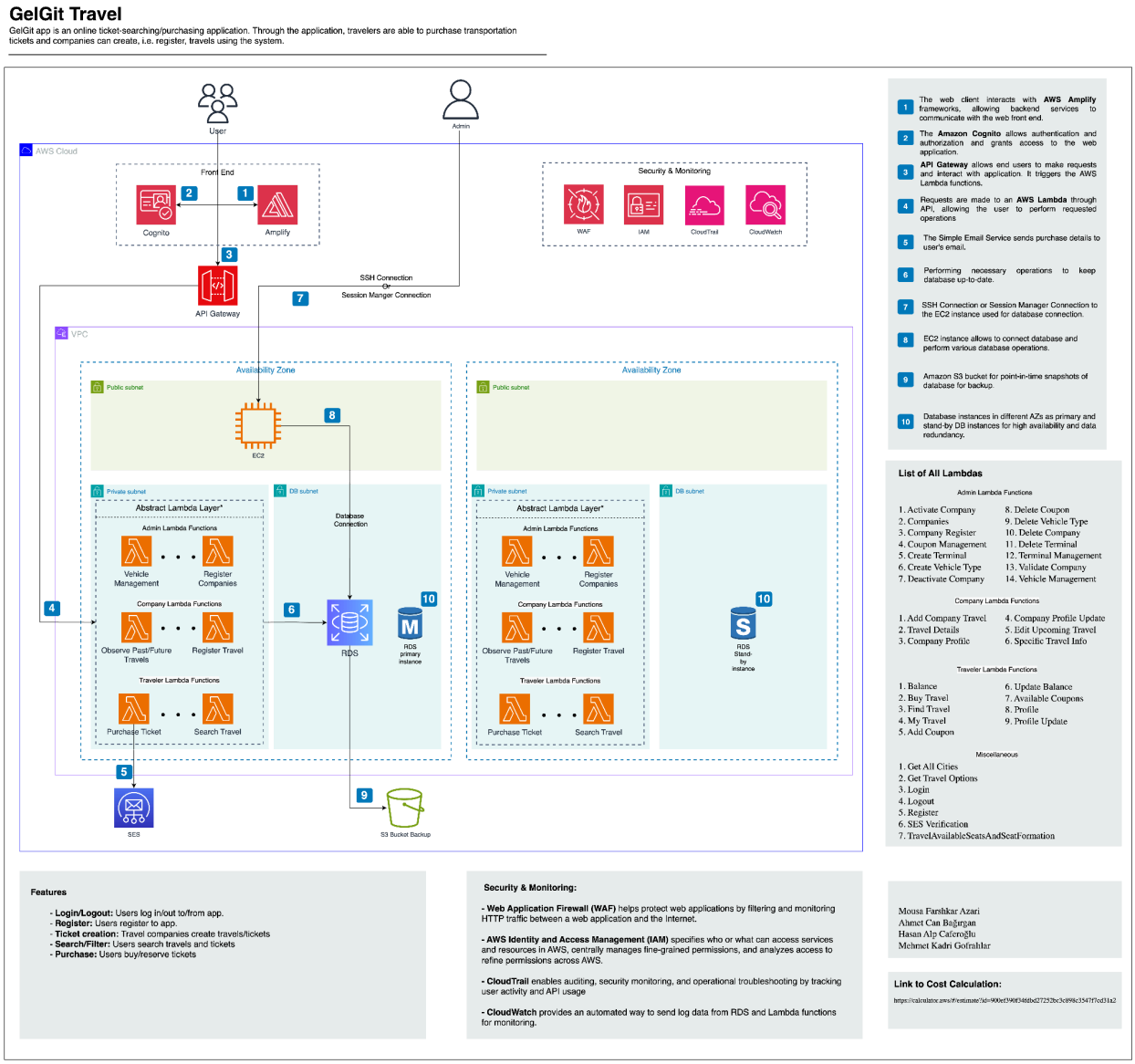
CS683 - Cloud Computing

GelGit Travel

Group A
Mousa Farshkar Azari
Ahmet Can Bağırgan
Hasan Alp Caferoğlu
Mehmet Kadri Gofralılar

System Architecture



Purpose and Configuration of AWS Services Used in GelGit Travel

Amplify: We set up AWS Amplify by connecting it to our GitHub repository and selecting a branch, then configuring the build settings and environment variables to build, deploy, and host our web application automatically. The Auto-build option provided by Amplify enables the CI/CD pipeline.

Cognito: The Amazon Cognito allows authentication and authorization to users and grants access to the web application. To use this service, we first created a userpool and set the authentication and authorization to be performed via our designed UI. User confirmation is done by utilizing the SES service.

API Gateway: For the API Gateway implementation, a new API Gateway Endpoint is created in the API Gateway console. API methods and resources are defined, and integration with backend services like Lambda is established. Security settings, such as IAM roles, are configured accordingly. Additionally, the authorization is handled by a Cognito Authorizer named *gelgitapp_userpool_authorizer*.

