In your initial post for this discussion, compare the Lambda compute model against one of the following traditional compute models: EC2, containers, virtual machines, or native hardware.

**compare the Lambda compute model against one of the following traditional compute models: EC2, containers, virtual machines, or native hardware.**

In consideration of the Lambda compute model, we should remember that the Lambda compute model manages all of our OS, network, security settings, and the application stack. Lambda does not expose these underlying features, merely letting the user focus on writing their code – given the runtime language, what code to run, and a handler method in say, NodeJs, AWS can take a request and pass it into an AWS-managed compute infrastructure, and upon puling the code, a new execution environment is spun up; upon passing in the Event and everything needed for execution to the compute environment via our Lambda function, our code will be executed without need for knowledge of our infrastructure.

Lambda functions can also utilize docker container images – code can be packaged with dependencies as a container image (using docker cli), then the image can be uploaded to a container registry hosted on Amazon elastic container registry (Musgrave).

EC2 (Amazon Elastic Compute service) provides scaling computing capacity in virtual computing environments called “instances” which reveal more of the underlying server infrastructure than a Lambda. If you need to manage your own compute resources, consider Amazon Elastic compute as you can customize the OS, the network and security settings, and the entire software stack (unlike Lambda). In this sense, Lambdas are best suited for shoter, event-driven workloads as they can run for up to 15 minutes per invocation, whereas EC2 is used if you need fine-grain tuning over you server instance (which is abstracted away when using a lambda).

Amazon’s Elastic Beanstalk offering is one layer of abstration away from an EC2 instance as Beanstalk will setup an environment for you containing a number of EC2 instances, with an optional database, load-balancer support, auto-scaling group, and a security group.

The key difference between virtual machines and containers is that virtual machines are emulating hardware, whereas containers have direct kernel access resulting in faster spin up size and smaller instances. Containers provide a layer of abstraction, and automation of OS level virtualization on Linux (Srivastav).

Srivastav, Prakhar. “A Docker Tutorial for Beginners.” A Docker Tutorial for Beginners, https://docker-curriculum.com/#getting-started.

MUSGRAVE, DAVID. “Lambda.” Amazon, EUROPA EDITIONS, 2022, https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

In posting responses to your classmates, consider the following:

Identify the key points.

Explain why you agree or disagree with the statements made by your peer.

4-2 Discussion: The Value of Gateway

In your initial post for this discussion, explain three to four features and benefits of using Amazon API Gateway. Explore and write about topics such as routing, load balancing, firewall protection, and so on.

The primary purpose of the AWS Amazon Gateway, is to map incoming API calls to a service to a service, handling the API call. REST is one protocol it supports. AWS Gateway is what triggers the lamba functions, telling them to start running. API Gateway can also act as a “security barrier”, meaning only the outside world can access the APIs we deploy, and the API Gateway is the only service with permissions to call our Lambdas. According to the article, “What is an Amazon API Gateway”, API Gateway provides authentication mechanisms such as AWS Identity, Access Management policies, Lambda authorizer functions, and Amazon cognito user pools.

Another key benefit is the ability to access API Gateway from either AWS SDKs + Gateway commands, AWS Management Console (web interface), older API Gateway versions, AWS CLI, or AWS tools for Powershell. API Gateway provides a scalable and secure by default gateway for web APIs that route HTTP requests to Lambda functions. AWS Gateway can be used to create a unified API front-end for several microservices, with DdoS protection, authentication and authorization for requests to the backend, and throttling of API usage. We can utilize API Gateway to handle app routing with the usage of path variables to route any incoming API requests to backend resources (Working with Routes for HTTP Apis). Gateway Load balancer can help by distributing traffic across multiple virtual instances with the ability to scale up or down on demand. AWS Firewall manager can help roll out firewall rules across Load balancers, API Gateways, and CloudFront distributions – Amazon ensures compatibility and interconnectedness between most of these AWS service offerings.

Working with Routes for HTTP Apis - Amazon API Gateway. https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-develop-routes.html.

What Is Amazon Api Gateway? - Docs.aws.amazon.com. <https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>.

SCREENSHOTS 4.1

First serverless compute function

Execution results

EchoWithQuery execution results

Screenshot or JSON for the created test event “EchoWithQuery”

Source code or screenshot for index.js (about 22 lines of code)

Screenshots 4.2

Screenshot of the GET method execution page

Screenshot of the GET test page

Screenshot of the browser running the URL

The invoke URL from your deployment