serverless

An Introduction to Microservices with the Serverless Framework

serverless

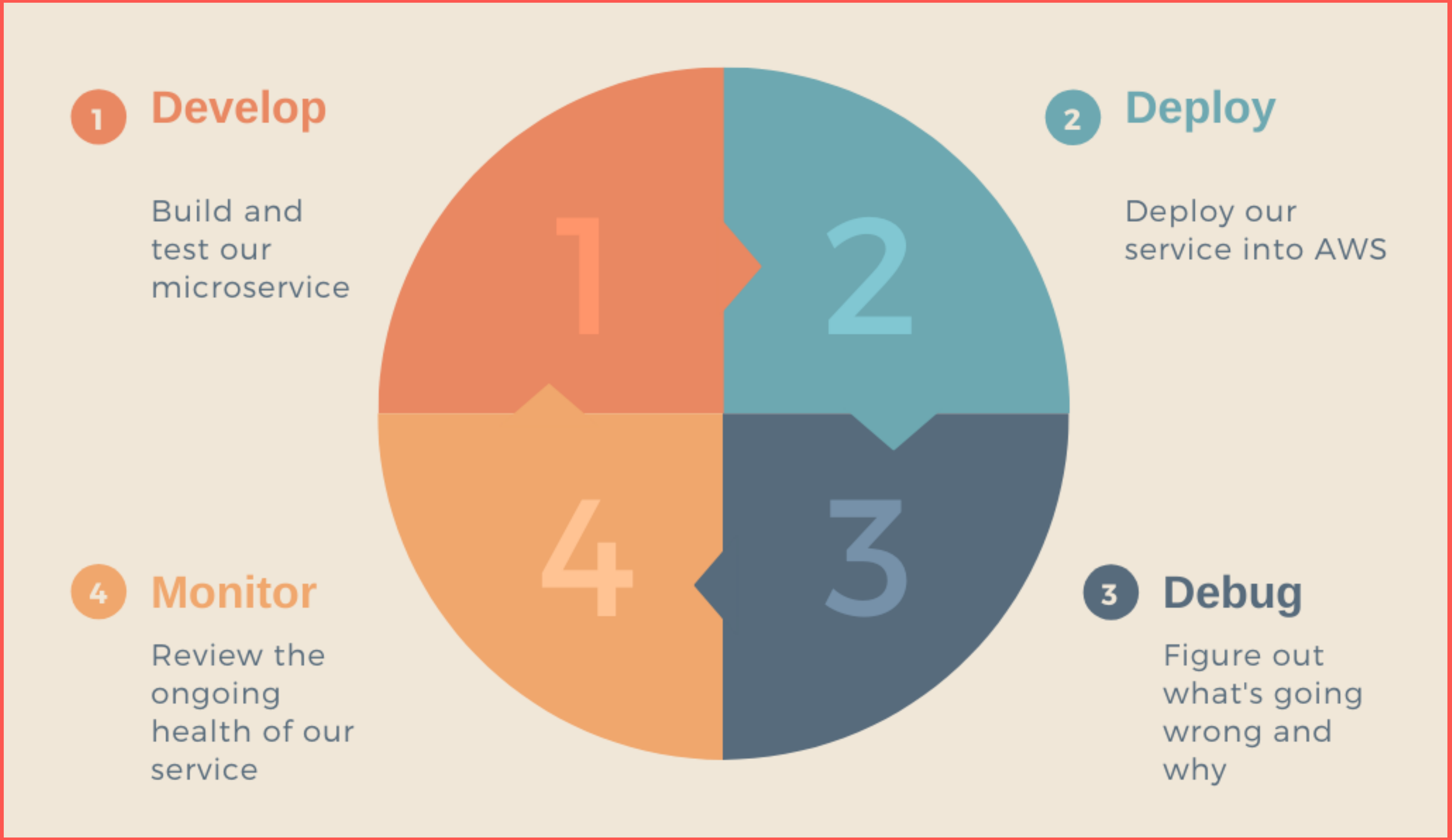
Fernando Medina Corey

Solutions Architect
Serverless Inc.

@fmc_sea

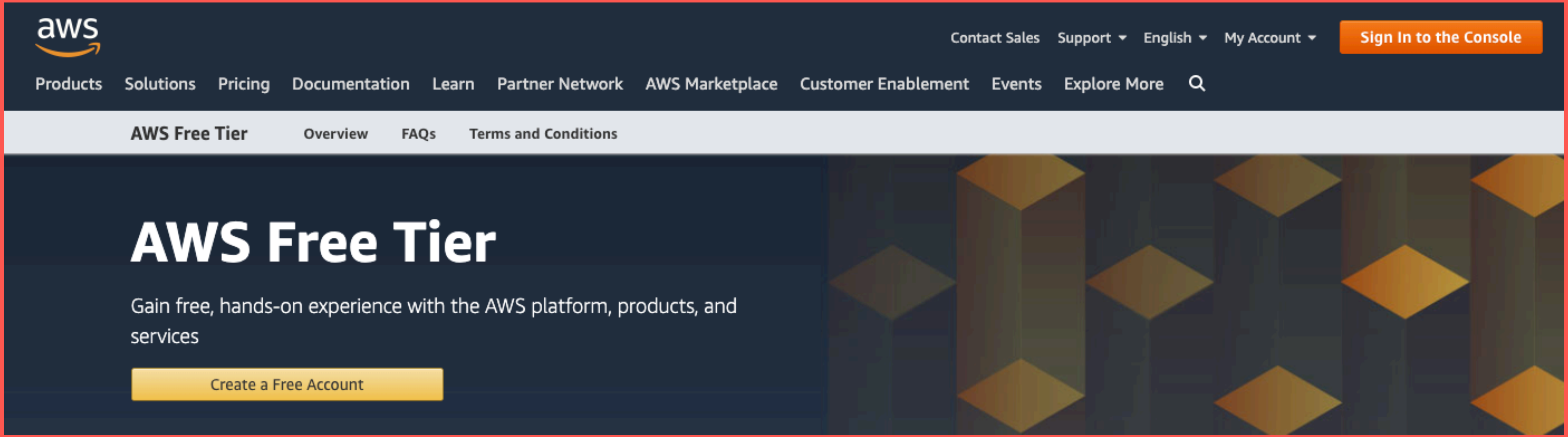
What are we doing?

Our Goals


1 Develop
Build and test our microservice
2 Deploy
Deploy our service into AWS
3 Debug
Figure out what's going wrong and why
4 Monitor
Review the ongoing health of our service
1
2
3
4

Let’s get our environment ready

Create an AWS Account

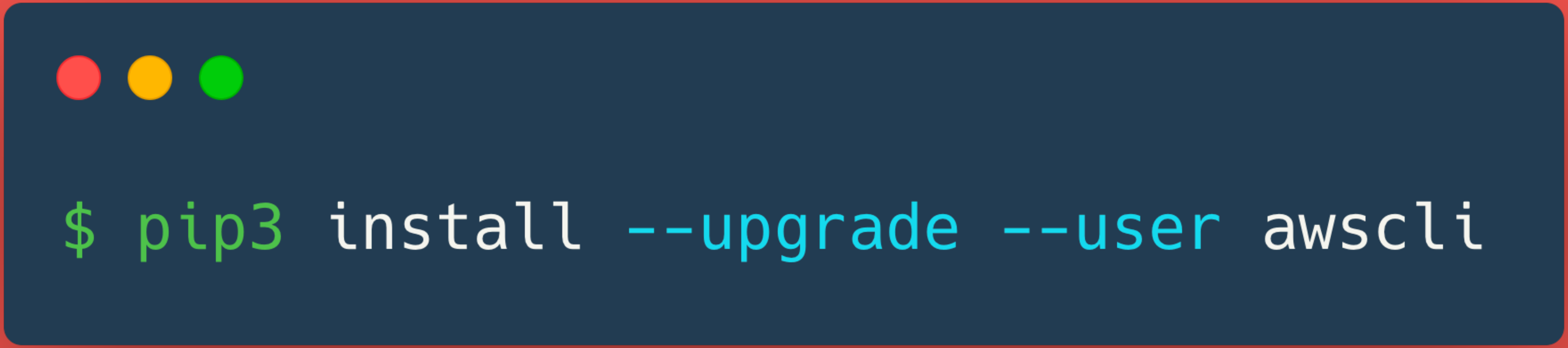


AWS Free Tier

Gain free, hands-on experience with the AWS platform, products, and services

https://aws.amazon.com/free

Install the AWS CLI

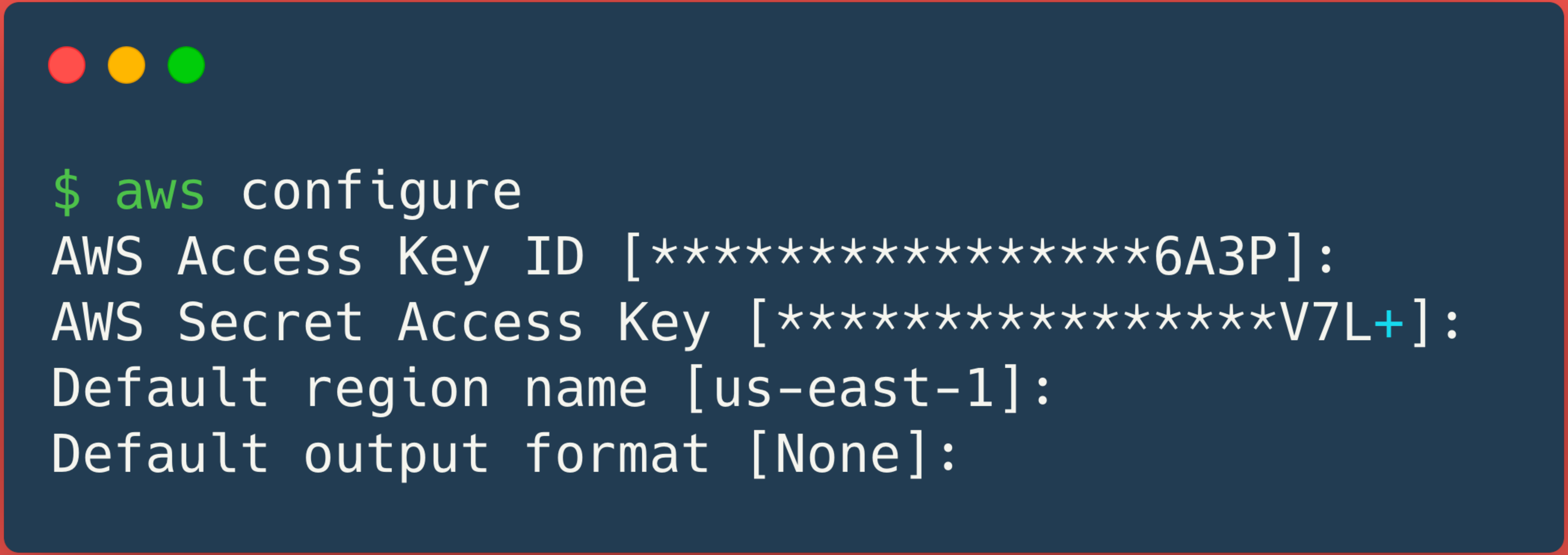


```
$ pip3 install --upgrade --user awscli
```

(This assumes you have Python3 installed)

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

Configure the AWS CLI



```
$ aws configure
AWS Access Key ID [****************6A3P]:
AWS Secret Access Key [****************V7L+]:
Default region name [us-east-1]:
Default output format [None]:
```

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html

Install the Serverless Framework

Mac/Linux

```
$ curl -o- -L https://slss.io/install | bash
```

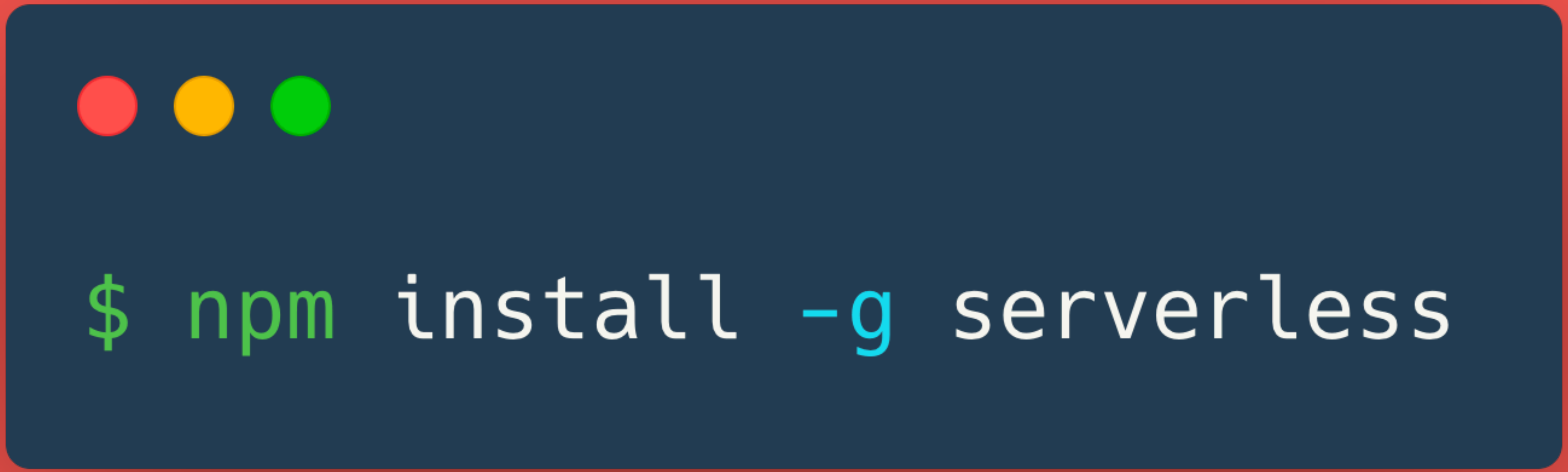
Windows

```
> choco install serverless
```

https://serverless.com/framework/docs/getting-started/

Install the Serverless Framework

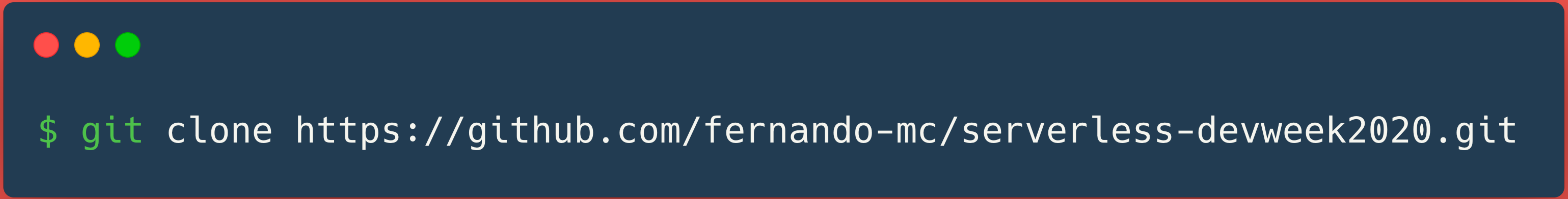
(With npm)



```
$ npm install -g serverless
```

https://serverless.com/framework/docs/getting-started/

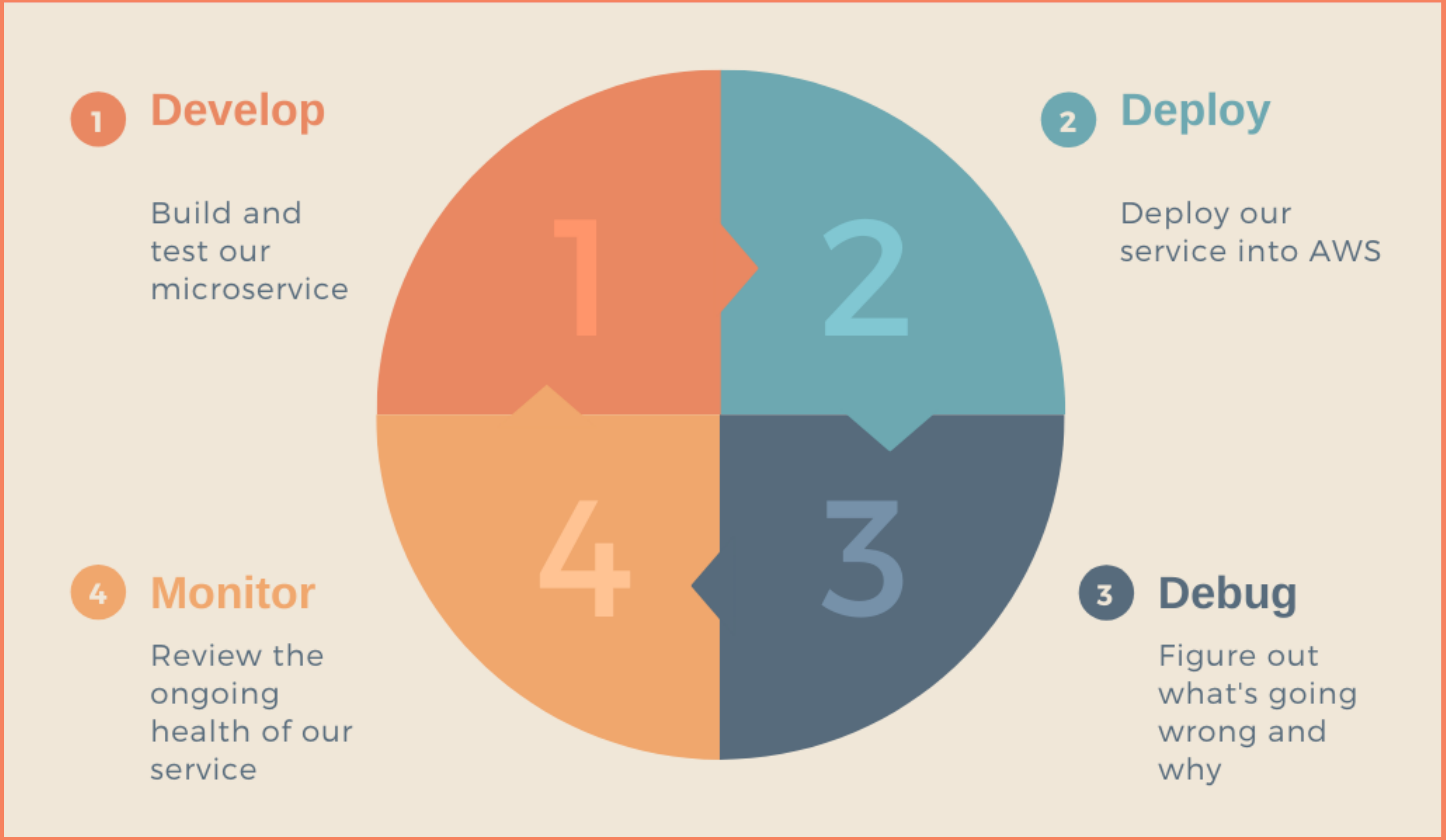
Clone the Project



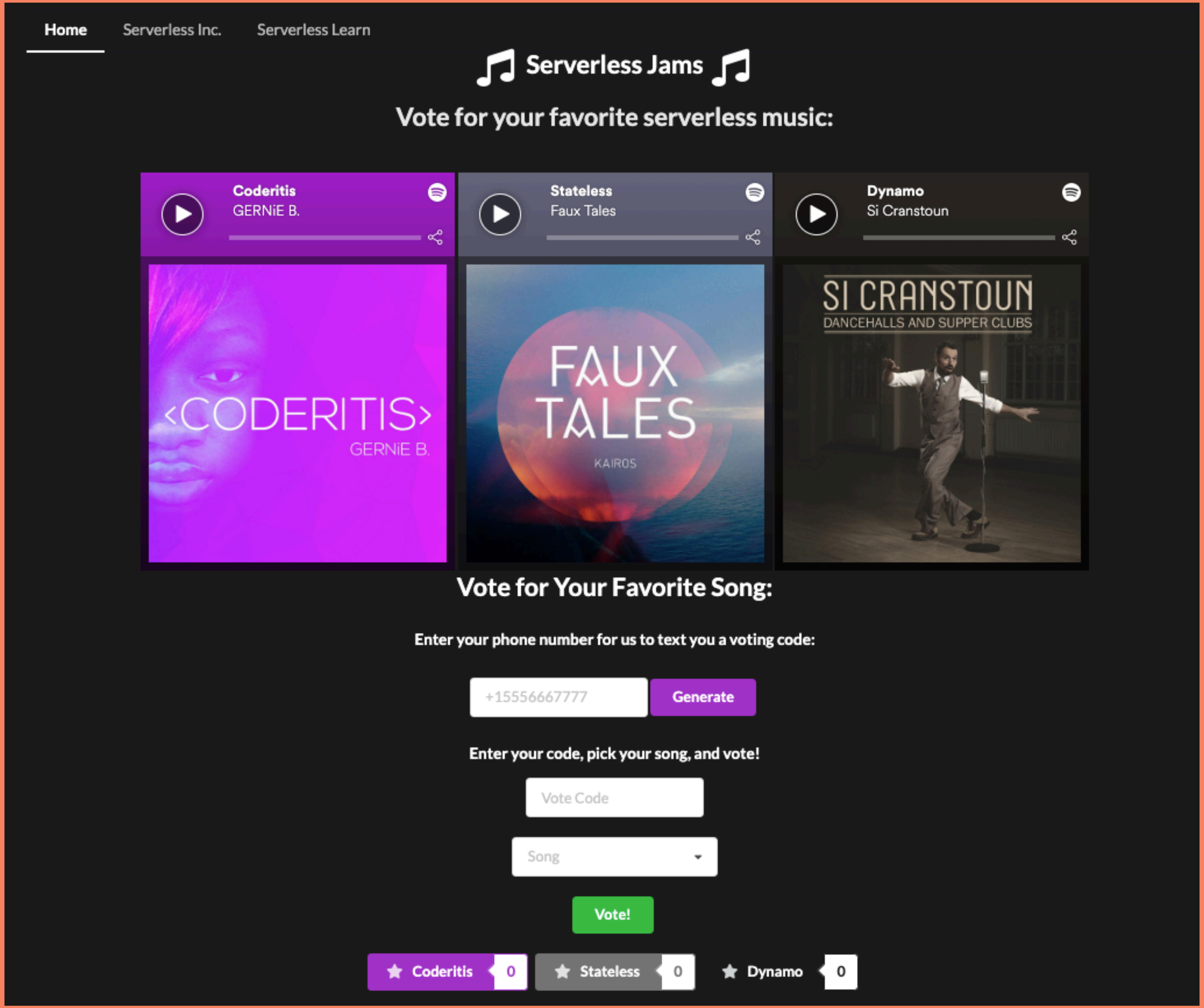
```
$ git clone https://github.com/fernando-mc/serverless-devweek2020.git
```

https://github.com/fernando-mc/serverless-devweek2020

Develop


1 Develop
Build and test our microservice
2 Deploy
Deploy our service into AWS
3 Debug
Figure out what's going wrong and why
4 Monitor
Review the ongoing health of our service
1
2
3
4

What are we developing?



Home Serverless Inc. Serverless Learn

Serverless Jams

Vote for your favorite serverless music:

Vote for Your Favorite Song:

Enter your phone number for us to text you a voting code:

+15556667777 Generate

Enter your code, pick your song, and vote!

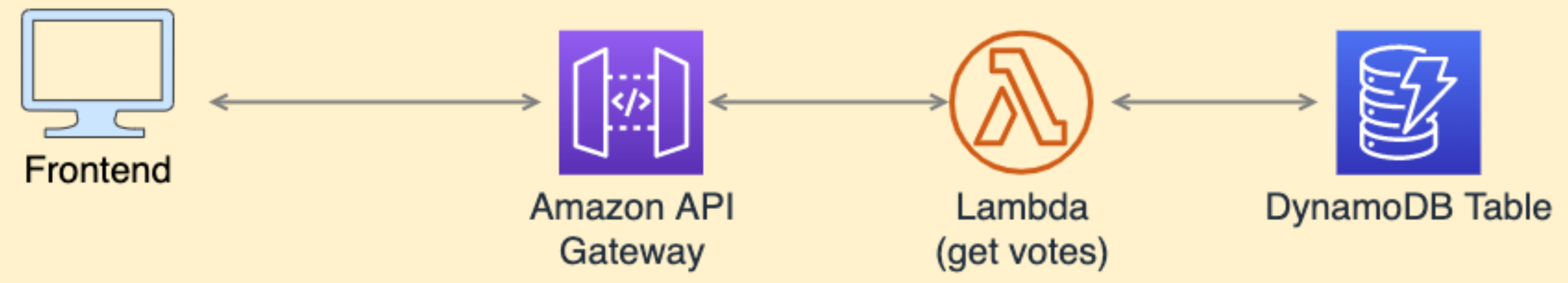
Vote Code

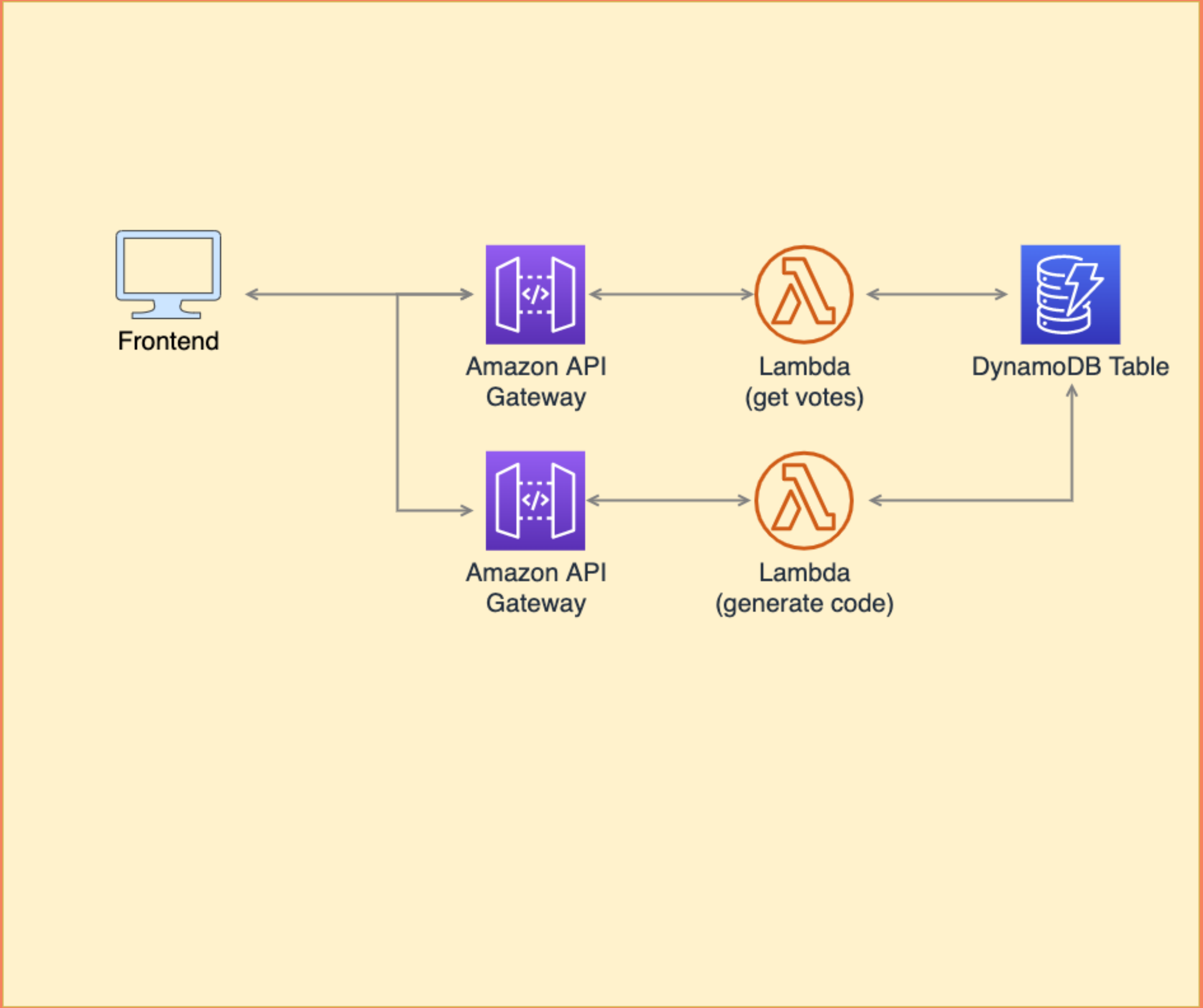
Song

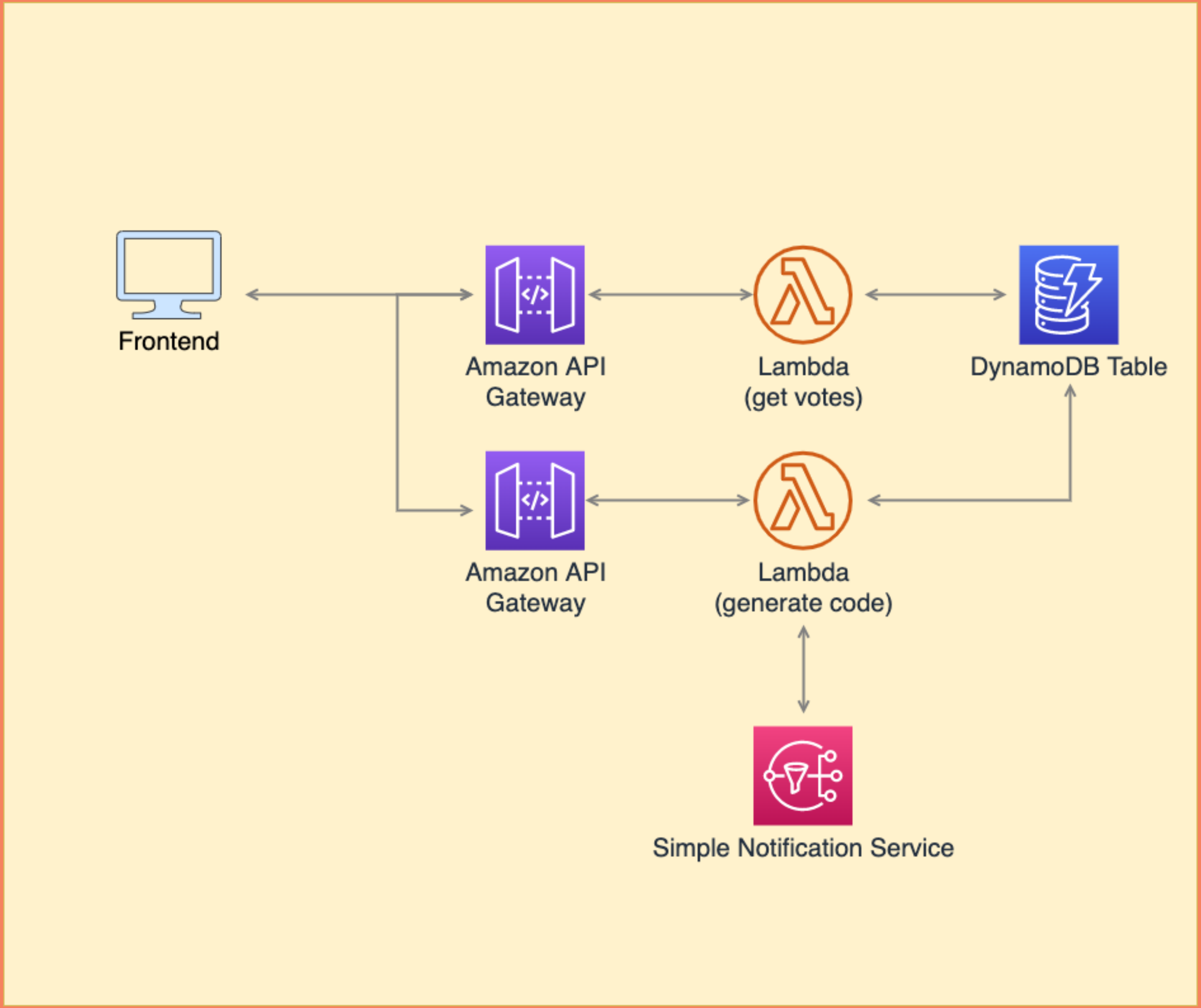
Vote!

