UpgradeRails
https://www.upgraderails.com/

Case Study: CodePen Rails Upgrade

by Joshua Wood

"This is the perfect thing to bring a consultant on for because we can be building user facing features and [UpgradeRails] can be working super hard in the background. And they specialize in this; this is what their firm does at this point, so to me this is a perfect use of our hard earned dollars to put back into the company." - Tim Sabat, Cofounder CodePen

Executive Summary

In 2016 CodePen selected UpgradeRails to upgrade the primary component of their platform -- a Rails application -- from Rails 3.2 to 4.2; a critical upgrade because Rails 3.2 has reached end-of-life and is open to security vulnerabilities. The CodePen development team is hyper-focused on product development, lacking the necessary time to complete the upgrade in-house.

CodePen chose UpgradeRails over several other options because we specialize exclusively in Rails upgrade projects. Our process makes us the most affordable as well as the most experienced option on the market.

Initially estimated at 1-2 weeks, the project was completed in exactly 1 week (32 hours) of development time. The upgrade progressed in tandem with the in-house product development, whose team were involved during a short integration phase only.

The end result is that UpgradeRails completed the upgrade with minimal disruption and in significantly less time than it would have taken a less experienced team to do the same work, saving CodePen tens of thousands of dollars. CodePen's Rails application is now up-to-date and secure, and the CodePen development team continues to meet product deadlines which would be otherwise impossible.

Background

CodePen was founded by Alex Vazquez, Tim Sabat and Chris Coyier in 2012, and since has grown to a team of 8. The web-based service, which provides front-end developers with shared code sandboxes and collaboration tools, has over half a million users and is ranked 1,249 out of all sites on the Internet by Alexa, with 2.3 million unique visitors per month.

The core of CodePen's technology is a web application built on Ruby on Rails; the application leverages the full Rails stack and also makes heavy use of JavaScript on the front-end.

As a small team serving a relatively large user base, one of the key challenges faced by CodePen is keeping their software stack up-to-date while under pressure to continually deliver new features to their customers.

When they contacted UpgradeRails they were on Rails 3.2, while the latest version of Rails -- version 5 -- was just completing beta testing. It became critical that they update to a newer version of Rails for several reasons:

- Rails 3.2 is no longer maintained. As a result security patches are not released, exposing CodePen to vulnerabilities and hacking attempts.
- It is much easier for developers to work with the latest version of Rails; the documentation and discussions on the web often reference the newest version of Rails, making it difficult for the development team to find answers for older versions.
- They would be able to use new features and methods, increasing the pace of development.

Alternatives

CodePen had several alternatives to consider when planning their Rails upgrade project, which we'll discuss in turn. In each case, they came to the conclusion it was more efficient to hire UpgradeRails.

Perform the upgrade in-house

CodePen potentially could have had their in-house development team perform the upgrade. Rails maintains adequate documentation for upgrading Rails itself to newer versions, and there are a number of resources online. Companies perform their own upgrades in-house all the time, and typically spend substantial time and money in the process.

The main reason an in-house project was not feasible for CodePen is the disruptive impact on their development team that already had a heavy workload. Stopping to work on maintenance projects (which if done correctly have no immediate user-facing impact, beyond better performance) was a distraction they couldn't afford to manage themselves.

Additionally, UpgradeRails specializes in upgrading Rails applications. Our process allows us to do the work far faster than a team which must multi-task, translating to lower costs and shorter duration. CodePen did the math and realized it was also cheaper to have us upgrade their Rails app vs. upgrading in-house.

Hire an agency

Another option was to outsource the upgrade to a more standard Rails agency. This solved the problem of distracting the in-house development team, but did not win on cost. Agency rates may vary, but confidence in the outcome of the project would necessitate hiring a top agency; the process we've developed at UpgradeRails allows us to upgrade Rails applications faster than other top agencies while bringing more experience to the table in our area of expertise. That translates to a higher quality and lower cost service.

Solution

When CodePen engaged UpgradeRails, we began by discussing their goal for the project - to modernize their Rails application with minimal impact to their development team.

