Disney Theme Parks Cloud Migration Strategy

By: Pravin Raja

**Table of Contents**

[**Introduction**](#_eon5w6ihor0w) **3**

[**On-Premise Assumptions**](#_ki39mcglb5nm) **4**

[**Cloud Assessment and Benefits of Cloud**](#_3sa31wknhmoq) **6**

[**Proposed Migration Strategy**](#_fspr4jt7qtzj) **9**

[**Conclusion**](#_sqc2nbw0onmv) **17**

[**References**](#_bx43jb1p10au) **18**

# **Introduction**

This research project will design, describe, and architect a potential long-term cloud migration strategy for The Walt Disney Company using Amazon Web Services as our cloud platform. This company is currently ranked 49th on the Fortune 500 List. Disney has already chosen and migrated to AWS as their cloud partner a few years back, but this project will mainly focus on the theme parks side opposed to other divisions. I will focus this project for both theme parks: DisneyLand in Anaheim, CA and DisneyWorld in Orlando, FL.

As of now, Disney’s revenue and profits have been down slightly this year due to Covid lockdowns. Despite this, cost management won’t be much of a problem as they are still worth billions of dollars and will come back strong after life comes back to normal and also because of their strong backing and ownerships in the media. Finally, DisneyLand will be expected to fully reopen in April of 2021. So, because of all this we can liberally spend to get the best cloud infrastructure, security, and customer experience possible.

This cloud migration project will focus on four important parts. The first part will explain our assumptions about current on-premises systems that could be used at Disney Theme Parks. The second part will go into high detail about the cloud assessment methods for migration and the benefits of moving to the cloud. The third part will show our proposed infrastructure, application, and data migration strategy along the specific AWS services we would use as part of the migration and why they were selected and how they map to Disney’s current on-premises systems through detailed architecture diagrams. Finally, the fourth part will talk about levagarging and the optimization of our plan for the future along with the conclusion.

**NOTE: ALL Diagrams were created using** [**Lucidchart**](https://www.lucidchart.com/)

# **On-Premise Assumptions**

First, we can currently assume Disney Theme Parks currently use traditional on premise EMC SAN/NAS storage servers for archiving older movies and videos for showtime and also the storage of terabytes of security camera footage. All this video footage could be migrated to AWS S3 (real-time data) and Glacier (archival data) to improve the speed, quality, security, and cost of storage. Next, we could also assume they are using relational database servers like Oracle, SQL Server, and Transact-SQL to manage their customer data, sales information, keeping inventory, and managing check-in and check-outs. Similarly, we can migrate this data to AWS RDS to modernize and improve the database management system. AWS can also take care of the backups, high availability, fault tolerance, and disaster recovery instead of the traditional database administrator activities. Disney would also most likely be storing at least 20 years of historical video data into a data warehouse system like Teradata / Vertica. This could be moved to AWS EMR, AWS Redshift (Platform as a Service), Snowflake (Software as a Service). Similarly, accessing lots of structured data can be done from AWS S3 using Athena or Presto. Finally, any on-premise NoSQL database like MongoDB or Cassandra could be replaced with DynamoDB.

Second, we can also assume that for connecting all the computer networks throughout the park they will most likely use a mix of traditional Cisco Routers, Switches, and Hubs within a server room to handle things like backroom production, public address systems, and customer support. This could be migrated to AWS Virtual Private Compute (VPC) with public and private subnets with VPC peering included to enable us to route traffic between 2 VPC’s (one for development and another for production use). VPC can also handle the Inbound rules which help control the incoming traffic to our instance, and outbound rules which control the outgoing traffic from our instance. This pretty much replaces the traditional network equipment, firewall, and internet gateway. Finally, we could add a demilitarized zone (DMZ). Lets say, Disney has public and private network connections. For the public, we could have websites or portal logins for workers off site. In private they have their p2p sharing, virtual machines, secret plans for new ideas, etc. If all of these are on the same network the public ones must be accessible so it can't be in the security of your WAN or LAN, it must be able to be seen by the public, so they put the publicly accessible ones in the DMZ.

Third, lots of legacy and software tools can be migrated onto AWS through Platform as a Service. We would just provide the application and data we want to be on the AWS platform. This is useful both for development and for deploying custom built applications. Assumptions of this could include Java, J2EE, Struts, Hibernate, video and sound editing software (Pixar Presto, Maya, Adobe Premiere & Audition), Tableau visualization dashboards, and ETL (Extract, Transform ,and Load) software tools like Informatica. Furthermore, we can migrate Informatica powercenter to Informatica IICS (Platform as a Service), Tableau to AWS Quick Insight and Elasticsearch service, and Java J2EE to a full stack (React.JS/Node.JS) Docker based container like ECS (Elastic Container Service).

Fourth, we could also move the Unix and Windows based operating systems to the cloud into EC2 instances with a mix of public, private, and hybrid.

Fifth, middleware based applications, namely web, messaging, and monitoring services can also be migrated to AWS ECS based full stack applications like Node.JS and React.JS, and AWS SNS, SQS, and Cloudwatch.

