Fault Tolerant, Self-Healing and Vendor Neutral Multi-Cloud Patterns and Framework Focusing on Deployment and Management

by

Andrey Rybka

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

School of Computer Science and Information Systems

Pace University

October 2017

Version 2.6

We hereby certify that this dissertation, submitted by Andrey Rybka, satisfies the dissertation requirements for the degree of *Doctor of Professional Studies in Computing* and has been approved.

Lixin Tao — 10/28/2017

Dr. Lixin Tao
Chairperson of Dissertation Committee

Date

Charles Tappert — 10/28/2017

Dr. Charles Tappert
Dissertation Committee Member

Date

Ronald S. Frank — 10/28/2017

Dr. Ronald Frank
Dissertation Committee Member

Date

School of Computer Science and Information Systems
Pace University

Abstract

Fault Tolerant, Self-Healing and Vendor Neutral Multi-Cloud Patterns and Framework Focused on Deployment and Management

by

Andrey Rybka

Submitted in partial fulfillment

of the requirements for the degree of

Doctor of Professional Studies in Computing

October 2017

Many organizations are looking to migrate to the cloud and looking for the best way to do it securely, reliably and without vendor lock in. Most organizations have to pick a cloud provider that uses proprietary APIs and Software. Most vendors currently do not implement any cloud API standards i.e. TOSCA or OASIS CAMP. Therefore, to date, the standard approaches to cloud computing have not been successful. In addition, cloud providers experience outages, frequently with serious business impact – so fault tolerance in cloud environment still needs more research and clear and prescriptive guidance. Variable performance is also an issue because most cloud providers overprovision their virtualized infrastructure and it results in degradation of performance and quality of service for customers depending on overprovisioning factor set by the cloud provider.

This study focuses on development of multi-cloud vendor neutral framework and patterns that deliver non-proprietary APIs and Software above IaaS layer with the functionality that will be on par with proprietary software or service offered by an individual cloud provider. We demonstrate how to design Fault Tolerant, Self-Healing, Performant, Secure and Cost Efficient Deployment using Patterns in the Multi-cloud environment without vendor lock in. We detail and catalog the developed solutions to common multi-cloud problems via patterns and multi-cloud framework using open source software which ensures portability across cloud providers. Framework and patterns can be re-produced by setup scripts, code and examples which are provided in the accompanying code repository. All of the failure scenarios are validated and demonstrated clearly showing the fault tolerance and limits of each solution.

Acknowledgements

First I would like to thank my advisor - Dr. Lixin Tao for all for all the help, patience, support and guidance provided over the years. In addition, I am deeply grateful to my family for their patience and support over all these years. I would also like to thank dissertation committee members Dr. Ronald Frank and Dr. Charles Tappert as well as DPS faculty who have helped me a great deal on this journey.

Table of Contents

List of Figures

Chapter 1 - Introduction

Most organizations are moving to the cloud or planning to move to the cloud. However, it quickly becomes clear that it's usually not a simple lift and load process moving from on-premises to the cloud.

So, what is one to do when they move to a cloud?

There are multiple stakeholders usually involved in this process:

- *Developers* just want to write code and deploy applications.
- *Operations and Support* professionals want to make sure that the solution is fault tolerant, not brittle and everyone can sleep well at night.
- *Security professionals* want to make sure there are no breaches and data is protected.
- *Business users* generally just want good SLA (Service Level Agreement) without any downtime for the cheapest price possible.

Some initial challenges and questions that are usually asked:

Which cloud provider should I choose?

Should I move to Amazon AWS, Azure, Google or someone else?

Is it just about a price? Should I go with the cheapest option?

What about data safety and security certifications?

My application or process needs a software stack i.e. J2EE, .NET etc. – how do I set this up in the cloud? How do I do it securely? How do I ensure there is no downtime/faults? More importantly, most organizations ask: I would like to move application X or process Y to the cloud – what is the best way of doing it?

