
AWS
re:Invent
DECEMBER 1 – 5, 2025 | LAS VEGAS, NV

CNS361

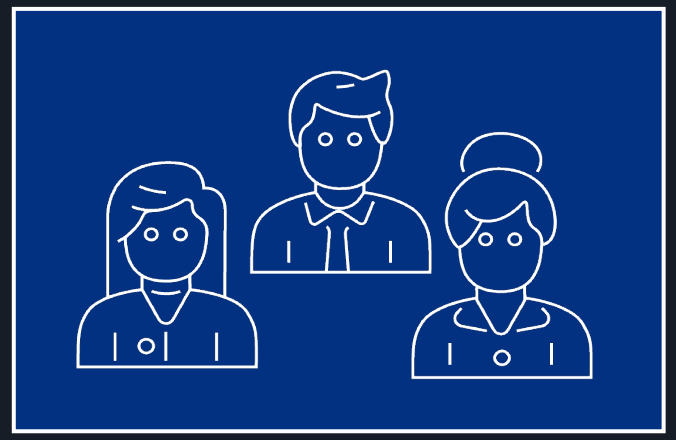
Scaling Serverless with platform engineering: A blueprint for success

Anton Aleksandrov
Principal Solutions Architect
AWS, Serverless

Ran Isenberg
Principal Software Architect, CyberArk
AWS Serverless Hero

aws

The operational model evolution

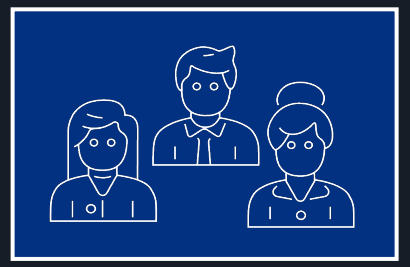


Application development team

Infrastructure/ops team

The operational model evolution

We ♥ DevOps

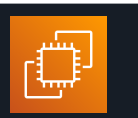


Application development team

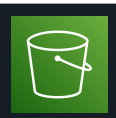
CI/CD tools and processes

The Application Layer

The Infrastructure Layer

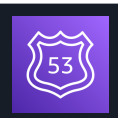


Amazon EC2



Amazon S3

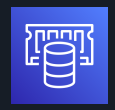
Amazon EBS



Amazon Route 53

Amazon EMR

Amazon RDS



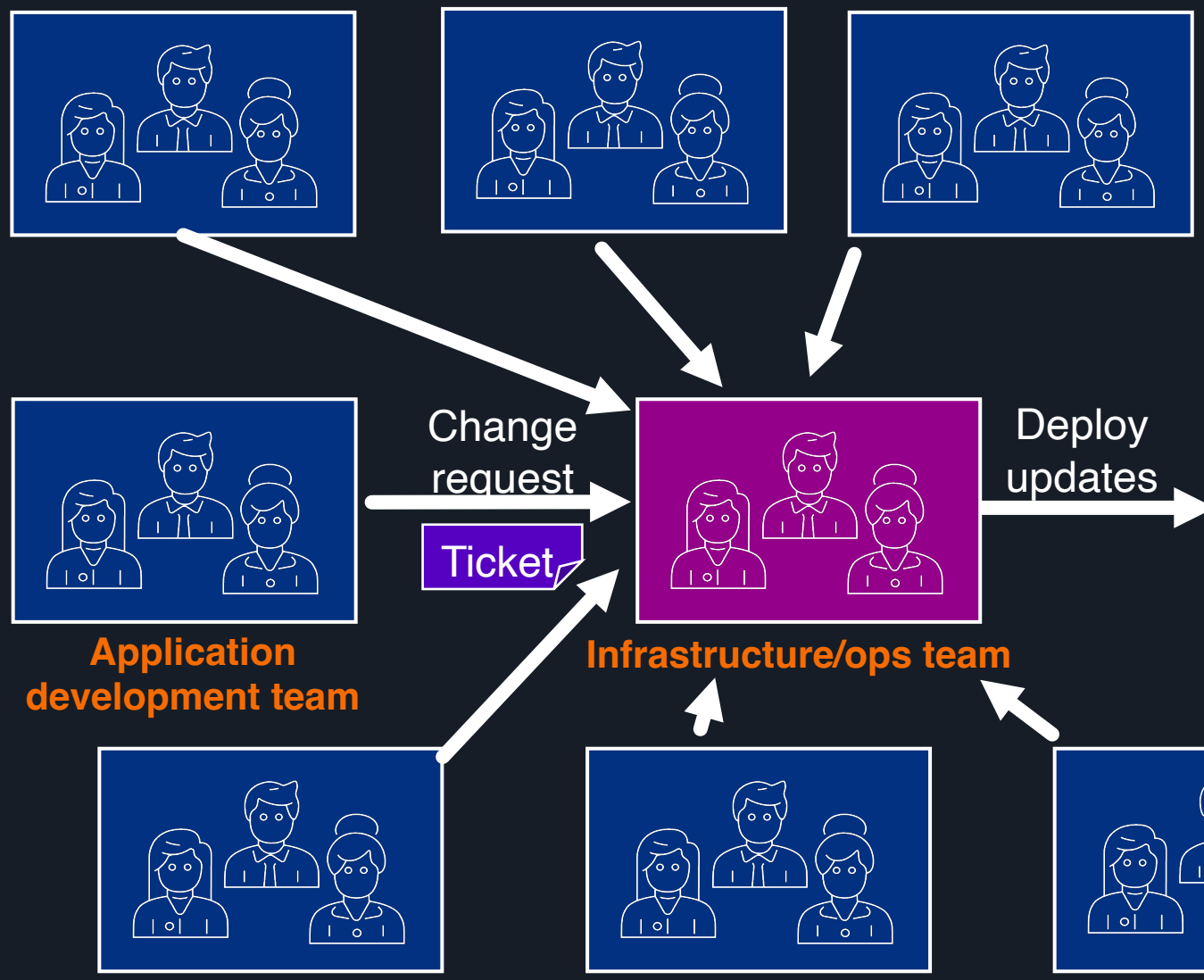
ElastiCache

Amazon SNS

IaC tools and processes

Infrastructure/ops team

The operational model evolution



Shift-left to the rescue?

?!

Application development teams

Cognitive Load

Code

Ship

Run

Secure

Optimize

Sustainable

2000

Today

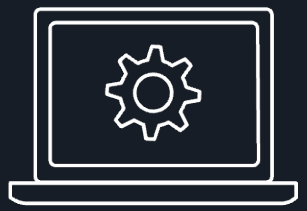
Why Serverless

Reduced operational overhead

Improved availability and scalability

Improved security posture

Native integrations

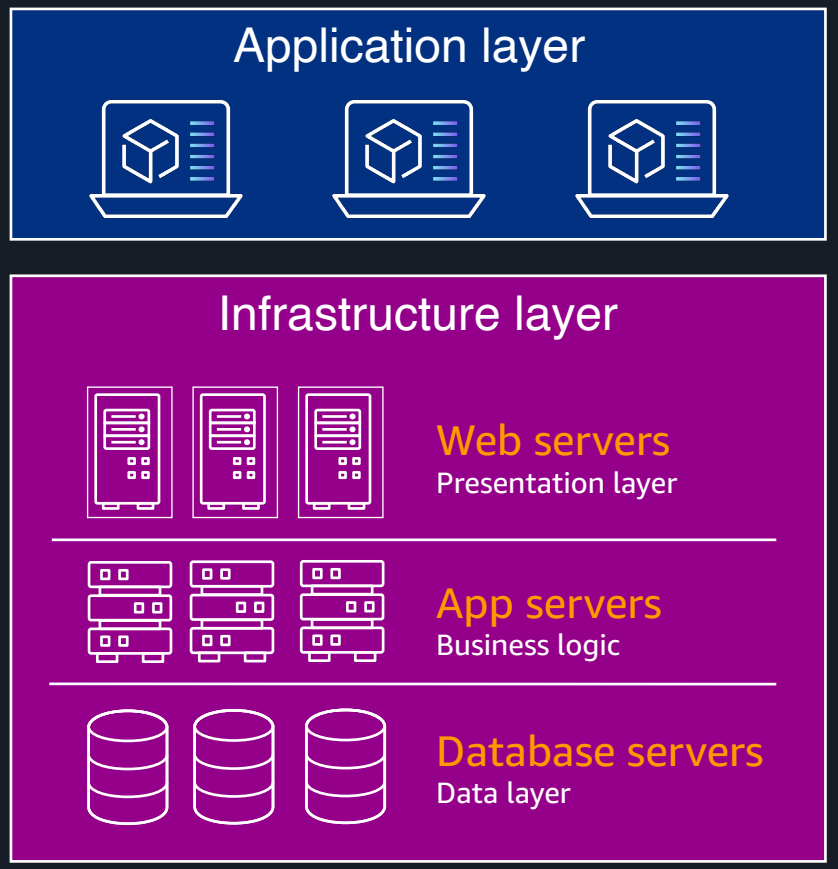


Flexible compute choice

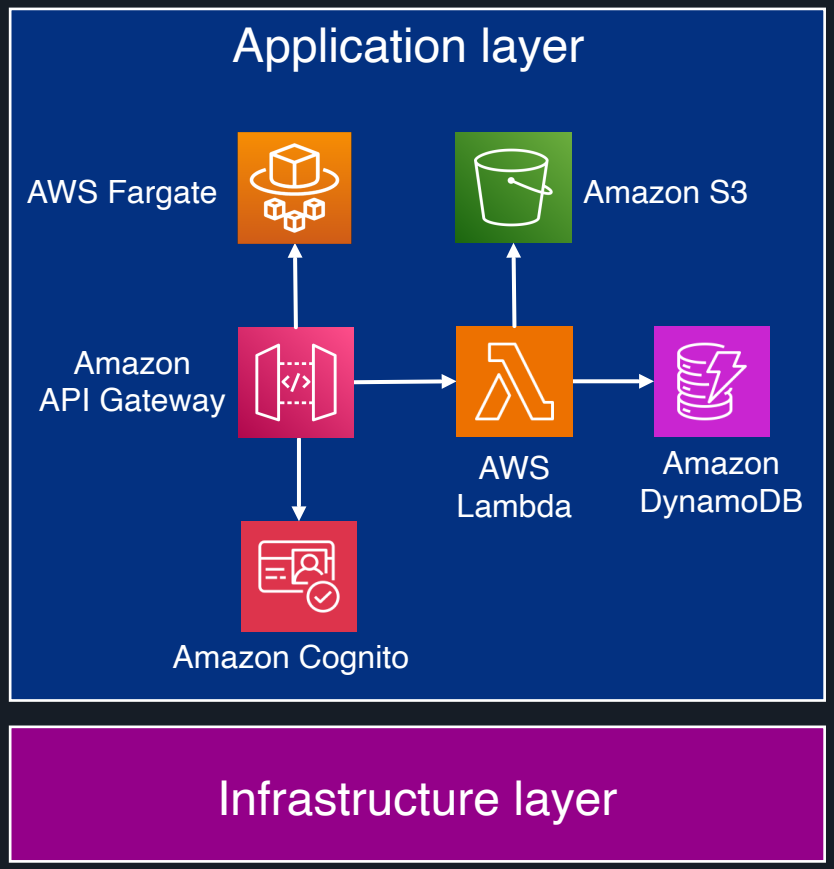
Accelerated application delivery

Application topology evolution

Traditional application

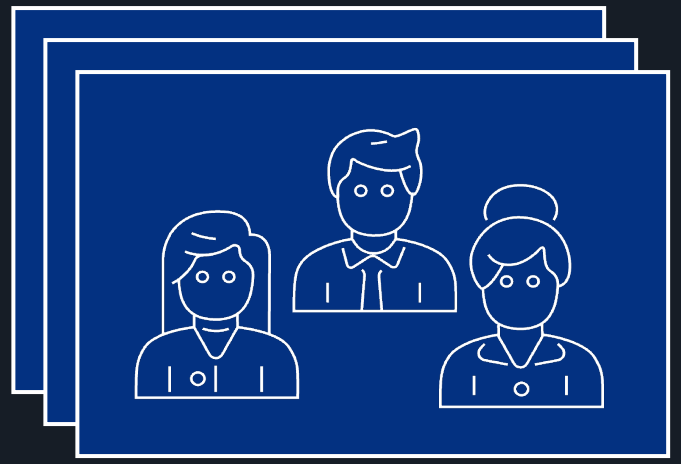


Serverless application



The ownership boundaries

This is an **application resource**! We need the flexibility to control it!

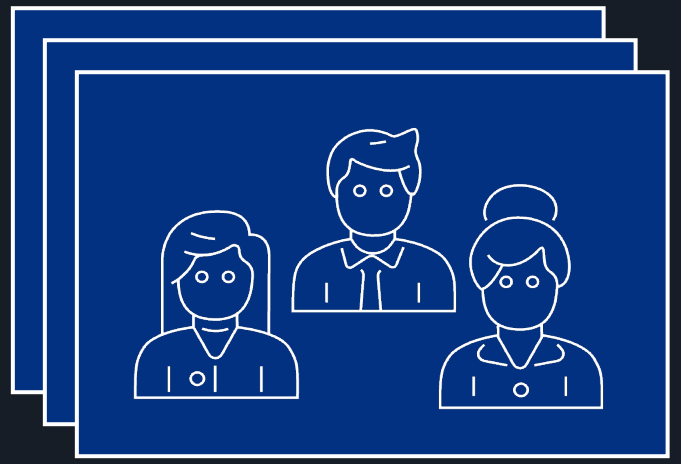


Application development teams

This is an **infrastructure resource**! We own and manage it!

Infrastructure/ops team

The ownership boundaries

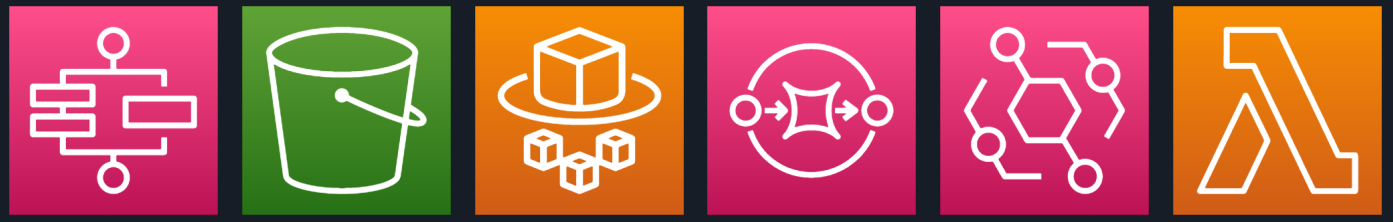


Application development teams

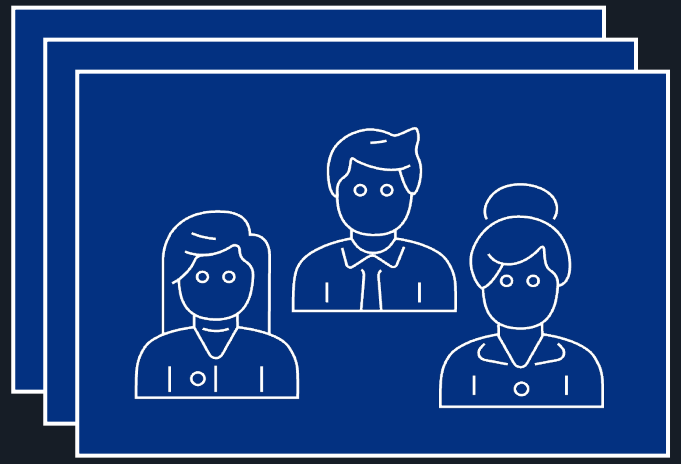
Rapid innovation! Agility! Time to market! Feedback cycle! New services!

Standardization! Security! Governance! Observability! Cost efficiency!

