ConFoo.CA
DEVELOPER CONFERENCE

Introduction to AWS Lambda Functions

Serverless Computing with AWS

Beginner | Cloud | Python

Santiago Paiva
@SantiagoPaiva

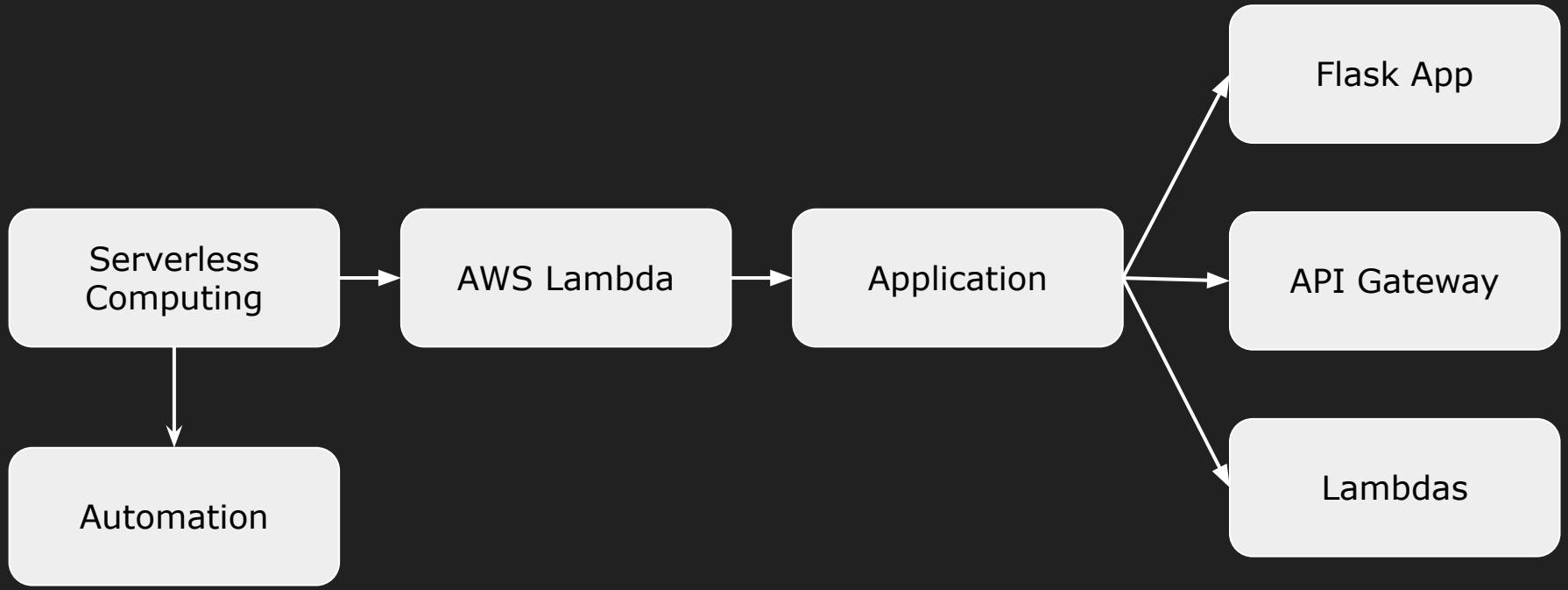
Slides & Source Code

github.com/**paiva**/**confoo-2024**

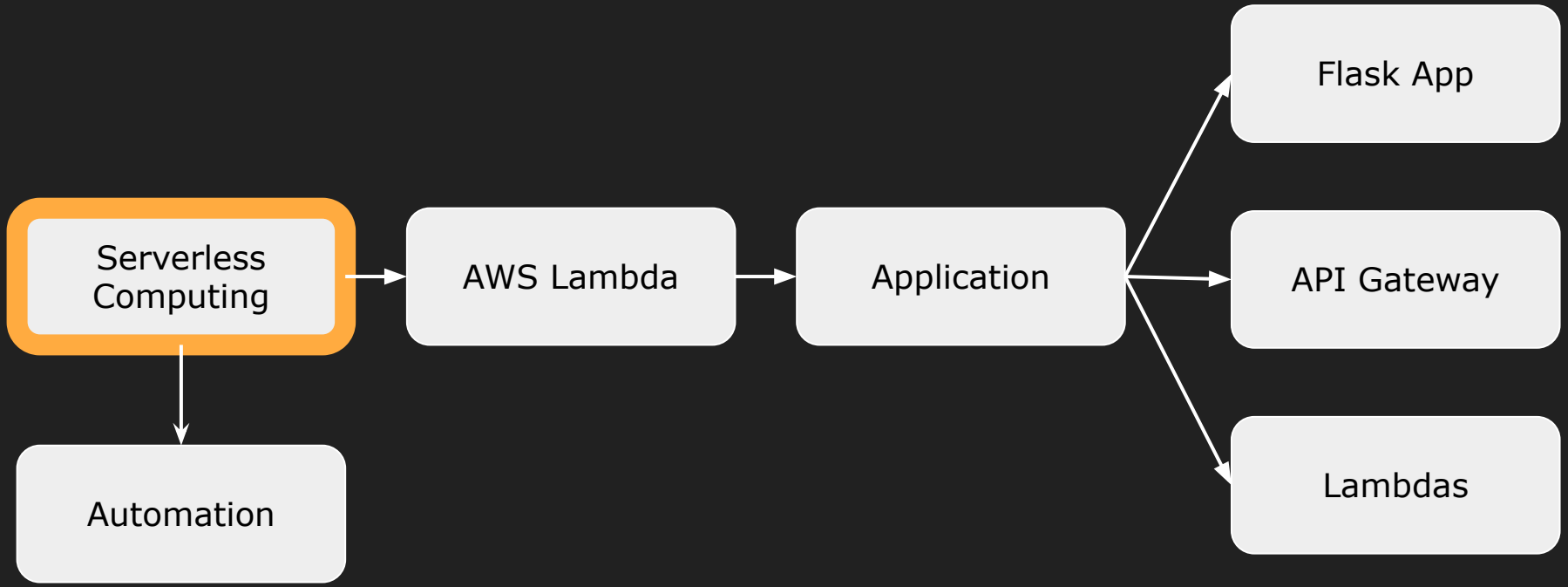
Presentation Overview

- Introduction
- Overview of Serverless Computing
- Automation with Serverless
- What is AWS Lambda?
- Use Cases of AWS Lambda
- How does AWS Lambda Works
- Demo: Flask Application with AWS Lambda
- Conclusion
- Q & A

Presentation Overview



Let’s start: Serverless Computing

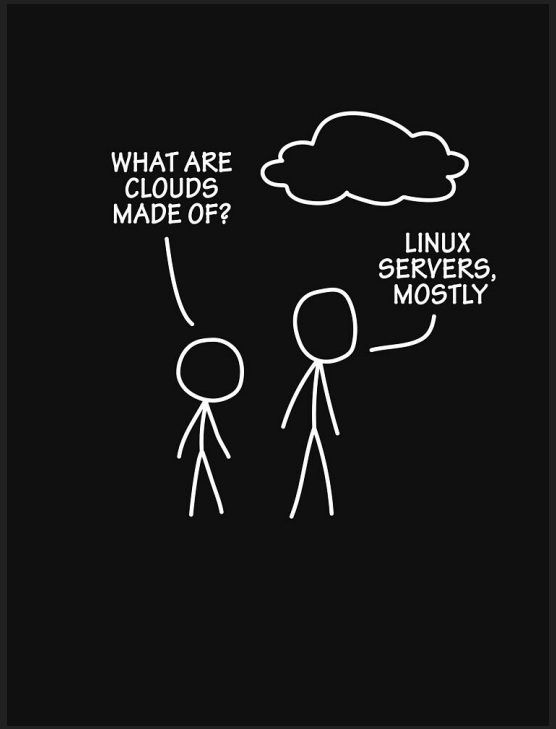


Overview of Serverless Computing

What is Serverless Computing?

