WHITEPAPER

The Benefits of Deployment Automation

Reducing the risk of production deployments

Contents

Executive Summary

In October 2013 we conducted a survey of 72 firms that had recently adopted automated deployment solutions, asking a series of questions about their deployment experiences before and after automation. We expected that automation would make deployments faster, and of course it did – computers are good at making things faster. However, the survey also revealed a number of unexpected results.

The first finding was a dramatic decrease in the perceived risk of a deployment going badly. 94% of respondents reported that deployments were now far less risky after adopting automation, while 90% reported that their team and stakeholders had more confidence in their ability to execute a production deployment successfully.

The second major finding was that after adopting automation, deployments became far more frequent. Automation hadn't just made deployments easier, it had made them less risky, and this translated to firms attempting deployments far more often than when deploying manually. Prior to automation, 60% of respondents deployed to pre-production (development, test, QA, staging, etc.) or production environments only one or two times a month, or even less. In contrast, only 19% of respondents attempted a deployment on a weekly or more frequent basis.

After adopting automation, these numbers changed dramatically. 43% of respondents now perform at least one deployment a day, and another 21% deploy more than once a week. Only 18% wait a week or more between deployments. 56% reported getting earlier and more frequent feedback from testers and end users faster thanks to this increased frequency of deployment.

For these firms, automation made deployments faster and reduced risk. When there is less risk and effort involved in deployment, firms are more likely to deploy smaller changes to production, and to get feedback faster.

In this whitepaper, we explore how automation can lead to shorter iteration and feedback cycles, by reducing the risk and effort involved in deploying software to production. We examine the findings of this survey in the context of Agile software delivery, and look at ways that you too can experience the benefits of deployment automation.

"How often do you deploy to any environment?"

Before automation

19%

Weekly or more

60%

Monthly or less

After automation

43%

Daily

21%

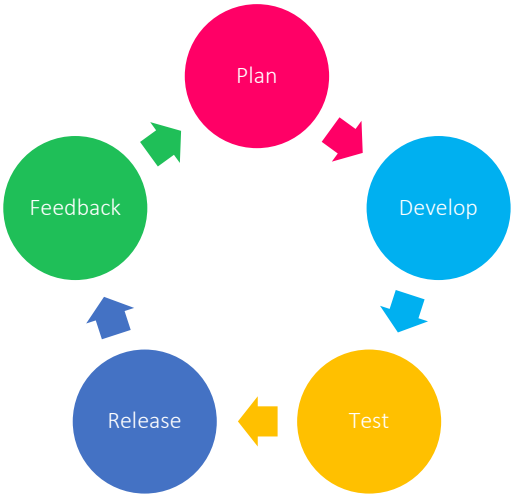
Weekly or more

"It's helped enormously to improve our reliability and repeatability of our deployments and given us the confidence to deploy much more often, with a direct benefit to our end users."

Marcus Denny
Software Development Manager
Visa Global Logistics

Deployment and Agile software development

Agile software development methodologies, such as Scrum, Kanban and Lean, have a focus on iterative, incremental development. The goal is to produce high-quality software with less waste, by delivering small increases in functionality to end-users frequently. Deployment is fundamental to Agile software development, because deployment is the process by which software gets into the hands of users.



In most Agile processes, at the end of each iteration cycle (typically one, two or four weeks long), feedback from stakeholders is supposed to inform the next iteration. All too often, however, software that is otherwise ready to deliver isn't actually delivered when the iteration ends. Since deployments can be time consuming, costly, and high-risk, they need time to plan and to execute.

Software is written to solve problems. Software that has been written, tested, and reviewed, and yet isn't in the hands of end users, isn't actually solving a problem. End-users and stakeholders can't begin to provide real, valuable feedback on something they can't actually use, no matter how many times they see a demonstration.

