CASE STUDY

LEGACY APPLICATION TRANSFORMATION

Contents

Introduction

Companies frequently evolve faster than their IT systems or Infrastructure and sometimes applications get forgotten. These applications are often key to the business and upgrading them is not as simple as just buying a new bit of hardware or software, it is generally intrinsically linked throughout the organisation with important touch points.

Not all legacy systems can be quickly re-written into cloud native applications or microservices. Even if they could it would more than likely present its own challenges.

- Do you have the skills within the team to do this?
- How much will it cost?
- How long will it take?
- Are you better off investing that time and money into new features?

A well-managed monolithic application can still work in the modern age. In many cases, the key challenge to tackle becomes – speed of deployment. How do you push new code out quickly, when you run a legacy, monolithic system?

The Challenge

One of our clients had a legacy .NET web-based application, which they sell to many customers. The customers love the product and it attracts new business every year. The challenge is that the core of the system, having been coded in the late 90s, it is taking an increased amount of time to carry out deployments to support new customers.

The sales side of the organisation is attracting more customers than their technology can deploy to, within an acceptable timeframe, creating a bottleneck, having a detrimental effect on the business's agility and as such, sales.

Additionally, the internal developers use their own build/release solutions for testing purposes, none of which is shared with the project team who must deploy the product.

To make things worse, support teams find issues and manually fix them on the servers, without informing the developers. That means that the next customer is likely to have that same issue at some point which will again be fixed by someone in the support. Master branch was not used and always known "good" because of the size of the application, it made merge days go on forever. Developers have decided to use tags

Finally, what the Quality Assurance team tests is not necessarily what is being deployed. This cannot be controlled with over 100-page documents that provide instructions on how to configure the application nor with the manual packages that are copied and installed.

To build a single server with the application running on it would take up to 4 hours. A typical deployment would include 6-8 servers and up to 32 hours.

Consult

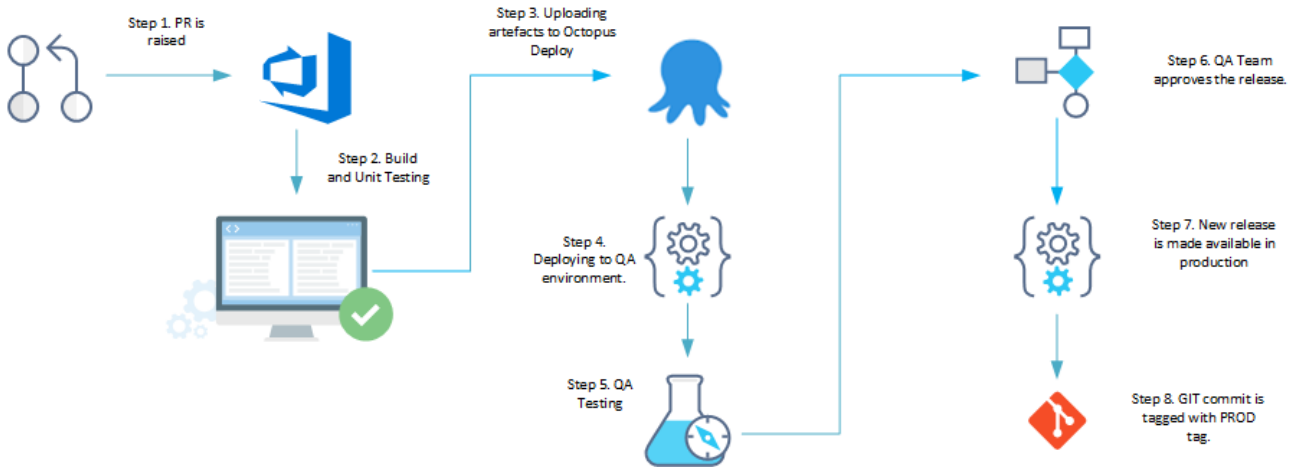
The client took advantage of our initial free consultancy offer and one we understood the challenges, we assured them that we could assist in the resolution.

Through discussion, it was decided that the introduction of a simple Continuous Integration/Continuous Delivery (CI/CD) pipeline was needed to ensure that there was a solid, automated process in place to replace the manual steps. AppDS spent some more time with the client and evaluated what they already had in place. The fact that the client ran across multi-tenant environments meant that AppDS came up with a hybrid solution for them using Azure DevOps and Octopus Deploy.

The following steps were agreed:

1. After a developer finishes working on a feature/issue – a Pull Request (PR) is raised to merge the code back into the master branch.
2. As a part of PR an automated build process would carry out the Unit tests and would produce the artefacts. If the PR is approved – the code is then merged into the master branch.
3. An Azure DevOps release pipeline would kick in and would upload the artefacts to Octopus Deploy and would create a new release in the Development (DEV) channel with a -dev suffix.
4. The release would be automatically deployed to a Quality Assurance (QA) environment as well as the servers developers use for their own testing.
5. QA team would carry out their standard testing procedure and would either approve or decline the release.
6. If QA team declined it – the release goes no further. An automated Microsoft Teams message is sent to developers notifying them of the status.
7. A new release is made available for all tenants with a -prod suffix via the live channel.
8. A GIT commit is tagged with a “PROD” tag.

Graphical representation of the process



Manage

The project engagement was over a 8 month period, during this time, AppDS worked with the client stakeholders to ensure that the solution worked in all area's so that we maximised the time saving and importantly that the application remained stable.

At the end of the process, the client had a well-tested release on Octopus Deploy ready to go out to different clients. By using per-tenant Octopus Deploy variables, we were able to manage different configurations of each client without having to maintain repositories of different configurations

Deliver

We were able to achieve all core objectives set out at the beginning of the project:

1. Fully automated release flow
2. No major changes to development process including the way master branch is used
3. A well tested release clearly marked with a suffix
4. A well tested release only available to be deployed in production environments
5. Dev releases only available in dev environments
6. Fully automated deployment and configuration of the application using per-tenant variables
7. Reduce deployment errors
8. Reduce amount of time needed to configure servers for each customer

Succeed

Using the automated deployment approach, our client was able to reduce the time required to a deploy a single server from 4h to 30 minutes. With our solution, you can deploy to multiple servers at the same time using the automated solution, as such, the total amount of time required to deploy a 8-server farm dropped from 32hrs to 30 minutes. In addition, as deployments were carried out using the same packages that were tested by the QA – there was a large reduction in deployment errors.

Through automation, we were able to modernise the deployment of a legacy application and provide our client with a competitive advantage over their competition.

If you are in a similar situation, AppDS will offer a free initial consultancy to determine the root cause, provide a fixed cost plan and resolution.

Resources

You can find a couple of handy scripts that we wrote as a part of this engagement on our GitHub page - https://github.com/AppDSConsult