“Serverless computing is a cloud computing model where the cloud provider dynamically manages the allocation and provisioning of servers. Developers focus on writing code without worrying about server management.”

Overview of Serverless Computing



Overview of Serverless Computing

Characteristics of serverless architecture

- **No server management**: Cloud provider handles server provisioning and maintenance.
- **Pay-per-use pricing**: Users are charged based on actual usage rather than provisioned capacity.
- **Auto-scaling**: Resources scale automatically based on demand.

Overview of Serverless Computing

Importance of serverless in modern Cloud Computing

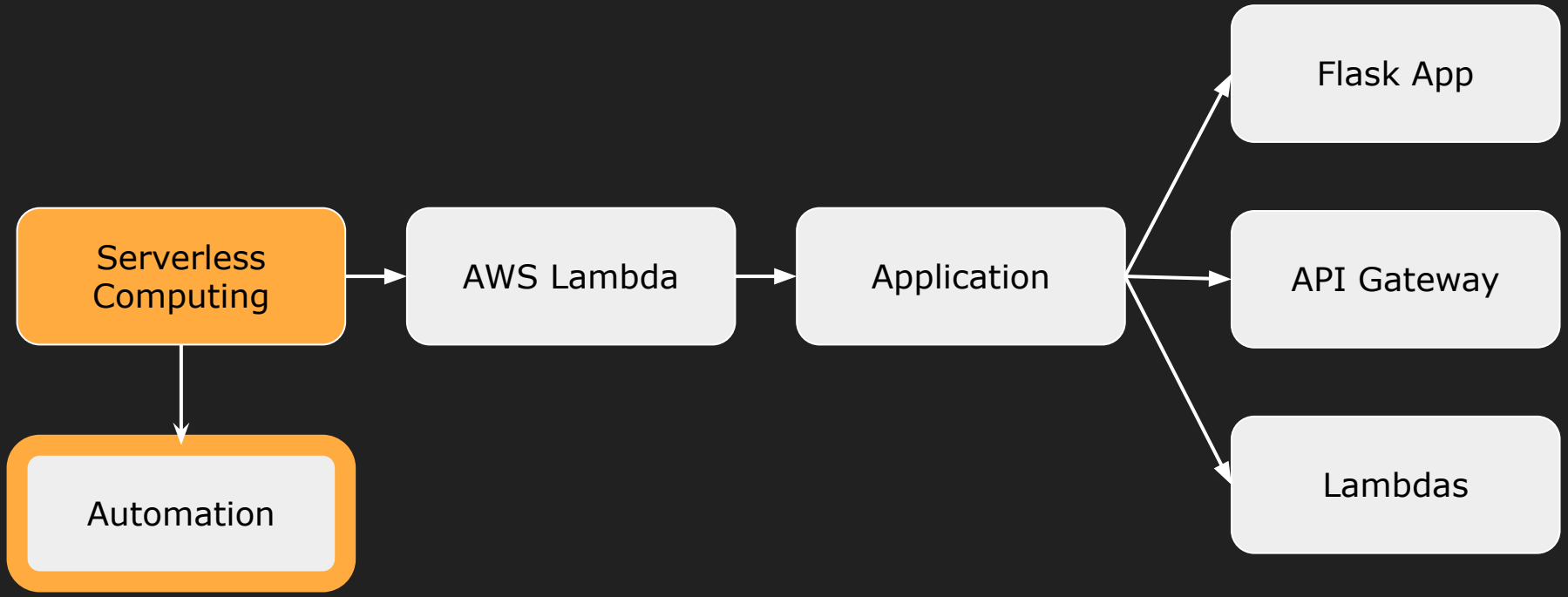
Serverless architectures offer:

