AWS Compute Blog

# Building well-architected serverless applications: Introduction

by Julian Wood | on 02 APR 2020 | in AWS Well-Architected Tool, Best Practices, Serverless, Technical How-to | Permalink | Share

The following posts are a multi-part series addressing each of the questions within the Serverless Lens of the Well-Architected Tool.

## Series Episodes: Building well-architected serverless applications

1. Introduction
2. Operational Excellence: Understanding serverless application health
    1. Out of the box metrics and alerts; structured and centralized logging
    2. Custom metrics and distributed tracing
3. Operational Excellence – Approaching serverless application lifecycle management
    1. Using infrastructure as code with version control
    2. Deploying to multiple stages using temporary environments, and rollout deployments
    3. Configuration management, CI/CD, and managing function runtime deprecation
4. Security – Controlling access to serverless APIs
    1. Authentication and authorization for public and private API endpoints
    2. Amazon Cognito user and identity pools, JSON web tokens, API keys, and usage plans
    3. Separating authenticated users into logical groups
5. Security – Managing serverless security boundaries
    1. Resource polices, controlling and blocking network access, and audit tools
    2. Using temporary credentials, smaller single-purpose functions, and auditing permissions
6. Security – Implementing serverless application security
    1. Security awareness documentation, inspecting code dependencies, and validating inbound events
    2. Securely storing, auditing, and rotating secrets used in application code
7. Reliability – Regulating inbound request rates
    1. Throttling and performance testing
    2. API quotas and mechanisms to protect non-scalable resources
8. Reliability – Building resiliency into your serverless application
    1. Managing failures using retries, dead-letter-queues, and using state machines

## Well-architected

Customers building production applications in the cloud are looking to build and operate their applications following best practices. Best practices are useful throughout the software development lifecycle and help to answer the question “am I well-architected?”

Serverless technologies provide a solid foundation for building well-architected applications with a goal to reduce and minimize the impact of issues that can happen. What does it mean to be “well architected”?

