School of Engineering & Computer Science

Department Of Computer Science and Engineering

BeeHome

(A software application designed for bed and breakfast)

Course Name: Advance Topic for Application Development (DevOps)

Course ID: CSE461

Semester: Summer 2023

Section: 01

Done by:

▪Adnan Akanda	1821541
▪Maruf Hasan	1821933
▪Azwad Fawad Hasan	2020222

Table of content:

Section 01

1.1 Introduction

BeeHome is a software application designed specifically for owners of small bed and breakfast establishments. As private family residences with an average of four to nine rooms, with six being the norm, these establishments provide overnight accommodation as well as breakfast for their guests. Typically, the hosts of a BeeHome also live in the home, making it a more intimate and personal experience for guests.

BeeHome provides a comprehensive solution for managing the daily operations of a bed and breakfast, allowing owners to focus on providing a comfortable and welcoming experience for their guests. The software automates and streamlines tasks such as booking management, guest information storage, and marketing efforts. With features such as automated booking confirmations and reminders, guest information management for personalized experiences, BeeHome helps owners efficiently manage their properties.

1.2 Product Description

BeeHome is a web-based software application that provides a user-friendly interface for managing small bed and breakfast establishments. The software is designed to automate and streamline daily operations, allowing owners to manage bookings, check-ins, and checkouts more efficiently.

BeeHome includes features for managing guest information and preferences, such as storing guest contact information, payment details, and special requests or dietary restrictions. Owners can use this information to personalize the guest experience and ensure that guests feel valued and cared for during their stay.

The software also includes reporting and analytics tools that provide owners with insights into their business performance, including occupancy rates, average daily rates, and revenue per available room. This information can be used to identify trends and opportunities for improvement, helping owners to make data-driven decisions about their business.

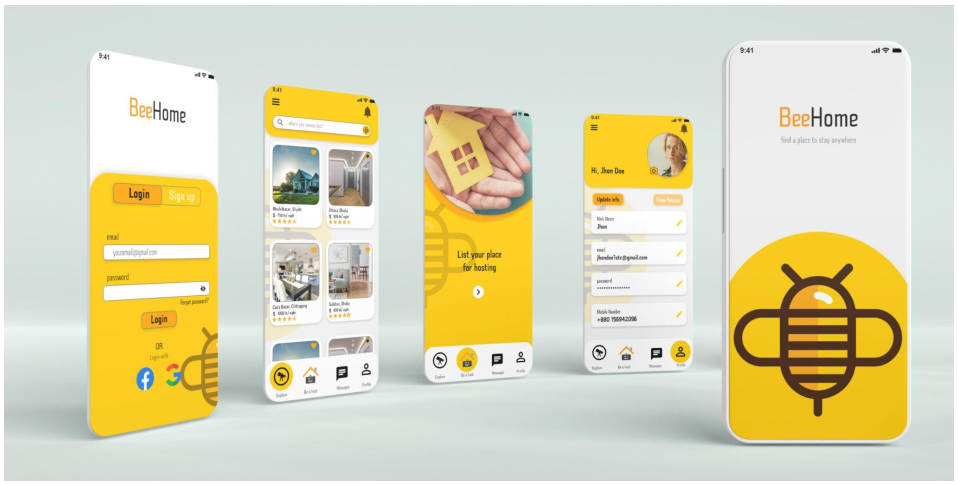
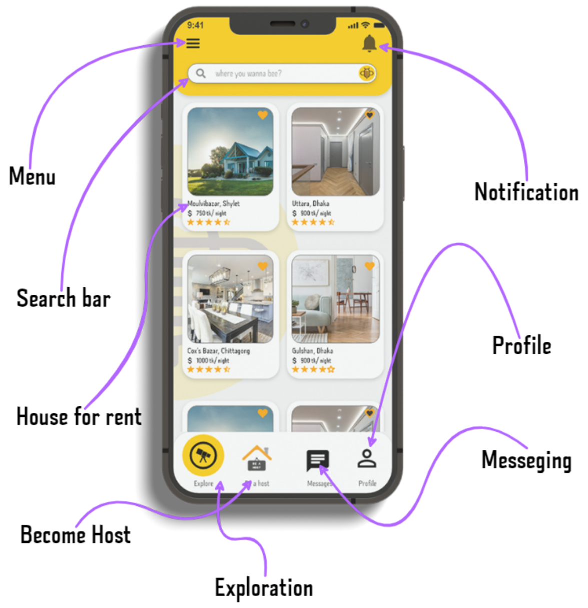


Figure: Beehome as the product

1.3 Key Technical Features of Software



· Mobile app and web-based application accessible from any device with an internet connection.

