AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

DOP302

How to reuse patterns when developing infrastructure as code

Ryan Bachman (He/Him)
Specialist SA – DevOps
AWS

Ethan Rucinski (He/Him)
Principal Architect
United Airlines

Ravi Palakodeti (He/Him)
Senior Solutions Architect
AWS

What am I going to talk about?

- How infrastructure is evolving
- Challenges organizations face
- CDK @ United
- IaC Community Engagement Program

“One of my most productive days was throwing away 1000 lines of code.”

Ken Thompson

American Computer Science Pioneer, Early UNIX Developer

How infrastructure is evolving

On-premises infrastructure

Build environments with available infrastructure

Lengthy delivery times

Limited innovation

Cloud infrastructure

Ubiquitous access to infrastructure through on-premises virtualization

Deliver infrastructure as code

Eliminate one roadblock to deploying application environments automatically

Application-based EaaS

Ubiquitous access to infrastructure and AWS Cloud services

Unique requirements for different applications

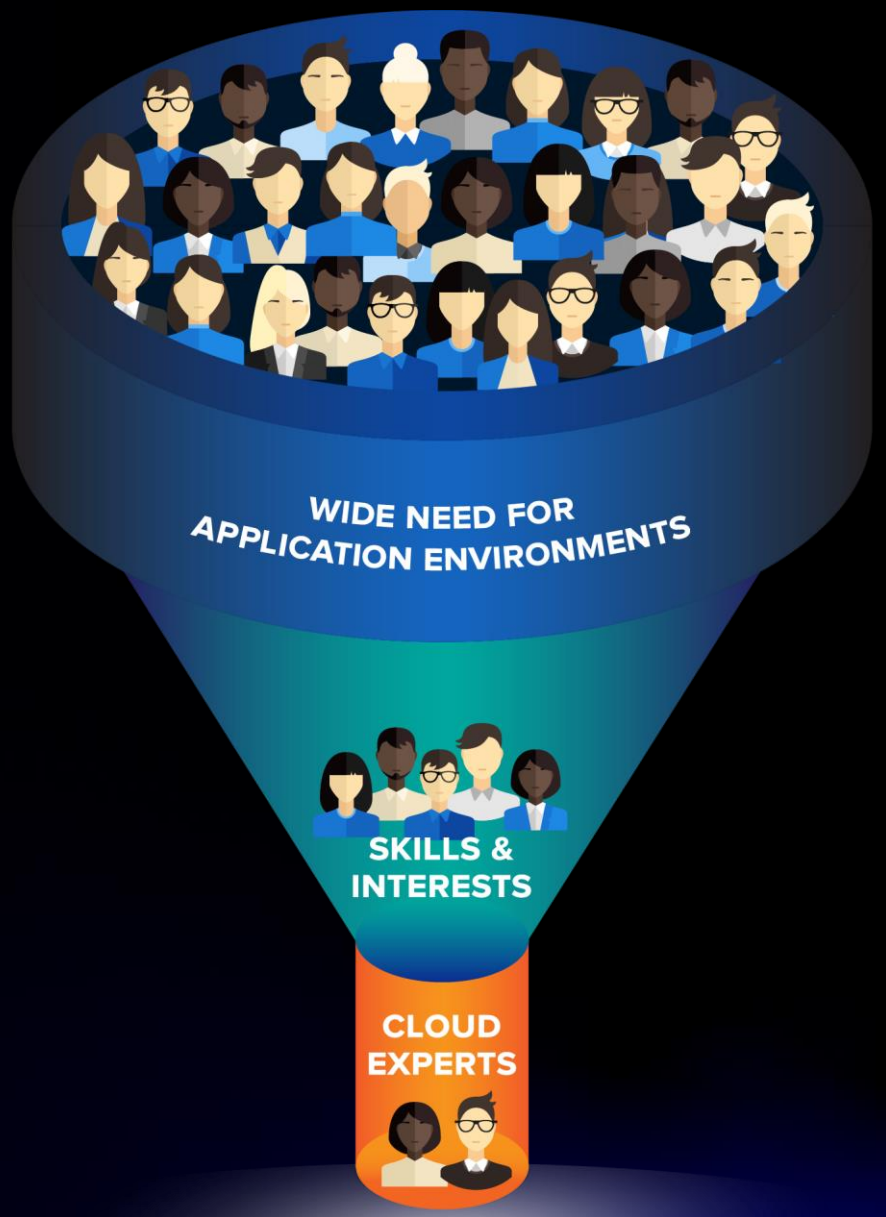
Ability to automate the process of deploying application environments

The skills gap makes this more difficult

Huge demand for application environments

Low supply of people with the skills to support them

Bottlenecks that slow down delivery and reduce quality (velocity)