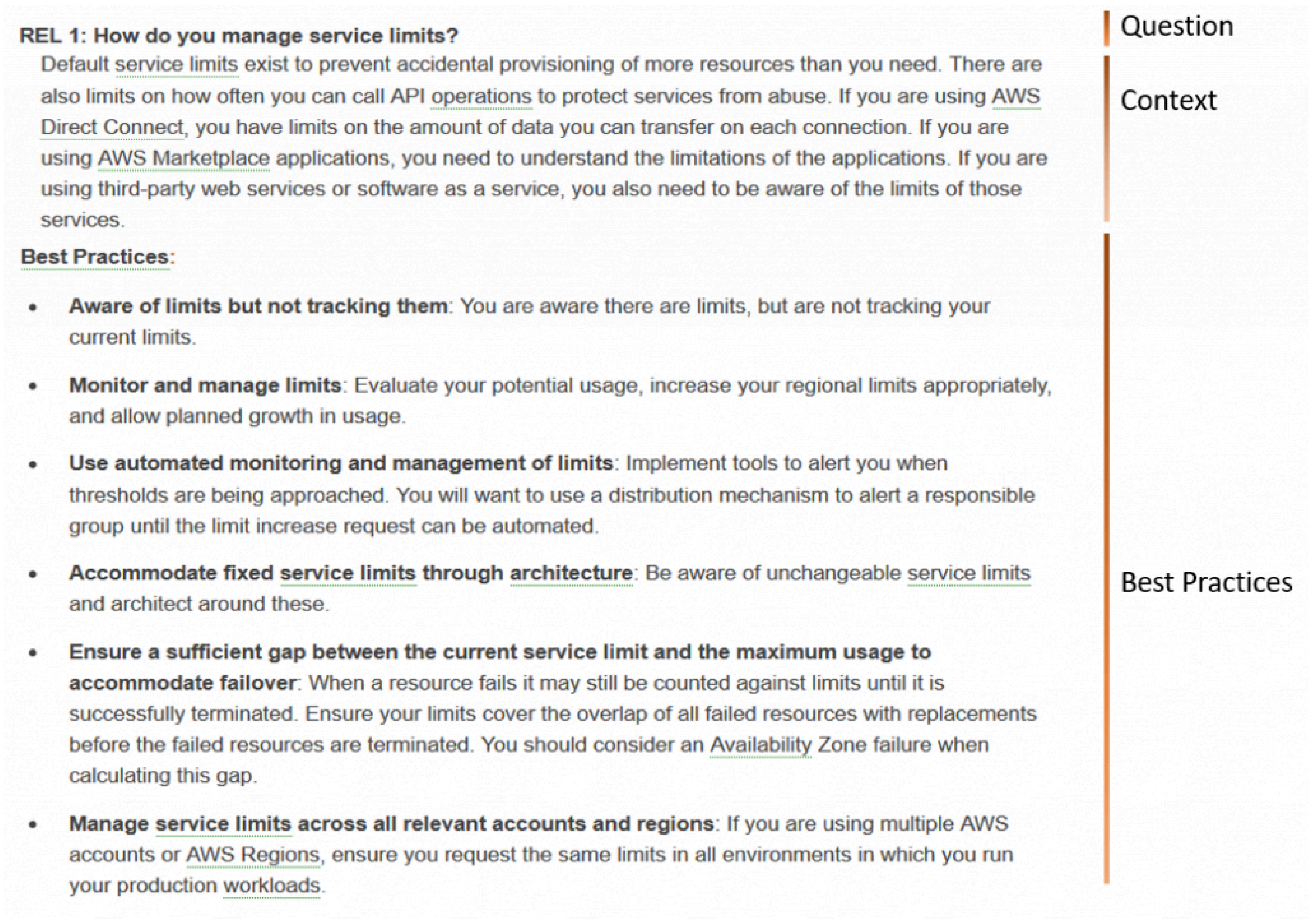
In 2015, AWS released the AWS Well-Architected Framework. It’s a structured way to compare applications against AWS architectural best practices with advice on how to improve. This formalized framework publicly documents the approach our solutions architects use when conducting architectural reviews. The Well-Architected Framework can be used by customers and partners to evaluate applications. It’s currently based on five pillars:

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization

In 2017, AWS extended the framework’s general advice to be more application domain specific with the concept of a “lens”. A lens adds additional questions for specific technology areas, which focus on what is different from the generic advice. Today, there are three technology domain lenses:

- Serverless
- High Performance Computing (HPC)
- IoT (Internet of Things)

In 2018, AWS added the AWS Well-Architected Tool within the AWS Management Console. The tool allows you to define a Workload and answer a number of questions based on each of the five pillars. Each question has context to explain what the question means and why it is important, and then provides a number of best practices.



**REL 1: How do you manage service limits?**

Default service limits exist to prevent accidental provisioning of more resources than you need. There are also limits on how often you can call API operations to protect services from abuse. If you are using AWS Direct Connect, you have limits on the amount of data you can transfer on each connection. If you are using AWS Marketplace applications, you need to understand the limitations of the applications. If you are using third-party web services or software as a service, you also need to be aware of the limits of those services.

