Infrastructure IS Code

Intro to AWS CDK

Dr. Michael Mourao,
Full Stack Developer

Vasos Koupparis,
Full Stack Developer

Nodes & Links

Agenda

Intro and Concepts (15 min) by MM

Demo: Use AWS CDK to create a small serverless application. (30 min) by MM

How we use AWS CDK at Nodes & Links to provision permanent and development environments using the same code. (45 min) by VK

Wrap Up - Questions!

History of Cloud Infrastructure Provisioning

From the Console

It all looks so pretty!! Now let me just provision another environment, … wait … how did I do this last time?

Using scripts

Ok we automated creation, but what about updating?

AWS CLI

Using templates

Great we can now update efficiently too. But templates are too difficult to write and edit!

CloudFormation

Using tools to help with template generation

This is much easier! But it still feels I'm repeating myself...

Terraform

serverless

ANSIBLE

AWS SAM FOR LAMBDA TESTING

CHEF

puppet

Using code

aws CDK

What is the Cloud Development Kit (AWS CDK)?

A multi-language software development framework for modelling cloud infrastructure as reusable components

```
export class MyEcsConstructStack extends core.Stack {
  constructor(scope: core.App, id: string, props?: core.StackProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, "MyVpc", {
      maxAzs: 3 // Default is all AZs in region
    });

    const cluster = new ecs.Cluster(this, "MyCluster", {
      vpc: vpc
    });

    // Create a load-balanced Fargate service and make it public
    new ecs_patterns.ApplicationLoadBalancedFargateService(this, "MyFargateService", {
      cluster: cluster, // Required
      cpu: 512, // Default is 256
      desiredCount: 6, // Default is 1
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry("amazon/amazon-ecs-sample")
      },
      memoryLimitMiB: 2048, // Default is 512
      publicLoadBalancer: true // Default is false
    });
  }
}
```

What it looks like on aws Cloud Formation?

```yaml
Resources:
  MyVpcF9F0CA6F:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
      InstanceTenancy: default
      Tags:
        - Key: Name
          Value: MyEcsConstruct/MyVpc
    Metadata:
      aws:cdk:path: MyEcsConstruct/MyVpc/Resource
  MyVpcPublicSubnet1SubnetF6608456:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.0/18
      VpcId:
        Ref: MyVpcF9F0CA6F
      AvailabilityZone:
        Fn::Select:
          - 0
          - Fn::GetAZs: ''
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: MyEcsConstruct/MyVpc/PublicSubnet1
        - Key: aws-cdk:subnet-name
          Value: Public
        - Key: aws-cdk:subnet-type
          Value: Public
    Metadata:
      aws:cdk:path:
MyEcsConstruct/MyVpc/PublicSubnet1/Subnet
  MyVpcPublicSubnet1RouteTableC46AB2F4:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId:
        Ref: MyVpcF9F0CA6F
      Tags:
        - Key: Name
          Value: MyEcsConstruct/MyVpc/PublicSubnet1
    Metadata:
      aws:cdk:path:
MyEcsConstruct/MyVpc/PublicSubnet1/RouteTable
  MyVpcPublicSubnet1RouteTableAssociation2ECEE1CB:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId:
        Ref: MyVpcPublicSubnet1RouteTableC46AB2F4
      SubnetId:
        Ref: MyVpcPublicSubnet1SubnetF6608456
    Metadata:
      aws:cdk:path:
MyEcsConstruct/MyVpc/PublicSubnet1/RouteTableAssociation
  MyVpcPublicSubnet1DefaultRoute95FDF9EB:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId:
        Ref: MyVpcPublicSubnet1RouteTableC46AB2F4
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId:
        Ref: MyVpcIGW5C4A4F63
    DependsOn:
      - MyVpcVPCGW488ACE0D
    Metadata:
      aws:cdk:path: MyEcsConstruct/MyVpc/PublicSubnet1/DefaultRoute
  MyVpcPublicSubnet1EIP096967CB:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
    Metadata:
      aws:cdk:path: MyEcsConstruct/MyVpc/PublicSubnet1/EIP
  MyVpcPublicSubnet1NATGatewayAD3400C1:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId:
        Fn::GetAtt:
          - MyVpcPublicSubnet1EIP096967CB
          - AllocationId
      SubnetId:
        Ref: MyVpcPublicSubnet1SubnetF6608456
      Tags:
        - Key: Name
          Value: MyEcsConstruct/MyVpc/PublicSubnet1
    Metadata:
      aws:cdk:path: MyEcsConstruct/MyVpc/PublicSubnet1/NATGateway
  MyVpcPublicSubnet2Subnet492B6BFB:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.64.0/18
      VpcId:
        Ref: MyVpcF9F0CA6F
      AvailabilityZone:
        Fn::Select:
          - 1
          - Fn::GetAZs: ''
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: MyEcsConstruct/MyVpc/PublicSubnet2
        - Key: aws-cdk:subnet-name
          Value: Public
        - Key: aws-cdk:subnet-type
          Value: Public
    Metadata:
      aws:cdk:path: MyEcsConstruct/MyVpc/PublicSubnet2/Subnet
...
...
```

