CS301 IT Solution Architecture G1

Project Proposal

Team Information

Github Team Name : project-2021-22t2-g1-project-2021-22t2-g1-team6

Team Members :

Lixiang Wu (01404879), Tian Mingze (01368170), Hartono Tjakrawinata, Jonathan Chow (01339300), Soh Yu Wei (01423316), Sia Yan Rui (01351538), Zhang HongXiang (01366948)

Background and Business Needs

A Hotel Search and Booking System will be developed as requested by Ascenda Loyalty and used to power whitelabelled hotel booking platforms on behalf of banks, airlines and loyalty programs. The key functional requirements that will be needed are: **1) Destination Search, 2) Hotel search results**, and **3) Booking data**.

The quality requirements needed are: **1) Speed**, **2) Scalability**, **3) Ease of Maintenance**, **4) Data security**, and **5) Resilience and Disaster recovery**.

Stakeholders

Stakeholder	Stakeholder Description	Permissions (if not applicable, write N.A.)
Ascenda Loyalty	Client	All resources (read + write)
Hotel Suppliers	Provides pricing data for hotels	N.A.
Users	Customers	N.A.
Frontend Devs	Develop frontend services and an intuitive UI/UX	S3 (read + write) Amplify (read + write) WAF (read + write) Route53 (read + write) API Gateway (read + write) CloudWatch (read + write)
Backend Devs	Develop backend services and integrate API into application	Lambda (read + write) API Gateway (read + write) S3 (read + write) Aurora (read + write) ElasticCache (read + write) CloudWatch (read + write)

Use Cases for Architectural Significant Requirements (ASRs)

Use Case Title - Enter destination search	
Use Case ID	UC-001
Description	This is a text-based auto-complete search for hotels in the destination. The autocomplete suggestions have to be fast for customers to help them in constructing their search query.
Actors	Customers, Hotel API, Cache
Main Flow of events	1. Customer enters the query destination in the search bar. 2. Relevant auto-completed suggestions are listed as the search query changes.
Alternative Flow of events	No suitable suggestions for customer's query
Pre-conditions	Customers have to know what destination(s) they want to search
Post-conditions	Customers can find and select the destination that they're looking for in the suggestions

Use Case Title - View and select hotel search results	
Use Case ID	UC-002
Description	For a given destination, dates of stay and number of rooms/guests, a list of matching hotels and the cheapest room for each hotel. The result list is populated as soon as responses are returned from the Hotel API, so customers do not have to wait for the slower responses to load.
Actors	Customers, Hotel API, Cache
Main Flow of events	1. Customer clicks on the search button 2. A list of hotels from the customer's searched destination will be displayed, in order of the cheapest room first. 3. More hotels will continue to be loaded as the responses are returned from the Hotel API 4. Customer clicks "View Details" on their desired hotel to be redirected to the room booking screen.
Alternative Flow of events	No hotels available for the chosen destination
Pre-conditions	Hotel pricing and static data APIs are available
Post-conditions	Customers are able to find a hotel that suits their needs

Use Case Title - Reserve hotel rooms	
Use Case ID	UC-003
Description	A customer should be able to book their chosen hotel room

Actors	Customer, Hotel API, Booking Lambda
Main Flow of events	1. Customer clicks “Reserve” button for their chosen hotel room 2. The hotel room is reserved temporarily for the customer for an hour until the payment process has been completed. 3. Room booking status is set to “Reserved”.
Alternative Flow of events	If hotel room has already been reserved by another user, customer will be prompted and redirected back to the hotel rooms page
Pre-conditions	The chosen hotel room has to be available and unreserved for the customer’s chosen booking period.
Post-conditions	1. Customers have reserved the room successfully 2. Room status has been changed to reserved for the customer’s reserved time period

