Following is the table discussing Amazon Web Services(AWS), Google Cloud Platform(GCP) and Microsoft Azure's computing platform:

Parameter	**AWS (Lambda)**	**Google Cloud (Cloud Functions)**	**Azure (Azure Functions)**
Initial Release	2014	2016	2016
Supported Languages	Node.js, Python, Ruby, Java, Go, .NET Core, and more	Node.js, Python, Go, Java, .NET Core, Ruby, PHP	.NET Core, Node.js, Java, Python, TypeScript, and more
Scaling			
Automatic Scaling	Yes	Yes	Yes
Max Execution Time	15 minutes	9 minutes	5 minutes (consumption plan)
Pricing			
Free Tier	1M requests per month and 400,000 GB-seconds of	2M requests per month	1M requests and 400,000 GB-seconds

	compute time per month		of execution per month
Integration			
Integrated Services	Many AWS services like S3, DynamoDB, etc.	Many GCP services like Cloud Storage, Pub/Sub, etc.	Azure services like Cosmos DB, Event Hub, etc.
Development			
Local Testing	AWS SAM (Serverless Application Model)	Local Emulator	Azure Functions Core Tools
Security			
IAM	AWS Identity and Access Management	Cloud IAM	Azure Active Directory
Cold Start Latency	Varies based on language, memory, and VPC. Java and VPC-connected functions tend to have	Typically faster for Python and Go, slower for Node.js and Java.	Varies; .NET Core tends to be faster. Premium plan offers pre-warmed instances.

	higher cold starts.		
Deployment & Packaging	Supports AWS SAM, CloudFormation, AWS CLI, and Serverless Framework. Deployment package size limit is 50MB (zipped) and 250MB (unzipped).	Supports gcloud CLI, Firebase tools, and ZIP uploads. Deployment package limit is 540MB.	Supports Azure CLI, Azure Portal, and Visual Studio. Limit is 1GB for deployment package.
State Management	No built-in state management. Integration with DynamoDB, ElastiCache, and others for maintaining state.	No built-in state management. Integration with Cloud Datastore, Firebase, etc.	No built-in state management. Can use Azure Cosmos DB, Azure Cache, etc.
Concurrency & Throttling	Default soft limit of 1000 concurrent executions per region (can be increased). Throttling occurs	No explicit limit mentioned, but there are quotas on the number of concurrent connections.	Default of 200 instances. Each instance can handle multiple invocations

	beyond this.		
Diagnostics & Monitoring	Integrated with CloudWatch for logs and monitoring. X-Ray for distributed tracing.	Stackdriver for logs and monitoring.	Integrated with Application Insights for monitoring, diagnostics, and logging.
Error Handling & Retries	Supports automatic retries. Dead Letter Queues (DLQ) for failed asynchronous invocations.	Automatic retries can be configured. No native DLQ, but can be implemented using Pub/Sub.	Supports retries and failed messages.

Considering Amazon Lambda, here’s how it evolved since its inception in 2014:

2014:

- November: AWS Lambda was launched at re:Invent 2014. Initially, it only supported Node.js.
- Allowed running code in response to events from Amazon S3, DynamoDB, Kinesis, and more.
- Automatic scaling and built-in fault tolerance.

2015:

- Introduced support for Java.
- VPC support was added, allowing Lambda functions to access resources inside a Virtual Private Cloud.
- Increased the maximum execution timeout.
- Amazon API Gateway was introduced, which can trigger AWS Lambda functions.

2016:

- Introduced support for Python.
- Released AWS Serverless Application Model (SAM) for defining serverless applications.
- Introduced environment variable support for Lambda functions.

2017:

- Increased the default safety limit for concurrent executions.
- Introduced support for .NET Core.
- Added support for Go.
- Launched AWS Lambda@Edge for running Lambda functions at CloudFront edge locations.

2018:

- Introduced support for Ruby.
- Increased maximum execution time to 15 minutes.
- Increased the payload size limit for asynchronous invocations.
- Released AWS Lambda Layers for code sharing and separation of responsibilities.

2019:

- Released Provisioned Concurrency, allowing for consistently low latency for applications at any scale.

- Introduced support for custom runtimes, allowing developers to bring in their own execution environments.

2020:

- Introduced Amazon EFS integration, allowing Lambda functions to access shared file systems.
- Launched Lambda Extensions Preview, offering a new way to integrate Lambda with operational tools.

2021:

- Introduced Lambda Insights for monitoring, troubleshooting, and optimizing Lambda applications.
- Improved the Lambda Console experience for better configuration and testing.
- Continued to integrate with many AWS services and improve the developer experience.

2022:

- Lambda now supports granting permissions to an organization in AWS Organizations
- Lambda now supports sharing test events with other users in the same AWS account.
- Lambda now supports function URLs, which are dedicated HTTP(S) endpoints for Lambda functions.
- SnapStart to reduce startup time for Java functions without provisioning additional resources or implementing complex performance optimizations.

2023:

- Lambda releases asynchronous invocation metrics
- Lambda now supports streaming responses from functions
- Lambda now supports a new runtime for Ruby 3.2.