**Best Practices:**

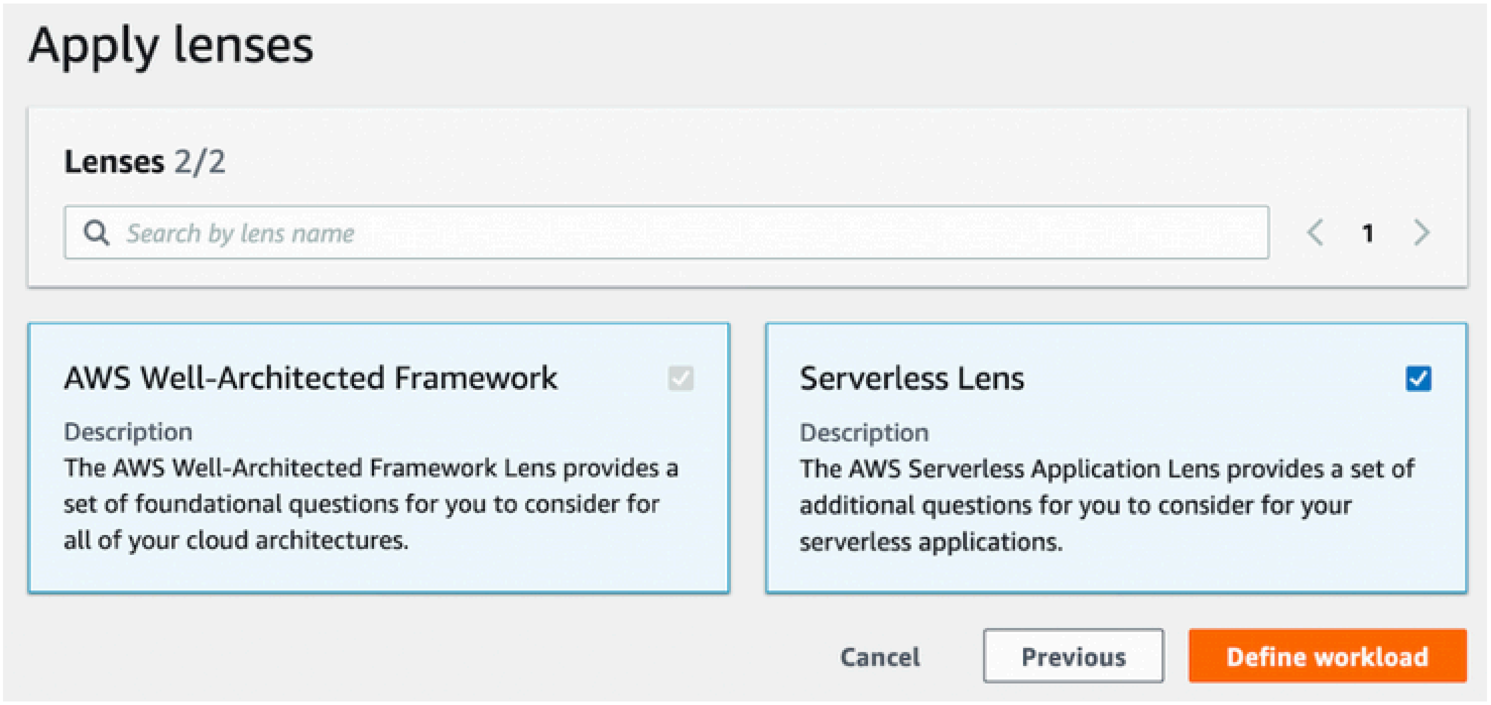
- **Aware of limits but not tracking them**: You are aware there are limits, but are not tracking your current limits.
- **Monitor and manage limits**: Evaluate your potential usage, increase your regional limits appropriately, and allow planned growth in usage.
- **Use automated monitoring and management of limits**: Implement tools to alert you when thresholds are being approached. You will want to use a distribution mechanism to alert a responsible group until the limit increase request can be automated.
- **Accommodate fixed service limits through architecture**: Be aware of unchangeable service limits and architect around these.
- **Ensure a sufficient gap between the current service limit and the maximum usage to accommodate failover**: When a resource fails it may still be counted against limits until it is successfully terminated. Ensure your limits cover the overlap of all failed resources with replacements before the failed resources are terminated. You should consider an Availability Zone failure when calculating this gap.
- **Manage service limits across all relevant accounts and regions**: If you are using multiple AWS accounts or AWS Regions, ensure you request the same limits in all environments in which you run your production workloads.

Well-Architected Tool question

The tool is used to assess risks, and find opportunities for improvement for workloads. It can also provide a broader view across multiple applications for a central team. Each question has a number of best practices to follow, categorized as high or medium risk. This can help you decide where to focus.

## Serverless Lens

In February, we added functionality to apply lenses to the Well-Architected Tool. The first one is the Serverless Lens. This includes nine questions, each with additional best practice recommendations to follow.


