Hybrid Cloud Migration Challenges
A case study at King
Master Thesis

Mikhail Boronin
Supervisor: David Johnson
June 14, 2020
Abstract

Migration to the cloud has been a popular topic in industry and academia in recent years. Despite many benefits that the cloud presents, such as high availability and scalability, most of the on-premise application architectures are not ready to fully exploit the benefits of this environment, and adapting them to this environment is a non-trivial task. Therefore, many organizations consider a gradual process of moving to the cloud with Hybrid Cloud architecture. In this paper, the author is making an effort of analyzing particular enterprise case in cloud migration topics like cloud deployment, cloud architecture and cloud management. This paper aims to identify, classify, and compare existing challenges in cloud migration, illustrate approaches to resolve these challenges and discover the best practices in cloud adoption and process of conversion teams to the cloud.
Acknowledgements

This work has been done in collaboration with King\(^1\). I want to thank the organization for providing all the required resources and information to be able to focus on this research.

I want to thank Swedish Institute\(^2\) for the support during my entire Master’s program. I want to thank the organization for all the support I have received during my Master program. Without this help, I would not be able to start and participate in the program.

I want to thank my supervisor David Johnson for his useful guidelines and advice he was giving me through the whole thesis writing. This work would have been impossible without his help.

I am grateful to Xiaochen Guo, and Carl Bring from King who initiated this project and were facilitating it with innovative ideas. They were always open for discussion, providing valuable information.

An important part of the work consists of interviews with several King employees. I want to thank everyone who participated in the interviews and made this work possible.

Finally, I would like to thank all my classmates and friends who were assisting me. Their reviews helped to analyze the report from different perspectives.

\(^1\)King.com - Play the Most Popular & Fun Games Online! (n.d.)

\(^2\)Swedish Institute (n.d.)
# Contents

List of Figures 5  
List of Tables 6  

## 1 Introduction 10  
1.1 Background ......................................... 10  
1.1.1 Overview ........................................ 10  
1.2 Cloud deployment models ............................ 11  
1.2.1 Service Models .................................. 12  
1.3 Aim of this work .................................... 13  
1.4 King ............................................... 13  
1.5 Research Plan ...................................... 14  
1.5.1 Research focus .................................. 14  
1.5.2 Questions ....................................... 14  
1.5.3 Topic Justification ............................... 14  
1.6 Delimitations ...................................... 15  
1.7 Thesis structure .................................... 15  

## 2 Literature review 17  
2.1 Cloud computing .................................... 17  
2.1.1 Essential characteristics .............................. 17  
2.1.2 Initial state of utility computing ..................... 17  
2.1.3 Current challenges in cloud computing .......... 18  
2.1.4 Hybrid Cloud .................................. 18  
2.2 Cloud deployment .................................... 19  
2.2.1 Environments .................................. 19  
2.2.2 Pipelines ...................................... 20  
2.2.3 Continuous Integration ............................ 20  
2.2.4 Continuous Delivery .............................. 20  
2.2.5 Namespaces .................................. 21  
2.3 Cloud management ................................... 21  
2.3.1 Infrastructure .................................. 21  
2.3.2 Imperative vs. Declarative approaches .......... 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.3 User benefits of GitOps</td>
<td>26</td>
</tr>
<tr>
<td>2.3.4 Version Control as a source of truth for all processes</td>
<td>26</td>
</tr>
<tr>
<td>2.3.5 Kubernetes</td>
<td>26</td>
</tr>
<tr>
<td>2.4 Cloud architecture</td>
<td>27</td>
</tr>
<tr>
<td>2.4.1 Cluster environments</td>
<td>27</td>
</tr>
<tr>
<td>2.4.2 Multi-tenancy</td>
<td>28</td>
</tr>
<tr>
<td>2.5 Migration</td>
<td>28</td>
</tr>
<tr>
<td>2.5.1 Cloud Migration</td>
<td>28</td>
</tr>
<tr>
<td>2.5.2 Application migration</td>
<td>29</td>
</tr>
<tr>
<td>2.5.3 SOA versus MSOA in Cloud Migration</td>
<td>30</td>
</tr>
<tr>
<td>2.5.4 Full enterprise migration</td>
<td>31</td>
</tr>
<tr>
<td>2.5.5 General Migration issues</td>
<td>33</td>
</tr>
<tr>
<td>2.5.6 Cloud Providers</td>
<td>33</td>
</tr>
<tr>
<td>2.5.7 Cloud Agnostic approach</td>
<td>33</td>
</tr>
<tr>
<td>2.5.8 Cloud cost</td>
<td>34</td>
</tr>
<tr>
<td>2.6 Summary</td>
<td>34</td>
</tr>
<tr>
<td>3 Methodology</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Study design</td>
<td>35</td>
</tr>
<tr>
<td>3.2 Approach</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Methods</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1 Interviews</td>
<td>36</td>
</tr>
<tr>
<td>3.3.2 Internal Knowledge portal</td>
<td>36</td>
</tr>
<tr>
<td>3.3.3 Developer forums</td>
<td>37</td>
</tr>
<tr>
<td>3.4 Data collection</td>
<td>37</td>
</tr>
<tr>
<td>3.5 Interviews</td>
<td>38</td>
</tr>
<tr>
<td>3.5.1 Workflow</td>
<td>38</td>
</tr>
<tr>
<td>3.5.2 Developer interviews</td>
<td>38</td>
</tr>
<tr>
<td>3.5.3 User Interviews</td>
<td>41</td>
</tr>
<tr>
<td>3.6 CI/CD Solutions analysis</td>
<td>45</td>
</tr>
<tr>
<td>3.7 Forums and other discussion platforms</td>
<td>46</td>
</tr>
<tr>
<td>3.8 Ethics</td>
<td>46</td>
</tr>
<tr>
<td>4 Empirical findings</td>
<td>47</td>
</tr>
<tr>
<td>4.1 Visualization</td>
<td>47</td>
</tr>
<tr>
<td>4.1.1 Word cloud</td>
<td>47</td>
</tr>
<tr>
<td>4.1.2 Charts</td>
<td>47</td>
</tr>
<tr>
<td>4.1.3 Concepts</td>
<td>47</td>
</tr>
<tr>
<td>4.2 Summary</td>
<td>55</td>
</tr>
</tbody>
</table>
5 Analysis

5.1 Changes in architecture ............................................. 57
  5.1.1 Gateways ......................................................... 58
  5.1.2 Firewall ......................................................... 58
  5.1.3 Overview ......................................................... 59

5.2 Declarative approach to infrastructure. GitOps Workflow .......... 59
  5.2.1 Provisioning ..................................................... 60
  5.2.2 Deployment ..................................................... 61

5.3 Separation of the environments ....................................... 61
  5.3.1 Approaches to tenancy ......................................... 61
  5.3.2 Rancher ........................................................ 62

5.4 Unification ........................................................ 62
  5.4.1 Helm ............................................................ 63
  5.4.2 King service .................................................... 63

6 Discussion ......................................................... 66

6.1 Sharing responsibilities among the teams ............................ 66
  6.1.1 Layers .......................................................... 66

6.2 Knowledge, Culture and Cloud Adoption ............................ 66
  6.2.1 Shift to DevOps approach ..................................... 67
  6.2.2 Vendor lock-in ................................................ 67
  6.2.3 Cloud Teams and Cloud Engagement ............................ 68
  6.2.4 Stack free approach .......................................... 68

6.3 Shift to third-party solutions ....................................... 68
  6.3.1 CNCF ............................................................ 68
  6.3.2 Open-source tools ............................................. 69

7 Conclusions ......................................................... 70

Bibliography .......................................................... 71

8 Appendix ............................................................ 78

8.1 Traffic ............................................................ 82
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Hybrid Cloud</td>
<td>12</td>
</tr>
<tr>
<td>1.2</td>
<td>Cloud service models</td>
<td>12</td>
</tr>
<tr>
<td>1.3</td>
<td>Thesis structure</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Word Cloud based on interviews’ transcripts</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>Challenges during cloud migration</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Positive impact of migration</td>
<td>49</td>
</tr>
<tr>
<td>5.1</td>
<td>Zones at King, Future state</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>Current networking state</td>
<td>59</td>
</tr>
<tr>
<td>5.3</td>
<td>Goal of the networking state</td>
<td>59</td>
</tr>
<tr>
<td>5.4</td>
<td>Global cluster overview</td>
<td>59</td>
</tr>
<tr>
<td>5.5</td>
<td>Infrastructure as code workflow</td>
<td>60</td>
</tr>
<tr>
<td>5.6</td>
<td>GitOps Deployment workflow</td>
<td>61</td>
</tr>
<tr>
<td>5.7</td>
<td>Multi-tenant, multi-environment cluster</td>
<td>63</td>
</tr>
<tr>
<td>5.8</td>
<td>Single-tenant, multi-environment cluster</td>
<td>64</td>
</tr>
<tr>
<td>5.9</td>
<td>Multi-tenant, single-environment cluster</td>
<td>65</td>
</tr>
<tr>
<td>5.10</td>
<td>Simple pipeline design</td>
<td>65</td>
</tr>
<tr>
<td>6.1</td>
<td>Architectural layers</td>
<td>67</td>
</tr>
<tr>
<td>8.1</td>
<td>Comparison of CI/CD Tools</td>
<td>79</td>
</tr>
<tr>
<td>8.2</td>
<td>Traffic overview</td>
<td>83</td>
</tr>
</tbody>
</table>
## List of Tables

2.1 Obstacles for cloud migration .................................................. 32  
3.1 Platform Developers interview summary ................................. 41  
3.2 Challenges revealed during interviews with frequency of their appearances .................................................. 42  
3.3 Positive concepts and opportunities revealed during interviews with frequency of their appearances .................................................. 43  
3.4 Target audience identification ................................................. 44  
3.5 Potential users of Hybrid cloud platform ................................. 45  
5.1 King Zones .............................................................................. 57  
5.2 Restrictions of traffic between zones ...................................... 62  
5.3 Approaches to tenancy ............................................................. 62
Keywords
"cloud migration", "hybrid cloud", "microservices", "devops", "on-premise to Cloud", "cloud architecture", "cloud infrastructure", "live migration", "kubernetes", "cloud providers", "private public cloud"

List of definitions and abbreviations

This section provides the definitions for terms and products, used in this work.

General definitions

**Kubernetes** - an open-source system for automating deployment, scaling, and management of containerized applications. *(Production-Grade Container Orchestration - Kubernetes n.d.a)*

Identity and access management (IAM) is the discipline that enables the right individuals to access the right resources at the right times for the right reasons. *(Identity and Access Management (IAM) n.d.)*

Interconnection is the physical linking of a carrier’s network with equipment or facilities not belonging to that network. The term may refer to a connection between a carrier’s facilities and the equipment belonging to its customer, or a connection between two or more carriers. *(Section n.d.)*

**SSO** - single sign-on

**DevOps** - approach to deployment

**GDPR** - General Data Protection Regulation, a new EU directive that will be enforced from 25th of May 2018. Read more at [http://www.eugdpr.org/](http://www.eugdpr.org/).

**PII** - Personally Identifiable Information as defined by GDPR.

**CNCF** - a Linux Foundation project that was founded in 2015 to help advance container technology and align the tech industry around its evolution.

**JWT** - JSON Web Token, an encoded and signed JSON string. Read more on [https://jwt.io/](https://jwt.io/).

**On-premise** - is installed and runs on computers on the premises of the person or organization using the software, rather than at a remote facility such as a server farm or cloud. *(Wikipedia n.d.)*

**SLA** - Service-level Agreement.

**Unique Request Identifier** - uniquely identifies a chain of requests. A chain of requests encompasses an external request, plus all the internal requests triggered as a result of it.

**Load Balancer** - device that acts as a reverse proxy and distributes network or application traffic across a number of servers.

**VM** - Virtual machine, a software-based computer that exists within another computer’s operating system, often used for the purposes of application testing and deployment.

**TSDB** - time series database
Environments: QA, DEV, LIVE

**Uptime** - is a measure of system reliability, expressed as the percentage of time a machine, typically a computer has been working and available. (*Wikipedia* n.d.)

**Orchestrator** is the automated configuration, coordination, and management of computer systems and software.

**Namespace** is a set of symbols that are used to identify and refer to objects of various kinds. A namespace ensures that all of a given set of objects have unique names so that they can be easily identified.

**Cloud provider specific terms**

**Google Cloud Platform (GCP)** - cloud platform, developed by Google (*Google Cloud Platform Overview* n.d.)


**BigQuery** - serverless and scalable cloud data warehouse designed for business agility. (*BigQuery: Cloud Data Warehouse — Google Cloud* n.d.)

**Shared VPC** - A Virtual Private Cloud (VPC) network solution developed by Google (*VPC network overview — Google Cloud* n.d.a)

**Cloud IAM** - an IAM solution developed by Google (*Cloud Identity and Access Management — Cloud IAM — Google Cloud* n.d.)

**Virtual Private Cloud (VPC) network** is a virtual version of a physical network, such as a data centre network. (*VPC network overview — Google Cloud* n.d.b)

**AWS** - Amazon Web Services

**GKE** - Google Kubernetes Engine

**Google AppEngine** is a Platform as a Service and cloud computing platform for developing and hosting web applications in Google-managed data centres. (*Wikipedia* n.d.)

**Enterprise specific/internal King products**

**Unified Platform (UP)** - an internal platform, containing software products designed to provide a central, high-quality, robust and extensible toolset for building, growing and monetizing enterprise products (games), with consistent, stable user experience.

**Unified Platform Hybrid cloud (UP Hybrid Cloud)** - internal King project, which intention is to extend UP ecosystem to GCP, more precisely to a production-ready multi-tenant Kubernetes cluster located in shared VPC.

**UPF Team** - the team at King, focused on the creation of UP and UP Hybrid Cloud.

**IE Team** - Infrastructure Engineering team at King, focused on on-premise infrastructure and UP Hybrid Cloud.
Incognito - Anonymization service for GDPR which strips away and/or substitutes all PII before storage in the data warehouse.

Kafka - An open-source stream processing platform developed by the Apache Software Foundation.

King-service - A specification of the requirements on a service running at King. Read more at the GitHub page.

King SDK - The SDK provided to client developers to connect and communicate with our games’ backend.

Machines-Meta - King’s in-house server inventory system that can be considered loosely as a service discovery system. Read more at the GitHub page.

Product - set of services that together provide a common, managed set of features satisfying the specific needs of a particular user segment.

Service - set of software functionalities deployed as a unit and reusable using its application programming interface (API) through calls over a defined protocol.

Unified Platform - product composed of all technology created by Shared Tech teams, exposed to end-users using the Unified Platform Portal.

Unified Platform Portal - multi-tenant end-user interface enabling interaction with the Unified Platform product.

Zone - A separated piece of infrastructure that often is split into three different environments/level 3 networks.

pFlight - internal Java framework
Rollo - internal IAM application
Releaso - internal tool for releasing software
Machines Meta - internal tool for machine discovery, used for deploying and maintaining software
KWS - King’s virtualization system based on Open Nebula.

3rd party tools
Grafana - monitoring tool(Grafana: The open observability platform — Grafana Labs n.d.)

Github - version control system used by King, internally hosted (Github Enterprise).

Minikube - Minikube is a tool for running Kubernetes locally. It is typically used for development and testing.

npm is a package manager for the JavaScript programming language.

Gradle - open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language instead of the XML form used by Maven for declaring the project configuration.

Jira is a proprietary issue tracking product developed by Atlassian

HA Proxy is open-source software that provides a high availability load balancer and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers.

