Serverless, a cloud native approach to building APIs

Modern cloud-first solutions have become what technology companies today see as their primary resource to get to market faster, scale more effortless, and offload expensive procurement of vital infrastructure. The largest company that offers these products is Amazon Web Services (AWS). One such product, AWS Lambda, allows for what AWS calls infinitely scalable compute processes and should prove to give any company the ability to offload processing to these services at a fraction of the cost. Introduced in 2014, Lambda is an event-driven serverless computing platform that ties into many other service offerings by AWS. AWS also defines this as the Serverless Application Model, a micro service event driven application that is distributed across the AWS footprint. One such application of this technology is the ability to build a public REST API and not be concerned with what programming language to use, a lengthy deployment time, and limiting the over or under capacity planning.

AWS Lambda is a serverless compute platform. Developers “can focus on [their] core product and business logic instead of responsibilities like operating system (OS) access control, OS patching, provisioning, right-sizing, scaling, and availability."\*\*\*{quote this -- https://d1.awsstatic.com/whitepapers/serverless-architectures-with-aws-lambda.pdf }. In short, serverless means developers can write and execute code without managing or configuring the underlying servers. A developer's primary focus is only writing code. AWS handles all compute and infrastructure in the background, and while you may lose some flexibility in some respects, applications only get billed when your code is executed. If you only use 1 ½ seconds of computer time, that is all you are billed for. A best practice for a piece of lambda code is to follow the SOLID principle of single responsibility. Building each Lambda as a single function that only handles one type of request or event trigger. A lambda functions lifecycle starts at some event trigger, the lambda function then accepts a JSON payload and will either respond to the requestor or pass along the payload to another service, and then the function completes. Most operations run in milliseconds, but there is the ability to extended process time to a maximum of five minutes. Another amazing feature of this service is that this function is spread across many locations allowing for concurrent requests to be handled without having to worry about a traditional server overload. Other AWS services initiate these triggers. One such trigger is API Gateway, a routing service that will allow a lambda to process and respond to HTTP requests. AWS also offers a command-line tool called SAM, which is the Serverless Application Model. The SAM tool allows the developer to create a cloud-formation template that provisions and deploys any AWS service and its corresponding event triggers.

An issue that we have discussed is how do we build our new API with less dependency on infrastructure and achieve on-demand scalability that saves the company money. Our standard Rest API, a massive monolith, is slow to change, hard to scale, and lacks testing. As we rebuild our public API, the focus for our company should be the ability to build quickly while also not having to worry about how this could impact stakeholders down the road. With the combination of API Gateway and Lambda, we will create small sections of the API at a time, focusing on quality.