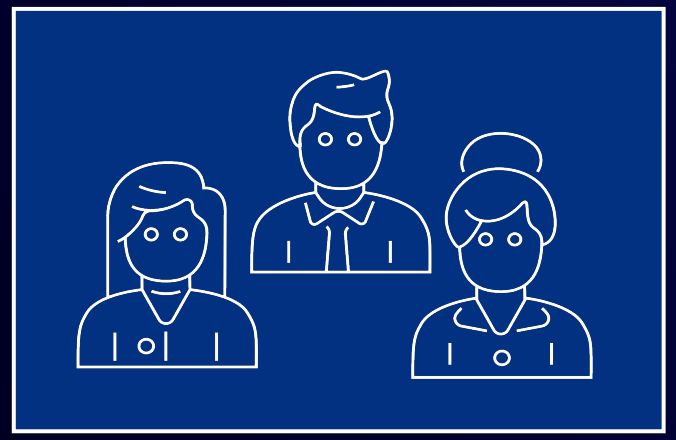
Scaling Serverless Development with Platform Engineering

Anton Aleksandrov
Principal Solution Architect, Serverless
AWS

The operational model evolution

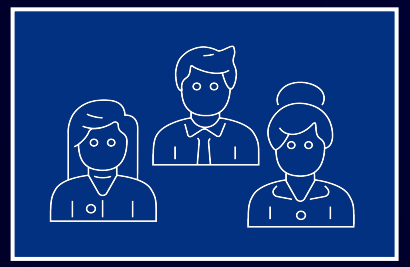


Application development team

Infrastructure/ops team

The operational model evolution

We ♥ DevOps

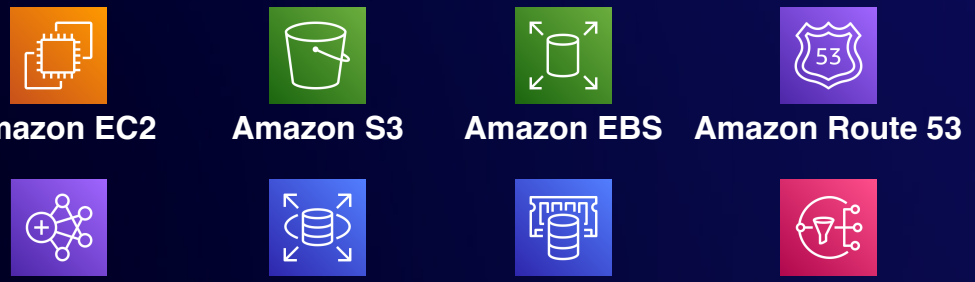


Application development team

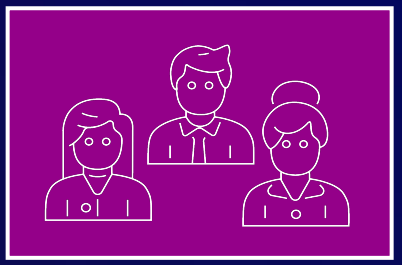
CI/CD tools and processes

The Application Layer

The Infrastructure Layer

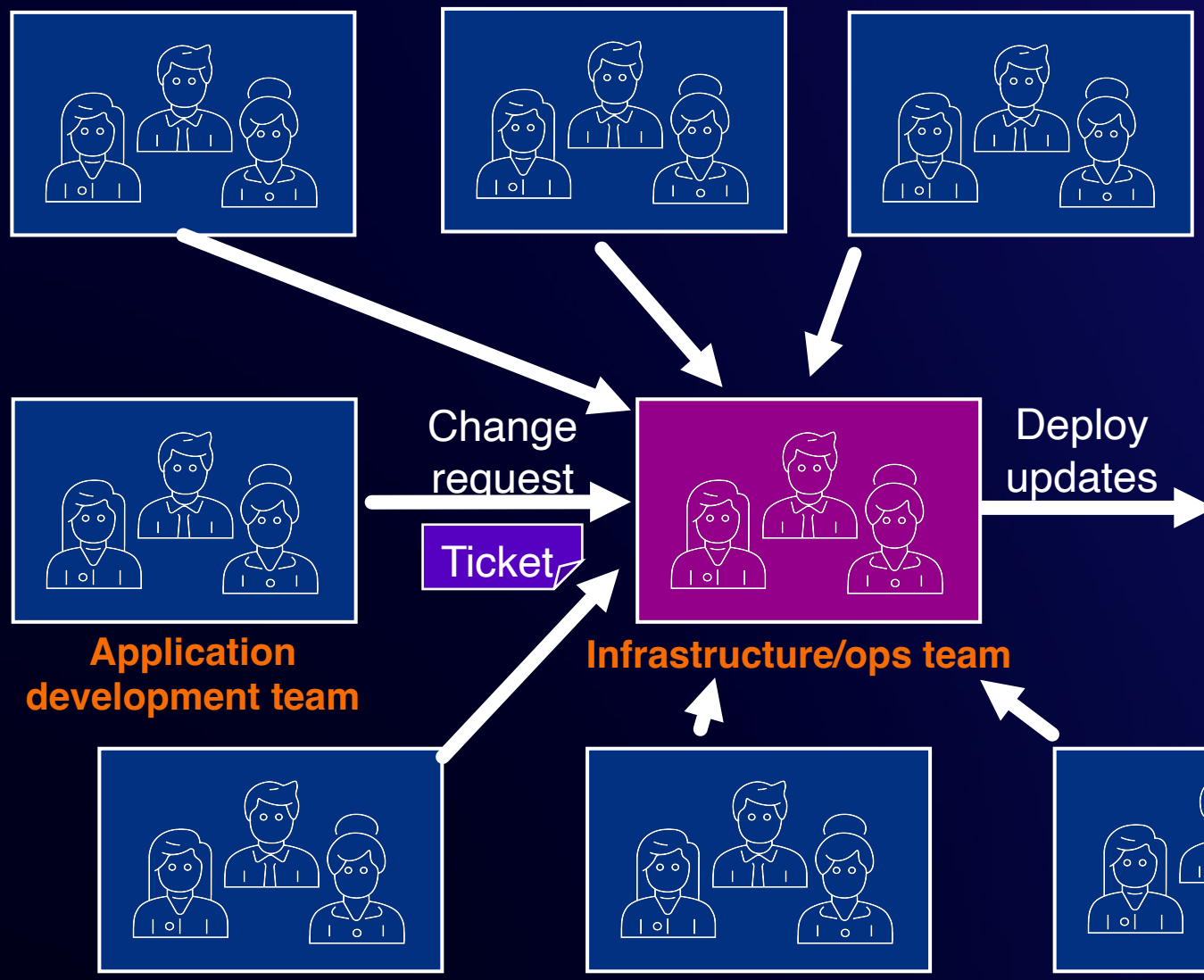


IaC tools and processes



Infrastructure/ops team

The operational model evolution



The Infrastructure Layer

Amazon EC2 | Amazon S3 | Amazon EBS | Amazon Route 53

Amazon EMR | Amazon RDS | ElastiCache | Amazon SNS

Shift-left to the rescue?

?!

Application development teams

Cognitive Load

Code

Ship

Run

Secure

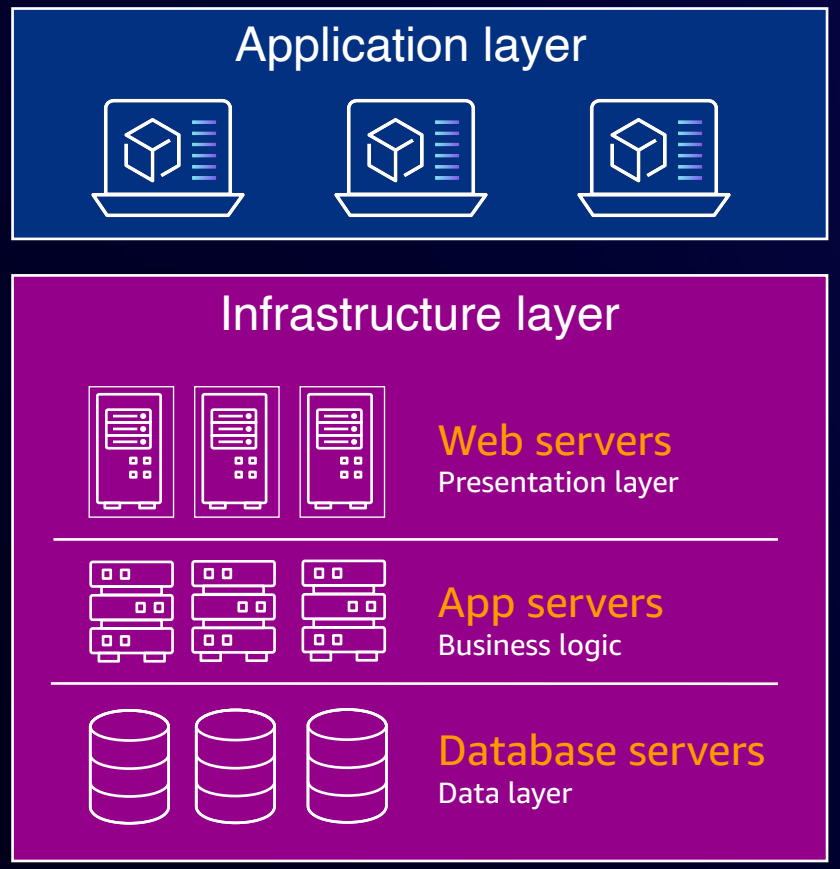
Optimize

Sustainable

2000

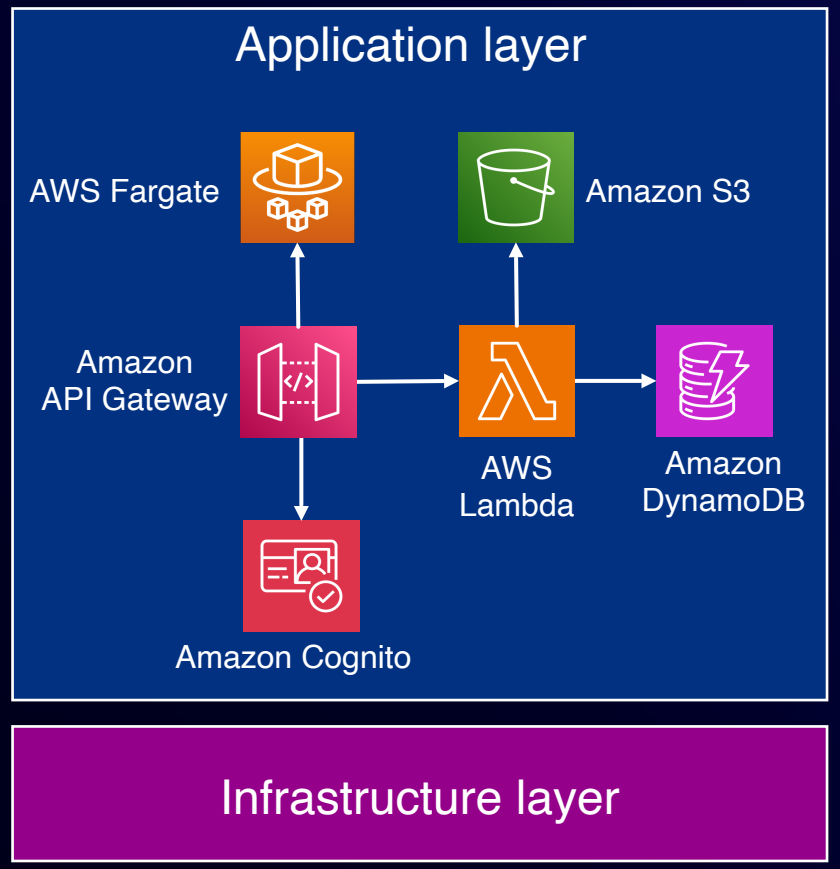
Today

Traditional applications



- **Rigid separation** between infrastructure and application teams, tools, processes
- Frequently **coupled, manual workflows** for release and quality control
- **Long cycles** - need a new database? Open a ticket, we'll get back to you…

