Cloud Application Development Continual Assessment 2021

Quick, Low Cost, Serverless Application Development in the Cloud.

James Mahoney

T00145442, TL810, MTU

Abstract

An application using a serverless architecture is an application that uses the Function as a Service model. This model has abstracted everything else away from the developer and serves functions for the developer to use on a use-by-use basis.

The objective of this report was to investigate different cloud providers and the serverless services they provide and develop and deploy an application. First AWS Amplify, AWS Cognito and AWS AppSync was investigated as possible services to develop and deploy an application with. Secondly Google Cloud Functions, Google Firebase, Google Firebase Realtime Database and Google Authentication was investigated. It was concluded to use AWS and an application was then developed using AWS services and deployed using AWS Amplify. The process was drastically simpler than the conventional development of an application that uses and API to communicate to a database and Authentication as all the processes were ready made for the developer. The report outlines the architecture used and the steps taken to develop and deploy the cloud application.

*Keywords:* Cloud services; Serverless architecture; GraphQL API; AWS Amplify; AWS AppSync;

Introduction

As part of the Cloud Application Development module for second semester of fourth year in Computing with Software Development course the class have been asked to research an area of Cloud Computing and to write a report of the chosen topic. The report must be an active report and accompanying application must be developed and delivered alongside the report. For this report I have chosen to write it on Serverless Computing i.e., Functions as a Service (FaaS).

Serverless architecture in Software Engineering has become more popular in recent years and with the rise in popularity it has paved the way for people with less programming skills and knowledge of software engineering to develop and deploy complex applications to the internet. This has been achieved by abstracting away the much of the process, serverless services don’t require the developer to provision any resources as this is all handled by the service provider. By Using a Serverless architecture a developer can develop a scalable, secure application and not have to worry about configuring any hardware.

This report will investigate available services from competing cloud service providers AWS and Google and will develop and deploy a serverless cloud application using services from one of the two providers in question.

1. Cloud Services

Cloud Computing is an alternative to traditional computing as it outsources the architecture and infrastructure to a cloud services provider. As opposed to traditional computing with cloud computing the user no longer has to host any hardware or software on site as it is all taken care of by the provider (Iosup, Yigitbasi and Epema, 2011). The types of services on offer by the provider are cloud computing, load balancing, web hosting, file storage, cloud databases, cloud API’s just to name a few. As of 2020 there were 191 cloud services on offer by Amazon Web Services and that list is ever growing. Two of the main competitors in the market offering cloud services are Amazon and Google, both offering their own versions of the same services. Mostly the only differing factor is the name. These services allow a customer to spin up as many cloud servers as they need for however long they need them allowing almost immediate scaling of infrastructure (Buyya, Broberg and Gościński, 2011).

* 1. Amazon Web Services

Amazon Web services was started in 2006 when it launched its service “Simple storage service” (S3) followed closely by Elastic Compute Cloud and EC2. It offers a wide variety of global cloud-based services. These services would stay with Amazon Web Services and become a staple of the company. Amazon offers a pay as you go model that starts with a free tier that allows a user to start most projects for free allowing startups to test out services and get off the ground before having any fees. If you are a startup using AWS as the infrastructure a lot of time and work is cut out of the architecture design and of the need for a quick scale up arises, AWS will scale up your service automatically greatly reducing the chances of down time. Using these services also lessens the chance of a security breach physically or by hackers as AWS will handle all the security also. As of February 2021, (Stalcup, 2021) AWS has 31% of the cloud services market making it the largest provider of cloud services in the world. As the initial launch of cloud computing allowed the customer not to have the need of setting up servers on site; clous service providers now offer serverless infrastructure as a service.

* 1. Serverless Architecture

Serverless architecture, also known as function as a service (FaaS) is a pattern where an application is hosted by a cloud service provider and the application is comprised of separate functions (Rajan and Arokia, 2018). These functions can scale separately to the depending on the traffic, leaving the customer to only pay for exactly what they use. Although the application is still hosted somewhere on servers, the developer will never have to worry about these servers. This type of architecture allows the customer to develop and manage the functionalities of the application without having to deal with the extra work of building and managing maintenance that is typically associated with developing applications. Serverless is not just used for application development, there is also a growing trend in deploying Neural Networks using the serverless model, which is stated in (Zu Zhucheng, Mengping and Jimmy, 2018) to scale up with zero engineering effort as all load balancing is managed by the cloud provider.