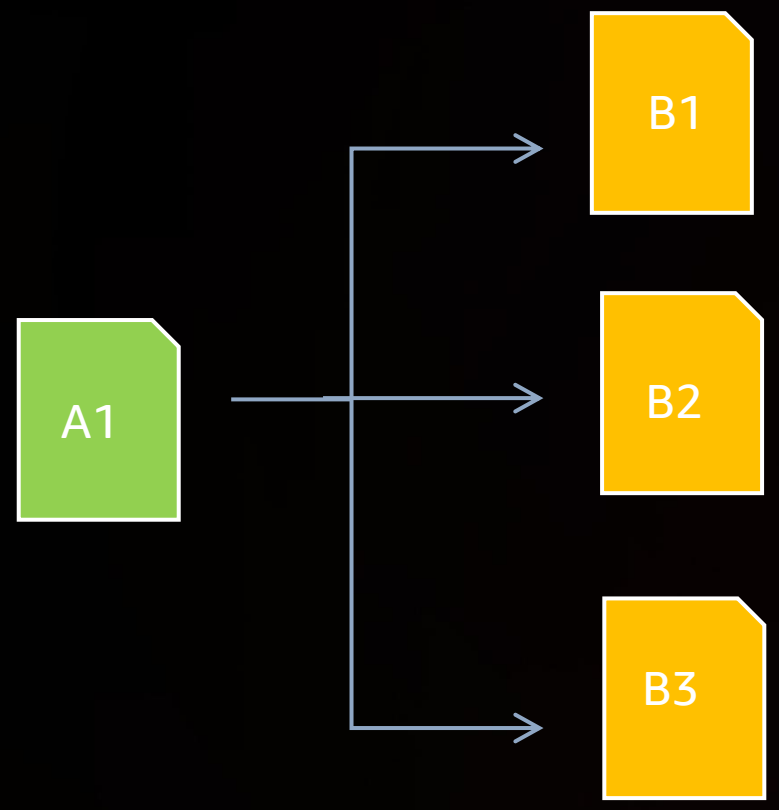


Challenges

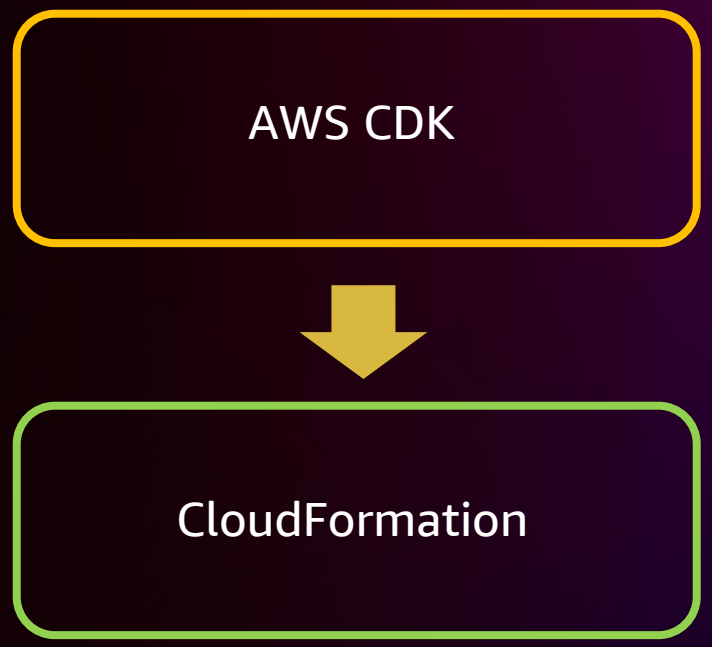
- Builders are spending too much time trying to figure out resource intricacies and best practices
- Applications often share common architectural elements
  - For example, one or more Lambda functions that can read/write to an Amazon DynamoDB table
- Modeling infrastructure becomes repetitive, “re-implementing the wheel”
  - Defining and configuring each service
  - Integrating with other services
  - Adhering to best practices
- Takes valuable development time from building unique features

AWS to the rescue

AWS CloudFormation modules



AWS Cloud Development Kit (AWS CDK)



AWS CDK at United Airlines

- Business considerations
- Agility
- People processes
- Cloud adoption

Agenda

About me

Where we were

Our roadmap to the solution

Key steps we took and decisions we made

How we solved the problem

Where we are today

About me

ETHAN RUCINSKI | PRINCIPAL ARCHITECT AT UNITED AIRLINES

- Chicago, IL
- Previously
  - Manager – network operations
  - Operations data analyst
- Fun facts
  - Have flown ~1M miles with United
  - Enjoy rock climbing
- 4 AWS Certifications

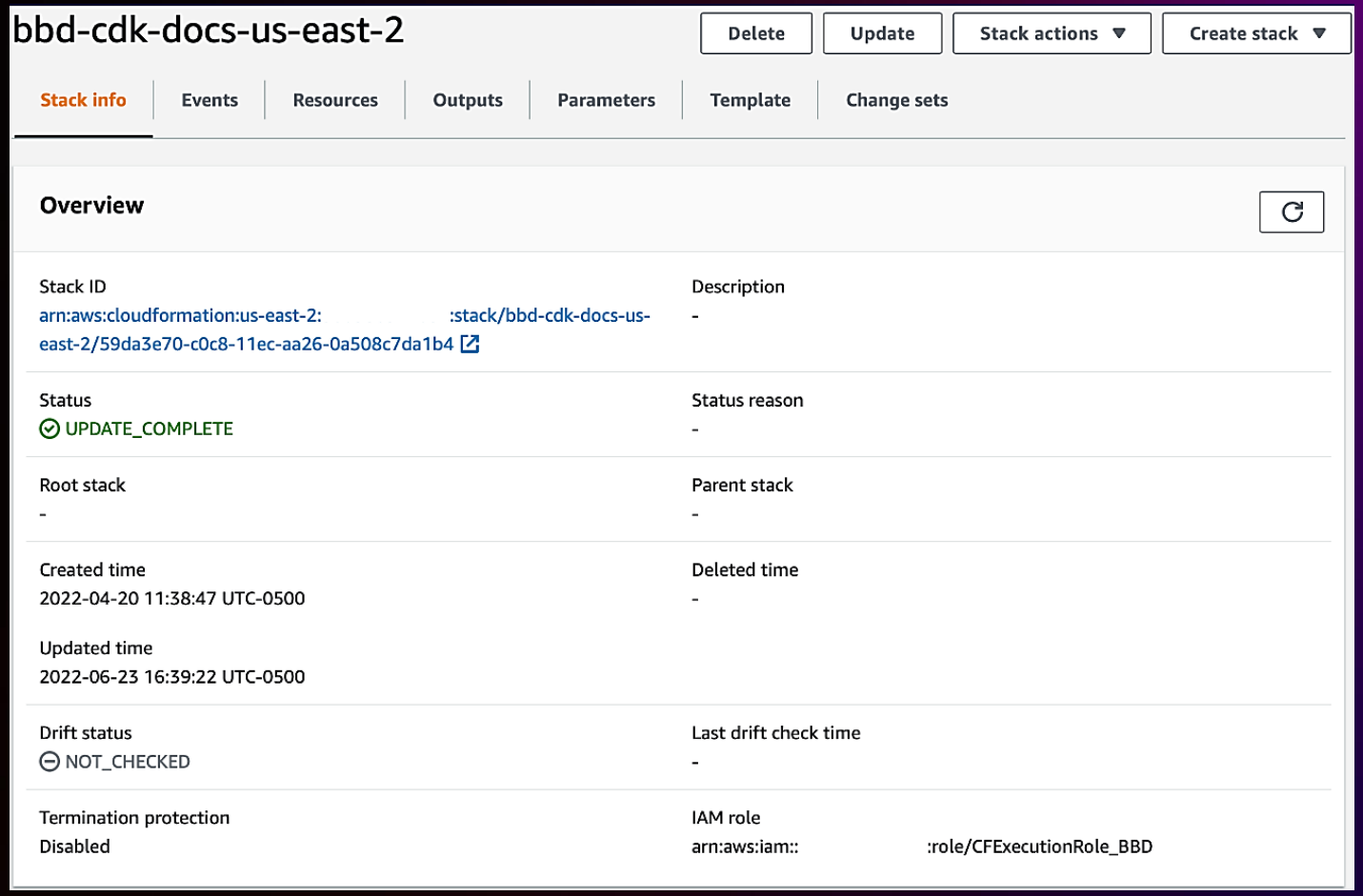
United has a large and growing cloud footprint

