clean up users. We created a small bash script to do this, and PKP also has a process that they recommend. You’ll need to create a user account that all accounts can be merged into if one doesn’t already exist. Keep in mind that while it’s nearly impossible to find all the spam users in a large instance of OJS, you may be able to significantly clean these up, thereby reducing the amount of data in your instance and making it easier for journal teams to manage their users.

- Checklist item #2: Consider cleaning up the content you don’t need to carry over, including inactive journals, spam users, etc.

Setting upgrade timelines

The time you are able to devote to planning and preparing for the upgrade will depend on a number of factors, including your department’s schedule and staff availability, the size of your instance and customizations to carry over, whether you will have demo instances for your journals to test, and whether you are planning on delivering pre- and/or post-upgrade training.

At UTL we gave our journals 3 months to test out their sandboxes, report issues, request theme migration assistance, and receive training, consultations, and resources. As journals encountered any issues, we reported these to Scholars Portal. While several journals took advantage of these resources, some did not — and subsequently required significant (and more time-sensitive) assistance after the upgrade. The bottom line is that setting aside enough time and support resources post-upgrade is just as important as pre-upgrade preparation.

- Checklist item #3: Decide if you will provide sandbox sites to journals for testing (or use PKP’s OJS test-drive instance)
- Checklist item #4: Set aside time BEFORE and AFTER the upgrade for testing, training & troubleshooting

Communication and training

As with any major change, advance communication with clear timelines and expectations is key. The first step is to ensure you have updated contact information and re-establish communication with any journals that may have had editorial changes. Announcing your upgrade plans early (regardless of whether you have the exact date scheduled) will go a long way to set expectations. If the upgrade involves downtime or a freeze for new content, make sure to request email receipt confirmation once the dates are announced. At UTL, we used a combination of our mailing lists for hosted journals and personal emails to journal editors for upgrade-related communication.

Training and/or one-on-one drop-in opportunities for journal teams may be another helpful way of announcing the upgrade, communicating your support, and resolving issues. Be prepared for some journal teams not to take advantage of these opportunities until after the upgrade happens, at which point the learning curve may be steeper and the support need more urgent.

At UTL, we offered an OJS 3 workshop 3 months before the upgrade and sent out information about journal sandboxes that were available for testing. Fifteen editors attended the 2-hour workshop from 13 journals (36% of our active journals), and these journals went on to be mostly self-sufficient throughout the upgrade. Because finding another session time that would work for the rest was challenging, we opted to offer drop-in hours for one-on-one training and consultations. By contrast, multiple drop-in sessions had only 1 attendee, with most editors opting for email communication.

- Checklist item #5: Check that your contact information for hosted journals is up-to-date and re-establish contact in case of editorial changes.
- Checklist item #6: Announce plans to upgrade early, even if you don’t have exact timelines yet.
- Checklist item #7: Set downtime for production upgrade – and request confirmation.
- Checklist item #8: Set content freeze dates if necessary.
Checklist item #9: Align your training/consultation offerings with demand – your editors may prefer emails or a call over workshops and drop-ins.

Defining the level of support

Support services and resources vary in institutions hosting multiple journals. The main change that users will experience in the upgrade is to the user interface, and effort is required to carry over any design or layout changes made in OJS 2. An institution may choose to only perform the upgrade and style its base homepage while letting journals do their own customizations, or it may be able to offer more hands-on support with theme migrations.

Scholars Portal provided assistance with customized theme migrations to institutions on a case-by-case basis, depending on the local resources available at each institution, the comfort of the institution in doing that work themselves, and the complexity of the original themes. The UTL supported its hosted journals with theme migrations on the following conditions:

- Support is only provided to active journals
- Requests have to be made before the cut-off date (one month before the upgrade), so we have enough time to do the styling
- Support requests are responded to on a first-come, first-served basis
- If there is no response to our offer of support, we assume that the journals will do their own styling after migration

Additionally, training sessions, individual drop-ins, and email support were offered to empower journal teams to do their own setup. In the end, we received requests for assistance from 4 journals (10% of our active journals). Two more identified the need for assistance after the upgrade. The rest either did their own styling or chose to go with the default theme. Overall, this change was not as drastic or disruptive as we had originally expected, and most journals were able to handle it on their own.

- Checklist item #10: Define your level of support and communicate it clearly.
- Checklist item #11: Set the cut-off date for pre-upgrade support requests.