We agreed that because Rails 5 was not released yet, the target version should be the latest stable version of Rails: version 4.2. With that in mind, we knew there were 3 steps to perform:

- Upgrade from Rails 3.2 to 4.0
- Upgrade from Rails 4.0 to 4.1
- Upgrade from Rails 4.1 to 4.2

It's important to perform each upgrade incrementally - it's easy to miss required steps when jumping versions. Additionally, there are several phases which upgrade steps must go through:

- Development - Our teams upgrade the Rails application; all changes are applied to a dedicated code branch.
- Integration - As CodePen teams make changes to their master code branch, we integrate their changes into the upgrade. When the upgrade is complete, we integrate our branch back into the master branch.
- Quality Assurance (QA) - CodePen is responsible to test deployment of the upgrade to staging environments to verify that the production deployment will go smoothly and that application features work correctly.
- Deployment - After completing QA, CodePen deploys the upgrade to production environments.

With these targets in mind, our team swung into action and began the upgrade from Rails 3.2 to 4.0. The upgrade was performed while CodePen's product development team continued their work on application features; periodically we integrated new work from their team into our upgrade branch of development. Our process is engineered in a manner which strategically avoid conflicts with other teams, resulting in minimal merge conflicts during integration.

After completing this initial 4.0 upgrade, we submitted our branch to CodePen for review and QA testing. We advised that they move to Rails 4.0 in production environments after testing was completed in order to avoid unnecessary integration cycles during the subsequent Rails 4.1 and 4.2 upgrades.

Next the upgrades to Rails 4.1 and 4.2 were performed. While each upgrade was performed sequentially, we opted to deploy 4.2 to production after both upgrades were completed saving time during QA testing (essentially cutting QA and deployment time in half). During the 4.1 and 4.2 upgrades we also offered several recommendations for necessary code changes which we back-ported to Rails 4.0 to minimize conflicts during integration.

When the 4.2 upgrade was ready for QA, we review with CodePen the important changes to the application and provided checklists for their developers to review. The final production deployment was performed successfully and in a timely manner.

Recommendations

We continue to refine our process as we complete further Rails upgrades. Our goal is to continually lower the cost and risk of upgrading as we acquire new methods, tools and knowledge to speed our pace of development. A few lessons from the CodePen upgrade follow.

Coordinating QA testing and deployment

It's extremely important that QA testing is performed in the proper environments and that all teams plan for the time required to respond to potential bugs and issues before deployment is completed. During the deployment of the Rails 4.0 upgrade there was a bug which caused user sessions to be reset, meaning that users had to re-login to the site. This was the result of a setting which was not configured properly during development. Normally this is something that would have been caught during QA, but QA was not performed, and the deployment happened without additional review. We used this experience to improve our process in several ways:

- The misconfigured setting has been added as an issue to our internal pre-QA checklist so that our teams do not miss it during future projects.
- We now provide customers with a more detailed process for QA and deployment phases, which are largely their responsibility. This includes checklists and other materials which our customers should use to test the deployment scenario for their applications.

Code improvement opportunities

New versions of Rails often include changes which attempt to curb bad development practices and plug gaps in prior versions. As a result, sometimes a workaround that was necessary on an

older version of Rails may conflict with the newer version. These types of issues are sometimes missed by the test suite and need to be fleshed out during the review and QA phase; occasionally there will be an issue which makes it through QA and into production, where more of the application is exercised in unique ways. While this can be irritating, the end result is that the quality of the code improves during the process of working through these issues. The upgrade process is a valuable opportunity to ensure that the application is using the current best practices in the Rails community.

Conclusion

When first contacted CodePen's current version of Rails -- version 3.2 -- had reached end-of-life, resulting in security vulnerabilities. It was imperative to upgrade, however their in-house development team didn't have the necessary capacity available in a near-term time frame. While the upgrade work was essential, it was not customer-facing, making it the perfect project for CodePen to outsource.

CodePen chose UpgradeRails because we specialize in upgrading Rails applications and have developed a process which allows us to perform Rails upgrades in dramatically less time than it would take an internal team or an external agency to complete the same effort.

Working with UpgradeRails allowed CodePen to continue to meet product deadlines for THEIR clients without missing a step while we performed the upgrade concurrently in the background.

The end result is that CodePen saved tens of thousands of dollars by engaging UpgradeRails while meeting their business and product goals. Their Rails application is now modern and secure, and their development team continues to meet product deadlines which would have not been possible otherwise.

Further reading/listening

- Joshua Wood discusses the upgrade on CodePen's weekly podcast
- Upgrading CodePen on The Hint Blog
- https://www.upgraderails.com/
- http://codepen.io/