- >200 apps in the cloud
- >120 active AWS accounts
- ~100 different services

United's mass cloud migration is in full swing

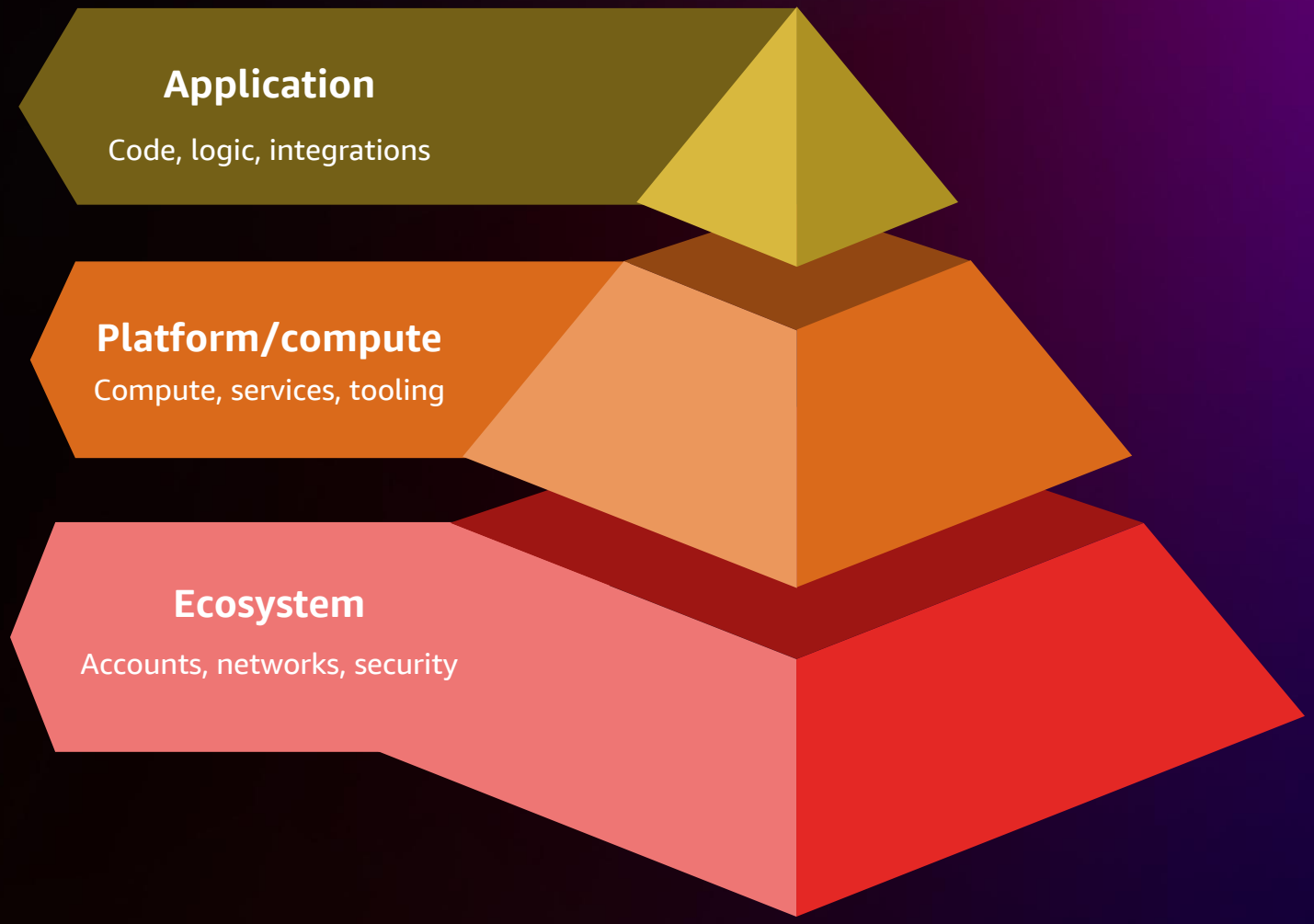
App teams own their cloud destiny at United

- Developers implement and deploy their own cloud infrastructure
- Strict policies protect higher environments
- Infrastructure as code enforces consistency across deployments



Where does PlatformOps fit in?

- CI/CD tooling
- Cloud reference architectures
- Support app teams new to the cloud



First-mover teams become experts in AWS CloudFormation

```yaml
Service:
    Type: AWS::ECS::Service
    Properties:
        Cluster:
            Fn::ImportValue: !Sub ${AppID}-ecs-cluster
        DesiredCount: !Sub ${DesiredCount}
        LoadBalancers:
            - ContainerName: !Sub ${AppID}-${ServiceName}
              ContainerPort: 3000
              TargetGroupArn: !Ref "TargetGroup"
        NetworkConfiguration:
            AwsvpcConfiguration:
                AssignPublicIp: DISABLED
                SecurityGroups:
                    - Fn::ImportValue: !Sub ${AppID}-ecs-service-security-group
                Subnets:
                    - !Ref "SubnetA"
                    - !Ref "SubnetB"
                    - !Ref "SubnetC"
        LaunchType: FARGATE
        ServiceName: !Sub ${AppID}-${ServiceName}
        TaskDefinition: !Ref TaskDefintion
        PropagateTags: SERVICE
```

First-mover teams connect everything

```yaml
TaskDefintion:
    Type: AWS::ECS::TaskDefinition
    Properties:
        Cpu: !Sub ${T
        RequiresCompa
            - FARGATE
        Family: !Sub
        NetworkMode:
        Memory: !Sub
        ExecutionRole
            Fn::Impor
        TaskRoleArn:
            Fn::Impor
        ContainerDefi
```

```yaml
Service:
    Type: AWS::ECS::Service
    Properties:
        Cluster:
            Fn::ImportValue: !Sub ${AppID}-ecs-cluster
        DesiredCount: !Sub ${De
        LoadBalancers:
            - ContainerName: !S
              ContainerPort: 30
              TargetGroupArn: !
        NetworkConfiguration:
            AwsvpcConfiguration
                AssignPublicIp:
                SecurityGroups:
                    - Fn::Impor
                Subnets:
                    - !Ref "Sub
                    - !Ref "Sub
                    - !Ref "Sub
```

```yaml
TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
        Name: !Sub $
        Port: 3000
        Protocol: HT
        TargetType:
        VpcId: !Ref
        HealthCheckP
        HealthCheckP
        Tags:
            - Key: A
              Value:
```

```yaml
ApiHttpsListenerRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
        Actions:
            - Type: forward
              TargetGroupArn: !Ref TargetGroup
        Conditions:
            - Field: path-pattern
              Values:
                - !Sub "/${RoutingRulePath}*"
        ListenerArn:
            Fn::ImportValue: !Sub ${AppID}-elb-https-listener
        Priority: !Ref LoadBalancerPriority
```

