PAGE INTRODUCTION & TITLE PAGE

SUGGESTED TIMELINE
TIME START: 00:00:00   TIME END: 00:01:00

SUGGESTED NARRATIVE
This is intended to be a very brief case history of our journey into data-driven Performance Testing & Engineering.

- After touching on just some of the reasons why Performance Testing was an excellent candidate for data transformation.
- What Data Driven Testing has delivered for us.
- A quick look at some of the technology we used, though this session is by no means intended to be about technology per-se, it's about benefiting from decision science.
- How we used DevOps pipelining techniques to achieve the event and context driven actions and behaviours across the stack.
- Where we found the value in the journey.
- Some lessons learned.

WHY PICK PERFORMANCE ENGINEERING

SUGGESTED TIMELINE
TIME START: 00:01:00   TIME END: 00:03:00

SUGGESTED NARRATIVE
Incontestably the best choice to commence our journey.

We knew there were quality issues to be addressed in terms of cost per test, value per test, difficult human based manual processes.

We knew that there was an enormous amount of operational/machine and unstructured context data which was not being fully exploited.

We knew that whatever value we did get from a performance test led to deeper issues downstream, for example the test might identify an issue, but the data was not being collected in such a way as to allow developers or test environment management to be able to actually fix it.

THIS IS THE MAIN THEME WE WANTED TO HUNT DOWN WITH DATA ANALYTICS.
If an anti-pattern happens – we need to be able to understand and extract value from it. Immediately.

As can be seen there are a hideous number of variables to be considered in Performance Testing, from test data, to environment and component versioning. While humans can and were making decisions around what to do next, based on their assessment of the situation – this was slow and error-prone.

We decided to build data-driven sequences to make these decisions for us and to enact on the events and their context while baking in lots of scope for feedback loops to enable our delivery chains (Developers/Project Managers and others) to consume the data and build their own solutions on a solid consistent foundation.

The solution is open. We designed it this way to ensure that the insight we are mining from Machine Data + Events + Context is presented as structured and consumable by the next part of the pipeline – whether that part of the pipeline is core performance testing or being designed and built by another team.

DATA DRIVEN PERFORMANCE ENGINEERING

SUGGESTED TIMELINE
TIME START: 00:03:15   TIME END: 00:07:00

OUR SHOPPING LIST / GOALS

This is important…… the one thing we are not getting into in this presentation is the test itself – the test is irrelevant … it's the surrounding activities that matter here. Triggers, test customer data, automation around the test, extraction of the test result data, analytics, movement of information…. All these things surround the test – but the test could be anything at all.

SUGGESTED NARRATIVE

Walk through each of the items on the list and detail what our objectives were for each.

Special notes:
Hammer home the importance of understanding the 'context' of things. We might be able to measure how long something took, but we needed to consume all possible data dimensions, for example: How busy was the component when it took that long, how mature was the component at the time, how many exceptions were being generated at that time, was the test data within tolerance at the time…. How different to the baseline was that response.

REMEMBER THE FOUR UNBREAKABLE TENETS OF PERFORMANCE:

LATENCY – Measuring how long something takes.

TRAFFIC / THROUGHPUT – Measuring how MUCH we can move through a system BEFORE the system begins to break down.

PERFORMANCE – While the THROUGHPUT is being achieved from above – what are the key metrics for how the systems are performing. TPS/Queuing Time Etc

ERRORS – While the THROUGHPUT is being achieved from above AND while the PERFORMANCE from above is being achieved, what is the average ERROR/EXCEPTONS rate?

IMPORTANT:

Context IS data too. Context can turn banal data into hero data. HERO DATA is a good line….

TOPOLOGY HIGHLIGHTS

SUGGESTED TIMELINE
TIME START: 00:07:00 TIME END: 00:09:00

TOPOLOGY HIGHLIGHTS

SUGGESTED NARRATIVE
Caveat that this is not intended to be a deep dive into the tech stack, but it does require some articulation.

We used Jenkins to schedule sophisticated pipelines using Java/Python and Go Code to achieve the orchestration and choreography we need.

We used Elastic and Influx with their preferred agents (Filebeat and Telegraf respectively) to ship the data into clusters.

We applied very many checks to the pre-test conditions.

We applied in flight checks to ensure the expected conditions while the test is running, including the expected changes within the clusters.

We applied multiple post-test processes to harvest and analyse results – then take any action required automatically as part of the data driven pipeline.

The machine data does not lie.

We needed DevOps tooling and mindset combined with the curiosity of a Data Scientist. Melted down together we achieved Performance Science and Operations.