Setting up your test instance

Each upgrade began with a testing environment that was upgraded from the current OJS 2 instance. This provided a space for journal managers to gain familiarity with the new system, ask questions about any changes in functionality, and complete any necessary customization.

While it may seem obvious to those of us that manage software, it bears repeating – create backups! Back up your submission files, your OJS system code, and your database consistently. This will save you a lot of time and headaches throughout the upgrade process. In particular, it can be helpful to back up your instance when it’s ready for the upgrade script to be run so you can quickly resume the upgrade once you resolve any issues that the script encountered.

As you work through any issues encountered during your test upgrades, PKP’s forums, GitHub issues, and administrator’s guide are essential resources. Older OJS instances in particular may have database encoding issues that need to be resolved before an upgrade will work successfully. After some research and troubleshooting on the PKP forums, we found a process to resolve this issue.

Once you’ve successfully upgraded your first test instance, there are a few settings you may wish to configure. First, disable any plugins within OJS that send data out to other services or systems, such as the Crossref plugin and the Acron plugin. We also recommend replacing all user email addresses in your database’s ‘users’ table so that users aren’t receiving emails. PKP has also recommended this in their upgrade guide [3]. The following database query can be used to set all user emails to mailinator addresses, which you could replace with any preferred domain.

```sql
UPDATE users SET email=CONCAT(username, "@mailinator.com");
```

If users need to have the correct email set in the testing instance later, then you can update the email in the test database via their account management pages in the user interface; for instance, if they need to reset their account password or want to test email notifications within the system.

- Checklist item #12: Create frequent backups of your code, database, and files as you work through each upgrade.
- Checklist item #13: When setting up your testing instance, disable any plugins that may send data out to other services or systems.
- Checklist item #14: Consult the PKP forums and open GitHub issues when troubleshooting problems.

Testing and feedback

After Scholars Portal (SP) set up the UTL test instance, every journal received a notice with their individual URL, user credentials, and testing instructions (see Appendix C). It was made clear that no changes made in the sandbox would be carried over to the upgraded instance. Journal teams were encouraged to get familiar with the system, test and tweak the new look, and report any missing functionality or unexpected behaviour.

As users found any potential issues or changes they’d like to make, we ensured that these were documented appropriately so they could be replicated in our production upgrade. We did so via our Confluence and JIRA platforms. As the administrator of the OJS instance, you may also want to monitor your server logs for any errors that users encounter as they are testing the new instance.

Only about 10% of editors contacted us about functionality that was missing or changed (e.g. thesis abstract plugin, public folder plugin, etc.). Some also kindly reported the plugins they relied on that worked fine. Most reports about changes or missing features came after the upgrade. There may not be much you can do about it (other than submitting a feature request on PKP forum), but providing a venue for journals to report these issues covers your bases.

- Checklist item #15: Put together an easy to follow test plan for the journal teams to test and report issues.

Redesigning your homepage

While most OJS 3 changes affect individual journal websites, the platform upgrade may be an opportunity to freshen up your OJS homepage, particularly if you use it as a front end for your journal publishing service.

At UTL we took advantage of this opportunity to redesign the homepage, for which we did a round of user testing with wireframes and subsequent modifications based on feedback. We also set up the default template that would identify the library as the host in the footer of all journals on the instance.
Figure 2. UTL’s old homepage design on OJS 2.

Figure 3. UTL’s new homepage design on OJS 3

Checklist item #16: If you are looking to redesign your OJS homepage, consider timing it with the upgrade to reduce the impact on users.

Improving server infrastructure

At Scholars Portal, we used the upgrade to OJS 3 as an opportunity to improve our server infrastructure for our hosting service. When the service began, each institution’s instance was hosted on a separate server, along with a SQL database hosted on the same server. As the service has grown significantly, we decided to use the upgrade to OJS 3 as an opportunity to move to a single server to host all instances, reducing the need to maintain 10 different servers.

The new server infrastructure included an updated version of PHP, which is essential as OJS is a PHP application. This new setup also allowed us to move from CentOS 6 to 7, and all of our databases were moved to a dedicated server hosting MariaDB 10, which contains a testing and production database for each instance. We also began enforcing HTTPS across all of our OJS instances, using either Let’s Encrypt or certificates provided by the institution. By making these changes to our servers during the upgrade process, we reduced the need for future downtime and improved the overall performance of the system.

