

Serverless Computing Architecture Security and Quality Analysis for Back-end Development

Clark Jason Ngo
clarkngo@cityuniversity.edu

Peng Wang
rocwang@cityuniversity.edu

Tuan Khai Tran
t.k.tran@cityuniversity.edu

Sam Chung
chungsam@cityu.edu

Center for Information Assurance Education (CAE-CDE)
Technology Institute
City University of Seattle
521 Wall Street Suite 100, Seattle, WA 98121

Abstract - The purpose of this paper is to propose how to improve both quality and security for the back-end of a modern software system through adapting to the serverless computing architecture. In order to achieve this, this paper will conduct the following three steps: 1) Show a complete back-end architecture using three serverless computing such as Amazon Web Service (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP). 2) Analyze each component's security and quality of each serverless computing provider and compare to show similarities and differences. 3) Describe how using a cloud service improves the quality and security of a system.

Keywords

Serverless Development, Amazon Web Services, Microsoft Azure, Google Cloud Platform, Security Design, Quality Improvement

1. INTRODUCTION

The on-premise solution allows complete control of the security level. However, on-premise tends to fall behind in its security because it is rigid, time-consuming, and expensive. Moreover, it is difficult to maintain and upgrade. It is not able to quickly adapt to the ever-changing security best practices and compliance to combat cyber attacks and meet the minimum regulatory requirements.

Serverless computing is an event-driven model or code execution where infrastructure is abstracted from the developer or end-user (Deloitte, 2018). These applications using serverless computing are run in the cloud. However, these serverless applications that are managed by third-parties are also run on physical servers (Stigler, 2018). Sounds familiar? Cloud is just someone else's computer. So why use serverless computing model then if it's just another physical server? Adopting a cloud-based service can downscale bringing the costs to almost 1/10 compared to maintaining our infrastructure (AWS, 2019d). It provides ease of provisioning, operating, and maintaining. Cloud providers are compliant with most industry standard and regulatory requirements. Cloud helps companies focus more on business logic rather than spend time and resources for provisioning, maintaining, and operating resources (Ayoub, 2012).

Serverless applications provide high availability, fault tolerant, scalability, and elasticity. With all the benefits serverless architecture brings, it also has some limitations. These limitations are controlling limit on infrastructure, locked-in with a vendor and the impact of cold start. Moreover, moving the cloud would directly expose our servers to the internet (Ayoub, 2012).

In managing our cloud services, the principle of do-not-trust-anyone should be applied. This paper will show how a serverless architecture can impact both security and quality for a web application. The analysis will compare the cloud services of Amazon Web Services (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP).

2. THE ARCHITECTURE FOR WEB APPS

AWS offers various serverless services that can allow engineers to make a fully serverless back-end application. Potential architectures for web and mobile applications are shown in Figure 1 for AWS and Figure 2 for Azure. GCP also have similar services which can help to make a similar architecture. The equivalent service names can be found in Table 1.

