Achieving Cloud Scalability with Microservices and DevOps in the Connected Car Domain

Dr. Tobias Schneider

Connected Car
Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen
tobias.schneider@elektrobit.com

Abstract: The connected car business has high demands on the exchange of data and files between the connected car on the road, and a variety of services in the backend. To solve current and upcoming challenges in the best way, a scalable and flexible architecture and team setup is needed. The paper has its background in the automotive IT industry at Elektrobit (EB) and describes our starting point, challenges and practical experiences. We have chosen microservices as an architectural paradigm in order to be able to replicate granular services for scalability, and to easily replace a deprecated service. For the development and operations of the services, we decided to have one team which is responsible for the backend infrastructure in the cloud. Also a DevOps culture was established. This allows us to deploy quickly services with increased operational efficiency and code quality.

1 Introduction

The Internet of Things poses new challenges on the software industry. This does not only apply to cars - but apparently connecting cars with backend services are one of the major topics in this field. The Google Android initiative for cars or the current successes in automated driving are just a few examples demonstrating this. One major aspect in connecting things is that there is little value by providing just the connection. The benefit from being connected always goes hand in hand with the provision of a specific service, which is generating the benefit from the connection.

In the context of software solutions for connected cars, this means that besides providing the connectivity for the car itself, and hosting a backend, one major task is to offer the right services for the right purpose. One example of such a service, could be offering online navigation. The routing algorithm is implemented by navigation software experts whereas the data exchange between the car and the backend, and the creation of the service is performed by connectivity experts.

2 Requirements on software development for connected car solutions

EB is a supplier for IT solutions in the automotive sector and offers solutions in a wide variety of areas, such as navigation, vehicle diagnostics and services around the car. As connectivity became a major topic in the automotive industry, EB founded the connected car domain in February 2014. The teams in this new domain were already experienced in agile methods like Scrum [SK04] or Kanban [AD2010] and continuous integration. The new key technology cloud computing was changing processes and requirements. From a technical perspective the following key points were identified:

- **Scalability (T1):** Some components are more often used or are more resource intense than other components. Hence, we need ways to replicate specific components in an efficient way.
- **Exchangeability (T2):** Any component in a major software solution is outdated at some point in time. We need a way to exchange a component easily without affecting other components (loose coupling).
- **Reuse (T3):** Many projects had a similar feature set for software to work with the cloud. Hence, we identified a set of standard services to be used in multiple projects.
- **Continuous deployment (T4):** A feature added to a service must be visible at once, so that the compatibility with other services can be tested automatically.
- **Code quality (T5):** The same code shall be used in different projects (T3). Hence, the demands on quality and genericity have increased.

From an organizational point of view the following points had the strongest impact:

- **Working mode (O1):** Is the development in a cloud environment compatible with agile methods, or is another working mode needed?
- **Team Setup (O2):** Have our team members the right skills? Do we need specialists within the team or specialized teams?
- **Self-Service (O3):** In cloud computing an instance or repository must be available at once, so no additional tasks in the process are acceptable.
- **Configuration Management (O4):** For every solution in the cloud an infrastructure is set up which needs to be reproducible for quality and controlling reasons.
- **Cost controlling (O5):** The scaling in cloud computing comes with new demands on cost controlling as instances are paid on an hourly basis and services like AWS Lambda [AC15] are payed per usage.

3 Microservices

The micro-service architecture is defined as developing an application as a set of small independent services, where each of the services is running in its own independent process [NS14]. Services communicate with some lightweight mechanisms like HTTP [FL14] and are deployed absolutely independently [NS14]. For us, the key reason to decide for an architecture based on microservices was the ability to replicate on demand across servers [FL14], which targets directly requirement T1. Moreover, an architectural paradigm with microservices enforces the single responsibility of an individual service and modularity via loose coupling (T2). From our experience the modularity of the service is also the key to reuse (T3). The requirements in different projects are most of the time similar but very seldom exactly the same. Additional requirements and features can be easily integrated into a microservice or outsourced in another service. When and how to perform this, is the crucial architectural design decision.