That's 100 lines , there are 400 more :)
https://github.com/awsdocs/aws-cdk-guide/blob/master/doc_source/my_ecs_construct-stack.yaml

Deploying the AWS CDK app produces more than 50 resources, in just 20 lines of code!

Other advantages of the AWS CDK include:

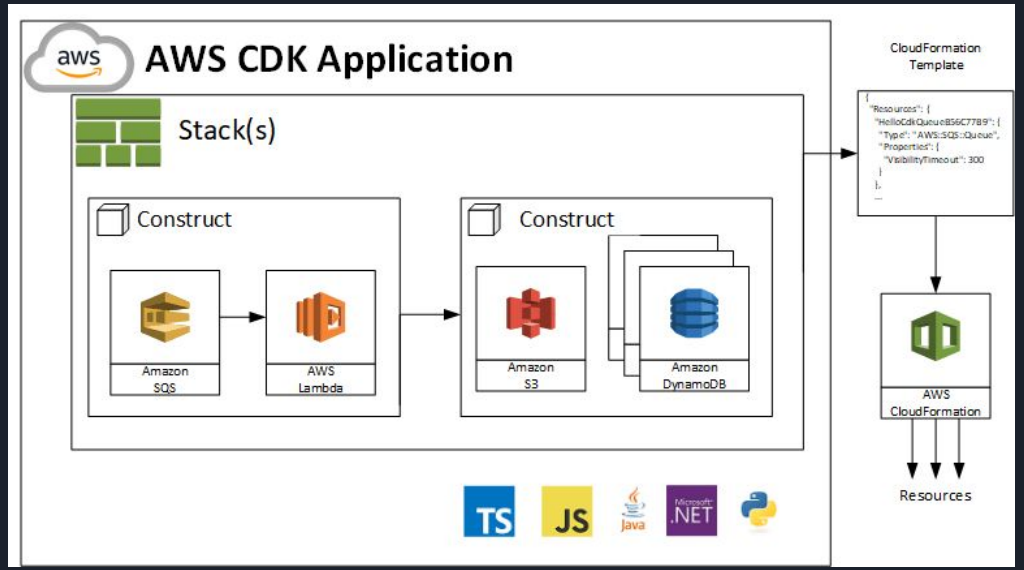
- Use logic (if statements, for-loops, etc) when defining your infrastructure.
- Smart defaults
- Use object-oriented techniques to create a model of your system.
- Define high level abstractions, share them, and publish them to your team, company, or community.
- Organize your project into logical modules.
- Share and reuse your infrastructure as a library.
- Testing your infrastructure code using industry-standard protocols.
- Use your existing code review workflow.
- Code completion within your IDE.

The CDK Hierarchy

Apps - A construct which represents an entire CDK app. This construct is normally the root of the construct tree.

Stacks - The unit of deployment in the AWS CDK is called a stack

Constructs - are the basic building blocks of AWS CDK apps. A construct represents a "cloud component" and encapsulates everything AWS CloudFormation needs to create the component.



Development Environments on the Cloud

Approach/Technology	PROS	CONS
Local (mock cloud dependencies)	😀 Cheap (if using free db offering) 😀 Quick on-demand deployments 😀 Multiple environments	😢 Doesn't reflect PROD
Cloud - Non Serverless	😀 Reflects PROD	😟 Prohibitively expensive
Cloud - Serverless	😀 Cheap 😀 Reflects PROD 😀 Quick on-demand deployments 😀 Multiple environments	😟 Difficult to set-up 😍 AWS CDK to the rescue!