Infrastructure/ops team



Striking the balance



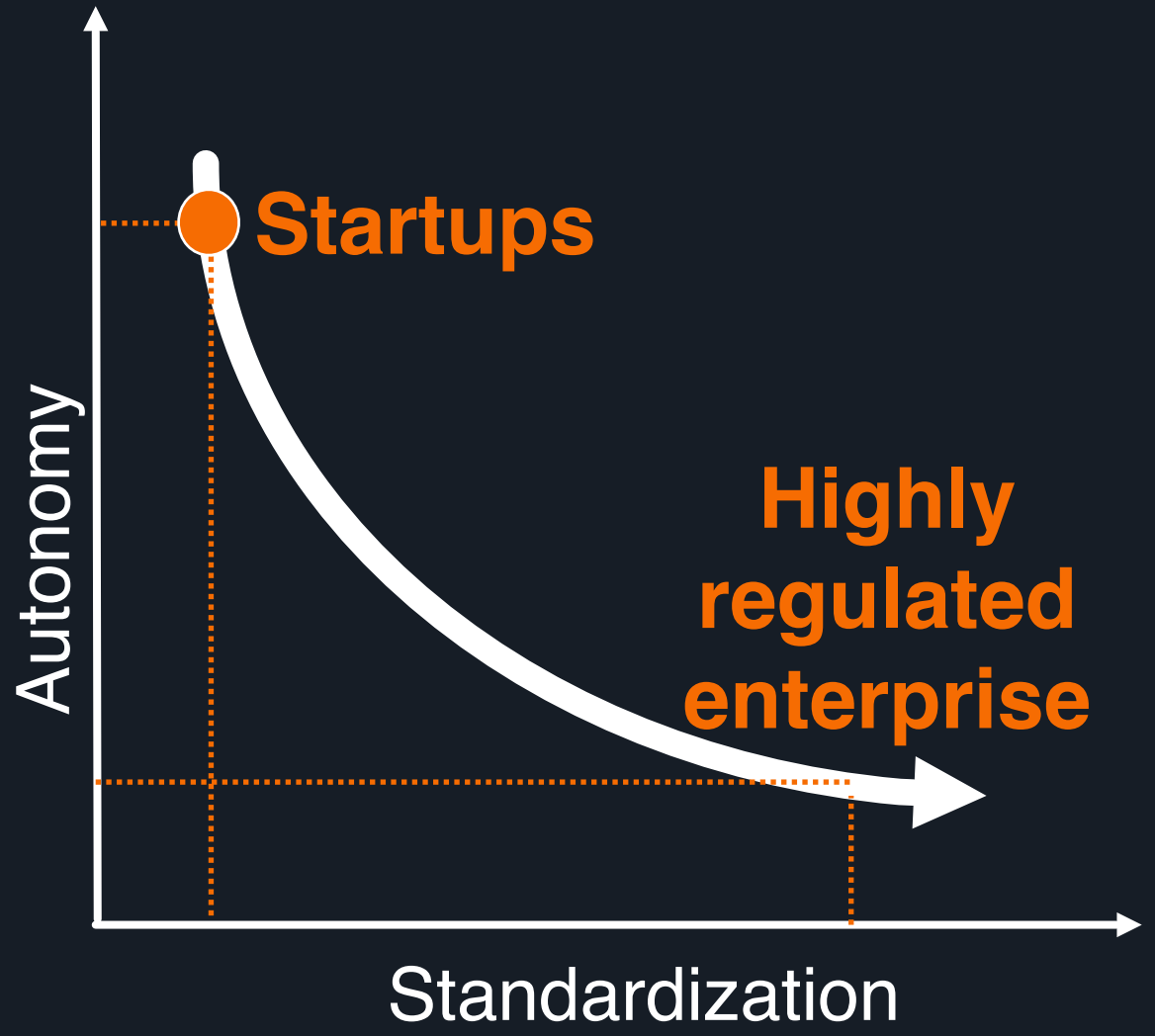
AppDev teams

Infra/ops team

Autonomy ⟷ **Standardization**

Striking the balance

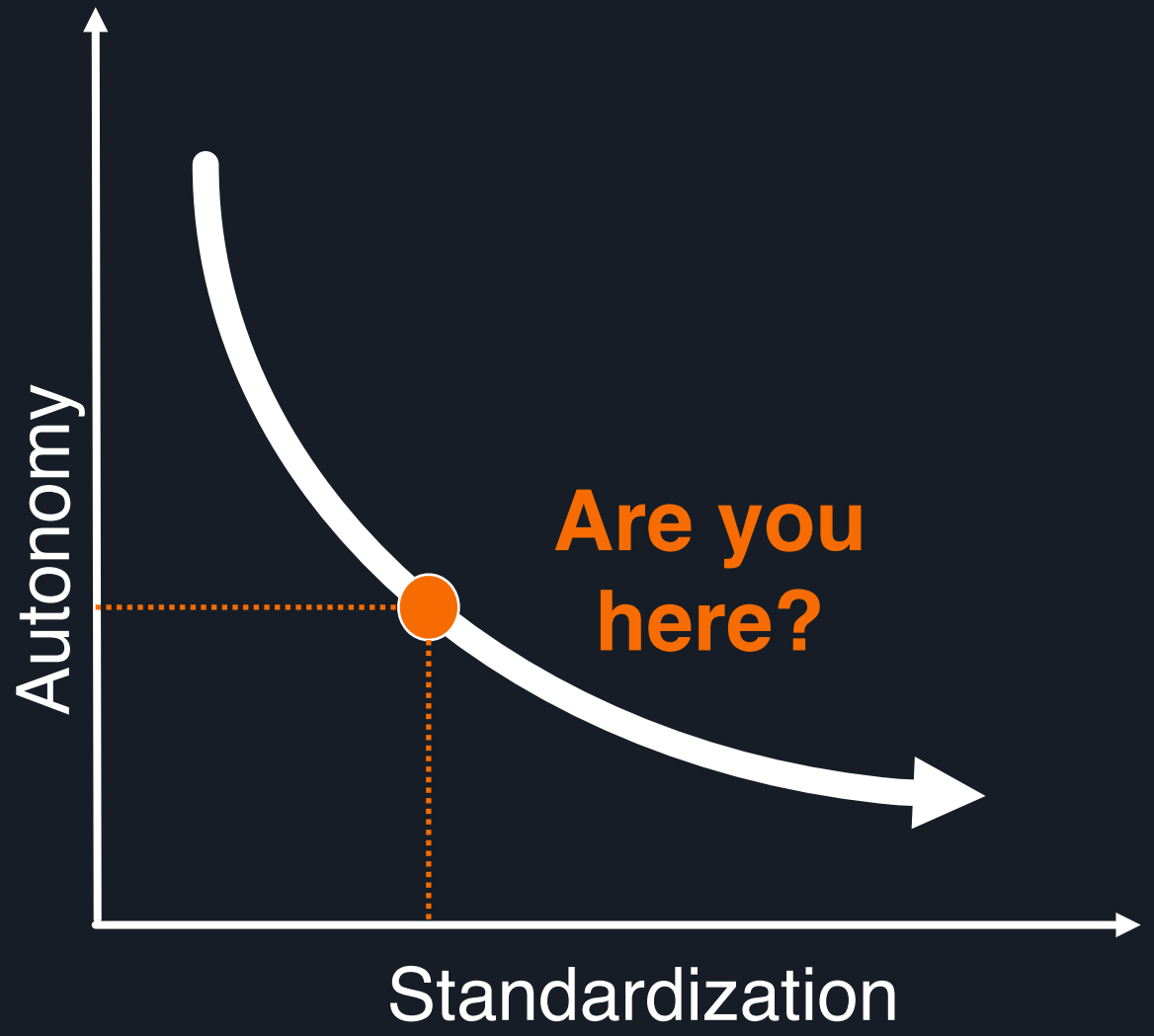
AppDev teams



Infra/ops team

Striking the balance

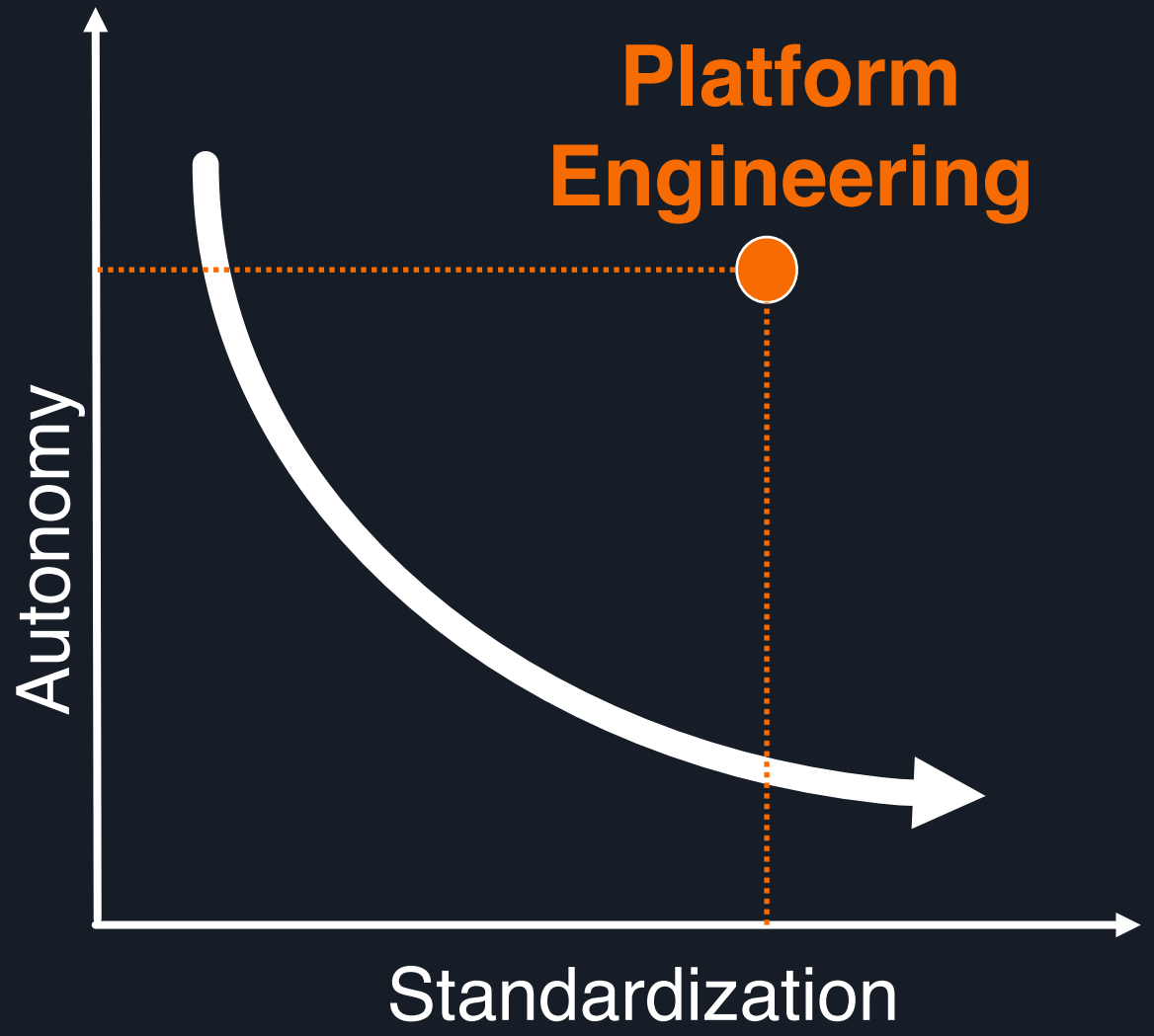
AppDev teams



Infra/ops team

Striking the balance

AppDev teams



Infra/ops team

CYBERARK®
THE IDENTITY SECURITY COMPANY®

"Implementing serverless blueprints, automation, and platform engineering practices allowed us to **reduce new service launch time from 23 weeks to 3 hours, saving over 5 months of work** for each new service we build"

- Ran Isenberg, AWS Serverless Hero, Principal Architect, CyberArk Platform Engineering Division

aws

https://aws.amazon.com/solutions/case-studies/cyberark-serverless-case-study

Can you be a little bit more specific?

Traditional ownership boundaries

Depends on

Codebase

CI/CD

Monitoring

Security

IaC

Codebase

CI/CD

Monitoring

Security

IaC

Codebase

CI/CD

Monitoring

Security

IaC

AppDev teams

Infra/ops teams

Shift-left without governance (aka wild west)

Codebase

CI/CD (GitOps)

Monitoring (CloudWatch)

Security (Top priority)

IaC (Terraform)

Codebase

CI/CD (AIOps)

Monitoring (Grafana)

Security (Maybe later)

IaC (CDK)

Codebase

CI/CD (ClickOps)

Monitoring (console.log)

Security (What?)

IaC (Bash + CLI)

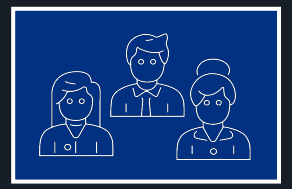
AppDev teams

?

Infra/ops teams

Shift-left empowered by Platform Engineering

Based on, empowered by



Cod
ebas
e

CI/
CD

Monitoring

Security

IaC

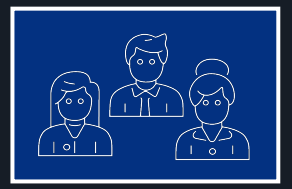
Cod
ebas
e

CI/
CD

Monitoring

Security

IaC



Cod
ebas
e

CI/
CD

Monitoring

Security

IaC

AppDev teams

App blueprints

Tools and processes

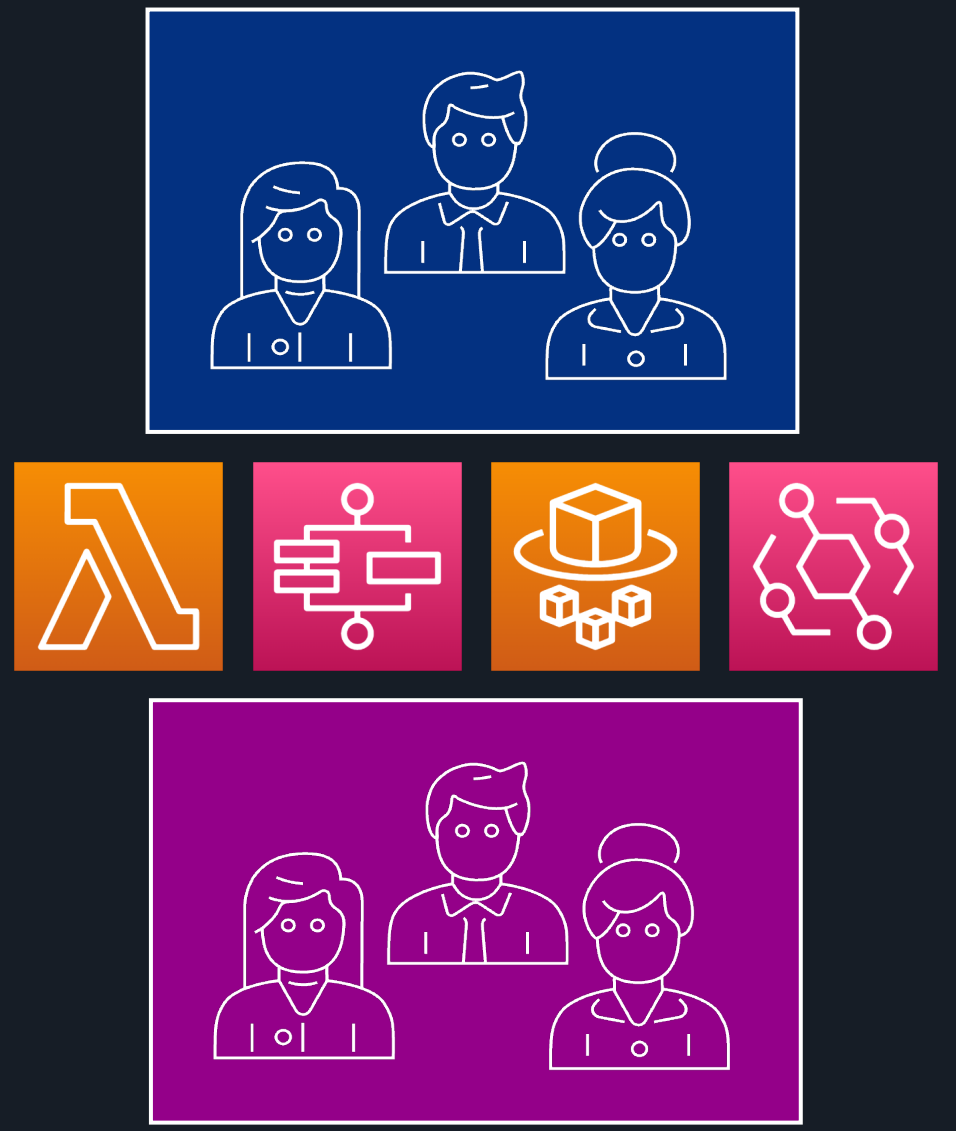
IaC templates

Security and governance policies

Platform team

Building a Serverless dev platform in practice

Define infrastructure and application resources ownership model, tools, processes



Internal Developer Platform

Partners

And many more …

Dev Portal

- Service Catalog
- CNOE
- Backstage
- GitLab

Governance

- CloudFormation Guard
- Organizations
- Control Tower
- cdk-nag
- Sentinel
- Checkov
- Open Policy Agent
- Kyverno

CI/CD

- CodeBuild
- CodeDeploy
- CodePipeline
- ArgoCD
- Jenkins
- GitLab Actions

IAC

- CloudFormation
- SAM
- CDK
- Terraform
- Pulumi
- Crossplane
- ACK

aws

- Management API
- Services

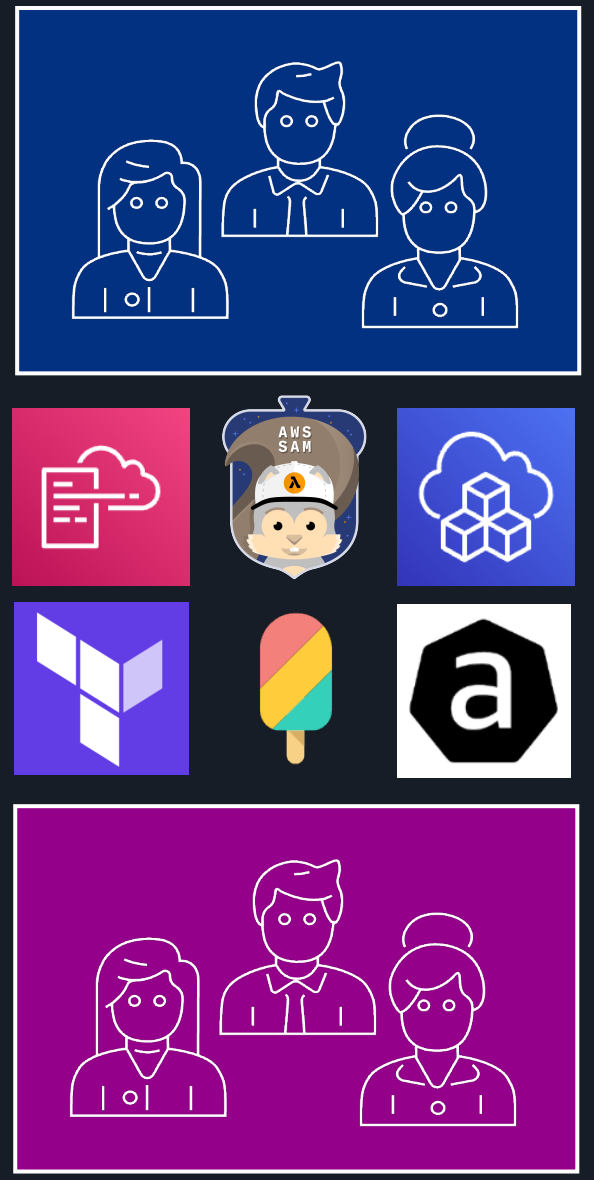
Blueprints

?

Building a Serverless dev platform in practice

Build a catalog of vetted IaC modules

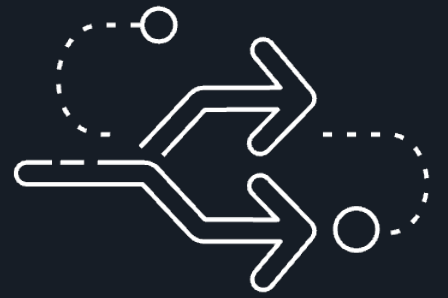
(aka constructs, aka templates)



Modularization

Best practices

Reusability



Composability

A Terraform Module

Input (variables.tf)

```
variable "function_name" {
    type = string
}

variable "runtime" {
    type = string
    default = "nodejs20.x"
}
```

Cloud resources configuration (main.tf)

```
resource "aws_lambda_function" "lambda_function" {
  function_name    = var.function_name
  runtime          = var.runtime
```

A Terraform Module

Input (variables.tf)

```
variable "function_name" {
    type = string
}

variable "runtime" {
    type = string
    default = "nodejs20.x"
}

variable "log_retention_days" {
    type = number
    default = 14
}
```

Cloud resources configuration (main.tf)

```
resource "aws_lambda_function" "lambda_function" {
  function_name    = var.function_name
  runtime          = var.runtime
  logging_config {
    log_group = aws_cloudwatch_log_group.this.arn
  }
}

resource "aws_cloudwatch_log_group" "this" {
  name              = "/aws/lambda/${var.function_name}"
  retention_in_days = var.log_retention_days
}

resource "aws_iam_policy" "this" {
    ....
}
```

A Terraform Module

Input (variables.tf)

```
variable "function_name" {
}

var
}

var
}
```

Cloud resources configuration (main.tf)

```
resource "aws_lambda_function" "lambda_function" {
  function_name = var.function_name
}
```

```
resource "aws_lambda_function" "products" {
  function_name = local.function_name
  logging_config {
    application_log_level = "INFO"
    system_log_level      = "INFO"
    log_format            = "JSON"
    log_group             = aws_cloudwatch_log_group.lambda_log_group.name
  }
  tracing_config {
    mode = "Active"
  }
}
```

A Terraform Module

```
resource "aws_lambda_function" "products" {
  function_name = local.function_name

  handler     = "index.handler"
  runtime     = "nodejs20.x"
  memory_size = 512
  role        = aws_iam_role.lambda_role.arn
  layers      = [local.powertools_layer_arn]

  environment {
    variables = {
      POWERTOOLS_SERVICE_NAME      = "${var.resource_name_prefix}-products",
      POWERTOOLS_METRICS_NAMESPACE = "${var.resource_name_prefix}-products",
      # Below properties can help with debugging
      POWERTOOLS_LOGGER_LOG_EVENT = false,  # Logs incoming event
      POWERTOOLS_LOG_LEVEL        = "INFO", # Changes log level
      POWERTOOLS_DEV              = false   # Increases JSON indentation
    }
  }
```

A Terraform Module

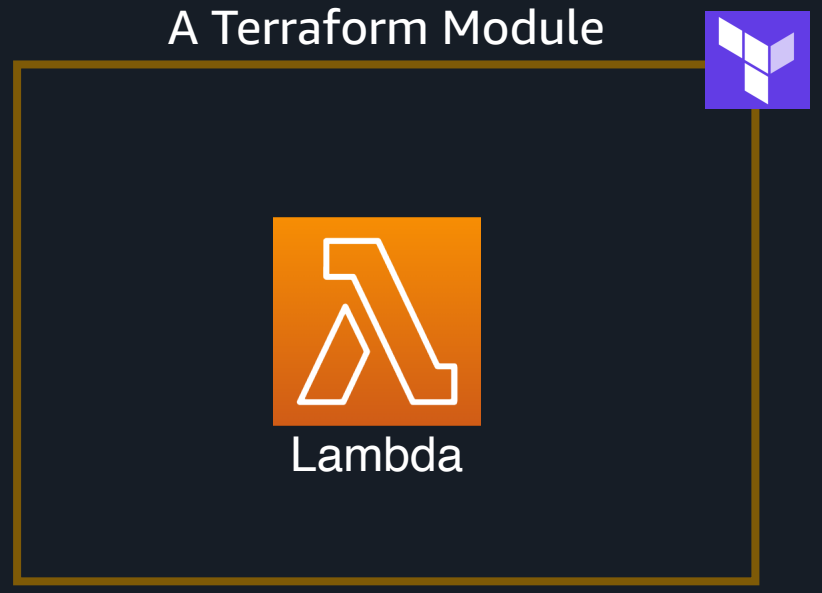
Input (variables.tf)

```
variable "function_name" {
    type = string
}

variable "runtime" {
    type = string
    default = "nodejs20.x"
}

variable "log_retention_days" {
    type = number
    default = 14
}
```

Cloud resources configuration (main.tf)

```
resource "aws_lambda_function" "lambda_function" {
  function_name    = var.function_name
  runtime          = var.runtime
  logging_config {
    log_group = aws_cloudwatch_log_group.this.arn
  }
}

resource "aws_cloudwatch_log_group" "this" {
  name              = "/aws/lambda/${var.function_name}"
  retention_in_days = var.log_retention_days
}

resource "aws_iam_policy" "this" {
    ....
}
```