Rancher is an open-source multi-cluster orchestration platform
Chapter 1

Introduction

1.1 Background

1.1.1 Overview

According to NIST (Peter M. Mell 2011), Cloud computing can be described as ‘a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models’

Cloud computing was introduced more than ten years ago (Armbrust, A. Fox, and R. Griffith 2009). It has been gaining popularity over the last decade with the majority of large enterprises using it to some extent. Organizations consider it as an efficient cost-saving model, which increases operational efficiency in comparison to on-premise infrastructure (Chang et al. 2016). Cloud provides the opportunity to use the exact needed amount of resources. Cloud computing paradigm allows workloads to be deployed and scaled-out quickly through the rapid provisioning of the virtualized resources. (Al-Dhuraibi et al. 2018) Cloud enables delivering computation as a utility with the features of elasticity, pooled resources, on-demand access, self-service and pay-as-you-go (Peter M. Mell 2011)

However, security, trust and privacy remain as a challenge for many enterprises, considering cloud migration. These issues hold many organizations from complete migration to the cloud (Ali et al. 2015). Cloud migration implies architectural changes in the applications and systems and therefore becomes a non-trivial task for development teams with a focus on monolith architecture. Migrating to the cloud through native cloud architectures such as microservices is a multidimensional problem and thus nontrivial (Balalaie et al. 2016a, Jamshidi et al. 2013)

Besides that, culture and lack of cloud knowledge and skills can become a significant obstacle for many enterprises with a long history of using on-premise infrastructure, as cloud introduces a shift to self-service approach to deployment (Tilkov 2015)

Enterprise applications are often faced with strict requirements in terms of performance,
delay, and service ‘uptime’. On the other hand, little can be said about the performance of applications in the cloud in general—the response time variation influenced by network latency, and the scale of applications suited for deployment. There has been significant interest in the industry in hybrid architectures where enterprise applications are partly hosted on-premise, and partly in the cloud. Hybrid architectures offer several advantages, which allow developers to choose, which components must be kept local, and which components should migrate is non-trivial. A key barrier to realizing hybrid migrations is the need to ensure that the reachability policies continue to be met. Multiple factors can motivate such hybrid deployments.

From a performance perspective, migrating the entire application to the cloud is likely to result in higher response times to users internal to the enterprise, as well as extensive wide-area communication. Replicating servers locally and remotely allows internal and external users to be served from different locations. From a data privacy perspective, enterprises may wish to store sensitive databases locally. This may, in turn, make it desirable to place other components that extensively interact with such databases locally to avoid wide-area communication costs and application response times. The critical requirement of reachability policy migration is to ensure correctness – if a packet between two nodes is permitted (denied) before migration, it must be permitted (denied) after migration.

1.2 Cloud deployment models

According to NIST (Peter M. Mell 2011) and several critical reviews (Rountree & Castrillo 2014, Goyal 2014) (2014), cloud deployments can be classified into four models:

- **Public cloud** is used by the general public and exists solely on the premises of the cloud provider. Usage of the public cloud allows scaling an organization infrastructure at any capacity. However, several privacy and performance issues can arise while relying solely on this model.

- **Private cloud** are built and maintained directly by an organization owning it. Usage of private cloud leads to limitations in scalability and increases responsibility level. In many cases, Private Cloud is considered to be a part of on-premise Infrastructure (Armbrust, A. Fox, and R. Griffith 2009)

- **Community cloud** can be described as infrastructure built for a specific community. This model simplifies maintenance for each member of the community but can lead to ownership and privacy issues.

- **Hybrid Cloud** is a composition of two or more distinct cloud infrastructures (in most cases Private and Public). This model is used by organizations as a temporary solution in the process of shifting to the cloud infrastructure. Some services can be kept private.


\(^1\)Hybrid Cloud Architecture: What Is It and Why You Should Care (n.d.)
until all security and performance issues are resolved. Hybrid cloud needs a high level of responsibility and expertise to keep data secure and synced.

![Hybrid Cloud Diagram](image)

Figure 1.1: Hybrid Cloud

### 1.2.1 Service Models

Three main service models can be distinguished in cloud computing:

- **Software as a Service (SaaS)** is a cloud computing offering that provides users with access to a vendor’s cloud-based software. Users do not install applications on their local devices. Applications reside on a remote cloud network accessed through the web or an API.

- **Platform as a Service (PaaS)** is a cloud computing offering that provides users with a cloud environment in which they can develop, manage and deliver applications. Users are provided with infrastructure abstraction like (storage and computing) and can use a suite of prebuilt tools to develop, customize and test their own applications.

- **Infrastructure as a Service (IaaS)** is a cloud-computing offering in which a vendor provides users access to computing resources such as servers, storage and networking. Organizations create and maintain platforms and applications within a service provider’s infrastructure.

![Cloud Service Models Diagram](image)

Figure 1.2: Cloud service models

This work excludes software as a service (SaaS) service model since it is more end-customer oriented and focuses mostly on infrastructure as a Service (IaaS) and Platform as a Service (PaaS) due to its higher demand among developer teams (Kepes 2013).

---

*IaaS PaaS SaaS Cloud Service Models — IBM Cloud (n.d.)*
IaaS provides resources similar to physical hardware. Users of the service are able to control entire software stack from OS and upwards. This service model makes scalability, and failover implementation difficult for the cloud provider since the majority of issues are application-dependent. PaaS provides the platform for building an application without the need for building and maintaining the infrastructure. This model is mainly oriented on web applications. (Armbrust, A. Fox, and R. Griffith 2009) Cloud computing introduced new application opportunities and increased the amount of analytics by adding large scalable data management services (Armbrust, A. Fox, and R. Griffith 2009).

1.3 Aim of this work

Based on previous studies in the cloud area, research has been conducted at a general level, where several aspects of the cloud environment have been researched, including highlighting advantages and disadvantages. Cloud migration can have both positive and negative impact on an organization, such as security risks and changed routines, processes and tools. Eliminating early-stage risks assist the organization in achieving a successful migration with minimal risks. Since the cloud migration area is a broad topic, this work focuses on Hybrid Cloud at King and developers infrastructure around it. It aims to investigate challenges, which developers may face while migrating applications to the cloud.

1.4 King

This work is conducted in collaboration with King. Therefore, a brief description of the company will be provided in this section.

King, also known as King Digital Entertainment is a video game developer with studios in Stockholm, London, Berlin, Malmö and Barcelona and offices in Malta, San Francisco and New York, that specializes in the creation of social games. King gained fame after releasing the cross-platform title Candy Crush Saga in 2012, considered one of the most financially successful games utilizing the freemium model. King was acquired by Activision Blizzard in February 2016 and operated as its own entity within that company. King has around 2000 employees and 273 million monthly active users, as of Q1 2020\(^3\).

King has begun its cloud history with migrating its data warehousing to Google Cloud Platform, which had a positive impact on the business.

The next stage, which is discussed in this work, is the creation of Hybrid Cloud platform and migration of game supporting applications to it.

Hybrid Cloud project at King involves several teams, collaborating together in order to create and maintain cloud infrastructure, which can be used by developers to create and migrate applications to the cloud.

\(^3\)Quatery results — Activision Blizzard, Inc. (n.d.)
• **Unified Platform Foundations.** Team is the owner of UP (Unified Platform), a platform for applications, using game data. Developers of these applications are the main future customers of this project.

• **Infrastructure Engineering.** Historically, the team is responsible for all the internal infrastructure and hardware at King. During this project, the team has started an investigation of Cloud technologies and had to shift focus with getting extra headcount to provide support for both on-premise and cloud.

• **Cloud Foundations.** team is responsible for cloud research and experiments, providing guidance for other teams and evangelizing migration to the cloud internally at King.

A virtual team, based mainly on members of three groups mentioned above, was created inside the organization to share the knowledge and experience while building the infrastructure required for Hybrid Platform.

### 1.5 Research Plan

#### 1.5.1 Research focus

To identify and characterize approaches to cloud migration both in technical and social aspects, research is limited to a general question, focused on cloud migration. However, the main focus of this work is based on the case at King. Therefore a major part of the research is based on challenges revealed during data collection, described in Chapter 4.

#### 1.5.2 Questions

- What are the challenges faced by software engineering teams during hybrid cloud migration?

- What are the ways to overcome these challenges?

#### 1.5.3 Topic Justification

Cloud computing is a large field in the modern IT business. It allows to optimize costs and improve organization flexibility. However, many organizations are still sceptical about migration to the cloud due to several reasons. Studies within this research show, why organizations adopt cloud computing and why they don’t. There are also patterns of cloud usage, specific to enterprises, mentioned in the research.

Several case studies show that the system infrastructure cost ±35% less over five years on one of the major cloud platforms, and using cloud computing could have potentially eliminated ±20% of the support calls for a system. (Khajeh-Hosseini et al. 2010) Emergence of cloud computing in the last decade made many companies of different sizes consider cloud as a target.
platform for migration. Many applications could not benefit from the cloud environment as long as migration strategy was to dump the existing legacy architecture to a virtualized environment and call it a cloud application. Cloud computing promises to reduce the cost of IT organizations by allowing organizations to purchase just as much compute and storage resources as needed, only when needed. (Hajjat et al. 2010)

The advantages and initial success stories of cloud computing are prompting many enterprise networks to explore how the cloud could be used to run their existing systems and applications. Considering a recent survey (Gholami et al. 2016a), over 36% of respondents indicated that a large number of applications and the complexity of managing data centres were huge problems that they faced. Over 82% of respondents indicated that reducing data centre costs was one of the most important objectives for the coming years. Over 72% of respondents indicated they were considering or using public cloud computing, although 94% of these respondents were still in the discussion, planning, trial or implementation stages. Migrating enterprise applications to cloud computing is a major challenge, despite the significant interest. (The Case Against Cloud Computing, Part One — CIO n.d.).

The result of this work should allow organizations to discover existing drawbacks during the cloud migration process and define a clear migration strategy in advance. This will lead to a decreased level of confusion among developers and clearer team roles and goals during the implementation of the project.

1.6 Delimitations

This study intends to review the case at King and clarify the modern challenges and benefits of cloud adoption in organizations and evaluate the process of cloud migration. The thesis is restricted to the analysis of cloud migration challenges and approaches to solve them. After the initial step of the research, it was narrowed down to a specific topic, which excluded several research questions from the original problem statement. This work focuses on Cloud challenges applicable to King’s business and Unified Platform project. Therefore it is narrowed down to some specific problems and cultural challenges, which, however, are mentioned in several other publications (Jamshidi et al. 2013, Tilkov 2015, Ali et al. 2015). The case study intends to provide a high-level overview of cloud migration in a large enterprise. This work focuses on a specific cloud provider - Google Cloud Platform; other providers were ignored due to irrelevance to this project.

1.7 Thesis structure

This work is structured into three main parts. In the first part, the main concepts and theoretical aspects are described in order to give the reader the understanding of the focus of this project and provide the required theory behind it. The existing state of the art is also reviewed in that part. The second part will describe the PoC concept, which was the result of
acquired knowledge and collected data. The third part will describe user-related issues during the system design and implementation process and mention the topics for future discussion.

This work is structured as follows:

1. **Introduction.** Consists of background a problem statement.

2. **Literature review.** Describes the main concepts in detail followed by relevant topics for this work work

3. **Research methodology.** Describes the approach to research and data collection.

4. **Empirical findings.** Describes research findings during data collection stage.

5. **Analysis.** Describes analysis, based on research findings and literature overview/

6. **Discussion.** Future challenges, opportunities and outcomes of the process of cloud migration

7. **Conclusion.** Summary of work.

Figure 1.3: Thesis structure
Chapter 2

Literature review

This chapter reviews the main theoretical concepts relevant to this work—theory regarding cloud computing, cloud architecture, cloud computing deployment models and cloud management. The chapter will end with the theory of cloud migration and its challenges. The literature reviewed in this chapter helps identify patterns in the cloud migration process and provides background for identification of challenges and benefits in the process of cloud migration, described in Chapters 4 and 5.

2.1 Cloud computing

2.1.1 Essential characteristics

According to Peter M. Mell (2011), five essential characteristics are specific for cloud computing:

On-demand self-service describes the possibility of automatically provisioning without requiring human interaction with a service provider.

Broad network access describes the feature of cloud applications and services, which makes them accessible from any location in the world, using any device.

Resource pooling describes how multi-tenant model can be used to dynamically assign and reassign resources according to consumer demand. Customer has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction.

Rapid elasticity allows applications and services to be elastically provisioned and released, both manually and automatically to match resources with business requirements or user demand.

Measured service resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer.

2.1.2 Initial state of utility computing

Utility computing is a term, which preceded cloud computing but introduced the same concept of rented infrastructure. It was introduced in 2000 by Intel but did not achieve success due
to requirements for a long-term contract (Armbrust, A. Fox, and R. Griffith 2009). Amazon introduced Amazon Web Services (AWS) with the ability to pay on a per-hour basis, which made it a first significant successful example in the cloud industry.

According to Yang (2012), there are good reasons for the growing popularity of cloud computing. Cost is a significant consideration in moving towards cloud computing, especially for business and industries that always look out for the reduction of operating expenses. Biggest challenges for cloud computing are data security, internet bandwidth, and the control of the IT infrastructure that customers are often reluctant to give up.

Cloud computing has been in high demand among many enterprises for around a decade (Peter M. Mell 2011). It was predicted to grow and have a significant impact on the entire IT industry over a decade ago. (Armbrust, A. Fox, and R. Griffith 2009). Cloud Computing provides an approach to computing as a utility.

2.1.3 Current challenges in cloud computing

According to more recent publication on the topic (Varghese & Buyya 2018), several main challenges can be identified in modern cloud computing. Data security keeps being the primary challenge, according to the majority of published works. Cloud professionals are more concerned with cloud security than other IT staff. The Crowd Research Partners survey\(^1\) found that 90% of security professionals are worried about cloud security: data loss and leakage (67%), data privacy (61%), and breach of confidentiality (53%). Lack of expertise and knowledge keeps being a major challenge since enterprises are still discovering the opportunities of cloud usage and are in the process of hiring employees with cloud skills and facilitate cloud training and certification for existing staff. Cloud management and governance are the main infrastructure problems that enterprises face. Different approach to architecture, when an enterprise uses a ‘pay-per-use’ concept and does not own hardware introduces problems in governance, compliance and cost management. According to Rightscale\(^2\) 81% of enterprises adopt a multi-cloud strategy, and 51% have a hybrid cloud strategy (public and private clouds combined). On average, companies use 4.8 different public and private clouds, which leads to a challenge of maintenance of the heterogeneous environment and keeping it reliable and consistent.

2.1.4 Hybrid Cloud

One of the types of multi-cloud architectures is a hybrid cloud, a combination of public and private clouds or a combination of public and private IT infrastructure (Zhang et al. 2011, Bernstein et al. 2009). The primary motivation for such an approach is having demands for bursty scaling. The benefit of using hybrid clouds for handling sensitive data was described by Xu et. al (Xu & Zhao 2015). It is estimated that 63% of organizations using the cloud have

\(^{1}\)Cloud Security Report - Crowd Research Partners (n.d.)