Finally we will discuss security needs, in Disney Theme Parks. We can assume they use an active directory system, LDAP (Lightweight Directory Access Protocol) with Kerberos for authentication. This could be better managed using Single Sign On with AWS Cognito or cloud based Okta. In on-premise Disney would SSL based certificates for HTTPS protocols, but through AWS, it is already HTTPS based by default. On-premise encryption based data trust is managed using AWS KMS (Key Management Service). Disney also needs to keep up their audit, complicanes, and governance up to to date such as 508 compliance for disabilities. This can be moved to AWS Audit Manager and CloudTrail. For physical security, tools like AWS Fraud Detector can manage credit card fraud and impersonations and AWS Rekognition can be implemented for user verification, people counting, contract tracing, and public safety use during entry and exit or even inside the park. Finally, AWS Web Application can help protect Disney against common web exploits and bots that may affect availability through their app or website.

# **Cloud Assessment and Benefits of Cloud**

Although, the cloud has many benefits for organizations. The organization should still look at its resources and IT environment and determine if it is capable of a cloud migration. formulating a checklist in agreement on how to do so can reduce the time and money it takes to migrate. This is known as a cloud assessment. Some key points we have planned out are checking out the current software licenses, tools for migration, the overall cost, technical assessments, and security compliances. After going through the cloud assessment process, we will give a high-level overview explaining the benefits of migrating to the cloud.

First, AWS License Manager can make it easier for Disney to manage their software licenses from software vendors (Microsoft, Teradata, Oracle, and Informatica) centrally across AWS and on-premises environments. This provides the control and visibility of usage of your licenses. This helps limit licensing overages and reduce any risk of non-compliance and misreporting. You can save also costs by using the bring-your-own-license (BYOL) concept, this means you can repurpose your existing license inventory for use with your cloud resources.

Second, we need to look at the tools we need to migrate to the cloud. One method we could use is AWS lift and shift . This is moving an application from on premises to cloud without making any changes in the tech stack or application itself. For example, Disney Resorts is running an application for hotel reservations from their own premises and they are planning to migrate to AWS cloud. They can just lift and shift, very minimal or zero code changes in the application. Just get the EC2 instance and replicate the application, and install DB,MQ. Alternatively, doing this may be less work, but it won’t leverage and take advantage of the cloud solutions as much as possible. So, going cloud native could be more beneficial for optimal efficiency and better performance at scale. This means you would rewrite your application to use resources offered by AWS. For example with AWS, using SQS, API gateway, Lambda, DynamoDB, etc.

Other good methods include AWS Application Discovery Service, this service gathers and collects data about on-premise servers. Ex. configurations, usage, and performance, which give users a thorough, accurate overview of their present set up. This data can then be fed into AWS Migration Hub, or used to calculate Total Cost of Ownership of migrating to and running these servers on AWS. The AWS Application Discovery Service is free to use: only pay for the AWS resources. Next, The Migration Acceleration Program (MAP) can use an evaluation of processes to work out an organization’s capabilities and create a plan that will build on these capabilities to effectively perform and support a cloud migration. Thirdly we could use AWS Migration Hub, this is like a deluxe dashboard for tracking the progress of your migration to the cloud. Through this one dashboard, users can audit their existing servers, plan out their project, and follow the status of each migration. Not to mention, AWS Server Migration Service, AWS Database Migration Service, CloudEndure Migration, ATADATA ATAmotion, and RiverMeadow Server Migration SaaS are all natively integrated with AWS Migration Hub, and can feed status updates to Migration Hub automatically.

Third, we need to focus on the costs. Yes, cloud computing can seem really cool and fun on paper, but it all comes down to a cost. From the research gathered we could use AWS cost explorer and use predictive analytics to predict the costs over a period of time using time-series or cost estimator to get a rough estimate based on current needs or even use Total Cost Ownership through AWS Migration Hub to get a clean and clear estimate. For a company like Disney, AWS cloud can be cheaper in the longer term because the organization doesn't have oncall network technicians to make sure load balancers and servers are constantly monitored for hardware, power or other faults and also spinning up a bunch of AWS resources is far cheaper because you can do it with a smaller staff compared to on-premise. The other consideration is that cloud services can easily switch regions such as between Orlando, FL and Anaheim, CA. So yes, cloud is worth it for Disney Theme Parks as it is one of their most profitable divisions pulling an all time high of $20.3 billion for the fiscal year of 2018. Although, this division has taken a bit of a hit due to Covid-19 in 2020, It can still be guaranteed that for the future once life comes back to normal, Disney Theme Parks will continue to be a money maker for The Walt Disney Company. As mentioned earlier, DisneyWorld and Disneyland should be fully operational by Q3 of 2021. To summarize, Disney can spend liberally on cloud services to greatly improve their customer experiences, infrastructure, and security throughout the park, but at the same time not spend too much as other needs throughout the business should also be carefully considered to increase and maintain the stock price and profitability.

The fourth point would be considering the focus on technical assessments. Moving to the cloud could be a daunting task for many employees that don’t have previous experience working in cloud based environments. Before doing a full fledged cloud migration it would be wise to conduct a company survey asking for what needs to be done, skills gaps, and any aptitude based questions on AWS cloud. After gathering and studying results, a training session and curriculum schedule can be created through help of educational programs like Coursera or Udemy to help smoothen the transition for employees during the cloud migration time. Both sites offer large course offerings and subject matter experts in the field of cloud computing.

The last point would focus on security compliances. The first step would be to ensure you have your compliance/legal teams address the impact of the compliance program and other security-risk management changes that using the cloud will introduce. This is absolutely vital as there are many compliances to manage and for a company like Disney Theme Parks. Things of this nature could range from disabilities, building zone codes, following federal regulations, accounting and tax, encryption policy standards, and even following cloud security guidelines from issuing bodies like ISO, FedRamp, PCI, and SOC. The best solution would be to clarify needs based on architecture diagrams and compliance controls while addressing the compliance-related challenges of migration to the cloud as time keeps progressing. Failure to do this could result in lawsuits, lost money, poor data security, and an overall failed cloud migration plan.

