SERVERLESS-FIRST FUNCTION

aws

Automating serverless application development workflows

@edjgeek

Eric Johnson, Sr. Developer Advocate - Serverless

May 28, 2020

Who am I?

- Senior Developer Advocate – Serverless, AWS
- @edjgeek

Session agenda

- What is CI/CD?
- Fresh Tracks architecture
- Tooling
- Testing
- Best practices

What is CI/CD?

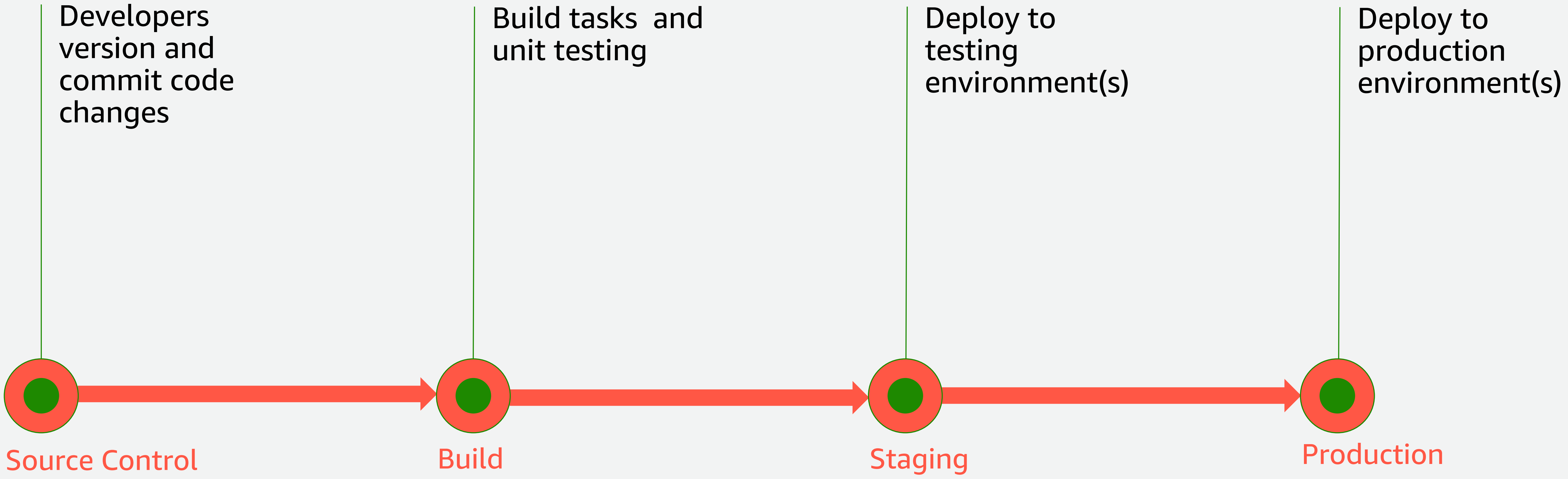
What is CI/CD?

CI: Continuous Integration

CD: Continuous Delivery

CD: Continuous Deployment

The pipeline



Developers version and commit code changes

Build tasks and unit testing

Deploy to testing environment(s)

Deploy to production environment(s)