A Terraform Module



Best Practices

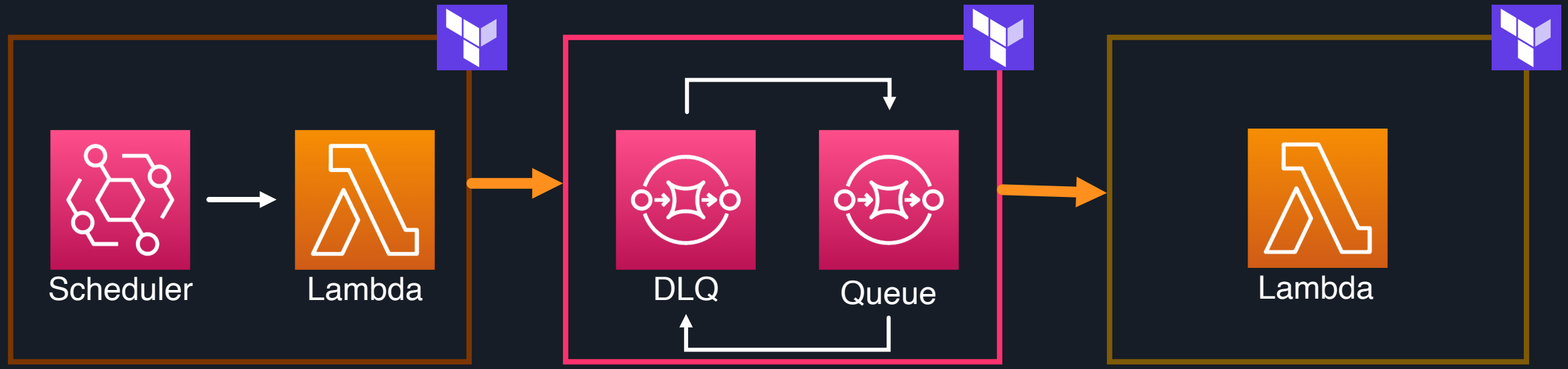
Variables

function_name
memory_size
source_path
runtime
handler
attach_vpc
log_retention_days
enable_furl
etc…

Outputs

function_arn
log_group_arn
function_url

Reusability and Composability

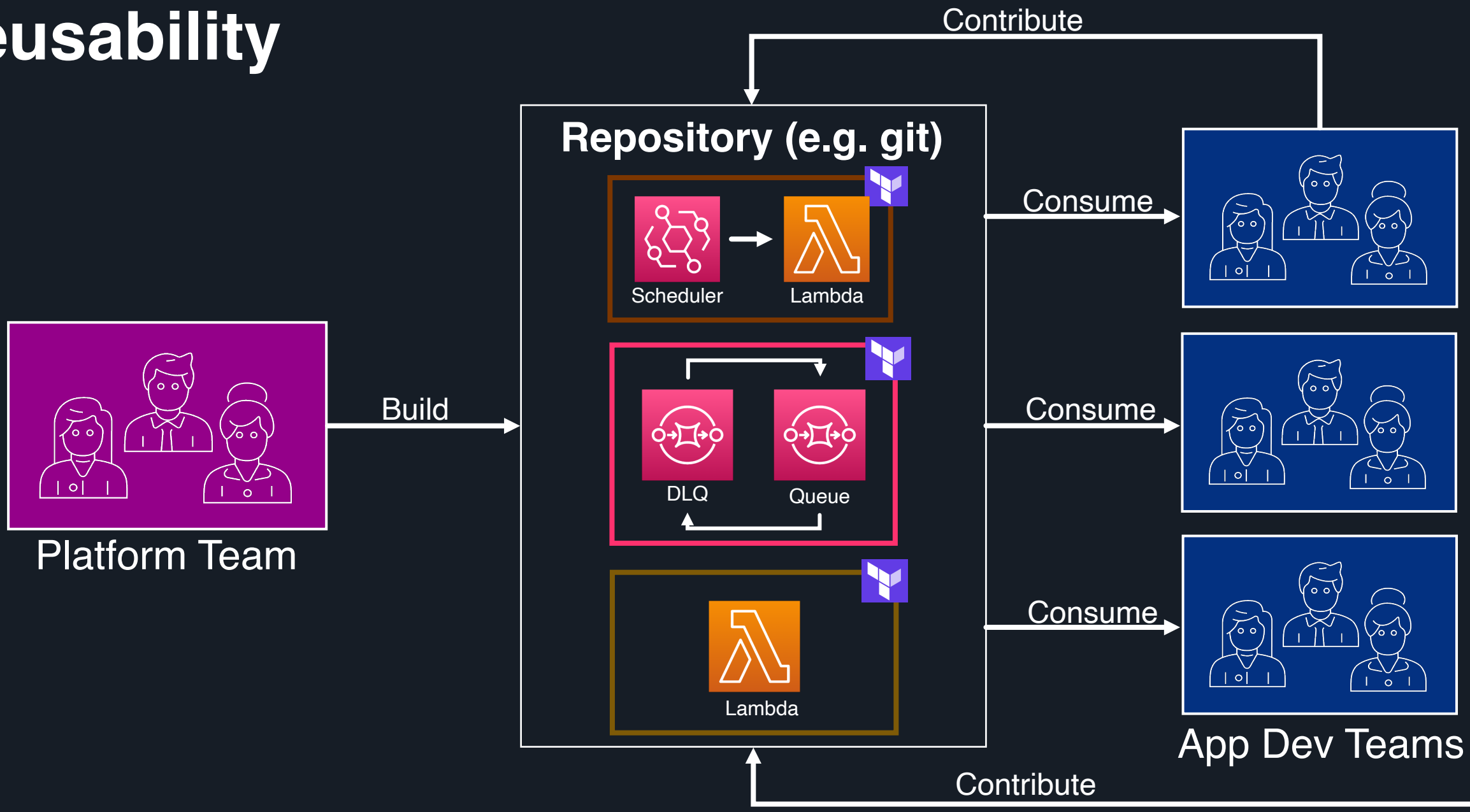


Lambda function with periodical scheduler

SQS Queue with redrive

Baseline Lambda function

Reusability



Open-source / community projects

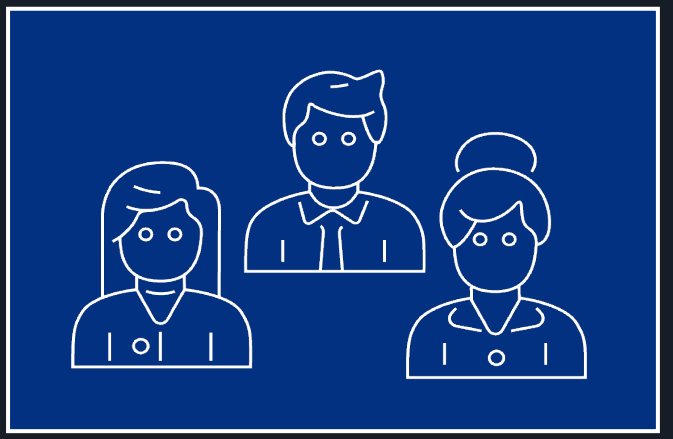
Serverless.tf

ConstructHub

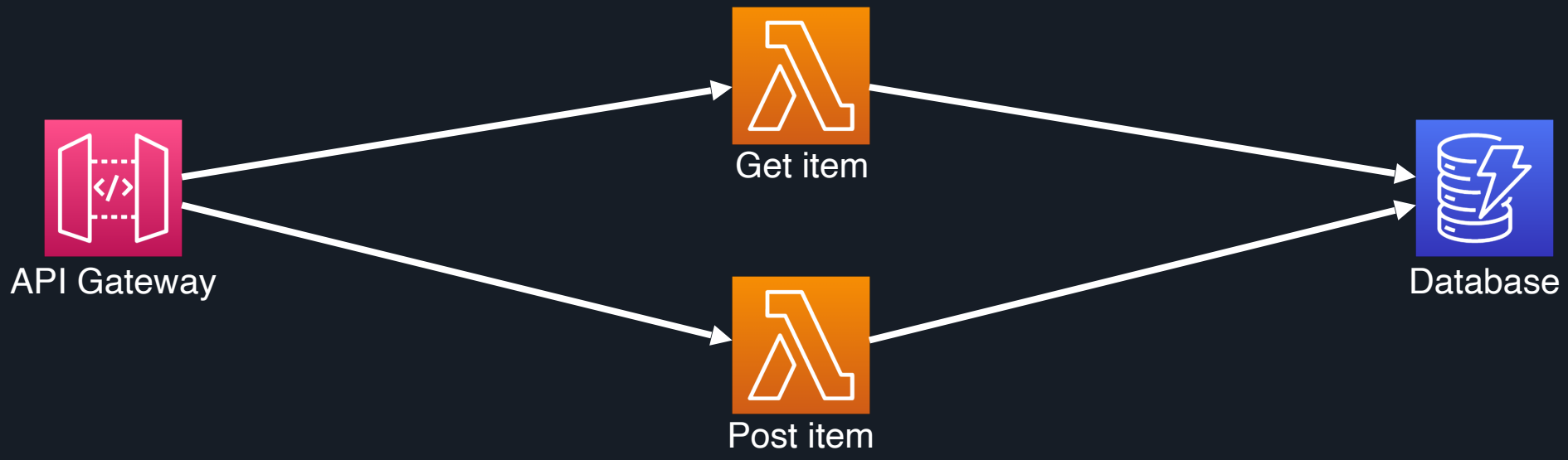
Awesome-serverless-blueprints

Building a Serverless dev platform in practice

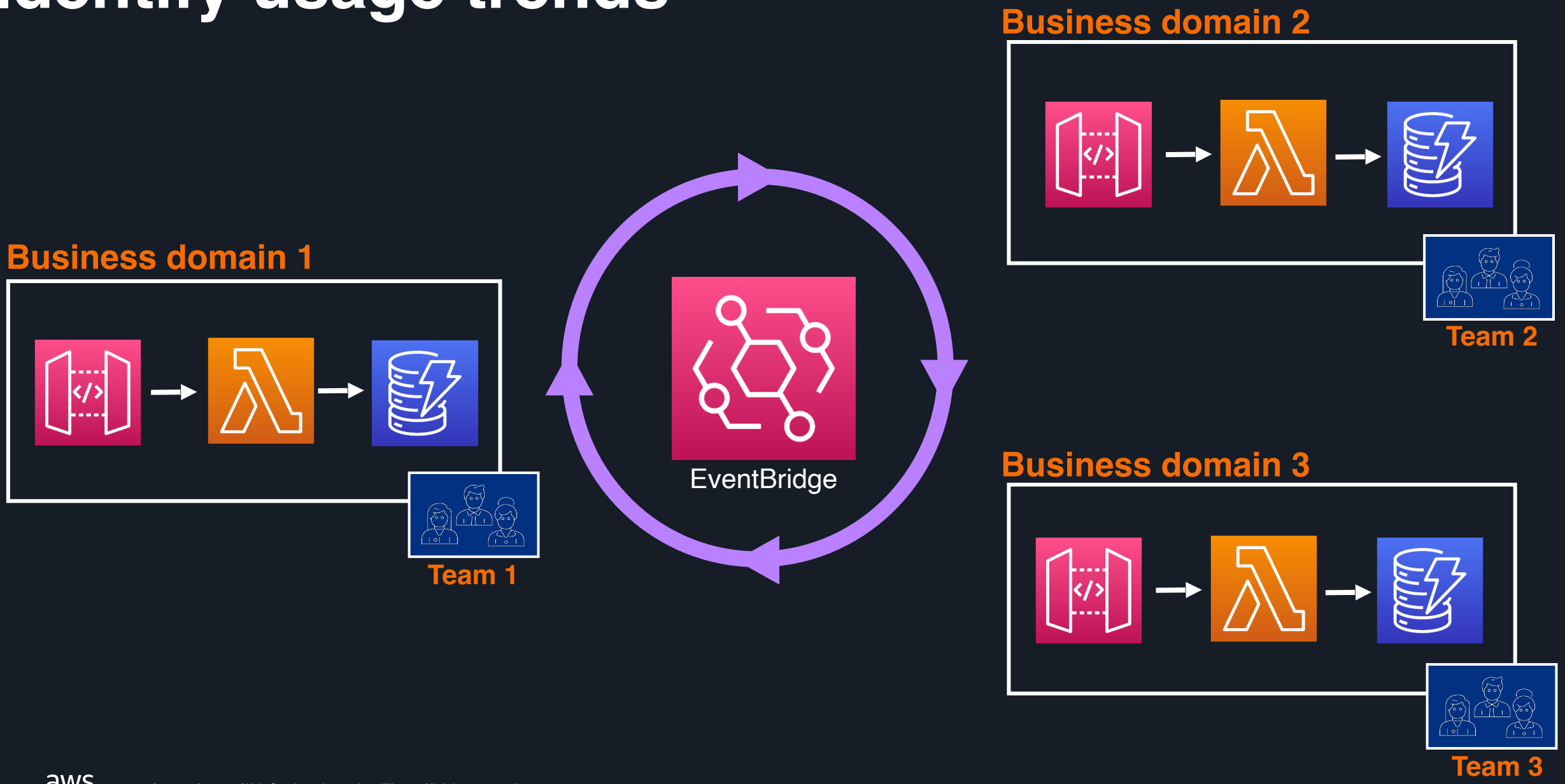
Build and evolve architectural blueprints



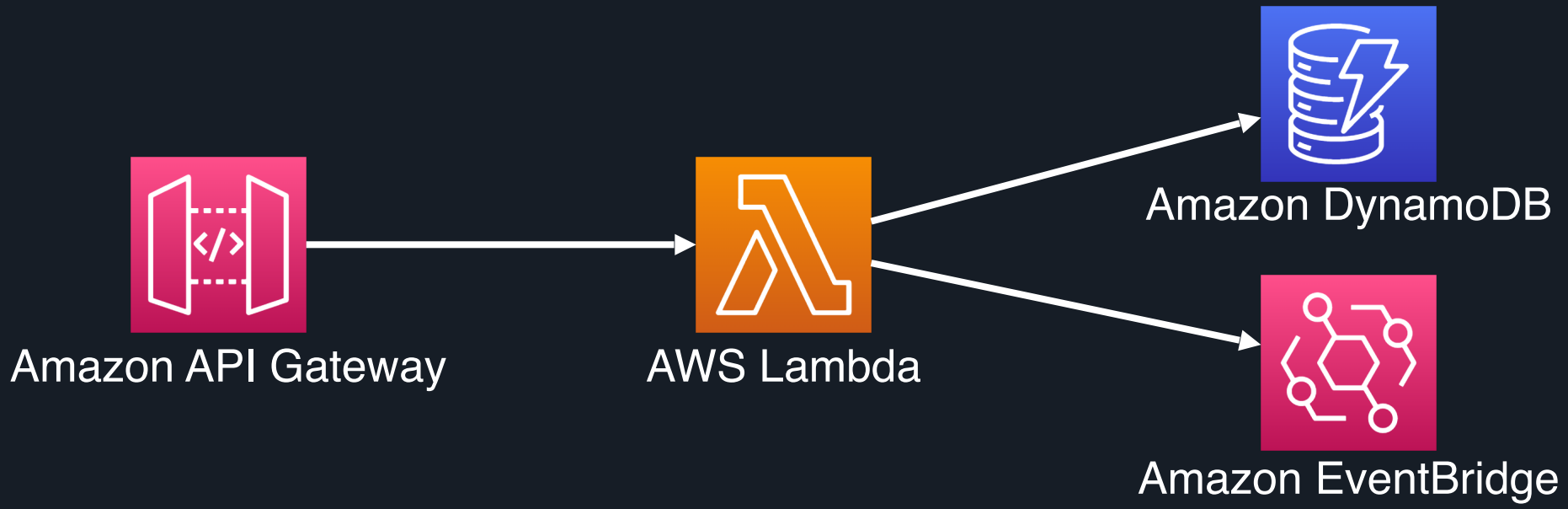
Architectural Patterns



Identify usage trends



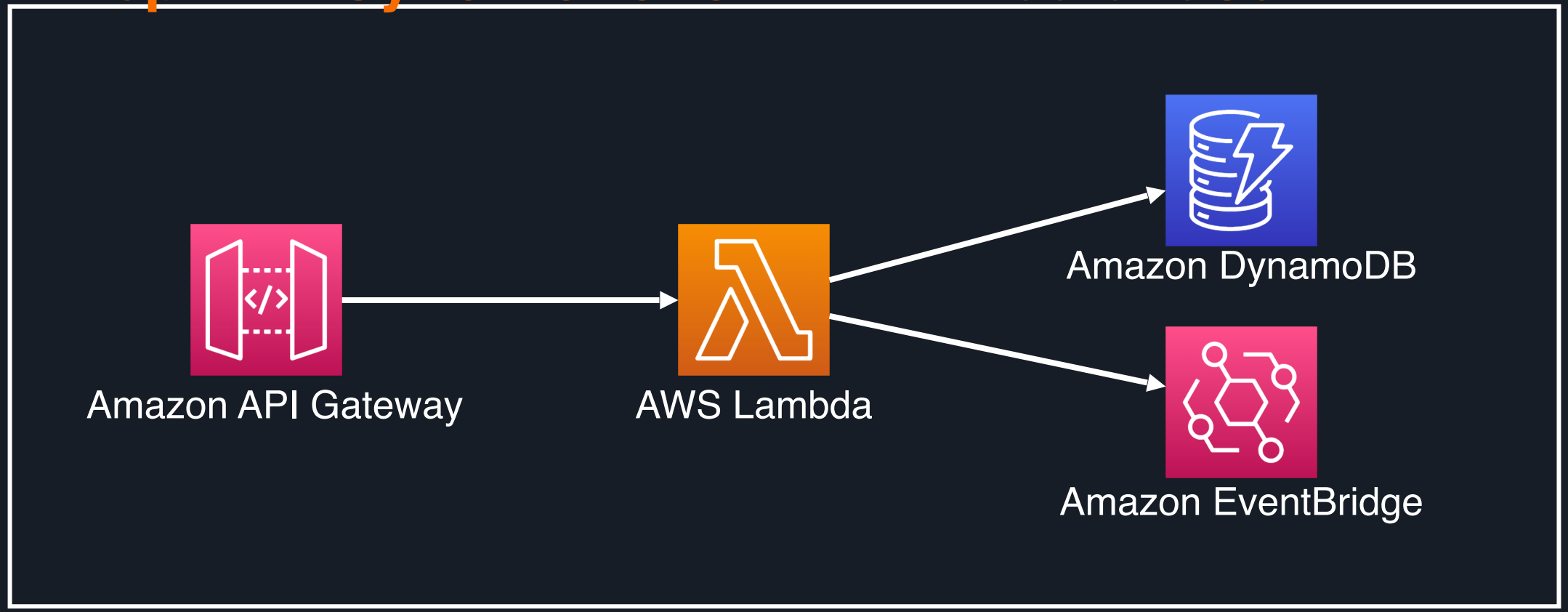
Bring it all together



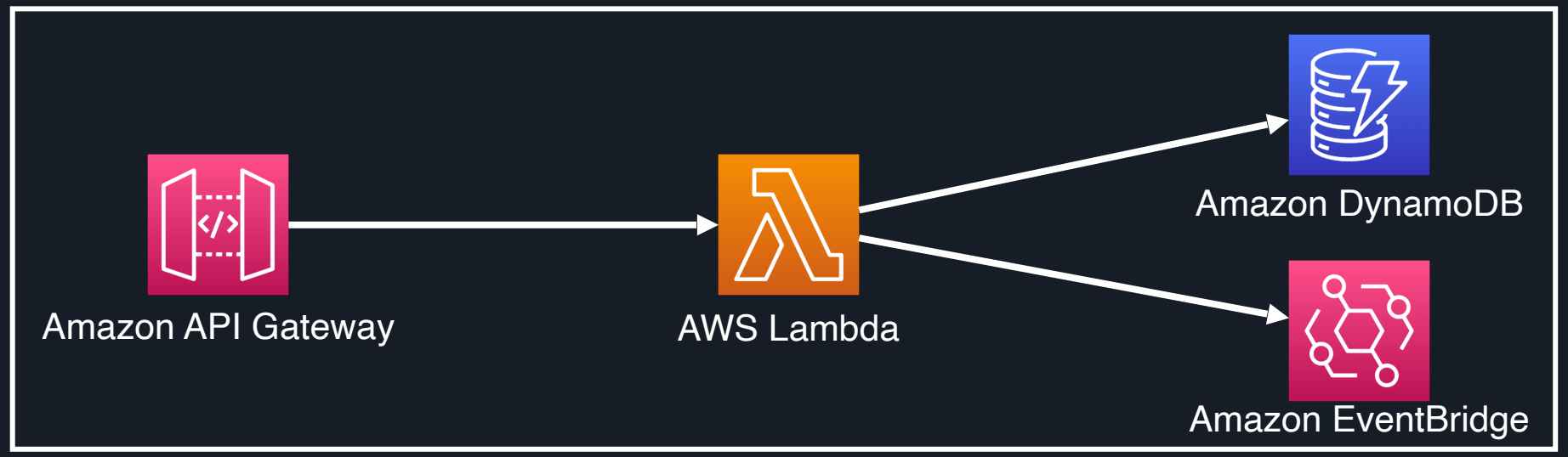
Arrows are as important as blocks

Serverless Blueprint

Blueprint - Synchronous API with database



Serverless Blueprint



Observability, security, best practices, governance, CI/CD, FinOps

- API consistency
- Authorization
- TLS/mTLS
- Rate limits
- Throttling
- Documentation
- WAF