Moving forward, part 2 of this section will give a high-level overview of why cloud is a better choice than on-premise for Disney Theme Parks in ten simple points. First as mentioned previously, AWS resources are far cheaper because you can do it with a smaller staff compared to on-premise and the organization doesn't have oncall network technicians to make servers are constantly monitored, and we can also work with multiple availability zones for both the east and west coast simultaneously to make work much easier and collaboration. Second, AWS cloud also offers cheaper and simpler backup options (Glacier), which can be useful for archival of old movie clips during performances and showtime for Disney Theme Parks. Third, cloud is infinitely scalable, dynamic, and the costs are op-ex instead of cap-ex, so If you're expecting lots of growth, or moves, or are trying to run lean, cloud is a great option. Fourth, AWS cloud makes more sense if you need to dynamically scale up and down depending on the customer traffic load through auto scaling. Fifth, upfront capital investment and initial management is much lower and easier through AWS as it would take time to create an entire infrastructure that would have taken months and months of hard work to put in place through physical labor while with AWS it can be done in a few clicks. Sixth, AWS allows you to pay for what you use. If your product/application is not always being used by your customers you are paying for wasted capacity. Seventh, cloud promises business continuity whether you're experiencing a natural disaster, power outage, or other crisis, having your data stored in the cloud ensures that it's backed up and protected in a safe place. Eighth, cloud can help Disney have a competitive edge by helping access the latest applications any time without spending precious time and money on installations. Ninth, would be processing multi-tenancy at scale, this means a single instance of the software and its supporting infrastructure can serve multiple customers. Each Disney customer shares the software application and also shares a single database. Each tenant's data is isolated and remains invisible to other tenants. The multi-tenant application will benefit Disney as it has an automated signup process, we can easily add new customers to the database and we also don’t need to change the codes or the data structure because the shared codes are basically common. In the end, multi-tenancy can help save costs and be more efficient and secure. Final point would be cloud security. AWS has a requirement to be more secure than any single business, as one failure can destroy them, We can see them provide services for 3-letter agencies, multiple governments, and all kinds of businesses. To add on, AWS provides services that help you protect company data, accounts, and workloads from any kind of unauthorized access. AWS data protection also offers services that provide encryption and key management and threat detection which continuously monitor and protect company accounts and workloads. In the end, it is important to know cloud security is the process to protect stored data from theft, leakage and deletion. It works in order to comply with administrative laws and to prevent unauthorized user access.

# **Proposed Migration Strategy**

After understanding why Disney would benefit from moving to AWS cloud. This section will cover our proposed cloud migration strategy in detail. Our plan will maintain the importance of having a good cloud infrastructure setup, security, and improving overall customer experience.

Our plan will completely be re-architecting everything becuase we are looking to move from the legacy codebase and architecture of Oracle based applications and want to take advantage of the AWS cloud proprietary features. Considering we have a monolithic architecture and are looking to move to a fully serverless architecture, we would have to refactor its entire code and possibly change its previous development frameworks.

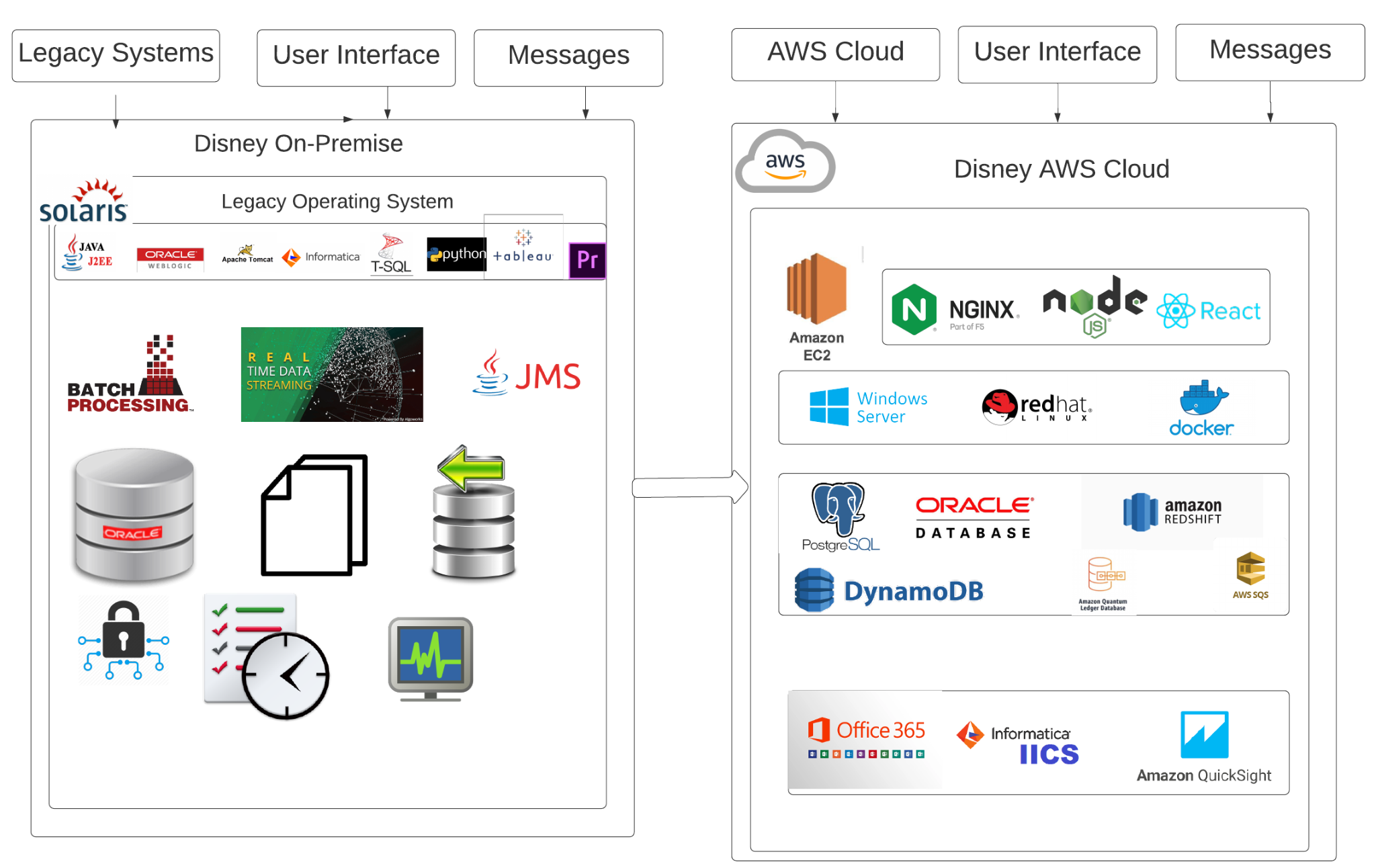
The table below shows the potential on-premise software and hardware tools that could be moved onto the cloud.