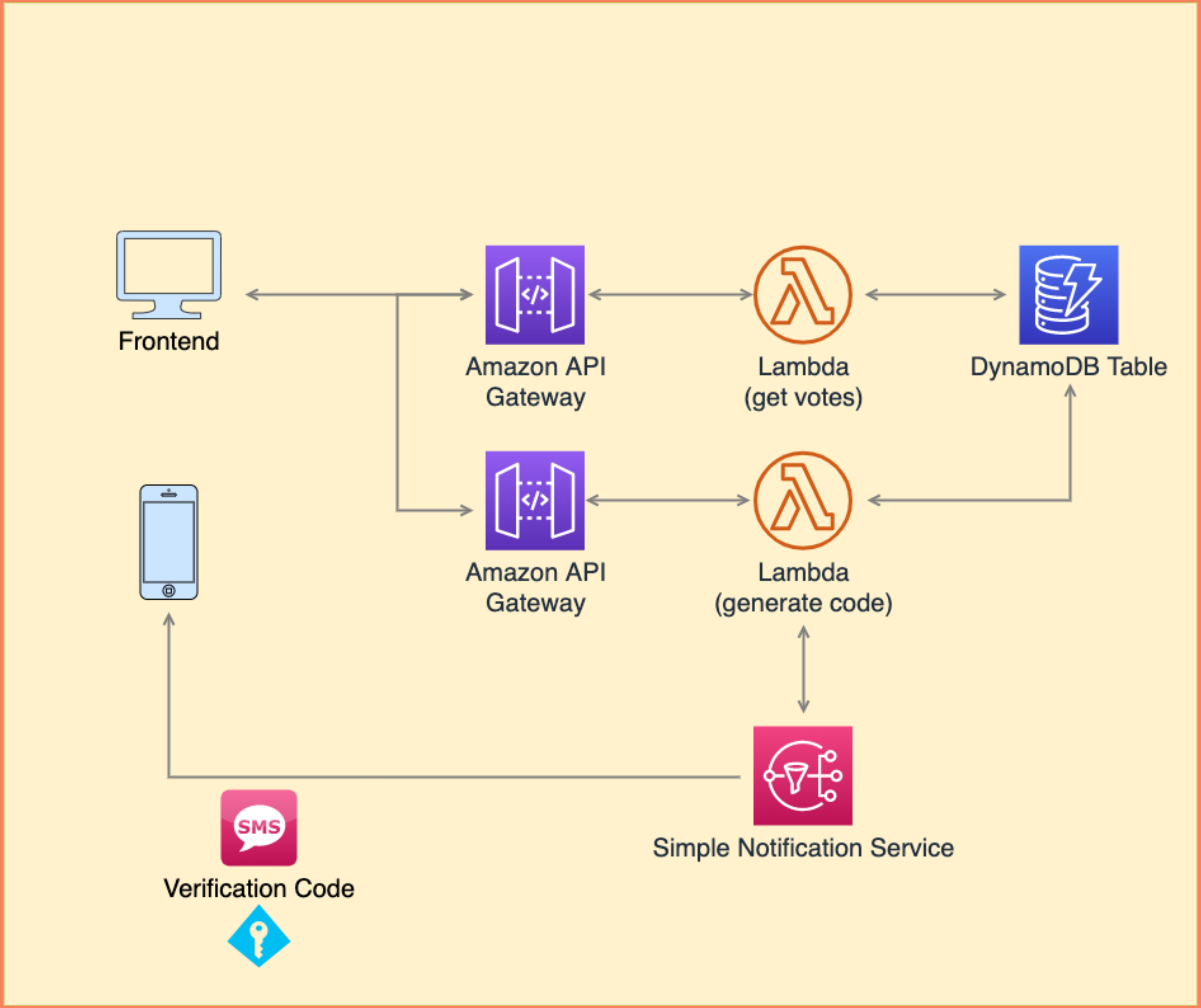
Coderitis	0
Stateless	0
Dynamo	0

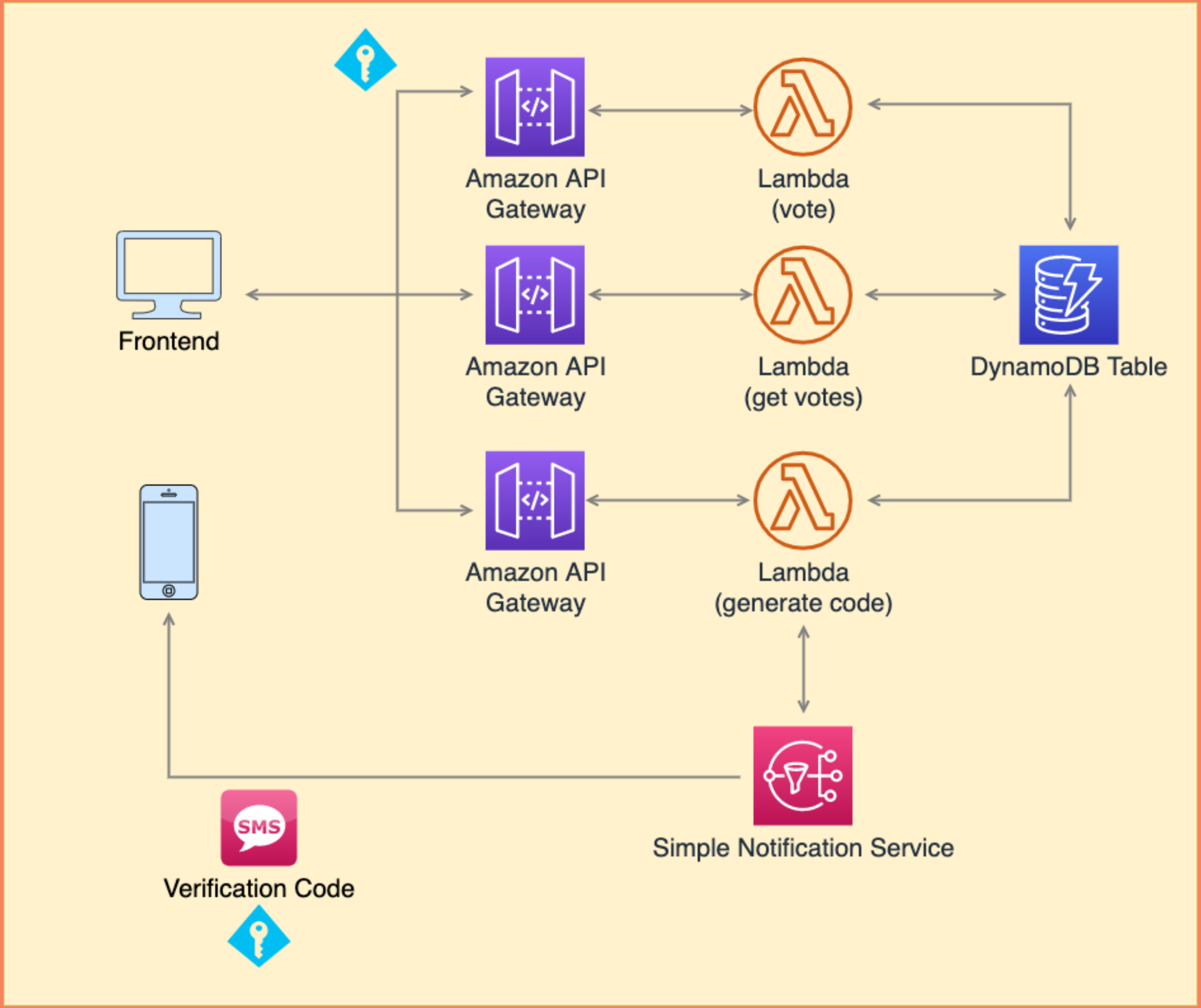
How does it work?


Frontend
Amazon API Gateway
Lambda (get votes)
DynamoDB Table


Frontend
Amazon API Gateway
Lambda (get votes)
DynamoDB Table
Amazon API Gateway
Lambda (generate code)

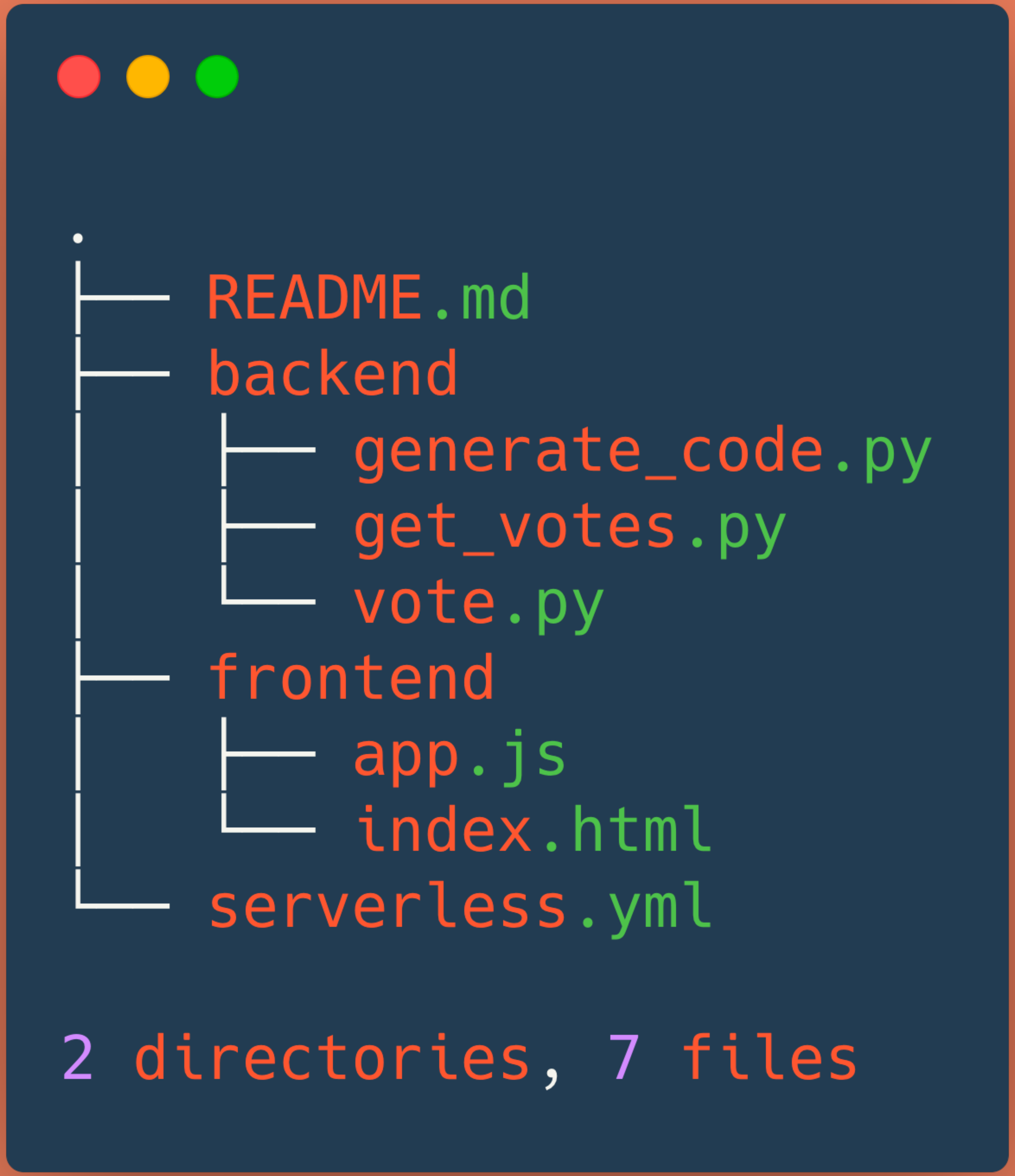

Frontend
Amazon API Gateway
Lambda (get votes)
DynamoDB Table
Amazon API Gateway
Lambda (generate code)
Simple Notification Service


Frontend
Amazon API Gateway
Lambda (get votes)
DynamoDB Table
Amazon API Gateway
Lambda (generate code)
Simple Notification Service
SMS
Verification Code


Amazon API Gateway
Lambda (vote)
Frontend
Amazon API Gateway
Lambda (get votes)
DynamoDB Table
Amazon API Gateway
Lambda (generate code)
Simple Notification Service
SMS
Verification Code

What’s the code look like?

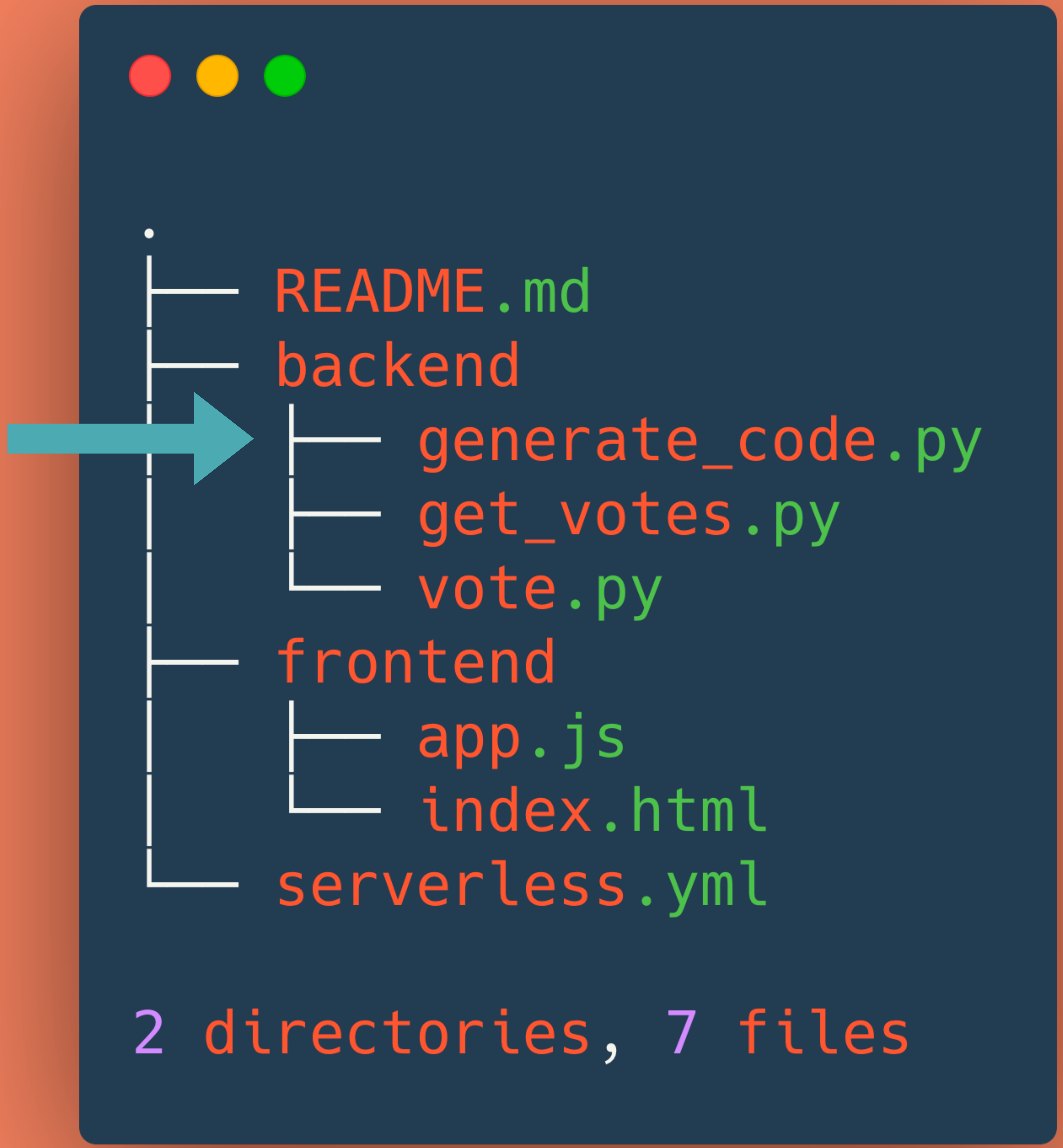
What's the code look like?



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

generate_code.py

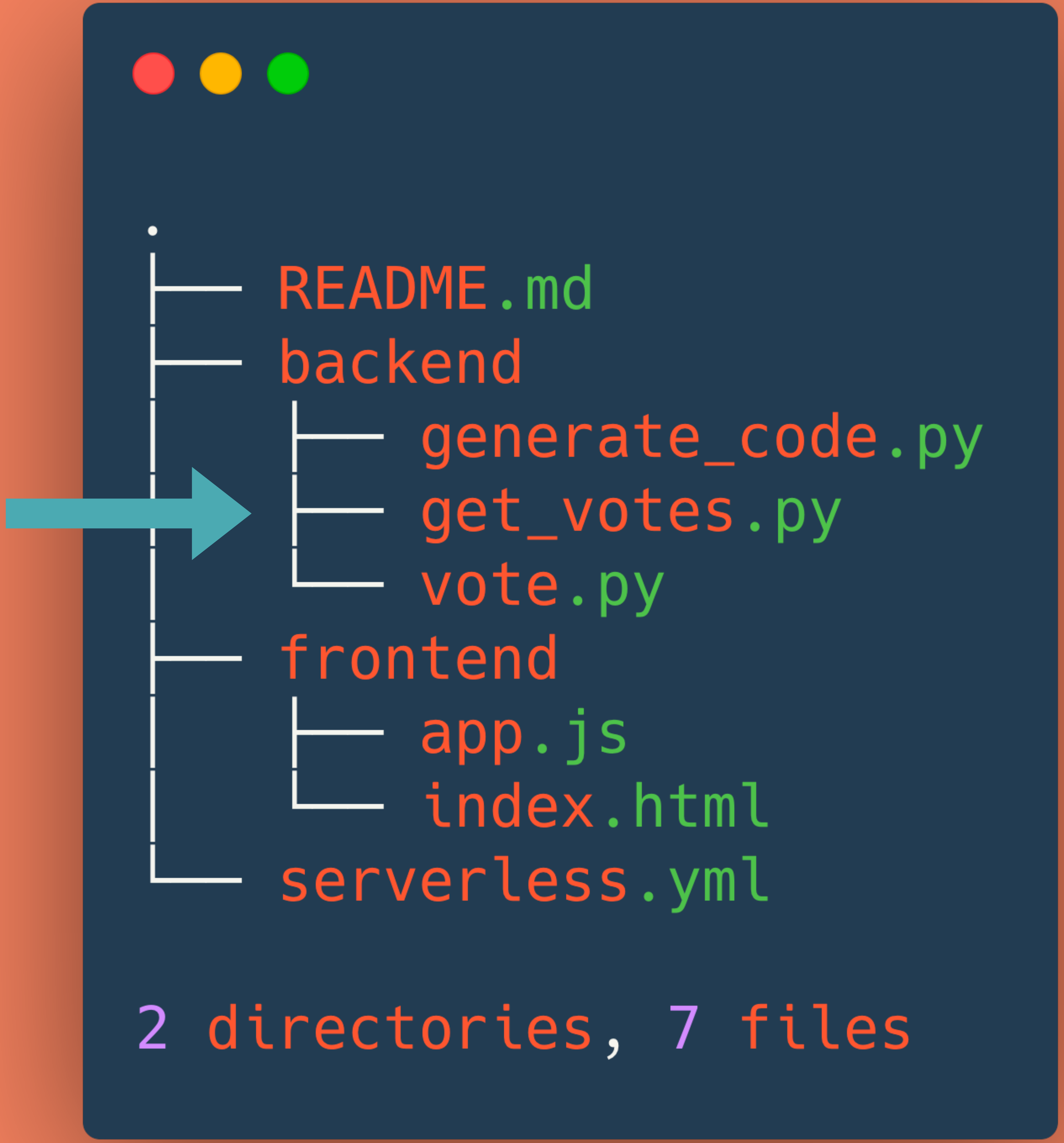
1. Phone number as input
2. (re)generates verification code
3. Sends the code to the number



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

get_votes.py

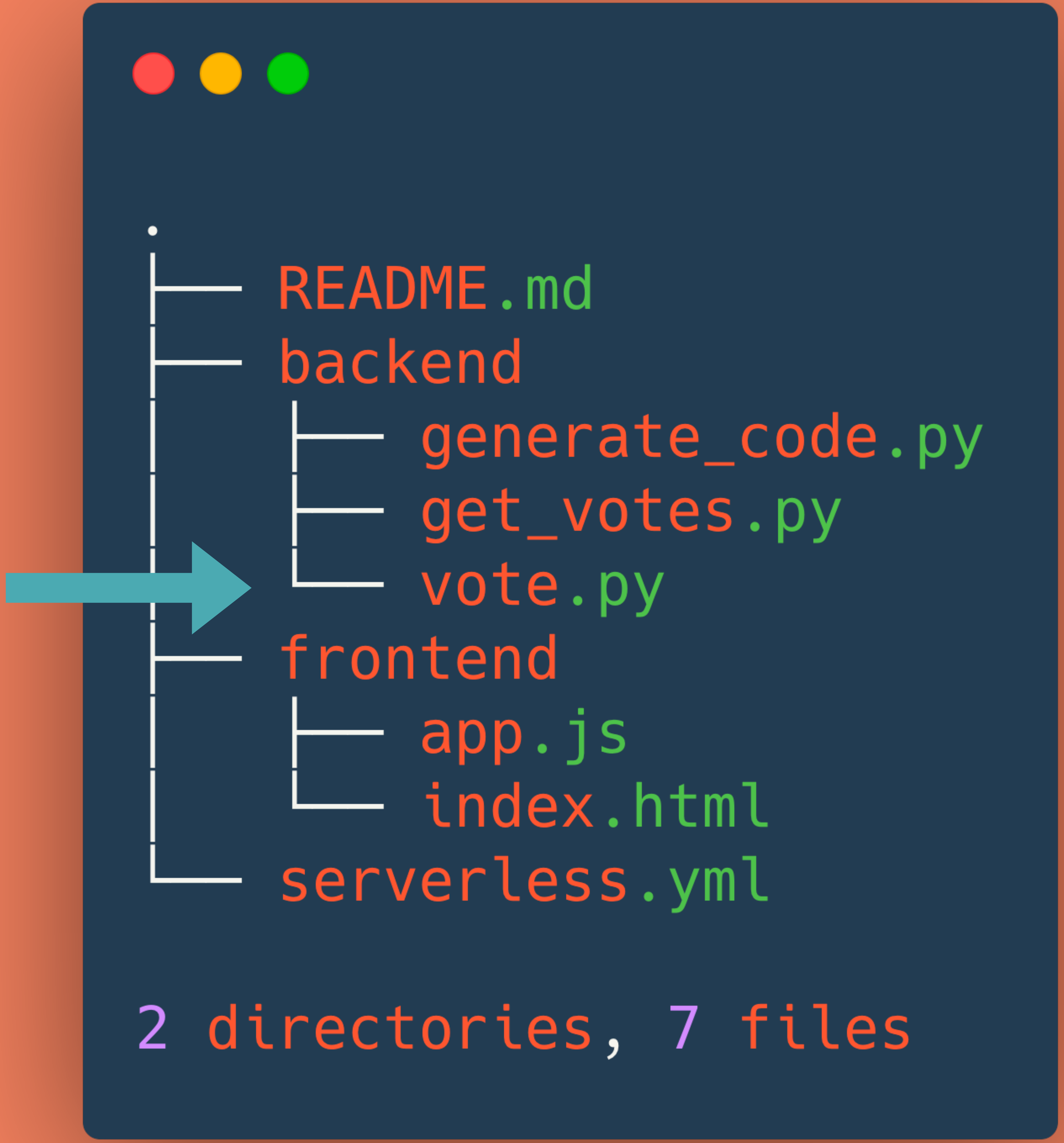
1. Gets vote counts for all songs
2. Returns them to the frontend



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

vote.py

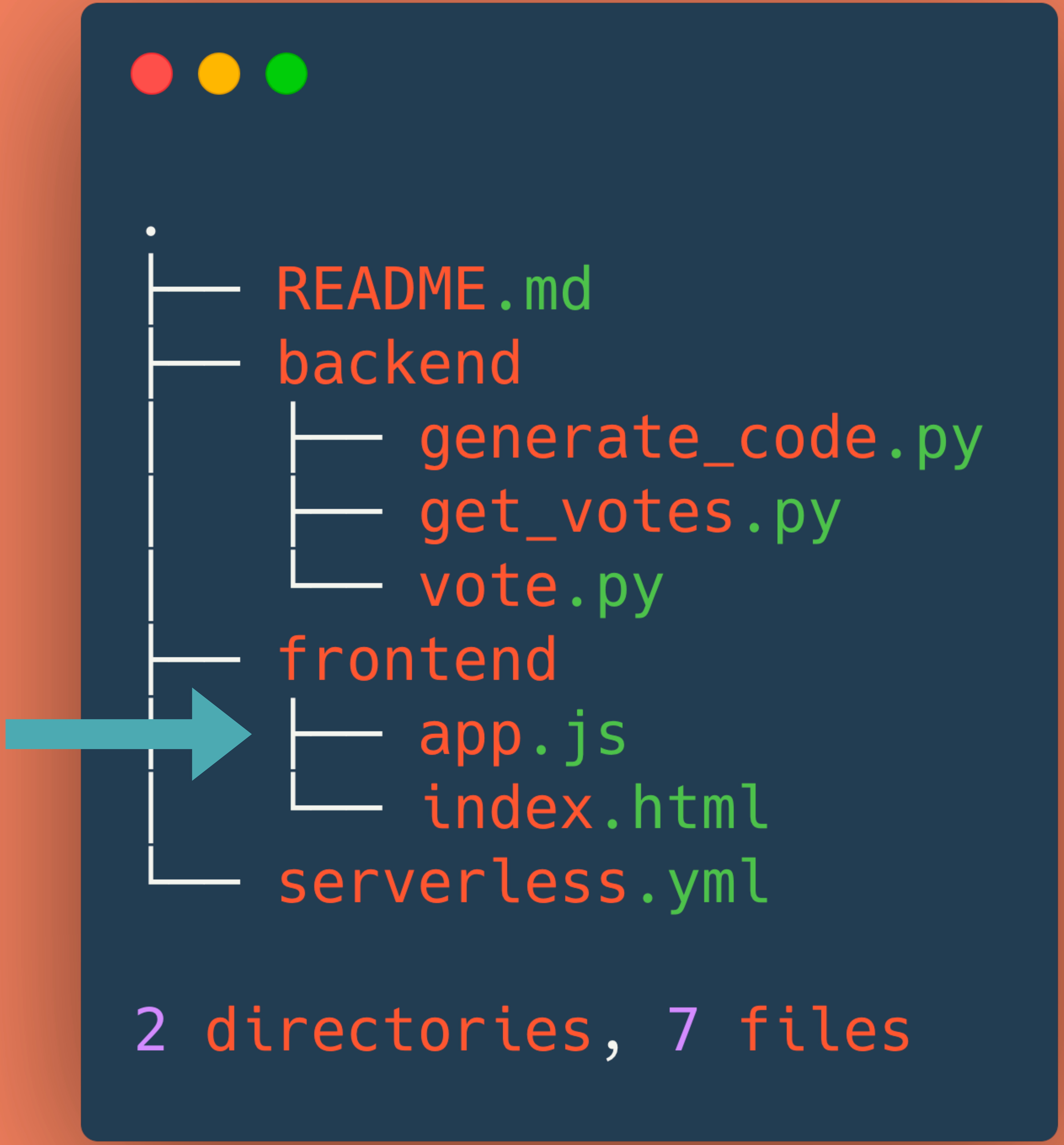
1. Verified phone number/code
2. Sets the code as used
3. Increments a vote counter
4. Returns the new vote count



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

app.js

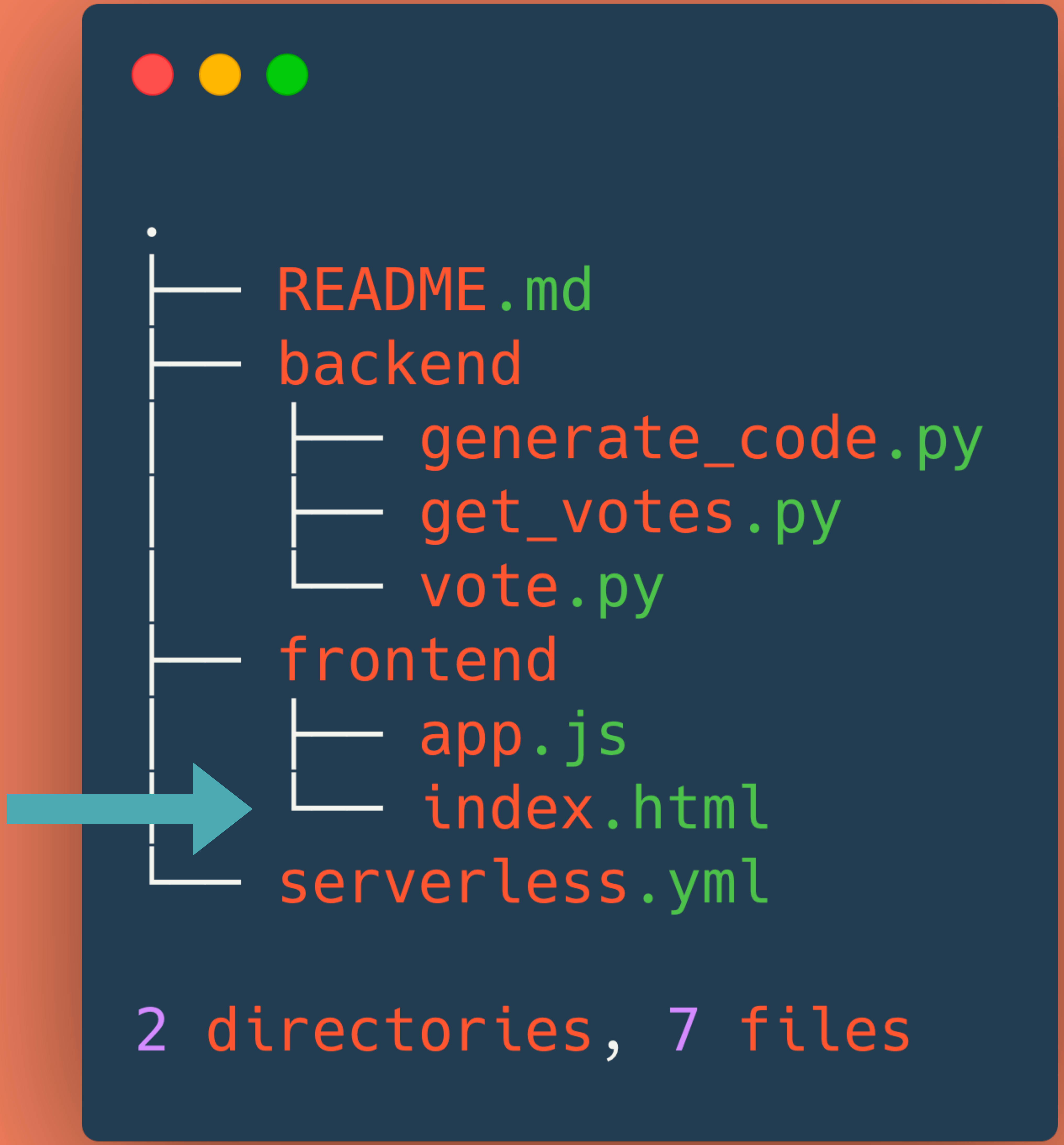
1. Handles interacting with the API
2. Updates vote counts in the UI
3. Provides helper text for the user



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

index.html

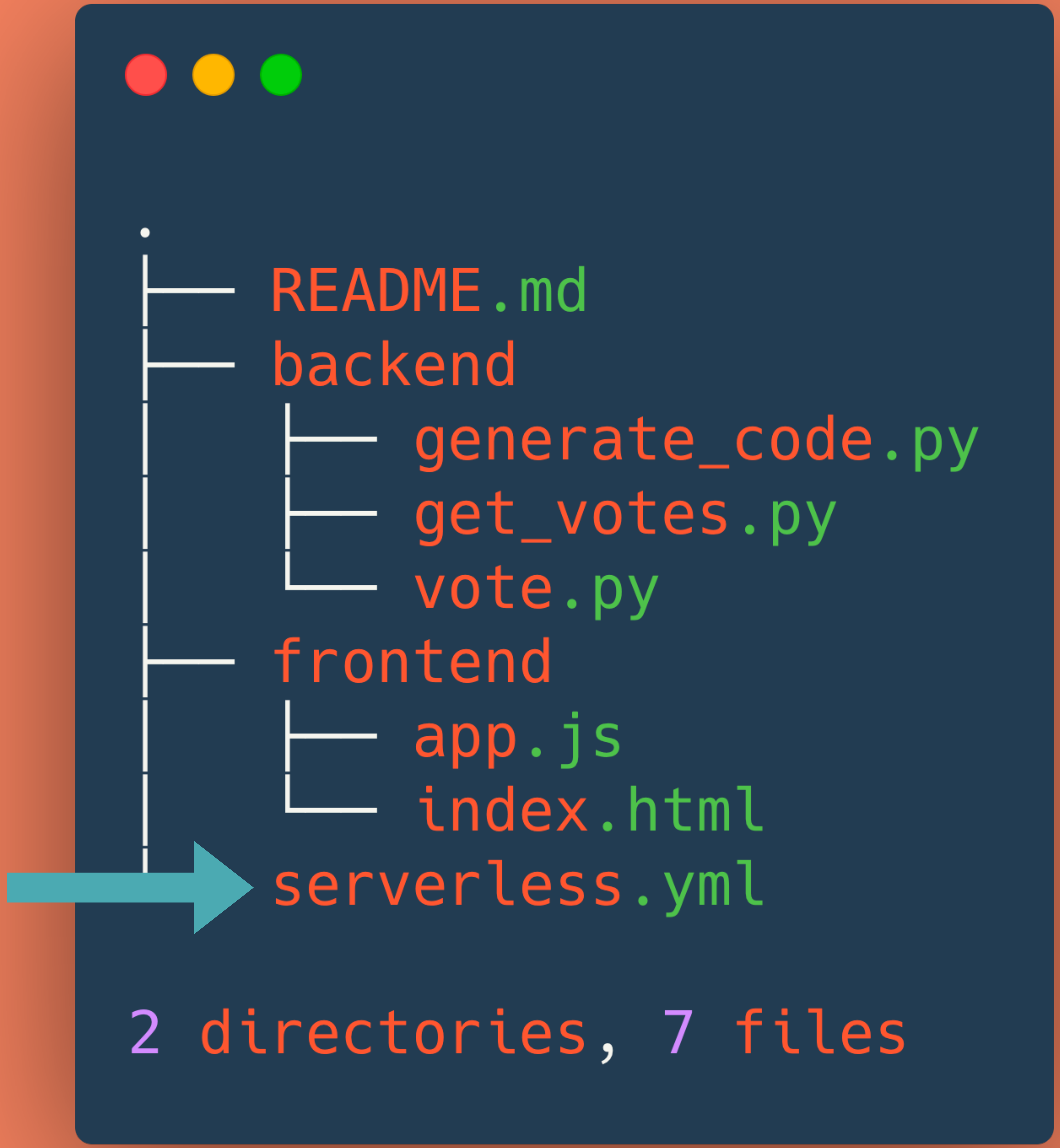
1. Snazzy Spotify embeds!
2. Pretty Semantic UI buttons!
3. Cool vote counters!



