Cloud services research document

TwitterV2

Daniil Blagoev

Table of Contents

[Introduction 2](#_Toc169513977)

[Research questions and methods 2](#_Toc169513978)

[Answers 2](#_Toc169513979)

[What types of cloud services are there and what are they used for? 2](#_Toc169513980)

[Which cloud services would be useful for TwitterV2 and why? 6](#_Toc169513981)

[How can the cloud services deemed as useful in the previous sub-question be implemented into TwitterV2? 8](#_Toc169513982)

[Conclusion 16](#_Toc169513983)

[Sources 17](#_Toc169513984)

# Introduction

The purpose of this research document is to answer the questions in the research plan with the outlined research methods. By the end of this research, the expectation is to have one or more implemented cloud services into TwitterV2. Specifically, I will be looking into Azure’s cloud services, due to the free credits for student accounts they offer.

# Research questions and methods

Main research question:

* How can I implement cloud services into TwitterV2?

Sub questions:

* What types of cloud services are there and what are they used for?
  + Literature study
    - Googling.
  + Expert interview
    - Interview with a DevOps engineer from Rabobank.
* Which cloud services would be useful for TwitterV2 and why?
  + Available product analysis
    - Analyze what TwitterV2’s requirements are and decide what would be beneficial for it.
* How can the cloud services deemed as useful in the previous sub-question be implemented into TwitterV2?
  + Literature study
    - Googling how to integrate whichever cloud service is deemed useful.
  + Joker
    - Integrating the cloud service.
  + System test
    - Test if TwitterV2 works with the implemented cloud services.

# Answers

## What types of cloud services are there and what are they used for?

The goal of this research question is to get an idea of what roles cloud services can play. There are 3 types of cloud services – IaaS, PaaS, SaaS.

#### IaaS

IaaS means a cloud service provider manages the infrastructure for you—the actual servers, network, virtualization, and data storage—through an internet connection. The user has access through an API or dashboard, and essentially rents the infrastructure. The user manages things like the operating system, apps, and middleware while the provider takes care of any hardware, networking, hard drives, data storage, and servers; and has the responsibility of taking care of outages, repairs, and hardware issues. This is the typical deployment model of cloud storage providers.

A screenshot of a computer

Description automatically generated

Figure 1 IaaS services available in Azure

#### PaaS

PaaS means the hardware and an application-software platform are provided and managed by an outside cloud service provider, but the user handles the apps running on top of the platform and the data the app relies on. Primarily for developers and programmers, PaaS gives users a shared cloud platform for application development and management (an important DevOps component) without having to build and maintain the infrastructure usually associated with the process.

A screenshot of a computer

Description automatically generated[[1]](#footnote-1)

Figure 2 PaaS services available in Azure

#### SaaS

SaaS is a service that delivers a software application—which the cloud service provider manages—to its users. Typically, SaaS apps are web applications or mobile apps that users can access via a web browser. Software updates, bug fixes, and other general software maintenance are taken care of for the user, and they connect to the cloud applications via a dashboard or API. SaaS also eliminates the need to have an app installed locally on each individual user’s computer, allowing greater methods of group or team access to the software.[[2]](#footnote-2)

A screenshot of a computer

Description automatically generated

Figure 3 SaaS services available in Azure

For the expert interview, I asked 2 questions, which would give me a general overview of the cloud service situation in Rabobank.

#### Question 1

What cloud platforms have you had the most experience with? Are there any specific services within those platforms that you find particularly useful?

#### Answer

At Rabobank, our primary cloud platform is Microsoft Azure. We leverage a lot of its services for our DevOps initiatives. Here's a breakdown of what I use most often:

* Azure DevOps: This is our main platform for managing our CI/CD pipelines. It integrates well with our Git repositories and allows for automated builds, testing, and deployments. We use features like Azure Boards for work management and Azure Artifacts for package management.
* Azure App Service: This comes in handy for deploying and scaling our web applications. It's easy to use and integrates seamlessly with other Azure services.
* Azure Functions: For serverless functionalities, Azure Functions is a great option. It allows us to write code snippets that execute on-demand without having to manage servers.
* Azure SQL Database: This managed SQL service is dependable for our relational databases in the cloud. It takes care of patching and managing the database infrastructure, freeing us up to focus on the application logic.

Of course, we also dabble in other cloud technologies depending on the project. There's always something new to learn in this field!

#### Question 2

What is an aspect in the project you are working on which greatly benefits from cloud services? How does it benefit?