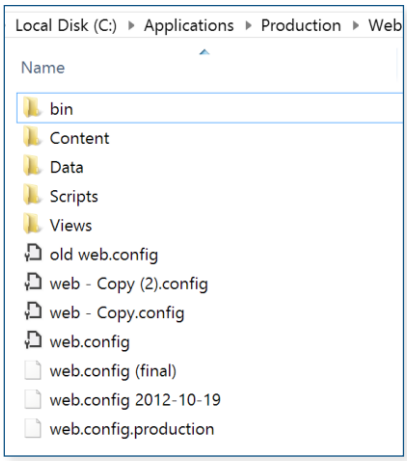
> "Deployment Automation has allowed our company to react quicker to changes in the market and the needs of our business without any measurable downtime. Automating the process has also removed the risk of human error and improved the confidence in our software solutions."
>
> Jasper Rijkeboer
> Software Engineer
> Tellus Leads

AIM TO DEPLOY AT LEAST ONCE PER ITERATION

Our goal, therefore, should be to narrow the gap between software being ready for delivery, and the time when it is actually delivered. We invest in source control systems, refactoring tools, automated continuous integration servers, and automated tests to improve the speed at which we can develop software. It therefore makes sense to invest in systems that allow us to get that software to the people that matter as quickly as possible, and with less risk.

Manual deployments

In most organizations today, deployments are done manually. One or more team members must manually copy files, edit configuration values, remote desktop to servers, restart services, synchronize folders, and apply database changes.



A sight that is, unfortunately, far too common on production web servers in organizations small and large

Deploying software this way creates a number of risks and poses some challenges, which we'll examine below.

MISTAKES HAPPEN

When deployment is reliant on a person following a process, it's very easy for a step to be overlooked or forgotten. Humans can also easily make mistakes when following the process, such as stopping the wrong service, being in the wrong remote desktop session, or forgetting the 'WHERE' clause on a SQL 'DELETE' statement. We are, after all, human.

DEPENDENCY ON ONE PERSON

This causes two problems. Firstly, even if the process is perfectly documented, often only one or two people have all the permissions necessary to execute the deployment. If, in an emergency, another team member is required to perform the deployment, it can take a long time to ensure they have all the necessary permissions. Other team members are often unprepared to take over deployments.

> "Removing dependency on a resource from the operations team to deploy packages means we no longer spend time on manual Azure deployments, while also removing possible human errors. Octopus in short gives us a shorter time to production."
>
> John Korsnes
> Norwegian Broadcasting Corp.

The second problem is that developers and testers are unable to perform development or test deployments, sometimes called “self-service deployments”. And if they do, there’s no guarantee they will follow the same process as the production release engineers. If a new dependency or configuration setting is needed, it might be done on-the-fly during deployment and might not be documented for the production deployment, causing more delays.

TRANSPARENCY AND ACCOUNTABILITY

There’s often no single place where team members can look to see what version of the application is deployed where, when it was deployed, or by whom, and if there is it’s often out of date. There’s also often not much in the way of auditing. Deployment logs are scattered across many servers and aren’t centralized.

> “We’ve been automating our production deploys through Octopus Deploy for well over a year and have seen almost 500 releases. We would have never released so often without the comfort and insights that Octopus Deploy gives us.”
>
> Martijn Laarman
> Software Engineer
> Q42

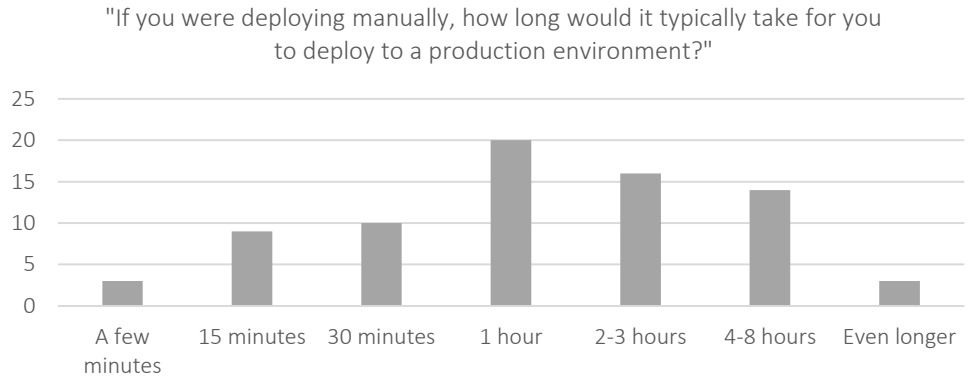
FRICTION

Manual deployments can be tedious and can take significant time, especially when multiple servers are involved. Furthermore, because of the time it takes, more co-ordination is needed to ensure the system is allowed to go down for an extended period of time.

