URL Shortener

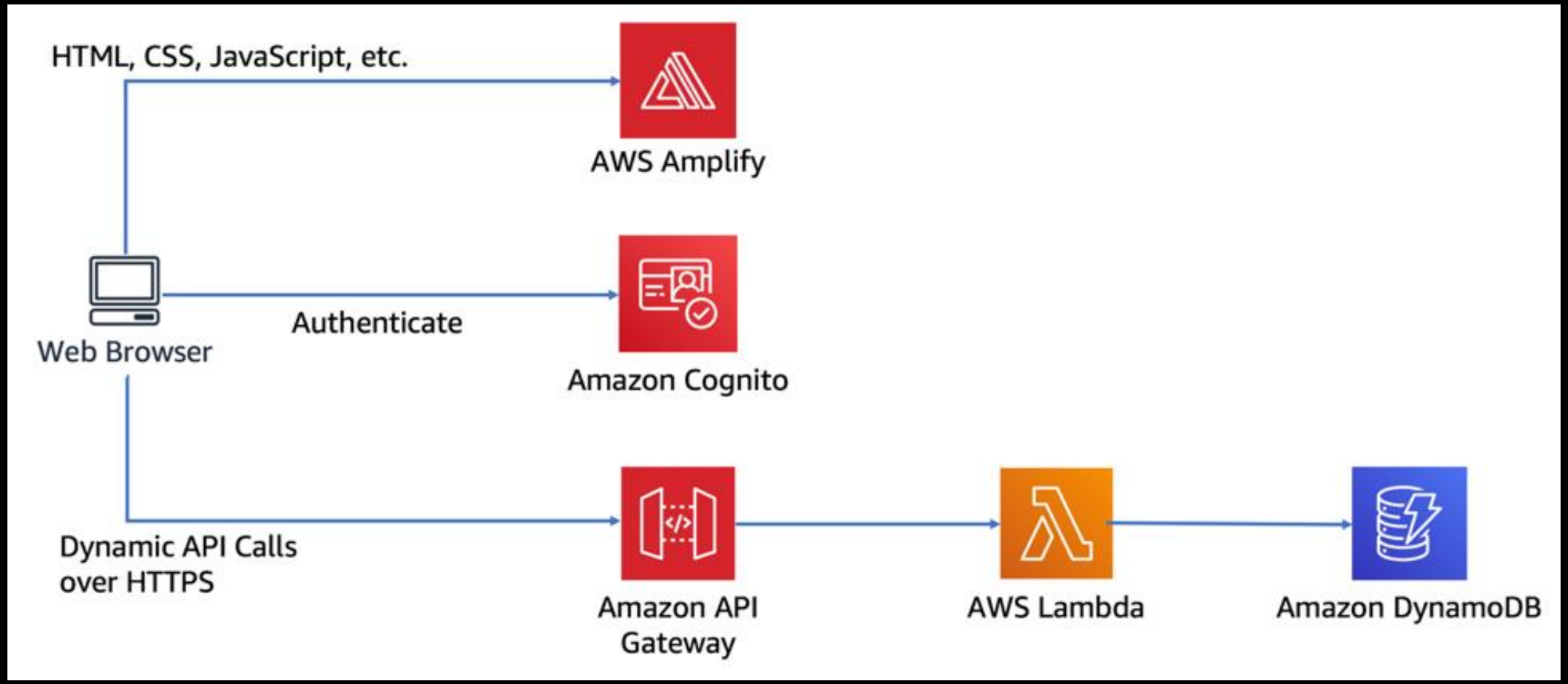
Noah H. Liu

Serverless

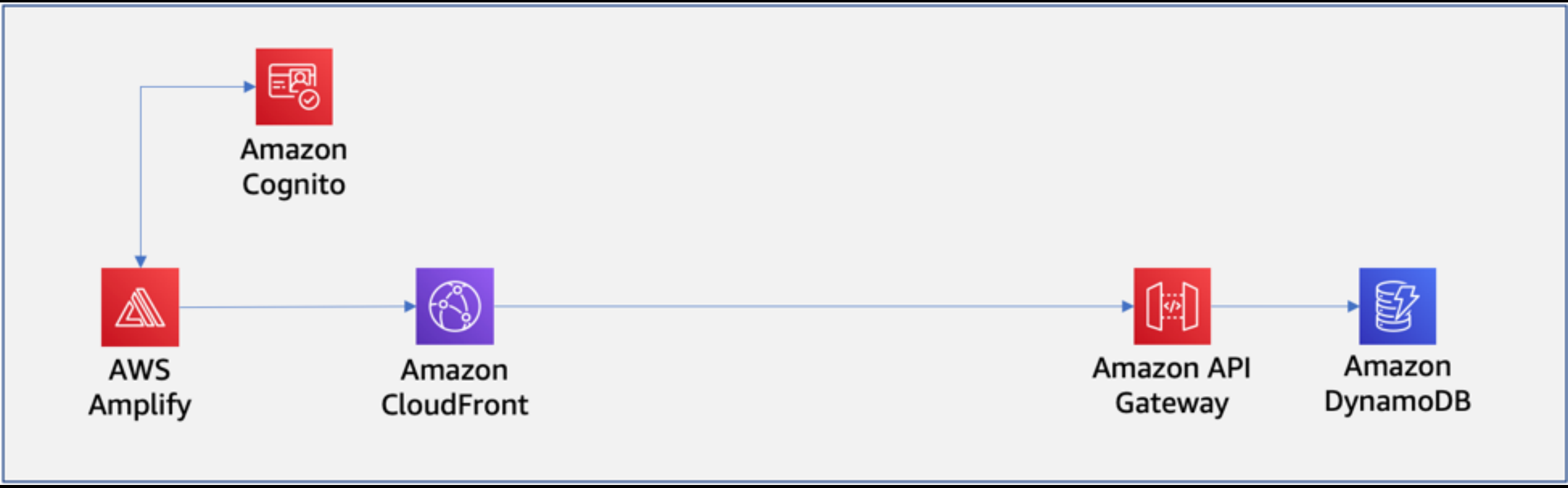
Build and run applications without thinking about servers

- Serverless technologies feature automatic scaling, built-in high availability, and a pay-for-use billing model to increase agility and optimize costs.
- Move from idea to market, faster
- Lower the costs
- Adapt at scale
- Build better applications, easier

My first idea: Build a Serverless Web Application



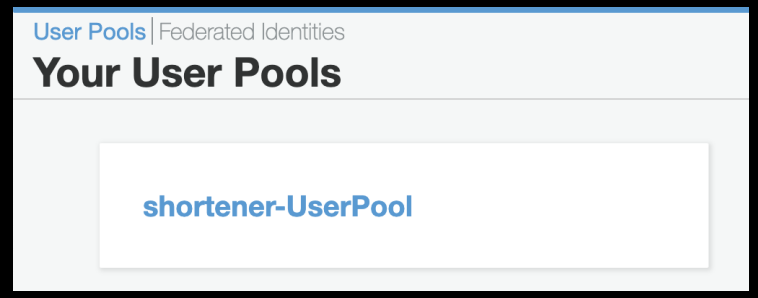
Architecture Overview

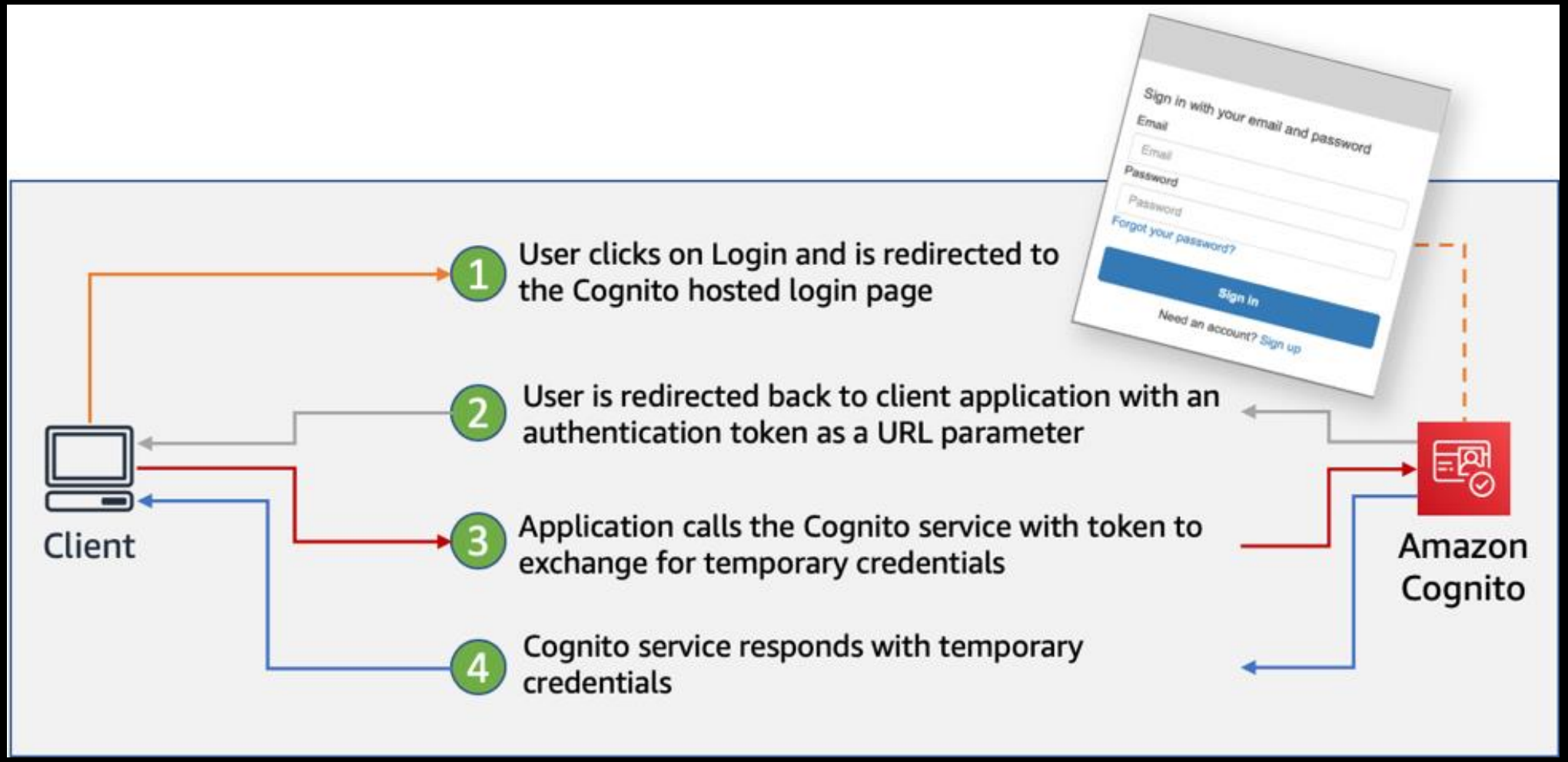


Application Architecture

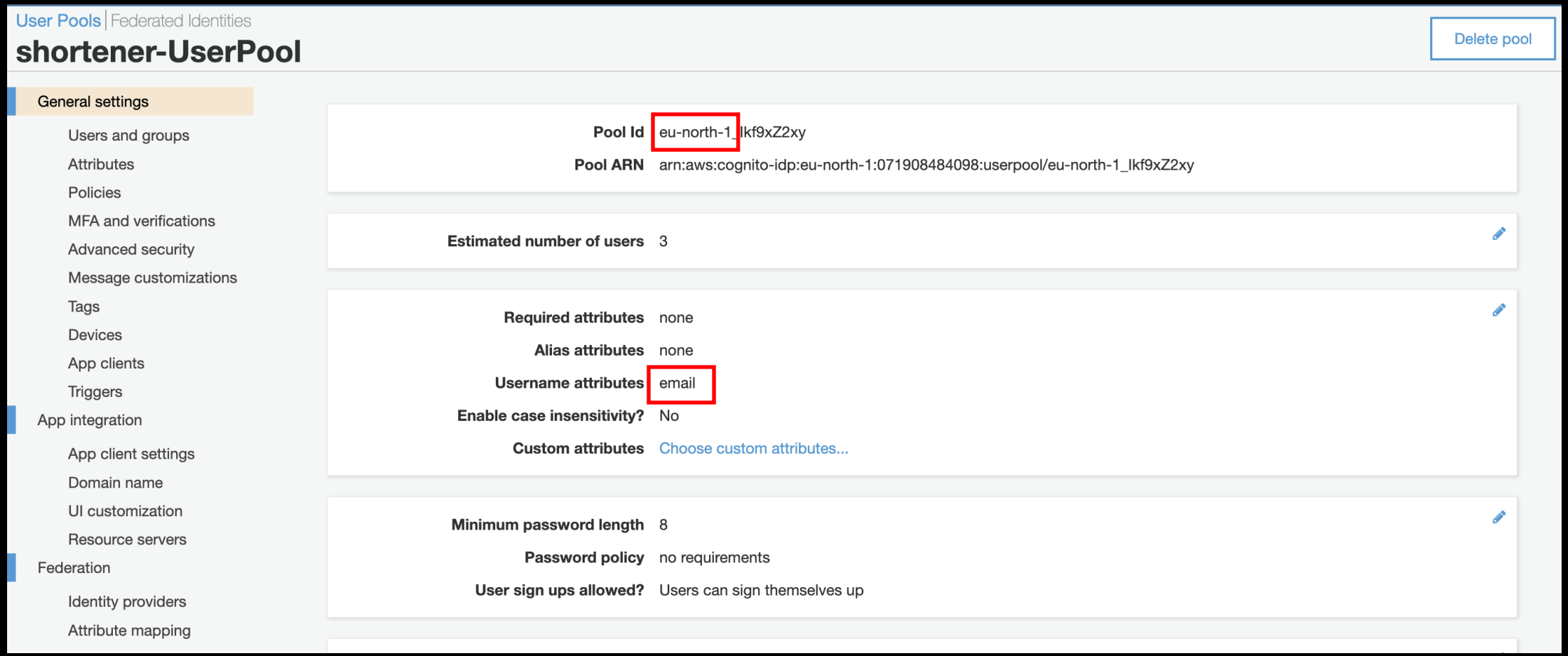
- My application architecture uses Amazon API Gateway, Amazon DynamoDB, Amazon Cognito, and AWS Amplify Console.
- Amplify Console provides continuous deployment and hosting of the static web resources including HTML, CSS, JavaScript, and image files which are loaded in the user's browser.
- JavaScript executed in the browser sends and receives data from a public backend API built using API Gateway.
- Amazon Cognito provides user management and authentication functions to secure the backend API.
- Finally, DynamoDB provides a persistence layer where data can be stored.

Cognito for authentication

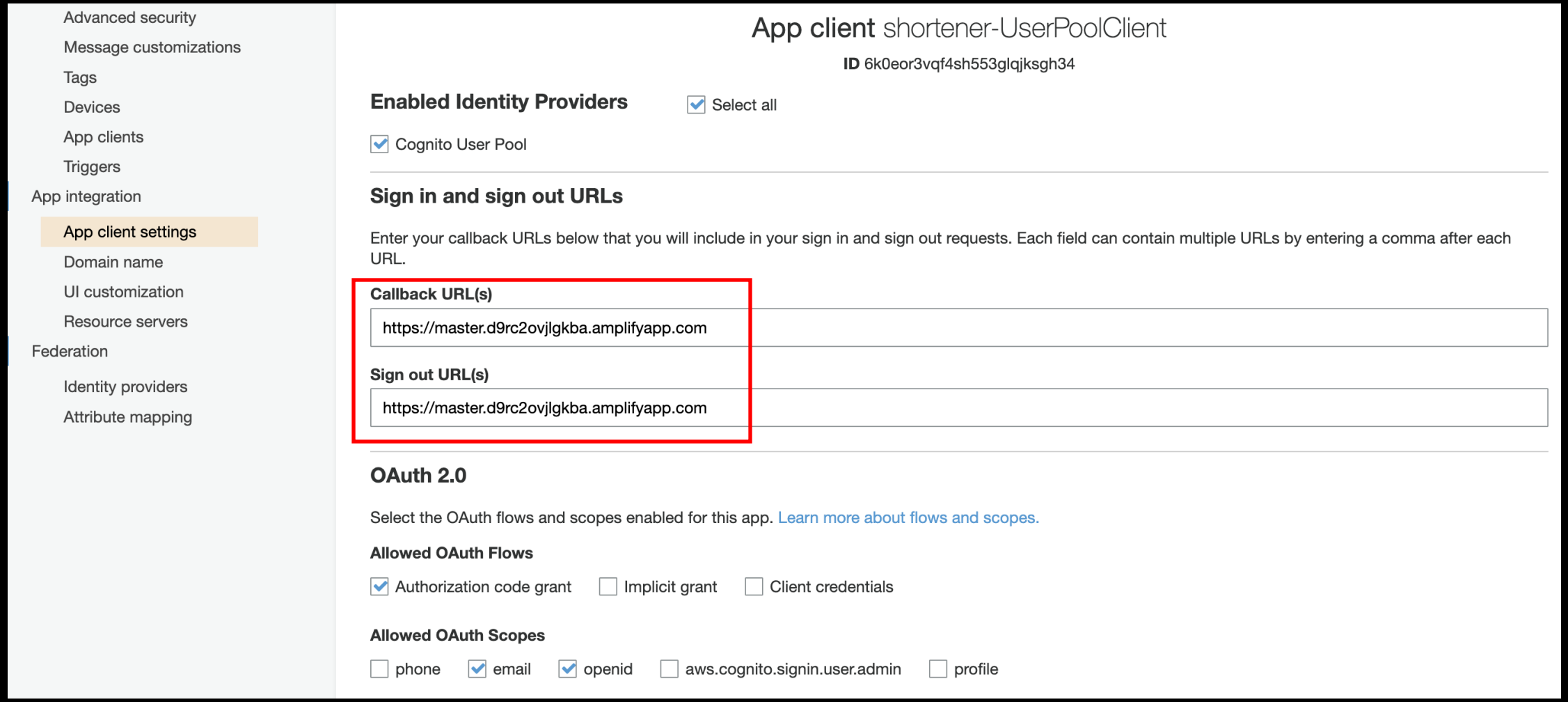




Cognito User Pool



Cognito User Pool App Client



Configure the client for secure requests

The last step for authorized requests is to configure the client. As explained above, the client interacts with Amazon Cognito to authenticate and obtain temporary credentials. The truncated temporary credentials follow the format:

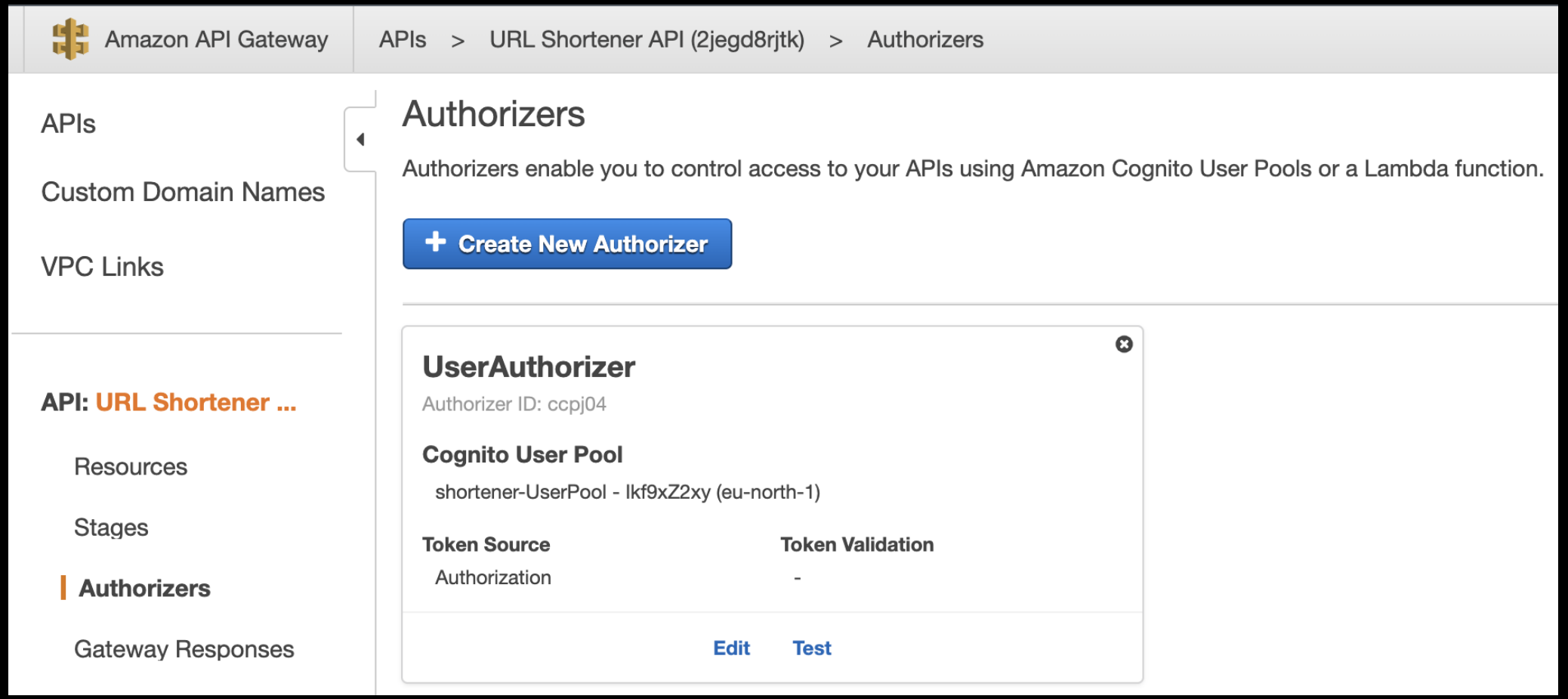

```
{
  "id_token": "eyJraWQiOiJnZ0pJZzBEV3F4SVUwZngreklE…",
  "access_token": "eyJraWQiOiJydVVHemFuYjJ0VlZicnV1…",
  "refresh_token": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU…",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

Temporary Credentials

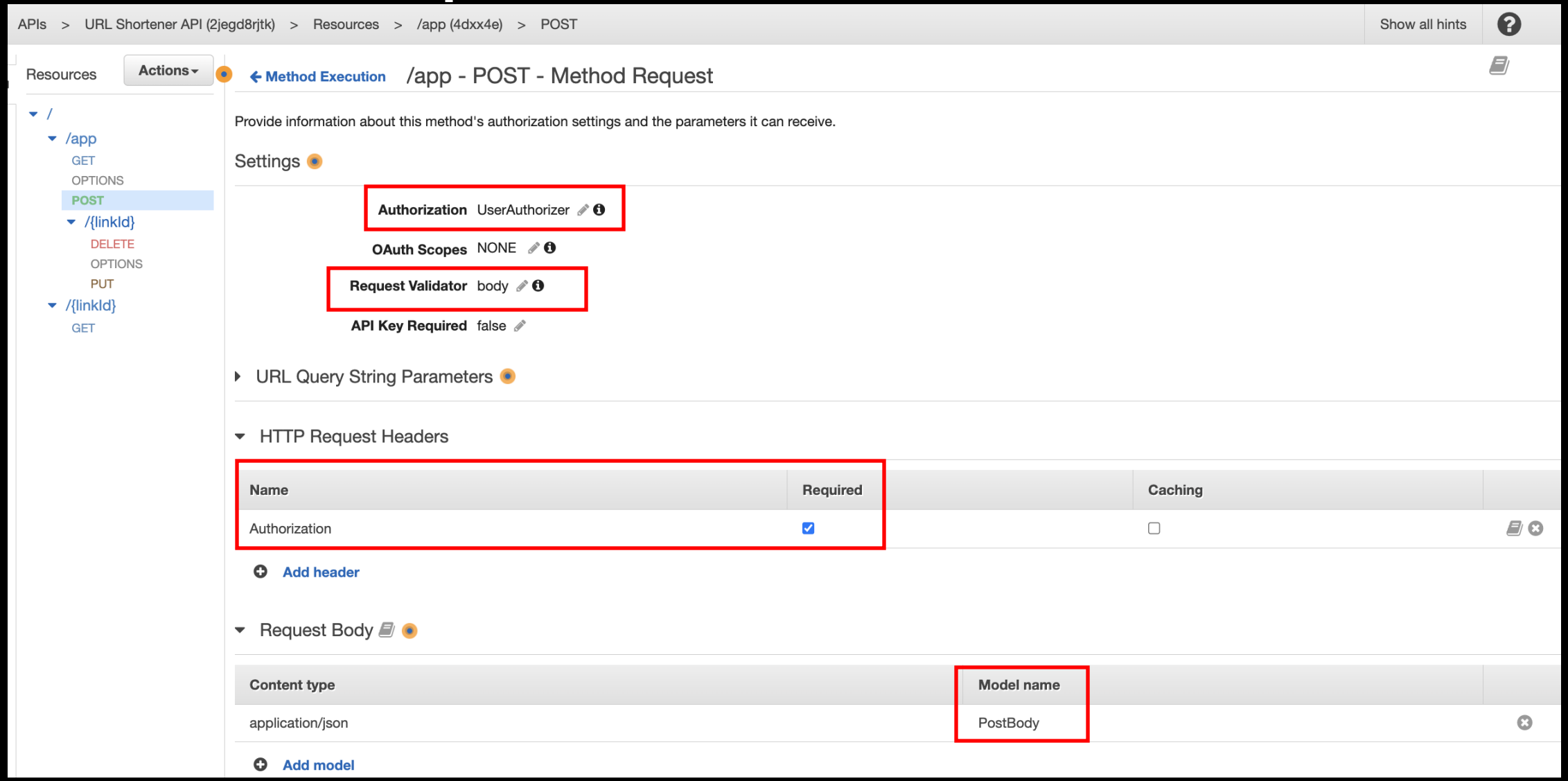
aws cognito-idp admin-respond-to-auth-challenge --user-pool-id --client-id ⌥⌘1