Should I pick Infrastructure as a Service (IaaS) or Platform as a Service (PaaS)? Let's briefly illustrate the difference between the two:

Infrastructure as a Service vs. Platform as a Service

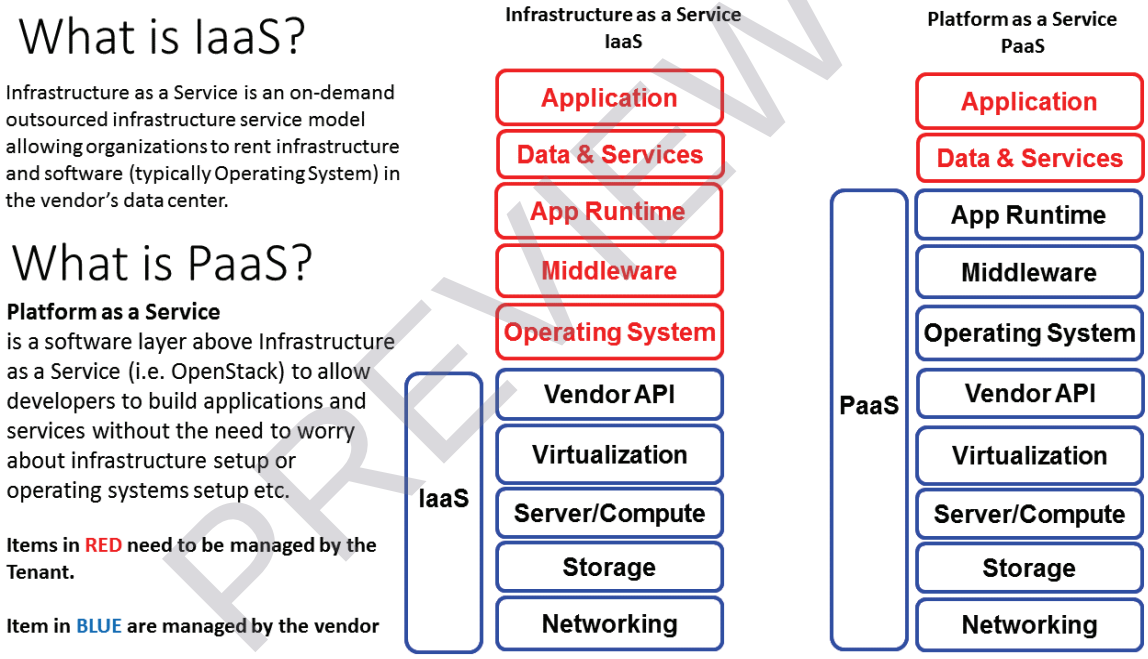
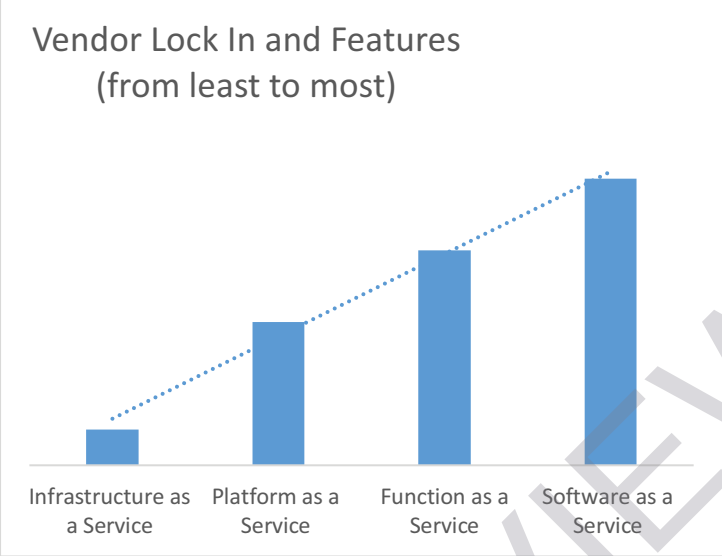


Figure 1 - Infrastructure as a Service vs. Platform as a Service

At this point, confused customers usually go to all cloud providers and every cloud provider promises that all of these will be addressed and they have excellent tools to help.

However, the reality is pretty grim. Even if a customer gets all of the questions answered and picks one cloud provider they end up locked in as most of these tools are proprietary and most cloud providers do not have or support standardized APIs.

The more features, automation and abstractions vendor provides usually results in more of lock-in for the customer. Here is high-level trend of vendor lock in of cloud offerings from least to most:



Infrastructure as a Service (IaaS): customer gets to control Resources, Virtual Machines/Compute, Networking, Storage. Example: AWS EC2, GCE.