Use Case Title - Confirm booking and make payment	
Use Case ID	UC-004
Description	A customer will proceed to pay for the reserved hotel room.
Actors	Customers, Payment Gateway, Booking Lambda, Payment Lambda
Main Flow of events	1. Customer enters their payment details and clicks ‘Book Now’ 2. A booking confirmation page will be shown if payment is successful 3. Room booking status will be changed to “Paid”
Alternative Flow of events	1. Payment is unsuccessful and customers will be prompted to try again. 2. Hotel room will be unreserved after 1 hour if payment has not been completed
Pre-conditions	Customer is able to reserve the room without any reservation conflicts with other customers
Post-conditions	Payment has been made successfully and customers have received their booking confirmation

Proposed Budgets*

*refer to Appendix for table of detailed calculations

Development Budget

Activities/AWS Services Used	Total Cost
Development of solution, WAF, Route 53, CloudWatch, Lambda, Aurora, API Gateway, ElastiCache, S3, Amplify	$62.25 USD + 420 man-hours for a 6-man team

Production Budget

AWS Services Used	Total Cost
Lambda, Aurora, API Gateway, ElastiCache, S3, Amplify, WAF, Route 53, CloudWatch	$369.02 USD per month

Quality Attributes

1. Speed

AWS Amplify leverages the Amazon CloudFront Global Edge Network to distribute the web app to users globally with low-latency and high speed transfer of data. Amazon Elasticache and Amazon Aurora creates a combination of top-tier speed of an in-memory cache with the reliability and flexibility of a relational database.

2. Scalability

AWS Lambda would also support scalability as it automatically scales when the number of events increases, Lambda will route those events to available instances and create new instances if needed. When the number of events decreases, Lambda will stop unused instances to free up scaling capacity for other functions. It is also able to do so without the need for configuration or deployment due to its serverless nature.

Other AWS services such as AWS S3 offer a pay-as-you-use model which allows us to start small and slowly scale up in the future without compromise on performance or reliability.

3. Ease of Maintenance

AWS API Gateway allows for easy maintenance of the many APIs used within our application as it is a fully managed service by AWS, allowing the concurrent handling and processing of many API calls. A CI/CD pipeline (consisting of AWS CloudFormation and GitHub Actions) will be implemented to automate the deployment of application changes to reduce manual errors as well as time spent on deployment and testing.

4. Security

AWS Amplify, AWS Web Application Firewall (WAF), and Amazon Route 53 work seamlessly together to create a flexible, layered security perimeter against multiple types of attacks. AWS WAF allows us to create rules to block web exploits like SQL injection and cross site scripting.

All of these services co-reside at the AWS edge and provide a scalable, reliable, and high-performance security perimeter for applications and content. All data would be encrypted at rest in Amazon Aurora using industry standard AES-256 encryption algorithm and keys that are managed under AWS Key Management Service.

AWS IAM would allow fine-grained access control over the different services and resources used by different stakeholders, and access can be analyzed across our entire AWS environment to further enforce the principle of least privilege.