· User-friendly interface designed for small bed and breakfast establishments.

· Automated booking management, including availability calendars, reservation creation, and booking confirmations and reminders.

· Guest information management, including contact information, payment details, and special requests or dietary restrictions.

Section 2:

2.1 Choice of language and framework and justification

Based on the requirements and features mentioned for the BeeHome software application, we recommend a web framework for developing the application using Python as the programming language and the Flask.

Justification:

- Python is a flexible, simple-to-learn programming language with a large community of third-party libraries, which can speed up development.
- Small to medium-sized web applications work well with Flask, a lightweight Python web framework. It offers flexibility in creating unique solutions for your application and is simple to learn. It also has a sizable community for support.
- Many libraries in the Python ecosystem, such as SQLAlchemy for database management, WTForms for handling forms, and libraries for handling payment processing like Stripe, can make it easier to implement different BeeHome features.
- The development of a web-based application that can be accessed from any device with an internet connection, as required by BeeHome, can be accomplished with ease using Python and Flask.
- A built-in templating engine called Jinja2 in Flask makes it simple to create dynamic and responsive HTML templates for the user interface, giving bed and breakfast owners a user-friendly experience.
- The integration of third-party APIs is well supported by Flask and Python, which is beneficial for marketing initiatives and the integration of analytics tools.
-
- Python is a popular choice for implementing machine learning and data analysis tools, which can be used to leverage the application's reporting and analytics features to provide insights into business performance.

In conclusion, because of their usability, flexibility, and wide support for third-party libraries and APIs, Python and Flask are a great choice for creating the BeeHome software application. The two factors work together to effectively create a web-based application that is user-friendly and specifically designed to meet the requirements of BeeHome.

2.2 Use of Version control, branching model, and justification

Git and the GitFlow branching model are the version control systems that we advise using for the development of the BeeHome software application.

Git's use as the version control system is justified by the following:

- Widely Supported and Used: Because Git is the most well-known and widely used version control system, a sizable community, extensive documentation, and a wide range of tools have been developed around it.
- Git is a distributed version control system that enables every developer to have a local repository that contains the entire project history. With no need for a centralized server, collaboration, offline work, and change merging are made simple.
- Speed and efficiency: Git is suited for a project like BeeHome because it is built to efficiently handle large projects with numerous files and developers.
- Git provides simple branching and merging, enabling programmers to make branches for testing, bug fixes, and new features without affecting the main codebase. Collaboration, code review, and integration of changes back into the main branch are all made simpler as a result.

Justification for using the GitFlow branching model:

- Clear Organization: GitFlow offers a branching model that clearly and logically distinguishes between development, release, and maintenance branches. This structure facilitates effective management of the various phases of the development process and makes it simpler for teams to work together and incorporate changes.
- GitFlow places a strong emphasis on the use of feature branches, which let individual developers work independently on either new features or bug fixes. This guarantees that new features are isolated until they are prepared to be merged and that the main development branch (develop) remains stable.
- GitFlow has a branch (release) specifically for creating and testing releases. Due to the separation, any last-minute fixes can be implemented without affecting ongoing development and the main development branch will remain unaffected throughout the release process.
- GitFlow offers separate branches for hotfixes and maintenance work, making it possible to apply urgent fixes to the production codebase without interfering with the main development branch.

- Scalability: GitFlow is a good option for BeeHome because it works well for projects of all sizes, from little teams to big businesses.

In conclusion, using Git as the version control system and implementing the GitFlow branching model offers a reliable and organized method to manage the BeeHome software application development. This enables effective teamwork, distinct stage separation of the development process, and streamlined release and maintenance procedures.

