Ran The Builder
Build Serverless Services

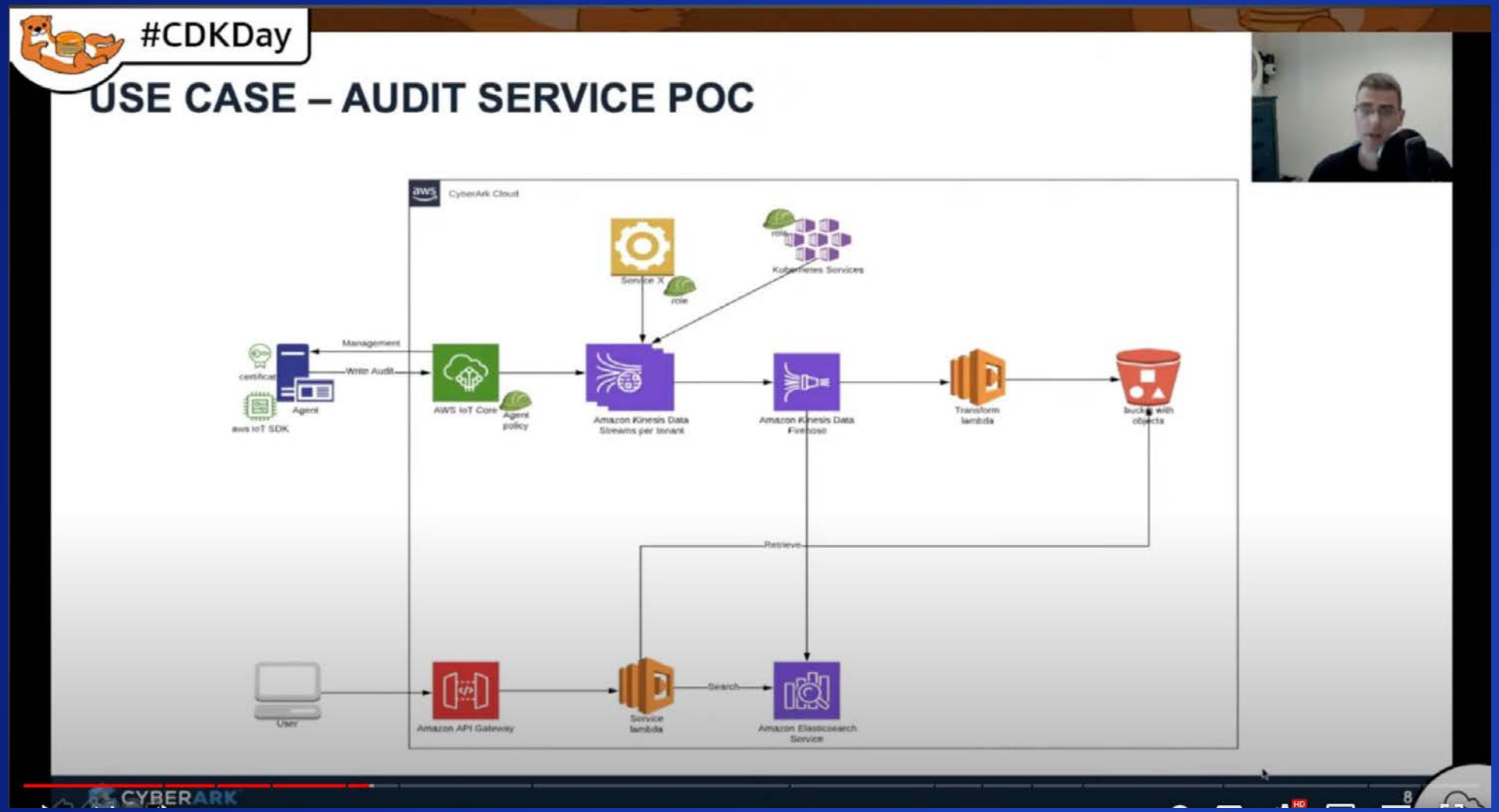
AWS CDK – BEST PRACTICES FROM THE TRENCHES

RAN ISENBERG, PRINCIPAL SOFTWARE ARCHITECT

CYBERARK®

"AWS CDK LETS YOU BUILD RELIABLE, SCALABLE, COST-EFFECTIVE APPLICATIONS IN THE CLOUD WITH THE CONSIDERABLE EXPRESSIVE POWER OF A PROGRAMMING LANGUAGE" – AWS DOCS

AWS CDK DAY 2020



HOWEVER,

Uncle ben, 2002, Spiderman

AGENDA

- CDK App guidelines
- Constructs guidelines
- CI/CD guidelines
- Security
- Resilience
- General development tips
- Summary