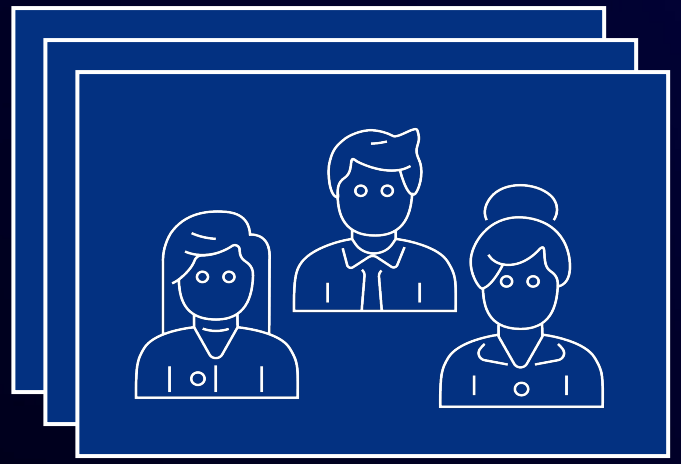
Serverlessapplications



- **“Infrastructure” is redefined.** A function, an event-source mapping, an event routing rule are **app resources** owned by the app team
- Updating app resources is commonly a part of **application developer’s responsibilities**.
- Guided **shift-left responsibilities** and trust between teams promote operational agility

The ownership boundaries

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

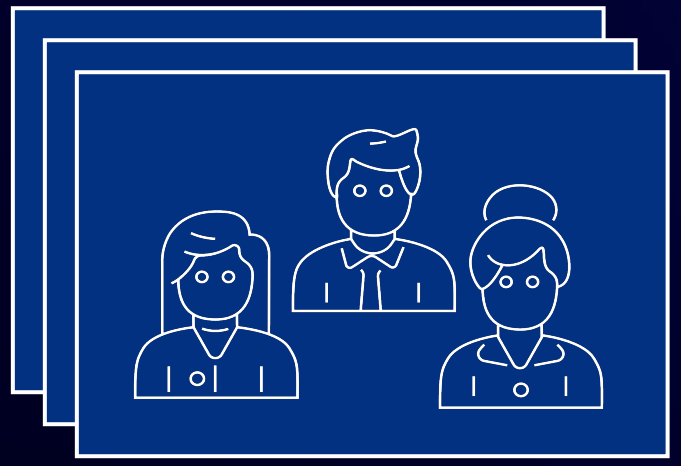


Application development teams

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

Infrastructure/ops team

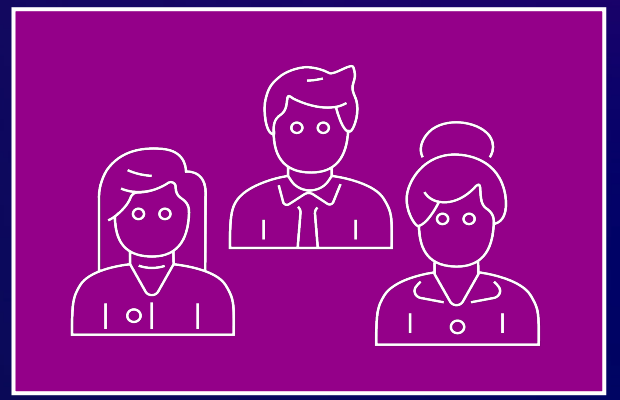
The ownership boundaries



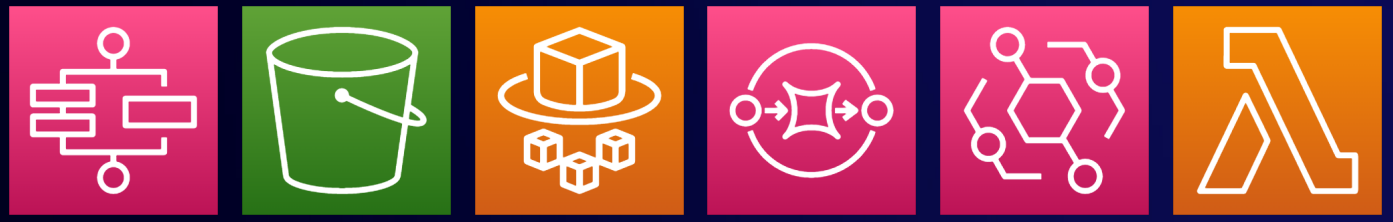
Application development teams

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

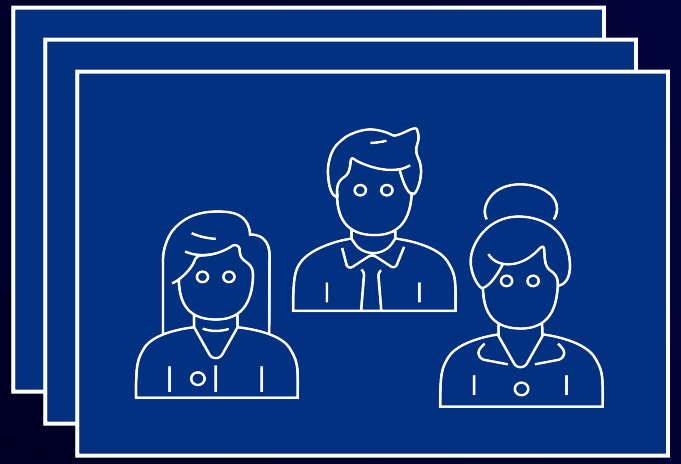
Standardization! Security! Governance! Observability! Cost efficiency!