|  |  |  |
| --- | --- | --- |
| # | On-premise | AWS Cloud |
| 1 | VM/OS   * Windows Server * Unix Server | EC2 Instance  -Windows Server  - Red Hat Linux |
| 2 | Network   * Cisco Router * Switches, Hubs, DMZ * Security Firewall | VPC (Virtual Private Compute)   * Security Group * NAT, Internet Gateway * Router * Elastic Network Interface * DMZ |
| 3 | Middleware   * Web Server * Application Server | Full Stack   * Nginx Reverse Proxy * Node.JS * React |
| 4 | Software / Tools   * Java /J2EE/ Struts/ Hibernate/ JDBC * Python * Informatica ETL * Tableau / Powerbi * Video Editing * Sound Editing | Software & Tools   * Javascript * React Native * JDBC/ODBC * Python & Jupyter Notebook * Informatica IICS * AWS Quicksight * AWS ElasticSearch * AWS Element and Media |
| 5 | Database   * Oracle * DB2/Unix * SQL Server | Database   * RDS Oracle * RDS Postgre * RDS SQL Server |
| 6 | Data Warehouse & NoSQL   * Teradata / Netezza * Cassandra, Mongodb | Data Warehouse & NoSQL   * Redshift * DynamoDB |
| 7 | Storage   * EMC Storage (NAS and SAN) | * EBS * S3, Glacier * EFS |
| 8 | Load Balancer and Fault Tolerance | * ALB & NLB * Autoscaling * AWS Backup Services |
| 9 | Security   * Burp Suite * Wireshark & Firewall * Security Camera | * IAM * Cognito * WAF * Cloudtrail * Audit Manager * Rekognition |

Using this table, we can map and visualize out a physical and logical diagram. The logical diagram will show high-level AWS services needed to migrate Disney on-premise to cloud for the optimal experience. The Physical Architecture diagram shows the detailed implementation of on-premise to AWS cloud for Disney Theme Parks. The physical diagram consists of six main parts, the VPC Network, Security, Infrastructure, Video Process, Managed Blockchain and Customer Experience.

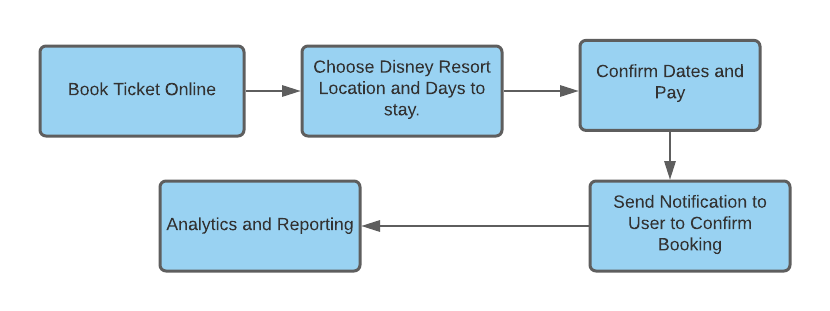
Both diagrams will be based on the three tier software architecture, which consists of the Front-end web, middleware, and backend database. The three tier software architecture organizes our application into three logical and physical computing tiers: the front-end: user interface, the application/middleware tier: where data is processed, and the database tier: where the data associated with the application is stored and managed. The benefit of this architecture is that each tier runs on its own infrastructure meaning that each tier can be developed simultaneously by a development team, and can also be updated or scaled as needed without impacting the other tiers.