\(^{2}\)2020 State of the Cloud Survey from Flexera (n.d.)
adopted a hybrid cloud approach with use-cases reported in the healthcare and energy sectors. The key challenge in setting up a hybrid cloud is networking issues as private, and public cloud needs to be connected directly and ensure the same privacy and security level. Bandwidth, latency and network topologies need to be considered for accessing a public cloud from a private cloud (Breitenbücher et al. 2012). Network limitations can result in an ineffective hybrid cloud. Dedicated direct networking between clouds may enable more effective infrastructure. However, it requires additional management of private resources, which can be a cumbersome task, considering existing maintenance of on-premise resources According to research from 2014 (Hsu et al. 2014) on enterprises using the cloud, cloud adoption is still at its initial stage, since the adoption rates are meagre. The perceived benefits, business concerns, and IT capability within the TOE framework are significant determinants of cloud computing adoption, while external pressure is not. Enterprises with extensive IT capability tend to choose the pay-as-you-go pricing mechanism. The most important factor influencing the choice of deployment model is a business concern, with higher concerns leading to private deployment options.

2.2 Cloud deployment

2.2.1 Environments

In order to fix or alter the behaviour of an application, developers make changes to its code. New software releases are deployed to each environment to facilitate phased release management, where at each phase software is rolled out, tested, and rolled back if needed.

Oracle distinguishes four development environments for the software development process:

- **Development environment.** Developers obtain a copy of the application code and make changes to it. They will also trial the changes they make by running the application locally, committing changes to be pushed on to the next environment

- **A Test environment** Combines completed work from all the project’s developers. Before promoting the code to the next stage, builds must originate in a single, consistent environment, so that the outputs are also consistent, and reproducible. Code is tested and evaluated

- **A QA environment** is used to test the application’s behaviour. This part can include acceptance testing by customers or stakeholders

- **A Production environment** is where an application is available for users.

It is essential to ensure as far as possible that all environments are as similar to one another as possible. Otherwise testing software in any environment other than the production environment is futile since there is no guarantee that the application will behave the same once in the production environment as it does where it is being tested.

---

3 Technology-organization-environment framework - EduTech Wiki (n.d.)
4 Development, Test, QA, and Production Environments (Oracle Waveset 8.1.1 Upgrade) (n.d.)
2.2.2 Pipelines

Waterfall methodology could take months or even years to deliver a product’s first version. Switching to Agile methods helped reduce programming cycles to weeks and introduced interval delivery. (Humble & Farley 2010) Today’s practice of continuous integration (CI) rolls out program updates even faster, within days or hours. That is the result of the frequent submission of code into a shared repository so that developers can easily track defects using automated tests, and then fix them as soon as possible.

2.2.3 Continuous Integration

The main aims of Continuous Integration are: to make the deployment process visible and clear to everyone involved, improve the feedback and allow automated deployment and releases to different environments. The first step of the pipeline is usually to create a binary or any other sort of executable, which should be triggered by any modification applied to source code. Successful build triggers the execution of tests, which usually run consequently. Each passed test indicates that the build is closer to a successful deployment candidate. Once all the tests pass, the release candidate can be released.

According to Humble & Farley (2010) three main rules for reliable software delivery can be identified:

- Deploying software in an automated way
- Always have a production environment ready during the development process in order to be able to test the current version of the software
- Having a production environment set in an automated way, so there are no changes on specific nodes of the entire software system.

All these ideas led to Infrastructure as Code approach, which became a standard for well-organized teams and enterprises, which in-fact led to modern GitOps approach, which will be covered later in this work. Three main practices Every change should trigger the feedback process Feedback must be received as soon as possible Everyone should be informed and be able to take actions Following these practices leads to a reduction of errors, lowering stress and confusion among team members and simplification of deployment for developers, which allows them to focus on the product itself.

2.2.4 Continuous Delivery

The workflow of the continuous delivery according to Humble & Farley (2010) consists of several important stages.

- Create a repeatable, reliable process for releasing
- Automate as much as possible
• Keep everything in version control (key to GitOps approach)
• Don’t postpone the testing and delivery, start early
• Test and fix immediately
• Define done as deployed
• Everyone is responsible
• Improve the process continuously.

Similarly to Agile Manifesto, only following all of the mentioned practices leads to a consistent process of delivering software regularly. Version control introduces the idea of not questioning the deletion since it rules out the risk of losing the code completely, as it is kept in the repository. Configuration should be treated the same way as code.

2.2.5 Namespaces

Namespaces are intended for use in environments with many users spread across multiple teams, or projects.

Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces. Namespaces can not be nested inside one another, and each resource can only be in one namespace.

Namespaces are a way to divide cluster resources between multiple users (via resource quota).\(^5\)

2.3 Cloud management

2.3.1 Infrastructure

Software delivery consists of a significant amount of work, which needs to be done in order to make code available to a customer. (Humble & Farley 2010)

The traditional approach to software development in organizations includes developers (“Dev”), who are focused on creating software, and operations (“Ops”), whose focus is software management. (Brikman 2019a)

Manual approach to software and hardware management, which is the central part of Operation teams work, requires many resources, once business grows. In order to keep systems reliable and keep live support available, software teams can come up with reducing release cadence, which goes in contradiction with the continuous delivery approach.

According to the state of DevOps report\(^6\) organizations, that use DevOps practices deploy 200 times more frequently, recover from failures 24 times faster, and have lead times that are 2555 times lower.

\(^5\) Production-Grade Container Orchestration - Kubernetes (n.d.b)
\(^6\) 2016 State of DevOps Report — Puppet.com (n.d.)
Appearance of cloud computing and infrastructure automation tools like Chef\textsuperscript{7}, Puppet\textsuperscript{8}, Ansible\textsuperscript{9}, SaltStack\textsuperscript{10}, Terraform\textsuperscript{11}simplified hardware management and configuration processes, which led to both Dev and Ops teams focusing on writing software. This lead to the appearance of DevOps paradigm to software development. According to Brikman (Brikman 2019\textsuperscript{b}), the goal of DevOps is to make software delivery vastly more efficient.

2.3.2 Imperative vs. Declarative approaches

An automation framework can be designed and implemented in two different ways: declarative vs imperative. These are called DevOps paradigms. While using an imperative paradigm, the user is responsible for defining exact steps which are necessary to achieve the end goal, such as instructions for software installation, configuration, database creation, etc. Those steps are later executed in a fully automated way. The ultimate state of the environment is a result of particular operations defined by the user. While keeping full control over the automation framework, users have to carefully plan every step and the sequence in which they are executed. Although suitable for small deployments, imperative DevOps does not scale and fails while deploying big software environments, such as OpenStack.

In turn, a declarative paradigm takes a different approach. Instead of defining exact steps to be executed, the ultimate state is defined. The user declares how many machines will be deployed, will workloads be virtualized or containerized, which applications will be deployed, how will they be configured, etc. However, the user does not define the steps to achieve it. Instead, a ‘magic’ code is executed, which takes care of all necessary operations to achieve the desired end state. By choosing a declarative paradigm, users not only save a lot of time usually spent on defining the exact steps but also benefit from the abstraction layer being introduced. Instead of focusing on the 'how', they can focus on the 'what'.

The Topology and Orchestration Specification for Cloud Applications (TOSCA)\textsuperscript{12} supports two different approaches to provisioning: declarative, which is based on defining goal state, and imperative, which uses provisioning plans that explicitly describe the tasks to be executed.

**Imperative** The main drawback of creating plans manually is the nature of time-consuming, costly, and error-prone task: complex management services need to be orchestrated, and data formats must be handled, as well as many other challenges. Also, plans are tightly coupled to a particular application topology and sensitive to structural changes: different combinations of components lead to different plans. Thus, plans for new applications often have to be created from scratch. (Breitenbücher et al. 2013)

**Declarative** The declarative approach is rather suited for simple applications that consist of shared components, relations, and technologies, due to the inability to define or infer

---

\textsuperscript{7} Chef: Enabling the Coded Enterprise through Infrastructure, Security and Application Automation (n.d.)
\textsuperscript{8} Powerful infrastructure automation and delivery — Puppet — Puppet.com (n.d.)
\textsuperscript{9} Ansible is Simple IT Automation (n.d.)
\textsuperscript{10} Home - SaltStack (n.d.)
\textsuperscript{11} Terraform by HashiCorp (n.d.)
\textsuperscript{12} Topology and Orchestration Specification for Cloud Applications Version 1.0 (n.d.)
provisioning logic.

The result of the discussion above is a combination of both flavours would enable application developers to benefit from automatically provided provisioning logic based on declarative processing and individual customization opportunities provided by adapting imperative plans.

**Declarative approach to infrastructure**

Approach to the automation of deployment and provisioning in enterprises started at the beginning of the century with appearing of Puppet in 2005. Puppet was used for automated provisioning and setup of Virtual Machines (VMs). The idea behind it was to automate the process of connecting existing VMs via network and executing the scripts on machines. Puppet was followed by Chef and Salt, which provided a higher level of abstraction and bigger functionality. However, market competition made these solutions comparable. Ansible appeared a few years later and introduced a completely new approach to infrastructure automation by providing next level of abstraction and ability to deploy most of the tools used by enterprises like DBs, Monitoring solutions and others. Tools mentioned before introduce a great level of automation with an imperative approach, allowing to define a procedure or script, which is a much more simplified approach in comparison to bash. In comparison to that, a declarative tool called Terraform appeared. It introduces state, which defines the services in a concrete moment of time. This, combined with a defined, resulting state, allows to compare and find the difference in provisioning and quick changes/rollbacks, whenever needed. Terraform provides templates in the form of modules, adapted to platforms, which can be used within Terraform.

**Infrastructure as Code**

The main idea behind Infrastructure (IAC) as code is to define, deploy, update and destroy the infrastructure using code. While the traditional approach to infrastructure encourages manual setup of each server by the execution of specific commands using shell or GUI, IAC allows automating the majority of the steps into scripts, which can be executed on every machine with minor customization depending on the logic.

Infrastructure as code can be defined by three steps:

1. Developers write the infrastructure specification in a domain-specific language.
2. The resulting files are sent to a master server, a management API, or a code repository.
3. The platform takes all the necessary steps to create and configure the computer resources.

Five categories of IAC tools can be distinguished according to Brikman (Brikman 2019b):

- **Ad Hoc Scripts** - putting the manual commands, typically executed on each server into a script and running it, whenever a new machine needs to be configured. This approach is not tool-specific and allows to use from a variety of programming languages, but it leads to problems in knowledge sharing and code maintenance issues, once the infrastructure grows or script logic becomes complex.
Configuration management tools. Chef\textsuperscript{13}, Puppet\textsuperscript{14}, Ansible\textsuperscript{15} and SaltStack\textsuperscript{16} are the examples. These tools simplify connection and execution steps for each server and introduce code conventions for automation scripts and enforce a consistent structure and documentation. They encourage code to be idempotent, i.e. correct execution independent from the number of repetitions.

Server templating tools. Such tools introduce an approach to the creation of server snapshots, which represent an image of an already configured operating system (OS) with required software and data. This approach enables the shift to immutable infrastructure, i.e. every deployed server will not be changed after the deployment process. Packer\textsuperscript{17}, Vagrant\textsuperscript{18} and Docker\textsuperscript{19} are the examples of such tools.

Orchestration tools. Introduction of server templating tools lead to the challenges in maintaining them: updating, health monitoring, scaling and load balancing. In order to solve these tasks, orchestration tools like Kubernetes\textsuperscript{20}, Docker Swarm\textsuperscript{21} and Nomad\textsuperscript{22} appeared. Kubernetes YAML files allow to define the deployment in a declarative way with specifying the single-unit (Pod) structure, it’s settings and the number of replicas to run. Kubernetes itself takes the responsibility of maintaining the desired state.

Provisioning tools, in contrast to other approaches, allow creating almost every part of the infrastructure like servers, services, rules and configurations themselves. Examples of such tools are Terraform\textsuperscript{23} and CloudFormation\textsuperscript{24}.

Recently, a large number of tools with similar approach were introduced by each major cloud provider in order to simplify the process of infrastructure management for users. A large list of tools available on the market can be found in XeniaLabs guide\textsuperscript{25}.

Infrastructure as code uses of various software engineering practices to improve software delivery processes like self-service, documentation, version control, validation and reuse.

**GitOps Approach**

GitOps is a declarative approach to infrastructure management and deployment. GitOps upholds the principle that Git is the one and only source of truth. GitOps requires the desired state of the system to be stored in version control such that anyone can view the entire audit

\textsuperscript{13}Chef: Enabling the Coded Enterprise through Infrastructure, Security and Application Automation (n.d.)
\textsuperscript{14}Powerful infrastructure automation and delivery — Puppet — Puppet.com (n.d.)
\textsuperscript{15}Ansible is Simple IT Automation (n.d.)
\textsuperscript{16}Home - SaltStack (n.d.)
\textsuperscript{17}Packer by HashiCorp (n.d.)
\textsuperscript{18}Vagrant by HashiCorp (n.d.)
\textsuperscript{19}Empowering App Development for Developers — Docker (n.d.)
\textsuperscript{20}Production-Grade Container Orchestration - Kubernetes (n.d.)
\textsuperscript{21}Swarm mode overview — Docker Documentation (n.d.)
\textsuperscript{22}Nomad by HashiCorp (n.d.)
\textsuperscript{23}Terraform by HashiCorp (n.d.)
\textsuperscript{24}AWS CloudFormation - Infrastructure as Code & AWS Resource Provisioning (n.d.)
\textsuperscript{25}The Ultimate List of Provisioning and Configuration Management Tools - XebiaLabs (n.d.)
trail of changes. All changes to the desired state are fully traceable commits associated with committer information, commit IDs and time stamps. This means that both the application and the infrastructure are now versioned artefacts and can be audited using the gold standards of software development and delivery. However, while setting up and managing Kubernetes clusters can be fun for folks who like to tinker with infrastructure, some application developers and testers do not want to get bogged down with logistical and administrative fire drills. Even folks who feel comfortable managing Kubernetes on their own admission that it inflates their total cost of ownership (TCO). (What is GitOps? | CloudBees 2020)

(Limoncelli 2018) was the first publication to describe the GitOps approach to Software development. GitOps lowers the cost of creating self-service IT systems and makes them faster and more convenient to its users. The author defines GitOps as Infrastructure as Code combined with a Pull Request approach to modifications. To be specific, the entire Pull request and merge operation should be automated as much as possible by introducing automated tests and build. Human intervention should include only manual checks, which require non-trivial automation solution.

Evolution of GitOps

1. Basic – configs in the repository as a storage or backup mechanism.
2. IaC – PRs from within the team trigger only CI-based deployments.
3. GitOps – PRs from outside the team, pre-vetted PRs, post-merge testing.

1. Find the Git repo that stores a logical description of the plumbing that connects the load balancer to various web application servers.
2. Edit that file to add your new application.
3. The proposed revision is submitted to the web team as a PR (pull request) the same way developers submit PRs for software projects.
4. At the same time that humans are reviewing the PR, your CI (continuous integration) system (i.e., Jenkins or similar) is linting and unit testing your changes to the load balancer config (possibly in a container or VM).
5. Once the PR is approved and "the builds are green," the CD (continuous deployment) pipeline (often another Jenkins job or similar) will take care of generating the new config file for the production load balancer and deploying it, usually with the help of a config management system such as Puppet or Chef.

This kind of workflow is known as GitOps: empowering users to do their own IT operations via PRs.

What’s new is enabling people outside the IT team to submit PRs, the extensive use of automated testing, and using a CI system to integrate all of this.
2.3.3 User benefits of GitOps

1. An operating model for Kubernetes and other cloud-native technologies, providing a set of best practices that unify deployment, management and monitoring for containerized clusters and applications.

2. A path towards a developer experience for managing applications; where end-to-end CI CD pipelines and git workflows are applied to both operations, and development.

2.3.4 Version Control as a source of truth for all processes

According to Continuous Delivery Book, Version Control should be used as a source of truth for all the project’s code, including infrastructure. This approach leads to convenient access of all version’s and states of the system and code. This approach leads to a Kubernetes concept, introduced by Weave works called GitOps.