- Resource allocation
- Access policies
- VPC-attachment
- Code-signing
- Versioning
- Canary deployments
- Tenancy management

- Billing mode
- Deletion protection
- Server-side encryption
- Event rules and targets
- Dead-letter queue
- Event archiving
- Secret management

Serverless Blueprint – FinOps with Tags

```
provider "aws" {
  region = "us-east-1"

  default_tags {
    tags = {
      Environment = var.tag_environment
      Department  = var.tag_department
      Service     = var.tag_service
    }
  }
}
```

✅Terraform ✅CDK ✅CloudFormation

Serverless Blueprint – FinOps with Tags

terraform > environments > development.tfvars

```
1   tag_environment = "development"
2   tag_department = "sales"
3   tag_service = "orders-processor"
```

terraform > environments > production.tfvars

```
1   tag_environment = "production"
2   tag_department = "sales"
3   tag_service = "orders-processor"
```

```
provider "aws" {
  region = "us-east-1"

  default_tags {
    tags = {
      Environment = var.tag_environment
      Department  = var.tag_department
      Service     = var.tag_service
    }
  }
}
```

✅Terraform ✅CDK ✅CloudFormation

Serverless Blueprint - DynamoDB

```
resource "aws_dynamodb_table" "data_table" {
  name           = "${var.resource_name_prefix}-data-table"
  billing_mode   = "PAY_PER_REQUEST"

  point_in_time_recovery {
    enabled = true
  }

  server_side_encryption {
    enabled = true
  }

  ttl {
    attribute_name = "TimeToLive"
    enabled        = true
  }
}
```

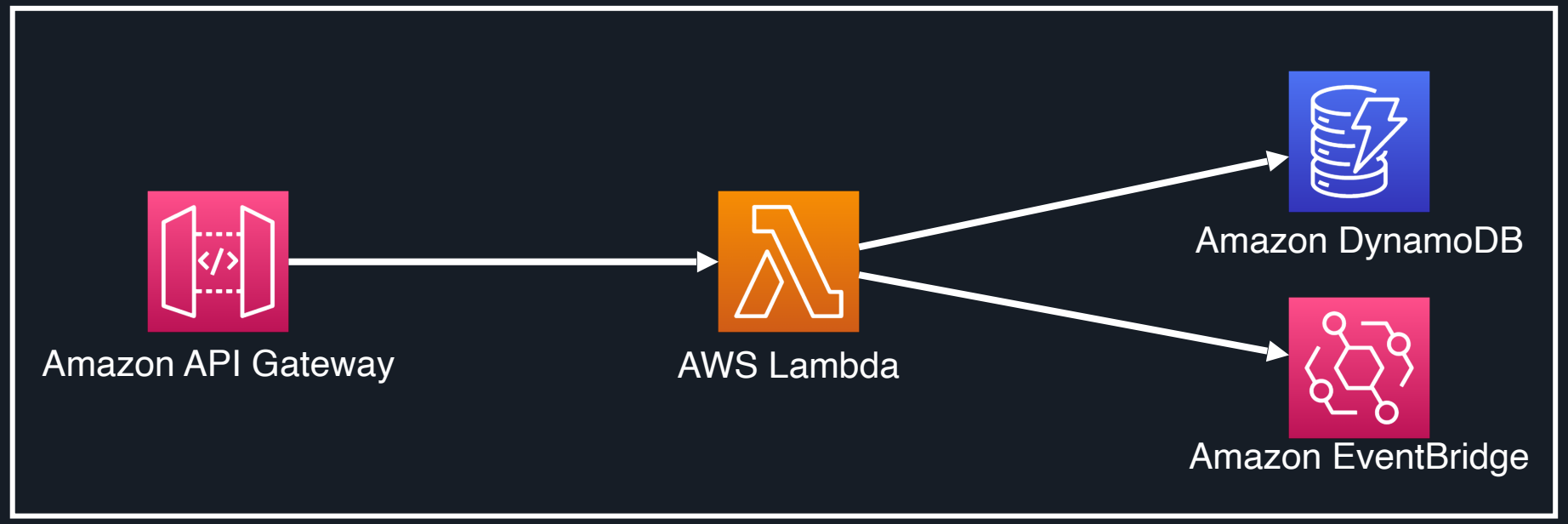
✅Terraform ✅CDK ✅CloudFormation

Serverless Blueprint – Lambda + DynamoDB

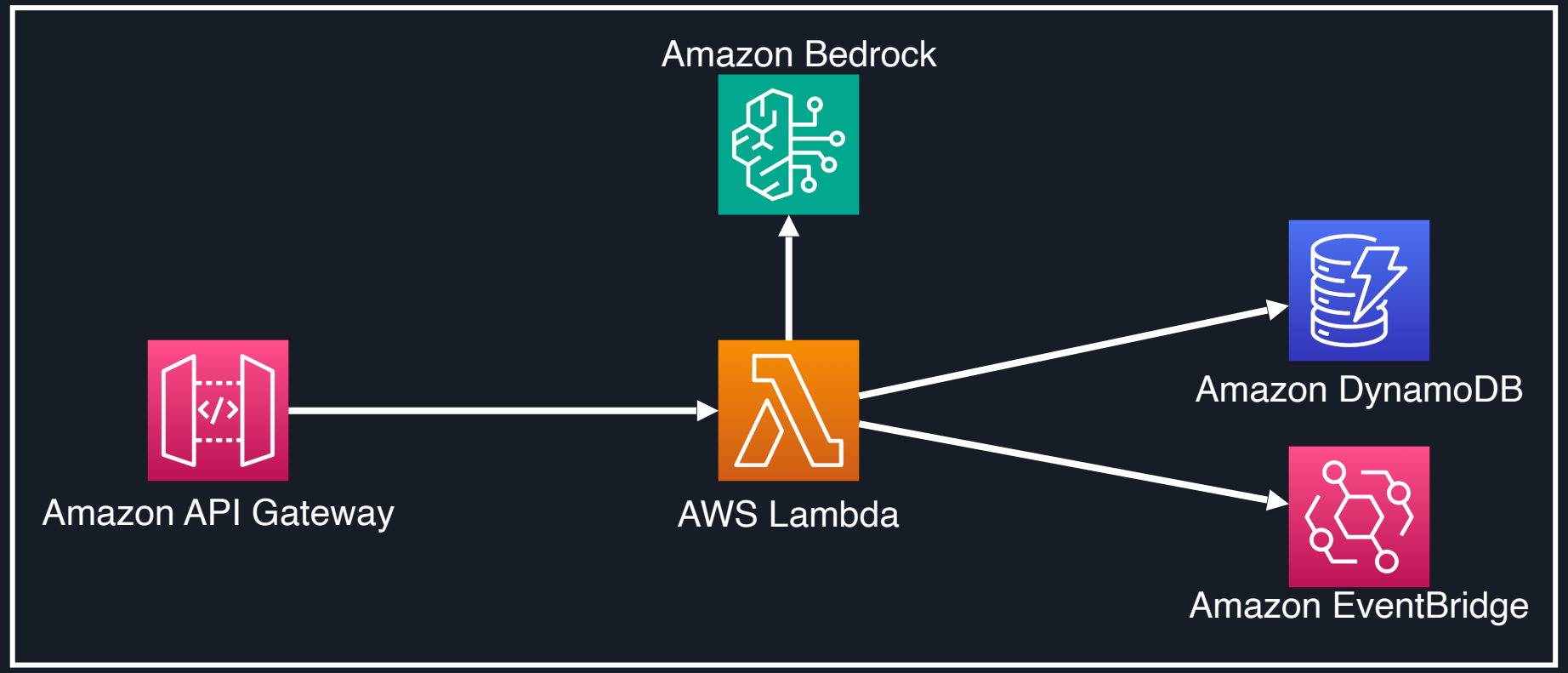
```
resource "aws_iam_policy" "lambda_ddb_data_table_access_policy" {
  name        = "${var.resource_name_prefix}-lambda-ddb-data-table-access-policy"
  description = "Grants GetItem and PutItem access to DynamoDB data table"

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Action = ["dynamodb:PutItem", "dynamodb:GetItem", "dynamodb:UpdateItem"],
        Effect   = "Allow",
        Resource = [var.ddb_data_table_arn]
      }
    ]
  })
}
```

✅Terraform ✅CDK ✅CloudFormation

Serverless Blueprint



Serverless Blueprint for genAI applications



Serverless Blueprint for genAI applications

```
resource "aws_iam_policy" "bedrock_invoke_only_specific_model" {
  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid      = "AllowInvokeSpecificModel"
        Effect   = "Allow"
        Action   = [
          "bedrock:InvokeModel",
          "bedrock:InvokeModelWithResponseStream"
        ]
        Resource = "arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-sonnet-4-20250514-v1:0"
      },
      {
        Sid      = "DenyInvokeAllOtherModels"
        # redacted
      }
    ]
  })
}
```

Serverless Blueprint for genAI applications

IaC

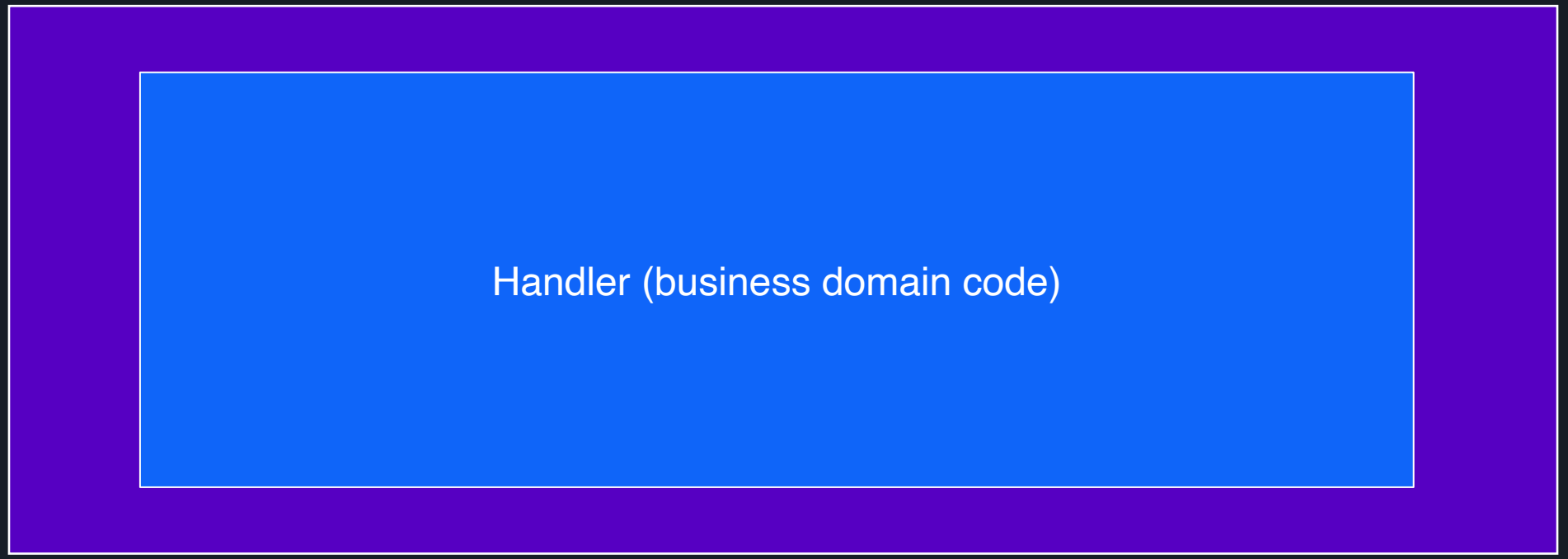
```
resource "aws_lambda_function" "bedrock_lambda" {
  # redacted
  environment {
    variables = {
      MODEL_ID    = "anthropic.claude-opus-4-20250514-v1:0"
      MAX_TOKENS  = "512"
      TEMPERATURE = "0.7"
      TOP_P       = "0.9"
    }
  }
}
```

Function handler code

```
def handler(event, context):

    model_id     = os.getenv("MODEL_ID")
    max_tokens   = os.getenv("MAX_TOKENS")
    temperature  = os.getenv("TEMPERATURE")
    top_p        = os.getenv("TOP_P")

    resp = bedrock.converse(
        modelId=model_id,
        messages=[...],
        inferenceConfig={
            "maxTokens": max_tokens,
            "temperature": temperature,
            "topP": top_p
        }
    )

    # redacted
```