* + 1. AWS Amplify

AWS Amplify is a serverless framework for frontend developers. AWS Amplify offers the user a Git based workflow to help developers to develop scalable full stack applications. Amplify supports most front-end frameworks including Reactjs, Angular, Vue and mobile development technologies such as Flutter, IOS, Android and React Native. AWS amplify allows the user to connect an Amplify application to a GitHub repository and offers the workflow to deploy a single page application with ease. Using the Amplify command line interface a user can configure backend aspects of the project locally and push the backend to be deployed on AWS. The user can also configure the backend using the Amplify console and pull the backend onto the local machine if the user wants o avoid the command line interface meaning the user does not have to have much programming experience to build and deploy a serverless backend (AWS, 2020a).

* + 1. AWS AppSync

AppSync is an API Gateway service offered by Amazon Web Services that makes it very easy to develop GraphQL APIs. GraphQL was first developed by Facebook in 2012 and was open source in 2015. It is a query language for APIs. It offers a high availability serverless infrastructure built in. When developing an AWS Amplify application you can easily add a GraphQL API by executing the “amplify add api” command in the terminal. This will start a dialog that will end in setting up an API for your application. It will generate any mutations, queries and subscriptions meaning the user will not even write the code to query the API. If the user wants to add more functionality to the API all that must be done is add a new model to the GraphQL schema in the application and then run “amplify api gql-compile” in the terminal. This will generate all the new query code the developer will need, and this can be pushed to the backend with the command “amplify push”(AWS, 2021). Mobile apps using AppSync can share data in real time and AppSync can interact with the data on the mobile when it is offline. Once the mobile app becomes online again, the AppSync will sync any local changes with the cloud and in turn update all connected apps in real time. AppSync also allows the developer to set up subscriptions that listen in to changes in specific data for changes and will run a specified function when triggered. AWS AppSync is the main

* + 1. AWS Amplify Authentication with AWS Cognito

Cognito is a cloud service offered by Amazon Web Services that handles user registration, authentication, password and account recovery and other authentication operations (Amplify Docs, 2020). Authentication can be added to am Amplify application through the Amplify command line interface. It can be added with the command “amplify add auth”. Once configured, the developer must simply run the command “amplify push” and authentication is added to the backend and deployed. Once authentication is added to the project a component is available “withAuthenticator()”, which is a higher order component that can be added to the application. This component can be used as a wrapper for the app component wrapping the whole application in the Cognito services. Once implemented and the application is run the application will now have a sign in page and depending on the implementation a user will have to sign up using details and will have to verify sign up details via email or SMS code before signing up can be completed.

* + 1. AWS Storage

AWS offers S3 storage as Infrastructure as a service (IaaS) service. This was the first service to be released by AWS and remains the most supported cloud storage service on the market. With the support it receives it makes it easier for developers to interact with their S3 buckets from different applications(AWS, 2020b). Since S3 is an object storage service, any type of object can be stored on the S3 service giving flexibility to developers and opening many different use cases from backup and recovery storage to data archives and application storage e.g. (Images, CSV files). An S3 bucket can also be accessed from the AWS CLI and even used for personal storage files can be copied from the terminal using the command “aws cp \*\*data\*\* \*\*s3 \*\*bucket\*\*” or vice versa.

* 1. Google Cloud

Google cloud platform is a suite of cloud services offered by google. It runs on the same platform as googles own services run on such as google search and google assistant. Google released app engine in 2008, that would allow developers to deploy applications on googles infrastructure(Paul, 2018). The service was available to everyone in 2011, Google Cloud platform provides Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and serverless computing packages. Google cloud offers services such as cloud computing, storage and databases, big data and identity and security tools (Vanian, 2016).

* + 1. Google Firebase

Google Firebase was originally called Envolve when it was developed in 2011 and was changed to Firebase when it was acquired by google in 2014 (Lardinoin, 2014). Firebase is used for developing mobile/web applications. Firebase contains tools for development, maintenance, and analysis of applications. These tools are developed to help developers to build deploy and maintain applications (Johnson, 2017).