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

serverless.yml

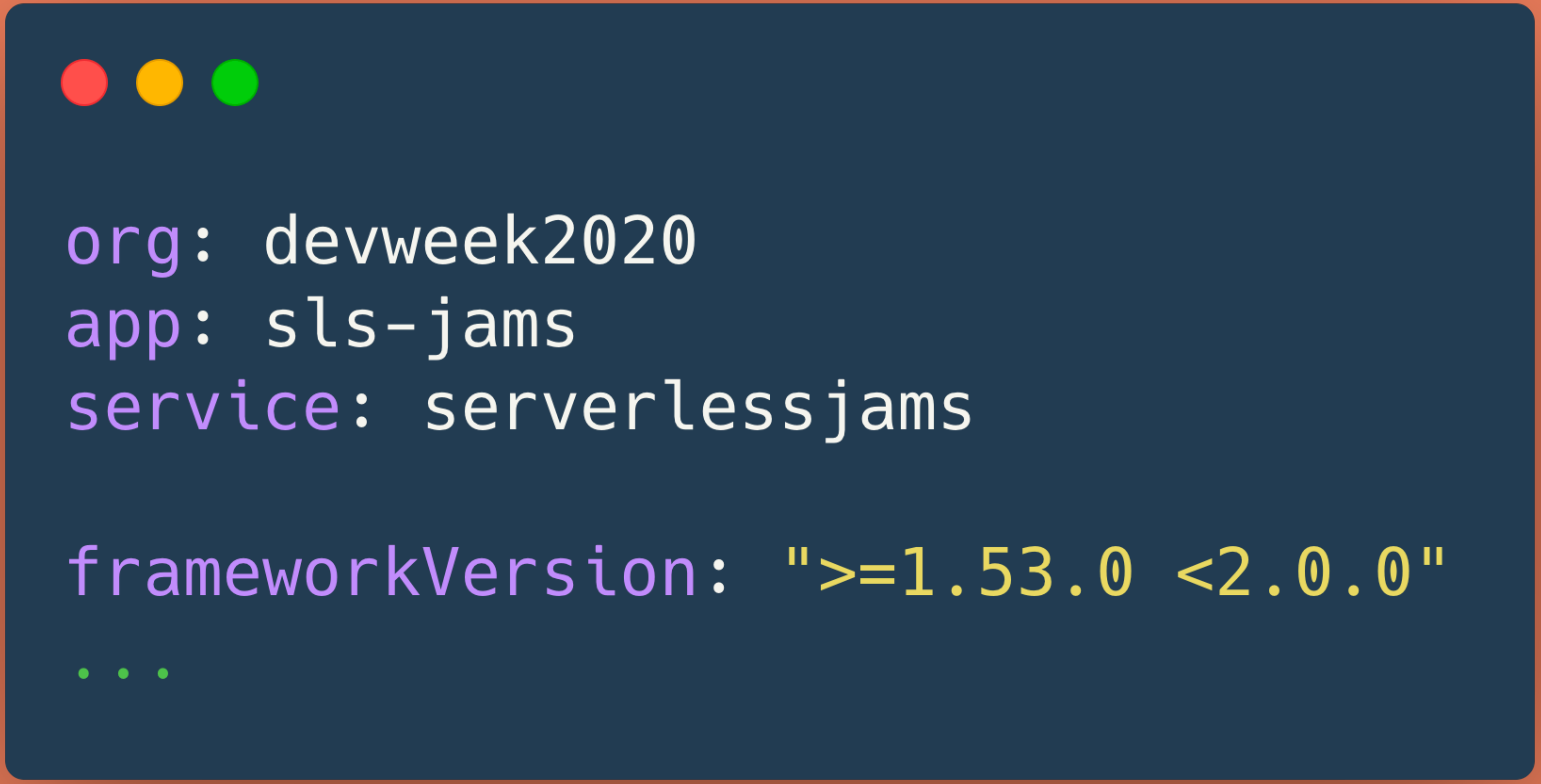
1. Creates Lambda Functions
2. Creates API Gateway Endpoints
3. Creates our DynamoDB table
4. Sets the service’s permissions



```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

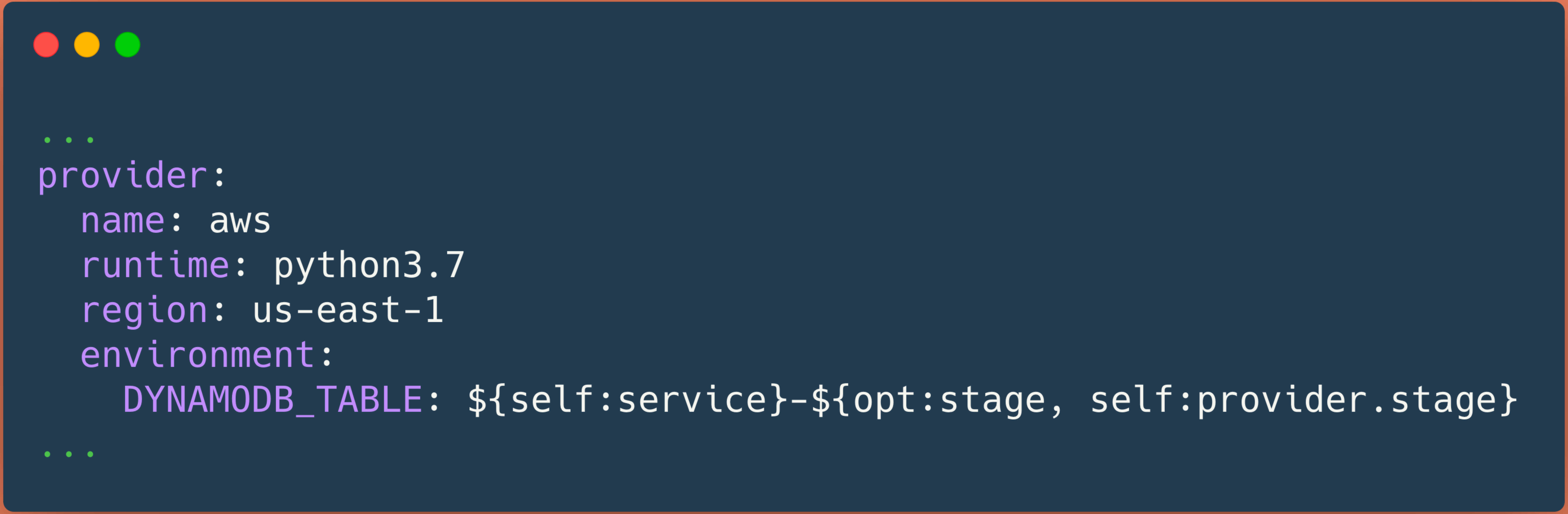
What’s in serverless.yml?

Service Configuration



```
org: devweek2020
app: sls-jams
service: serverlessjams

frameworkVersion: ">=1.53.0 <2.0.0"
...
```

Provider details

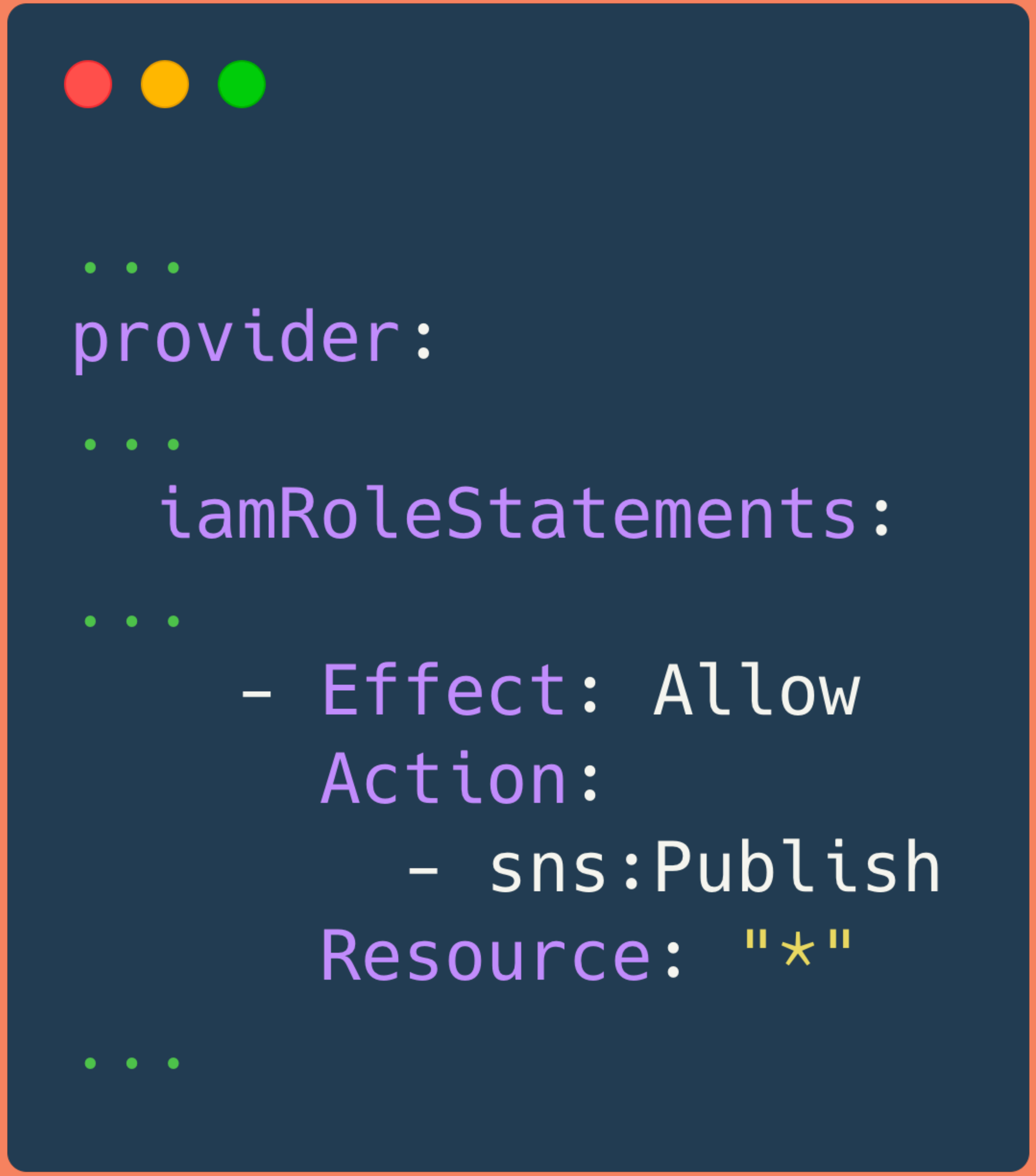


```
...
provider:
  name: aws
  runtime: python3.7
  region: us-east-1
  environment:
    DYNAMODB_TABLE: ${self:service}-${opt:stage, self:provider.stage}
...
```

Provider details - IAM Roles

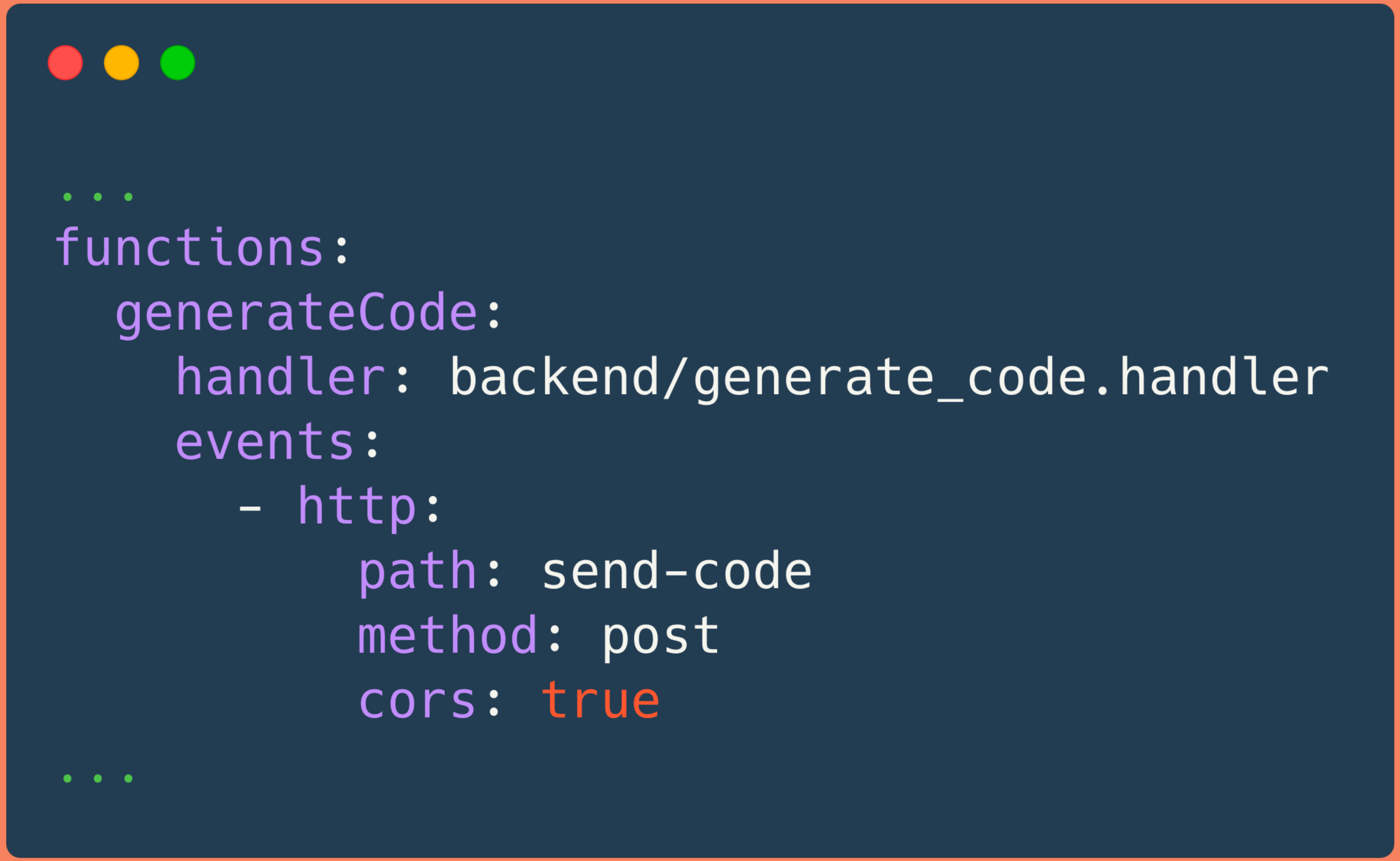
```
...
provider:
...
  iamRoleStatements:
    - Effect: Allow
      Action:
        - dynamodb:Query
        - dynamodb:Scan
        - dynamodb:PutItem
        - dynamodb:UpdateItem
      Resource: "arn:aws:dynamodb:${opt:region,
self:provider.region}:*:table/${self:provider.environment.DYNAMODB_TABLE}"
...
```

Provider details - IAM Roles

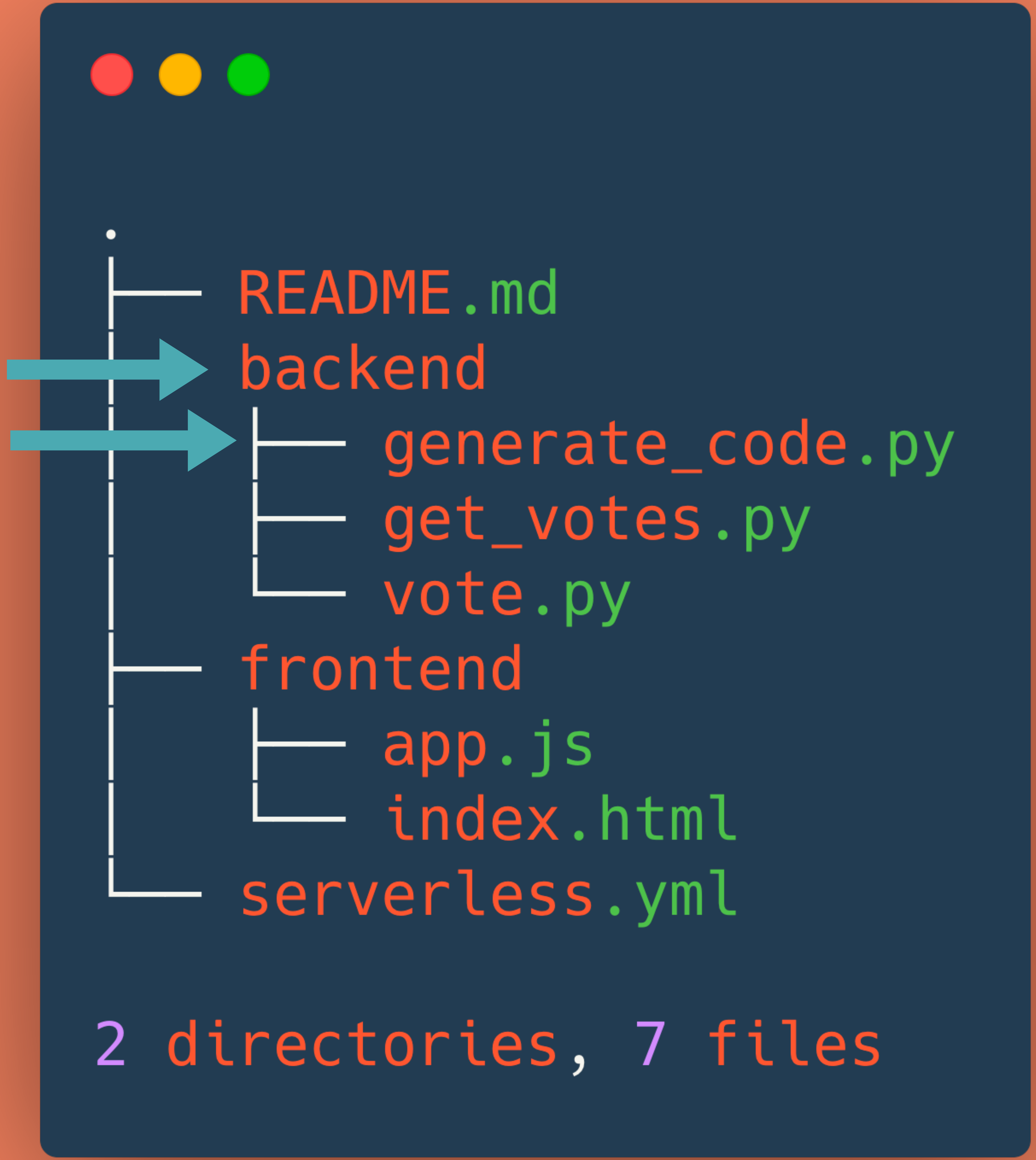


```
...
provider:
...
  iamRoleStatements:
...
    - Effect: Allow
      Action:
        - sns:Publish
      Resource: "*"
...
```

Function Definitions

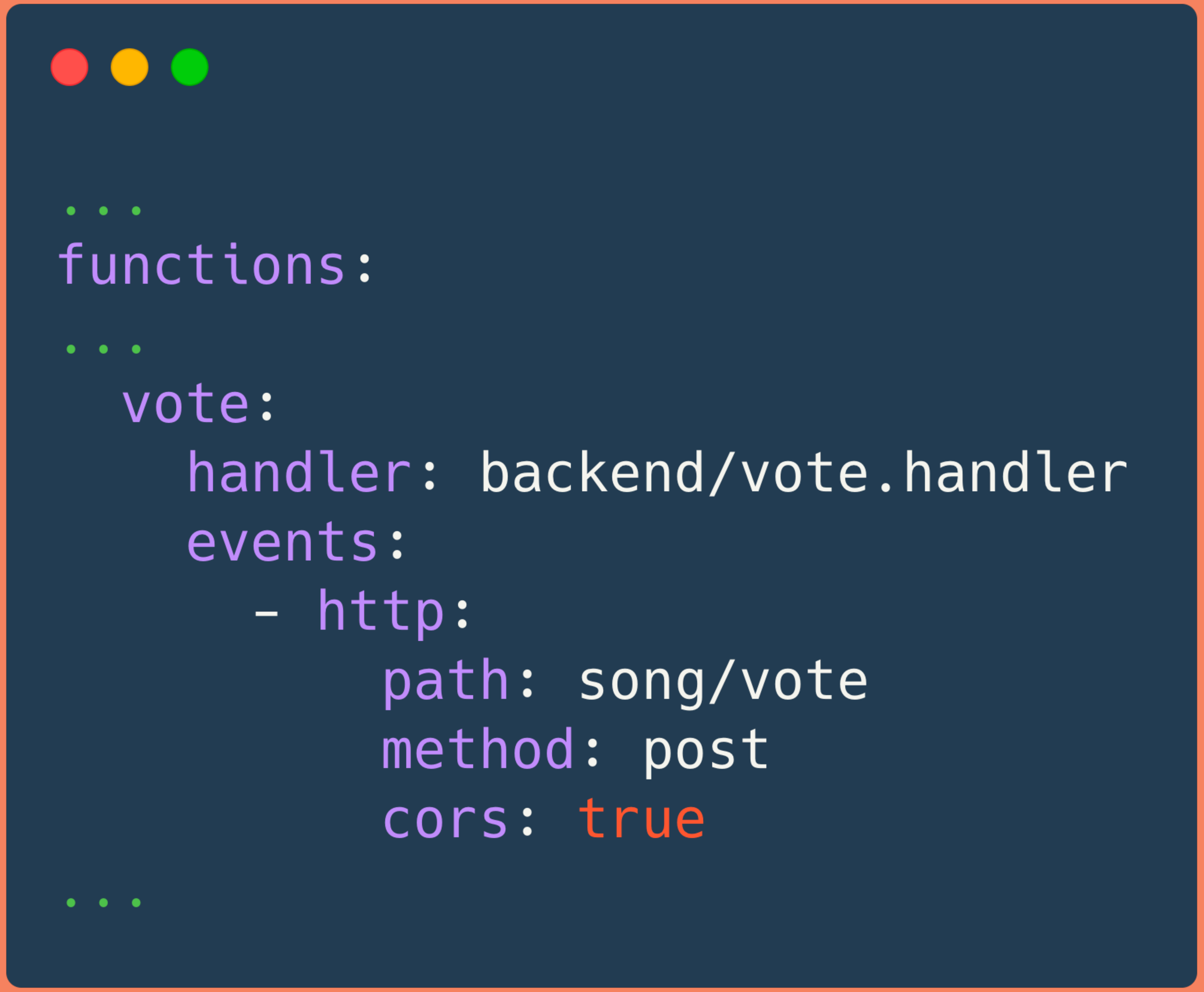


```
...
functions:
  generateCode:
    handler: backend/generate_code.handler
    events:
      - http:
          path: send-code
          method: post
          cors: true
...
```

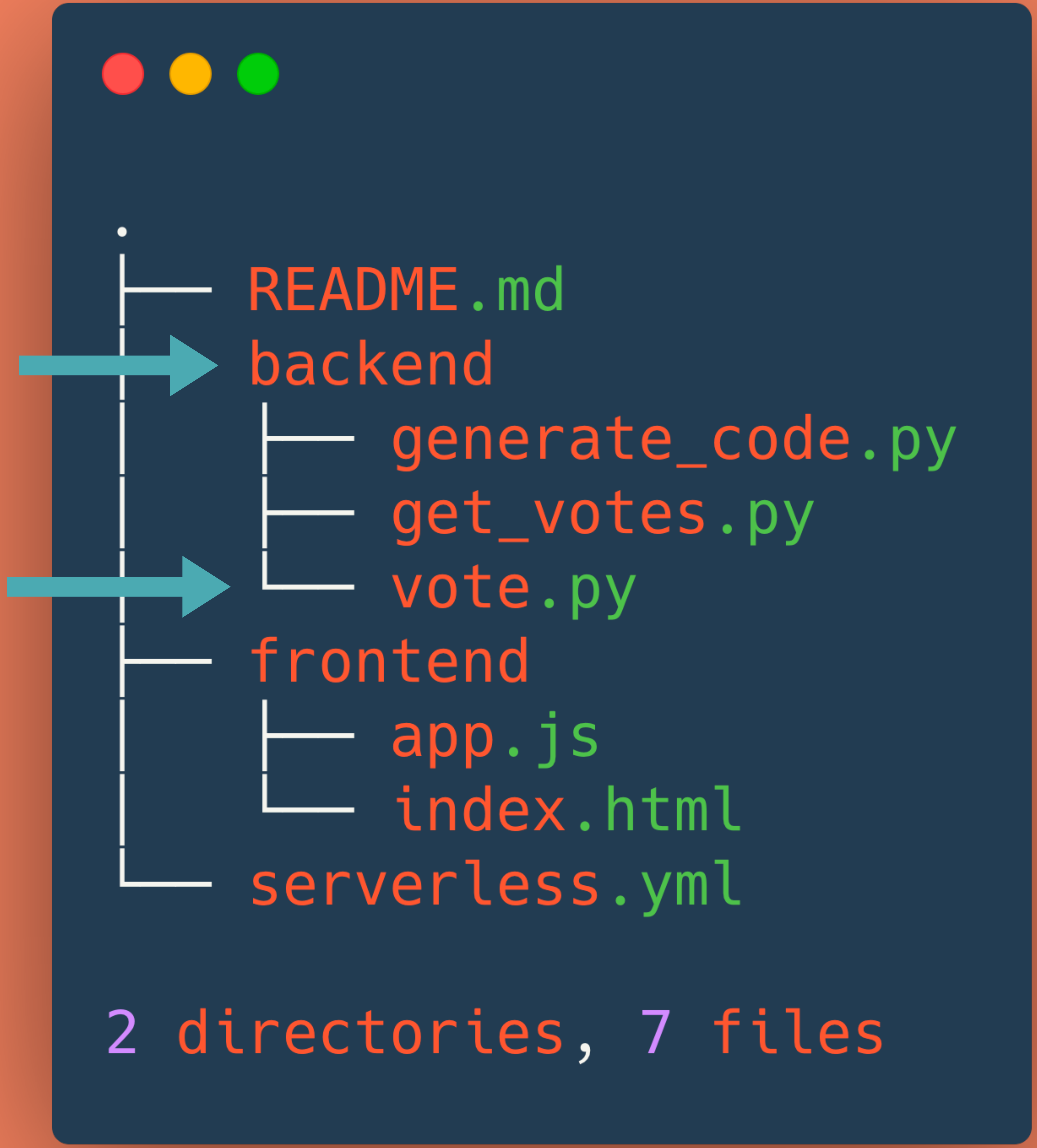


