AWS re:Invent

SVS336-R

CI/CD for serverless applications

Sanyogita Ranade
Senior MTS
Salesforce

Benjamin Susman
Lead MTS
Salesforce

Eric Johnson
Senior Developer Advocate
Amazon Web Services

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

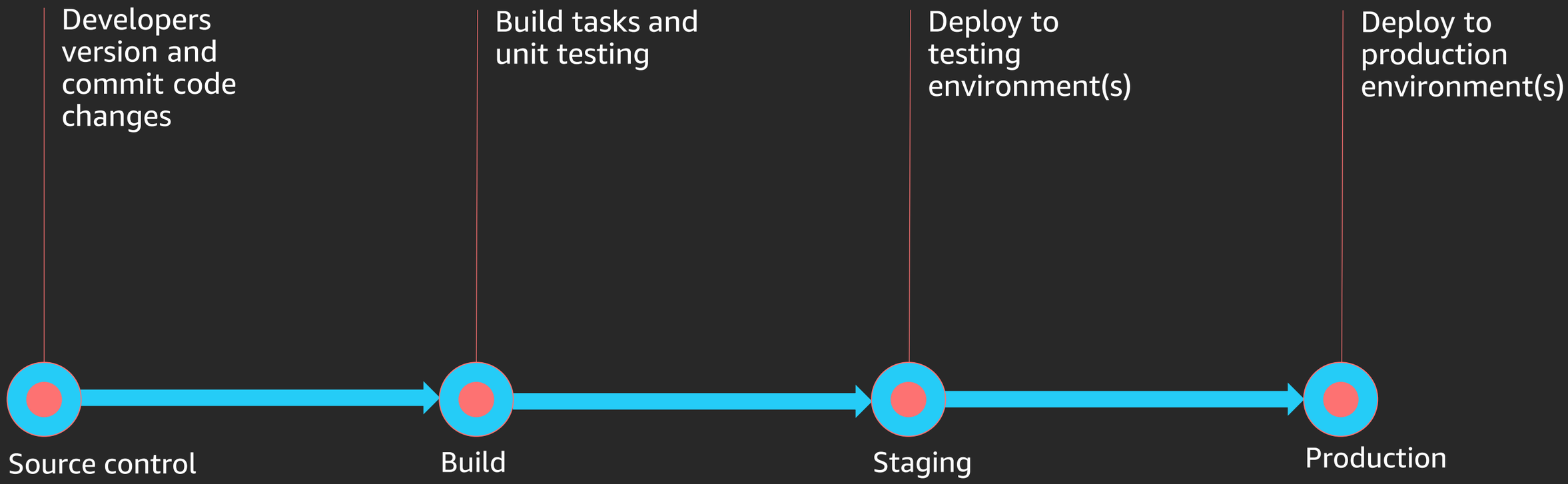
What is CI/CD and why is it important?

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

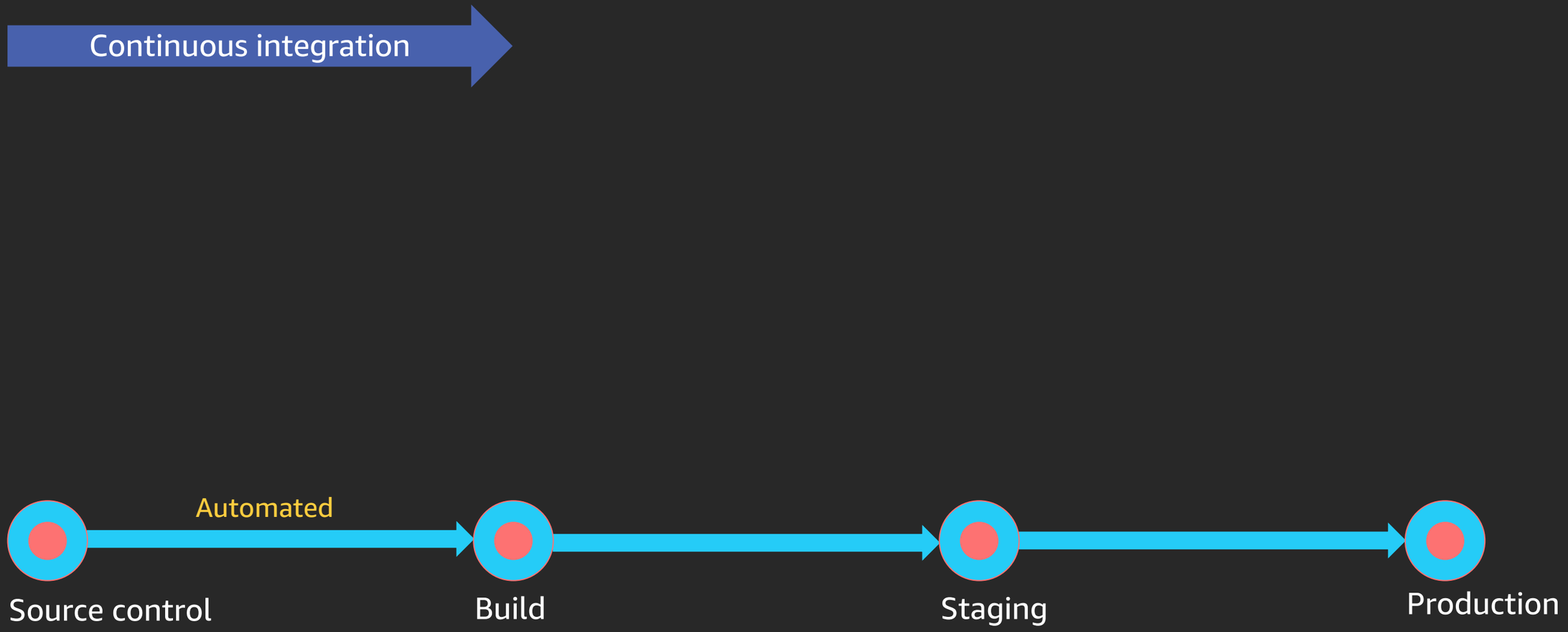
What is CI/CD?