More on this later...

Demo Time

Nodes & Links

Aegis

Demo of our SaaS product

Did you know

What percentage of the companies do you think complete their projects successfully?

- **Only 2.5 percent of companies complete 100 percent of their projects successfully. (Gallup)**
- **In 2015, For every $1 billion invested in the United States, $109 million was wasted due to lacking project performance. (PMI.org)**
- **On average, projects go over budget by 27 percent of their intended cost. (Harvard Business Review)**
- **Most organizations have a 70 percent project failure rate. (4PM)**

Nodes & Links

A startup company with offices in Nicosia and London.

We have raised £1.4m seed round led by ADV with participation from Seedcamp and EF.

Laser-focused on understanding and interpreting complexity for project leaders by introducing clarity and predictability.

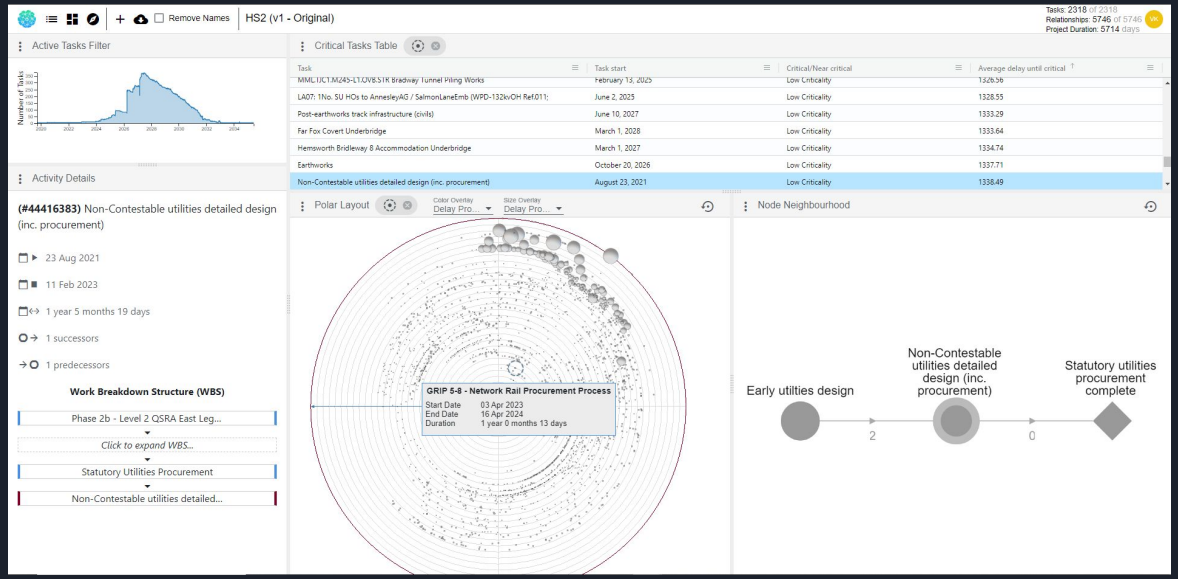
How it is done?

1. **You upload your data in a secure way.**
2. **Our algorithms decipher the patterns of complexity.**
3. **Our platform tells you how to improve performance.**
4. **You act and reap the benefits.**