Platform as a Service (PaaS): customer gets more features, but has less control than IaaS and more lock in. Example: Heroku.com

Function as a Service (FaaS): customer gets more features than PaaS but even less control than PaaS with even more lock in. Example: AWS Lambda, Azure Functions

Software as a Service (SaaS): customer gets more features and even less control than FaaS with even more lock in. Example: Salesforce

So is there a better way? It's usually very hard to quantify "better". However, one can argue the better way is to use a solution to a common problem which is generally defined as a pattern. In this research, we will provide patterns for common problems that organizations face when migrating to a cloud which ultimately will comprise multi-cloud framework. In addition, we will focus on building on lowest lock in layer which will be Infrastructure as a Service.

Originally most of the design patterns were software design patterns – i.e. Gang of Four Design Patterns. In this study, we will focus on infrastructure software deployment and design patterns.

What is a software infrastructure deployment pattern? It is a general reusable solution to a commonly occurring problem within a given context in software deployment. By infrastructure software we mean any software that is not application for example web service, application server software, database etc. This is also sometime referred to as software stack. Let us look at illustration of what Infrastructure Software or Stack is:

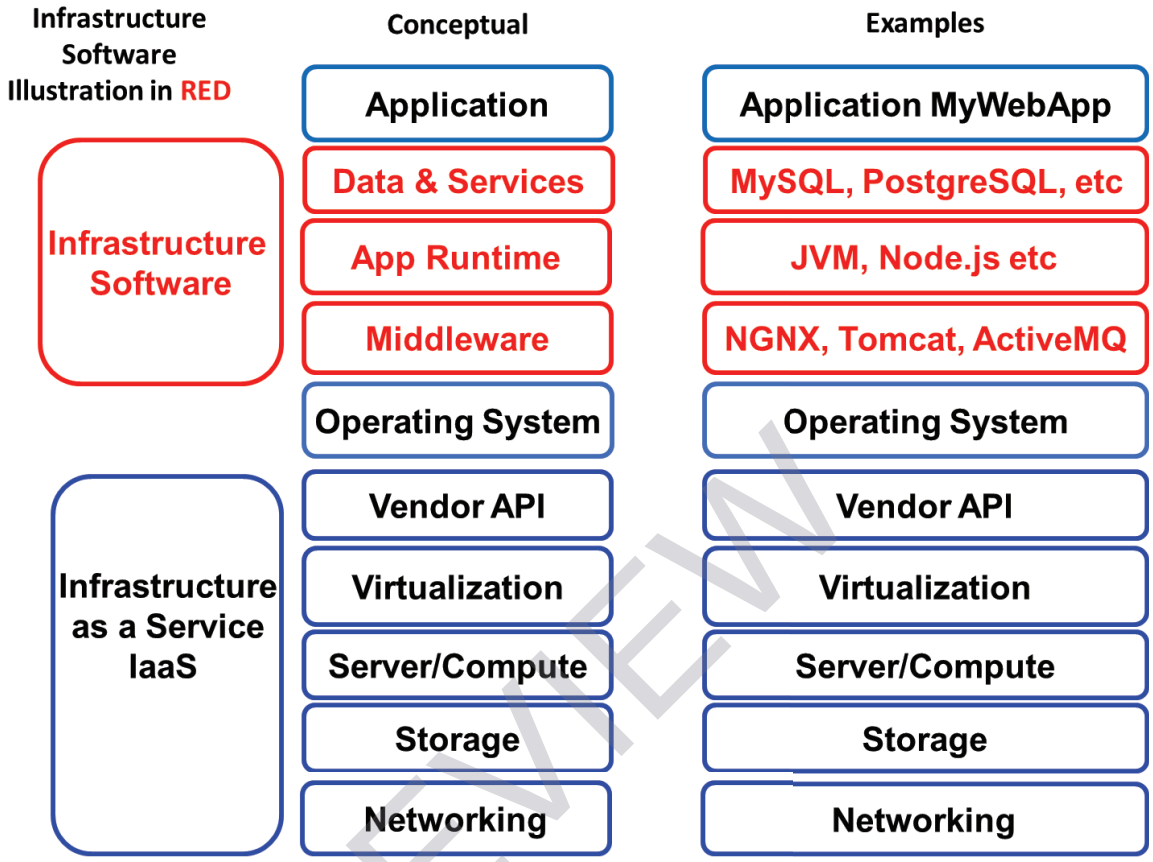


Figure 2 - Infrastructure Software

Applications need a specific software infrastructure stack to run on. Some of the common stacks can be classified as: Dynamic Web Application Stack, Media Streaming, Content Serving, Data Analytics Stack etc. These are examples of deployment stack patterns.