* + 1. Google Firebase Cloud Functions

Google cloud functions for Firebase is a service offered by google that serves a serverless framework for running code once an action has been triggered. A function can be triggered by events such as a user signs in, data has been added to a database or they can be triggered straight from a HTTP request (Google, 2021a).

* + 1. Google Firebase Authentication.

Google authentication for Firebase is used to authenticate a user and authenticate the actions a user can take and what a user can write, read, or execute. There is an SDK available for firebase that can be used to configure specific authentication flows or there are ready made Firebase UI components that can be added to an application and used such as the Firebase Auth component. This component handles sign in, sign out and cases such as account and password recovery (Google, 2021b).

* + 1. Google Firebase Realtime Database

Firebase Realtime Database is a NoSQL cloud hosted database that allows storage and synchronization of data between users in real time. Firebase Realtime Database can be used to build serverless applications with the available SDKs. FRD can be used in conjunction with Firebase Cloud Functions to execute functions triggered of Firebase Realtime Database events (Google, 2020). Firebase Realtime Database can be used offline to store changes to a user’s information on the local machine and once the application becomes online again these changes are synchronized with the database and all connected users. This makes Firebase Realtime database ideal for use with mobile applications.

1. Findings and Methodology
   1. Research Findings

AWS free tiers allow for a broad range of users, from large corporations to hobbyists.

AWS Amplify can be used along with AWS AppSync and GraphQL to create a serverless Application for no cost at all to the user to create. Since it is FaaS, even when the customer reaches the end of the free tier there will be no charges for server use as the customer will only be charged for computing time. So, if there are no functions being called or if the API is not getting any requests there is no charge at all.

AWS Cognito takes mist of the work out of setting up user authentication for applications and supplies the customer with premade wrapper functions that can wrap the whole application and tie directly into AWS Cognito, outsourcing security, and user/password management to AWS.

Google offer much of the same services for implementing a serverless application and Google Firebase is a competent rival to AWS Amplify.

Firebase authentication supplies components that can be used in JavaScript applications making developing authentication flows simple on these applications.

Firebase Realtime Database can be used alone to develop serverless applications such as chat applications or applications that need real time updates.

Firebase Cloud Functions can be used along with Firebase Realtime Database.

One thing to keep in mind when developing a serverless applications that due to GDPR European citizens information cannot be stored outside Europe so regions are very important when deploying the application.

* 1. Methodology

The aim of this paper was to research the technologies that a developer can use to develop a cloud application with and to develop an application quickly and at low cost.

The technologies used were ReactJS, AWS Amplify, AWS Cognito, AWS AppSync, and AWS DynamoDB. An Architecture diagram can be seen above in Figure 1.Graphical user interface, application