Checklist item #17: If there are aspects of your server architecture you could improve, consider making these changes during the upgrade to reduce future downtime or impact on users.

Dealing with pushback

If you encounter pushback due to significant changes in the functionality that a journal has come to rely on, evaluate possible alternatives or workarounds. If the functionality is critical, be prepared to reschedule the upgrade until it becomes available. In the case of UTL, no such changes were reported before the upgrade, and while non-critical changes were discovered afterward, we were able to come up with workarounds for all. One such example was that biographies in editors’ profiles linked to from the masthead were no longer public. We worked with the journal manager to create a page that would have the biographies outside of personal profiles and directed the links in the masthead to it.

Because the OJS 3 upgrade involves significant changes to the look and feel of the user interface, there may be some pushback from journals that have invested in their existing setup and may not have the resources to do this work again. In this case, consider offering assistance on a case-by-case basis.

At UTL, only one journal expressed concern about the upgrade since they did not want their old look to change. Curiously, the old look was not a custom design but one of the OJS 2 template themes. We recreated the look in the OJS 3 sandbox ahead of time and shared it with the journal for feedback. We received a letter of gratitude from the editors after the upgrade was complete congratulating us on how smoothly it went.

Checklist item #18: If critical functionality is not available in the new version, consider postponing the upgrade.

Checklist item #19: Be prepared to be flexible and offer additional assistance if the upgrade takes away previously used functionality, or where modifications are beyond the journal team’s skill level.

Production upgrade
Our estimated downtime for the UTL’s instance upgrade was 2 days. We communicated the dates to the editors three months in advance to give them the opportunity to notify us of potential date conflicts. We ended up completing the upgrade in 3 days. It was partially due to the fact that the upgrade was performed concurrently with a migration to a new server, and additional time was needed for DNS work that added a layer of communication with network services staff on both the SP and UTL side. Journal teams were understanding about the delay.

**Technical upgrade steps**

Below is an outline of the steps taken by Scholars Portal when completing a production upgrade from OJS 2 to 3. PKP has also outlined some of these steps in their documentation.

1. Get your OJS 3 code base set up in a separate directory, either from git or an official release package.
2. Disable your production OJS 2 instance and display a maintenance message for end users.
3. Copy the public and files directories from OJS 2 into their new locations for OJS 3.
4. Create your new configuration file (config.inc.php) from config.TEMPLATE.inc.php, leaving installed set to “off”, and copy over relevant settings from your old configuration file. Due to significant changes in the configuration file from OJS 2 to 3, it’s recommended to create a new file rather than using the previous version.
5. Back up your pre-upgrade setup so you can restore quickly if anything goes wrong.
6. Check that all of your file permissions are set appropriately for your environment, including the cache and public directories.
7. If you’re using a new database name or location, import your OJS 2 database into this location.
8. Open a browser window and navigate to your OJS 3 instance to make sure there aren’t any warnings or errors.
9. Run the upgrade script in tools/upgrade.php.
10. Set installed to “on” in your configuration file.
11. Archive your OJS 2 directory and move your OJS 3 directory into the production location (e.g. by renaming the folder to the previous name used by OJS 2).
12. Navigate to your OJS instance in a web browser and explore various journals, test logging in, make sure files are downloadable, test any custom domains, and complete any other necessary tests.
13. Apply any necessary customizations, like plugin installations, new themes, or other changes that were noted during the testing phase of the upgrade.
14. Let your journal managers and other stakeholders know that the upgrade is complete.

For larger OJS instances like UTL’s the upgrade script can take over 12 hours to run, primarily because of the size of the metrics tables, which is an issue that PKP is actively working to resolve.

We found it useful to turn off PHP warnings in our php.ini file – logs can become quite large otherwise (sometimes multiple gigabytes per log file). If you’re encountering bugs, you may need to change this temporarily to see error outputs. PKP is gradually working on reducing the PHP warnings, so this may not be necessary in the future.

| Checklist item #20: When planning upgrade time, ensure that the time to run the upgrade script is considered for larger instances.
| Checklist item #21: Consider turning off most PHP warnings so log files don’t become unmanageably large – you can always enable them later as needed for debugging.

**Post-upgrade**

The work does not end with the upgrade. In fact, the troubleshooting, support, and training post-upgrade turned out to be the most time-consuming phase of the process. It is important to take this into account and schedule staff time post-upgrade accordingly.

