DevOps for the Urban IoT

Conference or Workshop Item

DevOps for the Urban IoT

John Moore
The Open University
Milton Keynes, UK
john.moore@open.ac.uk

Gerd Kortuem
The Open University
Milton Keynes, UK
gerd.kortuem@open.ac.uk

Andrew Smith
The Open University
Milton Keynes, UK
andrew.f.g.smith@open.ac.uk

Niaz Chowdhury
The Open University
Milton Keynes, UK
niaz.chowdhury@open.ac.uk

Jose Cavero
The Open University
Milton Keynes, UK
jose.cavero@open.ac.uk

Daniel Gooch
The Open University
Milton Keynes, UK
daniel.gooch@open.ac.uk

ABSTRACT

Choosing the right technologies to build an urban-scale IoT system can be challenging. There is often a focus on low-level architectural details such as the scalability of message handling. In our experience building an IoT information system requires a high-level holistic approach that mixes traditional data collection from vendor-specific cloud backends, together with data collected directly from embedded hardware and mobile devices. Supporting this heterogeneous environment can prove challenging and lead to complex systems that are difficult to develop and deploy in a timely fashion. In this paper we describe how we address these challenges by proposing a three-tiered DevOps model which we used to build an information system that is capable of providing real-time analytics of Electric Vehicle (EV) mobility usage and management within a smart city project.

CCS Concepts

•Applied computing → **Enterprise computing; •Computer systems organization** → *Real-time systems;* **•Software and its engineering** → *Designing software;*

Keywords

DevOps; Urban IoT; Smart City; Solar Energy; EV Mobility

1. INTRODUCTION

Many cities around the world are expected to grow significantly over the coming years, creating unsustainable pressure on key local infrastructure, particularly transport, energy and water. The objective of a smart city initiative such as MK:Smart, is to develop novel approaches to manage future growth and to make Milton Keynes more sustainable. The MK:Smart project contains work streams such as Energy, Citizen engagement and Education. As part of the Energy stream we have a goal to research and gain an understanding of the concept of self-sufficient mobility. More specifically, we wish to examine how Electric Vehicle (EV) owners use solar energy to fulfill their transportation needs. To achieve this we have built a system to collect and examine the use of stored solar energy within a home and EV.

The technology required to support this effort is complex and involves staged deployment starting with a select number of trial participants. Figure 1 plots the households taking part in the energy trial within MK:Smart across the city of Milton Keynes. A key challenge for any smart city initiative is being able to scale up a trial infrastructure to an urban scale and sustain this growth during deployment. In this paper we focus on this challenge from a technological perspective and describe an approach to scaling up a real-time information system from its trial deployment. Our approach is based on a layered model influenced by the practice of DevOps within enterprise computing. According to Gartner [4], "DevOps emphasizes people (and culture), and seeks to improve collaboration between operations and development teams. DevOps implementations utilize technology – especially automation tools that can leverage an increasingly programmable and dynamic in-

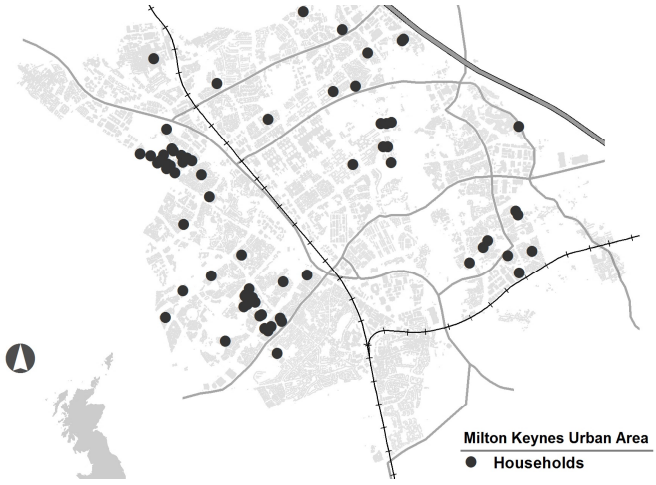

Figure 1: Household participation within the trial

Urb-IoT'16, May 24-25, 2016, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/2962735.2962747

frastructure from a life cycle perspective." Some popular DevOps technologies include Docker [8, 7] for software deployment, Jenkins [11] for continuous integration and Puppet [6] for configuration management. We will focus on Docker because we are most interested in how to deploy software in a scaleable way.