```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

Function Definitions

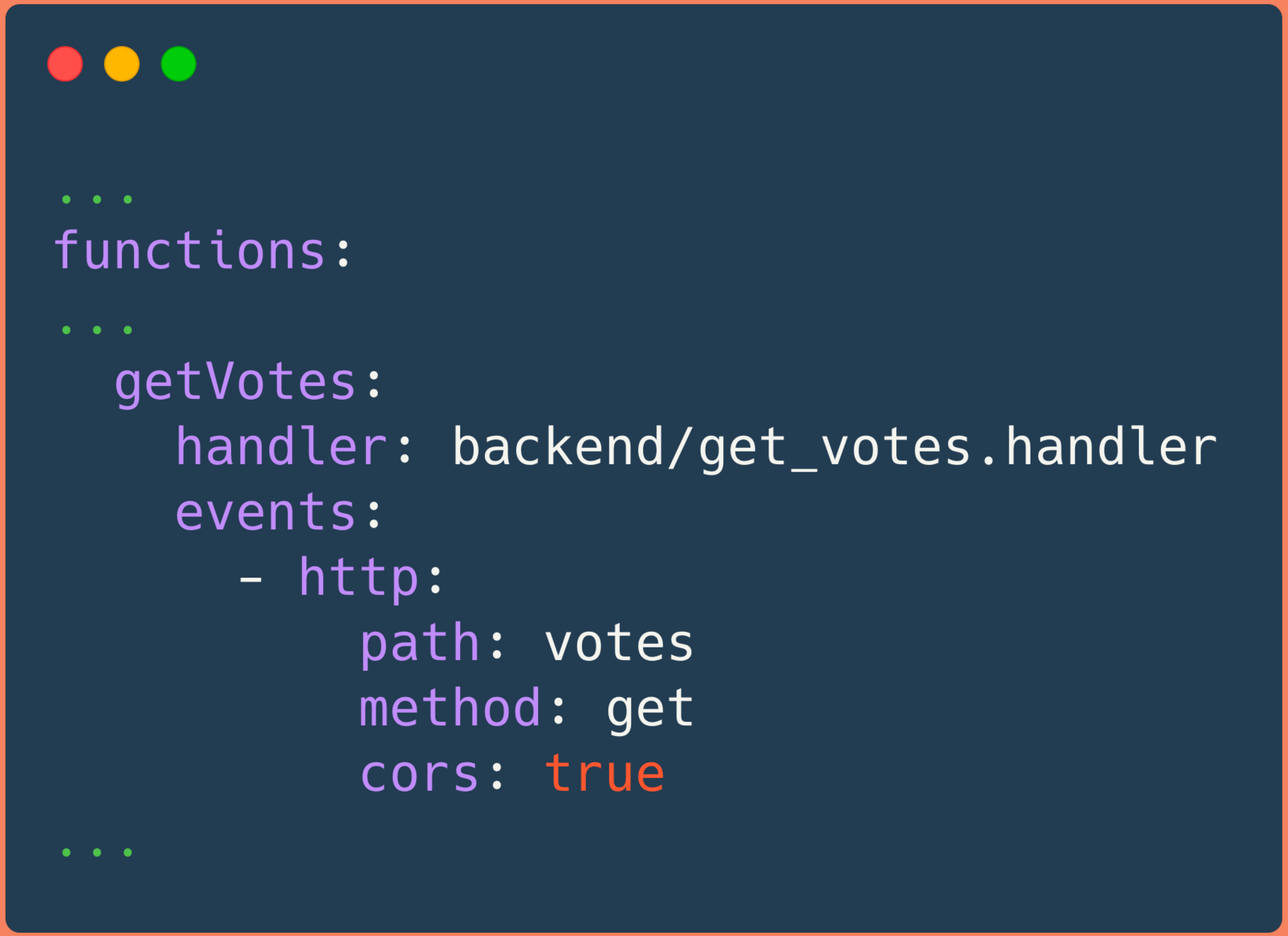


```
...
functions:
...
  vote:
    handler: backend/vote.handler
    events:
      - http:
          path: song/vote
          method: post
          cors: true
...
```

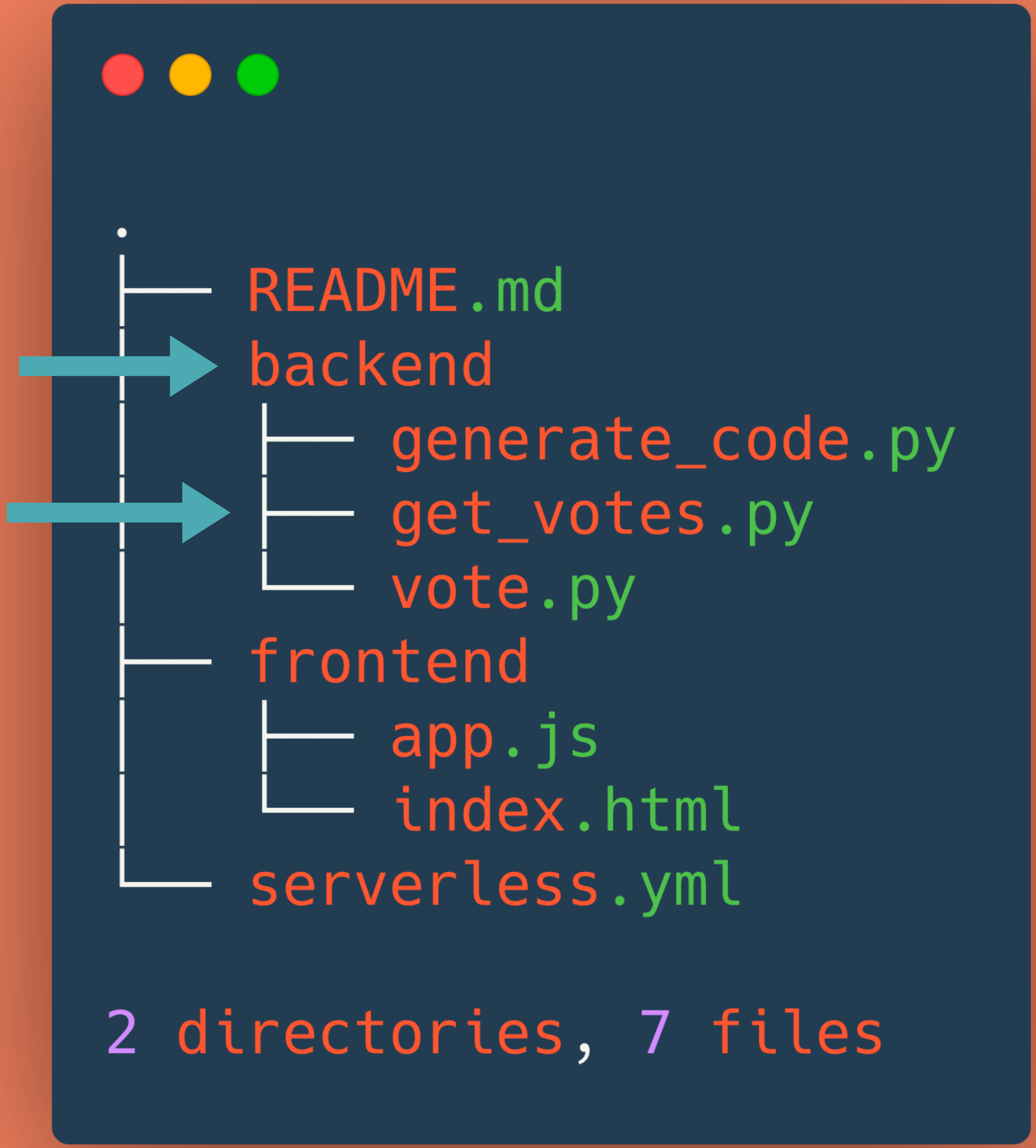


```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

Function Definitions



```
...
functions:
...
  getVotes:
    handler: backend/get_votes.handler
    events:
      - http:
          path: votes
          method: get
          cors: true
...
```

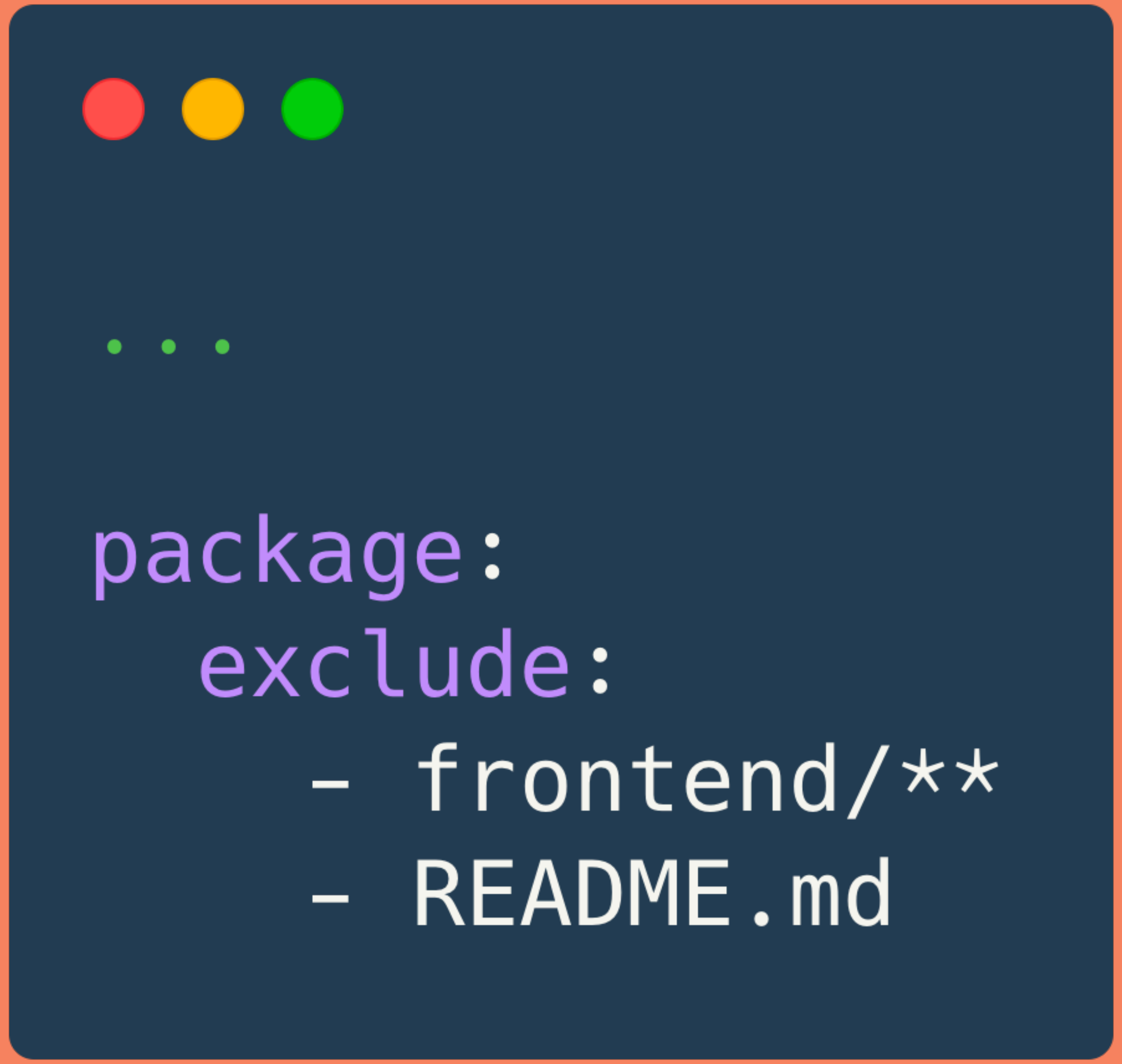


```
.
├── README.md
├── backend
│   ├── generate_code.py
│   ├── get_votes.py
│   └── vote.py
├── frontend
│   ├── app.js
│   └── index.html
└── serverless.yml

2 directories, 7 files
```

Resources

```yaml
...
resources:
  Resources:
    usersTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: ${self:provider.environment.DYNAMODB_TABLE}
        AttributeDefinitions:
          - AttributeName: pk
            AttributeType: S
          - AttributeName: sk
            AttributeType: S
        KeySchema:
          - AttributeName: pk
            KeyType: HASH
          - AttributeName: sk
            KeyType: RANGE
        ProvisionedThroughput:
          ReadCapacityUnits: 1
          WriteCapacityUnits: 1
...
```

Optional Packaging

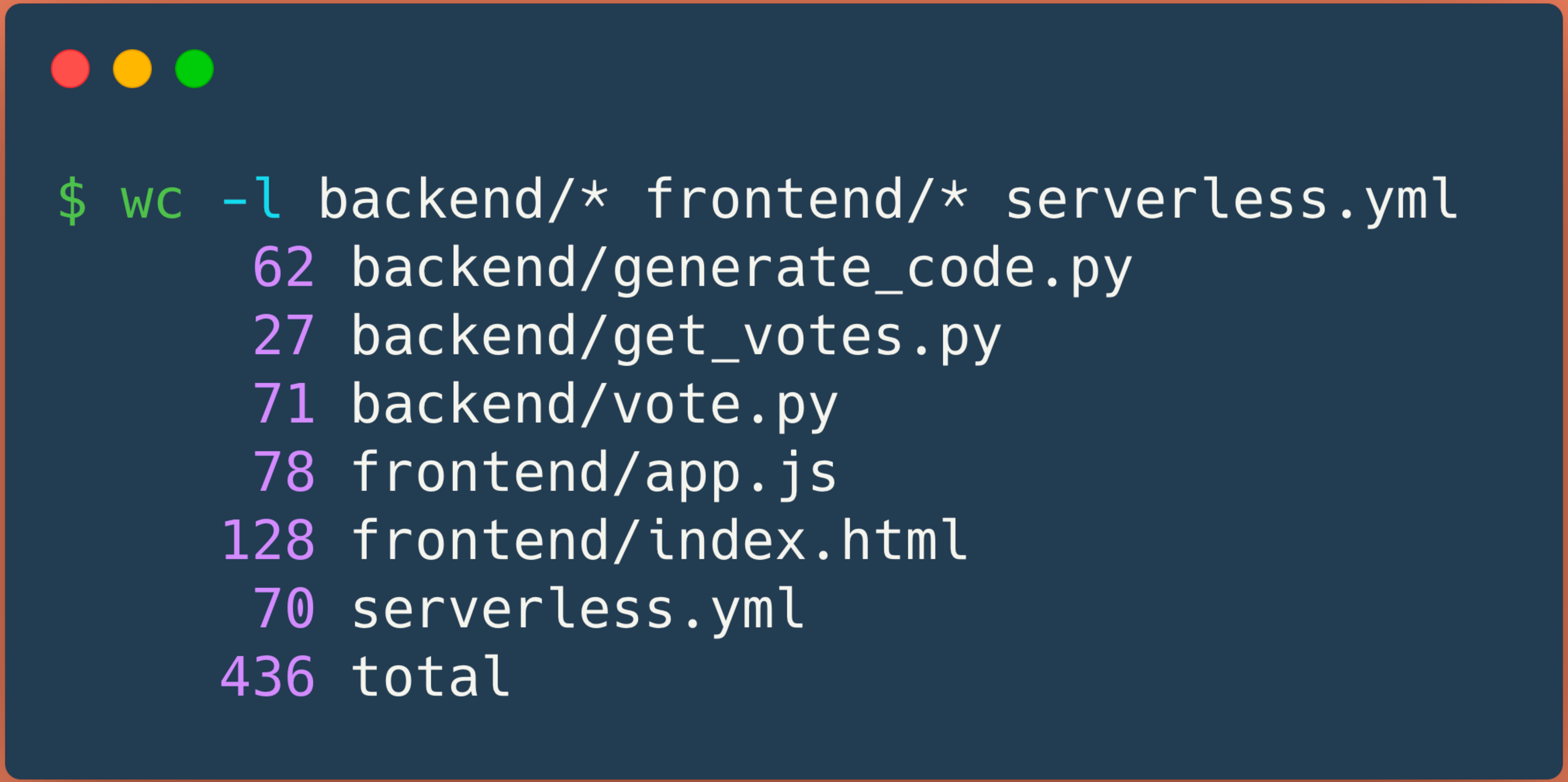


```
...