Apply lenses

Lenses 2/2

AWS Well-Architected Framework ✓

Description
The AWS Well-Architected Framework Lens provides a set of foundational questions for you to consider for all of your cloud architectures.

Serverless Lens ✓

Description
The AWS Serverless Application Lens provides a set of additional questions for you to consider for your serverless applications.

Serverless Lens in console

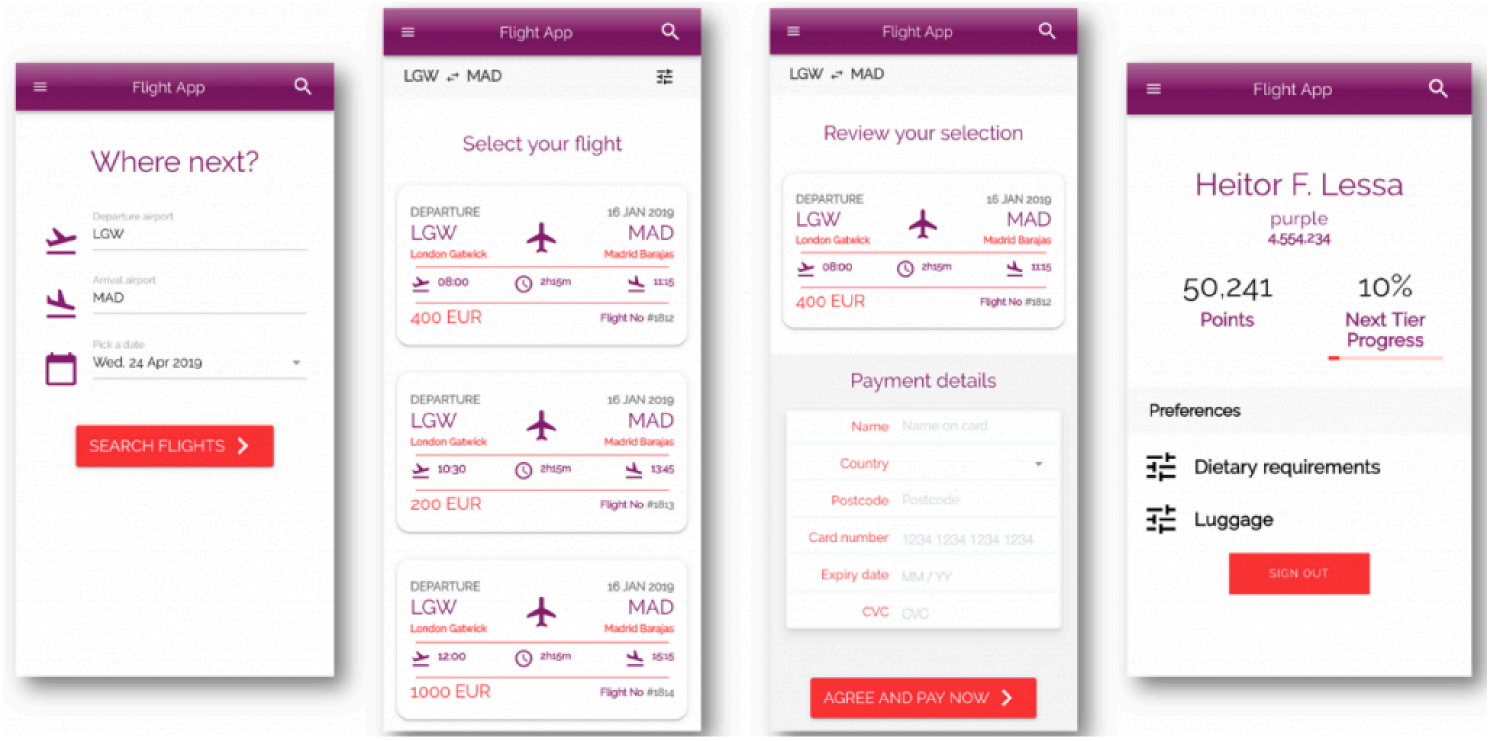
Applying best practices to production applications may need some more time and effort. The Well-Architected Tool and Serverless Lens are available to help and are not intended to only be a one-time check.