Gitflow Workflow is what we advise applying to the BeeHome project. Here is a flowchart showing how the Gitflow Workflow's branching model works:

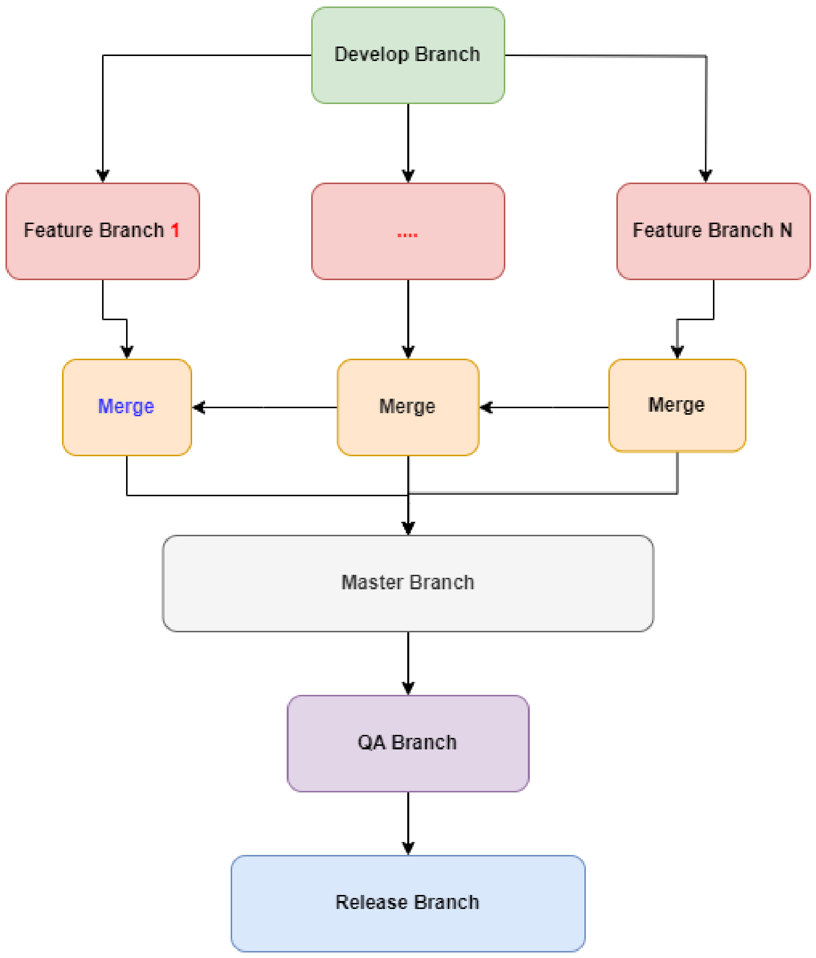


Figure: The branches in the Gitflow Workflow

2.3 Different environments to deploy the system and justification

Here are three different environments for deploying the BeeHome system, along with their justifications:

1. Development Environment:

The development environment is used by developers to build, test, and iterate on new features, bug fixes, and improvements. This environment should be isolated from the production environment to prevent any instability or errors from affecting end users.

Justification:

- Provides a safe space for developers to experiment and test new ideas without impacting the production environment.
- Ensures code quality by allowing developers to test changes thoroughly before pushing to other environments.
- Facilitates collaboration between team members, as they can share and review each other's work.

2. Staging Environment:

The staging environment is a replica of the production environment used to test and validate the application before deploying it to production. It allows for thorough testing of new features, bug fixes, and updates to ensure they work correctly and do not introduce new issues.

Justification:

- Reduces the risk of deploying untested or unstable code to production, which could impact end-users and the application's performance.
- Provides a reliable platform for quality assurance (QA) and user acceptance testing (UAT), ensuring that the application meets the desired requirements and standards.
- Helps identify and resolve potential issues and conflicts before they reach production, saving time and resources.

3. Production Environment:

The production environment is the live environment where the BeeHome system is deployed and accessible to end-users. This environment should be stable, secure, and optimized for performance.

Justification:
- Provides the final platform where end-users can access and use the BeeHome system.
- Ensures a stable and reliable experience for end-users by deploying only thoroughly tested and validated code.
- Enables monitoring and tracking of the application's performance, security, and usage, allowing for continuous improvement and maintenance.