4 DevOps

Cloud computing changes the ways a team cooperates as a team and cooperates with other teams. The deployment of software is an essential task in every release of a software component. Features which were formerly provided by the IT department are now directly part of applications via infrastructure as code [LM12]. This is blurring the line between traditional software development and operations. There are different setups to target this:

- Close collaboration between the operations team and the development team
- Developers and operators in one team
- Every team member performs operations and develops software

EB has decided to follow the latter approach for teams working primary with connectivity and backend infrastructure. The reason behind this is that cloud computing and connecting things are the key expertise of the employees working on connectivity solutions. In this context every software developer needs to be familiar with these techniques (O2) and needs knowledge in performing operations. In bigger projects several teams are working together. In this case, one team takes over the DevOps part. The cooperating teams concentrate purely on software development. Microservice architecture in this context means that every service has a single responsibility although the algorithm behind the service might be more complicated (e.g. a routing algorithm).

Regarding the working mode (O1) we have decided to follow a Kanban approach. Tasks in operations often need to be performed at once when they are occurring (e.g. the outage of a server). In Scrum, activities need to be planned. This is not possible with this kind of tasks. Kanban opens the possibilities to work immediately on high priority targets and to include feature development. Every DevOps team has the rights to start and stop instances and services in their respective stage which allows controlled self-services on cloud resources (O3+O5).

With this setup the next step is to setup a continuous deployment process (T4). In this case automation is the key for quality and efficiency with techniques like infrastructure as code [LM12] being the key factors. We are successfully working with AWS CloudFormation [DP15] to have our infrastructure as code which makes the configuration reproducible (O4). Code can be deployed directly to test instances from the build server (T4). Only the release needs manual interaction.

5 Software reuse with inner source

One key decision to enable software reuse (T3) is the adaption of open source techniques into our development processes. We followed here the inner source (IS) approach [CR15] by establishing a software forge where the source code was made available to all developers. A forge is a central place where a well-documented code base is kept forever with the possibility to search [CR15]. This allows the usage of the code and the review of the code by all developers. Following Linus´s law “Given enough eyeballs, all bugs are shallow” [RE01] we can achieve a higher code quality (T5) then with the common four-eye principle alone. Our experiences showed that the code quality improved and that teams were starting to work together, especially on common tooling. A set of microservices is available via the forge enabling software reuse in other projects and improving the overall development time.

6 Summary

With our current setup EB has adopted to change and new environments by changing the culture in software development from a classic agile setup to a DevOps culture. This enables EB to cope with the new technical challenges of cloud computing. From an architectural perspective, the decision for microservices enables us to scale our services, and gives us the necessary flexibility. This is supported from an organizational perspective with a DevOps culture embedded in an agile mind-set and the quality improvements and the possibility of software reuse from the inner source approach.

Literature

[AC15] Astakhov V., Chayel M.: Lambda Architecture for Batch and Real-Time Processing on AWS with Spark Streaming and Spark SQL. AWS Whitepaper: https://d0.awsstatic.com/whitepapers/lambda-architecure-on-for-batch-aws.pdf (retrieved 07.12.2015), 2015

[AD10] Anderson, David J. *Kanban*. Blue Hole Press, 2010.

[CR15] Capraro M., Riehle D.: Inner Source in Platform-based Product Engineering. Friedrich-Alexander University Erlangen Nürnberg Technical Report CS-2015-02, 2015

[DP15] Dalbahanian P.: Overview of deployment options on AWS, AWS whitepaper, https://d0.awsstatic.com/whitepapers/overview-of-deployment-options-on-aws.pdf (retrieved 07.12.2015), 2015

[FL14] Fowler M., Lewis J.: Microservices a definition of this new architectural term, http://martinfowler.com/articles/microservices.html (retrieved 07.12.2015), 2014

[LM12] Loukides M.: What is DevOps?, O’Really Media Inc., 2012

[NS14] Namiot, D., Sneps-Sneppe M.: On micro-services architecture. *International Journal of Open Information Technologies* 2.9: p- 24-27. 2014

[SK04] Schwaber, Ken. *Agile project management with Scrum*. Microsoft Press, 2004.

[RE01] Raymond, Eric S. *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. " O'Reilly Media, Inc.", 2001.