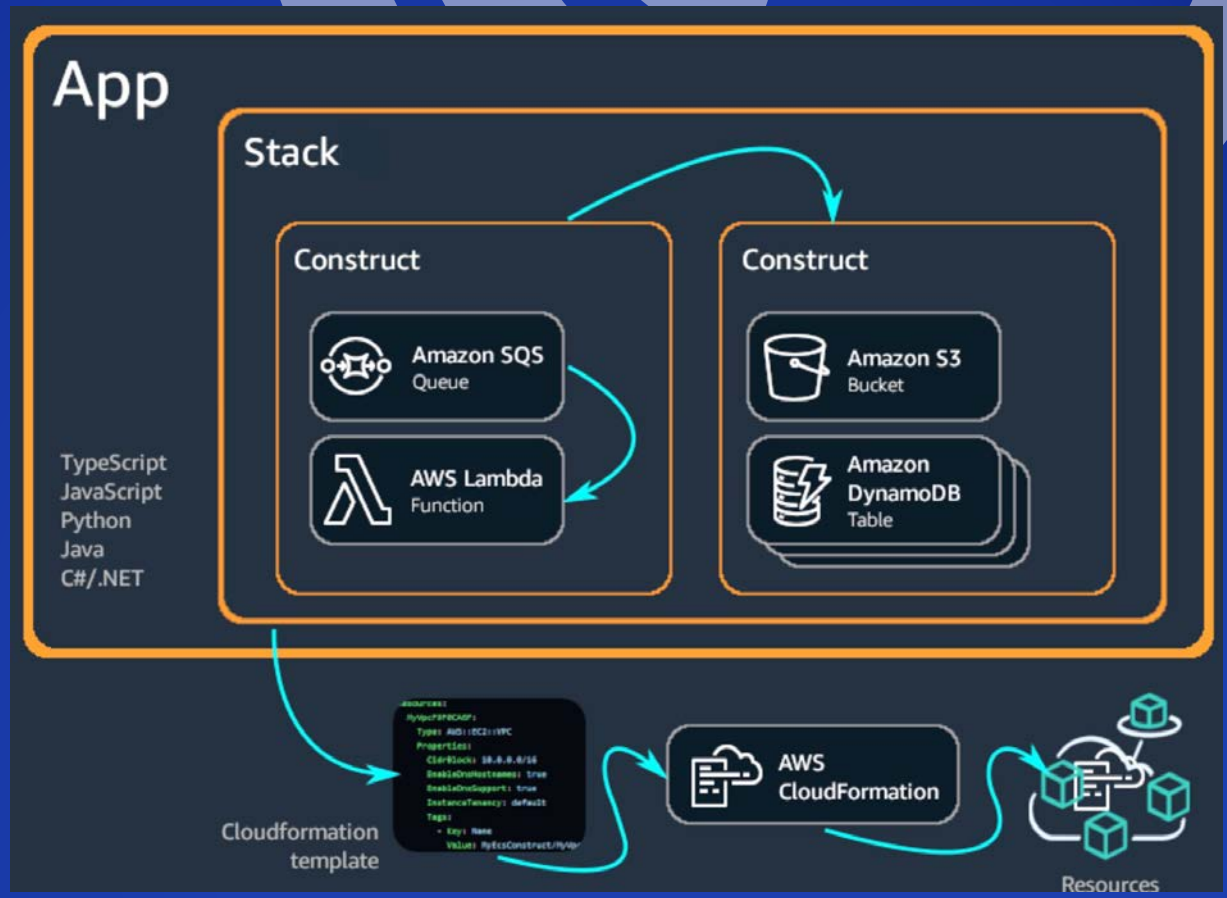
INTRODUCTION

- Principal Software Architect @CyberArk
- AWS Community Builder
- Owner & Blogger @RanTheBuilder.Cloud

CDK APP GUIDELINES

CDK APP GUIDELINES

- One business domain
- One repository & CI/CD pipeline
- Maintained by one team
- One CDK application & stack
- Small blast radius



