AWS CDK workshop

From zero to hero

Laimonas Sutkus, CTO @ Biomapas

Short intro

Laimonas Sutkus

Work experience

- CTO @ Biomapas
- Previously CTO @ iDenfy

Years experience

- ~10 years in IT industry (Cloud, SaaS, AWS, AI)
- ~5 years AWS experience

laimonas.sutkus@gmail.com

https://www.linkedin.com/in/laimonas-sutkus

https://github.com/laimonassutkus

Short intro

Biomapas

Expertise

- Clinical Operations
- Regulatory Affairs
- Pharmacovigilance
- Medical Information

Techical side

- AWS cloud
- Serverless architectures
- AI - NLP
- Best CI/CD practices

info@biomapas.com

https://www.linkedin.com/company/biomapas-clinical-regulatory-pharmacovigilance/

https://github.com/Biomapas

Agenda

- AWS CDK intro
- Prerequisites
- Install tools
- Create & deploy sample project
- Build serverless architecture
  - Create API Gateway HTTP API
  - Create Lambda backends for CRUD actions
  - Create DynamoDB
  - Use Lambda Layers and PynamoDB ORM
- Write integration tests
- Create GitHub repo & push code
- Create CI/CD pipeline
- Summary

aws

Cloud Development Kit

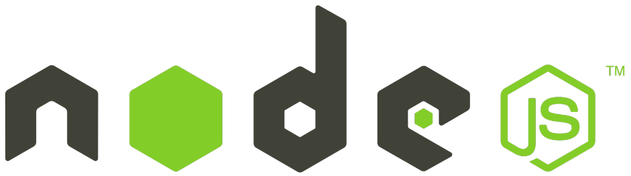
What is it?

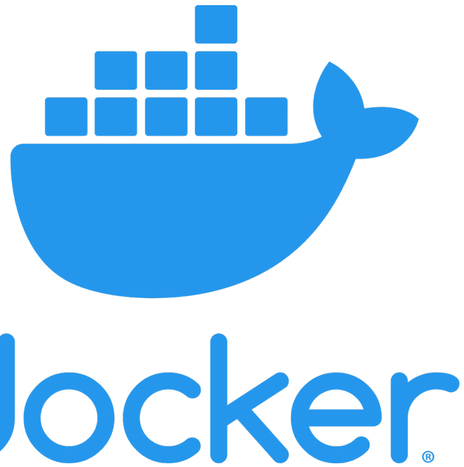
It is a framework that allows to define your infrastructure using a programming language. For example, using python you can define EC2 instances, RDS databases, IAM roles and so on. AWS CDK is just a wrapper for AWS CloudFormation, meaning AWS CDK produces CloudFormation templates. The actual deployment is being done by CloudFormation, not AWS CDK.

Alternatives?

Sort of. For example, the widely accepted Terraform has its own language. Troposphere is python-only. Chef is ruby-only. Ansible & Puppet are yaml / json / bash combination.

