UX in cloud platforms

Yaqing Zhu

1 INTRODUCTION

In recent years, web-based projects are one of the hot topics. In this paper, I selected top 3 popular cloud services: AWS, Azure, and Google Cloud Platform (GCP). Those are the most used cloud services based on income shared by those providers. I will empirically evaluate and analyze user experience of all stages of web-based project development of the three cloud services.

2 OBJECTIVES

The key aim of this paper is to evaluate the user experience of some commonly used products based on common web application development process introduced by [1] and [2].

3 RESEARCH AND ANALYSIS

3.1 Free Tier of Top 3 Cloud

Cost of cloud can be expensive. Besides, to migrate a project from one cloud to others can be complicated. Free tier is a good solution for this issue, which can allow potential users to try services without cost. Potential users can evaluate cloud providers from all aspects that they need to do in terms of finishing project development.

As we can see now, all of three cloud providers have free tiers. However, their free tier plans have huge differences. Both AWS and GCP offer a 12-month free trial whereas Azure only has 30 days. According to research [3], a small medium-sized project needs about 6 months to be implemented, and the coding stage need 1/3 time of the period which is about 2 months. Therefore, Azure's free trial cannot enough for developing even if users finished design work before starting their free trials. To evaluate a platform only from developing perspective is not enough. Availability, stability, and performance are also critical and can only be evaluated by deploying and maintaining.

During the 12-month free trial, there is no limit on usage and quota when you use AWS, but users can only use a maximum of $300 USD. However, this amount of money should be enough for a small medium-sized project based on prices of some GCP products, such as app engine which is a web server host product, spanner database, and Kubernetes [4]. My assumption is based on the price calculator provided by Google.

3.2 Frontend Hosting

Front-end is a must-have component of web-based project. All three cloud providers have products for users to host their front-end components: Azure App Service, AWS Amplify, and GCP App engine. In this section, I will evaluate UX of all three products from three aspects: deploying, developing, and managing. Both App Service and

App engine provides frontend and backend hosting services. In section 3.4, I will introduce more about backend related user experience analysis based on paper [9].

There are some common flexibilities provided. Firstly, all the three clouds allow users to select deployment locations during instance creation. This function is important since server location has significant impact on web page loading speed. Also, they all allow users to select own URLs and domains.

Integration of CI/CD is also an important feature that can affect user experience because this can allow users to deploy and test their codes faster and easier. This feature also provided by all three clouds.

It is hard to reach the load limitation for frontend only hosting because of the natural of how HTTP work. All the JavaScript execution will rely on clients' browses so that frontend only hosting just need to transfer web page sources which are plain texts to clients. Therefore, I will leave monitoring related user experience analysis of Azure and GCP in Section 3.4.

There are still many differences, and I suppose some of the differences have impact on UX. I did the analysis from 3 features: CI/CD function, Auto-scaling, and instance creation, and will introduce them in the following sub-sections and analyze impacts. Table 1 shows a summary of the analysis.

Table 1: summary of frontend products analysis.

Product Name	CI/CD function	Auto-scaling	Instance creation
App Service	Easy to implement with one extra step.	N/A	Easy form style
Amplify	Easy to implement during creation.	Fully auto process	Easy install wizard style
App engine	Hard to implement by using other product.	Fully auto process	CLI style with extra steps to install SDK

3.2.1 Azure App Service.

To create an instance, users just need to click "create" button on webpage. Instance name, instance location and instance coding language are mandatory fields and there are no default values. Without default values, users must write their own values accordingly in case users can click "finish" button by accident.

App Service supports CI/CD with Git, BitBucket. Besides, developers can download Azure visual studio code extension to deploy directly from coding workspace. To set up CI/CD process, users only need to go to your instance management page and follow the instructions.

However, App Service still have disadvantages in terms of UX. Firstly, App Service is the only one among three products that does not have auto scale function. Users must select server type by their own, and details are only available on other webpages. Secondly, Azure does not have a centralized search bar that can search everything. Each product has its own search bar to search settings of it. This is a common issue of Azure that is not only specific for App Service.

3.2.2 AWS Amplify.

Instance creation is as simple as Azure. Amplify provides an extra step during creation to allow users select CI/CD repository, which is the best among the three products because CI/CD is a mandatory component so that the easier users can set it, the less extra work they need to do. The downside of creation process of Amplify is the default value of URL. Since Amplify does not allow users to modify existing instances, they must delete it and re-create a new one

if anything wrong. Pre-populated default values void mandatory field checking so that users can create an instance with wrong URL.