package:
  exclude:
    - frontend/**
    - README.md
```

How many total lines of code?

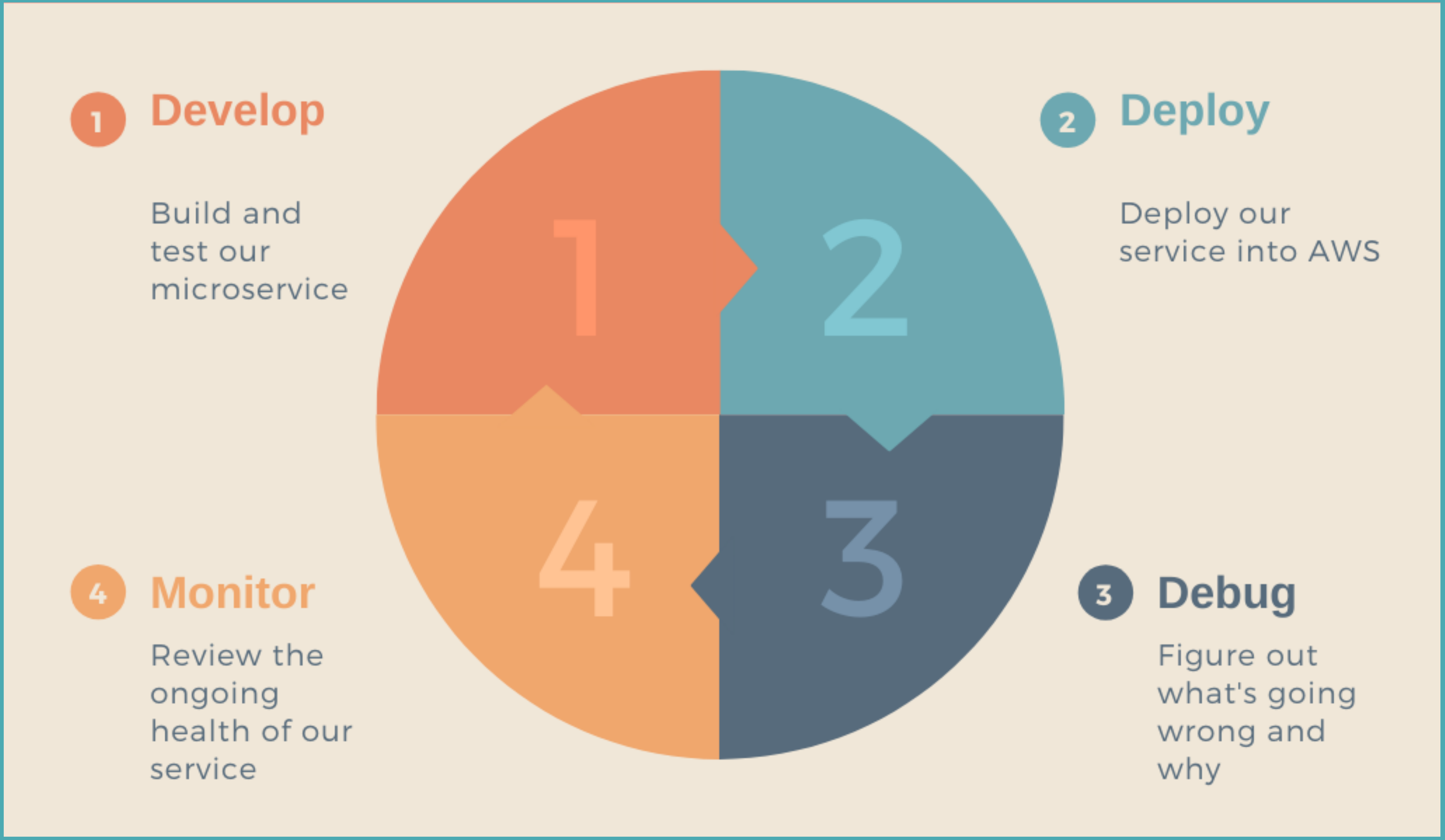
(README not included)

Total lines: 436

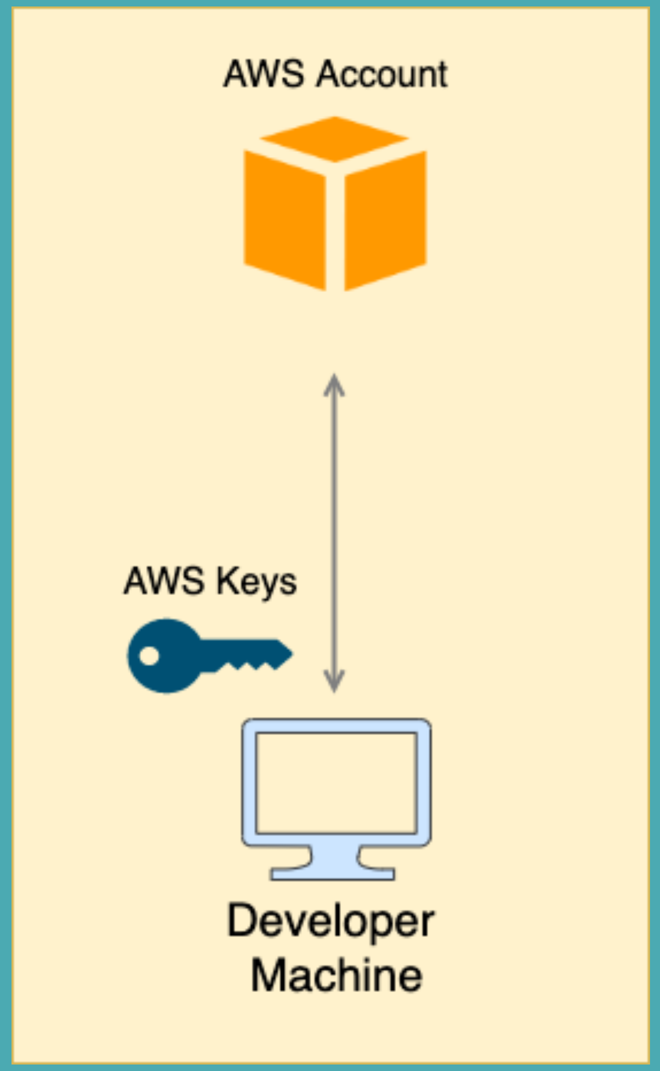


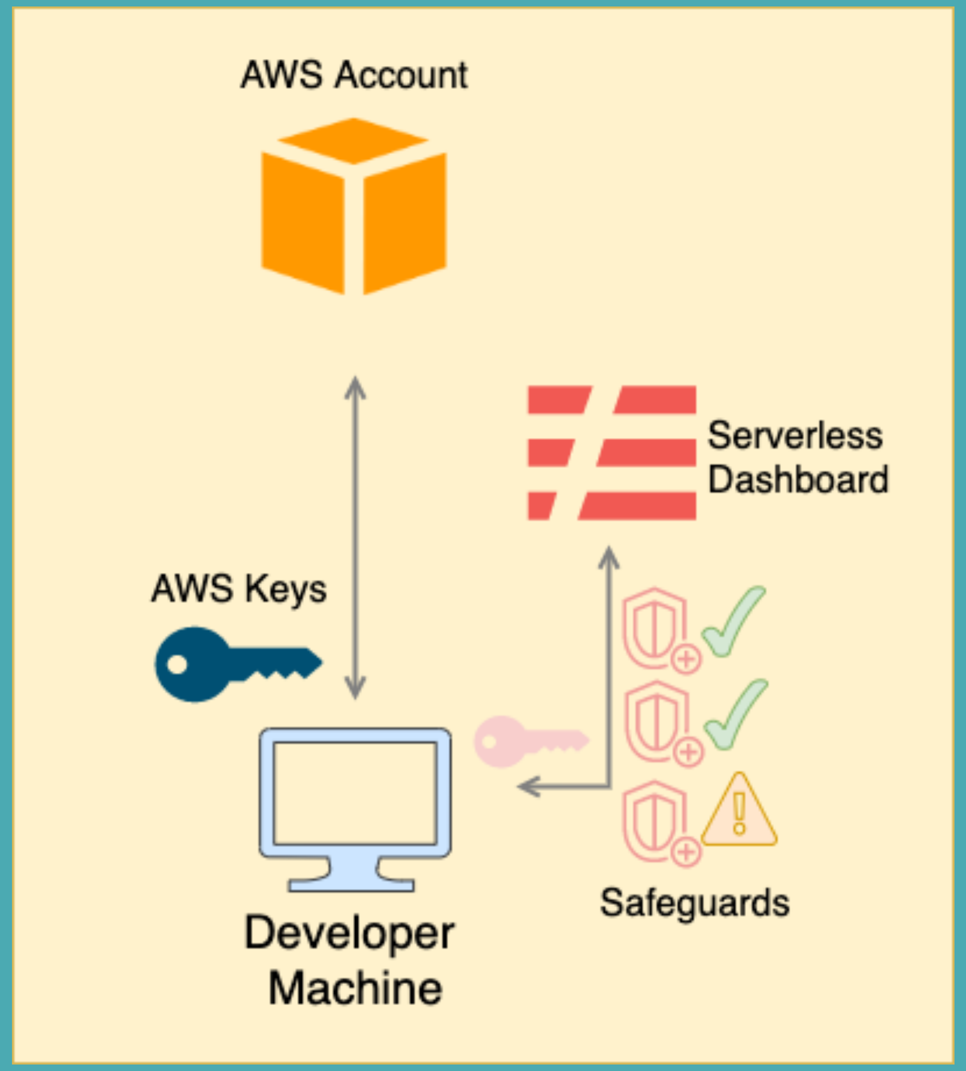
```
$ wc -l backend/* frontend/* serverless.yml
      62 backend/generate_code.py
      27 backend/get_votes.py
      71 backend/vote.py
      78 frontend/app.js
     128 frontend/index.html
      70 serverless.yml
     436 total
```

Deploy

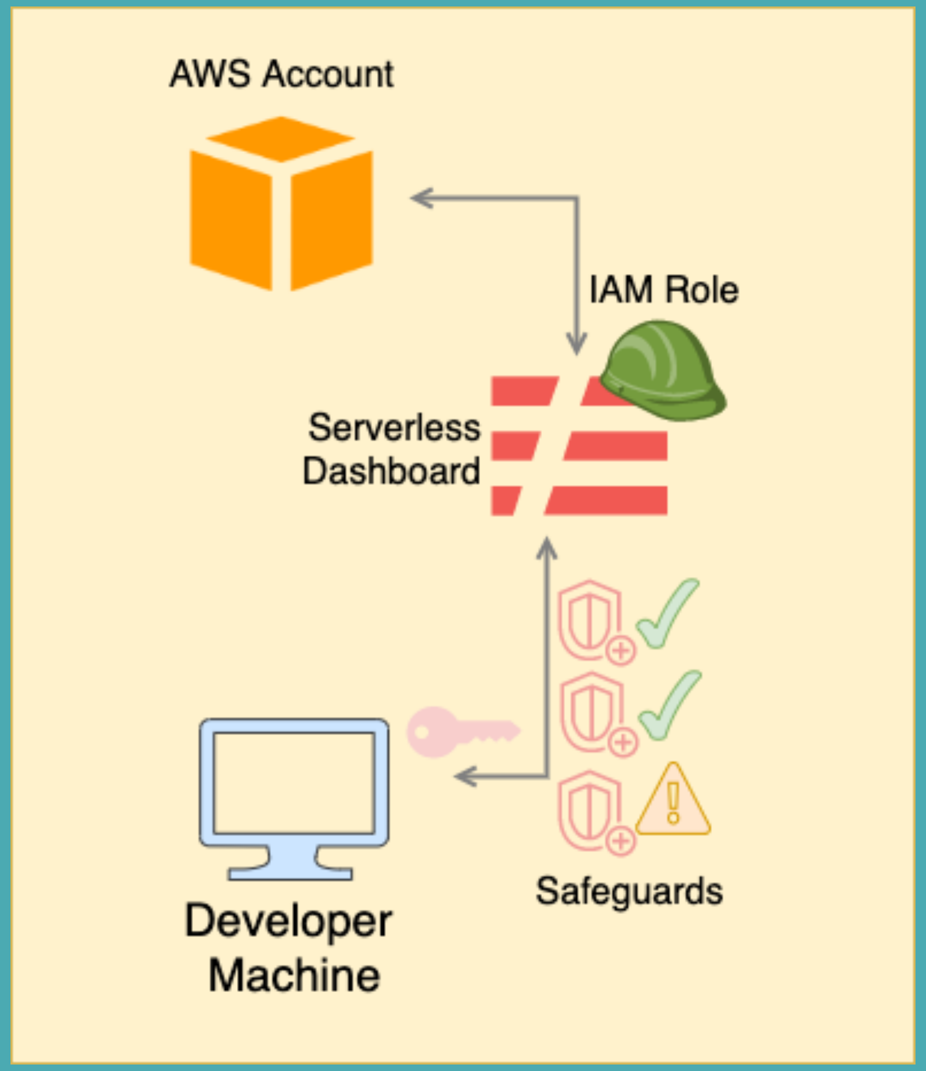

1 Develop
Build and test our microservice
2 Deploy
Deploy our service into AWS
3 Debug
Figure out what's going wrong and why
4 Monitor
Review the ongoing health of our service
1
2
3
4

Deployment Options

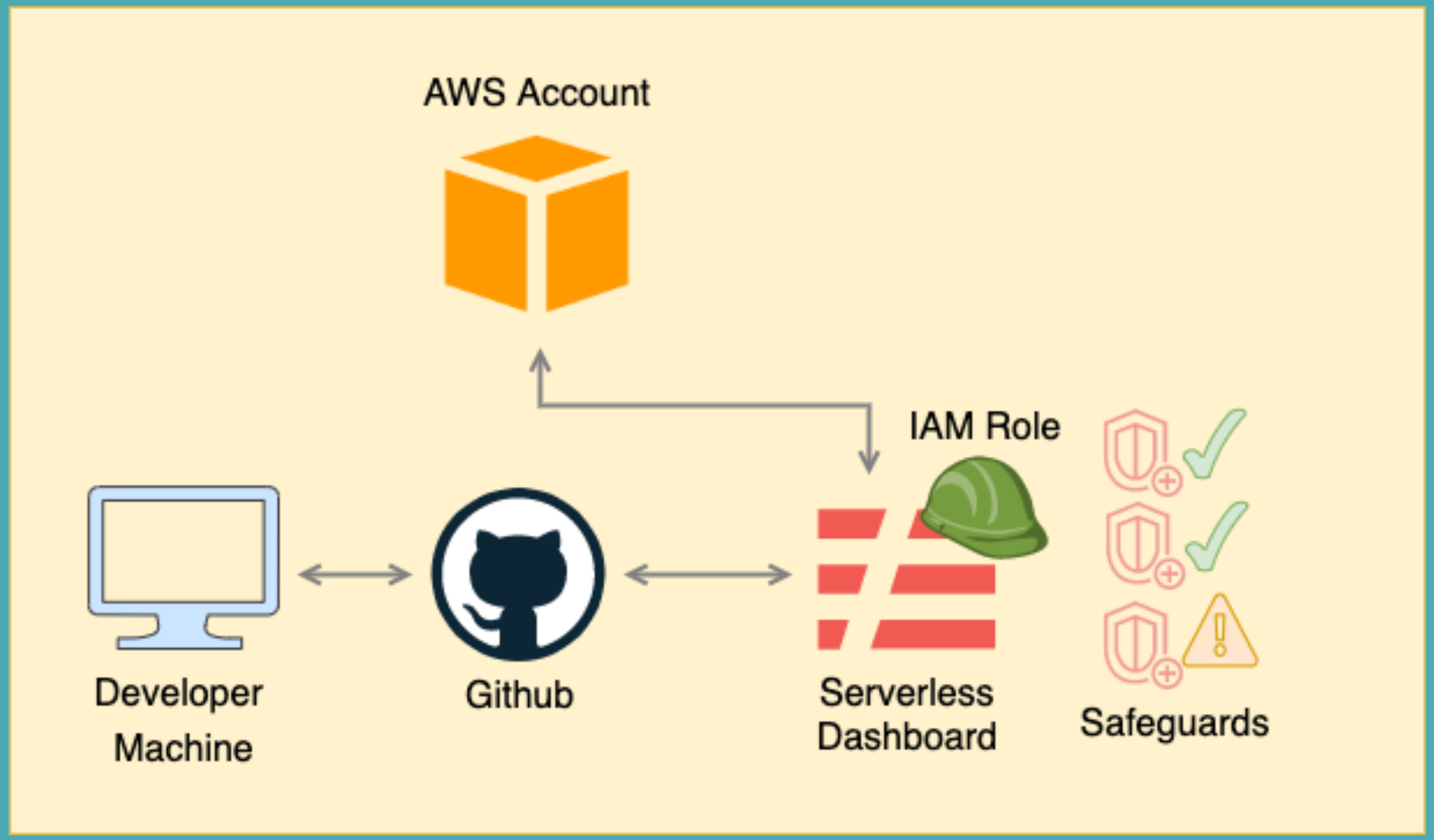




Local AWS Keys
(2 options)

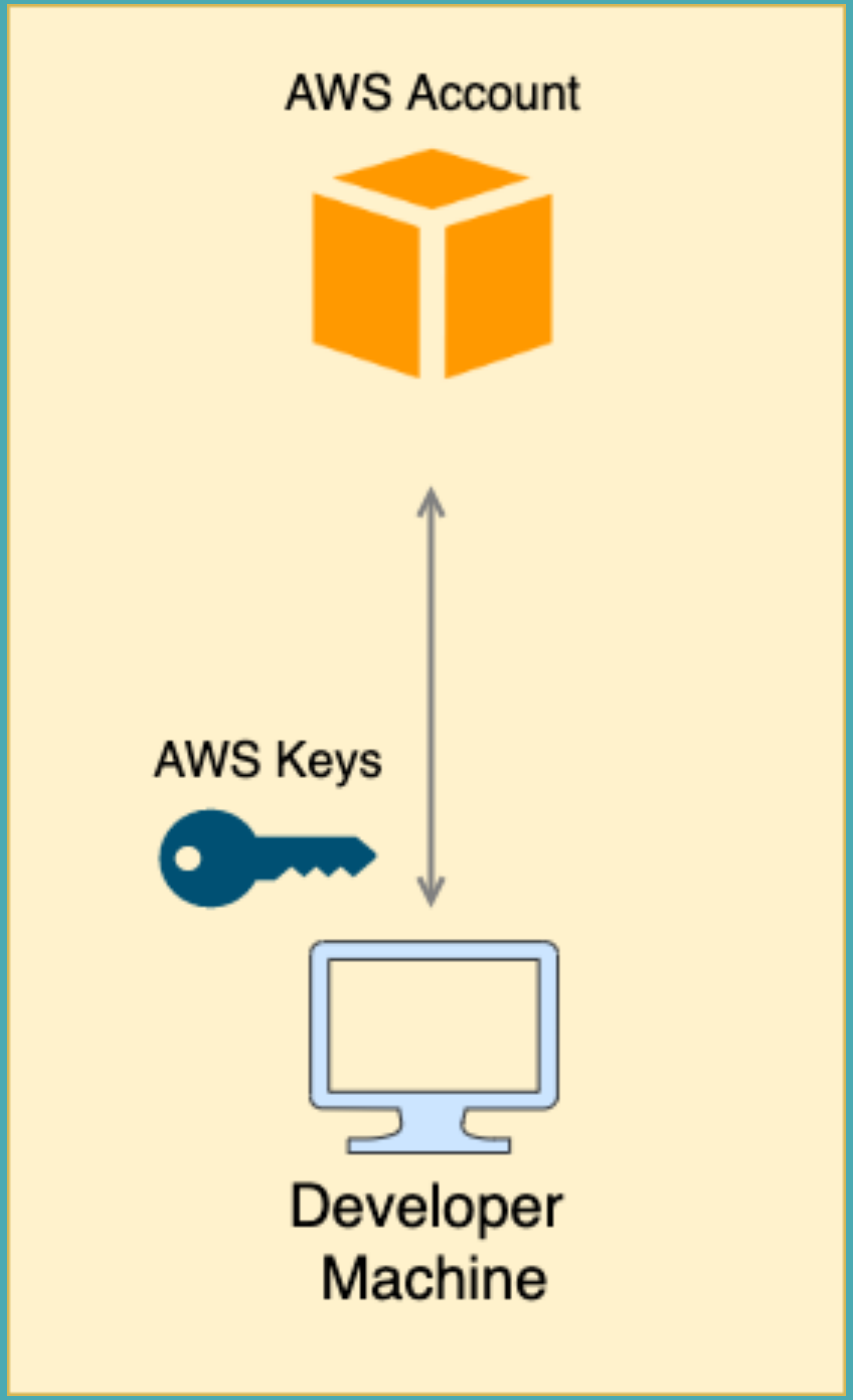


Using the Serverless Dashboard

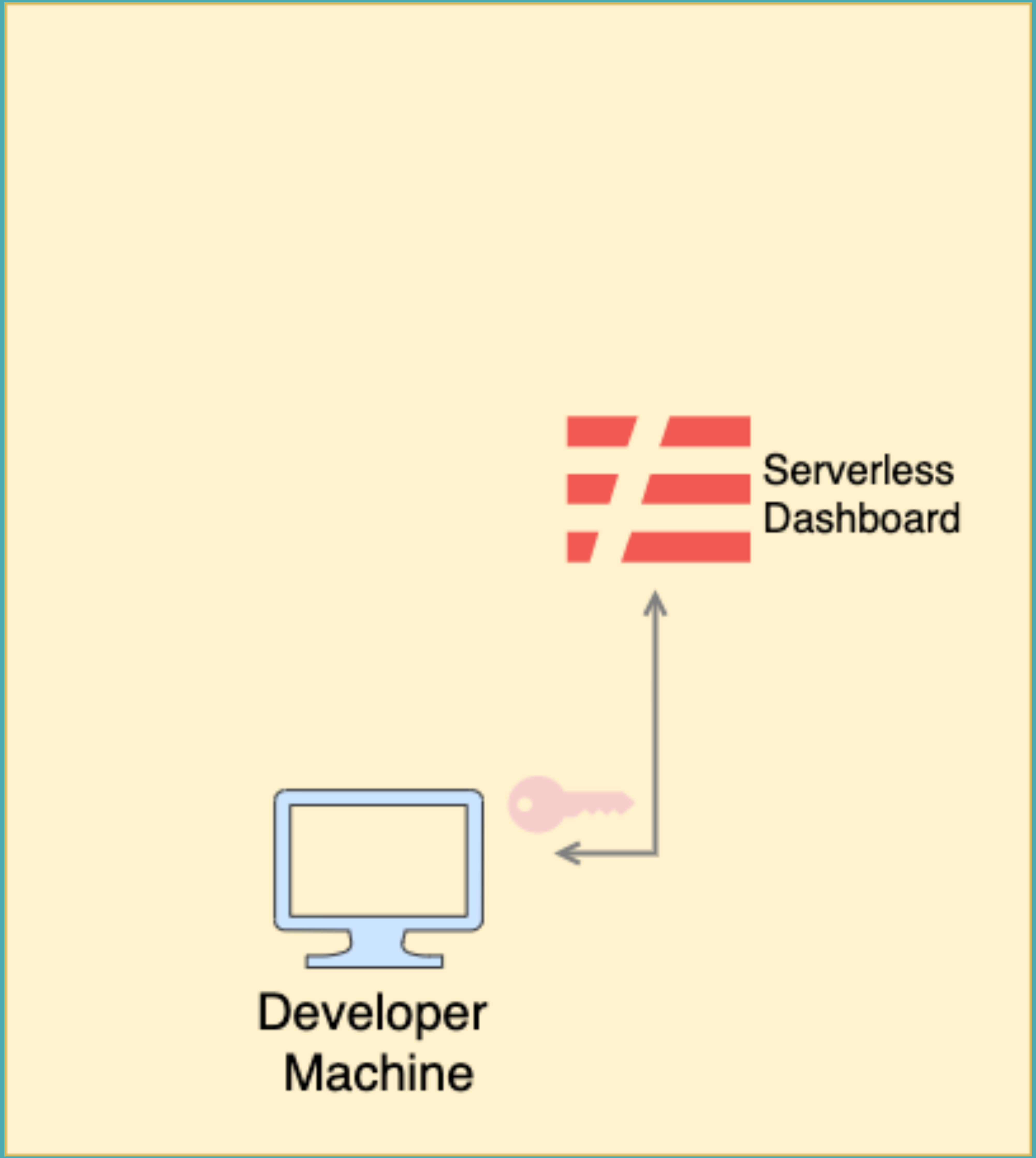


Using Serverless CI/CD

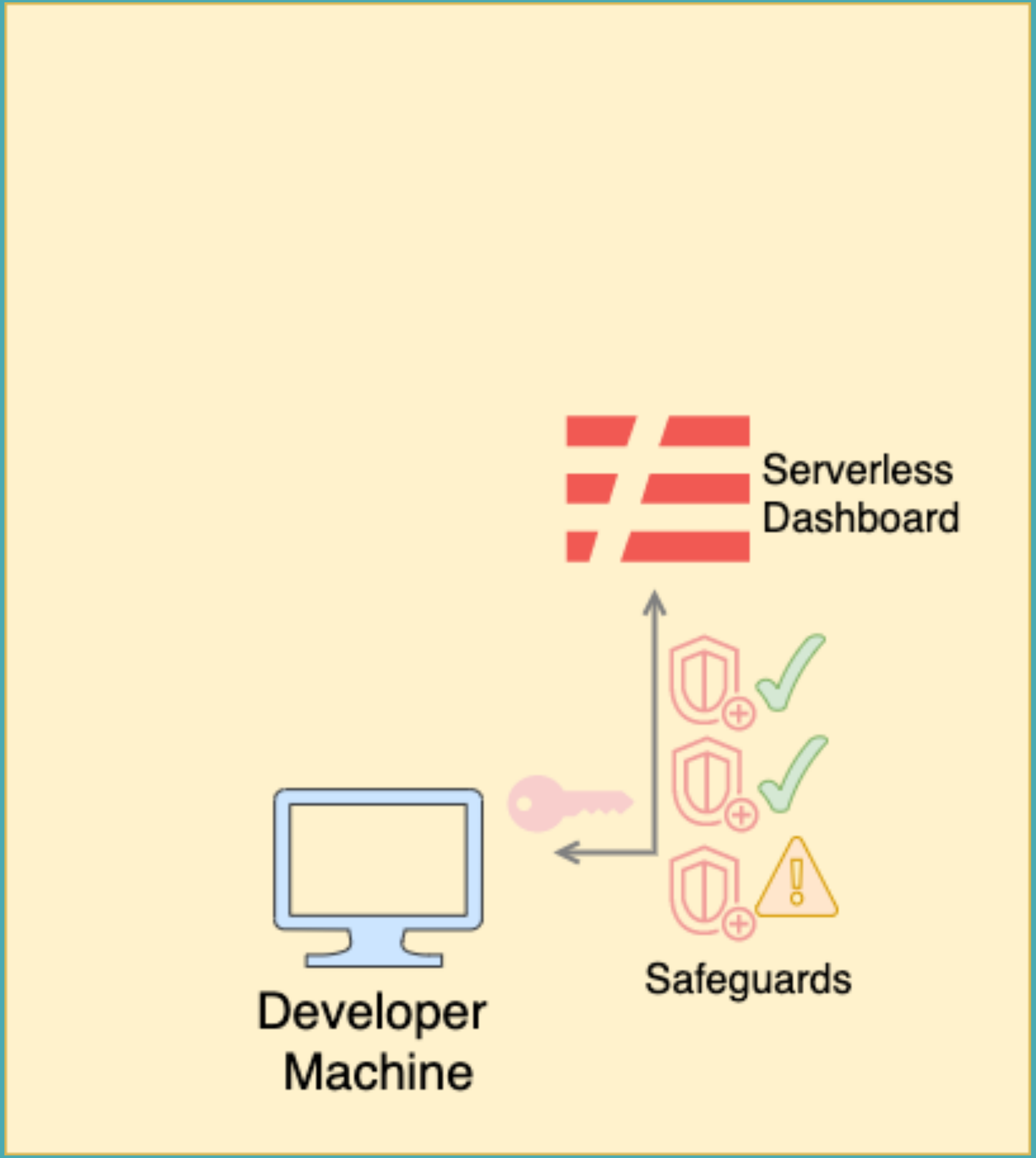
Local AWS Keys



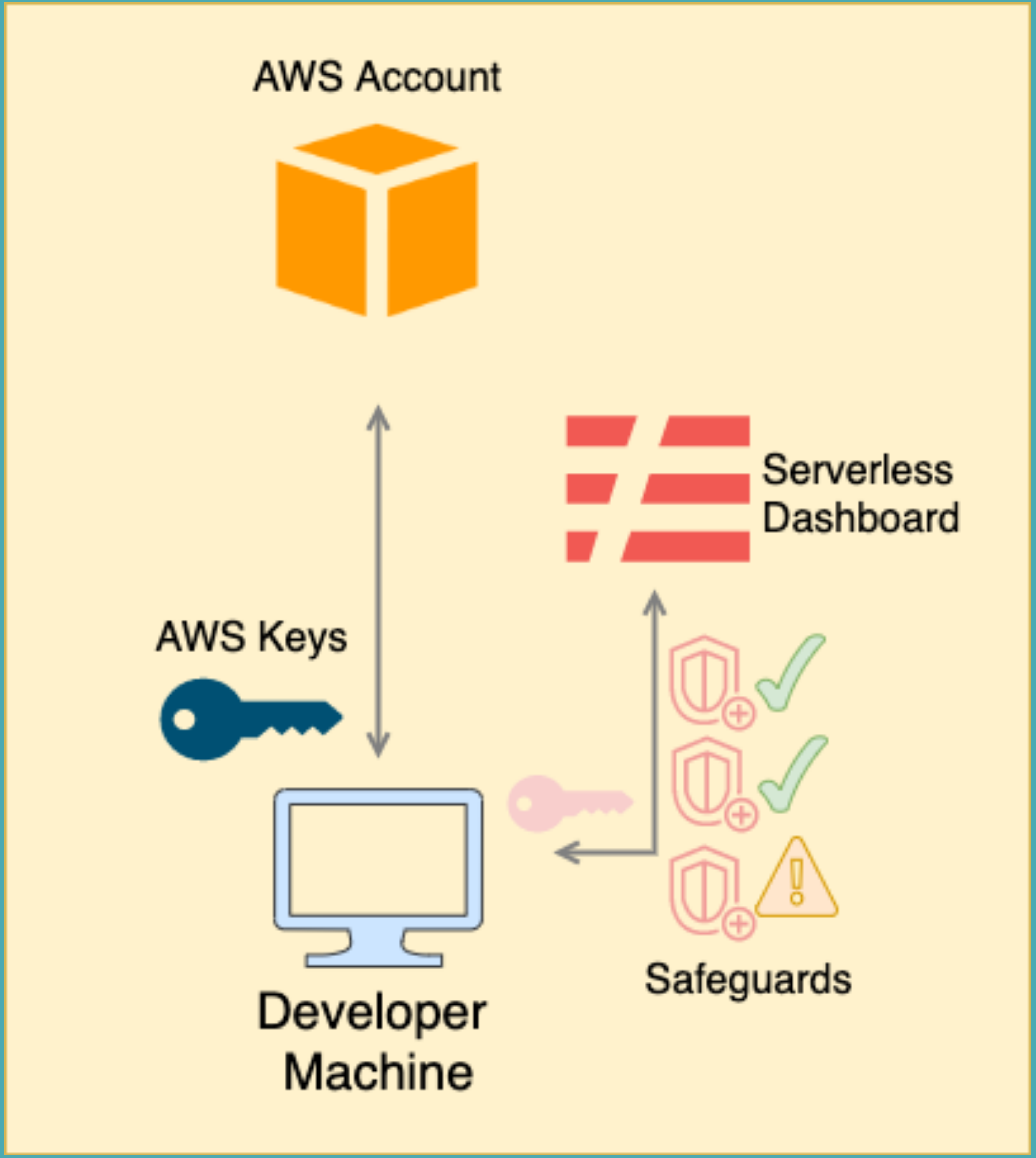
AWS Keys & Serverless Safeguards



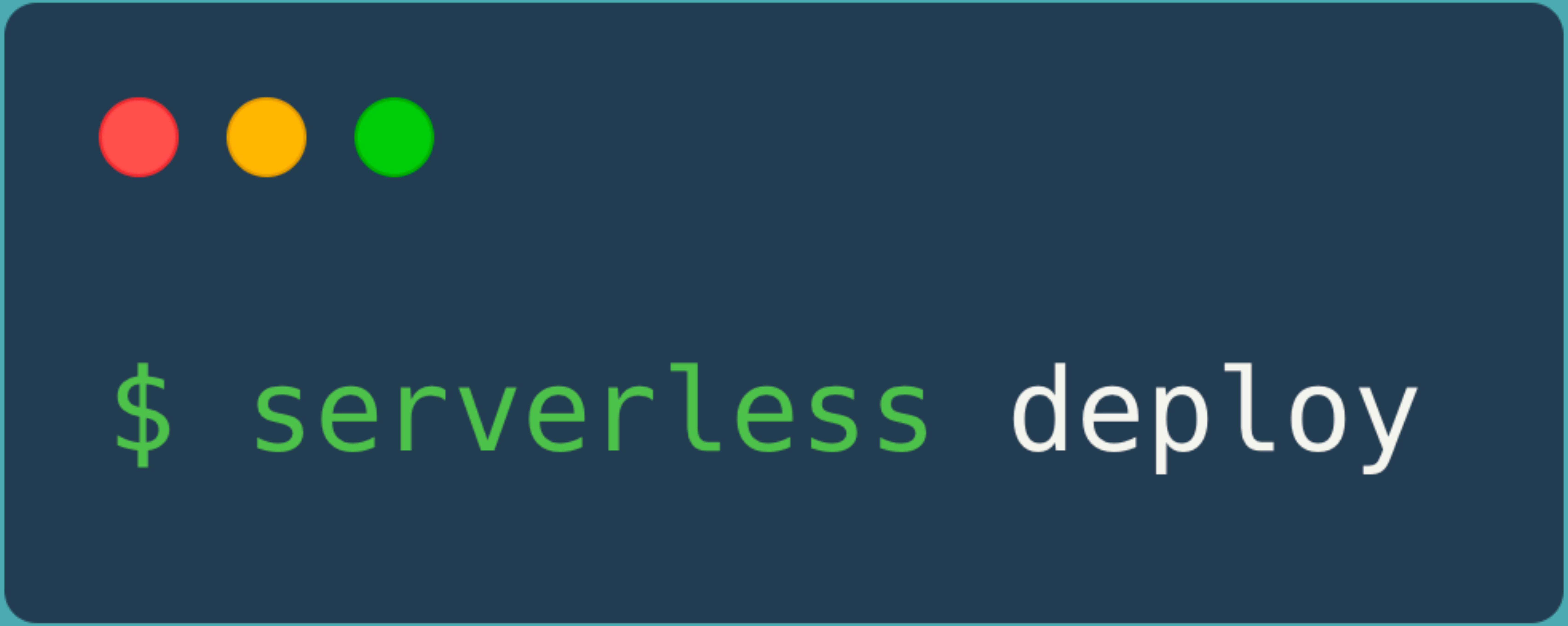
AWS Keys & Serverless Safeguards



AWS Keys & Serverless Safeguards

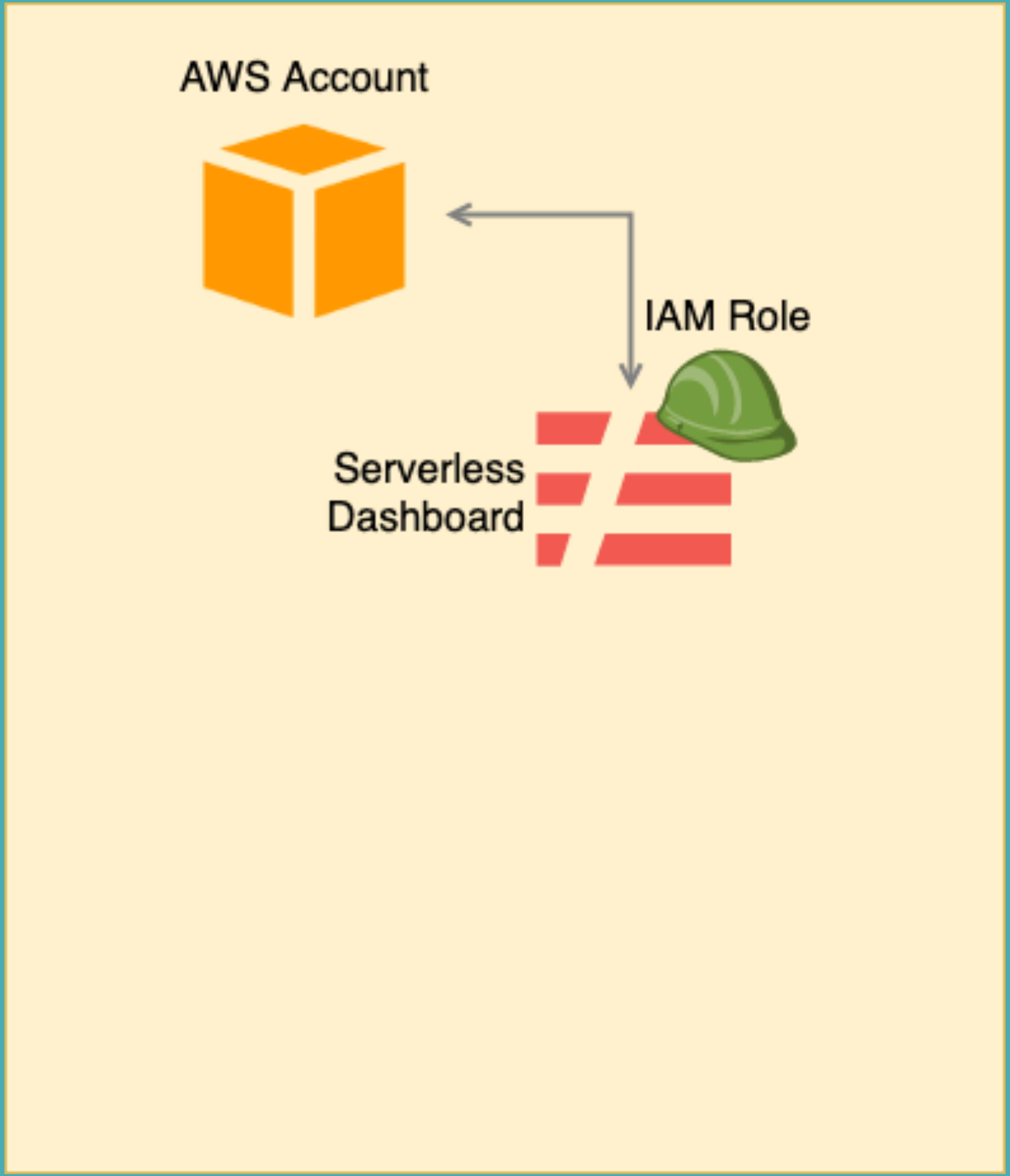


Deploying with Local Keys

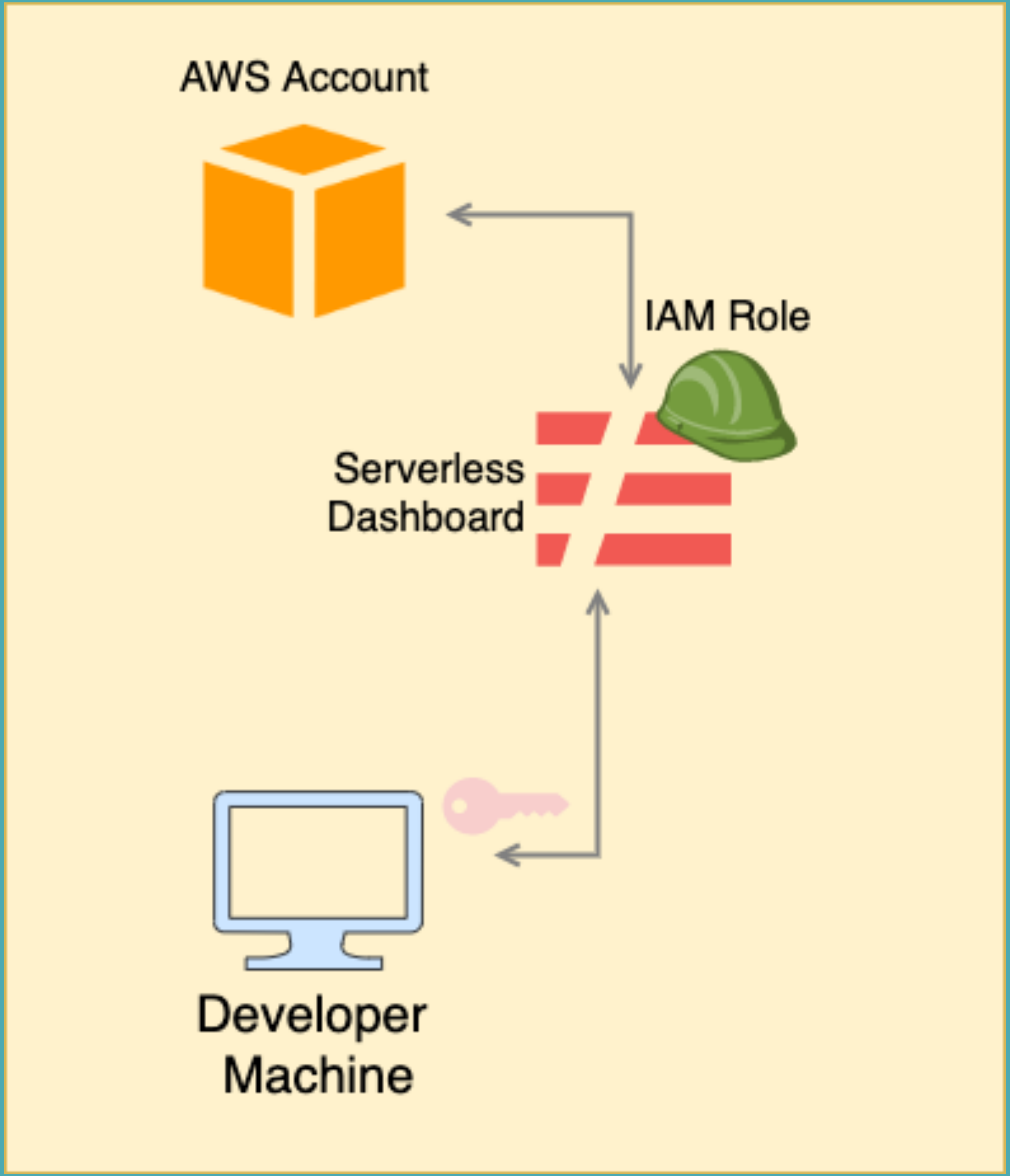


```
$ serverless deploy
```

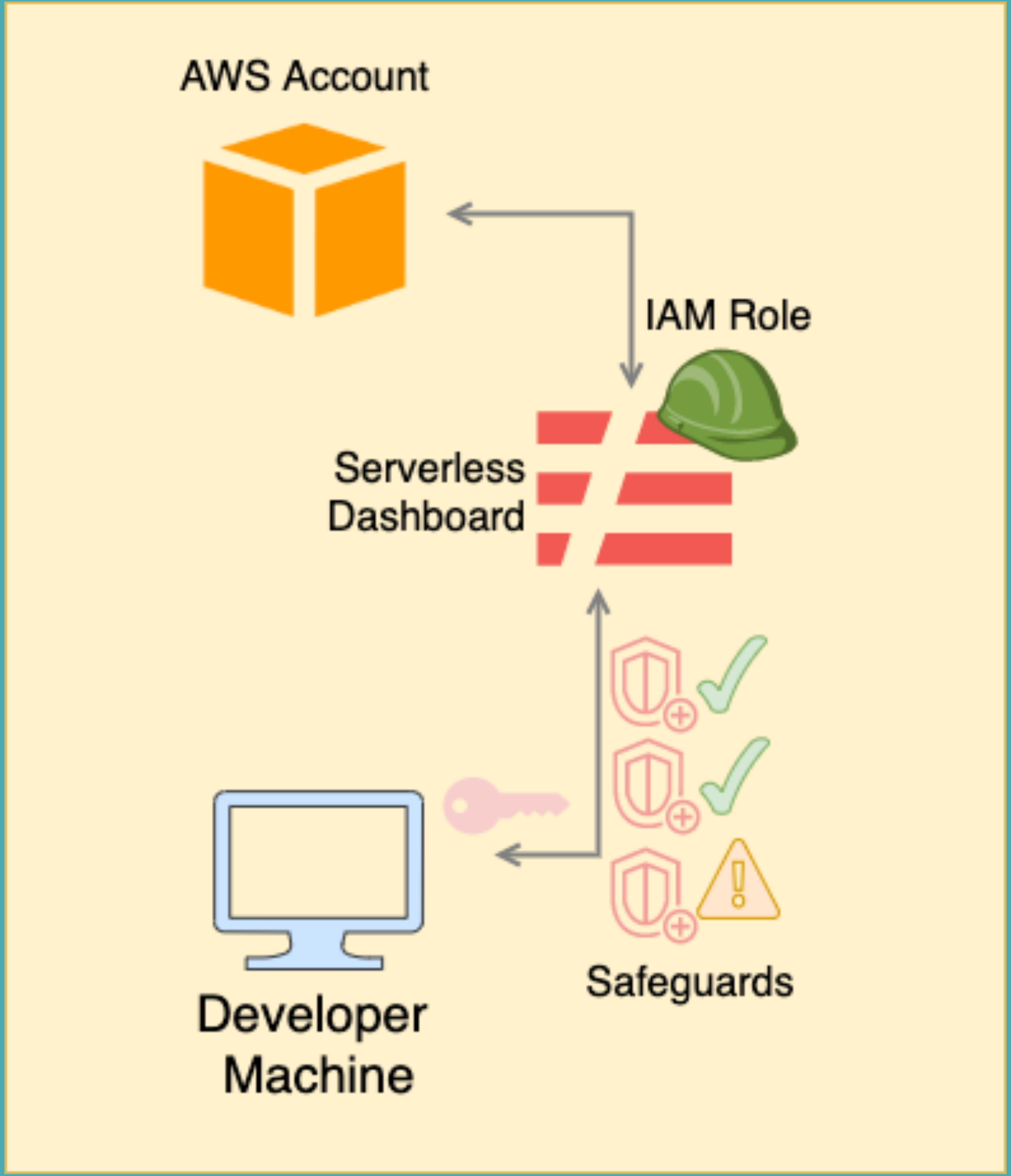
Deploying via the Serverless Dashboard



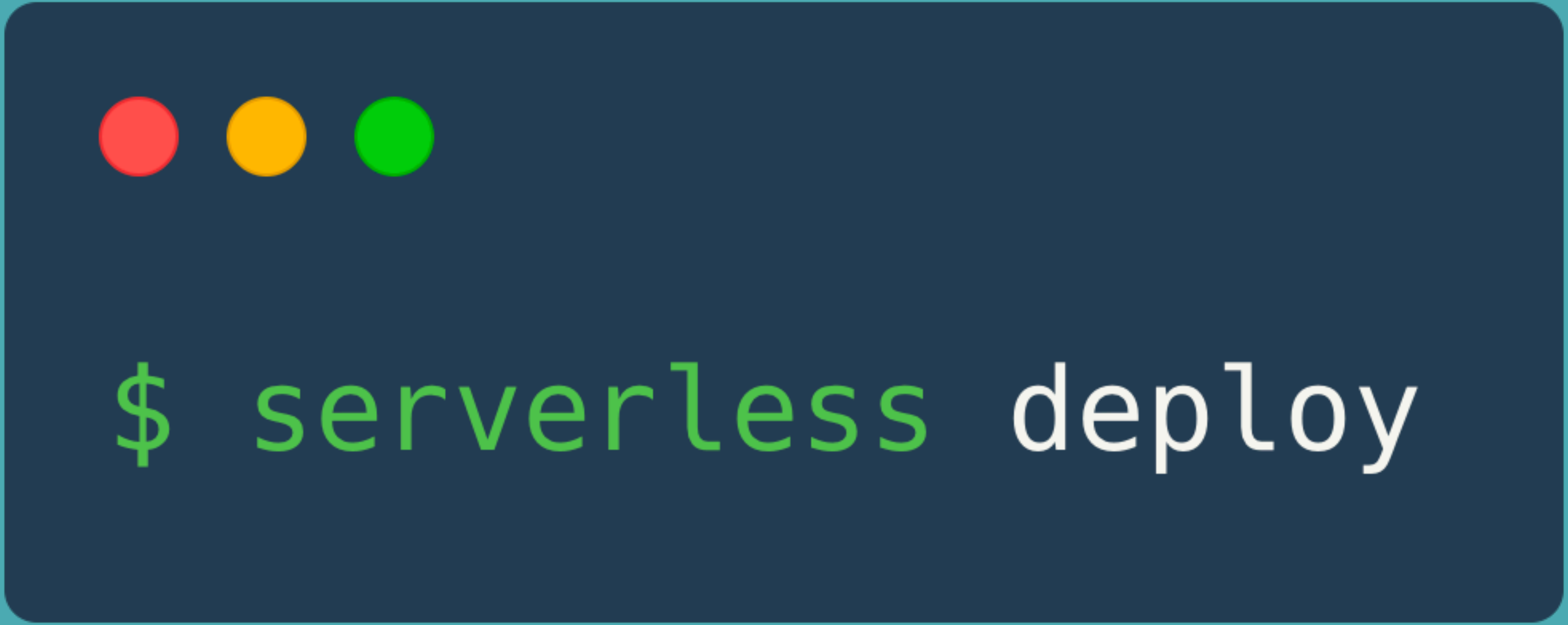
Deploying via the Serverless Dashboard



Deploying via the Serverless Dashboard

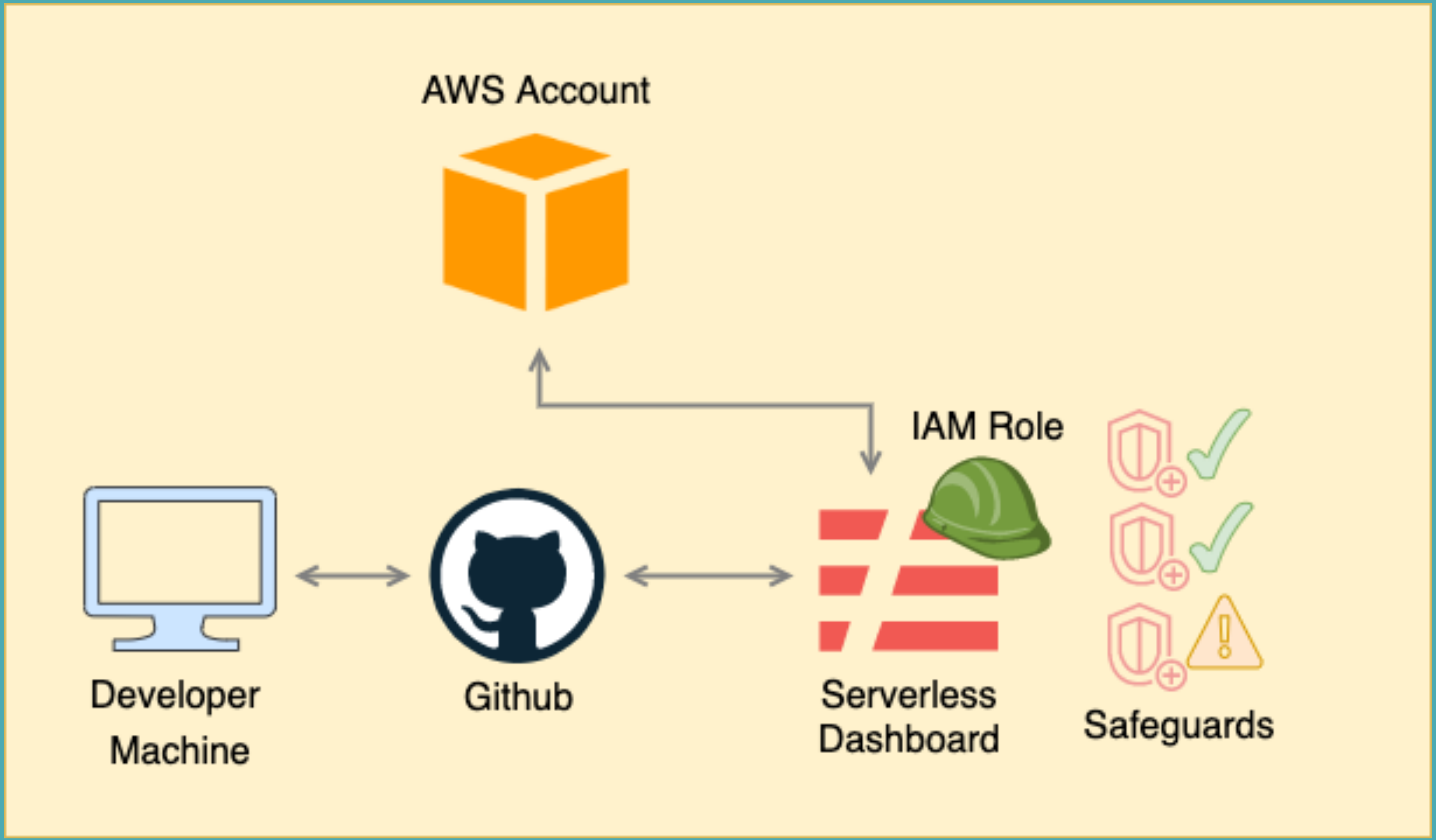


Deploying via the Serverless Dashboard



