Building serverless applications in Python with AWS Chalice

About me

- Software Engineer at Hourly.

- Physics lover.
- Python lover.
- Teaching lover.
- Dogs lover.

About me

- Software Engineer at Hourly.

- Physics lover.
- Python lover.
- Teaching lover.
- Dogs lover.

Garoso

MANIZALES
TECH TALKS

Table of contents

01 Introduction to Serverless

What is Serverless?

- Serverless is a cloud-native development model that allows developers to build and run applications **without having to manage servers**.
- Serverless is a way to describe the services that enable you to build and run applications **without thinking about servers**.
- Once deployed, serverless apps **respond to demand** and automatically scale up and down as needed.

AWS

Google Cloud

Azure Cloud

IBC Cloud

Alibaba Cloud

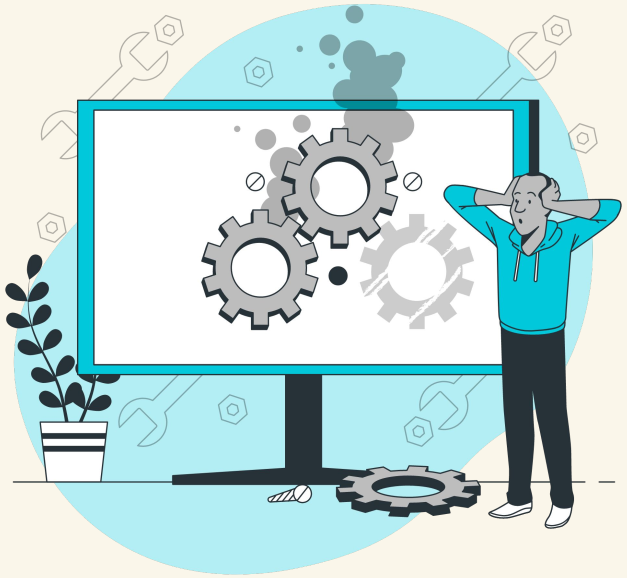
Other

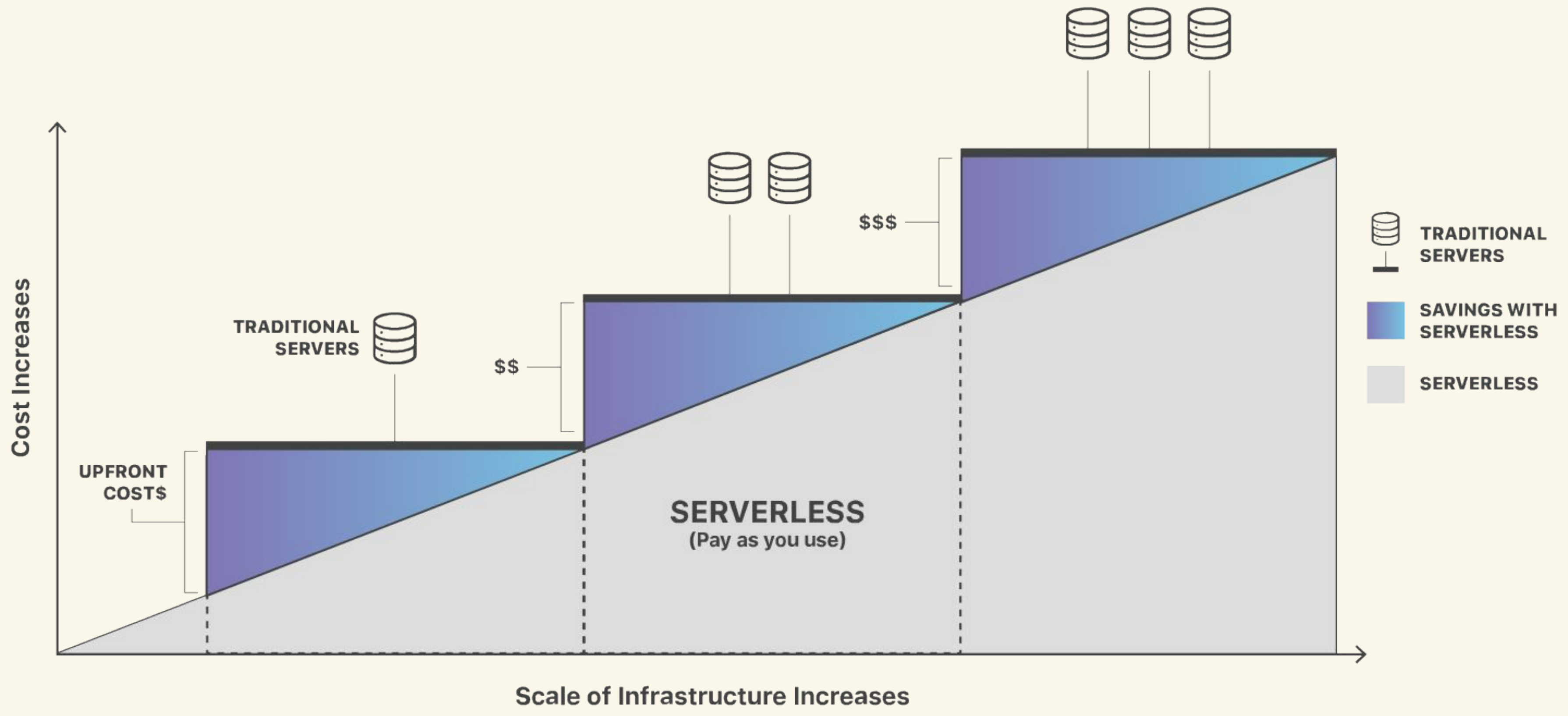
Serverless advantages

- Pay-as-you-go.
- No server management is necessary.