2.3.5 Kubernetes

This section focuses on the main design principles of Kubernetes and the main architectural ideas behind it. The main motivation of including this into the work is to describe, which ideas Kubernetes implements and how in order to follow the. Ideas while developing a hybrid architecture and migration process.

    Cloud resource orchestration involves the creation, management, manipulation and decommissioning of cloud resources, i.e., compute, storage and network, in order to realize customer requests, while conforming to operational objectives of the cloud service providers at the same time. (Liu et al. 2011)

1. Kubernetes is rather declarative than imperative. Level trigger instead of edge trigger. Each node and each component tries to talk to master and figure out what it is supposed to do. Leads to no missed events scheme, when a pod recovers and gets the information from master on what it is supposed to do. If the master goes down, all the components keep working according to the last desired state they have received, until the master recovers and updates the state.

   - Self-healing
   - Rollback
   - Extensible
   - Immutable.

2. Control plane is transparent, no hidden API. This makes the modification and configuration easy: to create a scheduler, a developer needs to create an application which will talk to Kubernetes API and retrieve unscheduled pods and update the state
Ease of adoption

3. Meet the user where they are. No application modifications are needed to migrate to Kubernetes, like exposing secrets as environment variables or files.

4. Workload portability. An abstraction of storage to persistent volume and persistent volume claim to keep the pods independent from storage implementation.

- Cloud/cluster agnostic

One of the containers’ noticeable features is that they can be managed specifically for application clustering, especially when used in a PaaS environment. Answering this need, at the June 2014 Google Developer Forum, Google announced Kubernetes, an open-source cluster manager for Docker containers. According to Google, “Kubernetes” is the decoupling of application containers from the details of the systems on which they run. Google Cloud Platform provides a homogeneous set of raw resources to Kubernetes, and in turn, Kubernetes schedules containers to use those resources. This decoupling simplifies application development since users only ask for abstract resources like cores and memory, and it also simplifies data centre operations. (Bernstein 2014)

2.4 Cloud architecture

2.4.1 Cluster environments

Multi-cluster environment

General Kubernetes architecture consists of several nodes (workloads), control plane(-s), which is usually called a Master. These nodes, which are virtual machines, run on top of shared compute and storage network. Control plane itself consists of API Server, Controller Manager, Scheduler, DNS, which is used for service discovery of all of the workloads. Worker nodes include service tools like kubelet and Kube-proxy, which is involved in networking. In fact, not only resources, which were implicitly mentioned as shared, are shared usually. It is all the Kubernetes components, which can be used by several projects. Nodes can be shared among several tenants of the cluster, i.e. several tenants can have pods on each of the nodes. The default approach of separation of these tenants is Kubernetes namespaces, which run in isolation. However, namespaces can not isolate node service tools, examples of which were mentioned before. Control plane does not get separated by any mechanisms, while all of its main parts are also being shared by all tenants. This cluster design is usually called soft multitenancy. The reason why this approach has become a general practice is that DNS, for example, can not be considered as ‘tenant-aware’ system. The implications of this are:

- each of the tenants can see, which services were published into DNS by others

- Ops affects all tenants because any sort of manipulation, i.e. upgrade of a node pool will lead to downtime of each project using that node pool
All tenants of the cluster are compelled to use the same version of Kubernetes cluster. Another approach, which can be called hard multitenancy to cluster architecture is to use a multi-cluster method, when both control plane and workers are individual to a tenant (project). At the same time, only the only hardware keeps being shared. This approach allows to isolate all software services from each other and make architecture more secure. Another implication of this is that all maintenance actions will affect only one tenant of the cluster. Clusters themselves can be configured differently and use different versions, so API server or controller manager can be configured individually for each project. This allows for more opportunities for tenants.

2.4.2 Multi-tenancy

There are several common patterns, each of which has pros/cons for setting up multiple environments for application teams. Probably the most common is a shared multi-tenant cluster where each of your app teams has access to namespaces which represent their development lifecycle. Some like to have traditional dev, test, QA, prod separation, whereas others want to split based on teams and apps/app groups. Rancher provides the ‘project’ construct which is pretty useful, although again is a divergence from vanilla K8s. Either way (and many others) namespaces can provide a management, resource and security boundary which is typically what you need for multi-tenant. Alternatively, you can go for multiple clusters. These provide more robust isolation, are in some ways easier to manage and secure but can increase your costs since there is less resource sharing and higher redundancy overall.

2.5 Migration

Cloud migration has several significant benefits for enterprises. The main ones are reducing capital expenditure and transforming it into operational costs (Armbrust, A. Fox, and R. Griffith 2009), which starts with reducing spending on buying infrastructure. Three main motivations behind the usage of Cloud computing (Armbrust, A. Fox, and R. Griffith 2009):

- Ability to scale up and down rapidly both for entire software systems and single applications
- Elimination of up-front commitment, which allows enterprises to start small
- Ability to pay on a ‘pay-per-use’ basis.

2.5.1 Cloud Migration

Two big surveys (Pahl et al. 2013) (Gholami et al. 2016b), describing cloud migration concludes that migration to the Cloud raises a range of questions currently. Standard procedures do not exist, and tool support is often not available. Migration experts rely on their own experience
and some essential tools to facilitate the process. The main source of motivation for another enterprises (Balalaie et al. 2016b) and teams for cloud migration was:

- Reusability
- Decentralization
- Automated deployment
- In-built scalability.

A common desired approach to this problem was to perform it incrementally without affecting the end-users. In terms of tools, a pattern of choosing Docker is visible; however, the choice of Docker registry varies. Similarly, how Cloud computing addresses both applications delivered as services over the internet and software systems that provide those services (Armbrust, A. Fox, and R. Griffith 2009), Cloud migration can be divided into two central topics: application migration and full enterprise infrastructure migration (How to choose a cloud migration strategy - Work Life by Atlassian n.d.).

2.5.2 Application migration

Several main issues preventing applications from migration can be identified (Andrikopoulos et al. 2013):

- business-critical value of the application, which requires it to be live and secure
- application purpose or architecture, e.g. embedded applications

Typical application can be split into three parts: presentation, business logic and data. Among large applications, not entire application needs to be migrated; in many cases, part of the application will provide expected benefits while allowing to keep the benefits of on-premise deployment for other parts. (Andrikopoulos et al. 2013) Four types of migration can be highlighted:

- Replace
- Partially migrate
- Migrate the whole network stack
- Cloudify the application.

Each of the mentioned methods introduces a replacement step for a component or a part of a component in an application. The decision on the migration approach should be based on business logic and value of the application.

Security and confidentiality concerns with respect to data migration, e.g. of application data are one of the main issues impeding the further adoption of Cloud computing in industry and research.
Isolated state, elasticity and automated management are mentioned among the main benefits of cloud computing for applications were described by Fehling et al. (2013).

**Isolated state**

A cloud application should handle the session, interaction and application states. Data handled by the application in as few application components as possible. Most cloud providers suggest handling state in communication offerings, i.e., messages or provider-supplied storage, making application components stateless.

**Elasticity**

A cloud application has to support that Cloud resources may be provisioned and decommissioned flexibly. The isolation of state is closely related to this property as the addition and removal of resources is significantly simplified if no state information has to be extracted or synchronized.

**Automated Management**

Manual changes to resource numbers are commonly not reactive enough to effectively benefit from usage-based billing supported by clouds. Also, cloud providers often do not assure availability for individual resources 4 suggesting automated failure handling.

### 2.5.3 SOA versus MSOA in Cloud Migration

Migration process starts from the estimation of current infrastructure and services in order to define a cloud migration strategy. Two most popular architectural style is considered: Service-Oriented Architecture (SOA) and Microservice-oriented architecture (MSA) (Microservices Architecture – SOA and MSA n.d.). The main difference between two styles is that SOA a service may be composed of other services, while in MSA we define a service as independent and self-contained, which implies that it cannot be composed of other services. MSA appeared as a successor of SOA to simplify the architecture of software systems and make it follow the single responsibility principle (Clean Coder Blog n.d.), microservices adopt SOA concepts that have been used during the last decade, but it is an architecture style more focused in achieving agility (Villamizar et al. 2015) MSA has proven to be the most efficient way of developing applications in the Cloud efficiently, to reduce the complexity, to grow development teams easily, and to achieve agility. However, there are still many challenges that must be kept into account when new companies want to use this pattern in their applications. Microservice implementations also require the use of DevOps (Development + Operation) strategies, where teams that develop applications also deploy, operate and monitor them in the Cloud. One of the benefits of using microservices because each microservice/gateway can scale independently using different policies. At a business level, this may represent a significant saving in IT infrastructure costs.
and a more efficient way to take advantage of the pay per use and on-demand benefits of the cloud model. (Villamizar et al. 2015)

"Microservices and SOA solve different problems. Microservices must be independently deployable, whereas SOA services are often implemented in deployment monoliths." - Eberhard Wolff (Microservices and SOA — Oracle Magazine n.d.)

2.5.4 Full enterprise migration

The success of cloud migration comes from two main advantages of Cloud technologies (Andrikopoulos et al. 2013):

- minimum modifications to existing applications, i.e. application can be simply migrated to VM in Cloud

- Ability to reduce cost while improving performance on average.

Cloud plans usually consist of pay-per-resource offerings, which allow paying for each resource in the project separately. Pay-as-you-go approach can charge the application separately for each type of utilized resource, which allows reducing underutilization (Armbrust, A. Fox, and R. Griffith 2009). Support and all the operational costs are shifted to a cloud provider’s responsibility. NIST (Armbrust, A. Fox, and R. Griffith 2009) lists ten main obstacles for cloud migration, which are presented in table 2.1

Note that these issues were mentioned back in 2009 when this report was published. Several of these issues are not so relevant nowadays since they were approached and improved by cloud providers (e.g. options to use licensed software, smarter scaling mechanisms)

Virtual Machine migration

The term virtual machine (VM) initially described a software abstraction with the looks of a computer system’s hardware. Nowadays, the term includes a broad range of concepts. For example, Java virtual machines that don’t match an existing real machine. This work focuses only on system-level virtualization, which is close to the original definition of VM. Virtualization layer sits between the operating system and the application programs that run on the operating system. The virtual machine runs applications that are written for the particular operating system being virtualized. Among the main advantages of virtual machines, Software compatibility, isolation, encapsulation and performance should be mentioned.26

Virtualization technologies have enabled new mechanisms for providing resources to users. In particular, these efforts have influenced hardware design to support transparent operating system hosting. Clark et al. (2005)

26The Reincarnation of Virtual Machines - ACM Queue (n.d.)
<table>
<thead>
<tr>
<th></th>
<th>Availability of Service</th>
<th>While users expect the same level of service availability as Google search for most of the services, Cloud providers struggles to provide that level of service and support for many resources. However, in most cases, the availability of VMs in the Cloud can be compared to average private cloud machine availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Data Lock-In</td>
<td>While cloud providers provide an API to access all the resources, it is still easy to access data and programs on-premise</td>
</tr>
<tr>
<td>3</td>
<td>Data Confidentiality and Auditability</td>
<td>Enterprises still consider Cloud as a more insecure environment than on-premise infrastructure</td>
</tr>
<tr>
<td>4</td>
<td>Data Transfer Bottlenecks</td>
<td>Connection between on-premise services and services hosted in the Cloud sometimes does not match service requirements</td>
</tr>
<tr>
<td>5</td>
<td>performance Unpredictability</td>
<td>In several use cases performance of VMs and its I/O operations speed is hard to predict</td>
</tr>
<tr>
<td>6</td>
<td>Scalable Storage</td>
<td>Scalable Store with high availability and low cost is hardly achievable</td>
</tr>
<tr>
<td>7</td>
<td>Bugs in Large-Scale Distributed Systems</td>
<td>It is harder to debug and troubleshoot applications and systems running in cloud environment</td>
</tr>
<tr>
<td>8</td>
<td>Scaling</td>
<td>It is challenging to scale up and down rapidly without performance loss on the user side</td>
</tr>
<tr>
<td>9</td>
<td>Reputation Fate Sharing</td>
<td>Customer actions can lead to IP addresses of a cloud provider can be blocked, which will affect other customers</td>
</tr>
<tr>
<td>10</td>
<td>Software Licensing</td>
<td>Some of the software licences are not adapted to use in the cloud</td>
</tr>
</tbody>
</table>

Table 2.1: Obstacles for cloud migration
2.5.5 General Migration issues

Many applications, of course, are not ready to be moved to the Cloud because the environment is not mature enough for this type of applications, e.g. safety-critical software. For others, it may not make sense to be migrated at all, e.g. embedded systems. Some software will be implemented for the Cloud (Cloud-native applications), but other systems must be adapted to be suitable for the Cloud, making them Cloud enabled. The latter category is the focus of this work.

2.5.6 Cloud Providers

Selection of Cloud provider is not a trivial task since it requires a lot of estimation and some high-risk assumptions. The process of selecting the provider, once the requirements are set, can be automated (Zhang et al. 2012).

2.5.7 Cloud Agnostic approach

How to keep the balance between cloud-agnostic and cloud-native approach? How to ensure platform migration simplicity?

Based on the performance tests executed for a monolithic architecture, a microservice architecture operated by the cloud customer, and a microservice architecture operated by AWS Lambda, we can conclude that the use of emerging cloud services such as AWS Lambda, exclusively designed to deploy microservices at a more granular level (per HTTP request/function), allows companies to reduce their infrastructure costs in up to 77.08%. Microservices also enable large applications to be developed as a set of small applications that can be independently implemented and operated, thus managing large codebases by using a more practical methodology, where incremental improvements are executed by small teams on independent codebases.

Agility, cost reduction, and granular scalability must be balanced with the development efforts, technical challenges, and costs incurred by companies resulting from microservices requiring the adoption of new practices, processes, and methodologies. Furthermore, the case study allows us to conclude that for applications with a small number of users (hundreds or thousands of users), the monolithic approach may be a more practical and faster way to start. In the reviewed practical cases, most applications using microservice architectures started as monolithic applications and were incrementally modified to implement microservices due to scaling problems at infrastructure and team management levels. (Villamizar et al. 2017)

Based on the performance tests executed for a monolithic architecture, a microservice architecture operated by the cloud customer, and microservice architecture, we can conclude that the use of emerging cloud service, exclusively designed to deploy microservices at a more granular level (per HTTP request/function) allows companies to reduce their infrastructure costs in up to 77.08%. Microservices also enable large applications to be developed as a set of small applications that can be independently implemented and operated, thus managing large codebases by using a more practical methodology, where incremental improvements are executed
by small teams on independent codebases. Agility, cost reduction, and granular scalability must be balanced with the development efforts, technical challenges, and costs incurred by companies resulting from microservices requiring the adoption of new practices, processes, and methodologies. Furthermore, the case study allows us to conclude that for applications with a small number of users (hundreds or thousands of users), the monolithic approach may be a more practical and faster way to start. In the reviewed practical cases, most applications using microservice architectures started as monolithic applications and were incrementally modified to implement microservices due to scaling (Villamizar et al. 2017).

It has been noted (Armbrust, A. Fox, and R. Griffith 2009) that Cloud providers can offer a more practical approach to infrastructure costs due to changes in prices of hardware and ownership of the risks they take. However, each enterprise is suggested to examine the expected average and peak utilization in order to estimate the cost of Cloud usage.

Second, these companies may already have the datacenter, networking, and software infrastructure in place for their mainline businesses, so Cloud Computing represents the opportunity for more income at little extra cost.