CI: Continuous integration
CD: Continuous delivery or
CD: Continuous deployment

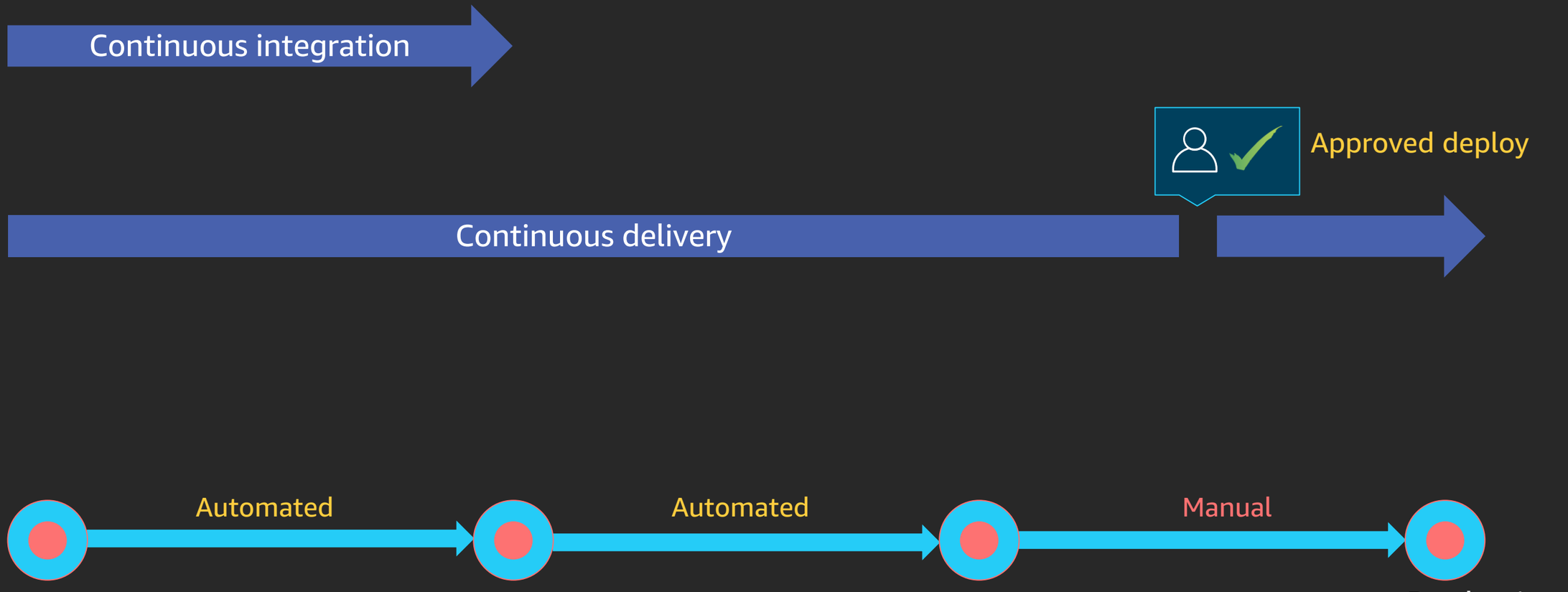
The pipeline



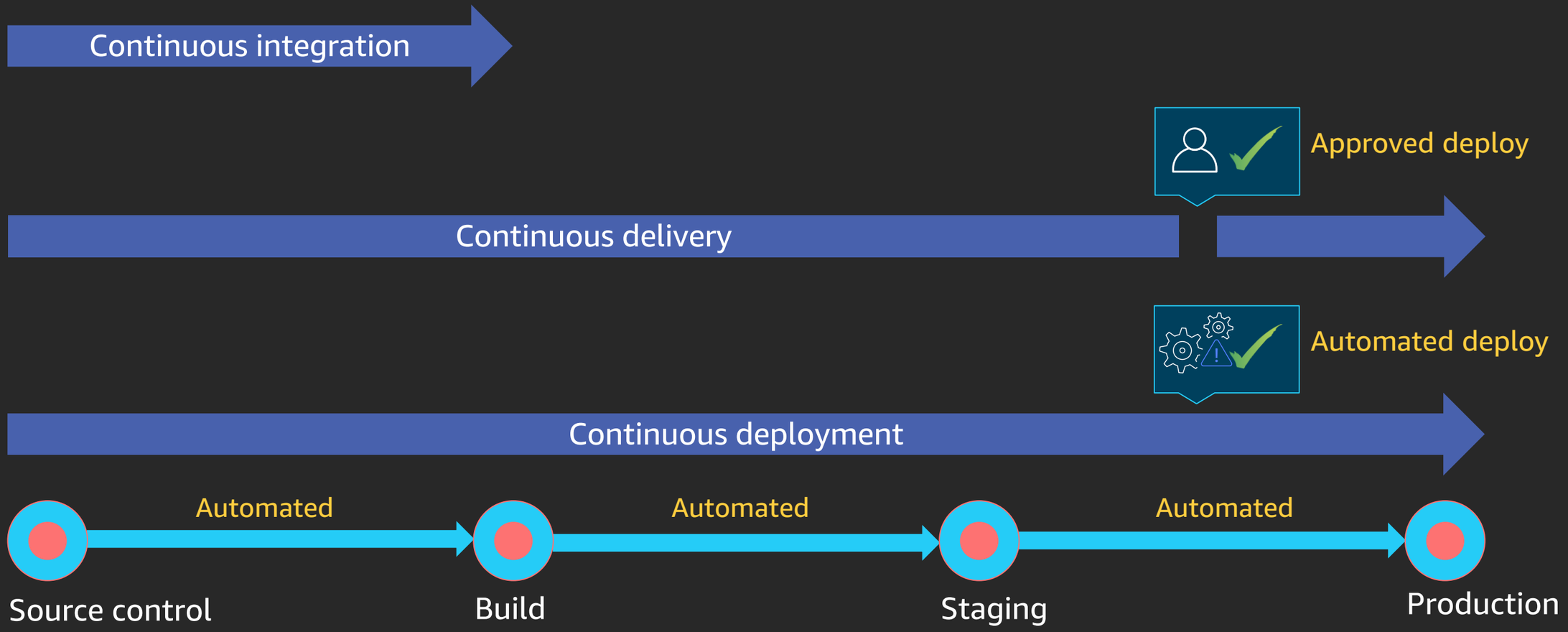
The pipeline: Continuous integration



The pipeline: Continuous delivery



The pipeline: Continuous deployment



The right tools

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Meet AWS SAM

- AWS Serverless Application Model
- Can mix in other traditional AWS CloudFormation resources in the same template (i.e., Amazon S3, Amazon Kinesis, AWS Step Functions)
- Supports use of parameters, mappings, outputs, etc.
- Supports intrinsic functions (i.e., Ref, Sub, Join, Select, Split)
- Can use ImportValue (exceptions for RestApiId, Policies, StageName attributes)
- YAML or JSON



AWS SAM templates

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs10.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

Just 20 lines to create:

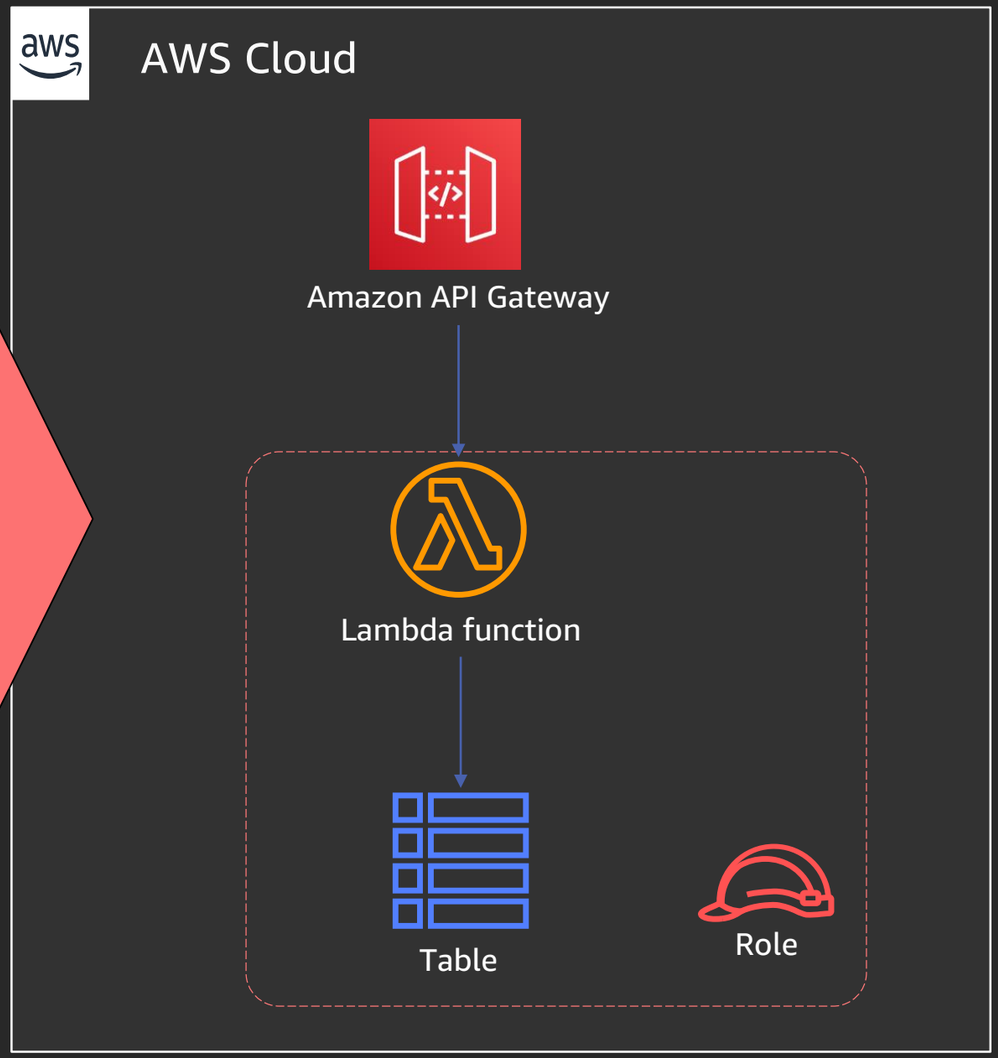
- Lambda function
- IAM role
- API gateway
- DynamoDB table

AWS SAM templates

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs10.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

Allowing this ← ===

To become this →



Code repository

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS CodeCommit

- Fully managed source-control service that hosts secure Git-based repositories
- Allows teams to collaborate on code in a secure and highly scalable ecosystem
- Automatically encrypts your files in transit and at rest
- Integrated with AWS Identity and Access Management (IAM)
- https://aws.amazon.com/codecommit/



Third-party code repositories

GitHub

Atlassian Bitbucket

GitLab

git private repo



Integrates with AWS CodeBuild and AWS CodePipeline



Integrates with AWS CodeBuild

AWS CodePipeline: The orchestrator

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS CodePipeline

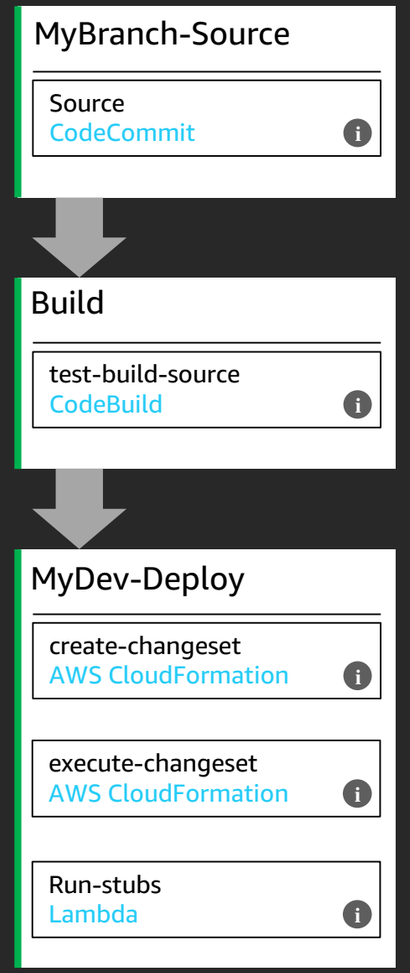
- Continuous delivery service for fast and reliable application updates
- Model and visualize your software release process
- Builds, tests, and deploys your code every time there is a code change
- Integrates with third-party tools and AWS
- https://aws.amazon.com/codepipeline/



Example of minimal developer's pipeline

This pipeline:

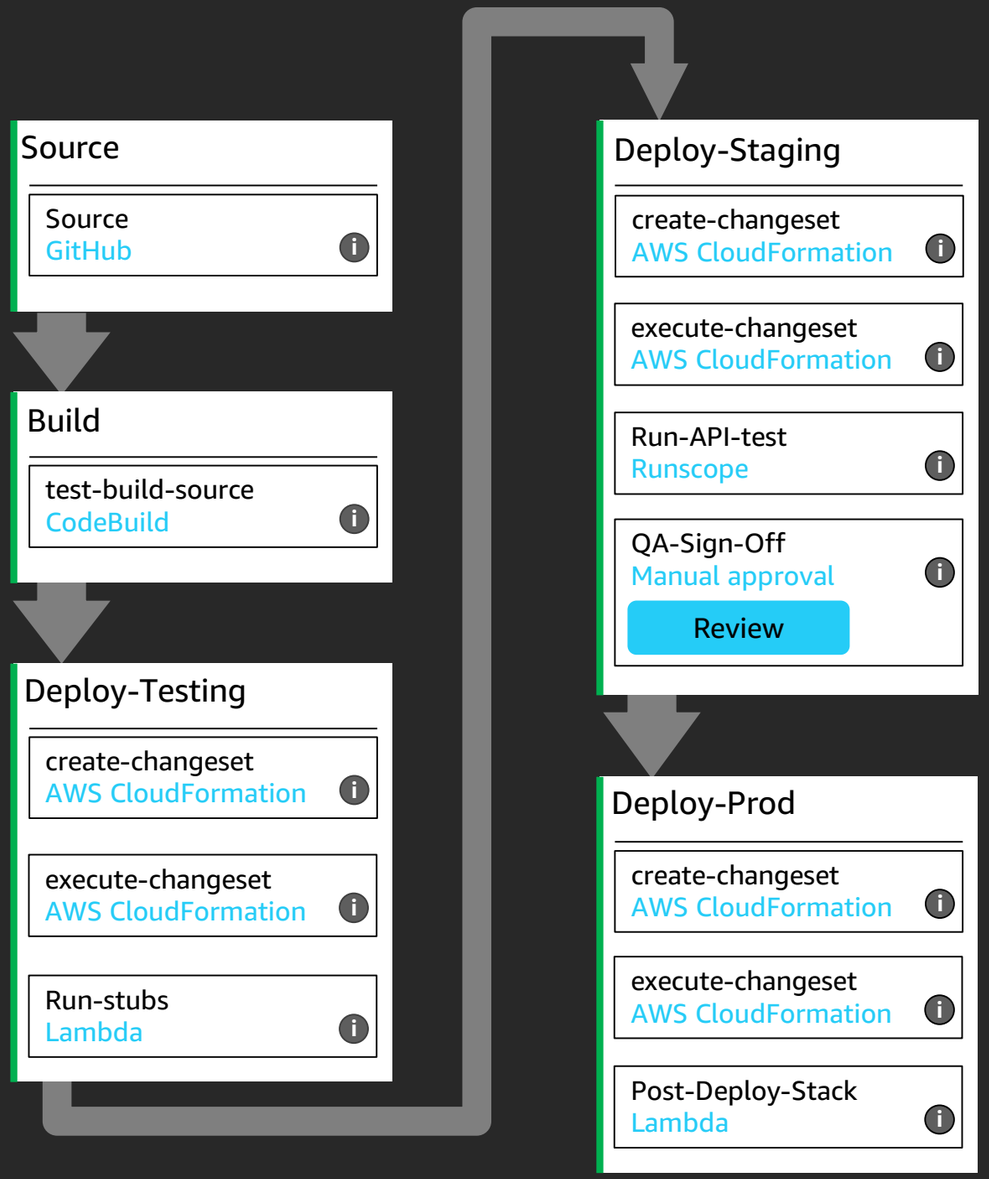
- Three stages
- Builds code artifact
- One development environment
- Uses AWS SAM/AWS CloudFormation to deploy artifact and other AWS resources
- Has Lambda custom actions for testing functions



Example of full pipeline

This pipeline:

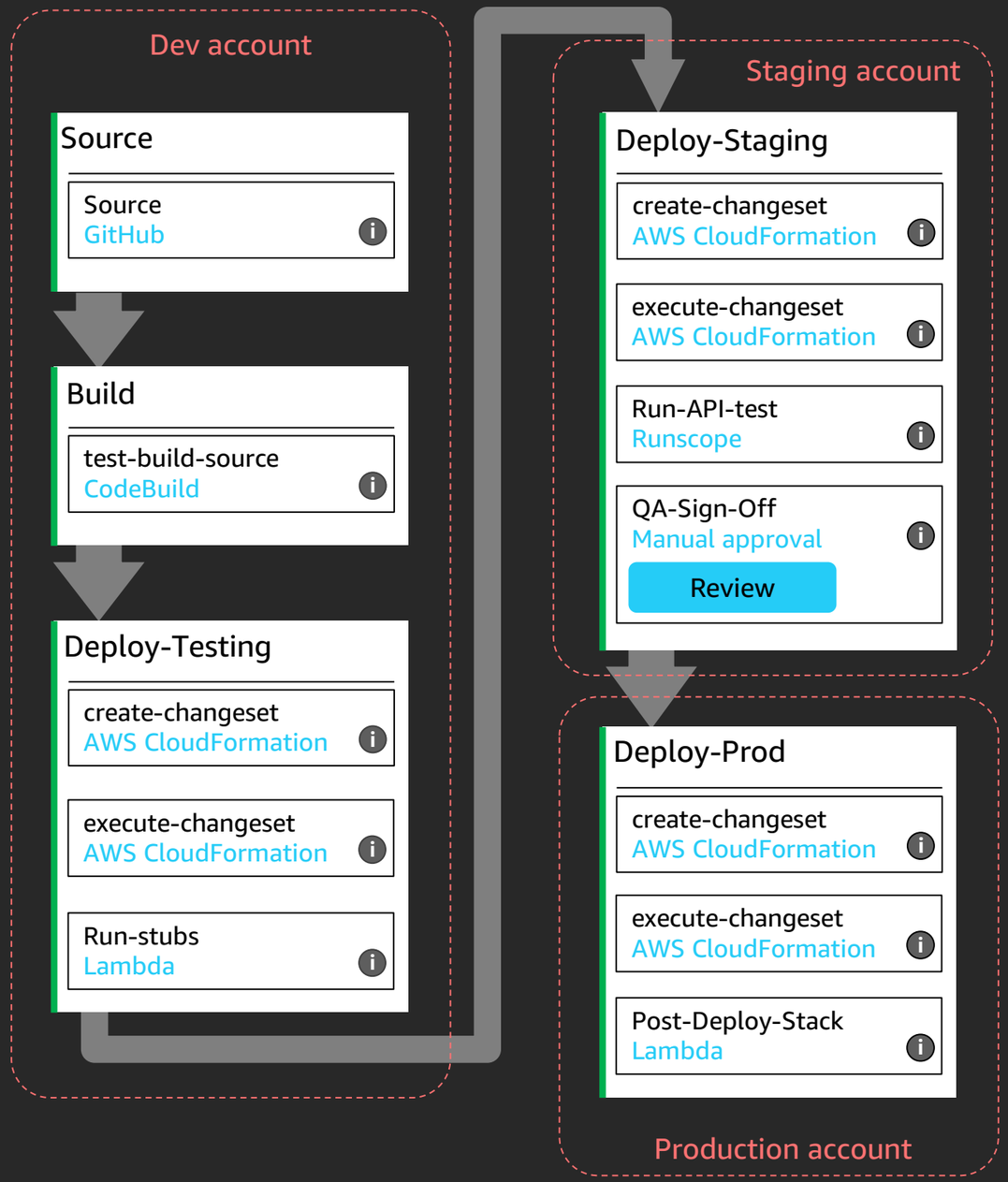
- Five stages
- Builds code artifact
- Three deployed to “environments”
- Uses AWS SAM/AWS CloudFormation to deploy artifact and other AWS resources
- Has Lambda custom actions for testing functions
- Integrates with a third-party tool/service
- Has a manual approval before deploying to production



Example of full pipeline

This pipeline:

- Five stages
- Builds code artifact
- Three deployed to “environments”
- Uses AWS SAM/AWS CloudFormation to deploy artifact and other AWS resources
- Has Lambda custom actions for testing functions
- Integrates with a third-party tool/service
- Has a manual approval before deploying to production



AWS CodeBuild

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS CodeBuild

- Fully managed build service that can compile source code, run tests, and produce software packages
- Scales continuously and processes multiple builds concurrently
- Can consume environment variables from AWS Systems Manager Parameter Store
- Can run in your VPC and locally
- Supports dependency caching
- https://aws.amazon.com/codebuild/



The buildspec.yaml file

```yaml
version: 0.2
variables:
  parameter-store:
    BUCKET_NAME: /CodeBuild/BucketName

phases:
  install:
    commands:
      - pip install --upgrade pip
      - pip install pipenv -user
      - pipenv install awscli aws-sam-cli
      - npm install
  pre_build:
    commands:
      - eslint *.js
  build:
    commands:
      - sam build
  post_build:
    commands:
      - sam package -template-file template.yaml -s3-bucket $BUCKET_NAME -output-template out.yaml

artifacts:
  type: zip
  files:
    - out.yaml
```

AWS CloudFormation

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS CloudFormation

- Infrastructure as code (IaC)
- Provides a common language for you to describe and provision all the infrastructure resources in your cloud environment
- Build and rebuild your infrastructure and applications, without having to perform manual actions or write custom scripts
- https://aws.amazon.com/cloudformation/