CDK documentation

CDK APP Guidelines

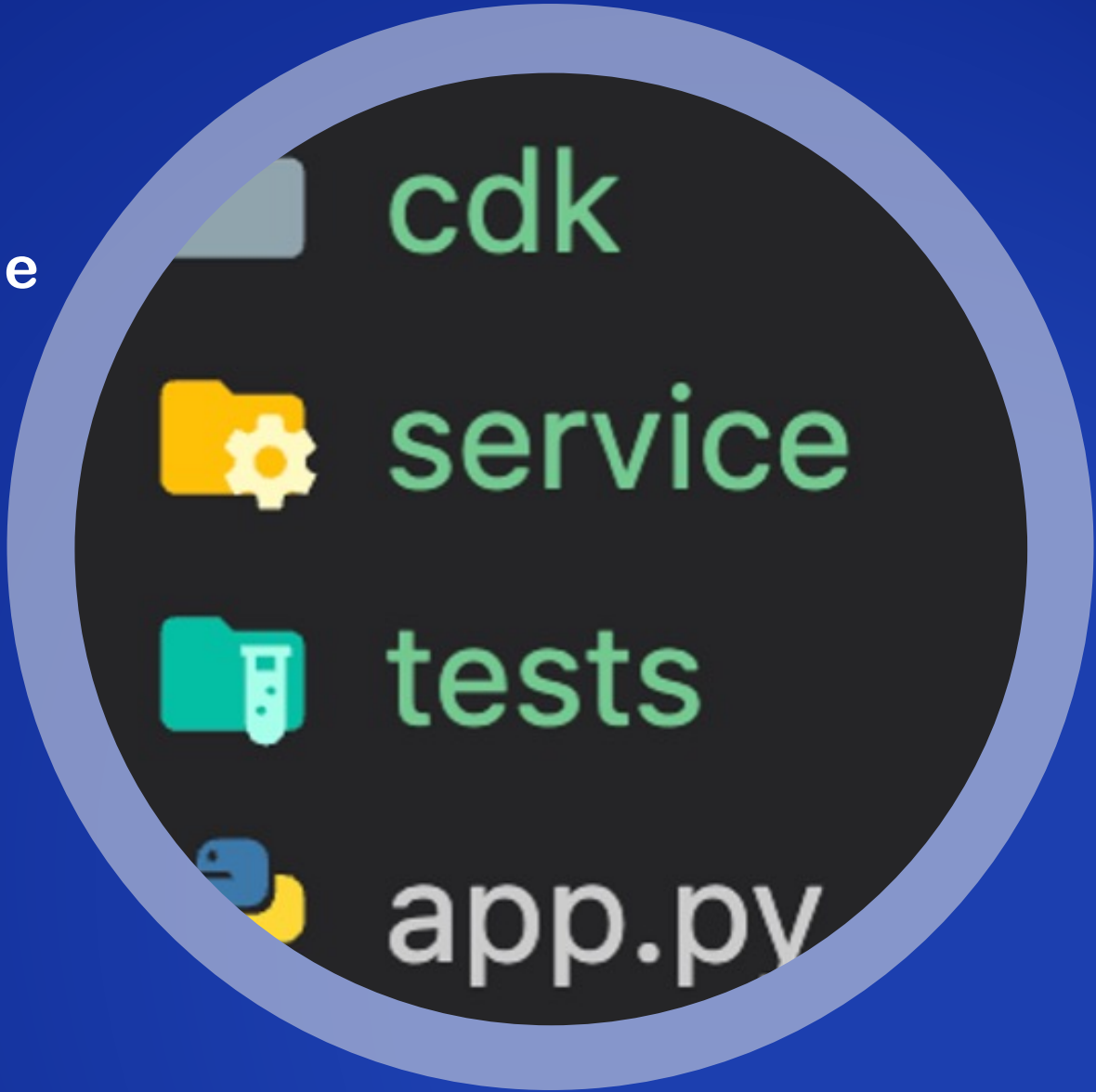
- When to split to a new application & repository:
  1. Different team will maintain the new application
  2. Different business domain

- Don’t over split! Balance is key

- Multiple repositories:
  - Increase development complexity of new cross repo features
  - Share deployment time parameters (SSM/CloudMap)

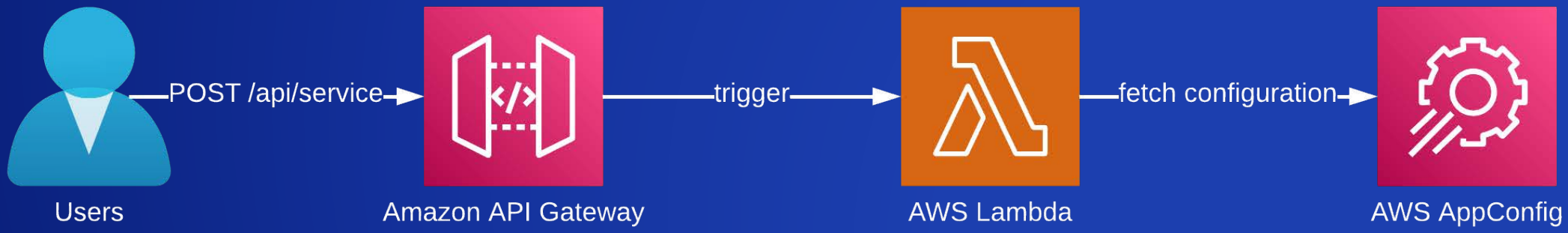
PROJECT STRUCTURE GUIDELINES

- IaC and business domain code together
- One CI/CD pipeline
- Tests:
  - Unit / integration / e2e
  - Security & CDK infra