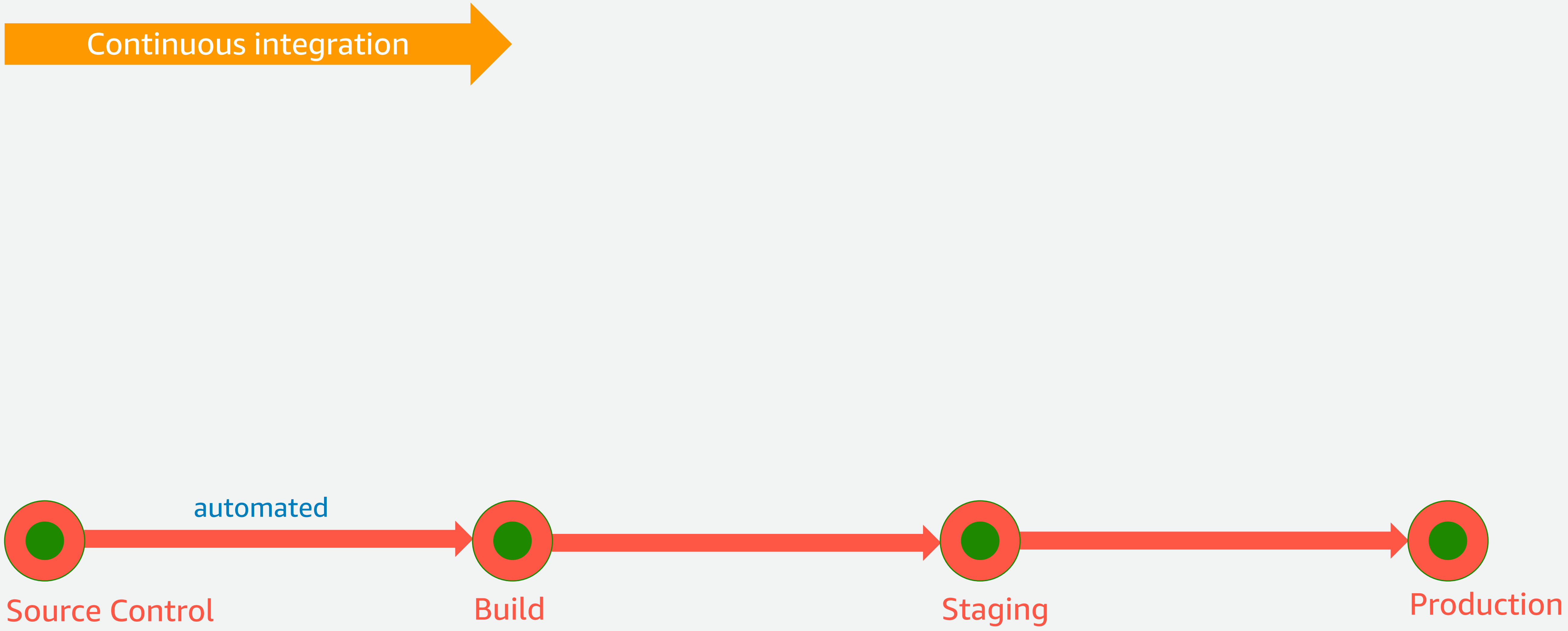
Source Control

Build

Staging

Production

The pipeline: continuous integration



Continuous integration

automated

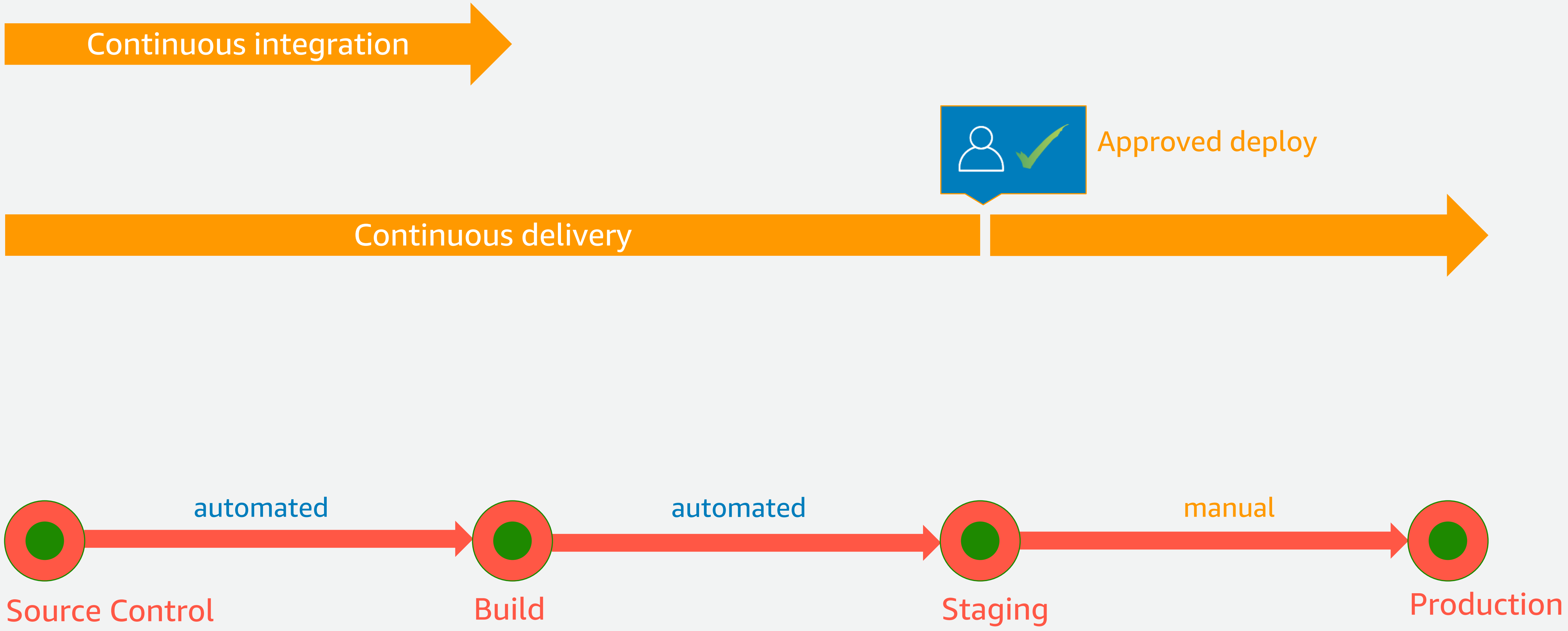
Source Control

Build

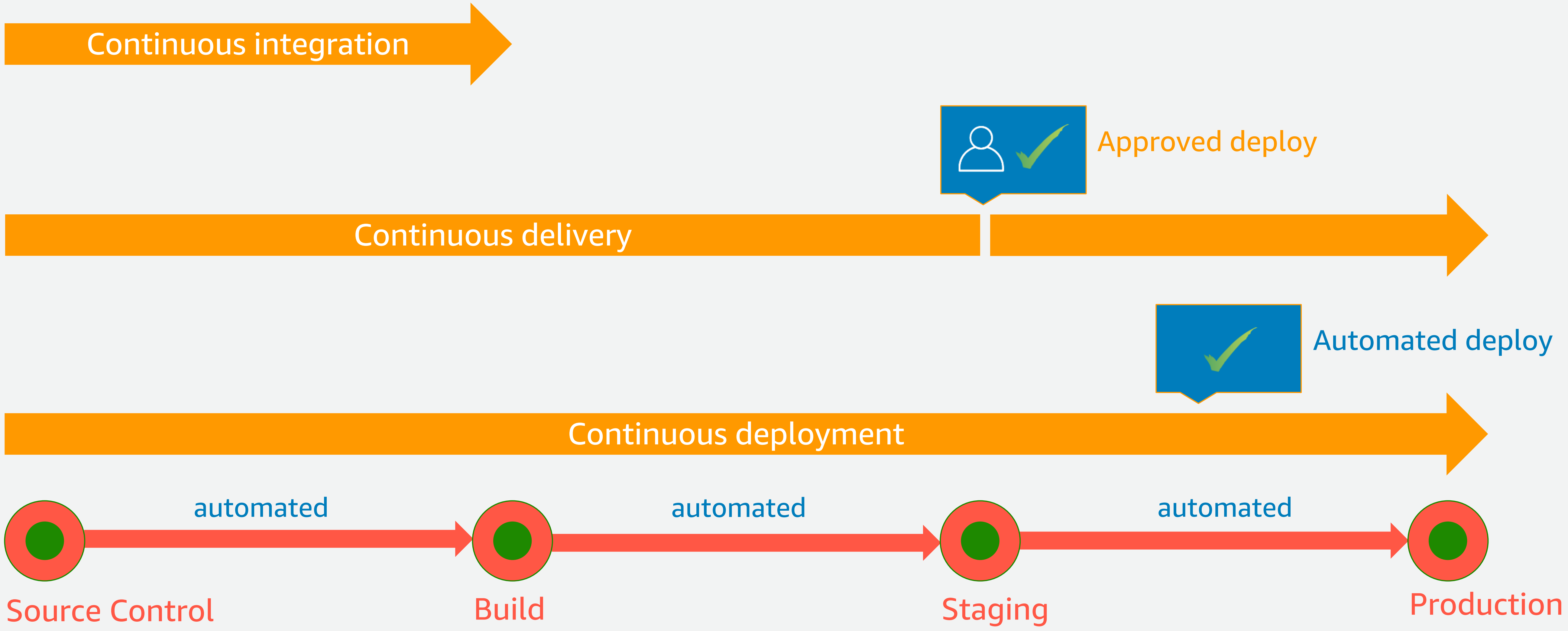
Staging

Production

The pipeline: continuous delivery

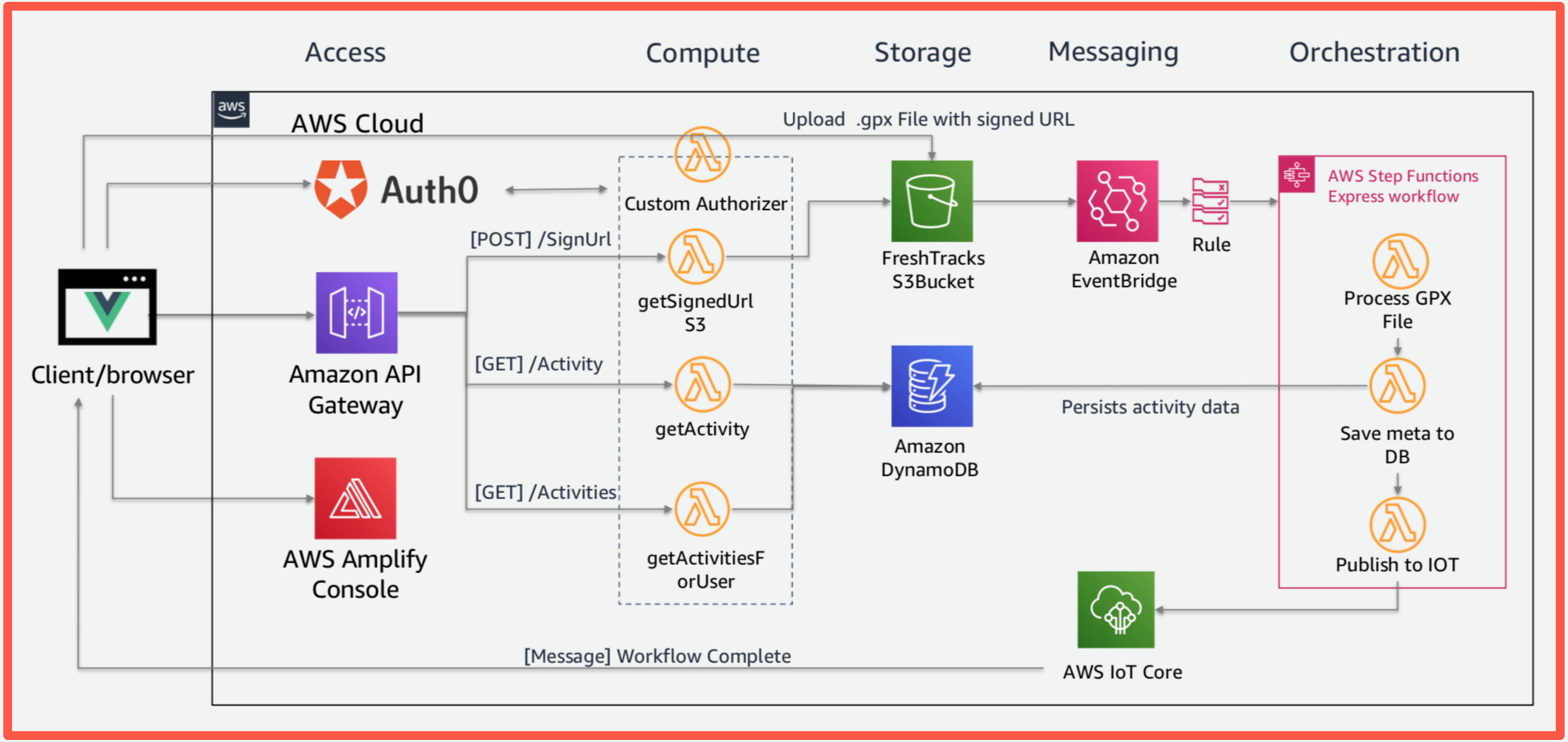


The pipeline: continuous deployment

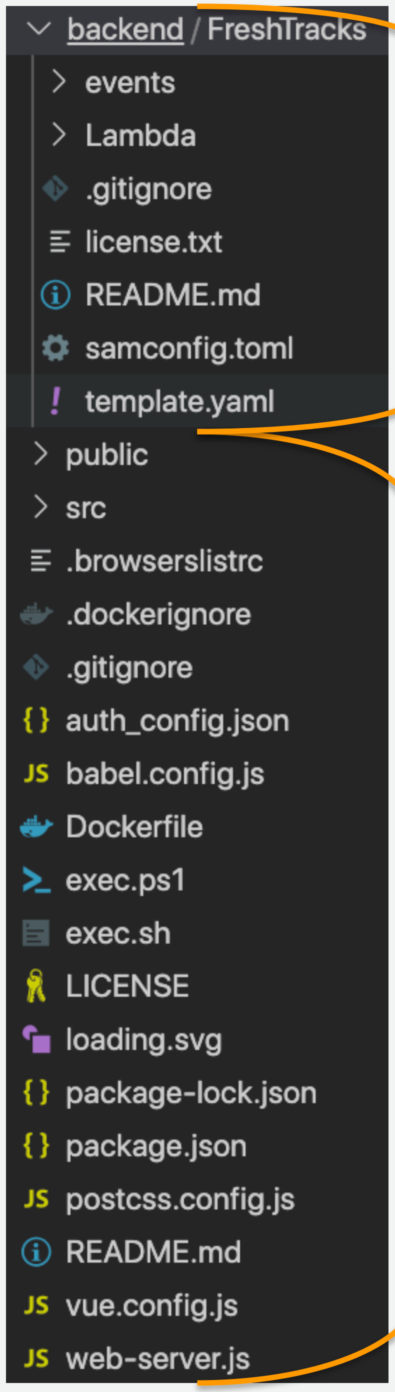


Fresh Tracks architecture

Fresh Tracks architecture



Fresh Tracks folder structure

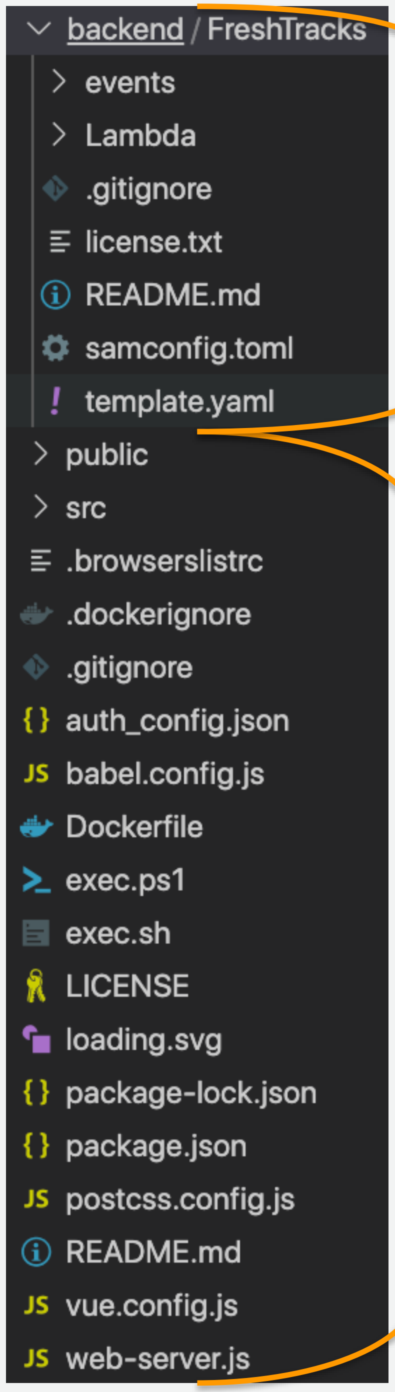


```
∨ backend / FreshTracks
    > events
    > Lambda
      .gitignore
      license.txt
      README.md
      samconfig.toml
      template.yaml
  > public
  > src
    .browserslistrc
    .dockerignore
    .gitignore
    auth_config.json
    babel.config.js
    Dockerfile
    exec.ps1
    exec.sh
    LICENSE
    loading.svg
    package-lock.json
    package.json
    postcss.config.js
    README.md
    vue.config.js
    web-server.js
```

Backend: Serverless

Client: Vuejs Application

Fresh Tracks folder structure



```
⌄ backend / FreshTracks
    > events
    > Lambda
    .gitignore
    license.txt
    README.md
    samconfig.toml
    template.yaml
  > public
  > src
  .browserslistrc
  .dockerignore
  .gitignore
  auth_config.json
  babel.config.js
  Dockerfile
  exec.ps1
  exec.sh
  LICENSE
  loading.svg
  package-lock.json
  package.json
  postcss.config.js
  README.md
  vue.config.js
  web-server.js
```

Backend: Serverless

The client can be easily separated into a separate repository if needed

Client: Vuejs Application

Tooling

Serverless Application Model

SAM

SAM comes in 2 parts

SAM comes in 2 parts

SAM templates

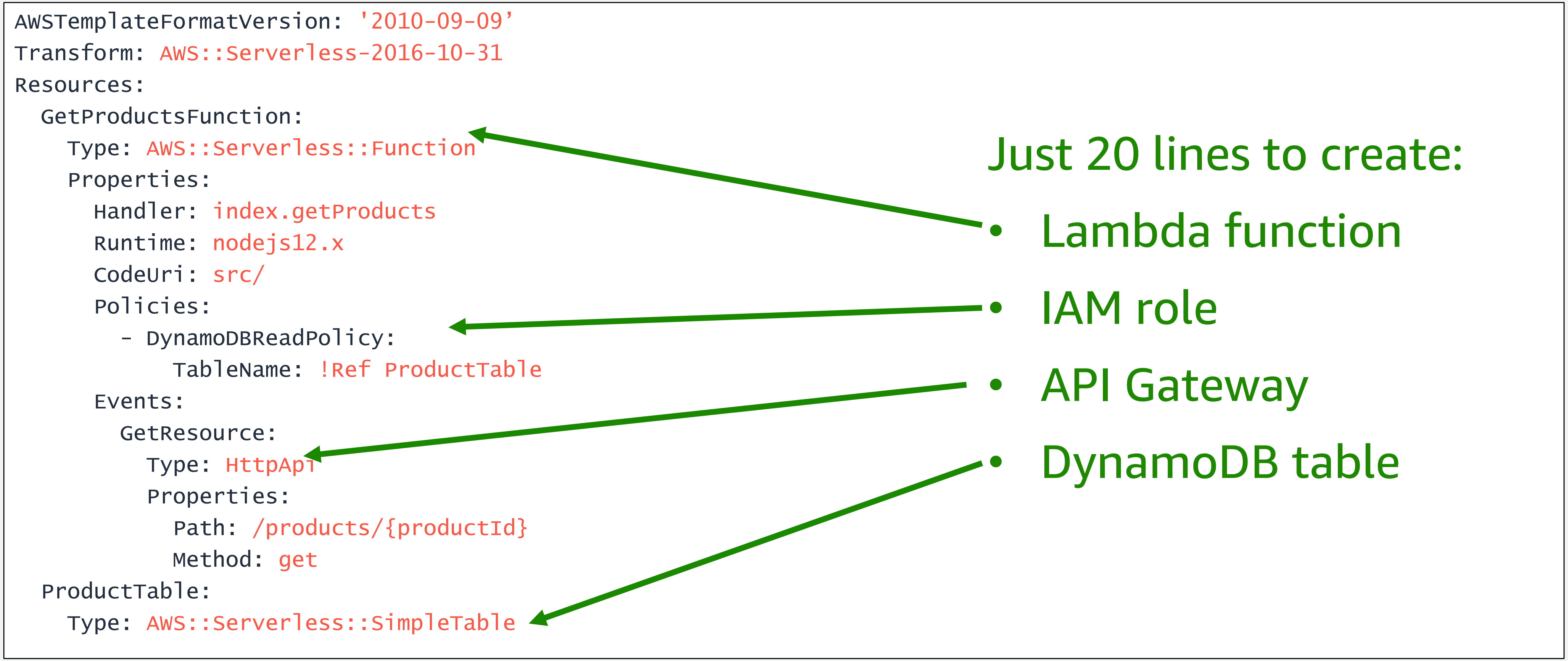
Using shorthand syntax to express resources and event source mappings, it provides infrastructure as code (IaC) for serverless applications.

SAM CLI

Provides tooling for local development, debugging, build, packaging, and deployment for serverless applications

https://aws.amazon.com/serverless/sam/

SAM templates



```yaml
AWSTemplateFormatVersion: '2010-09-09’
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs12.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: HttpApi
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

Just 20 lines to create:

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

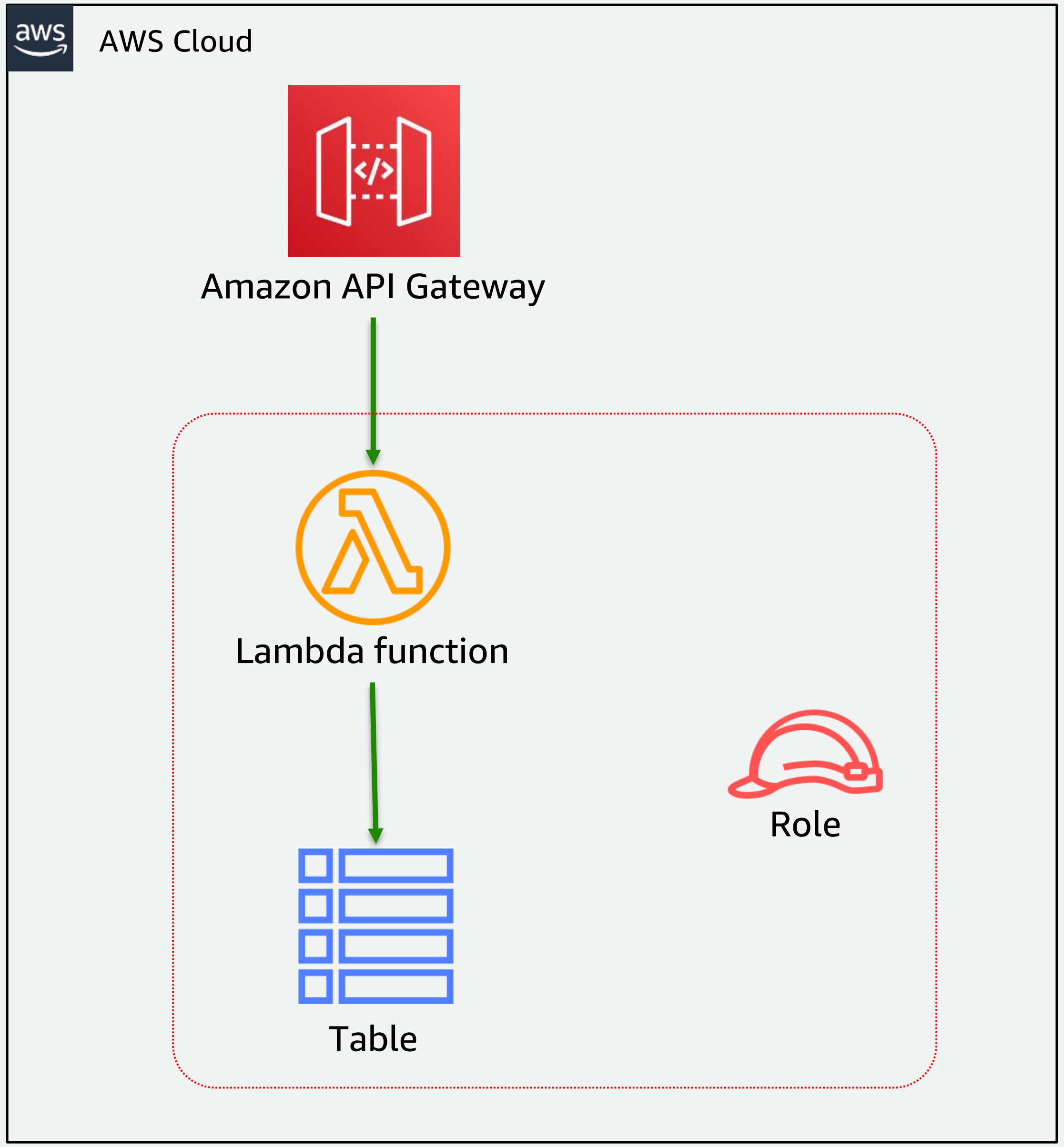
SAM templates