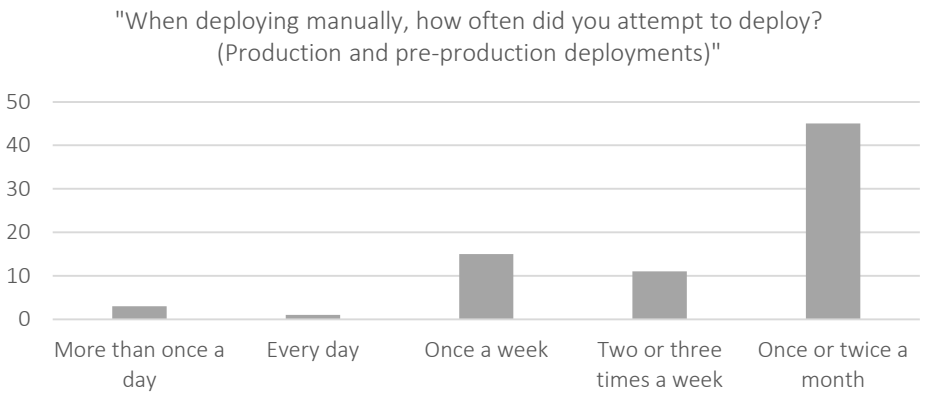
In the event that a critical problem is discovered with an application after deployment, this friction can also hinder the rollback or patching efforts.

THE IMPACT OF MANUAL DEPLOYMENTS

In our survey, 27% of respondents put the time it takes to perform a deployment manually as taking between 15 and 30 minutes. 40% of respondents estimated it to take between one to three hours, and 20% of respondents estimated four hours to a day or more to perform a successful deployment.



When deployment takes so long, and with so many things that can go wrong, it is hardly surprising that over 60% of the firms we surveyed only deployed once or twice a month (that's counting not just production deployments, but test and staging deployments too.)



The most disappointing part about this result, however, isn't the wasted time spent deploying software manually. It's in the opportunity cost of features and bug fixes, which are otherwise complete, tested and ready to be used, stuck in the process waiting for next month's deployment when they could be solving real problems for end-users. There needs to be a better way.

Automated deployments

While it's essential for humans to define the deployment process, is it essential for them to execute it? And even if a human must perform one or two steps, must all the steps be done manually? These are the questions that lay at the heart of deployment automation.

AUTOMATION GOALS

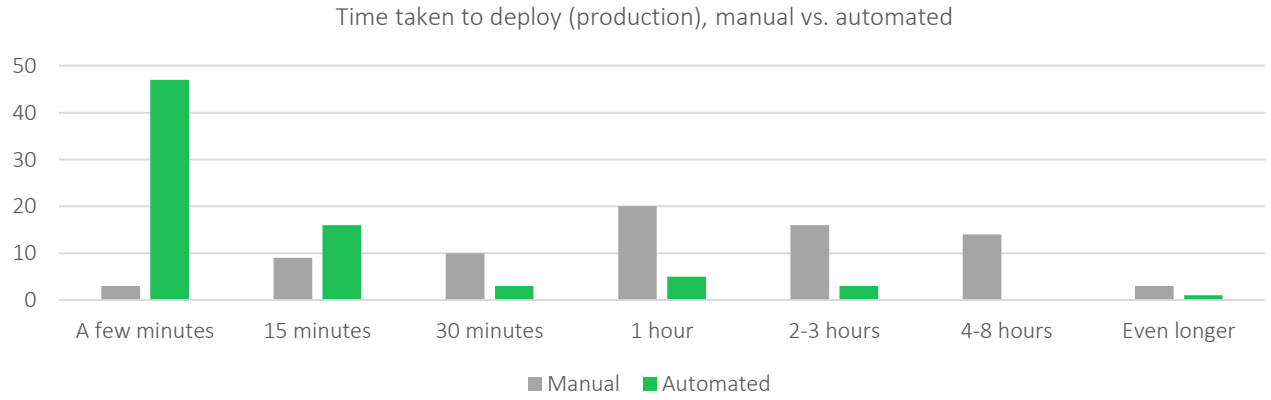
Deployment automation solutions can vary in sophistication depending on the project. They can be a collection of custom scripts, or a full-fledged automated deployment and release management system. They can be stitched together by a release engineer in a few hours, or could take a team of consultants to configure.