We suggest at a minimum reading the whitepapers and being familiar with the questions before the design phase. It's a good idea to check throughout the application lifecycle; halfway (or sooner), close to launch, and post-launch for each iteration. Although it is never too late to add best practices to your applications, starting early helps reduce the remediation process as well as making it part of the full application lifecycle journey.

## Example application

For this series, I am looking at an example application, AWS Serverless Airline Booking. This is a complete web application for a fictional airline booking service built using serverless technologies that provide Flight Search, Payment, Booking, and Loyalty Point features.

I show how to use the Well-Architected Tool with the Serverless Lens to help apply serverless best practices to the application.

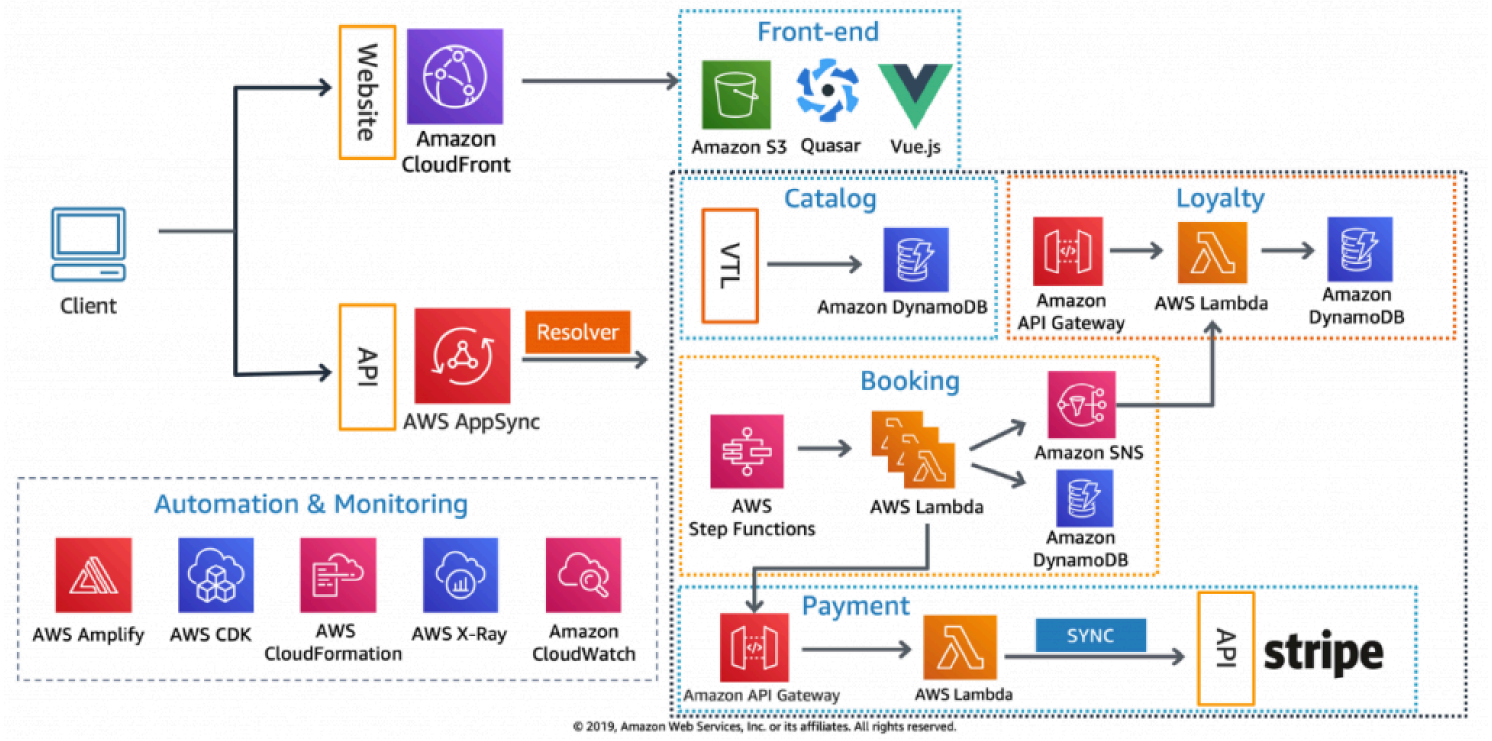


Airline mobile example screenshots

This web application is the theme of Season 2 of Build on Serverless, which is a video series on Twitch.tv, and presented at AWS re:Invent 2019.

You can view this application directly on GitHub or deploy into an AWS account using the Getting Started instructions. This is not required to follow this Building Well-Architected Serverless Applications series but allows you to explore the code.

The application architecture consists of a frontend web application, which interacts with a number of backend microservices to provide the airline booking functionality.