Function Blueprints

AWS Lambda

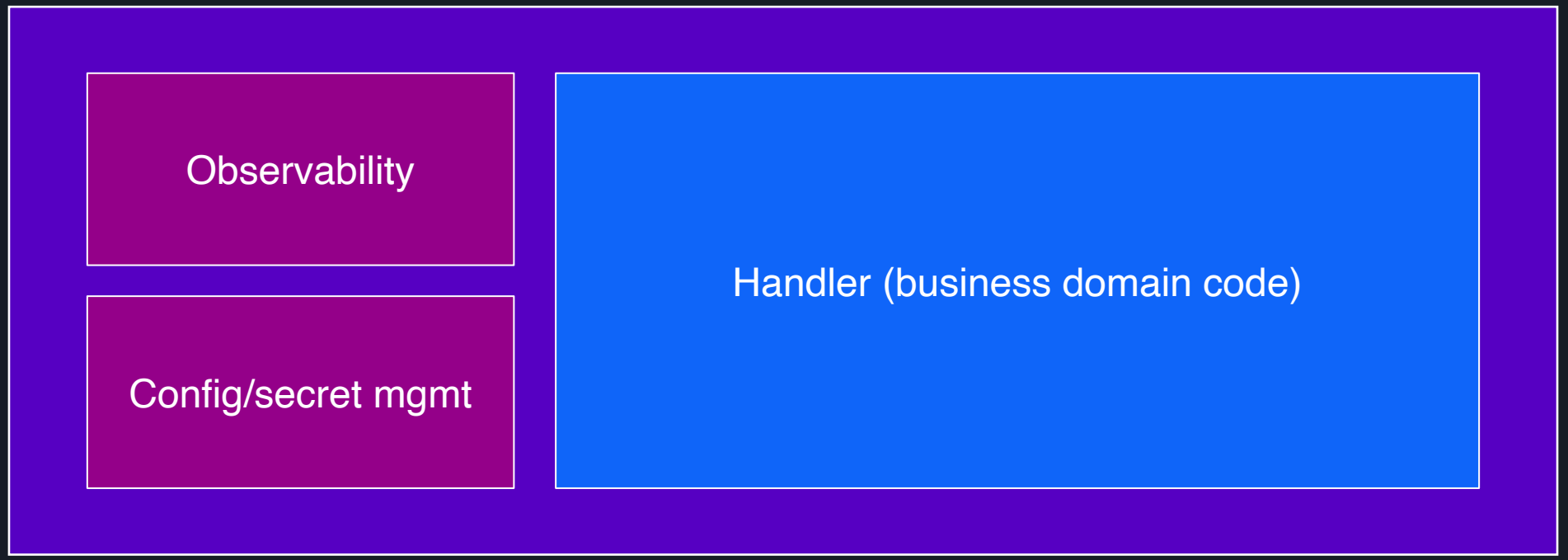


Function Blueprints

AWS Lambda

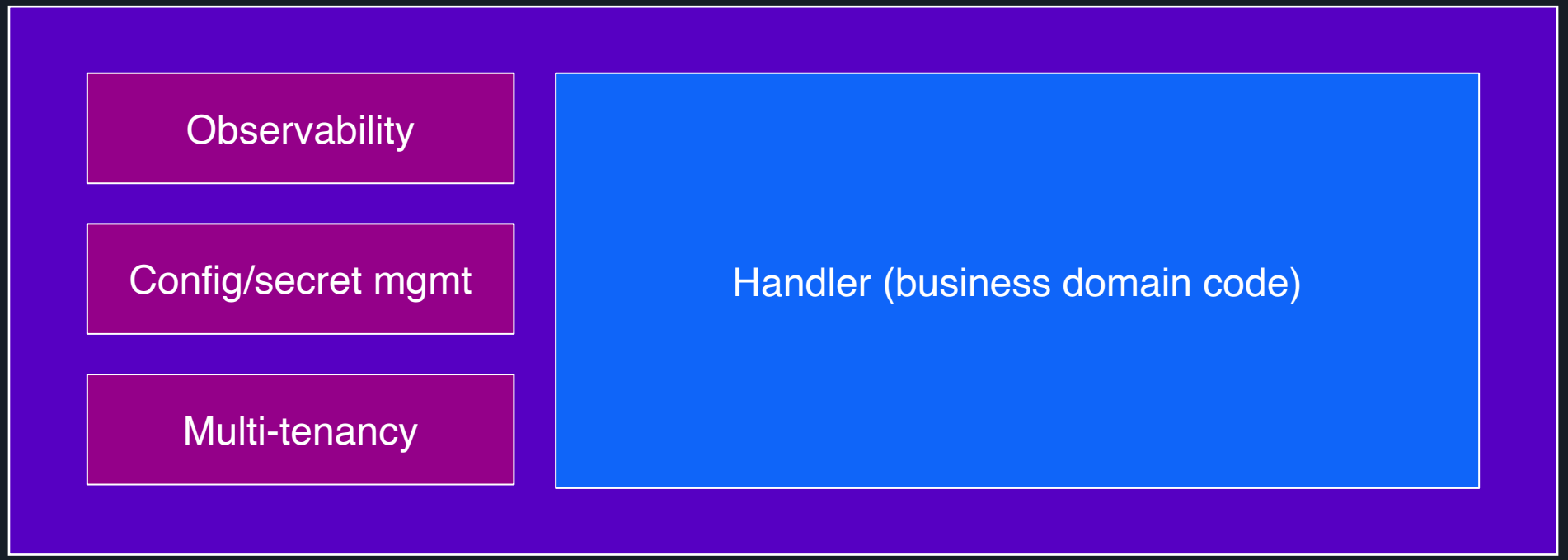
Function Blueprints

AWS Lambda



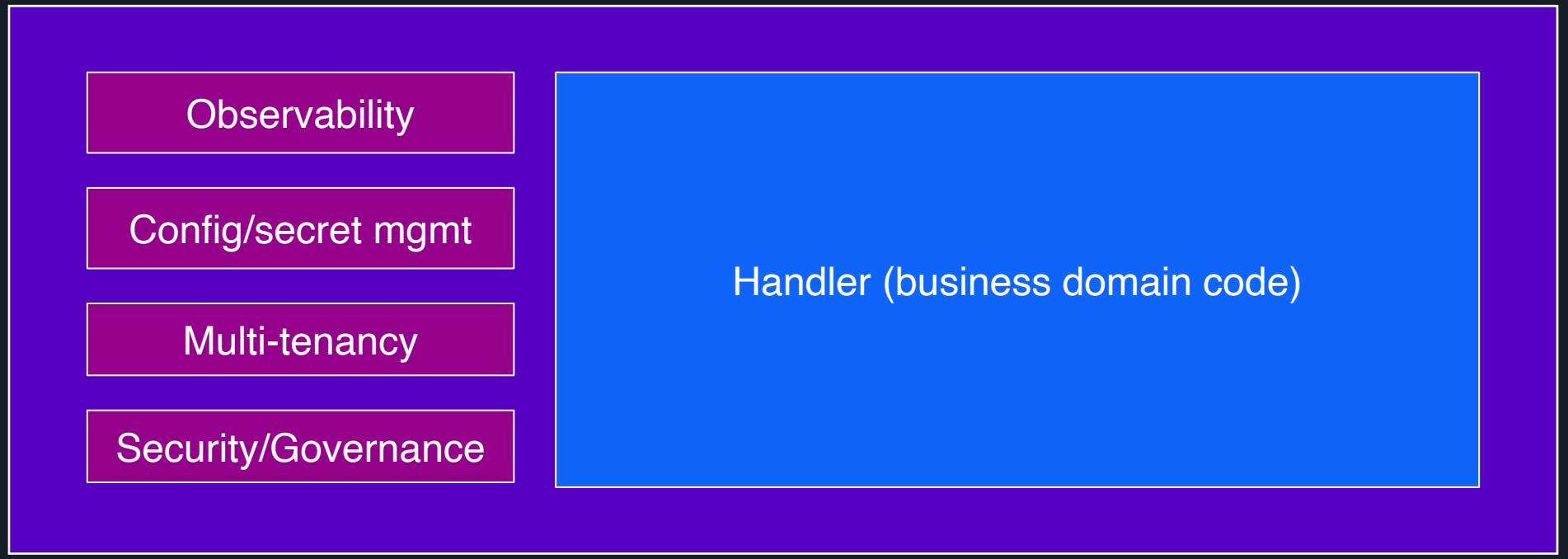
Function Blueprints

AWS Lambda



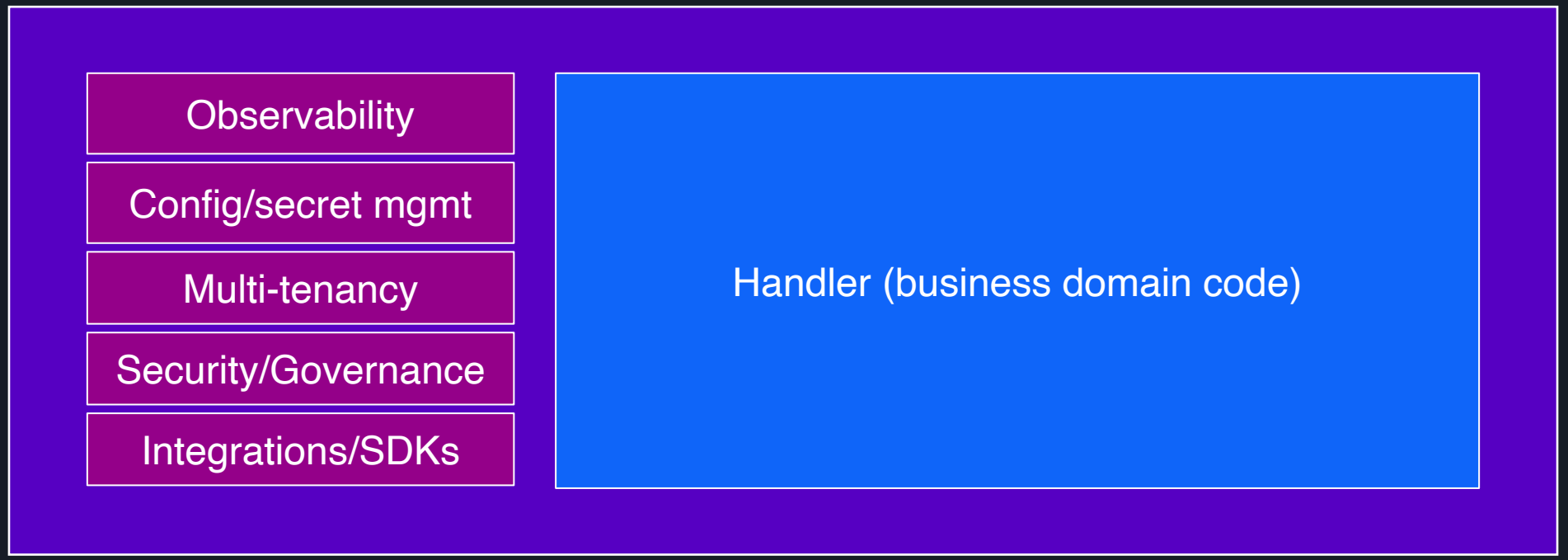
Function Blueprints

AWS Lambda



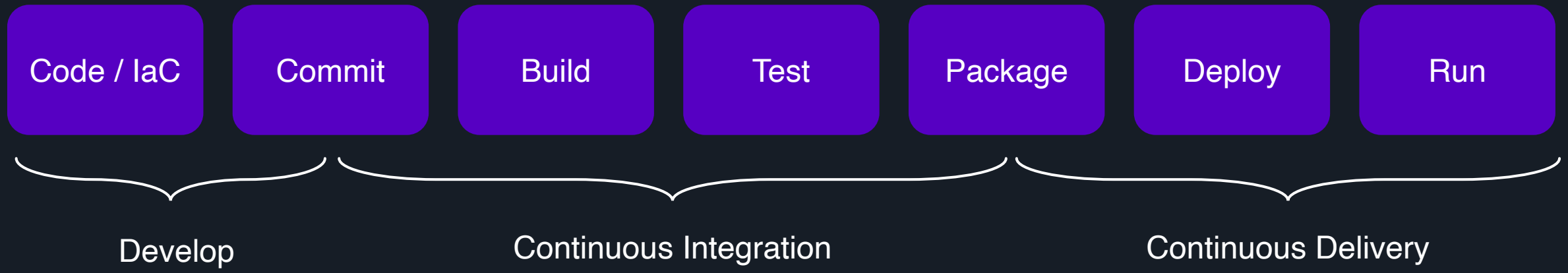
Function Blueprints

AWS Lambda



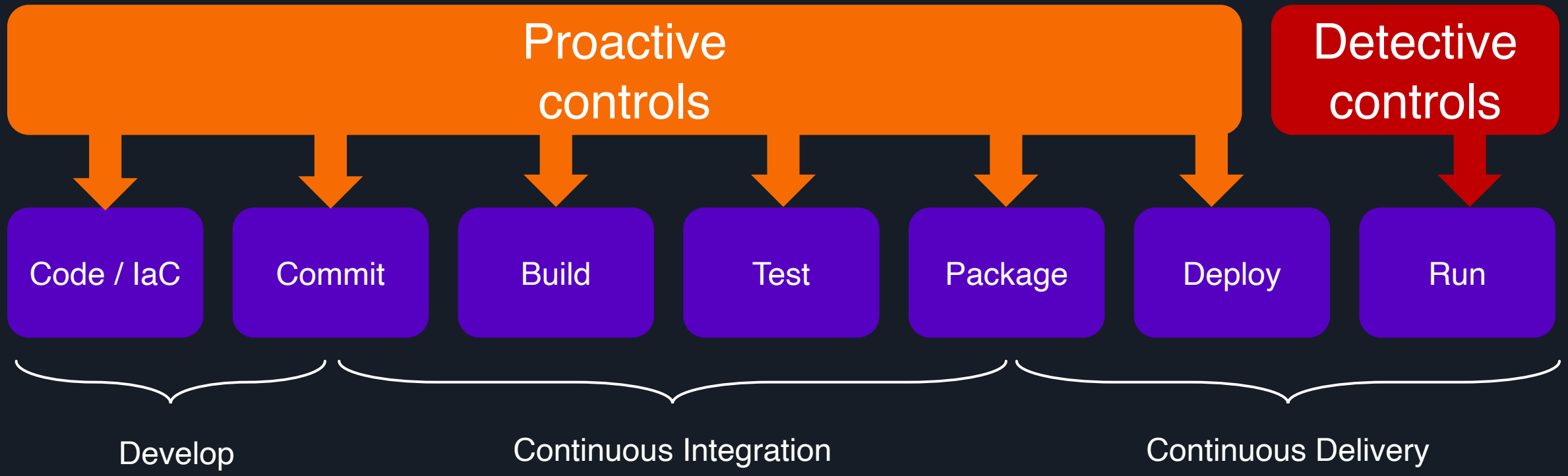
But what if developers want to make changes to the blueprint / pattern!?

End-to-end governance with CI/CD



https://serverlessland.com/content/service/lambda/guides/governance/1-introduction

End-to-end governance with CI/CD



https://serverlessland.com/content/service/lambda/guides/governance/1-introduction

End-to-end governance with CI/CD


AWS CloudFormation Guard
(template validation)


AWS Signer
(code signing)

checkov
BY PRISMA® CLOUD


AWS Config
(proactive/detective)


Amazon Inspector
(code scanning)


AWS CDK
(cdk-nag)


AWS Organizations
(proactive/detective)


AWS Control Tower
(multi-account environments)


HashiCorp Sentinel
Policies

Proactive controls

Detective controls

Code / IaC

Commit

Build

Test

Package

Deploy

Run

Develop

Continuous Integration

Continuous Delivery

https://serverlessland.com/content/service/lambda/guides/governance/1-introduction

Proactive Governance Controls

Checkov

AWS CloudFormation Guard

CDK-NAG

HashiCorp Sentinel

✅Terraform ✅CDK ✅CloudFormation

aws

Proactive Governance Controls


```
Passed checks: 41, Failed checks: 0, Skipped checks: 5

Check: CKV_AWS_274: "Disallow IAM roles, users, and groups from using the AWS AdministratorAccess policy"
        PASSED for resource: aws_iam_role.lambda_role
        File: /main.tf:7-23
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-iam-policies/bc-aws-274
Check: CKV_AWS_61: "Ensure AWS IAM policy does not allow assume role permission across all services"
        PASSED for resource: aws_iam_role.lambda_role
        File: /main.tf:7-23
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-iam-policies/bc-aws-iam-45
Check: CKV_AWS_60: "Ensure IAM role allows only specific services or principals to assume it"
        PASSED for resource: aws_iam_role.lambda_role
        File: /main.tf:7-23
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-iam-policies/bc-aws-iam-44
Check: CKV_AWS_66: "Ensure that CloudWatch Log Group specifies retention days"
        PASSED for resource: aws_cloudwatch_log_group.lambda_log_group
        File: /main.tf:31-35
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-logging-policies/logging-13
Check: CKV_AWS_338: "Ensure CloudWatch log groups retains logs for at least 1 year"
        PASSED for resource: aws_cloudwatch_log_group.lambda_log_group
        File: /main.tf:31-35
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-logging-policies/bc-aws-338
```

✅Terraform ✅CDK ✅CloudFormation

Proactive Governance Controls

checkov
BY PRISMA® CLOUD

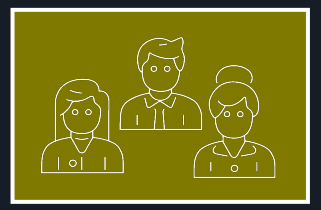
```
Check: CKV_AWS_66: "Ensure that CloudWatch Log Group specifies retention days"
        PASSED for resource: aws_cloudwatch_log_group.lambda_log_group
        File: /main.tf:31-35
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-logging-policies/logging-13
Check: CKV_AWS_338: "Ensure CloudWatch log groups retains logs for at least 1 year"
        PASSED for resource: aws_cloudwatch_log_group.lambda_log_group
        File: /main.tf:31-35
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-logging-policies/bc-aws-338
```

```
Check: CKV_AWS_66: "Ensure that CloudWatch Log Group specifies retention days"
        FAILED for resource: aws_cloudwatch_log_group.lambda_log_group
        File: /main.tf:31-35
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-logging-policies/logging-13
Check: CKV_AWS_338: "Ensure CloudWatch log groups retains logs for at least 1 year"
        FAILED for resource: aws_cloudwatch_log_group.lambda_log_group
        File: /main.tf:31-35
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-logging-policies/bc-aws-338
```

✅Terraform ✅CDK ✅CloudFormation

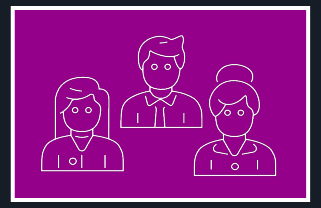
aws

Blueprint Lifecycle



Security team

Feedback



Platform team

Build

Maintain

Curated Blueprint Catalog

(e.g. Backstage, Stackspot, CNOE, Service Catalog)

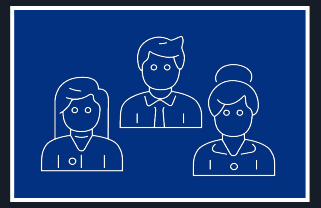
Sync REST API with database

Async WebSockets API

Distributed S3 data processing

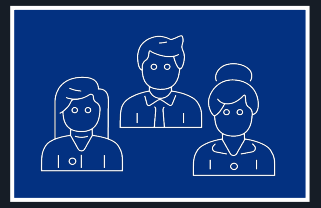
Other common pattern…

Contribute



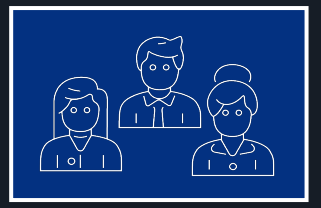
Application development team

Consume



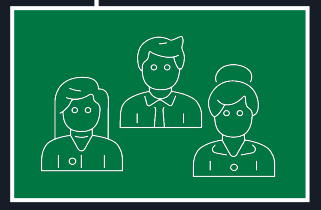
Application development team

Contribute



Application development team

Compliance team



Feedback

CYBERARK®

THE IDENTITY SECURITY COMPANY®

CyberArk identity security platform enables secure access for any identity — human or machine — to any resource or environment from anywhere, using any device.

Founded 1999

Identity and Access Management Leader

4,000+ employees
1,000+ developers

Cloud-native, serverless-first SaaS company

The SaaS Provider Evolution

Fragmented:
- Developer experience
- Tech stack
- Architecture
- Observability
- User Experience

🚀 Platform Engineering team established with 15 engineers

- Streamlined **tech stack**, **observability**, security, governance, **customer experience,** tooling…
- Used by 100s of developers
- Saved **years** of dev time
- 100+ engineers, 2 divisions

Early days | **2020** | **Today**

One down, hundreds to go

Adopt and scale **Serverless** across organization

Maintain **standards and best-practices** for architecture, governance, security, observability

Help organization to **deliver value faster** for our customers

So, we've built a serverless platform

Encapsulated automation and best practices into blueprints **used by 100s of developers for dozens of services**

New service creation time reduced by **99%**

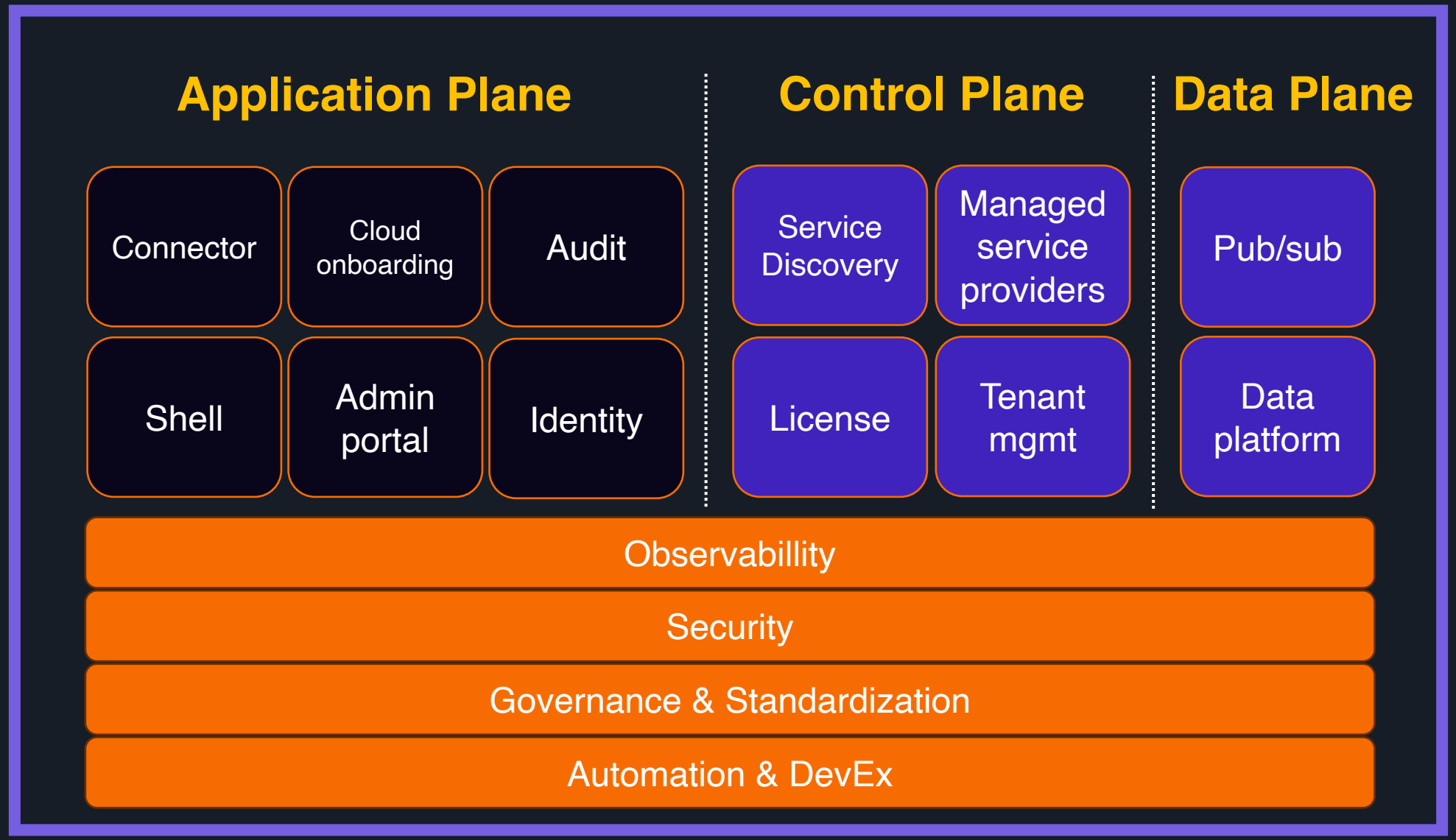
Saved **YEARS of development time and MILLIONS of $$$**

Ran Isenberg

CYBERARK®

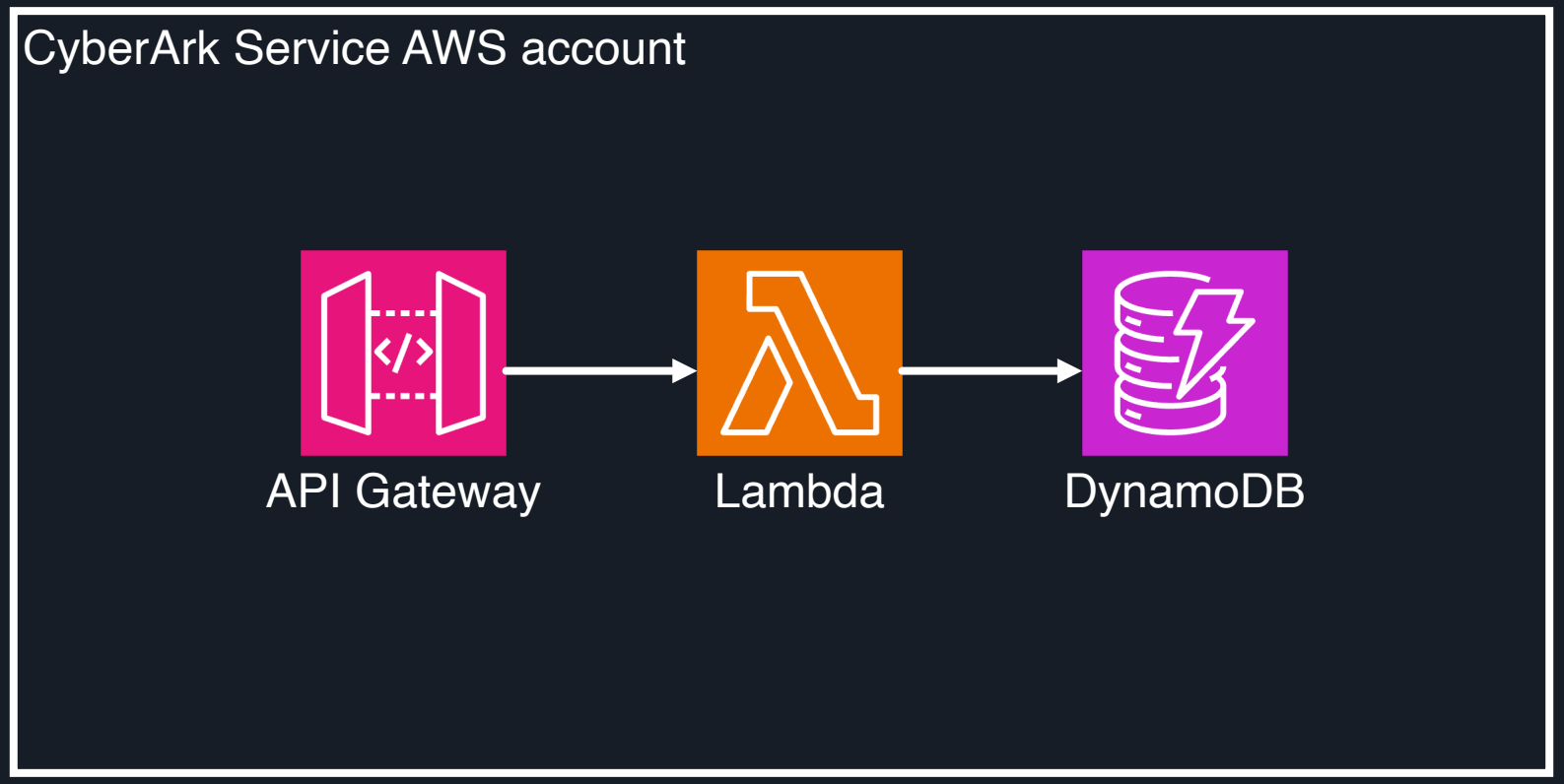
- Principal Software Architect @CyberArk
- AWS Serverless Hero
- https://www.RanTheBuilder.Cloud
- RanBuilder @RanTheBuilder.cloud