CDK Template Project

- Self service
- Internal training
- Reduce cognitive load
- Jump start into SaaS development
- Organization level: same tools, CI/CD pipeline, tech stack



- https://github.com/ran-isenberg/aws-lambda-handler-cookbook

CONSTRUCT GUIDELINES

Stack/Construct Composition

- Don’t define all resources in the stack
  - Use constructs
  - Exception - Lambda layer used in multiple constructs
- Constructs are easy to share

Shareable Constructs

- Platform engineers own & maintain

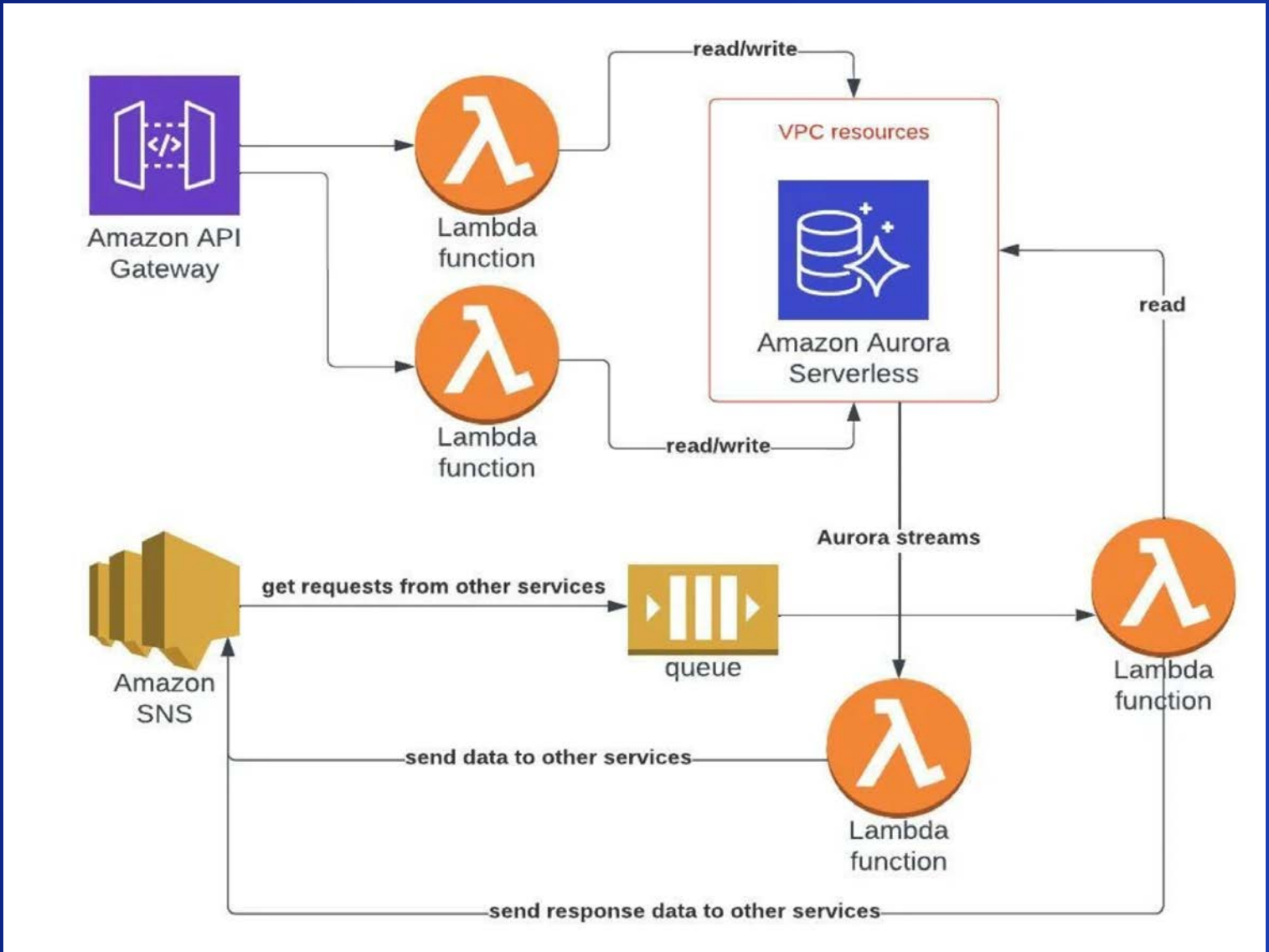
- Pros:
  - Secure, cost effective, tested constructs
  - Save time for developers
- Cons:
  - Versioned
  - Can cause breakage/resource deletion on upgrades