```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "eyJraWQiOiJtV0Q0R2J0aW9RdGY1TWRwVVlmcExxTFZKWjNPdzlueXdvbWZJNmMwaXZJPSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI2NzEzMWUzMy0zMGNhLTQ4ODYtOTQ1My0wNDAyYTZkMTY1OWQiLCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAuZXUtbm9ydGgtMS5hbWF6b25hd3MuY29tXC9ldS1ub3J0aC0xX0lrZjl4WjJ4eSIsImNsaWVudF9pZCI6IjZrMGVvcjN2cWY0c2g1NTNnbHFqa3NnaDM0Iiwib3JpZ2luX2p0aSI6ImY2N2ZjZTJkLTM0NzItNDhkYy1iMTNlLTUzYzc5MmI3ZWE0OSIsImV2ZW50X2lkIjoiNGU0NjNlOWUtZWMyOS00NjFhLTk0YmEtYWFkMDA2ZTAxZThmIiwidG9rZW5fdXNlIjoiYWNjZXNzIiwic2NvcGUiOiJhd3MuY29nbml0by5zaWduaW4udXNlci5hZG1pbiIsImF1dGhfdGltZSI6MTY1NjAxMzIxMywiZXhwIjoxNjU2MDE2ODEzLCJpYXQiOjE2NTYwMTMyMTMsImp0aSI6ImY0MWM2OWYzLTg2NWMtNGU0MS1hZTI1LWM4MzZhMjcyN2UzOSIsInVzZXJuYW1lIjoiNjcxMzFlMzMtMzBjYS00ODg2LTk0NTMtMDQwMmE2ZDE2NTlkIn0.U8YNJCKJZrNtczxZwd0QfR7LDocb4sWZFXrKLzRIdJzeoXuEXkWI0SE0i70z0QoXD0BXlgEdScUmnZT4TvmqTOynm-yYX0682FSl1HJjxXgETYcgFpf_O0UzM-AnfKH4wO0j5YXcmpeeS0gg8qQVnkNiXLCxM4-1O8WfHYJD16WdNIvuMKGfdEJzbnCvyKgfgN1vrHN4dDOmTn7jtSAD60SvDgynMJ3q1E_I_70z2O2OgYeIeunKzGArEJEWrZlpQbFoyff5km6NI3laV-3c9Gr5gujTUmXWdng5wpsHlU3b0HjaR7NkvyuS2STOL3pWZlQPabMPu__FRiC0C8cdtw",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.VQPaINK6HveDMubJUP8sIqAx3UulCk4DmkRx78onhrCUaU1YhuaM8mHS0dVKUWKE5e2HR91utbewdZuE4A6r4zauFIR9erTfzyvsQGXpPE15qcYhYsEz73oVmdjH9AknLROeXEhtY__UdGJp8zPD7sG7UeNrP1vQC8RZTwruDbdIOJYZDYU2e8Isvc06UPWeBaTMOBS-LBroczNCVtkqiQKxyxu-MBBH5Vo5WHchhi-tg122wNoM2y4fGO2EO9ctiqNgxFllmVFyfmManab_AYIIuj0ElQ8jbg83-f-rcIBQKiSEwgbJ93HN2jF_EI-U2Vb4Mqf4eCmS5LQ0iFlYXQ.8vh3Nlr-WgI70Pve.dbvPoPQm34o9cn7QM7bI4n5kWn7ErfB2shbroXd-o7asBCMyS0EREWZTiKfuKZQTxI3P6C91WtNT4vmRAmvlSt2aPLzCsgusyflq-bz7fKa6YkQz83BtY1SwrF5EMHVDotYqcijYKjrtgWr08aot1Vy-mqBZV_idoFsk8IDgi2OPN92CrccO-WX6x96aIdHXUTVscITqrTq9gCoz9zZQ0wwJnPcZ7gHdyZLksoPZu4-pbiOGO5Bo2BaP4wagCJQjhrHX9I22lqIsdS8jOKghXlz5Q4uA2W2ESZrGtir73uHPjp4kk2BDVnaNXcbtTTeVq3vXUqRLGmHj4qTuB_yB5MVP5AJv66GHETaNxDAVlNVBEl6CmBf0oYiOvNnuV7MdKSFb0xHb3nzNAkElpvYnjZvxHEF7Ho5jBMUokzAEoBdS9ef7V1Jcdu1VNlTWG4OpSsUJ41_nwmF-8VX_2dEkAdLOTOAiuEVHNq4fD3OvLWa0DHlsKPa6AbVmjKbhX8ZcmP77Y__YIEeHfn9fgniG7Qz2NjA-OFm5tEYKDY6DYURRU89PS2c8pxXthTEMd7a0OaDo4xG1bccq1e-2HN8LB75siR4C1hTITwIEHWyWPSR67gTLC66oQ19tR4W8YCRELmf6LZsSFqWTCZ1IUM_fXmEVlEbcRddkvLQmUBmKpn7GCOK_7EvVZjgPEfCOnY_a5mhm2IFrzsyxNWWDNb5GYZ2LvY4UQD9n1Yqm7WtBZ3RTiGVdLciWY8SJdqBH1oz6ECyumNO1TIpXT69D_j9nUwvQwdU0WPUYdcwI9kVRR_7wizjxBEhLkOD4oYzLjnYCX6t6DLffL2uYN6CkVtds3hZYoMdCV8iqTU33-JmbAYi0x_L9DhGB1Wzsy4wiHkG_QpSpm15vV_YndreexVwi71E_MoiLv6_7sxQyuMcHb-GmGtdGi2kRVCG5SM8KNC2ATBDbZEpi5svz3nrS4TU3tuME_keNktppoNmrPT_VdTo7Vv7RBEvBoIuALb3mf6DIUgiwLaVEJSyiv6lcrZ6tqtRE16I6feD8UmQxbLNsdKe7G4-GQT3CtUSgpBsaXIhfNWECPvbVuMTpy-yf3sY3NTgrjSZQMqui2NCJ-dW3jSm5KLCwda2FDZlmQP1G3dcGlkC1hn3DVGfBA3Sgx4jVdbeDWMINtya-NxVB6GOQfWyYWk10yaOZee53LAW-M6ZF0OKKowiEM7PLw8va3LSxiP0IMXikWb4Kv6eIMFqTCNnGNdFeQJD6kA9g7tozp446970Eeqv7BGHI3JkEzxVTVOu_gqENa4XMLmJ03iPO-tciqGkkL2WOVj25A_3rtJc.IUfoymlYj1S1zN4VbHPevw",
        "IdToken": "eyJraWQiOiJXQlBXdjZkZU45cmRjblpmdm0xMko4RithNnc4TmdKbmZZTUNHTnZsWWdzPSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI2NzEzMWUzMy0zMGNhLTQ4ODYtOTQ1My0wNDAyYTZkMTY1OWQiLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwiaXNzIjoiaHR0cHM6XC9cL2NvZ25pdG8taWRwLmV1LW5vcnRoLTEuYW1hem9uYXdzLmNvbVwvZXUtbm9ydGgtMV9Ja2Y5eFoyeHkiLCJwaG9uZV9udW1iZXJfdmVyaWZpZWQiOnRydWUsImNvZ25pdG86dXNlcm5hbWUiOiI2NzEzMWUzMy0zMGNhLTQ4ODYtOTQ1My0wNDAyYTZkMTY1OWQiLCJvcmlnaW5fanRpIjoiZjY3ZmNlMmQtMzQ3Mi00OGRjLWIxM2UtNTNjNzkyYjdlYTQ5IiwiYXVkIjoiNmswZW9yM3ZxZjRzaDU1M2dscWprc2doMzQiLCJldmVudF9pZCI6IjRlNDYzZTllLWVjMjktNDYxYS05NGJhLWFhZDAwNmUwMWU4ZiIsInRva2VuX3VzZSI6ImlkIiwiYXV0aF90aW1lIjoxNjU2MDEzMjEzLCJwaG9uZV9udW1iZXIiOiIrNDkxNjM3NjQwODI3IiwiZXhwIjoxNjU2MDE2ODEzLCJpYXQiOjE2NTYwMTMyMTMsImp0aSI6ImViOWYwMmQ4LTA4ZGUtNGZiZS1iYzI2LWYxMzcxNGUyNzBjOSIsImVtYWlsIjoiYm9sZWRhZGlhbkBnbWFpbC5jb20ifQ.AbojB4j7ftXwy2KjpA0uxu5cnh5-Y3h3_03U-NWdoERauP9bP0uj6rRmgJpFGZDE-eiXLsTIAKH_oOiabGA2oSwfxivZsQbv50104IYjOinEB_7Hp4DCrN6GSYX8jZ4O65AtN50ldmAN-Mj0SVrTtidqUfi2nXni03vqpIBM6qWXL1K6l9eJVP60Zpb_tGQlGiiMpxHcRkHHJ6W0BzVQlLB-lr1GnDWI-cRimFfXz8zG4eoXj47RUzUiOKv2NE7zLe1PeCN8ah16kudA83QtMdSBlXXdGJpyR0bAW7Mnbq7XBFwr-VKoDxHqxsqT9-2ekaibo4ZgbuZyTPlmj9Jblg"
    }
}
```

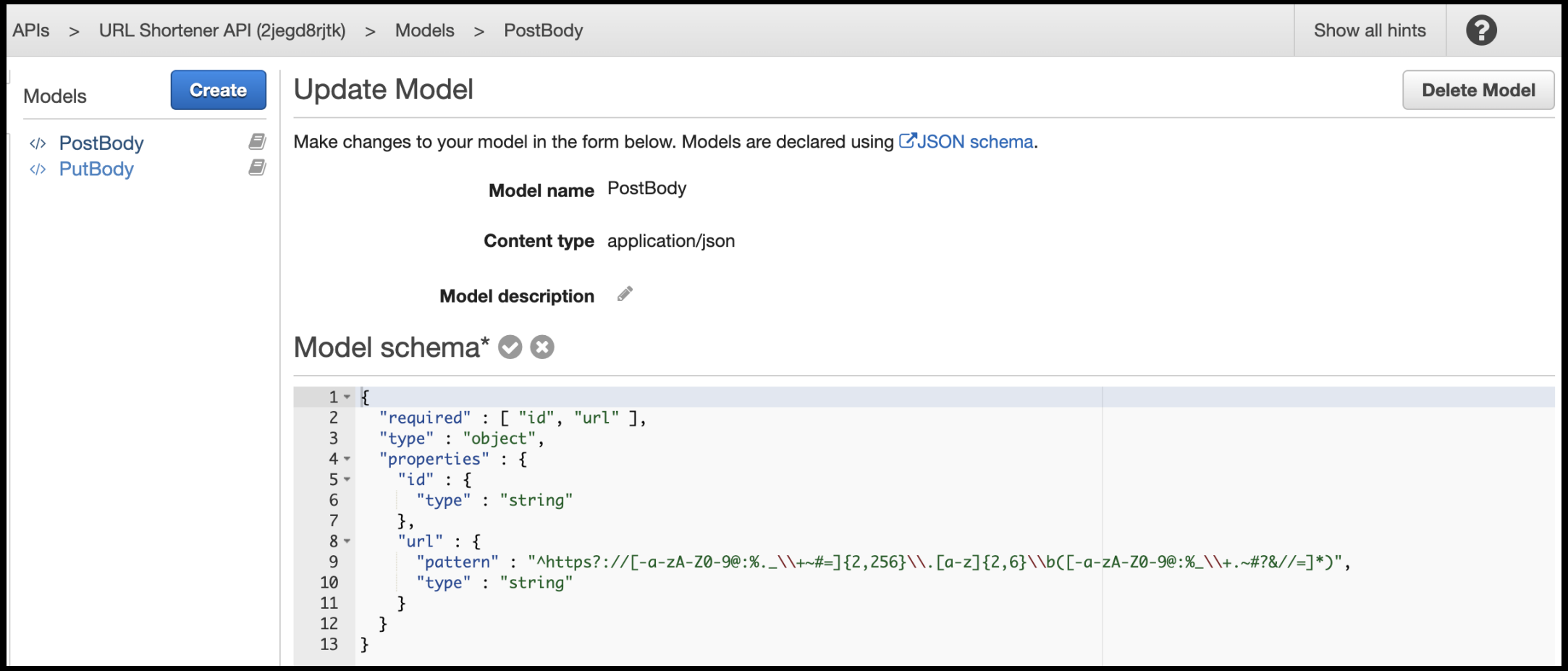
Cognito Authorizer



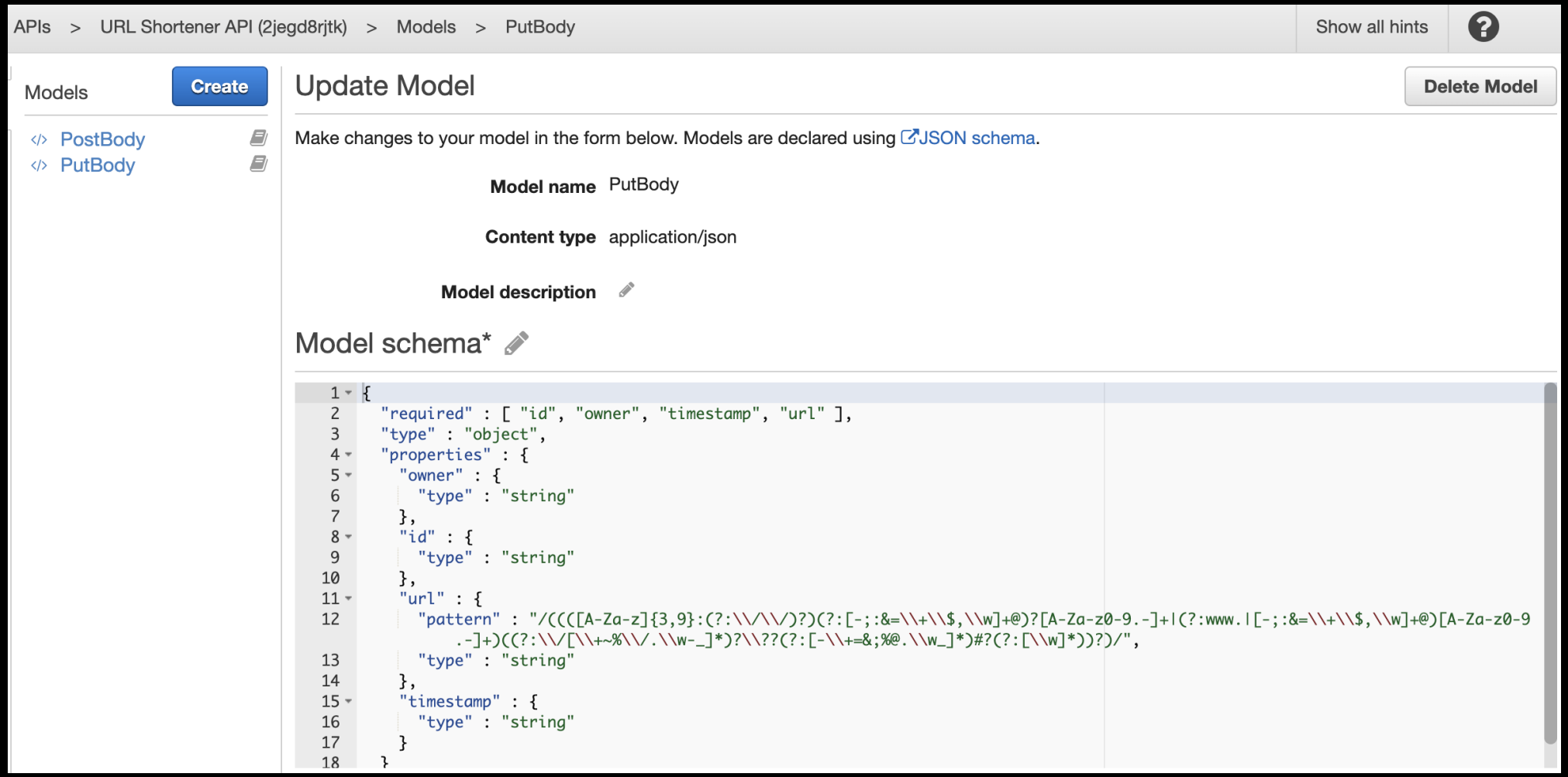
Method Request



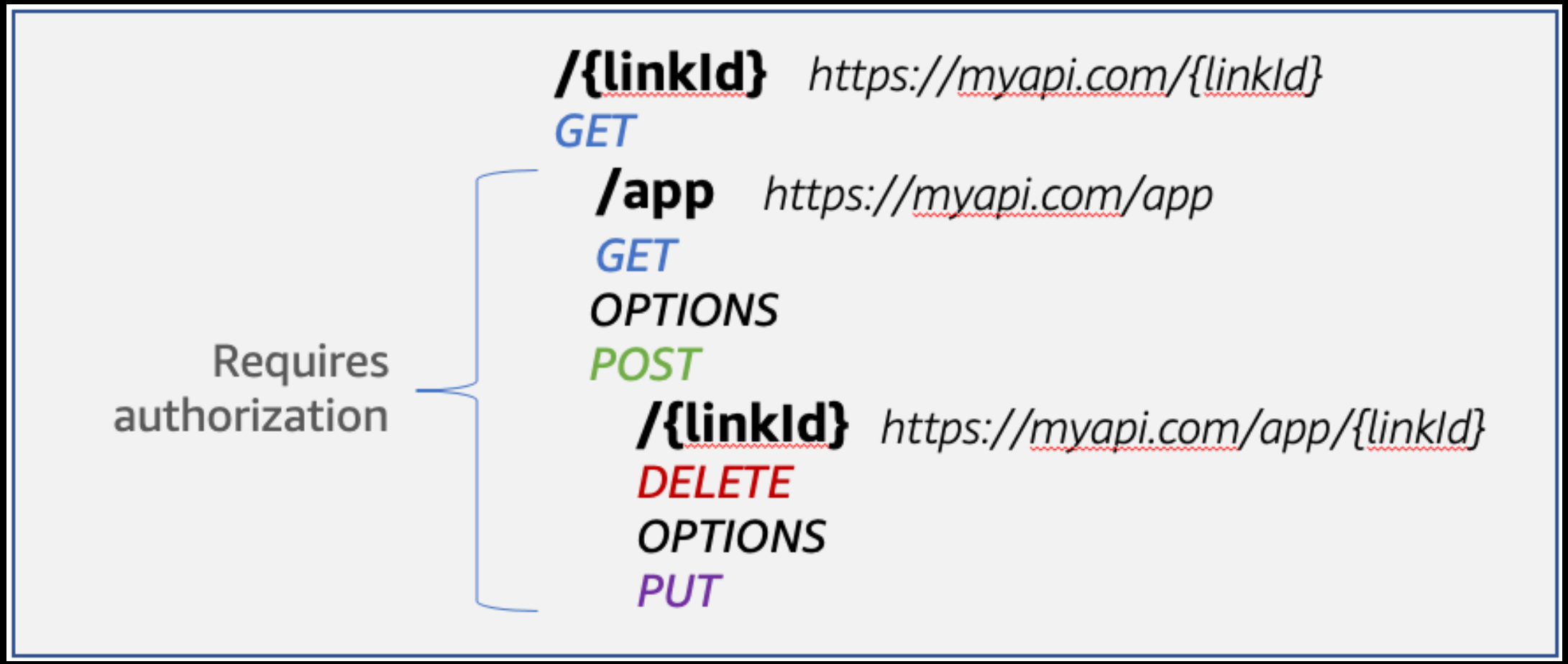
Request validation: PostBody Model



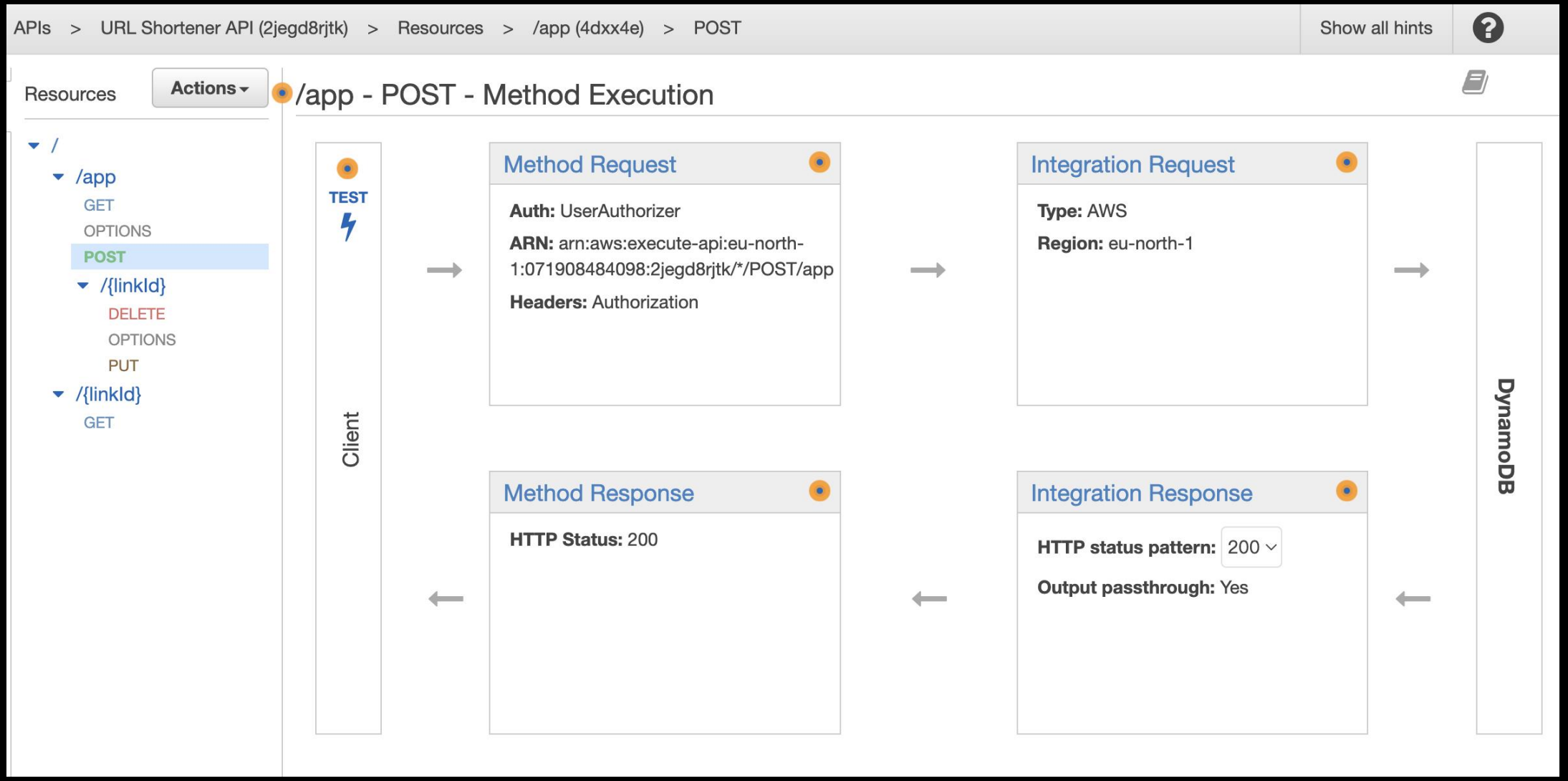
Request validation: PutBody Model



Layout of the API endpoints

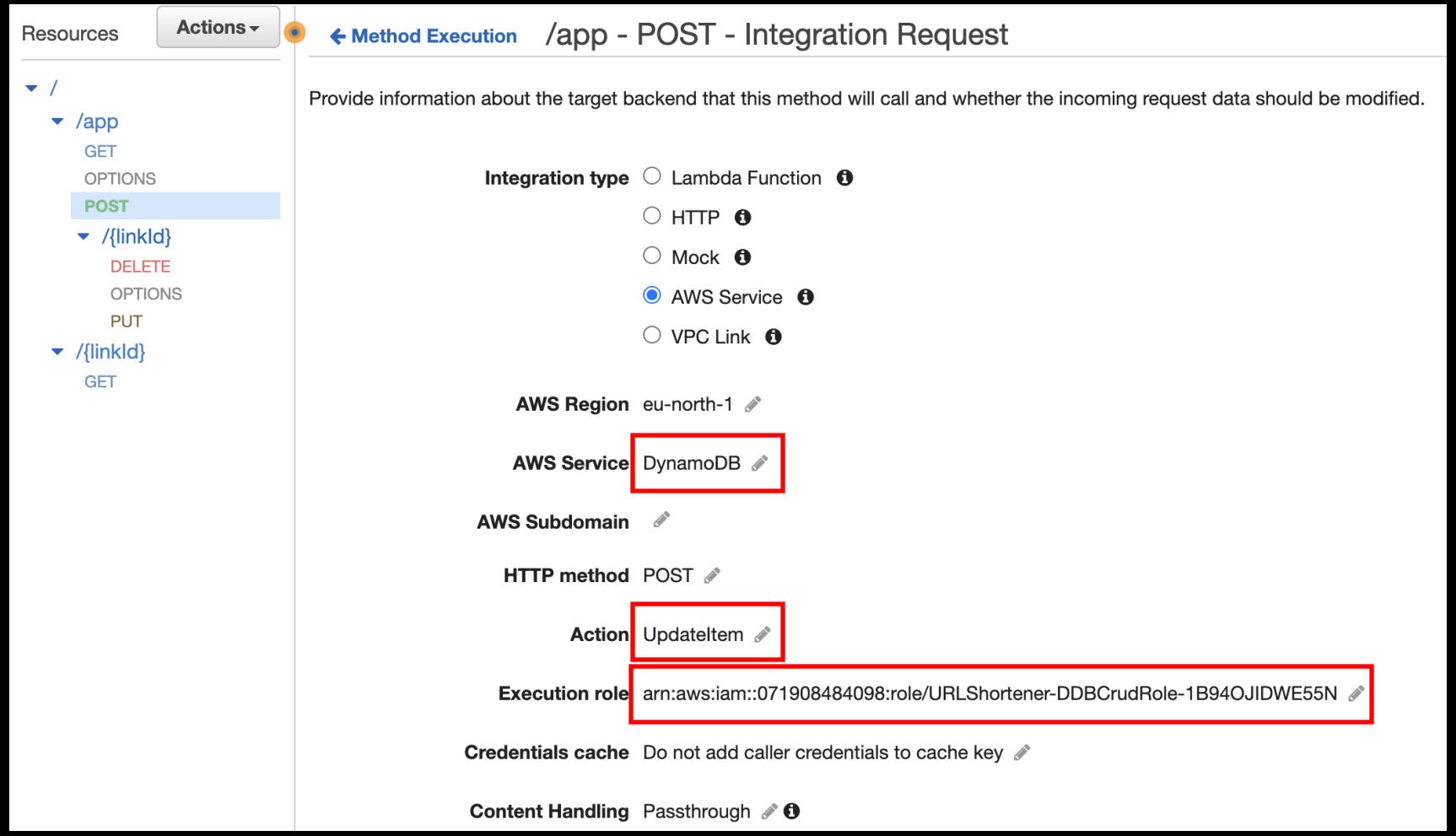


API Gateway REST API Endpoint

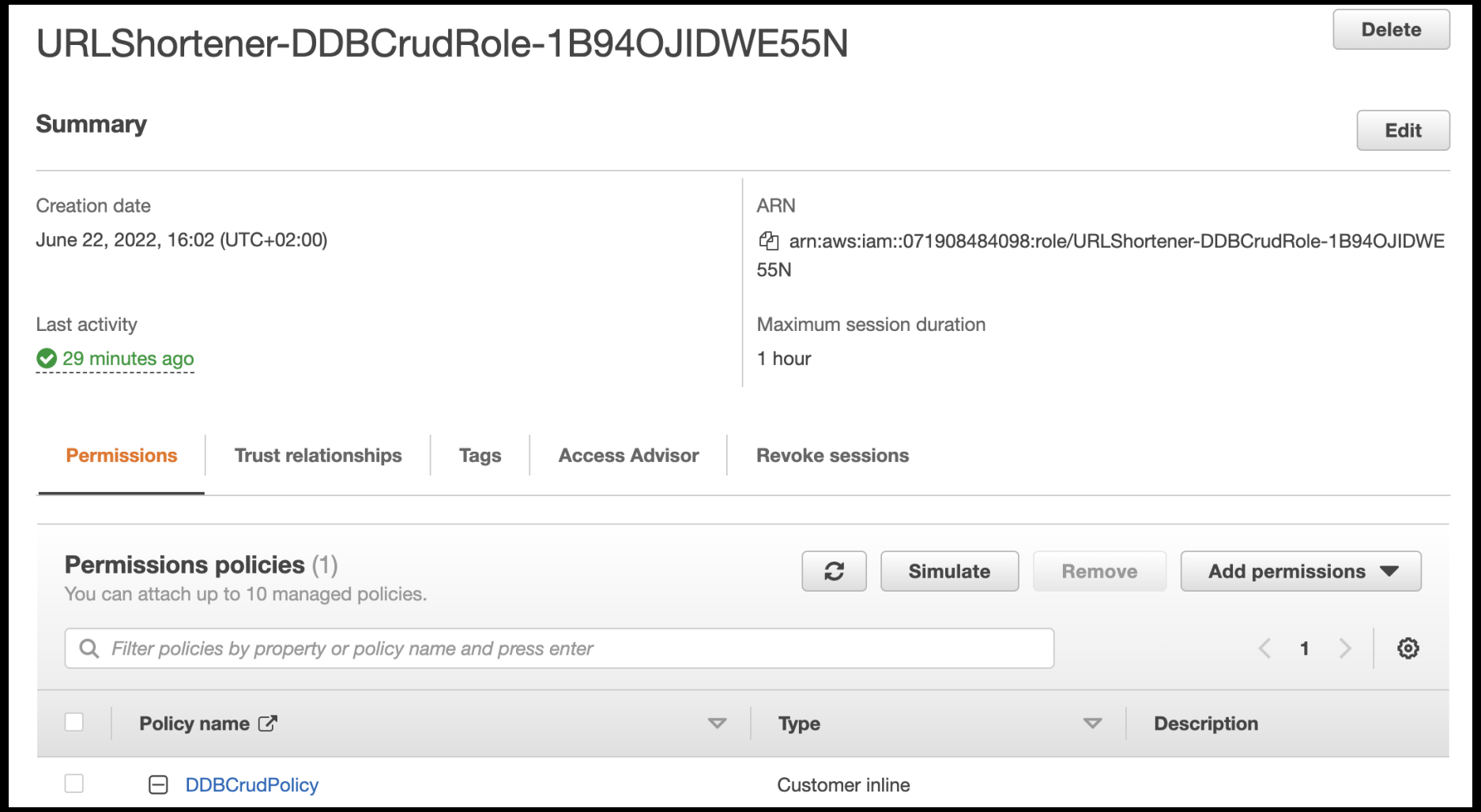


Integration Request

POST method is required to update items of DynamoDB table



IAM Role for changing DynamoDB Table



DDBCrudPolicy

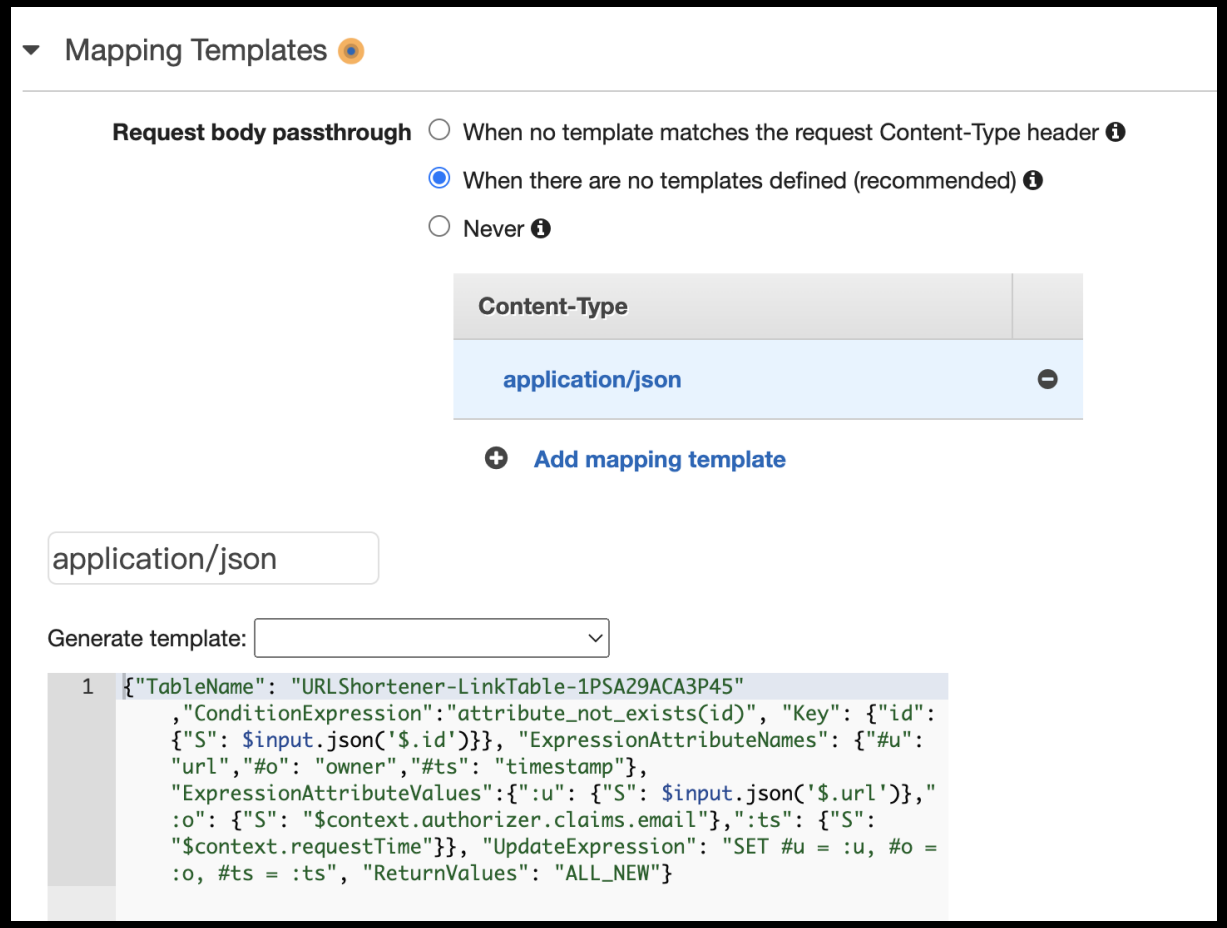
```
{
   "Version": "2012-10-17",
   "Statement": {
      "Action": [
         "dynamodb:DeleteItem",
         "dynamodb:UpdateItem"
      ],
      "Resource": "arn:aws:dynamodb:eu-north-1:071908484098:table/URLShortener-LinkTable-1PSA29ACA3P45",
      "Effect": "Allow"
   }
}
```

This policy is used for all create, read, update, and delete (CRUD) operations, and *UpdateItem* is used for both create and update.

IAM Role in SAM

```
##   Dynamo DB Read/Write Role
 DDBCrudRole:
  Type: "AWS::IAM::Role"
  Properties:
   AssumeRolePolicyDocument:
    Version: "2012-10-17"
    Statement:
     -
      Effect: "Allow"
      Principal:
       Service: "apigateway.amazonaws.com"
      Action:
       - "sts:AssumeRole"
   Policies:
    - PolicyName: DDBCrudPolicy
      PolicyDocument:
       Version: '2012-10-17'
       Statement:
        Action:
        - dynamodb:DeleteItem
        - dynamodb:UpdateItem
        Effect: Allow
        Resource: !GetAtt LinkTable.Arn
```

Mapping Templates