Infrastructure/ops team



Striking the balance



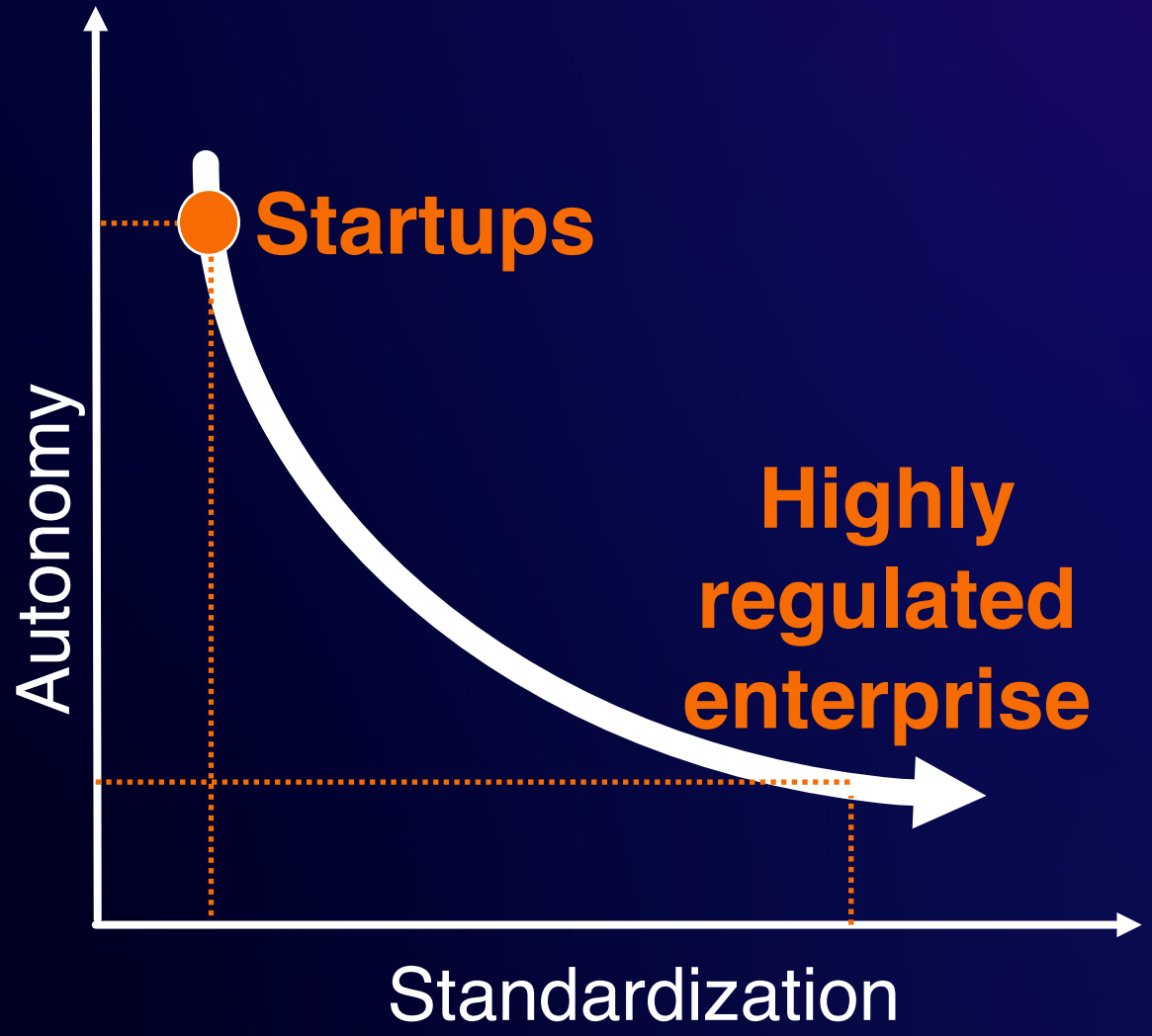
AppDev teams

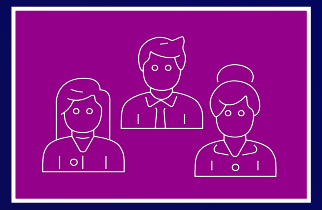
Infra/ops team

Autonomy ⟷ **Standardization**

Striking the balance

AppDev teams

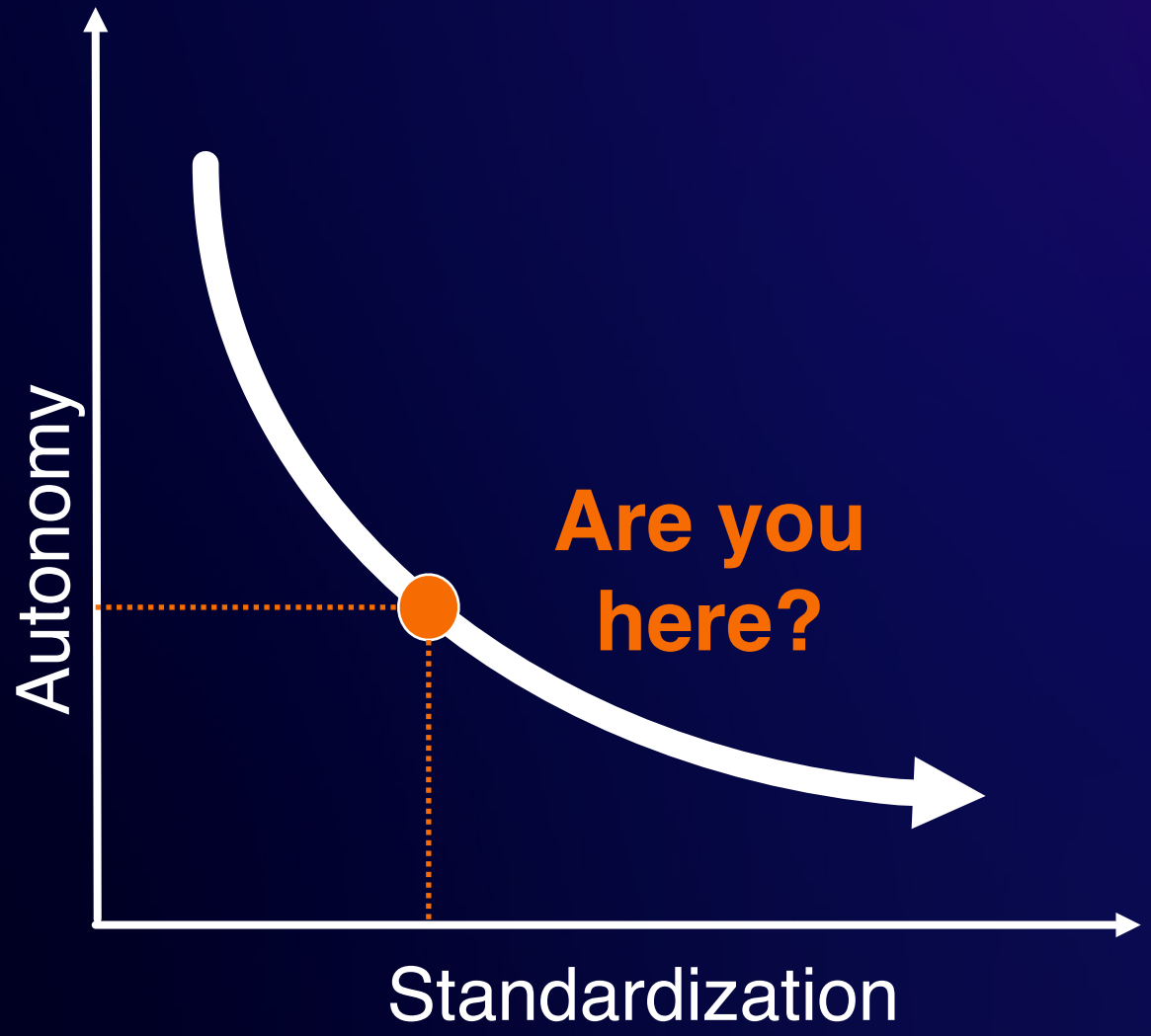


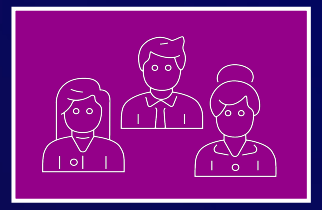


Infra/ops team

Striking the balance

AppDev teams

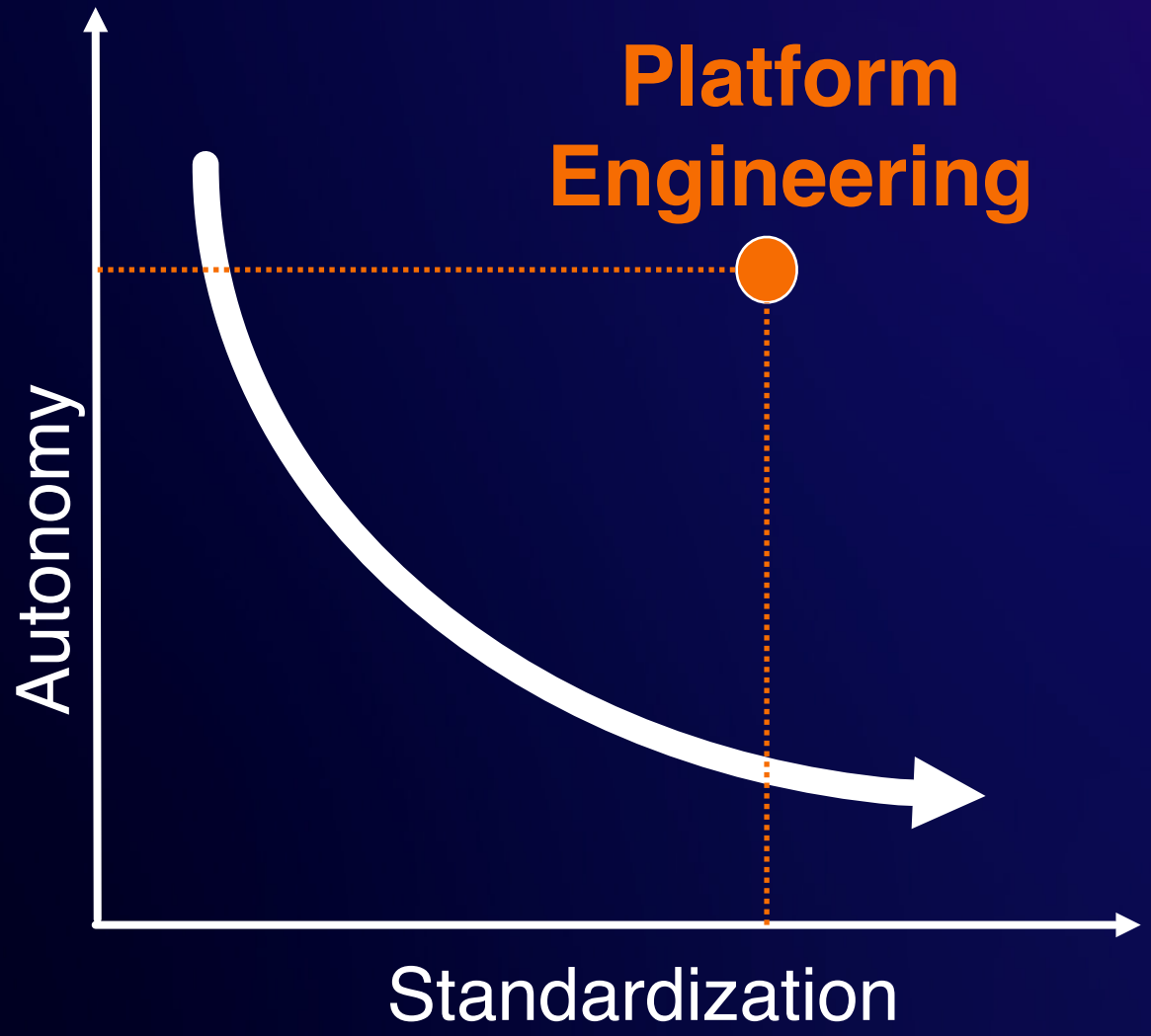


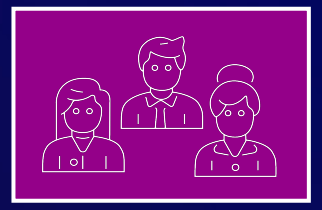


Infra/ops team

Striking the balance

AppDev teams





Infra/ops team

A digital platform is a foundation of self-service APIs, tools, services, knowledge and support which are arranged as a compelling internal product.

Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced co-ordination.

Evan Bottcher

“What I Talk About When I Talk About Platforms”, martinfowler.com

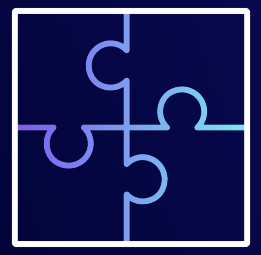
https://martinfowler.com/articles/talk-about-platforms.html

Internal Developer Platforms (IDP) empower organizations to achieve economies of scale in their cloud adoption journey

Internal Developer Platforms (IDP) empower organizations to achieve economies of scale in their cloud adoption journey

Velocity

Reduce the time it takes to get new applications in front of customers



Governance

Ensure application teams operate safety and securely in the cloud

Efficiency

Optimize spend via efficient resource utilization and specialization

Customer example

"Using a serverless platform on AWS, we've gone from weekly deployments to deployments that we do multiple times a day, and from what used to be hours of at least two engineers' time, we've been able to shave that down to several minutes.

We were able to streamline the deployment process, eliminate manual upgrades and patches, and ensure consistent and reliable workloads."

- Skylar Graika, Senior Principal Software Engineer, Smartsheet

https://aws.amazon.com/solutions/case-studies/smartsheet-ecs-fargate-case-study/

Where do I start

Traditional ownership boundaries

Depends on

Codebase

Codebase

Codebase

CI/CD

Monitoring

Security

IaC

CI/CD

Monitoring

Security

IaC

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)

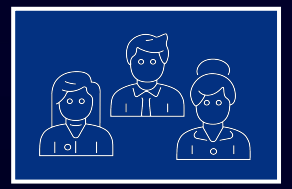
AppDev teams

?

