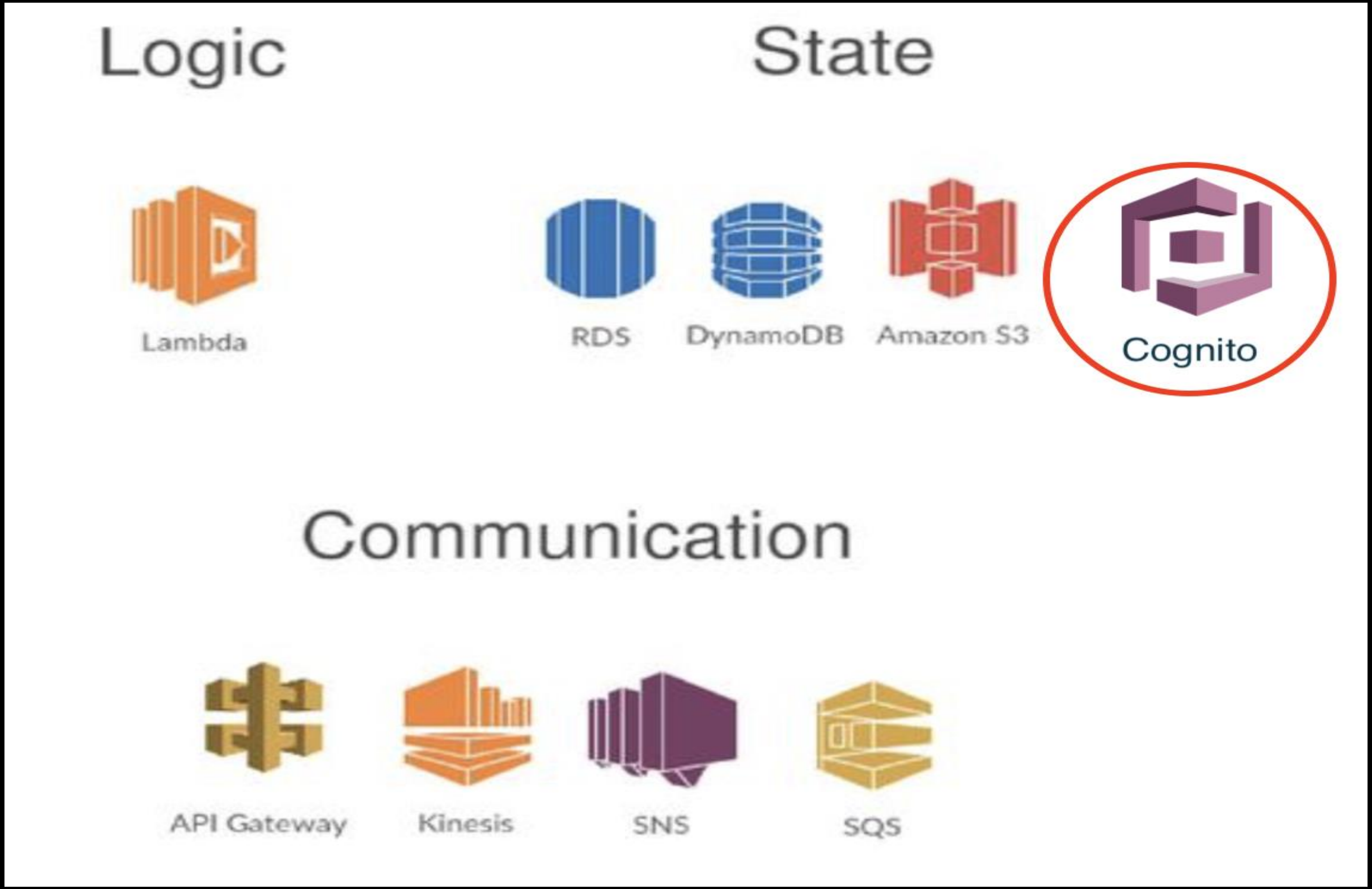
Amazon Cognito

Components of a Serverless app



Amazon Cognito

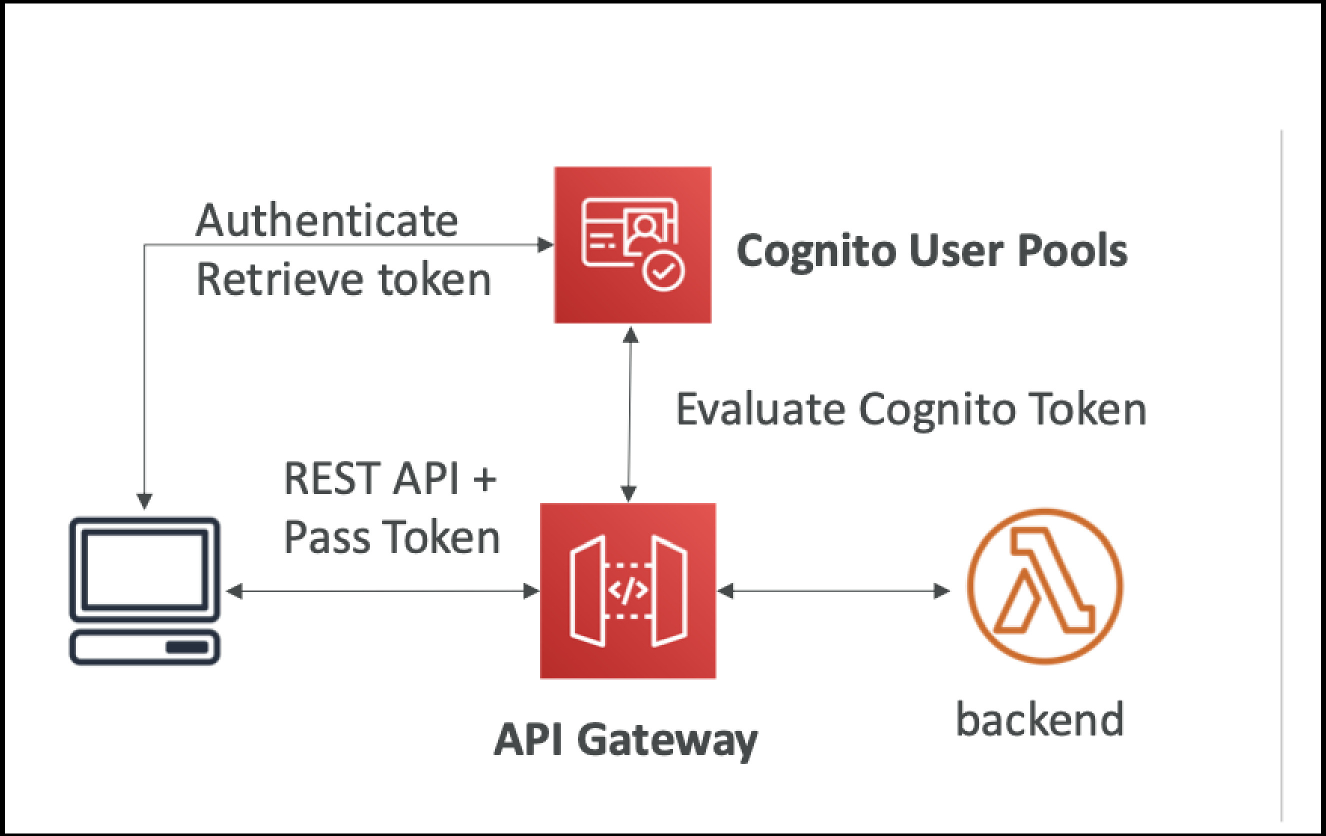
- We want to give users an identity so that they can interact with our application.
- Cognito User Pools:
  - Sign in functionality for app users.
  - Integrate with API Gateway & Application Load Balancer.
- Cognito Identity Pools (Federated Identity):
  - Provide AWS credentials to users so they can access AWS resources directly.
  - Integrate with Cognito User Pools as an identity provider.

- Cognito vs IAM: “hundreds of users”, ”mobile users”.