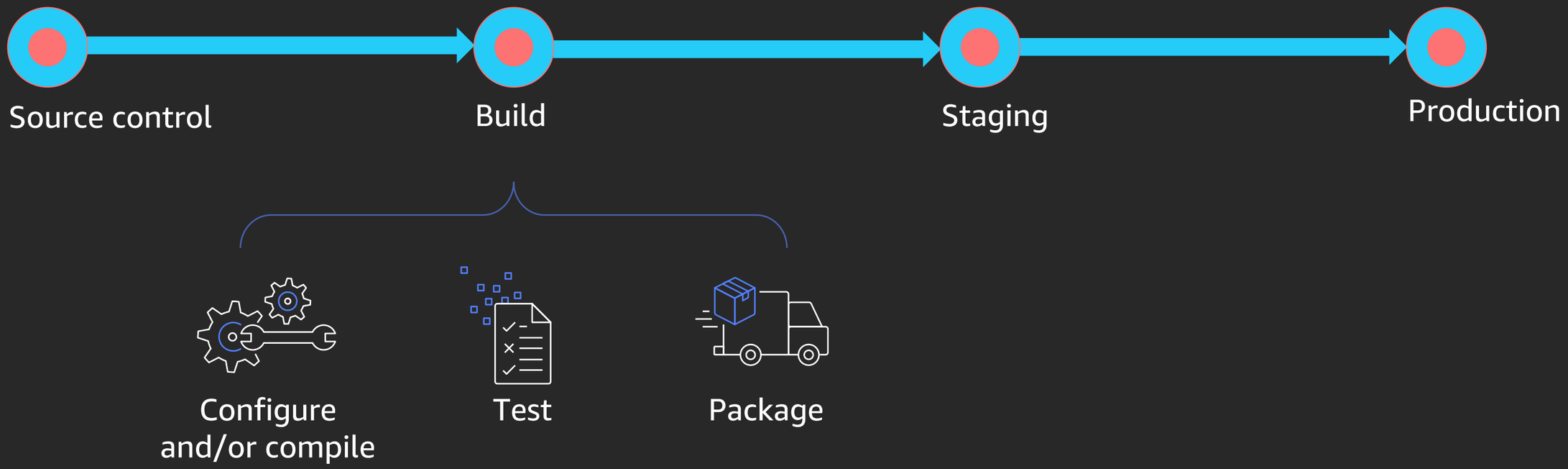


Testing

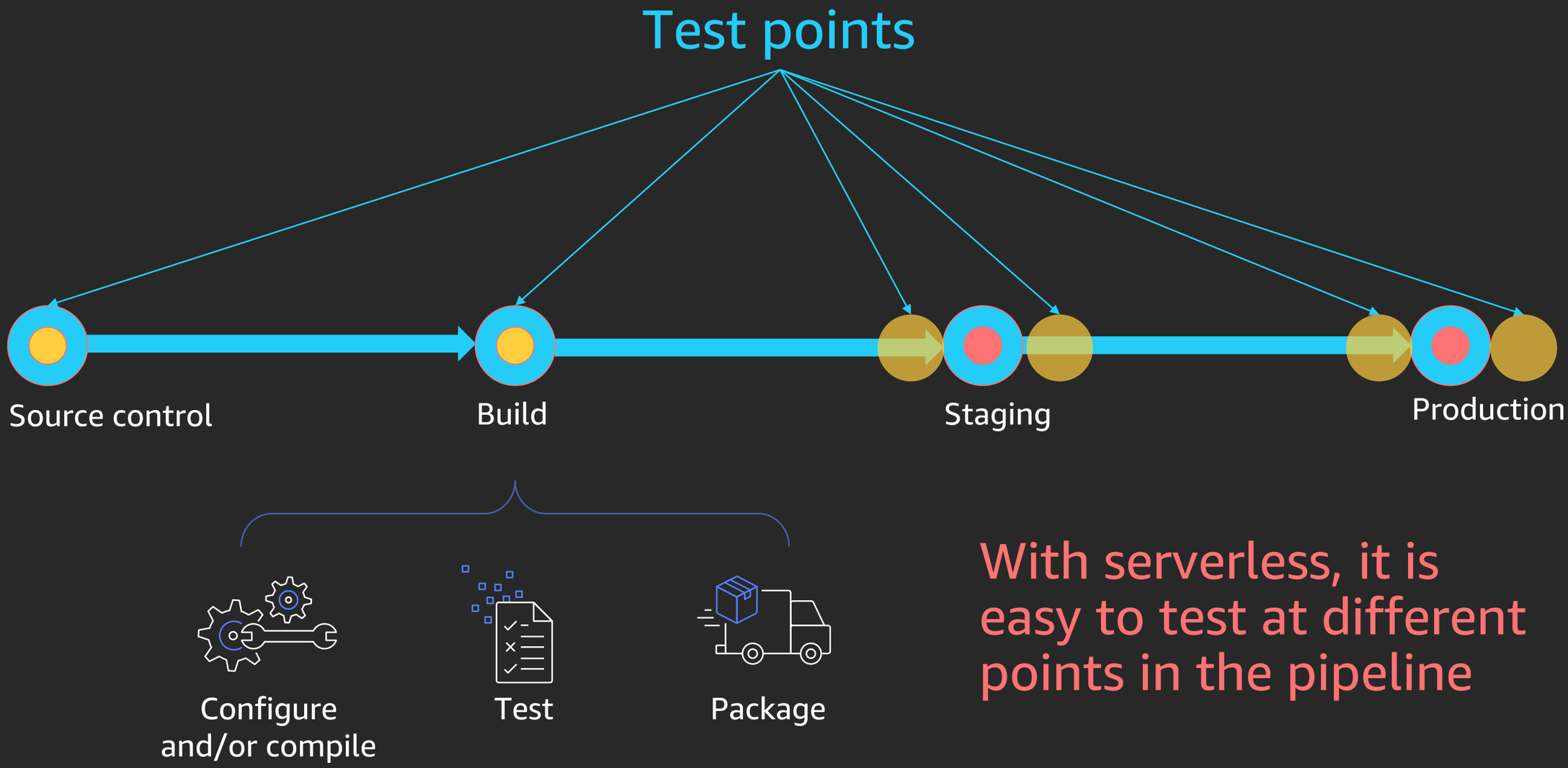
© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The pipeline: Testing

The build phase is a common place for testing



The pipeline: Testing



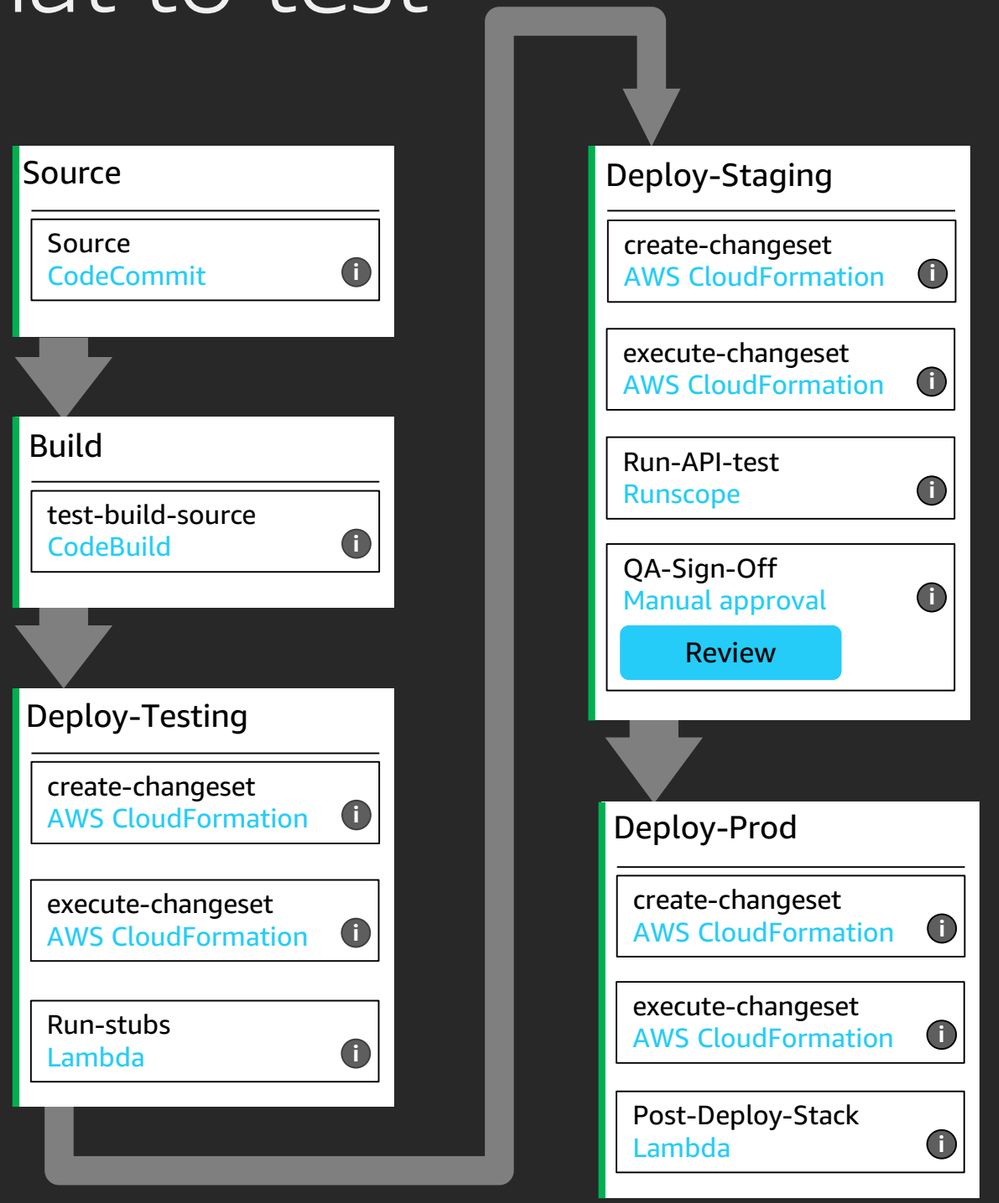
With serverless, it is easy to test at different points in the pipeline

Where and what to test

- Code review via pull request

- Lint/syntax check
- Unit test pass
- Code successfully compiles

- Application deploys successfully
- Mocked/stubbed integration tests



- Application deploys successfully
- Test against real services (potentially against production dependencies)

- Run pre-traffic Lambda tests
- Deploy canaries
- Complete wait period successfully
- Deploy 100%
- Run post-traffic Lambda tests

Testing using safe deployments

```
MyLambdaFunction:
  Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs10.x
      AutoPublishAlias: !Ref ENVIRONMENT
      DeploymentPreference:
        Type: Linear10PercentEvery10Minutes
        Alarms:
          # A list of alarms that you want to monitor
          - !Ref AliasErrorMetricGreaterThanZeroAlarm
          - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
        Hooks:
          # Validation Lambda functions that are run before & after traffic shifting
          PreTraffic: !Ref PreTrafficLambdaFunction
          PostTraffic: !Ref PostTrafficLambdaFunction
```