```
{
  "TableName":"URLShortener-LinkTable-1PSA29ACA3P45",
  "ConditionExpression":"attribute_not_exists(id)",
  "Key":{
    "id":{
      "S":"$input.json(""$.id"")"
    }
  },
  "ExpressionAttributeNames":{
    "#u":"url",
    "#o":"owner",
    "#ts":"timestamp"
  },
  "ExpressionAttributeValues":{
    ":u":{
      "S":"$input.json(""$.url"")"
    },
    ":o":{
      "S":"$context.authorizer.claims.email"
    },
    ":ts":{
      "S":"$context.requestTime"
    }
  },
  "UpdateExpression":"SET #u = :u, #o = :o, #ts = :ts",
  "ReturnValues":"ALL_NEW"
}
```

Mapping Templates

The TableName indicates which table to use in the call.

The ConditionExpression value ensures that the id passed does not already exist. The value for id is extracted from the request body using *$input.json('$.id')*.

In the ExpressionAttributeValues I have set ':o' to *$context.authorizer.claims.email*. This extracts the authenticated user's email from the request context and maps it to owner. This allows me to uniquely group a single user's links into a global secondary index (GSI). Querying the GSI is much more efficient than scanning the entire table.

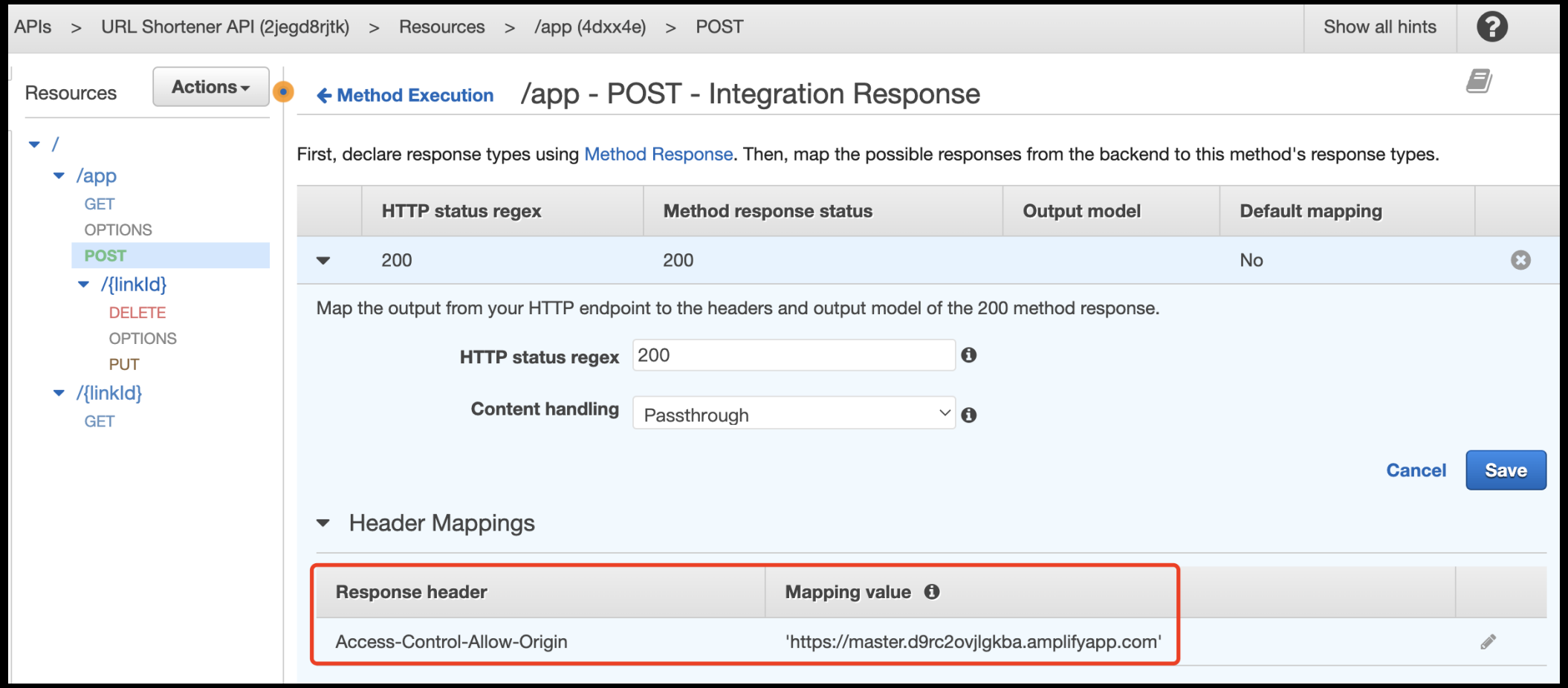
I also retrieve the requestTime from the context object, allowing me to place a timestamp in the record.

I set the ReturnValues to return all new values for the record. Finally, the UpdateExpression maps the values to the proper names and inserts the item into DynamoDB

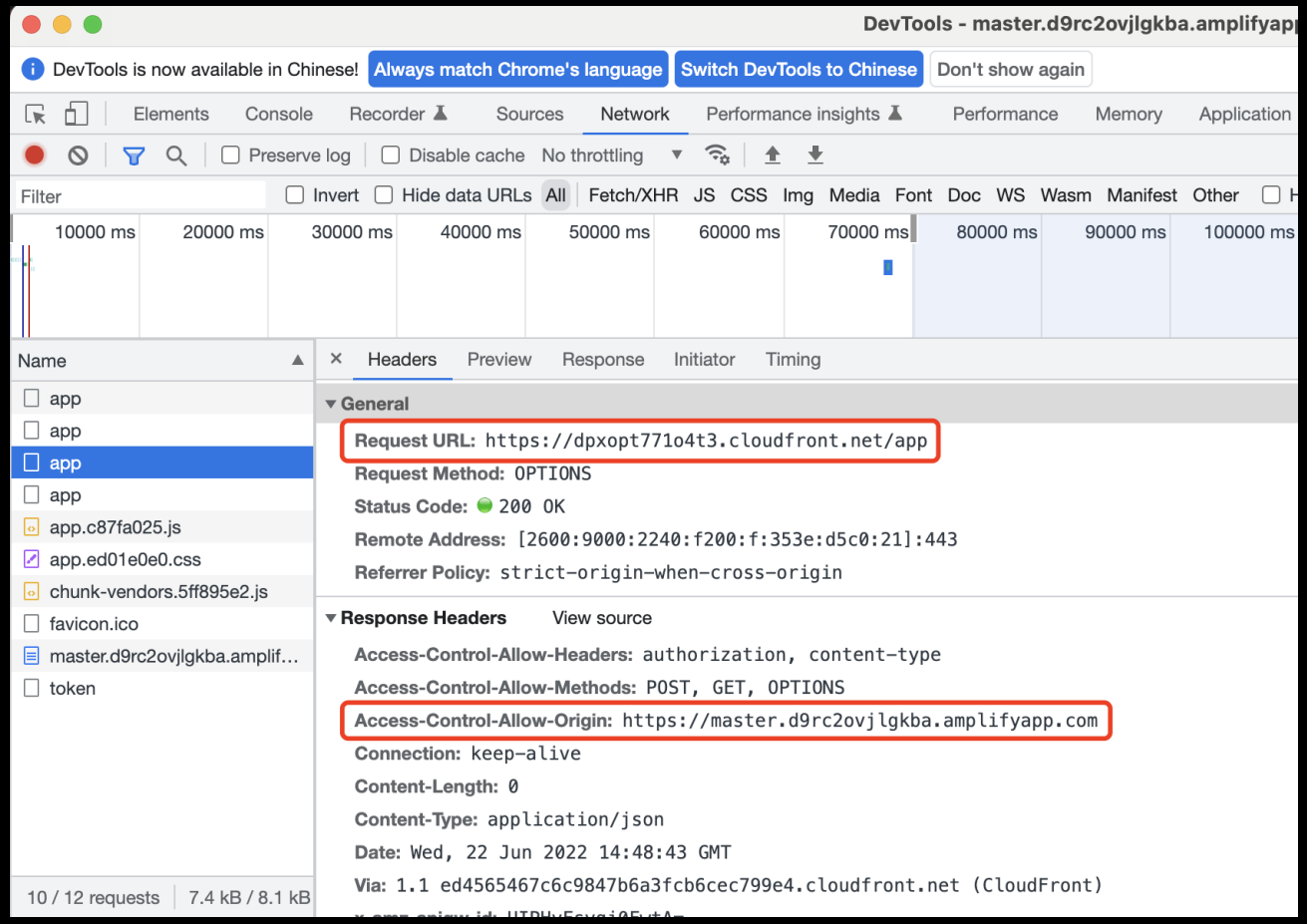
```
{
  "TableName":"URLShortener-LinkTable-1PSA29ACA3P45",
  "ConditionExpression":"attribute_not_exists(id)",
  "Key":{
    "id":{
      "S":"$input.json(""$.id"")"
    }
  },
  "ExpressionAttributeNames":{
    "#u":"url",
    "#o":"owner",
    "#ts":"timestamp"
  },
  "ExpressionAttributeValues":{
    ":u":{
      "S":"$input.json(""$.url"")"
    },
    ":o":{
      "S":"$context.authorizer.claims.email"
    },
    ":ts":{
      "S":"$context.requestTime"
    }
  },
  "UpdateExpression":"SET #u = :u, #o = :o, #ts = :ts",
  "ReturnValues":"ALL_NEW"
}
```

Integration Response

Allow AWS Amplify to access the CloudFront



Access-Control-Allow-Origin Header

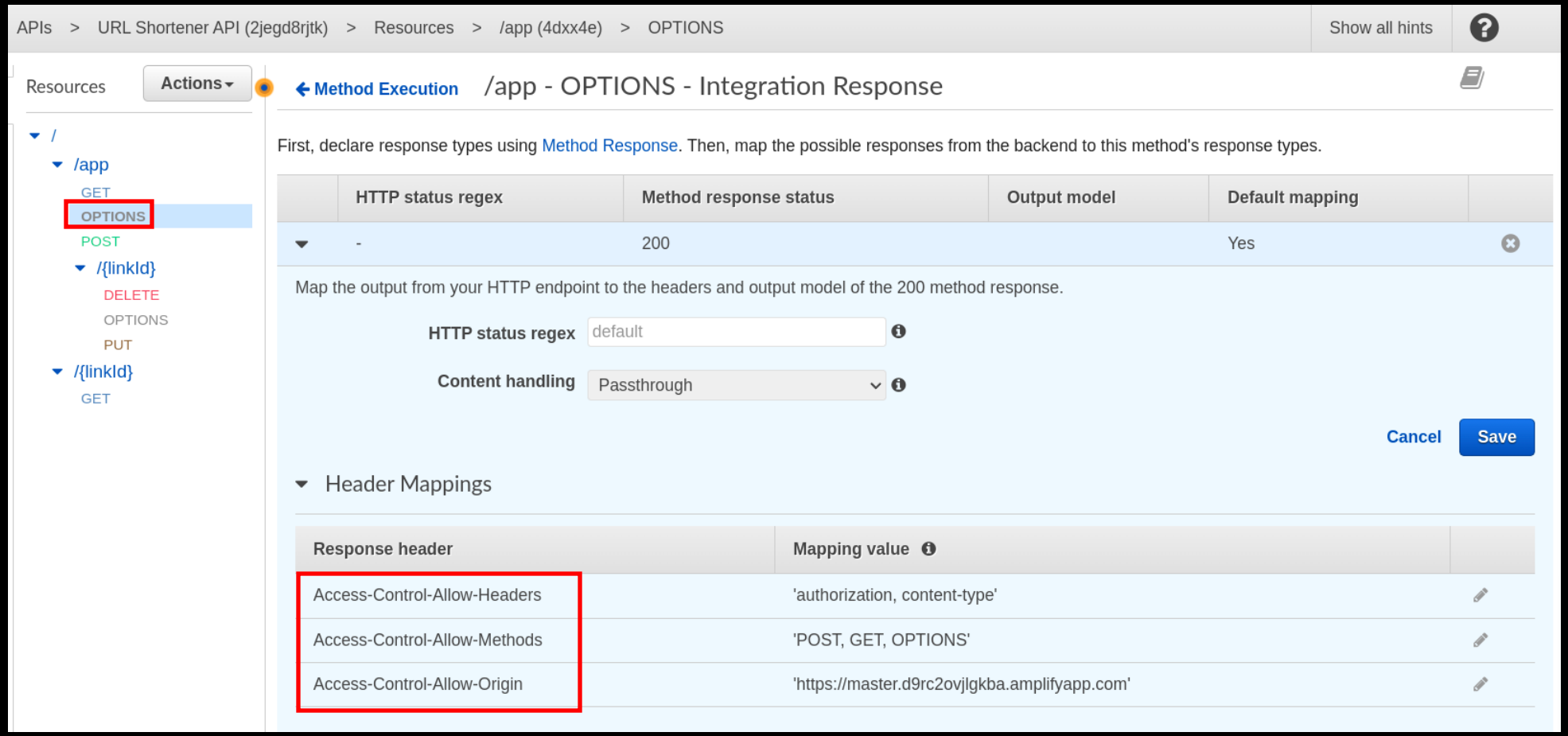


For cross-origin POST method requests, the response from my resource needs to include the header Access-Control-Allow-Origin, where the value of the header key is set to amplify in order to allow the access of that resource.

CORS

- Cross-Origin Resource Sharing is a mechanism for allowing applications from different domains to communicate. The limitations are based on exchanged headers between the client and the server. For example, the server passes the Access-Control-Allow-Origin header, which indicates which client domain is allowed to interact with the server. The client passes the Origin header that indicates what domain the request is coming from. If the two headers do not match exactly, then the request is rejected.
-

Set up mock integrations in API Gateway



- This is the header configuration for the /app OPTIONS call:

```
options:
  responses:
    "200":
      description: "200 response"
      headers:
        Access-Control-Allow-Origin:
          schema:
            type: "string"
        Access-Control-Allow-Methods:
          schema:
            type: "string"
        Access-Control-Allow-Headers:
          schema:
            type: "string"
```

AWS Amplify

All apps > Url-Shortener-Client > master

master

View latest build | View build history

Build 2

Redeploy this version

Domain
https://master.d9rc2ovjlgkba.amplifyapp.com

Source repository
https://github.com/bolecodex/url-shortener/tree/master

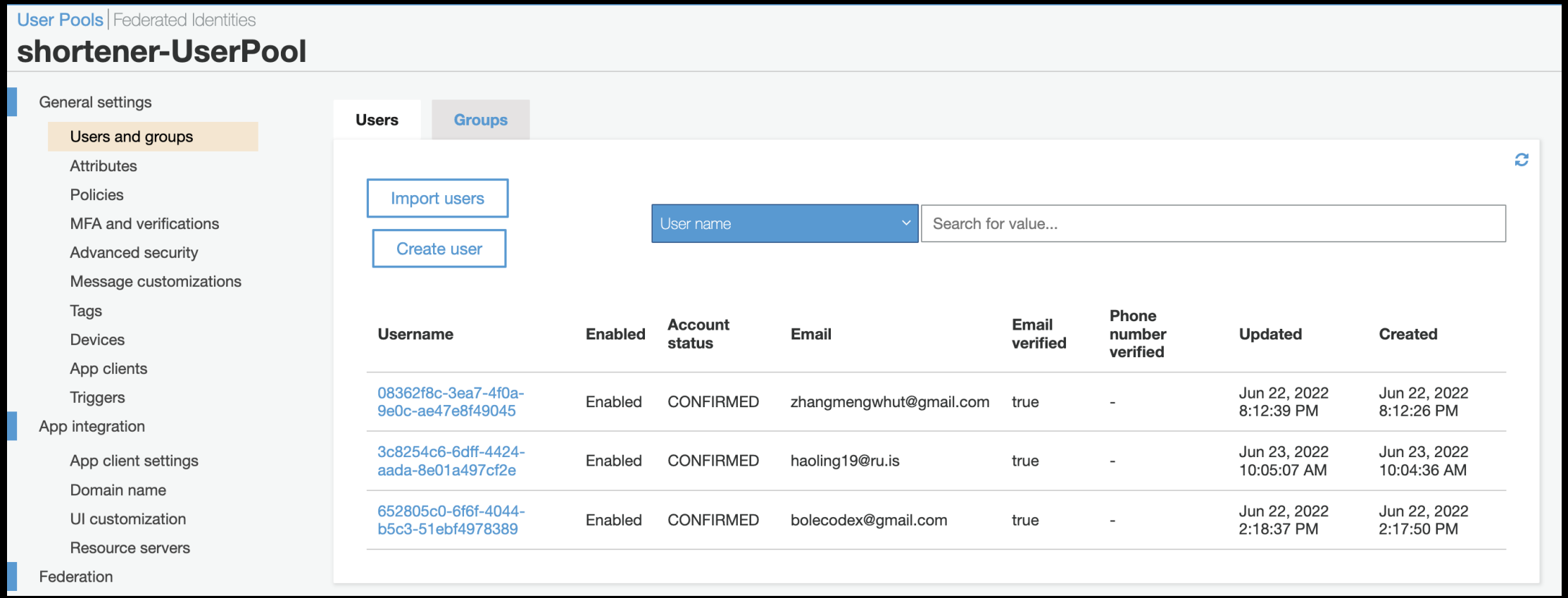
Started at
2022/6/22 16:13:27

Last commit message

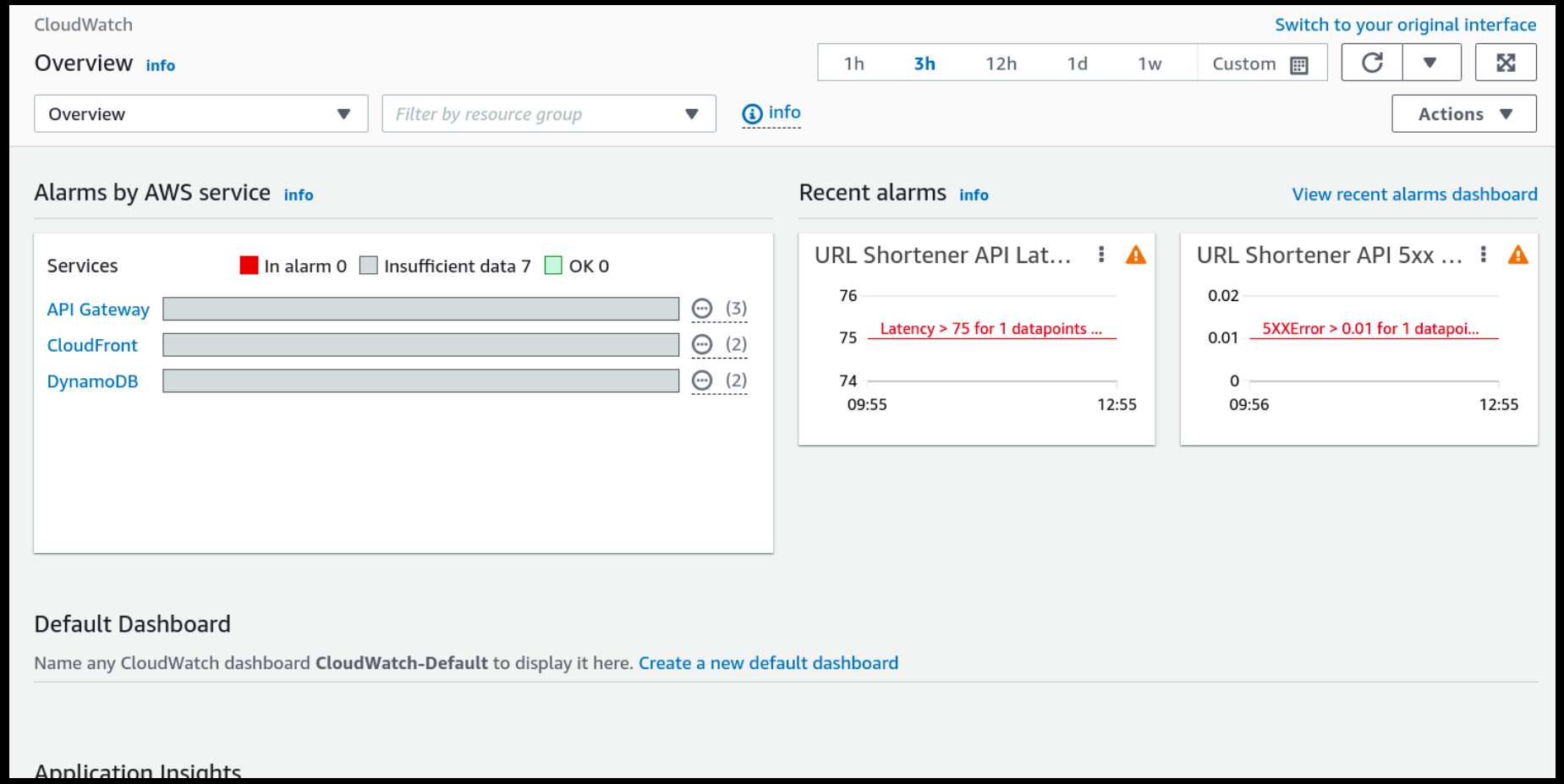
Build duration
3 minutes 3 seconds

- sam deploy –g
- -g, --guided
- Specify this flag to allow SAM CLI to guide you through the deployment using guided prompts.

Registerd Users



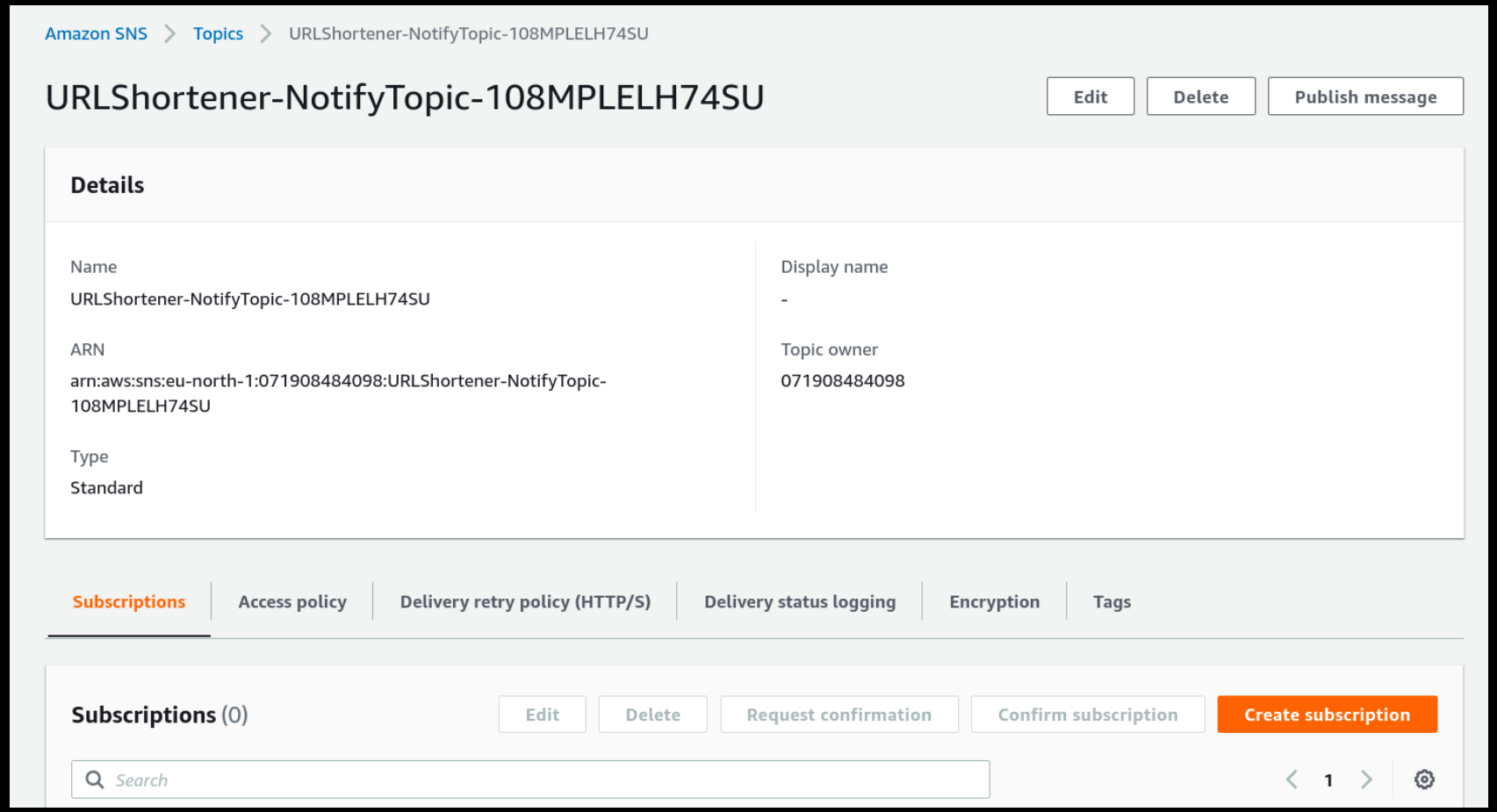
CloudWatch Dashboard



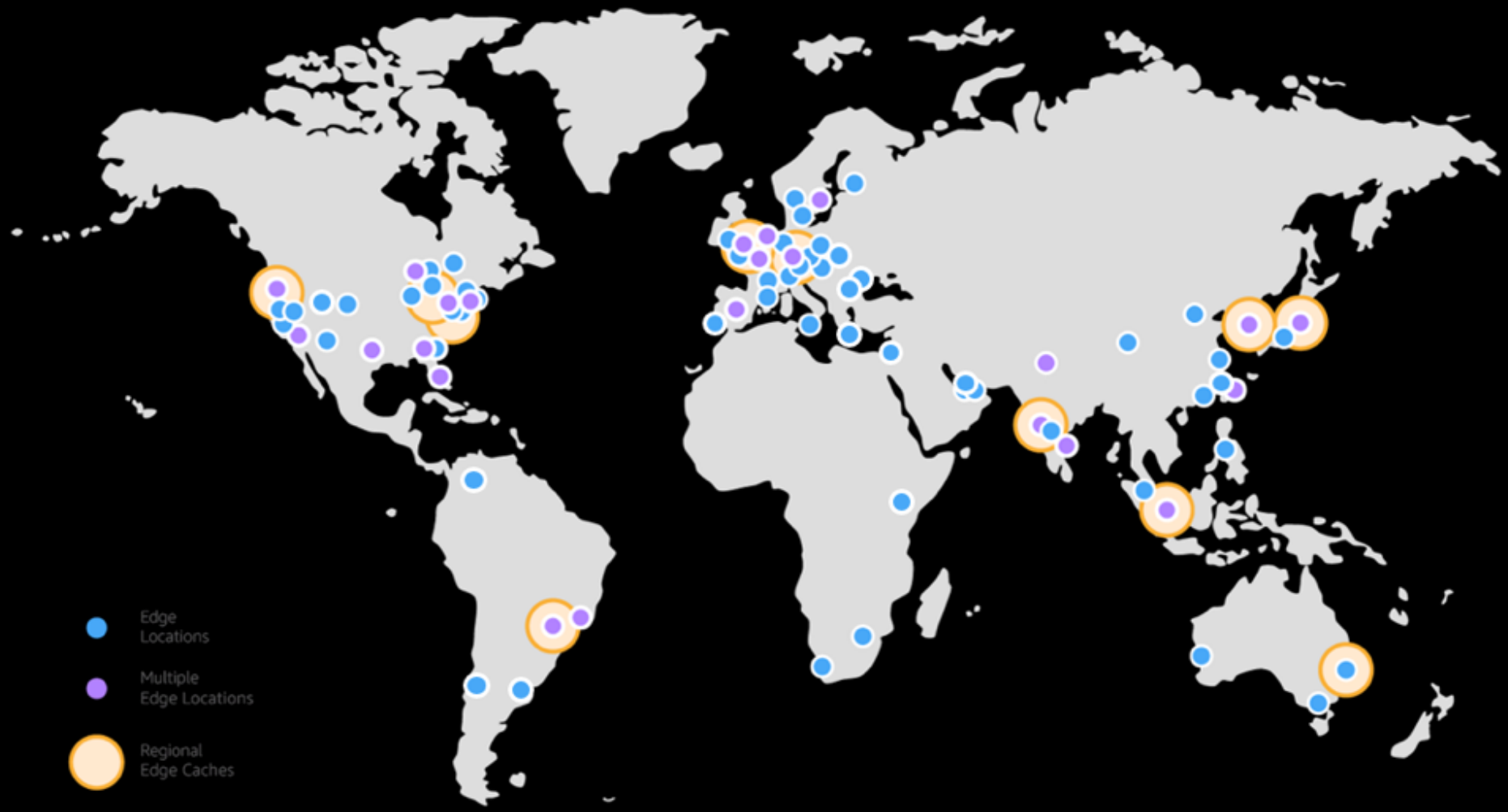
CloudWatch Alarms

Alarm	Trigger
APIGateway4xxAlarm	One percent of the API calls result in a 4xx error over a one-minute period.
APIGateway5xxAlarm	One percent of the API calls result in a 5xx error over a one-minute period.
APIGatewayLatencyAlarm	The p99 latency experience is over 75 ms over a five-minute period.
DDB4xxAlarm	One percent of the DynamoDB requests result in a 4xx error over a one-minute period.
DDB5xxAlarm	One percent of the DynamoDB requests result in a 5xx error over a one-minute period.
CloudFrontTotalErrorRateAlarm	Five requests to CloudFront result in a 4xx or 5xx error over a one-minute period.
CloudFrontTotalCacheHitRAteAlarm	80% or less of the requests to CloudFront result in a cache hit over a five-minute period. While this is not an error or a problem, it indicates the need for a more aggressive caching story.

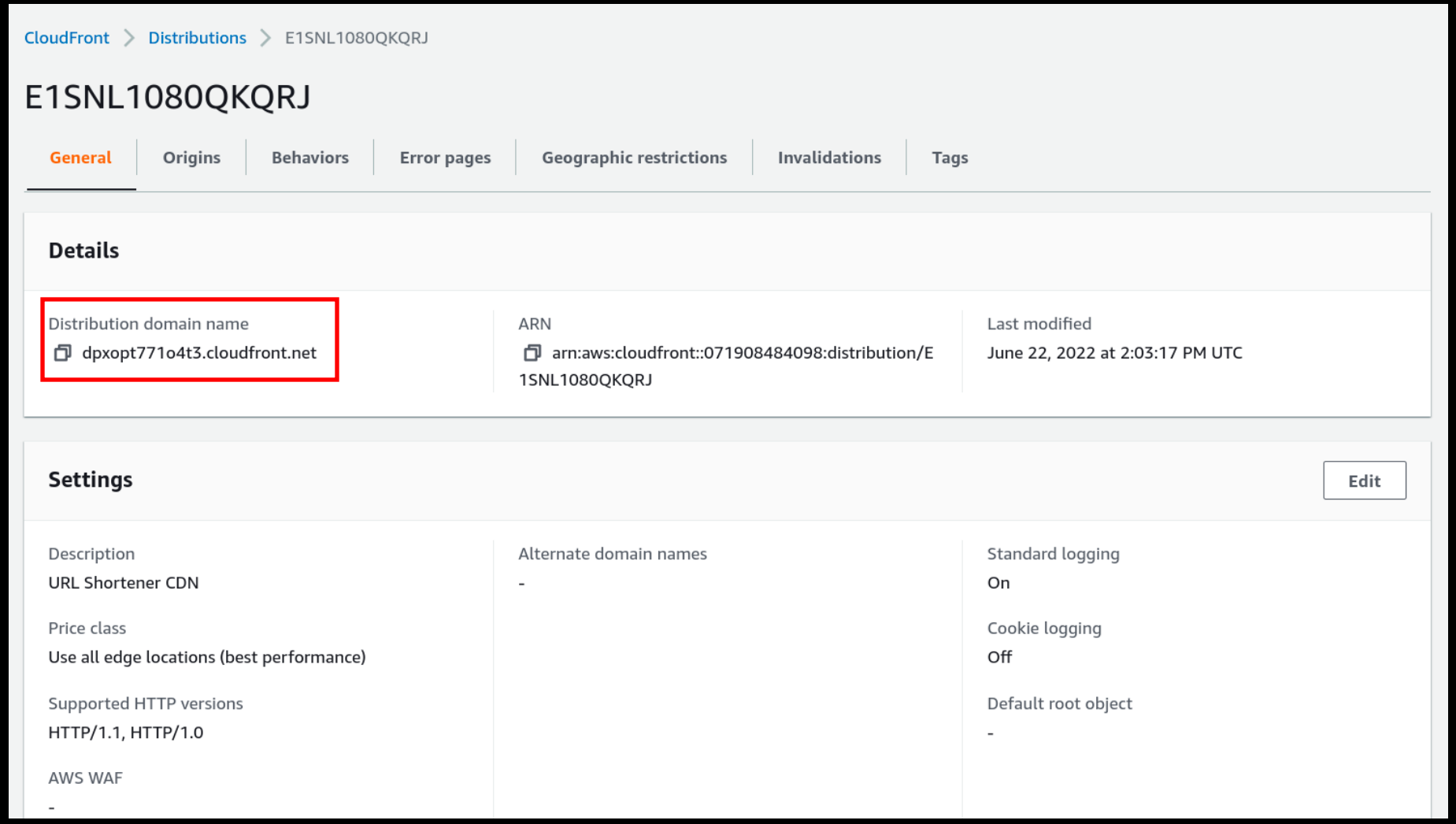
Configure email address as a subscriber to the SNS topic



Content delivery network



CloudFront:


CloudFront > Distributions > E1SNL1080QKQRJ
E1SNL1080QKQRJ
General
Origins
Behaviors
Error pages
Geographic restrictions
Invalidations
Tags
Details
Distribution domain name
dpxopt771o4t3.cloudfront.net
ARN
arn:aws:cloudfront::071908484098:distribution/E1SNL1080QKQRJ
Last modified
June 22, 2022 at 2:03:17 PM UTC
Settings
Edit
Description
URL Shortener CDN
Alternate domain names
-
Standard logging
On
Price class
Use all edge locations (best performance)
Cookie logging
Off
Supported HTTP versions
HTTP/1.1, HTTP/1.0
Default root object
-
AWS WAF
-

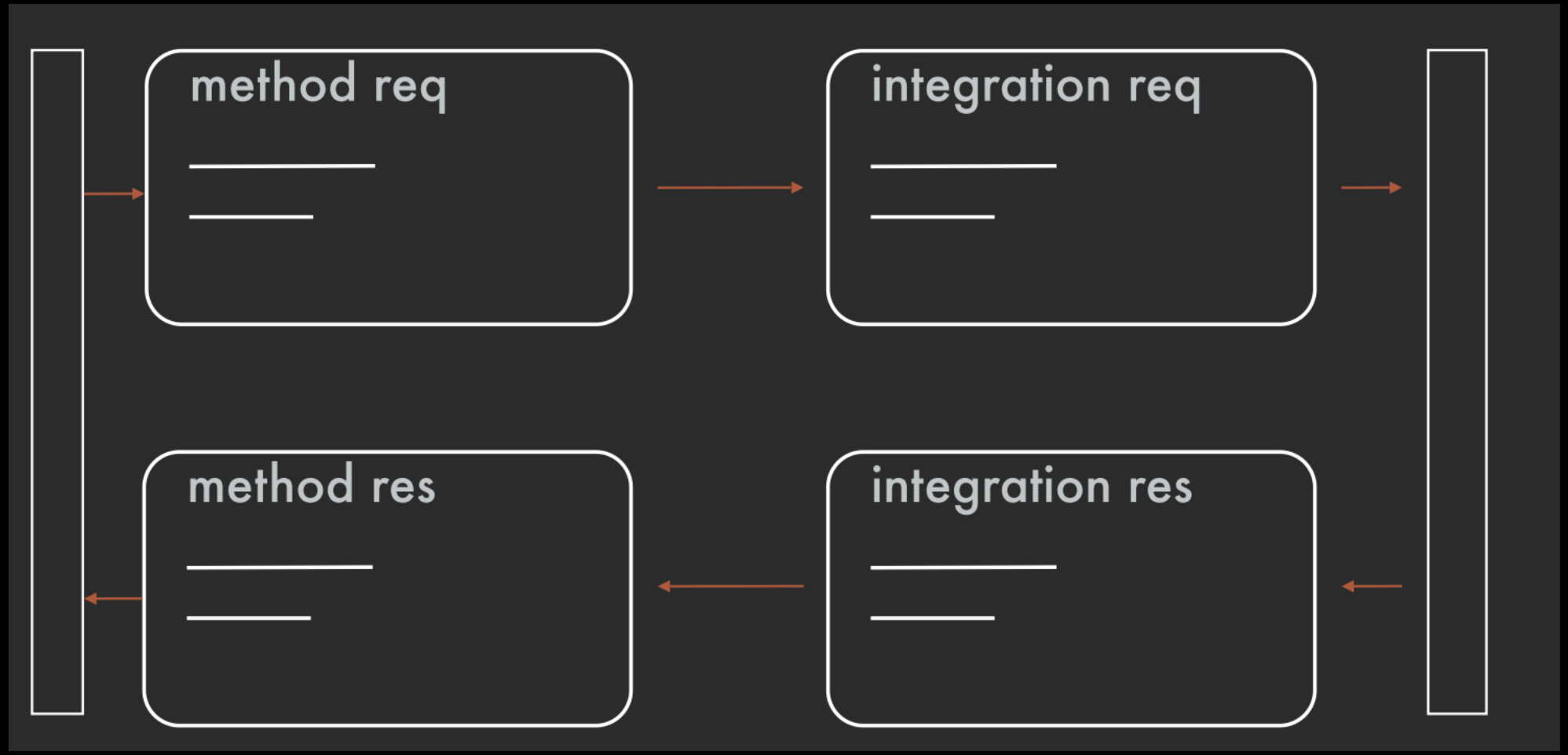
Conclusion

- This application is built by integrating multiple managed services together and applying business logic via mapping templates on API Gateway. Because these are all serverless managed services, they provide inherent availability and scale to meet the client load as needed.

Benefits

- The advantage to this pattern is also in its extensibility. With the data saved to DynamoDB, I can use the DynamoDB streaming feature to connect additional processing as needed.
- I can also use CloudFront access logs to evaluate internal application metrics.

Basic Components

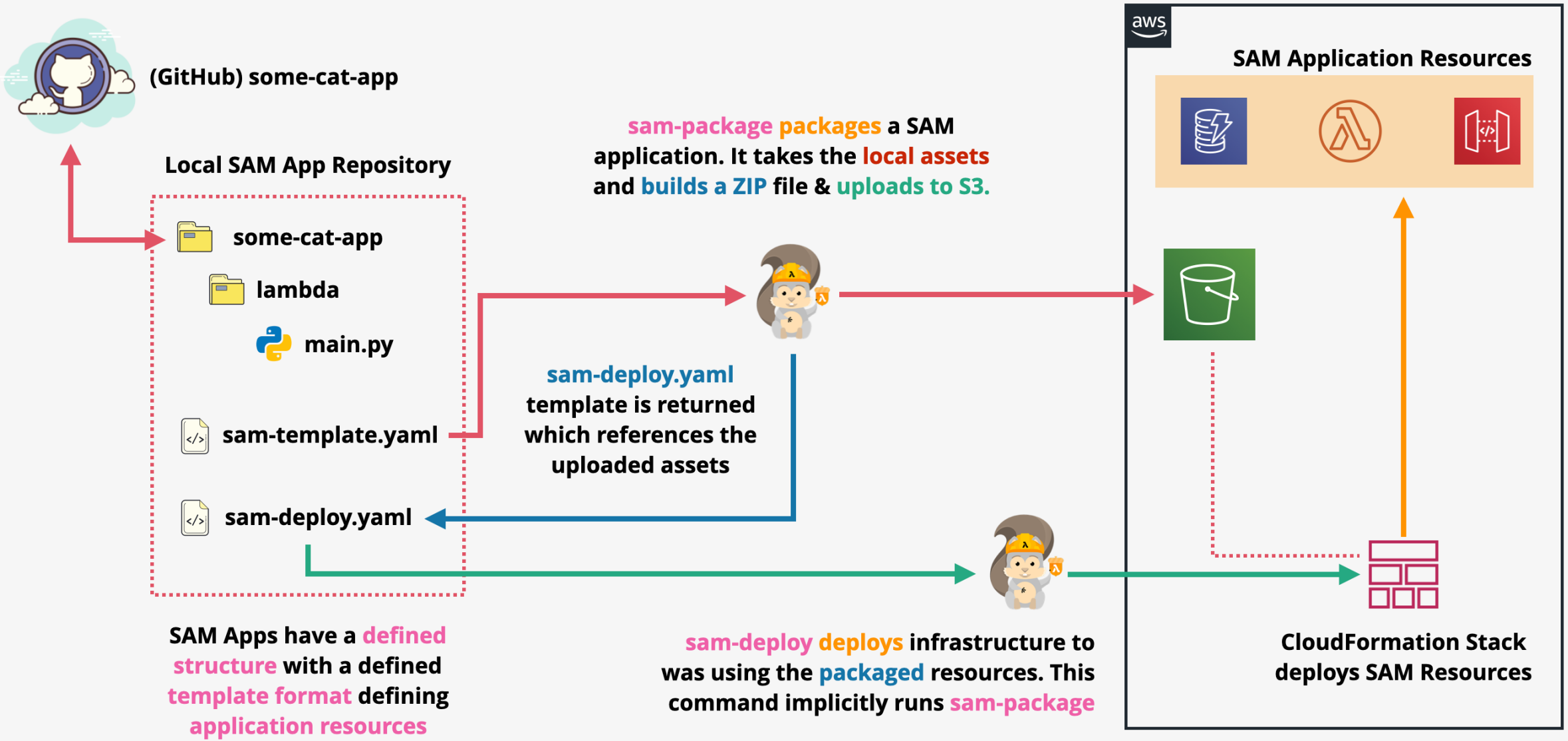


Request API Flow

AWS SAM

Serverless Application Model

Serverless Application Model (SAM)



- **AWS Serverless Application Model (AWS SAM)**

- AWS Serverless Application Model (AWS SAM) is an open source framework for building serverless applications on AWS. It provides you with template specifications for defining serverless applications and a command line interface (CLI) tool.

- A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that serverless applications are not just Lambda functions, but can also contain other resources such as APIs, databases, and event source maps.

- You can use AWS SAM to define serverless applications. AWS SAM includes the following components:

- AWS SAM template specification. You can use this specification to define serverless applications. It gives you a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application. You can use AWS SAM template files to run on a single deployable versioned entity, which is your serverless application.

- AWS SAM Command Line Interface (AWS SAMCLI). You can use this tool to generate SAM templates by AWS. The commands provided by the CLI enable you to verify that the AWS SAM template file is written according to the specification, invoke a Lambda function locally, step through the Lambda function, package and deploy a serverless application to the AWS cloud, and more.

Introduction to SAM

SAM is both an **AWS CLI tool** and a **CloudFormation Macro** which makes it effortless to define and deploy serverless applications.

What is a Macro?
A replacement output sequence according to a defined procedure. The mapping process that instantiates a macro use into a specific sequence is known as macro expansion.

A Macro **allows you to change the rules on how code works**, allowing you to embed a language with a language. Macros serve to make code **more human readable** or **allow to write less code**. Creating a language within another launch is called a DSL (Domain specific language) By using macros you are creating a DSL.

CloudFormation allows you to specify macros through the **Transform** attribute. This is how SAM is used.

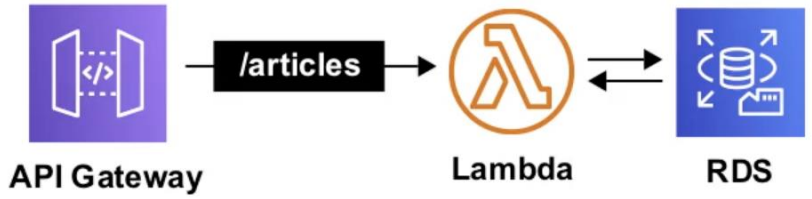
```
Transform: 'AWS::Serverless-2016-10-31'
```

SAM gives you new Resource types:

- AWS::Serverless::Function
- AWS::Serverless::API
- AWS::Serverles::SimpleTable

SAM vs CloudFormation

An example of API Gateway calling a Lambda and that Lambda gets data from database.



Without SAM

With SAM

You write **at least 50% less code** with SAM