```
$ serverless deploy
```

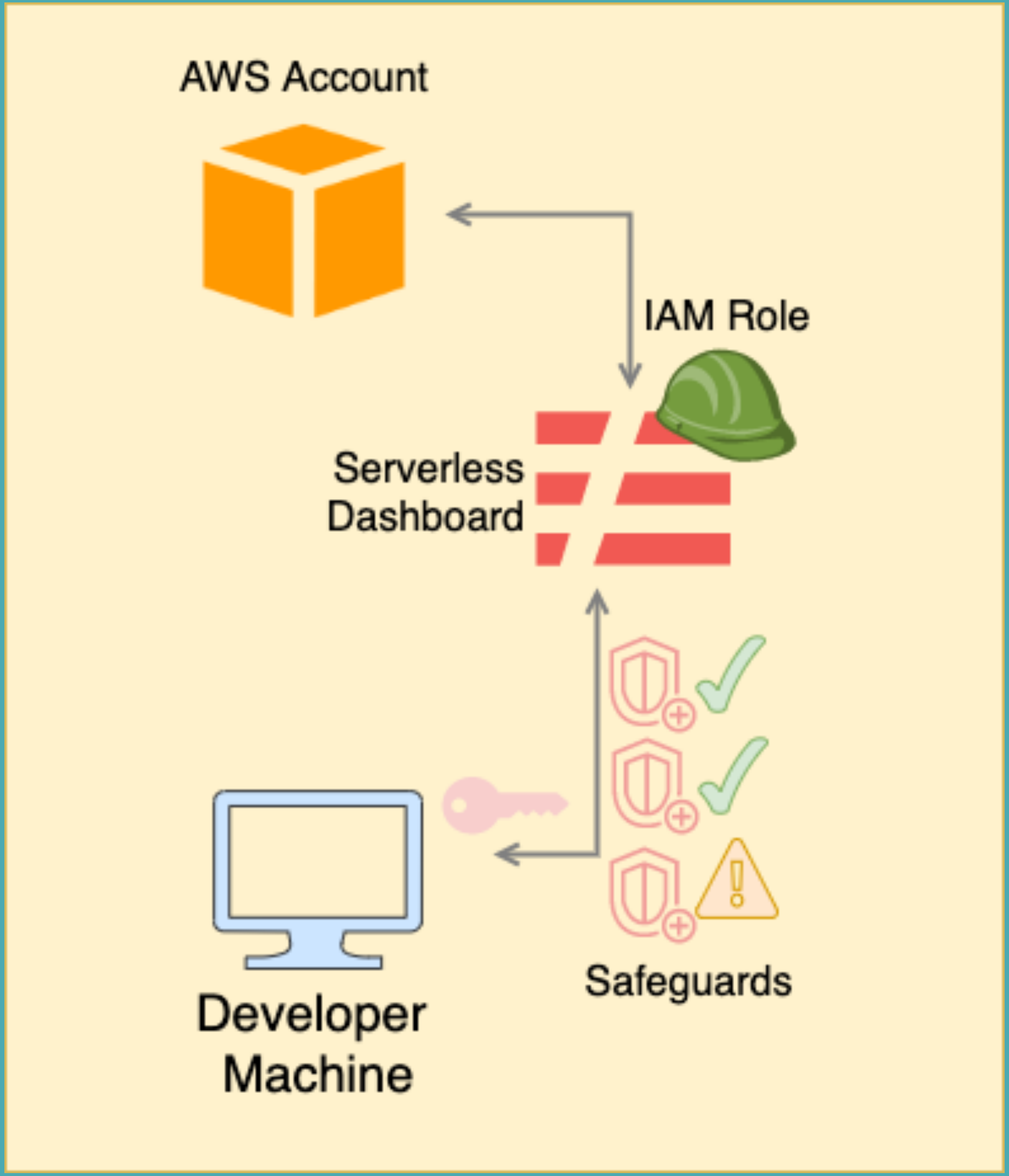
Deploying with Serverless CI/CD



What are we doing?

DEMO TIME!

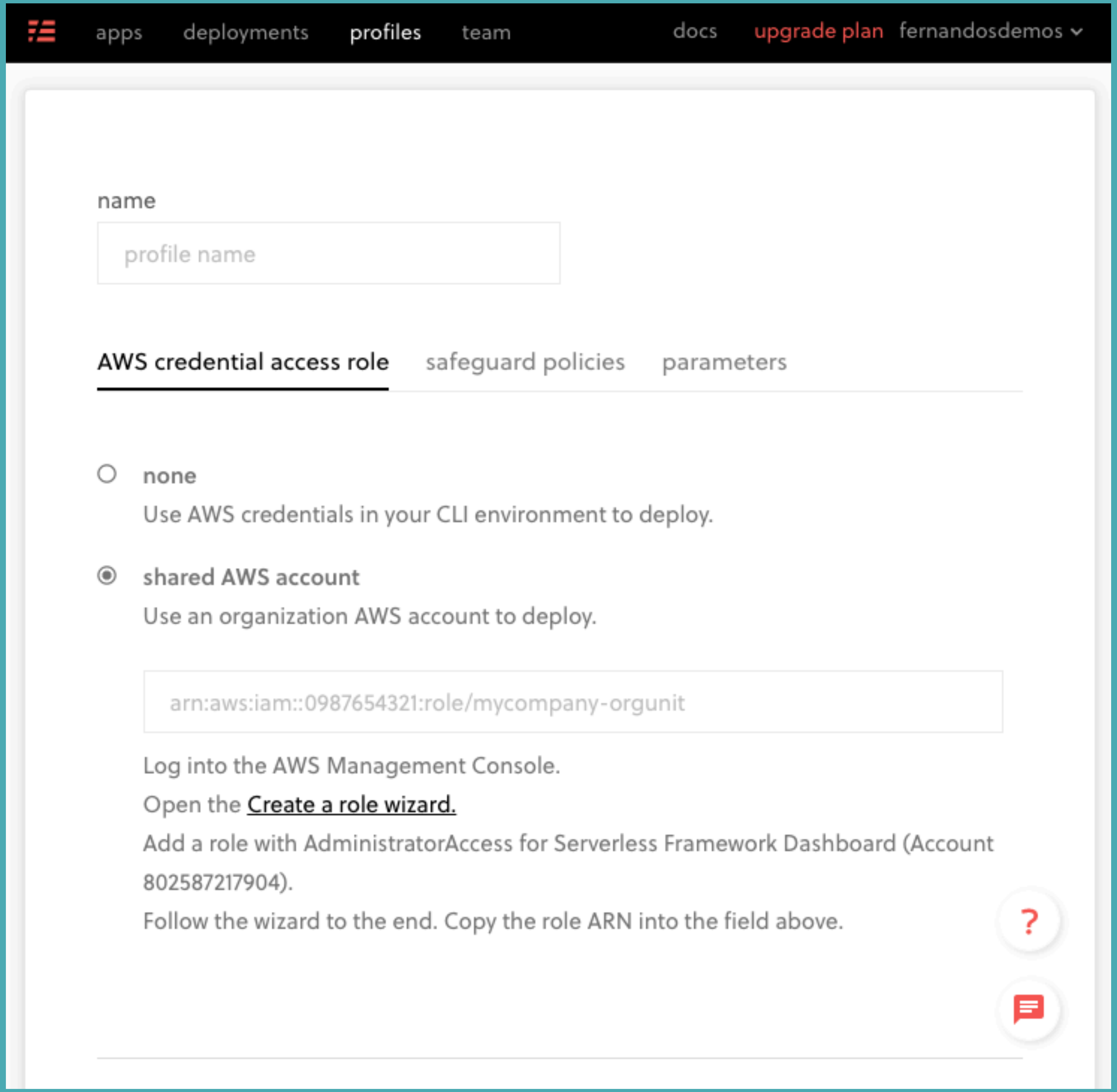
1. Review app/org values
2. Create/add an IAM role
3. Create Profiles and Safeguards
4. Deploy a `dev` and `prod` service
5. Add API Endpoints to our frontend



Just in case…

The demo went great

Why did I do a live demo?



name

profile name

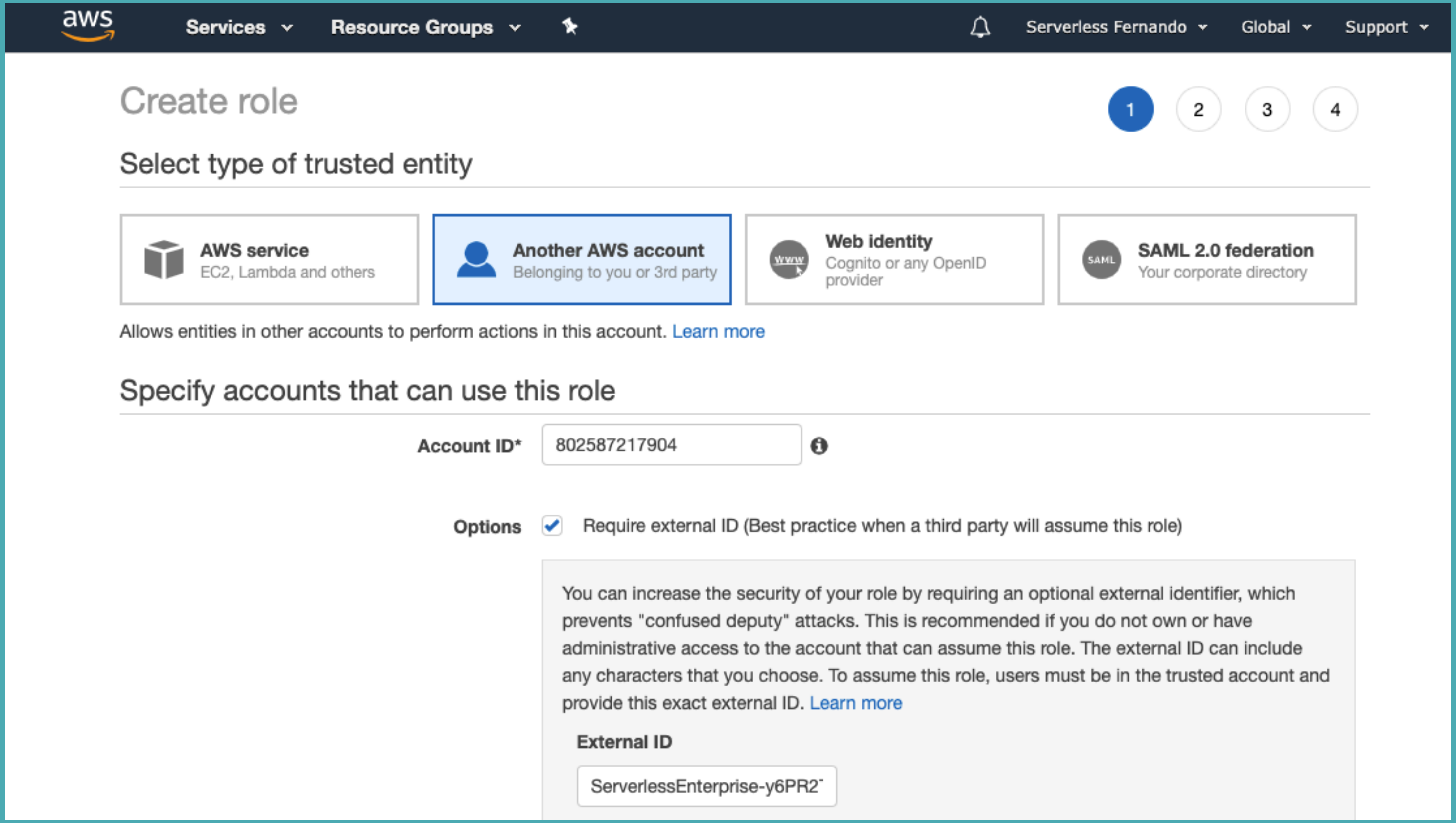
AWS credential access role | safeguard policies | parameters

○ none
Use AWS credentials in your CLI environment to deploy.

● shared AWS account
Use an organization AWS account to deploy.

arn:aws:iam::0987654321:role/mycompany-orgunit

Log into the AWS Management Console.
Open the Create a role wizard.
Add a role with AdministratorAccess for Serverless Framework Dashboard (Account 802587217904).
Follow the wizard to the end. Copy the role ARN into the field above.



Create role

Select type of trusted entity

Allows entities in other accounts to perform actions in this account. Learn more

Specify accounts that can use this role

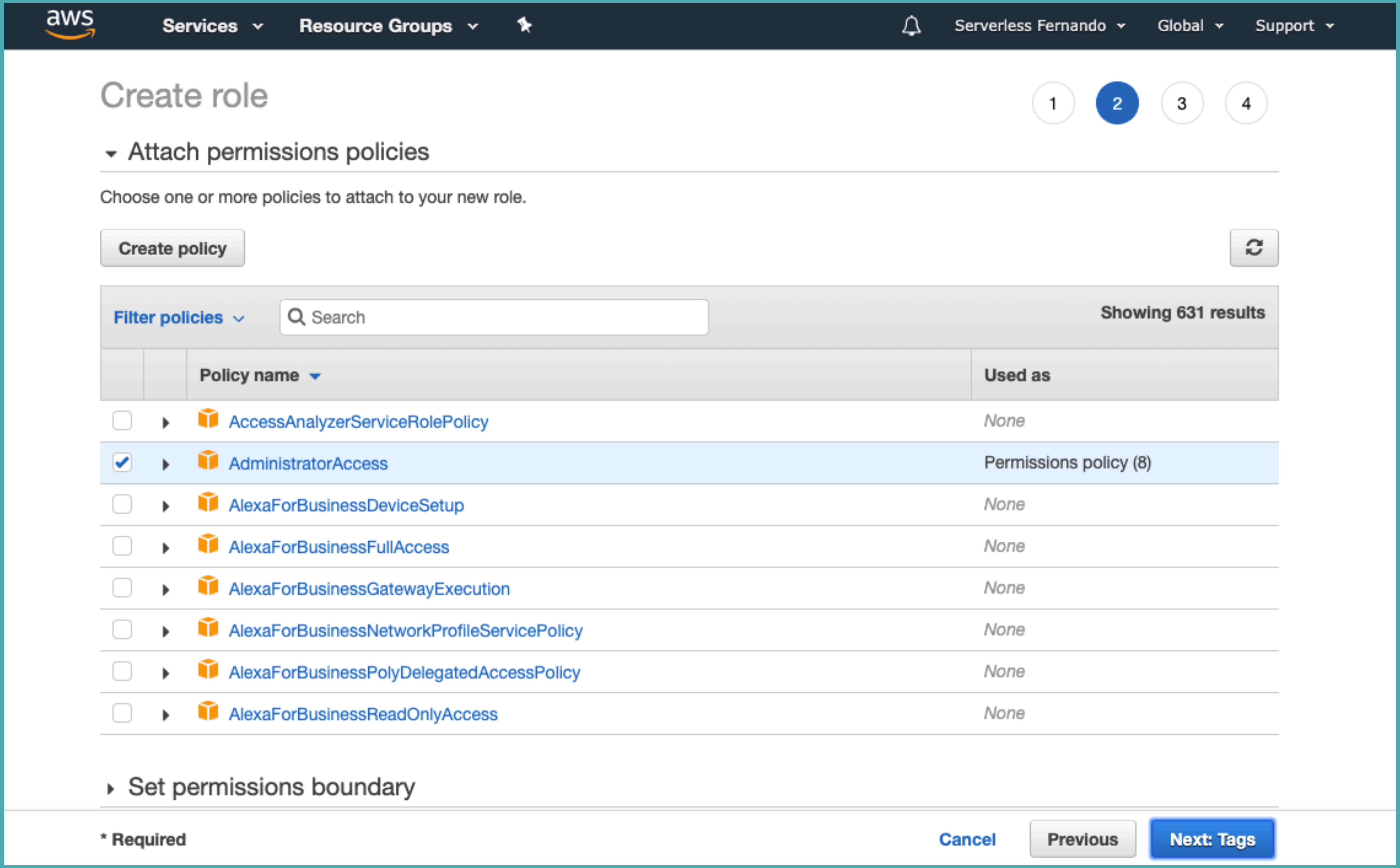
Account ID* 802587217904

Options [✓] Require external ID (Best practice when a third party will assume this role)

You can increase the security of your role by requiring an optional external identifier, which prevents "confused deputy" attacks. This is recommended if you do not own or have administrative access to the account that can assume this role. The external ID can include any characters that you choose. To assume this role, users must be in the trusted account and provide this exact external ID. Learn more

External ID

ServerlessEnterprise-y6PR2



Create role

1 2 3 4

Attach permissions policies

Choose one or more policies to attach to your new role.

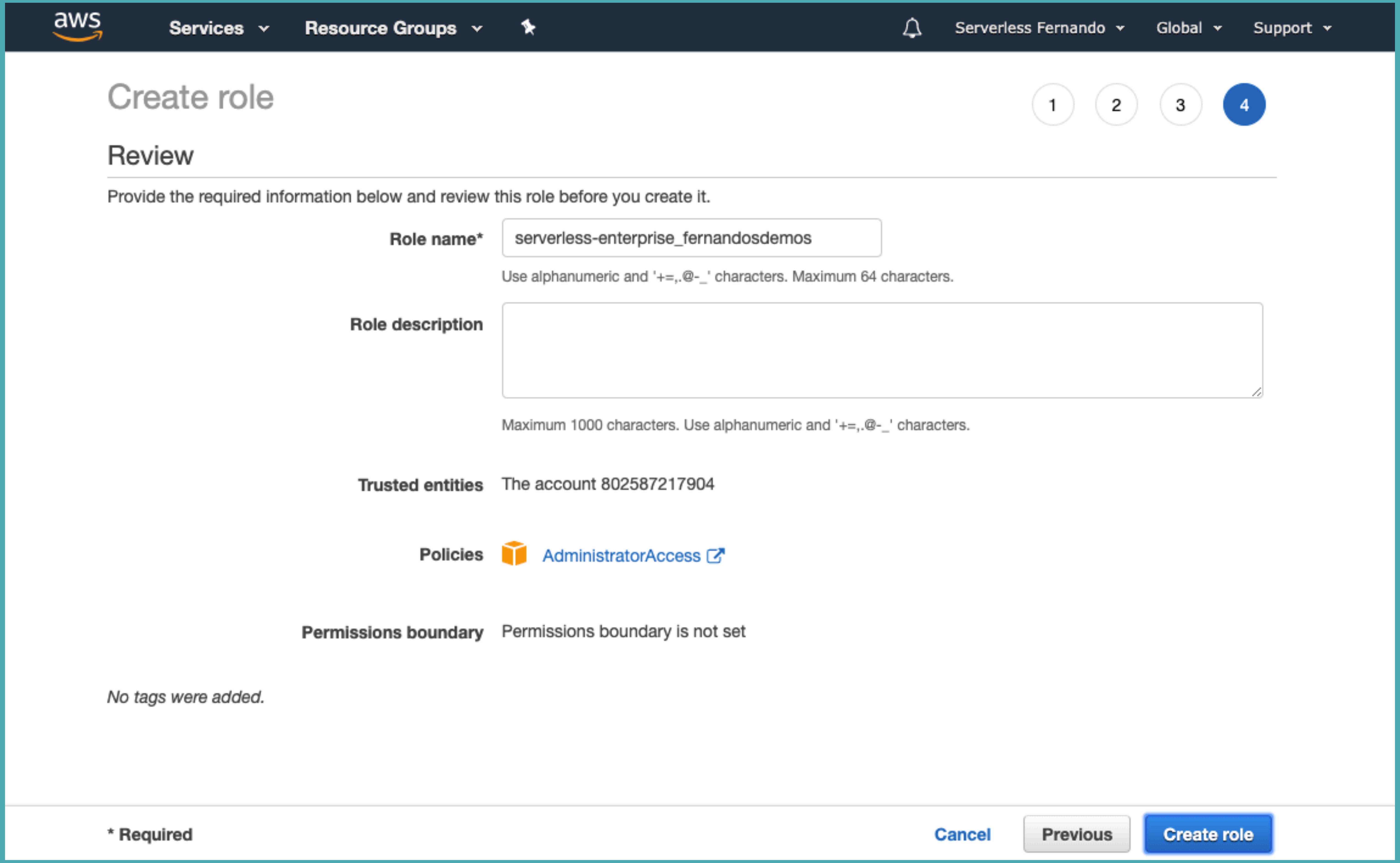
Create policy

Filter policies Search Showing 631 results

	Policy name	Used as
☐	AccessAnalyzerServiceRolePolicy	None
✓	AdministratorAccess	Permissions policy (8)
☐	AlexaForBusinessDeviceSetup	None
☐	AlexaForBusinessFullAccess	None
☐	AlexaForBusinessGatewayExecution	None
☐	AlexaForBusinessNetworkProfileServicePolicy	None
☐	AlexaForBusinessPolyDelegatedAccessPolicy	None
☐	AlexaForBusinessReadOnlyAccess	None

Set permissions boundary

* Required Cancel Previous Next: Tags



Create role

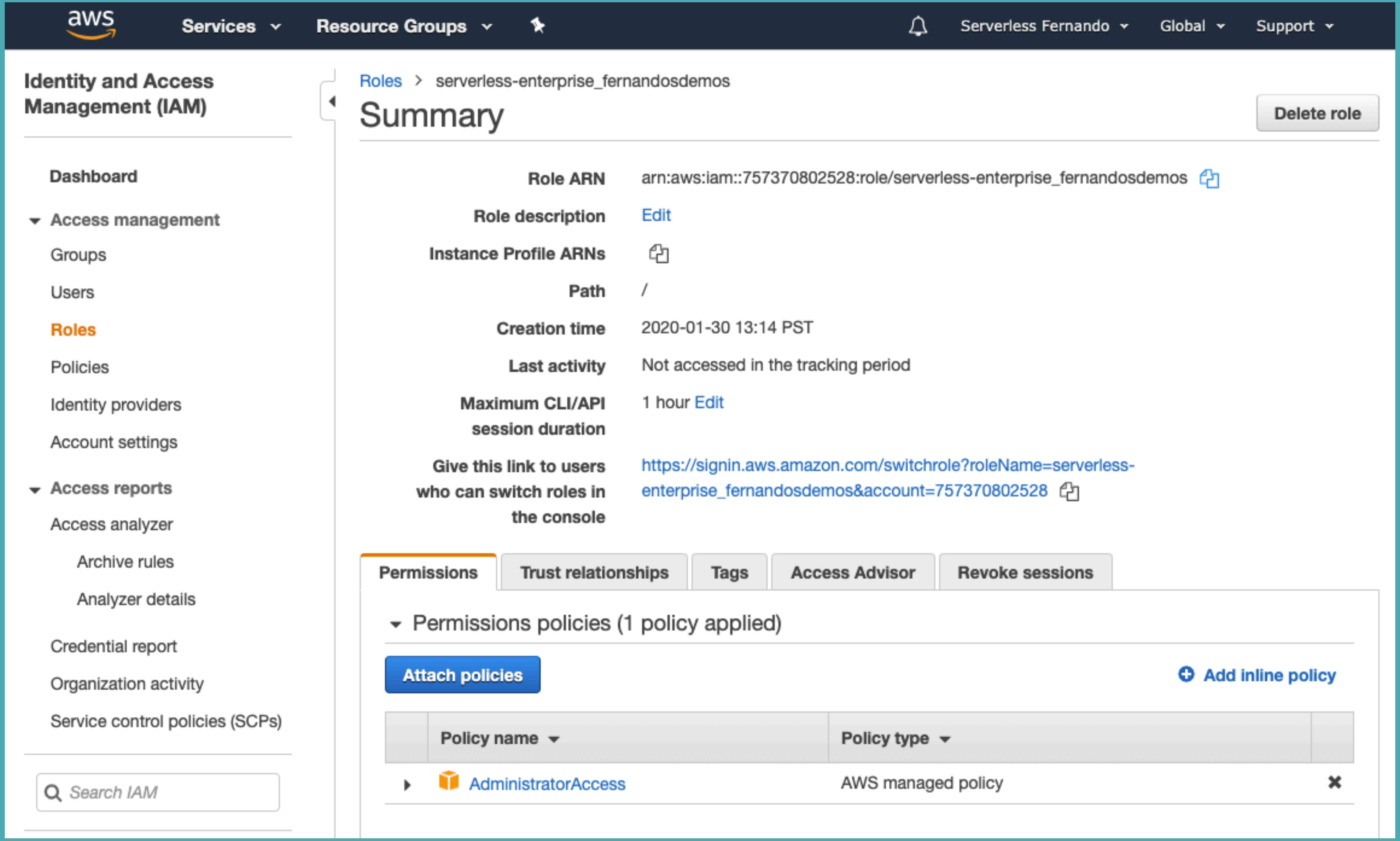
Review

Provide the required information below and review this role before you create it.

Role name*	serverless-enterprise_fernandosdemos
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	The account 802587217904
Policies	AdministratorAccess
Permissions boundary	Permissions boundary is not set

No tags were added.

* **Required**

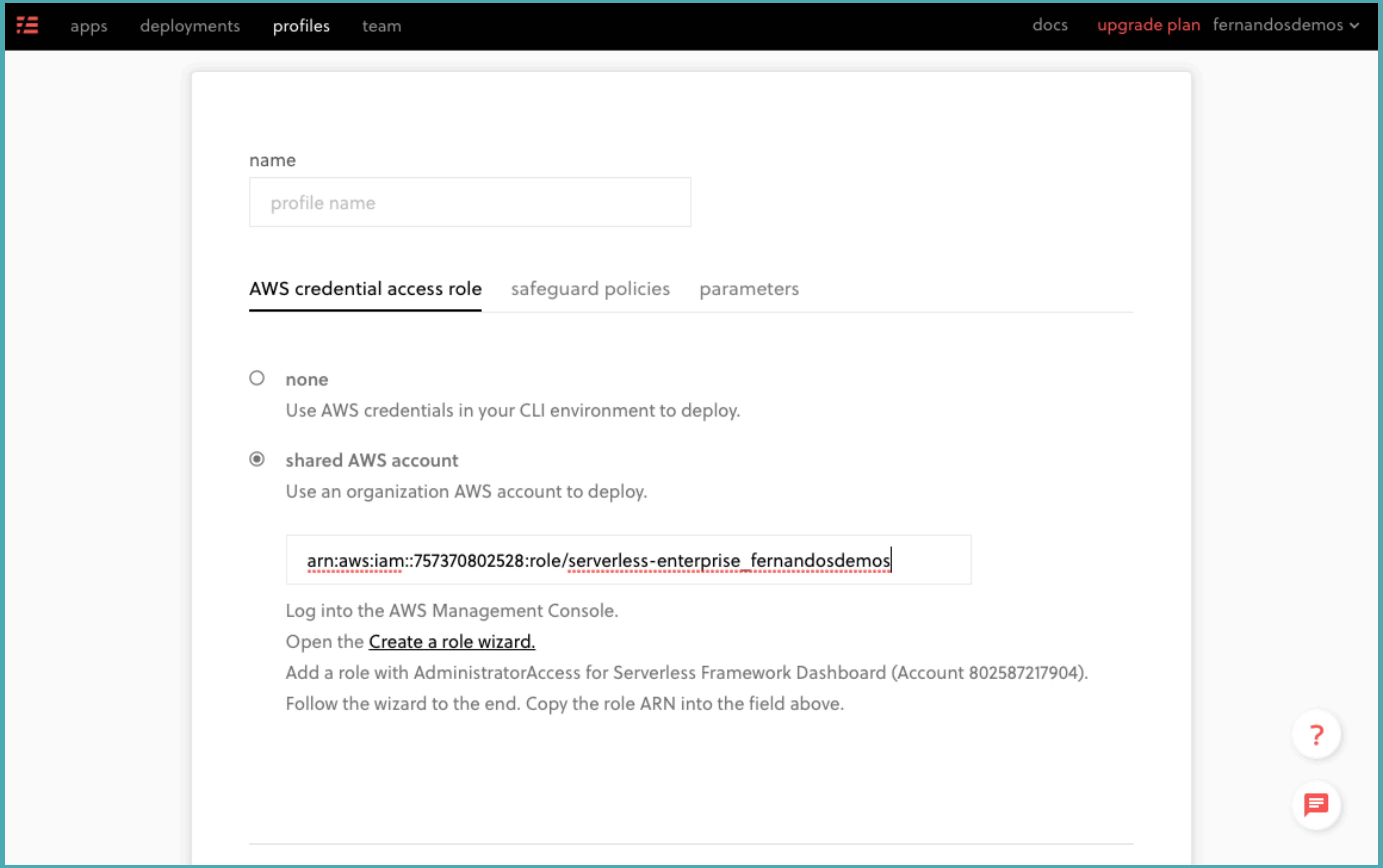


Summary

Role ARN	arn:aws:iam::757370802528:role/serverless-enterprise_fernandosdemos
Role description	Edit
Instance Profile ARNs	
Path	/
Creation time	2020-01-30 13:14 PST
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour Edit
Give this link to users who can switch roles in the console	https://signin.aws.amazon.com/switchrole?roleName=serverless-enterprise_fernandosdemos&account=757370802528

Permissions policies (1 policy applied)

Policy name	Policy type
AdministratorAccess	AWS managed policy



name

profile name

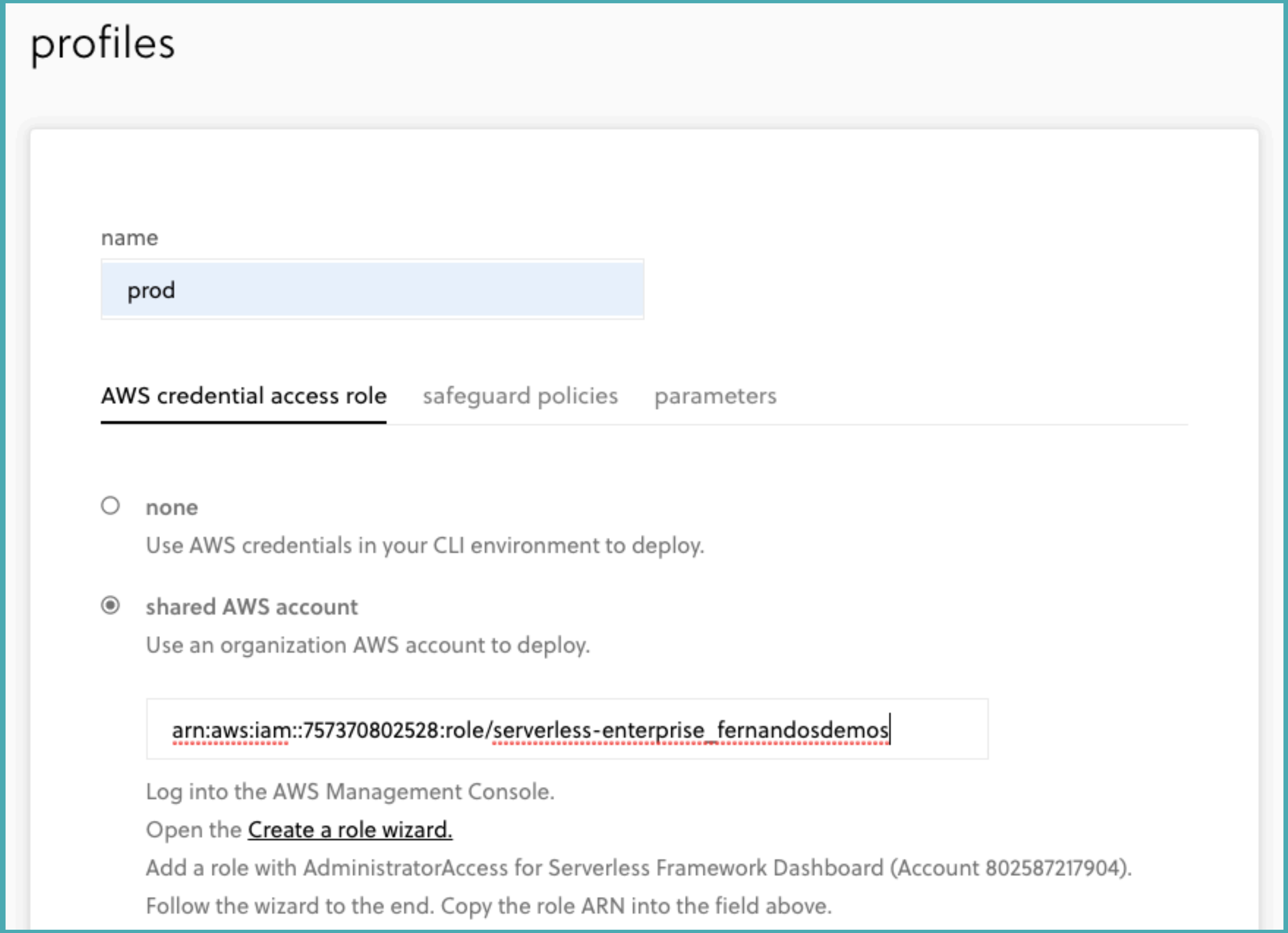
AWS credential access role | safeguard policies | parameters

( ) none
Use AWS credentials in your CLI environment to deploy.

(●) shared AWS account
Use an organization AWS account to deploy.

arn:aws:iam::757370802528:role/serverless-enterprise_fernandosdemos

Log into the AWS Management Console.
Open the Create a role wizard.
Add a role with AdministratorAccess for Serverless Framework Dashboard (Account 802587217904).
Follow the wizard to the end. Copy the role ARN into the field above.



profiles

name

prod

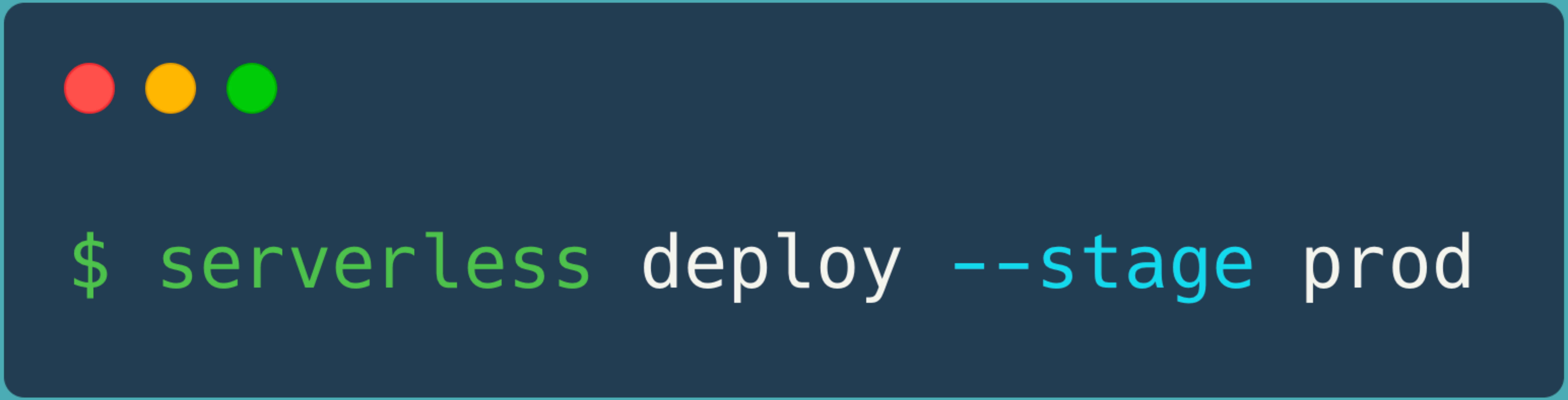
AWS credential access role | safeguard policies | parameters

○ none
Use AWS credentials in your CLI environment to deploy.

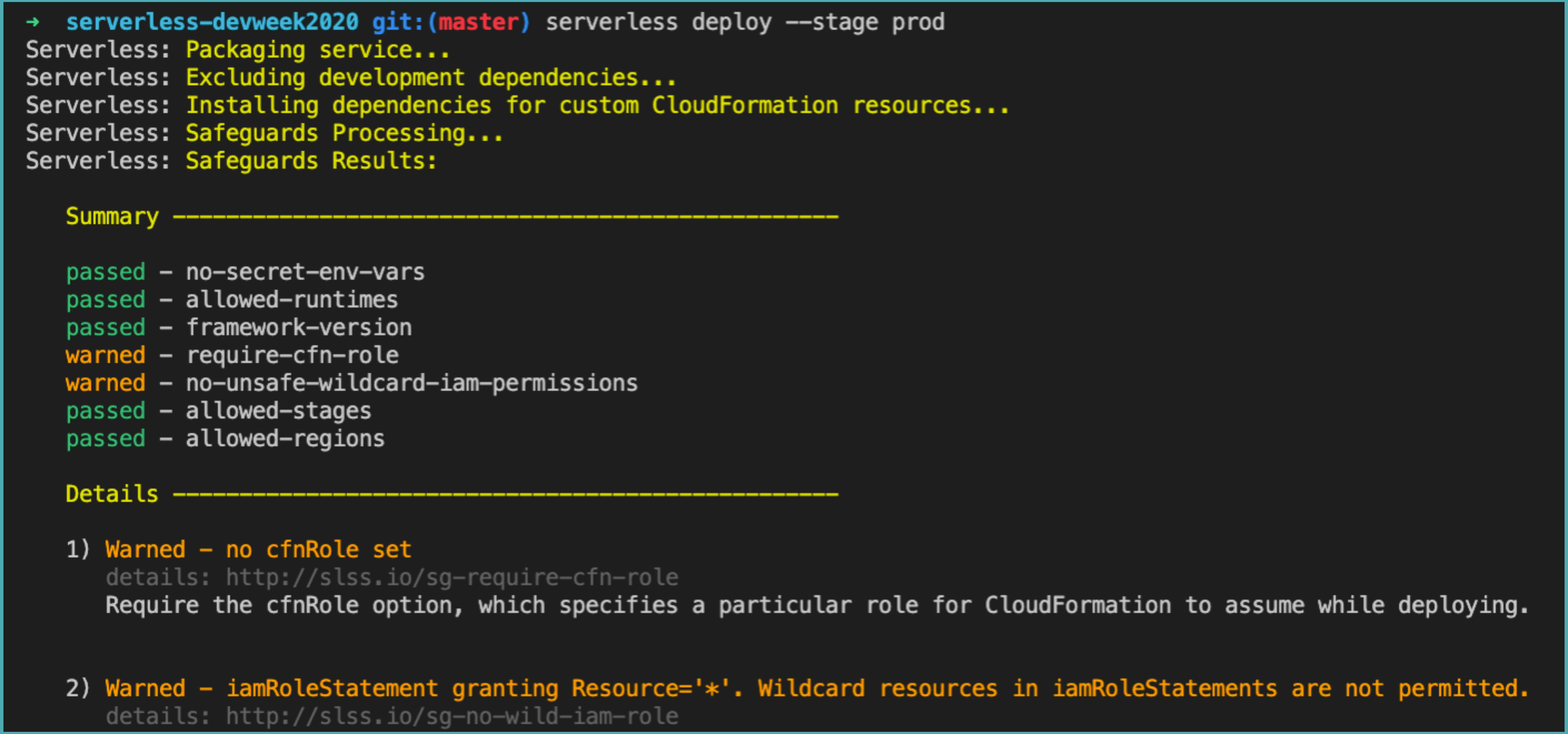
● shared AWS account
Use an organization AWS account to deploy.

arn:aws:iam::757370802528:role/serverless-enterprise_fernandosdemos

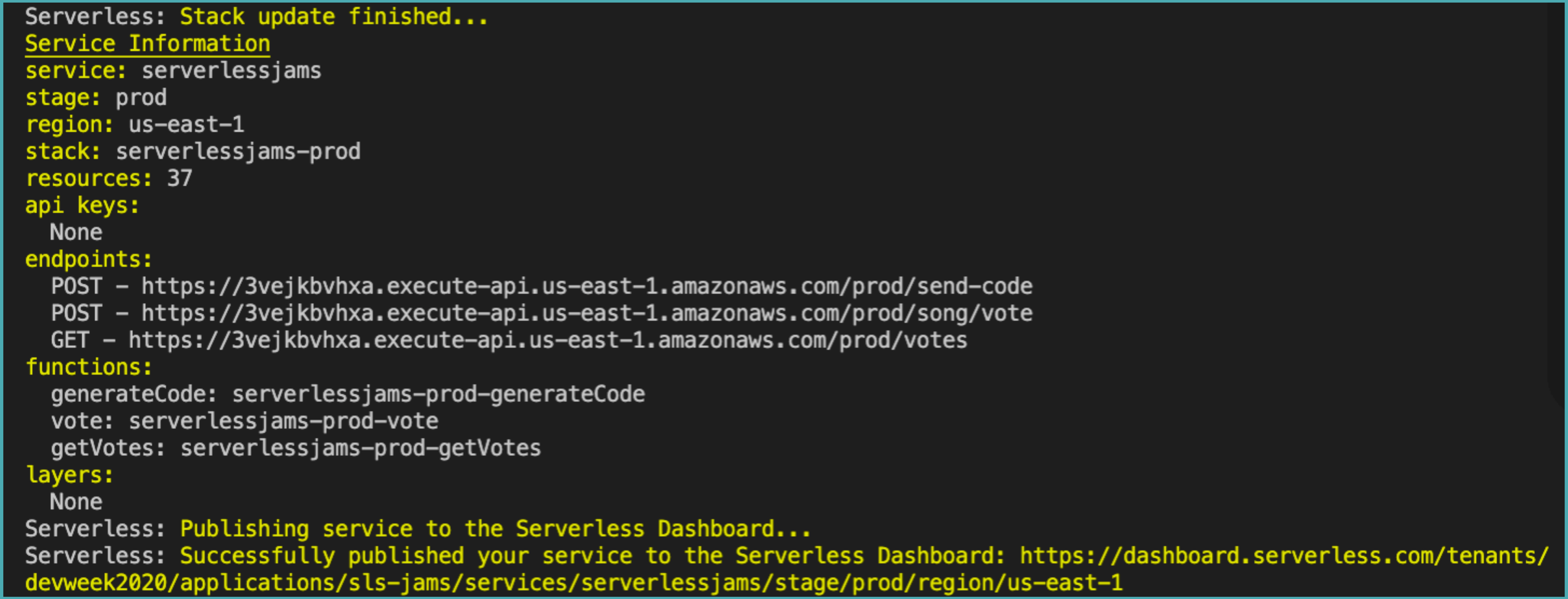
Log into the AWS Management Console.
Open the Create a role wizard.
Add a role with AdministratorAccess for Serverless Framework Dashboard (Account 802587217904).
Follow the wizard to the end. Copy the role ARN into the field above.



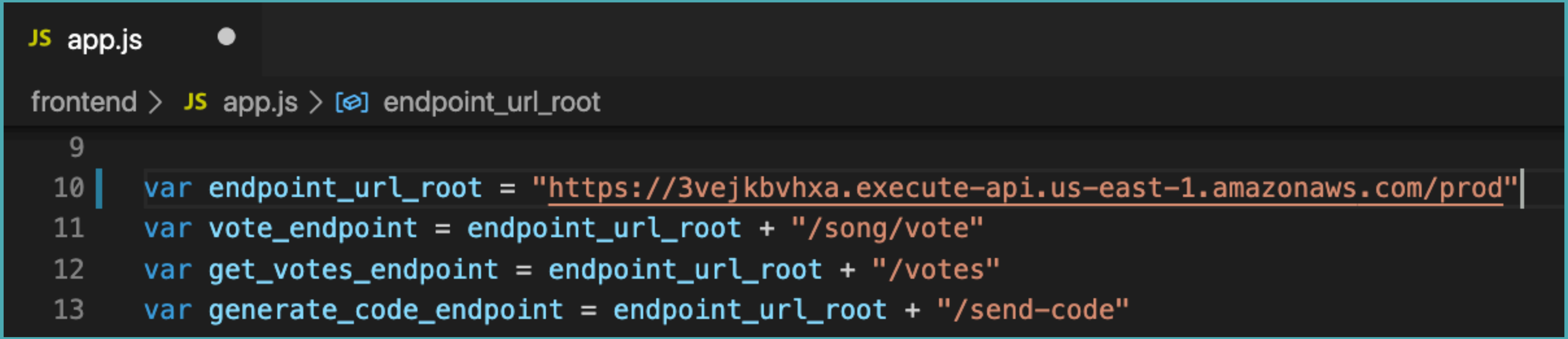
```
$ serverless deploy --stage prod
```