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

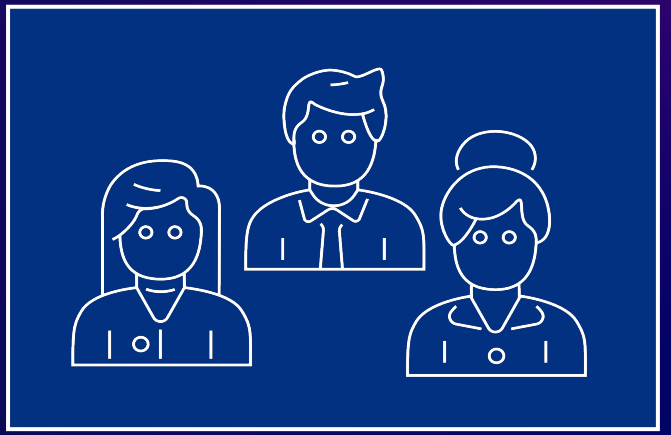
IaC templates

Security and governance policies

Platform team

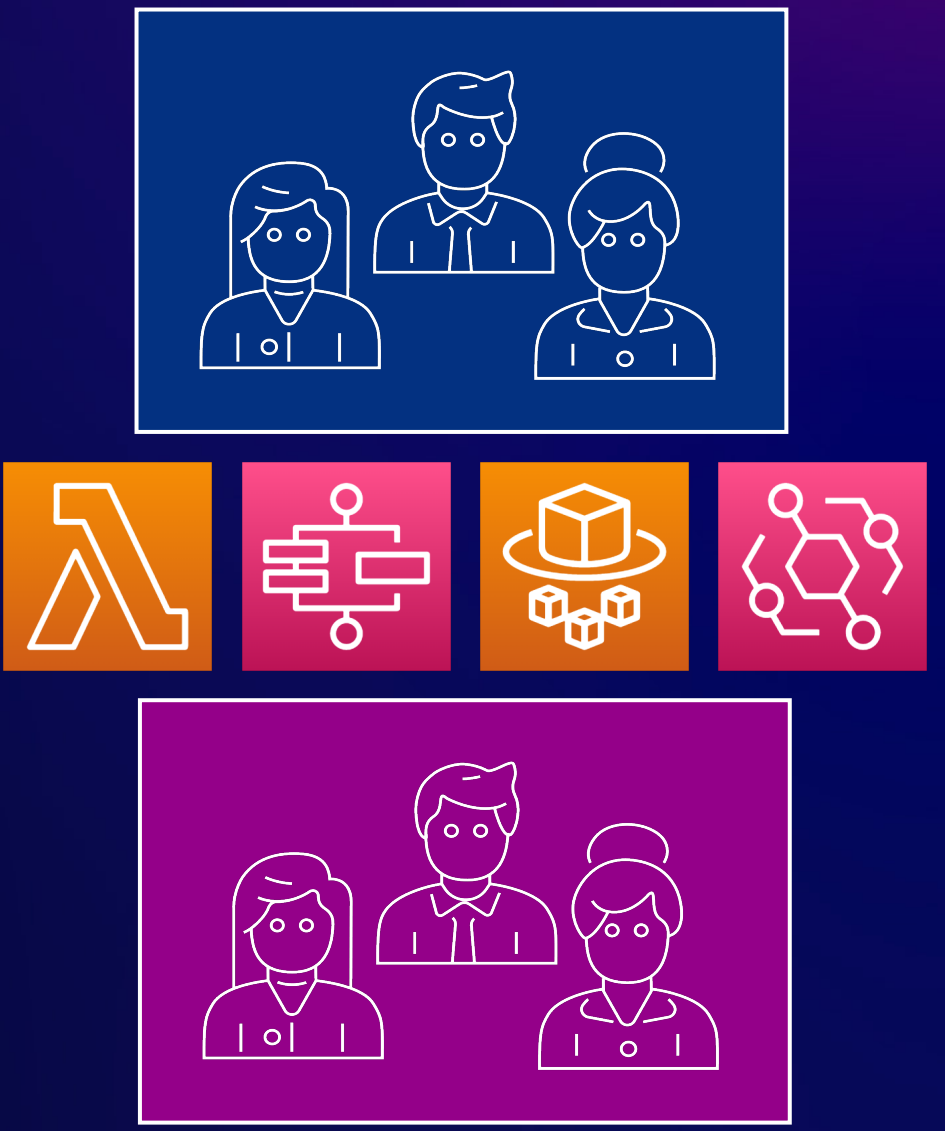
Building a Serverless Developer Platform in practice

Build for your adopters
with your adopters

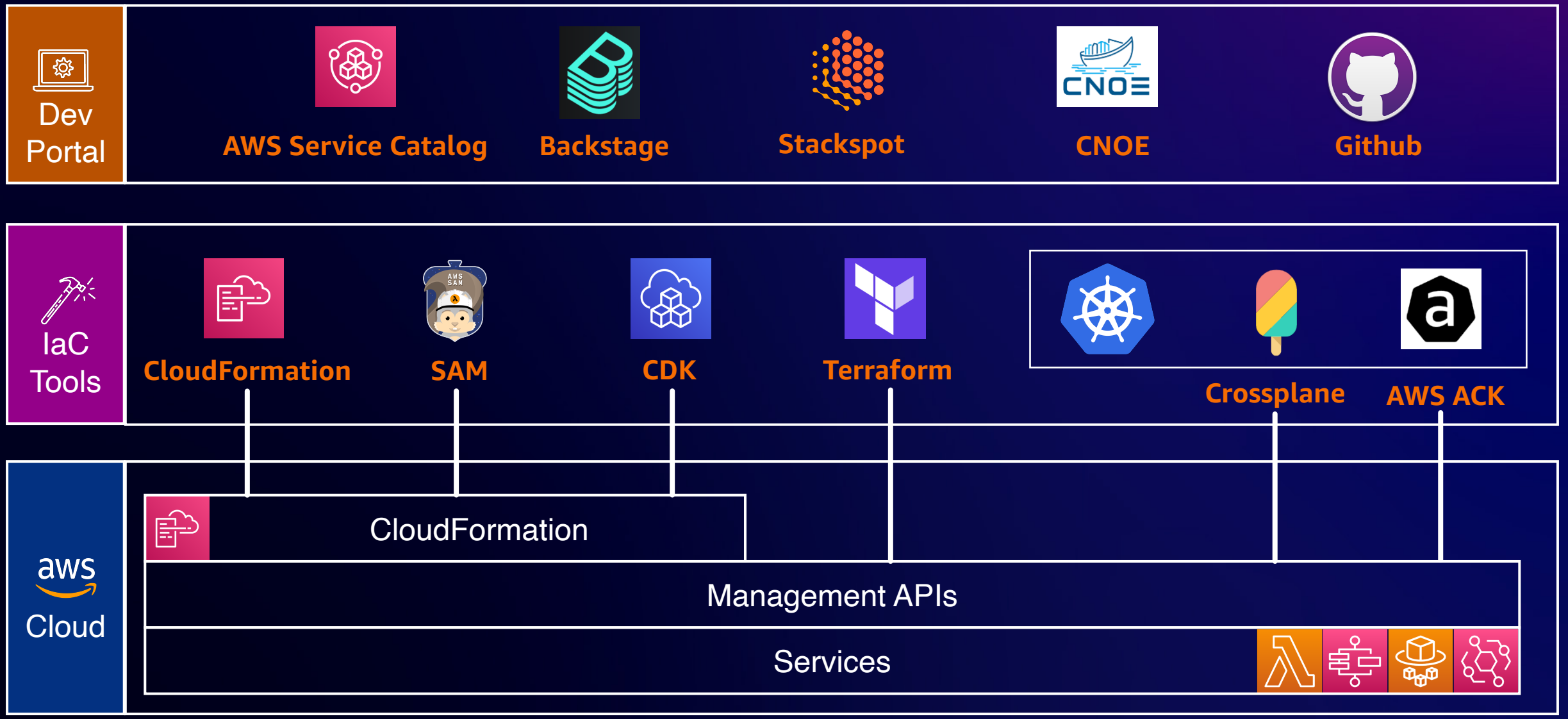


Building a Serverless Developer Platform in practice

Define infrastructure and application resources ownership model, tools, processes

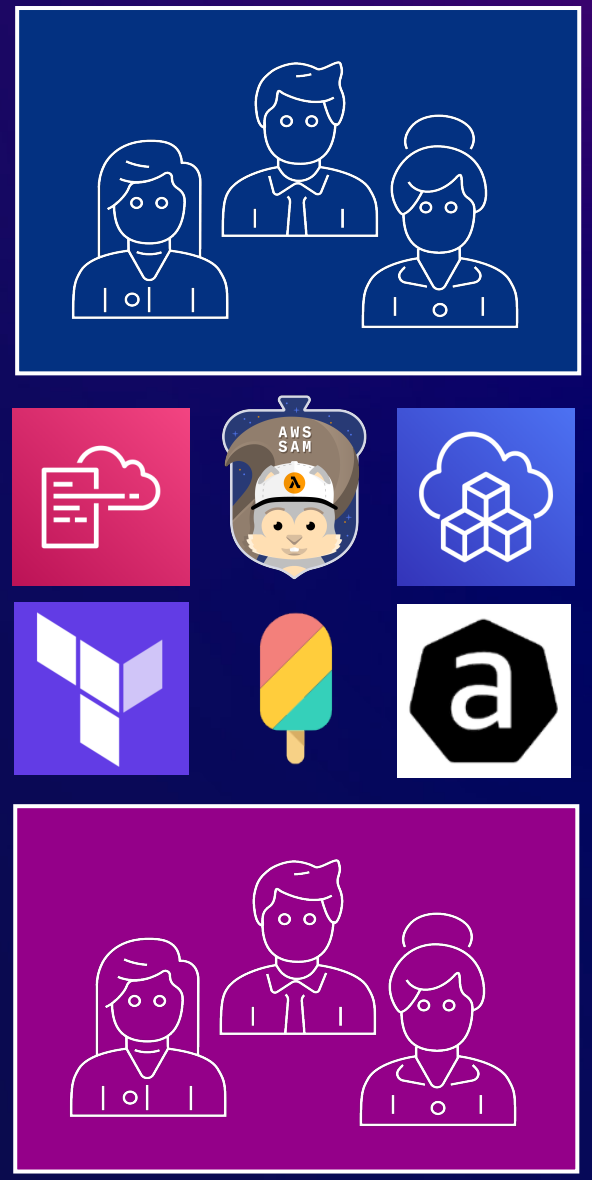


Internal Developer Platform



Building a Serverless Developer Platform in practice

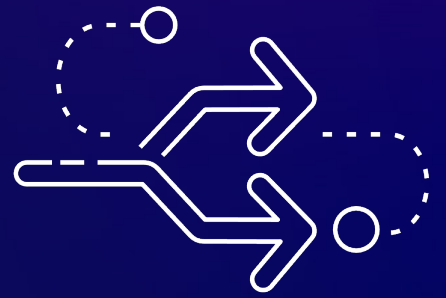
Build a catalog of vetted IaC modules
(aka constructs, aka templates)



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)

Cloud resources configuration (main.tf)

```
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)

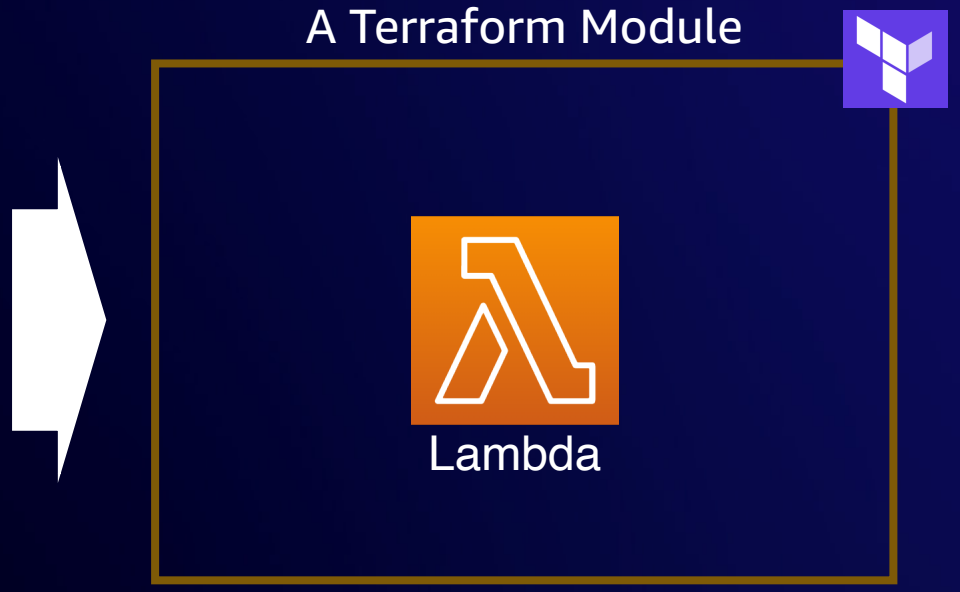
```
resource "aws_lambda_function" "lambda_function" {
  function_name    = var.function_name
  runtime          = var.runtime
  logging_config {
    log_group = aws_cloudwatch_log_group.this.arn
  }
}