Platform Engineering: SaaS Products Overview

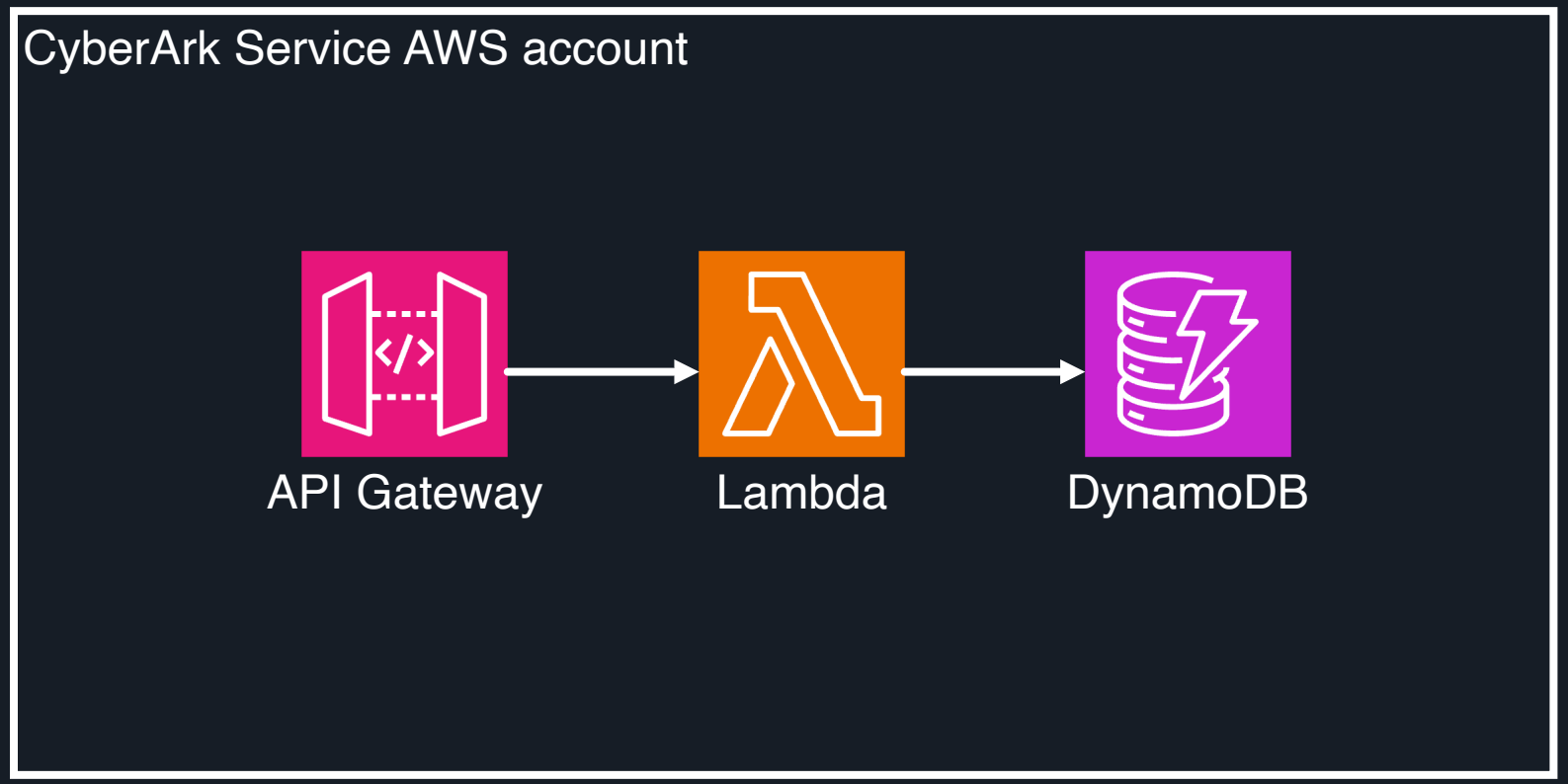


Scaling Enterprise-Grade Serverless Services

Challenges for a new enterprise-grade SaaS Service

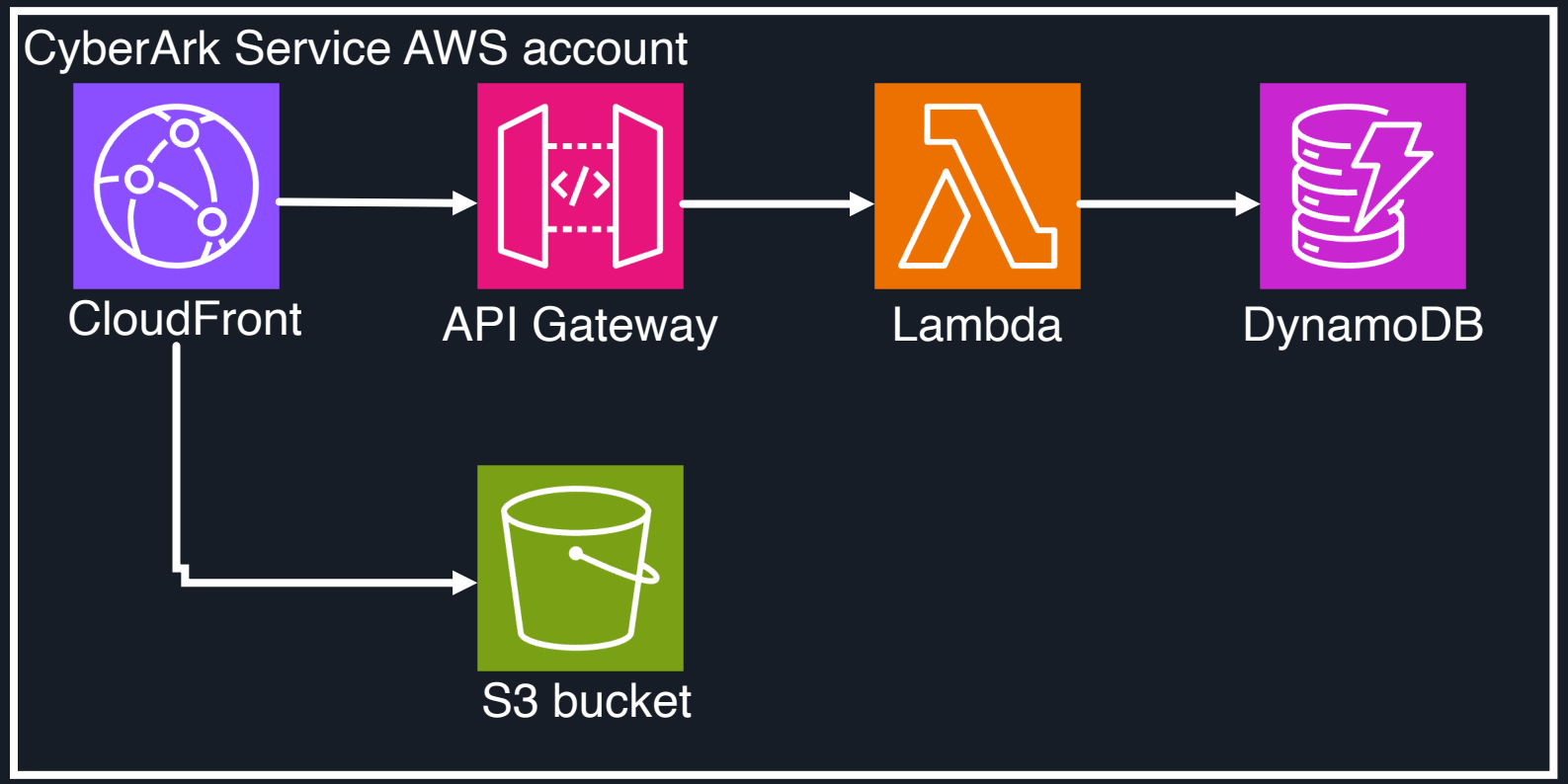


Challenges for a new enterprise-grade SaaS Service



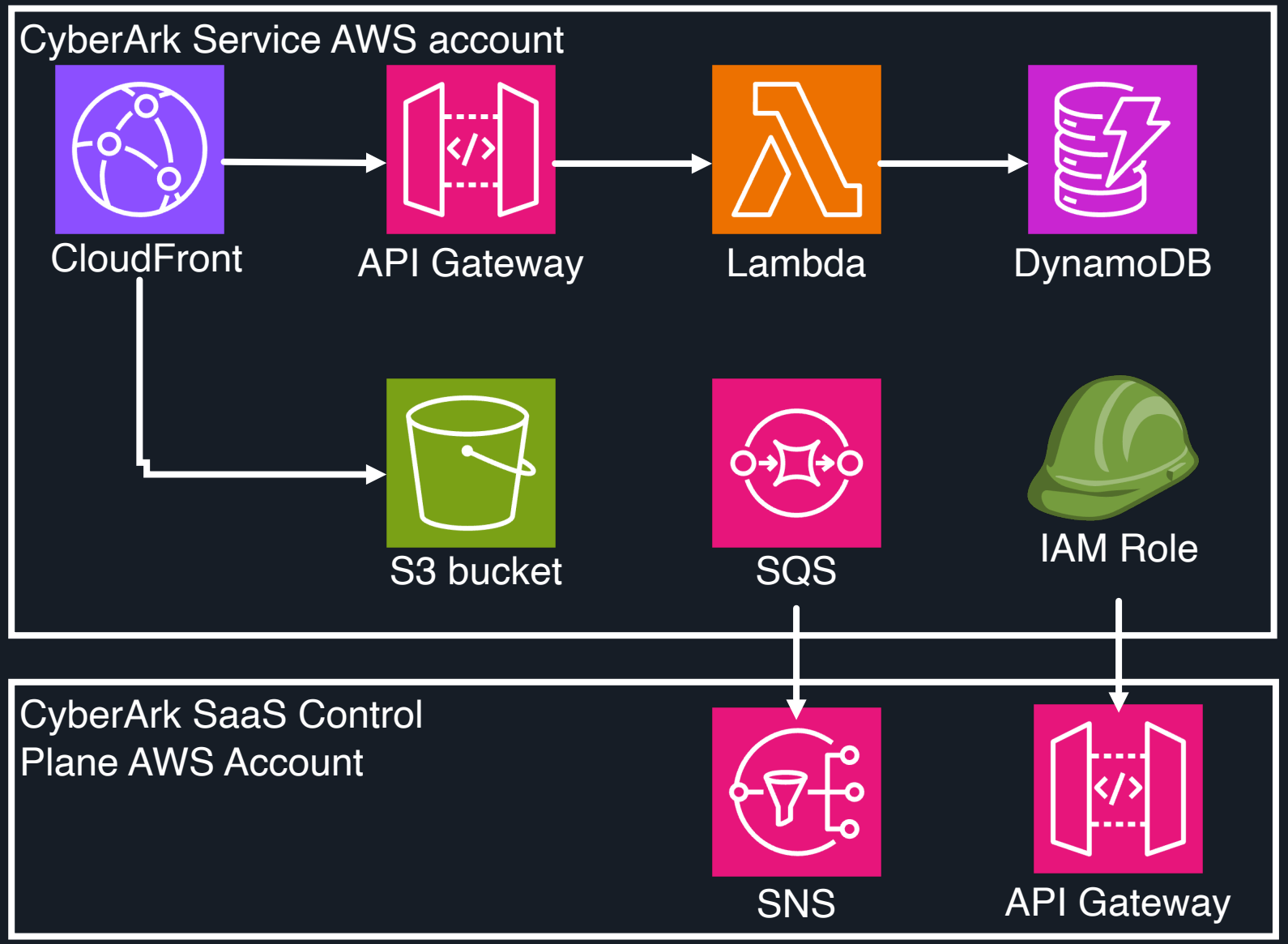
IaC **CI/CD pipeline** **Best practices** **Testing** **Observability**

Challenges for a new enterprise-grade SaaS Service



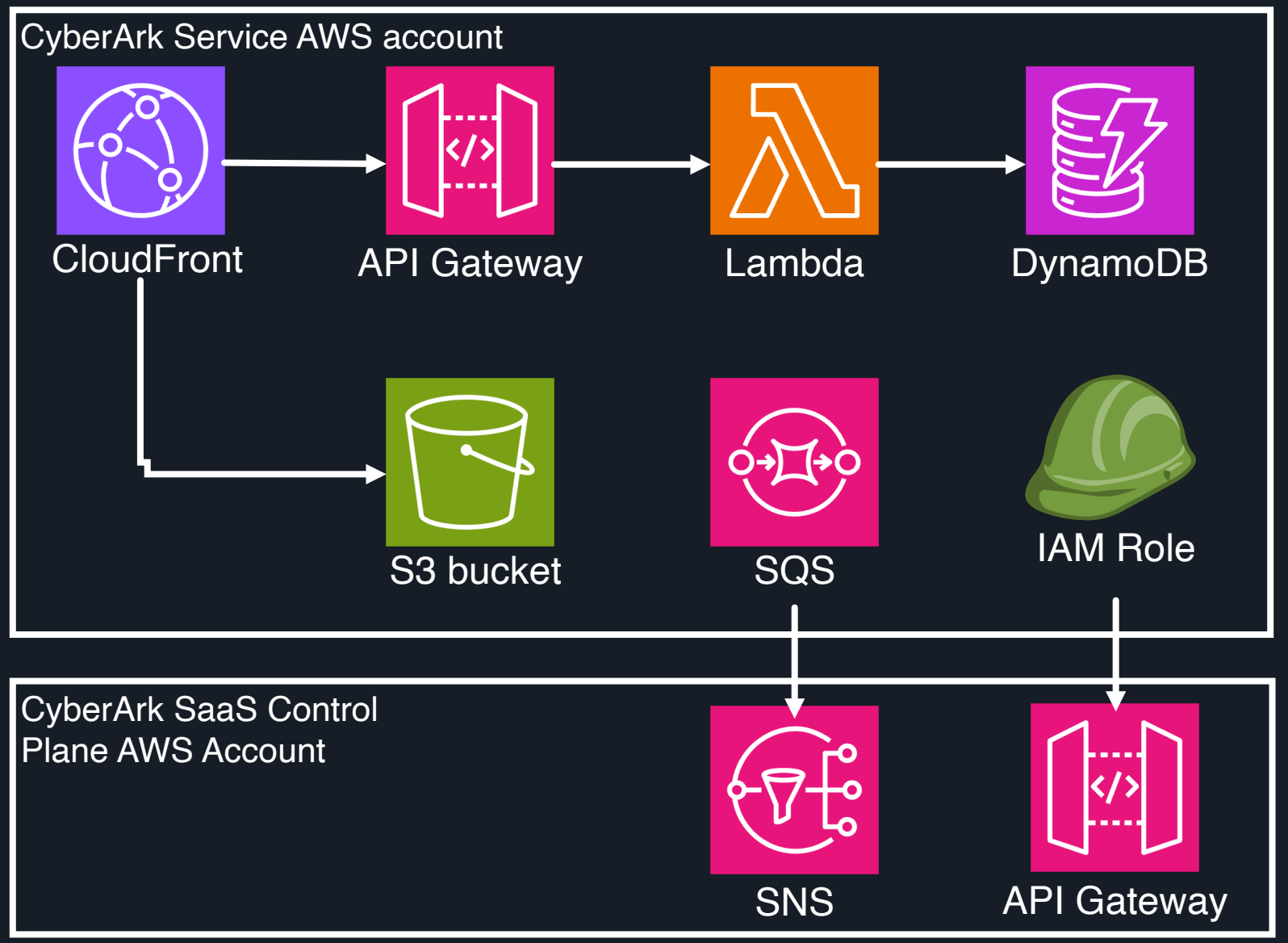
IaC **CI/CD pipeline** **Best practices** **Testing** **Telemetry**

Challenges for a new enterprise-grade SaaS Service



Integrating with the SaaS Control Plane

- Tenant management integration
- Subdomain registration
  (hostname, certificate)
- Cross account access
  (role delegation, topic registration)



Enterprise-grade is complex!