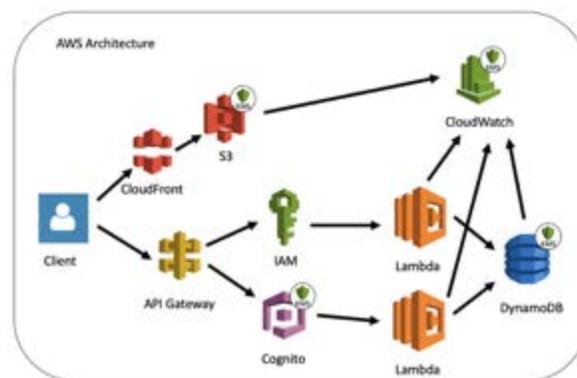


Figure 1. AWS Serverless Architecture

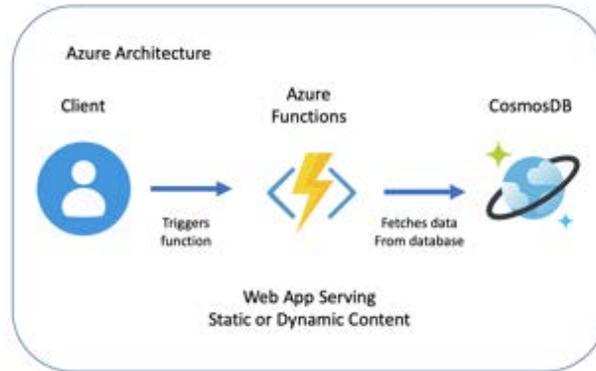


Figure 2. Azure Serverless Architecture

Table 1. A Comparison of Service Names of AWS, Azure, and GCP

AWS	GCP	Azure
S3	Cloud Storage	Azure Blob
API Gateway	Cloud VPN	VPN Gateway
IAM	Cloud IAM	Active Directory
Lambda	Cloud Functions	Azure Functions
DynamoDB	Cloud SQL	Cosmos DB
CloudWatch	Google Stackdriver	Azure Monitor

3. SECURITY AND QUALITY ANALYSIS

Each component in the architecture, as shown in Figure 1, will be analyzed from security and quality perspectives in order to see whether using serverless technologies can increase both security and quality.

3.1 Storage Service

In the AWS architecture, the Simple Storage Service (S3) is chosen for the storage service that will store all static content such as HTML files, JavaScript files, images, Lambda function codes, and so forth. By using S3, four methods can be used for conducting encryption at rest in order to protect users' data (AWS, 2019d). The four methods are: 1) Data at rest inside the volume, 2) All data moving between the volume and the instance, 3) All snapshots created from the volume, and 4) All volumes created from those snapshots. The CloudFront will help S3 to set up the Secure Sockets Layer (SSL) to apply the encryption in transit, which can protect data between clients and the cloud (AWS, 2019b). From the quality perspective, according to AWS (2019c), S3 can automatically give users at least 3,500 write with 5,500 read requests per second per prefix in a bucket. The document also mentioned that there are no limits to the number of prefixes in a bucket, which means S3 gives users ability to conduct unlimited read and write concurrent requests by adding more prefix for the same file. It is extremely hard and costly to do the same with a non-serverless solution due to the hardware or provision needs to be prepared ahead.

In the Google Cloud architecture, Cloud Storage is unified object storage to store live or archive data (Google Cloud, 2019p). Data is encrypted at rest and located in the server-side with Cloud Storage. At the storage system layer, Google uses the Advanced Encryption Standard (AES) for encrypting data at rest. At the storage device layer, data is encrypted with AES 128 for hard disk drives and AES256 for solid state drives. This encryption at the storage device layer is done with the device-level key. At backups, files are encrypted with Data Encrypted Key (DEK) (Google Cloud, 2019c). According to Google Cloud (2019a), best practices for securing a Google Cloud Storage are as follows: 1) Have distinct credentials per user, never share the credentials, and securely store them. 2) Always use Transport Layer Security (TLS) such as HTTPS to transmit data. 3) Use Hypertext Transfer Protocol Secure (HTTPS) library that validates server certificates to mitigate the risk of man-in-the-middle-attack. 4) Revoke authentication credentials if an application does not need access anymore. 5) Create a bucket name with names difficult to guess and do not add sensitive information such as our account or Application Programming Interface (API) secret keys as part of the bucket name. In terms of quality, Cloud Storage can provide 1000 writes per second and unlimited reads. Scaling is provided for both writes and reads (Google Cloud, 2019n). Availability is improved through scheduled deletion instead of immediate. This allows recovery from accidentally deletions or deletion from internal bugs or processes (Google Cloud, 2018).