```
AWSTemplateFormatVersion: '2010-09-09’
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs12.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: HttpApi
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

Allowing this

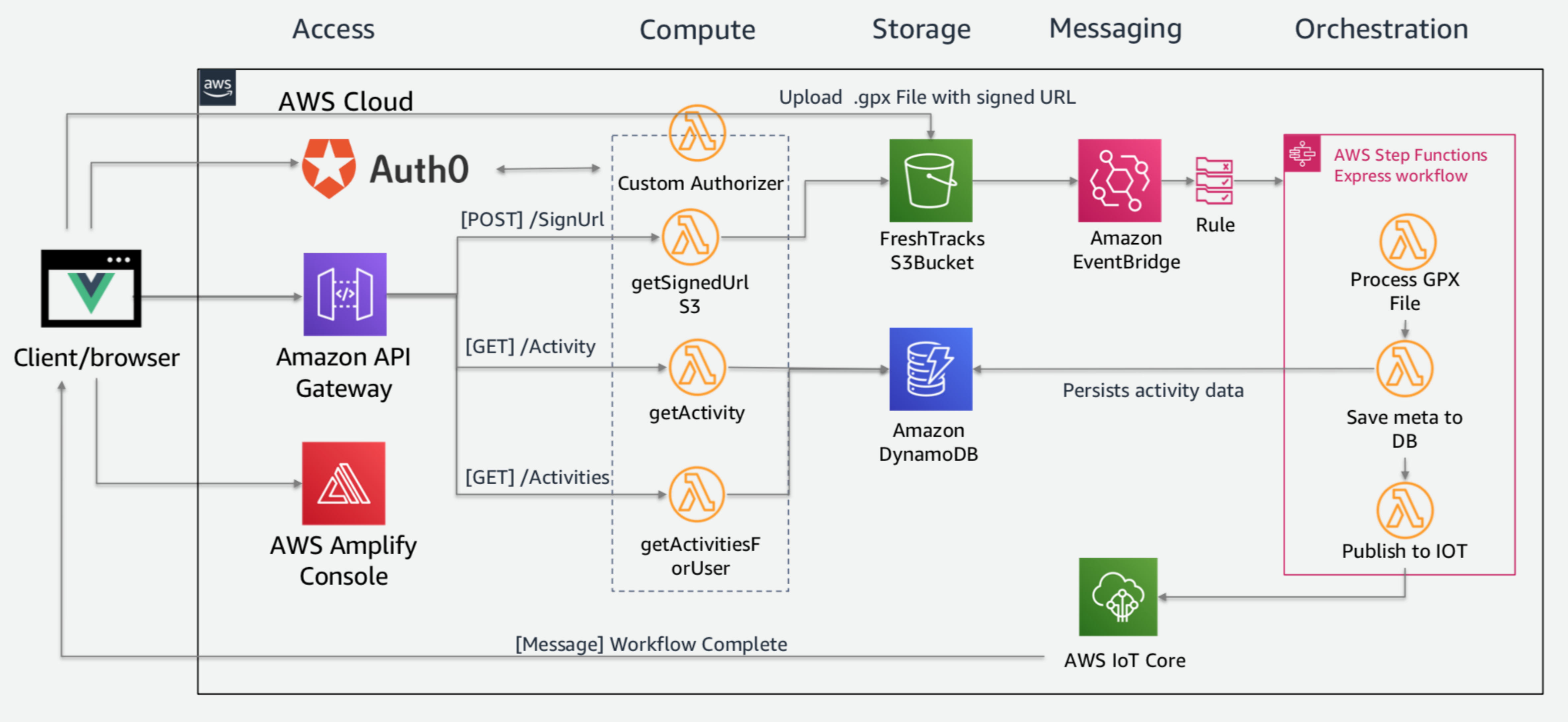
===

To become this



Fresh Tracks

Roughly 90% of this application is managed and deployed with SAM.

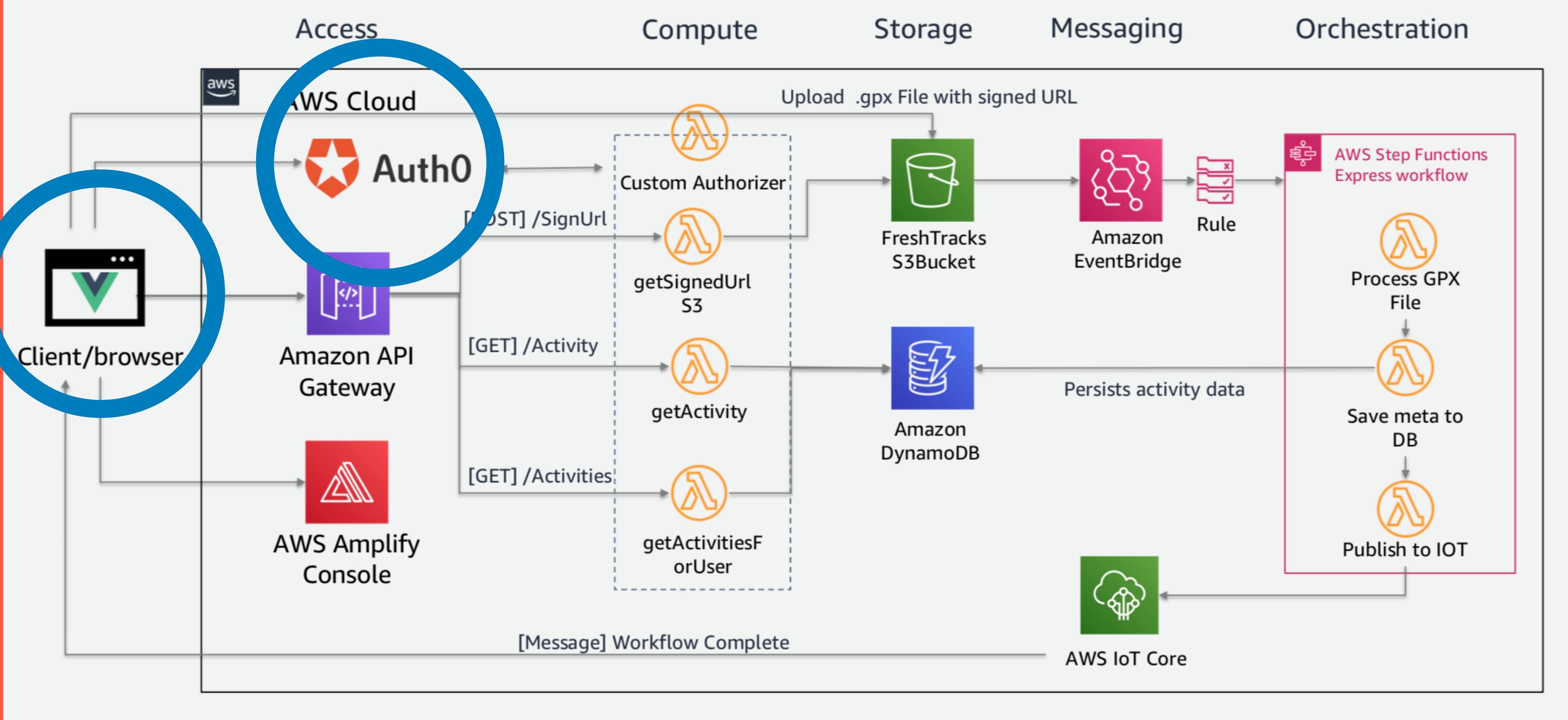


Fresh Tracks

Auth0 is deployed separately

The Amplify app will be created

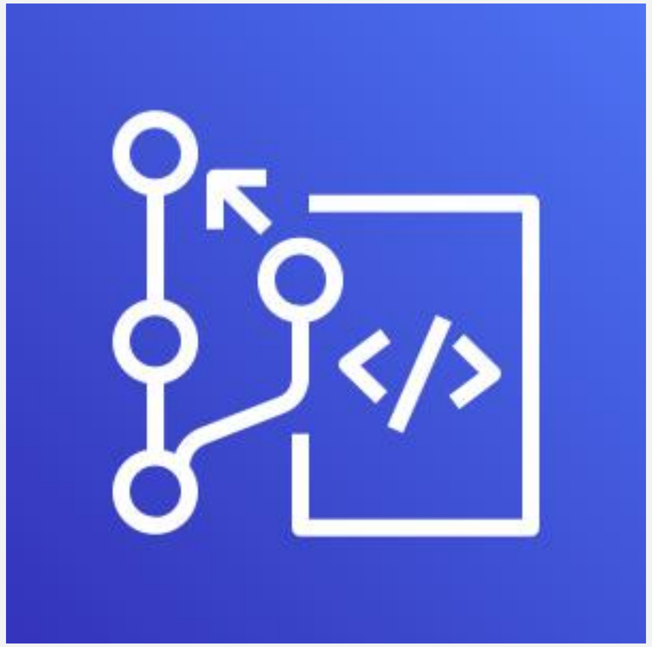
The client code will be deployed separately



Code repository

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 CodeBuild and CodePipeline

Integrates with CodeBuild

AWS Amplify Console

Deploying the client

AWS Amplify Console

- Powered by Lambda@Edge, Amazon S3, and Amazon CloudFront
- Integrated CI/CD
- Build configurations
- Feature branch deployments
- Global availability (CDN)
- Basic password protection

Amazon CloudFront

AWS Lambda

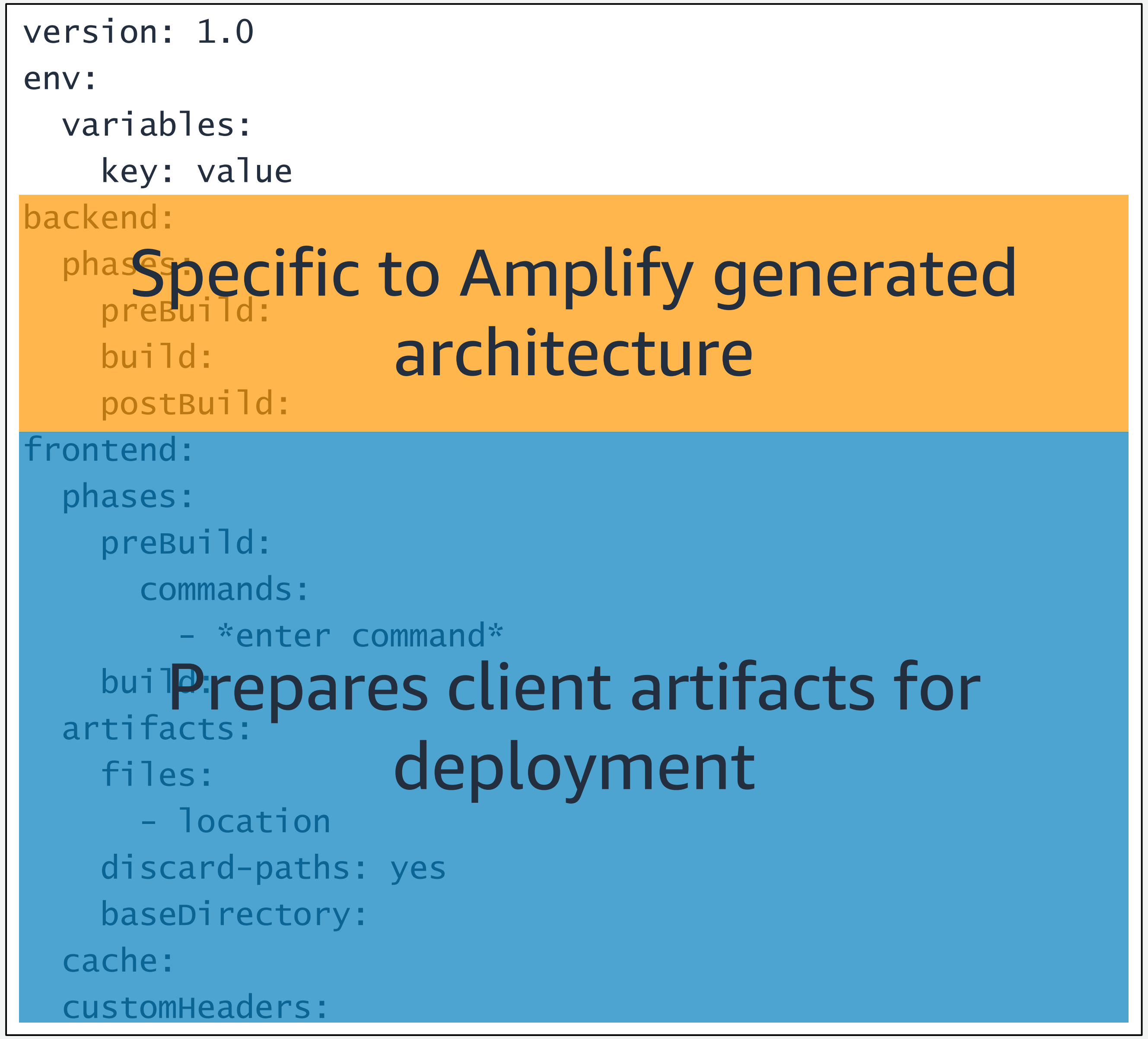
Amazon S3

Amplify Console buildspec.yaml

```
version: 1.0
env:
  variables:
    key: value
backend:
  phases:
    preBuild:
    build:
    postBuild:
frontend:
  phases:
    preBuild:
      commands:
        - *enter command*
    build:
  artifacts:
    files:
      - location
    discard-paths: yes
    baseDirectory:
  cache:
  customHeaders:
```

```
test:
  phases:
    preTest:
      commands:
        - *enter command*
    test:
      commands:
        - *enter command*
    postTest:
      commands:
        - *enter command*
artifacts:
  files:
    - location
    - location
  configFilePath: *location*
  baseDirectory: *location*
```