Amplify also has a management page that can view information related to this instance. However, users need to switch to the management page first. This adds one extra step compared with Azure. The management page of Amplify does not have as many as information of Azure. The only information on it is access log and domain information. All other information such as system metrics and quota usage need to be checked in separate products. This type of UX design do add too many extra works when troubleshooting. Troubleshooting on complex bugs or issues need to check much information together, which means Amplify users need to go to many different AWS products and select this Amplify instance every time when switching to a new AWS product. According to [5], the user happiness constantly dropping when the network communication time increasing. Frequently switching between different products needs extra network communication time so that user experience can become significantly bad during complex troubleshooting.

Amplify has one unique function that can determines framework/language automatically so that you do not need to worry about language, and language version anymore.

3.2.3 GCP App engine.

Both App Service and Amplify allow users to create instances from webpage. However, to create App engine instance, users need to use GCP SDK which needs many extra steps. I tested to install the SDK and set up account login, which spent about 10 mins as an experienced user so that the time needed can be larger for first-time users.

CI/CD integration is not directly supported by App engine. Users need to leverage another product called cloud build. Also, in App engine instance page, there is no hints or suggestions tell customers that cloud build can provide CI/CD function. As CI/CD is a must-have component of web applications, users must find out how to implement it by searching online.

3.3 SQL Database

Data storage is the place to store data related to the web applications. According to [2], web applications usually need serval different types of data storages, such as SQL database for strong consistency, NoSQL database to get faster distributed performance, and GFS [6] or similar products for large immutable files. Since this paper is only focus on user experience of existing products, I will not analyze the product diversity such as does this cloud provide all types (key-value, document, column-oriented, and graph) of NoSQL databases. All three clouds provide VM products and Kubernetes to all users deploy their own data storage clusters.

In this section, I will evaluate SQL database and analyze file storage in next section. The reason I skip NoSQL is that this paper only evaluates user experience and NoSQL databases (Azure Cosmos DB, AWS DynamoDB, and GCP Bigtable) in all three clouds have similar user interface and features from user experience perspective. However, file storage has many different functions such file URLs, file organizations, etc. Table 2 is the summary of the analysis of SQL database.

Table 2: summary of SQL database products analysis.

Product	Security	Stability	Easy to Integrate	Note
Azure SQL	Standard firewall and backup.	Allow users to select replication locations	Standard IP connection	Only one has auto-scaling feature.
RDS	Standard firewall and backup plus IAM authentication.	Auto zone/country level of replications.	Standard IP connection	N/A

Product	Security	Stability	Easy to Integrate	Note
GCP SQL	Standard firewall and backup plus IAM authentication. Only one can select backup time.	Auto zone/country level of replications.	Standard IP connection plus connection text feature.	Service label allows users to connect other GCP products without IP address

As a user of database, the important features of database are data security, service stability, and easy to integrate with backend [8]. Azure SQL, AWS RDS, and GCP SQL all provide a variety of features to fulfill users' requirements.

All three clouds are using form style to let users create database. All required fields are clear and easy to understand because they all have hover over hints indicating detailed explanation of each attribute.

Unlike the differences of monitoring features of frontend products, all three database products provide integrated monitoring functions even though different cloud has different coverage. The details will be introduced in the following paragraphs as well as user experience analysis on important features listed above.

One important user experience related feature that Azure fall behind is Azure SQL is the only one type of database whereas RDS and GCP SQL have multiple selections, such as MySQL, SQL Server, and PostgreSQL. This is not the same as product diversity exclusion that I talked above since database type selection is a feature of single cloud product and this feature do have impact on user experience: according to [7], when people deciding to use a cloud product, collaboration is an important factor. So, the more people know how to use this database the higher likelihood they will select it. Azure SQL is only provided by Azure cloud, which means less people know how to use it than MySQL and PostgreSQL.

3.3.1 Azure SQL.

To keep users' data security, Azure SQL provides two features: firewall and backup. To setup a firewall rule, users only need to add a range of IPs. The setup process is straightforward, and users can easily find the setup page on the top of database instance management page. Backup is not configurable and managed by Azure as well as RDS. This kind of approach does not have any flexibility that allows customers to select backup time based on business needs compared with similar features provided by GCP SQL.

To increase stability, the most common method is adding replications, including geo- replications and failover. To add replications in Azure, users only need to click geo- replications link from database instance management page. In the new page, users need to select secondary replications locations. Azure is the only cloud among the three that needs user to select replications locations. Unlike backup time, replications locations do not have impact on user business. Most common selection is to select same zone/country as the location of replications.

Besides of replications, the server machine type also can impact the stability. Too high CPU and memory usage usually leads to high queue time for queries, which can reduce the overall performance. Therefore, to select the right server machine type is important. However, GCP SQL still require users to select machine type, which requires many extra works to determine business requirements.

3.3.2 AWS RDS.

AWS RDS also has above two features to keep data security. RDS has one extra feature that allow users to select authentication type: normal database password or AWS IAM. IAM authentication has many advantages over database password such as one password for all services deployed on AWS, all protection features provided by IAM

including password cracking protection. Therefore, IAM authentication is a positive point of user experience. As mentioned above, RDS also uses pre-defined time for automatic backup.