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

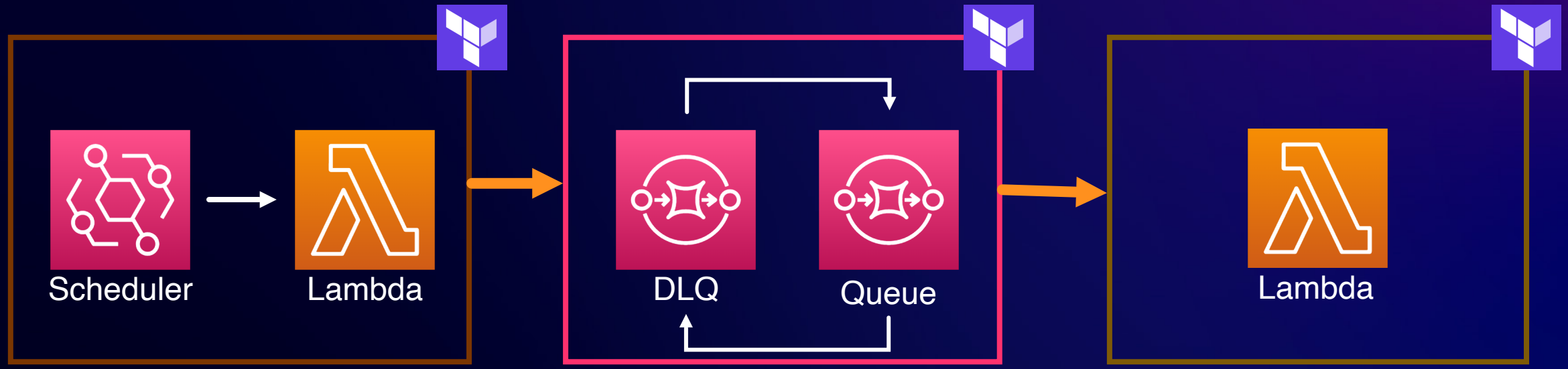
function_name
memory_size
source_path
runtime
handler
attach_vpc
log_retention_days
enable_furl
etc…



Outputs

function_arn
log_group_arn
function_url

Reusability and Composability

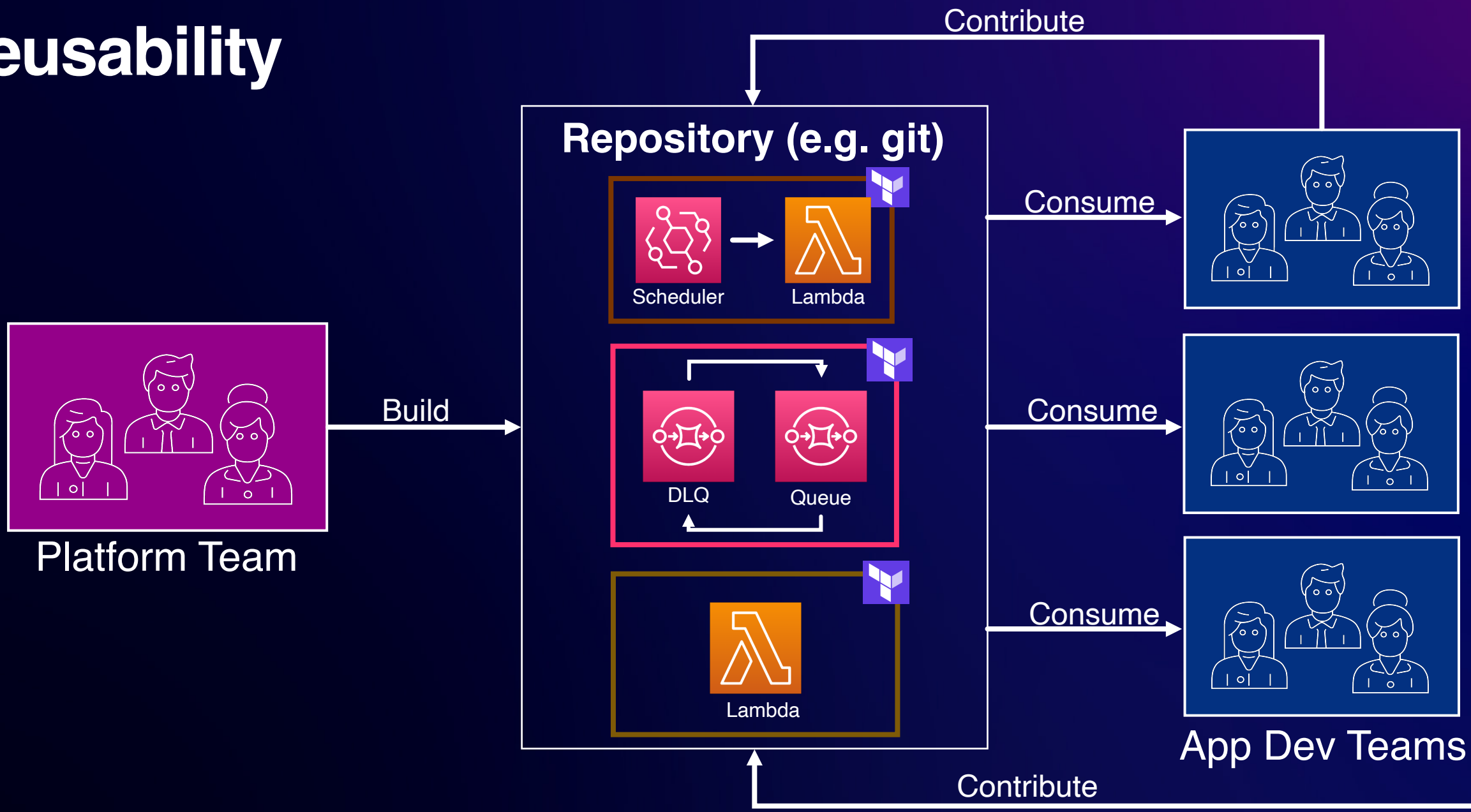


Lambda function with periodical scheduler

SQS Queue with redrive

Baseline Lambda function

Reusability



Serverless.tf – community project

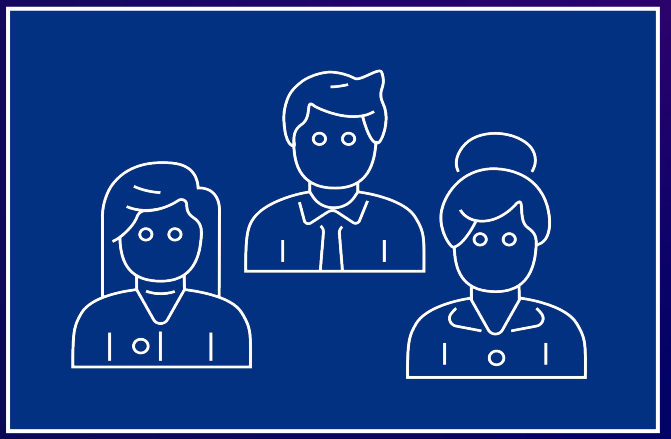
An opinionated, 100% open-source community framework for developing, building, deploying, and securing serverless applications on AWS using Terraform.

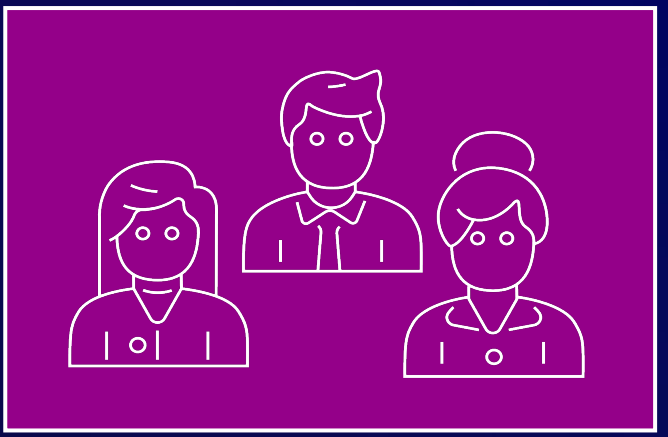
Over **120 million** downloads

https://serverless.tf

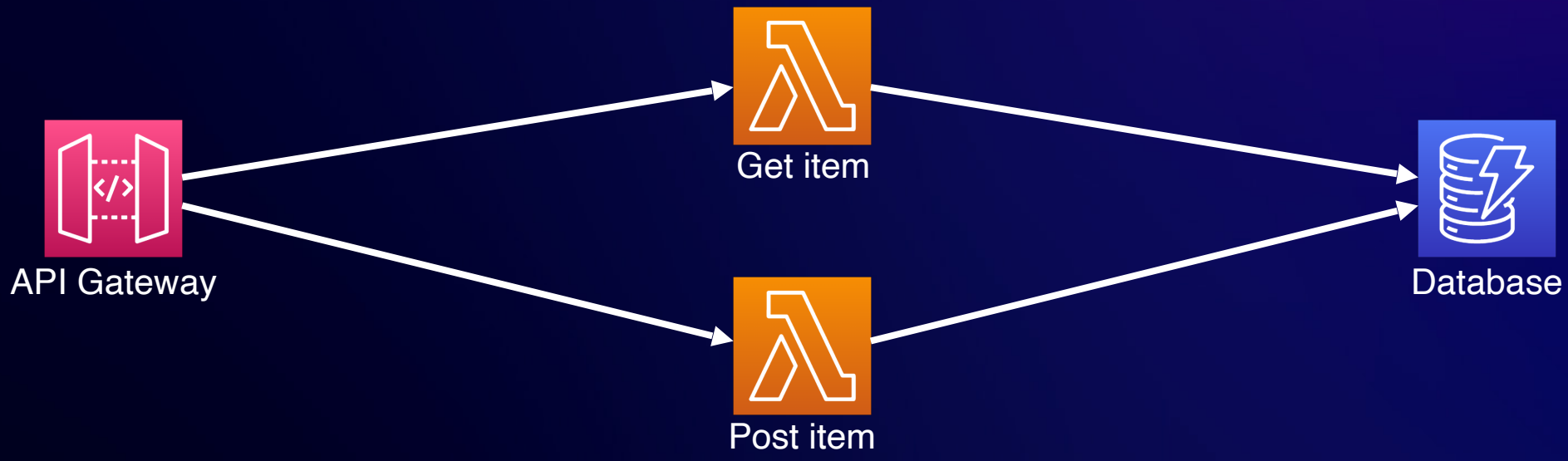
Building a Serverless Developer Platform in practice