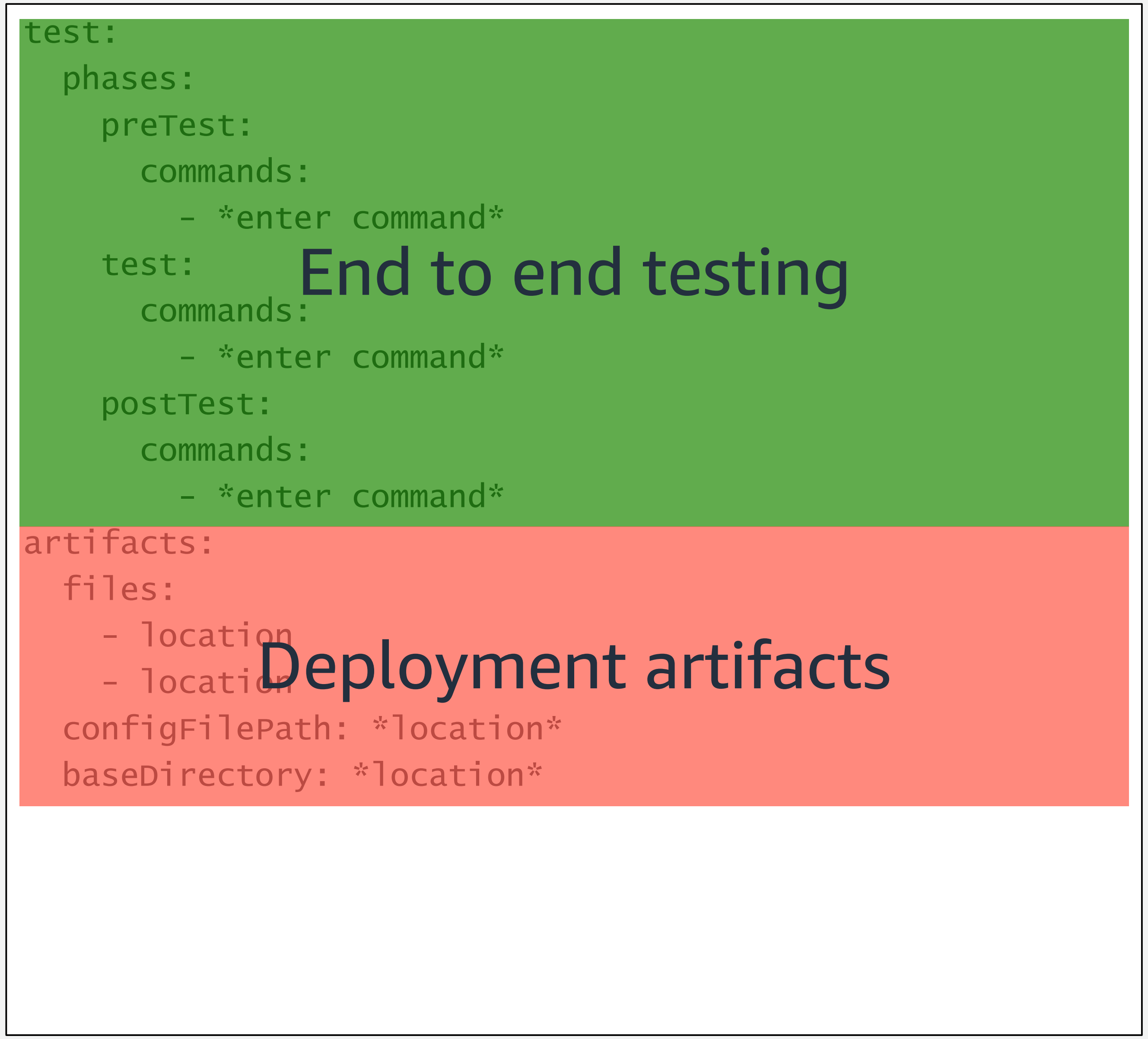
Amplify Console buildspec.yaml;



```
version: 1.0
env:
  variables:
    key: value
backend:
  phases:
    preBuild:
    build:
    postBuild:
frontend:
  phases:
    preBuild:
      commands:
        - *enter command*
    build:
  artifacts:
    files:
      - location
    discard-paths: yes
    baseDirectory:
  cache:
  customHeaders:
```

Specific to Amplify generated architecture

Prepares client artifacts for deployment

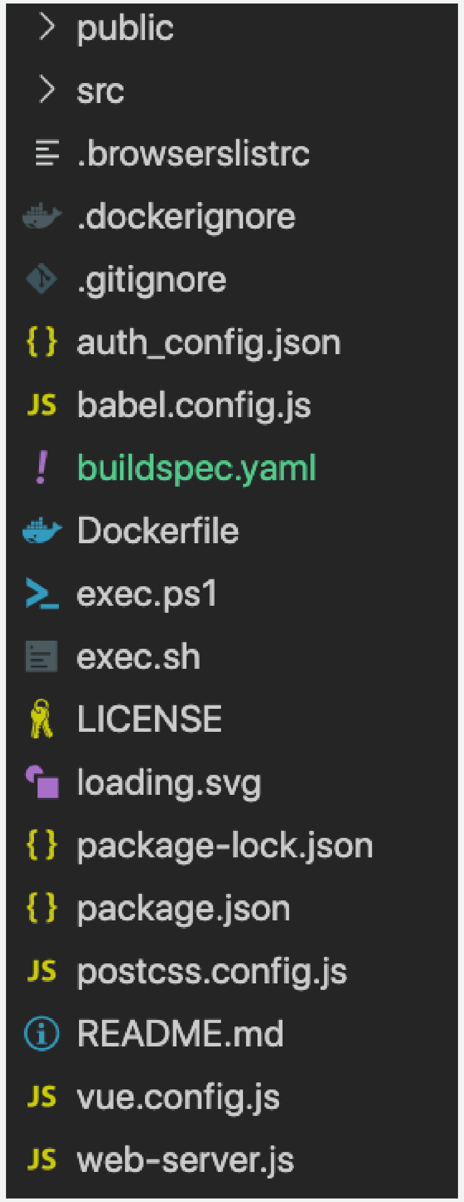


```
test:
  phases:
    preTest:
      commands:
        - *enter command*
    test:
      commands:
        - *enter command*
    postTest:
      commands:
        - *enter command*
artifacts:
  files:
    - location
    - location
  configFilePath: *location*
  baseDirectory: *location*
```

End to end testing

Deployment artifacts

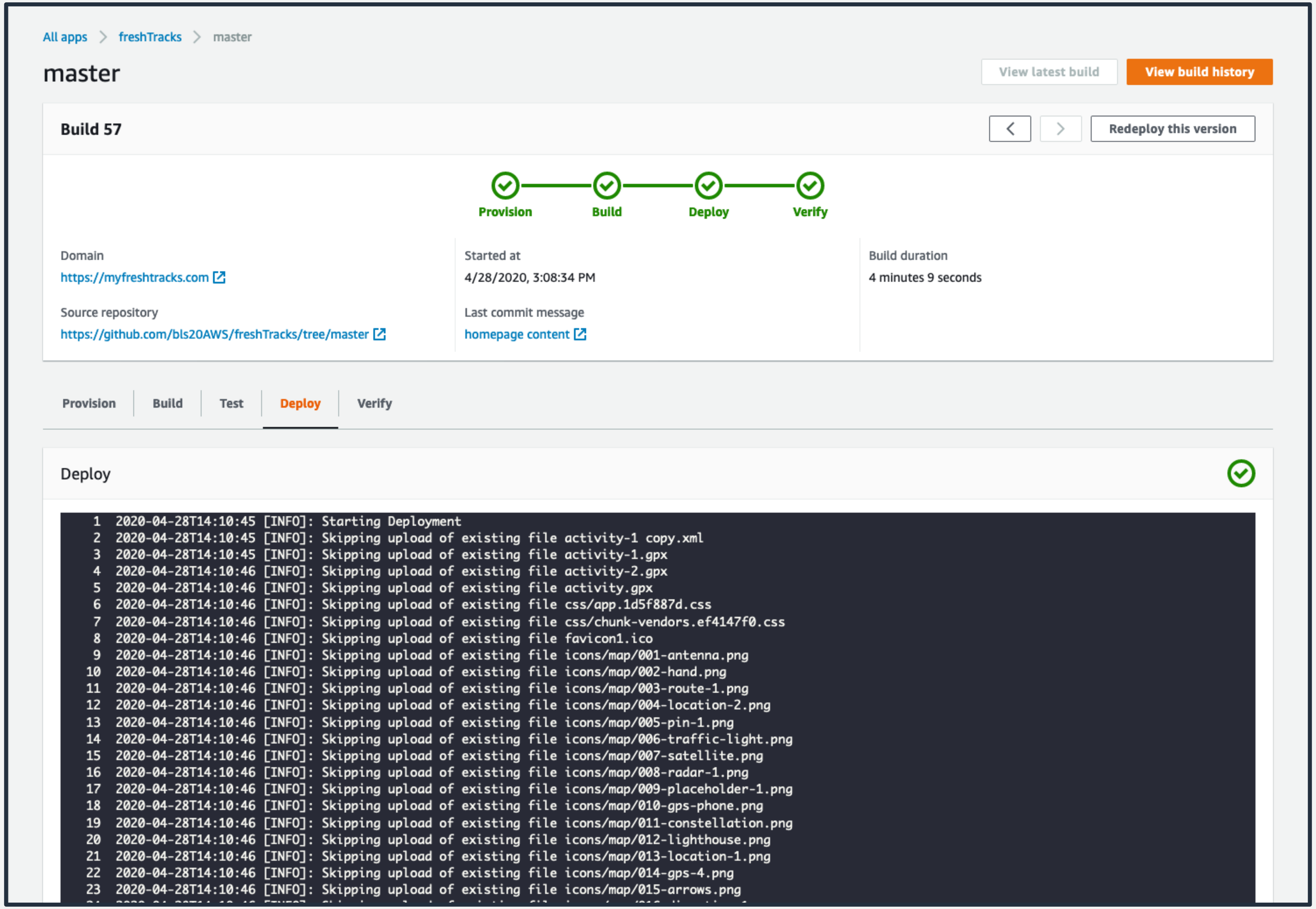
Deploying the client



```
> public
> src
.browserslistrc
.dockerignore
.gitignore
auth_config.json
babel.config.js
buildspec.yaml
Dockerfile
exec.ps1
exec.sh
LICENSE
loading.svg
package-lock.json
package.json
postcss.config.js
README.md
vue.config.js
web-server.js
```

1. Developer commits code to repository
2. Amplify console is triggered
3. Code is prepared and tests are run according to the buildspec specifications
4. The client is deployed

Deploying the client



master

Build 57

Domain
https://myfreshtracks.com

Source repository
https://github.com/bls20AWS/freshTracks/tree/master

Started at
4/28/2020, 3:08:34 PM

Last commit message
homepage content

Build duration
4 minutes 9 seconds

Deploy

```
2020-04-28T14:10:45 [INFO]: Starting Deployment
2020-04-28T14:10:45 [INFO]: Skipping upload of existing file activity-1 copy.xml
2020-04-28T14:10:45 [INFO]: Skipping upload of existing file activity-1.gpx
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file activity-2.gpx
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file activity.gpx
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file css/app.1d5f887d.css
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file css/chunk-vendors.ef4147f0.css
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file favicon1.ico
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/001-antenna.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/002-hand.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/003-route-1.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/004-location-2.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/005-pin-1.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/006-traffic-light.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/007-satellite.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/008-radar-1.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/009-placeholder-1.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/010-gps-phone.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/011-constellation.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/012-lighthouse.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/013-location-1.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/014-gps-4.png
2020-04-28T14:10:46 [INFO]: Skipping upload of existing file icons/map/015-arrows.png
```