In the Microsoft Azure architecture, Azure also provides the number of services for data storage including Azure Blob, Azure Files, Azure queue, and Azure tables. Azure storage provides data stores for different objects such as data file repository, archive for messaging channels, and NoSQL data store. Azure Blob is probably the equivalent to Amazon S3 as an object storage solution that can be used to store all type of unstructured data (Microsoft Azure, 2019a). With the use of Representational State Transfer (REST)-ful, Blob store manages to reduce, and Azure platform ensures that objects will be stored and available on multiple storage copies which enhances the Availability in Confidentiality, Integrity, and Availability (CIA) triad. By replicating data, it also helps to protect user's data in both planned and unplanned events (Microsoft Azure, 2019b). Microsoft Azure storage services are in general also known for its collection of security capabilities. Role-Based Access Control (RBAC) allows users to secure their storage account by restricting accesses following the least privilege principle. Data transition across networks will also be secured by encrypting with HTTPS, SMB 3.0, and client-side encryption. They also provide Azure Disk Encryption for virtual machines and data disks, Advanced Threat Protection for monitor Storage logs and check for any suspicious requests to Block Blob storage.

3.2 Authentication and Authorization Service

Three services that include API Gateway, Identity and Access Management (IAM) and Cognito are used in the AWS architecture to achieve the authentication and authorization functionality. A user's request will reach the endpoint that offers by API Gateway and then passes to either IAM for providing accesses with their infrastructure or Cognito for providing accesses to outside users. Cognito supports both encryptions at rest and in transit when it is Health Insurance Portability and Accountability Act (HIPAA) eligible and Payment Card Industry Data Security Standard (PCI DSS), Service Organization Control (SOC), ISO/EIC 27001, International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) 27017, ISO/IEC 27018, and ISO 9001 compliant (AWS, 2019d). From the quality perspective, Cognito offers the capabilities of fully managing user lifecycle and automatically verifying identities from

API Gateway without customized implementation required out of the box. It can improve the system quality a great deal through avoiding to use an incorrect or weak authentication implementation. API Gateway also offers the cache capability without requiring to change any code plus the ability to control the API usage for each of our clients for free.

GCP has Cloud IAM to allow administrators to have a platform to manage and control user and group access for cloud services (Google Cloud, 2019e). IAM enforces policies to member identities and roles. Roles are derived from the collection of permissions. Figure 2 shows GCP's IAM. It adopts the principle of least privilege. IAM configurations are on a per-project basis. A project must exist first before adding users and roles. Cloud Endpoints manages our API's development, deployment, and monitoring (Google Cloud, 2019d). Cloud Endpoints works in conjunction with Cloud IAM to grant and revoke API access. Google Cloud is compliant with ISO 27001, ISO 27017, ISO 27018, SOC 1/2/3, PCI DSS and Cloud Security Assurance (CSA) Security Trust Assurance and Risk (STAR) (Google Cloud, 2019p).

In GCP, the system is improved with encryption of inter-service communication, which provides automatic encryption for infrastructure Remote Procedure Call (RPC). End-user data access is provided with automatic mutual encryption, see Figure 3 (Google Cloud, 2018). Insider risk is reduced through two-party approval implementation for certain actions and API debugging without exposure to sensitive information (Google Cloud, 2018).

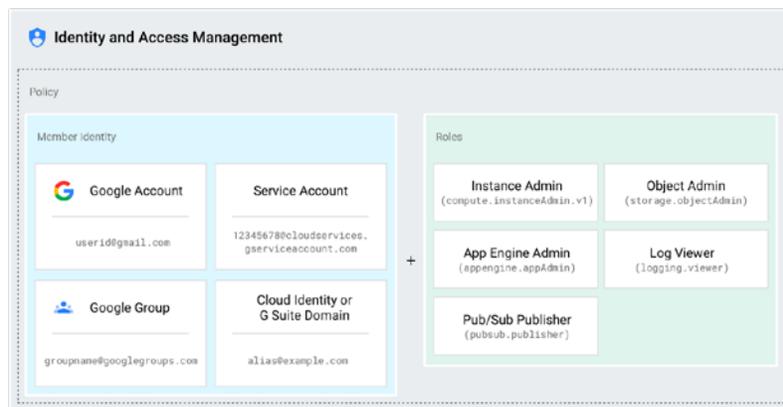


Figure 2. GCP Identity and Access Management (Google Cloud, 2019p)