### 2.5.8 Cloud cost

It has been argued that the cost is outweighed by the significant Cloud Computing economic benefits of elasticity and transference of risk, especially the risks of overprovisioning (underutilization) and underprovisioning (saturation).

Although the commercial appeal of Cloud Computing is often described as "converting capital expenses to operating expenses" (CapEx to OpEx), it is believed the phrase "pay as you go" (usage-based pricing) more directly captures the economic benefit to the buyer. Hours purchased via Cloud Computing can be distributed non-uniformly in time. Also, the absence of up-front capital expense allows capital to be redirected to core business investment. (Armbrust, A. Fox, and R. Griffith 2009)

Even if peak load can be correctly anticipated, without elasticity, resources are wasted during non-peak times. Underprovisioning can occur, when potential revenue from users not served (shaded area) is sacrificed or some users quit the service permanently due to poor experience; this result in a permanent loss of a portion of the revenue stream.

### 2.6 Summary

This chapter identified main concepts in approaching cloud deployment and management, as well as architectural templates, patterns and tools used by the majority of enterprises, using cloud computing and hybrid cloud, in particular. Concepts like Infrastructure as Code, GitOps, CI/CD pipelines, Kubernetes and clusters, in general, were mentioned to provide a background for further work.
Chapter 3

Methodology

In this chapter, the methodological approach used in this research is described. Methods of data collection, evaluation and analysis are also presented. The chapter is concluded with notes on ethics. The general approach is to describe the nature of the study and the way it evolved.

3.1 Study design

Both Single-Case or Multi-case design can be chosen to study in information systems (Benbasat et al. 1987). The reasons to choose a single-case study according to Gustafsson (2017) are:

- A situation previously inaccessible to scientific investigation
- Critical case
- Extreme or unique case.

Considering the specifics of game business and significant differences in enterprises system architectures and their approach to Cloud computing, each case can be considered unique. Therefore this study was focused on a single enterprise.

3.2 Approach

Creswell (Creswell 2006) states that three research approaches can be used in a research study: quantitative, qualitative and mixed methods approach. Quantitative research was designed to study natural phenomena in natural sciences. Examples of quantitative techniques are surveys, registers, and experiments (Myers & Newman 2007). Qualitative research involves an in-depth study of a phenomenon, which includes the participants’ aspects to try to determine the meaning of the specific phenomena (Creswell 2006). The mixed-methods approach is a method that is used in research studies to collect both quantitative and qualitative data by integrating the two forms of data by using various patterns that may involve philosophical assumptions and theoretical frameworks (Creswell 2006). Since the purpose is to determine the challenges in the process of cloud migration, qualitative research is conducted. Data is to
be collected from participants in an environment where cloud services are being used. This will help the researcher to get a better insight into the phenomena. The reason the qualitative approach has been chosen is to collect data from participants that are working in the cloud environment and sharing their experience. The purpose of a qualitative study is to understand issues by investigating different perspectives of people in a specific situation. These types of studies explore the influence of the social, cultural, and organizational context of a study (Maxwell & Kaplan 2005).

3.3 Methods

3.3.1 Interviews

Several interviews were conducted with platform developers and application developers at King. Platform developers, in this case, represent individuals, who are responsible for the process of designing, implementing and maintaining Hybrid Cloud architecture. Application developers, who are also defined as users in this work, are focusing on UP application development in their daily routine.

Platform Developers

Semi-structured individual interviews with platform developers were conducted in order to identify the effects of cloud migration on platform teams and collect the opinions on obstacles, stopping the enterprise from the migration.

Users (Application Developers)

Structured interviews were used as a framework for interviewing application developers, who will become future users of the Hybrid Cloud platform were conducted. The reason to select structured interviews was to follow a specific protocol for obtaining the requirements and expectations. Each migration candidate application was evaluated individually based on the answers provided. This material was later used to evaluate the rationality of application migration.

3.3.2 Internal Knowledge portal

To collect the data specific to this project, and systems architecture and details, data from internal knowledge portal was collected. Internal knowledge portal represents a set of pages, grouped into categories and organized by teams or individuals. It is used for collaboration and knowledge management inside an organization.

Internal meetings with King employees allowed to gain new knowledge and prioritize the challenges discovered.
3.3.3 Developer forums

To collect the information on practices, tools and services, which can be relevant to the cloud migration project, developer portal and forums were analyzed. This stage included analysis of forums (Stackoverflow, Reddit) and GitHub repositories of evaluated projects. The developer community has become the primary source of knowledge sharing between different enterprises and developers in the IT field. Unlike forums, they allow providing concepts and implementation of shared ideas in order to describe them in detail. By investigating several social networks, popular among cloud developers, with a list of keywords, we tried to summarize opinions on resolving migration issues and creating reliable Hybrid infrastructure.

Stackshare

Stackshare (Feed — StackShare n.d.) is a platform for anonymous sharing of internal stack used by teams and enterprises. It is a community that helps developers and engineers discover and compare software tools. By seeing how big is the community around a solution and how much support it gets, developers can simplify R&D process of finding solution fitting their requirements. Stack share was used in this research as a method of initial data collection with the aim of summarizing the current CI/CD solutions market. Result of the research are presented in the 8.1 on page 79.

3.4 Data collection

This section describes data collection. In order to keep the project relevant and beneficial for its users (i.e. target audience), a set of interviews was organized to collect opinions and expectations from the project. Users were asked about their cloud experience and main issues they have faced during application migration or creation new cloud application.

The main outcome of the project is the set of practices and a workflow, which can be used by teams, wishing to build a Hybrid Cloud platform or have an application, which needs to migrate to the cloud with the least effort and least amount of security issues. By Cloud platform we mean a set of services, provided in order to have a way to deploy and application in a created project in minutes and see it working live. Developers of the application should not be in charge of setting up logging, monitoring and alerting systems. They should not also be in charge of deploying CI/CD tools, which should be provided as a part of the platform.

Note

In order to keep data private, all interviews were anonymized. The original interview recordings are available on request.

This section describes the main challenges application developer may face during creation or migration process in the cloud.
3.5 Interviews

3.5.1 Workflow

To make the migration process more effective, semi-structured interviews (Kvale 1996) with the developers, whose projects can be applicable to the Cloud environment, were conducted. The same strategy was applied to the developers at UPF Team and other experts, focusing on the Cloud platform. These project typically included the process of simulating live environments and services in the sandbox (Sandbox (computer security) - Wikipedia n.d.) and workflow, which expects application to scale up and down rapidly.

The workflow for encoding the interview included several iterations. Interviews used in this research revealed that skills and knowledge are more important for building tools than performance metrics since market competition leads to the unification of such tools. The interview was focused on the challenges and problems faced. Therefore, the amount of positive factors is fewer than the number of challenges. The idea was not to evaluate whether the migration is reasonable or not.

3.5.2 Developer interviews

The participants for the study were selected by using purposive sampling. Purposive sampling assumes participants to be chosen based on their roles and experience in order to provide detailed insights into the researched problem (Tongco 2007). The interviewees were selected based on their knowledge and participation in cloud activities within the studied enterprise. Description of interviewees can be found in the table 3.1. We have interviewed several developers, both with and without prior experience on developing in the cloud to see, what is the main motivation for them to consider the cloud and what are the biggest challenges, stopping them from migrating their applications.

Our goal was to get collect information from the participants involved in the platform development process and developers, creating the application for future cloud platform, both with and without cloud knowledge and experience. The main reason for omitting cloud experience was the aim to discover the challenges any developer can face in the process of cloud migration. By getting their experience, it makes it easier to get an overview of what they considered to be positive and negative, which can be seen in the summarized tables 3.2 and 3.3. This approach prevents future mistakes and incidents that can cost time and money for an enterprise.

Therefore, the selection of participants was based on their working role and the knowledge of internal systems and projects. The study tries to interview people from different teams involved in cloud migration in order to obtain a full picture. The participants had different working roles such as Principal Engineer, Senior Software Developer, Engineering Manager, DevOps Engineer, Site Reliability and Associate Tech Director. All of the participants are based in King’s office in Stockholm. They were in various parts of Sweden. Considering the probabilities of withdrawal, fourteen participants were chosen for this study. Out of fourteen
participants, two of them chose not to participate in the research study. However, the size of the twelve was enough to gather information for the research.

Age and gender were not considered when selecting the participants since we believe that it should not have any influence on the quality of the data. The participants were given the freedom to answer the questions without any limitations. Choosing this method gave the opportunity to make an analysis between the interviews answers and the chosen literature (Longhurst 2003).

<table>
<thead>
<tr>
<th>Position and Cloud Experience</th>
<th>Challenges</th>
<th>Positive factors</th>
<th>Consent collected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Senior Developer, Less than a year</td>
<td>Lack of knowledge, lack of experience, investment in tools on-premise, no clear strategy/timeline, network segregation and policies issues, inter-connectivity, architectural specifics, tools migration/replacement</td>
<td>Ability to tune access in a granular way, a new approach to ownership, usage of generic tools (Kubernetes), simplify infrastructure maintenance, clear objectives</td>
<td>X</td>
</tr>
<tr>
<td>Senior Developer, Less than a year</td>
<td>Finding best practices, lack of experience, lack of staff, economic feasibility, cultural shift, inter-connectivity, choosing and evaluating third-party tools, difficulty with making decisions, data migration and access, security and integration, more stressful in the team, usage of general cloud tools</td>
<td>Chance to do things better, freedom of choice for developers, simplify developer roadmap, self-service approach to save effort</td>
<td>X</td>
</tr>
<tr>
<td>Principal Engineer, Less than a year</td>
<td>Lack of interaction, networking and separation of environments, more responsibility, tools discovery and evaluation, more pressure, losing focus, confusion about team future, lack of experience, a different approach to databases, work separation among teams</td>
<td>Simplify developer roadmap, easier to provision servers, freedom of language choice, provide developers with more tools, finding more suitable tools, unified platform, opportunity to automate and unify tooling, tools on-premise are not so good</td>
<td>X</td>
</tr>
<tr>
<td>Position and Cloud Experience</td>
<td>Challenges</td>
<td>Positive factors</td>
<td>Consent collected</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Principal Engineer, Less than a year</td>
<td>Networking and interconnection issues, no defined environments, unable to use tools in the cloud, no clear vision, huge investment in the temporary hybrid architecture may not worth it, need for R&amp;D, database usage, service discovery, full cloud focus was a mistake, the unclear role of the team, new architectural challenges, work separation among teams, hard to focus</td>
<td>Ability to create a safe testing environment, ability to create an abstraction for developers, can create a more suitable solution for us (service discovery, building), a shift towards DevOps mentality, ownership, simplify developers roadmap</td>
<td>X</td>
</tr>
<tr>
<td>Senior Developer, Less than a year</td>
<td>dependency on a third-party, two separate environments, hard to focus and progress, collaboration, access issues, integration issues, unclear vision, integration of on-premise</td>
<td>More freedom for developers, automation of tooling, simplify data developers roadmap, can simplify the roadmap for developers, easier and more flexible for developers, flexible environment in the cloud</td>
<td>X</td>
</tr>
<tr>
<td>Site Reliability Engineer, Several years</td>
<td>A lot of dependencies, a lot of steps, legacy applications, third-party solutions dependency, lack of experience and knowledge, research for best-practices, extra tools to maintain, extend some current systems, hard to migrate data, lack of cloud-native applications</td>
<td>Opportunities for automation, the culture of knowledge sharing, responsibility, sharing, shift to DevOps, define SLA clearer, possible help from a cloud provider, abstraction for developers</td>
<td>X</td>
</tr>
<tr>
<td>Position and Cloud Experience</td>
<td>Challenges</td>
<td>Positive factors</td>
<td>Consent collected</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Senior DevOps Engineer, Several years</td>
<td>Cultural shift, lack of knowledge, lack of experience, team leaders vision, approach to knowledge sharing, extra work, no clear strategy, some things are hard to predict, lack of trust, a new layer in architecture (not able to just migrate VMs)</td>
<td>Can reduce cost, chance to standardize solutions, many tools are not business-critical, ability to build an abstraction over the environment, good provider choice</td>
<td>X</td>
</tr>
<tr>
<td>Engineering Manager, Several years</td>
<td>Lack of knowledge and experience, finding the balance between support, development and research, lack of culture, lack of strategy, slow decision-making, separation of responsibilities,</td>
<td>Unified, extendable platform, the team became more determined, DevOps culture, an important role for the team</td>
<td>X</td>
</tr>
<tr>
<td>Associate Tech Director, Several years</td>
<td>Systems are optimized for on-premise performance: latency, local resources; immediate costs will increase; elasticity; extra work during the transition period; switching to third-party tools will cause problems; teams will have more responsibilities.</td>
<td>Easier to manage Kafka; hard to understand benefits; potentially will save money long-term; disaster-resilient setup; easier adoption of GCP products in the future</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 3.1: Platform Developers interview summary

**Interviews summary**

After summarizing of the interviews, the tables with migration positive and negative impacts were created. The most mentioned issues determined the focus of future work in the project. The summary can be seen in tables 3.2 and 3.3

### 3.5.3 User Interviews

User interviews were conducted using structured approach in order to clearly identify the details of the projects and well as user expectations and challenges.
<table>
<thead>
<tr>
<th>Challenge</th>
<th>Mentions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of knowledge/experience</td>
<td>6</td>
<td>Lack of knowledge or experience in the enterprise or among developers</td>
</tr>
<tr>
<td>Investment in on-premise tools, tools migration</td>
<td>6</td>
<td>Usage of custom tools specific to on-premise, which migration can be challenging</td>
</tr>
<tr>
<td>No clear strategy/timeline</td>
<td>5</td>
<td>Enterprise strategy seems to be unclear</td>
</tr>
<tr>
<td>Networking</td>
<td>5</td>
<td>Connections between zones and services, network policies, interconnect to the cloud issues</td>
</tr>
<tr>
<td>More stress/responsibility</td>
<td>5</td>
<td>Cloud migration introduces extra work and extra responsibility for services deployed in the cloud</td>
</tr>
<tr>
<td>Access &amp; Migration of data</td>
<td>4</td>
<td>Migration large amounts of data can be challenging as well as accessing all the needed data from on-premise or from the cloud</td>
</tr>
<tr>
<td>Architectural specifics</td>
<td>3</td>
<td>Architecture of the systems, which were built for on-premise specifically</td>
</tr>
<tr>
<td>Research &amp; Finding best-practice solutions</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Cultural shift</td>
<td>3</td>
<td>Changes in enterprise culture and difficulties in adoption of new approach to producing and maintaining software</td>
</tr>
<tr>
<td>Work separation for teams</td>
<td>3</td>
<td>New roles and responsibilities lead to difficulties in division of labor</td>
</tr>
<tr>
<td>Losing focus</td>
<td>3</td>
<td>Work on maintaining current solution and R&amp;D leads to lack of focus</td>
</tr>
<tr>
<td>Difficulty with making decisions</td>
<td>2</td>
<td>Involvement of several teams makes it difficult to make decisions</td>
</tr>
<tr>
<td>Lack of interaction</td>
<td>2</td>
<td>Lack of interaction between teams</td>
</tr>
<tr>
<td>Confusion about team future</td>
<td>2</td>
<td>Changed role of the team</td>
</tr>
<tr>
<td>Lack of cloud-native applications</td>
<td>2</td>
<td>Majority of applications are designed for on-premise</td>
</tr>
<tr>
<td>Lack of staff</td>
<td>1</td>
<td>Lack of staff for the migration</td>
</tr>
<tr>
<td>Economical reasons</td>
<td>1</td>
<td>Some systems are too expensive to migrate to the cloud</td>
</tr>
<tr>
<td>Security</td>
<td>1</td>
<td>Hard to ensure security in Hybrid environment</td>
</tr>
<tr>
<td>Hybrid investment</td>
<td>1</td>
<td>Hybrid environment requires investment which may not worth it</td>
</tr>
<tr>
<td>Approach to knowledge sharing</td>
<td>1</td>
<td>Teams do not share knowledge between each other</td>
</tr>
<tr>
<td>Future</td>
<td>1</td>
<td>Hard to predict how situation will develop in the future</td>
</tr>
</tbody>
</table>