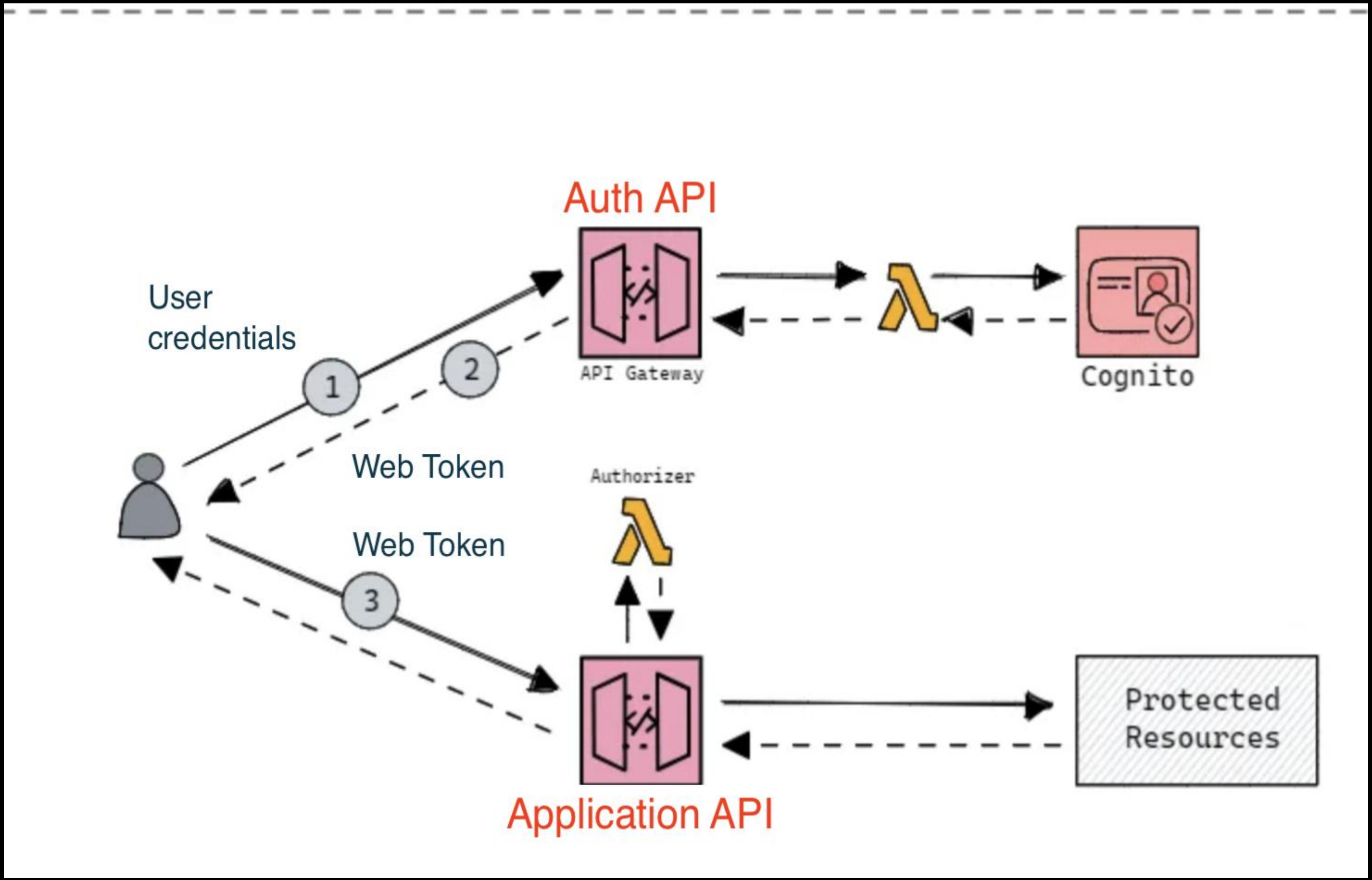
Cognito User Pool.

- Creates a serverless database of user for your web & mobile apps.
- Simple login: Username (or email) / password combination.
- Password reset.
- Email & Phone Number Verification.
- Multi-factor authentication (MFA).
- Federated Identities: Facebook, Google...
- Feature: block users if their credentials are compromised elsewhere
- Include JSON Web Token (JWT) in Login response.