RDS and GCP SQL both use automatic zone/country level replication strategy. However, there are cases that users want to deploy the replications in different zones so Azure allows you to do that at the cost of one extra step during configuration.

RDS has the most convenience feature: auto-scaling. User can only specific the upper and lower bound of resources such as CPU number, RDS will automatically scale up or down based on workload. This is the same as front-end products.

3.3.3 GCP SQL.

GCP SQL provides the most powerful data security features that can lead to a better user experience. First, GCP SQL provides all similar features as RDS, plus users can select their own backup time according to their business requirement. To setup backup time just needs one extra step during creation. Secondly, GCP SQL firewall supports not only IP also service label which is a feature of GCP that can allow other GCP products to connect to database without IP address. This one has positive impact on user experience because users do not need to check each IP address one by one and add it to firewall rule.

All three products provide IP based connection. GCP SQL has one unique feature which is connection text. Each database instance has one connection text which can allow other GCP products connect this instance by using only the text and IAM authentication. This does not have directly impact on user experience but to develop applications using this feature can have better user experience since connecting to a database becomes easier than using IP address.

3.4 File Storage

File storage, as a special type of database, has many common user experience requirements of SQL database. In addition to those common areas, file storage usually is used to store large files which means users care about latency when do collaborations with others indicated by [8]. In this paper, collaborations mean distributed servers working with file storage products.

Usually, a common solution to reduce latency is to use global distribution system or GDS. GDS will distribute your files to many physical locations globally so that backend servers can populate files from the nearest location. All three file storage products, Azure storage account, AWS S3, and GCP Storage, support GDS features. However, there are still some differences in terms of user experience. Table 3 is the summary of the analysis of file storage.

Table 3: summary of file storage products analysis.

Product	GDS/endpoints	File Access
Storage account	Needs to go to other product before setup	Need to open cloud console and search manually.
S3	Easy to implement at instance management page.	Need to open cloud console and search manually.
GCP storage	Easy to implement at instance management page.	Can have GCP URL to share with others who can directly view the file via URL.

3.4.1 Azure storage account.

To enable multiple endpoints access on Azure storage account, users must go to Networking product to setup endpoints first. Then come back to storage and link endpoints to your files.

3.4.2 AWS S3.

In AWS S3 and GCP storage, you can add your endpoints just in the instance page. Therefore, Azure needs one more step to implement this feature causing negative impact on user experience.

3.4.3 GCP storage.

Besides GDS/endpoint, GCP storage provides one extra way to access files. Usually, users need to use API provided by clouds to do operations on files. GCP storage provides URL like style which generates a GCP URL to allow users share the files with authorized persons. Without this feature, storage account and S3 need to ask the persons who want to view files logging into cloud console and find the files from the file list. This needs many extra steps.

3.5 Backend Hosting

Backend is the layer between frontend and data storage. Most of the logic and process work will be done at backend. Paper [9] did a load test on top three backend hosting products: Azure App Service, AWS EC2, and GCP App engine. From the result of [9], we can notice that how easy for users to config the products to be ready for heavy workload, and to find exception stack trace are important. Developers can have many ways to reproduce frontend issues such as Chrome developer console, Visual Studio Code. However, backend logging system usually is the only option to get related error messages since personal computer is not powerful enough to host the whole backend. Based on above analysis, I will evaluate backend hosting user experience from two points: how easy for users to config the products to be ready for heavy workload; how easy to use logging system. Since auto-scaling is covered by frontend section, I will only focus on diagnose and metrics to help users to determine current healthy of their web application backend when talking about config. Please note, EC2 do support auto-scaling. Table 4 shows the summary of this section.

Table 4: summary of file backend products analysis.

Product Name	Monitoring	logging
App Service	Monitoring functions in the instance page to view key metrics. Do not have flexibility to change time range	Easy form style
EC2	Monitoring functions in the instance page to view more key metrics than App Service and have flexibility to change time range.	Easy install wizard style
App engine	Almost same as EC2 but details of metrics are in different pages.	CLI style with extra steps to install SDK

3.5.1 Azure App Service.

There are two ways to diagnose performance of users' application on App Service. Users can use a tool provided by Azure to automatically investigate their instance by using keywords wrote defined by users. Those keywords include HTTP error code, resource usage, and server status. This unique tool gives users ability to find common issues quicker, which can be a significant improvement on user experience.

From App Service management menu on the left of instance page, users can easily find all related information such as quota and metrics. The kind of design is very helpful for DevOps engineers, which can allow them to find all information quickly without switch between webpages or products. This way is traditional: users check metrics and do their own judgements. App Service provides key metrics (5xx errors, 4xx errors, CPU usage, QPS, etc.) in dashboard page of your instance so that users do not need to go anywhere else. Users can also view details by clicking each metric to re-direct to other page.