Build and evolve architectural blueprints

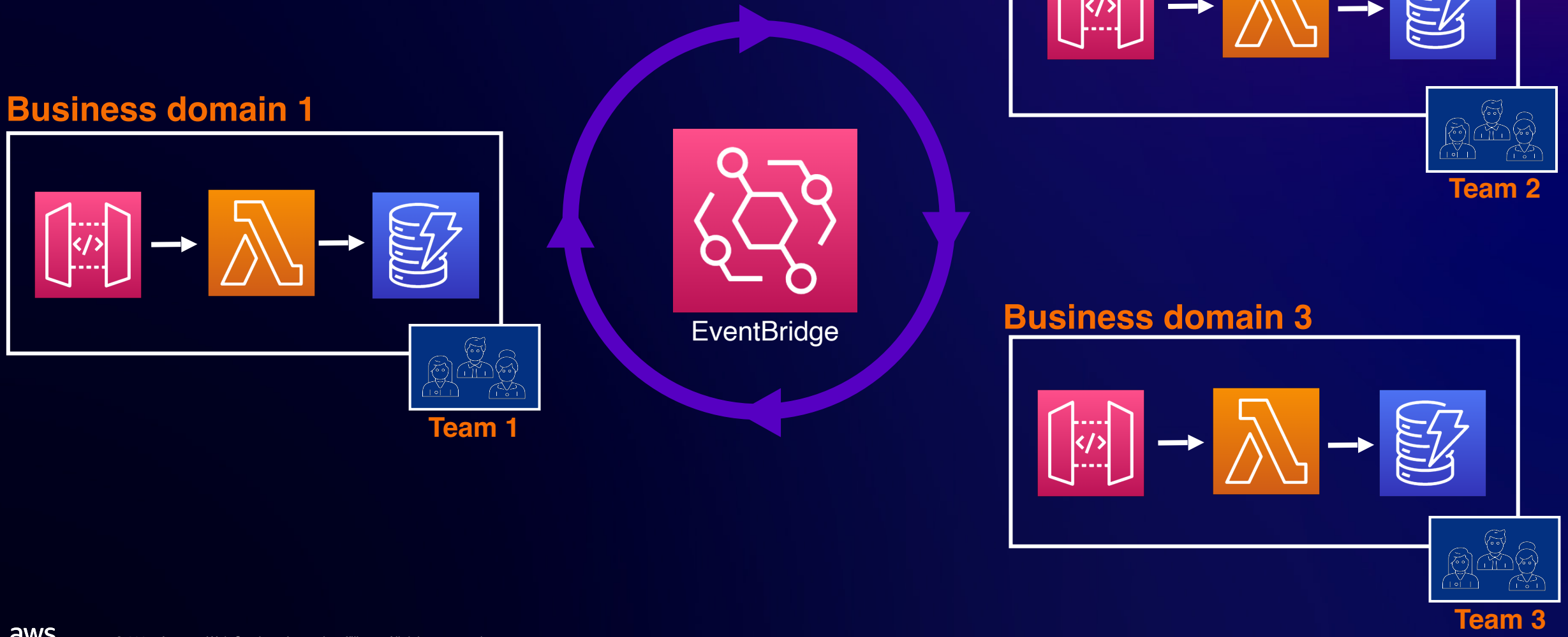




Architectural pattern



Usage trend

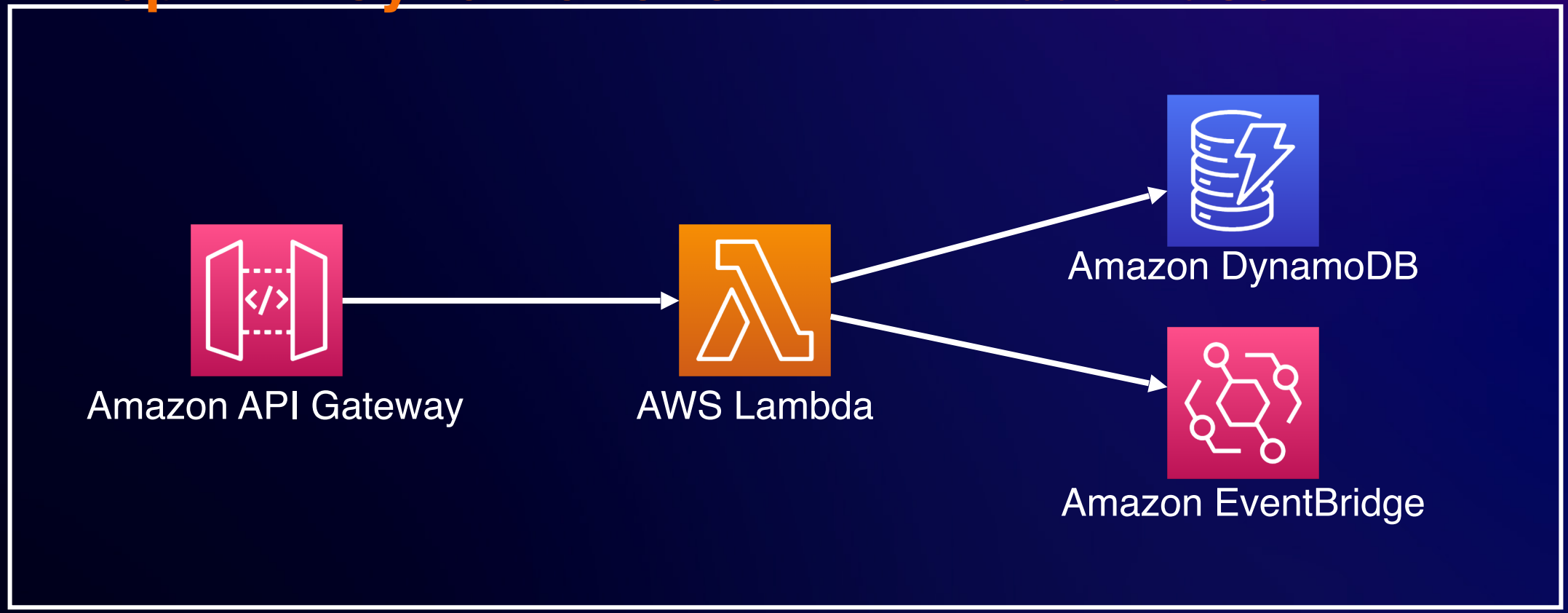


Bringing it all together

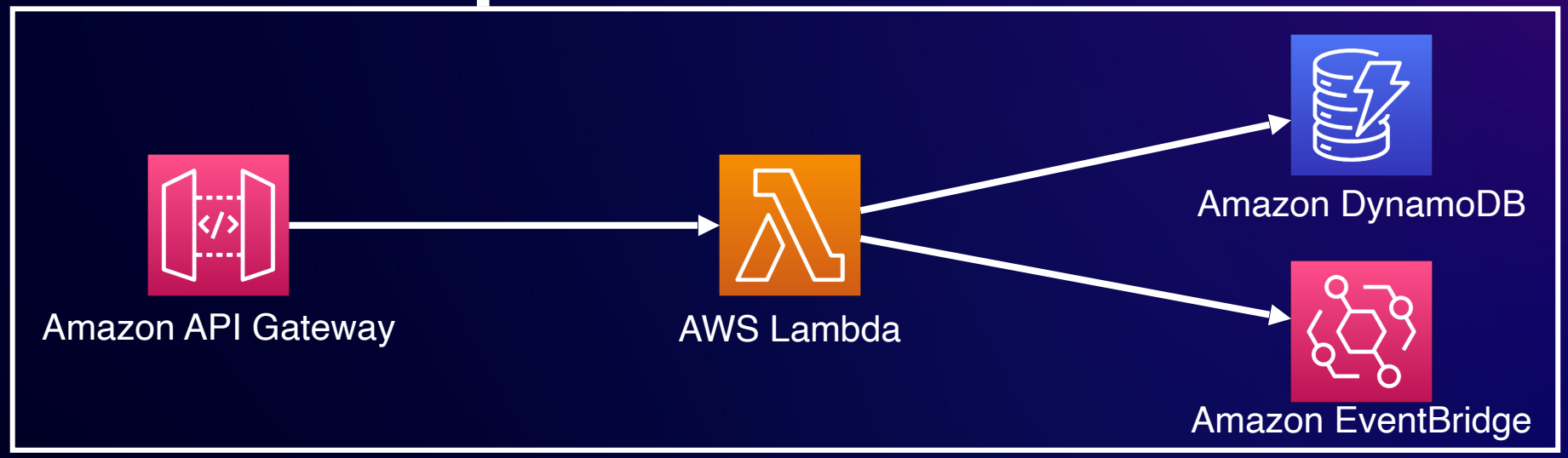
Arrows are as important as blocks

A Serverless Blueprint

Blueprint - Synchronous API with database



A 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 Blueprints – 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 Blueprints – 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 Blueprints – 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