- Serverless architectures are inherently scalable.
- Quick deployments and updates are possible.
- Code can run closer to the end user, decreasing latency.

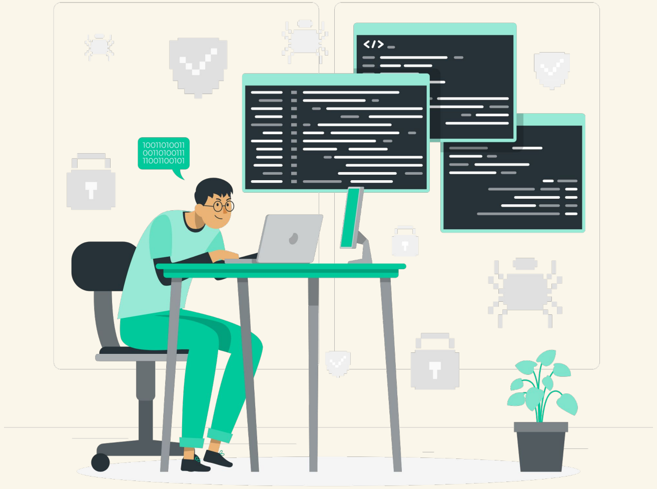
Serverless disadvantages

- Testing and debugging become more challenging.
- Serverless computing introduces new security concerns.
- Serverless architectures are not built for long-running processes.
- Cold start delays.
- Vendor lock-in is a risk.
- Stateless.




Cost Increases
UPFRONT COST$
TRADITIONAL SERVERS
$$
$$$
SERVERLESS
(Pay as you use)
Scale of Infrastructure Increases
TRADITIONAL SERVERS
SAVINGS WITH SERVERLESS
SERVERLESS

When to use Serverless?



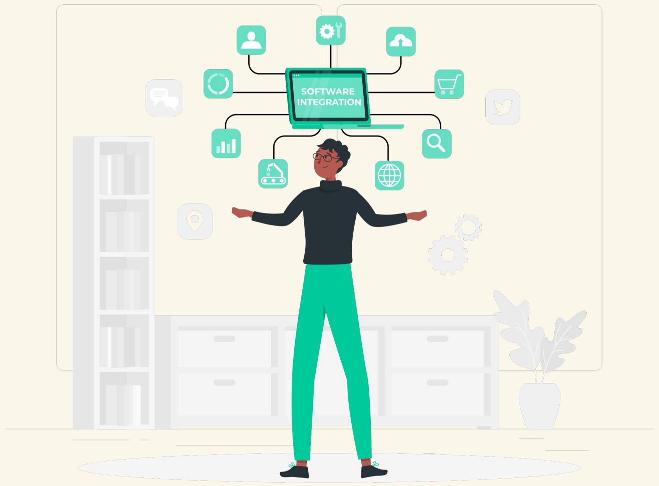
- Developers who want to decrease their go-to-market time and build lightweight, flexible applications that can be expanded or updated quickly.
- Inconsistent usage applications.
- When applications must be close to end users to reduce latency.