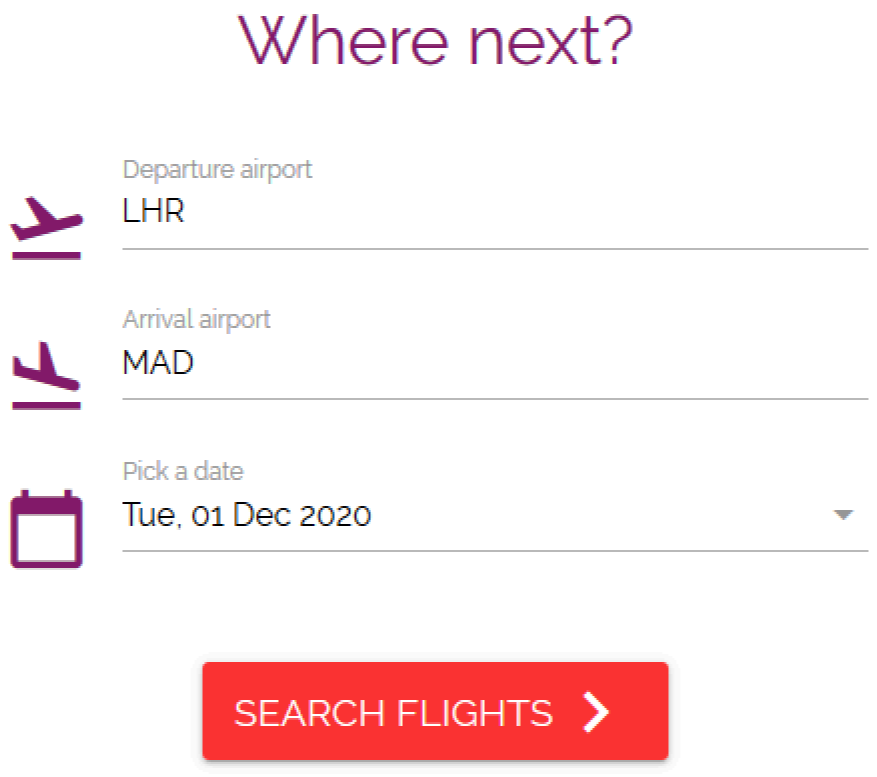


Airline architecture diagram

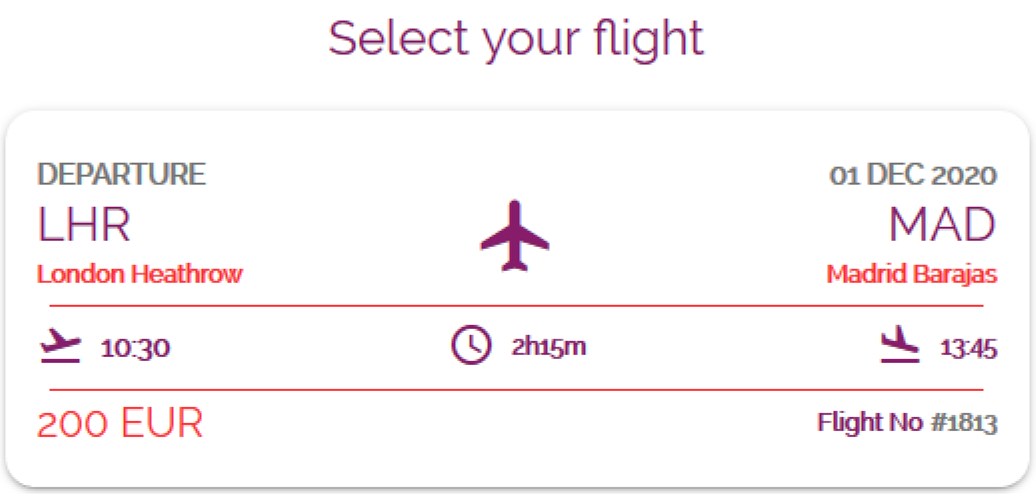
There are four backend services that make up the application:

| Service | Description |
|---|---|
| Catalog | Provides flight search |
| Booking | Provides new and list bookings. Business workflow implemented in Python. |
| Payment | Provides payment authorization, collection, and refund workflows using Stripe. Business workflow implemented in Python |