**Troubleshooting**

While it would be ideal if all journal teams did rigorous testing in the pre-upgrade phase, this is frequently not the case. Many of the troubleshooting questions that came in post-upgrade had to do with carrying over OJS 2 set up or changed/missing functionality.

For earlier upgrades, this phase was especially challenging as users encountered bugs that existed in the earlier versions of OJS 3. As these bugs have been resolved by the PKP team, these issues became less frequent for each upgrade. For these earlier upgrades, a significant amount of time was spent investigating bugs encountered by users. Before posting in the forums, make sure to search both the PKP forum and GitHub issues. Encourage your users to engage with and search through the PKP forums as well, where possible.

With the move to OJS 3, PKP is updating the software more frequently now, including more small releases and bug fixes. Monitoring for the latest updates will make for a more smooth user experience as bugs are resolved. We recommend managing your OJS code with git so you can pull in small code changes from the PKP repository; at Scholars Portal, we manage our code in a local GitLab instance. We also use RSS to stay on top of major releases, and Zapier integrated with GitHub to monitor commits to the branch of OJS that we’re actively using.

**Training and communication**

Rather than hosting another OJS 3 workshop at UTL, we opted for a combination of:

- Online training resources via the updated Journal Publishing Guide (thoroughly put together by our graduate library assistant)
- Individual consultations via email and phone – that proved the most in demand
- Scheduled drop-in sessions – that proved to be poorly attended
- “New in OJS” listserv series that we launched based on the frequently asked questions about the changed and new features

We soon realized that the extensive pre-upgrade communication put our team on the radar of journal editors who have not been previously in touch beyond occasional troubleshooting requests – and we started receiving more in-depth questions about various aspects of the publishing process. The upgrade turned into a welcome outreach opportunity and helped us establish more meaningful connections with the journal teams and offer more valuable support.

**Reflection, lessons learned, & next steps**

Looking back at the OJS 3 upgrade process, we can say that it went quite smoothly, both for the UTL and SP staff doing the upgrade as well as for the journal editors. Considering the size of the UTL instance, we were happy to only encounter minor issues throughout the process that were all efficiently resolved. We credit our success to these three major factors:

- Generally stable and bug-free state of the version we were migrating to
- Sufficient experience with prior upgrades, and general expertise of SP staff
• Enough prep time and extensive communication with our journal teams

That being said, there is always space for improving and reflecting on what we at SP might have done differently, such as:

• Completing more test upgrades for each instance leading up to the final migration date.

• Creating a testing workflow that journals can follow which might point to any issues that happened in the upgrade process, such as missing data. This can also help journal managers begin to get familiar with the new system. UTL created one for their users (see Appendix B), but it would have been useful to develop one at the consortial level that each institution could edit as needed.

• Evaluate the feasibility of using automated testing tools like Katalon Recorder, or writing Selenium tests, in order to spot-check the data migration in the upgrade process. Tests could be created during the first upgrade to OJS 3, and run for each upgrade that follows. This may be especially useful for tracking persistent and recurring issues.

One of the most impactful side effects of the upgrade process was establishing (or re-establishing) communication and rapport with journal teams that had previously mostly worked on their own. On one hand, this has resulted in an increase in the number of questions received by UTL’s service, which have gradually morphed from upgrade-related issues to more in-depth publishing inquiries. This is something we recommend accounting for if the support for journal teams at your institution is limited. At UTL, 0.3 FTE is dedicated to the journal production service support, and while the increase could not be called overwhelming, we did benefit from having a student assistant help with basic triage and troubleshooting initial inquiries.

On the other hand, and more importantly, this re-established communication with the editorial community has motivated the UTL to proceed with planning more in-depth support and services, such as centralized DOI minting, assistance with DOAJ applications, and additional workshops for editors, which we will be rolling out in the future.

Additionally, we hope that OJS 3 with its intuitive navigation and flexible interface will attract journal teams (particularly student journals) that had previously moved off of OJS 2 to other platforms in search of a simple and flexible content management system. These improvements to OJS, along with other library services for student publishers, will be highlighted at our Fourth Annual Student Journal Forum planned for January 2019.

In conclusion, software maintenance and upgrades may not be the most exciting aspect of library technology, but there is a rewarding challenge in making the transition seamless for end-users and painless for staff. With careful advance planning, user support, and communication, the upgrade may bring more than just an improvement in functionality – it may strengthen your connections with your users and facilitate further outreach and valuable engagement opportunities.