5. Resilience & Data Recovery

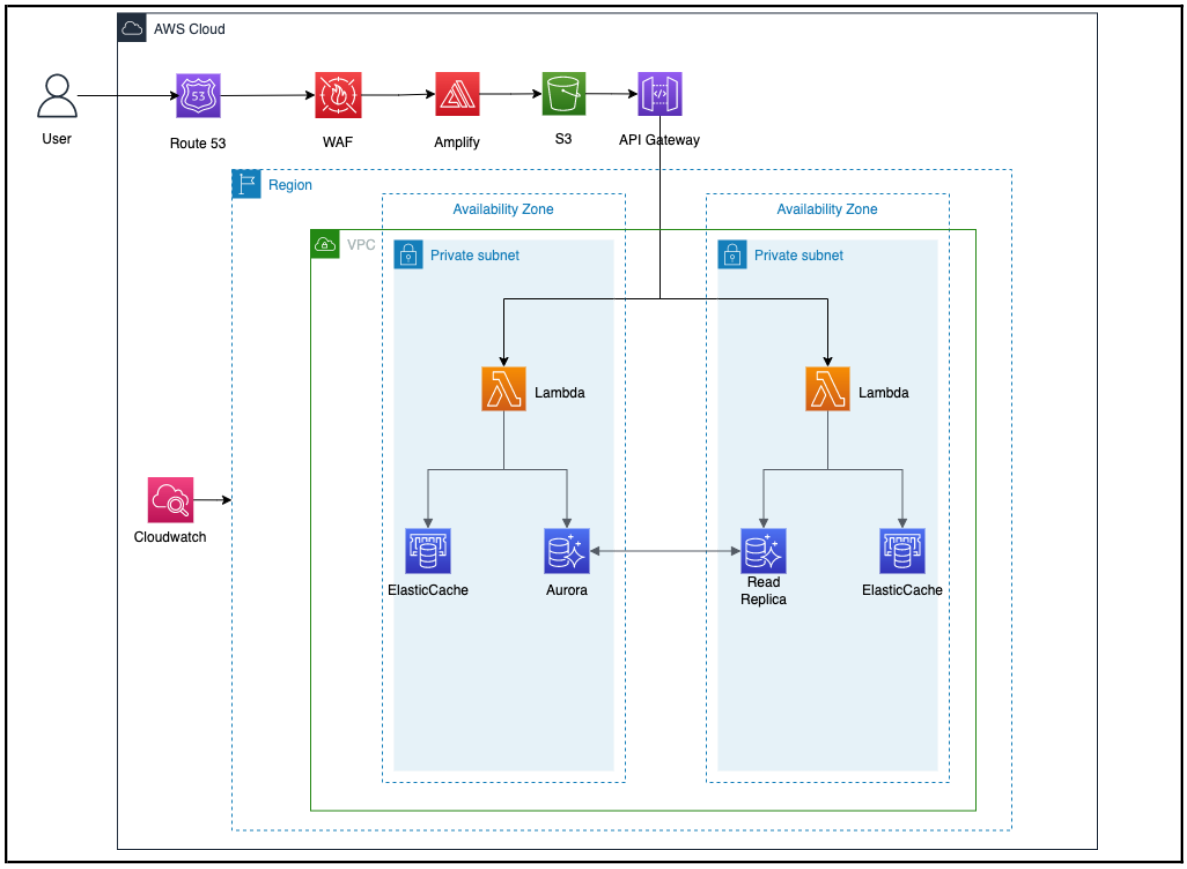
AWS Lambda offers several features to help support your data resiliency and backup needs. With a hot-standby, databases can be recovered quickly without affecting the user experience. AWS Aurora not only ensures high availability for data but also ensures high availability for the DB instance. It can tolerate a failure of an Availability Zone without any loss of data and only a brief interruption of service. Besides, AWS API Gateway is also acting as a load balancer by directing requests to specific resources based on the endpoints being requested, ensuring the system to be highly available.

We propose a 3-step recovery process:

1) Communication: Notifying all relevant stakeholders
    a) Amazon CloudWatch alarms will be configured for all AWS resources. When a resource experiences outages, an email will be sent to notify all relevant stakeholders.
2) Recovery: Restoring AWS resources
    a) After detecting a downed instance (eg. Aurora or ElasticCache), we can check the status of the instance to see if it still exists. If not, we can restore the instance through the last automated snapshot.
    b) If Lambda service(s) is/are down, we will check the AWS CloudWatch logs, resolve the error and push our changes to production again.
3) Testing: Conduct testing to ensure that all operations are working.
    a) After restoring all downed resources, we will perform load testing and continual monitoring to ensure all operations have resumed and are working properly.