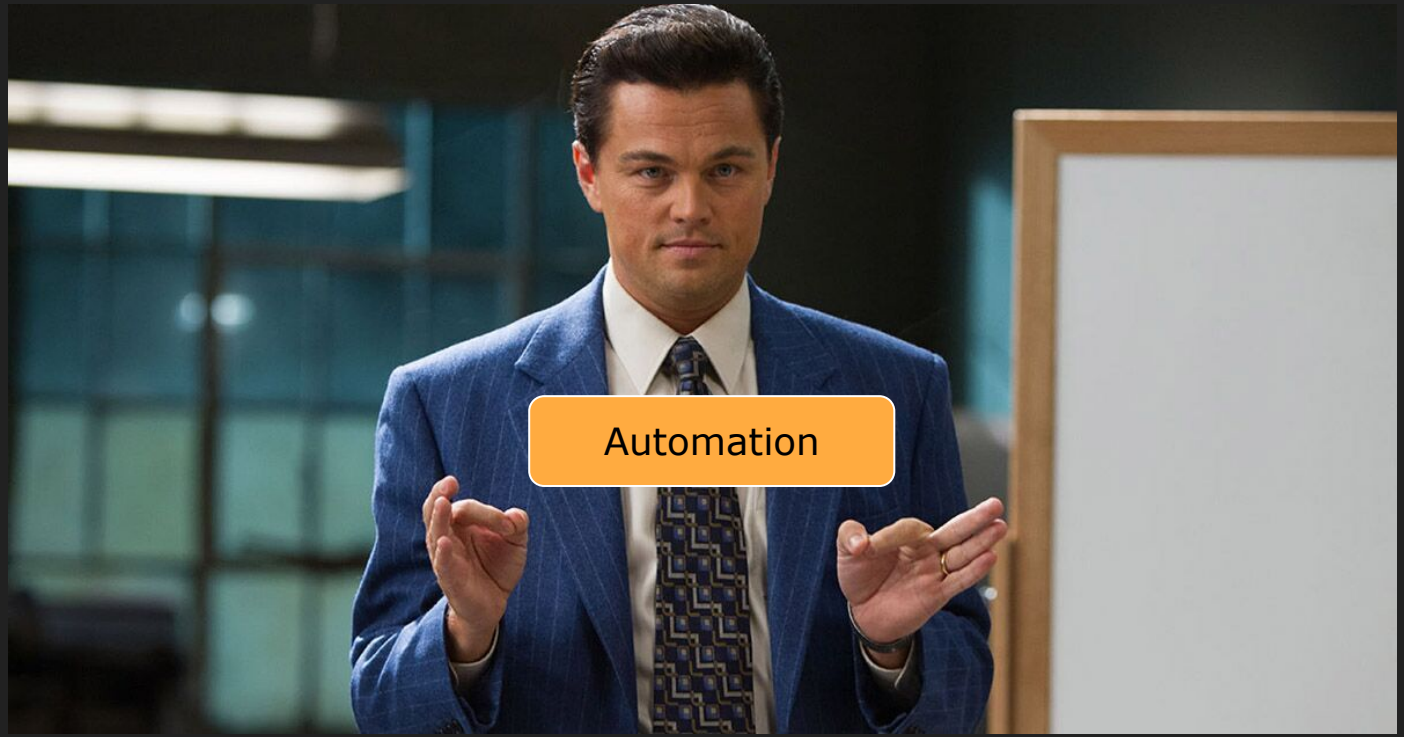
- **Agility**
- **Scalability**
- **Cost-effectiveness**

for modern application development

There are still servers in serverless!

Next: Automation




Automation

Automation with Serverless

1) **Serverless Architecture**: serverless functions operate on a serverless model, which means you don't need to provision or manage servers. This significantly reduces the overhead associated with server maintenance, patching, and administration, allowing you to focus on writing code that serves your business logic.
2) **Event-Driven**: serverless functions are inherently event-driven, making them ideal for automation tasks. This capability enables you to automate responses to specific changes in your environment, such as processing files uploaded to S3, reacting to database changes, or handling web request events via API Gateway.

Automation with Serverless

3) **Scalability**: serverless functions can automatically scale your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload, from a few requests per day to thousands per second. This makes it highly suitable for automation tasks that might experience variable loads
4) **Cost-Effectiveness**: with serverless functions, you pay only for the compute time you consume. There is no charge when your code is not running. For automation tasks that are sporadic or only need to run in response to specific events, this can lead to significant cost savings compared to running dedicated infrastructure 24/7

Automation with Serverless

5) **Rapid Development and Deployment**: serverless functions allow for quick iterations and deployments since you can easily update your function code to add or modify automation tasks. This agility is beneficial for evolving automation needs and experimenting with new automation strategies without extensive infrastructure changes.
6) **Customization and Control**: serverless functions provide a high degree of customization, allowing you to write functions in various programming languages such as Python, Node.js, Java, and Go. This flexibility lets you tailor your automation logic to your specific requirements and leverage existing libraries and SDKs.

Automation with Serverless

7) **Security and Compliance**: In the case of AWS Lambda: these serverless functions adhere to AWS's high standards of security, ensuring that your automation tasks are executed in a secure environment. It integrates with AWS IAM (Identity and Access Management), allowing you to set fine-grained access controls on your Lambda functions. Additionally, AWS Lambda is compliant with many compliance programs, which can be a crucial consideration for automation tasks in regulated industries.