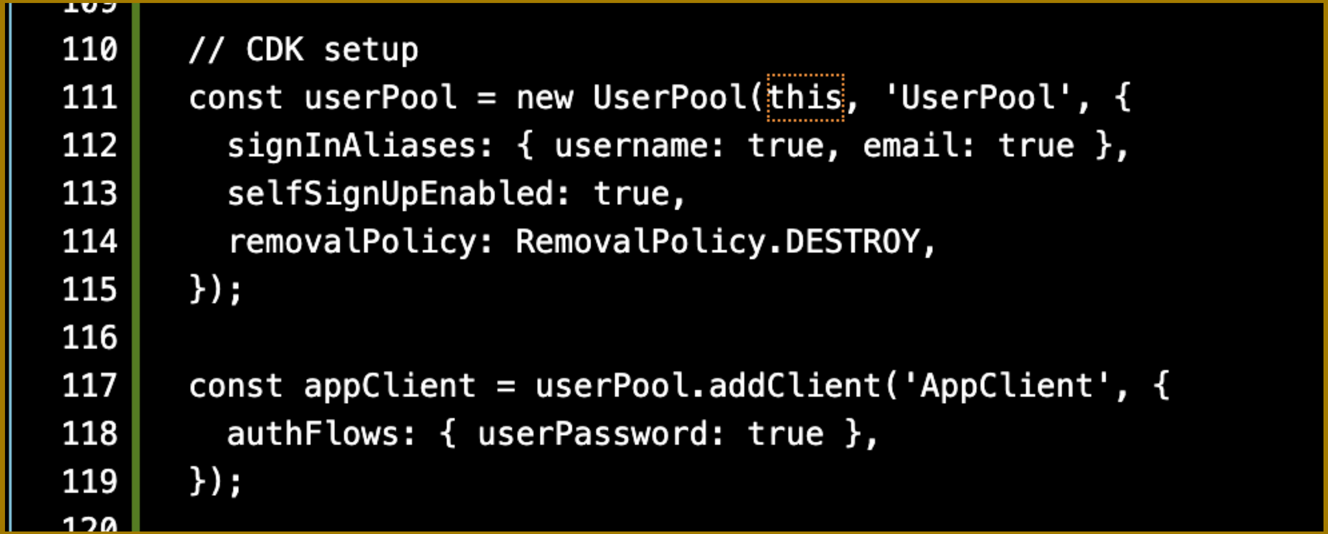
Cognito's role



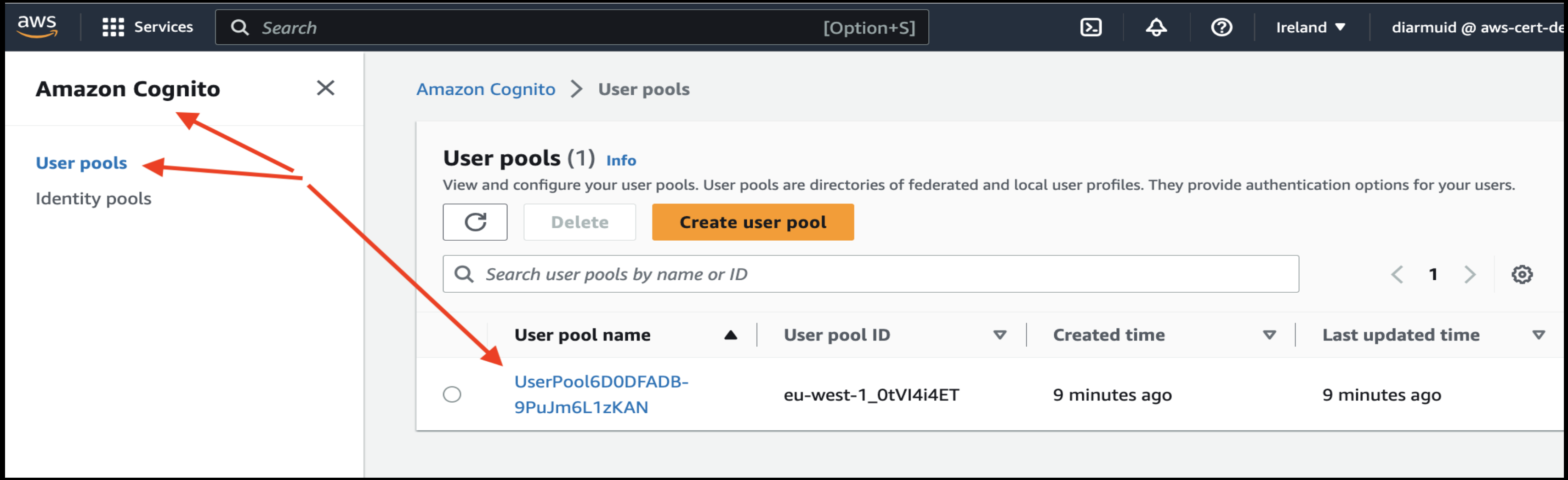
Typical architecture



Demo – User pool



```
// CDK setup
const userPool = new UserPool(this, 'UserPool', {
  signInAliases: { username: true, email: true },
  selfSignUpEnabled: true,
  removalPolicy: RemovalPolicy.DESTROY,
});

const appClient = userPool.addClient('AppClient', {
  authFlows: { userPassword: true },
});
```