```yaml
Resources:
  RestAPI:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: MyApi
      EndpointConfiguration:
        Types:
        - REGIONAL
      Body:
        swagger: '2.0'
        info:
          version: 1.0.0
          title: MyApi
        schemes:
        - https
        paths:
          "/": {}
          "/articles":
            get:
              produces:
              - application/json
              - application/customer-error
              - application/api_not_found
              - application/no_record
              - application/api_error
              responses:
                '200':
                  description: 200 response
                  schema:
                    "$ref": "#/definitions/Empty"
                '201':
                  description: 201 response
                '204':
                  description: 204 response
                '230':
                  description: 230 response
                '231':
                  description: 231 response
                '232':
                  description: 232 response
                '233':
                  description: 233 response
              security:
              - sigv4: []
              x-amazon-apigateway-integration:
                responses:
                  default:
                    statusCode: '200'
                uri:
                  Fn::Join:
                  - ":"
                  - - arn:aws:apigateway
                    - Ref: AWS::Region
                    - lambda:path/functions/arn:aws:lambda
                    - Ref: AWS::Region
                    - Ref: AWS::AccountId
                    - function
                    - Fn::Join:
                      - ''
                      - - Ref: APIFunction
                        - "/invocations"
                passthroughBehavior: when_no_match
                httpMethod: POST
                contentHandling: CONVERT_TO_TEXT
                credentials:
                  Fn::GetAtt:
                  - IAMAPIServiceRole
                  - Arn
  APIFunction:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: GetArticles
      Code:
        S3Bucket: my-app
        S3Key: functions/articles.index.zip
      Handler: index.handler
      Runtime: nodejs10.x
      MemorySize: 128
      Role:
        Fn::GetAtt:
        - IAMLambdaServiceRole
        - Arn
      Timeout: 60
  IAMLambdaServiceRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: LambdaServiceRole
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Sid: ''
          Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: sts:AssumeRole
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs
      Policies:
      - PolicyName: LambdaServiceRolePolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Action:
            - s3:Get*
            - s3:List*
            Resource: "*"
            Effect: Allow
```

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Resources:
  MyAPI:
    Type: AWS::Serverless::Api
    Properties:
      Name: MyApi
      StageName: prod
      EndpointConfiguration: REGIONAL
      Cors:
        AllowMethods: "'GET,POST,OPTIONS'"
        AllowHeaders: "'Content-Type,X-Api-Key,X-Amz-Security-Token'"
        AllowOrigin: "'*'"
  ArticlesIndex:
    Type: 'AWS::Serverless::Function'
    Properties:
      Runtime: nodejs10.x
      Handler: index.handler
      CodeUri: s3://my-app/functions/articles.index.zip
      MemorySize: 128
      Timeout: 10
      AutoPublishAlias: live
      DeploymentPreference:
        Type: AllAtOnce
      Events:
        Post:
          Type: Api
          Properties:
            Path: /
            Method: Get
            RestApiId:
              Ref: MyAPI
      Policies:
        - AWSLambdaExecute # Managed Policy
        - AWSLambdaVPCAccessExecutionRole
        - Version: '2012-10-17' # Policy Document
          Statement:
            - Effect: Allow
              Action:
                - secretsmanager:GetSecretValue
                - secretsmanager:PutResourcePolicy
                - secretsmanager:PutSecretValue
                - secretsmanager:DeleteSecret
                - secretsmanager:DescribeSecret
                - secretsmanager:TagResourc
                - secretsmanager:CreateSecret
                - secretsmanager:ListSecrets
                - secretsmanager:GetRandomPassword
                - tag:GetResources
                - rds-data:BatchExecuteStatement
                - rds-data:BeginTransaction
                - rds-data:CommitTransaction
                - rds-data:ExecuteStatement
                - rds-data:RollbackTransaction
              Resource: "*"
```

SAM CLI

SAM has a CLI which makes it easy to **run, package and deploy** Serverless Applications or Lambdas

sam build	Prepare the lambda source code to be deployed by packaging it for upload
sam deploy	Upload the lambda package code (artifact) and then deploy
sam init	Initialize a new serverless project, creating all the folders an files to get started
sam local generate-event	Generates sample payloads from different event sources, such as Amazon S3, Amazon API Gateway, and Amazon SNS
sam local invoke	Run a single lambda locally
sam local start-api	run your serverless application locally for quick development and testing
sam local start-lambda	Run a single lambda locally (much easier)
sam logs	Fetches logs that are generated by your Lambda function.
sam package	Packages an AWS SAM application. Creates a zip, and uploads it to S3
sam publish	Publishes your packaged SAM application to AWS Serverless Application Repository
sam validate	Validates an AWS SAM template (great for finding syntax bugs)

Self-service serverless

Developers want:

- No blockers from idea -> staging -> production

Security wants:

- Security to review and sign off code & configuration
- “Least privilege” applications

The tension between these two drives automation

AWS SAM

- Framework for building serverless applications
- Shorthand syntax to express functions, APIs, databases, and event source mappings
- Model with YAML, deploy using AWS CloudFormation
- Open source!
- https://github.com/awslabs/serverless-application-model

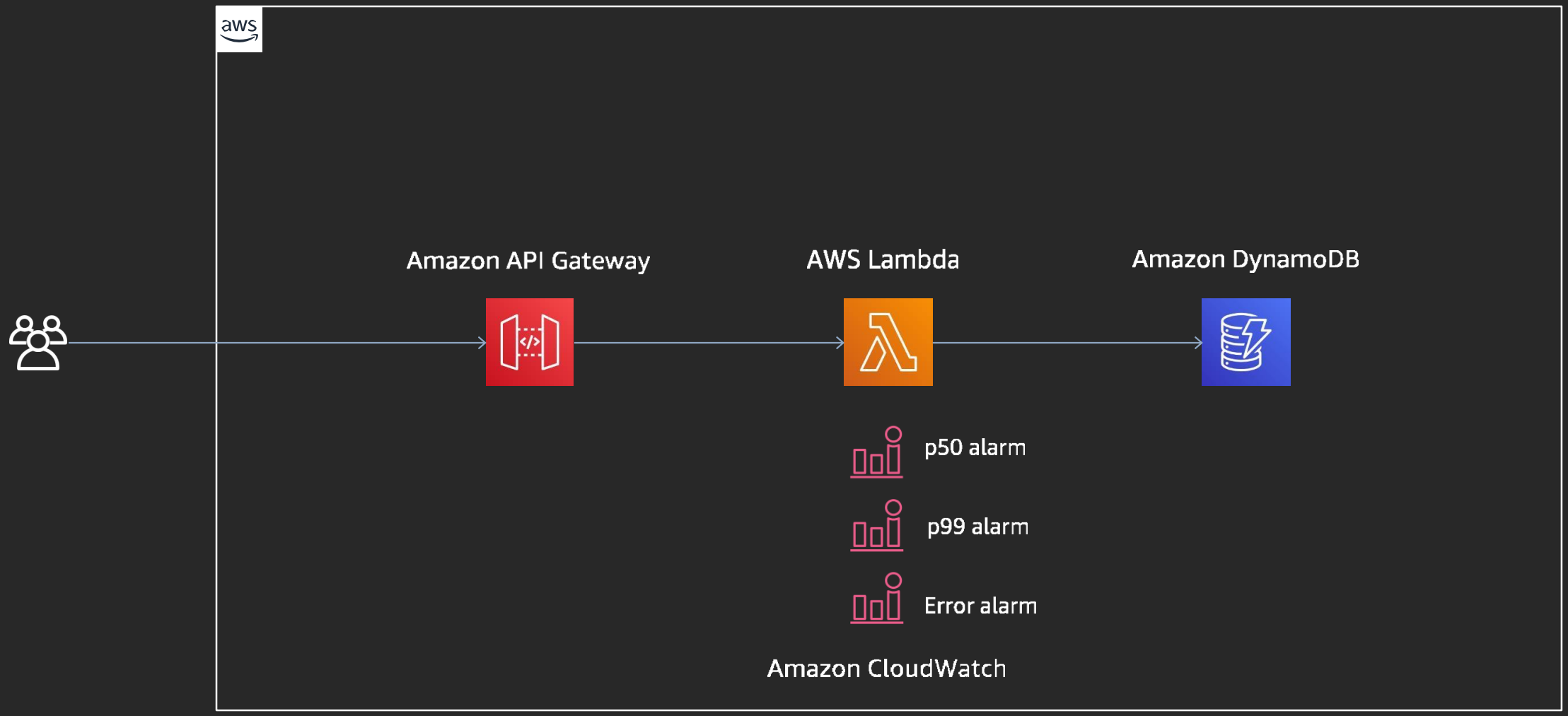
AWS SAM CLI

- Create, build, test, and deploy AWS SAM applications
- Step-through debugging and IDE support
- Open source!
- https://github.com/awslabs/aws-sam-cli

AWS SAM CLI supports custom init sources

```
sam init --location https://github.com/awood45/aws-sam-sinatra-template
```

App template structure



Example: Built-in alarms

```
SinatraErrorAlarm:
     Type: AWS::CloudWatch::Alarm
     Properties:
       ComparisonOperator:
GreaterThanOrEqualToThreshold
       EvaluationPeriods: 3
       Threshold: 1
       Dimensions:
         - Name: FunctionName
           Value: !Ref SinatraFunction
       Statistic: Sum
       TreatMissingData: missing
       Namespace: AWS/Lambda
       Period: 60
       MetricName: Errors
```

Example: Hide marshalling work

```
APP = Rack::Builder.parse_file('config.ru').first

def handler(event:,context:)
  $baseHost ||= event.fetch(
    'headers',
    {}
  ).fetch('Host', nil)
  body =
    if event['isBase64Encoded']
      Base64.decode64(event['body'])
    else
      event['body']
    end
# etc...
```

```
require 'sinatra'
require_relative 'app_table'

get "/" do
  # your app logic here
end
```

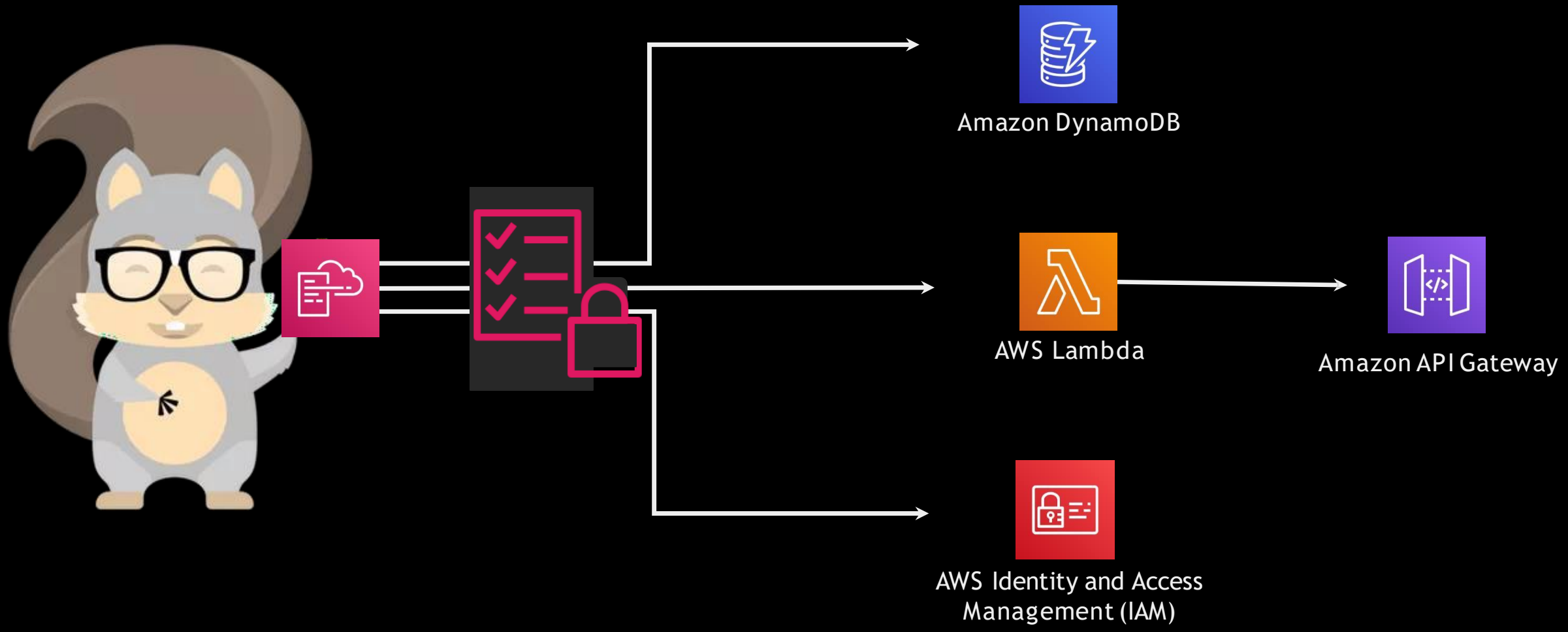
AWS SAM CLI Commands

```
sam init --location https://github.com/awood45/aws-sam-sinatra-template

sam build

sam deploy --guided
```

AWS Serverless Application Model (SAM)

```
main:
  Type: AWS::Serverless::Function
  Properties:
    Handler: main
    Runtime: go1.x
    CodeUri: ./services/main
    Role: "lambda@default"
    Timeout: 30
```

Self-service security automation AWS SAM



Serverless frameworks

AWS

AWS Amplify

AWS SAM

Third-party

APEX

Claudia.js

Serverless Framework

Sparta

Zappa

AWS Serverless Application Model (SAM)

https://github.com/awslabs/serverless-application-model

- AWS CloudFormation extension, optimized for serverless
- Serverless resource types: Functions, APIs, tables, layers, and applications
- AWS CloudFormation supports (e.g., Amazon Kinesis)
- Supports parameters, mappings, outputs, global variables, intrinsic functions, and some import values
- YAML or JSON

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  CalculatorAdd:
    Type: AWS::Serverless::Function
    Properties:
      Handler: calculator
      Runtime: go1.x
      Events:
        Add:
          Type: Api
          Properties:
            Path: /
            Method: get
```

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  CalculatorAdd:
    Type: AWS::Serverless::Function
    Properties:
      Handler: calculator
      Runtime: go1.x
      Events:
        Add:
          Type: Api
          Properties:
            Path: /
            Method: get
```

Lambda function

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  CalculatorAdd:
    Type: AWS::Serverless::Function
    Properties:
      Handler: calculator
      Runtime: go1.x
      Events:
        Add:
          Type: Api
          Properties:
            Path: /
            Method: get
```

API Gateway

Transformed template


```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "AwesomeCalculatorRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecut
        ],
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "lambda.amazonaws.com"
                ]
              }
            }
          ]
        }
      }
    },
    "AwesomeCalculatorAddPermissionTest": {
      "Type": "AWS::Lambda::Permission",
      "Properties": {
        "Action": "lambda:invokeFunction",
        "Principal": "apigateway.amazonaws.com",
        "FunctionName": {
          "Ref": "AwesomeCalculator"
        },
        "SourceArn": {
          "Fn::Sub": [
            "arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:${
            {
              "__Stage__": "*",
              "__ApiId__": {
                "Ref": "ServerlessRestApi"
              }
            }
          ]
        }
      }
    },
    "ServerlessRestApiProdStage": {
      "Type": "AWS::ApiGateway::Stage",
      "Properties": {
        "DeploymentId": {
          "Ref": "ServerlessRestApiDeployment9bcc4decb0"
        },
        "RestApiId": {
          "Ref": "ServerlessRestApi"
        },
        "StageName": "Prod"
      }
    },
    "AwesomeCalculator": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Code": {
          "S3Bucket": "sydney-summit-sam",
          "S3Key": "001f1d29dc238636e2b94fe7a71aa00c"
        },
        "Handler": "index.handler",
        "Role": {
          "Fn::GetAtt": [
            "AwesomeCalculatorRole",
            "Arn"
          ]
        },
        "Runtime": "nodejs6.10",
        "Tags": [
          {
            "Value": "SAM",
            "Key": "lambda:createdBy"
          }
        ]
      }
    },
    "ServerlessRestApi": {
      "Type": "AWS::ApiGateway::RestApi",
      "Properties": {
        "Body": {
          "info": {
            "version": "1.0",
            "title": {
              "Ref": "AWS::StackName"
            }
          },
          "paths": {
            "/": {
              "get": {
                "x-amazon-apigateway-integration": {
                  "httpMethod": "POST",
                  "type": "aws_proxy",
                  "uri": {
                    "Fn::Sub": "arn:aws:apigateway:${AWS::Region}:l
                  }
                },
                "responses": {}
              }
            }
          },
          "swagger": "2.0"
        }
      }
    },
    "AwesomeCalculatorAddPermissionProd": {
      "Type": "AWS::Lambda::Permission",
      "Properties": {
        "Action": "lambda:invokeFunction",
        "Principal": "apigateway.amazonaws.com",
        "FunctionName": {
          "Ref": "AwesomeCalculator"
        },
        "SourceArn": {
          "Fn::Sub": [
            "arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:$
            {
              "__Stage__": "Prod",
              "__ApiId__": {
                "Ref": "ServerlessRestApi"
              }
            }
          ]
        }
      }
    },
    "ServerlessRestApiDeployment9bcc4decb0": {
      "Type": "AWS::ApiGateway::Deployment",
      "Properties": {
        "RestApiId": {
          "Ref": "ServerlessRestApi"
        },
        "Description": "RestApi deployment id: 9bcc4decb055b4dac489
        "StageName": "Stage"
      }
    }
  }
}
```

SAM commands

Package

- Creates a deployment package (.zip file)
- Uploads deployment package to an Amazon S3 bucket
- Adds a CodeUri property with S3 URI

Deploy

- Calls AWS CloudFormation "*CreateChangeSet*" API
- Calls AWS CloudFormation "*ExecuteChangeSet*" API

General AWS CloudFormation security policy automation

- Always prefix resource names (Stack, Lambda, API Gateway, etc.)
  - e.g., constant-project-stage, prefix-staging-hello-world-
- JSON schema to limit and validate all resources
  - Enable and disable resources, properties, and features
- Validate that external resources are allowed to be used
  - Security groups, IAM roles, subnets
- Correctly tag resources
  - Tags are free, and most resources have them

IAM cloud formation role

IAM actions must use the permissions boundary