| Loyalty | Provides loyalty points for customers including tiers. Implemented in TypeScript. |

Once the application is deployed, I can create a booking by searching for flights.

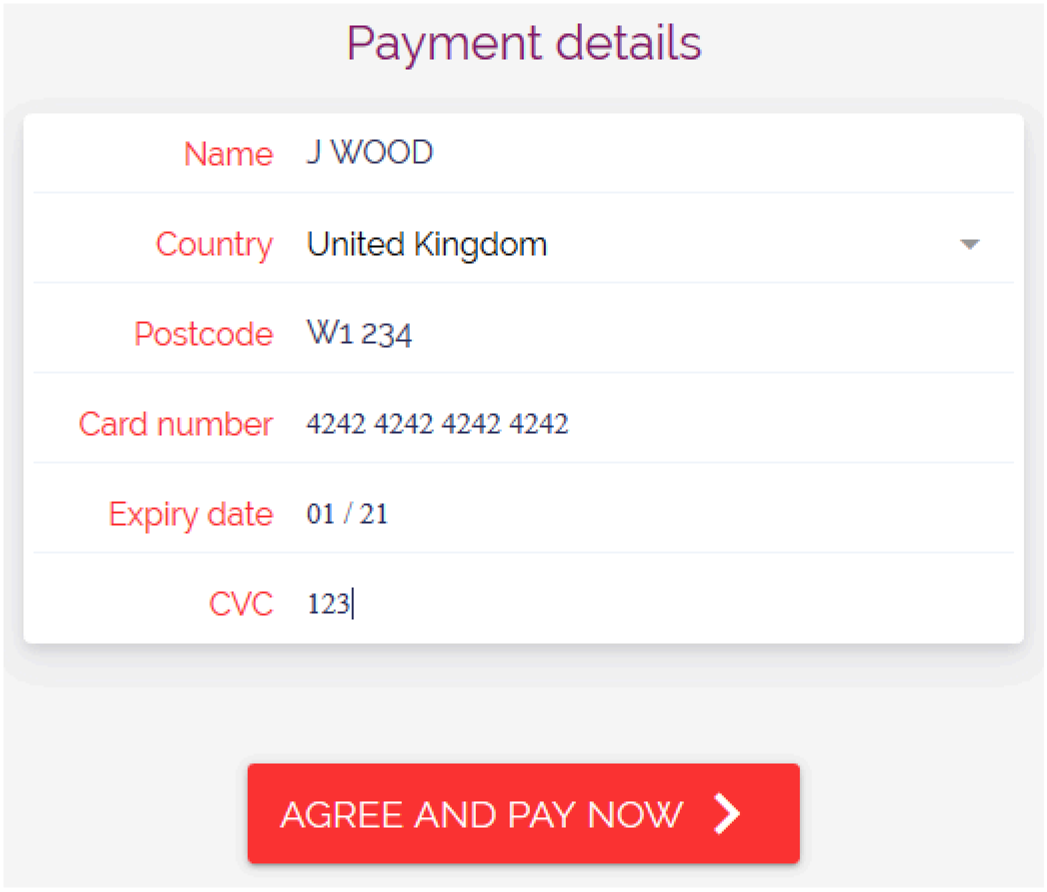


Search flight



Select flight

Once I select an available flight, I enter payment details using a test Stripe credit card.