The goal of an automated deployment solution is to:

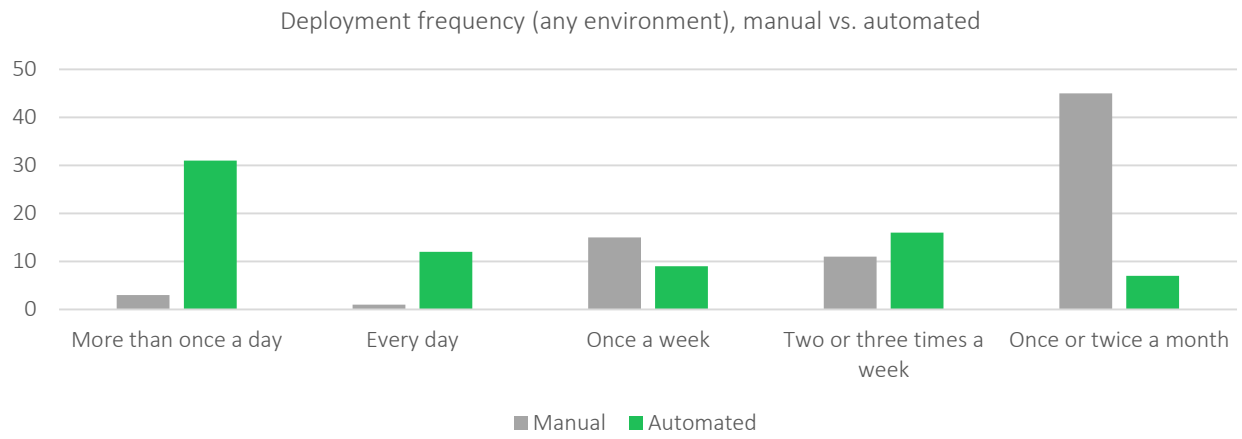
- Automate as many of the deployment steps as possible
- Automate them in a way that is consistent, so that the same automation can be used for development and test deployments as well as production deployments
- Account for configuration differences between environments
- Not compromise the security of any servers involved in the deployment
- Provide a central place for the team to view the state of all deployments, and inspect any failures
- Support some level of auditing

THE IMPACT OF AUTOMATION

For the organizations we surveyed, implementing a deployment automation solution brought a number of advantages. Firstly, the time taken to perform a deployment reduced significantly – 90% put it at less than 30 minutes, with 64% of those taking a few minutes or less. In the graph below you can see how that compares to the time taken prior to automation.



The most important difference, however, was in the frequency of deployments. 43% of respondents now perform at least one deployment a day, and another 21% deploy more than once a week. Only 18% wait a week or more between deployments. Again, the graph below shows the changes:



We also asked about the effects automation had created.

- 94% reported that they felt deployments were less risky thanks to automation
- 90% reported that their team was more confident in their ability to deploy successfully to production
- 72% reported that their team benefitted from the increased visibility into the state of deployments created by the automation system
- 56% reported getting feedback from testers and end users faster thanks to this increased frequency
- 45% reported that more members of their team are now able to initiate a deployment to non-production environments thanks to automation