**Figure 1: Logical Architecture Diagram**

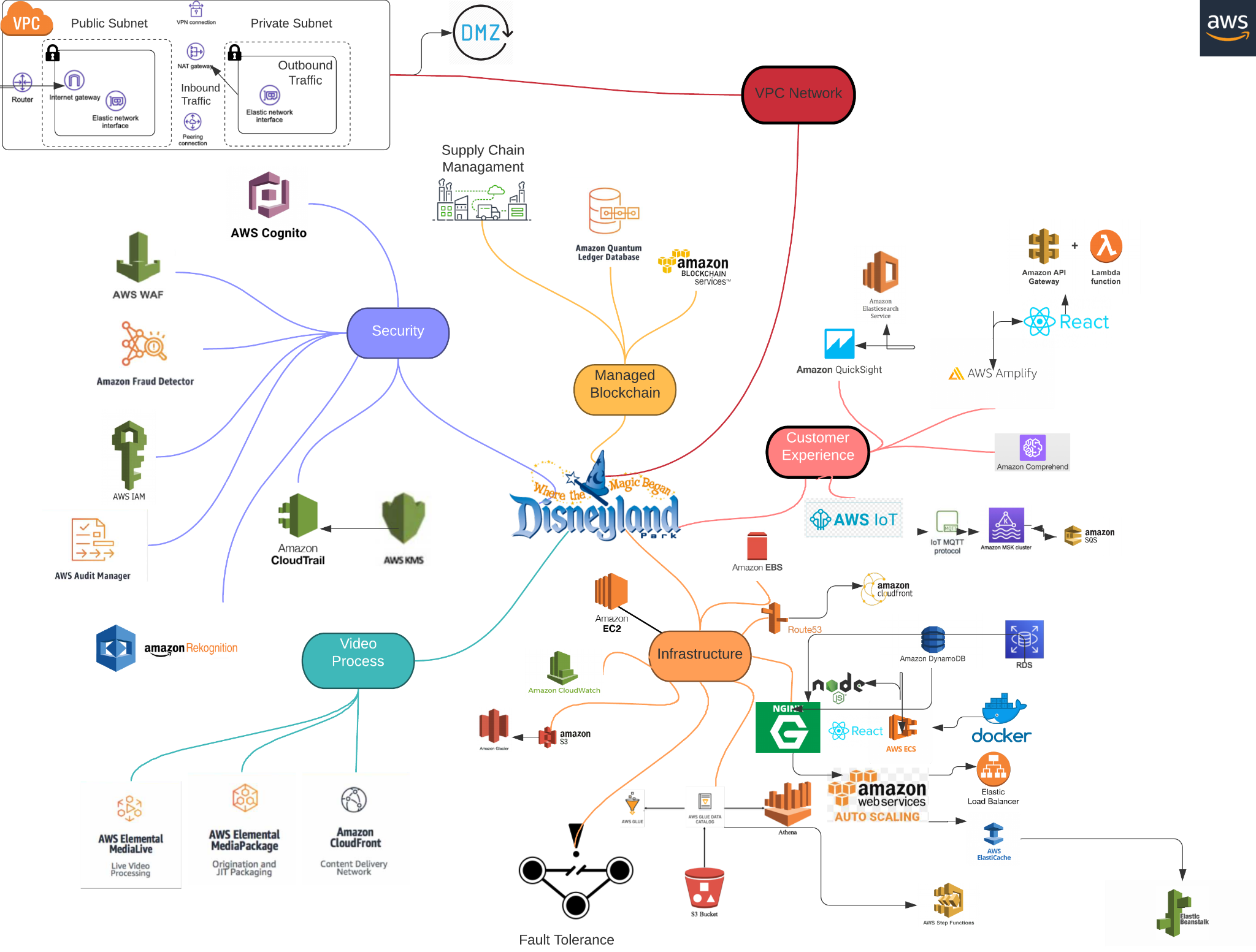


In Disney Theme Parks, the IT department used Unix Sun Solaris, Legacy Struts based applications, Oracle Database, Apache Tomcat, Oracle Web Logic, Tableau, Adobe Premiere, Python for on-premise datacenter. As per this project, we need to move to Infrastructure as a Service (IAAS) namely AWS EC2 instance with Red Hat Linux We will also use a Docker based Nginx with React and Node.JS. Next we need to move legacy databases to Platform as a Service (PAAS) such as RDS PostgreSQL, RDS Oracle, Redshift, etc. Fourth, we will use legacy Microsoft Office installation to Office 365, similarly Java Messaging Service will be replaced with Amazon Simple Queuing Service (SQS). Finally Tableau will be converted to AWS Quicksight and Informatica to Informatica (IICS) cloud based solution.

**Figure 2: Disney Theme Parks Online Booking Business Flow**



Disney Theme Parks has a global presence and a growing customer base. To provide the best experience to its customers, the company decided to move its business operations on AWS cloud that would help to scale the server infrastructure in no time without accumulating a heavy cost. Also, it will provide a scalable infrastructure based on business requirements with continuous server availability with higher security and data durability for peak time especially summer time in Anaheim, CA and Orlando, FL.

**Figure 3: Physical Architecture Diagram**

Above is the brief summary of the AWS services used in the physical architecture diagram. This shows the detailed architecture and implementation of all AWS services. This can also be visualized like a mind map. To explain, a mind map is a diagram used to visually organize our information. It is based on a hierarchical structure and shows relationships among pieces of the whole.

Below are the AWS services used to migrate on-premise to cloud:

**VPC Network:**

The VPC network is implemented to show how the company network is set up and how the network resources are shared.