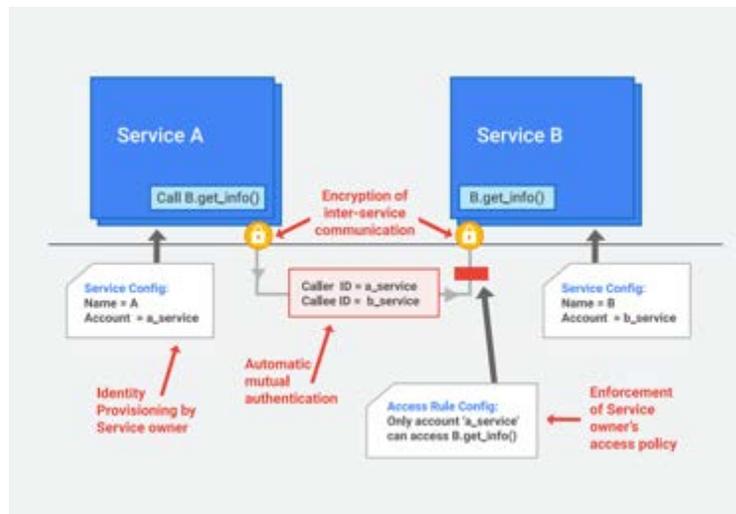


Figure 3. GCP Service Identity, Mutual Authentication, and Encrypted Inter-Service (Google Cloud, 2018).

Azure Active Directory is a Microsoft Azure solution for IAM. The service is used to manage access of employees, partners, customers, and corporate assets. Users can synchronize their on-premise local infrastructure with Windows Azure Active Directory to provide single sign-on for users to access cloud applications (Microsoft Corporation, 2018). Azure Active Directory also brings multi-authentication to the table as an additional defense to help a business protect sensitive information and applications. Some of the best practices that need to be considered when working with Azure Active Directory include enabling single sign-on, enforcing multi-authentication, applying role-based access control, centralizing identity management, and being aware of locations where resources are placed (Microsoft Azure, 2018g).

3.3 Computing Service

The Lambda service is chosen for AWS architecture's computing component. Lambda itself does not help to improve the security directly since it is hiding behind API Gateway. However, the security level can be raised indirectly because the Lambda service will handle the OS level and runtime level patching automatically, which will reduce the risk of having vulnerabilities significantly. Additionally, it also helps in raising the quality of the system a great deal. By using Lambda, users can automatically scale in and out based on the real-time traffic without even thinking about the provision. It gives end users a much better user experience.

Cloud Function is a computing service provided by GCP that lets us run code in the cloud. It shines best when used for microservices as it adds agility by having small independent functionalities running instead that one huge service (Google Cloud, 2019h). As of February 27, 2019, the quota for function calls is at 40,000,000 in all regions. Security is still in the hands of the developer. The developer should follow best practices such as passing a JSON Web Token (JWT) to our functions, attaching a role with least privileges, hard to guess function name, and logging and monitoring using Stackdriver to enhance security (Google Cloud, 2019g).

Similar to the Lambda and Cloud Function, Azure Function is also a service that allows users to run small pieces code or function on the cloud to solve a specific problem without bothering other components related to the application as well as the infrastructure to run this function. Azure Function was built based on a trigger and binding mechanisms. Triggers are used to make the functions run, which will rely on a specific event to activate our function. Binding is a way to declare used services to simplify the interaction with input and output data (Microsoft Azure, 2017f). To optimize the performance, Azure Function can be integrated with some other Azure services such as Cosmos DB, Azure Event Hubs, Azure Storage Blob. These services can be used to execute our functions through some provided templates including CosmosDBTrigger, BlobTrigger, QueueTrigger, and EventHubTrigger.