When to avoid using Serverless?

- Long-Running functions.
- Fear vendor lock-In.
- You need advanced monitoring.
- Large applications with a fairly constant, predictable workload may require a traditional setup (probably less expensive).
- It may be difficult to migrate legacy applications to a new infrastructure with an entirely different architecture.

FaaS (Function as a Service)

- FaaS is a cloud-computing service that allows customers to execute code in response to events.
- Faas ≠ Serverless.
- FasS ⊆ Serverless.

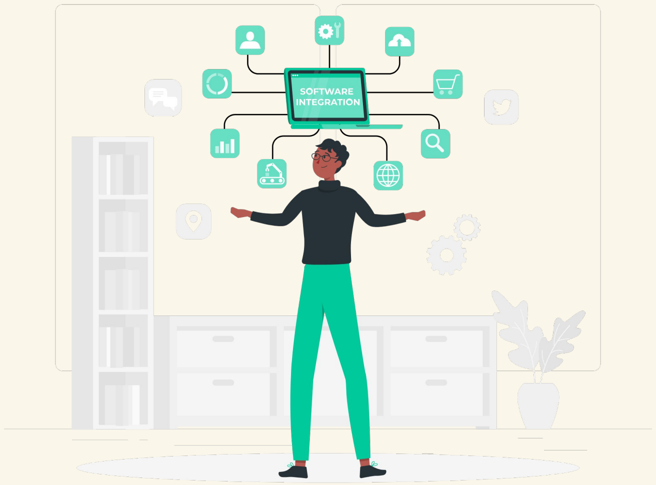


FaaS (Function as a Service)

- FaaS is a cloud-computing service that allows customers to execute code in response to events.
- Faas ≠ Serverless.
- FasS ⊆ Serverless.

Good Practices:

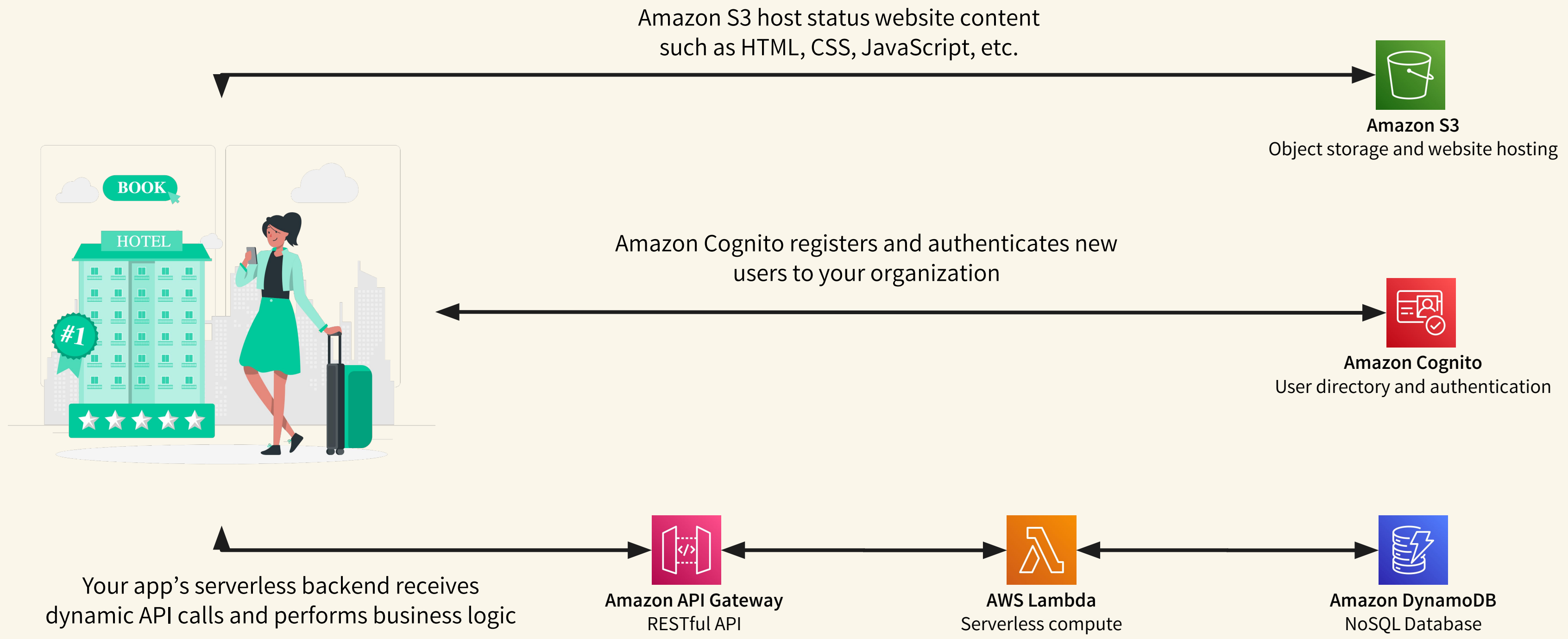
- Make each function perform only one action.
- Don't make functions call other functions.
- Use as few libraries in your functions as possible.