Shareable Constructs

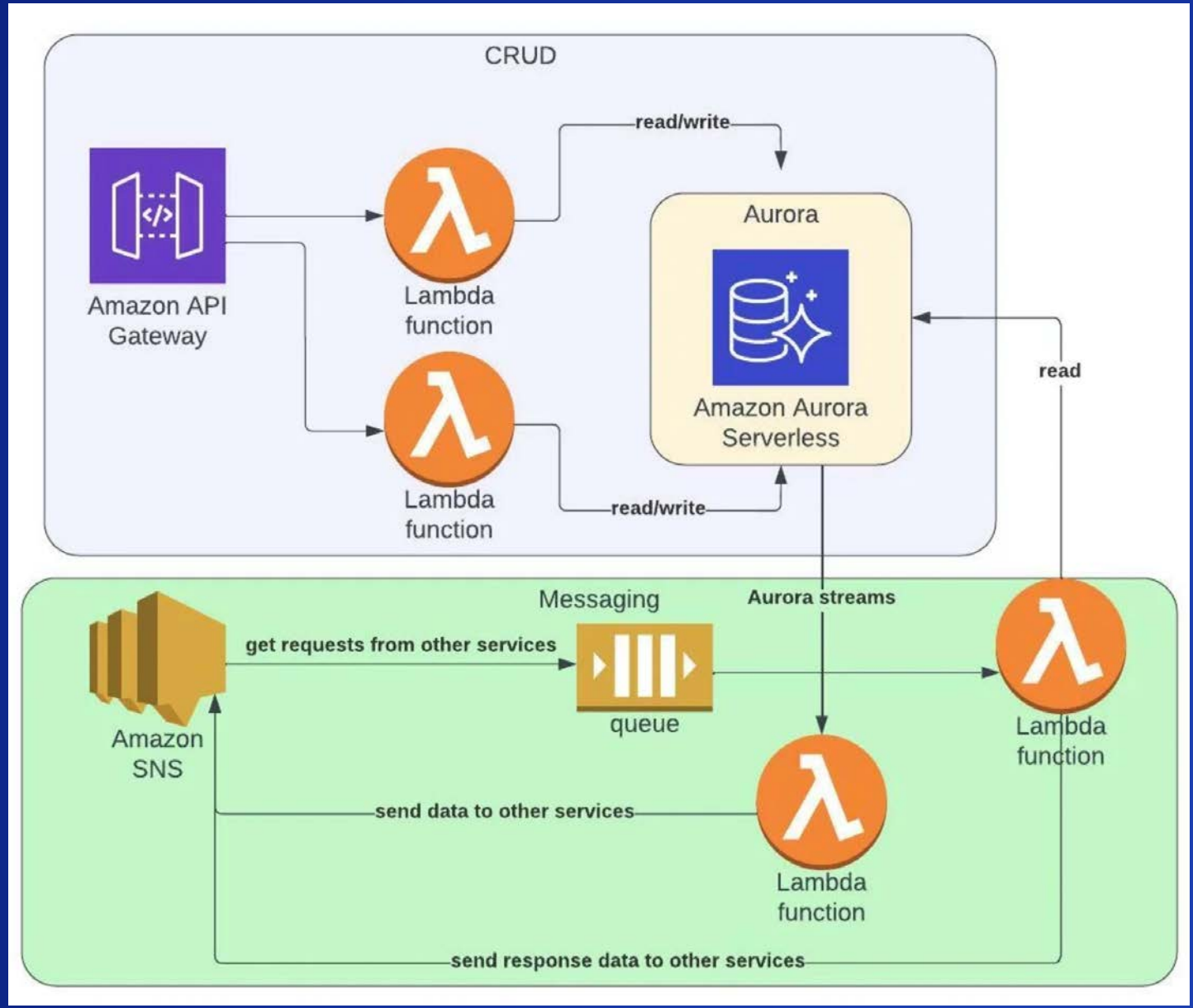
- Internal library of common constructs
  - WAF rules for API Gateway/CloudFront distributions.
  - SNS -> SQS pattern with encryption at REST
  - AWS AppConfig dynamic configuration construct.
  - Datadog logs shipper/log PII sanitizer

- External resources:
  - https://constructs.dev
  - Serverless land
  - cdkpatterns.com
  - https://aws.amazon.com/solutions/constructs