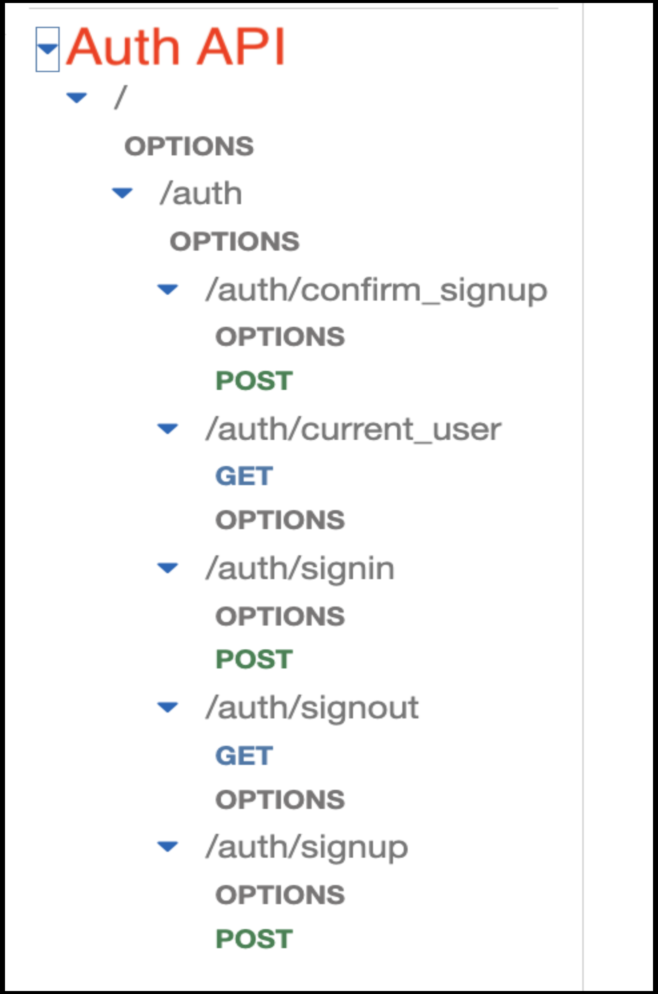


User pools (1) Info

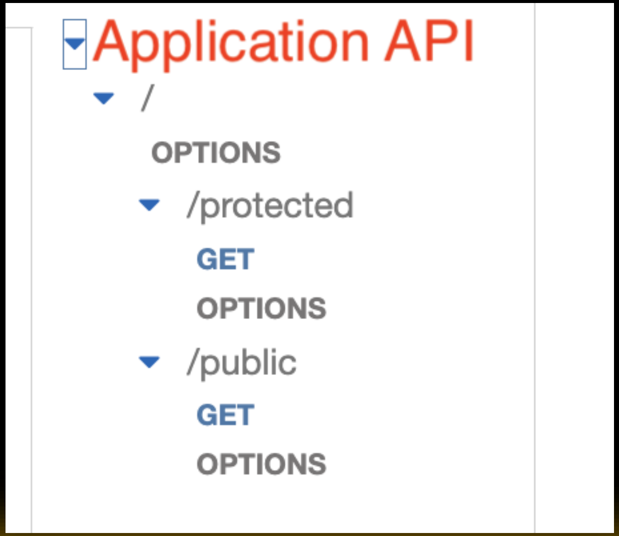
View and configure your user pools. User pools are directories of federated and local user profiles. They provide authentication options for your users.

User pool name	User pool ID	Created time	Last updated time
UserPool6D0DFADB-9PuJm6L1zKAN	eu-west-1_0tVI4i4ET	9 minutes ago	9 minutes ago

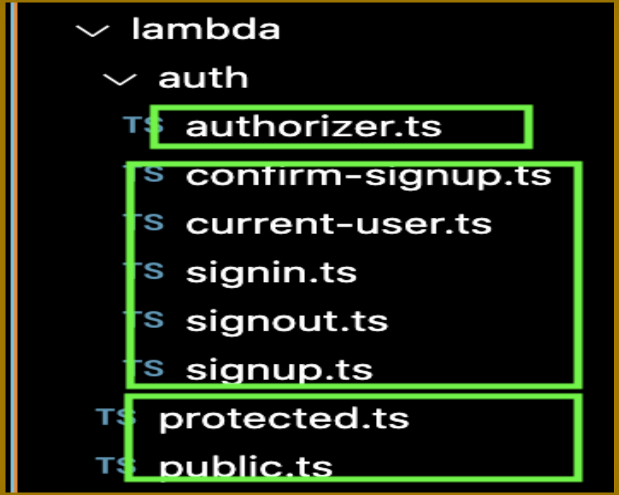
Demo – APIs.



