

Foize is an emergent startup that operates globally from the Netherlands. Foize creates software solutions for Digital Social Communication, Geo-fencing, Phone as a Sensor and other location-based innovations for both enterprise customers and consumers.

One of their applications, 24Coms, adds geo-tracking data to conventional communication mechanisms, including information exchange and notifications. In order to meet rapidly growing demand for these services, Foize took a strategic decision to start moving from a hybrid solution, which combines some cloud-based services with on-premises resources, to a fully cloud-based solution. However, since this would take some time, Foize partnered with Microsoft to run a 3-day Hackathon in order to improve their overall model and approach.

## Value Stream Mapping

The Hackathon started off with an exercise called Value Stream Mapping (VSM). After examining the value stream from inception of ideas all the way to deployment to production, the team identified three key focus areas:

Don’t try to change everything at once. Focus on small changes that bring immediate value.

“

1. Improve current DevOps Processes
2. Create a blueprint for future service cloud migrations
3. Keep production running smoothly while the team experimented

Or to phrase these in DevOps terminology:

1. Infrastructure as Code
2. Continuous Integration and Continuous Delivery
3. Staging and Production Environments

Foize's team historically used a mix of tools, including Atlassian Bamboo with build servers in Amazon Web Services (AWS). Since the team decided to use Azure for their cloud-based services, the team also decided to move to Visual Studio Team Services for source code management, build and release services.

## Using Visual Studio Team Services (VSTS) to Deliver CI

The team started off by checking their current code into a VSTS Git repository. This worked great for the team, since not all of the developers use Visual Studio and the Microsoft stack for development - there is also an Android team that builds Android apps. The non-Microsoft developers were pleasantly surprised at how easy it was to integrate their workflows into VSTS while continuing to remain productive.

Once the source code was in the Git repo, it was simple to create a build that would compile the web app and worker role code and run unit tests. The team configured the build to run whenever a commit is pushed to the server - Continuous Integration was configured!

## Infrastructure as Code

The cloud services communicate using the Azure Service Bus via Queues and Topics. One of the goals the team wanted to achieve was to be able to describe their infrastructure as code so that setup and configuration in any environment (new or existing) could be automatically enforced during deployments.

Make sure your goal is measurable. Nothing helps drive DevOps adoption like seeing measureable improvements.

“

Initially the team attempted to use Azure Resource Manager (ARM) json templates to specify their infrastructure as code. Unfortunately, Queues and Topics were resources that had not yet been released in the ARM template schema. The team had to fall back to using a PowerShell script to create/update Service Bus Queues and Topics. The team discovered that in order for the script to work, they had to reference Microsoft.ServiceBus.dll. This had to then be added to the source repository so that the build/deploy agents have access to this assembly.

## Continuous Integration (CI) to Continuous Delivery (CD)

The team then set about integrating the Infrastructure as Code PowerShell scripts into their build so that the build would compile and test source code, then configure the target environments, deploy worker roles and finally deploy the web application. The team had to work around a bug in the out-of-the-box Azure Cloud PowerShell Deployment task - but since the source code for all the VSTS Build Tasks is publically available in a GitHub repo, the team was able to customize the task quickly.

## Instant Value

One easily measurable outcome from the Hackathon is the Mean Time to Recovery (MTTR). Before configuring the Continuous Deployment pipeline, the team would take between 2 and 4 hours to deploy the application once a bug was fixed. With the new build, the application deployment time reduced to around 15 minutes - an 8 to 16x improvement.

Besides realizing immediate improvement, the team now have a blueprint they can reproduce to bring better DevOps to other applications and projects. Their success is inspiring them to keep improving and to keep changing for the better.

This case study is for informational purposes only.

MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Document published November 2015

*Ronald Koster (Android Dev, DBA), Leo Winder (Owner, Ops), Michael Vlaar (C# Dev, Ops) in front of their Value Stream Map.*