2. BACKGROUND AND RELATED WORK

It is now possible to purchase IoT devices which contain embedded networking stacks that connect directly to publisher/subscriber service provider companies such as PubNub [1]. However, it is unlikely that a large IoT project will be able to exclusively adopt this approach. Within our project, we can classify data as originating from either physical hardware such as a mobile phone or software-oriented sources such as cloud APIs provided by the manufacturers of the hardware used. Furthermore, the data that is collected comes in a range of formats from semi-structured text in flat files to more structured data such as XML retrieved over web protocols. Managing the collection of this data per user in a scalable way requires a systematic approach or methodology. However, current literature within the field of IoT tends to focus on low-level details such as the physical layer communication, messaging protocols used [3, 14, 10] and how to scale the communications architecture [1, 2]. At the other end of the spectrum, there has been research conducted on designing technology-oriented frameworks and architectures to support the development of smart cities [5, 13, 9]. Rather than base our work on these contrasting approaches, we demonstrate how DevOps can be applied to turn traditional information systems into IoT information systems. We achieve this by combining well known IoT implementation techniques into a simple three-tiered model.

3. ARCHITECTURE

We will describe in detail what we consider are the three key components when building a real-time IoT system using a DevOps approach. Namely, containerisation, publisher/subscriber messaging and unified logging.

3.1 Containerisation

Containerisation can be viewed as a more lightweight alternative to virtualisation popularised by the technology Docker. In Docker terminology, a container is a sandboxed environment for running software. Multiple containers can be built based on different operating systems and can be deployed on a single host machine. A container can run on any machine which supports Docker which makes migrating software from different hosting solutions straightforward. Containerisation eliminates the need for running and maintaining a complete operating system and kernel which is required when a virtual machine is deployed. In addition to this simplification, Docker provides a complete ecosystem for easily managing and working with containers. Figure 3 summarises some of its key features. A Docker image is described in a Dockerfile file which details its build requirements. This includes what operating system to use as well as what software packages to add and what scripts to execute. A registry exists containing a wealth of predefined build configurations. By issuing commands such as those shown in figure 3, it is possible to build an image and run it on the host system as a container. Although each container runs in its own secure sandbox it is possible to link containers so that they can share information. In our system each user can provision and configure their own container to act as a virtual hub that can collect the data and send it to our publisher/subscriber infrastructure. This container also provides code to carry out event processing on the data stream with the results being communicated back to the user. This is implemented in a traditional DevOps fashion by writing a series of small scripts per event stream that are managed within a traditional Unix-style environment. Once we have a working system for a single user we simply need to replicate this behaviour by building and running another container.

3.2 Publisher/Subscriber messaging

A key consideration when designing any pub/sub architecture is how you handle the server-side channel subscriptions as this helps manage the overall complexity of your server infrastructure. Ideally you will have a fixed number of server-side channel subscriptions which do not increase as the number of publishers joining the system increases. We use a single server-side subscription to a log channel. This channel is used to collect all household data for every participant. If the volume of messages flowing through this single channel becomes too high it is possible to scale this by introducing more log channels and/or servers and reading the channel messages in a predetermined fashion such as round-robin. The log channel can be visualised as raw data coming into the system that will processed and sent back out to an individual household as high-level events. The information sent back takes place over a channel supporting push notifications to services such as those offered by Apple, Google and Microsoft. The purpose of a push channel is to allow mobile platforms to deliver messages in the form of notifications through to the operating system when the client application is not running or suspended in the background.

3.3 Unified logging

Unified logging [12] abstracts the problem of routing data received from multiple sources to multiple potential consumers of the data. Technologies such as fluentd [2] achieve this by providing a consistent interface layer controlled by a declarative language. This makes it possible to write a single configuration file that works with numerous technologies. Additional technologies can be supported by developing plugins. In our system we route from a PubNub log channel to MongoDB and the Elasticsearch, Logstach, and Kibana (ELK) stack. MongoDB records all the JSON received which is then processed in real-time to generate events back out to users. ELK is used to provide real-time data analytics on the log channel such as live heatmaps of EV location events. The following section will showcase some of this work.