Acknowledgments

Thank you to the following staff and students at Scholars Portal and UTL that helped make this upgrade a success: Steve Baroti, Gordon Belray, Bartek Kawula, Bikramjit Singh, Cordelia Tang, and Sean Xiao Zhao.

Thank you to Alec Smecher at PKP for pointing us in the right direction to resolving database encoding issues.

Thank you to James MacGregor at PKP for providing valuable feedback on a draft of this article.

References

[1] Quick Facts: Student Enrolment (Fall 2017 – 18), Faculty & Staff (Fall 2016) [Internet]. [updated 2018]. University of Toronto; [cited 2018 Dec 17] Available from: https://www.utoronto.ca/about-u-of-t/quick-facts


Appendix A: Upgrade Prep Checklist

1. Work with your journals to identify their essential features and make sure they are available in the new version.

2. Consider cleaning up the content you don’t need to carry over, including inactive journals, spam users, etc.

3. Decide if you will provide sandbox sites to journals for testing (or use PKP’s OJS test-drive instance).

4. Set aside time BEFORE and AFTER the upgrade for testing, training & troubleshooting.

5. Check that your contact information for hosted journals is up-to-date and re-establish contact in case of editorial changes.

6. Announce plans to upgrade early, even if you don’t have exact timelines yet.

7. Set downtime for production upgrade – and request confirmation.

8. Set content freeze dates if necessary.

9. Align your training/consultation offerings with demand — your editors may prefer emails or a call over workshops and drop-ins.

10. Define your level of support and communicate it clearly.

11. Set the cut-off date for pre-upgrade support requests.

12. Create frequent backups of your code, database, and files as you work through each upgrade.

13. When setting up your testing instance, disable any plugins that may send data out to other services or systems.

14. Consult the PKP forums and open GitHub issues when troubleshooting problems.

15. Put together an easy to follow the test plan for the journal teams to test and report issues.

16. If you are looking to redesign your OJS homepage, consider timing it with the upgrade to reduce the impact on users.

17. If there are aspects of your server architecture you could improve, consider making these changes during the upgrade to reduce future downtime or impact on users.

18. If critical functionality is not available in the new version, consider postponing the upgrade.

19. Be prepared to be flexible and offer additional assistance if the upgrade takes away previously used functionality, or where modifications are beyond the journal team’s skill level.

20. When planning upgrade time, ensure that the time to run the upgrade script is considered for larger instances.

21. Consider turning off most PHP warnings so log files don’t become unmanageably large — you can always enable them later as needed for debugging.

Appendix B: Upgrade Resources
The following resources are linked throughout this article and are compiled here for reference.

- PKP's OJS testing instance: https://pkp.sfu.ca/ojs/ojs_demo/
- Fixing database encoding issues in OJS: https://github.com/kaitlinnewson/ojs-tools/blob/master/fix-database-encoding.md
- Cleaning up spam users
  - PKP's documented process: https://docs.pkp.sfu.ca/admin-guide/en/securing-your-system#cleaning-lots-of-users
- PKP upgrade documentation: https://github.com/pkp/ojs/tree/master/docs
- PKP community forum: https://forum.pkp.sfu.ca/
- PKP Administrator's Guide: https://docs.pkp.sfu.ca/admin-guide/
- Get emails with new GitHub commits in Zapier: https://zapier.com/apps/github/integrations/email/11485/get-emails-with-new-github-commits
- The University of Toronto's journal publishing guide: https://jps.library.utoronto.ca/index.php/pubguide

**Appendix C: Pre- and Post-Upgrade Test Plan for Journal Managers**

**OJS 3.1 UPGRADE TEST PLAN**

**OJS 3.1 Upgrade Schedule**

March – July – pre-upgrade preparation: editorial teams review OJS 3 documentation, explore their test instances, identify customizations and report any issues to the JPS team at jps@library.utoronto.ca

August 7-8 – upgrade: JPS team performs platform upgrade. Journal websites will not be available at this time.

August 9-17 – post-upgrade set up: editorial teams test their new journal sites, apply customizations and report any issues to the JPS team at jps@library.utoronto.ca

**Drop-in sessions and consultations**

JPS team offers drop-in sessions to help you test and set up your journal before and after the upgrade.