AWS CodePipeline

The orchestrator

AWS CodePipeline

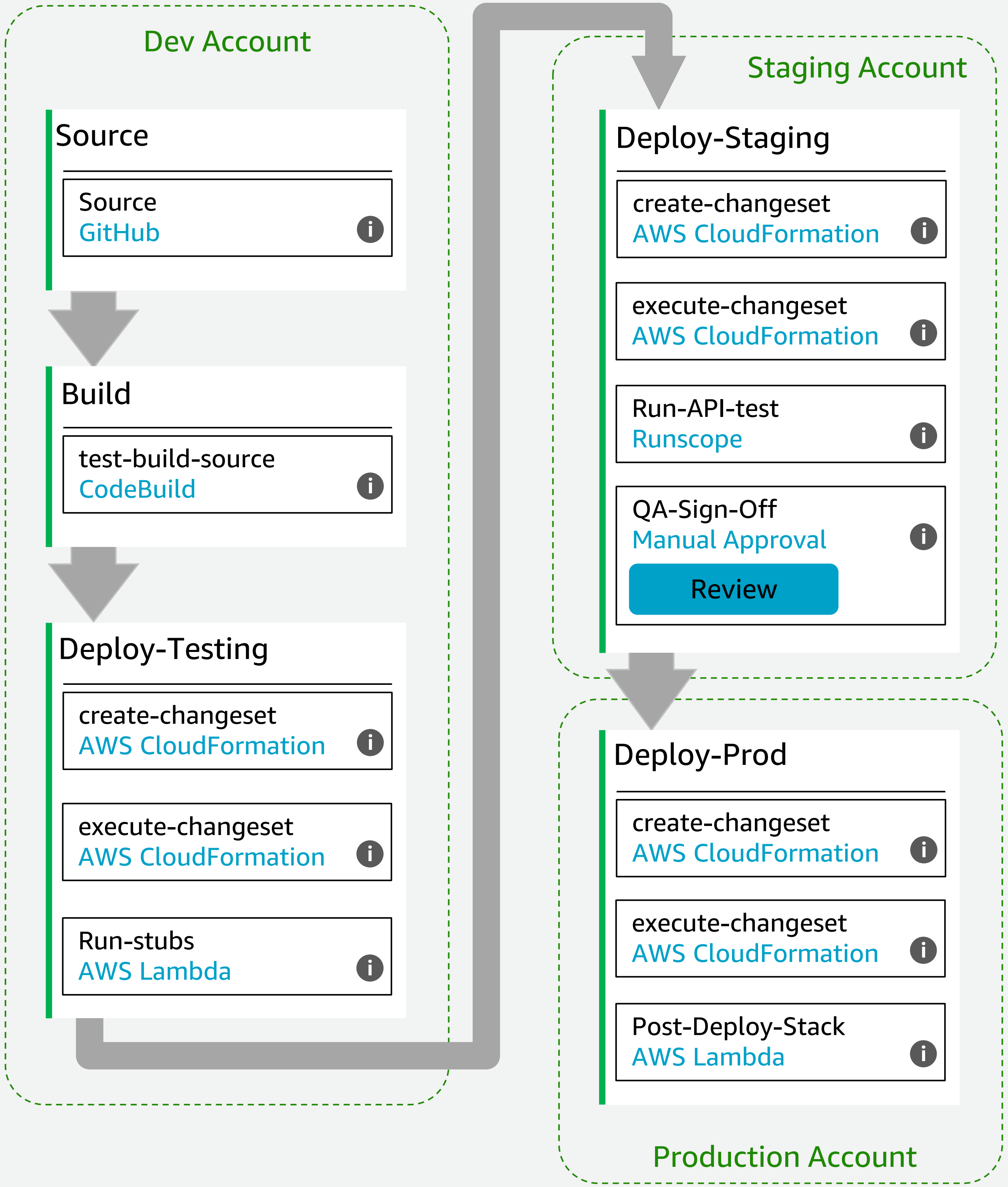
- Continuous delivery service for fast and reliable application updates
- Model and visualize your software release process
- Builds, tests, and deploys your code every time there is a code change
- Integrates with third-party tools and AWS

https://aws.amazon.com/codepipeline/

Example of full pipeline

This pipeline:

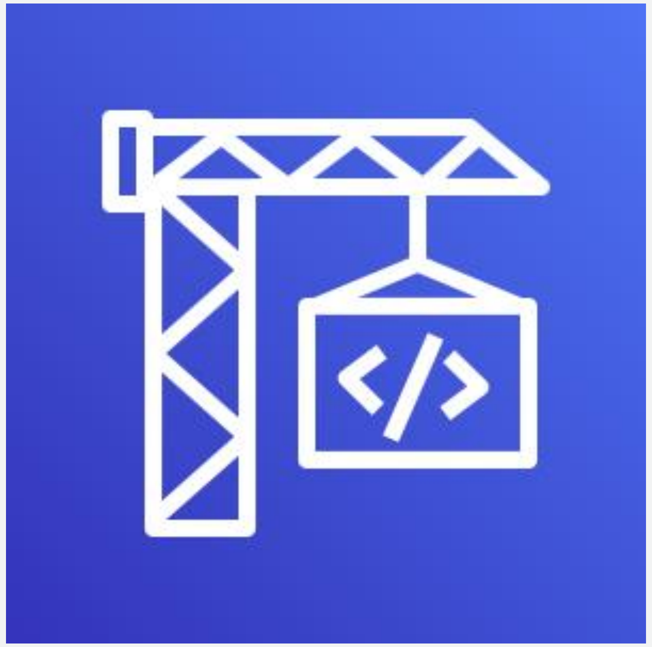
- Five stages
- Builds code artifact
- Three deployed to “environments”
- Uses SAM/CloudFormation to deploy artifact and other AWS resources
- Has Lambda custom actions for testing functions
- Integrates with a 3rd party tool/service
- Has a manual approval before deploying to production



AWS CodeBuild

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 SSM 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:
      - 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
```

The buildspec.yaml file

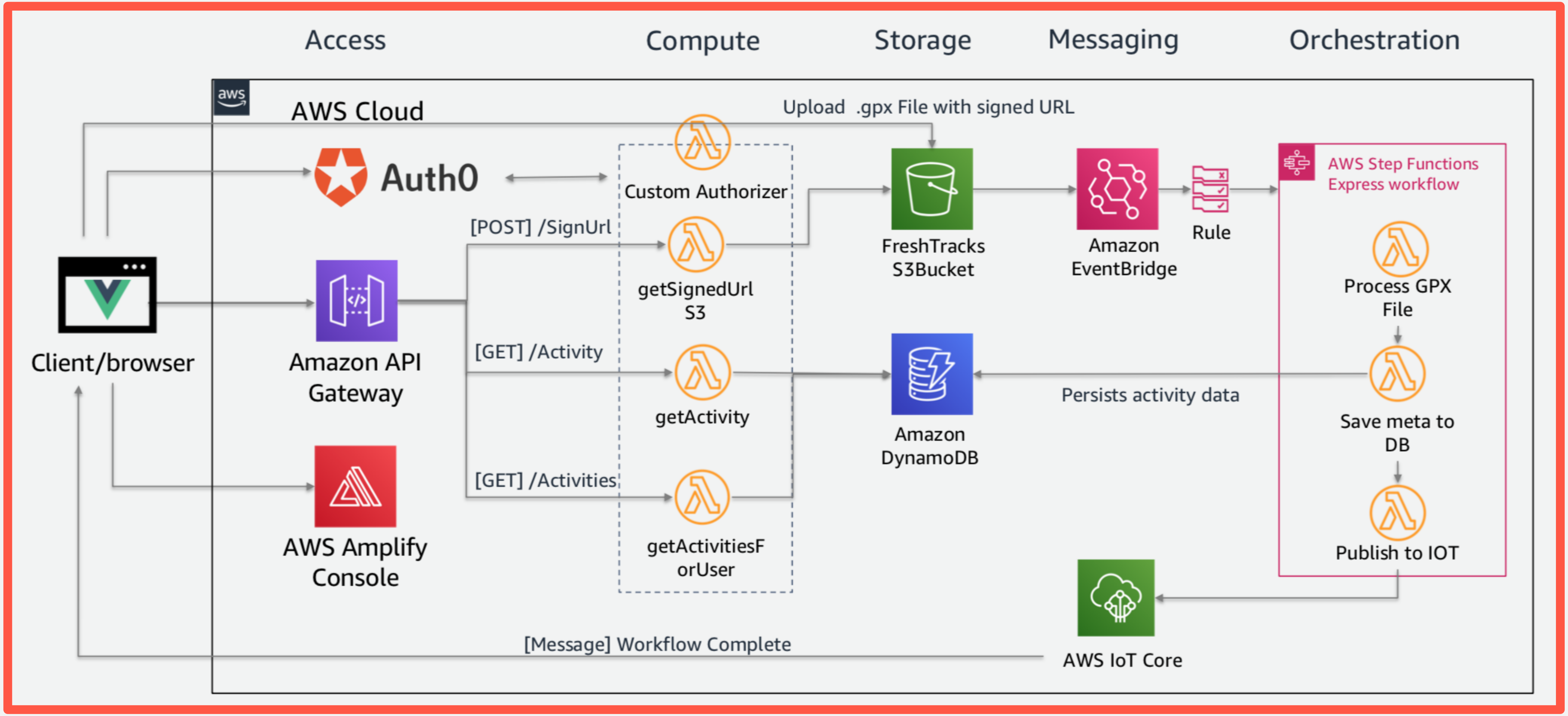
```yaml
version: 0.2
variables:
  parameter-store:
    BUCKET_NAME: /CodeBuild/BucketName

phases:
  install:
    commands:
      - 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
```

Prepare and test code

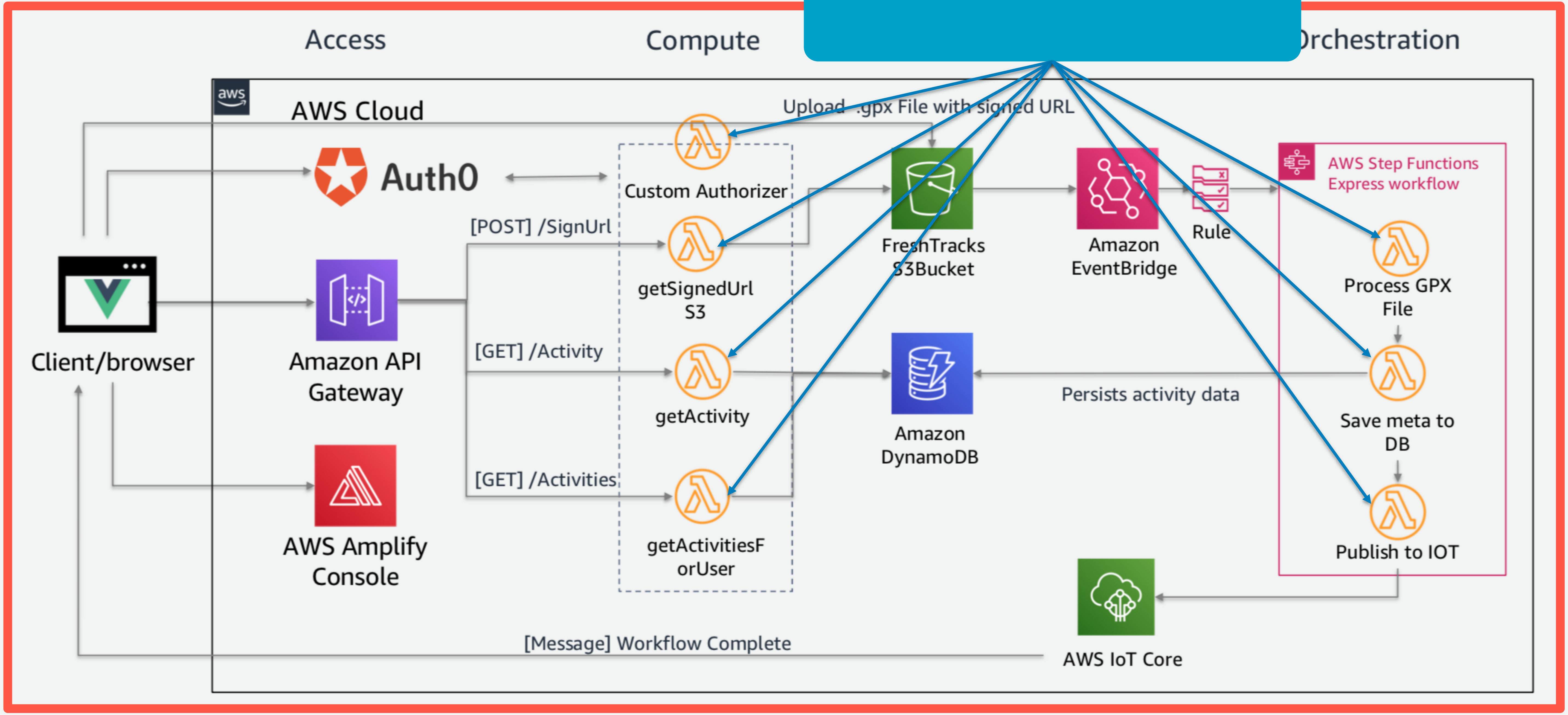
Deployment artifacts

Fresh Tracks



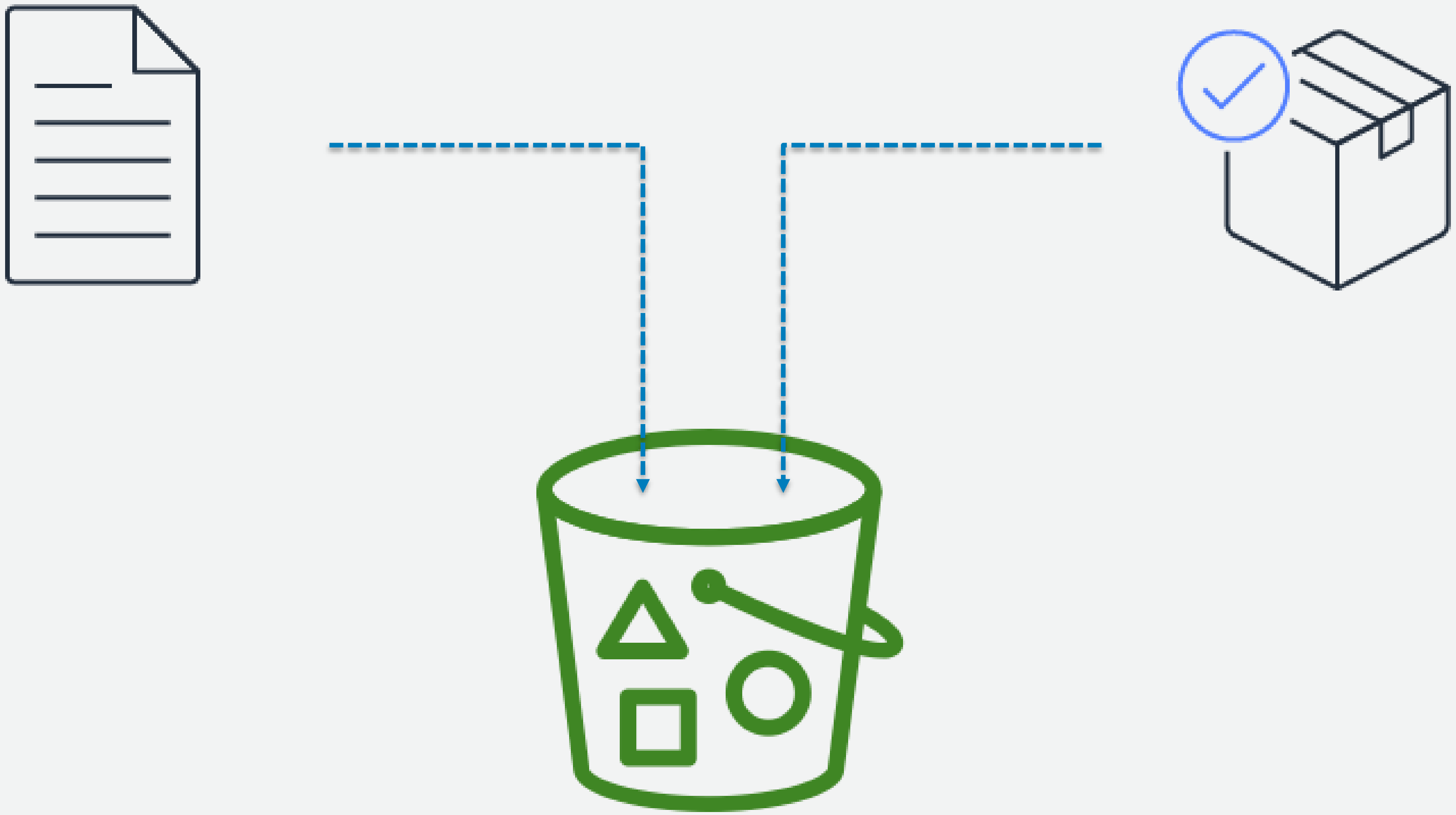
Fresh Tracks

Code built and tested



Artifacts