- One microservice is **NOT** enough
  - We need multiple microservices
- **Integration** with other SaaS components
- **Deploy** to **multiple** AWS accounts & regions
- **Adhere** to serverless and enterprise best-practices

Enterprise-grade “Hello World” Saas Service

BEFORE

AFTER

Serverless Platform Engineering

5+ months

3 hours

Automation DevEx

Let’s Build a “One Click” SaaS Serverless Service

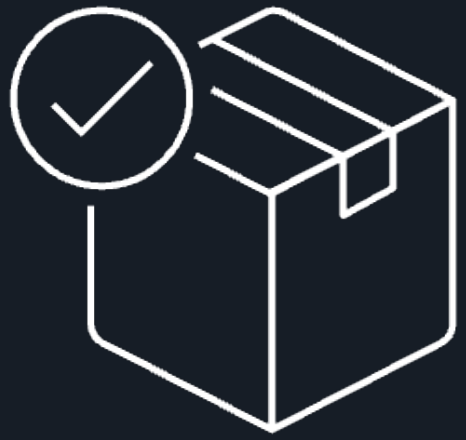
CRUD API

BEST PRACTICES

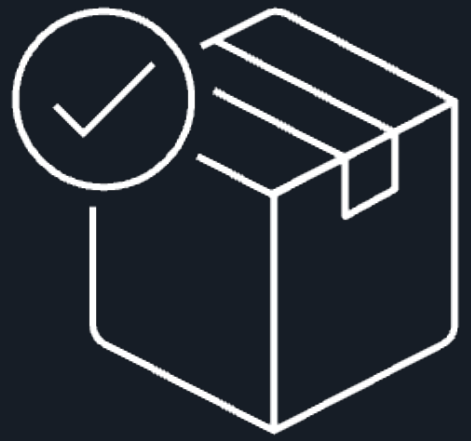
FRONTEND

CYBERARK UNIFIED UI

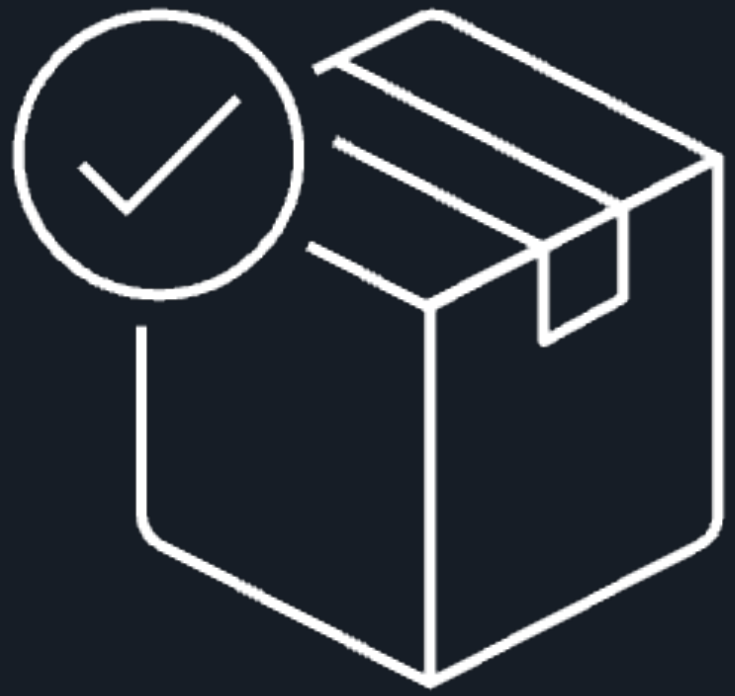
SAAS CONTROL PLANE



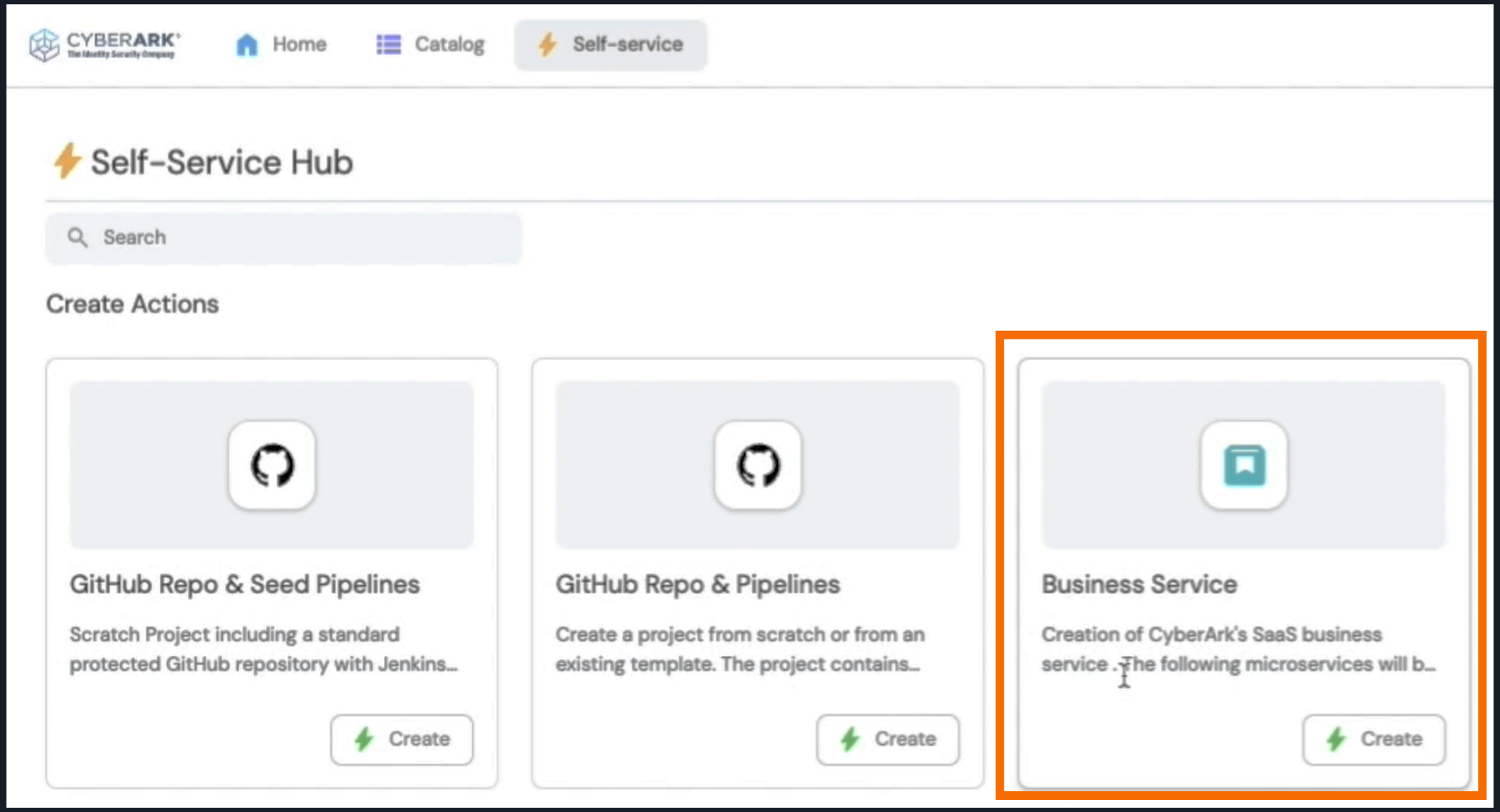
Let’s Build a “One Click” SaaS Serverless Service



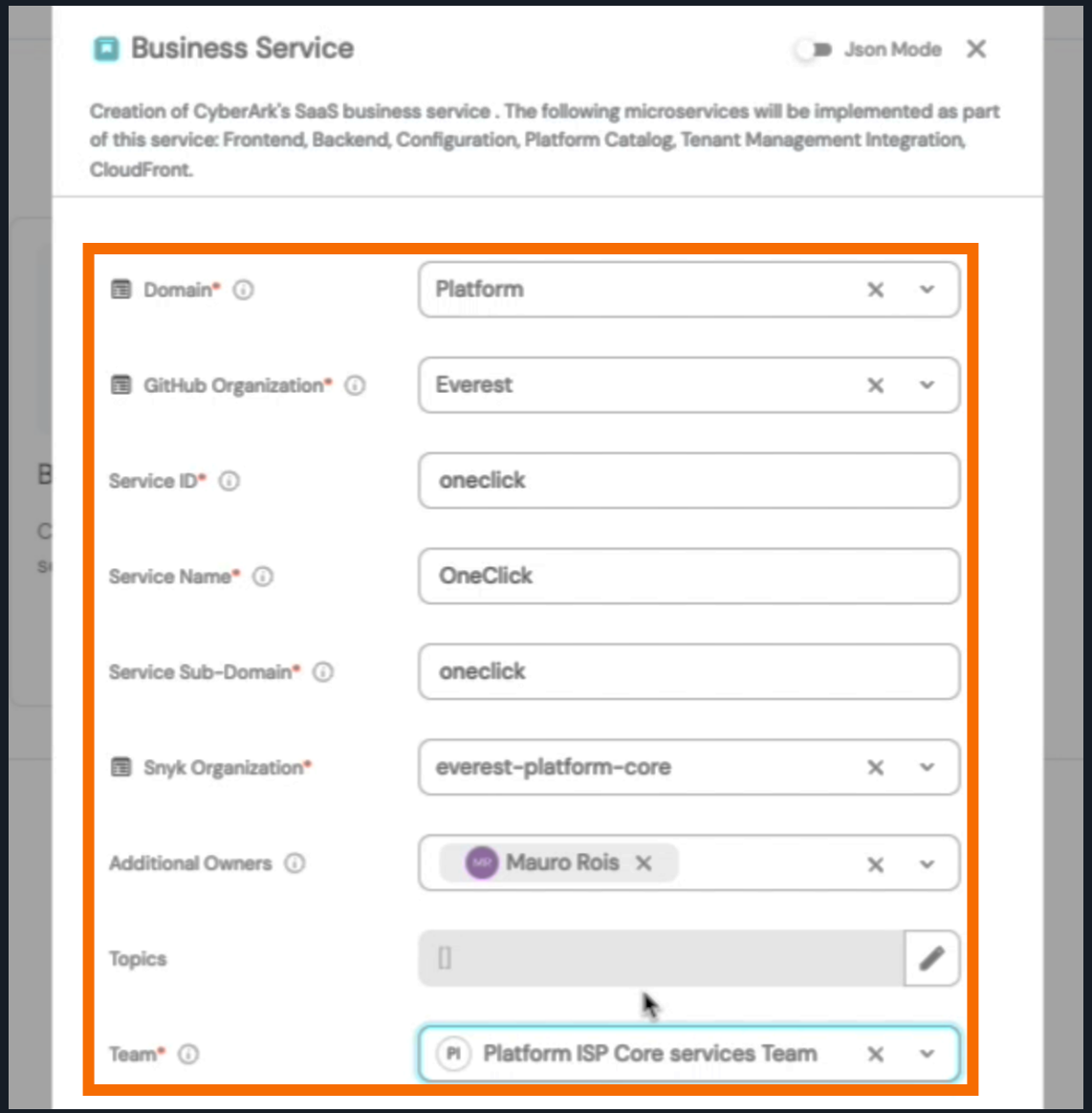
Let’s Build a “One Click” SaaS Serverless Service



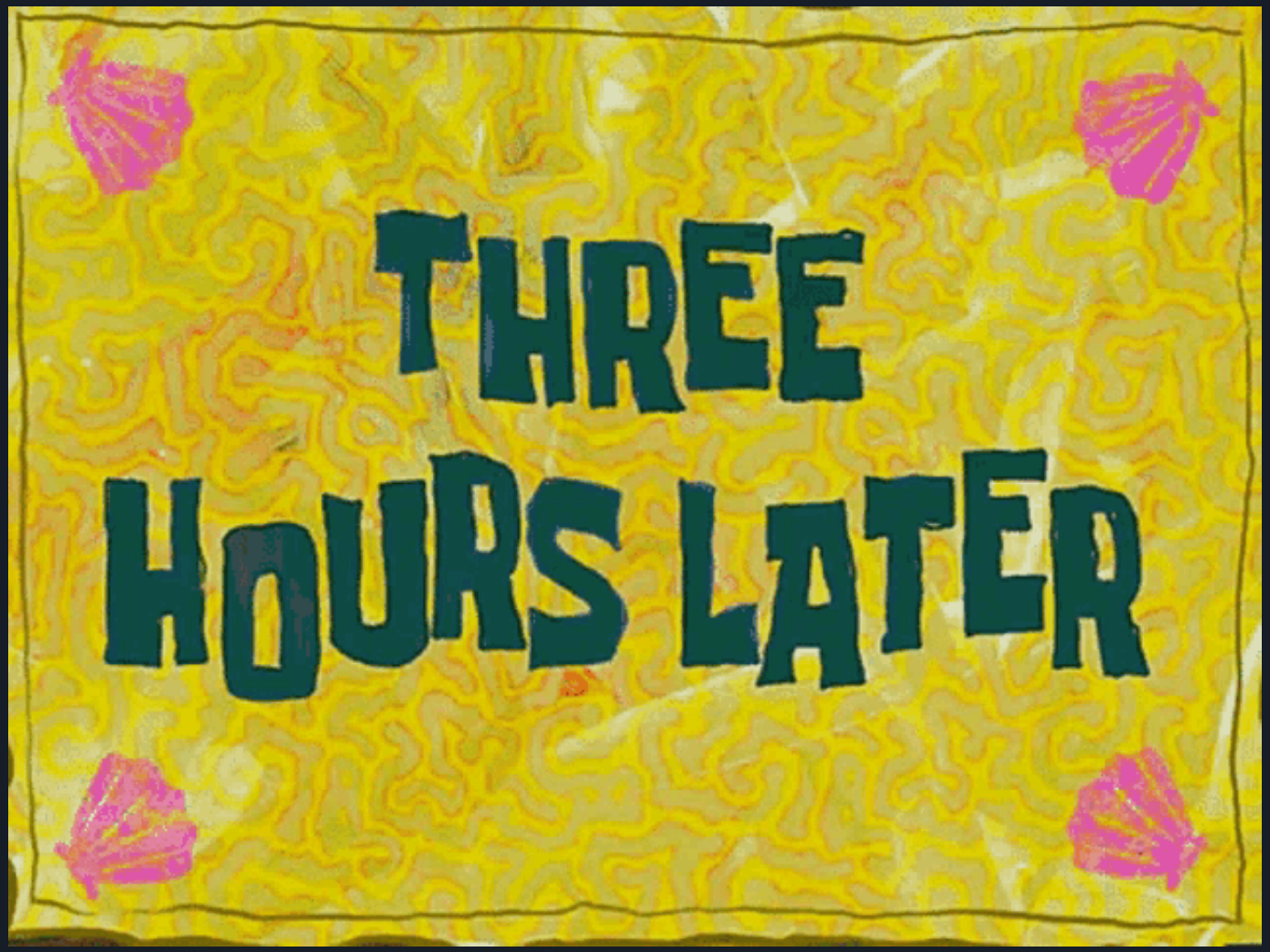
It all Starts with the Developer Portal



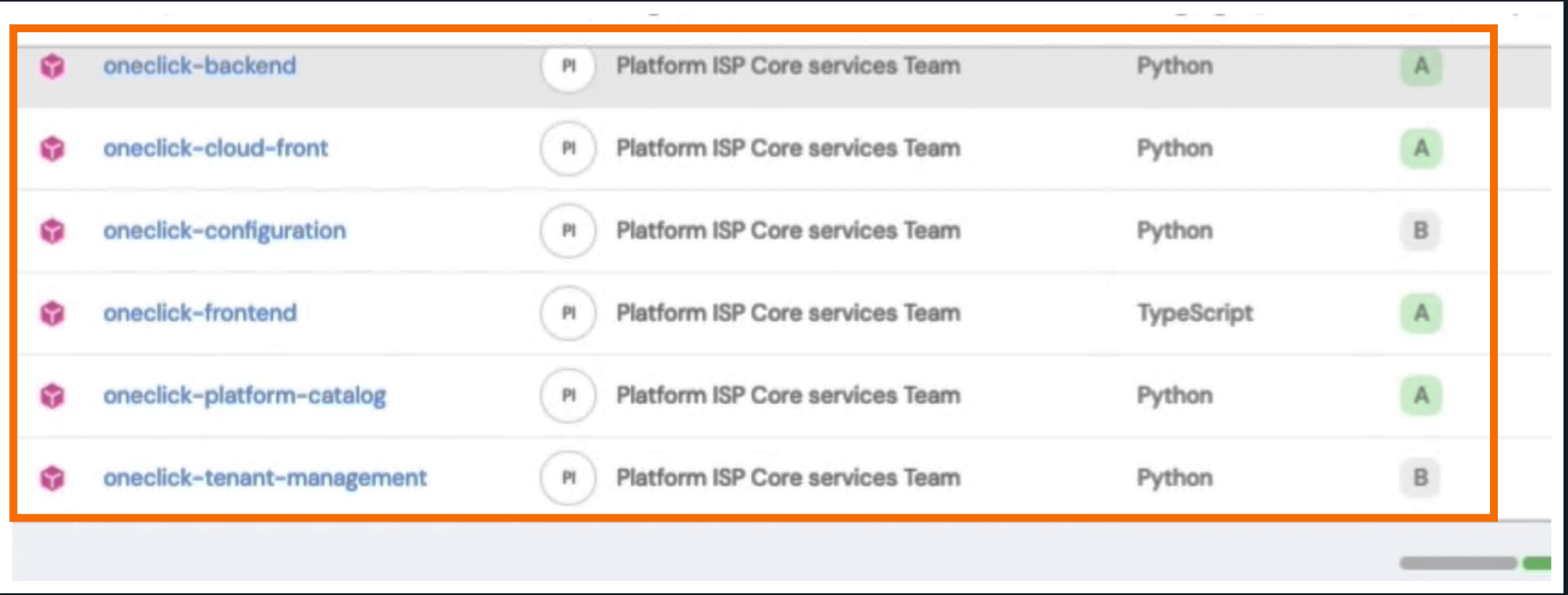
It all Starts with the Developer Portal



It all Starts with the Developer Portal



New components and integrations provisioned

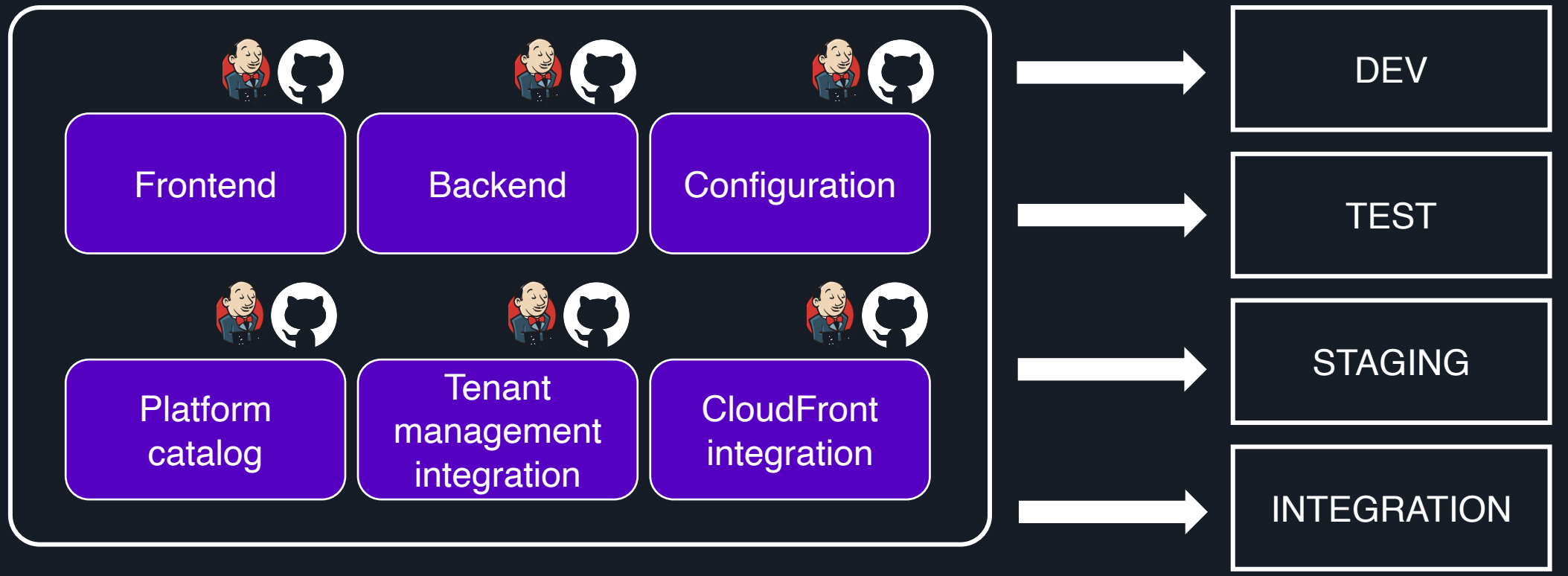


oneclick-backend	PI	Platform ISP Core services Team	Python	A
oneclick-cloud-front	PI	Platform ISP Core services Team	Python	A
oneclick-configuration	PI	Platform ISP Core services Team	Python	B
oneclick-frontend	PI	Platform ISP Core services Team	TypeScript	A
oneclick-platform-catalog	PI	Platform ISP Core services Team	Python	A
oneclick-tenant-management	PI	Platform ISP Core services Team	Python	B

New service deployed to production, UI included

What did we see?

6 blueprints. Best practices, security, observability, tooling baked-in.



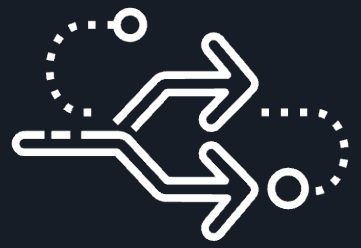
x2 regions

Go and build value!

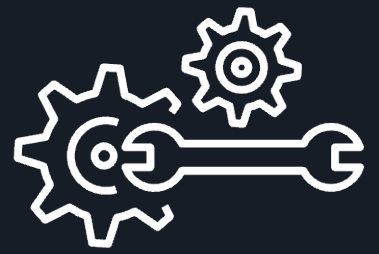
Focus on business domain instead of undifferentiated heavy lifting

Automation DevEx Tips

Simple to use, with **minimal prerequisites**



Retry-able, customize-able, ability to clean-up failures.