#### Answer

One of the biggest wins for using cloud services in our current project is the speed and efficiency of our API integration process. Traditionally, integrating APIs involved a lot of manual configuration and coordination between development and operations teams.

Here's how cloud services streamline things:

* Pre-built Integrations: Cloud platforms like Azure offer pre-built connectors and APIs for many popular services. This saves us a ton of time compared to building integrations from scratch.
* Infrastructure as Code (IaC): With IaC tools like Azure Resource Manager (ARM) templates, we can define our infrastructure configuration as code. This allows for automated provisioning and deployment of the resources needed for the API integration, ensuring consistency and repeatability.
* Scalability: Cloud services are inherently scalable. As the volume of data flowing through the API increases, we can easily scale up the resources allocated without having to worry about managing physical servers. This keeps our integrations running smoothly even during peak loads.
* Monitoring and Logging: Cloud platforms provide robust monitoring and logging services. This allows us to proactively identify and troubleshoot any issues with the API integration, ensuring uptime and reliability.

Overall, cloud services take the heavy lifting out of API integration, allowing us to focus on the core logic and business value of the integration itself.

## Which cloud services would be useful for TwitterV2 and why?

As it can be seen from the screenshots in the answer to the previous research question, there are numerous Azure products for each type of cloud service. It is the goal of this sub question to find out which ones would be beneficial for TwitterV2 based on non-functional requirements and implement them. I will also be basing my choices on what I think will be useful for me to have experience with as a professional in the field.

It is important to note that the number of cloud services I can implement is limited by the number of credits available in the student account. If I implement too many, it will result in me losing all my credits before the semester is over. Therefore, I will not be able to implement every single service which is remotely useful.

#### Non-functional requirements of TwitterV2

TwitterV2’s non-functional requirements include:

* Scalability & Performance – Ideally, TwitterV2 will be able to perform whether it has 500 or 500 million active users.
* Reliability – TwitterV2 needs to have the highest uptime possible.
* Security – Authentication & Authorization, Data encryption, common attack protection.
* GDPR Compliance – Ensuring data privacy and consent management.
* Monitoring – Monitoring of system metrics and application events for troubleshooting.

#### Choice of cloud services to implement

Before I choose which cloud services I am going to implement, it is important to note that I will be dealing with the Security and GDPR Compliance non-functional requirements myself. Therefore, I will not take them into account when making a choice.

##### Choice 1: Azure Kubernetes Service (AKS)

Reasons:

* Scalability - AKS excels at automatically scaling web applications based on traffic demands. During peak usage, AKS can spin up additional instances to handle the load, and then scale down when traffic subsides. This ensures TwitterV2 remains responsive and avoids bottlenecks.
* Reliability - AKS utilizes a distributed architecture, meaning TwitterV2’s services will run across multiple containers on different virtual machines. If one container or VM fails, AKS will automatically restart the container on another healthy VM, minimizing downtime and ensuring high availability.
* Relevancy to my future as a professional in the field – Having experience with deploying a web application to the cloud would be a big plus for my CV in the coming years, as the vast majority of software products are deployed on a cloud platform.

##### Choice 2: Prometheus metrics + Grafana

Reasons:

* Seamless integration with existing AKS deployments.
* Monitoring - Metrics like CPU and memory usage, network traffic, and restarts at the container level can be monitored. This helps identify issues specific to containers, such as resource starvation or unexpected restarts.
* Performance - Gives the ability to analyze resource utilization across pods and nodes in the AKS cluster. This helps identify resource bottlenecks and optimize container placement for better performance.

##### Choice 3: Cosmos DB for MongoDB

Reasons:

* Scalability - It can handle fluctuating traffic demands without manual intervention, ensuring TwitterV2’s database can accommodate growth.
* Globally Distributed - Cosmos DB offers geographically distributed databases, allowing replication of TwitterV2’s data across multiple Azure regions. This enhances data availability, reduces latency for geographically dispersed users, and improves disaster recovery capabilities. This will help me with achieving the Distributed Data learning outcome.

Now that I have chosen the cloud services, it’s time to learn how to implement them into TwitterV2.

## How can the cloud services deemed as useful in the previous sub-question be implemented into TwitterV2?

#### Azure Kubernetes Service

##### Creating a Kubernetes cluster

I started off by logging into Azure, going to the marketplace, selecting AKS and filling everything in. That’s all it took to create the cluster.