Serverless (AWS)

- Compute
  - AWS Lambda
- Containers
  - AWS Fargate
- Storage
  - Amazon S3
  - Amazon EFS
- Database
  - Amazon RDS Proxy
  - Amazon DynamoDB
  - Amazon Aurora Serverless
  - Amazon Neptune Serverless
  - Amazon Timestream
  - Amazon Redshift Serverless
- Application Integration
  - Amazon SNS
  - Amazon SQS
  - AWS AppSync
  - Amazon EventBridge
  - AWS Step Functions
  - Amazon API Gateway

Serverless Example



Amazon S3 host status website content such as HTML, CSS, JavaScript, etc.

Amazon Cognito registers and authenticates new users to your organization

Your app's serverless backend receives dynamic API calls and performs business logic

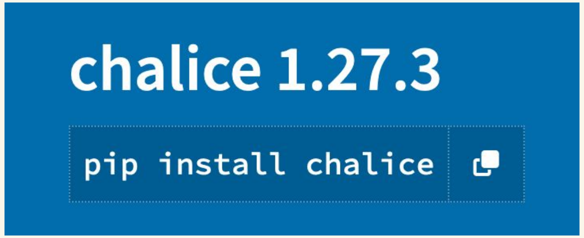
CHALICE

02

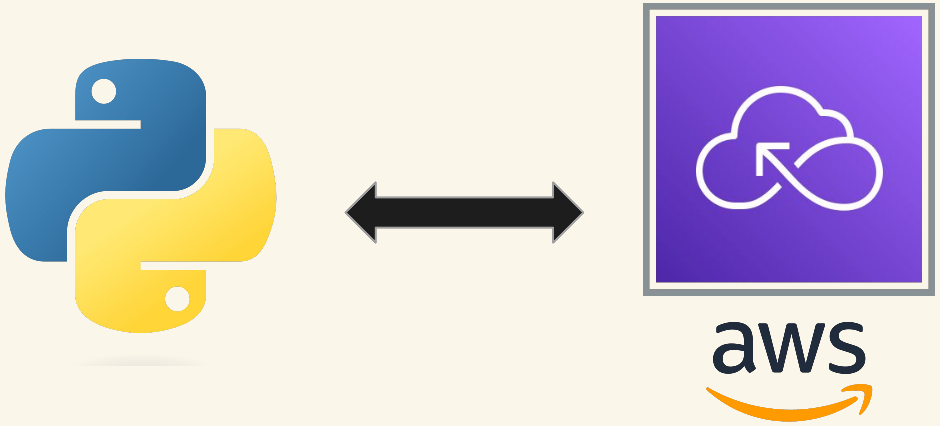
Chalice

What is Chalice?