Testing using safe deployments

```
MyLambdaFunction:
  Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs10.x
      AutoPublishAlias: !Ref ENVIRONMENT
      DeploymentPreference:
        Type: Linear10PercentEvery10Minutes
        Alarms:
          # A list of alarms that you want to monitor
          - !Ref AliasErrorMetricGreaterThanZeroAlarm
          - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
        Hooks:
          # Validation Lambda functions that are run before & after traffic shifting
          PreTraffic: !Ref PreTrafficLambdaFunction
          PostTraffic: !Ref PostTrafficLambdaFunction
```

Lambda alias traffic shifting and AWS SAM

By adding the AutoPublishAlias property and specifying an alias name, AWS SAM will do the following:

- Detect when new code is being deployed based on changes to the Lambda function's Amazon S3 URI
- Create and publish an updated version of that function with the latest code
- Create an alias with a name you provide (unless an alias already exists) and points to the updated version of the Lambda function

Deployment preference type

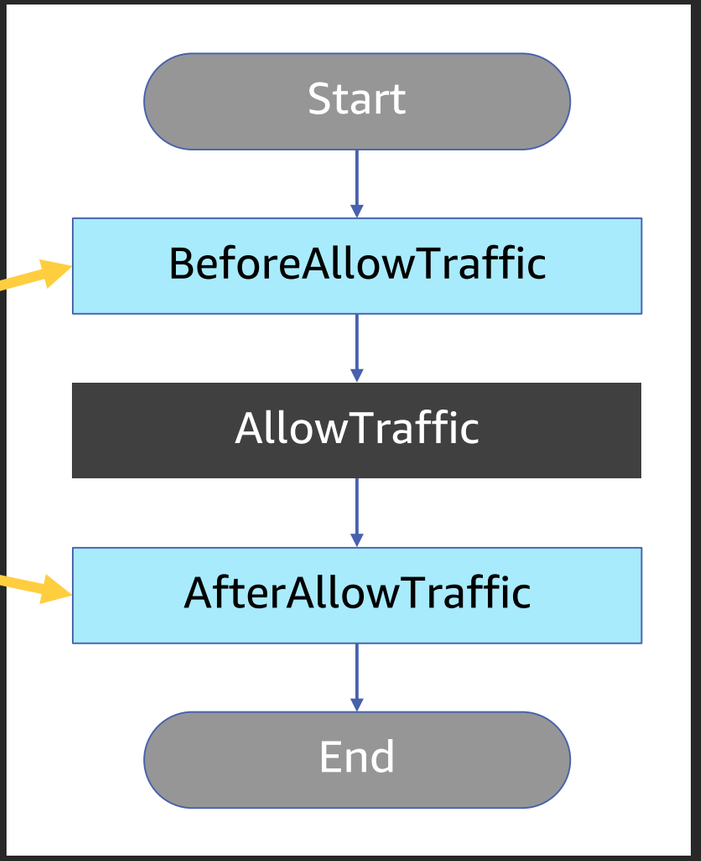
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear10PercentEvery10Minutes
Linear10PercentEvery1Minute
Linear10PercentEvery2Minutes
Linear10PercentEvery3Minutes
AllAtOnce

Lambda alias traffic shifting and AWS SAM

In AWS SAM:

```
Alarms:
  # A list of alarms that you want to monitor
  - !Ref AliasErrorMetricGreaterThanZeroAlarm
  - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
Hooks:
# Validation Lambda functions that are run before &
after traffic shifting
  PreTraffic: !Ref PreTrafficLambdaFunction
  PostTraffic: !Ref PostTrafficLambdaFunction
```

Note: You can specify a maximum of 10 alarms



Best practices

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

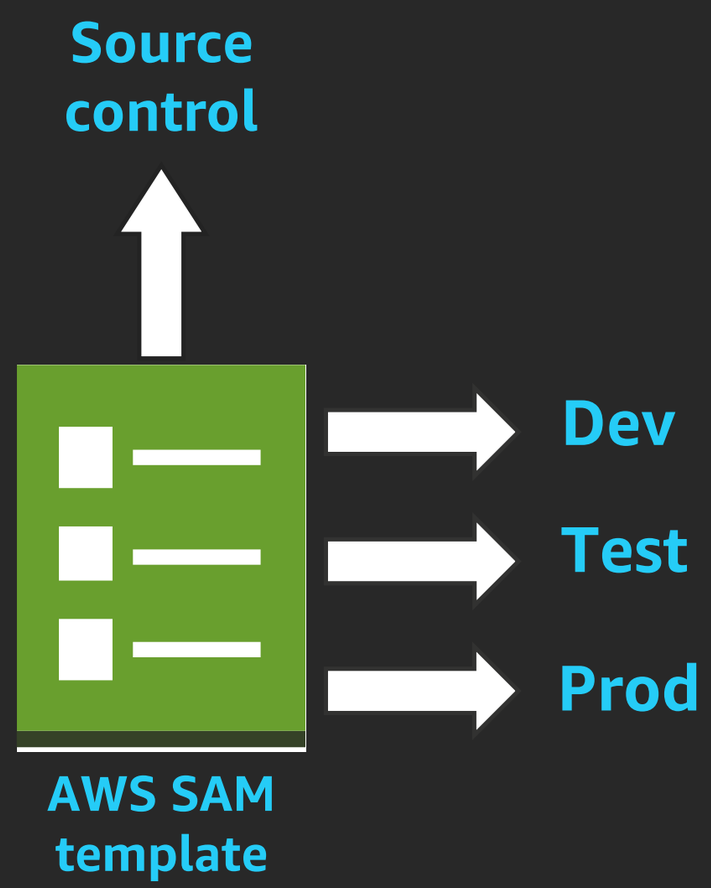
AWS SAM

- Unless function handlers share code, split them into their own independent Lambda function files or binaries.
    - Another option is to use language-specific packages to share common code between functions.
- Unless independent Lambda functions share event sources, split them into their own code repositories with their own AWS SAM templates.
- Locally lint and validate your YAML or JSON SAM files before committing them. Then do it again in your CI/CD process.

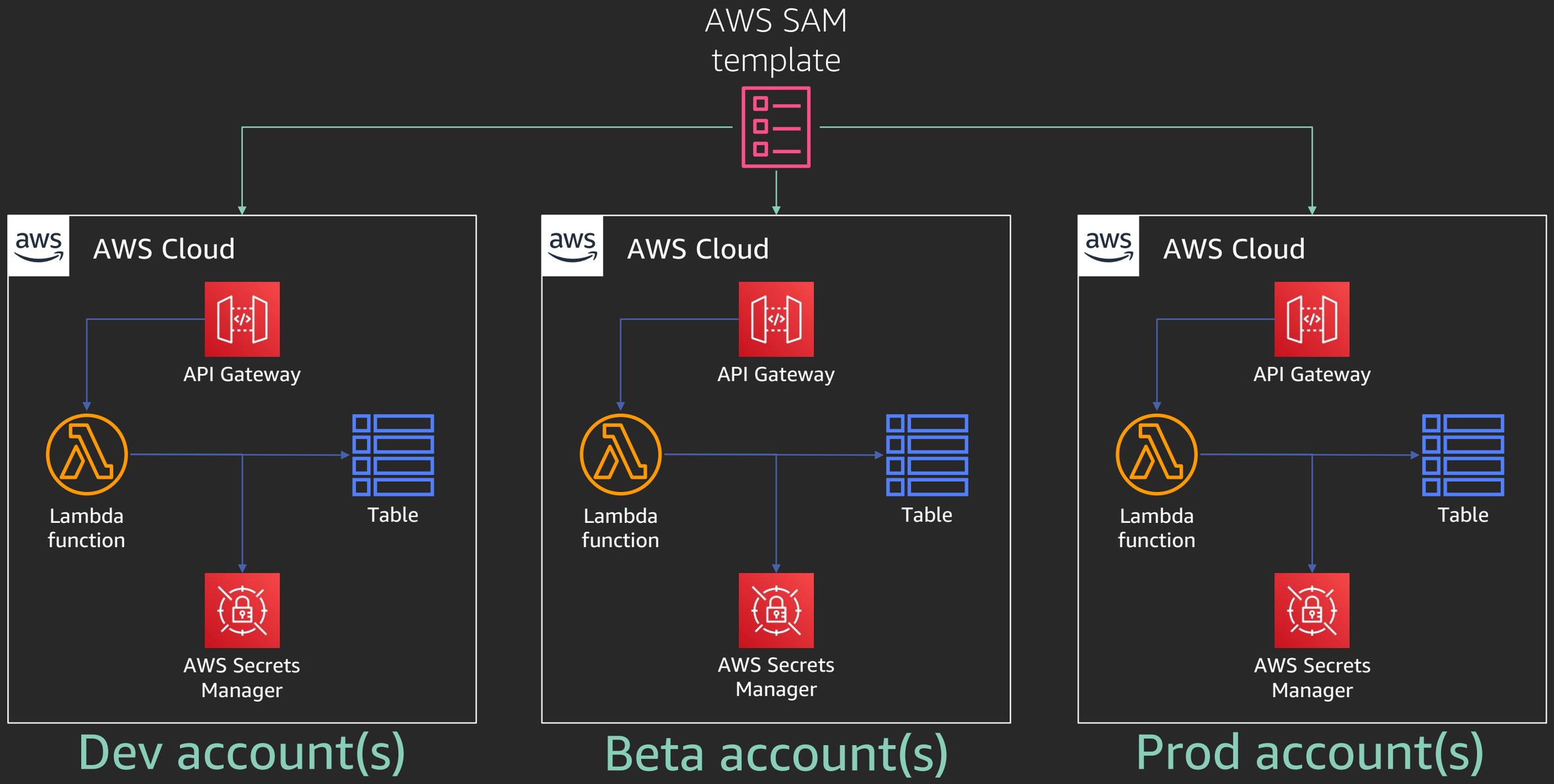


AWS SAM

- Use parameters and mappings when possible to build dynamic templates based on user inputs and pseudo parameters such as AWS::Region
- Use the Globals section to simplify templates
- Use ExportValue and ImportValue to share resource information across stacks
- Build out multiple environments, such as for development, test, production, and even DR using the same template, even across accounts



