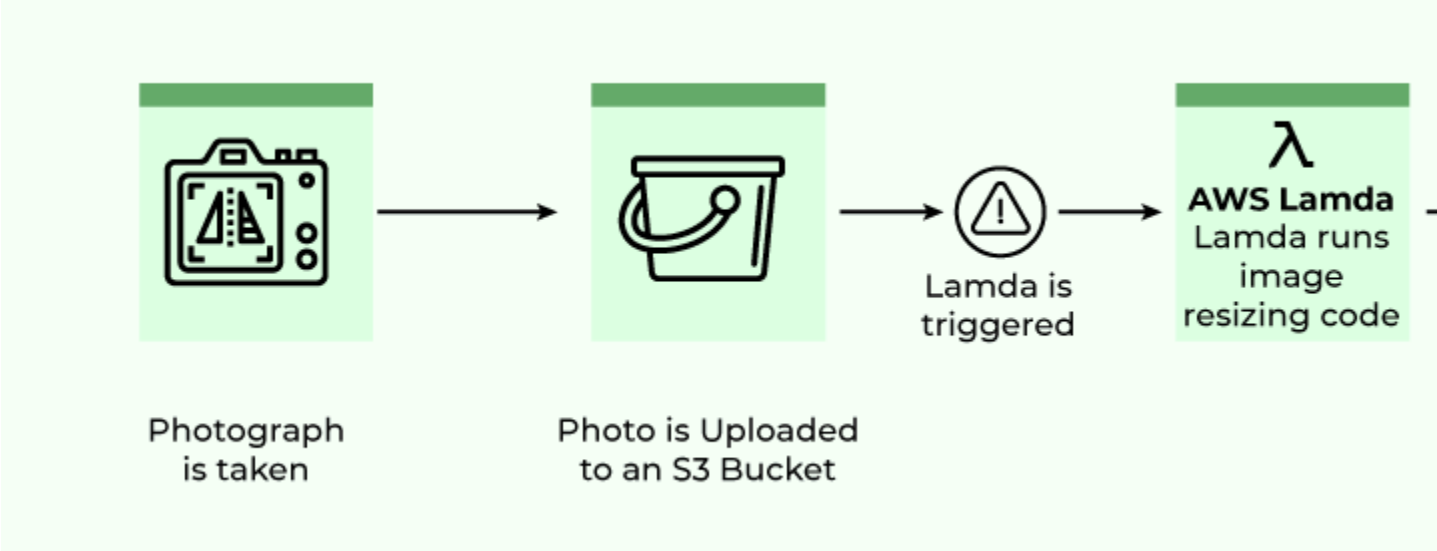
AWS Lambda

AWS Lambda is an Amazon serverless computing system that runs code and automatically manages the underlying computing resources like (EC2). It is an event-driven computing service. It lets a person automatically run code in response to many types of events, such as HTTP requests from the Amazon API gateway, table updates in Amazon DynamoDB, and state transitions. It also enables the person to extend to other AWS services with custom logic and even creates its own back-end services. For example, just write the code and then upload it as a .zip file or any container image. The service works by running code on high-availability computer infrastructure. It then performs all the administrative duties of that compute resource such as:



providing maintenance on the server and operating system, auto-scaling and managing the person's capacity provisions, handling security patch deployment lambda extensions code, monitoring, logging, and concurrency function blueprints. AWS lambda is very helpful when you know how to write code but you don't know how to provision the underlying infrastructure in AWS. AWS lambda will scale up the applications rapidly when there is sudden incoming traffic and scale down to zero when the incoming traffic is reduced.

Why Use Azure Functions

1. **Cost Efficiency**: Azure Functions follow a pay-as-you-go model. You only pay for the compute resources your function uses while it is running. This can be cost-effective for workloads that don't run continuously.
2. **Scalability**: Azure Functions automatically scales based on demand. This means that if your function needs to handle more requests, Azure will automatically allocate more resources without any manual intervention.
3. **Simplified Development**: You can write functions in various languages (such as C#, JavaScript, Python, and more) and focus on just the logic without worrying about the underlying infrastructure.
4. **Event-Driven**: Azure Functions are designed to respond to events. This can be a change in data, a timer, an HTTP request, or a message from a queue.
5. **Integration**: Azure Functions integrates well with other Azure services (like Azure Blob Storage, Azure Cosmos DB, Azure Event Hubs) and external services (through webhooks, APIs).

How to Use Azure Functions

1. **Create a Function App**: This is the container for your functions. You can create a Function App using the Azure Portal, Azure CLI, or Azure PowerShell.
2. **Define a Function**: Within the Function App, you can define individual functions. You specify the trigger for each function, such as an HTTP request, a timer, or a message from a queue.
3. **Write Your Code**: Write the code for your function in the language of your choice. Azure Functions supports languages such as C#, JavaScript, Python, Java, and PowerShell.
4. **Deploy**: Deploy your function to Azure. You can do this through the Azure Portal, Azure CLI, or by using a CI/CD pipeline from tools like GitHub or Azure DevOps.
5. **Monitor**: Use Azure Monitor and Application Insights to track the performance and health of your functions.

Where to Use Azure Functions

1. **Microservices**: When you need to implement microservices architecture, Azure Functions can handle individual service components efficiently.
2. **Data Processing**: For tasks like processing data as it arrives in a storage account or database, Azure Functions can react to events and process data.
3. **Automation**: You can use Azure Functions to automate tasks, such as sending notifications, processing files, or executing scheduled jobs.
4. **API Development**: Create serverless APIs that can handle various HTTP requests and provide responses without the need for managing server infrastructure.
5. **Integration**: Use Azure Functions to integrate different systems, services, or applications by responding to events and triggering workflows.

When to Use Azure Functions

1. **When You Need to Execute Code in Response to Events**: Ideal for scenarios where code needs to run in response to specific triggers, like file uploads or message arrivals.
2. **When You Have Intermittent or Irregular Workloads**: Use Azure Functions for workloads that don't run continuously but need to handle spikes in traffic or demand efficiently.
3. **When You Want to Focus on Code Rather Than Infrastructure**: Azure Functions abstracts away the infrastructure management, allowing you to concentrate on writing the function logic.
4. **For Prototyping or Proof of Concepts**: When you need to quickly prototype an idea or build a proof of concept, Azure Functions offer a fast and flexible development environment.
5. **For Cost Management**: When you want to manage costs effectively by only paying for the compute time your functions actually use, especially for sporadic workloads.