```
"Action": "iam:*",
"Condition": {
  "StringEquals": {
    "iam:PermissionsBoundary": "arn:aws:iam::0000:policy/pb"
```

Limit actions to prefixed resources

```
"Resource": "arn:aws:iam::0000:role/prefix-*"
```

Limit to the tags

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/prefix": "true"
```

The challenge

- Following the principle of least privilege is important
- Most AWS SAM templates have permissions to create IAM roles
  - We want to limit what those created roles can do, systemically
- Good intentions are not enough

2 ways to write IAM policies in AWS SAM

```
ServerlessFunctionCannedPolicy:
  Type: AWS::Serverless::Function
  Properties:
    Handler: lambda.handler
    Runtime: ruby2.5
    Policies:
      - DynamoDBCrudPolicy:
          TableName: !Ref AppTable
```

```
ServerlessFunctionCustomPolicy:
  Type: AWS::Serverless::Function
  Properties:
    Handler: lambda.handler
    Runtime: ruby2.5
    Policies:
      - Version: ”2012-10-17”
        Statement:
          - Effect: Allow
            Action:
              - dynamodb:GetItem
              - dynamodb:PutItem
          -     dynamodb:UpdateItem
        Resource:
-         !Sub
arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::
AccountId}:table/${AppTable}
```

IAM permissions boundaries

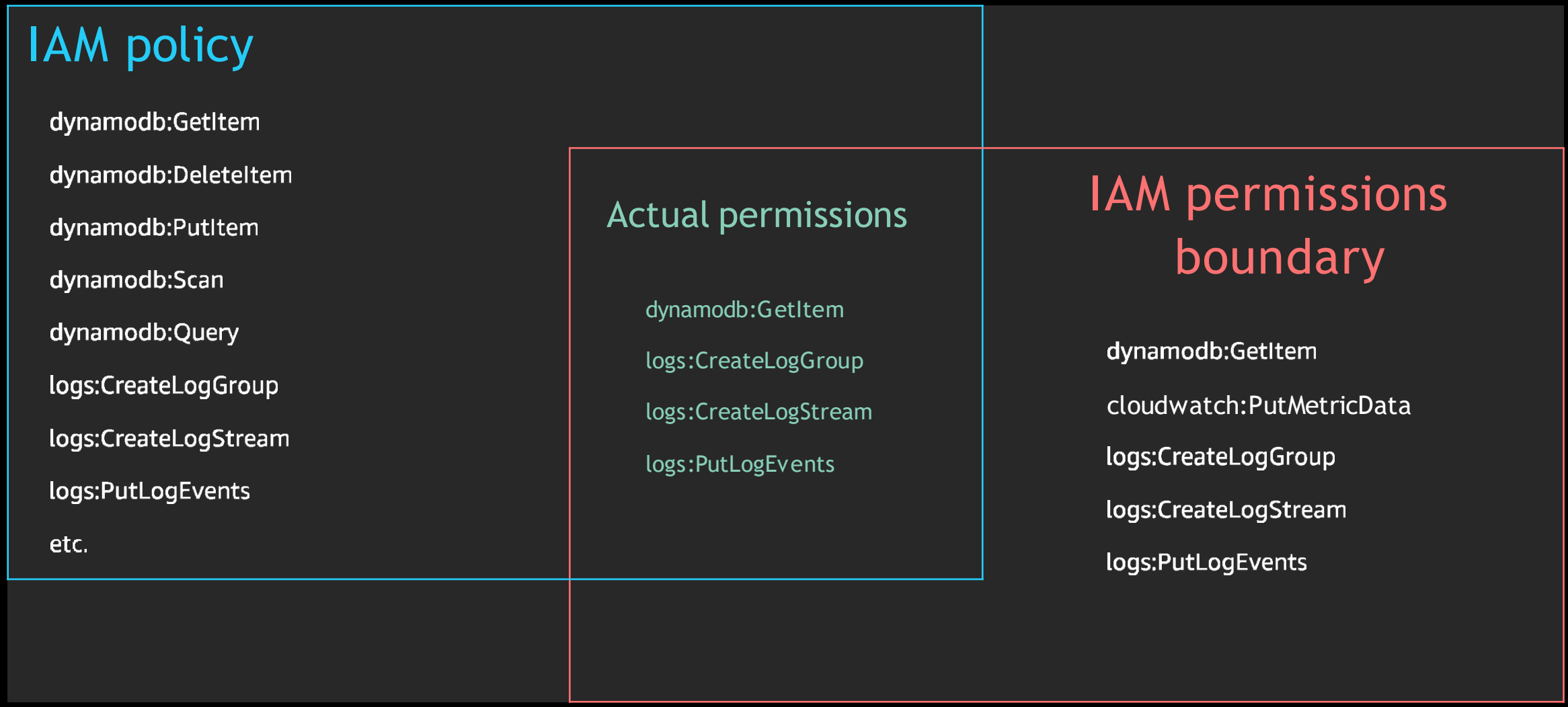
- Apply to IAM entities (users or roles)
- Use a managed policy to set maximum permissions that can be granted to an IAM entity
- That entity can only perform actions allowed by *both* its identity-based policies and the permissions boundary
- The AWS::Serverless::Function resource allows you to pass a permission boundary policy

Permissions boundary in action

```
TestPermissionsBoundary:
   Type: AWS::IAM::ManagedPolicy
   Properties:
     PolicyDocument:
       Version: 2012-10-17
       Statement:
         - Effect: Allow
           Action:
             - dynamodb:GetItem
             - cloudwatch:PutMetricData
             - logs:CreateLogGroup
             - logs:CreateLogStream
             - logs:PutLogEvents
           Resource: "*"
```

```
 HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: ruby2.5
      Policies:
        - DynamoDBCrudPolicy:
            TableName: !Ref AppTable
      PermissionsBoundary: !Ref
TestPermissionsBoundary
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
```

Resulting permissions



Permissions boundary from AWS CodePipeline

- Pattern: Create a permissions boundary policy in your CI/CD toolchain
- Configure your CloudFormation role to only be allowed to create AWS IAM roles that include this permissions boundary.
- Pass the permissions boundary managed policy ARN to your application template.

Permissions boundary in action

```
TestPermissionsBoundary:
   Type: AWS::IAM::ManagedPolicy
   Properties:
     PolicyDocument:
       Version: 2012-10-17
       Statement:
         - Effect: Allow
           Action:
             - dynamodb:GetItem
             - logs:CreateLogGroup
             - logs:CreateLogStream
             - logs:PutLogEvents
           Resource: "*"
```

```
def lambda_handler(event:, context:)
  # this uses GetItem
  item = AppTable.find(hkey: "foo")
  return {
    statusCode: 200,
    body: item.body
  }
end
```

Permissions boundary in action

```
TestPermissionsBoundary:
   Type: AWS::IAM::ManagedPolicy
   Properties:
     PolicyDocument:
       Version: 2012-10-17
       Statement:
         - Effect: Allow
           Action:
             - dynamodb:GetItem
             - logs:CreateLogGroup
             - logs:CreateLogStream
             - logs:PutLogEvents
           Resource: "*"
```

```
def lambda_handler(event:, context:)
  # this uses Scan
  item = AppTable.scan.first
  return {
    statusCode: 200,
    body: item.body
  }
end
```

AWS CloudFormation

Name the stack with a prefix

Validate resource types and properties

```
main:
    Type: AWS::Serverless::Function
    Properties:
       Handler: main
       Runtime: go1.x
       CodeUri: ./services/main
       Role: "lambda@default"
       Timeout: 30
```

AWS Lambda

Name the Lambda with a prefix

VpcConfig subnets and security groups should be **allowed**

```
main:
    Type: AWS::Serverless::Function
    Properties:
      Name: <replaced>
      VpcConfig:
        SecurityGroupIds:
          - sg-123
        SubnetIds:
          - subnet-123
      Handler: main
      Runtime: go1.x
      CodeUri: ./services/main
```

AWS Lambda: Events

Events (e.g., Amazon S3, Amazon SNS, Amazon SQS)

- Should be **valid**
- Resources should be **allowed**

Internal references for most properties

```
main:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        SQS:
          Type: SQS
          Properties:
            Queue: arn::sqs:queue
        APIGet:
          Type: Api
          Properties:
            RestApiId: !Ref api
            Path: /
            Method: GET
```

IAM

Use **IAM permissions boundaries** with `PermissionsBoundary`

Role is **allowed** to be attached to Lambda

Policy templates like `DynamoDBCrudPolicy` force `TableName` to be a reference to a table in the same template

Check tags on resources, e.g., AWS KMS keys

```
main:
    Type: AWS::Serverless::Function
    Properties:
      PermissionsBoundary: <replaced>
      Role: "lambda@default"
      Policies:
      - DynamoDBCrudPolicy:
          TableName: !Ref dynamodb
      - KMSDecryptPolicy:
          KeyId: "alias/env"
      - VPCAccessPolicy: {}
```

Amazon DynamoDB (SimpleTable)

Name database with a prefix

Deletion policy default to `Retain`

```
dynamodb:
    Type: AWS::Serverless::SimpleTable
    DeletionPolicy: <default>
    Properties:
      Name: <replaced>
      PrimaryKey:
        Name: primary_key
        Type: String
```

Amazon API Gateway

- Name gateway with a prefix
- Default `EndpointConfiguration` to `PRIVATE`
- Replace `x-amazon-apigateway-policy`

```
api:
  Type: AWS::Serverless::Api
  Properties:
    Name: <replaced>
    StageName: api
    EndpointConfiguration: <default>
    DefinitionBody:
      openapi: "3.0.0"
      x-amazon-apigateway-policy: <replaced>
      paths:
        "/": ...
```

Amazon API Gateway

Name gateway with a prefix

Default `EndpointConfiguration` to `PRIVATE`

Replace `x-amazon-apigateway-policy`

```
api:
  Type: AWS::Serverless::Api
  Properties:
    DefinitionBody:
      openapi: "3.0.0"
      x-amazon-apigateway-policy: <replaced>
      paths:
        "/": ...
```

```
x.amazon-apigateway-policy:
  Version: "2012-10-17"
  Statement:
    - Effect: "Allow"
      Principal: "*"
      Action:
      -   "execute-api:Invoke"
      Resource: "execute-api:/*"
    - Effect: "Deny"
      Principal: "*"
      Action:
      - "execute-api:Invoke"
      Resource: "execute-api:/*"
      Condition:
        StringNotEquals:
          aws:SourceVpc:
            - "vpc-123"
```

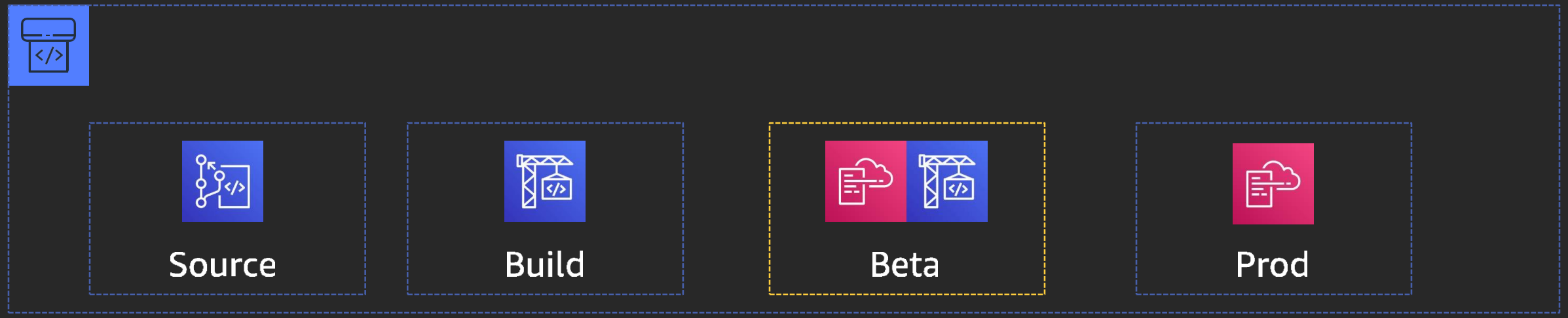
AWS SAM Local

- Uses Docker to simulate execution on AWS Lambda
- Several supported modes:
  - sam local start-api (Create an endpoint that simulates your API Gateway endpoint)
  - sam local start-lambda (Create an endpoint that simulates the Lambda API)
  - sam local invoke (Single invocation)
- Useful for quick development cycles/iterations
- With IDEs, can do step-through debugging

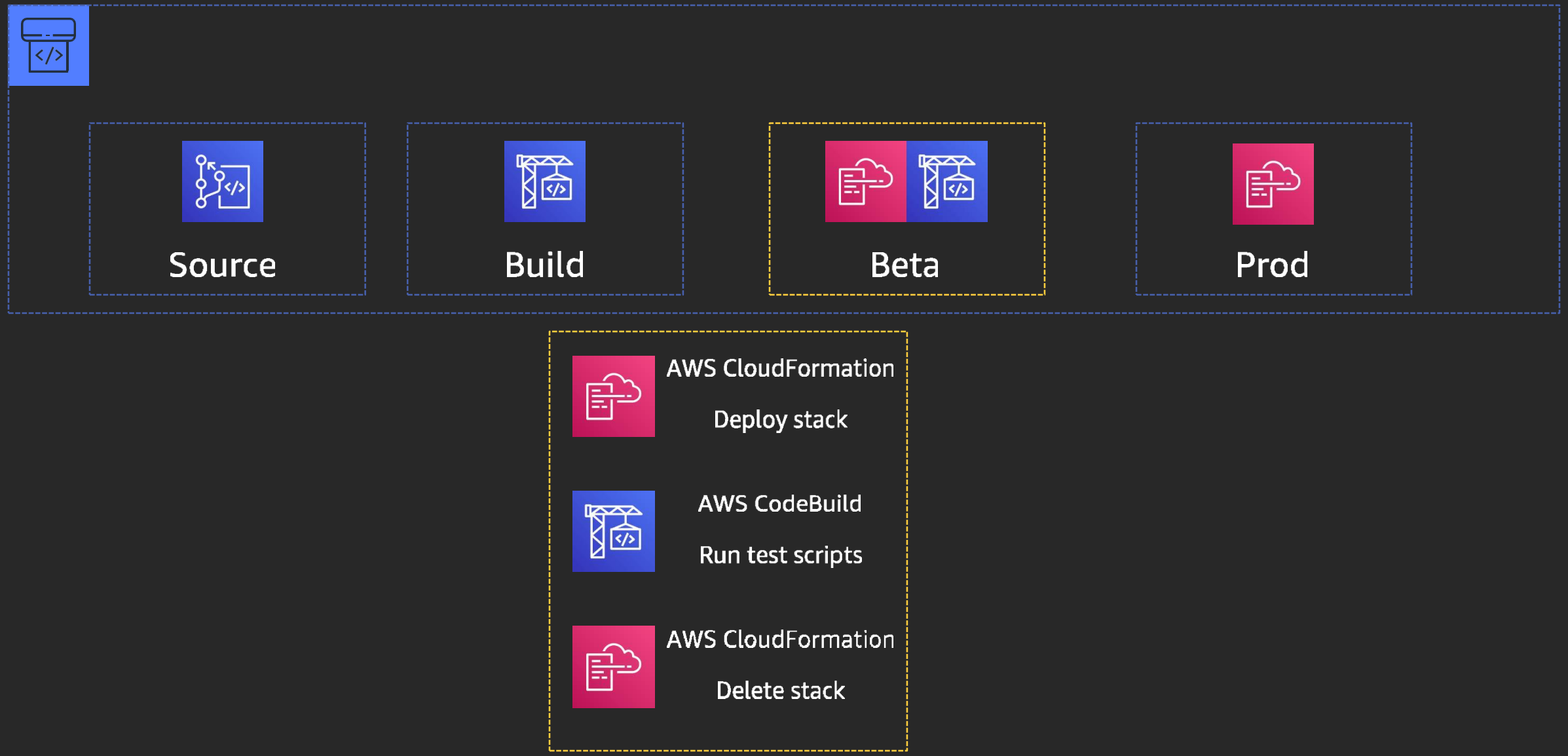
Live integration testing

- Serverless + AWS CloudFormation = Easy to test on prod-like environment
- Example AWS CodePipeline stage:
  - AWS CloudFormation action: Create change set (beta stack)
  - AWS CloudFormation action: Execute change set (beta stack)
  - AWS CodeBuild: Run integration test scripts against beta environment
  - AWS CloudFormation action: Delete stack (beta stack)

Live integration testing



Live integration testing



Testing summarized

- AWS SAM Local testing is helpful for experimental testing on a developer machine
- AWS CloudFormation + AWS CodePipeline = 😎
- All of this provides a feedback loop as you develop, and as you ship changes to your application

Summary

- Use AWS SAM CLI's init function to ingrain your best practices
- IAM provides a number of ways to secure your applications
- Local and remote testing
- AWS SAM provides safe deployment helpers

Building and Hosting with AWS Amplify

AWS Amplify

- Deploy and manage backend resource categories using the Command Line Interface
- Integrate your frontend application with those resources using Client Libraries
- Expedite development with UI Components
- Manage your application hosting and CI/CD with the Amplify Console

Amplify Categories

Authentication Add user signup, signin, and access control

API Make HTTP requests using REST and GraphQL, easily and securely

Analytics Drop in analytics to track user sessions, attributes, and in app metrics

Storage Manage user content securely in the cloud or on the device

DataStore Shared data for offline and online scenarios

PubSub Connectivity with cloud based, message oriented middleware

Push Notifications Push notifications with analytics and targeting

Interactions Enable AI powered chatbots in your web or mobile apps

Predictions Use AI and ML cloud services to enhance your application

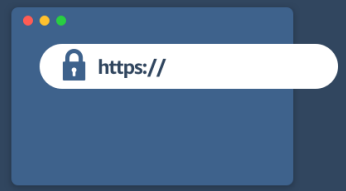
Geo APIs and map UI components for maps and location search

Amplify Command Line Interface

```
#Install the CLI
npm install -g @aws-amplify/cli

#Initialize your amplify project
amplify init

#Add an API to your project
amplify api add

#Use custom plugins to extend your projects scope
amplify video add

#Push your local project to the cloud to auto-deploy your resources
amplify push
```

Amplify Hosting

Globally available



Easy custom domain setup

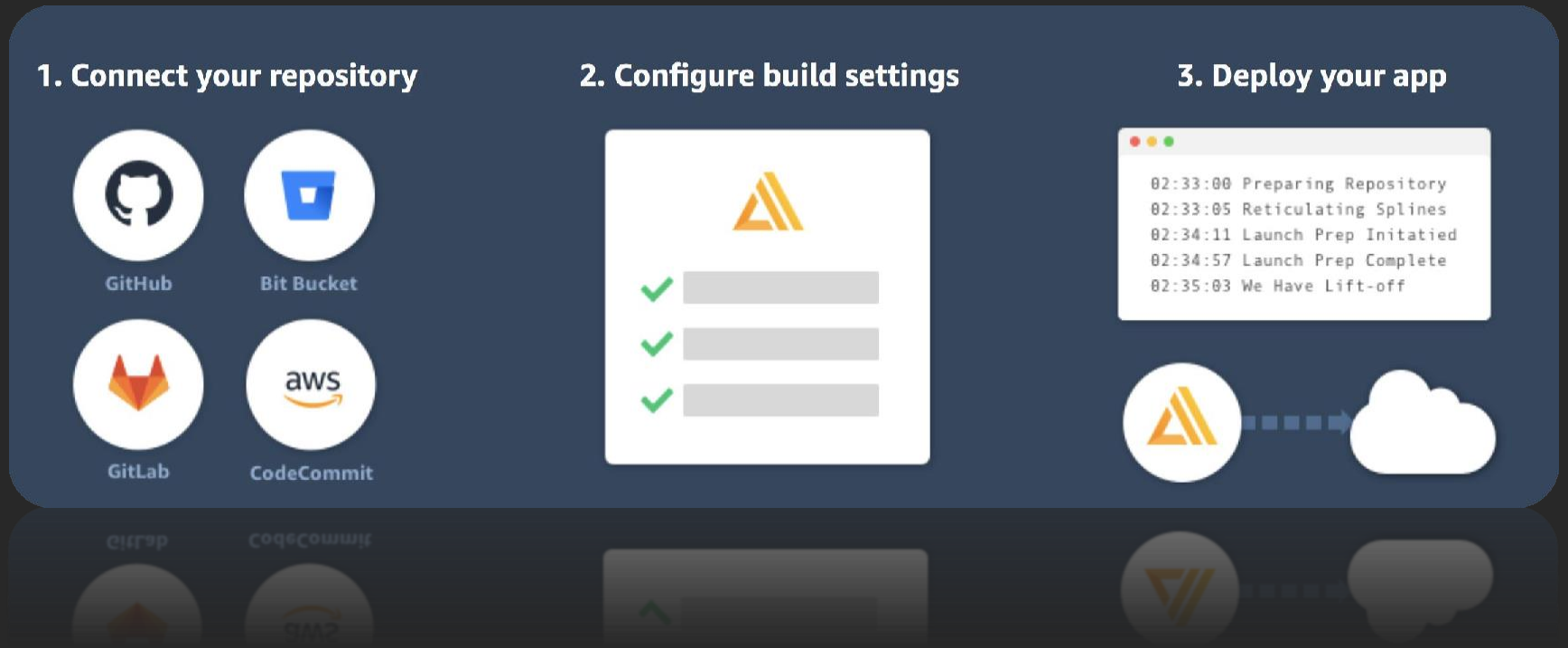
Simplified continuous workflows

Feature branch deployments

Atomic deployments

Password protection

Amplify Hosting: How it Works



AWS Amplify

Create mobile and web applications

Amplify Studio

Visually build a full-stack app, both front-end UI and a backend.

Amplify CLI

Configure an Amplify backend with a guided CLI workflow

Amplify Libraries

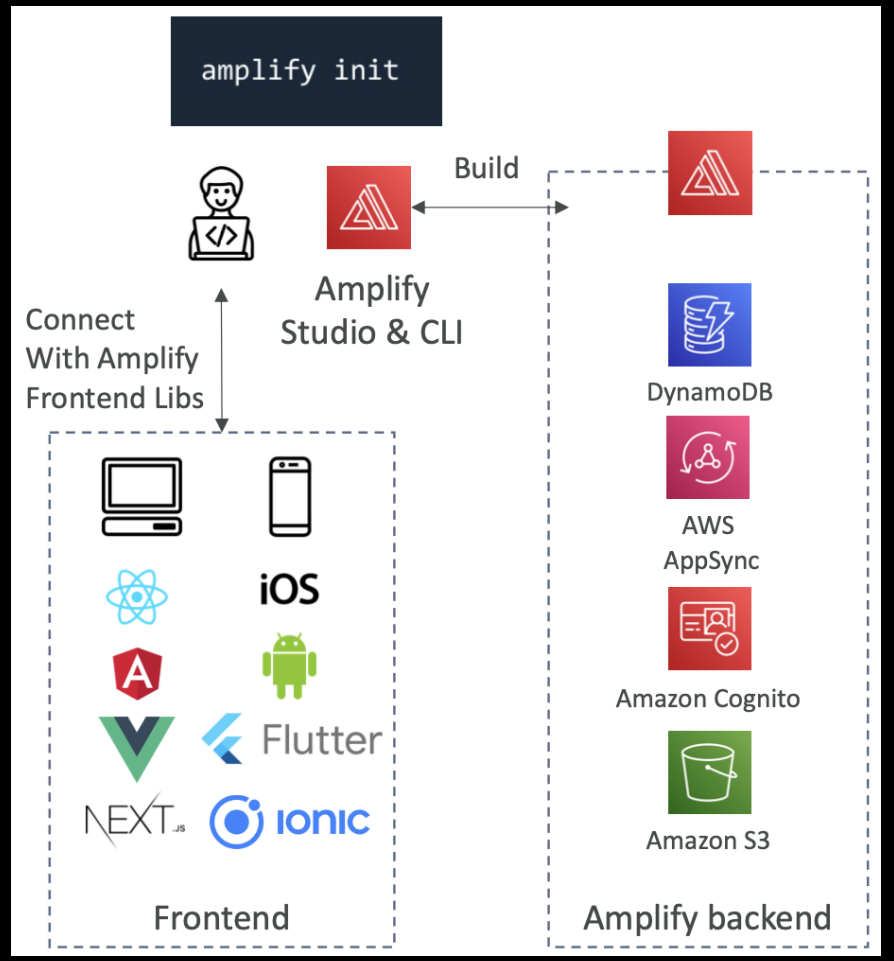
Connect your app to existing AWS Services (Cognito, S3 and more)

Amplify Hosting

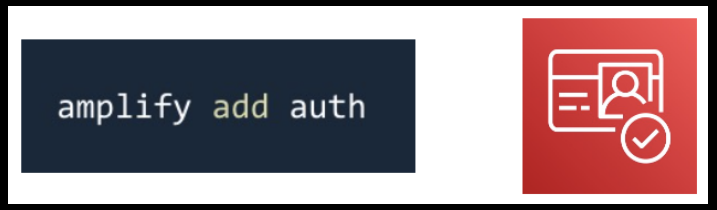
Host secure, reliable, fast web apps or websites via the AWS content delivery network.

AWS Amplify

- Set of tools to get started with creating mobile and web applications
- “Elastic Beanstalk for mobile and web applications”
- Must-have features such as data storage, authentication, storage, and machine-learning, all powered by AWS services
- Front-end libraries with ready-to-use components for React.js, Vue, Javascript, iOS, Android, Flutter, etc...
- Incorporates AWS best practices to for reliability, security, scalability
- Build and deploy with the Amplify CLI or Amplify Studio



AWS Amplify – Important Features



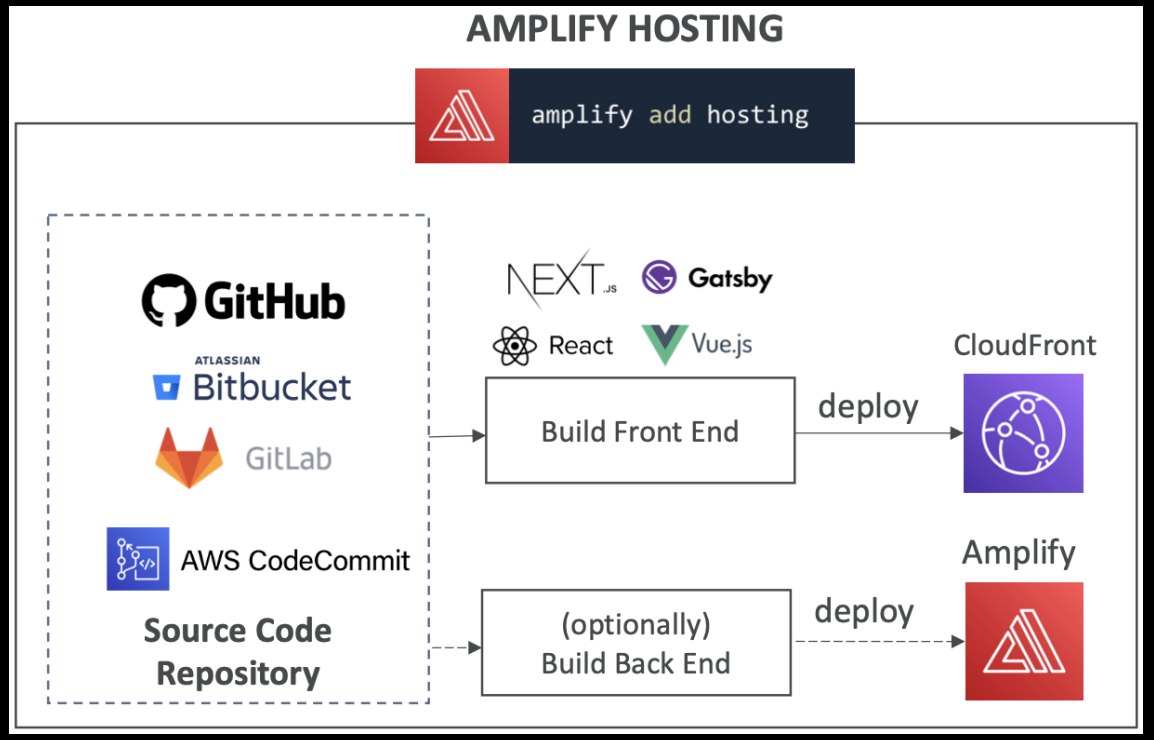
AUTHENTICATION
- Leverages Amazon Cognito

- User registration, authentication, account recovery & other operations
- Support MFA, Social Sign-in, etc...
- Pre-built UI components
- Fine-grained authorization

DATASTORE
Leverages Amazon AppSync and Amazon DynamoDB
Work with local data and have automatic synchronization to the cloud without complex code
Powered by GraphQL
Offline and real-time capabilities
Visual data modeling with Amplify Studio

AWS Amplify Hosting

- Build and Host Modern Web Apps
- CICD (build, test, deploy)
  - Pull Request Previews
  - Custom Domains
- Monitoring
  - Redirect and Custom Headers
- Password protection