```
sam package –template-file template.yaml –s3-bucket $BUCKET_NAME –output-template out.yaml
```



AWS CloudFormation

Deploying the backend

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/

CloudFormation deploy

1

A template is submitted to CloudFormation

2

A change set is created and validated

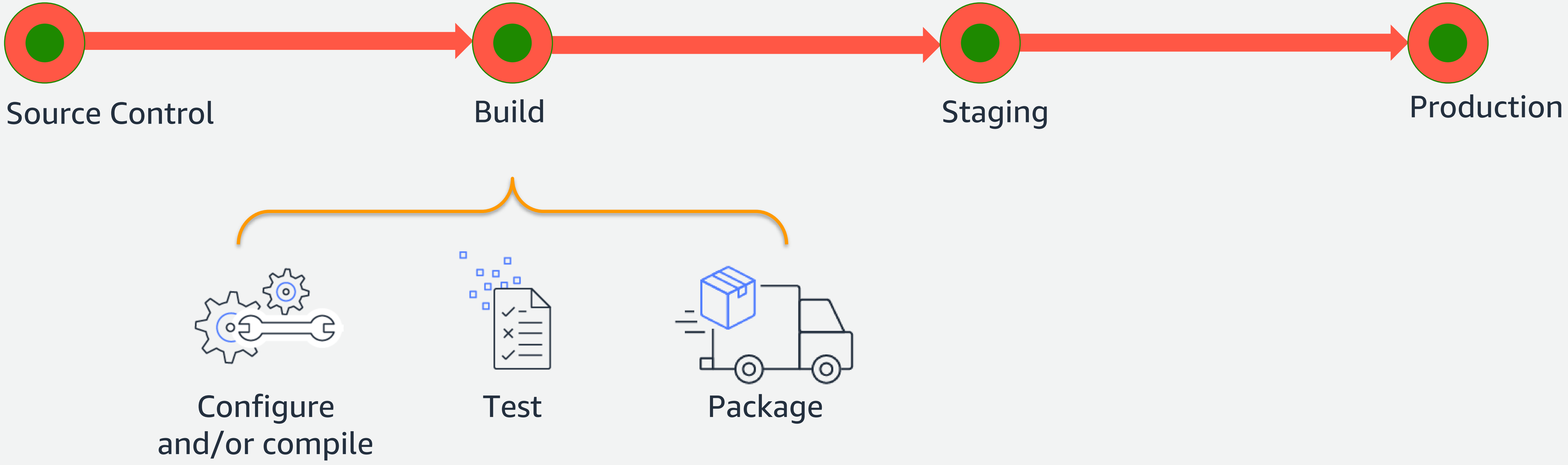
3

The change set is executed to create or update a stack

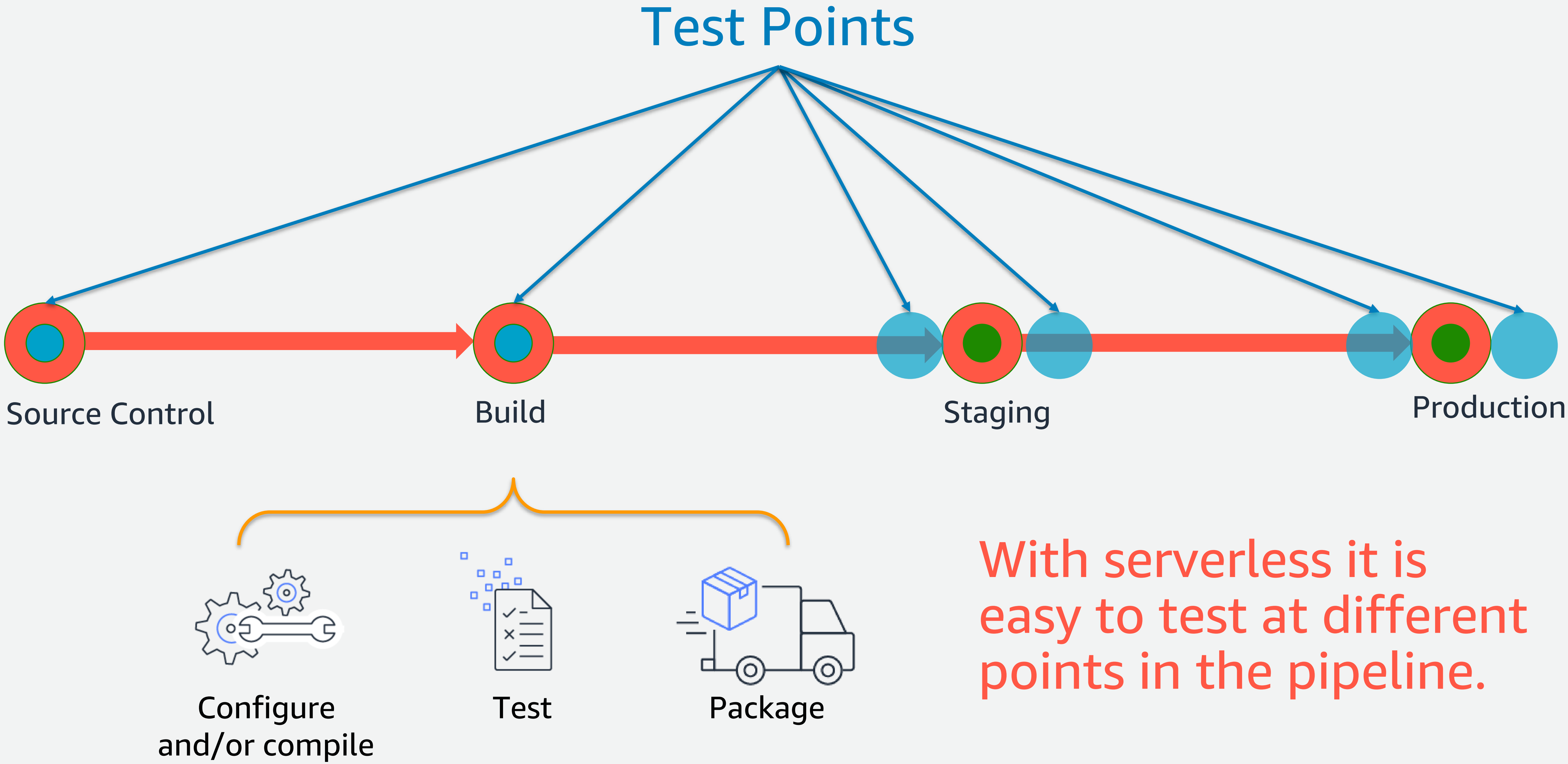
Testing

The pipeline: testing

The build phase is a common place for testing.



The pipeline: testing



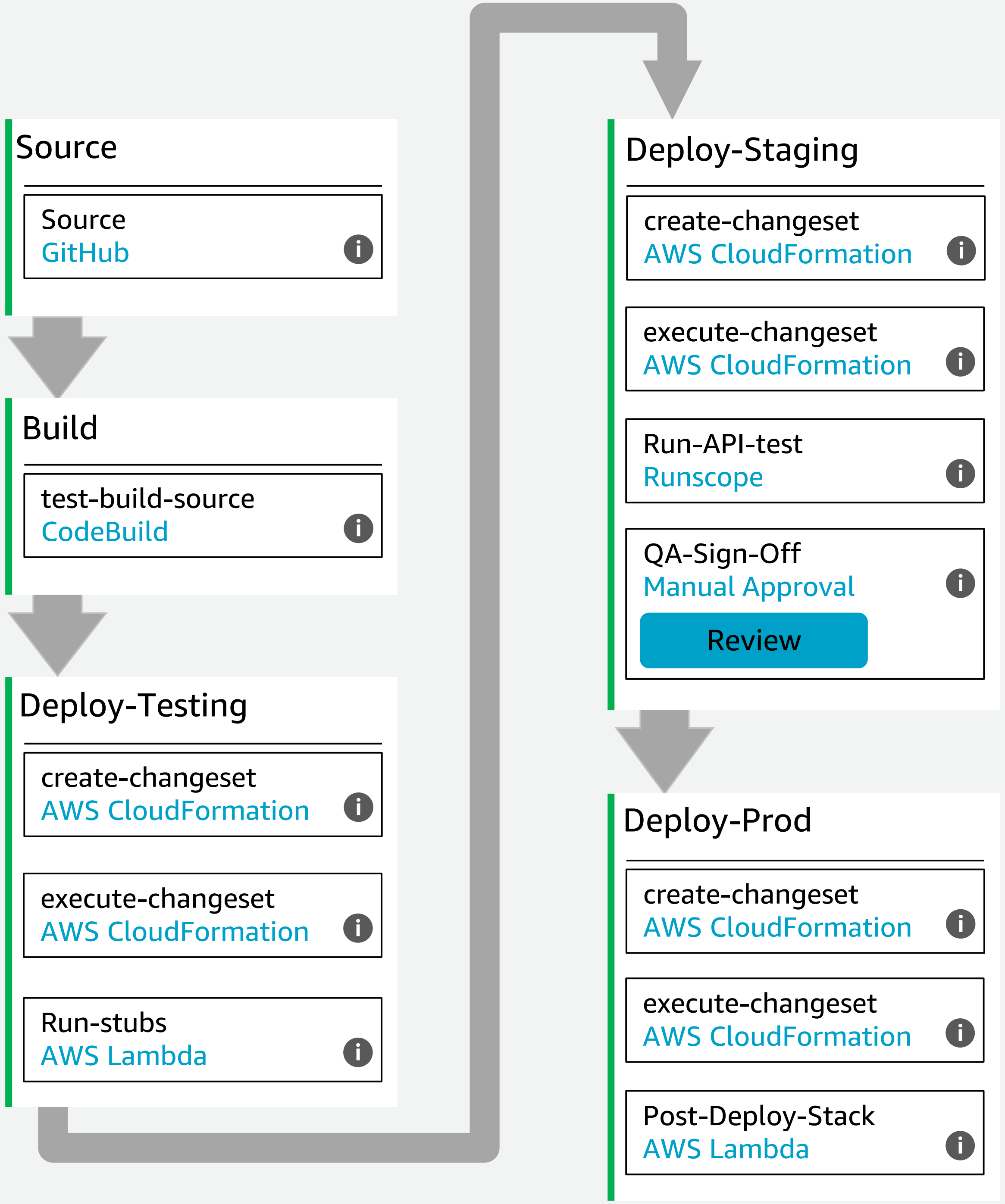
With serverless it is easy to test at different points in the pipeline.

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



Source

Source
GitHub

Build

test-build-source
CodeBuild

Deploy-Testing

create-changeset
AWS CloudFormation

execute-changeset
AWS CloudFormation

Run-stubs
AWS Lambda

Deploy-Staging

create-changeset
AWS CloudFormation

execute-changeset
AWS CloudFormation

Run-API-test
Runscope

QA-Sign-Off
Manual Approval

Deploy-Prod

create-changeset
AWS CloudFormation

execute-changeset
AWS CloudFormation

Post-Deploy-Stack
AWS Lambda

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

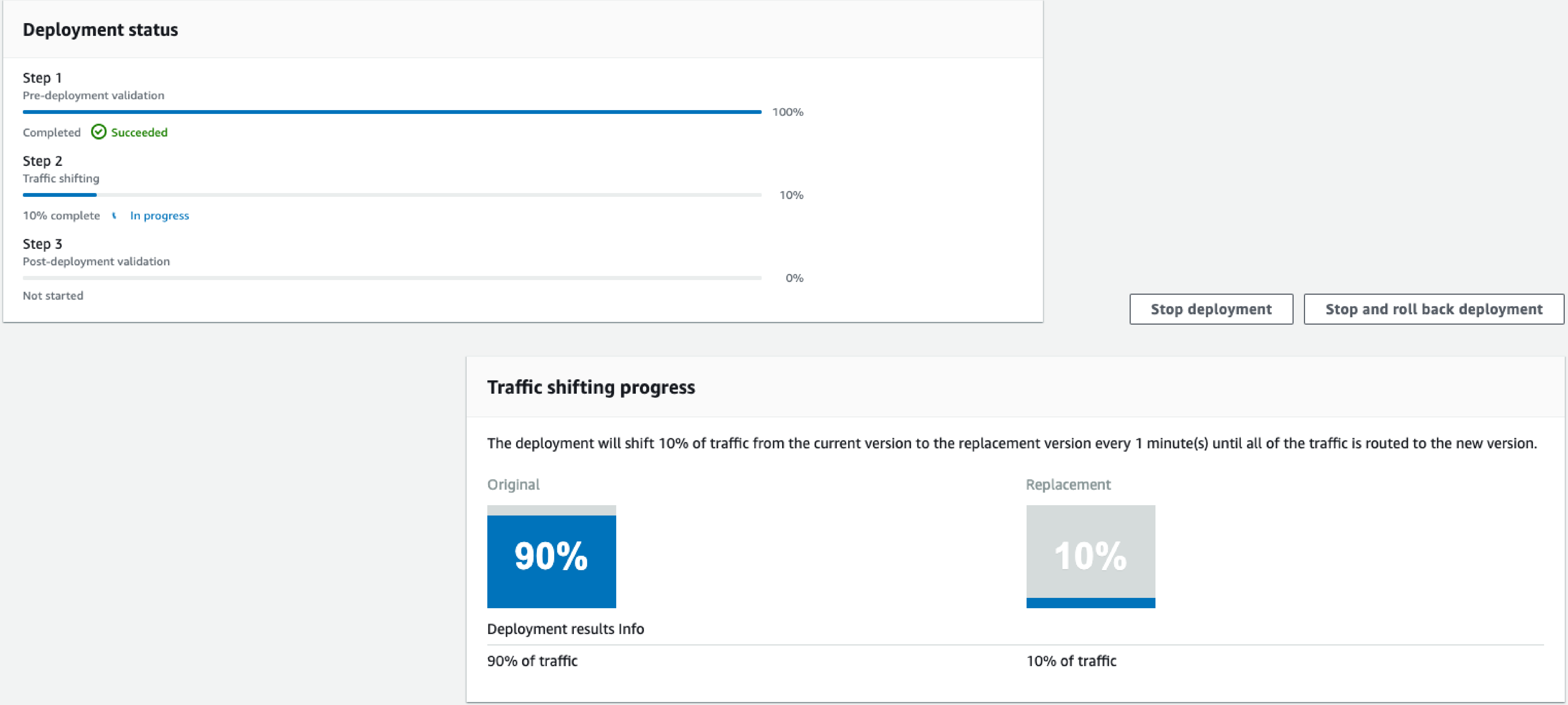
- Run pre-traffic Lambda tests
- Deploy canaries
- Complete wait period successfully
- Deploy 100%
- Run post-traffic Lambda tests

Testing using safe deployments