By deploying the BeeHome system in these different environments, you can maintain a clear separation between the development, testing, and production stages. This separation helps ensure the stability and reliability of the application while facilitating efficient collaboration and continuous improvement.

2.4 CI/CD Pipeline for different environments and justification.

We recommend using Jenkins as the CI/CD pipeline tool for the different environments (development, staging, and production) for the following reasons:

1. Open-source and widely used:

Jenkins is a popular open-source CI/CD tool with a large community and extensive plugin ecosystem. Finding support, documentation, and extensions for different needs is now simpler as a result.

Justification:
- Cost-effective as it is free and open-source.
- A sizable community guarantees ongoing updates and improvements.
- A large plugin ecosystem makes it simple to integrate with other programs and services.

2. Flexibility and customization:

Jenkins can be configured and customized with a lot of ease, making it suitable for various environments and unique project requirements.

Justification:

- Enables the development of pipelines that are specifically suited to each environment's requirements (development, staging, and production).
- Supports declarative as well as scripted pipelines, giving the build and deployment process more control.
- Easily customizable and extensible using the Groovy scripting language.

3. Scalability and performance:

Jenkins can be easily scaled to handle large projects and heavy workloads and supports distributed builds.

Justification:

- Assures that the project's growth and increased demand won't overwhelm the CI/CD pipeline.
- Distributed builds allow for task execution in parallel, cutting down on build times and enhancing pipeline efficiency.
- Provides flexibility in terms of infrastructure choices by being able to be deployed on-premises or in the cloud.

In conclusion, Jenkins is a popular, adaptable, and flexible CI/CD tool that can meet the demands of various environments and projects. It is a good fit for the CI/CD pipeline of the BeeHome project due to its open-source nature, customizability abilities, integrations with other tools, and scalability.

2.5 Unit test approach and justification

Since the Test-Driven Development (TDD) methodology has a number of advantages that can raise the overall quality and maintainability of the software, we advise using it for unit testing in the BeeHome project. Prior to writing the implementation code, TDD practices include writing unit tests; the tests and code are then iterated upon until they pass.

Justification for using the TDD approach:

1. Improved code quality:

Developers are compelled to consider the requirements and design of their code before writing it by writing unit tests first. This results in better overall code quality and more thoughtful implementations.TDD makes sure the code complies with the requirements.Writing tests prior to writing code promotes better design and lowers the probability of bugs and problems.Tests act as documentation for the code, facilitating the understanding and maintenance of the code by other developers.

2. Easier debugging and maintenance:

Writing tests for every piece of functionality is a requirement of TDD, making it simpler to find and address problems as they arise. When a test fails, it is obvious which area of the code is to blame, which speeds up and improves the efficiency of debugging.
TDD reduces the amount of time needed for debugging by making it simpler to find the root of problems.Developers can refactor and modify the code with confidence when there is thorough test coverage because they know that any errors will be discovered by the tests.TDD promotes the creation of maintainable, modular code that is simpler to debug.

3. Faster development process:

Although it might seem as though writing tests before the implementation code would slow down development, TDD can ultimately speed up development. Early issue detection makes the development process more efficient because developers don't have to spend as much time troubleshooting and fixing problems.

TDD streamlines the development process by cutting down on the time needed for resolving problems and debugging.TDD helps avoid expensive late-stage bug fixes and redesigns by spotting issues early.TDD promotes iterative development and continuous improvement, which over time results in more productive development.

Developers should use a testing framework appropriate for the selected programming language and framework in the BeeHome project to implement the TDD methodology. Developers can write and run unit tests using the unittest or pytest testing frameworks, for instance, if the project is built in Python using the Flask framework.

Section 3:

3.1 Choice of Cloud Service for deployment and justification.

When implementing the BeeHome application, we advise using Amazon Web Services (AWS). Leading cloud service provider AWS provides a variety of tools and services to make it easier to build, maintain, and scale the application.

Justification for choosing AWS:

1. Comprehensive service offerings:
AWS offers a wide range of services to address different application-related needs, including compute (EC2), storage (S3), databases (RDS), and networking (VPC). You can easily manage and scale your application as necessary thanks to this.