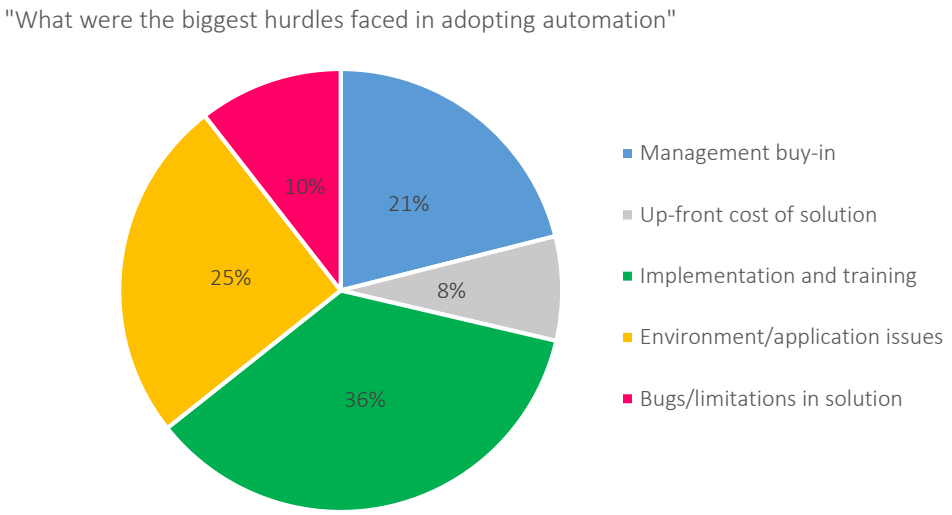
> "Deployment automation is one of the most important changes I have introduced to reduce the risk of deployments, increase the agility of business value delivery and improve software quality."
>
> Rob Moore
> Senior Consultant
> Readify

BARRIERS TO AUTOMATION

Any automation solution requires an up-front investment to implement, with the expectation that there will be a return on this investment in terms of productivity over time. 62% of the respondents to our survey pointed out that they did need to spend significant time in setting up the automation, training their team to use it, and so on. As one would expect, it's clear that there's no such thing as a free lunch when it comes to implementing an automation solution.

After the initial adoption costs, 43% of respondents rated difficulties encountered due to their environment as being a significant hurdle in adopting automation. In some environments, infrastructure issues or application complexity naturally make automation harder than in other environments. Sometimes these issues can be overcome by making

changes to the environment or application; where possible, we recommend finding ways to simplify applications or environments rather than investing time in automating for very complicated scenarios. It's often easier to change the way an application reads its configuration data, for example, than to try and automate for a very complicated configuration system.



The final significant hurdle faced by our respondents was in getting organizational or management buy-in to adopt the automation solution, affecting 36% of respondents.

COMPLIANCE

The centralized auditing, logging, configuration management and permissions management capabilities of an automated deployment system can also make it easier to comply with regulations, standards and internal controls, including:

- ✔ PCI/DSS
- ✔ ITIL and ITGC
- ✔ FISMA

> "It has reduced the human error and given us more confidence over releasing. By using tried and trusted build and deployment scripts we have made ourselves more efficient and productive. We have more time to write quality code and not spend it trying to get it into test/production"
>
> Chris Hey
> Lead Developer
> McCann Manchester

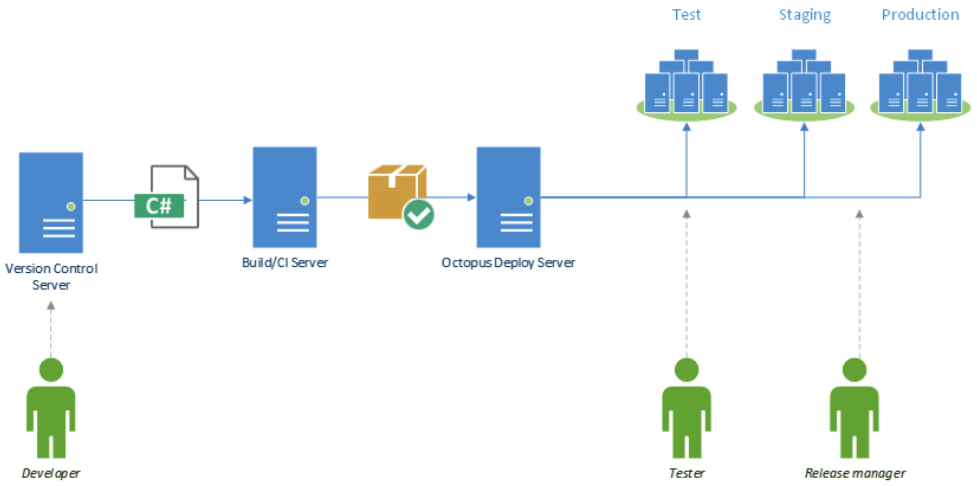
How Octopus Deploy enables automation

So far in this whitepaper we've looked at the benefits of automated deployment compared to deploying manually. In this section, we'd like to show you how easy deployment automation can be to implement using Octopus Deploy.