```yaml
MyLambdaFunction:
  Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs12.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
```

The deployment

d-45W1IYDA3



Deployment status

Step 1
Pre-deployment validation
100%
Completed ✓ Succeeded

Step 2
Traffic shifting
10%
10% complete In progress

Step 3
Post-deployment validation
0%
Not started

Traffic shifting progress

The deployment will shift 10% of traffic from the current version to the replacement version every 1 minute(s) until all of the traffic is routed to the new version.

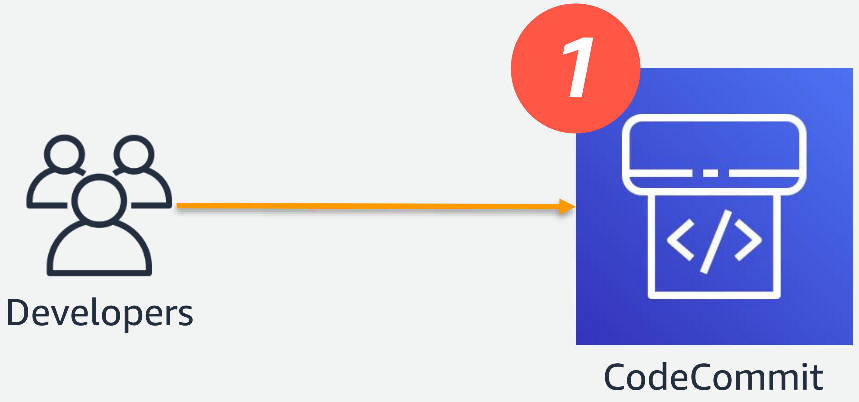
Original	Replacement
90%	10%

Deployment results Info

90% of traffic	10% of traffic

By the numbers

By the numbers



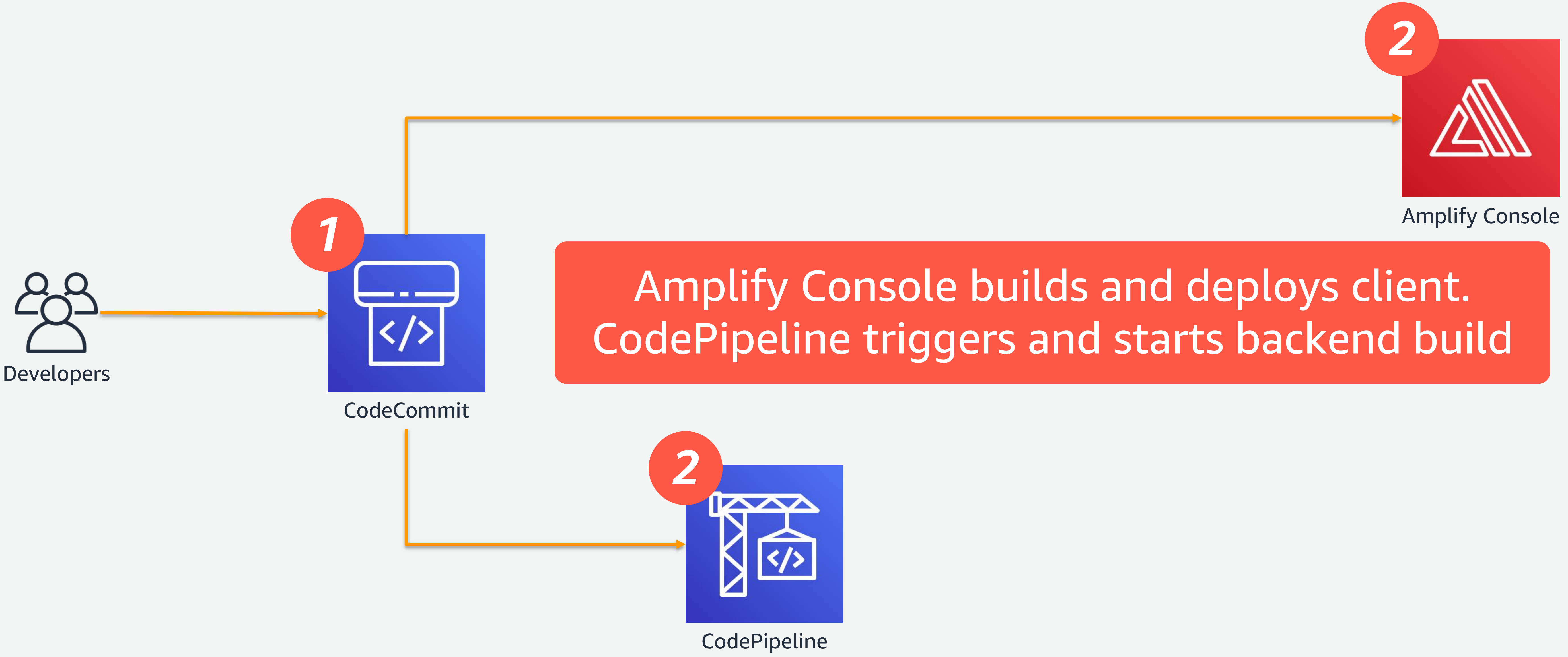
1

Developers

CodeCommit

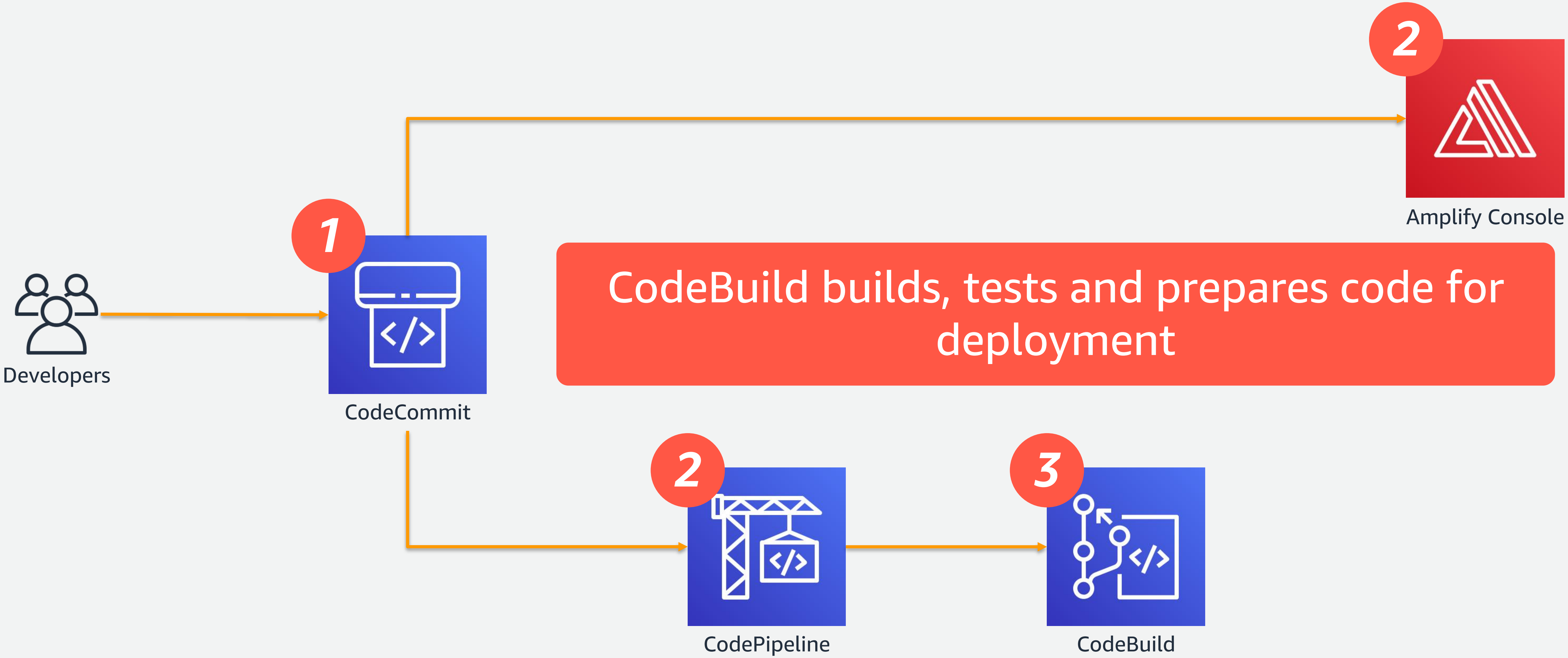
Developers commit code to repository

By the numbers



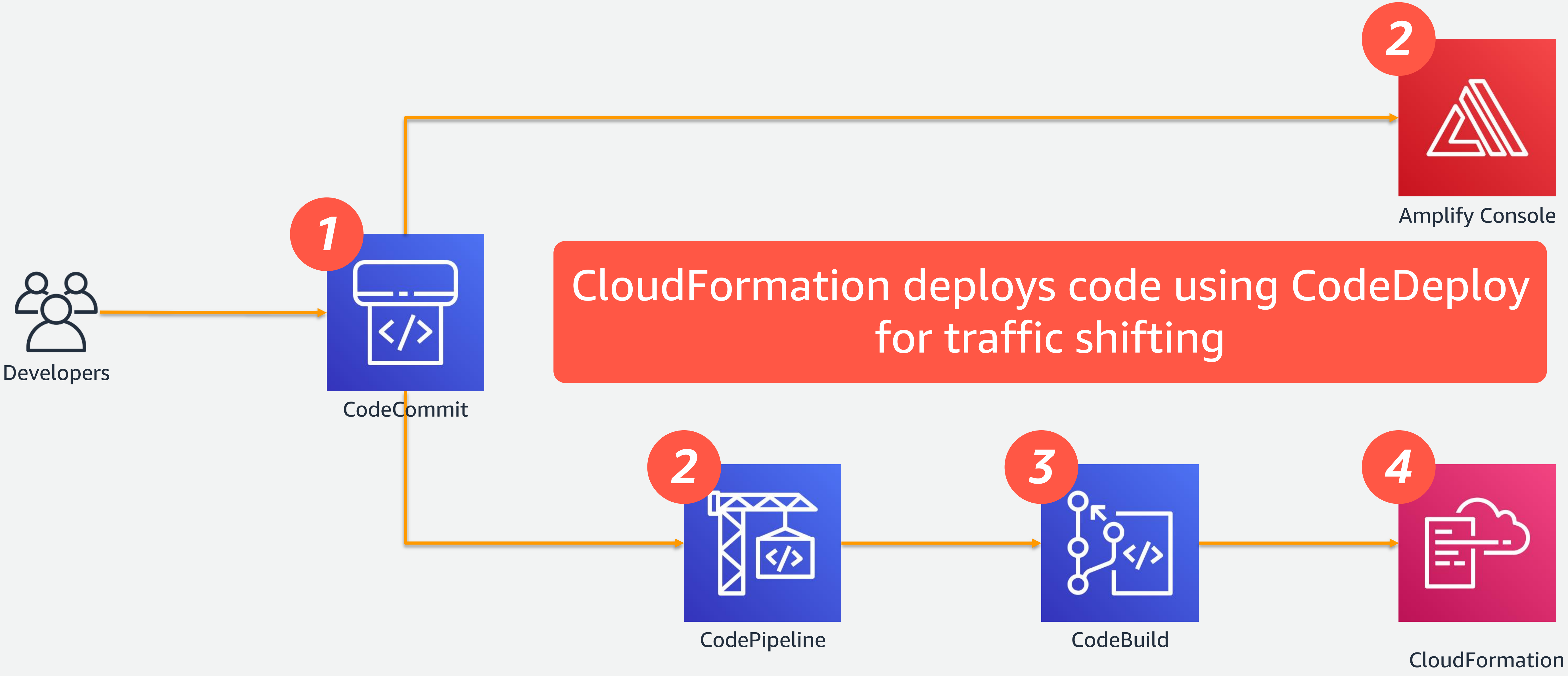
Amplify Console builds and deploys client. CodePipeline triggers and starts backend build

By the numbers



CodeBuild builds, tests and prepares code for deployment

By the numbers

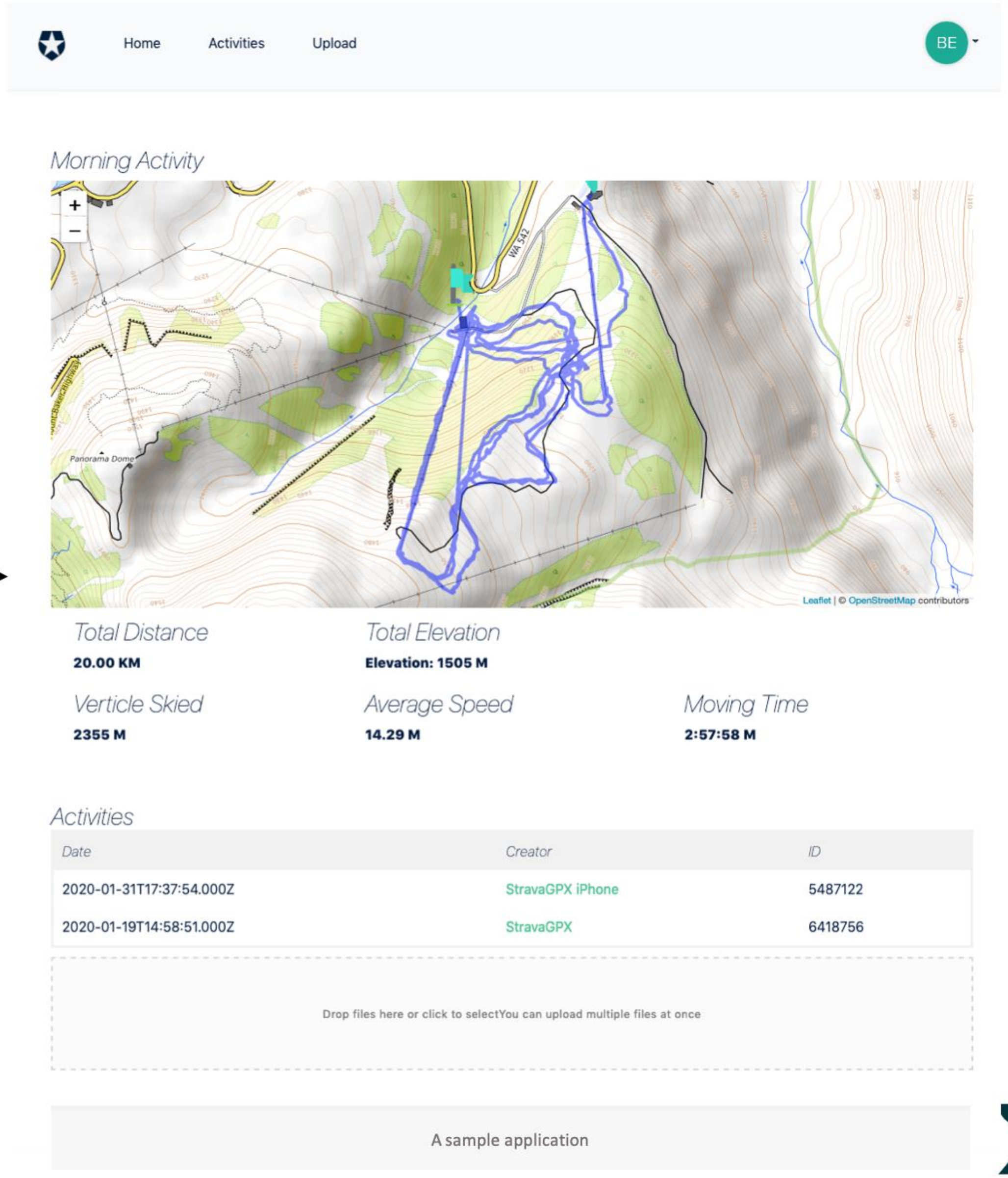


CloudFormation deploys code using CodeDeploy for traffic shifting

Fresh Tracks deployed

Fresh Tracks

A new sample application that demonstrates how to connect to multiple 3P SaaS providers, eliminate the data sprawl that occurs when using multiple SaaS integrations and automate workloads between them.



Home Activities Upload BE

Morning Activity

Total Distance	Total Elevation	
20.00 KM	Elevation: 1505 M	
Verticle Skied	Average Speed	Moving Time
2355 M	14.29 M	2:57:58 M

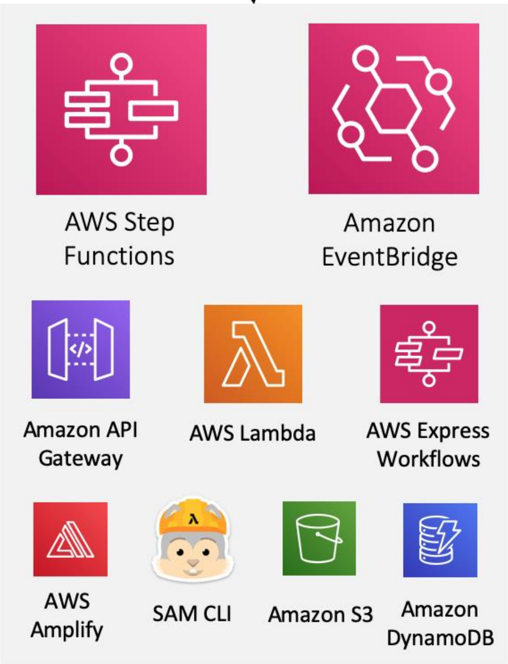
Activities

Date	Creator	ID
2020-01-31T17:37:54.000Z	StravaGPX iPhone	5487122
2020-01-19T14:58:51.000Z	StravaGPX	6418756

Drop files here or click to selectYou can upload multiple files at once

A sample application

Built with these AWS services



AWS Step Functions

Amazon EventBridge

Amazon API Gateway

AWS Lambda

AWS Express Workflows

AWS Amplify

SAM CLI

Amazon S3

Amazon DynamoDB

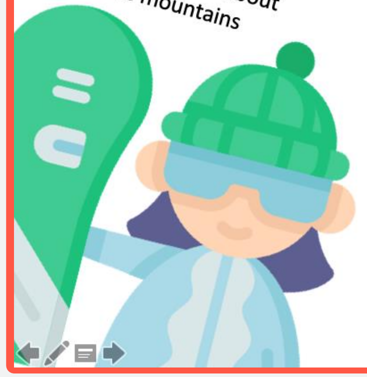
To automate these SaaS integration partners

