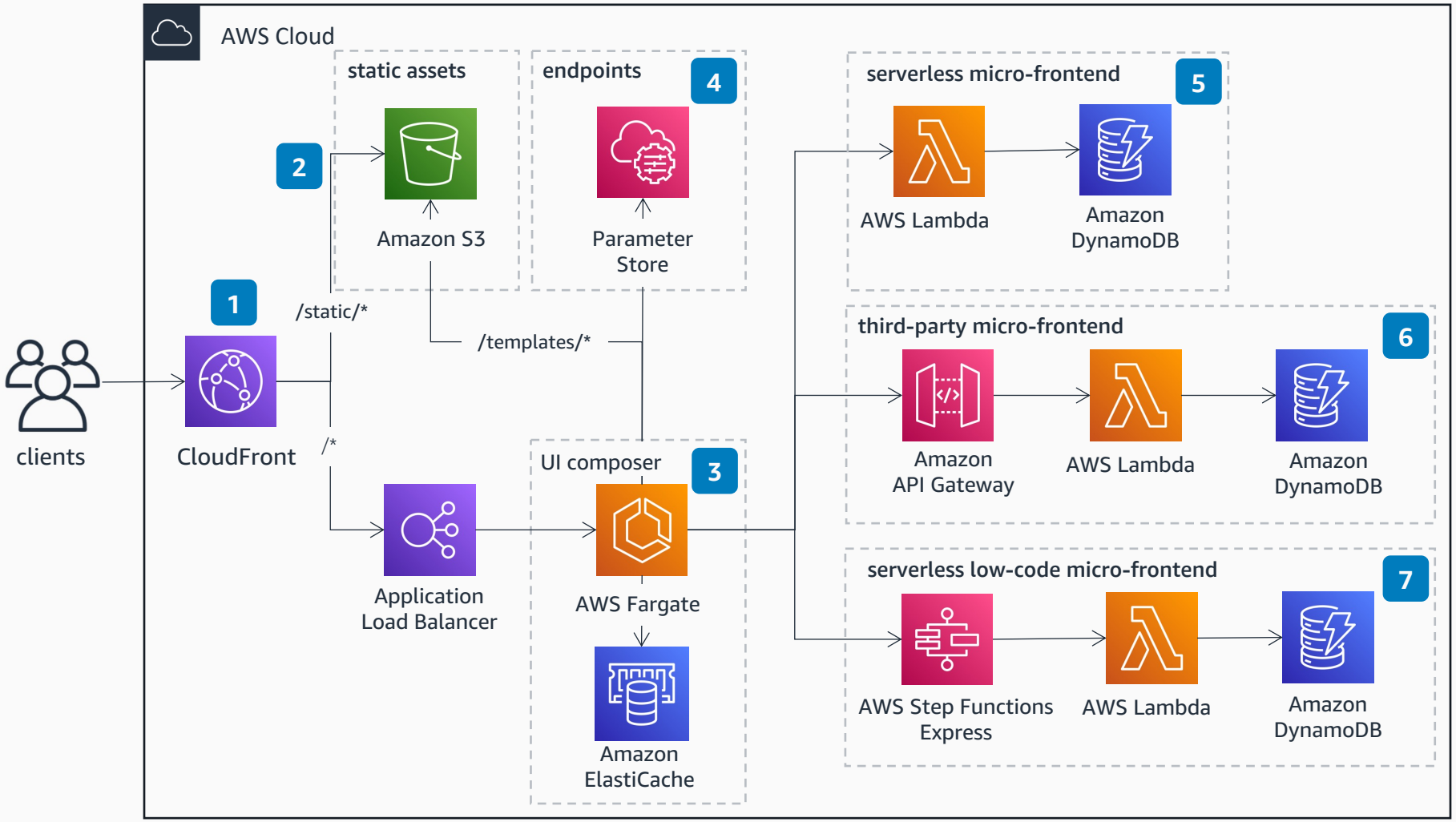
Server-Side Rendering Micro-Frontends in AWS

Embrace distributed systems on the frontend using micro-frontends. This approach shows how to implement server-side rendering micro-frontends in AWS using a serverless approach. Every serverless micro-frontend returns an HTML fragment (HTML-on-the-wire), and a UI composer is responsible for stitching together these independent parts, creating a seamless experience for users.



1. **Amazon CloudFront** is the entry point of this architecture. It has two origins: an **Amazon Simple Storage Service** (**Amazon S3**) bucket, and a public **Application Load Balancer**.

2. The **Amazon S3** bucket contains all the static files to be served for the browser, such as common micro-frontends dependencies, images, or CSS files. It also contains the templates needed by the UI composer for placing every micro-frontend in the right place on an HTML page.

3. The UI composer is an **AWS Fargate** cluster that stiches together different micro-frontends and serves them to the browser, streaming the response to improve web application performance. To increase performance even further, you can use an **Amazon ElastiCache** cluster for caching some micro-frontends output, or even an entire page.

4. Use **AWS Systems Manger Parameter Store** to collect all the micro-services endpoints. They can be HTTP endpoints, or Amazon Resource Names (ARNs) of a specific service such as **AWS Lambda** or **AWS Step Functions**. This decoupling lets you maintain independence between teams working in the application.

5. This serverless micro-frontend is composed by **AWS Lambda** and **Amazon DynamoDB** for storing data to render. The output is an HTML fragment ready to be embedded in the template composed by the UI composer.

6. When you work with third-party companies in the same application, you can enhance the security of your requests using **Amazon API Gateway** for validating tokens or API keys, ensuring only your application can access that endpoint.

7. Use **Step Functions Express** as a low-code solution for generating micro-frontends. Thanks to **Step Functions** integration with over 200 services, you can reduce the amount of computation needed and retrieve data from **Amazon DynamoDB** natively, for example, and delegate just the rendering of these data to an **AWS Lambda** function.

Reviewed for technical accuracy September 9, 2022

AWS Reference Architecture