Table 3.2: Challenges revealed during interviews with frequency of their appearances
<table>
<thead>
<tr>
<th>Positive side</th>
<th>Mentions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplify developer roadmap</td>
<td>7</td>
<td>Simplify developer work and deployment process</td>
</tr>
<tr>
<td>Ability to unify solutions and tolling</td>
<td>5</td>
<td>Make tooling automated and use unified services for application environments</td>
</tr>
<tr>
<td>New approach to ownership, DevOps mentality</td>
<td>5</td>
<td>Changes in the culture</td>
</tr>
<tr>
<td>Freedom of choice for developers</td>
<td>4</td>
<td>Free stack choice</td>
</tr>
<tr>
<td>Usage of better/generic tools</td>
<td>3</td>
<td>Ability to use best-practices and 3rd party solutions</td>
</tr>
<tr>
<td>Reduce cost</td>
<td>2</td>
<td>Costs to support infrastructure</td>
</tr>
<tr>
<td>Ability to improve systems</td>
<td>2</td>
<td>Make systems more reliable</td>
</tr>
<tr>
<td>Simplify infrastructure maintenance</td>
<td>1</td>
<td>Make maintenance simpler</td>
</tr>
<tr>
<td>Clear objectives</td>
<td>1</td>
<td>Clear objectives of migration</td>
</tr>
<tr>
<td>Ability to create safe testing environment</td>
<td>1</td>
<td>Allow developers to test their solutions properly</td>
</tr>
<tr>
<td>Knowledge and culture sharing</td>
<td>1</td>
<td>Improved knowledge sharing between teams</td>
</tr>
<tr>
<td>Clearer defined SLA</td>
<td>1</td>
<td>Ability to define SLA clearer</td>
</tr>
<tr>
<td>Getting help from Cloud providers</td>
<td>1</td>
<td>Ability to get support from Cloud provider</td>
</tr>
<tr>
<td>Determined team, team role importance</td>
<td>1</td>
<td>More important team role in the organization, clearer vision of team objectives</td>
</tr>
<tr>
<td>Support for managed service</td>
<td>1</td>
<td>Ability to get a managed service with support</td>
</tr>
<tr>
<td>Disaster resiliency</td>
<td>1</td>
<td>Improved disaster resiliency</td>
</tr>
<tr>
<td>Easier Cloud products adoption in the future</td>
<td>1</td>
<td>Easier future use of cloud products</td>
</tr>
</tbody>
</table>

Table 3.3: Positive concepts and opportunities revealed during interviews with frequency of their appearances

To select the interview candidates a discussion, followed up by the workshop, was conducted with product manager and developers to identify the target audience of the Hybrid Cloud platform. Discussion included both facts about internal systems at King and assumptions about other projects and developers. The output of this process was the list of developer teams invited for an interview.

The process of creating a cloud platform and setting up the environment within the team, which was in charge of the migration process, was also discussed. It helped to identify how the initial idea evolved into system architecture and how different solutions were considered.

Transcripts of the interviews can be found in the appendix.

The process of target audience identification is presented in table 3.4

In order to discover the challenges and challenges patterns in the process of migration, team interviews were conducted, which followed a set of questions.
<table>
<thead>
<tr>
<th>Question</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who is the target audience?</td>
<td>Primarily developers of UP applications that want to leverage Cloud-specific features and/or data (Game events etc.), but in the long run it will most likely be applicable to all developers in UP</td>
</tr>
<tr>
<td>What are the specific attributes of the developers affected by this opportunity?</td>
<td>They want to use Cloud-specific features, scalability requirements not suitable for on-premise, need easy access to King data stored in the cloud and independence when running their applications. They don’t want the constraints related to running things on-premise.</td>
</tr>
<tr>
<td>What are they trying to get done?</td>
<td>Build or migrate the application to the cloud, so it has a better performance or cost results in comparison to the on-premise application, spin up test environments for their applications (for each feature-branch for example)</td>
</tr>
<tr>
<td>What is the current condition?</td>
<td>The current way to run a UP Application is on-premise. All the documentation and existing solutions primarily target this. The UP Hybrid Cloud Project is investigating how to support a hybrid scenario, where UP applications can run in GCP as well. Right now we have three candidate projects for cloud migration</td>
</tr>
<tr>
<td>What does the current application developer flow look like?</td>
<td>It has two major development parts: Application (front-end) and service (back-end). There are also dependencies to external parties: UPF team (To get the application/service added to the UP Portal and to configure IAM), IE team (To setup servers on-premise)</td>
</tr>
<tr>
<td>How is this affecting the application developers? What are their pain points?</td>
<td>Current decision to separate environments in clusters does not suit all projects and becomes an obstacle for developers, migrating applications. The setup for hybrid connectivity between on-premise and GCP introduces several limitations to which features that can be used in GCP</td>
</tr>
<tr>
<td>What is the target condition?</td>
<td>Extend UP ecosystem to GCP, more precisely to a production-ready multi-tenant cluster located in shared VPC. Provide tooling such as identity management, monitoring, alerting, logging. Enable hybrid cloud solution by providing the interconnection between on-premise environment and UP cluster on GCP</td>
</tr>
<tr>
<td>How should this improve application developers’ happiness, productivity, engagement, retention?</td>
<td>Empower teams to choose a hosting environment that suits their needs and improve productivity by using tools supported by GCP</td>
</tr>
<tr>
<td>Which pains should we relieve?</td>
<td>Avoid having developers to choose between all available tools and instead provide ready to use solutions and recommendations.</td>
</tr>
<tr>
<td>Which gains should we provide?</td>
<td>Provide developers with a possibility to use an extended amount of computation resources Provide easy access to King’s data already in the cloud.</td>
</tr>
</tbody>
</table>

Table 3.4: Target audience identification
User interview questions

1. What level of experience with the cloud do you have?
2. What are you trying to build?
3. Why do you want to use the cloud for your task?
4. Do you need to use interconnection between cloud and on-premise? How, to which environments?
5. What would you expect from UP Portal in terms of cloud self-service?
6. What stops from migration to the cloud or what would motivate you to start working on it?
7. What are your current pain points when building for the cloud?
8. What would you appreciate instead?

Summary of the user interviews is presented in the table 3.5 on page 45.

<table>
<thead>
<tr>
<th>Team</th>
<th>Motivation</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>Create a sandbox environment for testing</td>
<td>No possibility to run majority of applications in the cloud environment</td>
</tr>
<tr>
<td>Teams 2</td>
<td>Migrate a data application with a need of rapid scale up/down</td>
<td>Inability to access data on-premise, absence of support, logging and monitor-ing for the application in the cloud</td>
</tr>
<tr>
<td>Team 3</td>
<td>Create an application for running application instances and testing performance</td>
<td>Lack of knowledge and experience with the cloud in a team and limited time for application creation</td>
</tr>
</tbody>
</table>

Table 3.5: Potential users of Hybrid cloud platform

3.6 CI/CD Solutions analysis

In order to select the tools for the research, data collected in the data analysis section was used. In this section, the review of CI/CD tools is described. The review has been done in order to select the most suitable solution for the majority of King internal projects. During the CI/CD tools research, a platform dependency pattern was revealed. The major part of the solutions is tied to a specific VCS system (Gitlab, Github or Bitbucket) or Cloud Provider (Google Cloud Build, AWS Code Build).
3.7 Forums and other discussion platforms

During developers’ portals and forums evaluation, a comparison of Continuous Integration tools was conducted. Solutions were evaluated based on the license, architecture and applicable domain (cloud-native etc.). Also, metrics regarding its popularity in the community and level of maintenance were collected using Github, forums on Reddit, Stackshare, Stackoverflow. The main motivation behind this work was to find the solution, which suits developers and helps simplify developer roadmap. Work proceeded with the installation of one of the evaluated solutions in the UP cluster. Summary of evaluation can be found in Appendix B.

3.8 Ethics

When working with research, it is important to consider ethics for all participants. When conducting a study with other participants, informed consent and confidentiality should be considered. (CODEX - rules and guidelines for research n.d.) Ethical considerations can also be described as more important in qualitative research, as we interact with people as subjects. Creswell (Creswell 2006) states that ethical issues should be discussed prior to conducting the study, during data collection and analysis, and in the study report.

Due to unforeseen circumstances, interviews were conducted through the web-conferencing solution, transcripts and consents were exchanged using the internet-based messaging platform. Participants who volunteered to participate in the study were aware of the risks of participation related to the future publication of the thesis. Interview transcripts were presented to them with the possibility of making minor modifications to clarify the concepts. The purpose of the study was presented to participants, and participation was voluntary. It was important that participants knew what was about to happen. The purpose of the study and background information were explained to the participants, and we also requested their permission to record the interview.

Neither name nor surname of the participants was not disclosed in the study, and they knew about the anonymity of their interviews. The participant also had the opportunity to cancel the meeting at any time or to skip the question, which, in his opinion, was uncomfortable.

Members also have the right to privacy. It is important that personal information is not disclosed; otherwise, approval will be required. Participants were informed that these interviews were used for educational purposes only.
Chapter 4

Empirical findings

This chapter will present the results that were collected from the interviews and data collection. They will be presented and divided into concepts, which represent the challenges developers have faced in this particular case. However, considering the literature discussed in the previous chapter, the majority of these concepts can be generalized to other cases of cloud migration. Ten concepts are identified and discussed based on that assumption, surrounded by quotes from the interviews.

4.1 Visualization

This section summarizes the data collection and reveals the patterns.

4.1.1 Word cloud

A word cloud was built based on the interview transcripts.

4.1.2 Charts

For visibility, platform developers interview data was converted into charts.

4.1.3 Concepts

Based on the data collected in the previous chapter, several main concepts were selected for future analysis:

1. Cloud Deployment models, Feasibility of Hybrid Cloud environment
2. Build tools automation
3. Cloud Agnostic
4. Unification
Figure 4.1: Word Cloud based on interviews’ transcripts

- Lack of Knowledge and experience, Investment in on-premise
- No clear strategy, Networks and policies, More stress & responsibility
- Data migration & access
- Architectural specifics
- R&D Challenges 3
- Loosing focus
- Difficulty with making decisions
- Lack of interaction
- Confusion about team future
- Lack of cloud-native applications
- Lack of staff
- Economical reasons
- Security
- Hybrid investment
- Approach to knowledge sharing
- Future

Figure 4.2: Challenges during cloud migration

5. Freedom for developers
6. Networking, policies
7. Cost
8. Culture, Knowledge sharing
9. Multi-tenancy and environment separation
10. Changes in team roles.
Concept 1

Build tools automation

Majority of participants mentioned simplification developer workflow and build tools automation as an important step in the process of cloud migration.

One participant described that developers expect automated pipelines for the deployment and would wish to have a level of abstraction which would not make them go into deployment details.

"We can make more automated pipelines and abstract the way for developers, so they do not have to care about it"

Another participant mentioned that all required tools and services should be already integrated into the platform, therefore they do not have to work on that part.

"Developers do not have to think about the integration themselves"

Concept 2

Vendor lock-in and cloud-agnostic approach

During the interviews, not so many developers mentioned concerns regarding vendor lock-in. Therefore several developers were asked about it on purpose based on their level of expertise.

It was discovered, that data tools like BigQuery can keep and enterprise locked to a specific provider due to tool uniqueness.

*BigQuery is used a lot in the rest of the organization, and it is a bit of a unique product. That ties us quite a lot. The hardest thing to move would be the data.*
Another developer noted that abstractions could provide flexibility in the cloud. However, he did not see it as a major advantage since the majority of services used by the team are generic to a large extent.

Certain things are impossible to abstract, there are many things which are equivalent to all the major providers, but there are still some specific ones. I do not think we will get a big advantage of abstracting it too much. From a developer point of view in Kubernetes, you just deploy a container. From the approach of Infrastructure, I would not be too concerned about it. But so far we have not tried anything provider-specific in this project.

Concept 3
Unification

Developers believe that unification will become an important part of the future Hybrid Cloud as it will keep the application and services defragmented and will help maintaining them.

By being completely free, scanning the images that we build, will not be opposed by anyone, because that provides a more unified way of doing it.

I think platform team should play an important role and unify the solutions to work altogether, so we do not have fragmented solutions everywhere and we can leverage all the learning that we do along the migration path.

Concept 4
User freedom

Platform developers are sure that migration to the cloud brings more freedom to application developers, who are considered as users of the platform and Hybrid Cloud.

Of course, with much freedom, there comes more responsibility, the open policy can help us control it. However, it should be easy to get started. Containers provide much more freedom in choosing the images and Java versions.

The whole point of moving to the cloud is the freedom of choice. They can use what they want

However, developers note that control and restrictions should apply in order to make applications follow specifications and policies.
It is the way to go: you have to give people some freedom, but they cannot create an application which does and work however they want. You have to be able to come back to specification and show them what they do not follow: we manage this cluster, and you have to be a good citizen in it

Concept 5

Networking

Networking was noted as one of the main problems in the process of migration by platform team.

I think when we talked about connecting tools zone to cloud, we expected that to be dedicated interconnect or at least purposeful interconnect, what we have had is a general interconnect to support zone, and then there is some additional bridging over to tools zone, and that causes confusion because not everyone knows how the environments and networks segments are set up on-premise, so it took a while to get that to work

we cannot wait 6 hours because of the problem with the firewall, so we do not want to rely on the public, the corporate firewall in games production and right now we are doing that

the big limiting factor is the network connectivity for me, but probably that is not everything

However, a single logical network is considered a reachable goal by developers with experience in infrastructure.

I think we can make a single logical network because we are using some tools for that

Concept 6

Cost

One of the developers noted that at this stage, running on enterprise data-centre will be cheaper due to planning and wise scaling in the past. Elasticity was not an issue on-premise since the rate is already flat as enterprise owns its hardware. However, in the cloud infrastructure, teams will have to consider elasticity in order to reduce the price, which was not the case while using on-premise.
The data centre and running on physical hardware is very cheap for us because we
are already at scale. We are going to have to start thinking about elasticity before
we just had a big data centre. And elasticity is rather a bad thing than a good thing
in my mind. And now we have to pay for all the compute we are using.

Another developer mentioned that usage of SaaS solutions could be cheaper to buy than to
implement if it is not the main focus of the business.

...there are many tools built which we are paying for already. And we do not need
a team to maintain it. Buying is sometimes cheaper. Managed services add value.
We should look at the first year and save money. We should focus on the business
value and not try to reinvent the wheel.

**Concept 7**

**Culture, sharing knowledge and responsibility**

Even though several developers noted a good culture of knowledge sharing internally at King.

_Actually, King has a culture of sharing knowledge. We have an internal portal with
information. We can access the source code of any team and contribute_

Majority of developers still believe that migration leads to a significant change in team culture
and responsibility.

_Companies’ organization and culture need to be changed in a cloud way, so this kind
of consulting can be helpful for us_

_I think from the cultural perspective. The big challenge is that we always run our
games on-premise in a particular way. It has always worked, and now it is a big
push back in changing how things work_

However, the general view of this problem is optimistic.