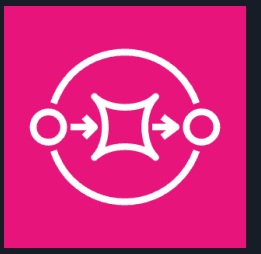
Ongoing maintenance, ”fake services” runs

Architectural Bluprints

Architectural Bluprints Examples

Lambda with dynamic provisioned concurrency

Secure S3 bucket



SQS Queue with DLQ and redrive

KMS with CMK

Tenant aware Idempotency table

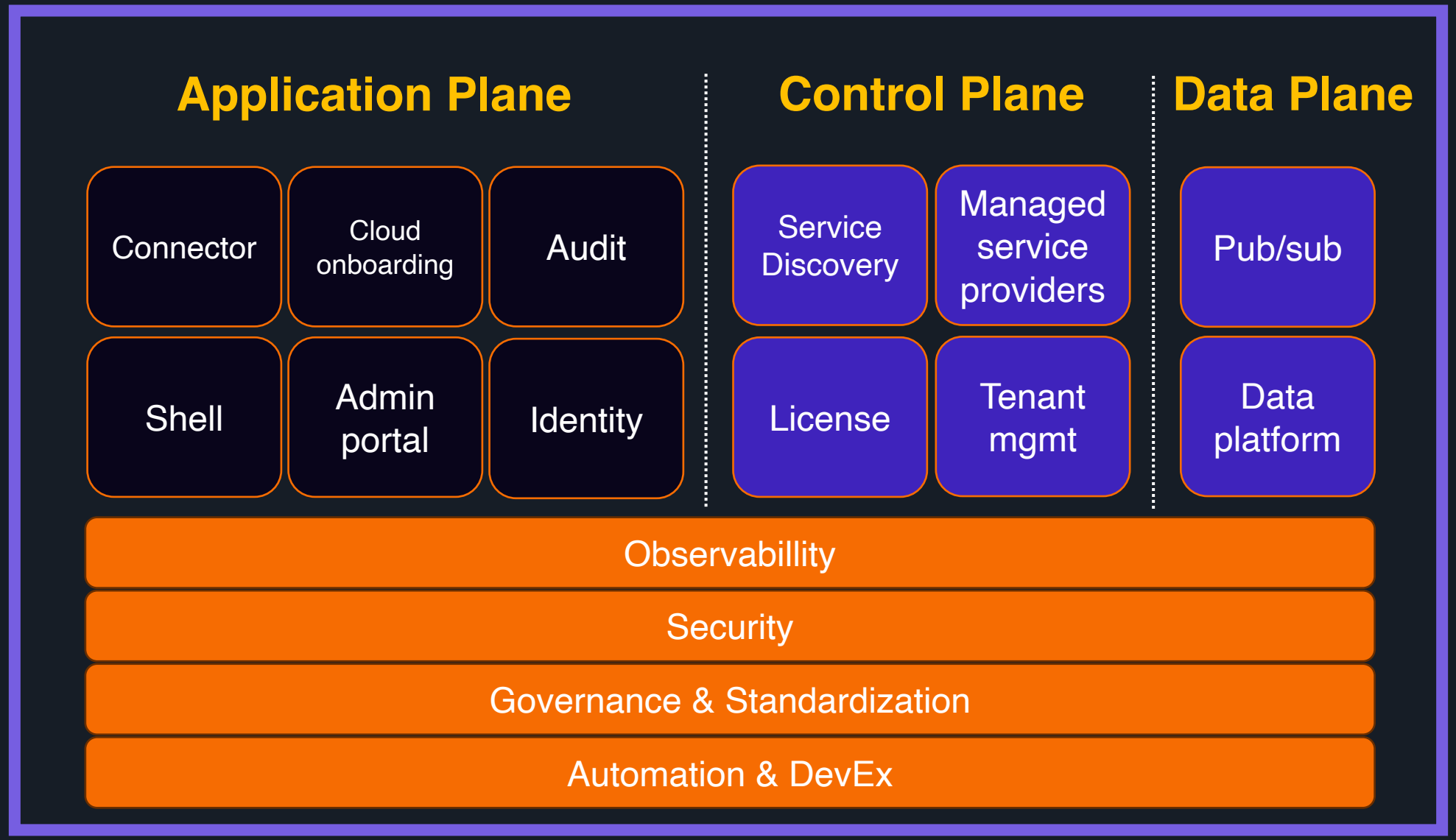
WAF ACL association

Secure S3 Bucket CDK Construct

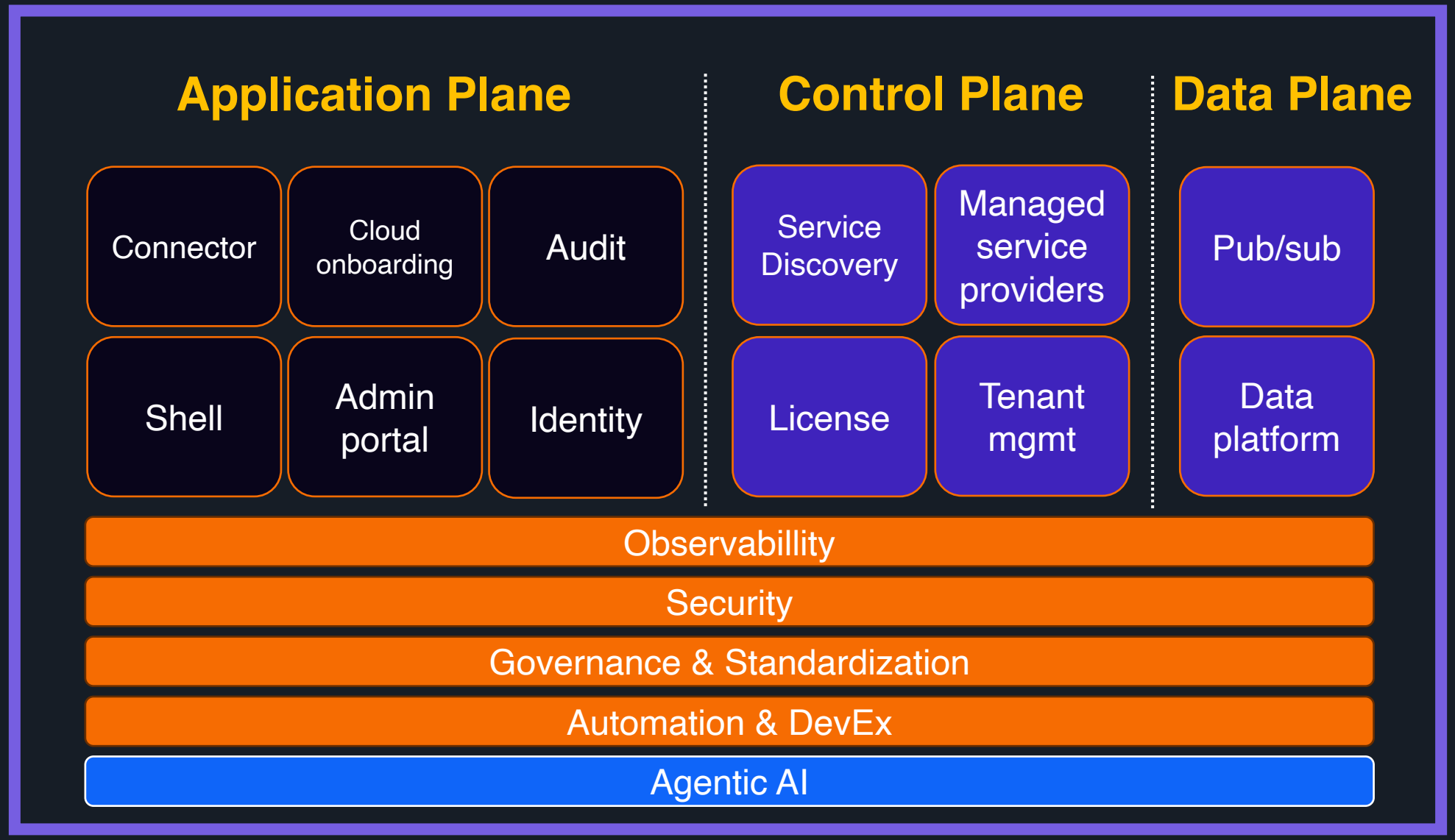
```python
from aws_cdk import RemovalPolicy
from aws_cdk import aws_s3 as s3
from constructs import Construct

class SecureBucket(Construct):
    def __init__(self, scope, identifier, bucket_name, is_versioned):
        super().__init__(scope, identifier)
        production_env = ...
        self.bucket = s3.Bucket(
            self,
            id=f'{identifier}Bucket',
            bucket_name=bucket_name,
            enforce_ssl=True,
            removal_policy=RemovalPolicy.RETAIN if production_env else RemovalPolicy.DESTROY,
            auto_delete_objects=False if production_env else True,
            block_public_access=s3.BlockPublicAccess.BLOCK_ALL,
            encryption=s3.BucketEncryption.S3_MANAGED,
            versioned=is_versioned)
```

Your products are evolving

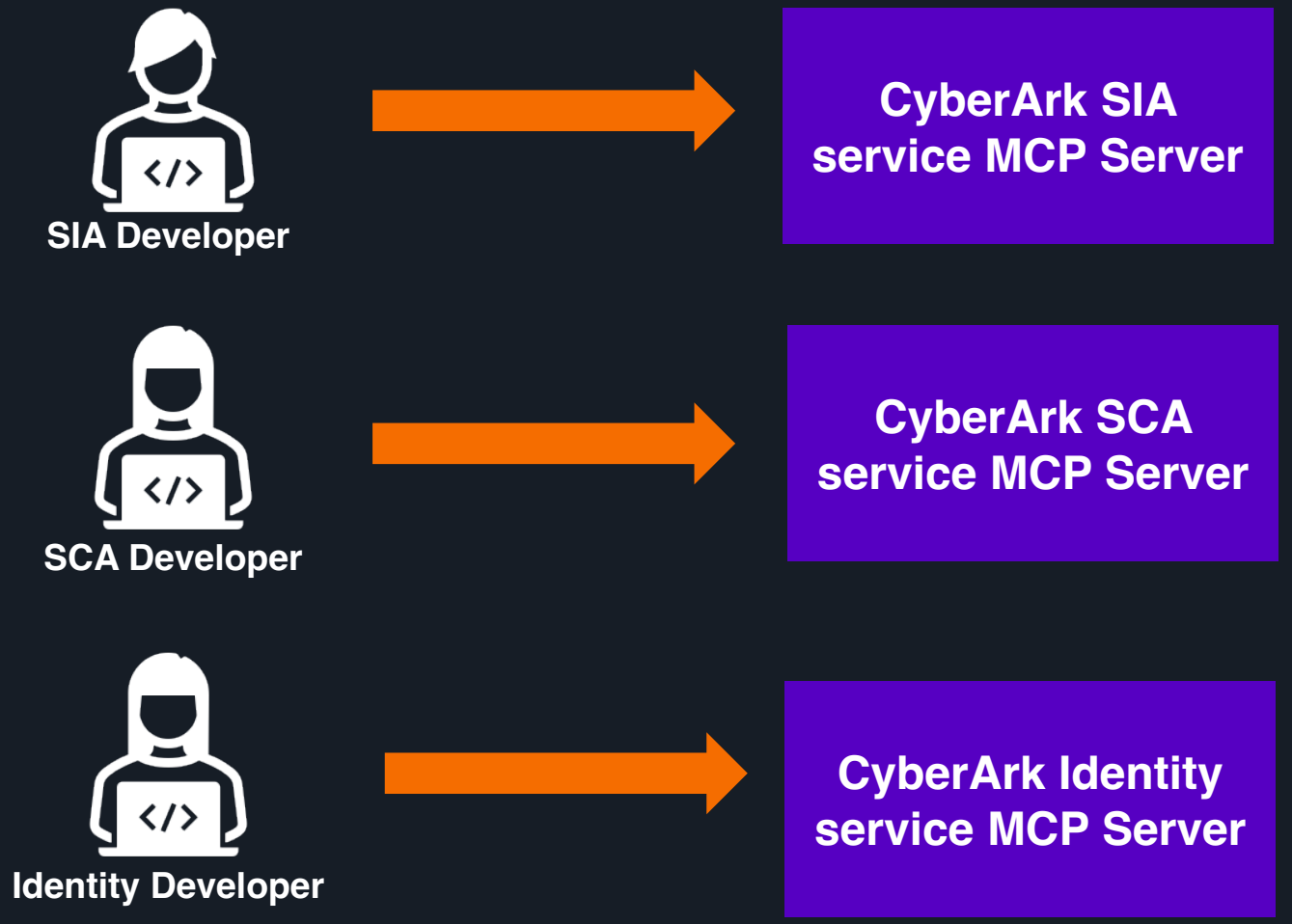
Platform Engineering: SaaS Products Overview



Platform Engineering: SaaS Products Overview



Building MCP Servers at scale can be challenging



Building MCP Servers at scale can be challenging

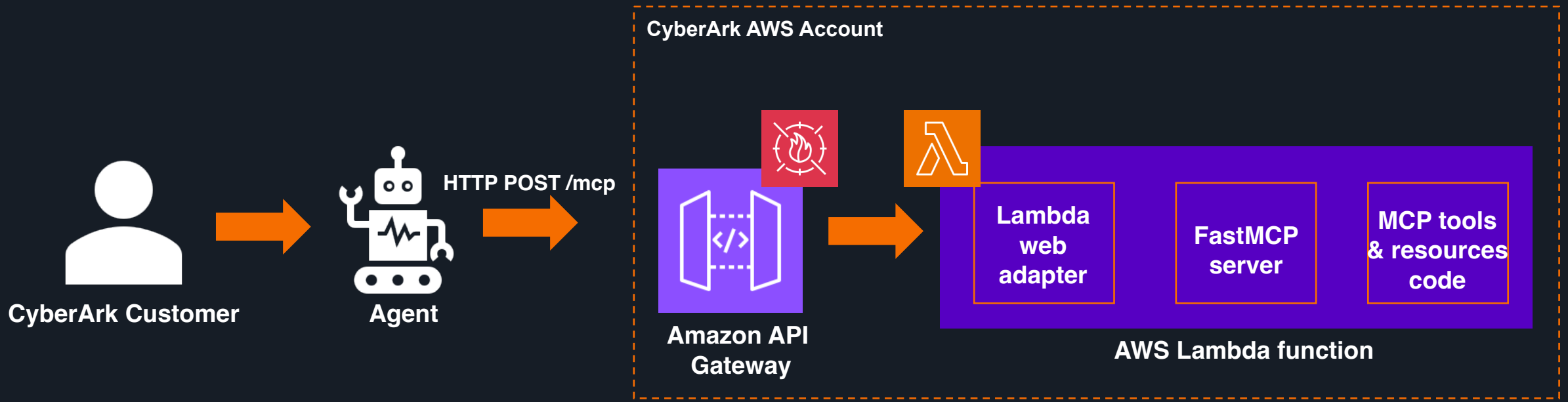
- **Teams "reinventing the wheel"**
  - Authentication & authorization
  - CDK IaC
  - CI/CD pipeline
  - Testing with MCP client
  - Observability
- **Duplicated efforts, architectural inconsistencies, technical debt**

CyberArk SIA service MCP Server

CyberArk SCA service MCP Server

CyberArk Identity service MCP Server

Platform Engineering and MCP Blueprints



IaC

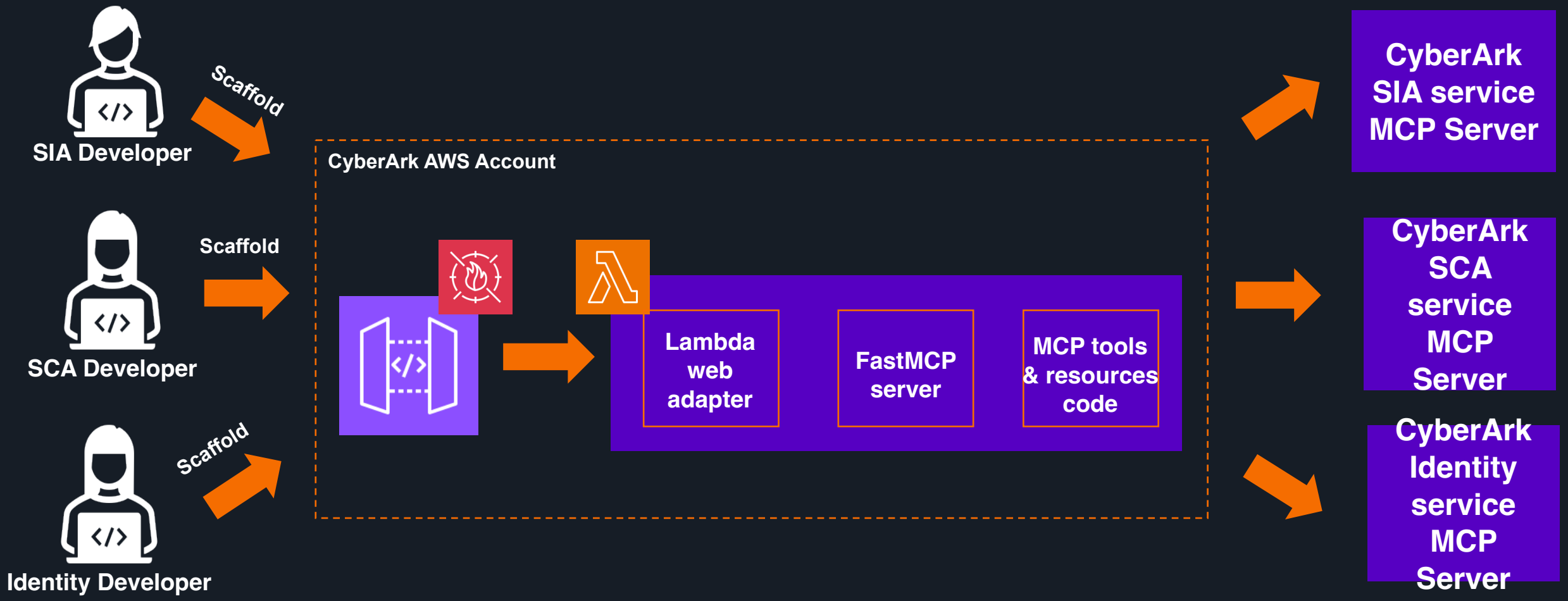
CI/CD pipeline

Security best practices

Testing

Observability

Platform Engineering and MCP Blueprints



In conclusion


Don’t boil the ocean


One size doesn’t fit all workloads


Documentation and education are key

Build with your customers

Check out these other sessions

CNS206: Building Agentic AI Architectures with AWS Serverless

Monday (Dec 1) @ 10:00a – MGM 304 – Builders Session

CNS403: Best practices for serverless developers

Tuesday (Dec 2) @ 12:00p – Mandalay Bay Islander F – Breakout Session

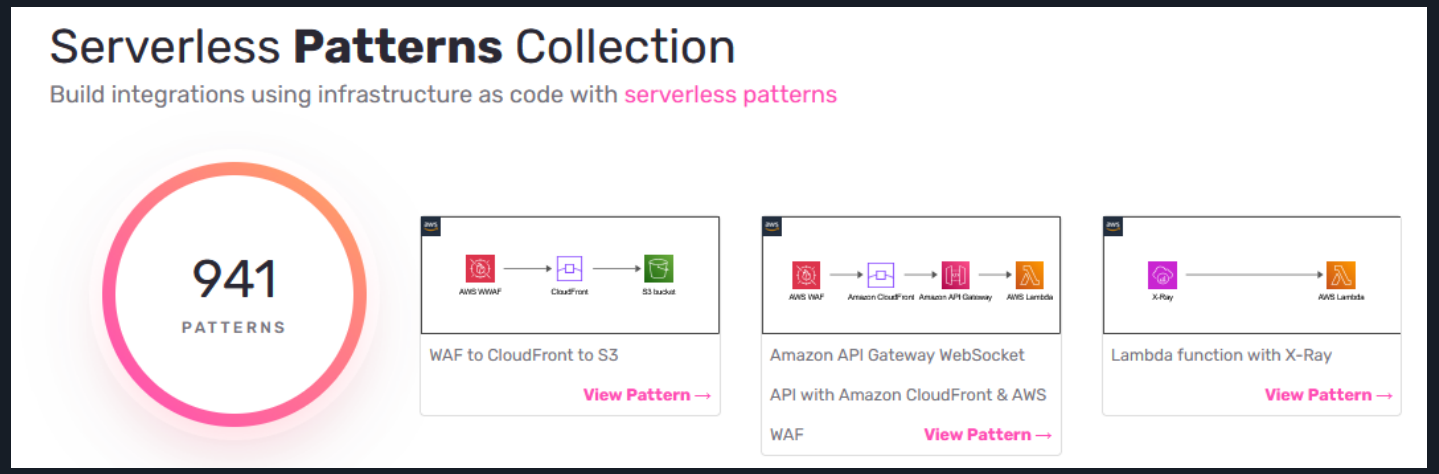
API309: Serverlesspresso: Build an event-driven application from the ground up

Thursday (Dec 4) @ 12:00p – MGM Grand 120 - Workshop

Serverless & AppInt Resources

https://s12d.com/RIV2025

Continue your AWS serverless learning
Powertools for AWS Lambda
Serverless Land patterns



Thank You!

This session's content
and much more

Anton Aleksandrov
antonal80
aal80.github.io/whoami

Ran Isenberg
ranbuilder
ranthebuilder.cloud

Please complete the session survey in the mobile app