Automation

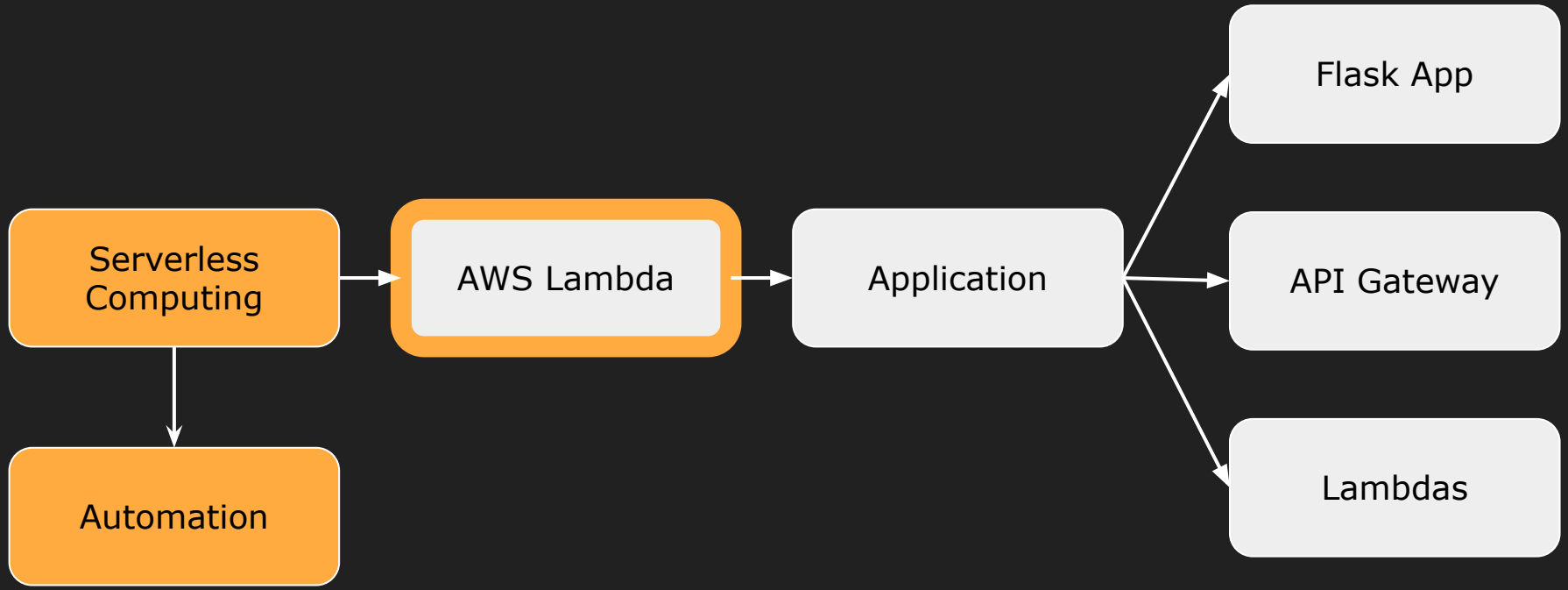
Different Cloud Providers for Serverless Automations

AWS
Lambda

Azure
Functions

Google
Functions

Next: AWS Lambda



What is AWS Lambda?

Definition of AWS Lambda

- AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS) that lets you run code without provisioning or managing servers
- It automatically scales to handle incoming requests and charges only for the compute time consumed

What is AWS Lambda?

Key features and Capabilities

- **Event-driven programming**: Lambda functions can be triggered by events from various AWS services or custom sources
- **Support for multiple programming languages**: Lambda supports popular programming languages like Node.js, **Python**, Java, and more
- **Integration with other AWS services**: Lambda seamlessly integrates with other AWS services such as S3, DynamoDB, API Gateway, etc

What is AWS Lambda?

Benefits of using AWS Lambda

- **Cost-effective**: Pay only for the compute time consumed by the function
- **Scalable**: Automatically scales to handle any workload
- **Reduced operational overhead**: Eliminates the need for server provisioning, maintenance, and scaling

What is AWS Lambda?

Lambda Pricing

AWS Lambda Pricing

Region: US East (N. Virginia)

Architecture	Duration	Requests
x86 Price		
First 6 Billion GB-seconds / month	$0.0000166667 for every GB-second	$0.20 per 1M requests
Next 9 Billion GB-seconds / month	$0.000015 for every GB-second	$0.20 per 1M requests
Over 15 Billion GB-seconds / month	$0.0000133334 for every GB-second	$0.20 per 1M requests
Arm Price		
First 7.5 Billion GB-seconds / month	$0.0000133334 for every GB-second	$0.20 per 1M requests
Next 11.25 Billion GB-seconds / month	$0.0000120001 for every GB-second	$0.20 per 1M requests
Over 18.75 Billion GB-seconds / month	$0.0000106667 for every GB-second	$0.20 per 1M requests

https://aws.amazon.com/lambda/pricing/

Use Cases of AWS Lambda (Examples)

Example 1

- **Real-time file processing**: Process files uploaded to S3 buckets in real-time

Example 2

- **IoT data processing**: Handle and analyze data from IoT devices

Example 3

- ***Web application backends***: Power serverless APIs and backend services for web applications

Use Cases of AWS Lambda (Examples)

Example 4

- ***Scheduled tasks and batch processing***: Run scheduled tasks or batch jobs at specified intervals