<table>
<thead>
<tr>
<th>Drop-ins pre-upgrade*</th>
<th>Drop-ins post-upgrade*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday July 12, 2-4pm</td>
<td>Tuesday August 14, 10am-12pm</td>
</tr>
<tr>
<td>Tuesday July 17, 10am-12pm</td>
<td>Thursday August 16, 2-4pm</td>
</tr>
<tr>
<td>Wednesday July 25, 2-4pm</td>
<td>Monday August 20, 10am-12pm</td>
</tr>
<tr>
<td>Thursday August 2, 10am-12pm</td>
<td>Wednesday August 22, 2-4pm</td>
</tr>
</tbody>
</table>

* All drop-ins are held in Study Room #2, ground floor Robarts Library, 130 St. George St, Toronto

If you would like to schedule a virtual session or a meeting outside of the drop-in hours, contact us at jps@library.utoronto.ca

**Your journal's demo instance**

An upgraded demo instance of your journal is available at https://ojs.scholarsportal.info/toronto/ for testing, learning, and tweaking of your journal's looks and settings. **None of the changes made in the demo instance will be carried over to your upgraded website** – they will need to be re-applied. Contact the JPS team if you have not received your demo instance or are experiencing access issues.

**Pre-upgrade Test Checklist (on demo instance*) – report issues by July 20**

**Access and Subscriptions**

- Issue and article pages including full text files are accessible (recent issues published since the demo instance was created in the spring will not be there)
- For subscription journals – individual and institutional subscribers have access to their content

**Website Content**

- The content on About, Editorial Team, Announcements, and other pages is in place
- All internal and external links from website pages work
- All uploaded files (e.g. Guidelines for Authors, etc.) are accessible

**Website Appearance**

- The items in the top navigation menu and sidebar menu are in place (these may be reset to default as a result of the upgrade and may need to be re-added in the demo and post-upgrade instance)
- Your preferred theme is in place (you can select from one of the existing themes or customize and upload a CSS; save your CSS for re-upload after the upgrade)
- Your journal's logo/banner/favicon are in place (these may not display in demo instance; save them from the current OJS 2 instance for re-upload after the upgrade)

**Submissions and Review Workflow**

- Submitted, under review and published articles are present in the workflow (recent articles submitted since the demo instance was created in the spring will not be there)
- Workflow settings reflect your journal's editorial process
- Perform a test submission, review and publication of an article/issue
- If you are using a Quick Submit plugin rather than full review workflow – perform a test submission via the plugin

**User Roles**

- Team members are assigned to their respective roles (Journal Manager, Editor, Reviewer, etc.)
Your team roles reflect your journal’s management process (these can be adjusted or removed)

Test roles at each level including Journal Manager, Editor, Author, Reviewer, Reader, etc.

Plugins

- The plugins your journal uses function as expected on the demo instance (report ASAP any missing or non-functioning plugins)

Communications

- Notify your users and team members of the upcoming upgrade and impact on scheduling, if any

Post-upgrade Test Checklist (on upgraded live website) – after August 8

- Repeat testing steps from the pre-upgrade test checklist
- Re-upload your customized CSS, if any, and re-add menu items where needed
- Report any technical issues to jps@library.utoronto.ca
- Notify your users of the completed upgrade

OJS 3 resources

- OJS 3: Setting up a Journal in OJS 3 (PKP School video course, free with registration) – http://pkpschool.sfu.ca/courses/setting-up-a-journal-in-ojs-3/
- PKP Community Forum – https://forum.pkp.sfu.ca/
- UTL’s Journal Production Services support team – jps@library.utoronto.ca

Contact the JPS team at jps@library.utoronto.ca with any questions or issues related to the upgrade or your journal website

This test plan is adapted from the OJS 3.1.1 Test Plan kindly shared by the UBC Library.

About the Authors

Mariya Maistrovskaya (mariya.maistrovskaya@utoronto.ca) is the Institutional Repositories Librarian at the University of Toronto Libraries. She manages the TSpace research repository, Journal Production Services, and a suite of other services that support scholarly communications and advance open access at the University of Toronto.

Kaitlin Newson (kaitlin@scholarsportal.info) is the Digital Projects Librarian at Scholars Portal, the technology service arm of the Ontario Council of University Libraries. You can find her online at https://www.kaitlinnewson.com and on twitter as @kaitlinnewson.

Subscribe to comments: For this article | For all articles

This work is licensed under a Creative Commons Attribution 3.0 United States License.