Environment



Secrets







AWS SAM parameters

- Supports default values
- Supports data type and allowed values
- Can be overridden on deployment
- Can be seen by other developers with access to account
- Can be passed to Lambda functions as environment variables

Stage variables

- Declared in API Gateway
- Use in stage-specific situations
- Can be overridden in canary releases

Parameter Store

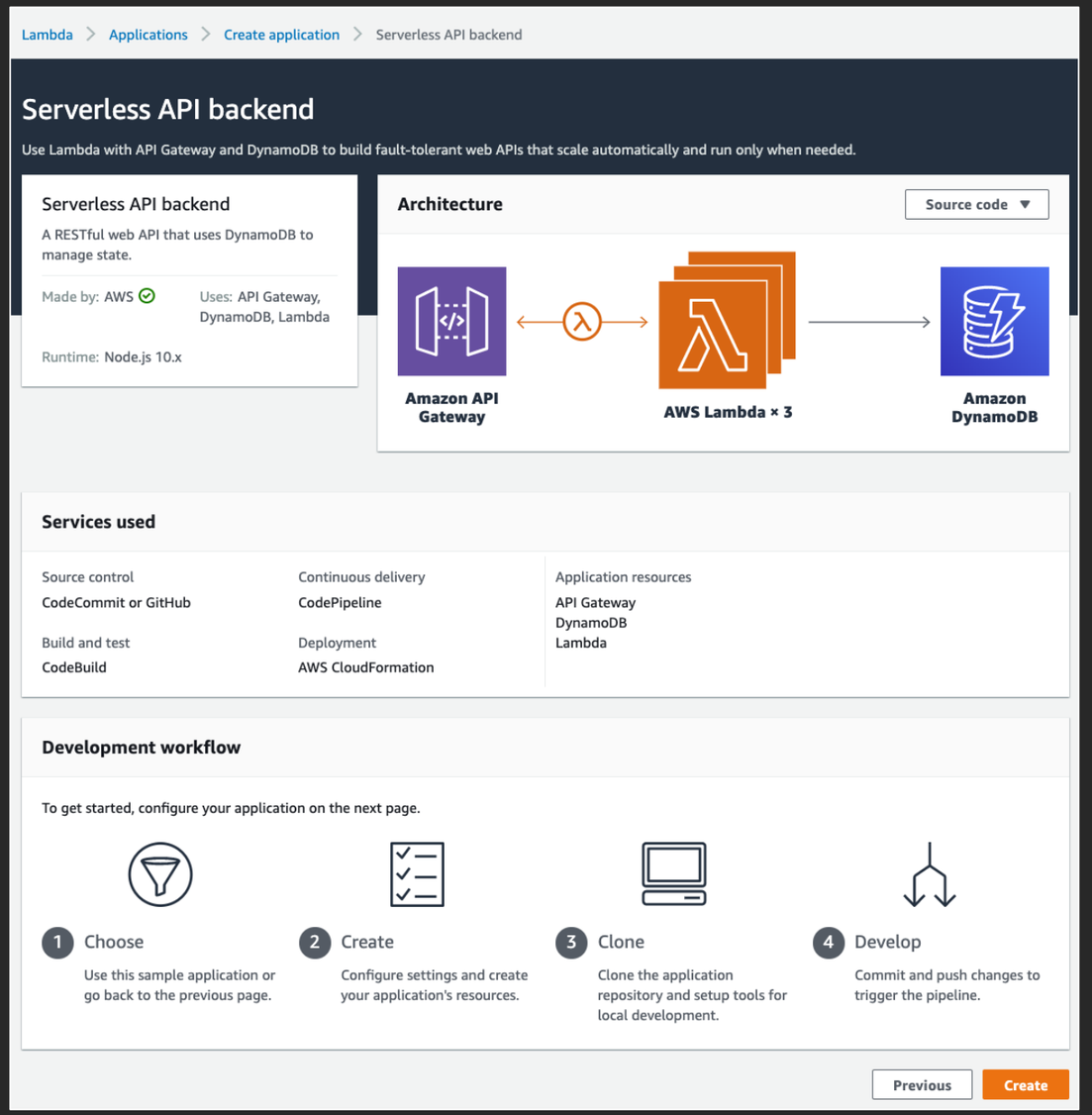
- Accessible through AWS SAM template at deployment time
- Accessible from code at runtime
- Supports encrypted values
- Account specific

“Create application” experience

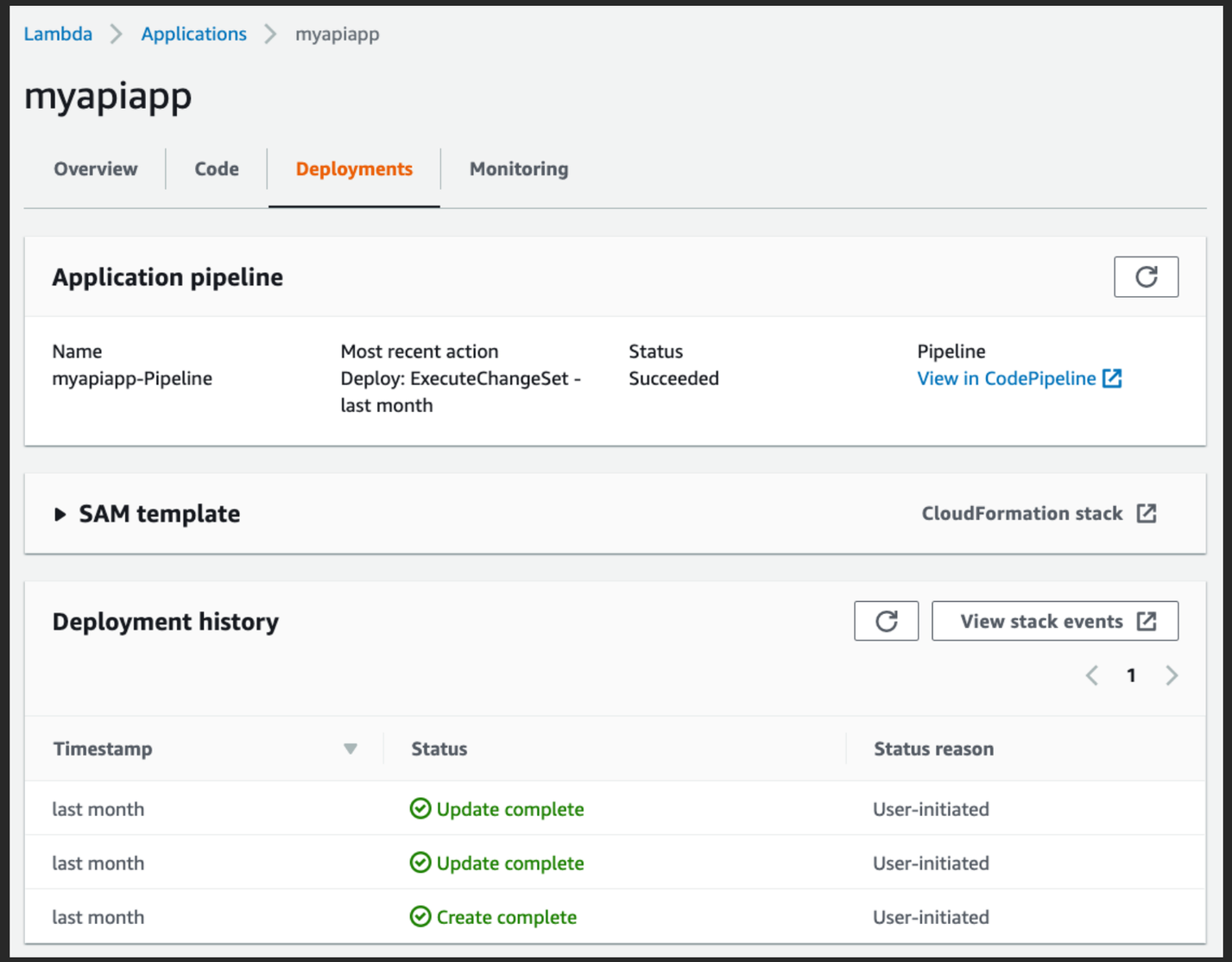
Create a serverless application from a starter template

- AWS SAM template for IaC
- CI/CD pipeline
- Clone locally for development with AWS SAM CLI

Create application



“Create application” experience



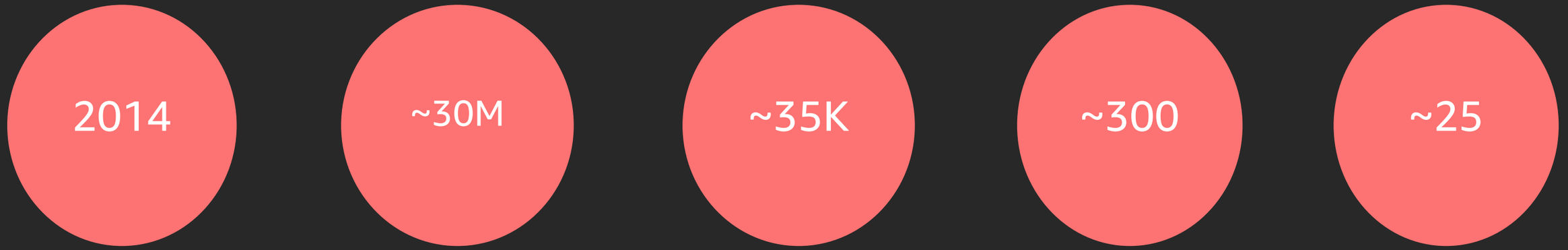
The Salesforce CI/CD story

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Who we are and what we do

- We support highly available monitoring solutions
  - 30M+ metrics/minute consumed in custom time-series metrics store
  - Monitoring and real-time alerting service
  - Lambdas covering many areas, including:
    - Import and export of monitoring data to various central Salesforce subsystems
    - Complimentary service monitors
    - AWS admin and security automation
    - Event annotations support

Metrics by the numbers



AWS account created … and so began the journey

Unique metrics/min.