3.4 Database Service

Several serverless database options can be chosen for the AWS architecture. DynamoDB is one of them that can deliver single-digit millisecond performance at any scale (AWS, 2019e). The official document (AWS, 2019e) also mentioned that “*DynamoDB can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second.*” It makes DynamoDB is an excellent candidate for any mission-critical workload. DynamoDB encrypts all customers’ data at rest by default, which is an excellent security feature that users can get for free (AWS, 2019f). According to the official document (AWS, 2019f), using DynamoDB also can help to improve the system quality through different features it has. First of all, it has an auto-scaling and on-demand mode that allows users to scale up and down based on real-time traffic ultimately. Secondly, DynamoDB can be used with DynamoDB Accelerator to improve the read performance by up to 10 times. Moreover, the DynamoDB Streaming feature could help users to take advantage of real-time data processing. Additionally, the build in point-in-time recovery feature gives users the ability to restore a table to any point of time up to the second during the previous 35 days.

GCP offers various database services. Cloud SQL gives us high performance, scalable, and manageable PostgreSQL and MySQL databases in the cloud (Google Cloud, 2019f). Cloud SQL has two levels of access controls. First, instance-level access authorizes access from an applicant, client, or other GCP services to our Cloud SQL instance. Second, database access uses MySQL Access Privilege System to manage MySQL users in our Cloud SQL instance (Google Cloud, 2019k). Cloud SQL access and permissions can also be configured using Cloud IAM (Google Cloud, 2019m). High availability is an option to a Cloud SQL project that works by creating a replica of an instance through semi-asynchronous replication in different zones (Google Cloud, 2019l).

Some of the more popular products when it comes to database services for Microsoft Azure are SQL Database and Cosmos DB. Azure Cosmos DB is known for its support on the multi-model and the ability to globally distribute for any scale. It is capable of replicating all the data in the database to a global scope with more than 30 Microsoft Azure data center located throughout the world. With the ability to replicate such global scope, application connected to Cosmos DB will have very low latency with 99% data processing, which will be guaranteed at less than 10ms for both read and write (Microsoft Azure, 2018d). In term of security, Azure Cosmos DB uses master keys and resource tokens to perform authentication and provide appropriate access to its data and resources. It also adds to the database another layer of protection in case of crashes based on the "failover" mechanism. When a problem such as power outage on all the

datacenter or a national, multinational scale incident, Cosmos DB can automatically handle the situation (Microsoft Azure, 2018d). Cosmos DB is certified for various standards such as CSA START and ISO 20000-1:2011(Microsoft Azure 2018e). All the data stored is also well encrypted.

3.5 Logging and Monitor Service

The CloudWatch is a service that is integrated with almost all of other AWS service to offer logging and motoring capability. It is also used for this AWS serverless architecture. The CloudWatch logs are PCI and FedRamp compliant and also offers encrypted at rest as well as encrypted in transit as its standard security features (AWS, 2019g). The system quality also can be raised by using CloudWatch. According to the official document (AWS, 2019g), the CloudWatch offers an easy way to collect and store logs, collect built-in metrics, push custom metrics, and create dashboards. Most importantly, the CloudWatch allows users to create alarms based on some specific metrics, which can be used to trigger other AWS services. It gives users a great ability to automate tasks.

Google Stackdriver is a logging and monitoring tool for GCP, AWS, on-premise, or hybrid. Stackdriver features are debugger, error reporting, rapid discovery, uptime monitoring, integrations, smart defaults, alerts, tracing, logging, dashboards, and profiling. It helps enable observability, working with multiple cloud services, create fast fixes, and gather full-stack insights (Google Cloud, 2019o).