Description automatically generated

Figure 1 System Architecture

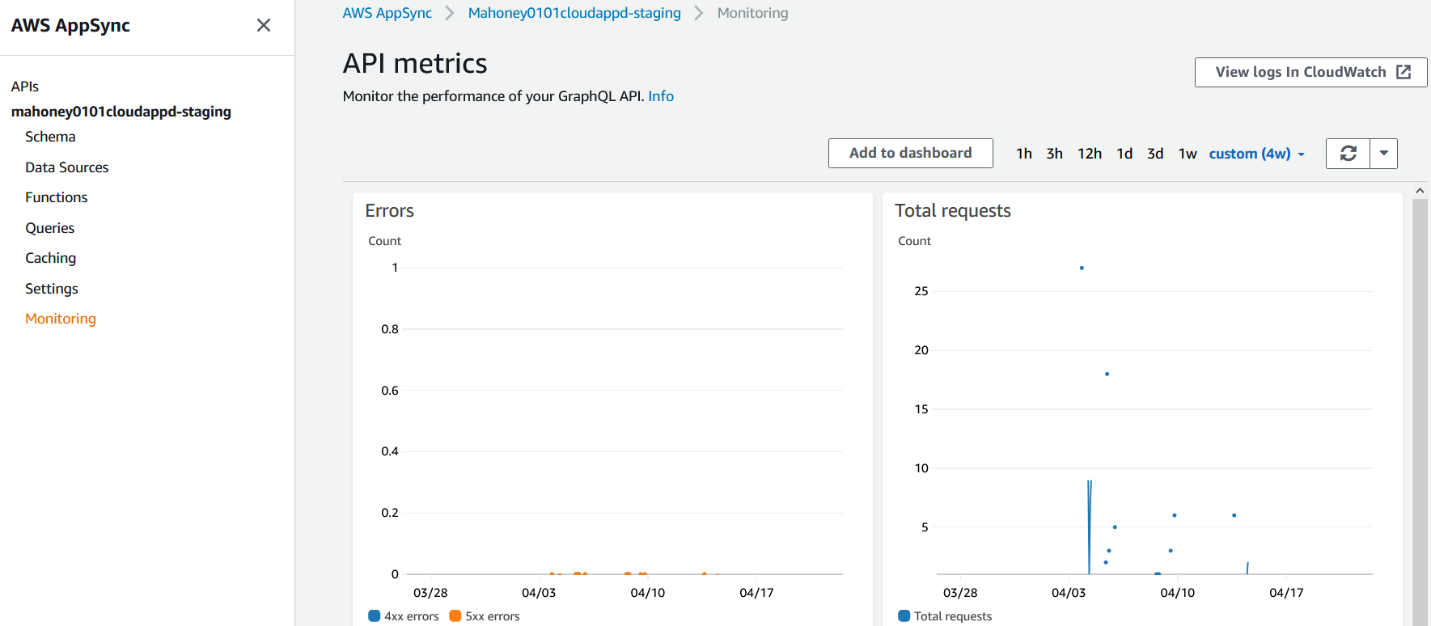
The Frontend was created using ReactJS and communication between the backend was enabled using AWS AppSync GraphQL API as an API endpoint. This was enabled easily by adding an api to the Amplify project through the Amplify CLI. The command used to create the API was “amplify add api”. The user was prompted to input the type of API and region among other settings for the API. This command also created a DynamoDB instance for the application to use as storage connected to the API. By running that once command both the API and DynamoDB table was created as can be seen in Figure 2 and Figure 3 below.

Figure 2 GraphQL API metrics

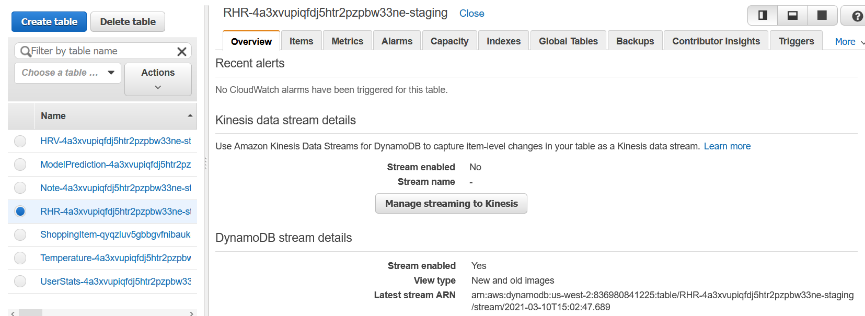
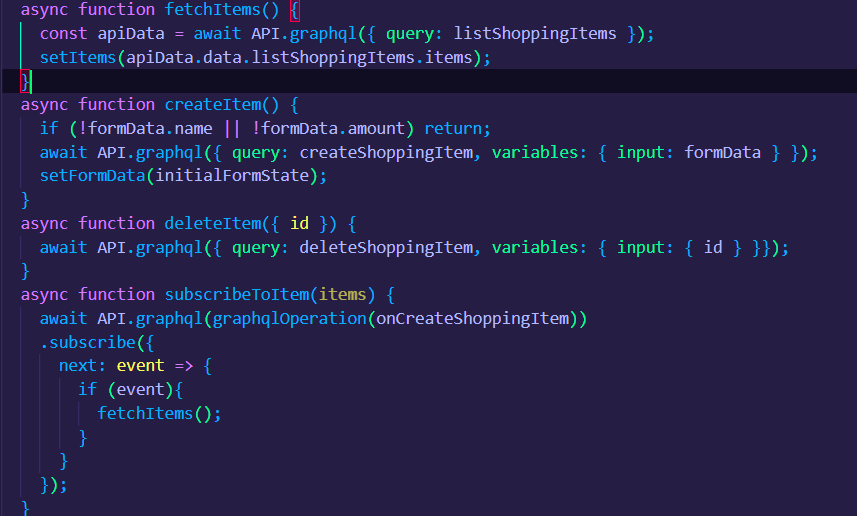
Next code could be written in the frontend to interact with the API and DB. An example of code that will interact with the API can be seen below in Figure 4. These are some CRUD functionalities along with a subscription Webhook. Which will allow for the application to sync across different devices when logged in. When a user creates or deletes something this will affect every other user logged into the application and update the application in real time.