Alert checks/min.

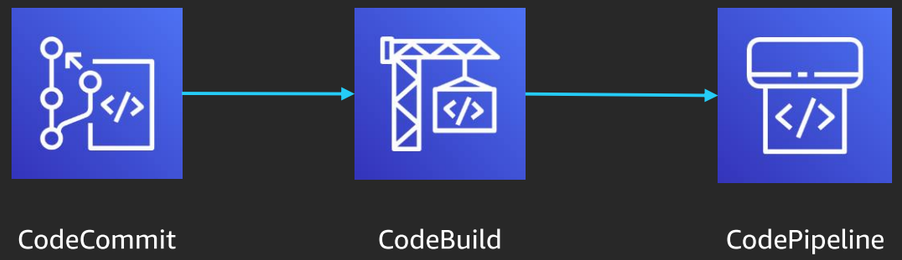
Monthly users

Serverless applications

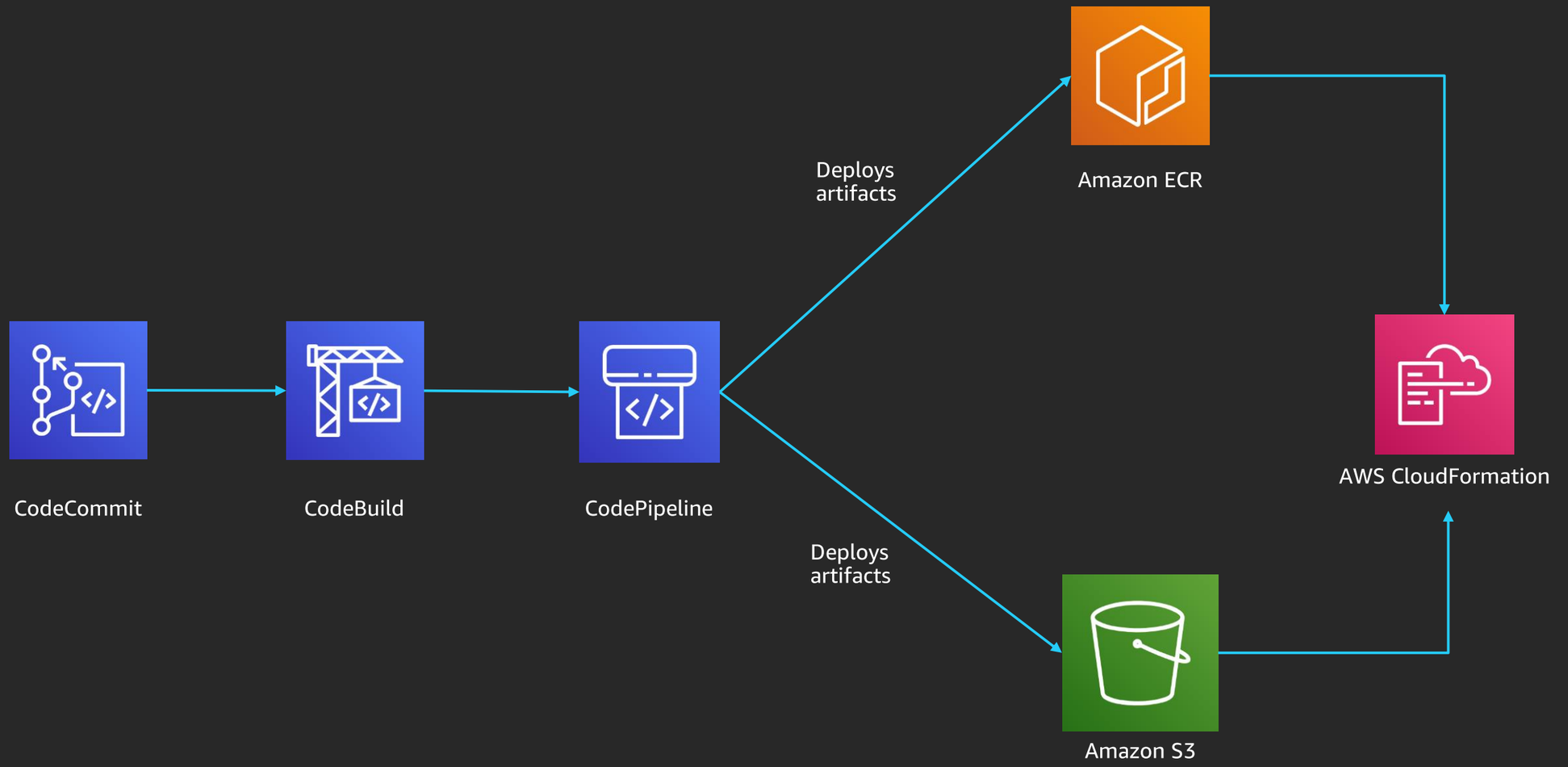
Our use case

- Custom containerized apps in Amazon EKS
- Lambdas and Amazon ECS/AWS Fargate
- Requirements and goals
  - AWS CloudFormation templates
  - Fully featured developer environment
  - Developer stacks separated from staging and production
  - All infrastructure in code
  - PR merge triggers automatic deployment of changes