Auth API
- /
  - OPTIONS
  - /auth
    - OPTIONS
    - /auth/confirm_signup
      - OPTIONS
      - POST
    - /auth/current_user
      - GET
      - OPTIONS
    - /auth/signin
      - OPTIONS
      - POST
    - /auth/signout
      - GET
      - OPTIONS
    - /auth/signup
      - OPTIONS
      - POST

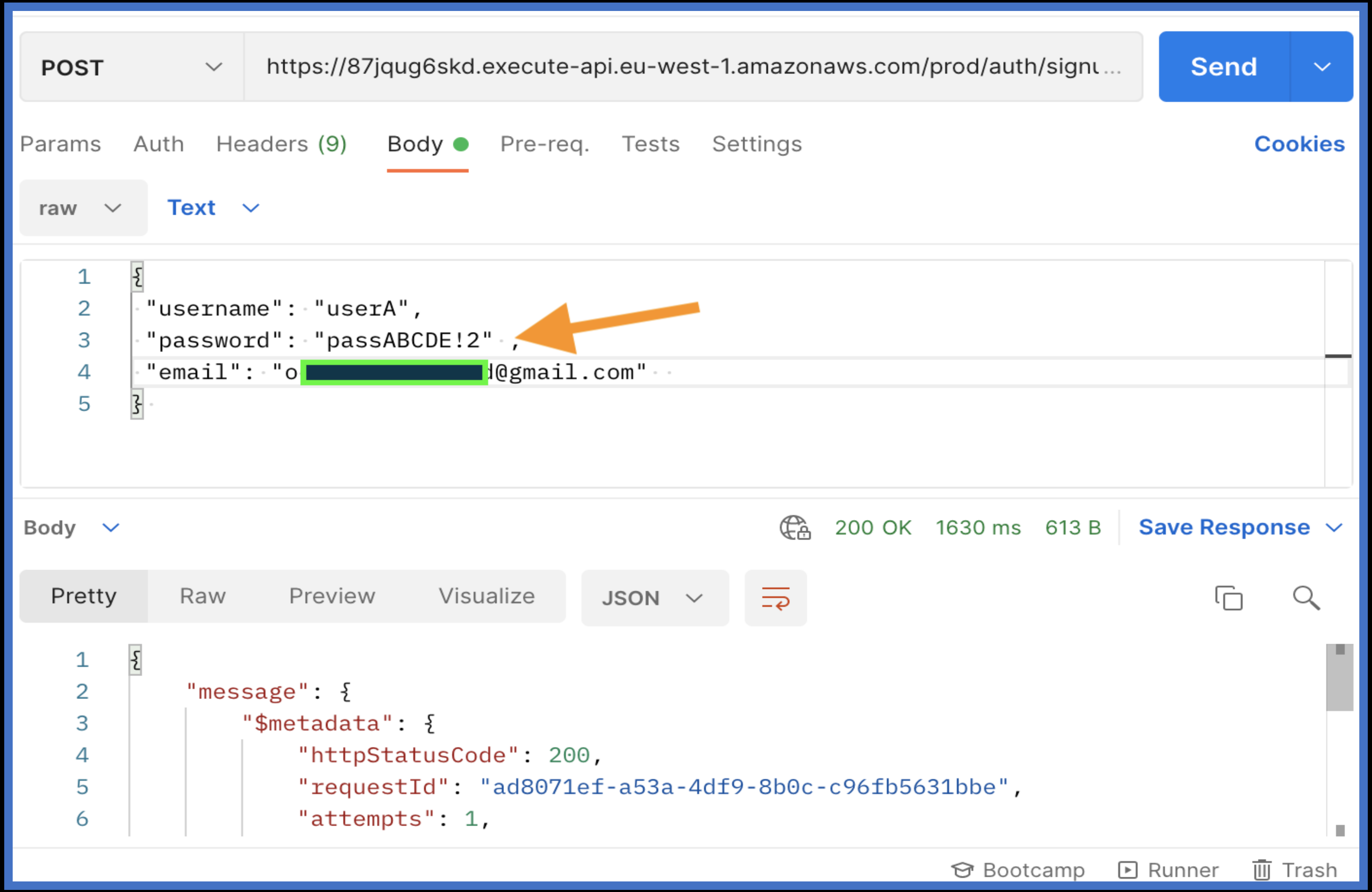


Application API
- /
  - OPTIONS
  - /protected
    - GET
    - OPTIONS
  - /public
    - GET
    - OPTIONS



```
lambda
  auth
    authorizer.ts
    confirm-signup.ts
    current-user.ts
    signin.ts
    signout.ts
    signup.ts
  protected.ts
  public.ts
```