Azure Monitor is the selection for logging and monitoring. There are two categories that data collected by the service fall into which are metrics and logs. Metrics refer to a numerical value which interprets some data of a system. In contrast, logs contain records that organized into a different set of properties. Azure Monitor also has responding mechanism to critical situations. The respondent can be an alert or sending emails to administrators who are responsible for solving the issue, or it can launch an automated procedure and attempt to fix the issue (Microsoft Azure, 2019h). Another feature that Azure Monitor can provide is to visualize monitoring data. Collected Data and elements can be added into their dashboard, which allows combining both metrics and logs into charts and diagrams. Thus, it can leverage other Azure services for publishing to different audiences (Microsoft Azure 2019h).

4. SUMMARY OF SECURITY FOR CLOUD SERVICES PROVIDERS

AWS, GCP, and Azure all support encrypting data at rest and in transit that uses AES-256 encryption, which is the industry standard and a secure block cipher. IAM for all three cloud providers addresses the required access to security and resource control. However, the implementation of authorization for AWS, GCP, and Azure differs. In AWS, we can immediately create users, groups, policies, and roles. In GCP, we need to create a project or choose an existing project before we can assign users, roles, etc. In Azure, it uses the subscription scope that has an owner, contributor, or reader which grants different levels of access to resource groups depending on the role. Resource group would also have an owner, contributor, or reader. The concept of resource groups is perfect for isolating resources, shown in Figure 4 (Power, 2018).

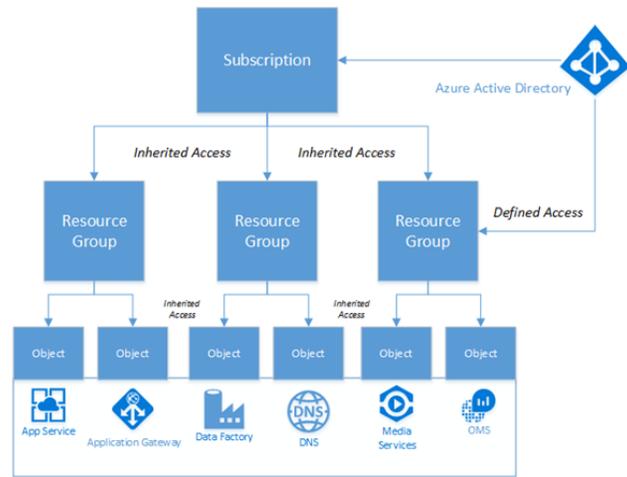


Figure 4. Azure Active Directory Subscription and Resource Group (Sandbu, 2018)

5. FINDINGS

5.1 In Terms of Improving Security

Serverless architecture improves all aspects of security such as authentication, authorization, encryption, confidentiality, integrity, and availability. Most providers comes with built-in monitoring and logging (Deloitte, 2018). It delivers ease of control, management that requires less technical knowledge to implement better security. Moreover, cloud providers are having better security standards as they specialize in the business of securing resources and applications. It is important to note that while cloud vendors improves cloud security such as network or infrastructure security, cloud users must ensure the security of their APIs. See Table 2 for API access types.

Table 2. API access types

API access type	Explanation	Example
Public APIs	Public content	Your homepage index.html
Internal APIs	APIs only called by other your functions	Function to call the data from the database
Authenticated APIs	APIs only usable for users	Unregistered users are not allowed to call API to mitigate DDoS attacks from API call abuse
APIs for third parties	APIs shareable to others	Integration for other third-party applications

5.2 In terms of Improving Quality

Improving quality improves security. Serverless architecture improves quality by enforcing a consistent configuration standard for instantiating resources. Tools for monitoring, logging and reviewing are readily available to vulnerability check. Less code to maintain means less technical debt and fewer vulnerabilities from the code base. Availability is vastly improved through auto-scaling, durability, and redundancy.

On the downside, serverless architecture have the following considerations that are detrimental to quality if not managed properly. These considerations are ephemeral functions, cold start, limited database configuration, and no system level access. See Table 3 for architectural considerations for serverless computing (Deloitte, 2018).