1.1 Multi-Cloud Deployment for Fault-Tolerance, Performance, Security and Cost Efficiency

What is Multi-cloud deployment? It is essentially the use of multiple cloud computing resources/services seamlessly as one coherent architecture. This might include Public and Private Cloud, as well as mix of SaaS, PaaS and IaaS computing resources.

Why does multi-cloud approach matter?

Fault Tolerance and High Availability

Every cloud provider experiences outages. These might be contained to one particular location, but in many cases those locations serve a very large group of clients for example U.S. Northeast or Europe. Many outages have demonstrated that high density deployments in one region with one cloud result in high impact outages to many customers. [Appendix A]. There are even services that track cloud outages in real time https://cloudharmony.com/status. (Valid by July 31, 2017)

Here the summary of total hourly impact of outages in 2015 by major cloud providers:

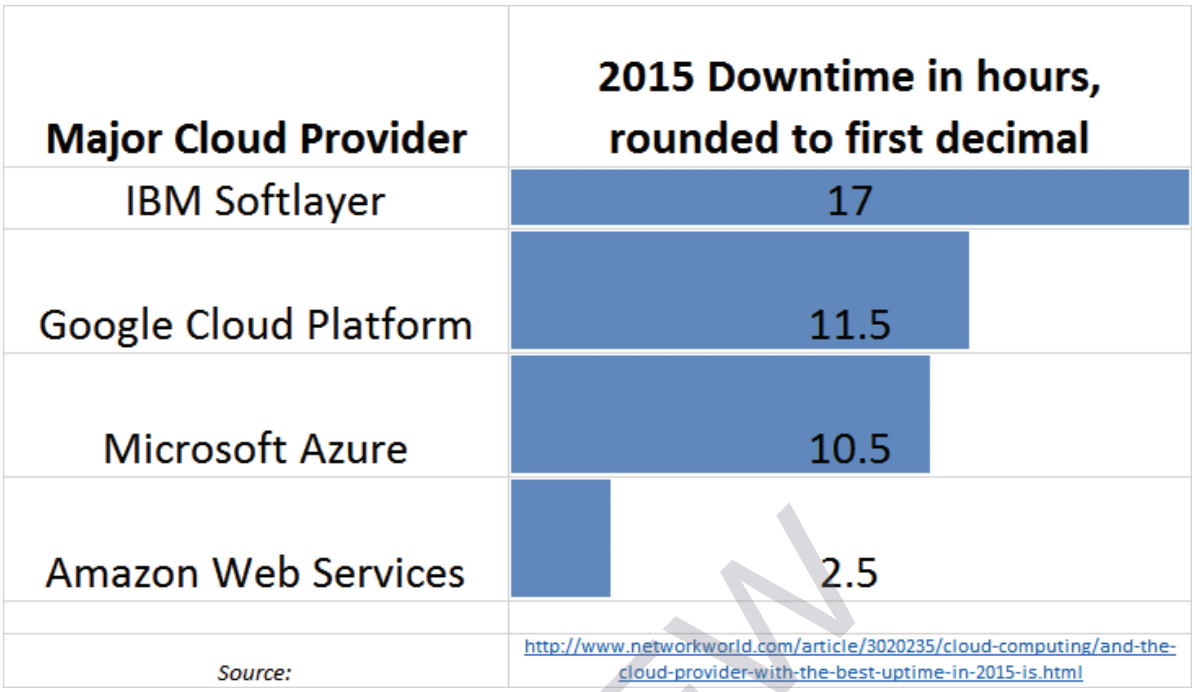


Figure 3 - 2015 Major Cloud Provider Outage in Hours

This is just a one year example - please reference a more detailed list of major outages of cloud providers for past 3 years:

Appendix A – Major Cloud Provider Outages

Performance

Performance generally has to do with how fast computer systems can perform a task X but more precisely it has to do with following computer system quality attributes: response time, throughput, latency and scalability.