Example 5

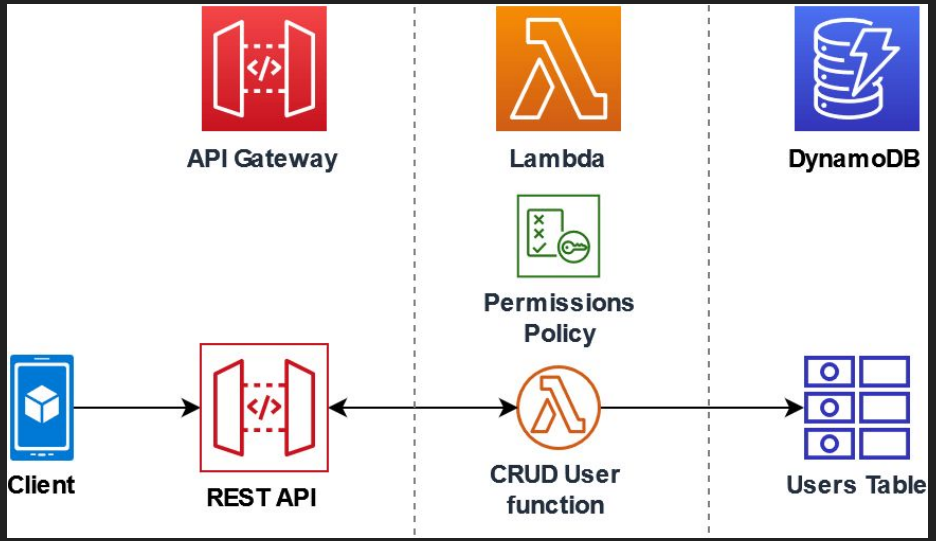
- **Chatbots and AI/ML applications**: Implement chatbots or perform real-time data analysis using machine learning models

Example 6

- **Mobile and web application backends**: Build scalable backends for mobile and web applications without managing servers

How AWS Lambda Works

Lambda functions are triggered by events, and AWS automatically provisions and manages the infrastructure to execute the code



How AWS Lambda Works

Each Lambda function consists of code, associated dependencies, and configuration.

[VS Code]

How AWS Lambda Works

API Gateway Trigger: For a function triggered by an API Gateway event, the event object would contain information about the HTTP request, such as the HTTP method, path, headers, query string parameters, and, if applicable, the request body.

```
{
  "httpMethod": "POST",
  "path": "/example/path",
  "headers": {
        "Content-Type": "application/json"
  },
  "queryStringParameters": {
        "param1": "value1"
  },
  "body": "{\"key\":\"value\"}"
}
```


SERVERLESS IS PERFECT
ME
SERVERLESS IS PERFECT
LIMITATIONS
ME

Limitations of Lambda Functions

- **Execution Time Limit**: Lambda functions have a maximum execution time limit, which, as of my last update, is set at 15 minutes.

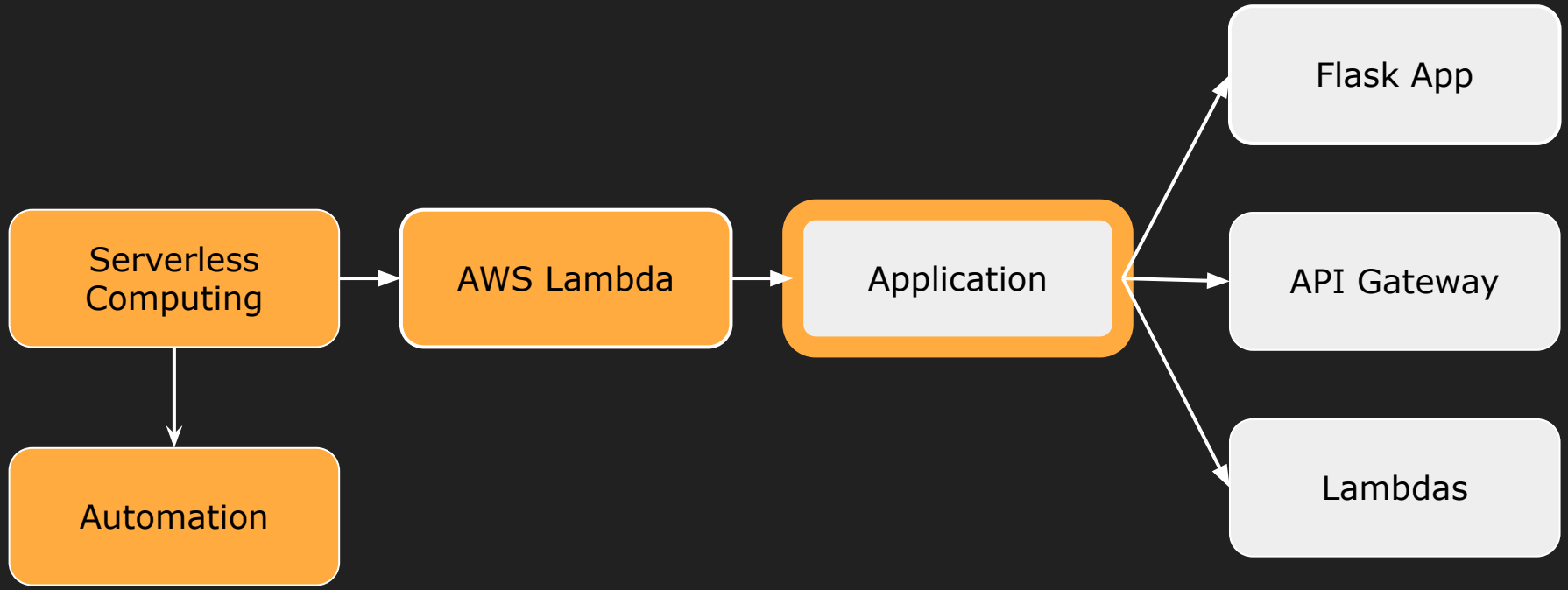
- **“Cold Starts”**: Lambda functions can experience what's known as "cold starts," which is the latency experienced when invoking a Lambda function after it has been idle for some time. This can be a concern for applications requiring consistent, low-latency responses