Table 3. Architectural Considerations for Serverless Computing (Deloitte, 2018)

Considerations	Explanation	Workaround
Ephemeral functions	Functions are deployed in a container that persist for a certain period of time only	Don't use for large processing requirements
Cold start	Invoking function for the first time after it's was inactive increases execution time	Keep the function hot (stay alive) before execution
Limited DB configuration	Limit to simultaneous connection in relational DBs	Use NoSQL
No system level access	No support for reading attributes from configuration files or spilling over to in-memory cache to disk	Don't use for applications needing file system level or operating system level access

6. CONCLUSION

Serverless architecture drastically reduces security vulnerabilities from legacy code and resource configurations. AWS, GCP, and Azure all currently provide the minimum regulatory requirement and best practices. Choosing a cloud service provider would depend on the use case and extra security features on a particular service. A cloud provider might provide more insightful logging and review over the other or provide better availability that does not immediately shut down our resources after hitting the maximum instance or call made to a service.

To leverage the use of serverless architecture and improve overall security and quality of the application, it is important to follow industry guidelines. See Table 4 for industry guidelines on serverless architecture implementation.

Table 4. Serverless Architecture Industry Guiding Principles (Deloitte, 2018)

Principles	Explanation
Develop simple purpose functions	Use single-purpose codes for easier deploy and test
Design push-based, event-driven patterns	Allow event chain to propagate without the need for user input
Create thicker and powerful frontends	Reduce function calls by executing more complex frond-end through rich client-side application framework
Incorporate appropriate security mechanism across the technology stack	Use of API Gateway layer and other mechanisms such as access controls, authentication, IAM, encryption, etc.

Identify performance bottleneck	Measure performance of bottlenecks and identity functions slowing a service
Leverage third party services	Enterprise tools may not be compatible. Assess and choose the right third party for monitoring, logging, etc.

REFERENCES

- Adzic, G., & Chatley, R. (2017). Serverless Computing: Economic and Architectural Impact. Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 884-889
- AWS. (2019d). Protecting Data Using Server-Side Encryption. Retrieved from <https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html>
- AWS. (2019b). Amazon CloudFront custom SSL. Retrieved from <https://aws.amazon.com/cloudfront/custom-ssl-domains>
- AWS. (2019c). Request Rate and Performance Guidelines. Retrieved from <https://docs.aws.amazon.com/AmazonS3/latest/dev/request-rate-perf-considerations.html>
- AWS. (2019d). AWS Compliance Programs. Retrieved from <https://aws.amazon.com/compliance/programs>
- AWS. (2019e). Amazon DynamoDB. Retrieved from <https://aws.amazon.com/dynamodb>
- AWS. (2019f). Amazon DynamoDB Features. Retrieved from <https://aws.amazon.com/dynamodb/features>
- AWS. (2019g). Amazon CloudWatch Features. Retrieved from <https://aws.amazon.com/cloudwatch/features>
- Ayoub, D., & Wacker, R. (2012). Security: On-premise or in the cloud? Network World, 29(20), 18-19. Retrieved from <http://proxy.cityu.edu/login?url=https://search-proquest-com.proxy.cityu.edu/docview/1265769745?accountid=1230>
- Chen, L. (2018). Microservices: Architecting for Continuous Delivery and DevOps. IEEE International Conference on Software Architecture.
- Deloitte. (2018). Google Cloud Security Retrieved from <https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media-telecommunications/Serverless%20Computing.pdf>
- Google Cloud. (2019). Serverless Computing – Architectural Considerations & Principles. Retrieved from https://services.google.com/fh/files/misc/security_whitepapers_march2018.pdf
- Google Cloud. (2019a). Best Practices for Google Cloud Storage. Retrieved from <https://cloud.google.com/storage/docs/best-practices>
- Google Cloud. (2019b). Data Encryption Options. Retrieved from <https://cloud.google.com/storage/docs/encryption/>
- Google Cloud. (2019c). Encryption at Rest in Google Cloud Platform. Retrieved from <https://cloud.google.com/security/encryption-at-rest/default-encryption/>
- Google Cloud. (2019d). Cloud Endpoints. Retrieved from <https://cloud.google.com/endpoints/>
- Google Cloud. (2019e). Cloud Identity & Access Management. Retrieved from <https://cloud.google.com/iam/>
- Google Cloud. (2019f). Cloud SQL. Retrieved from <https://cloud.google.com/sql/>
- Google Cloud. (2019g). Function Identity. Retrieved from <https://cloud.google.com/functions/docs/securing/function-identity>
- Google Cloud. (2019h). Google Cloud Functions. Retrieved from <https://cloud.google.com/functions/>