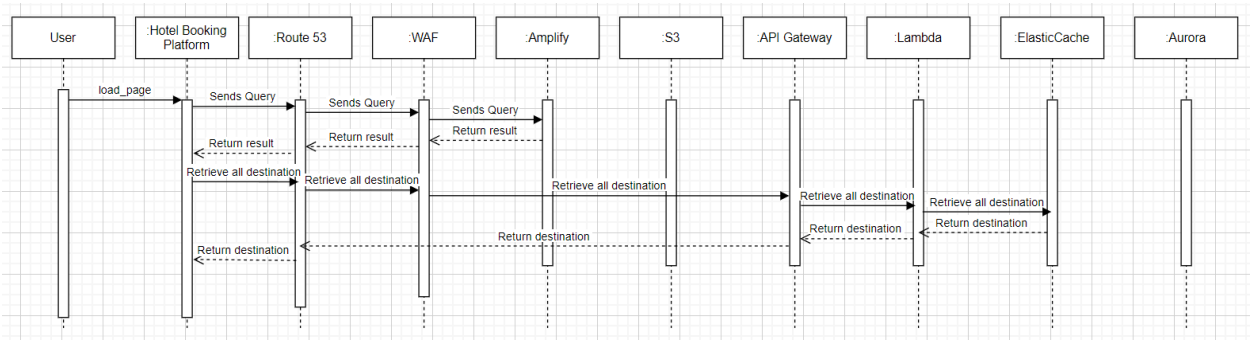
Solution Views

Architecture Diagram

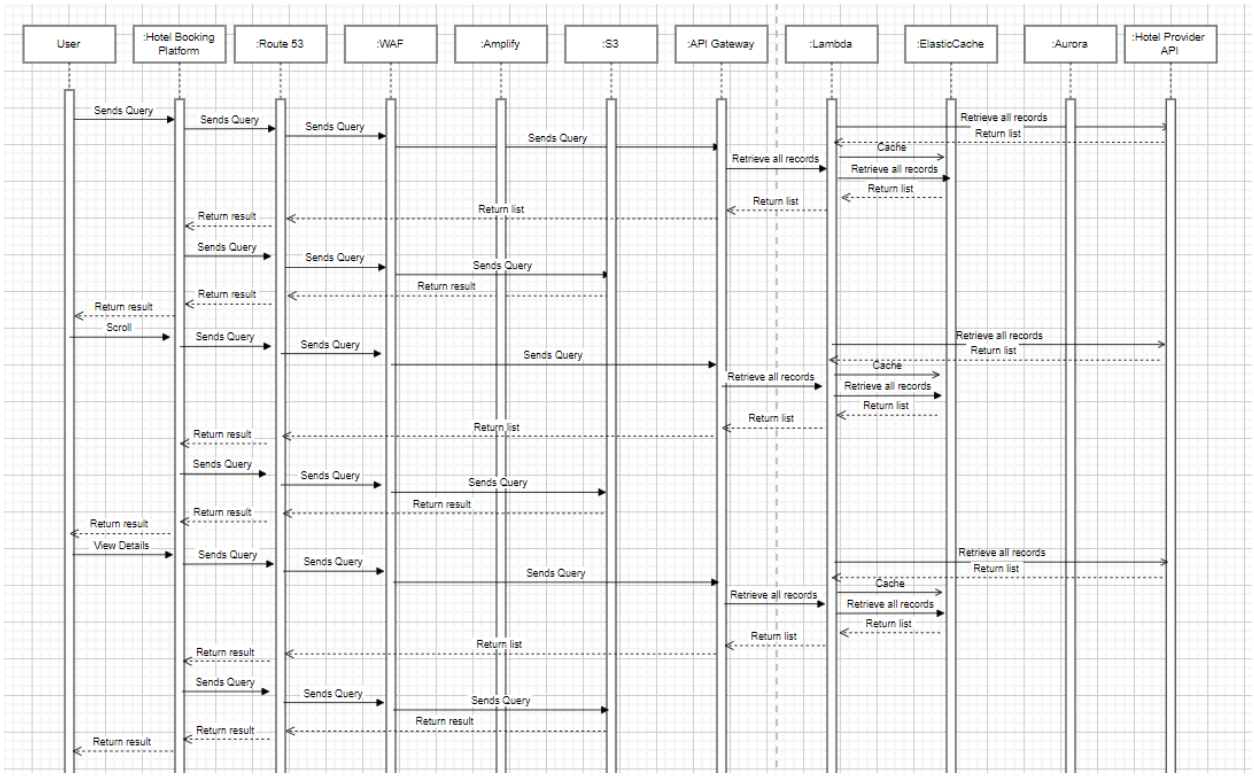


Sequence Diagram

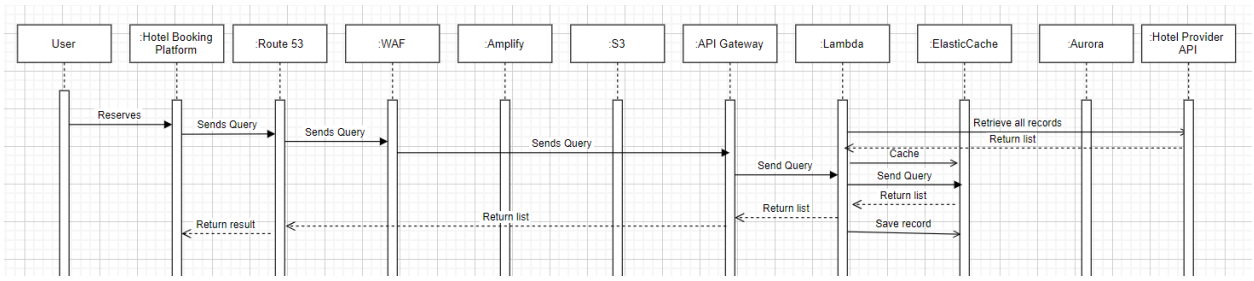
Use Case 1



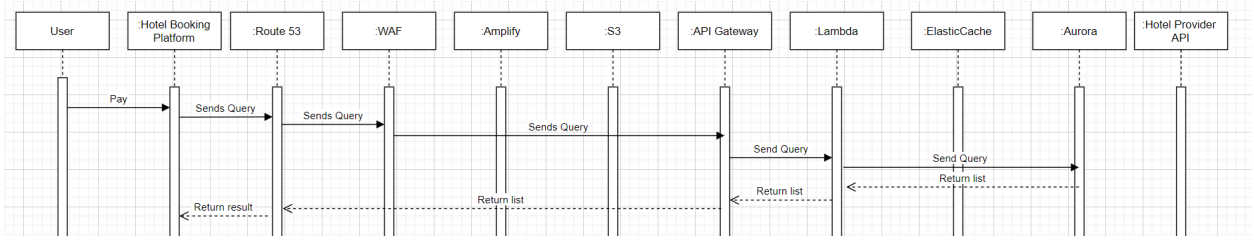
Use Case 2



Use Case 3



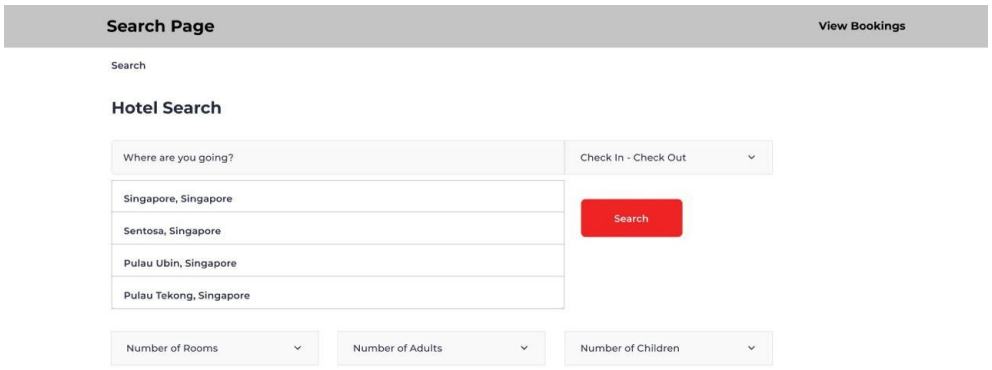
Use Case 4



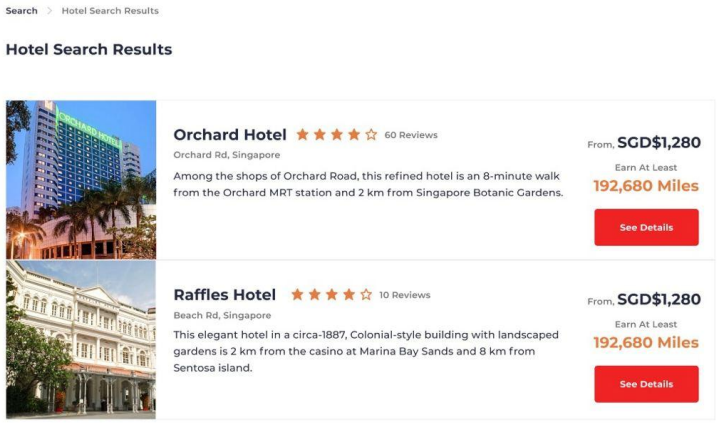
Appendix

1. Figma Screenshots of UI

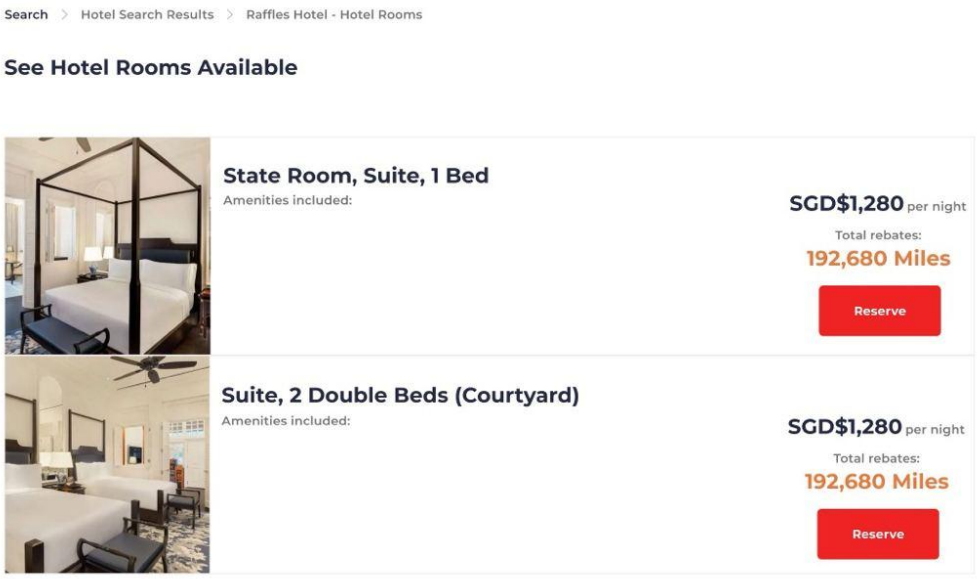
ASR Use Case 1: Enter destination search (autocomplete)



ASR Use Case 2:View and select hotel search results

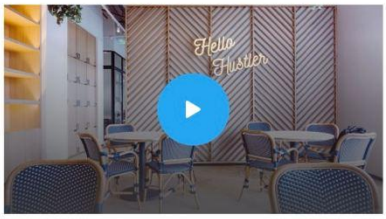


ASR Use Case 3: Reserve hotel rooms



ASR Use Case 4: Confirm booking and make payment

Booking Page



Hotel Room Name

$200

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu
stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcat cher synth. Cosby sweater eu banh mi

Name

Name

We'll never share your email with anyone else.

Credit Card *

Select

Choose from the list.

Credit Card *

Credit Card

Book Now

Booking Confirmation Page

Booking Confirmation #32032jkd28j

Hotel Room Name

$200

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu
stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcat cher synth. Cosby sweater eu banh mi

Details
Name
Identity Number
Credit Card

Back to Home

2. Detailed Budget Calculation Tables

Development Budget

Activity / Hardware / Software / Service	Description	Cost
Development of Solution	Implementation of frontend and backend logic; Setting up of AWS services; Testing	**420 man-hours for a 6-man team**
AWS Lambda	1 million requests per month and 400,000 GB-seconds of compute time per month, usable for functions powered by both x86, and Graviton2 processors.	**Free Tier** https://calculator.aws/#/createCalculator/Lambda
Amazon Aurora	1 x db.t3.small instance	1 instance(s) x 0.041 USD hourly x 730 hours in a month = 29.9300 USD **Amazon Aurora MySQL Compatible cost (monthly): 29.93 USD** Amazon Aurora MySQL Compatible cost (upfront): 0.00 USD https://calculator.aws/#/createCalculator/AuroraMySQL
Amazon API Gateway	1 million API calls received for REST APIs 1 million API calls received for HTTP APIs 1 million messages	**Free Tier** https://calculator.aws/#/createCalculator/APIGateway