Business Domain Driven Constructs

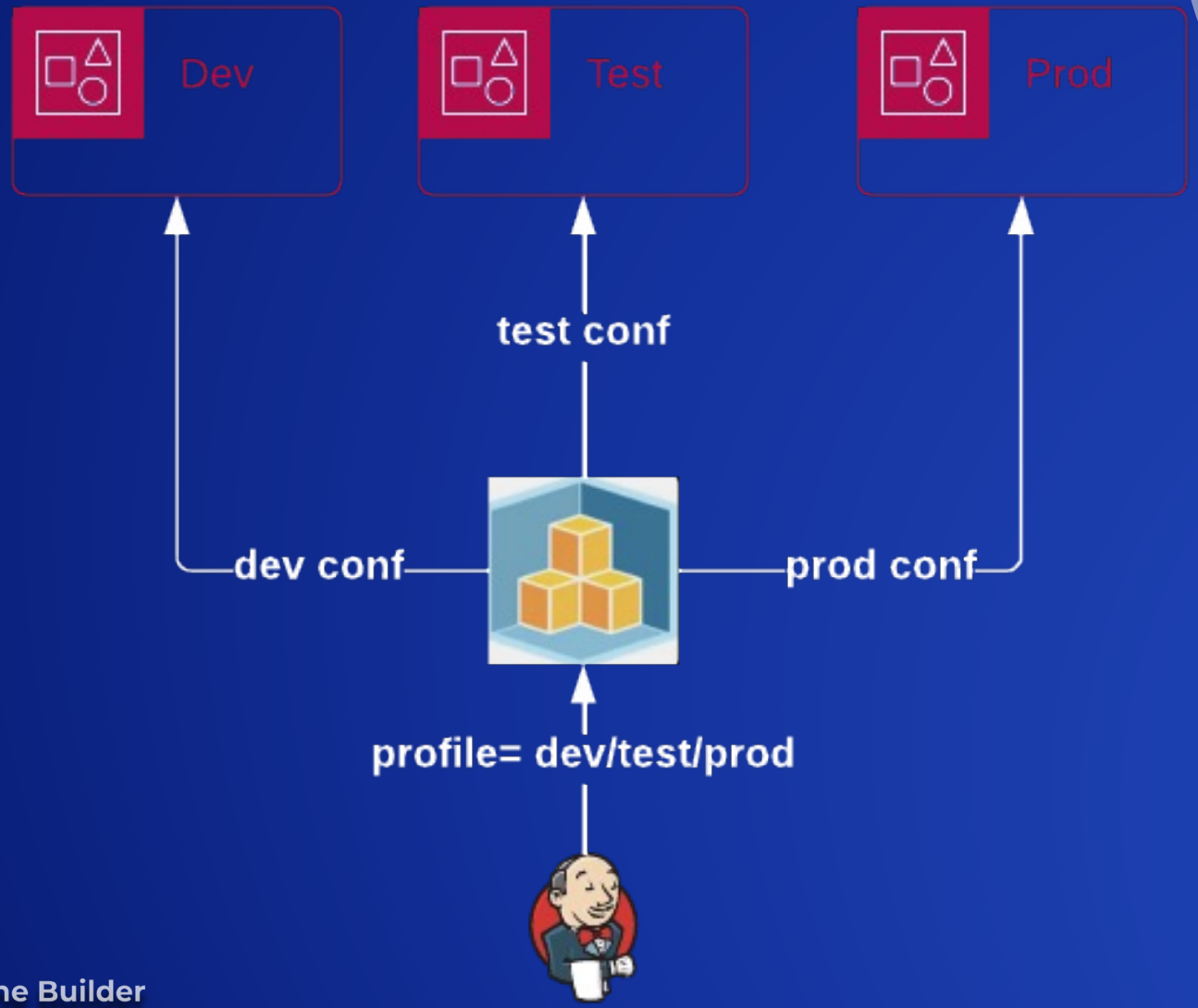


Business Domain Driven Constructs



CI/CD GUIDELINES

MODEL YOUR CI/CD PIPELINE STAGES IN CODE



Model Your CI/CD Pipeline Stages in Code

- Why multiple accounts?
  - Account breach smaller blast radius
  - AWS resource quota limits

- How to model stages in CDK?
  - Environment variables
  - ‘if statements’ for the win
  - Apply different configuration

Model Your CI/CD Pipeline Stages in Code

```
profile = os.getenv('PROFILE')
table = dynamodb.Table(
    self,
    table_id,
    table_name=table_id,
    partition_key=dynamodb.Attribute(name='order_id', type=dynamodb.AttributeType.STRING),
    billing_mode=dynamodb.BillingMode.PAY_PER_REQUEST,
    point_in_time_recovery=False if profile == 'dev' else True,
    removal_policy=RemovalPolicy.DESTROY if profile == 'dev' else RemovalPolicy.RETAIN,
)
```