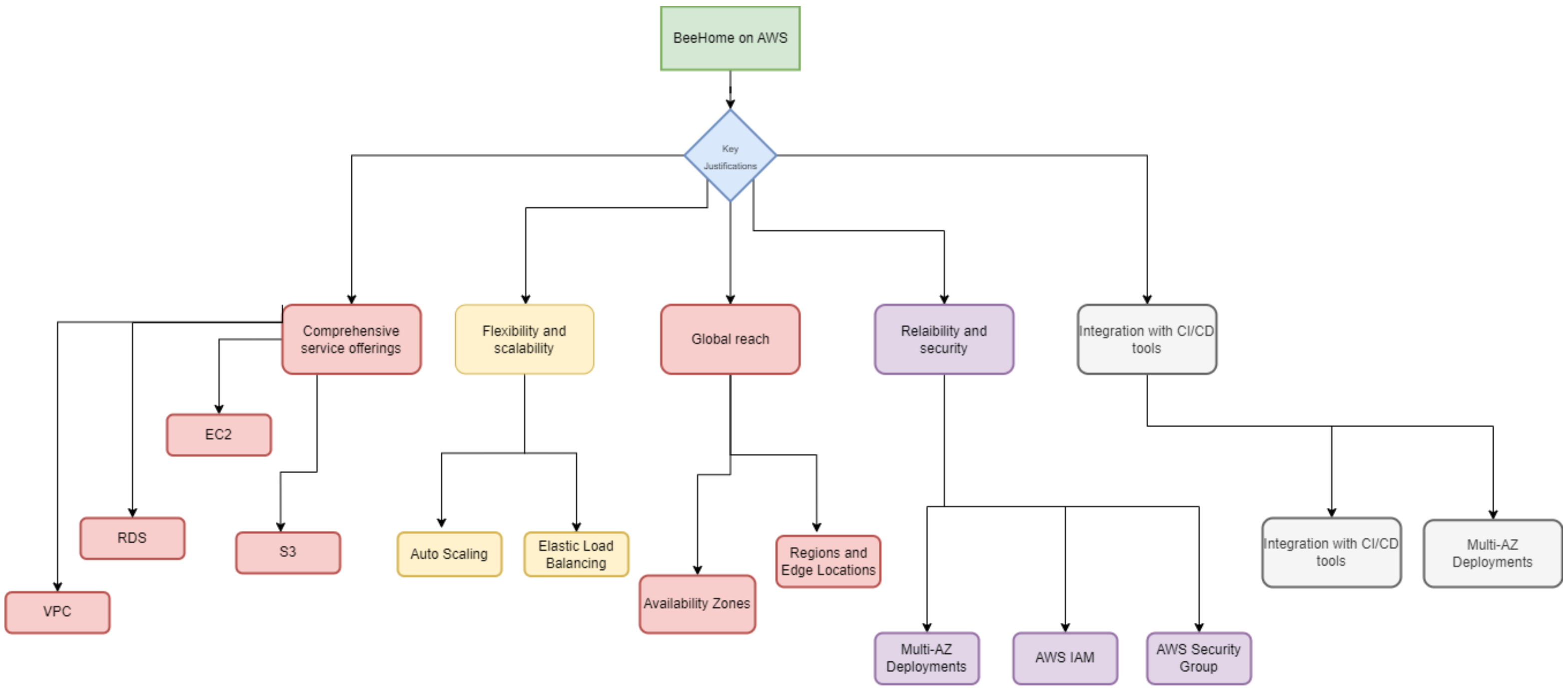
2. Flexibility and scalability:
AWS allows you to scale your resources up or down according to your requirements, ensuring that you only pay for what you use. This flexibility makes it cost-effective for small businesses like BeeHome that may have fluctuating resource needs.

3. Global reach:
AWS enables us to deploy your application in an area that best serves our target audience thanks to its global network of data centers. This can assist our users experience less latency and increase the performance of the application.

4. Reliability and security:
AWS is well known for offering dependable and secure cloud services. They make significant infrastructure and security investments to safeguard your application and data. AWS also offers features and tools like AWS Identity and Access Management (IAM) and AWS Security Hub that assist you in managing and keeping an eye on the security of your application.