[1] http://pubnub.com

[2] http://fluentd.org

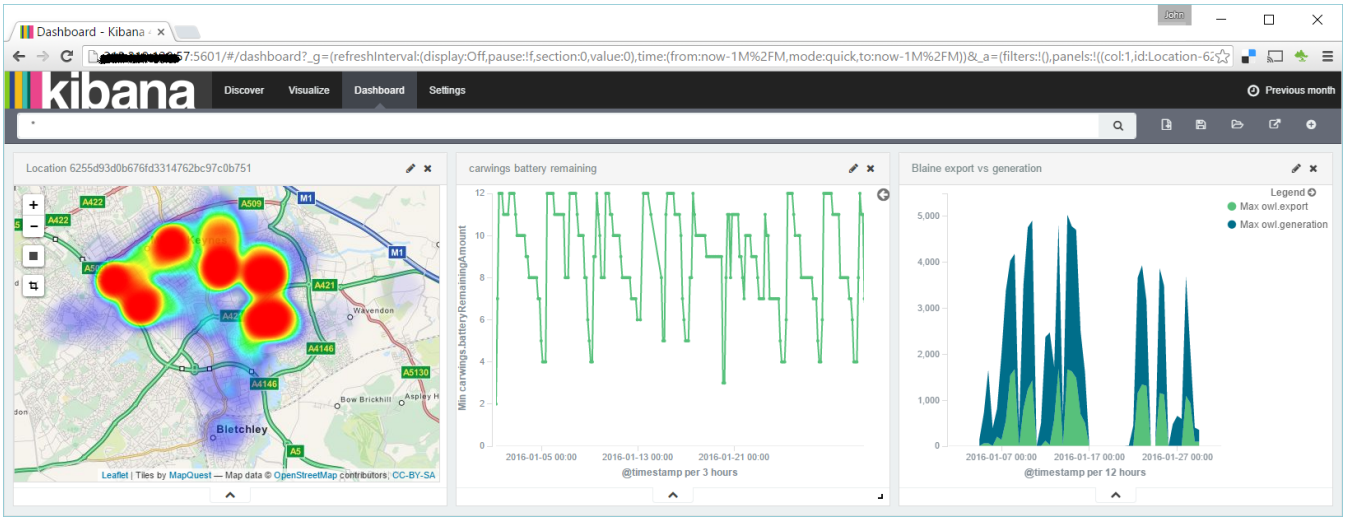

Figure 2: Kibana dashboard

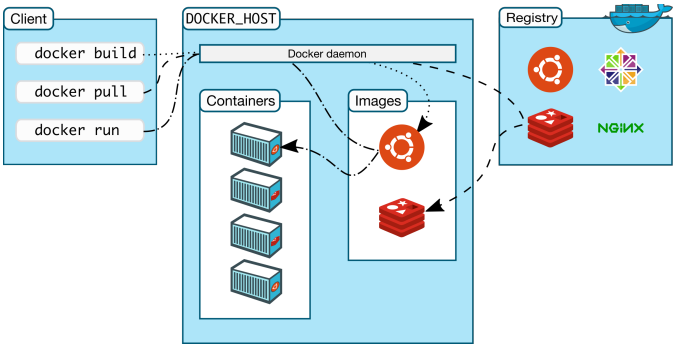

Figure 3: Provisioning services using Docker

4. TECHNOLOGY DEMONSTRATOR

Figure 2 is a screenshot of the data visualisation tool, Kibana, which is built on top of Elasticsearch. We use this tool to provide real-time data analytics of each participant in the trial in the form of a dashboard. The information shown is for a specific participant in our trial over the period of one month. The tool provides flexibility to examine data across different time periods as well as graph data as it is received in real-time.