* **Public and Private Subnet**: The public subnet sends the **outbound traffic** directly to the Internet. The instances in the private subnet access the Internet by using a network address translation **(NAT) gateway** that resides in the public subnet.
* **Inbound** means incoming **traffic** coming to the **EC2** instances.
* **Elastic Network Interface**: represents a virtual network card. It will be the primary private IPv4 address with a fixed Mac Address.
* An **internet gateway** is a virtual **router** we add to to enable direct connectivity to the internet.
* **VPN** allows Disney to connect to another computer over the internet using a secured tunnel.VPC **peering** connection is the connection between two VPCs that enables us to route traffic between them using private IPv4 addresses or IPv6 addresses.
* **DMZ** the isolation between the public and private internet traffic, it was explained in greater detail in the assumptions section stated above.

**Security:**

We deployed various and extensive security measures to ensure security across the physical, virtual, and legal aspects of the park.

* **AWS Cognito:** provides authentication, authorization, and user management for web and mobile apps through login credentials through social media or traditional username and password.
* **AWS Web Application Firewall (WAF):**  This is programmable within the infrastructure. It makes it easy to set policies using FirewallManager to ensure it’s deployed on all resources with rules you define. This is huge in our multi-account environment and also helps prevent web exploit attacks like bot attacks.
* **AWS Identity Access Management:** IAM permissions are for people inside your organization who need to do stuff in AWS - your developers for example. IAM permissions are also for AWS resources that need access to other AWS resources.
* **AWS Fraud Detector:** Can manage credit card fraud and impersonations throughout the system by using machine learning automation to build, deploy, and manage fraud detection models with a very low false-positive rate.
* **AWS Rekognition** Can be implemented for user verification, people counting, contract tracing, and public safety use during entry and exit or even inside the park by identifying the objects and scenes in images that are specific to Disney business needs or somewhat sketchy. This can improve physical safety in the park. This is done through Deep Learning and Image Processing.
* **AWS CloudTrail:** Can manage the governance, compliance, operational auditing, and risk auditing side of things through log analysis and making sure everything is done right. **AWS KMS** is integrated with **AWS CloudTrail** to provide us with logs of all key usage to help meet regulatory and compliance needs.
* **AWS Audit Manager:** Can continuously audit AWS usage to simplify assessing risk and compliance with regulations and industry standards

**Infrastructure:**

* **EC2 Instance**: We plan to use multiple M4 Large and some M4 Extra Large **EC2** RedHat and Windows Server as our servers and operating systems.
* **EBS**: We are using SSD based **Elastic Block Storage** as an efficient storage method opposed to NAS or SAN based storage solutions.
* **S3:**  It provides us 99.999999999% durability for objects stored in the services and supports multiple security and compliance certifications. It is an affordable storage solution.
* **S3 + Glacier:** This will be used to create a lifecycle to store one year of storage into S3 and archive 2-5 years of data in glacier.
* **Route53 + Cloudfront**: **Route53:** Will be used to create and manage the DNS (Domain Name Service). We can speed up delivery of our web content by distributing traffic in our **Route53** by using **Cloudfront** for different regions.
* **Docker + ECS**: **Docker** is a container management tool. It packages applications and libraries into container images. It makes it easy to develop applications and ship them in containers for deployment anywhere. Amazon Elastic Container Service (ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage **Docker** containers on a cluster.
* **React.JS + Node.JS:** Both Javascript based services. **React.JS** acts as a front-end user interface and **Node.JS** is the back-end for API calls and database connectivity for retrieving data.
* **Nginx:** Will be our Web Server. It is lightweight and can be an extra layer of security as it can act as a reverse proxy. **Nginx** also offers load balancing needs.
* **Beanstalk:** Automation for deploying, scaling, and orchestrating our load balancing. Instead of manually having to configure instances, auto scaling, RDS, load balancing, etc. **Beanstalk** does it all for us.
* **RDS + DynamoDB:** Semi structured JSON based data will be stored and managed with **DynamoDB** and **RDS (Relational Database Server)** is for the relational databases namely Oracle and SQL server
* **S3 + Athena + Glue + Step Functions:** We will store structured data into **S3** and access the S3 as a database and table format using **Athena.** For semi-structured data we need to use a classifier to tag the attributes and crawl the XML and JSON files into structured database and table format. **AWS Glue** jobs can Extract, Transform, and Load the data into a different database management system. We use **Step Functions** to automatically trigger and track each step, and retry when there are errors, so our system executes in order and as expected.
* **Elasticache:** In-memory data store and cache service. Helps improve the performance of web applications by retrieving information from managed in-memory caches instead of relying entirely on slower disk-based databases. **ElastiCache** for Redis with cluster mode can be enabled to enhance reliability & availability with little change to an existing workload.
* **Elastic Load Balancer: ELB:** Distributes the requests to whichever server has the lowest load. One IP to rule them all and can elastically scale up and down without anyone knowing. This evenly distributes the amount of connections from multiple clients to multiple servers, especially non-busy servers. This is very useful when you have a bunch of people who want to use a service at once especially during entry and exit time during peak hours and also multiple purchase orders going on at the same time.
* **Auto Scaling:** Scales up or scales down the computing resources automatically according to the current needs of our software and our park infrastructure in real-time. Using this, the cloud computing capacity can be increased or decreased according to the current needs required. This helps Disney boost customer experience without the need of increasing computational resource consumption.
* **Fault Tolerance:** Give us the ability for our to remain in operation even if some of the components shown used to build our system fail without any downtime.
* **CloudWatch:** This is our metrics repository. This helps us with monitoring and log management for data insights of current performance. Can also be used to set alarms and send SNS notifications based on a certain requirement such as deploying the correct resources at the right times.