Limitations of Lambda Functions

- **Resource Limits**: AWS Lambda imposes limits on the amount of compute and memory resources available to each function. If your application requires more resources than Lambda can provide, it might not be the best fit

- **Continuous Processing**: Lambda is designed to respond to events and is not intended for continuous processing or long-running applications. Workloads that require continuous data processing might be better served by other compute services like EC2 or ECS

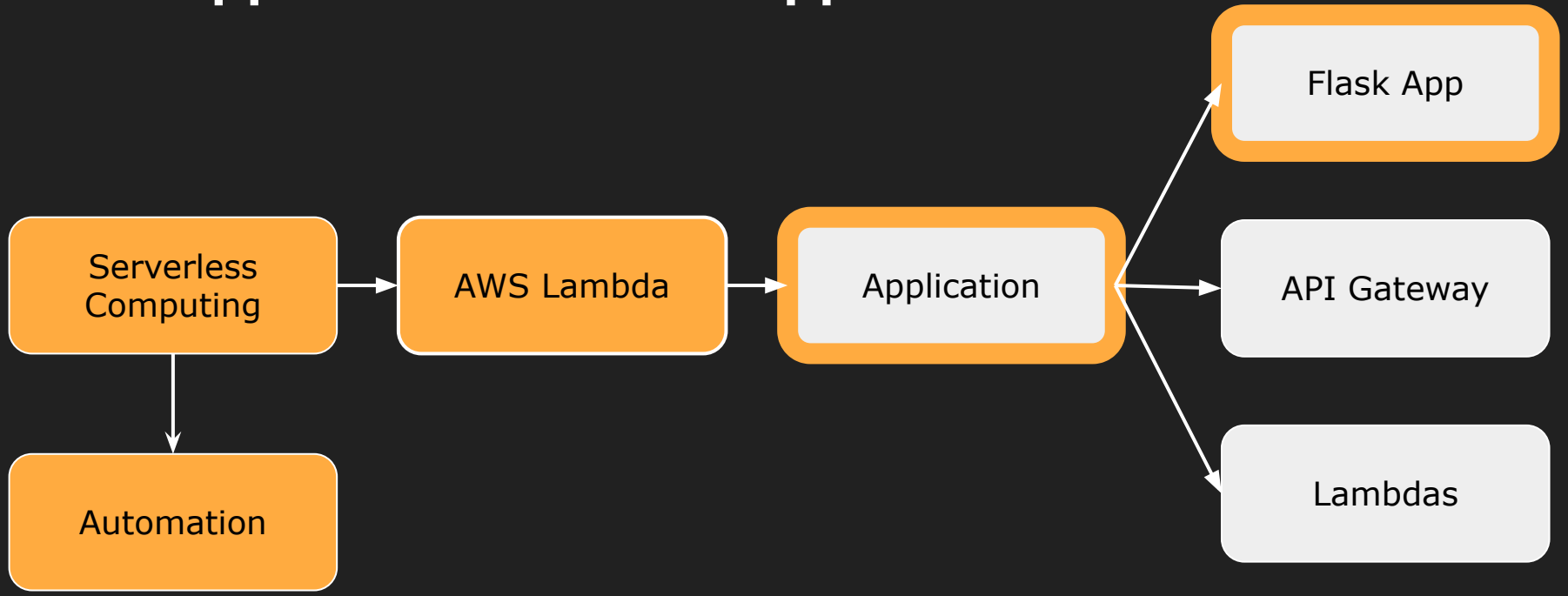
Next: Application (Demo)



Application

Personal Stand: *Not to convince anyone that this serverless architecture is the perfect solution for everything, but rather present the approach as a different way to decouple a web application*

Next: Application → Flask App



Application: Flask App

- The demo showcases a Flask application frontend communicating with an AWS Lambda function through API Gateway
- To simplify things, we will run Flask locally (Render deployment next time)

What is Flask?

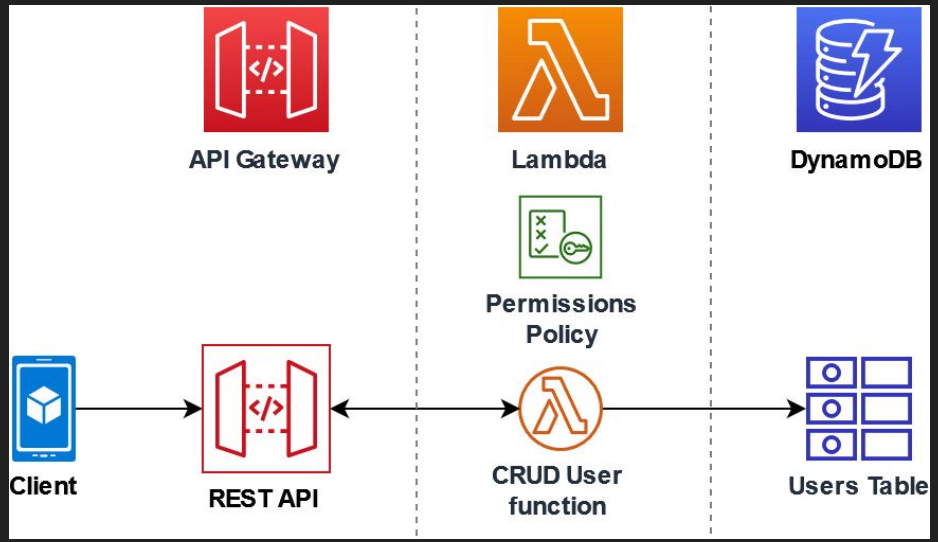
- Flask is a lightweight *micro* web application framework written in Python
- We will use the requests library to send a GET and POST request to the Lambda function
- Highly recommend Miguel Grinberg : **The Flask Mega-Tutorial**