Figure 4 CRUD and WebHook for API communication

Figure 3 DynamoDB tables overview

Then AWS Cognito was set up for authentication. Cognito was added through the CLI using the command “amplify add auth”. This command prompts the developer to add authentication and once added ready packed ReactUI components were added to the application. The component withAuthenticator and AmplifySignOut were imported and implemented. This addition added authentication with a sign in screen for the user to sign up or sign in. There is also functionality to verify email and change password.

Users were able to create an account through the implementation of Cognito authentication and were able to create records in the DB once authenticated. The backend environment can be seen below in Figure 5.

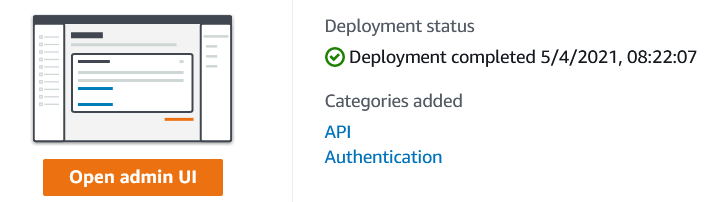


Figure 5 Amplify backend environment.

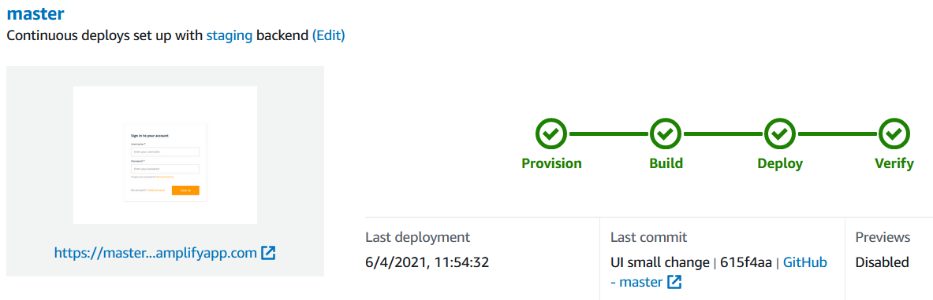
A workflow was then created by adding a WebSocket to a GitHub repository. This was done automatically by Amplify through a GUI by selecting the GitHub repository to use for the application. Every time there is a push to the selected working branch the workflow starts and runs through provisioning, building, deploying a verification. This enables CI and CD for the project. The option is there to manually edit the build file if a developer wishes to add testing or change the default build settings. The workflow for the application created for this report can be seen below in Figure 6.

Figure 6 Frontend workflow

1. Results

An application was developed and deployed using AWS serverless services. The application uses AWS Cognito and requires the user to sign up and verify their email. Once logged in, the user can post messages to a message board. The application was developed very quickly and at no cost to the developer. The application is an example of a serverless architecture. The method of creating the application proved to be quick, high quality with loose coupling and free of charges.

1. Discussion

The aim of this report was to investigate serverless cloud services and to develop and deploy an application using available serverless cloud services quickly and at low cost. The ease of developing an application and deploying it to AWS using serverless services has proven that today it is a much simpler task than it would have been before the dawn of the Functions as a Service model. The resources needed to gain the knowledge for one person to be able to complete such a task alone can be easily found by a simple internet search. The fact that one person can develop and deploy an application on the same network and architecture that large companies deploy their applications to; and do it for free, leaves the door open for innovation. A startup would only have to pay once they start to receive traffic or when the free period ends. When there is no traffic, the payments are at a minimum especially since there is no cloud server to keep running.