**Video Process:**

AWS Media Services can be used to transport, prepare, process, and deliver live and on-demand content in the cloud. These managed services let us build and adapt video workflows quickly, eliminate capacity planning, easily scale with growth. They are useful for live performance and according to need for any scenario. Ex. AWS Mediaconvert can be used for Multiscreen delivery across the park

AWS Media Services makes it fast and easy to produce, process, and deliver broadcast and over-the-top video for Disney showtime. These pay-as-you-go services and appliance products offer the video infrastructure you need to deliver great viewing experiences to any screen while giving the ability to transform video codec and improve video quality for all video footage for improving entertainment for all customers.

**Managed Blockchain:**

* **Point of Sale Through AWS Blockchain Services**: Disney Restaurants and gift shops can also improve, secure, and speed up their point of sale process for non credit card or cash users. They can use a web-based point-of-sale system that lets you process payments through the Internet. Without the need of clunky and expensive servers, or pricy software that makes you buy a new version to upgrade.
* **Supply Chain Management**: Disney can use an Amazon Managed Blockchain network as a transparent and efficient way to keep track of ownership of goods such as food or souvenirs that move across global supply chain networks. They can also connect with distributors easier to manage warehouse inventory . Finally this technology can check for defects and poisoning throughout the supply chain sources ranging from the seed to the farmer to even the factory of where the defect came from. This can stop further defects from worsening.
* **QLDB**: This is a NoSQL database that comes alongside Blockchain that helps provide an immutable, transparent, and cryptographically verifiable transaction log ‎owned by a central authority so all customer payments information can be secure and safe with histories saved.

**Customer Experience:**

* **AWS Comprehend:** Can get keywords and recognize relationships with other documents EX. Customer reviews, blogs, sentiments, entities, and even personal survey data. So this will make it easier to tie documents together and look to improve on issues from extracted elements. Can also be used for language conversion to work against language barriers for non-english customers. This is Natural Language Processing.
* **React.JS/Amplify:** React.JS is used for client-side UI code to provide functionality like login/signup, search and confirmation pages, also to send search requests, Search responses to and from AWS Lambda. Amplify UI Component is used for AWS Cognito connectivity for authentication and authorisation. This is mainly used for the Disneyland Mobile app experience side. EX. **Play Disney Parks app** was originally used to play music and games in queues, but has become far more interactive.
* **Lambda:**  This is used for serverless computing platforms to implement business logic such as search and reserve vacation rental. Also can be used to trigger data transfer automatically from DynamoDB to Glacier using Kinesis Firehose for data archival purposes.
* **Amazon API Gateway:** API Gateway service is used to create RESTful APIs which can enable real-time two-way communication between Client-side code and AWS Lambda functions.
* **IOT:** Also known as the **Internet of Things (IOT)** . Disney MagicBands use long-range devices inside, which send out signals to beacons all over Disney World's parks to make the guest experiences personalised. Disney also has sensors all over the parks, which stream real-time data about what guests are doing and when they're doing it. This part can be managed through various message queue protocols like **Message Queuing Telemetry Transport** (**MQTT), Managed Kafka Service (MKS), Simple Queuing Service (SQS)**. Disney uses IOT (“Magic Bands”) to simplify experiences like food deliveries, payment methods, avoiding paperwork based registration, photopass access, and the usage of fast pass for long lines and also adding personal entertainment for various interests like Star Wars or Epcot. It’s like a one-stop shop.
* **AWS ElasticSearch Service + Quicksight: AWS ElasticSearch** is an enterprise search Cloud tool, Disney can store a copy of all survey, customer, and various data in **Elasticsearch** that they want to be able to search for. This can be up to terabytes of data and **Elasticsearch** can search, archive, log, and visualize all results with ease and also Sort, Search, Rank search results according to criteria. Using these indexed logs from **ElasticSearch** Disney can build an interactive dashboard in the Cloud using **AWS Quicksight** to visualize things like customer data and completing market research to improve customer experience and sales by making informed decisions. This would be Customer Relationship Management (CRM) and Business Intelligence.

# **Conclusion**

As per the current technology trend, lots of companies are migrating from on-premise to cloud. Doing this can save cost, improve performance, and help have a good user experience. Furthermore, we should hire a cloud architect to provide and evaluate the 6R’s of cloud migration patterns and implement good practices. The patterns are Remove, Retain, Replatform, Rehost, Repurchase, and Refactor. In our project, we used, recommended, and focused on the refactor pattern also known as re-architecting.

From here the architect should consistently monitor the cloudwatch to deploy the correct resources and recommend pay as you go instances, reserved, or even spot instances depending on the scenario. Pay as you go is billed on a per second basis and you can start or stop the service at any time, for Reserved Instances (RI), we are just reserving the specific instance size, if any VM matches that size in the environment then the RI is applied to that VM, we don't have to do anything to the VM itself. If in case we delete this VM and deploy another VM of the same size, the RI will still get matched to the new VM. Finally, spot instances are an unused EC2 instance which is available for less than the On-Demand price. Spot instances are up to 90% cheaper than On-Demand instances, which can significantly reduce your EC2 costs. A Spot Price is the hourly rate for a Spot instance.