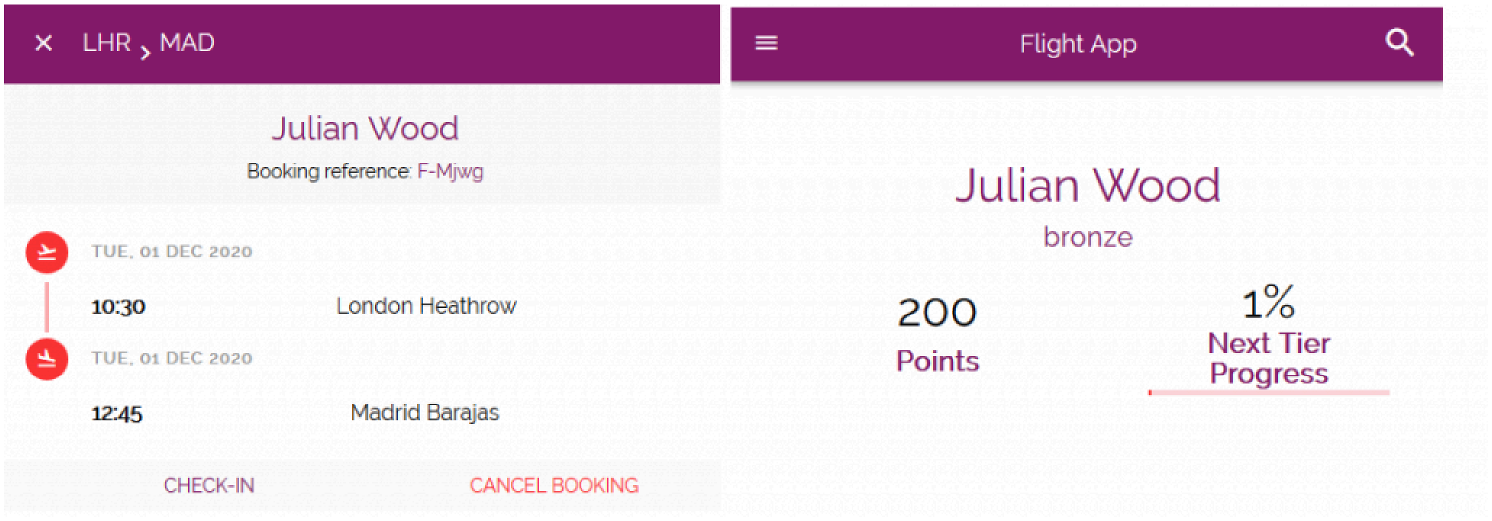


Payment details

| | |
|---|---|
| Name | J WOOD |
| Country | United Kingdom |
| Postcode | W1 234 |
| Card number | 4242 4242 4242 4242 |
| Expiry date | 01 / 21 |
| CVC | 123 |

Flight Payment

The application authorizes the payment, confirms, and stores the booking and then updates the loyalty points.



Flight confirmed

## Next steps

Building serverless applications using best practices can give you more confidence in the architecture and operations of your workloads.

Using the Well-Architected Tool and Serverless Lens can give you more visibility into your applications. They can help pinpoint and rank areas to improve. Well-Architected works best when integrated into your application lifecycle processes.

Continue learning in the next post, which dives into the first Well-Architected Serverless Lens question: Operational Excellence – Understanding Serverless Application Health – part1.

TAGS: serverless, well-architected