A screenshot of a computer

Description automatically generated

Figure 4 The AKS configuration screen

A screenshot of a computer

Description automatically generated

Figure 5 The created AKS twitterv2 cluster (at the top of the list)

##### Automating the deployment to AKS

Now that I have a cluster, the next step was to automate the deployment through my GitHub CI/CD pipeline. For this, I am going to follow the official Microsoft documentation[[3]](#footnote-3).

Here are the steps I took to add automatic deployment to my pipeline:

1. Navigate to the newly created cluster -> Automated deployment -> Create -> Deploy an application (since my application is already containerized).
2. Fill all the needed information:

A screenshot of a computer

Description automatically generated

Figure 6 Pointing Azure's automated deployment to my GitHub repository

1. After filling in the fields in the 3 steps from the screenshot above, Azure automatically sent a pull request to my user service GitHub repository, which contained the deployment workflow file. I then merged it. The pull request also added all the needed Azure credentials as secrets.

A screenshot of a computer

Description automatically generated

Figure 7 Automatically generated pull request by Azure to my GitHub repository, which provides the workflow file for deployment to AKS

A black rectangular object with white lines

Description automatically generated

Figure 8 The automated pull request also added GitHub secrets needed for the AKS deployment in the workflow

1. Repeat for the frontend and gateway.

##### End result

This is a screenshot from the Azure shell on which it can be seen that all 3 deployments were successful and the pods are running.

A screenshot of a computer

Description automatically generated

Figure 9 The AKS deployment was successful and the pods are running

#### Prometheus metrics + Grafana

Adding Prometheus and Grafana was an option during the AKS cluster creation process, so I just chose to add them prior to the AKS cluster was created:

A screenshot of a computer

Description automatically generated

Figure 10 The steps needed to add monitoring to the AKS cluster

Afterwards, I navigated to the Monitoring section of my AKS cluster and opened my Grafana workspace.

A screenshot of a computer

Description automatically generated

Figure 11 Where the Grafana workspace can be opened from

There were already some pre-created dashboards in Grafana such as this one, which displays CPU and Memory metrics (default is the namespace in which TwitterV2’s pods are):

A screenshot of a computer

Description automatically generated

Figure 12 Pre-created dashboard in the Grafana workspace for the AKS deployment

#### Cosmos DB for MongoDB

For this one, I will be following this guide in the official Microsoft documentation.[[4]](#footnote-4)

First, I created a CosmosDB for MongoDB resource in Azure:

A white background with black text

Description automatically generated

Figure 13 The created MongoDB resource in Azure

Next, I copied the connection string:

A screenshot of a computer

Description automatically generated

Figure 14 The connection string needed to connect TwitterV2 to Azure's MongoDB

And replaced my previous connection string with the new one:

A screenshot of a computer

Description automatically generated

Figure 15 Added Azure MongoDB connection string to my code

After all implementations from above, I tested the deployment to see if everything was working correctly and if TwitterV2 could properly scale in the cloud environment:

[Testing the AKS deployment (with scalability).](../aks%20deployment%20with%20scaling.mp4)

# Conclusion

This research successfully identified and implemented cloud services to enhance TwitterV2. By leveraging Azure Kubernetes Service (AKS), Prometheus metrics with Grafana, and Cosmos DB for MongoDB, the project achieved significant improvements in scalability, reliability, distributed data storage, and performance monitoring.

Key takeaways include:

* Understanding the three main cloud service models (IaaS, PaaS, SaaS) and their use cases.
* Learning how a professional organization utilizes cloud services (through the Rabobank interview).
* Gaining practical experience deploying and integrating various Azure services with TwitterV2.

# Sources

*Types of cloud computing*. (n.d.). https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud

Polkovnikov, A. (n.d.). *Azure Services Overview*. https://azurecharts.com/overview

Q. (2023, May 23). *Automated deployments for Azure Kubernetes Service (Preview) - Azure Kubernetes Service*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/aks/automated-deployments

G. L. (2023, March 30). *Connect a MongoDB application to Azure Cosmos DB*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/connect-account

1. Polkovnikov, A. (n.d.). *Azure Services Overview*. https://azurecharts.com/overview [↑](#footnote-ref-1)
2. *Types of cloud computing*. (n.d.). https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud [↑](#footnote-ref-2)
3. Q. (2023, May 23). *Automated deployments for Azure Kubernetes Service (Preview) - Azure Kubernetes Service*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/aks/automated-deployments [↑](#footnote-ref-3)
4. G. L. (2023, March 30). *Connect a MongoDB application to Azure Cosmos DB*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/connect-account [↑](#footnote-ref-4)