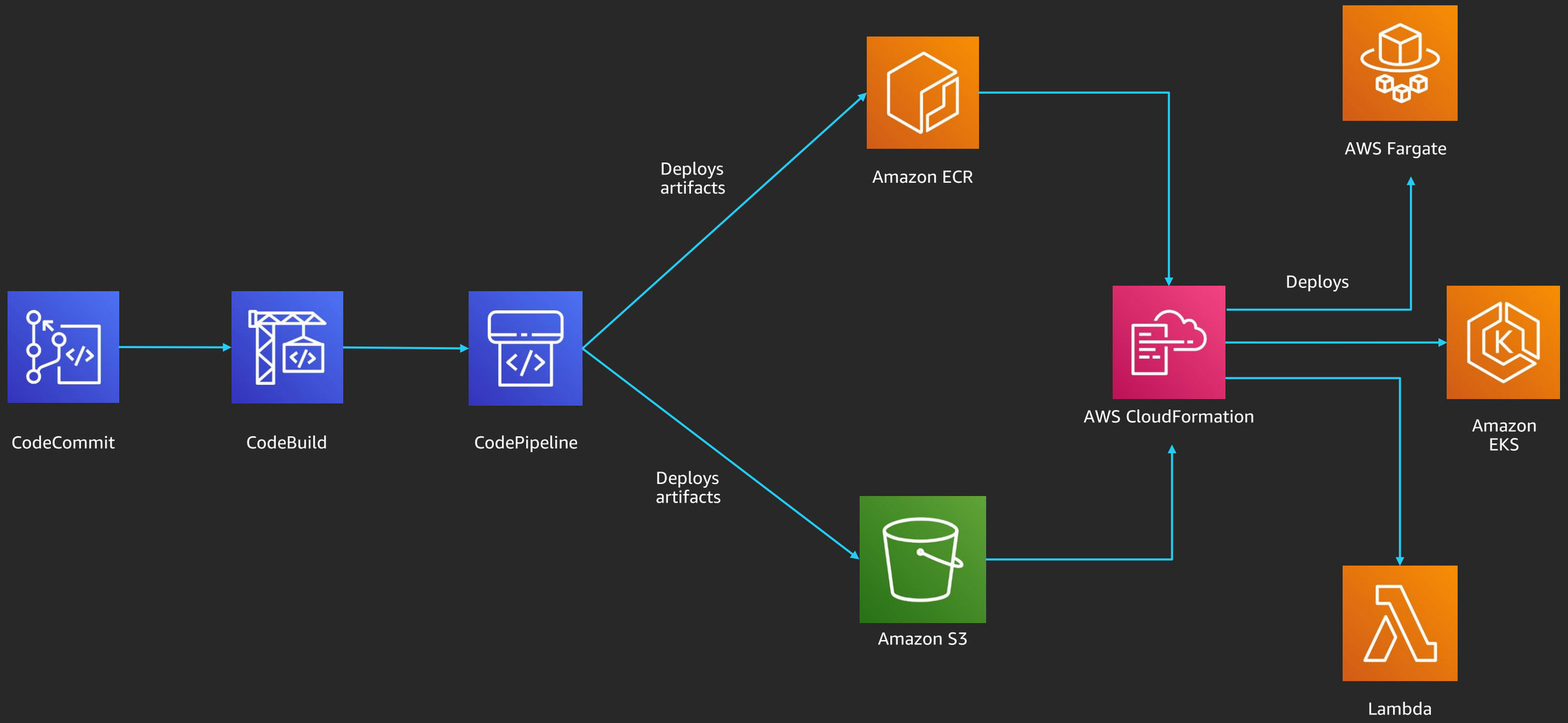
Continuous-deployment pipeline I



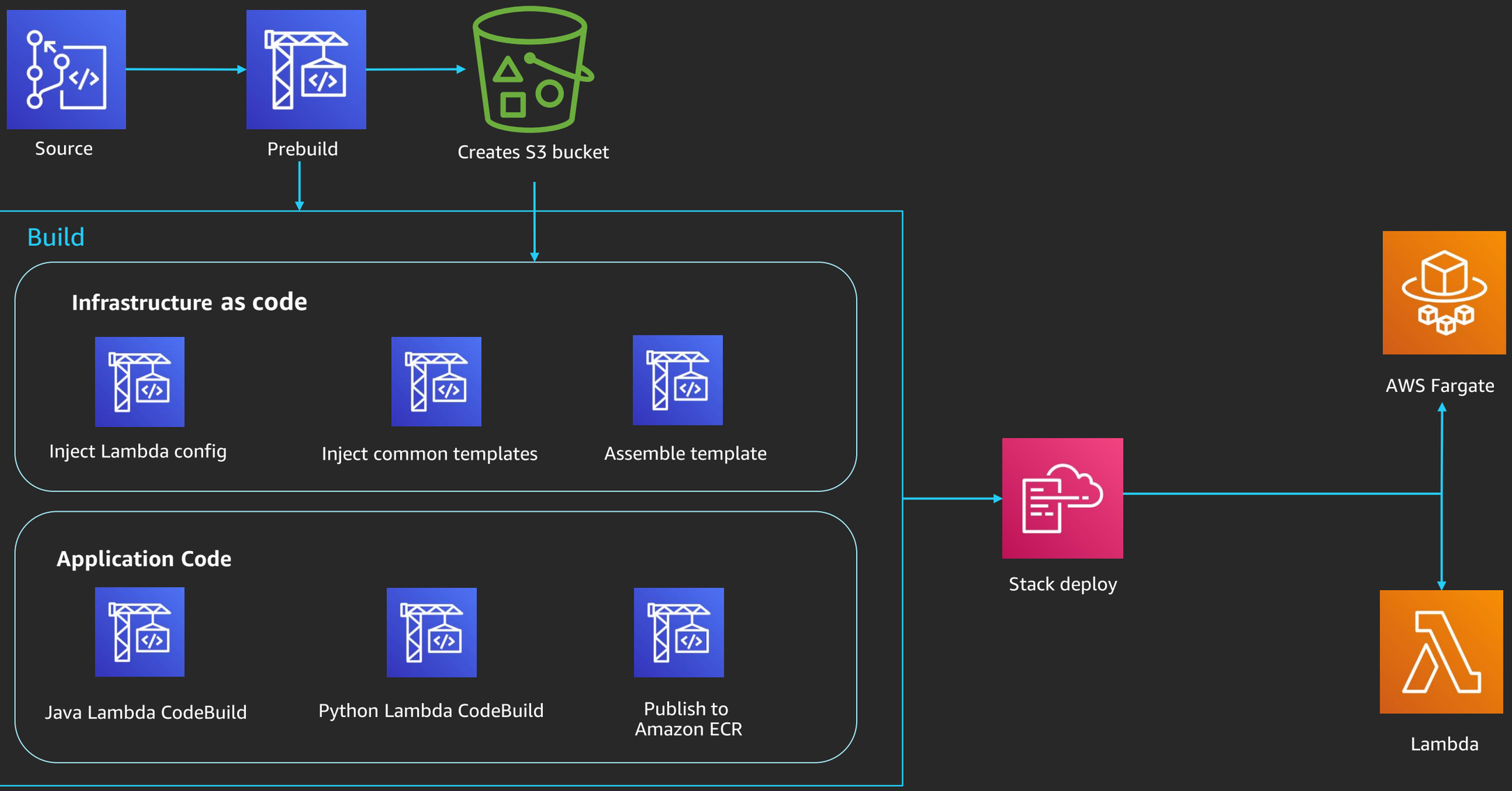
Continuous-deployment pipeline I



Continuous-deployment pipeline I



Serverless pipeline I



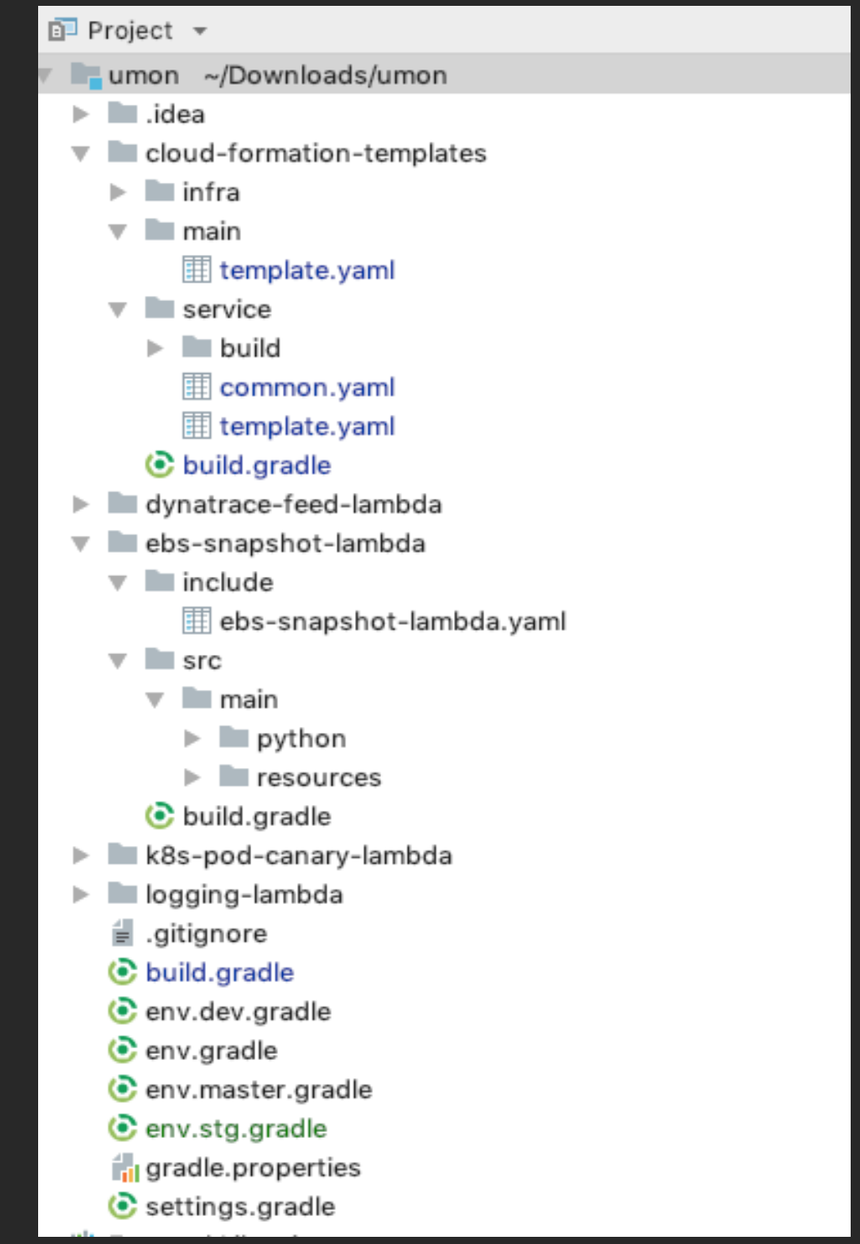
Serverless pipeline II

Application, configuration, and infra code bundled in one project

Inject environment-specific configuration (stg/prd)

Each Lambda augmented with Amazon CloudWatch metrics, alerts, and notifications if Lambda fails

- Avoids duplication of boilerplate code
- Streamlines build environment between JS/Java/Python



Serverless pipeline III

Lambda is built and build artifact referenced in template for each Lambda

Each commit is a new version – Build artifact changes based on commit hash

Redeployment of all Lambdas for each build

- Cheap/fast
- Deploy often, fail fast
- Detects breaking changes

```
LoggingLambda:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri:
      Bucket: !Ref s3bucket
      Key: !Sub "${s3Bucket}/${commitHash}/logging-lambda.zip"
    Events:
      Stream:
        Type: Kinesis
        Properties:
          Stream:
            "Fn::Sub":
            - arn:aws:kinesis:${AWS::Region}:${AWS::AccountId}:stream/${loggingStream}
            - loggingStream:
                "Fn::ImportValue": !Sub ${Env}-LoggingStream
          StartingPosition: LATEST
          BatchSize: 25
    Handler: com.salesforce.cc.infra.logging.LoggingLambda
    MemorySize: 2048
    Runtime: java8
    Timeout: 120
```

Best practices I

Consistent developer experience

All infra, apps, and config in code make it easy to replicate full environment

Share costly resources

Deploy often, fail fast

One code repository for everything*

AWS CloudFormation makes rollback atomic, cheap, and easy

Think about time savings as well—45 minutes to production!

* https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext

Best practices II

Use tracing, Amazon CloudWatch Logs, and built-in metrics to monitor Lambdas

More detailed metrics from Lambdas: Leverage open-source monitoring tools when cost-effective

Create common code packages to share across Lambdas (all code in one repo)

Provide logical defaults with the ability to easily override

Externalize configuration for easier templating environments (DEV/STG/PRD)

Set concurrency limit for Lambdas wherever applicable

Best practices III

Some developers are lazy

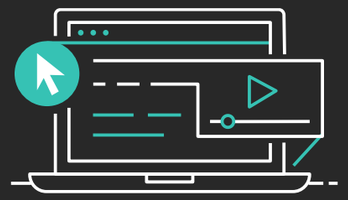
- Limit number of costly resources to absolute minimum for DEV/STG (Elasticsearch, etc.)
- Delete unused resources and automatically

Common pitfalls/lessons learned

- AWS CloudFormation updates should be tested in the CI/CD pipeline
- AWS CloudFormation feature support
- Not all resources can be updated easily (e.g., immutable NLBs, DNS names)
- Stack names have to be unique – Issue for dev environments

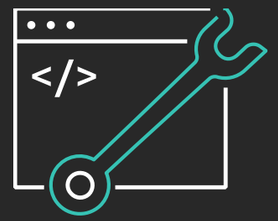
Learn serverless with AWS Training and Certification

Resources created by the experts at AWS to help you learn modern application development



Free, on-demand courses on serverless, including

- Introduction to Serverless Development
- Getting into the Serverless Mindset
- AWS Lambda Foundations
- Amazon API Gateway for Serverless Applications
- Amazon DynamoDB for Serverless Architectures



Additional digital and classroom trainings cover modern application development and computing

Visit the Learning Library at https://aws.training

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.