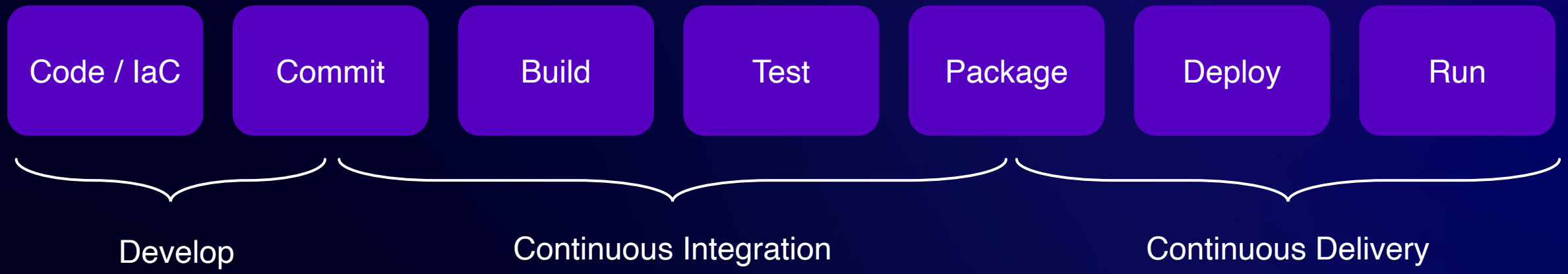
Serverless Blueprints – Lambda + DynamoDB

```
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

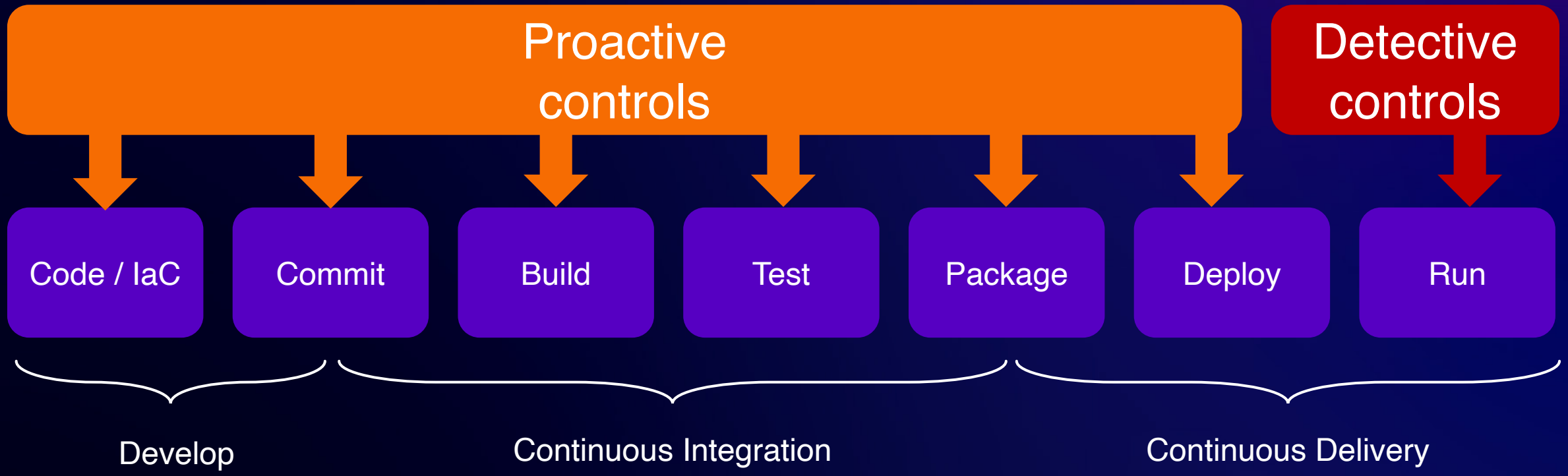
But what if developers want to make changes to the blueprint / pattern!?

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)

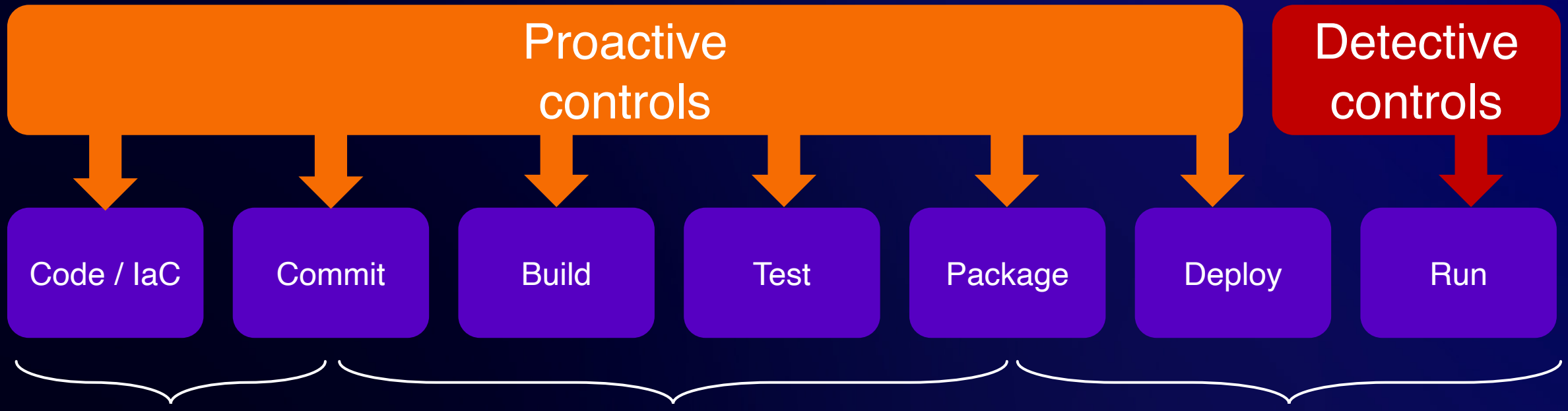
Amazon Inspector (code scanning)

AWS CDK (cdk-nag)

AWS Organizations (proactive/detective)

AWS Control Tower (multi-account environments)

HashiCorp Sentinel Policies



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

Proactive governance controls

Checkov

AWS CloudFormation Guard

CDK-NAG

HashiCorp Sentinel

✅Terraform ✅CDK ✅CloudFormation

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

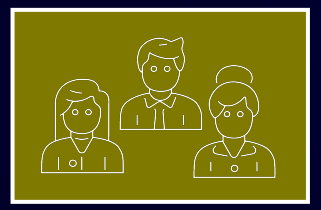
Proactive governance controls

checkov
BY PRISMA® CLOUD

```yaml
---
metadata:
  name: "Function should use organizationally approved runtimes"
  id: "ACMECORP_0002"
  category: "GENERAL_SECURITY"
definition:
  or:
    - cond_type: "attribute"
      resource_types:
        - "aws_lambda_function"
      attribute: "runtime"
      operator: "subset"
      value:
        - "nodejs20.x"
        - "nodejs18.x"
        - "python3.12"
        - "python3.11"
```

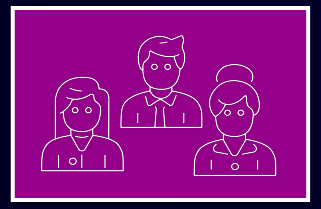
✅Terraform ✅CDK ✅CloudFormation

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

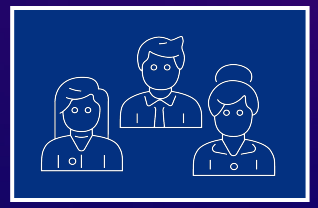
Distributed S3 data processing

Other common pattern…

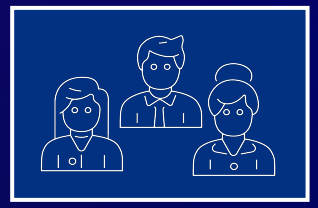
Contribute

Consume

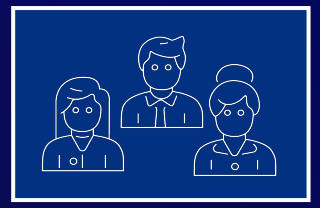
Contribute



Application development team

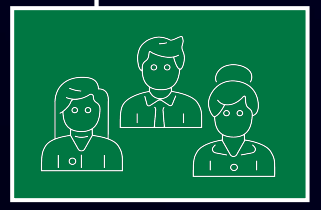


Application development team



Application development team

Compliance team



Feedback

Customer example

"Implementing serverless blueprints, automation, and platform engineering practices allowed us to save up to 5 months of development time for each new service we build"

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

Serverless Blueprints

AWS Lambda

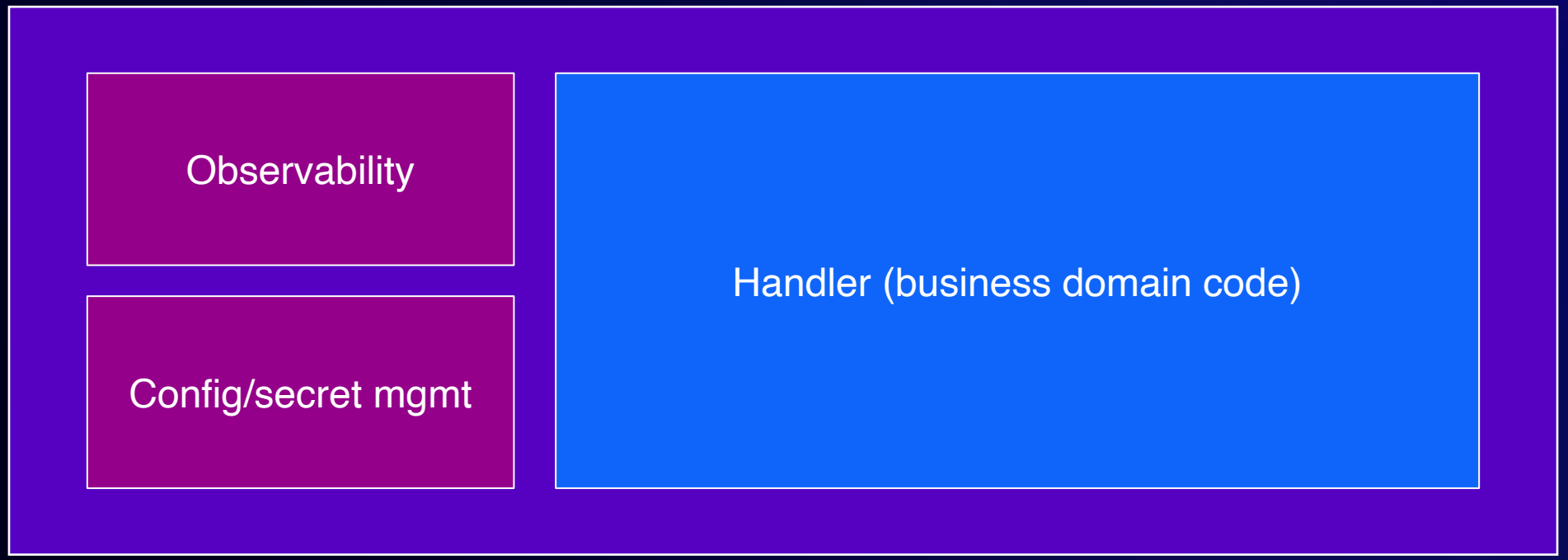
Handler (business domain code)

Serverless Blueprints

AWS Lambda

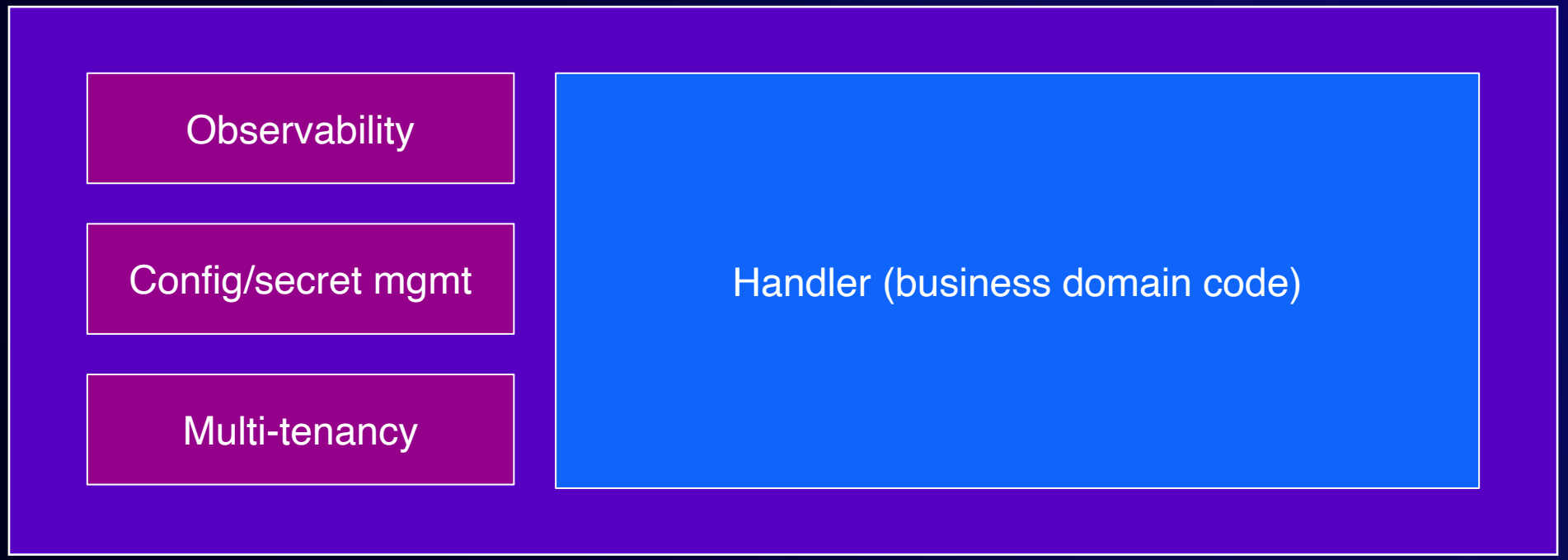
Serverless Blueprints

AWS Lambda



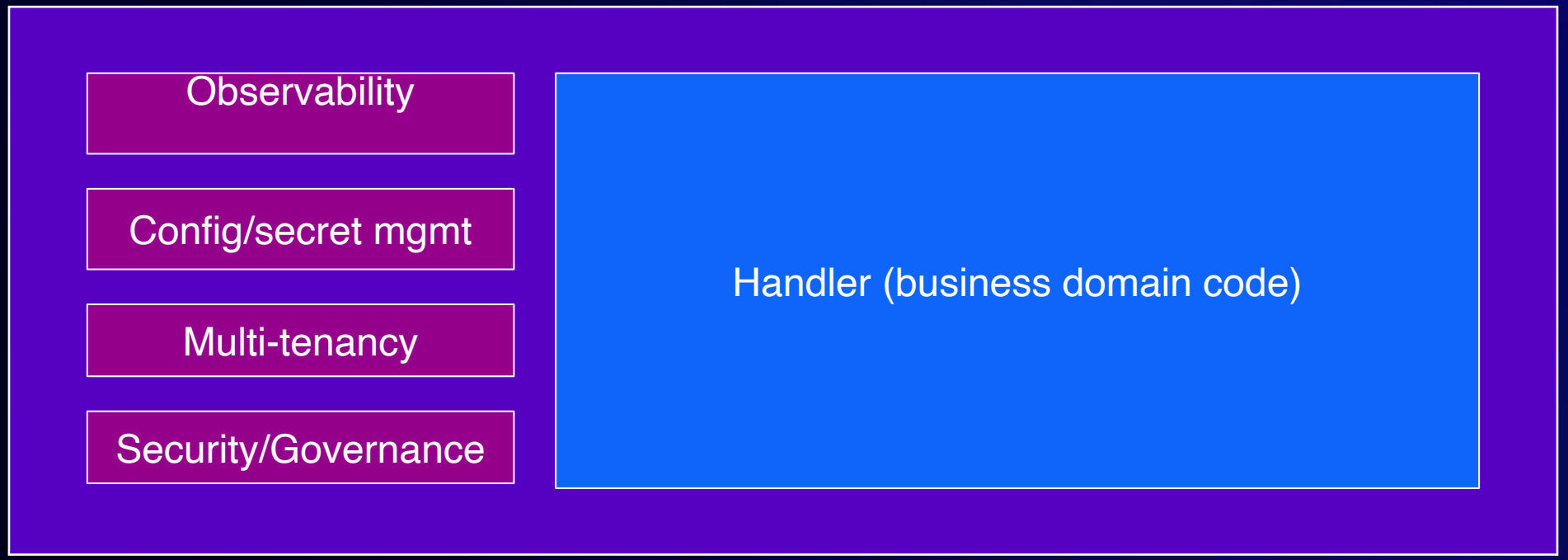
Serverless Blueprints

AWS Lambda



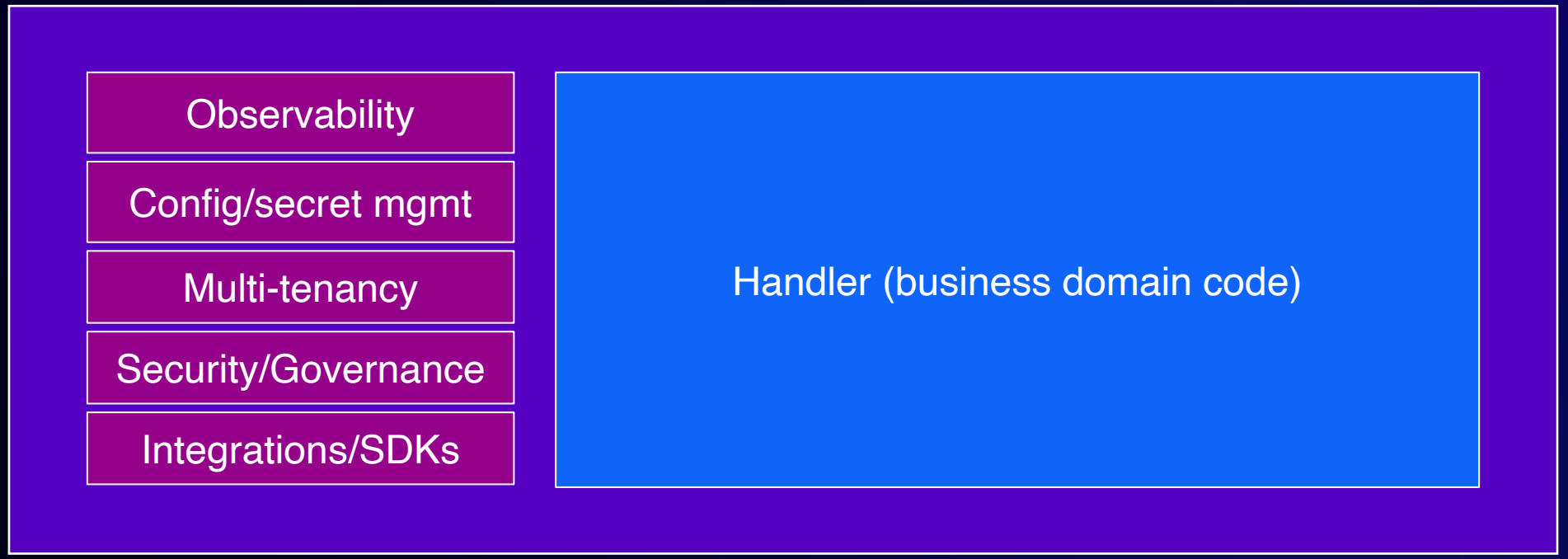
Serverless Blueprints

AWS Lambda



Serverless Blueprints

AWS Lambda



Serverless Blueprints

