CASE STUDY

Daily Deployments at CSG International (2013)

CSG International is North America's largest SaaS-based customer care and billing provider, with over sixty-five million subscribers and a tech stack that covers everything from Java to mainframe.[7] Scott Prugh, Chief Architect and VP of Development, led an effort to improve the predictability and reliability of their software releases. To achieve this, they doubled their release frequency from two per year to four per year (halving their deployment interval from twenty-eight weeks to fourteen weeks).[8]

Although the Development teams were using continuous integration to deploy their code into test environments daily, the production releases were being performed by the Operations team. Prugh observed,

> It was as if we had a "practice team" that practiced daily (or even more frequently) in low-risk test environments, perfecting their processes

> and tools. But our production "game team" got very few attempts to practice, only twice per year. Worse, they were practicing in the high-risk production environments, which were often very different than the pre-production environments with different constraints—the development environments were missing many production assets such as security, firewalls, load balancers, and a SAN [storage area network].[9]

To solve this problem, they created a Shared Operations Team (SOT) that was responsible for managing all the environments (development, test, production) performing daily deployments into those development and test environments, as well as doing production deployments and releases every fourteen weeks. Because the SOT was doing deployments every day, any problems they encountered that were left unfixed would simply occur again the next day. This created tremendous motivation to automate tedious or error-prone manual steps and to fix any issues that could potentially happen again. Because the deployments were performed nearly one hundred times before the production release, most problems were found and fixed long before then.[10]

Doing this revealed problems that were previously only experienced by the Ops team, which were then problems for the entire value stream to solve. The daily deployments enabled daily feedback on which practices worked and which didn't.[11]

They also focused on making all their environments look as similar as possible, including the restricted security access rights and load balancers. Prugh writes, "We made non-production environments as similar to production as possible, and we sought to emulate production constraints in as many ways as possible. Early exposure to production-class environments altered the designs of the architecture to make them friendlier in these constrained or different environments. Everyone gets smarter from this approach."[12]

Prugh also observes:

> We have experienced many cases where changes to database schemas are either 1) handed off to a DBA team for them to "go and figure it out" or 2) automated tests that run on unrealistically small data sets (i.e., "100's of MB vs. 100's of GBs"), which led to production failures. In our old way of working, this would become a late-night blame game between teams trying to unwind the mess.
>
> We created a development and deployment process that removed the need for handoffs to DBAs by cross-training developers, automating schema changes, and executing them daily. We created realistic load testing against sanitized customer data, ideally running migrations every day. By doing this, we run our service hundreds of times with realistic scenarios before seeing actual production traffic.[13‡]

Their results were astonishing. By doing daily deployments and doubling the frequency of production releases, the number of production incidents went down by 91%, MTTR went down by 80%, and the deployment lead time required for the service to run in production in a "fully hands-off state" went from fourteen days to one day.[15]

Prugh reported that deployments became so routine that the Ops team was playing video games by the end of the first day. In addition to deployments going more smoothly for Dev and Ops, 50% of the time the customer received the value in half the time.[16]

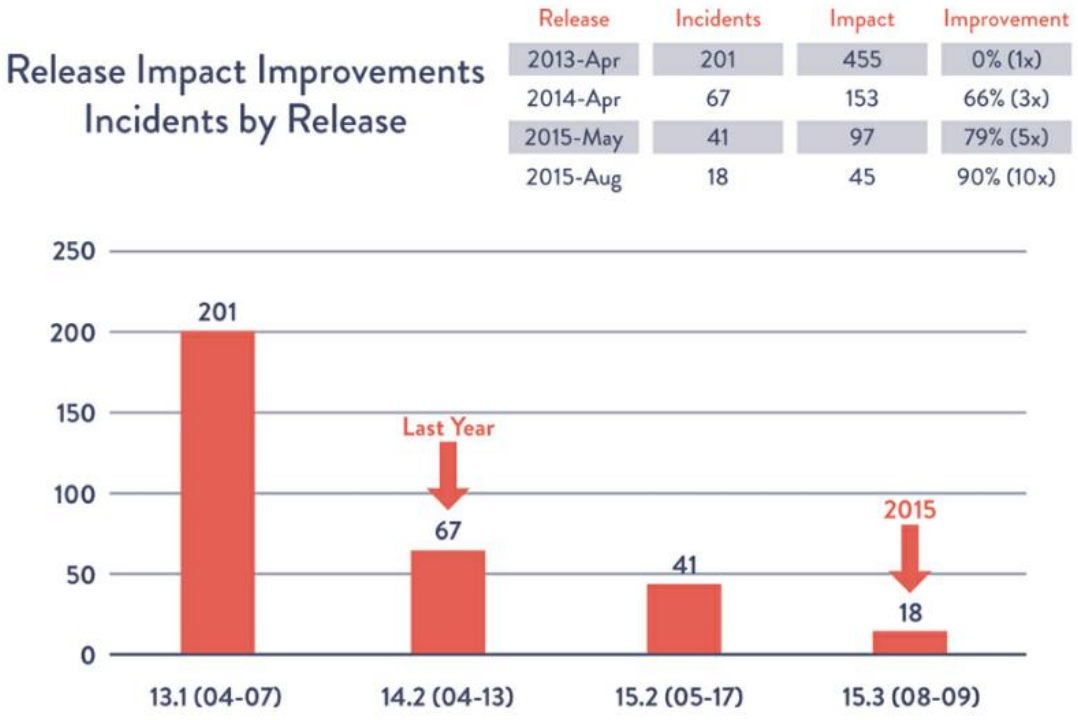

Figure 12.2: Daily Deployments at CSG International

Daily deployments and increasing release frequency resulted in a decrease in the number of production incidents and MTTR.

Source: "DOES15 - Scott Prugh & Erica Morrison—Conway & Taylor Meet the Strangler (v2.0)," YouTube video, 29:39, posted by DevOps Enterprise Summit, November 5, 2015, https://www.youtube.com/watch?v=tKdIHCL0DUg.

This case study underscores how more frequent deployments are good for Development, QA, Operations, and the customer. Frequent deployments allowed problems to be identified earlier, motivated teams to fix errors sooner and led to the delivery of cleaner code faster.