Octopus Deploy is a user-friendly automated deployment server designed and optimized for .NET developers. We built Octopus with the goal of making repeatable, reliable automated deployments available to everyone.

OCTOPUS DEPLOY IN YOUR DELIVERY PROCESS

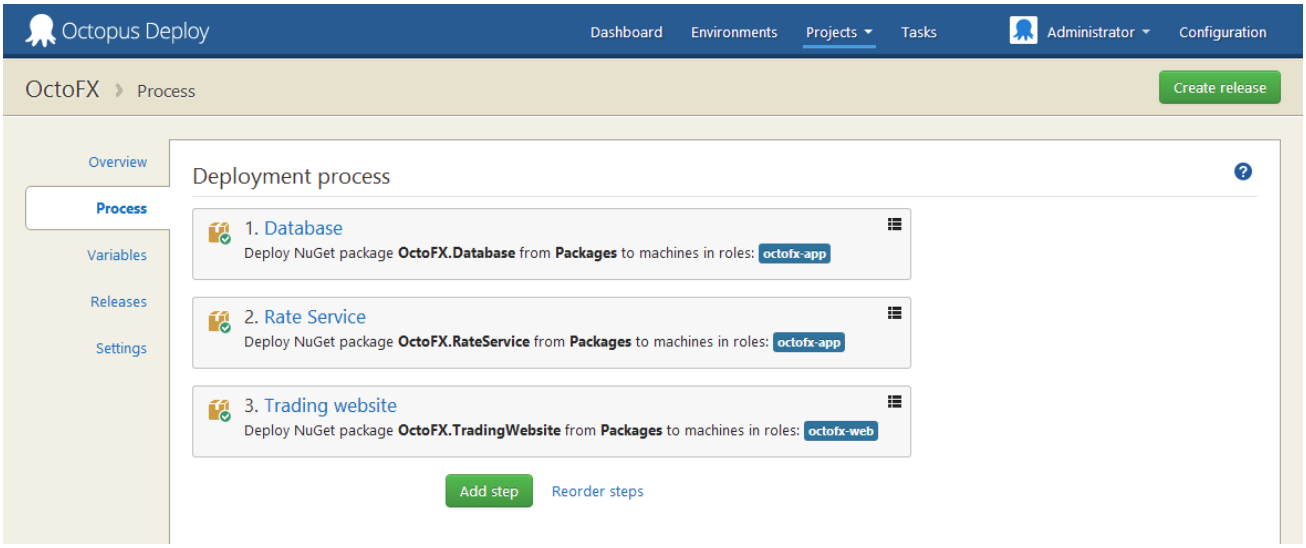
We designed Octopus to fit into teams that follow agile delivery practices. Octopus compliments your existing delivery pipeline:



- **Developers commit their code into your existing source control system**
  You might be using Git, Team Foundation Server, Subversion or Mercurial; the choice is up to you.
- **Your CI/build server compiles the code and runs unit tests**
  Again, you might be using TeamCity, Jenkins, Bamboo, Team Foundation Server or CruiseControl.NET; the choice is up to you.
- **Your application is packaged into a NuGet package**
  When the build is done, your CI/build server bundles all of the files – the binaries, images, scripts, configuration files and so on – needed to deploy your application into a NuGet package

The job of Octopus, is to take these packages and push them to the machines that they will be deployed to.