These results matter because in the opinion of the author, a new generation of developers are emerging that have access to all these resources means that the future of application development will be on the cloud. Even more they will be serverless models. Although cloud services make developing and deploying software much easier, a layer of control is lost. The provider is trusted with security and storing data in a secure manner. This may make life easier for the users of these services but there is a level of unknowing.

An issue that the author tackled during the development of the application was sparsity of documentation for the GraphQL API. Although the creators of the framework make it possible to generate queries, mutators and resolvers from the scheme on a simple terminal command. “amplify gql-compile”; the documentation for that specific part of the process is sparce. This makes it a difficult task to create custom processes that are not generated automatically.

1. Conclusion

The application developed for this report acts in a serverless fashion. There is no server that must be provisioned and kept running the application to exist. This cuts down on pricing and experience needed to develop the application. The application was developed quickly and deployed along with the backend for free. In the future more functionalities could be added to the application such as upload files to S3 Storage and AWS Lambda could be added to the application to process the files or serve any other process needed. This would still fit into the FaaS model.

References

Amplify (2021) *API (GraphQL) - Getting started - Amplify Docs*, *Amplify Framework Documentation*. Available at: https://docs.amplify.aws/lib/graphqlapi/getting-started/q/platform/js (Accessed: 26 March 2021).

Amplify Docs (2020) *Authentication - Getting started - Amplify Docs*, *Amplify Framework Documentation*. Available at: https://docs.amplify.aws/lib/auth/getting-started/q/platform/js (Accessed: 13 March 2021).

AWS (2020a) *AWS Amplify Console - Modern Application Development on AWS*. Available at: https://docs.aws.amazon.com/whitepapers/latest/modern-application-development-on-aws/aws-amplify-console.html (Accessed: 8 March 2021).

AWS (2020b) *Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3)*, *Amazon Web Services, Inc.* Available at: https://aws.amazon.com/s3/ (Accessed: 15 March 2021).

AWS (2021) *API (GraphQL) - Overview - Amplify Docs*, *Amplify Framework Documentation*. Available at: https://docs.amplify.aws/cli/graphql-transformer/overview (Accessed: 12 March 2021).

Buyya, R., Broberg, J. and Gościński, A., 2011. *Cloud computing*.

Google (2020) *Firebase Realtime Database*. Available at: https://firebase.google.com/docs/database (Accessed: 10 March 2021).

Google (2021a) *Cloud Functions for Firebase*. Available at: https://firebase.google.com/docs/functions (Accessed: 23 March 2021).

Google (2021b) *Firebase Authentication*, *Firebase*. Available at: https://firebase.google.com/docs/auth (Accessed: 23 March 2021).

Iosup, A., Yigitbasi, N. and Epema, D. (2011) *On the Performance Variability of Production Cloud Services*, *Proceedings - 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2011*, p. 113. doi: 10.1109/CCGrid.2011.22.

Johnson, H. (2017) *Firebase Platform Overview - DZone Mobile*, *dzone.com*. Available at: https://dzone.com/articles/firebase-platform-overview (Accessed: 23 March 2021).

Lardinoin, F. (2014) *Google Acquires Firebase To Help Developers Build Better Real-Time Apps | TechCrunch*. Available at: https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/ (Accessed: 23 March 2021).

Paul, S. (2018) *The History of Google Cloud Platform*, *A Cloud Guru*. Available at: https://acloudguru.com/blog/engineering/history-google-cloud-platform (Accessed: 23 March 2021).

Rajan, R. and Arokia, P., 2018. Serverless Architecture - A Revolution in Cloud Computing. *2018 Tenth International Conference on Advanced Computing (ICoAC)*, pp.88-93.

Stalcup, K. (2021) *AWS vs Azure vs Google Cloud Market Share 2021: What the Latest Data Shows - ParkMyCloud*. Available at: https://www.parkmycloud.com/blog/aws-vs-azure-vs-google-cloud-market-share/ (Accessed: 12 March 2021).

Vanian, J. (2016) *Google Changes Names of Enterprise Units to Google Cloud And G Suite | Fortune*. Available at: https://fortune.com/2016/09/29/ggoogle-cloud-branding-g-suite/ (Accessed: 23 March 2021).

Zu Zhucheng, T., Mengping, L. and Jimmy, L., 2018. Pay-per-request deployment of neural network models using serverless architectures. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pp.6--10.