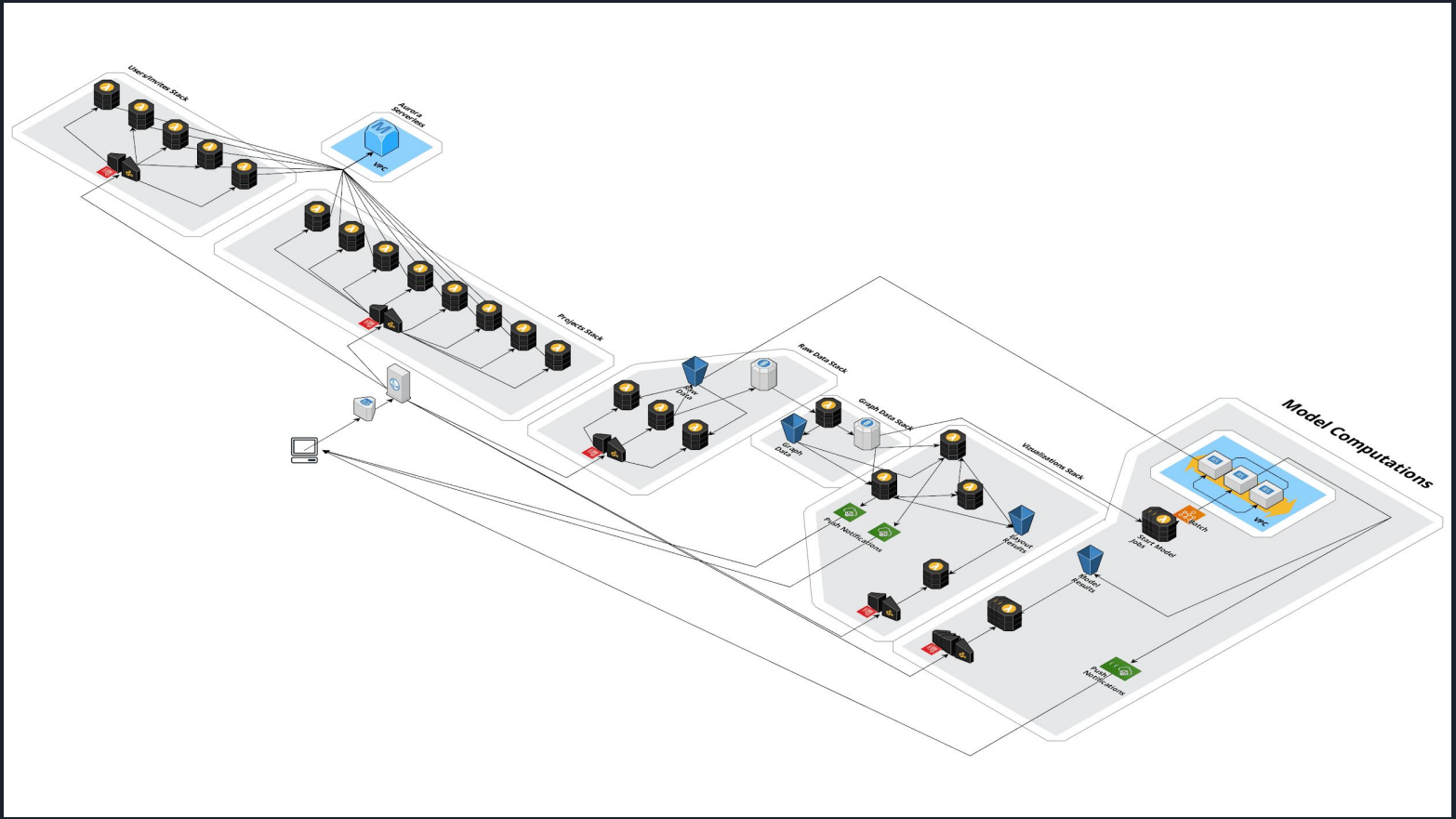
Aegis - currently alpha version

SaaS product for project managers, project controls, planners.