One of the core issues is that not every provider has the same performance profile and at any point of time it might change due to multi-tenant nature of the cloud and because

most of workloads and component are virtualized. For example, let's take hypervisor which runs multiple virtual machines. You will most likely be sharing a hypervisor with multiple tenants and this results in a noisy neighbor problem – where one tenant might negatively impact performance for another tenant sharing the hypervisor.

In addition will focus on ability to scale Up / Vertically within the same Virtual Machine Instance and Out / Creating new instances / components.

Locality

Not every cloud provider has presence in all countries and in many cases there are different regulations that might drive the need to be in specific country. Latency and proximity matters to many use cases. In many cases you might want to run some workloads on premises and some in the cloud which can be described as Hybrid Private/Public Cloud deployment pattern.

Here is a high-level example of Multi-Cloud or Hybrid Deployment Pattern:

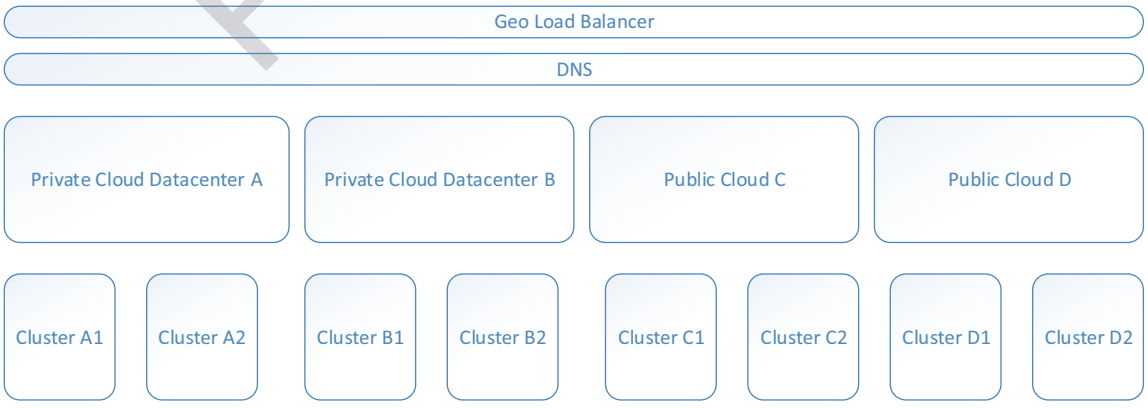


Figure 4 - Multi-Cloud Hybrid Deployment Pattern

Security

Another extremely important factor in any cloud deployment needs to be security. Cloud Providers have different services that solve this, but most of these are non-standard and proprietary which result in cloud provider lock in and customer confusion. One of the key areas we will focus on is how to protect secrets and data at rest. Security arguably should be part of every cloud pattern.

Cost Efficiency

Cloud providers change their prices quite frequently and generally the trend is to lower the price of services. In addition, there are discounted instances that cloud providers offer due to idling capacity. Multi-cloud approach allows you to shift workloads to leverage cheaper prices depending which cloud provider offers the better price. Furthermore, this body of work will demonstrate how this can be dynamically.

1.2 Current Solutions and Their Limitations

There are multiple cloud providers and some provide their own specific deployment patterns.

However, there are couple of problems with vendor provided patterns:

1). Patterns provided are usually vendor specific with specific APIs and software stacks that lock in the customer. The vendor generally has no insensitive to support standards so the customer can leave at will to another provider. Furthermore, some providers such AWS have stated that they will not support any partner that offers solution that runs on competitor's cloud platform. [36]

2). Generally, these proprietary services also cost more due to the fact that vendor can charge more for proprietary technology and convenience.

3). These patterns do not always take in account important factors such as system resilience properties i.e. fault tolerance.

4). Other patterns are very high level and theoretical and do not demonstrate the concrete software technology implementation in vendor neutral way i.e. with open source software which make these not very useful.

This leads to vendor lock in, poor deployment design and is prone to faults and poor service availability ultimately resulting in poor end user experience and data loss.