Chalice is a framework for writing serverless apps in python. It allows you to quickly create and deploy applications that use AWS Lambda.

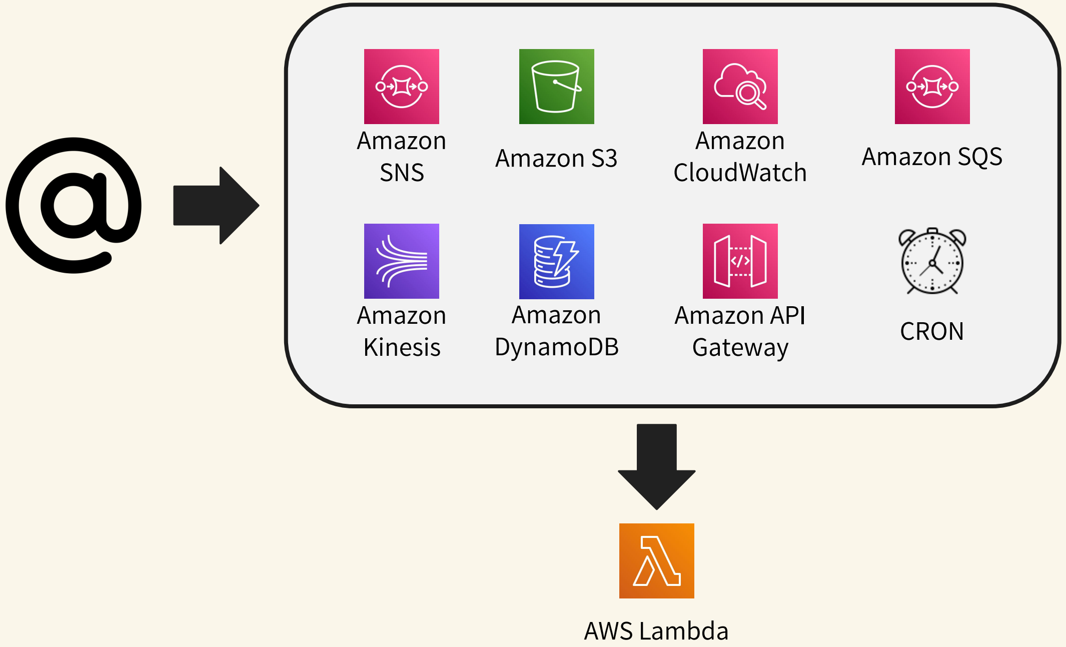

chalice 1.27.3

```
pip install chalice
```



What does Chalice provide?

A decorator based API



What does Chalice provide?

A command line tool for creating, deploying, and managing your app

AWS Identity and Access Management (IAM)

Automatic IAM policy generation

AWS CloudFormation

Automatic CloudFormation template generation

CLI command to export terraform template file

AWS CodePipeline

AWS CodeBuild

CLI command to set up a basic Continuous Deployment pipeline using AWS CodePipeline and AWS CodeBuild