First-mover teams repeat each other

```yaml
Service:
    Type: AWS::ECS::Service
    Properties:
        Cluster:
            Fn::ImportValue: !Sub ${AppID}-ecs-cluster
        DesiredCount: !Sub ${DesiredCount}
        LoadBalancers:
            - ContainerName: !Sub ${AppID}-${ServiceNam
              ContainerPort: 3000
              TargetGroupArn: !Ref "TargetGroup"
        NetworkConfiguration:
            AwsvpcConfiguration:
                AssignPublicIp: DISABLED
                SecurityGroups:
                    - Fn::ImportValue: !Sub ${AppID}-ec
                Subnets:
                    - !Ref "SubnetA"
                    - !Ref "SubnetB"
                    - !Ref "SubnetC"
        LaunchType: FARGATE
        ServiceName: !Sub ${AppID}-${ServiceName}
        TaskDefinition: !Ref TaskDefintion
        PropagateTags: SERVICE
        Tags:
            - Key: ApplicationID
              Value: !Ref AppID
```

```yaml
Service:
    Type: AWS::ECS::Service
    Properties:
        Cluster:
            Fn::ImportValue: !Sub ${AppID}-ecs-cluster
        DesiredCount: !Sub ${DesiredCount}
        LoadBalancers:
            - ContainerName: !Sub ${AppID}-${ServiceNam
              ContainerPort: 80
              TargetGroupArn: !Ref "TargetGroup"
        NetworkConfiguration:
            AwsvpcConfiguration:
                AssignPublicIp: DISABLED
                SecurityGroups:
                    - Fn::ImportValue: !Sub ${AppID}-ec
                Subnets:
                    - !Ref "SubnetA"
                    - !Ref "SubnetB"
                    - !Ref "SubnetC"
        LaunchType: FARGATE
        ServiceName: !Sub ${AppID}-${ServiceName}
        TaskDefinition: !Ref TaskDefintion
        PropagateTags: SERVICE
        Tags:
            - Key: ApplicationID
              Value: !Ref AppID
```

```yaml
Service:
    Type: AWS::ECS::Service
    Properties:
        Cluster:
            Fn::ImportValue: !Sub ${AppID}-ecs-cluster
        DesiredCount: !Sub ${DesiredCount}
        LoadBalancers:
            - ContainerName: !Sub ${AppID}-${ServiceName}
              ContainerPort: 8080
              TargetGroupArn: !Ref "TargetGroup"
        NetworkConfiguration:
            AwsvpcConfiguration:
                AssignPublicIp: DISABLED
                SecurityGroups:
                    - Fn::ImportValue: !Sub ${AppID}-ecs-service-security-group
                Subnets:
                    - !Ref "SubnetA"
                    - !Ref "SubnetB"
                    - !Ref "SubnetC"
        LaunchType: FARGATE
        ServiceName: !Sub ${AppID}-${ServiceName}
        TaskDefinition: !Ref TaskDefintion
        PropagateTags: SERVICE
        Tags:
            - Key: ApplicationID
              Value: !Ref AppID
```

Parameterized templates reduce repetition

- DevOps teams produce standard templates
- Applications leverage latest templates for current best practices
- Expert teams build their own templates for new features

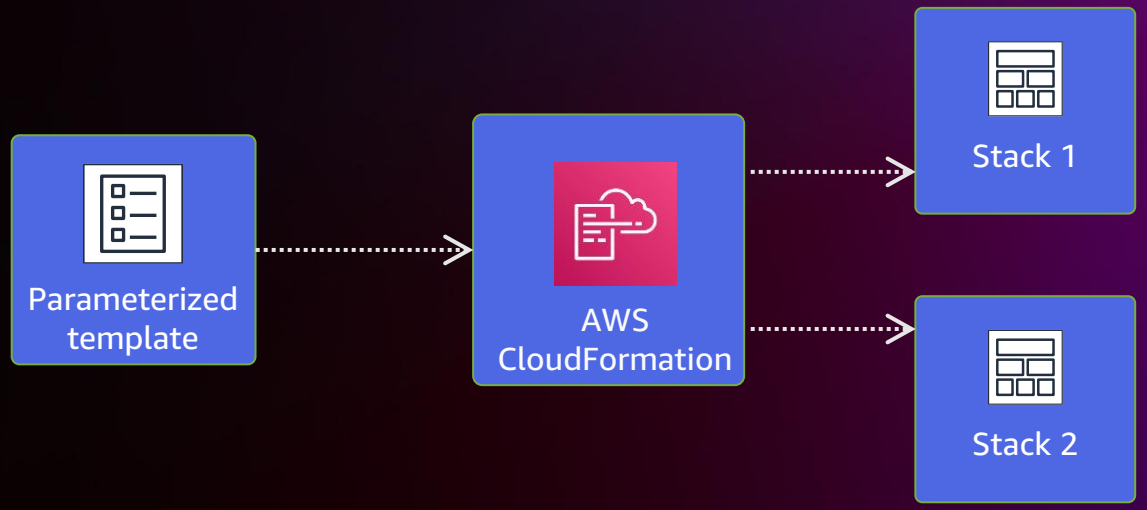
```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: Template for standing up generic Lambda
Parameters:
    LambdaName:
        Type: String
        Description: Name for the Lambda and associated resources
    AppCI:
        Type: String
        Description: Lowercase AppCI Example abc
    LambdaFunctionHandler:
        Description: The handler for your Lambda function
        Type: String
        Default: main.lambda_handler
    LambdaPackageS3Bucket:
        Description: Bucket where the lambda package is placed Example this-is-my-s3-bucket-name
        Type: String
    LambdaPackage:
        Description: The name of your Lambda deployment package Example lambda.zip
        Type: String
        Default: lambda.zip
    LambdaRuntimeEnvironment:
        Description: The runtime environment for your Lambda function
        Type: String
        Default: python3.6
    ExecutionRoleArn:
        Description: ARN of the role for the lambda to use
        Type: String
    SecurityGroup:
        Description: ID of a security group to attach to this lambda
        Type: AWS::EC2::SecurityGroup::Id
    SubnetA:
        Description: ID of a subnet in which to run the lambda
        Type: AWS::EC2::Subnet::Id
    SubnetB:
        Description: ID of a subnet in which to run the lambda
        Type: AWS::EC2::Subnet::Id

Resources:
    Lambda:
        Type: AWS::Lambda::Function
        Properties:
            FunctionName: !Ref LambdaName
```