```
➜  serverless-devweek2020 git:(master) serverless deploy --stage prod
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Installing dependencies for custom CloudFormation resources...
Serverless: Safeguards Processing...
Serverless: Safeguards Results:

   Summary --------------------------------------------------

   passed - no-secret-env-vars
   passed - allowed-runtimes
   passed - framework-version
   warned - require-cfn-role
   warned - no-unsafe-wildcard-iam-permissions
   passed - allowed-stages
   passed - allowed-regions

   Details --------------------------------------------------

   1) Warned - no cfnRole set
      details: http://slss.io/sg-require-cfn-role
      Require the cfnRole option, which specifies a particular role for CloudFormation to assume while deploying.


   2) Warned - iamRoleStatement granting Resource='*'. Wildcard resources in iamRoleStatements are not permitted.
      details: http://slss.io/sg-no-wild-iam-role
```



```
Serverless: Stack update finished...
Service Information
service: serverlessjams
stage: prod
region: us-east-1
stack: serverlessjams-prod
resources: 37
api keys:
  None
endpoints:
  POST - https://3vejkbvhxa.execute-api.us-east-1.amazonaws.com/prod/send-code
  POST - https://3vejkbvhxa.execute-api.us-east-1.amazonaws.com/prod/song/vote
  GET - https://3vejkbvhxa.execute-api.us-east-1.amazonaws.com/prod/votes
functions:
  generateCode: serverlessjams-prod-generateCode
  vote: serverlessjams-prod-vote
  getVotes: serverlessjams-prod-getVotes
layers:
  None
Serverless: Publishing service to the Serverless Dashboard...
Serverless: Successfully published your service to the Serverless Dashboard: https://dashboard.serverless.com/tenants/
devweek2020/applications/sls-jams/services/serverlessjams/stage/prod/region/us-east-1
```

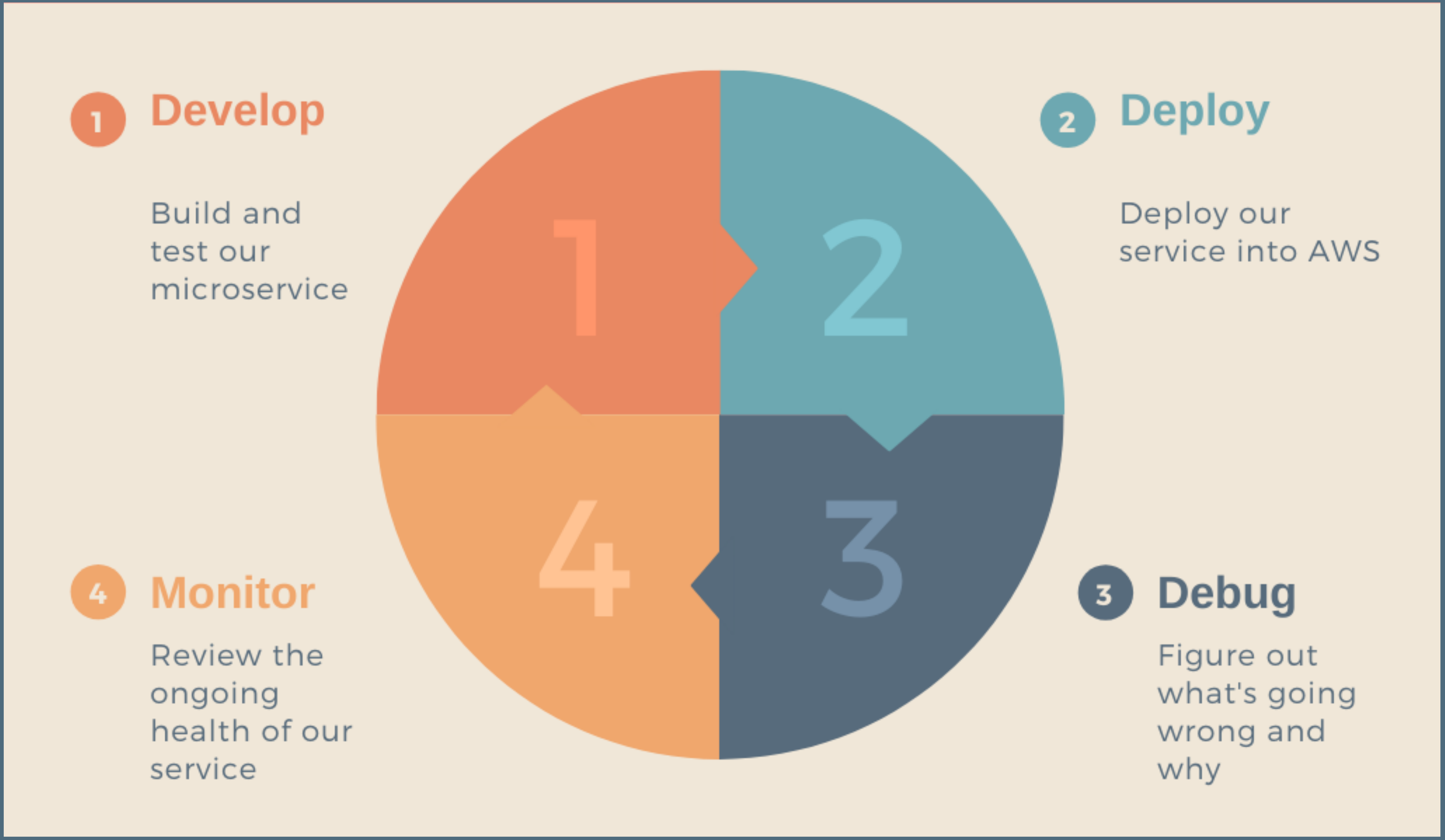


app.js

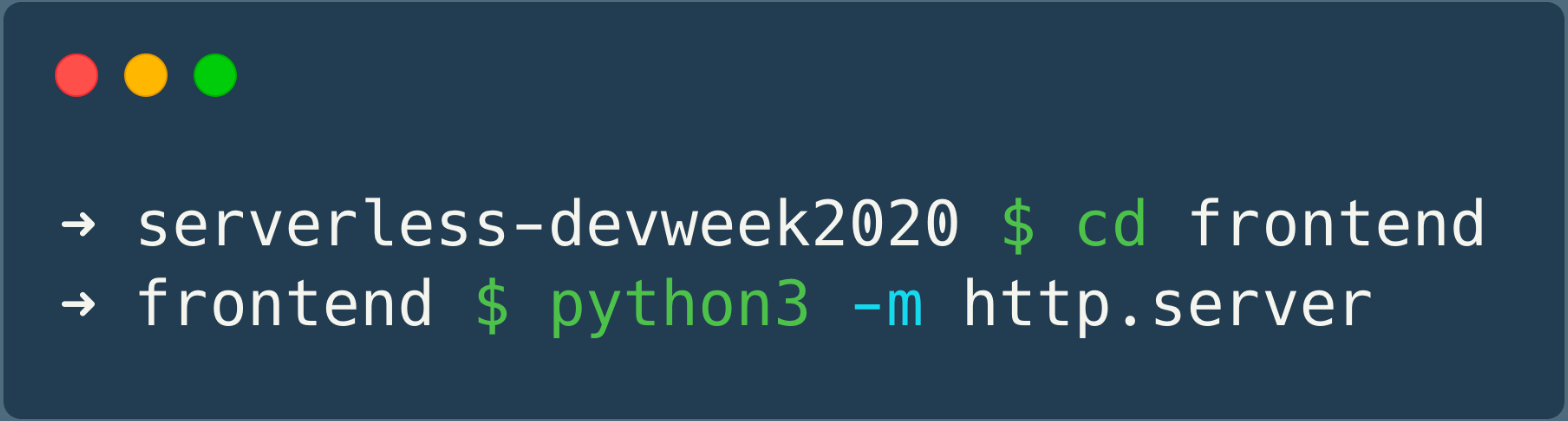
frontend > app.js > endpoint_url_root

```
var endpoint_url_root = "https://3vejkbvhxa.execute-api.us-east-1.amazonaws.com/prod"
var vote_endpoint = endpoint_url_root + "/song/vote"
var get_votes_endpoint = endpoint_url_root + "/votes"
var generate_code_endpoint = endpoint_url_root + "/send-code"
```

Debug

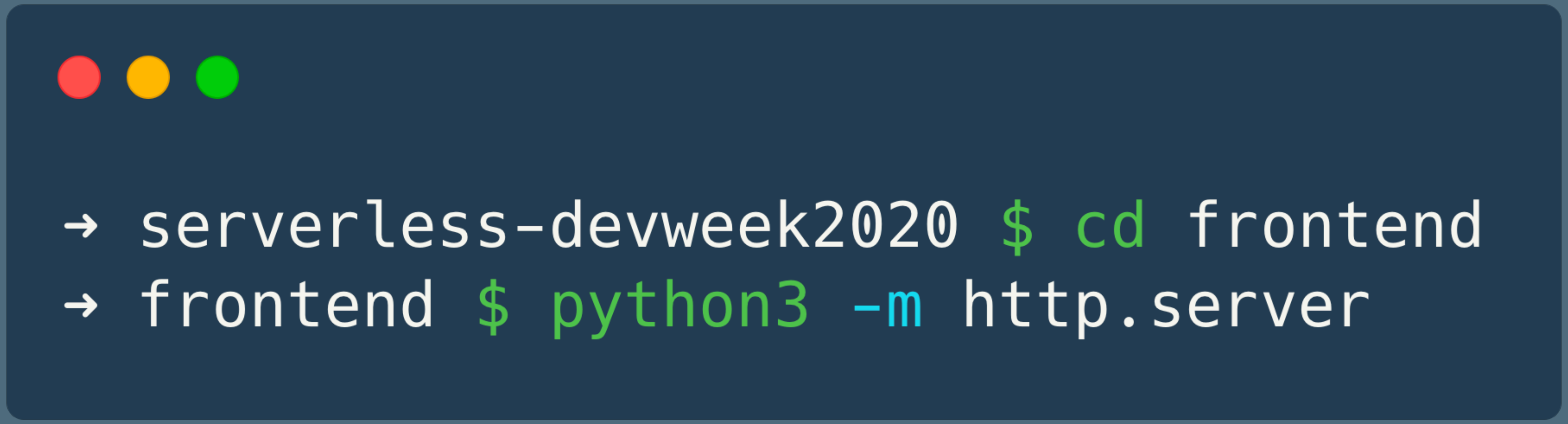

1 Develop
Build and test our microservice
2 Deploy
Deploy our service into AWS
3 Debug
Figure out what's going wrong and why
4 Monitor
Review the ongoing health of our service
1
2
3
4

Running Our Frontend



```
→ serverless-devweek2020 $ cd frontend
→ frontend $ python3 -m http.server
```

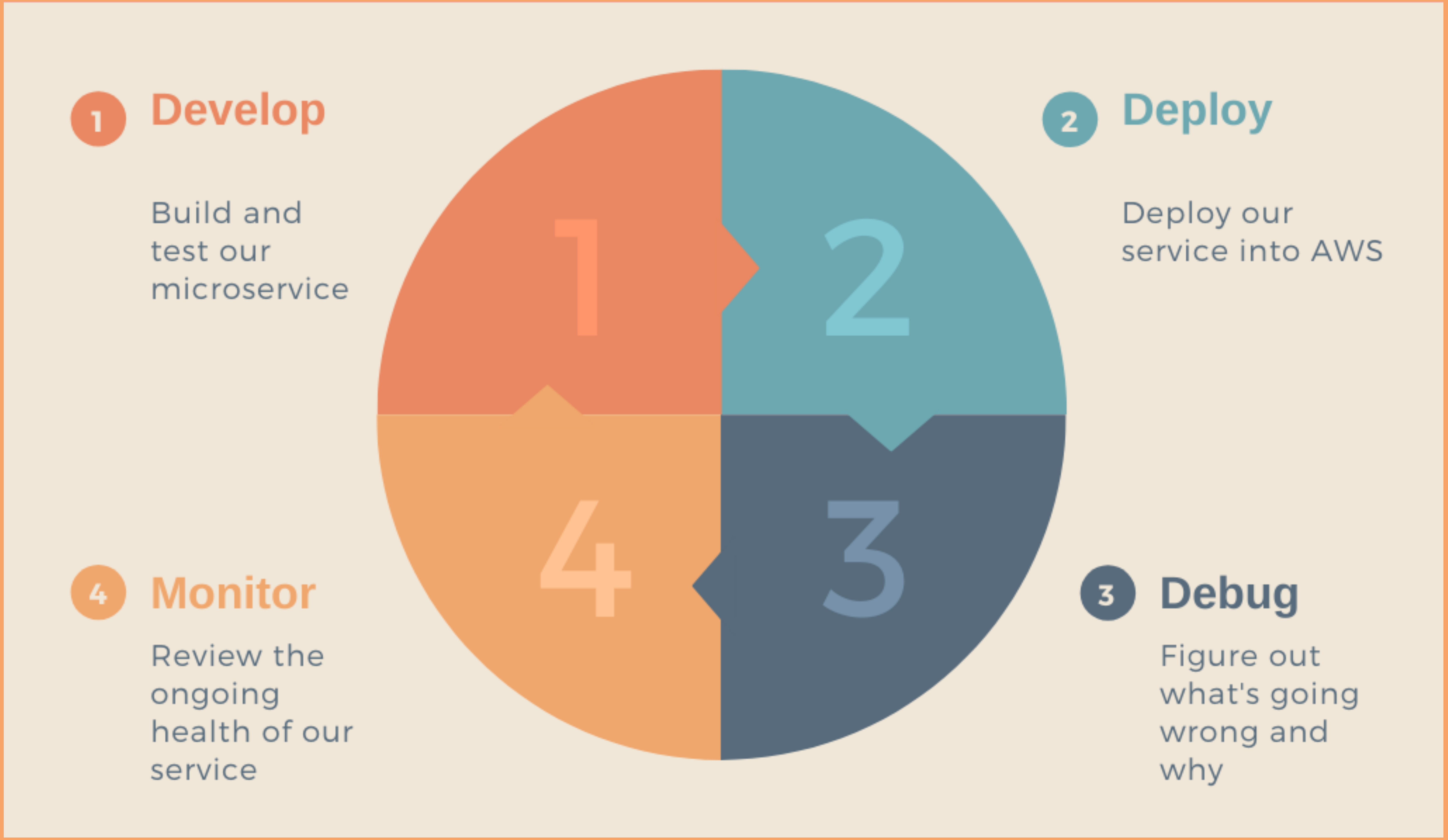
DEMO TIME!