- Google Cloud. (2019i). Google Cloud Storage. Retrieved from <https://cloud.google.com/storage/>
- Google Cloud. (2019j). Instance Access Control. Retrieved from <https://cloud.google.com/sql/docs/mysql/instance-access-control>
- Google Cloud. (2019k). Overview of High Availability Configuration. Retrieved from <https://cloud.google.com/sql/docs/mysql/high-availability>
- Google Cloud. (2019l). Project Access Control. Retrieved from <https://cloud.google.com/sql/docs/mysql/project-access-control>
- Google Cloud. (2019m). Project Access Control. Retrieved from <https://cloud.google.com/storage/quotas>
- Google Cloud. (2019n). Stackdriver. Retrieved from <https://cloud.google.com/stackdriver/>
- Google Cloud. (2019o). Trust & Security. Retrieved from <https://cloud.google.com/security/>
- Hong, S., Srivastava, A., Shambrook, W., & Dumitras, T. (2019). Go Serverless: Securing Cloud via Serverless Design Patterns. Retrieved from <https://www.usenix.org/system/files/conference/hotcloud18/hotcloud18-paper-hong.pdf>
- Jiménez, V., & Zeiner, H. (2018). Serverless cloud computing: a comparison between "function as a service" platforms. CS & IT-CSCP, pp. 15–22. DOI: 10.5121/csit.2018.80702
- Microsoft Azure. (2019a). Blob storage. Retrieved from <https://azure.microsoft.com/en-us/services/storage/blobs/>
- Microsoft Azure. (2019b). Storage redundancy. Retrieved from <https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy>
- Microsoft Azure. (2019c). Azure Storage security overview. Retrieved from <https://docs.microsoft.com/en-us/azure/security/security-storage-overview>
- Microsoft Azure. (2018d). Welcome to Microsoft Azure Cosmos DB. Retrieved from: <https://docs.microsoft.com/en-us/azure/cosmos-db/introduction>
- Microsoft Azure. (2017f). An Introduction of Azure Function. Retrieved from: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview>
- Microsoft Azure. 2018g. Azure Identity Management and access control security best practices. Retrieved from: <https://docs.microsoft.com/en-us/azure/security/azure-security-identity-management-best-practices>
- Microsoft Azure. 2018h. Azure Monitor overview. Retrieved from: <https://docs.microsoft.com/en-us/azure/azure-monitor/overview>
- Microsoft Corporation. 2018. Azure Functions and Serverless platform security. Retrieved from: <https://azure.microsoft.com/mediahandler/files/resourcefiles/azure-functions-serverless-platform-security/Microsoft%20Serverless%20Platform.pdf>
- Power, Vince. (2018). IAM Roundup: AWS vs. Azure vs. GCP. Retrieved from <https://www.twistlock.com/2018/09/25/iam-roundup-aws-vs-azure-vs-gcp/>
- Sandbu, M. (2018). Microsoft Azure and Security Best Practices – Part 1 Identity. Retrieved from <https://msandbu.org/microsoft-azure-and-security-best-practices-part-1-identity/>