AWS CloudFormation

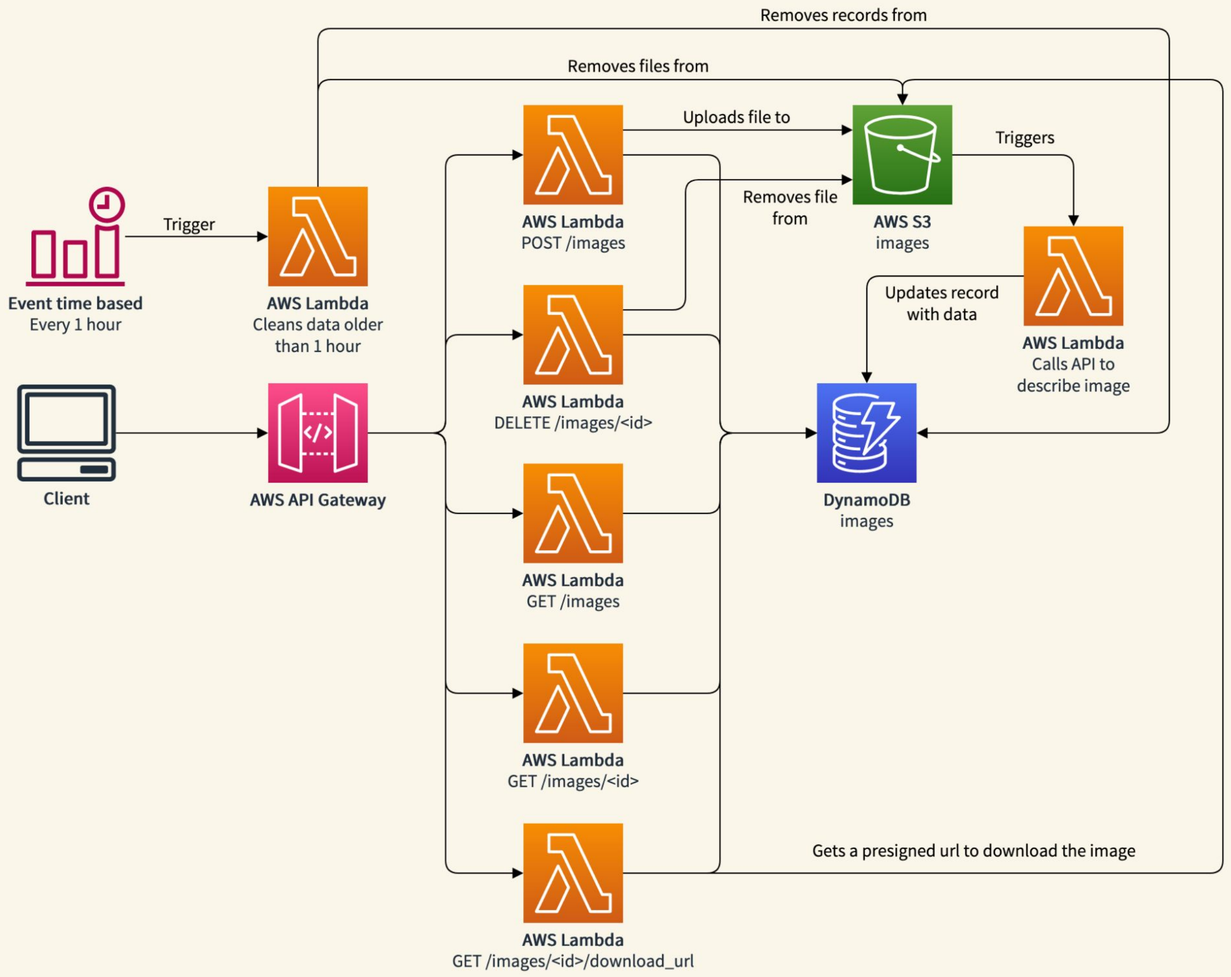
Automatic CloudFormation template generation