PIPELINING & REINFORCEMENT OF THE MAIN MESSAGE THEME

SUGGESTED TIMELINE
TIME START: 00:09:00 TIME END: 00:11:00

SUGGESTED NARRATIVE

REINFORCEMENT OF:
Testing running as a machine. End to end. Data in. Insight and actionable content out.

Now we have the data … we can analyse and begin to make predictions and forecasting.

TIMELINE & VALUE CHART

SUGGESTED TIMELINE
TIME START: 00:11:00 TIME END: 00:13:00

SUGGESTED NARRATIVE

Walk audience through the business processes associated with buy-in and alliance building.

Talk about the discrete platforms and how they all had their own ideas for innovations and needed to be 'won over'.

Talk about the data extraction and how this is hugely important… as the parsing and structuring of the data early makes it easier later.

The analysis…. This is data science based and is designed to produce actionable content for our customers and ourselves. This allows us to DRIVE PERFORMANCE.

The decision science… this is about doing the analysis to expose new opportunities, shine light into oddities and describe correlations and factors in smarter ways.

The transformation speaks of re-using our new insight to change how development is done. Change the delivery model. Predict issues and problems and essentially leverage the data driven decisions in order to work smarter and help our customers.

NEW JOURNEY FROM HERE

SUGGESTED TIMELINE
TIME START: 00:13:00 TIME END: 00:15:00

SUMMARY

SUGGESTED NARRATIVE

Walk through key points.

ODD:
If you trust your data (how it was procured and the quality of its enrichment) then what it is telling you is probably true.

But there is a delta between the truth as the component sees it and the truth as viewed through the lens of the entire user journey.

Components do not tell deliberate lies, only people can program them to do that.

COURAGEOUS:
This is a call to arms. An incitement – give people the licence to fails and they will fail… they will also learn.

The best way to become Data Driven is to make a start.

ADDITIONAL INFORMATION

A WORD ABOUT THE TOOLS SELECTED

ELASTICSEARCH + LOGSTASH + FILEBEAT + KIBANA
Industry leading analytics and search solution for managing, blending and analysing textual data with advanced graphing and correlation capabilities.

INFLUXDB + TELEGRAF + KAPACITOR + GRAFANA
Industry leading time series metric data ideally suited to collecting operational values from components and hosts.

JENKINS
Industry leading automation orchestration tool. Loved by DevOps all over the world. Fantastically expressive automation solutions.

JAVA (Spring) & PYTHON
Mature and trusted programming languages and frameworks offering superb features for pushing and pulling data and processes. Easy to learn, easy to buy the skills and a pleasure to develop with.

BLAZEMETER
Modern cloud-based orchestration of ultra-high load injection. Superbly integrated into Java and JMeter.

SLACK
Industry leading collaboration tool. Easily build and deploy bots and chat based digital assistants. Keeps everyone on the same page each day.

TENSORFLOW
Google package for Machine Learning. We are using this for forecasting and pattern analysis.

QUESTIONS

SUGGESTED TIMELINE
TIME START: 00:15:00   TIME END: 00:17:00

Q1:

What was the hardest part of the journey with Performance Engineering becoming Data Driven?

A1:

There were many challenging situations and decisions to be made.

It's hard not to stay true to requirements when the current technology is moving so fast. It's easy to be influenced by what the technology can do, rather than what you actually 'need'. We had to continually return to home base and say 'we can do all that cool stuff later, for now lets just focus on the problem statement'

Another problem, due to the extreme complexity of the banking architecture and diverse platforms was deciding which customer journeys we would prioritise first. In the end we chose to go with incoming projects and programmes which reflected the changing priorities for the bank itself, such as Open Banking API's and some of the other transformation programmes. Along *the way we touched surfaces that made joining it all up later easier while providing some early value to lots of areas.*

NOTE: This question should only be asked if there were questions or interest piqued during the tools slide.

Q2:

What factors influenced your technology stack selection?

A2:

The post DevOps world is now flooded with innovative and shiny tools, with so many offerings now available it was difficult to narrow things down. It would be virtually impossible to try out everything before committing. Instead we decided to go with solid, well established tools and solutions which were specifically designed to solve the problem for each well-defined problem domain. For example, choosing to develop much of the Data Science product out of Python made clear sense as Python is emerging as the market favourite in that space. Also, Python skills are fairly straightforward to procure from the skills markets. Similarly, for developing our own APIs to push and pull information pertaining to the solution, it was a simple choice to go with Java and Spring as they make it so simple, fast and repeatable.

There will always be new tools which look shinier and more fun; that’s the market we are in. Rather than getting jealous about what the kid next door got for Christmas – we are focussing on making our stack produce the best work for us while protecting and maintaining the modularity and compartmentalization we have built into the stack which allows us, should we wish to, to replace tech A with tech B.