SECURITY GUIDELINES

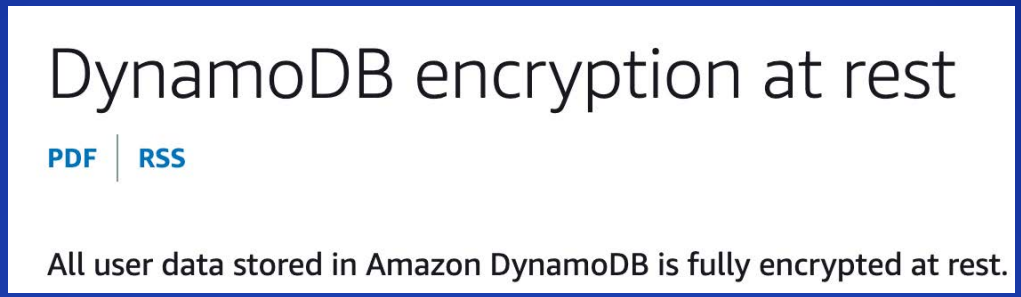
Secrets in CDK

- NEVER write hardcoded secrets in CDK or config files
- Store as GitHub/Jenkins/pipeline secret
  - Inject to CDK as environment variable/parameter during deploy

- Deploy secrets:
  - AWS Secrets Manager
  - SSM parameter store encrypted string

- Consume in Lambda from SSM/Secrets manager:
  - Secret name as lambda env. variable

Some Security Defaults Are Not Good Enough



- What about SNS encryption at rest?
  - Disabled by default
- Security defaults differ by service
- AWS sets better defaults over time

Some Security Defaults Are Not Good Enough

- Shared responsibility model
- Don’t expect AWS to do all the work for you
- Enable security best practices for all resources
- Security Review, scheduled PT
- Run CDK security tests – CDK nag

AWS CDK Security Tests

```python
from aws_cdk import App, Aspects
from cdk_nag import AwsSolutionsChecks, HIPAASecurityChecks
from cdk.my_service.service_stack import ServiceStack

def test_cdk_nag_default():
  app = App()

  service_stack = ServiceStack(app, 'service-test')
  Aspects.of(service_stack).add(AwsSolutionsChecks(verbose=True))


def test_cdk_nag_hipaa():
  app = App()

  service_stack = ServiceStack(app, 'service-test')
  Aspects.of(service_stack).add(HIPAASecurityChecks(verbose=True))
```

Write Your Own IAM Policies

```
def _build_db(self, id_prefix: str, my_role: iam.Role) -> dynamodb.Table:
table_id = f'{id_prefix}{constants.TABLE_NAME}'
table = dynamodb.Table(
    self,
    table_id,
    table_name=table_id,
    partition_key=dynamodb.Attribute(name='order_id', type=dynamodb.AttributeType.STRING),
    billing_mode=dynamodb.BillingMode.PAY_PER_REQUEST,
    point_in_time_recovery=True,
    removal_policy=RemovalPolicy.DESTROY,
)
table.grant_read_write_data(my_role)
return table
```

Grants: BatchGetItem, GetRecords, GetShardIterator, Query, GetItem, Scan, BatchWriteItem, PutItem, UpdateItem, DeleteItem, DescribeTable

Write Your Own IAM Policies

```
def _build_lambda_role(self, db: dynamodb.Table) -> iam.Role:
return iam.Role(
    self,
    'ServiceRole',
    assumed_by=iam.ServicePrincipal('lambda.amazonaws.com'),
    inline_policies={
    'dynamodb_db':
      iam.PolicyDocument(statements=[
        iam.PolicyStatement(actions=['dynamodb:PutItem', 'dynamodb:GetItem'], resources=[db.table_arn],
                            effect=iam.Effect.ALLOW)
    ]),},)
```

- Grants only GetItem, PutItem
- Prefer least privilege method – assign only what you need, no more, no less
- Better developers understand IAM policies