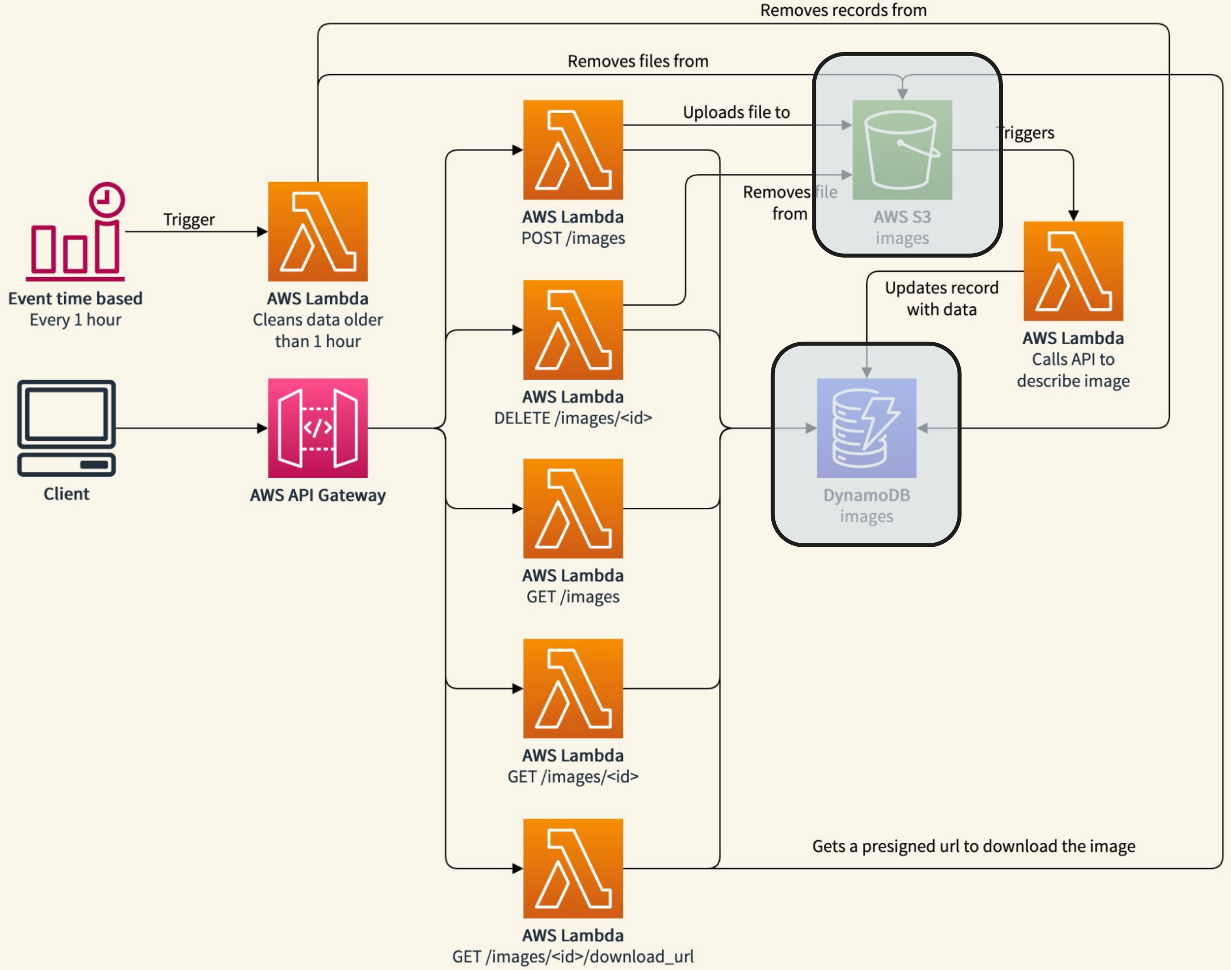
What can we do using Chalice?

- Rest APIs.
- Websockets.
- Tasks that run on a periodic basis.
- Connect a lambda function to an:
  - S3 event
  - SQS queue
  - SNS topic
  - CloudWatch event
  - Kinesis stream
  - DynamoDB stream

03

Live coding


Removes records from
Removes files from
Uploads file to
Triggers
Removes file from
Trigger
Event time based
Every 1 hour
AWS Lambda
Cleans data older than 1 hour
AWS Lambda
POST /images
AWS S3
images
Updates record with data
AWS Lambda
Calls API to describe image
AWS Lambda
DELETE /images/<id>
Client
AWS API Gateway
DynamoDB
images
AWS Lambda
GET /images
AWS Lambda
GET /images/<id>
Gets a presigned url to download the image
AWS Lambda
GET /images/<id>/download_url


Removes records from
Removes files from
Uploads file to
Triggers
Removes file from
Trigger
Event time based
Every 1 hour
AWS Lambda
Cleans data older than 1 hour
AWS Lambda
POST /images
AWS S3
images
Updates record with data
AWS Lambda
Calls API to describe image
AWS Lambda
DELETE /images/<id>
Client
AWS API Gateway
DynamoDB
images
AWS Lambda
GET /images
AWS Lambda
GET /images/<id>
Gets a presigned url to download the image
AWS Lambda
GET /images/<id>/download_url

Live coding

https://github.com/jdalzatec/mtt-2023-chalice/

https://bit.ly/mtt-chalice

Thanks!

Do you have any questions?
jdalzatec@gmail.com