_King, but we are moving in the right direction. But it will take time for this culture
to take in place_

**Concept 8**

**Environments**

Two developers have mentioned that the environment has become a challenge for many devel-
opers, and they should be defined more properly in the new architecture.
The problem is we don’t have well-defined environments, we don’t even think about it, now that we are developing the hybrid cloud, we have a live UP, that is hybrid with GCP cluster, but how do we replicate that in QA and DEV for UP so we can test things, so we don’t use LIVE as a testing ground as we do now since cloud tools are still in the test mode.

Environments are the challenge

One of the future platform users mentioned the need for a sandbox environment for each PR request made in the code repository.

From an infrastructure point of view - have a sandbox environment for each PR.

There is a whole bunch of things around networking and separation of the environments, which are needed to be taken care of

Current decision to separate environments in clusters does not suit all projects and therefore becomes an obstacle for developers, migrating applications. The setup for hybrid connectivity between on-premise and GCP introduces several limitations to which features that can be used in GCP.

Target condition of the project can be defined as:

• Extend UP ecosystem to GCP, more precisely to a production-ready multi-tenant k8S cluster located in shared VPC

• Provide tooling such as identity management, monitoring, alerting, logging

• Enable hybrid cloud solution by providing the inter-connectivity between on-premise environment and UP k8s cluster.

KPIs that can be set, to verify, that the goal is matched can look like:

• Live applications running in the cloud

• Lead time between starting building something on the cloud to get it running

• Lead time from starting the migration of existing app to running on cloud.

There are several approaches, which can be used to speed up and simplify cloud migration for the teams, which mostly depends on the level of expertise in the team.
Concept 9

Unified platform

Creation of a unified system for application deployment and maintenance was mentioned by several developers and also set as an organization priority at King.

Several developers mentioned the significant role of the future Hybrid Cloud in the process of solutions’ unification.

“I think platform team should play an important role and unify the solutions to work altogether”

However, the majority of developers consider this project to be large and not achievable in a short amount of time.

There is a lot of tooling and systems and monitoring and service we have for on-premise, maybe not all of them will be needed in the cloud, but at least some of them do, a good way of handling on-call and ownership concepts and responsibilities.

Hybrid cloud is considered to be a temporary solution by some developers, since they see the enterprise operating, fully in the cloud in the future.

I do not think so, because we are moving to the cloud, so the hybrid solution is temporary, so we should not invest much to make it a golden solution

However, some people do not think it is feasible.

We can move all UP application, but things like game servers will not go there

Therefore, concerns about the clarity of the strategy can be raised.

Concept 10

Changes in team roles

Platform developers believe, that shift to the cloud added extra work and stress to daily routines.

Maybe it is more stressful now

It has been quite different because initially, we were very development focused team, writing code, using Github enterprise and deploying it. Also, using existing tools, when deploying it and monitoring it a bit. But when we started this project, it has been a lot more interaction and requirement to learn about how the infrastructure works, and what are the benefits and issues with various kinds of choices there

It is a bit more pressure now since we have it in the company strategy. It was a bit spread out
It shifted the team focus and led to misunderstanding, which was resolved as the time was going by.

There was one of the big issues, we suddenly started doing much stuff, and we stopped improving the other stuff, and I did not understand that. So I think it could have been handled in another way from the beginning, but now it is more in place.

I think it was a mistake when we started fully focusing on the cloud, and it was hard to focus for everyone 100%.

The positive side is that the migration process allowed developers to spend more time on learning and collaborating.

People become more determined, and it helps the team to focus.

4.2 Summary

This chapter described the interviews’ summary, main concepts, including challenges and advantages. Concepts then were grouped into ten by relevancy and discussed in detail. Conclusions from the interviews and further work on these concepts are described in the next chapter.
Chapter 5

Analysis

This chapter discusses the main architectural and tooling challenges, decisions and approaches taken, based on the concepts and challenges identified in the previous chapters. Solutions and decisions are based on the literature review, described in chapter 2.

According to King cloud strategy, The main motivation behind Hybrid Cloud architecture was to:

- Empower teams to choose GCP as a hosting environment that suits better their business
- Enable teams to develop faster, ship faster and scale better within UP:
  - Leverage existing self-services in GCP within UP
  - Solve some current UP development pain points.
- Get learning experiences from using GCP and Kubernetes in production
- Involve teams in the global cloud community

Based on the concepts described in the previous section and goals above, several actions were identified in order to proceed in the process of Cloud migration. This chapter describes these actions.

This work consists of the description of the execution of steps and overcoming the challenges described in this section. To estimate the process of cloud migration, several stages were defined:

1. Determine the current state of cloud usage and cloud experience in the enterprise
2. Determine the priority of cloud migration
3. Determine the possibility and rationality of cloud migration for the majority of the applications
4. Determine the required components of the Hybrid Cloud platform
5. Determine the steps required to create a hybrid platform for application migration
6. Develop a hybrid platform
7. Identify the challenges that users of the platform face in the process of migration
8. Identify the challenges the platform engineers face in the process of hybrid platform creation.

**Note** Keeping in mind, that this section describes specific case at King, term cloud can refer to Google Cloud Platform and King projects running on this platform of it specifically.

### 5.1 Changes in architecture

King defines 4 service tiers in its systems.

1. Business critical
2. Operationally important
3. Other services
4. BYOSupport (self-service)

In order to keep systems secure, network segregation splits networks into different zones. Traffic rules can be visible in the 5.1 on page 58. The network zones at King are described in table 5.1 and visualized in figure 5.1.

<table>
<thead>
<tr>
<th>Zone</th>
<th>Subzone</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td></td>
<td>T2/T3</td>
</tr>
<tr>
<td>Game</td>
<td>DEV</td>
<td>T2</td>
</tr>
<tr>
<td></td>
<td>QA</td>
<td>T2</td>
</tr>
<tr>
<td></td>
<td>LIVE</td>
<td>T2</td>
</tr>
<tr>
<td>Tools</td>
<td>DEV</td>
<td>T3</td>
</tr>
<tr>
<td></td>
<td>QA</td>
<td>T3</td>
</tr>
<tr>
<td></td>
<td>LIVE</td>
<td>T3</td>
</tr>
<tr>
<td>O-Zone</td>
<td></td>
<td>T1</td>
</tr>
<tr>
<td>DMZ</td>
<td></td>
<td>T3</td>
</tr>
</tbody>
</table>

**Table 5.1: King Zones**

The initial problem with architecture at King was the inability to access the on-premise systems and services from the cloud. The interconnect solution (*Dedicated Interconnect Overview* — Google Cloud n.d.) was connected to the Support zone, which raised large amount of issues, since the only way to access tools zone is to use the gateway, which leads provides several obstacles in accessing the game zones, which the majority of the developers need.

In order to overcome the architectural challenges, several changes in the systems architecture have been made after the discussion involving several teams. The final suggestion was made by Platform Architect, which introduces increased level of access from the Cloud to on-premise tools and systems. The new architecture can be be seen in 5.1 on page 58.
5.1.1 Gateways

In order to allow traffic to enter the zone, gateways are used. Internal gateway is responsible for handling the traffic coming outside of the tools zone at King. The internal gateway resides within the tools zone. In this network zone the gateway is able to access the support zone (including the external support zone such as GCP VPC), while only being accessible from within the internal office network. See this link for more details. The main gateway, which is commonly used is responsible for proxying traffic from tools zones to games zones. This instance of the gateway has the capabilities of proxying, authorization and obtaining access using tokens. The traffic from UP applications to O-Zone, which represents the Game data is described in the figure, which can be found in Appendix B.

5.1.2 Firewall

Another issue caused by existing architecture was presence of Corporate firewall in-between on-premise and Support zone. Corporate firewall is designed to protect the enterprise from unrestricted access to data and services. Therefore, in this scenario it is causing interference to significant amount of traffic. Such interference makes it impossible to use this architecture for live services since no guarantee on delivery/access can be provided and troubleshooting can take unacceptable amount of time. To overcome this issue a decision to create a shared VPC in Tools zone was made. This allows direct connection to Game and O-Zone. Initial state is visualized on 5.2, goal state is described on 5.3.
5.1.3 Overview

Figure 5.4 represents the overview of Cluster architecture.

5.2 Declarative approach to infrastructure. GitOps Workflow

Knowledge sharing and automation were mentioned among main issues, developers have faced during the migration. During the process of infrastructure creation, which was done by several teams, knowledge gaps and problems in collaboration appeared, which were mainly caused by different focus of which team. Some of the solution and tools had to be re-configured and
redeployed several times in order to achieve optimal configuration and ensure reliability. The need of automation of these steps in the future were mentioned be majority of interviewees.

5.2.1 Provisioning

During initial steps of cluster creation, the process of sharing responsibilities between teams was unclear, which led to desynchronization of team work. In order to keep knowledge in one place and not to separate it from code in infrastructure-as-code approach was used. Among several advantages of infrastructure-as-code approach three have been noted as they satisfy the concerns mentioned by the platform team.

- **Automated provisioning** by sharing modules for common infrastructure patterns and creating repeatable workflow
- **Collaboration and knowledge sharing** using configurations files, and state management.
- **Unification** with enabling reproducible production, staging, and development environments and mapping resource dependencies

A diagram of infrastructure-as-code workflow is presented on figure 5.5.

![Infrastructure as code workflow](image)

**Figure 5.5: Infrastructure as code workflow**

**Atlantis**

Atlantis(*Terraform Pull Request Automation — Atlantis n.d.*) is a git pull request automation tool for Terraform, which allows to keep the workflow of infrastructure changes inside git repository. Every change has to be done through pull request, which enables a clear and simple process of review and application of changes. It ensures that history of changes is kept by using git and allows to require approvals before any changes in order to keep the infrastructure secure and reliable.

In order to keep the cluster infrastructure organized as code to simplify the changes and modifications, Infrastructure as Code was used as an approach. The sample configuration of Kubernetes cluster using Terraform can be found in Appendix B.
5.2.2 Deployment

In order to keep the cluster in the defined state and keep the state reliable, GitOps approach to infrastructure was applied. GitOps is a way of implementing Continuous Deployment for infrastructure and cloud native applications. It provides a developer-centric experience by using tools developers are already familiar with, including Git and Continuous Deployment tools. Flux uses an operator (Operator pattern - Kubernetes n.d.) in the cluster to trigger deployments inside Kubernetes, which means you don’t need a separate CD tool.

*Flux is most useful when used as a deployment tool at the end of a Continuous Delivery pipeline. Flux will make sure that your new container images and configuration changes are propagated to the cluster.* (Flux CD n.d.)

There are three main advantages of using Flux CD (Flux CD n.d.).

- **Declarative approach.** Desired state of a system is described through in Git. This includes apps, configuration, dashboards, monitoring etc.

- **Automation.** Configuration in YAML is automatically managed by the tool, which analyzes differences between git repository and a cluster and keeps the state, according to the first.

- **Knowledge management.** Everything is controlled through git pull requests, which allows to keep all the changes in git history and avoid learning curve for new developers.

A simple diagram, describing the Flux workflow is presented in figure 5.6. Simple Flux deployment file can be found in Appendix B.

![Figure 5.6: GitOps Deployment workflow](image)

5.3 Separation of the environments

King on-premise solutions have

5.3.1 Approaches to tenancy

Three main approaches to tenancy can be identified in the cloud. As a result of analysis at King, a comparison table of approaches appeared. Analysis helped identify, that only two
<table>
<thead>
<tr>
<th>From Zone-env</th>
<th>To Zone-env</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool-Live</td>
<td>Game-Live</td>
<td>Allow</td>
</tr>
<tr>
<td>Tool-Live</td>
<td>Game-QA</td>
<td>Allow</td>
</tr>
<tr>
<td>Tool-Live</td>
<td>Game-Dev</td>
<td>Allow</td>
</tr>
<tr>
<td>Any other</td>
<td>Any other</td>
<td>Deny</td>
</tr>
</tbody>
</table>

Table 5.2: Restrictions of traffic between zones

options can be suitable for the enterprise. Based on a collective agreement an option with Multiple tenants in one cluster with one environment in each cluster was chosen.

In order to keep same level of access as on-premise, rules need to be established.

<table>
<thead>
<tr>
<th>Design</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-tenant, multi-environment</td>
<td>Easy to setup, Shared MGMT components (CI, Logging, Monitoring)</td>
<td>How to set up granular access to on-premise, Operational risk, Shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>components (such as Ingress) between environments, Interference between</td>
</tr>
<tr>
<td></td>
<td></td>
<td>environments, Hard to maintenance, Limitation of scalability due to address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>limit</td>
</tr>
<tr>
<td>Single-tenant, multi-environment</td>
<td>Easy to control network segmentation, Each environment has its own network</td>
<td>Need to define policies to allow access between environments, Separated</td>
</tr>
<tr>
<td></td>
<td>policy and network scalability</td>
<td>MGMT components (CI, Logging, Monitoring)</td>
</tr>
<tr>
<td>Multi-tenant, single-environment</td>
<td>More opportunities for self-service approach, Manage physical resources</td>
<td>High operational cost, Needs a strict project resource design concept or</td>
</tr>
<tr>
<td></td>
<td>per application</td>
<td>policy</td>
</tr>
</tbody>
</table>

Table 5.3: Approaches to tenancy

### 5.3.2 Rancher

Rancher is an open source project that provides a container management platform built for organizations that deploy containers in production. Rancher makes it easy to run Kubernetes everywhere, meet IT requirements, and empower DevOps teams. It allows to separate access to environments and projects and deploy sufficient infrastructure for development teams. It provides a Web UI with control panel to simplify usage of kubectl (Kubernetes control tool) and simplify deployments for developers. In order to create a multi-tenant environment for the project, a dedicated cluster for Rancher was created. Such approach allows Rancher to manage other clusters, each representing one of the environments.

### 5.4 Unification

Several developers raised an issue of inconsistency in existing platform architecture. It is impossible to control the compliance with King service in all cases and
Figure 5.7: Multi-tenant, multi-environment cluster

5.4.1 Helm

Helm (Helm n.d.) is one of the projects in CNCF landscape, that is widely used by developers deploying to cloud.

*Helm helps you manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application. Charts are easy to create, version, share, and publish — so start using Helm and stop the copy-and-paste.*

Helm combines several YAML files, normally used to deploy an application using Kubernetes control tool, kubectl, into single chart, which can be easily installed and maintained using helm client, install inside Kubernetes cluster.

Helm is used for:

- Finding and using popular software in a simple way
- Sharing own applications as Helm Charts
- Creating reproducible builds of Kubernetes applications
- Managing Kubernetes manifest files
- Managing releases of Helm packages

5.4.2 King service

King service is an internal specification for services running on King infrastructure. It defines required endpoints, health checks, ways to start/stop a service and other requirements. Every
application, developed for UP is expected to conform these requirements.

In order to create a template for application cloud migration, a Helm Chart template was created. Template defines all the configuration in order to conform to King service requirements. Such template makes it easier and faster for developers to comply with all the requirements. In order to convert an application into a helm chart, developer needs to create an image of the application and make modifications to values file, example of which can be found in the Appendix B.

Default repository for images was configured to allow storing Helm charts, so charts can be stored and updated. Corresponding plugin for Drone, CI tool used for the cloud, was used to allow automated deployments.