5. Integration with CI/CD tools:
Jenkins, Travis CI, and AWS CodePipeline are just a few of the well-liked CI/CD solutions that AWS enables seamless connectivity with. By making it simpler to automate your deployment process, we can give our users new features and upgrades with less time and work.

By setting up the BeeHome application on AWS, we can effectively manage and expand our bed and breakfast management software while benefiting from the platform's adaptability, scalability, and wide range of service options.



.

Figure : Choice of Cloud Service for deployment

3.2 Choice of computation service and justification.

We may utilize Kubernetes with Amazon Elastic Kubernetes Service (EKS) or Docker containers with Amazon Elastic Container Service (ECS) to deploy the BeeHome application. Both solutions offer adaptability, scalability, and effective resource management..

1. Amazon Elastic Container Service (ECS):

Docker containers are supported by Amazon ECS, a service for container orchestration. It enables the deployment, management, and scaling of containerized applications without requiring you to handle the supporting infrastructure.

Justification for choosing Amazon ECS:

- Simple integration with other AWS services: Amazon ECS integrates easily with other AWS services, including CloudWatch for monitoring and logging, S3 for storage, and RDS for databases.
- Flexible and scalable: With ECS, you can deploy applications of varying complexity and scale them up or down in response to demand.
- Effective resource use: By utilizing Docker containers, you can make sure that your application functions effectively and consistently in various environments.
- Economical: Amazon ECS is a cost-effective choice for small businesses like BeeHome because you only pay for the resources used by your containers.

2. Amazon Elastic Kubernetes Service (EKS):

You can use Amazon EKS, a managed Kubernetes service, to deploy, administer, and scale containerized applications.

Reasons for selecting Amazon EKS:

Strong Kubernetes support: EKS offers a fully managed control plane for Kubernetes, ensuring a dependable and current Kubernetes environment.
- Integration with AWS services: Similar to how ECS integrates with different AWS services, EKS does the same, making it simple to create a comprehensive solution for your application.
- Scalability and dependability: To ensure high availability and fault tolerance, EKS automatically scales your application to meet demand and distributes load across multiple availability zones.
- Ecosystem and community support: Kubernetes has a sizable and vibrant community that gives users access to a wealth of plugins, tools, and best practices for managing containerized applications.

Excellent choices for deploying the BeeHome application include Amazon ECS and EKS. Your preference for either Kubernetes or Docker containers, as well as your particular needs for scalability, management, and integration with other services, will determine which one you should use.

3.3 Details of additional cloud services and justification

Here are some additional cloud services that can be used with the BeeHome application, along with justifications for their inclusion:

1. Amazon RDS (Relational Database Service):
Amazon RDS is a managed relational database service that simplifies the process of setting up, operating, and scaling a relational database in the cloud. It is compatible with a number of database engines, including PostgreSQL, MySQL, and MariaDB, which can be used to store data for the BeeHome application. We can simply build, manage, and grow databases using Amazon RDS, automate backups, and use security best practices.

2. Amazon S3 (Simple Storage Service):
Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is a great option for storing pictures, movies, and other static files for the BeeHome application because it can be used to store and retrieve unlimited quantity of data. Our data will be stored securely thanks to its robustness, accessibility, and security features.

3. Amazon CloudFront:
Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to users globally with low latency and high transfer speeds. By caching and providing content from edge locations nearer to the users, CloudFront can enhance the BeeHome application's performance for them. By doing so, latency can be decreased and user experience can be improved.

4. AWS Lambda:
The serverless compute service AWS Lambda executes our code in response to events and manages the underlying compute resources for you automatically. The BeeHome application can use Lambda to handle a variety of background processes, like processing uploaded photos and sending email messages. We can create a scalable and economical architecture with Lambda that only costs for the compute time needed when the functions are active.

By leveraging these additional cloud services, you can build a robust, scalable, and cost-efficient infrastructure for the BeeHome application, ensuring a seamless user experience and efficient management of resources.

5. AWS Cognito: Justification: Both bed and breakfast owners and visitors may need user authentication and access management for BeeHome. You can quickly add sign-up and sign-in functionality to your application using AWS Cognito's user management and authentication services. For increased security, it also supports multi-factor authentication and social media logins. You can effectively manage user access and permissions by utilizing AWS Cognito, ensuring a safe and customized experience for each user.

