ASSIGNMENT 1 **AVI AJMERA**

1. Serverless Computing: The Big Three Cloud Providers

Among the plethora of cloud providers offering serverless solutions, the big three - Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) - stand out due to their comprehensive services, global reach, and innovative solutions. Each platform brings its unique features, pricing models, and ecosystem integrations.

Parameter/Platform	**Amazon Web Services (AWS Lambda)**	**Microsoft Azure (Azure Functions)**	**Google Cloud Platform (Google Cloud Functions)**
Overview	AWS Lambda enables code execution without server provisioning, automatically scaling with workloads.	Azure Functions is an event-driven serverless compute platform integrated with Azure services.	Google Cloud Functions allows code execution in response to events without server management.
Age	11 years old	5 years old	6 years old
Notable Users	Coursera, Netflix, Airbnb, Coca Cola	Bosch, Starbucks, FedEx, Walmart	Toyota, Spotify, Twitter, Paypal
Pricing Model	Per second with a 60-second minimum	Per-minute basis	Per-minute basis
Primary Compute Service	EC2	Azure Virtual Machines	Google Compute Engine

Storage Characteristics	Transient storage with services like S3 and persistent with EBS	Transient with ID drives. Object Storage with Blobs and Files	Both transient and persistent disks. Object Storage with Google Cloud Storage
Key Tools	Gluon for AI/ML, SageMaker, AWS Elastic Beanstalk	Cognitive services (Bing Web Search API, Face API), Support for Microsoft software	TensorFlow for AI/ML, APIs for natural language, speech, translation
Hybrid/Multi-cloud Options	Amazon ECS Anywhere, AWS Snowball, AWS Outposts	Azure Arc, Azure Backup, Azure Blob Storage	Anthos, Traffic Director
Pros	Most mature, Reliable, Vast computational capacity	Easy integration for Microsoft services, Affordable, Strong hybrid cloud support	Integration with Google services, Advanced AI tools, Container support
Cons	Overwhelming options, Fewer hybrid cloud alternatives	Fewer service choices, Designed for business	Limited services compared to AWS and Azure, Limited enterprise support

The Evolutionary Journey of AWS Lambda

The evolutionary trajectory of Amazon Web Services' serverless compute service AWS Lambda from its inception to its current state, underscoring its significance in shaping the serverless landscape.

1. The Dawn of Lambda (2014-2016):

In 2014, the cloud world buzzed with excitement as AWS unveiled Lambda, initially supporting just Node.js. Over the subsequent two years, AWS expanded Lambda's horizons, incorporating support for languages like Java, Python, and C#.

2016 marked a significant enhancement with the introduction of "Scheduled Events," ushering in cron-like functionalities to the serverless realm.

2. Strengthening the Foundation (2017-2018):

The subsequent phase in Lambda's journey was characterized by expansion and integration. 2017 saw the advent of Lambda@Edge, a feature that enabled Lambda functions to operate at CloudFront edge locations.

Simultaneously, AWS augmented Lambda's language support, adding Go and .NET Core 2.0 to the roster. The integration of AWS Step Functions was another highlight, offering developers an avenue to create complex workflows with Lambda at the helm.

2018 further solidified Lambda's position in the serverless sphere. AWS unveiled Lambda Layers and the Lambda Runtime API. While Layers promoted code reusability, the Runtime API was a game-changer, allowing developers to introduce their own language runtimes to the platform.

3. Enterprise Adaptations (2019-2021):

With serverless computing gaining traction in enterprise ecosystems, Lambda underwent transformations to cater to this demographic. AWS introduced Provisioned Concurrency in 2019, a feature that addressed the perennial "cold start" dilemma, enhancing Lambda's enterprise appeal. The integration of Lambda with Amazon EFS the following year marked another milestone, optimizing file-based workloads.

4. Innovations and Beyond (2022+):

Venturing into 2022 and beyond, AWS showcased its commitment to innovation with a slew of features:

- Lambda SnapStart: An enhancement aimed at Java functions, boosting their cold start performance.
- Function URLs: Simplifying function invocation with dedicated HTTP(S) endpoints.
- Lambda States: Facilitating intricate workflow orchestration involving multiple Lambda functions.
- Lambda Layers & Runtime API: A continued focus on code reusability and language runtime adaptability.

Furthermore, AWS's strategy to intertwine Lambda with its AI and ML services, such as Amazon SageMaker and Rekognition, underscores its vision for a cohesive and powerful serverless ecosystem.

New Feature Suggestions for AWS Lambda as a Prospective Product Manager

Having dabbled with AWS Lambda after our recent introduction to the serverless, I've identified several potential enhancements that could significantly benefit users, especially those new to the serverless ecosystem. Here are my feature suggestions:

User-Friendly Interface:

- The current dashboard, though comprehensive, can be daunting for beginners.
- A more visual approach, perhaps with drag-and-drop functionalities and color-coding, might make the initiation process smoother.

Sharing Capabilities:

- Sharing my Lambda functions with peers turned out to be quite a task.
- A simple sharing mechanism, akin to sharing a link for a Google Doc, would be immensely beneficial.

Predefined Setups:

- Integrating Lambda with other AWS services feels like a complex puzzle.
- Offering templates or quick setups for common tasks could be a game-changer for those still getting their bearings.

Integrated Community Space:

- A platform within Lambda where users can:
    - Ask questions and seek guidance.
    - Share their creations and learn from the innovations of others.
    - Engage in discussions to broaden their understanding.