Amazon API Gateway概要

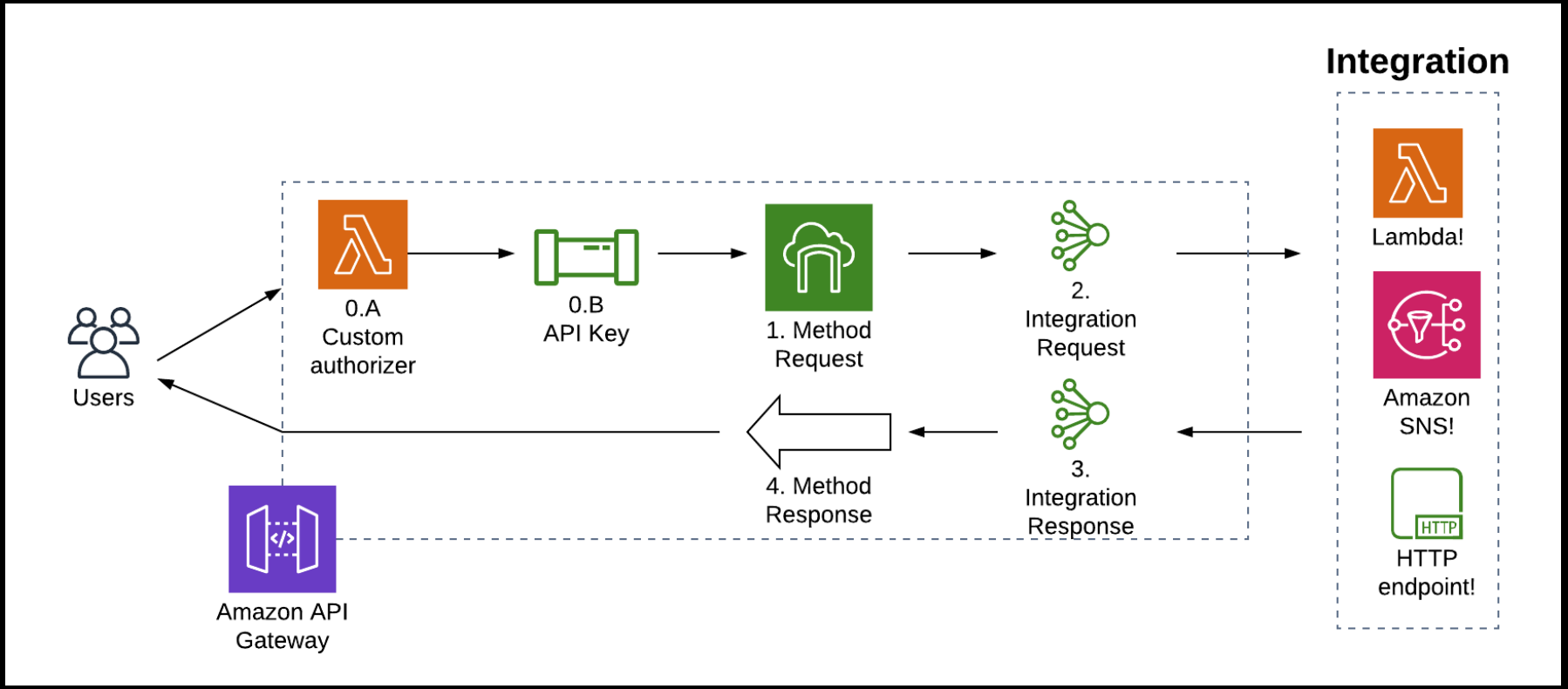
- 是一项全面管理的服务，使开发人员能够轻松地创建、发布、维护、监控和保护任何规模的API。API是应用程序从您的后端服务访问数据、业务逻辑或功能的 "前门"。使用API网关，可以创建RESTful APIs、HTTP APIs或WebSocket APIs（实现实时双向通信应用）。API Gateway支持容器化和无服务器工作负载，以及网络应用。
- API Gateway处理所有涉及接受和处理多达数十万个并发API调用的任务，包括流量管理、CORS支持、授权和访问控制、节流、监控和API版本管理。API网关没有最低费用或启动成本。你为你收到的API调用和传输出去的数据量付费，通过API Gateway的分级定价模式，你可以随着你的API使用量的增加而降低你的成本。

Amazon API Gateway

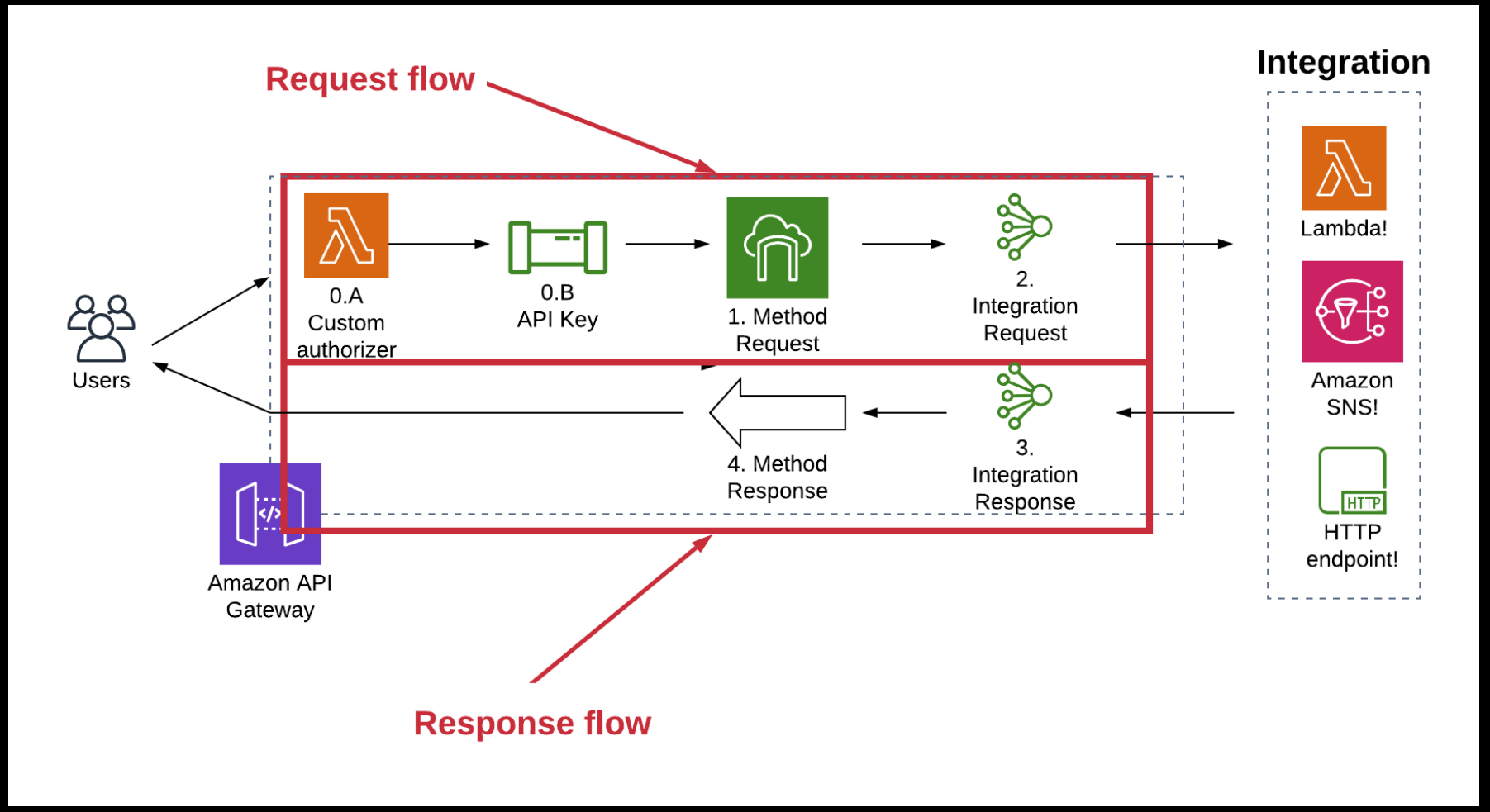
- Before this class, your understanding of API Gateway might be like the following:

Amazon API Gateway

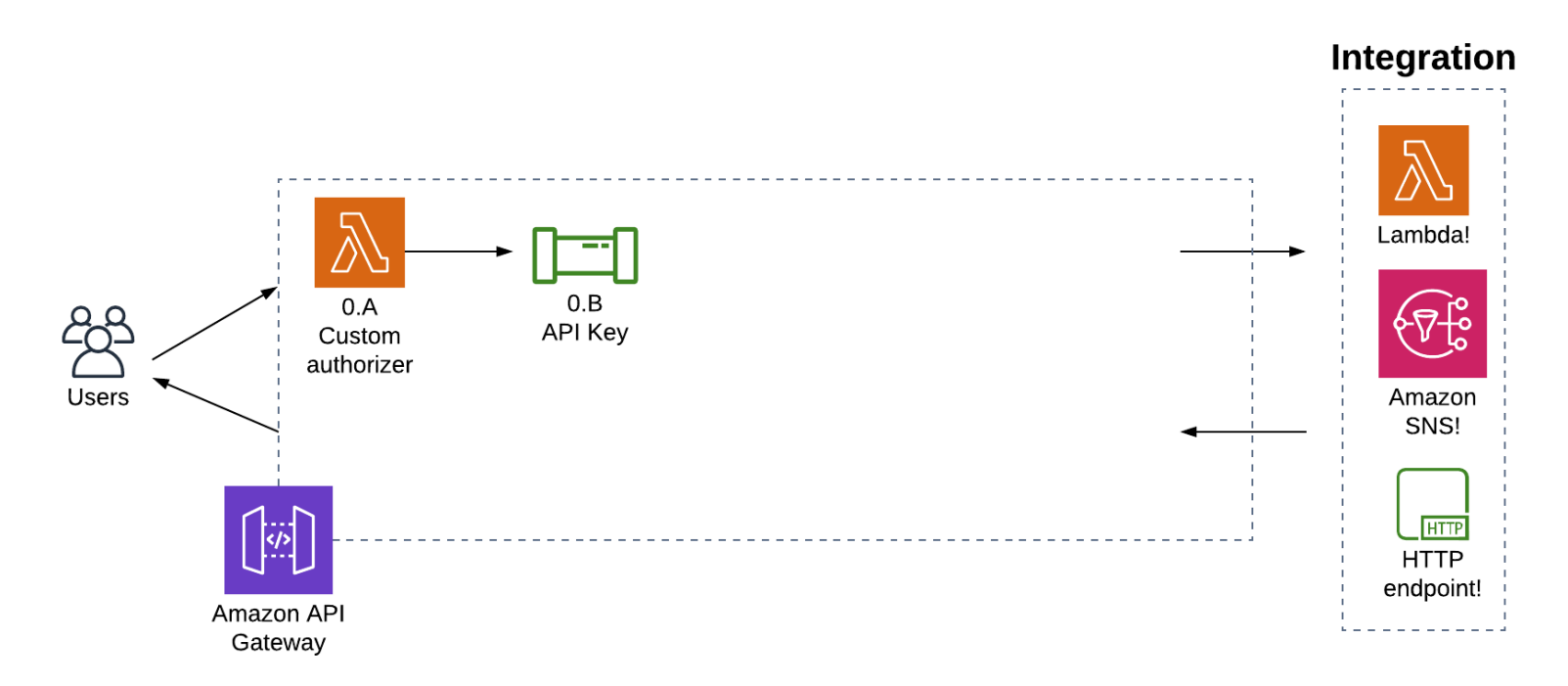
- After this class, you'll understand all the pieces in the following diagram:



HTTP request flow



Step 0: Protecting your API with Authorization and Usage Plans



- You can throttle a particular user by using *API keys*.
- The authorization check runs before the API key check.
- You may use just an authorizer, just an API key, both, or neither.

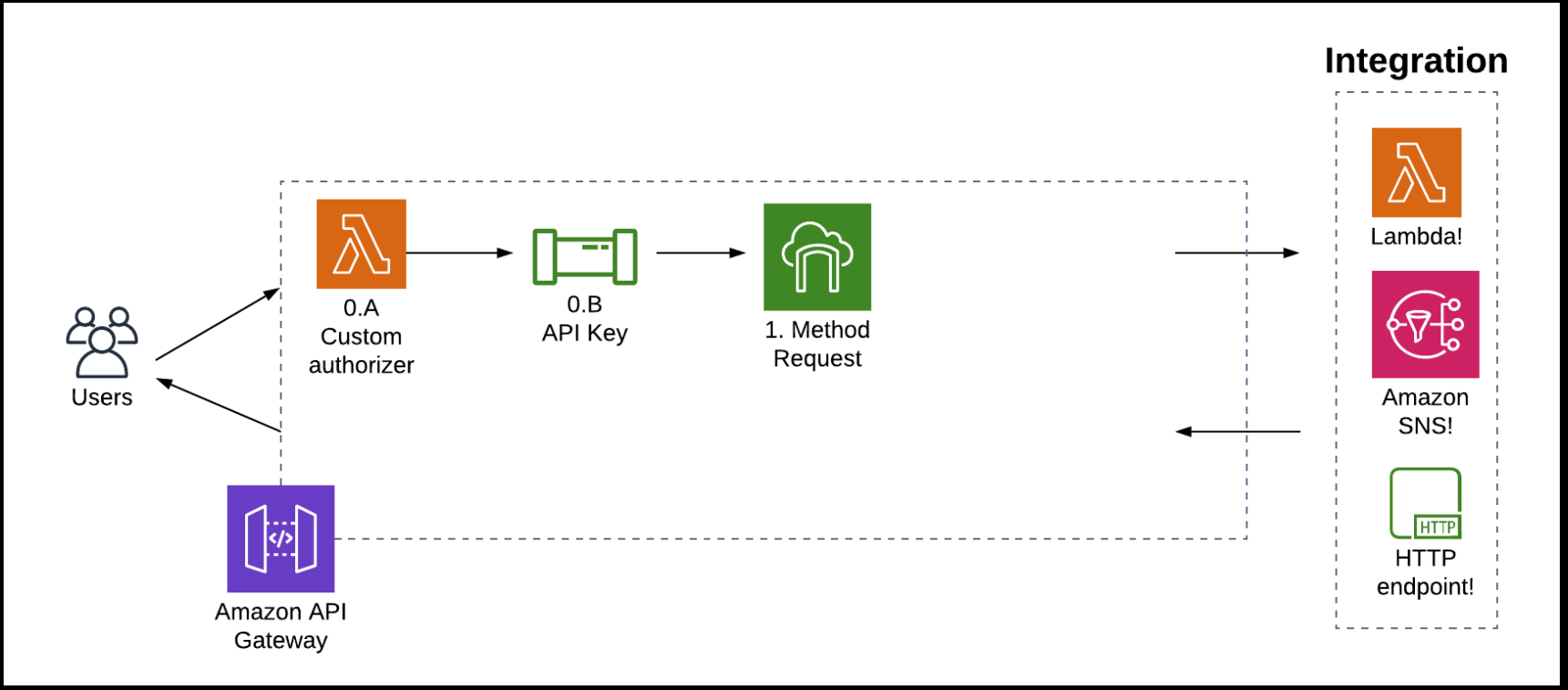
Key Takeaways from Step 0: Authorization

Here are the key takeaways from this section:

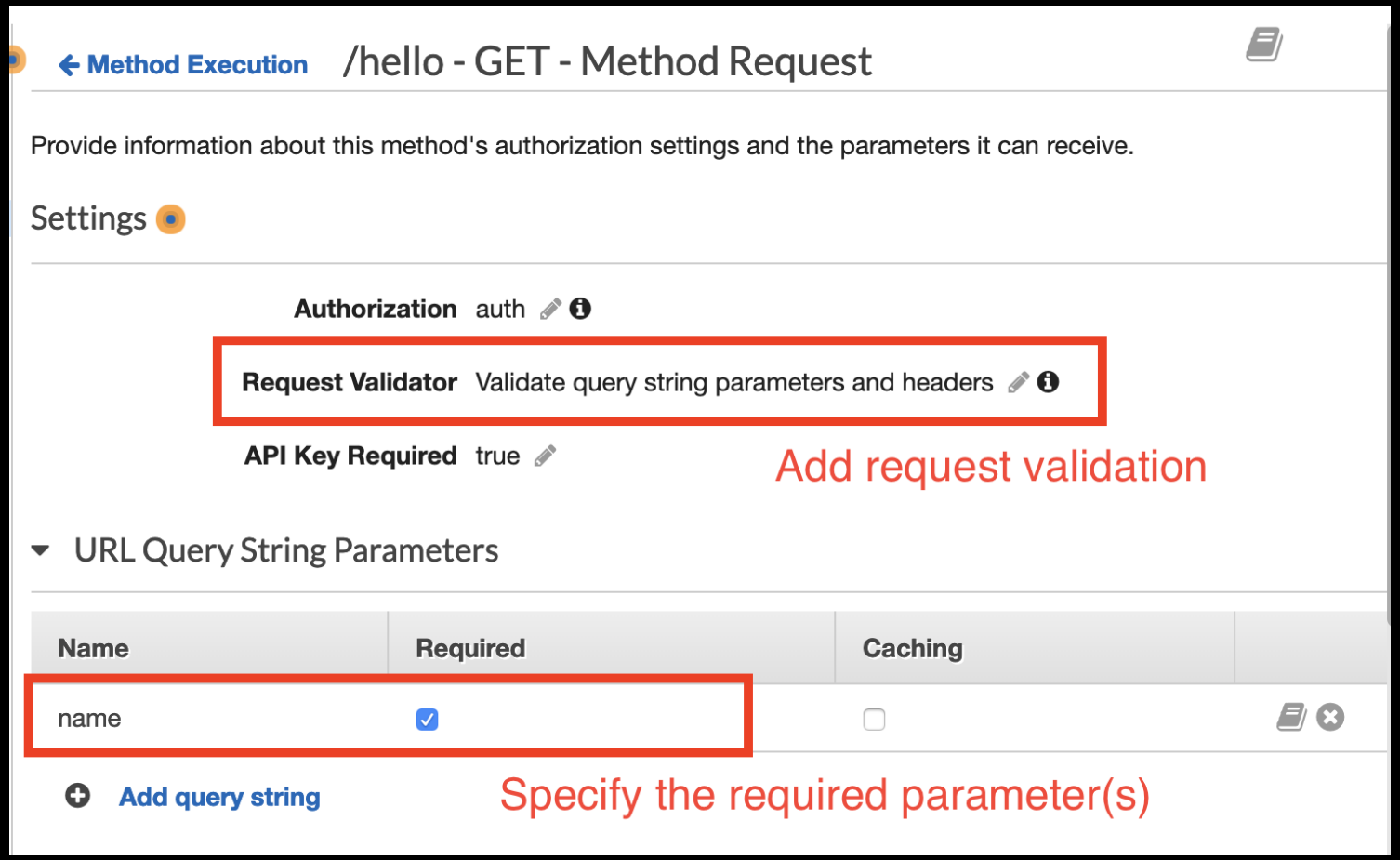
- Authorization is a completely optional step. You don't need to include any authorization on your API.
- Using authorization in API Gateway can protect your downstream resources from excess load.
- You can *authorize* a request by using Cognito User Pools, AWS IAM, or a Lambda custom authorizer.

Step 1: Validation with Method Requests

Method requests are like the public interface of your API: they define what your endpoint expects, which elements are required, and more.



Validating Parameters



With API Gateway method requests, you can specify these parameters and make them required if desired. If a client fails to provide the parameter, the client will receive a 400 Bad Request response with a payload.

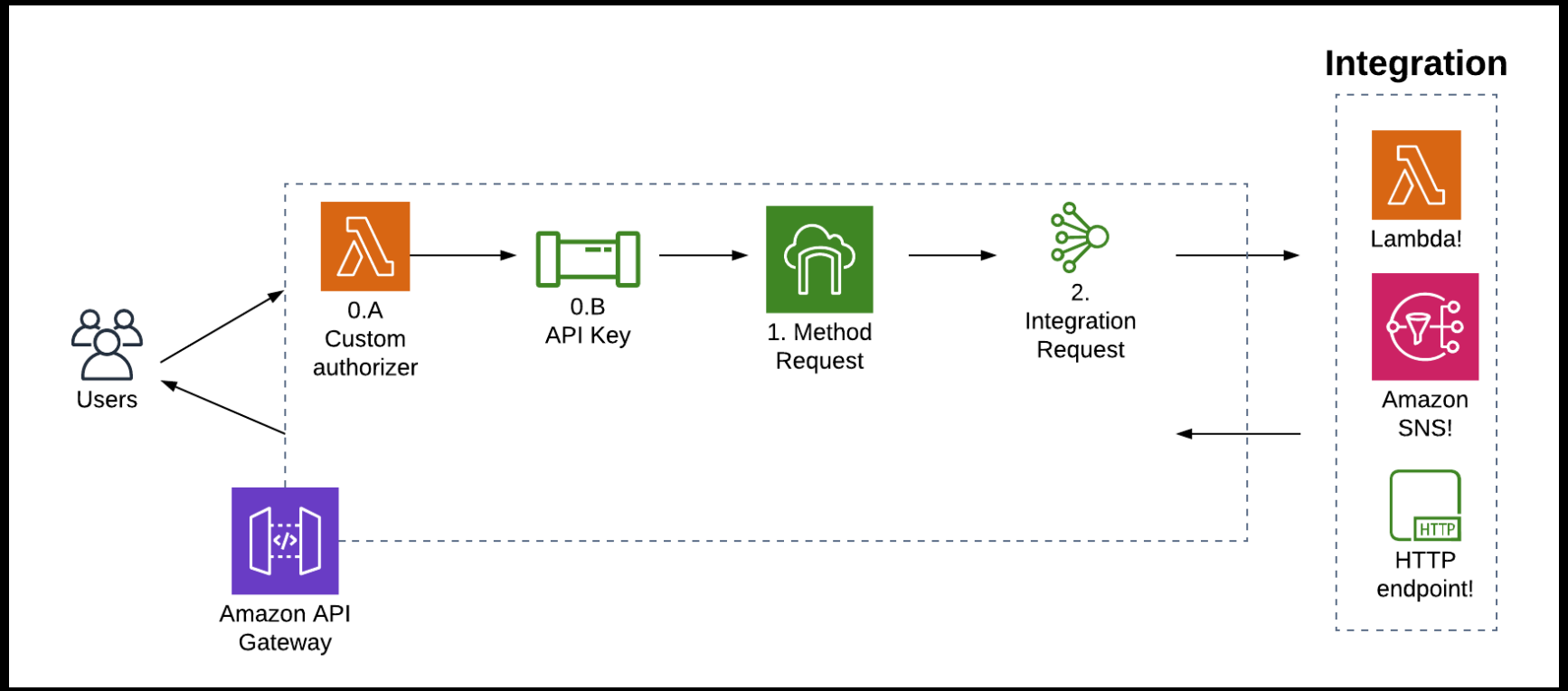
```
{ "message": "Missing required request parameters: [<parameter>]" }
```

Validating the Request Body using Request Models

```
UserModel:
  Type: AWS::ApiGateway::Model
  Properties:
    RestApiId:
      Ref: RestApi
    ContentType: "application/json"
    Description: "User Model"
    Name: UserModel
    Schema:
      "$schema": "http://json-schema.org/draft-04/schema#"
      title: UserModel
      type: object
      properties:
        name:
          type: string
        age:
          type: integer
        username:
          type: string
```

Step 2: Transforming the request with the Integration Request

The integration request step is about transformation.
Our request has come in from our client, but we may need to reshape that request to make it ready for our backend integration.



Writing Mapping Templates with VTL

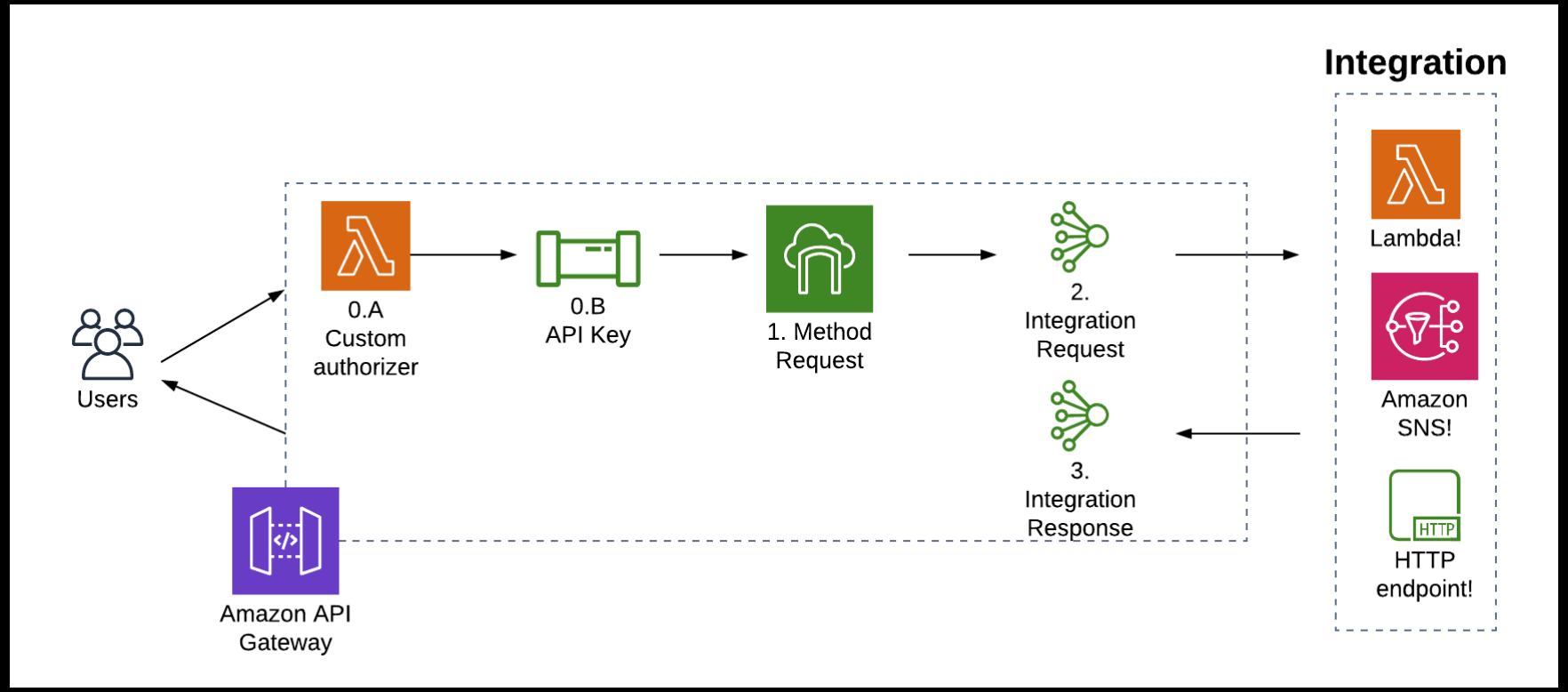
To transform the initial request into your integration request, you need to write a mapping template. Like request body validation, a mapping template is associated with a particular Content-Type for an HTTP request.

```
"Action=Publish
&TopicArn=$util.urlEncode('<yourTopicArn>')
&Message=$util.urlEncode($input.body)"
```

This example returns a simple x-www-form-urlencoded string that uses some utility methods to URL encode some other properties. It uses a hard-coded SNS Topic ARN that only API Gateway knows, as well as the request body (accessed using $input.body).

```
#set($inputRoot = $input.path('$'))
{
  "DueDate": "$inputRoot.DueDate",
  "Balance": $inputRoot.Balance,
  "DocNumber": "$inputRoot.DocNumber",
  "Status": "$inputRoot.Status",
  "Line": [
#foreach($elem in $inputRoot.Line)
    {
      "Description": "$elem.Description",
      "Amount": $elem.Amount,
      "DetailType": "$elem.DetailType",
      "ExpenseDetail": {
        "Customer": {
          "value": "$elem.ExpenseDetail.Customer.value",
          "name": "$elem.ExpenseDetail.Customer.name"
        },
        "Ref": {
          "value": "$elem.ExpenseDetail.Ref.value",
          "name": "$elem.ExpenseDetail.Ref.name"
        },
        "Account": {
          "value": "$elem.ExpenseDetail.Account.value",
          "name": "$elem.ExpenseDetail.Account.name"
        },
        "LineStatus": "$elem.ExpenseDetail.LineStatus"
      }
    }#if($foreach.hasNext),#end

#end
  ],
  "Vendor": {
    "value": "$inputRoot.Vendor.value",
    "name": "$inputRoot.Vendor.name"
  },
  "APRef": {
    "value": "$inputRoot.APRef.value",
    "name": "$inputRoot.APRef.name"
  },
  "TotalAmt": $inputRoot.TotalAmt
}
```

Step 3: Handling your response with Integration Responses



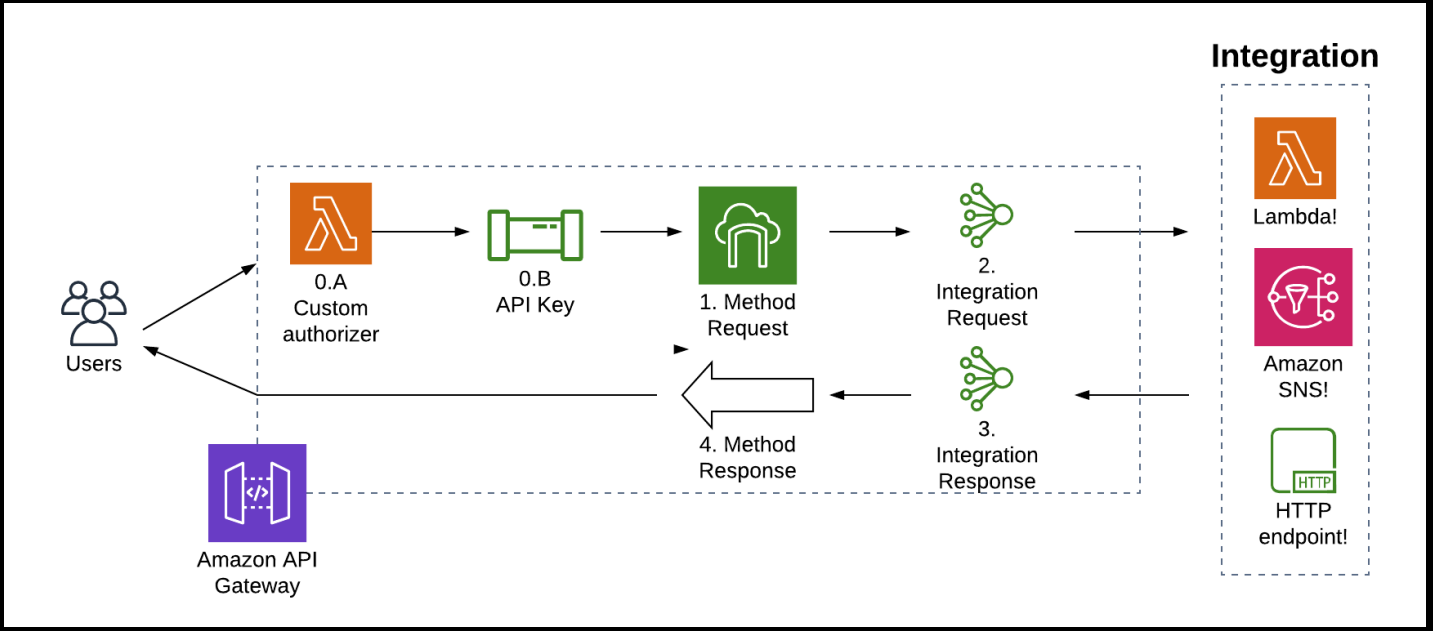
The integration response is a way to internally standardize the response from our integration into something that can be handled next, by our method response. It's essentially the inverse of the integration request.