6. Simple Notification Service (AWS SNS): Justification Real-time notifications and alerts for booking confirmations, cancellations, or other critical updates may be advantageous for BeeHome. You can send messages to multiple subscribers at once using AWS SNS, a fully managed pub/sub messaging service. You can send notifications to mobile devices using SNS via email, SMS, or even push notifications. By informing guests and bed and breakfast owners of significant updates, this will improve the user experience.

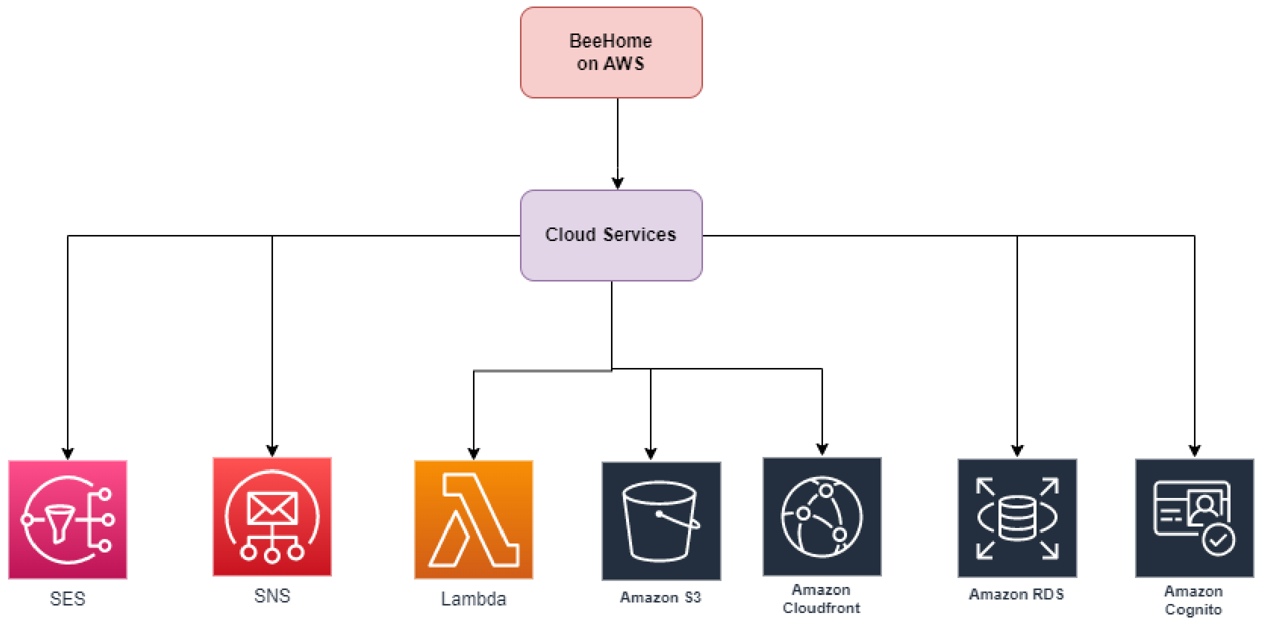


Figure: Additional cloud services

3.4 Deployment Architecture on the Cloud and Justification.

For the BeeHome application, a microservices-based deployment architecture on the cloud is recommended. This architecture involves breaking the application into smaller, loosely coupled services that can be independently developed, deployed, and scaled. The deployment architecture for the BeeHome application should include the following components:

1. API Gateway:

   An API Gateway, such as Amazon API Gateway, should be used to manage and route incoming requests to the appropriate microservices.

2. Microservices:

   Depending on the unique needs of each service, each microservice should be deployed utilizing a compute service like AWS Lambda, Amazon ECS, or Amazon EKS.

3. Data storage:

   Depending on the data storage requirements, use a combination of Amazon RDS for relational databases and Amazon S3 for object storage.

4. Caching and content delivery:

   Leverage Amazon CloudFront to cache and serve static assets, improving the application's performance and reducing latency for users.

5. Load balancing:

   Use AWS Elastic Load Balancing to distribute incoming traffic across multiple instances of your services to ensure high availability and fault tolerance.

