Senior Design Business Analysis: Agility

Serverless Deployment Made Easy

Team: Abhishek Pandya, Aditya Bhati, Anish Bikmal, Jared Asch

Executive Summary

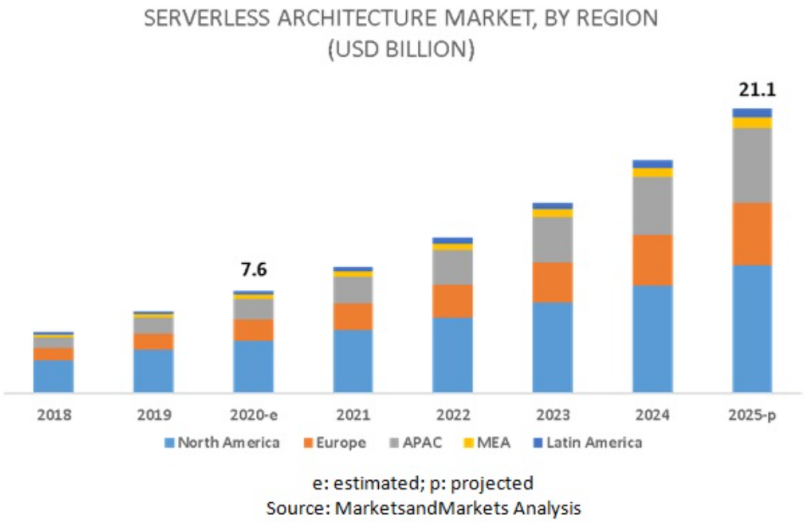
Agility is a simple, effortless way of creating and deploying serverless applications. First, create your architecture diagram, connecting any resources you need with supported edge types (ex: create a “write” edge from your Lambda function to a DynamoDB table). Then, with the simple click of a button, Agility will transform your architecture into a deployable yaml template using a conversion algorithm. Lastly, using just a single command, deploy your application!

Value Proposition

Our application will accelerate the process of planning, developing, and deploying serverless applications at scale. Currently, design diagrams are developed with tools such as Google Drive and draw.io, then undergo review from management, then must be programmed into CloudFormation configuration files. These configuration files are repetitive and it’s challenging, even for experienced developers, to properly implement proper security measures. Not only do the files contain repetitive configuration, they don’t include any additional information from the service diagram, making them redundant. Our application captures the end-to-end workflow experience, allowing teams to collaborate on architecture diagrams, and deploy their applications securely with a single command.

Market Research

According to MarketsandMarkets, the global serverless architecture market size is projected to grow from 7.6 billion USD to 21.1 billion by 2025. This can also seen in the graph in which we can also see that there is a 22.7% Compound Annual Growth Rate between 2018 and 2025. North America, the geography we are targeting, accounts for the largest market size during the forecast period. As the serverless market continues to grow, the need for a tool that helps translate serverless design diagrams into CloudFormation configuration files will grow correspondingly.



Engineering Innovation

There are a few primary areas of engineering innovation that this project involves. The first is the collaborative editing on the frontend of the application. Different users have the ability to open shared diagrams and do live collaboration with each other. This is similar to Figma and uses sockets to transfer information between the users.

The second is the conversion algorithm that our backend uses to convert diagrams into CloudFormation yaml templates. Each diagram is stored as a graph, a series of vertices and edges. Our algorithm traverses this graph and produces a minimum CloudFormation templates which contain resources representing the vertices and connectors representing the edges.

Lastly, we utilize the fact that it takes a single command (sam deploy –guided) to deploy a serverless application with the appropriate yaml files to empower the user to deploy their applications with ease.

Stakeholders

Customer Segments

1. Agile Teams
    a. Student Teams
    b. Startup Companies (not enterprise yet)
2. Developers with minimal dev-ops experiences

Partners

Cloud Providers: Cloud providers such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure are a key partner for our product. Our plan is to start with Amazon Web Services as it is currently the most commonly used cloud platform. Users who are creating CloudFormation configuration files must use a cloud provider and our product increases ease of use for our customers to integrate with CloudFormation thereby reducing friction for users of cloud providers.

Competition

Amazon has a tool for designing CloudFormation templates, but it doesn't target the same market segments as our product. Their tool is overly complex, and likely only possible to use if one is already a seasoned AWS engineer. Their product suffers from information overload, making it challenging to accomplish tasks that serverless is commonly used for. Additionally, their product doesn't handle security configuration automatically, instead leaving that to the user to define.

Another option is for developers to use tools such as Terraform or CloudFormation and write the configuration files themselves. This provides more control to the developer, but involves redundant processes, is not well-suited for team collaboration, and is error-prone.

Another competitor is Juju GUI for OpenStack. This provides a GUI for editing architecture diagrams, but is targeted at the OpenStack cloud computing platform. This platform is open-sourced, but has not seen widespread adoption in the same way that AWS has. Although they have a good core product, it doesn't translate to a business use case because AWS is still the industry standard for deployment. Additionally, they don't target serverless design, which prevents them from implementing optimization and automatic generation of certain aspects of configuration.

Revenue and Cost Structure

Our primary revenue stream is a small fixed percentage on the deployment for managing it. This fee could be <1% can be determined exactly based on similar products and further market research. As deployments are necessary and the number of them continue to grow, this will ensure a constant stream of income.

Our application will not incur any costs from the user's deployments, so our only costs are those associated with deploying our own application. Although this depends on the number of users and number of requests that our web application needs to serve, total costs of $20/month would be a very loose upper bound on cost. Our project requires no physical assets, so the only incurred

costs are monthly deployment costs, and potentially also a one-time fee for domain name registration.