Application: Flask App

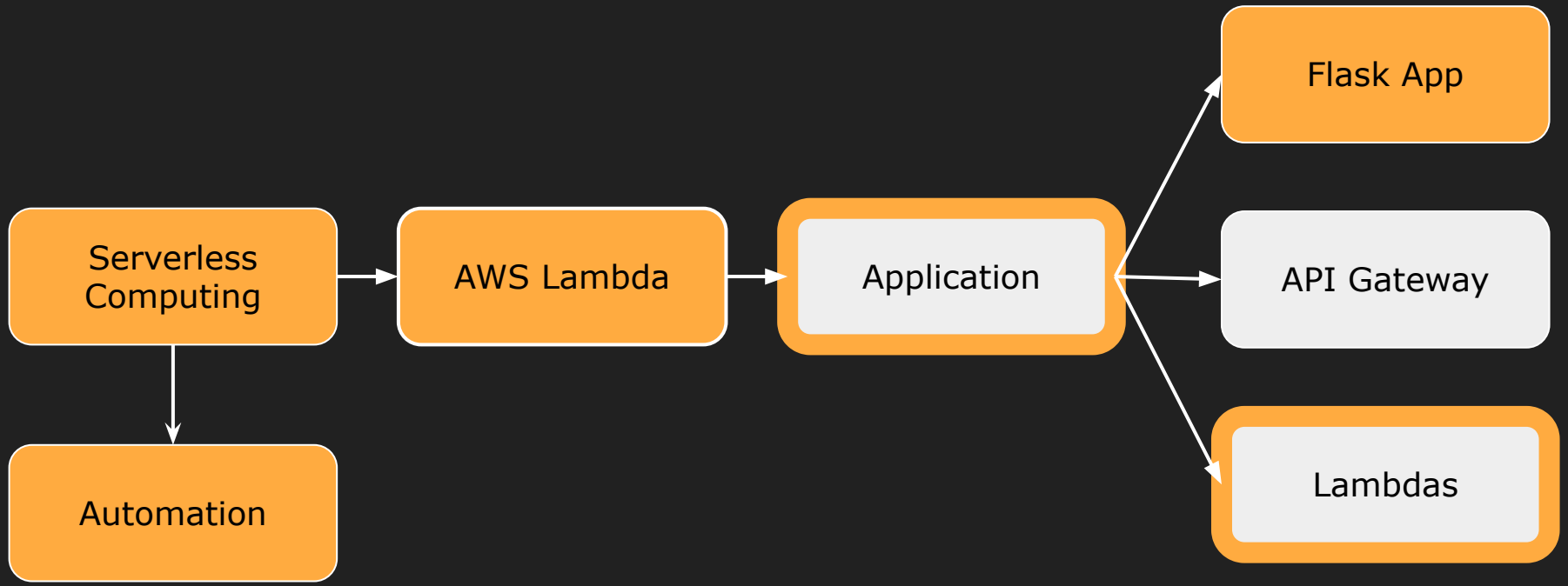
[VS Code]

Application: Flask App

- AWS Lambda function `api_get_users`: A serverless function hosted on AWS Lambda, processes the incoming JSON data and responds with JSON data back



Next: Application → Lambdas

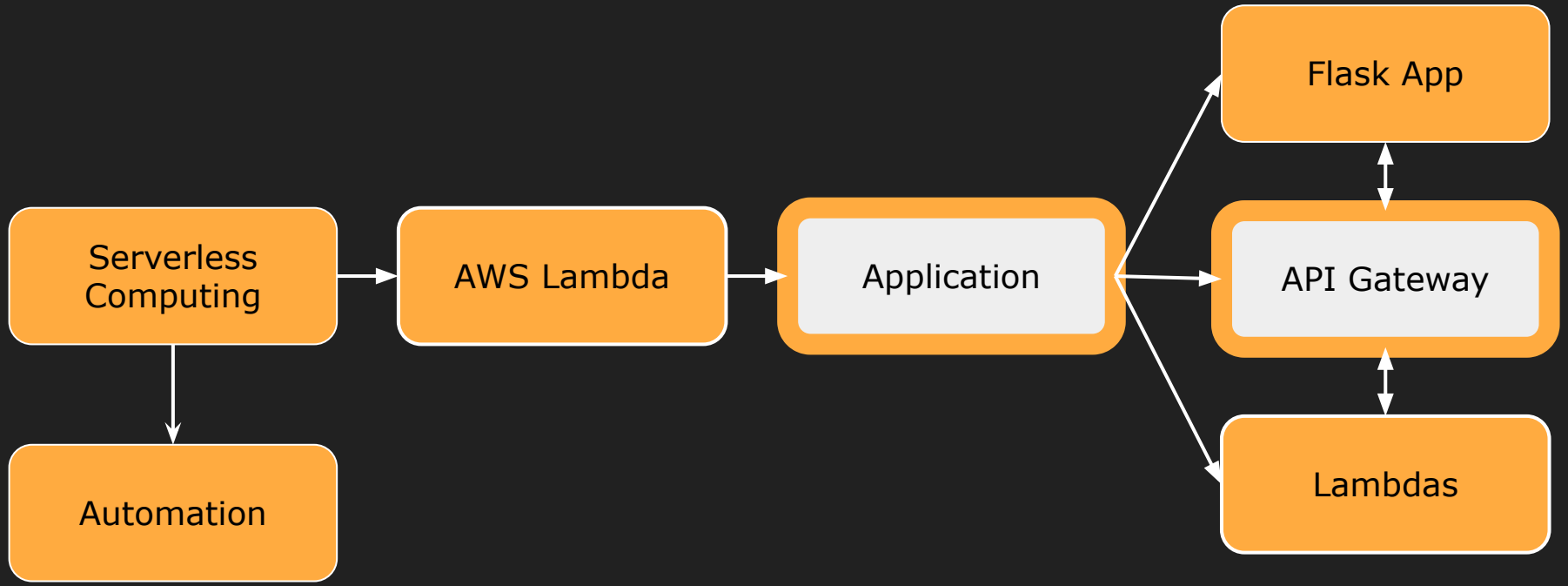


Let’s look at two Lambda Functions

- `api_get_users`: Gets data from the users table on DynamoDB [GET Request]