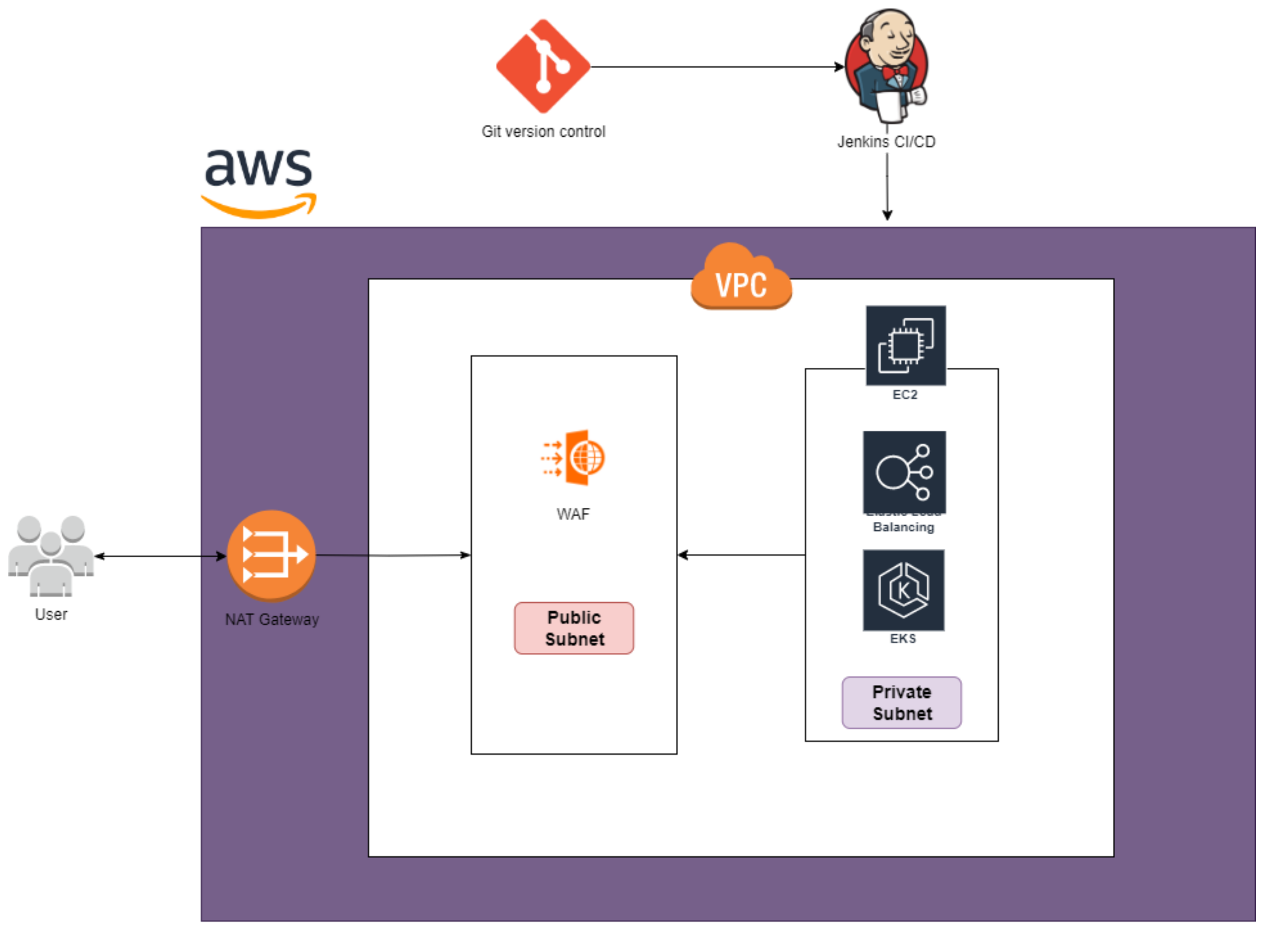


Figure: BeeHome Cloud architecture

By implementing a microservices-based deployment architecture on the cloud, you can build a scalable, flexible, and resilient application that efficiently utilizes resources and provides a seamless user experience.