The first graph shows an EV mobility heatmap generated through the detection of an iBeacon situated in the EV. Each time the driver enters or exits their vehicle we are able to capture the location of this event using an application running in the background of their mobile phone. We can view the mobility habits of an individual participant of the trial or plot the pattern of all participants within the city. The second graph shown plots the state of a participant's EV battery. Monitoring this data over time provides us with information on EV charging habits. For example, we can see that this particular EV owner tends to keep their battery level above 50%. This data originates from Nissan's telematics service. The third graph shown plots the amount of solar generation (shown in blue) and how much of this is exported back to the grid (shown in green) for a single home. This user is exporting back to the grid as they have not yet been fitted with battery storage capability. The data is collected using OWL Intuition energy monitoring hardware [3]. By looking across all three graphs we can answer questions such as: based on mobility habits is it possible for this participant to power their EV by using solar generation alone and can this be supported by adding latent solar storage in the form of a battery in the home? The technology we have described can not only help provide these kind of answers for one user but can also start to provide answers for many users across the city.

5. CONCLUSIONS

DevOps is a hot topic within enterprise computing. We believe this agile approach to building systems can equally be applied to tackling development projects within the IoT domain. To help with this, we defined a three-tiered approach to building a system which separates what we consider are the fundamental building blocks of an IoT information system. If tasked with building an IoT system, we believe this model provides a good starting point to focus your design efforts. Our model comprises of containerisation, publisher/subscriber messaging and unified logging. Containerisation is about employing technologies such as Docker to manage building systems that scale from one user to many. This is a key challenge that is encountered when scaling the deployment of a trial from a small number of participants up to an urban scale. Publisher/-subscriber messaging is about providing the underlying communication system required. Finally unified logging is about making sure the data is routed in the right directions so that it can be easily stored and analysed in real-time. To demonstrate our model we built an information system cable of providing real-time analytics on Electric Vehicle (EV) mobility and behaviour that forms part of a smart city project.

[3]http://www.theowl.com/owl-intuition/

6. REFERENCES

[1] A. Cenedese, A. Zanella, L. Vangelista, and M. Zorzi. Padova Smart City: An urban Internet of Things experimentation. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–6, June 2014.

[2] M. Collina, G. Corazza, and A. Vanelli-Coralli. Introducing the qest broker: Scaling the iot by bridging mqtt and rest. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pages 36–41, Sept 2012.

[3] K. Framling, S. Kubler, and A. Buda. Universal messaging standards for the iot from a lifecycle management perspective. *Internet of Things Journal, IEEE*, 1(4):319–327, Aug 2014.

[4] Gartner. Gartner IT Glossary > DevOps. http://www.gartner.com/it-glossary/devops. (last accessed: 04/02/2016).

[5] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami. An information framework for creating a smart city through internet of things. *Internet of Things Journal, IEEE*, 1(2):112–121, April 2014.

[6] J. Loope. *Managing Infrastructure with Puppet*. O'Reilly Media, Inc., 2011.

[7] K. Matthias and S. P. Kane. *Docker: Up & Running*. O'Reilly Media, Inc, 2015.

[8] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014.

[9] L. Sanchez, V. Gutierrez, J. Galache, P. Sotres, J. Santana, J. Casanueva, and L. Munoz. Smartsantander: Experimentation and service provision in the smart city. In *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*, pages 1–6, June 2013.

[10] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. McCann, and K. Leung. A survey on the IETF protocol suite for the Internet of Things: standards, challenges, and opportunities. *Wireless Communications, IEEE*, 20(6):91–98, December 2013.

[11] J. F. Smart. *Jenkins: the definitive guide*. O'Reilly Media, Inc., 2011.

[12] K. Tamura. Unified Logging Layer: Turning Data into Action. http://www.fluentd.org/blog/unified-logging-layer, Aug. 2014. (last accessed: 04/02/2016).

[13] R. Wenge, X. Zhang, C. Dave, L. Chao, and S. Hao. Smart city architecture: A technology guide for implementation and design challenges. *Communications, China*, 11(3):56–69, March 2014.

[14] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin. IOT Gateway: BridgingWireless Sensor Networks into Internet of Things. In *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, pages 347–352, Dec 2010.