Lambda: We implemented over 40 Lambda functions for three different user types, namely admin, company, and traveler. Another Lambda function handles the email sending process for notifying our users. We perform almost all of our backend operations with lambda functions which enables us to provide an automatically scalable and cost effective application. Note that while implementing AWS Lambda functions, necessary libraries that are not provided by default are uploaded.

SES: To implement SES, domain or email addresses are verified in the SES console. Sending and receiving email settings are configured, and email templates are created. SES is integrated with the application to send emails (such as sending informative emails when the user makes a purchase).

EC2: The EC2 instance allows the admin to connect to the database and perform various database operations. We implement EC2 by selecting an appropriate AMI (Amazon Machine Image), choosing our instance type, configuring instance details such as VPC and IAM roles, and then launching the instance. The Session Manager instance profile is also attached to the existing EC2 instance to be able to connect through Session Manager.

RDS: We begin by selecting the appropriate database engine (which is MySQL in our case) and version for our use case. We then configure the DB instance with the necessary settings, such as instance class, storage, VPC subnet, and security groups. Finally, we enable automated backups, set up monitoring and alarm for free storage space with CloudWatch, and configure Multi-AZ deployment for high availability and failover support.

Security and Monitoring:

AWS Identity and Access Management (IAM): Users, groups, and roles with specific permissions are created for IAM. Policies to grant or restrict access to AWS services are configured.

Web Application Firewall (WAF): We implement AWS WAF to protect our web applications from common web exposures by creating a web ACL and defining rules to allow, block, or count web requests based on specified conditions.

CloudTrail: We enable AWS CloudTrail to log all API calls and operations done by AWS user accounts in the AWS console to secure our organization.

CloudWatch: We set up CloudWatch to monitor our AWS resources and services by default and created custom metrics (i.e. number of invocations for email lambda since we have email limit for SES service), dashboards, and alarms. Log groups are configured to collect and store log data from various services such as Lambda, enabling real-time monitoring and alerting.

VPC:

Public Subnet: The Public Subnet is the segment within the VPC that includes an EC2 instance in one of the availability zones. The admin user of the app connects to the EC2 instance through an SSH connection or session manager connection.

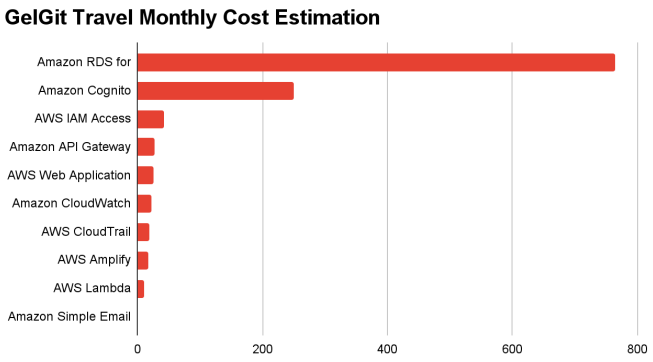
Private Subnet: Private Subnet is the segment within the VPC that does not have a direct route to the internet. Resources in this subnet, such as databases or application servers, are only accessible from within the VPC, enhancing security.

DB Subnet is a specific type of subnet group used for RDS instances, which spans multiple AZs to ensure database availability and failover support.

Cost Estimation

Link: https://calculator.aws/#/estimate?id=23a4229d509703a916eb58d866d64648e69ab357

Service Name	Status	Upfront cost	Monthly cost
Amazon RDS for MySQL	-	0.00 USD	764.50 USD
Amazon Cognito	-	0.00 USD	250.00 USD
AWS IAM Access Analyzer	-	0.00 USD	42.00 USD
Amazon API Gateway	-	0.00 USD	27.68 USD
AWS Web Application Firewall (WAF)	-	0.00 USD	26.00 USD
Amazon CloudWatch	-	0.00 USD	22.09 USD
AWS CloudTrail	-	0.00 USD	20.00 USD
AWS Amplify	-	0.00 USD	17.88 USD
AWS Lambda	-	0.00 USD	11.19 USD
Amazon Simple Email Service (SES)	-	0.00 USD	0.72 USD



Upfront Cost : 0.00 USD

Monthly Cost : 1,182.06 USD

Total 12 Months Cost : 14,184.72 USD

This cost estimation assumes the following scenario:

- Monthly Active User count: 5000
- Monthly Purchase count: 500
    - API Request per Purchase: 20 on average
- Monthly Search count: 5000
    - API Request per Search: 50 on average

This cost estimation shows that the service with the highest cost (i.e. approximately two-thirds of the total cost) is Amazon RDS for MySQL. Meanwhile, even though there are 36 actively called Lambda functions, AWS Lambda is the service with the second lowest cost. This is one of the biggest benefits of using Lambda as the computing service.

The Amazon EC2 instance that is previously shown in the architecture is not included in the cost estimation. This is because the instance is used only to allow admin access to the RDS instance via SSH Connection or Session Manager Connection and can mostly be omitted.