Step 4: Standardizing your responses with Method Responses

Method responses are validating the output to a client.

Method responses are responsible for two things:

1. Defining the status codes that will be returned by your API.
2. Defining the response bodies that are returned by your API.



Key Takeaways

- API Gateway only returns a 200 OK status code by default. You can add additional status codes by adding method responses.
- You must create your method responses before you can use a given status code in an integration response.

Amazon Cognito

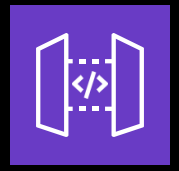
API Gateway authorization

Amazon Cognito user pools  User pool authorizers

API Gateway + Amazon Cognito

Amazon Cognito
user pools

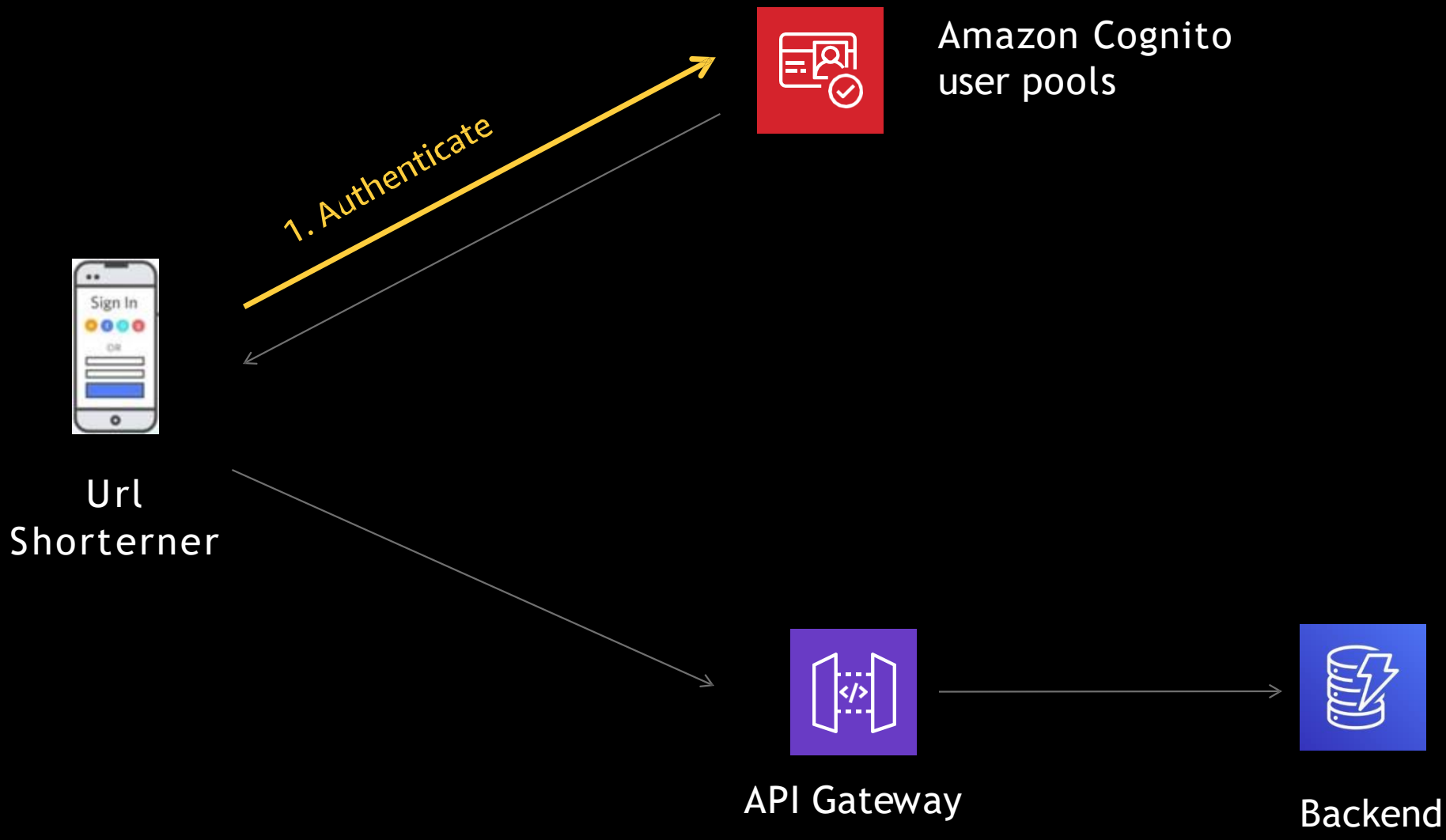
Url
Shorterner



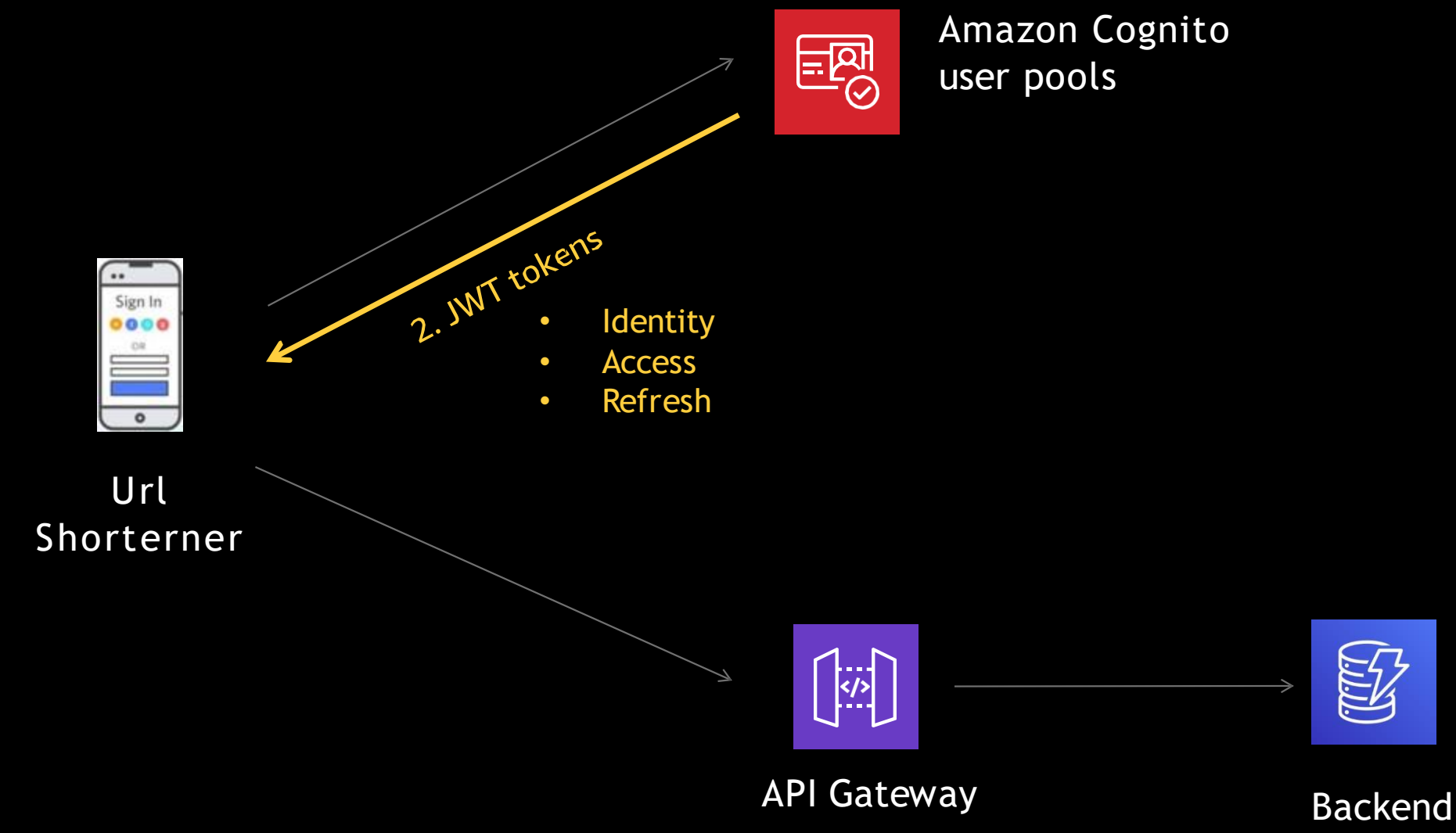
API Gateway

Backend

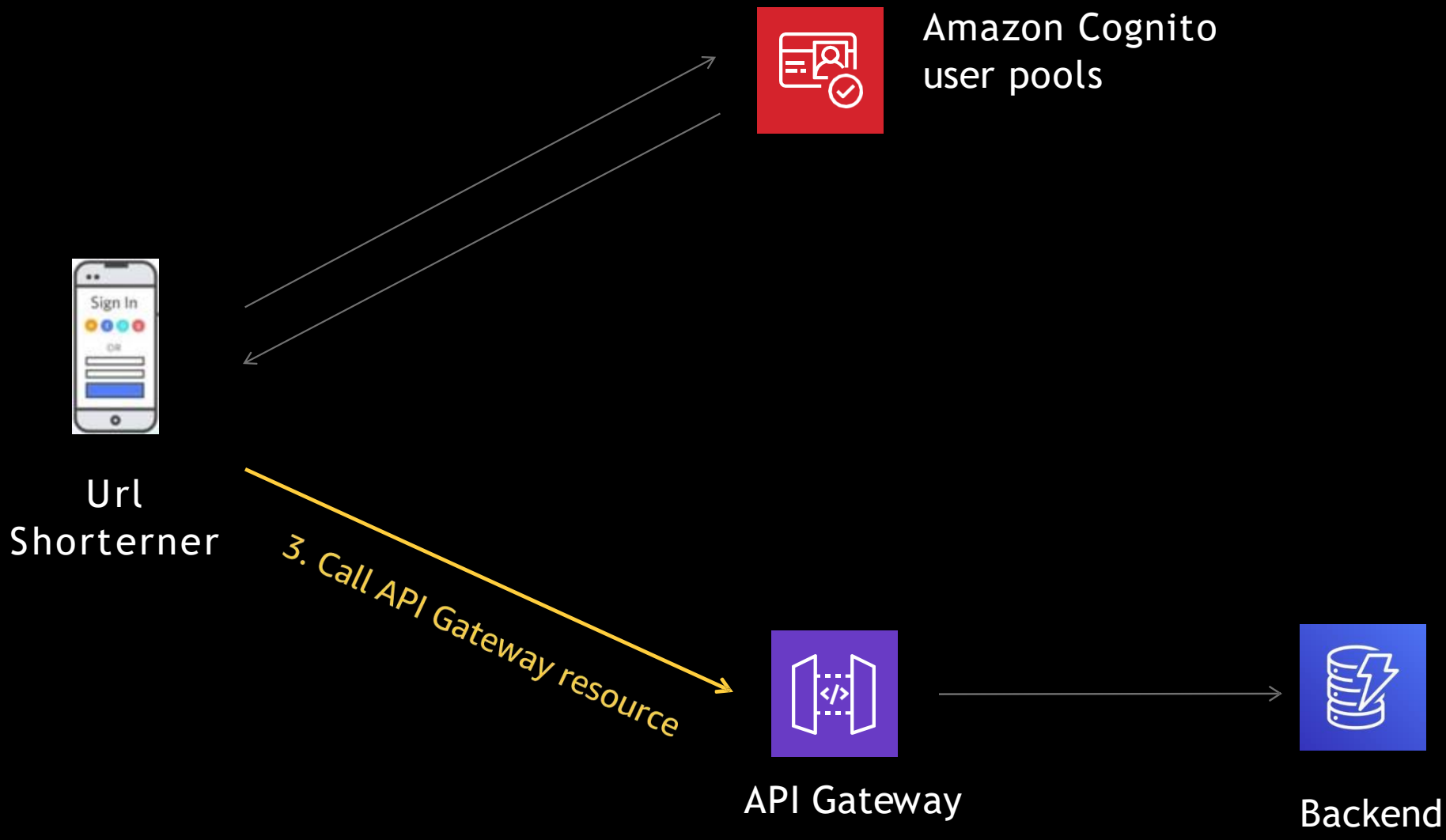
User pool authorizers



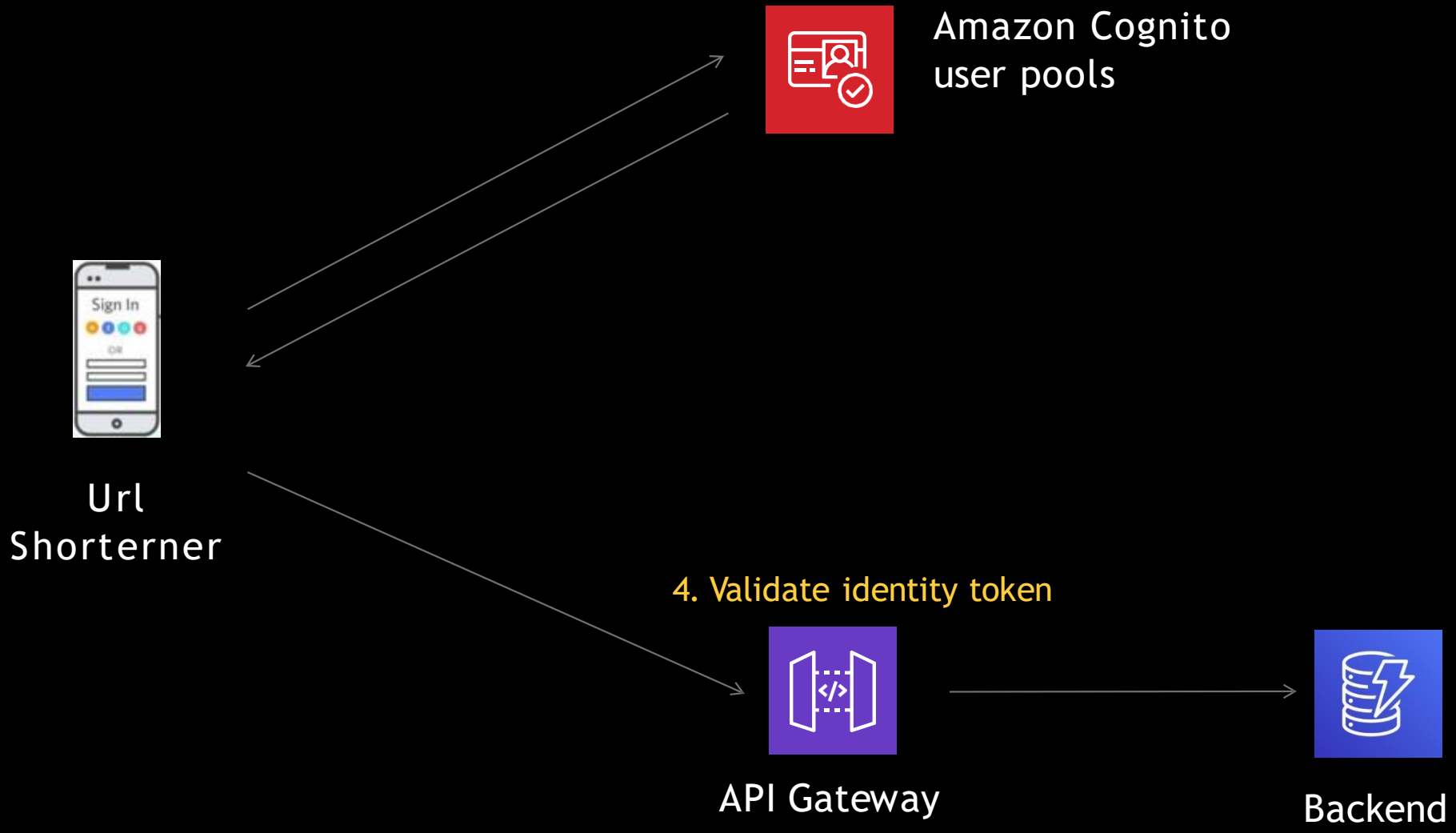
User pool authorizers



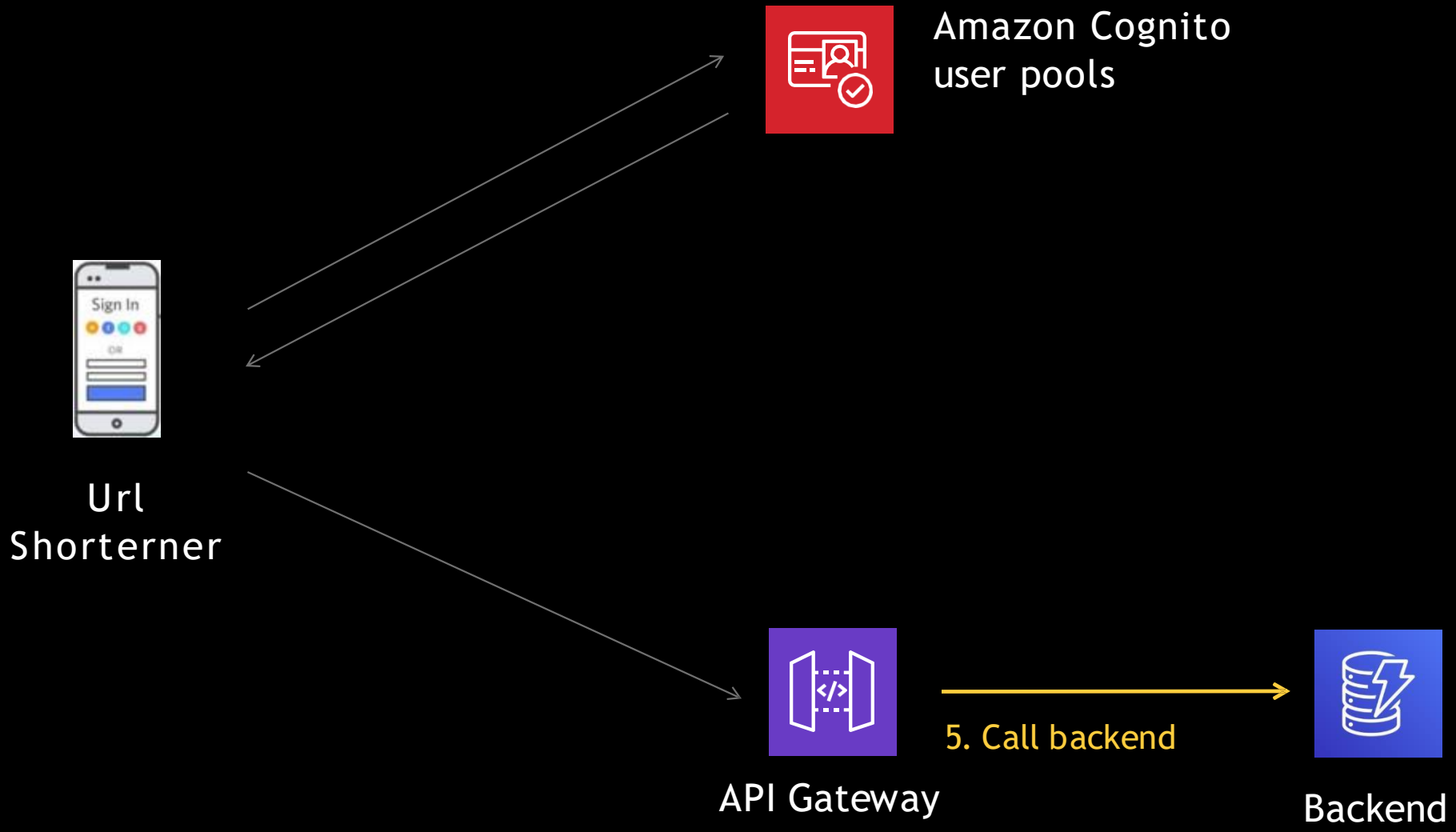
User pool authorizers



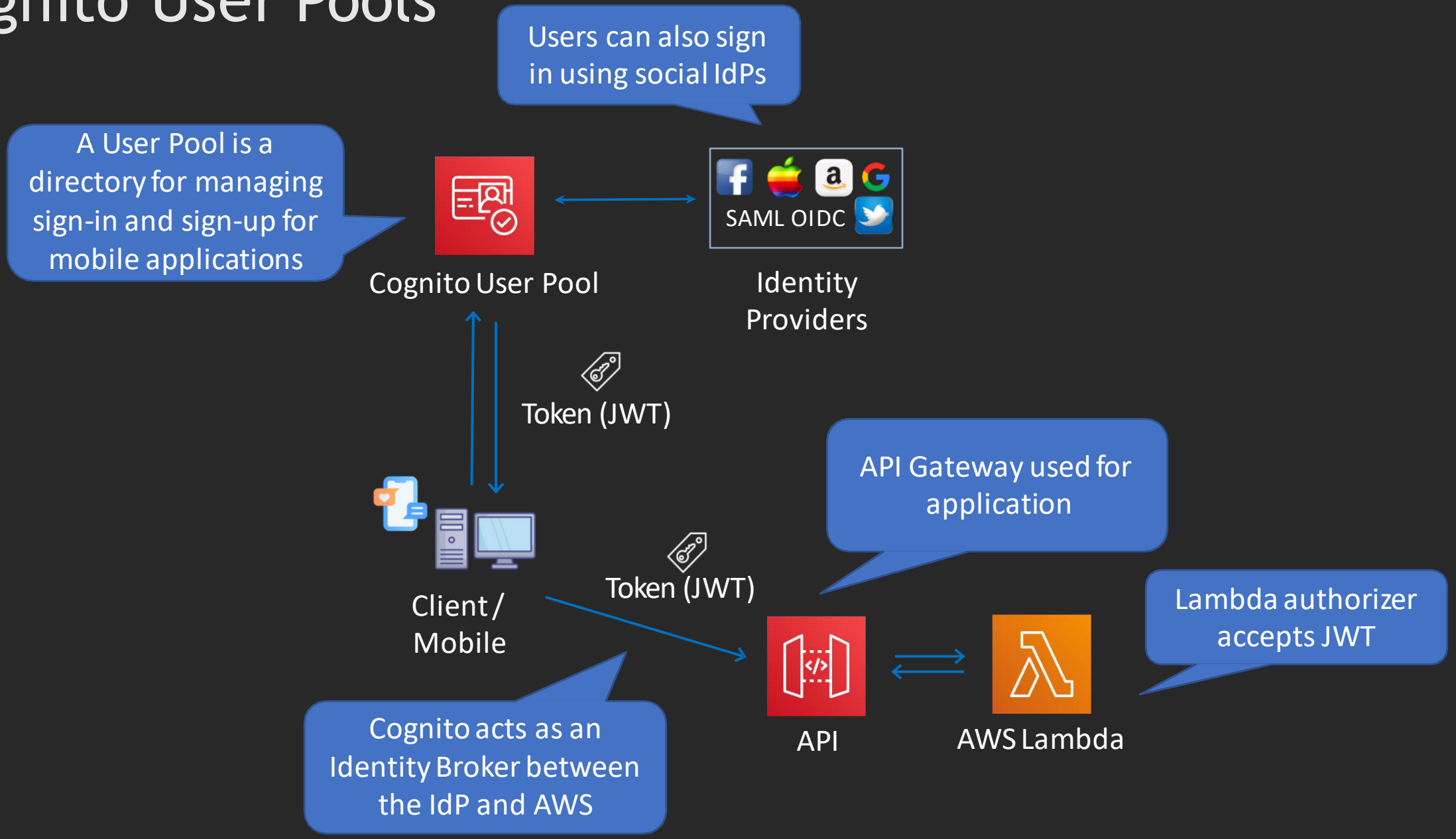
User pool authorizers



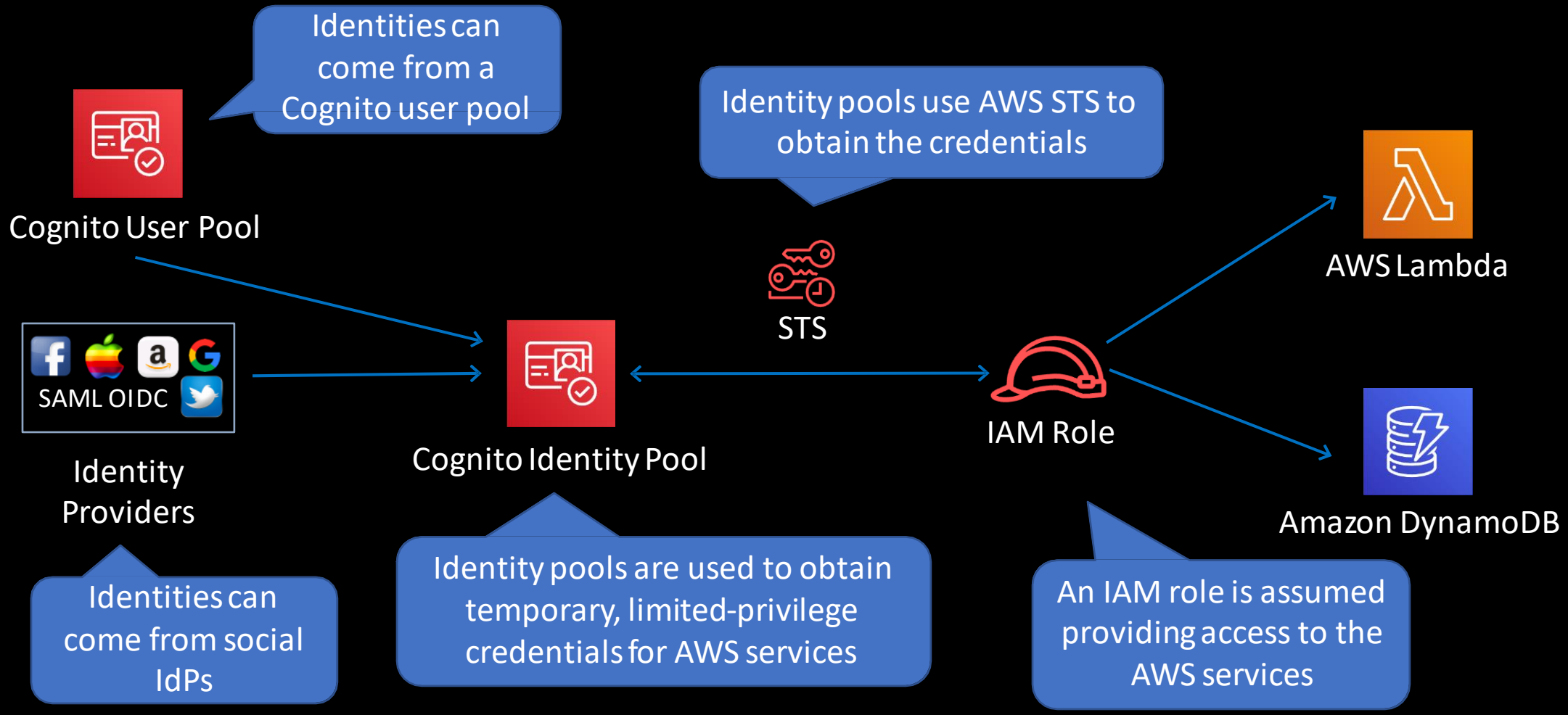
User pool authorizers



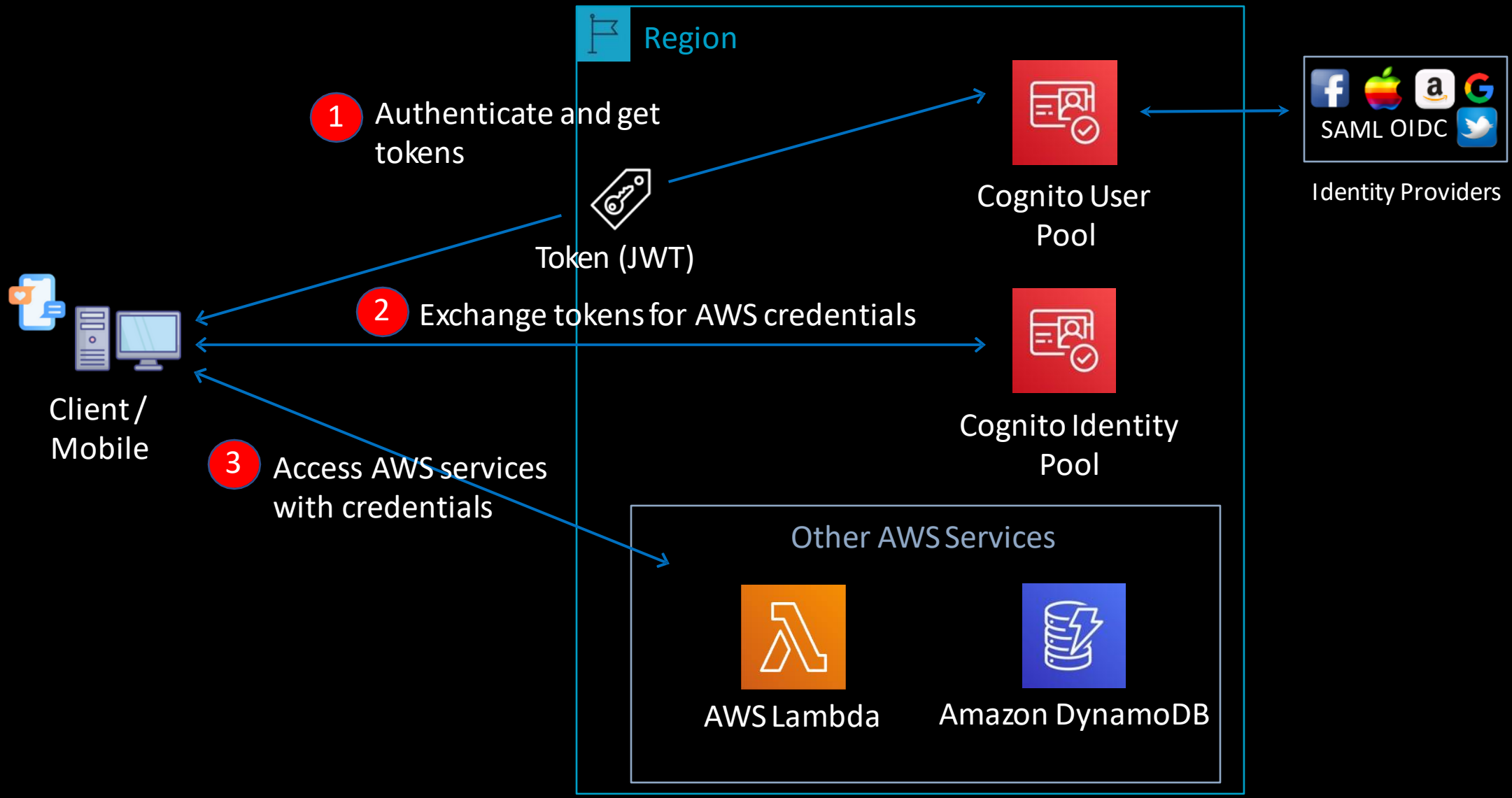
Cognito User Pools



Cognito Identity Pool



User Pools + Identity Pools



CloudFront Architecture

https://learn.cantrill.io adriancantrill

Distribution

http://animals4life.org

http://d111111abcdef8.cloudfront.net

Integrates with ACM for HTTPS

http://animals4life.org

Configure Distribution

Set Origin

Regional Edge Cache

2 Checks Regional Edge

3 Origin Fetch

4

5

1 Checks Edge

Julie

Moss

7 Checks Regional Edge

8

9

6 Checks Edge

Upload Original Content

Upload direct to origins

No Caching

Edge Locations

whiskers.jpg

CloudFront Architecture

Origins

Origins are used by behaviours as content sources

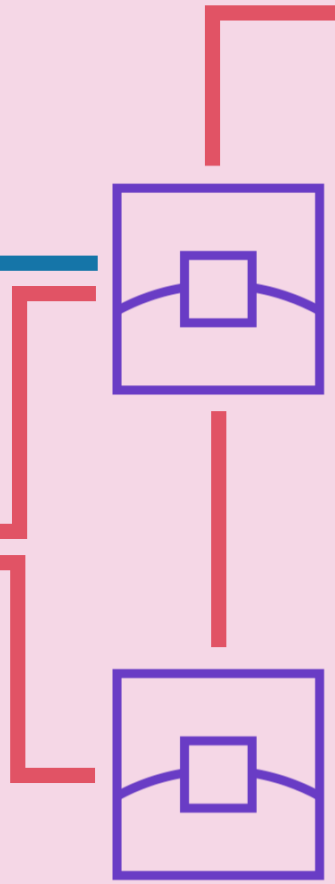
Cache Behaviour

private (img/*)

default(*)