Let’s take a look at Microsoft Azure Big Data offerings – all components with exception of HortonWorks Data Platform and HBase are proprietary:

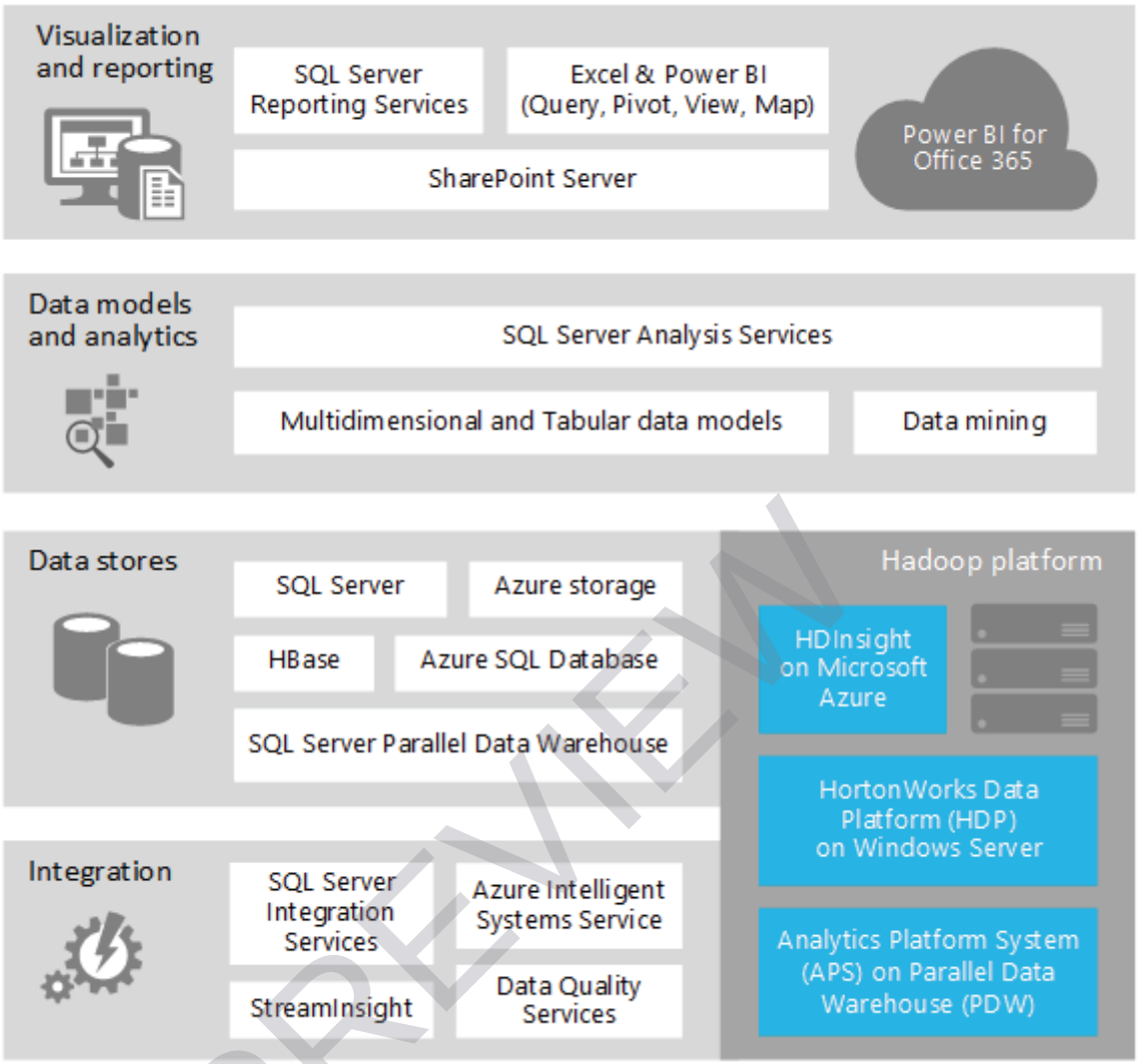


Figure 5 - Microsoft Azure Big Data Offerings Circa 2016

Source: https://msdn.microsoft.com/en-us/library/dn749804.aspx