zendesk

Auth0

DATADOG

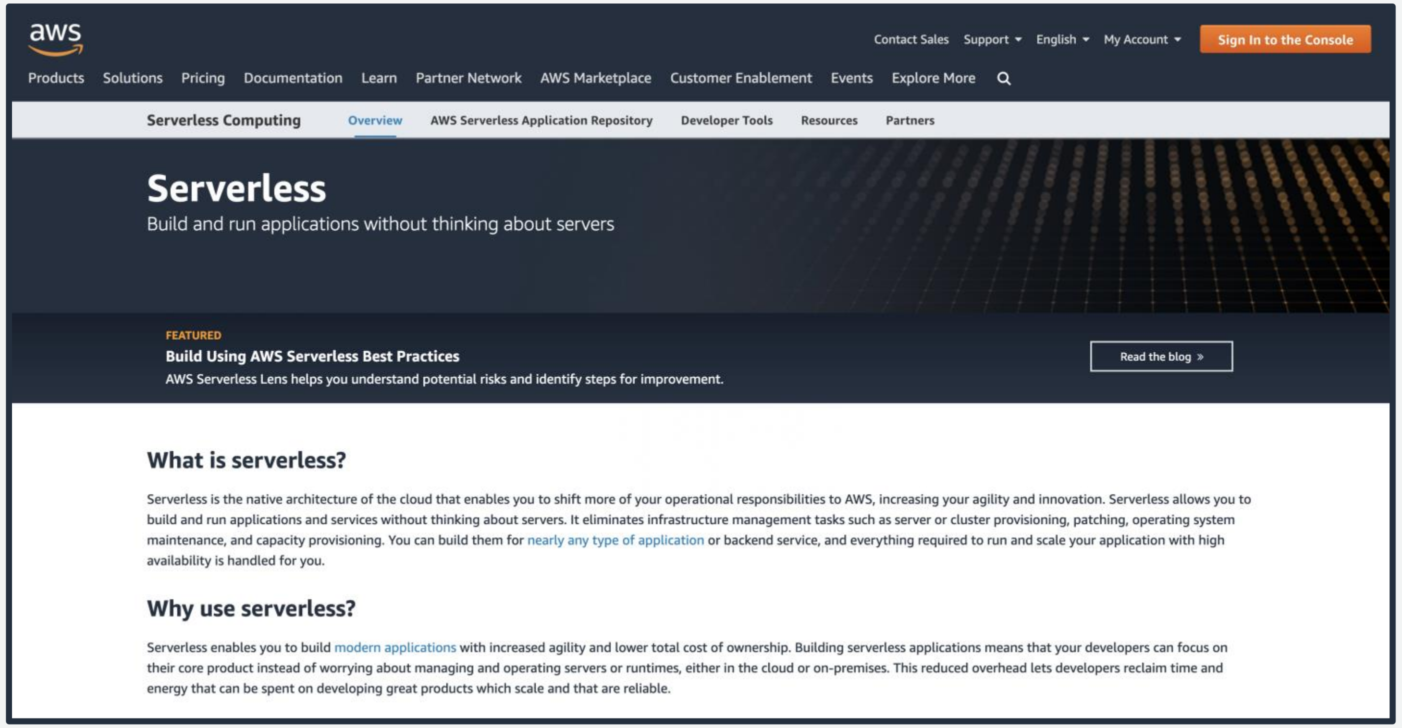
..And it's all about the mountains



© 2020, Amazon Web Services, Inc. or its Affiliates.

CI/CD Partners

Final resources



Serverless

Build and run applications without thinking about servers

FEATURED

Build Using AWS Serverless Best Practices

AWS Serverless Lens helps you understand potential risks and identify steps for improvement.

What is serverless?

Serverless is the native architecture of the cloud that enables you to shift more of your operational responsibilities to AWS, increasing your agility and innovation. Serverless allows you to build and run applications and services without thinking about servers. It eliminates infrastructure management tasks such as server or cluster provisioning, patching, operating system maintenance, and capacity provisioning. You can build them for nearly any type of application or backend service, and everything required to run and scale your application with high availability is handled for you.

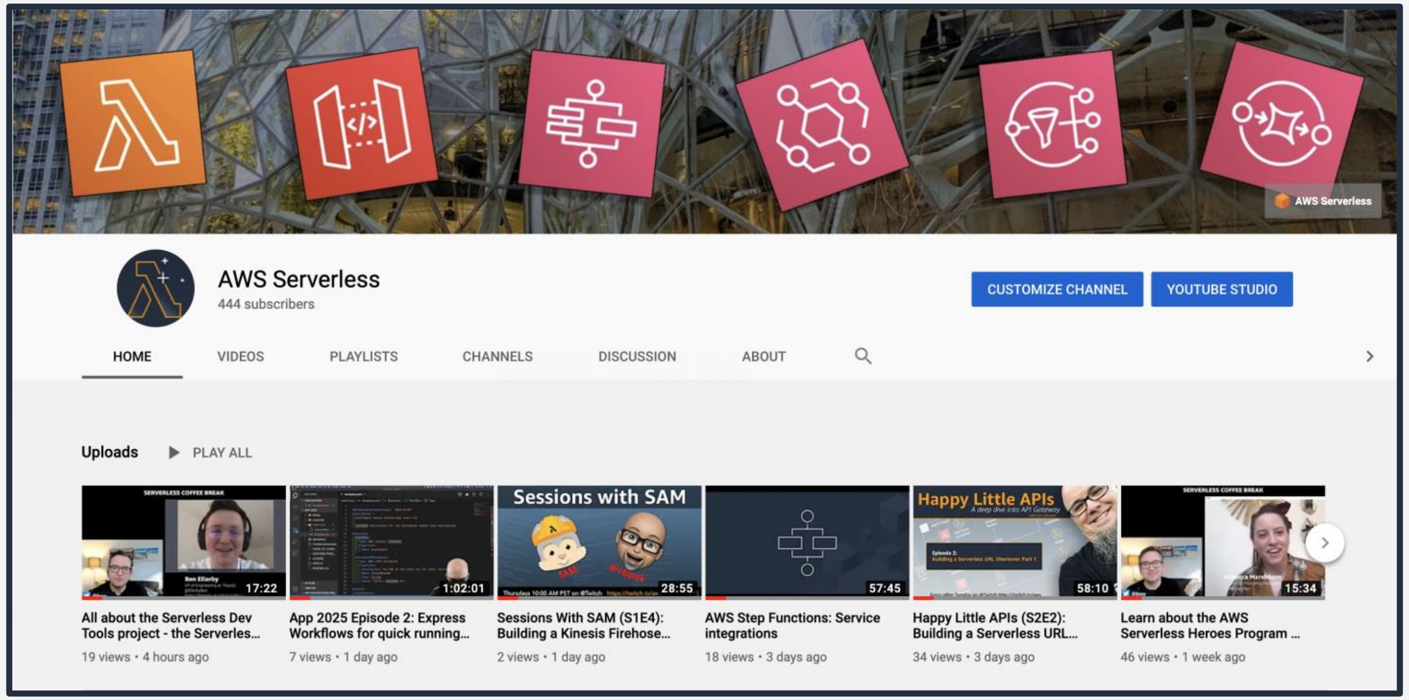
Why use serverless?

Serverless enables you to build modern applications with increased agility and lower total cost of ownership. Building serverless applications means that your developers can focus on their core product instead of worrying about managing and operating servers or runtimes, either in the cloud or on-premises. This reduced overhead lets developers reclaim time and energy that can be spent on developing great products which scale and that are reliable.

AWS Serverless

slip.link/aws-serverless

Final resources



All about the Serverless Dev Tools project - the Serverles... 19 views • 4 hours ago

App 2025 Episode 2: Express Workflows for quick running... 7 views • 1 day ago

Sessions With SAM (S1E4): Building a Kinesis Firehose... 2 views • 1 day ago

AWS Step Functions: Service integrations 18 views • 3 days ago

Happy Little APIs (S2E2): Building a Serverless URL... 34 views • 3 days ago

Learn about the AWS Serverless Heroes Program ... 46 views • 1 week ago

AWS Serverless YouTube Channel

slip.link/serverless

Thank You!

@edjgeek