Growing pains

CloudFormation

Parameters and
intrinsic functions



Things that get stale	Why they got stale
Best practices	Best practices change constantly
Policies	Policies adapt to new features and new risks
The code I got from Bob	Bob's code had bugs, and he wrote it last year

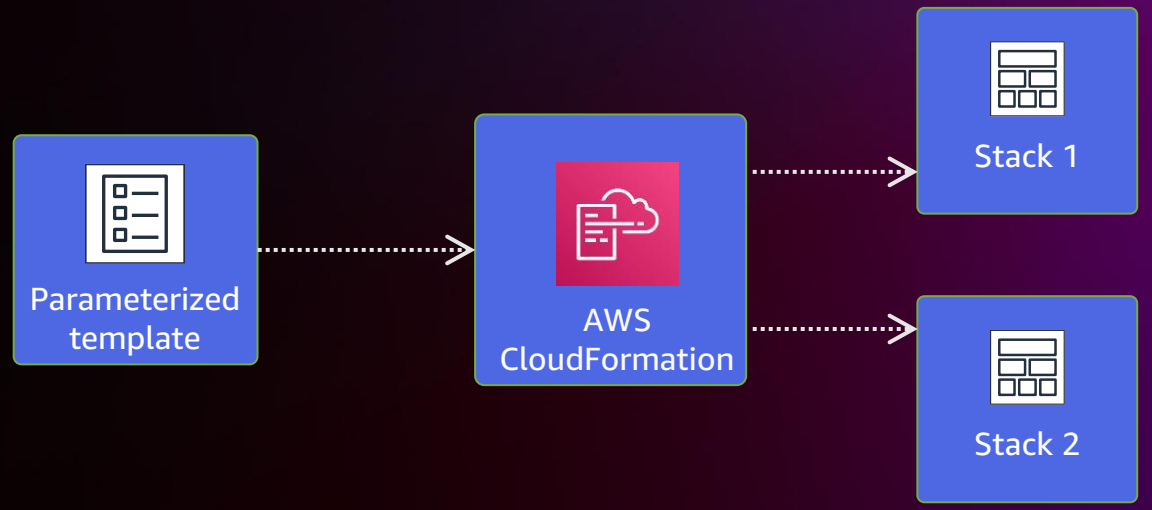
Supporting United’s new cloud workforce

- Meeting teams where they’re at
- Centrally managed living common patterns
- Policies encapsulated and updated behind the scenes
- Extensive customization for advanced teams

Advanced IaC tooling

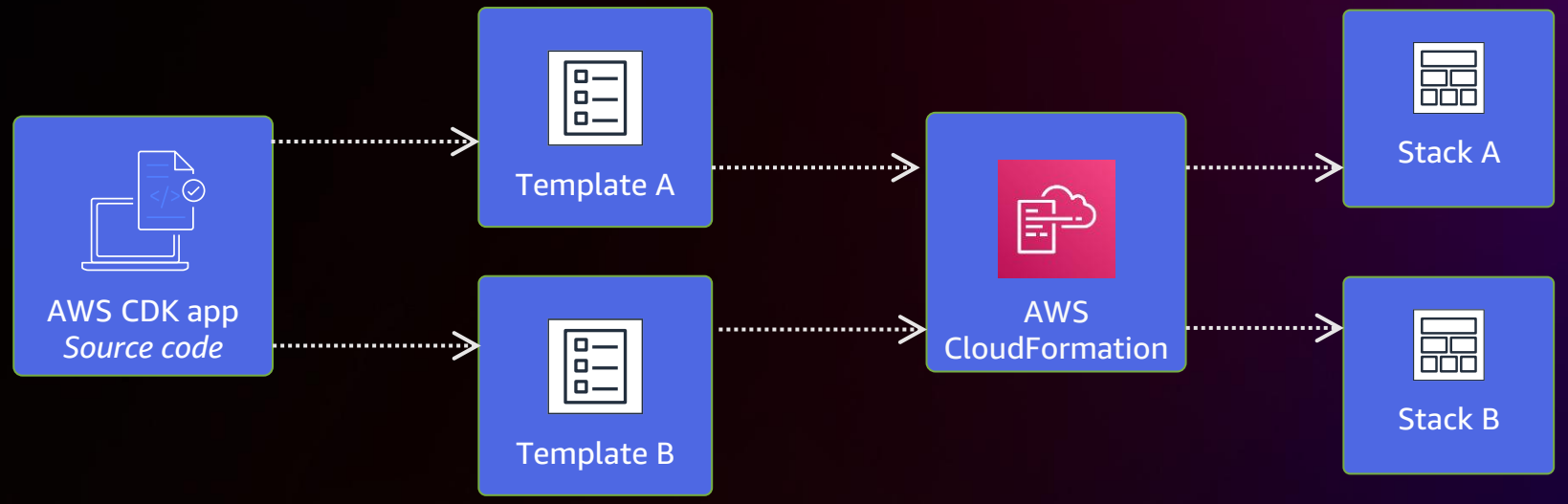
CloudFormation

Parameters and
intrinsic functions



AWS CDK

Typed OO language:
loops, conditions,
inheritance, etc.



AWS CDK

A MULTI-LANGUAGE SOFTWARE DEVELOPMENT FRAMEWORK FOR MODELING CLOUD INFRASTRUCTURE AS REUSABLE COMPONENTS

```
class UrlShortener extends Stack {
  constructor(scope: App, id: string, props?: UrlShortenerProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
    const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
      cluster,
      taskImageOptions: {
        image: ecs.ContainerImage.fromAsset('ping'),
      },
      dom
    });
    // Setup AutoScaling policy
    const scaling = service.service.autoScaleTasl
    scaling.scaleOnCpuUtilization('CpuScaling',
      targetUtilizationPercent: 50,
      scaleInCooldown: Duration.seconds(60),
      scaleOutCooldown: Duration.seconds(60)
    });
  }
}
```

domainName
domainZone

(property) patterns.NetworkLoadBalancedServiceBaseProps.domainName?: string | undefined

The domain name for the service, e.g. "api.example.com."