DEPLOYMENT PROCESS



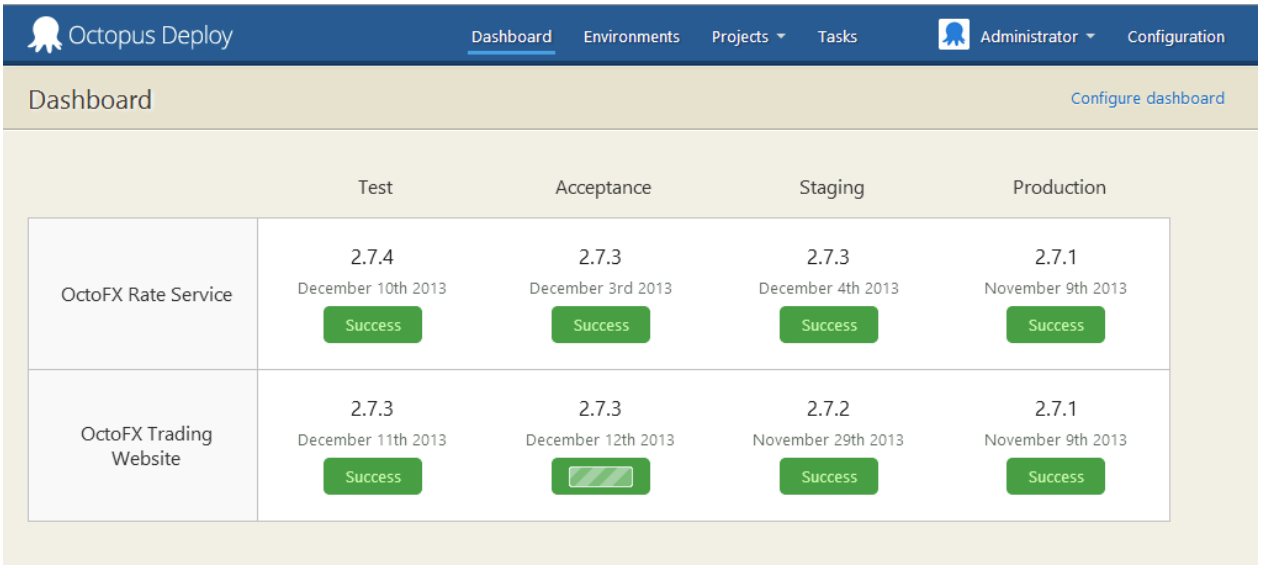
In the Octopus Deploy server, you define the steps or "recipe" for your deployment. The steps for a multi-tier e-commerce website, for example, may consist of:

- ✔ Deploying changes to a SQL database
- ✔ Deploying a Windows Service for back-end processing of orders to two application servers
- ✔ Deploying an ASP.NET MVC web application for the front-end to five web servers

Using permissions, you can then allow other members of your team to queue deployments; for example, you might enable testers to deploy applications to a test environment, but not to production. This approach means that even if different people are triggering the deployments, the deployment process is still consistent.

> "We've gone from manual to automatic and we're reaping the rewards. We're continuously deploying our projects to various environments on a daily basis, our developers are spending more time coding and less time fixing broken deployments. As a result of all this our clients get to see our progress much sooner than before, and we're able to get feedback much sooner than before."
>
> Tom Hall
> Senior Software Engineer
> Isobar

DEPLOYMENT DASHBOARD



As discussed on page 4, one of the downsides of manual deployments is the lack of transparency into the state of deployments. In Octopus Deploy, we provide a shared deployment dashboard that team members can use to easily answer questions like:

- What version of an application is deployed in each environment
- When the application was last deployed, and who deployed it
- How long the deployment took
- What went wrong with the deployment

This web-based interface is also where deployments are configured and initiated, where configuration variables are managed, and where environments and machines are maintained.

Conclusions

This whitepaper examined the benefits of automated deployment, based on the responses to a survey of firms that had recently adopted an automated deployment solution. Overwhelmingly, the survey found that deployment automation had reduced the risks involved in deployment, enabling customers to deploy more frequently and reliably.

Implementing an automated deployment solution may not be free, but the benefits of reduced risk, increased reliability and transparency compared to manual deployments make it a very compelling investment.

For more information about Octopus Deploy, visit: octopusdeploy.com. You can download the free Community edition of Octopus Deploy to get started with automated deployments today, or you can register for a fortnightly demo to see the software in action.

Happy deployments!