```
➜ serverless-devweek2020 $ cd frontend
➜ frontend $ python3 -m http.server
```

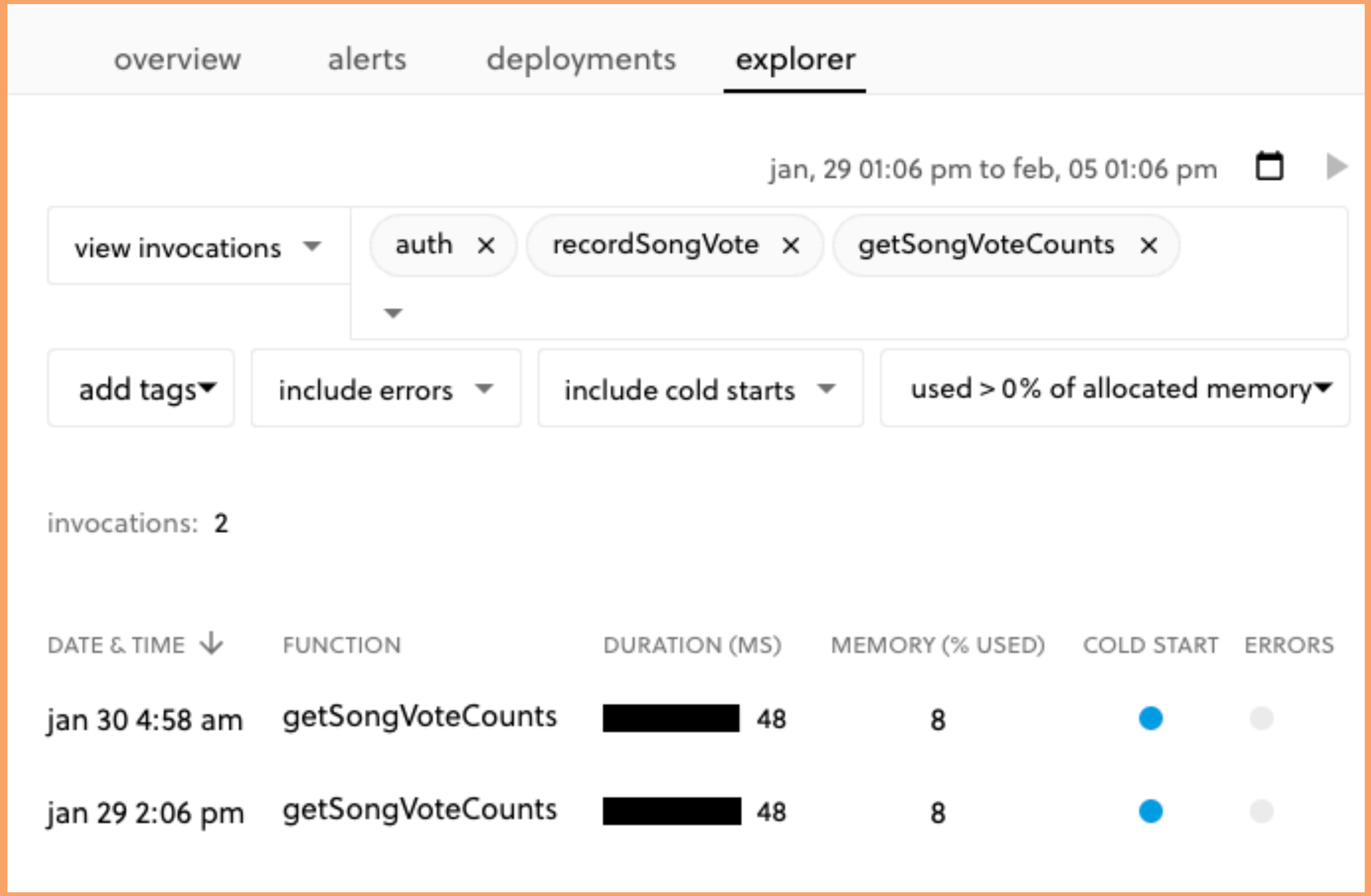
1. Run our frontend
2. Try to vote
3. Try to vote again!

Monitoring


1 Develop
Build and test our microservice
2 Deploy
Deploy our service into AWS
3 Debug
Figure out what's going wrong and why
4 Monitor
Review the ongoing health of our service
1
2
3
4

DEMO TIME!

1. Explore function logs
2. Review specific invocations



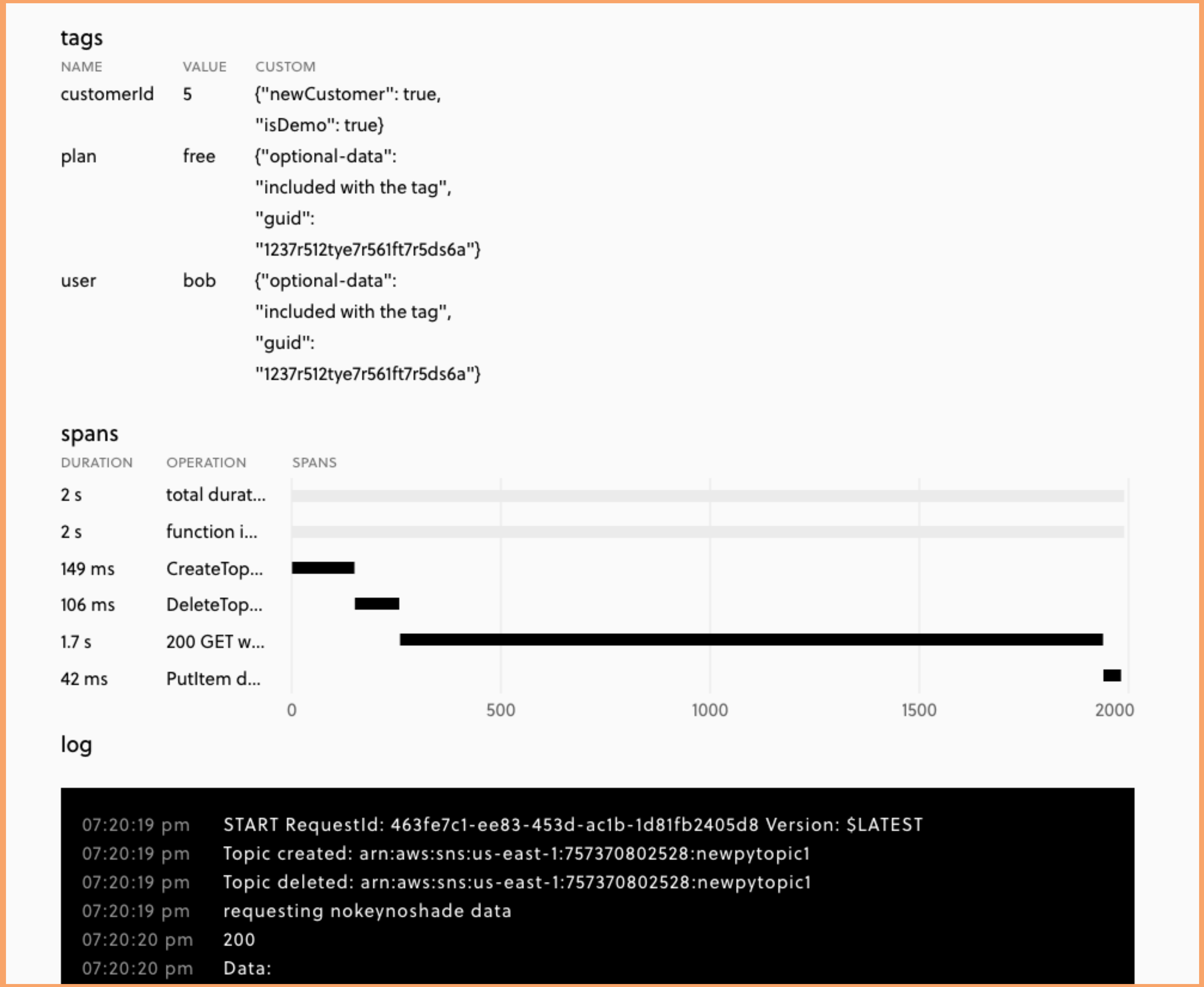
jan, 29 01:06 pm to feb, 05 01:06 pm

view invocations | auth × | recordSongVote × | getSongVoteCounts ×

add tags | include errors | include cold starts | used > 0% of allocated memory

invocations: 2

DATE & TIME	FUNCTION	DURATION (MS)	MEMORY (% USED)	COLD START	ERRORS
jan 30 4:58 am	getSongVoteCounts	48	8	●	○
jan 29 2:06 pm	getSongVoteCounts	48	8	●	○



tags

NAME	VALUE	CUSTOM
customerId	5	{"newCustomer": true, "isDemo": true}
plan	free	{"optional-data": "included with the tag", "guid": "1237r512tye7r561ft7r5ds6a"}
user	bob	{"optional-data": "included with the tag", "guid": "1237r512tye7r561ft7r5ds6a"}

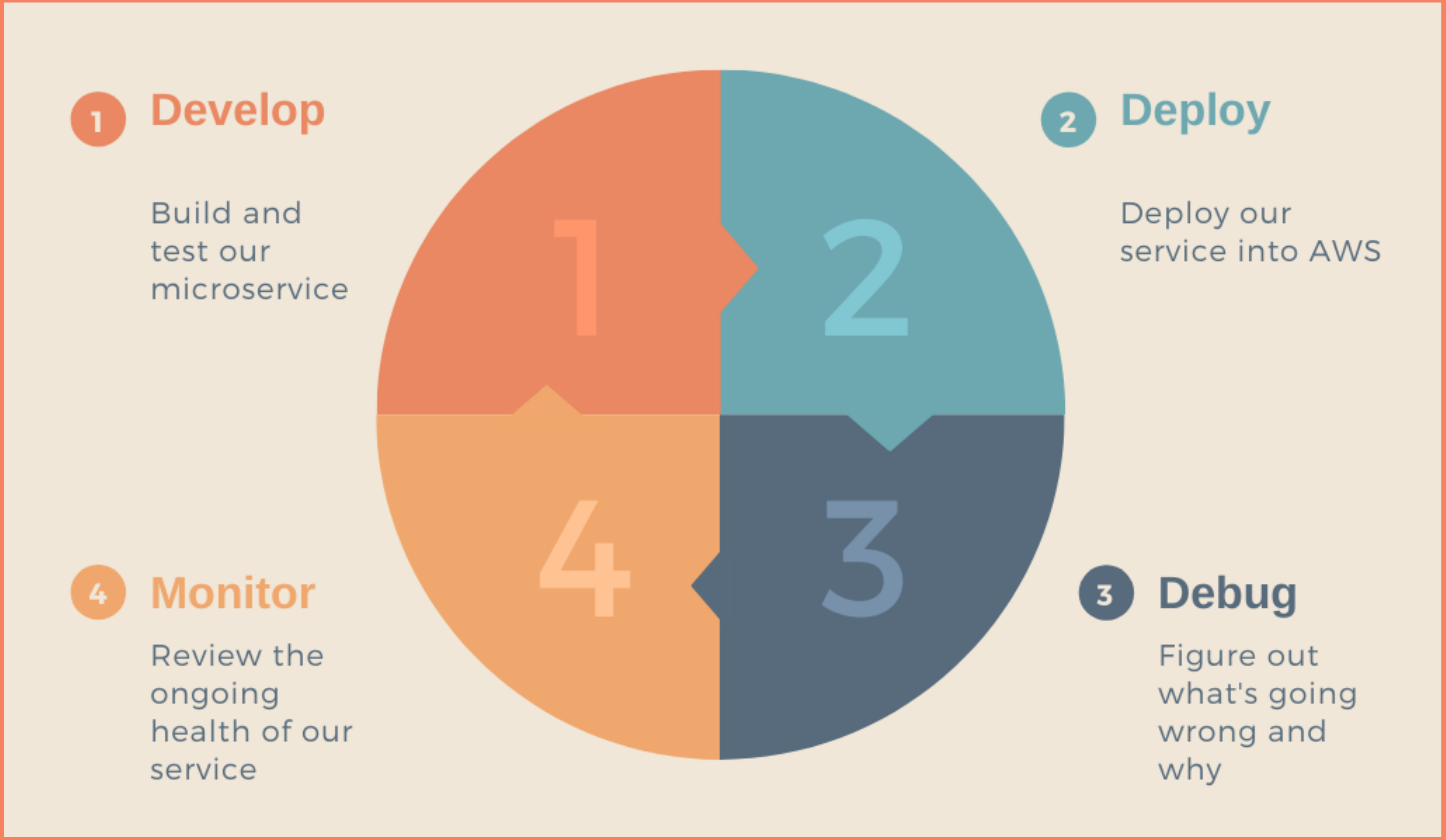
spans

DURATION	OPERATION
2 s	total durat...
2 s	function i...
149 ms	CreateTop...
106 ms	DeleteTop...
1.7 s	200 GET w...
42 ms	PutItem d...

log

```
07:20:19 pm   START RequestId: 463fe7c1-ee83-453d-ac1b-1d81fb2405d8 Version: $LATEST
07:20:19 pm   Topic created: arn:aws:sns:us-east-1:757370802528:newpytopic1
07:20:19 pm   Topic deleted: arn:aws:sns:us-east-1:757370802528:newpytopic1
07:20:19 pm   requesting nokeynoshade data
07:20:20 pm   200
07:20:20 pm   Data:
```

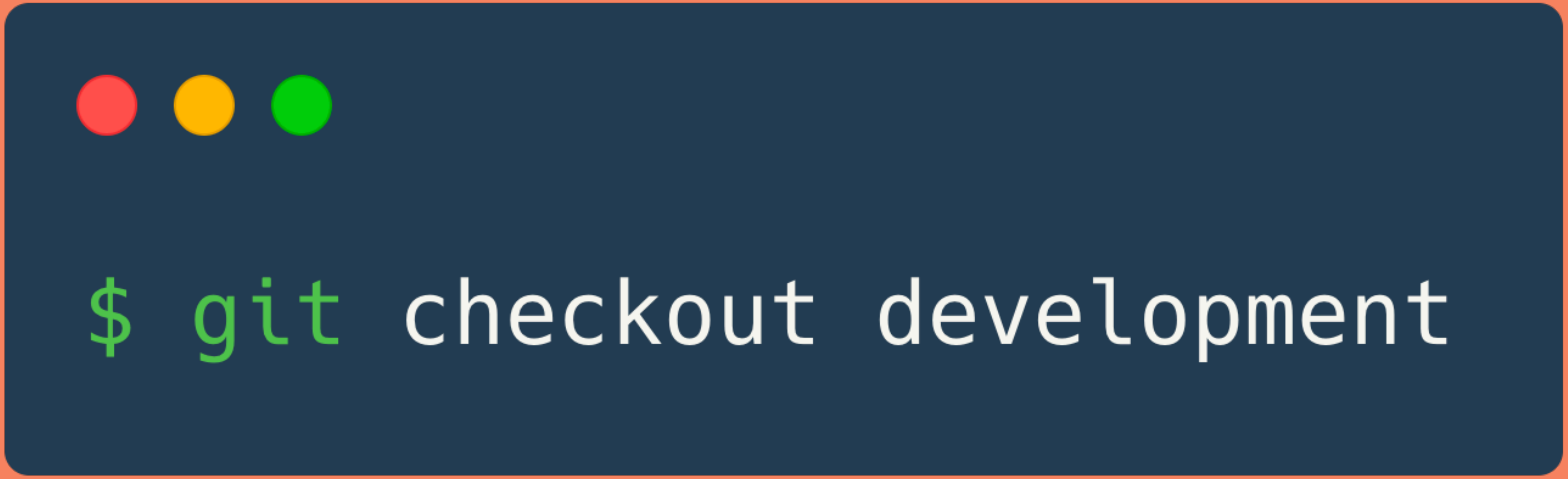
Develop


1 Develop
Build and test our microservice
2 Deploy
Deploy our service into AWS
3 Debug
Figure out what's going wrong and why
4 Monitor
Review the ongoing health of our service
1
2
3
4

New Requirements:

Users may vote every five minutes

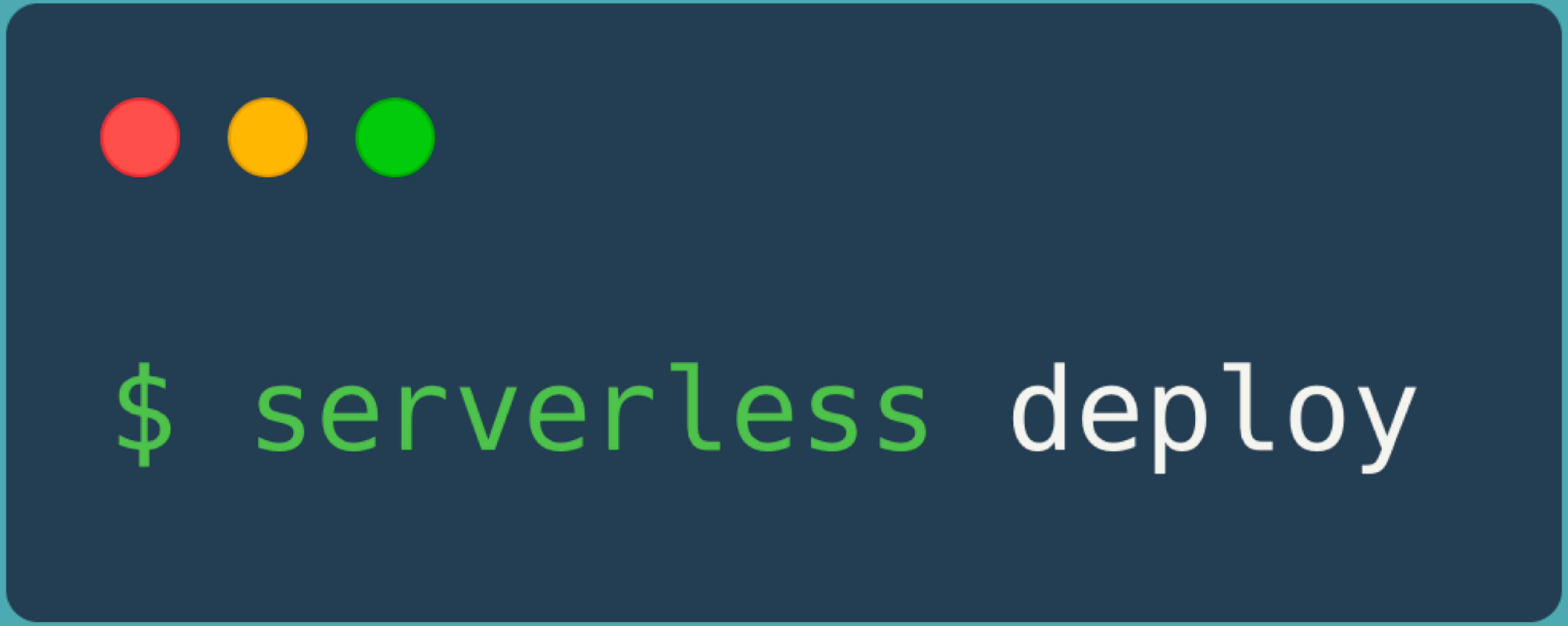
Let’s “Build” that



```
$ git checkout development
```

DEMO TIME!

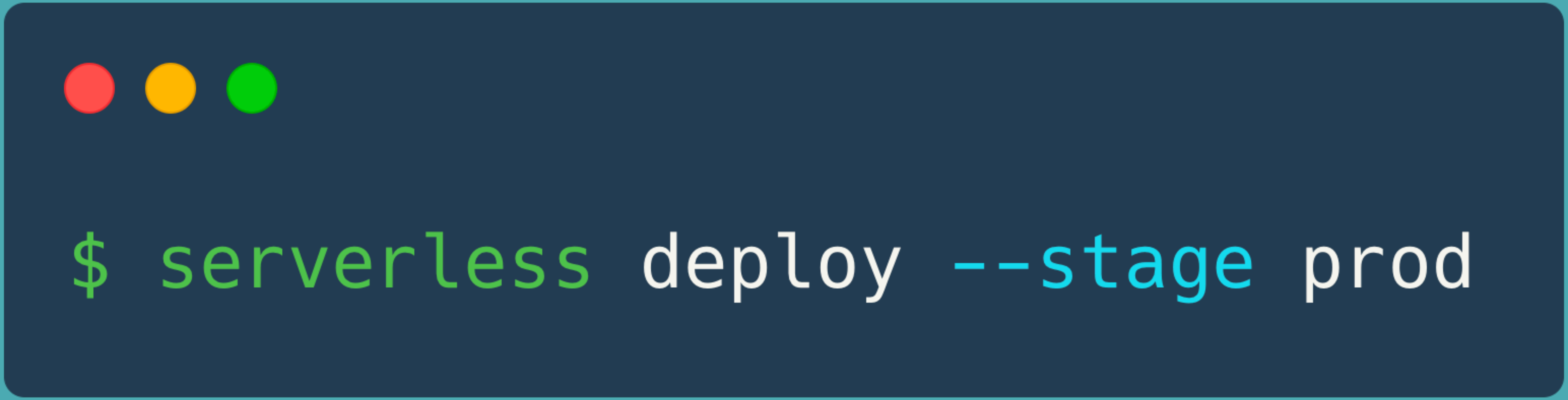
Deploy to `dev`



```
$ serverless deploy
```

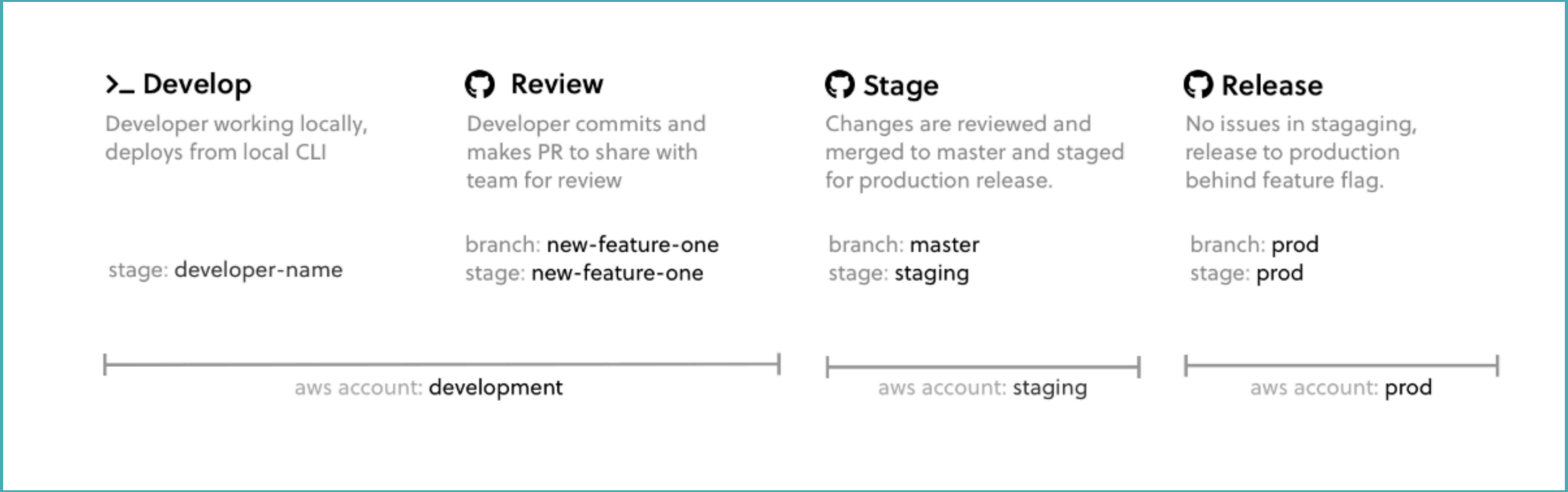
DEMO TIME!

How would we get this to prod?



```
$ serverless deploy --stage prod
```

Reality is more complicated…



>_ Develop
Developer working locally, deploys from local CLI

stage: **developer-name**

Review
Developer commits and makes PR to share with team for review

branch: **new-feature-one**
stage: **new-feature-one**

Stage
Changes are reviewed and merged to master and staged for production release.

branch: **master**
stage: **staging**

Release
No issues in stagaging, release to production behind feature flag.

branch: **prod**
stage: **prod**

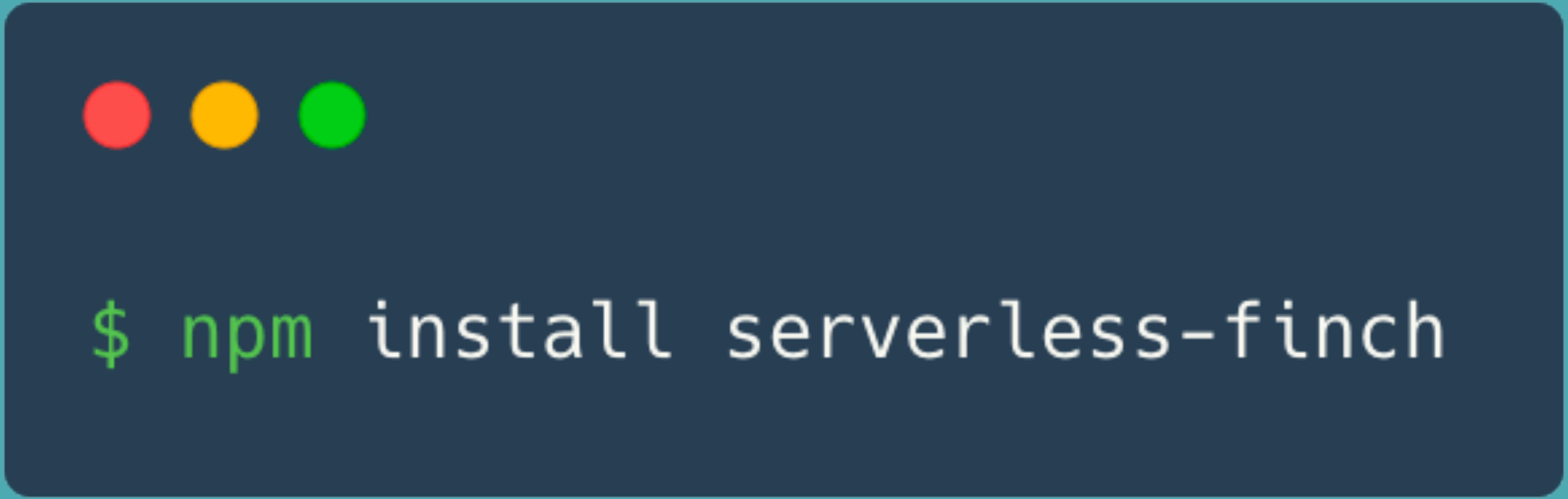
aws account: **development**

aws account: **staging**

aws account: **prod**

https://serverless.com/learn/guides/cicd/

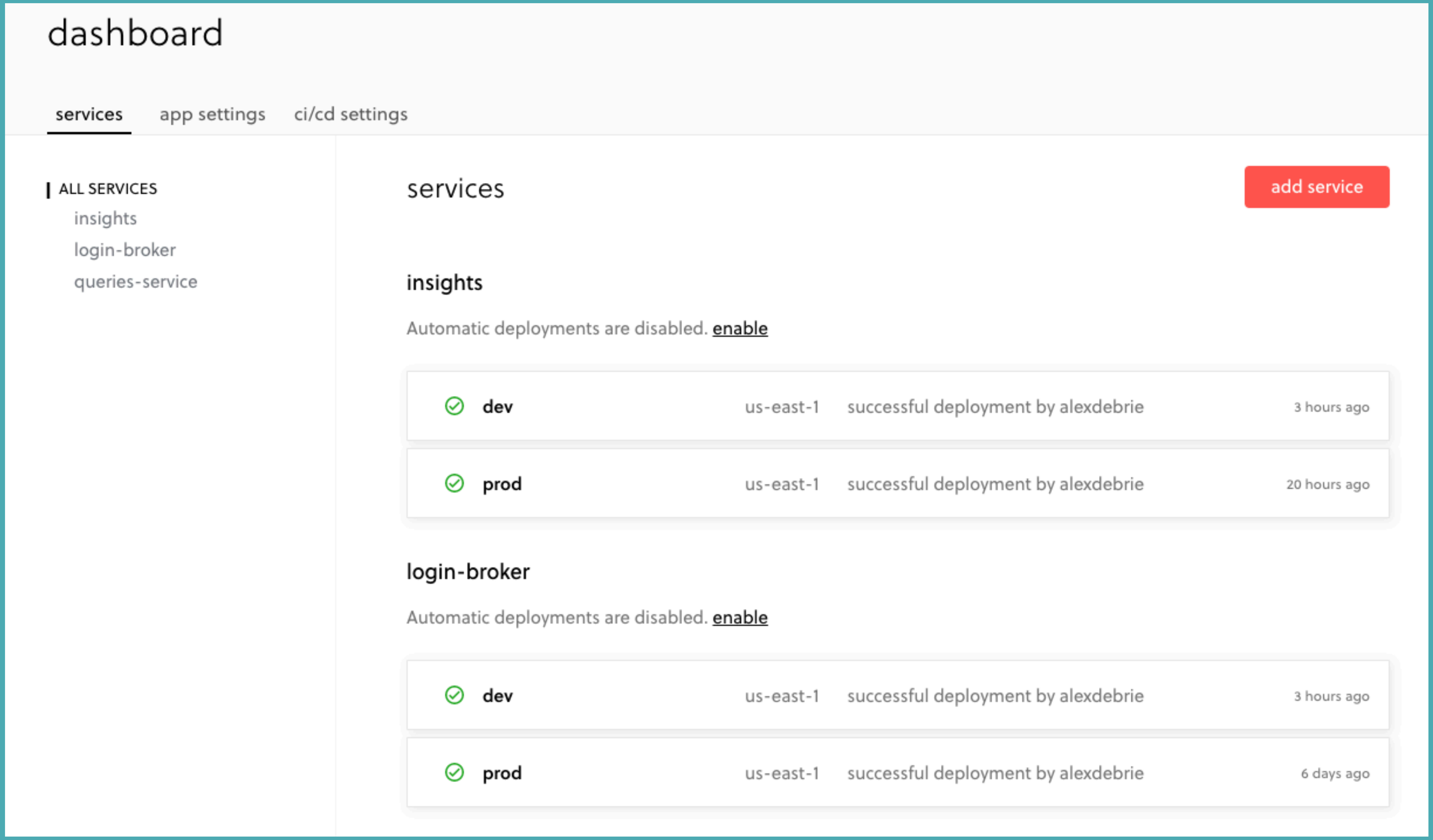
BONUS DEMO TIME?



```
$ npm install serverless-finch
```

Deploying Our Frontend

We eat our own dog food



dashboard

services | app settings | ci/cd settings

ALL SERVICES
insights
login-broker
queries-service

services

add service

insights

Automatic deployments are disabled. enable

✓ dev	us-east-1	successful deployment by alexdebrie	3 hours ago
✓ prod	us-east-1	successful deployment by alexdebrie	20 hours ago

login-broker

Automatic deployments are disabled. enable

✓ dev	us-east-1	successful deployment by alexdebrie	3 hours ago
✓ prod	us-east-1	successful deployment by alexdebrie	6 days ago

Thank you!

Office Hours:
9am tomorrow - Room 212

Unofficial Office Hours:
Right now
By appointment - (DM me)

@fmc_sea