@default

- No domain name.

Familiar
Your language
Just code

Tool support
Autocomplete
Inline documentation

Abstraction
Sane defaults
Reusable classes

JS TS Java C# GO

AWS CDK

A MULTI-LANGUAGE SOFTWARE DEVELOPMENT FRAMEWORK FOR MODELING CLOUD INFRASTRUCTURE AS REUSABLE COMPONENTS

```
class UrlShortener extends Stack {
  constructor(scope: App, id: string, props?: UrlShortenerProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
    const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
      cluster,
      taskImageOptions: {
        image: ecs.ContainerImage.fromAsset('ping'),
      },
      dom
    });
    // Setup AutoScaling policy
    const scaling = service.service.autoScaleTasl
    scaling.scaleOnCpuUtilization('CpuScaling',
      targetUtilizationPercent: 50,
      scaleInCooldown: Duration.seconds(60),
      scaleOutCooldown: Duration.seconds(60)
    });
  }
}
```

domainName
domainZone

(property) patterns.NetworkLoadBalancedServiceBaseProps.domainName?: string | undefined

The domain name for the service, e.g. "api.example.com."

@default

- No domain name.

Familiar
Your language
Just code

Tool support
Autocomplete
Inline documentation

Abstraction
Sane defaults
Reusable classes

JS TS Java C# GO

AWS CDK

A MULTI-LANGUAGE SOFTWARE DEVELOPMENT FRAMEWORK FOR MODELING CLOUD INFRASTRUCTURE AS REUSABLE COMPONENTS

```
class UrlShortener extends Stack {
  constructor(scope: App, id: string, props?: UrlShortenerProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
    const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
      cluster,
      taskImageOptions: {
        image: ecs.ContainerImage.fromAsset('ping'),
      },
      dom
    });
    // Setup AutoScaling policy
    const scaling = service.service.autoScaleTasl
    scaling.scaleOnCpuUtilization('CpuScaling',
      targetUtilizationPercent: 50,
      scaleInCooldown: Duration.seconds(60),
      scaleOutCooldown: Duration.seconds(60)
    });
  }
}
```

domainName
domainZone

(property) patterns.NetworkLoadBalancedServiceBaseProps.domainName?: string | undefined

The domain name for the service, e.g. "api.example.com."

@default

- No domain name.

Familiar
Your language
Just code

Tool support
Autocomplete
Inline documentation

Abstraction
Sane defaults
Reusable classes

Java

AWS CDK

A MULTI-LANGUAGE SOFTWARE DEVELOPMENT FRAMEWORK FOR MODELING CLOUD INFRASTRUCTURE AS REUSABLE COMPONENTS

```
class UrlShortener extends Stack {
  constructor(scope: App, id: string, props?: UrlShortenerProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
    const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
      cluster,
      taskImageOptions: {
        image: ecs.ContainerImage.fromAsset('ping'),
      },
      dom
    });
    // Setup AutoScaling policy
    const scaling = service.service.autoScaleTasl
    scaling.scaleOnCpuUtilization('CpuScaling',
      targetUtilizationPercent: 50,
      scaleInCooldown: Duration.seconds(60),
      scaleOutCooldown: Duration.seconds(60)
    });
  }
}
```

domainName
domainZone

(property) patterns.NetworkLoadBalancedServiceBaseProps.domainName?: string | undefined

The domain name for the service, e.g. "api.example.com."

@default

- No domain name.

Familiar
Your language
Just code

Tool support
Autocomplete
Inline documentation

Abstraction
Sane defaults
Reusable classes

Java

Construct levels

L3+	Purpose-built constructs	Opinionated abstractions
L2	AWS constructs	High-level service constructs
L1	CloudFormation resources	Automatically generated

AWS Construct Library

Serverless

- AWS Lambda
- Amazon API Gateway
- Amazon DynamoDB

Application integration/foundational services

- Amazon S3
- Amazon SNS
- Amazon SQS
- AWS Step Functions
- Amazon CloudWatch
- AWS Identity and Access Management

Containers

- Amazon ECS
- AWS Fargate
- Amazon VPC
- Amazon EC2

CI/CD

- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline

L1

```
new CfnBucket(this, 'MyBucket', {bucketName: 'my-bucket'})
```

cdk synth

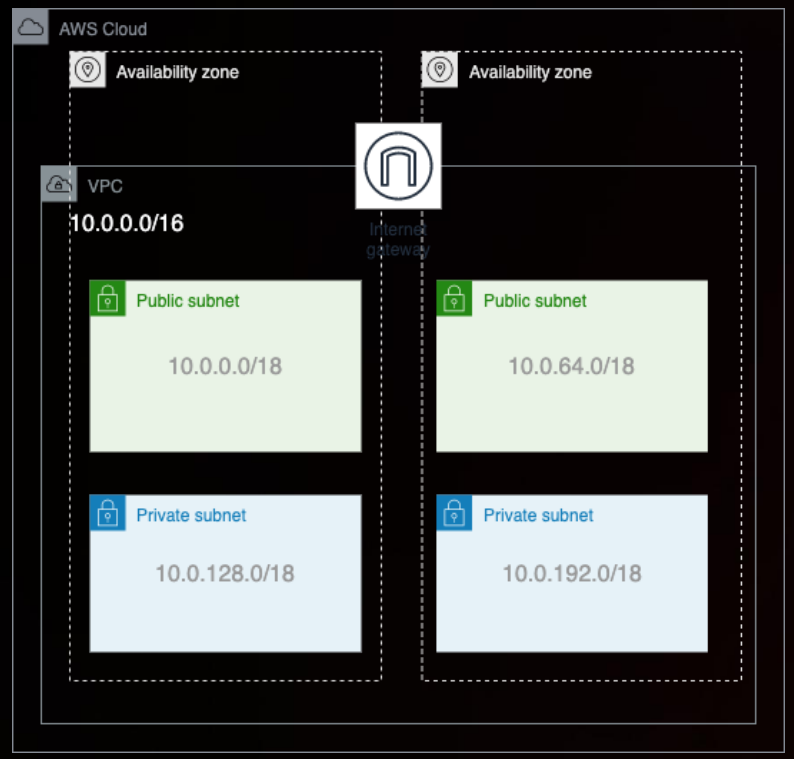
```
Resources:
  myBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: my-bucket
```