In terms of logging, users must learn a new query language called KQL. Query style can be helpful when finding single query from huge amount of data but can be slow when users just want to list all logs from a certain period.

3.5.2 AWS EC2.

To view metrics of EC2 instance, users just simply need to go to instance overview page. Metrics listed at overview page are more than App Service. Also, same as GCP App engine, users can select time range to view different range of metrics, which is not supported by Azure. What is more, EC2 supports users to view some details by just clicking on metric without re-directing to other pages.

Same as Amplify, EC2 also do not support viewing logs from instance page. Users need to go to logging products to find log.

3.5.3 GCP App engine.

User experience of metric monitoring provided by App engine has a score between Azure and AWS. First, App engine management page has fully covered information of the instance, and management page has unique dashboard which displays more important metrics than Azure so that users can see them easily. However, the reason why I give a lower score than AWS is the detail of each information locate at other GCP products. App engine provides links which can directly go to that product with parameters to load detailed information. This process still needs extra page switching even though it is better than Azure.

App engine has the most convenient logging system. Firstly, App engine dashboard lists most recent errors so that users can view them directly. Secondly, App engine allow users to view logs based on datetime range, severity, and keywords, as well as query-like languages. Therefore, users can have the most flexibility here to find the logs that they want.

4 THREATS TO VALIDITY

Threats to validity of those two references [1] and [2] that are using as guidance of web application development are rapid growing of web development. There are more and more advanced techniques and frameworks in recent years so that the modern web application structure can be different. However, this paper only focuses on user experience and most of web applications' high-level structure is still frontend, backend, and data storage.

Paper [7] and [8] was evaluated based on other type of cloud storage, e.g., Google drive and Dropbox. The only difference is the users. Google drive and Dropbox are facing normal customers whereas database and file storage are facing developers who developed the former. Therefore, performance of latter is the one of the causes of performance issues of former but there are still other causes such as network issues. What is more, [8] states that security is the most concern when using cloud storage. However, there is no benchmark to compare it with other storages. So that security can be one of top concerns in all kind of storages, online or offline.

Paper [9] did load testing against three clouds. However, in this paper, authors did not explicitly indicate if they are enabled auto-scaling or not. Since auto-scaling double-edged sword where can reduce the performance if scaling algorithm is not good enough. Therefore, I assumed all three clouds are having positive algorithm to make auto-scaling does not have negative impact on speed.

5 CONCLUSION

In this paper, I analyzed user experience of three most popular cloud from a web application developer perspective. Each product has its own strength and weakness.

Azure and GCP use integrated products for both frontend and backend therefore, App Service and App engine have more powerful monitoring and logging systems than Amplify, which can have better user experience since users can use less effort to collect useful information. AWS logging system has the worst integration with other AWS products because both Amplify and EC2 must go to logging product to view logs. However, AWS does best in terms of CI/CD integration and users can make it done by just one click whereas App engine needs more than 10 mins to do the same work.

For both SQL database and file storage, GCP products have the most flexibility to support users where leading to better user experience.

REFERENCES

[1] Darwin, Peter Bacon, and Pawel Kozlowski. 2013. AngularJS web application development. Packt Publ.

[2] Chen, Jim Q., and Richard D. Heath. Web application development methodologies. Web Engineering: Principles and Techniques. IGI Global, (2005), 76-96.

[3] Huang, W., Li, R., Maple, C., Yang, H., Foskett, D., & Cleaver, V. Web Application Development Lifecycle for Small Medium-Sized Enterprises (SMEs) (Short Paper). In 2008 The Eighth International Conference on Quality Software (2008, August), 247-252.

[4] Google cloud platform price calculator. Retrieved Jan 17, 2021 from https://cloud.google.com/products/calculator?skip_cache=true

[5] Casas, P., & Schatz, R. Quality of experience in cloud services: Survey and measurements. Computer Networks, 68 (2014), 149-165.

[6] Ghemawat, S., Gobioff, H., & Leung, S. T. The Google file system. In Proceedings of the nineteenth ACM symposium on Operating systems principles, (2003, October), 29-43.

[7] Tang, John C., Jed R. Brubaker, and Catherine C. Marshall. "What do you see in the cloud? Understanding the cloud-based user experience through practices." IFIP Conference on Human-Computer Interaction. Springer, Berlin, Heidelberg, 2013.

[8] Wu, Jiyi, et al. "Cloud storage as the infrastructure of cloud computing." 2010 International Conference on Intelligent Computing and Cognitive Informatics. IEEE, 2010.

[9] Zhao, Liang, Anna Liu, and Jacky Keung. "Evaluating cloud platform architecture with the care framework." 2010 Asia Pacific Software Engineering Conference. IEEE, 2010.