```python
@init_environment_variables(model=MyHandlerEnvVars)
@logger.inject_lambda_context(correlation_id_path=correlation_paths.API_GATEWAY_REST)
@metrics.log_metrics
@tracer.capture_lambda_handler(capture_response=False)
def lambda_handler(event: dict[str, Any], context: LambdaContext) -> dict[str, Any]:
    return app.resolve(event, context)
```

Serverless Blueprints

```python
@init_environment_variables(model=MyHandlerEnvVars)
@logger.inject_lambda_context(correlation_id_path=correlation_paths.API_GATEWAY_REST)
@metrics.log_metrics
@tracer.capture_lambda_handler(capture_response=False)
def lambda_handler(event: dict[str, Any], context: LambdaContext) -> dict[str, Any]:
    return app.resolve(event, context)
```

Serverless Blueprints

```python
def handle_create_order(create_input: Annotated[CreateOrderRequest, Body(embed=False, media_t
    env_vars: MyHandlerEnvVars = get_environment_variables(model=MyHandlerEnvVars)
    logger.debug('environment variables', env_vars=env_vars.model_dump())
    logger.info('got create order request', order=create_input.model_dump())

    my_configuration = parse_configuration(model=MyConfiguration)
    logger.debug('fetched dynamic configuration', configuration=my_configuration.model_dump())

    metrics.add_metric(name='ValidCreateOrderEvents', unit=MetricUnit.Count, value=1)
    response: CreateOrderOutput = create_order(
        order_request=create_input,
        table_name=env_vars.TABLE_NAME,
        context=app.lambda_context,
    )

    logger.info('finished handling create order request')
    return response
```

In conclusion


Build with your customers


Don’t boil the ocean


One size doesn’t fit all workloads


Allow for customizations


Documentation and education are key

Next steps

Next steps

820+ serverless patterns
on Serverlessland.com

Serverless patterns
for CDK

Open-source serverless application blueprint

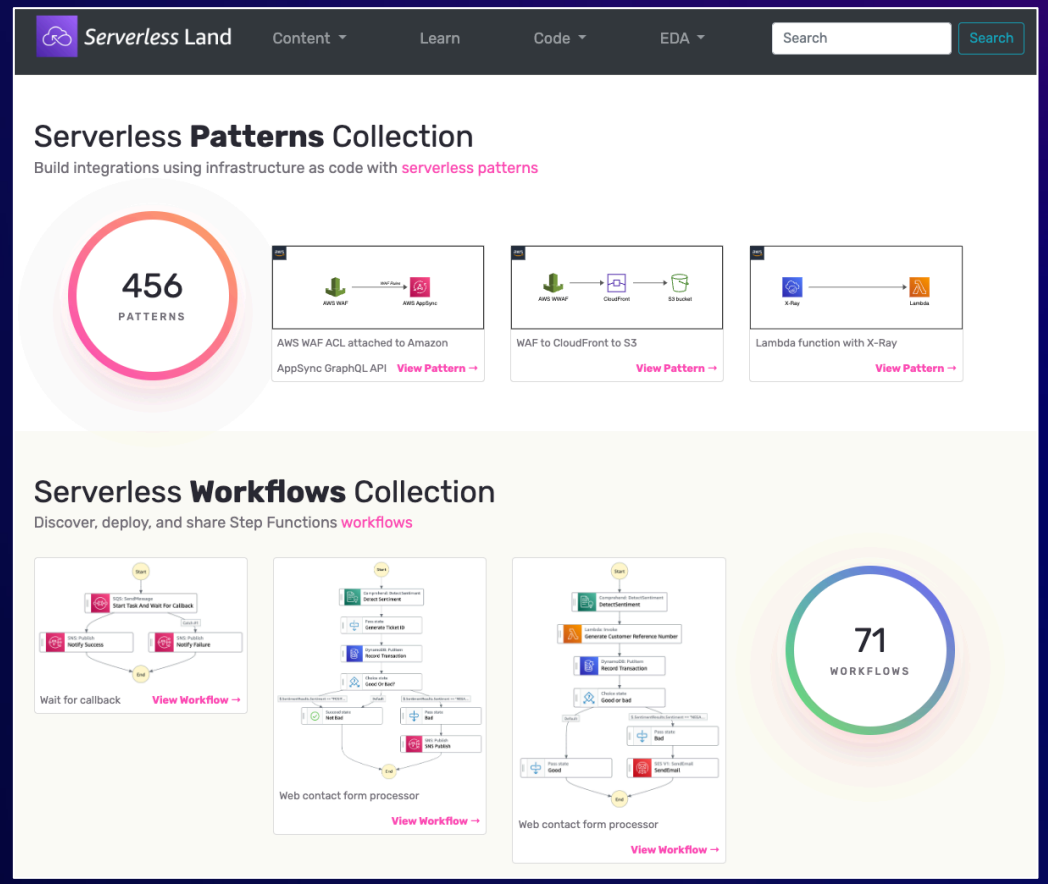
Building Serverless Applications with Terraform

Serverless.tf

Governance for Serverless Applications

Accelerate your Serverless Learning

- 450+ patterns
- 60+ guides
- 70+ workflows
- 300+ posts
- 190+ videos
- Weekly office hours



https://serverlessland.com

Thank you!

Anton Aleksandrov
Principal Solution Architect, Serverless
AWS