RESILIENCE GUIDELINES

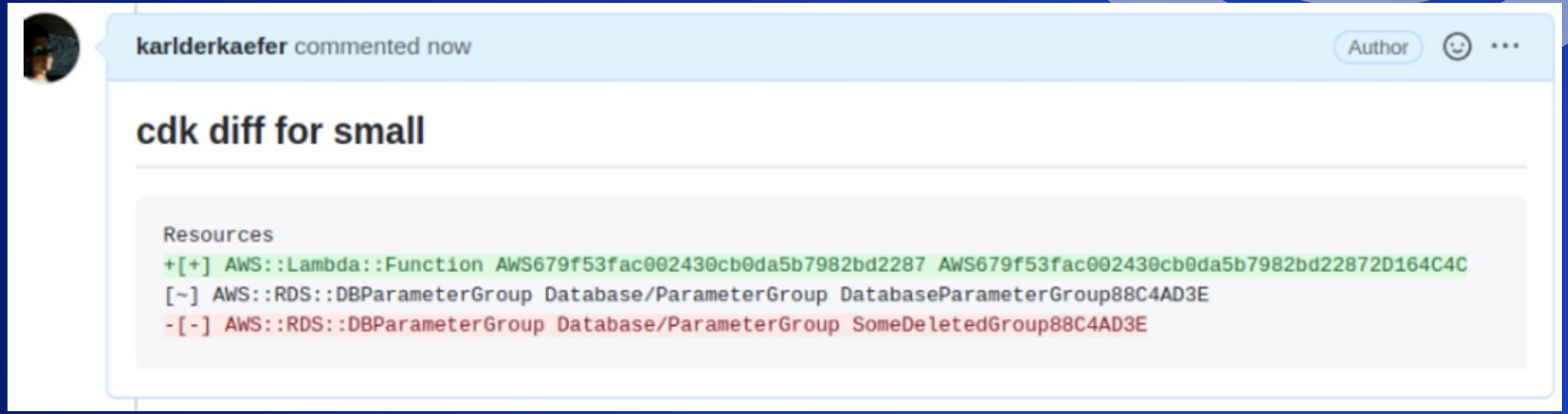
Changing Logical ID is Dangerous

- Unique resource ID
- Innocent refactor can be hazardous:
  - Stateful logical ids must NEVER change
  - Cross account trust role can break
- Critical resources can get deleted due to bugs
- Write CDK unit tests

CDK Unit Tests

```python
from aws_cdk import App
from aws_cdk.assertions import Template
from cdk.my_service.service_stack import ServiceStack

def test_synthesizes_properly():
  app = App()

  service_stack = ServiceStack(app, 'service-test’)

  # Prepare the stack for assertions.
  template = Template.from_stack(service_stack)

  # verify that we have one API GW, that is it not deleted by mistake
  template.resource_count_is('AWS::ApiGateway::RestApi', 1)
  table = template.find_resources('AWS::DynamoDB::Table’)
  # assert table’s key matches the logical id
```

CHANGES VISIBILITY



Backups

- Retain policy – RETAIN in production
  - Restore vs. lose customer data forever
- Backup your stateful resources:
  - DynamoDB point in time
  - AWS Backup

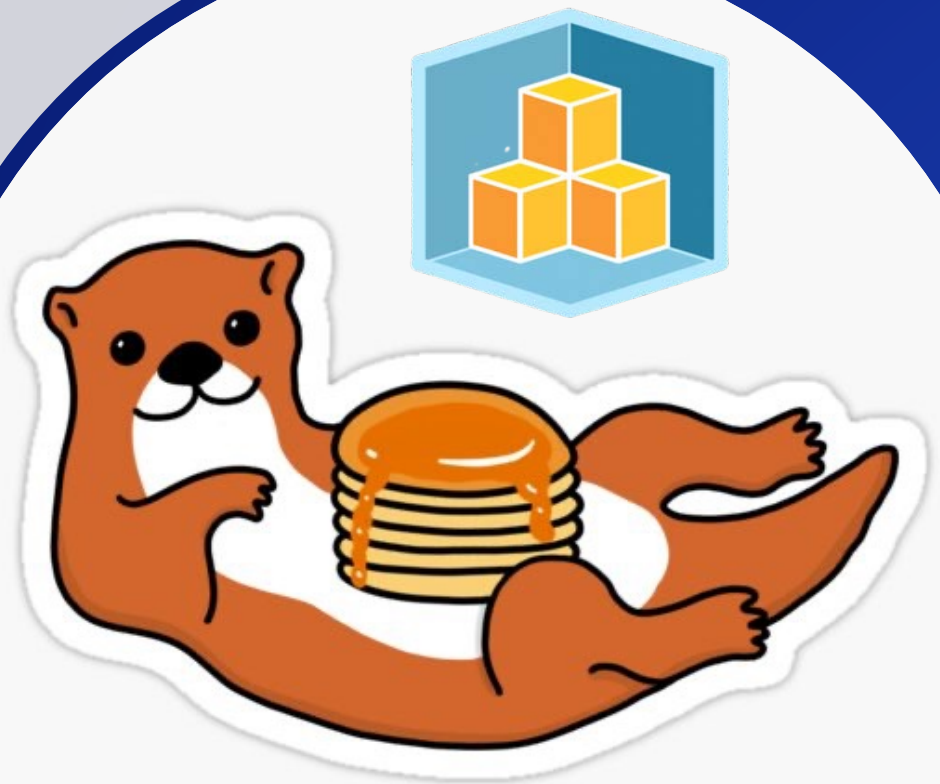
GENERAL DEVELOPMENT GUIDELINES

General Development Tips

- Console first approach
- CFN low level FTW
- Tag it!
- CDK code maintainability > abstraction
  - Avoid “cool” factory methods
  - Keep it simple
  - IaC must be readable and easy to follow

Summary

- With great power comes great responsibility
- Shared responsibility model
- Enforce best practices in organization:
  - CDK App, stack & construct guidelines
  - Share constructs
  - CDK Template self service
  - Security
  - Resilience



Ran The Builder
Build Serverless Services

CYBERARK®

THANK YOU!

@RANISENBERG

HTTPS://WWW.RANTHEBUILDER.CLOUD

@ISENBERGRAN