A distribution can have many behaviours each configured with a path pattern. If requests match that pattern - that behaviour is used .. otherwise the default

The distribution contains the configuration deployed to the edge locations



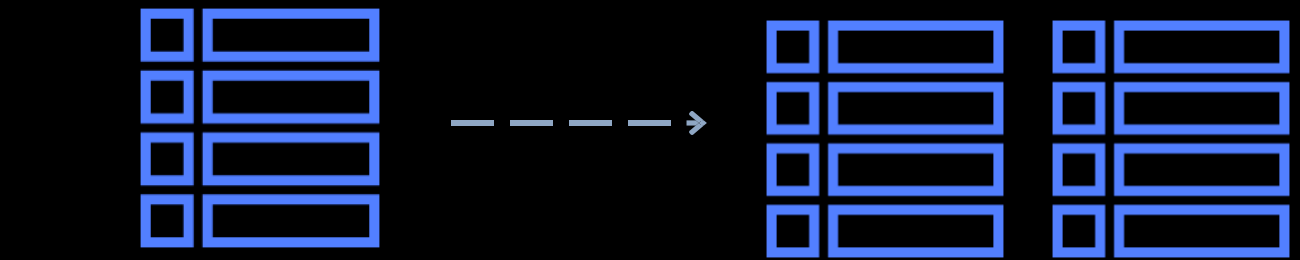
Origins, Origin Groups, TTL, Protocol Policies and restricted access are configured via Behaviours

Amazon DynamoDB Core Knowledge

© Digital Cloud Training | https://digitalcloud.training

Amazon DynamoDB Core Knowledge

- Fully managed NoSQL database service
- Key/value store and document store
- It is a non-relational, key-value type of database
- Fully serverless service
- Push button scaling

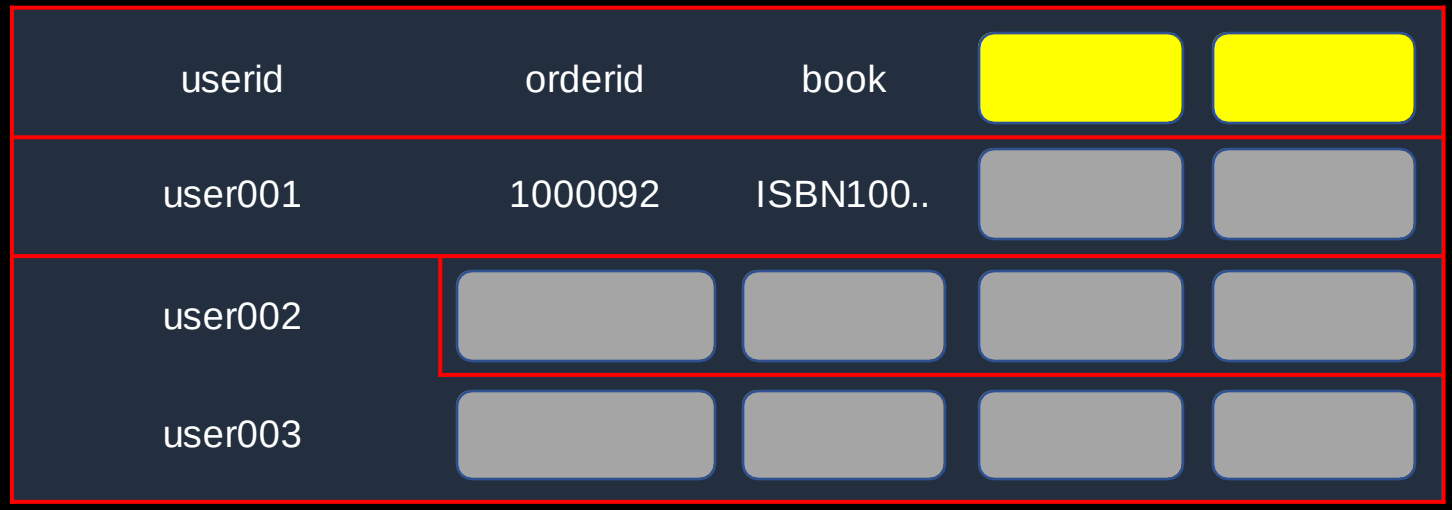


DynamoDB Table

© Digital Cloud Training | https://digitalcloud.training

Amazon DynamoDB Core Knowledge

- DynamoDB is made up of:
  - Tables
  - Items
  - Attributes



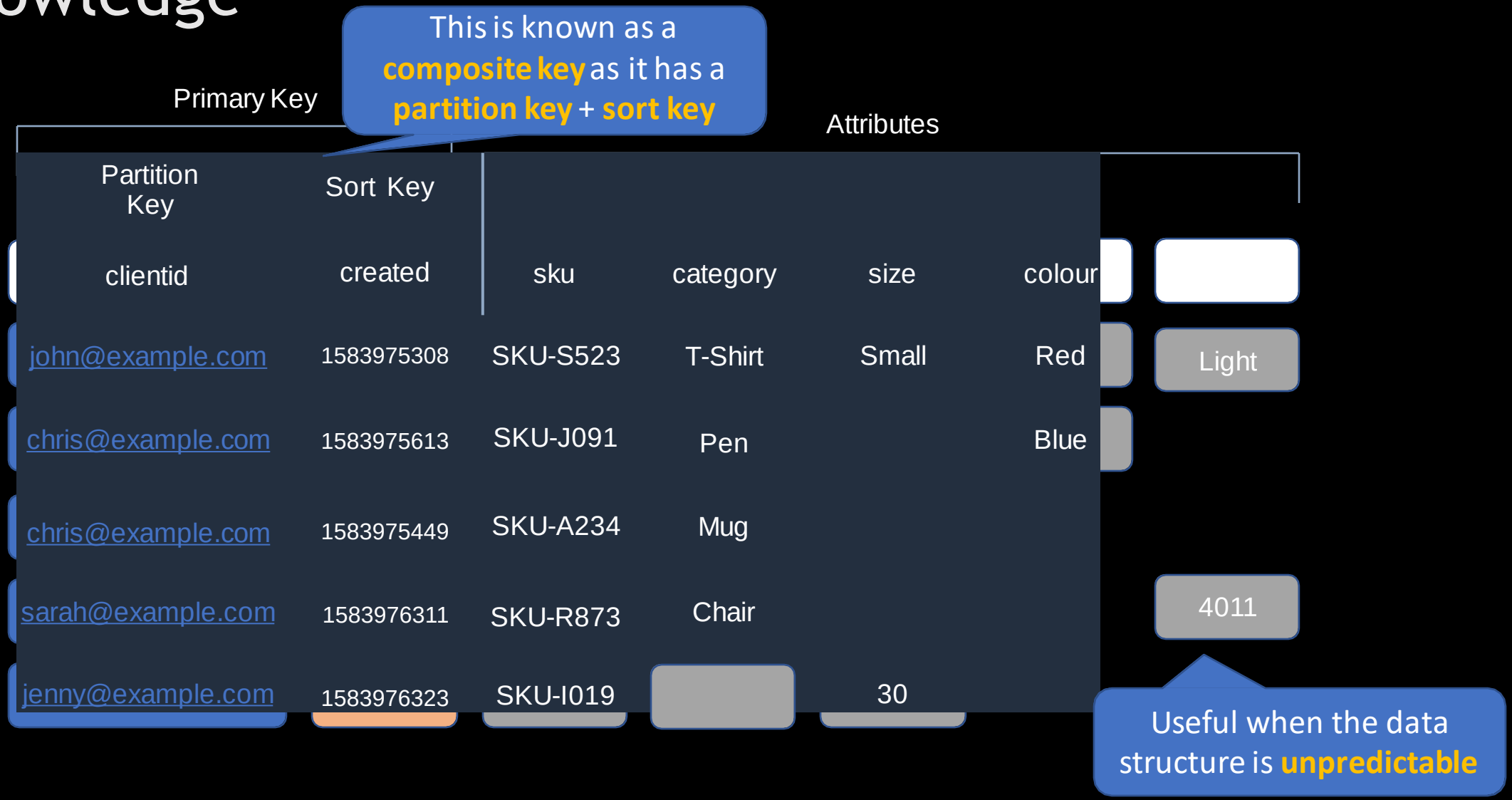
© Digital Cloud Training | https://digitalcloud.training

DynamoDB Time to Live (TTL)

- TTL lets you define when items in a table expire so that they can be automatically deleted from the database
- With TTL enabled on a table, you can set a timestamp for deletion on a per-item basis
- No extra cost and does not use WCU / RCU
- Helps reduce storage and manage the table size over time

© Digital Cloud Training | https://digitalcloud.training

Amazon DynamoDB Core Knowledge

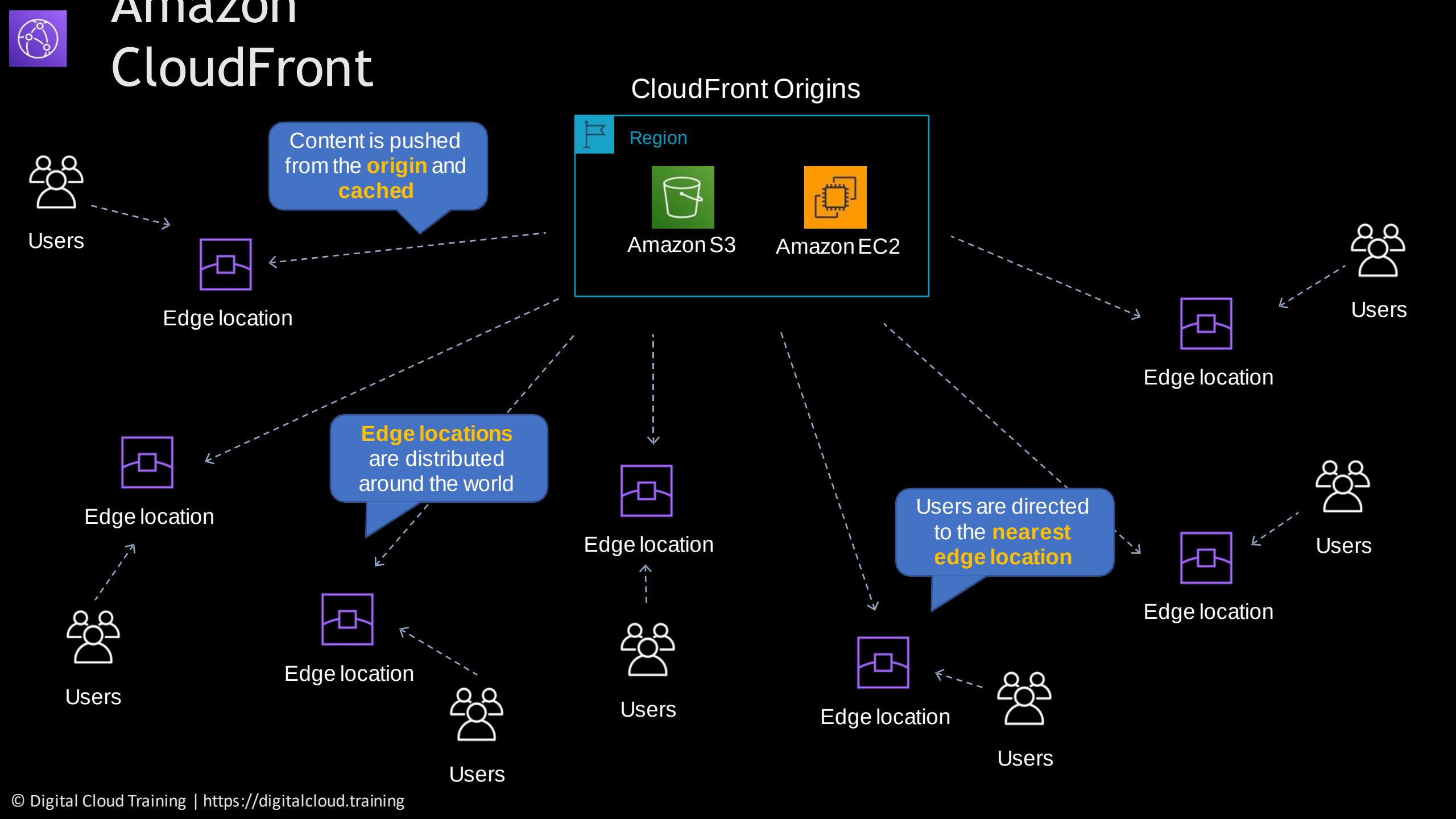


© Digital Cloud Training | https://digitalcloud.training

Amazon CloudFront Origins and Distributions

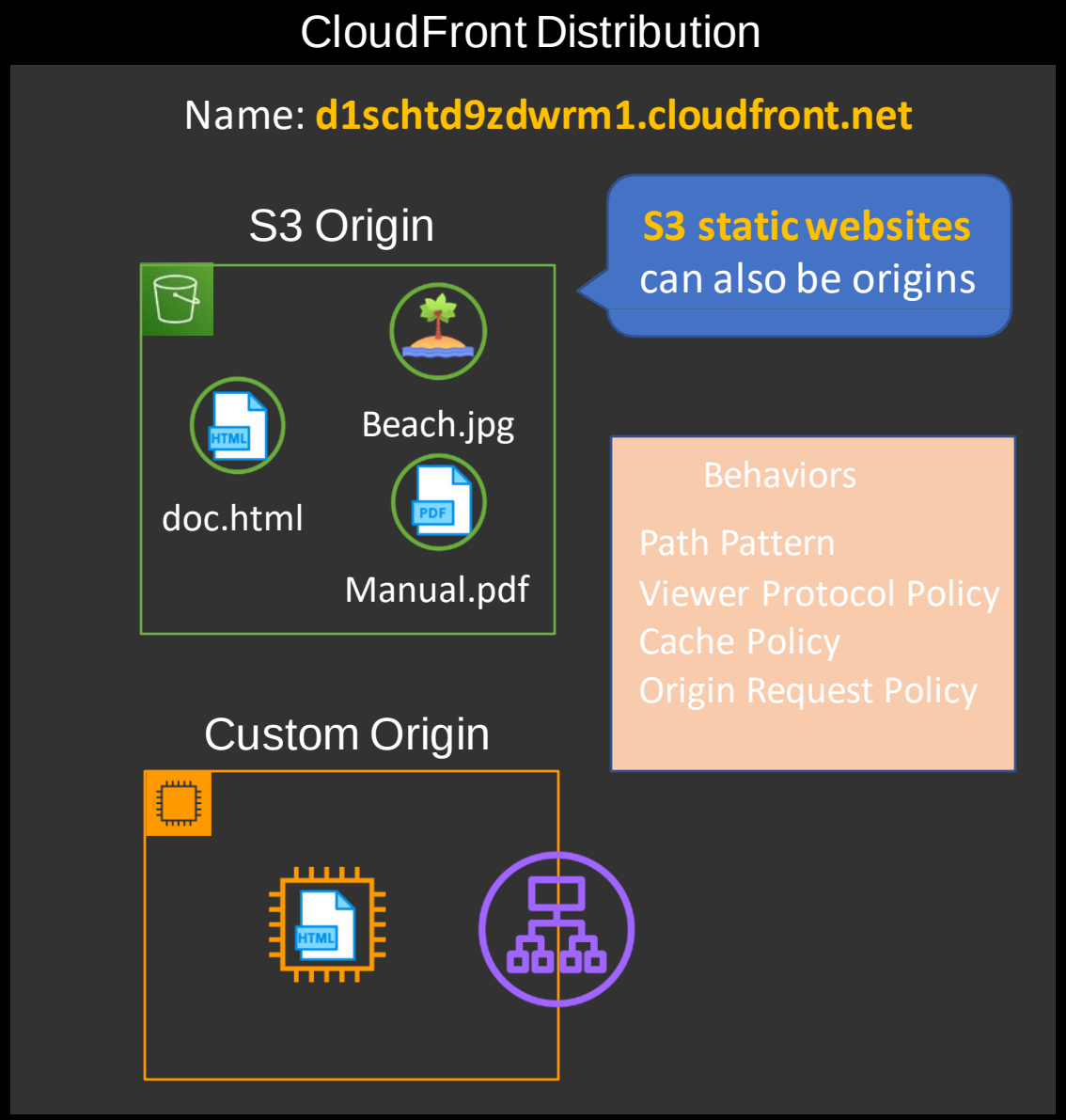
© Digital Cloud Training | https://digitalcloud.training

Amazon CloudFront



© Digital Cloud Training | https://digitalcloud.training

CloudFront Origins and Distributions



RTMP distributions were discontinued so only web distributions are currently available

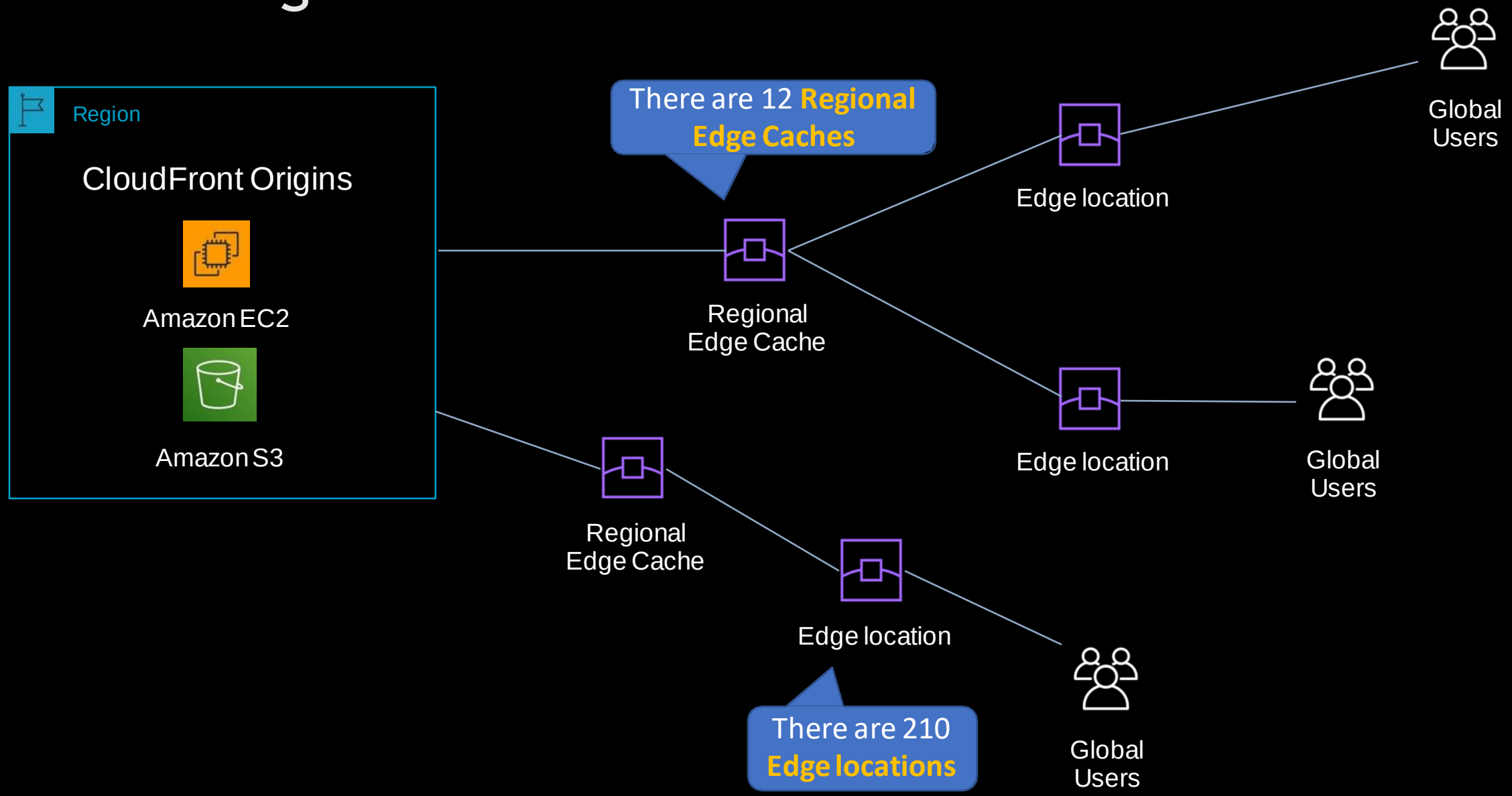
CloudFront Web Distribution:

- Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.
- Distribute media files using HTTP or HTTPS.
- Add, update, or delete objects, and submit data from web forms.
- Use live streaming to stream an event in real time.

© Digital Cloud Training | https://digitalcloud.training

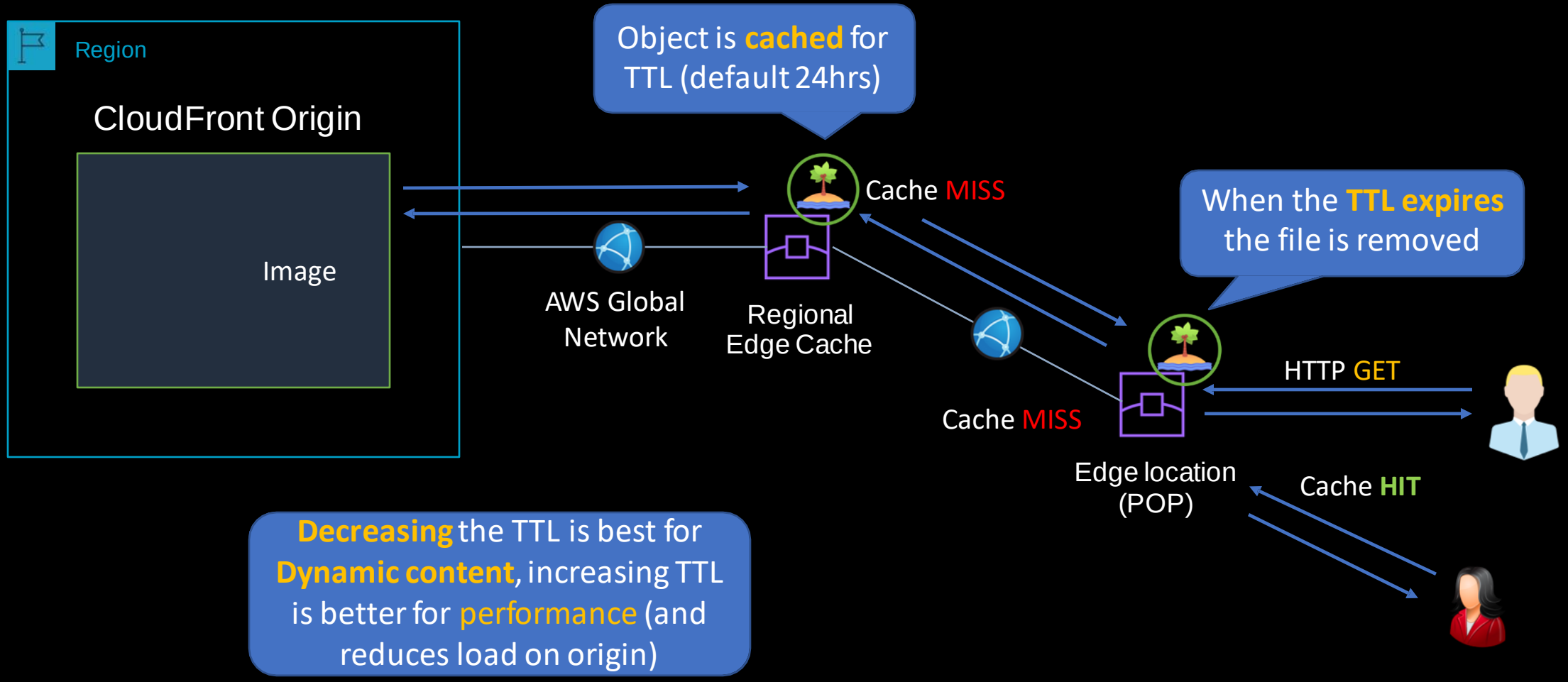
Amazon CloudFront Caching and Behaviors

Amazon CloudFront Caching



© Digital Cloud Training | https://digitalcloud.training

Amazon CloudFront Caching



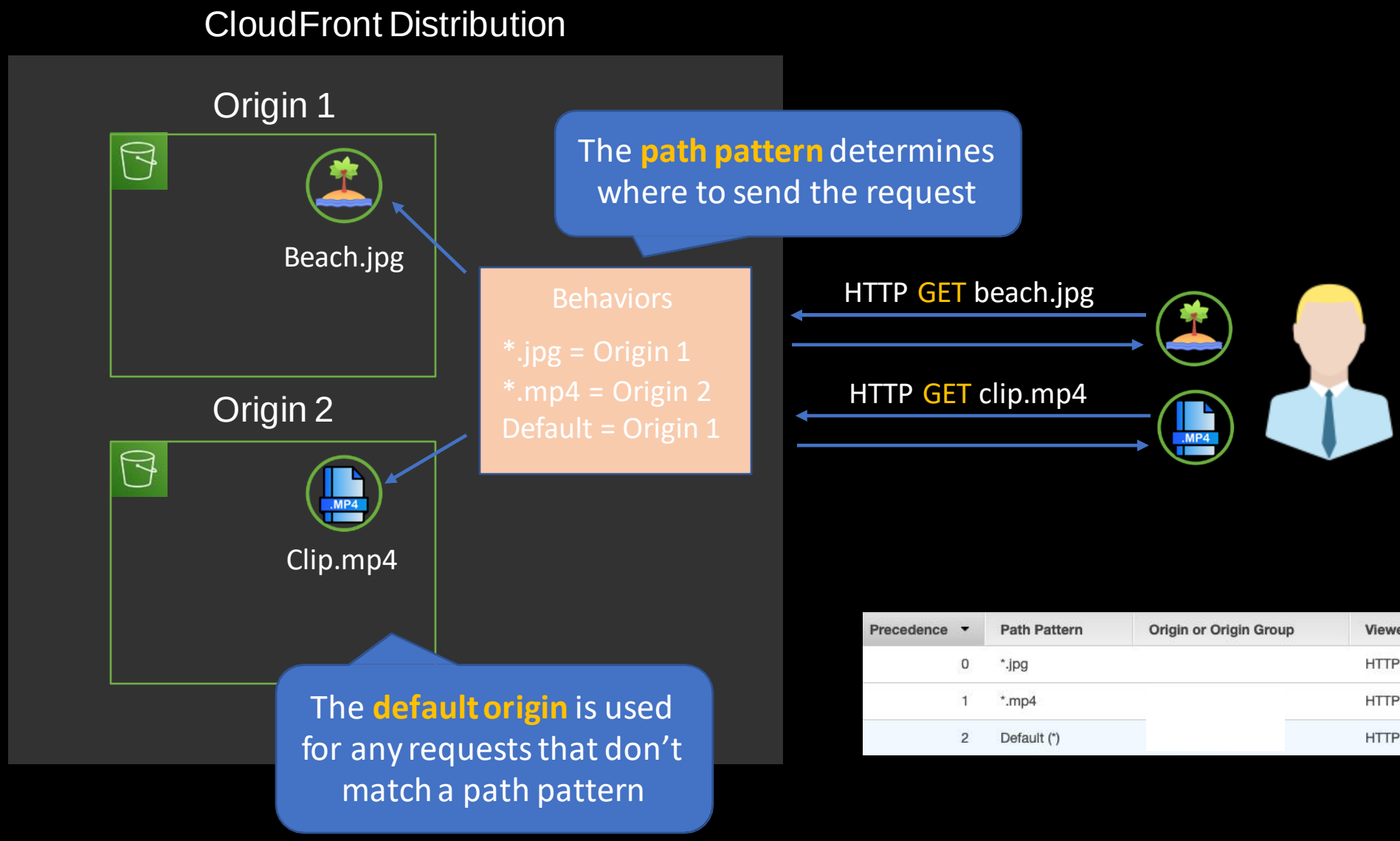
© Digital Cloud Training | https://digitalcloud.training

Amazon CloudFront Caching

- You can define a maximum **Time To Live** (TTL) and a default TTL
- TTL is defined at the **behavior** level
- This can be used to define different TTLs for different file types (e.g. png vs jpg)
- After expiration, CloudFront checks the origin for any new requests (check the file is the latest version)
- Headers can be used to control the cache:
  - `Cache-Control max-age=(seconds)` - specify how long before CloudFront gets the object again from the origin server
  - `Expires` – specify an expiration date and time

© Digital Cloud Training | https://digitalcloud.training

CloudFront Path Patterns



Precedence	Path Pattern	Origin or Origin Group	Viewer Protocol Policy	Cache Policy Name
0	*.jpg		HTTP and HTTPS	Managed-CachingOptimized
1	*.mp4		HTTP and HTTPS	Managed-CachingOptimized
2	Default (*)		HTTP and HTTPS	Managed-CachingOptimized

© Digital Cloud Training | https://digitalcloud.training

Caching Based on Request Headers

- You can configure CloudFront to forward **headers** in the **viewer request** to the origin
- CloudFront can then cache multiple versions of an object based on the values in one or more request headers
- Controlled in a behavior to do one of the following:
  - Forward all headers to your origin (objects are not cached)
  - Forward a whitelist of headers that you specify
  - Forward only the default headers (doesn't cache objects based on values in request headers)

© Digital Cloud Training | https://digitalcloud.training