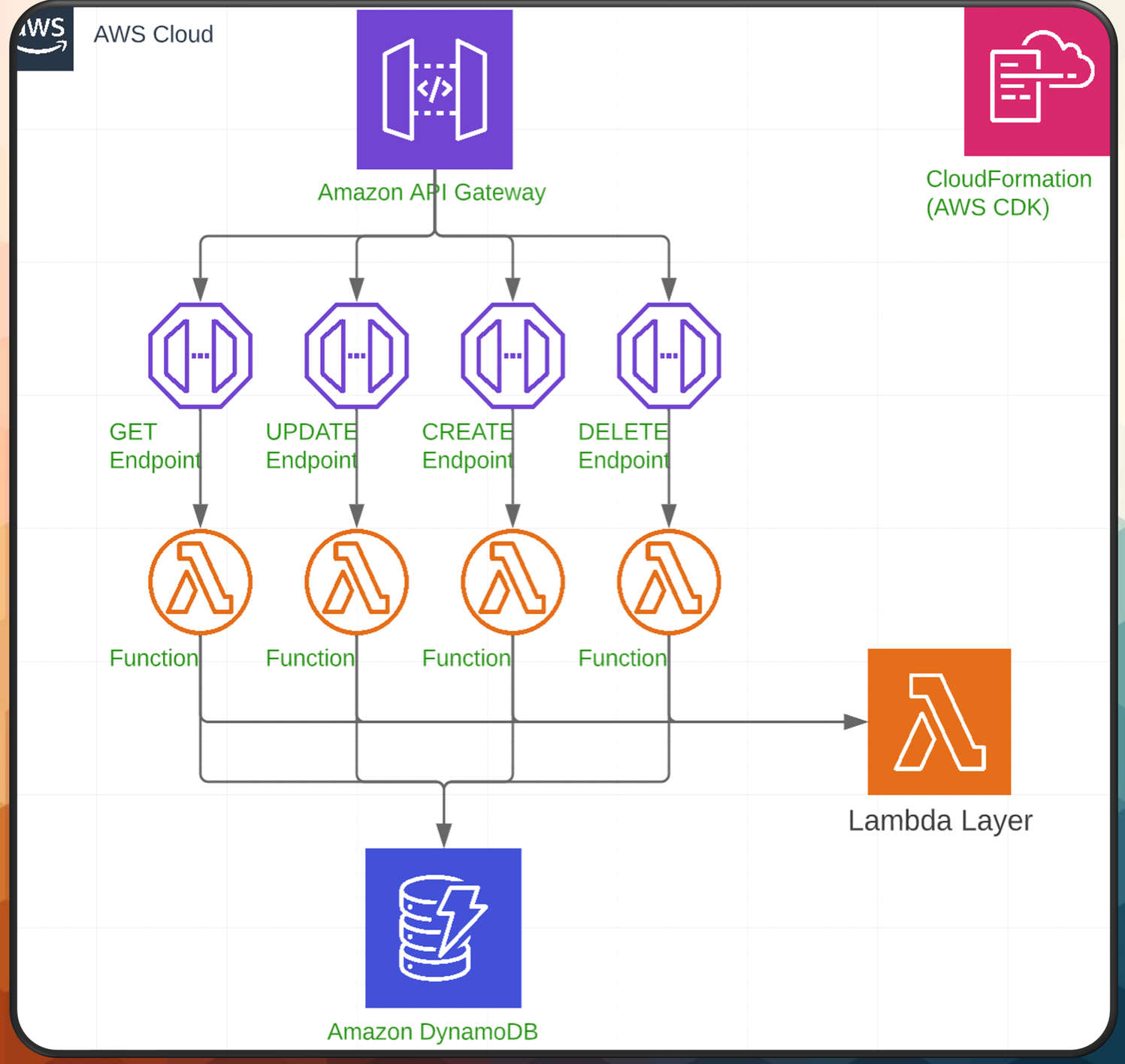
Prerequisites





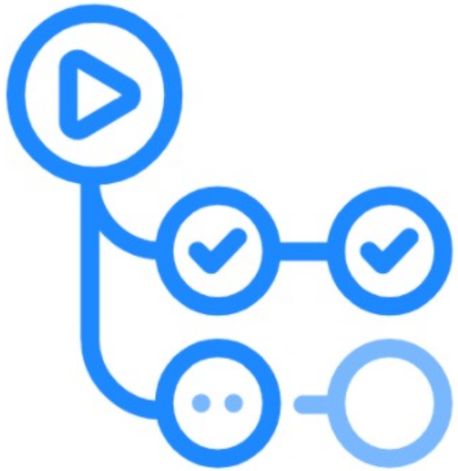

npm install -g aws-cdk@1.x


cdk init --language=python
cdk bootstrap
cdk deploy --all


AWS Cloud
Amazon API Gateway
CloudFormation
(AWS CDK)
GET
Endpoint
UPDATE
Endpoint
CREATE
Endpoint
DELETE
Endpoint
Function
Function
Function
Function
Lambda Layer
Amazon DynamoDB

Unit tests

Integration tests



GitHub Actions

Pro TIPS

- Be aware of stack limits.
- Have separate stacks for storage-type resources.
- Reduce dependencies between stacks to minimum.
- Do not share lambda layers between stacks.
- Use custom resources.
- Use 3rd party constructs (Biomapas, iDenfy, etc.).
- Use asynchronous stack deployments.
- Have automated CI/CD pipelines for DEV, STAGE & PROD.
- Split your application into multiple AWS CDK apps.

Summary

- AWS CDK is a great tool to manage your infrastructure
- Write integration tests to test your infrastructure AND application
- Use open-source libraries (like Biomapas open-source) to boost productivity

Thank you!