Demo – Sign up



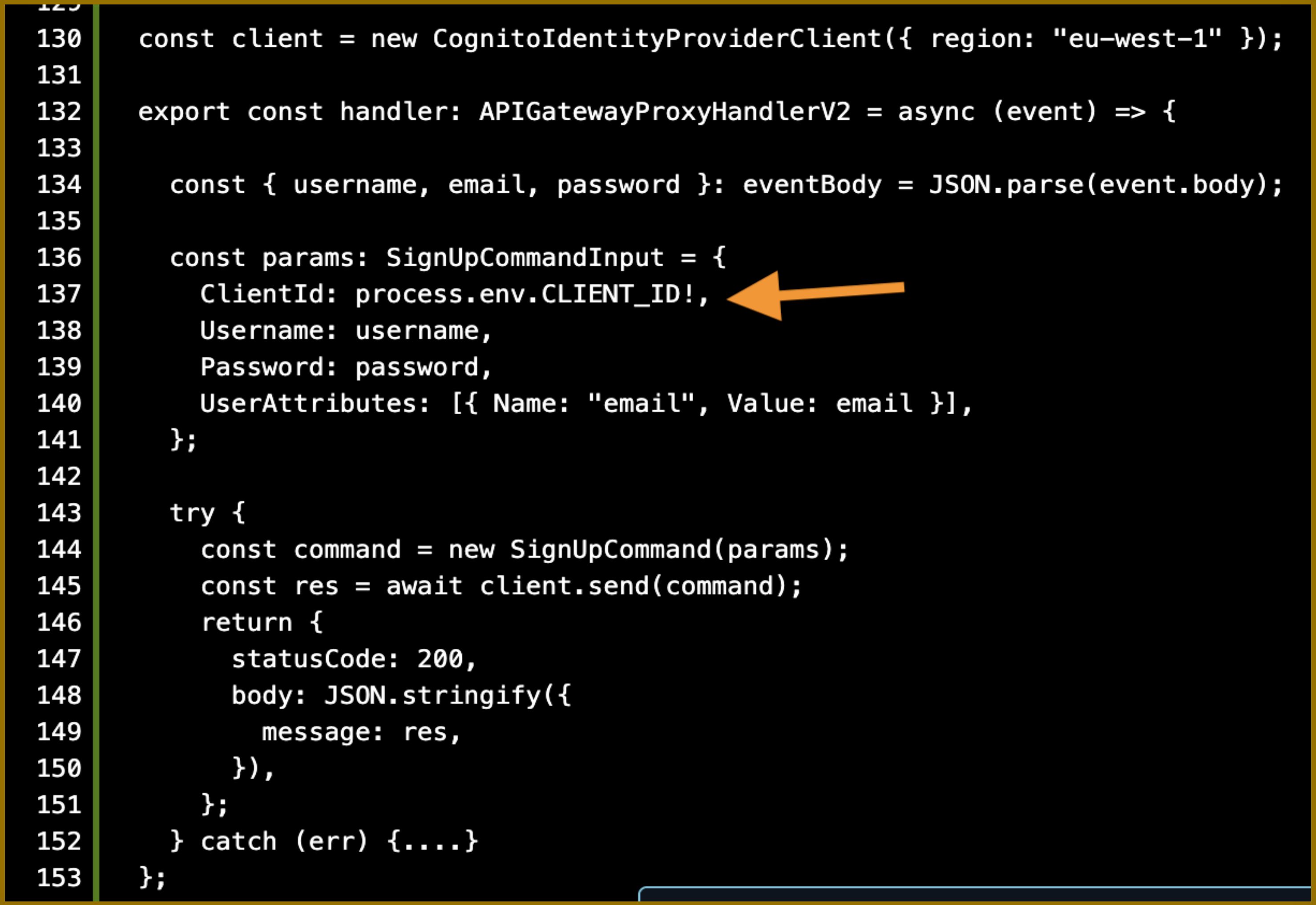
https://87jqug6skd.execute-api.eu-west-1.amazonaws.com/prod/auth/signu...

```
{
"username": "userA",
"password": "passABCDE!2" ,
"email": "o[redacted]d@gmail.com"
}
```