- Generated mappings from CloudFormation specification
- abc.CfnXyz → AWS::ABC::XYZ CloudFormation resource
- ec2.CfnInstance → AWS::EC2::Instance
- kms.CfnKey → AWS::KMS::Key

L2

```
new ec2.Vpc(this, 'MyVPC')
```

cdk synth



- Ready-to-use VPC setup
- 65,536 IPs split equally between 4 subnets
- If you provide a Region → adjust to 3 AZs
- Everything is optional; change any parameter
- Sane default values

L3

```
new ecs_patterns.ApplicationLoadBalancedFargateService(this, "FargateService", {
  cluster,
  taskImageOptions: {
    image: ecs.ContainerImage.fromRegistry("amazon/amazon-ecs-sample");
  }
});
```

829-line CloudFormation template

Amazon VPC

- Subnets
- EIP
- NAT gateways
- Internet gateway
- Route
- Route table

Elastic Load Balancing

- Security group
- Security group egress
- Security group ingress
- Task definition
- Listener
- Target group

AWS Fargate

- IAM roles
- IAM policies
- Log group
- Configuration

Amazon ECS task definition

- Image
- CPU
- Memory
- Port

Connecting stuff with code

```
table.grantReadData(role);
```

- Least-privilege policy
- No advanced knowledge of IAM
- No advanced use of CloudFormation

```yaml
MyTableReaderRoleDefaultPolicyB4C9E27D:
  Type: AWS::IAM::Policy
  Properties:
    PolicyDocument:
      Statement:
        - Action:
            - dynamodb:BatchGetItem
            - dynamodb:ConditionCheckItem
            - dynamodb:DescribeTable
            - dynamodb:GetItem
            - dynamodb:GetRecords
            - dynamodb:GetShardIterator
            - dynamodb:Query
            - dynamodb:Scan
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - MyTableTableAE194613
                - Arn
            - Ref: AWS::NoValue
      Version: "2012-10-17"
    PolicyName: MyTableReaderRoleDefaultPolicyB4C9E27D
    Roles:
      - Ref: MyTableReaderRole7F6B2E9A
```

CDK Quick Wins at United

- Tagging
- Permissions boundaries
- Complex Amazon ECS reference architecture

Aspects remove traditional policy speed bumps

- New cloud users don’t have PhDs in IAM
- Tagging, naming, etc. don’t generally affect applications
- Policies need to appear in infrastructure as code

```
const boundary = iam.ManagedPolicy.fromManagedPolicyArn(
  this,
  'permissions-boundary',
  `arn:aws:iam::${
    this.account
  }:policy/AppPolicy_${props.applicationCI.toUpperCase()}`
);
iam.PermissionsBoundary.of(this).apply(boundary);

cdk.Tags.of(this).add('ApplicationCI', this.applicationCI);
cdk.Tags.of(this).add('Environment', this.ualEnvironment);
cdk.Tags.of(this).add('Region', this.region);
```

L2 at United

```
const table = new UalTable(this, "MyTable", {
    partitionKey: {
        name: "pk",
        type: AttributeType.STRING,
    },
});
```

```
MyTableTableAE194613:
  Type: AWS::DynamoDB::Table
  Properties:
    KeySchema:
      - AttributeName: pk
        KeyType: HASH
    AttributeDefinitions:
      - AttributeName: pk
        AttributeType: S
    BillingMode: PAY_PER_REQUEST
    SSESpecification:
      SSEEnabled: true
    Tags:
      - Key: ApplicationCI
        Value: abc
      - Key: Environment
        Value: dev
      - Key: Region
        Value: us-east-2
      - Key: ual-cdk:version
        Value: 4.4.0
```

- Policy-compliant Amazon DynamoDB table
- Encryption enabled
- All tags enforced
- SSM Parameter for CloudFormation-generated name

```
MyTableTableMyTableSSMParameter86BAEA87:
  Type: AWS::SSM::Parameter
  Properties:
    Type: String
    Value:
      Ref: MyTableTableAE194613
    Name: /abc/dynamodb/dev/MyTable_TABLE_NAME
    Tags:
      ApplicationCI: abc
      Environment: dev
      Region: us-east-2
      ual-cdk:version: 4.4.0
```

Aspects at United

```
const role = new Role(this, "MyTableReaderRole", {
    assumedBy: new AccountPrincipal(this.account),
});
```

- Policy-compliant IAM role
- All tags enforced
- Permissions boundary applied

```yaml
MyTableReaderRole7F6B2E9A:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: sts:AssumeRole
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - :iam::12345678910:root
      Version: "2012-10-17"
    PermissionsBoundary: arn:aws:iam::12345678910:policy/AppPolicy_ABC
    Tags:
      - Key: ApplicationCI
        Value: abc
      - Key: Environment
        Value: dev
      - Key: Region
        Value: us-east-2
      - Key: ual-cdk:version
        Value: 4.4.0
```

United-specific ECS reference architecture

- 34 lines of TypeScript = 451 lines of CloudFormation
- Private hosted zone, Amazon ECS cluster, VPC, and subnet lookups
- Built in instrumentation

```typescript
import { UalStack, UalStackProps } from "ual-cdk";
import { ecs } from "ual-cdk";
import { aws_ecs as awsEcs } from "aws-cdk-lib";
import { ecs_patterns as ecsPatterns } from "ual-cdk";
import { Construct } from "constructs";

export class UalCdkDocsStack extends UalStack {
    constructor(scope: Construct, id: string, props: UalStackProps) {
        super(scope, id, props);

        const cluster = new ecs.UalCluster(this, "DocsCluster", {
            useIdInClusterNameSSMParameterName: true,
        });

        const albfs = new ecsPatterns.UalApplicationLoadBalancedFargateService(
            this,
            "ualcdk",
            {
                certificateArn:
                    this.region == "us-east-1"
                        ? "arn:aws:acm:us-east-1:12345678910:certificate/*****"
                        : "arn:aws:acm:us-east-2:12345678910:certificate/*****",
                cluster: cluster,
                containerPort: 80,
                taskImageOptions: {
                    image: awsEcs.ContainerImage.fromAsset("../"),
                    containerPort: 80,
                    enableLogging: true,
                },
                circuitBreaker: { rollback: true },
            }
        );
    }
}
```

Happy surprises

- No more hardcoded VPC IDs
- Amazon ECS workloads on AWS Fargate Spot in dev
- Change management
- Auto-generated documentation