Next after all this, the architect should look into the cost explorer and reduce the cost, they must recommend to run the non-production AWS instances during business hours from 8AM to 8PM Monday to Friday. During peak periods the architect must take use of load balancers, auto scaling groups, and fault tolerance to maintain high performance and have the best end user experience.

To finally end, Disney Theme Parks can see a huge return of investment migrating to AWS cloud as research and development shows if presented with a good strategy within 2-3 years of given time.

# **References**

Marr, Bernard. “Disney: Using Big Data, the IoT and Machine Learning to Enhance the Customer Experience.” *Bernard Marr*, www.bernardmarr.com/default.asp?contentID=1278.

Coffey, Kelly, et al. “Disney Parks Revenue Drops Over 50%, $119 Million Lost.” *Inside the Magic*, Inside the Magic, 11 Feb. 2021, insidethemagic.net/2021/02/disney-parks-revenue-decrease-kc1/.

Sylt, Christian. “Inside Disney's Internet Of Things.” *Forbes*, Forbes Magazine, 11 Jan. 2021, www.forbes.com/sites/csylt/2020/06/06/inside-disneys-internet-of-things/?sh=706c6275459c.

Johnston, Matthew. “How Disney Makes Money: Linear Networks Generate the Most Income.” *Investopedia*, Investopedia, 15 Feb. 2021, www.investopedia.com/how-disney-makes-money-4799164.

“Choosing the Right Cloud Migration Strategy in 2020 (The 6 Rs): Cloud Services.” *d3v Technology Solutions*, D3V Tech, 25 June 2020, www.d3vtech.com/insights/choosing-the-right-cloud-migration-strategy-in-2020-the-6-rs.

“Understanding the Importance of Cloud Compliance.” *RingStor*, RingStor Team, 11 Sept. 2019, ringstor.com/blog/understanding-the-importance-of-cloud-compliance/.

Divya, M. S., & Goyal, S. K. (2013). ElasticSearch: An advanced and quick search technique to handle voluminous data. *Compusoft*, *2*(6), 171.

Nadipalli, R. (2017). *Effective Business Intelligence with QuickSight*. Packt Publishing Ltd.

Todorov, D., & Ozkan, Y. (2013). AWS security best practices. *Amazon Web Services [Online]. Available from: http://media. amazonwebservices. com/AWS\_Security\_Best\_Practices. p df*.

Tavis, M., & Fitzsimons, P. (2012). Web application hosting in the AWS cloud. *Site Point*.

Varia, Jinesh. "Best practices in architecting cloud applications in the AWS cloud." *Cloud Computing: Principles and Paradigms* 18 (2011): 459-490.

Tiwari, Animesh, and Megha Jain. "Analysis of supply chain management in cloud computing." *International Journal of Innovative Technology and Exploring Engineering* 3.5 (2013): 152-155.

Steckel, Joel H., Sunil Gupta, and Anirvan Banerji. "Supply chain decision making: Will shorter cycle times and shared point-of-sale information necessarily help?." *Management Science* 50.4 (2004): 458-464.

Wheeldon, Johannes, and Jacqueline Faubert. "Framing experience: Concept maps, mind maps, and data collection in qualitative research." *International journal of qualitative methods* 8.3 (2009): 68-83.

Sudhakar, Kalyan. "Amazon Web Services (AWS) GLUE." *International Journal of Management, IT and Engineering* 8.9 (2018): 108-122

Vithayathil, Joseph. "Will cloud computing make the Information Technology (IT) department obsolete?." *Information Systems Journal* 28.4 (2018): 634-649.

Fernandez, Eduardo B., et al. "The secure three-tier architecture pattern." *2008 International Conference on Complex, Intelligent and Software Intensive Systems*. IEEE, 2008.

Khedekar, Vinay, and Yun Tian. "Multi-Tenant Big Data Analytics on AWS Cloud Platform." *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020.

Villamizar, Mario, et al. "Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures." *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2016.

Vijayakumar, Thurupathan. "API Gateways." *Practical API Architecture and Development with Azure and AWS*. Apress, Berkeley, CA, 2018. 51-96

Sbarski, Peter, and Sam Kroonenburg. *Serverless architectures on Aws: with examples using Aws Lambda*. Shelter Island: Manning Publications Company, 2017.

Nadarajah, Paneer Mala, and Nazatul Aini Abd Majid. "Mobile App Development Using Cloud-Based Platforms." *Advanced Science Letters* 24.2 (2018): 1080-1084.

Mukherjee, Sourav. "Benefits of AWS in Modern Cloud." *Available at SSRN 3415956* (2019).

Barr, Jeff, Attila Narin, and Jinesh Varia. "Building fault-tolerant applications on AWS." *Amazon Web Services* (2011): 1-15.

Němeček, J., and L. Vaňková. "CRM and cloud computing." *Proceeding in AICT’11 Proceedings of the 2nd International Conference on Applied Informatics and Computing Theory*. 2011.

Nathan, Senthil, et al. "Blockchain meets database: Design and implementation of a blockchain relational database." *arXiv preprint arXiv:1903.01919* (2019).

Abu-Libdeh, Hussam, Lonnie Princehouse, and Hakim Weatherspoon. "RACS: a case for cloud storage diversity." *Proceedings of the 1st ACM symposium on Cloud computing*. 2010.