	750,000 connection minutes for WebSocket APIs per month Free tier - up to 12 months	
Amazon ElastiCache	2 x Memcache Standard cache t3.micro OnDemand (Hourly) per month	2 instance(s) x 0.01700000 USD hourly x 730 hours in a month = 24.82 USD **ElastiCache cost (monthly): 24.82 USD** ElastiCache cost (upfront): 0.00 USD https://calculator.aws/#/createCalculator/ElastiCache
Amazon S3	5GB of Amazon S3 storage in the S3 Standard storage class 20,000 GET Requests; 2,000 PUT, COPY, POST, or LIST Requests 100 GB of Data Transfer Out each month	**Free Tier** https://calculator.aws/#/createCalculator/S3
AWS Amplify	**Build & Deploy** 1000 build minutes per month **Hosting** 5 GB stored per month and 15 GB served per month	**Free Tier** https://calculator.aws/#/createCalculator/Amplify
AWS WAF	1 x Web Access Control Lists (Web ACLs) 1x Rules added per Web ACL 1 million requests	1 Web ACLs per month x 5.00 USD = 5.00 USD (WAF Web ACLs cost) 1 Web ACLs per month x 1.00 Billable Rules per web ACL per month x 1.00 USD = 1.00 USD (WAF Rules cost) 1 requests per month x 1000000 multiplier for million x 0.0000006 USD = 0.60 USD (WAF Requests cost) 5.00 USD + 1.00 USD + 0.60 USD = 6.60 USD **AWS WAF cost (monthly): 6.60 USD**

		https://calculator.aws/#/createCalculator/WAF
AWS Route 53	1 hosted zone with 1 million queries	The calculations below exclude free tier discounts. Tiered price for: 1 1 x 0.5000000000 USD = 0.50 USD Total tier cost = 0.50 USD (Hosted Zone cost) 1 million queries x 1000000 multiplier for million = 1,000,000.00 Standard queries in million Tiered price for: 1000000.00 Standard queries 1000000 Standard queries x 0.0000004000 USD = 0.40 USD Total tier cost = 0.40 USD (Standard queries cost) 0.50 USD + 0.40 USD = 0.90 USD **Route53 Hosted Zone cost (monthly): 0.90 USD** https://calculator.aws/#/createCalculator/Route53
Amazon CloudWatch	**Metrics** Basic Monitoring Metrics (at 5-minute frequency) 10 Detailed Monitoring Metrics (at 1-minute frequency) 1 Million API requests (not applicable to GetMetricData and GetMetricWidgetImage) **Dashboard** 3 Dashboards for up to 50 metrics per month	**Free Tier** https://calculator.aws/#/createCalculator/CloudWatch

	Alarms 10 Alarm metrics (not applicable to high-resolution alarms) **Logs** 5GB Data (ingestion, archive storage, and data scanned by Logs Insights queries) **Events** All events except custom events are included **Contributor Insights** 1 Contributor Insights rule per month The first one million log events that match the rule per month **Synthetics** 100 canary runs per month **Evidently** First time free trial includes 3 million Evidently events and 10 million Evidently analysis units per account **RUM** First time free trial includes 1 million RUM events per account	
Total Cost:		**Total Monthly Cost: 62.25 USD**

Production Budget

Activity / Hardware / Software / Service	Description	Cost
AWS Lambda	1 million requests per month and 400,000 GB-seconds of compute time per month, usable for functions powered by both x86, and Graviton2 processors. + 100 Concurrency + 30 hours per month + 1 million request with provision concurrency + 30000ms Duration + 500mb memory	Free Tier + 164.59 USD **Total (monthly): 164.59 USD** https://calculator.aws/#/createCalculator/Lambda
Amazon Aurora	2 x db.t2.medium 100 GB Storage 100 GB Backup Storage	Amazon Aurora MySQL Compatible cost (monthly): 119.72 USD Total Storage Cost (monthly): 10.53 USD Additional backup storage cost (monthly): 2.10 USD **Total (monthly): 132.35 USD** https://calculator.aws/#/createCalculator/AuroraMySQL
Amazon API Gateway	1 million API calls received for REST APIs	**Free Tier** + HTTP API request cost (monthly):