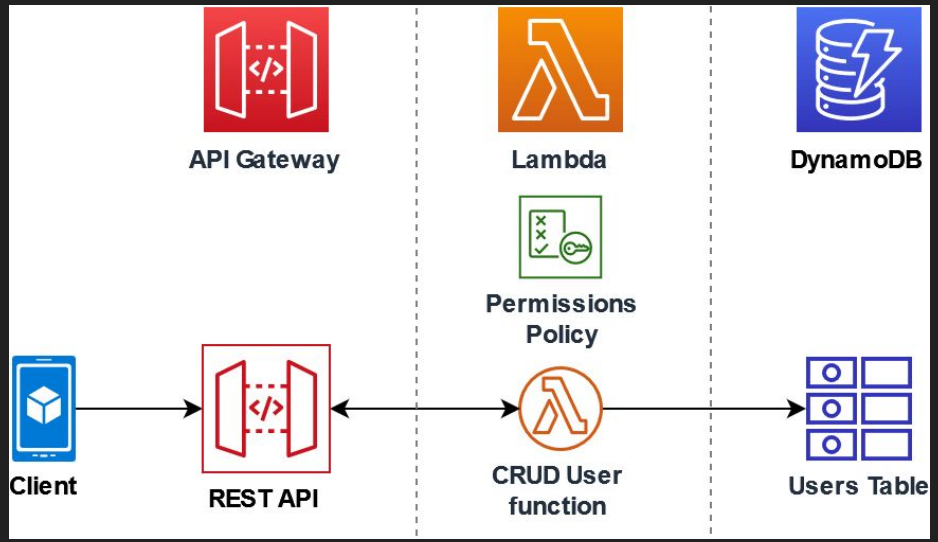
- `api_add_user`: Inserts data from Flask to the users table on DynamoDB [POST Request]

Next: Application → API Gateway



Final Step: The API Gateway

API Gateway for communication: API Gateway acts as an interface between the Flask application and Lambda function, enabling HTTP communication.



Conclusion

- AWS Lambda enables serverless computing by running code without managing servers.
- It supports multiple programming languages and integrates seamlessly with other AWS services.
- Lambda is ideal for various use cases, including web applications, IoT, and real-time data processing.

Conclusion

Benefits of using AWS Lambda:

- Cost-effective
- Scalable
- Reduces operational overhead

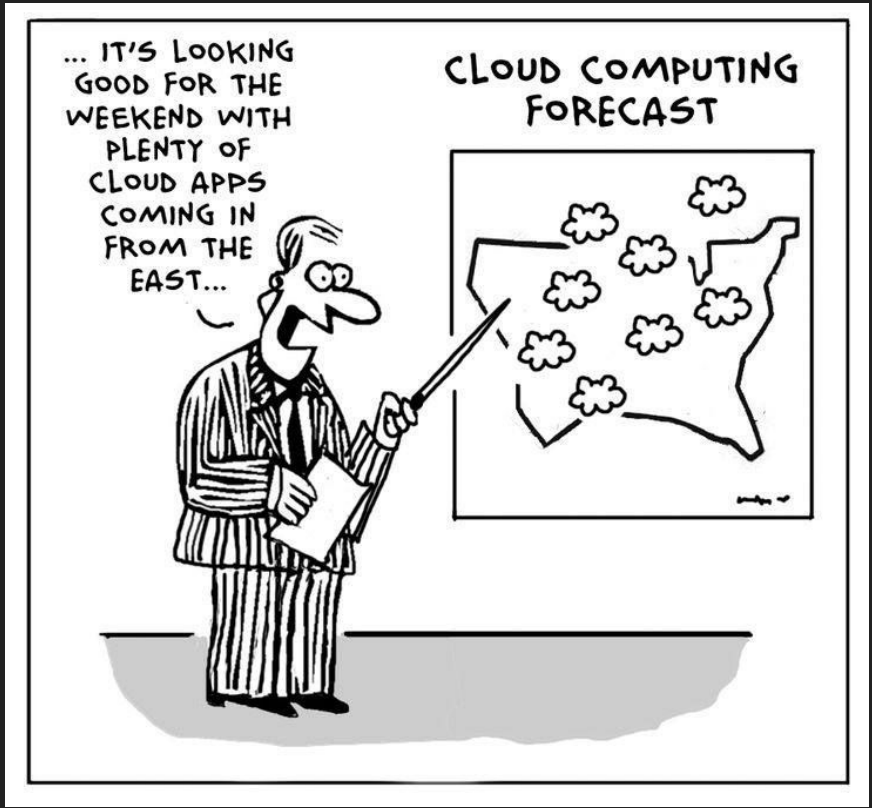
Limitation:

- 15-min timeout
- Possible “cold starts”
- Not meant for heavy computation

Slides & Source Code (Again)

github.com/**paiva**/**confoo-2024**

Q & A


... IT'S LOOKING GOOD FOR THE WEEKEND WITH PLENTY OF CLOUD APPS COMING IN FROM THE EAST...
CLOUD COMPUTING FORECAST