Policies

In order to control the compliance with King service and control tools was used. Open Policy Agents (OPA) is used to verify, that service matches the requirements provided. OPA allows to express policies in a high-level, declarative language that promotes safe, performant, fine-grained controls. This allows to unify the solutions and restrict the deployment of solutions, not complying with King service.
Figure 5.9: Multi-tenant, single-environment cluster

Figure 5.10: Simple pipeline design
Chapter 6

Discussion

This chapter discusses the strength of the results and the analysis, relating it back to the methodology and method application. Discuss weaknesses and what would be needed to strengthen the results. Discussion and conclusions are sometimes contained in the same chapter.

6.1 Sharing responsibilities among the teams

In order to define the scope of the responsibilities, a diagram for the cloud was presented by the IE team. It separates the responsibility between UPF and IE team in terms of cloud tools. The figure 5.4 can be found on page 59 and knowledge sharing.

6.1.1 Layers

Cloud introduces an extra layer in the architecture, which needs to be mentioned. Instead of maintaining the hardware, which expects and effort from infrastructure teams, a management layer appears, which allows to spread out the support among several teams, which however adds a challenge in the team role definition. The question, which has to be answered is how to perform a smooth shift to a different approach to infrastructure and redefine the approach to support and SLAs while keeping development teams effective.

6.2 Knowledge, Culture and Cloud Adoption

Lack of cloud knowledge and skills and cultural specifics inside the enterprise was discovered to be the main issues stopping the migration in the discussed enterprise during interviews.

Enterprise culture that empowers knowledge sharing and collaboration makes a significant impact on the migration process, allowing teams to overcome challenges faster and gaining the experience together. Usage of internal knowledge portals provides developers and users with a framework, which can speed up the cloud migration of individual applications.
All members of the platform team are convinced, that enforced Cloud migration is not an option, and the only way to move to the cloud is to create a platform or developer portal, which will be attractive for the developers. Creation of references and examples increases the level of trust and attitude to the cloud solution among the teams.

### 6.2.1 Shift to DevOps approach

What has been noted from the platform and infrastructure teams, is that the migration added extra responsibility to these teams, so they have to focus on both supporting on-premise applications and systems as well as devote time for R&D in the cloud. Team members noted that creation of the focus groups and improving collaboration among team members simplifies the migration process.

Moving to the cloud also introduces a shift to DevOps approach, which means more self-service solutions and shifting away from the teams focused specifically on support. It takes the ownership principle on another level and gives the technical part of the enterprise more flat hierarchy.

Migration to the cloud leads to a reevaluation of all internal systems and increases system understanding among developers.

### 6.2.2 Vendor lock-in

Platform teams believe that usage of Kubernetes makes possible system migration feasible. Majority of the tools in the cloud have alternative solutions by each of the major providers. Therefore it should not come out as a challenge.
6.2.3 Cloud Teams and Cloud Engagement

Several teams mentioned that cloud migration as a process influences team focus and team role in the enterprise. It can lead to a negative impact on a team focus and can introduce several challenges in the delivery process, at least during the transition period. To overcome this challenge, recruiting extra staff and to change priorities as suggested by the majority of developers.

6.2.4 Stack free approach

Cloud empowers teams to use whatever stack they prefer in their projects. Unlike on-premise environment, it is easier to build systems, which are stack-independent. The trade-off between stack-free approach and the restricted stack is different support level. Developers, who stick to suggested stack tend to get a higher level of support from platform and infrastructure team, as well as getting a lot of features out of the box, while wish to stick to more flexible solutions requires a higher level of knowledge of internal systems and more effort in order to make the application perform well and secure. One of the main positive sides, noted by the platform team, in the cloud migration process is the opportunity to provide developers more freedom to develop the solutions they want using the tools they want. This leads to developing systems and portals, as well as deploying tools, which intension is to simplify the development process of coding teams and let them focus solely on application development.

6.3 Shift to third-party solutions

6.3.1 CNCF

Tools used in the cloud can be completely different to what enterprise uses on-premise. It turns out that migration on-premise tools to the cloud can be as challenging, as migration cloud tools to the on-premise systems. Therefore the Hybrid solution tries to keep two different environments, which are however interconnected and can exchange data between each other.

The risk of a third-party solution is that there is no guarantee of support in the future, so one day the enterprise may become the owner of the tool, it never developed in order to keep it stable and reliable. According to the majority of the developers, external tools should be used only if they add significant business value or can help introduce new valuable features in the system.

The big cultural and development change for King as an enterprise is a shift to usage of the third-party solutions. King has a long history of using its own custom tools to solve specific problems. The cloud-native approach assumes higher usage of cloud-native third-party tools instead. Therefore, the question about the future use of such tools in the enterprise should be raised.
CNCF\textsuperscript{1} is a large community of enterprises, trying to apply the best practices on building and using cloud applications and building reliable infrastructure and cloud services. *Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. The Cloud Native Computing Foundation seeks to drive the adoption of this paradigm by fostering and sustaining an ecosystem of open-source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.*

Such communities provide enterprises missing expertise and knowledge in the process of third-party solutions discovery and evaluation. By accumulating information about cloud projects and empowering collaboration among enterprises and open-source, a higher level of cloud adoption is achieved. It has to be noted, however, that such communities and projects exist with large support of major cloud providers, acting in their interest.\textsuperscript{2}

### 6.3.2 Open-source tools

A general trend of cloud migration among enterprises leads to a higher level of contribution to open-source, that empowers knowledge sharing between tech teams in different enterprises. Spotify\textsuperscript{3} released its developer portal to open-source in the beginning of 2020\textsuperscript{4} as a solution to improve engineers productivity. The main issue the portal is supposed to resolve: introduce a standardized platform for developers, that can provide a full workflow during the process of creating and maintaining an application. Backstage consist of plugins, developed by each team in order to make it useful for each part of the enterprise. Teams use the portal to create and manage all the software that they own. Create section consists of templates, which allow developing a cloud application within minutes. Manage section consists of plugins, which represent data related to projects, which are owned by a team. Even though the open-source version is still lacking a lot of features it has in production, it describes a set of necessary components needed in order to create a cloud platform for application development. The solution is not intended for Hybrid use and is supposed to be used in the cloud, which reveals a common aim of the enterprises to migrate fully to the cloud.

\textsuperscript{1}Home Page - Cloud Native Computing Foundation (n.d.)
\textsuperscript{2}https://www.cncf.io/about/members/
\textsuperscript{3}Musik för alla - Spotify (n.d.)
\textsuperscript{4}spotify/backstage: Backstage is an open platform for building developer portals (n.d.)
Chapter 7

Conclusions

This work evaluates the migration process of UP to Hybrid Cloud architecture, mainly focusing on infrastructure deployment and redefining developers pipelines.

Among the many challenges facing the migration, the lack of knowledge and culture became one of the main concerns from teams at King. This research focused on challenges overview issues and risks facing organizations and companies planning to move towards cloud solutions. Changes in the architecture is another major concern for the decision to migrate, as both the staff and the organization can be positively or negatively affected. King has assisted this research in giving a wide coverage on how to identify and recognize the challenges. However as there are more changes and major advances in the cloud technology, it was important to investigate and identify the elements the application developers individually and teams need to understand and keep in focus before, during and after the migration.

By using individuals with experience within the cloud area, the research questions could be answered. The qualitative research methodology was adopted to understand the cloud area. Structured and semi-structured interviews from different teams at King were used as a data collection technique. System analysis and system design work have been done in order to define the way major changes have to be made in the process of migration to Hybrid Cloud.

Based on the findings, an assessment of current solutions was made, as well as evaluation of third-party solutions available on the market. Several changes in systems and approach to deployment were suggested in order to provide better control over the application and unify the process of logging and monitoring. The proposed solution allowed better isolation for each team and environment while keeping the applications flexible and accessible by other systems and services by providing more granular access policies in comparison with on-premise.
Bibliography


Longhurst, R. (2003), ‘Semi-structured interviews and focus groups’, *Key methods in geography* 3(2), 143–156.


**URL:** [https://linkinghub.elsevier.com/retrieve/pii/B9780124059320000037](https://linkinghub.elsevier.com/retrieve/pii/B9780124059320000037)


URL: http://dx.doi.org/10.1016/j.future.2017.09.020


Yang, S. Q. (2012), ‘Move into the Cloud, shall we?’.


Chapter 8

Appendix
<table>
<thead>
<tr>
<th>CI</th>
<th>License</th>
<th>Details</th>
<th>Configuration methods</th>
<th>Companies using</th>
<th>Reddit points</th>
<th>Stackshare Votes</th>
<th>Hacker News mentions</th>
<th>Stackoverflow</th>
<th>Github stars</th>
<th>Github Commits</th>
<th>Extra notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buddy</td>
<td>Proprietary</td>
<td>Container based</td>
<td>GUI/YAML</td>
<td>~200</td>
<td>~645</td>
<td>~500</td>
<td>~&lt;50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CircleCI</td>
<td>Proprietary</td>
<td>Container based. Both cloud and self-hosted options available</td>
<td>YAML</td>
<td>~5.5K</td>
<td>~1.1K</td>
<td>~1K</td>
<td>~4K</td>
<td>~&lt;1.3K</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Codefresh</td>
<td>Proprietary</td>
<td>Container based. Isolated container for each step of a pipeline.</td>
<td>GUI/YAML</td>
<td>~50</td>
<td>~1.1K</td>
<td>~50</td>
<td>~&lt;50</td>
<td>~&lt;50</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Concourse</td>
<td>Open Source</td>
<td>Container based, pipelines are built around resources (external states) and jobs</td>
<td>YAML</td>
<td>~200</td>
<td>~100</td>
<td>~&lt;50</td>
<td>~500</td>
<td>~100</td>
<td>~500</td>
<td>~22K</td>
<td>Been tested by Cloud Foundations team at King before</td>
</tr>
<tr>
<td>Drone</td>
<td>Open source</td>
<td>Isolated container for each step of a pipeline. Plugin centric</td>
<td>YAML</td>
<td>~300</td>
<td>~500</td>
<td>~300</td>
<td>~&lt;200</td>
<td>~&lt;21.1K</td>
<td>~3.7K</td>
<td>-</td>
<td>Unable to use due to usage of Github Enterprise</td>
</tr>
<tr>
<td>Github Actions</td>
<td>Proprietary</td>
<td>Container and task based</td>
<td>GUI/YAML</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>~3K</td>
<td>~&lt;1K</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Google Cloud Build</td>
<td>Proprietary</td>
<td>Builders and jobs. Plugin centric. Server required</td>
<td>GUI/Jenkinsfile (Groovy)</td>
<td>~25K</td>
<td>~700</td>
<td>~&lt;2.2K</td>
<td>~200</td>
<td>~&lt;15.5K</td>
<td>~30K</td>
<td>-</td>
<td>Main CI solution at King</td>
</tr>
<tr>
<td>Jenkins</td>
<td>Open source, Creative Commons and MIT</td>
<td>Container based, cloud native approach</td>
<td>GUI/Jenkinsfile (Groovy)</td>
<td>~100</td>
<td>~&lt;50</td>
<td>-</td>
<td>-</td>
<td>~&lt;3.5K</td>
<td>~&lt;10K</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>JenkinsX</td>
<td>Open source, Creative Commons and MIT</td>
<td>Container based with build matrix support</td>
<td>YAML</td>
<td>~9K</td>
<td>~1.4K</td>
<td>~&lt;1.7K</td>
<td>~&lt;400</td>
<td>~&lt;4K</td>
<td>~&lt;8K</td>
<td>~&lt;3.6K</td>
<td></td>
</tr>
<tr>
<td>Travis CI</td>
<td>Open source, MIT</td>
<td>Gitlab specific. Docker, shell, VirtualBox, or Kubernetes for execution</td>
<td>YAML</td>
<td>~1.4K</td>
<td>~&lt;5.2K</td>
<td>~50</td>
<td>~&lt;3.5K</td>
<td>~&lt;50</td>
<td>~&lt;8K</td>
<td>-</td>
<td>Unable due to usage of Github Enterprise</td>
</tr>
<tr>
<td>Gitlab CI</td>
<td>Proprietary, MIT</td>
<td>Gitlab specific. Docker, shell, VirtualBox, or Kubernetes for execution</td>
<td>GUI/YAML</td>
<td>~1.4K</td>
<td>~5.2K</td>
<td>~50</td>
<td>~&lt;3.5K</td>
<td>~&lt;50</td>
<td>~&lt;8K</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8.1: Comparison of CI/CD Tools
locals {
    authorized_networks = var.authorized_networks
    workload_identity_namespace = var.use_identity_namespace ? [format("%s.svc.id.goog", var.project_id)] : []
    maintenance_policy = var.maintenance_policy
}

data "google_container_engine_versions" "gke-version" {
    location = var.cluster_location
    project = var.project_id
    version_prefix = var.gke_version_prefix
}

resource "google_container_cluster" "cluster" {
    provider = "google-beta"
    project = var.project_id
    name = var.cluster_name
    location = var.cluster_location
    network = format("projects/%s/global/networks/%s", var.xvpc_project_id, var.xvpc_network)
    subnetwork = format("projects/%s/regions/%s/subnetworks/%s", var.xvpc_project_id, var.cluster_location, var.xvpc_subnetwork)
    remove_default_node_pool = true

    # Master config #
    min_master_version = data.google_container_engine_versions.gke-version.latest_master_version
    monitoring_service = var.monitoring_service
    logging_service = var.logging_service

    resource_usage_export_config {
        enable_network_egress_metering = var.bigquery_usage_export_enable_network_egress_metering

        bigquery_destination {
            dataset_id = var.bigquery_usage_export_dataset_id
        }
    }

    maintenance_policy {
        dynamic "recurring_window" {
            for_each = local.maintenance_policy

            content {
                start_time = lookup(recurring_window.value, "start_time", "")
                end_time = lookup(recurring_window.value, "end_time", "")
            }
        }
    }
}
recurrence = lookup(recurring_window.value, "recurrence", "")
}
}

enable_binary_authorization = var.enable_binary_authorization

# Cluster settings

dynamic "workload_identity_config" {
  for_each = local.workload_identity_namespace
  content {
    identity_namespace = workload_identity_config.value
  }
}

// var.use_identity_namespace ? format("%s.svc.id.goog", var.project_id) : ""

cluster_ipv4_cidr = var.cluster_ipv4_cidr

private_cluster_config {
  enable_private_endpoint = var.enable_private_endpoint
  master_ipv4_cidr_block = var.master_ipv4_cidr
  enable_private_nodes = var.enable_private_nodes
}

cluster_autoscaling {
  enabled = var.enable_cluster_autoscaling
}

ip_allocation_policy {
  use_ip_aliases = var.use_ip_aliases
  services_secondary_range_name = var.services_range_name
  cluster_secondary_range_name = var.cluster_range_name
}

network_policy {
  enabled = var.network_security_policy_enabled
  provider = var.network_security_policy_provider
}

pod_security_policy_config {
  enabled = var.pod_security_policy_enabled
}

# Features
addons_config {
  http_load_balancing {
disabled = var.http_load_balancing_disabled
}

horizontal_pod_autoscaling {
  disabled = var.horizontal_pod_autoscaling_disabled
}
}

vertical_pod_autoscaling {
  enabled = var.vertical_pod_autoscaling_enabled
}

# Node config #
node_locations = var.node_locations

master_authorized_networks_config {
  dynamic "cidr_blocks" {
    for_each = local.authorized_networks
    content {
      cidr_block = lookup(cidr_blocks.value, "cidr_block","
      display_name = lookup(cidr_blocks.value, "display_name",""
    }
  }
}

# Prevent cluster recreation
lifecycle {
  ignore_changes = ["node_locations"]
  prevent_destroy = true
}
}

8.1 Traffic
Figure 8.2: Traffic overview