United CDK apps look up their own networks

```
// Load vpc by id or default logic
if (props && props.vpcId) {
  this.vpc = ec2.Vpc.fromLookup(this, `${id}Vpc`, {
    vpcId: props.vpcId,
    subnetGroupNameTag: 'Name',
    isDefault: props.isDefault != null ? props.isDefault : false,
  });
} else {
  if (props?.isDefault) {
    console.warn(
      'isDefault defaults to false if vpcId is not provided to UalNetwork construct'
    );
  }
  this.vpc = ec2.Vpc.fromLookup(this, `${id}Vpc`, {
    isDefault: false,
    vpcName: `VPC-${ualEnvironment.toUpperCase()}*-${stack.region}*`,
    subnetGroupNameTag: 'Name',
  });
}
```

```
let ualNetwork: UalNetwork;
if (!props.vpcId) {
  ualNetwork = new UalNetwork(scope, `${id}UalNetwork`, {
    vpcId: props.vpcId,
  });
} else {
  ualNetwork = stack.getUalNetwork();
}
generated.vpcSubnets = {
  subnets: ualNetwork.subnets.appTier,
};
```

Fargate applications use Fargate Spot

```
generated.capacityProviderStrategies = [
  {
    capacityProvider:
      stack.ualEnvironment == UalEnvironment.DEV
        ? 'FARGATE_SPOT'
        : 'FARGATE',
    weight: 100,
  },
];
```

CDK construct libraries have versions and releases

v4.7.5

4.7.5 (2022-10-05)

Bug Fixes

- expose UalCluster nodegroup (#112) (a80b42d)
- make cdk bootstrap qualifier optional, or must be base (#110) (ae90814)

▸ Assets 2

United's CDK construct library auto-generates documentation

ual-cdk / appmesh_patterns / UalNetworkLoadBalancedVirtualGatewayServiceProps /

Interface UalNetworkLoadBalancedVirtualGatewayServiceProps

Properties required for defining a UalNetworkLoadBalancedVirtualGatewayService

Hierarchy

- **UalNetworkLoadBalancedVirtualGatewayServiceProps**

Index

Properties

- backendPorts
- backendSubjectAlternativeNames
- cloudExternal
- createDnsRecord
- domainNamePrefix
- ecsCluster
- enforceBackendTls
- envoyProxyImage
- envoyProxyPort
- gatewayNameSSMParameterName
- hostedZoneDomainName
- loadBalancerListenerCertificateArns
- loadBalancerPort
- loadBalancerProtocol
- mesh
- serviceName
- virtualGateway
- virtualGatewayAccessLog
- virtualGatewayListener
- virtualGatewayName
- vpcId
- weight

Exports

- appmesh
- **appmesh_patterns**
- dynamodb
- ecs
- ecs_patterns
- eks
- elbv2
- lambda
- servicediscovery
- sns
- sqs

UalNetworkLoadBalancedVirtualGatewayServiceProps

- backendPorts
- backendSubjectAlternativeNames
- cloudExternal
- createDnsRecord
- domainNamePrefix
- ecsCluster

It gets more fun

This . . .

```
export class MyTableStack extends UalStack {
    constructor(scope: Construct, id: string, props: UalStackProps) {
        super(scope, id, props);

        const role = new Role(this, "MyTableReaderRole", {
            assumedBy: new AccountPrincipal(this.account),
        });

        const table = new UalTable(this, "MyTable", {
            partitionKey: {
                name: "pk",
                type: AttributeType.STRING,
            },
        });

        table.grantReadData(role);
    }
}
```

. . . is the same as this . . .

```python
class MyTableStack(UalStack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        role = iam.Role(self, "MyTableReaderRole",
                        iam.AccountPrincipal(self.account))

        table = dynamodb.UalTable(self, "MyTable", partition_key=aws_dynamodb.Attribute(
            name="pk", type=aws_dynamodb.AttributeType.STRING))

        table.grant_read_data(role)
```

. . . is the same as this . . .

```java
public class MyTableStack extends UalStack {

    public MyTableStack(final Construct scope, final String id, final UalStackProps props) {
        super(scope, id, props);

        final Role role = new Role(this, id: "MyTableReaderRole",
                RoleProps.builder().assumedBy(new AccountPrincipal(this.getAccount())).build());

        final UalTable table = new UalTable(this, "MyTable",
                UalTableProps.builder()
                        .partitionKey(Attribute.builder().name(name: "pk").type(AttributeType.STRING).build()).build());

        table.grantReadData(role);
    }
}
```

Java

. . . is the same as this

```
public class MyTableStack : UalStack
{
    0 references
    internal MyTableStack(Construct scope, string id, IUalStackProps props) : base(scope, id, props)
    {
        var role = new Role(this, "MyTableReaderRole", new RoleProps
        {
            AssumedBy = new AccountPrincipal(this.Account)
        });

        var table = new UalTable(this, "MyTable", new UalTableProps
        {
            PartitionKey = new Attribute
            {
                Name = "pk",
                Type = AttributeType.STRING
            }
        });

        table.GrantReadData(role);
    }
}
```

“Good programmers know what to write. Great ones know what to rewrite (and reuse).”

Eric S. Raymond

American software developer, open-source software advocate

CloudFormation Community Extensions

aws-cloudformation / community-registry-extensions Public

Edit Pins | Unwatch

Code | Issues 13 | Pull requests 4 | Discussions | Actions | Projects 1 | Wiki | Security 2

main | 4 branches | 7 tags | Go to file | Add file | Code

ericzbeard Update README.md 00d07fa yesterday 68 commits

.github	Fix working directory for GitHub action (#90)	2 days ago
config	chore(release): Add bandit to CICD for Python projects (#45)	last month
hooks	chore(release): Prod redesign (#81)	3 days ago
packages	chore(package) - Handle errors in cfn guard rs hook (#88)	2 days ago
release	chore(release): Publish to multiple regions (#97)	2 days ago
resources	chore(release): Release script and CLI publishing (#95)	2 days ago
scripts	chore(release): Fix roles for publishing (#96)	2 days ago

Learn in-demand AWS Cloud skills

AWS Skill Builder

Access **500+ free** digital courses and Learning Plans

Explore resources with a variety of skill levels and **16+** languages to meet your learning needs

Deepen your skills with digital learning on demand

Train now

AWS Certifications

Earn an industry-recognized credential

Receive Foundational, Associate, Professional, and Specialty certifications

Join the **AWS Certified community** and get exclusive benefits

Access **new** exam guides

Thank you!

Ravi Palakodeti
rpalakod@amazon.com

Ryan Bachman
ryanbach@amazon.com

Ethan Rucinski
ethan.rucinski@united.com

Please complete the session survey in the **mobile app**