	1 million API calls received for HTTP APIs 1 million messages 750,000 connection minutes for WebSocket APIs per month Free tier - up to 12 months After 12 months: 1 million API calls received for REST APIs 1 million API calls received for HTTP APIs	1.00 USD REST API cost (monthly): 31.24 USD **Total (monthly): 32.24 USD** https://calculator.aws/#/createCalculator/APIGateway
Amazon ElastiCache	2 x Memcache Standard cache t3.micro OnDemand (Hourly) per month	2 instance(s) x 0.017 USD hourly x 730 hours in a month = 24.82 USD **ElastiCache cost (monthly): 24.82 USD** ElastiCache cost (upfront): 0.00 USD https://calculator.aws/#/createCalculator/ElastiCache
Amazon S3	5GB of Amazon S3 storage in the S3 Standard storage class 1 million GET Requests; 1 million PUT, COPY, POST, or LIST Requests 100 GB of Data Transfer Out each month	S3 (monthly): 5.52 USD Data Transfer (monthly): 2 USD **Total (monthly): 7.52 USD** https://calculator.aws/#/createCalculator/S3

AWS Amplify	**Build & Deploy** 1000 build minutes per month **Hosting** 5 GB stored per month and 15 GB served per month	**Free Tier** https://calculator.aws/#/createCalculator/Amplify
AWS WAF	1 x Web Access Control Lists (Web ACLs) 1x Rules added per Web ACL 1 million requests	1 Web ACLs per month x 5.00 USD = 5.00 USD (WAF Web ACLs cost) 1 Web ACLs per month x 1.00 Billable Rules per web ACL per month x 1.00 USD = 1.00 USD (WAF Rules cost) 1 requests per month x 1000000 multiplier for million x 0.0000006 USD = 0.60 USD (WAF Requests cost) 5.00 USD + 1.00 USD + 0.60 USD = 6.60 USD **AWS WAF cost (monthly): 6.60 USD** https://calculator.aws/#/createCalculator/WAF
AWS Route 53	1 hosted zone with 1 million queries	The calculations below exclude free tier discounts. Tiered price for: 1 1 x 0.5000000000 USD = 0.50 USD Total tier cost = 0.50 USD (Hosted Zone cost) 1 million queries x 1000000 multiplier for million = 1,000,000.00 Standard queries in million Tiered price for: 1000000.00 Standard queries 1000000 Standard queries x 0.0000004000 USD = 0.40 USD Total tier cost = 0.40 USD (Standard queries cost) 0.50 USD + 0.40 USD = 0.90 USD

		Route53 Hosted Zone cost (monthly): 0.90 USD https://calculator.aws/#/createCalculator/Route5

Amazon CloudWatch	**Metrics** Basic Monitoring Metrics (at 5-minute frequency) 10 Detailed Monitoring Metrics (at 1-minute frequency) 1 Million API requests (not applicable to GetMetricData and GetMetricWidgetImage) **Dashboard** 3 Dashboards for up to 50 metrics per month **Alarms** 10 Alarm metrics (not applicable to high-resolution alarms) **Logs** 5GB Data (ingestion, archive storage, and data scanned by Logs Insights queries) **Events** All events except custom events are included **Contributor Insights** 1 Contributor Insights rule per month The first one million log events that match the rule per month **Synthetics** 100 canary runs per month	**Free Tier** https://calculator.aws/#/createCalculator/CloudWatch

	Evidently First time free trial includes 3 million Evidently events and 10 million Evidently analysis units per account **RUM** First time free trial includes 1 million RUM events per account	
Total Cost:		**Total Monthly Cost: 369.02 USD**