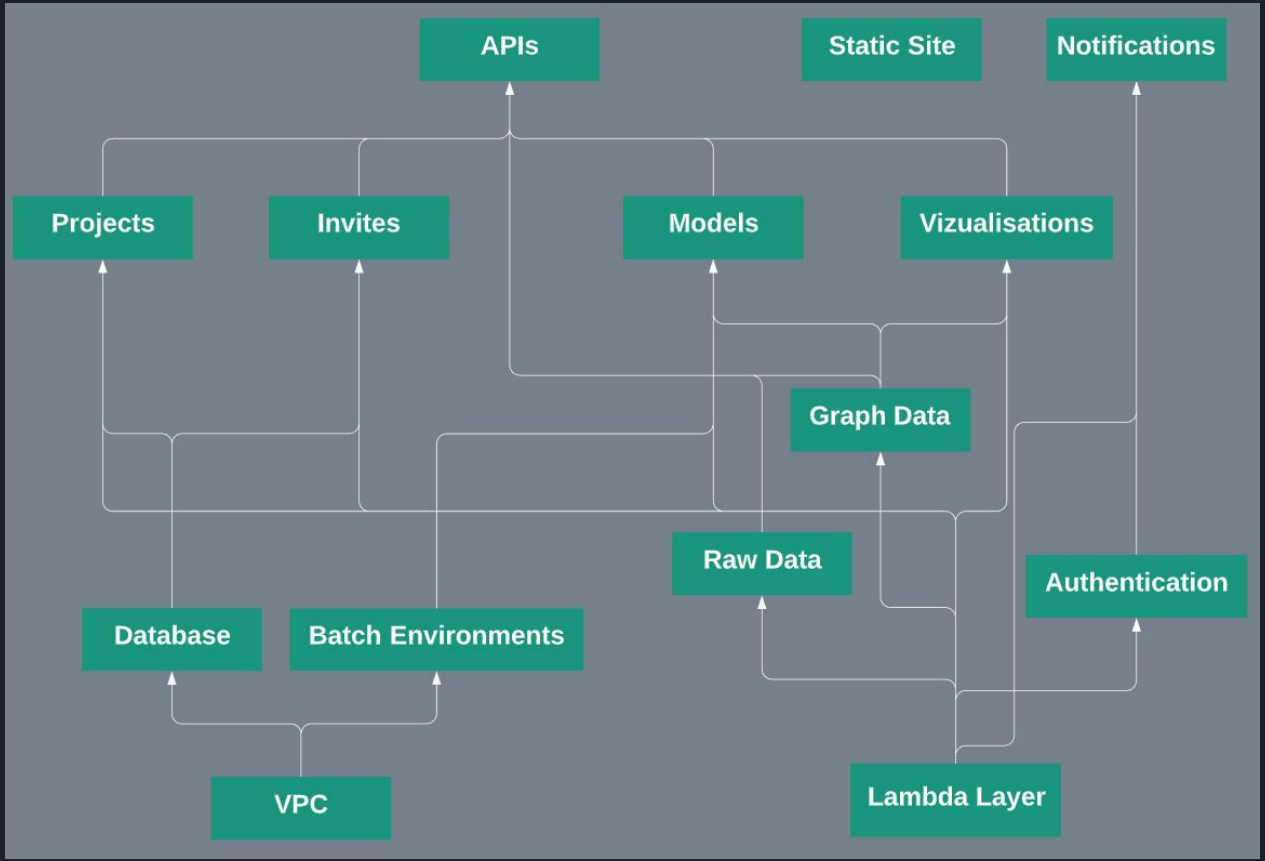
- Uncover hidden risks in the project.
- Identify the critical activities in the project.
- Explore Activities, resources, costs.



Our infrastructure ...



... which is deployed based on these CDK Stacks...



… replicated for each of these environments

Environment	Permanent	PROD-like	DEV-like	Experimental
PROD	✓	✓	✗	✗
Staging	✓	✓	✗	✗
Dev	✓	✗	✓	✗
Preview	✓	✗	✗	✓
Explorers	✓	✗	✗	✓
One for each feature development	✗	✗	✓	✓

Tagging is easy!

AWS CDK provides the **Tag** class which includes two methods that you can use to create and delete tags:

- **Tag.add()** applies a new tag to a construct and **all of its children recursively.**
- **Tag.remove()** removes a tag from a construct and any of its children, including tags a child construct may have applied to itself.

Let's look at a couple of examples using Tags at Nodes & Links.

```
Tag.add(stack, 'stack', VPCStack.Name);
Tag.add(stack, 'env-type', envProps.environment);
Tag.add(stack, 'env-name', envProps.prefix);
```

Both methods supports properties that fine-tune how tags are applied to resources. https://docs.aws.amazon.com/cdk/latest/guide/tagging.html

Next Steps

Get Started!

https://docs.aws.amazon.com/cdk/index.html
https://docs.aws.amazon.com/cdk/api/latest/typescript/api/index.html
https://cdkworkshop.com/
https://aws.amazon.com/developer/tools/
https://gitter.im/awslabs/aws-cdk

Our demo

https://github.com/nodes-links/gdg-talk-demo

Engage with Us!

https://www.nodeslinks.com/

Contribute!

https://github.com/aws/aws-cdk

Thank you !