200 OK 1630 ms 613 B

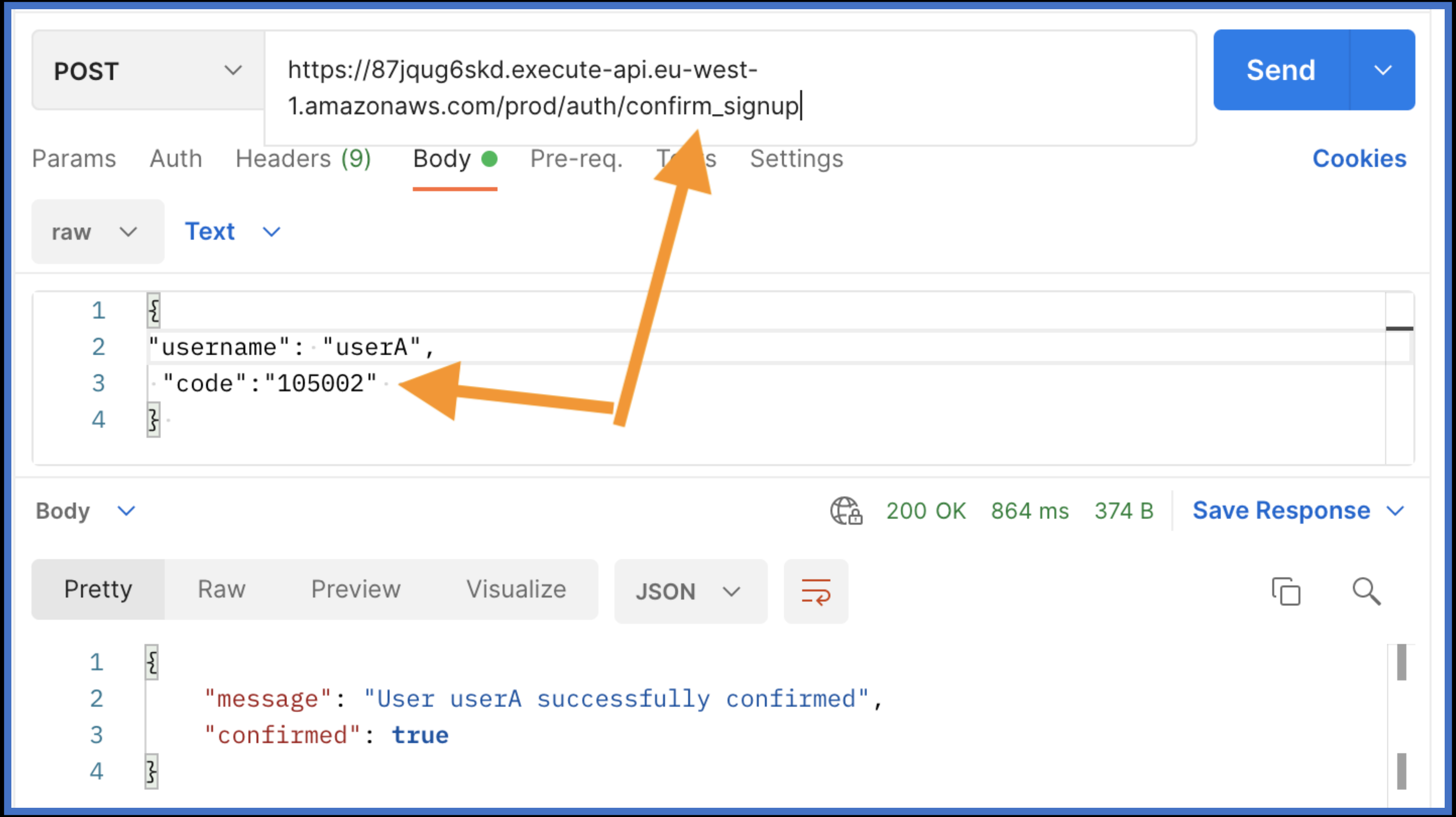
```
{
    "message": {
        "$metadata": {
            "httpStatusCode": 200,
            "requestId": "ad8071ef-a53a-4df9-8b0c-c96fb5631bbe",
            "attempts": 1,
```

Demo – SignUp



```
const client = new CognitoIdentityProviderClient({ region: "eu-west-1" });

export const handler: APIGatewayProxyHandlerV2 = async (event) => {

  const { username, email, password }: eventBody = JSON.parse(event.body);

  const params: SignUpCommandInput = {
    ClientId: process.env.CLIENT_ID!,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  };

  try {
    const command = new SignUpCommand(params);
    const res = await client.send(command);
    return {
      statusCode: 200,
      body: JSON.stringify({
        message: res,
      }),
    };
  } catch (err) {....}
};
```

Demo – Confirm SignUp



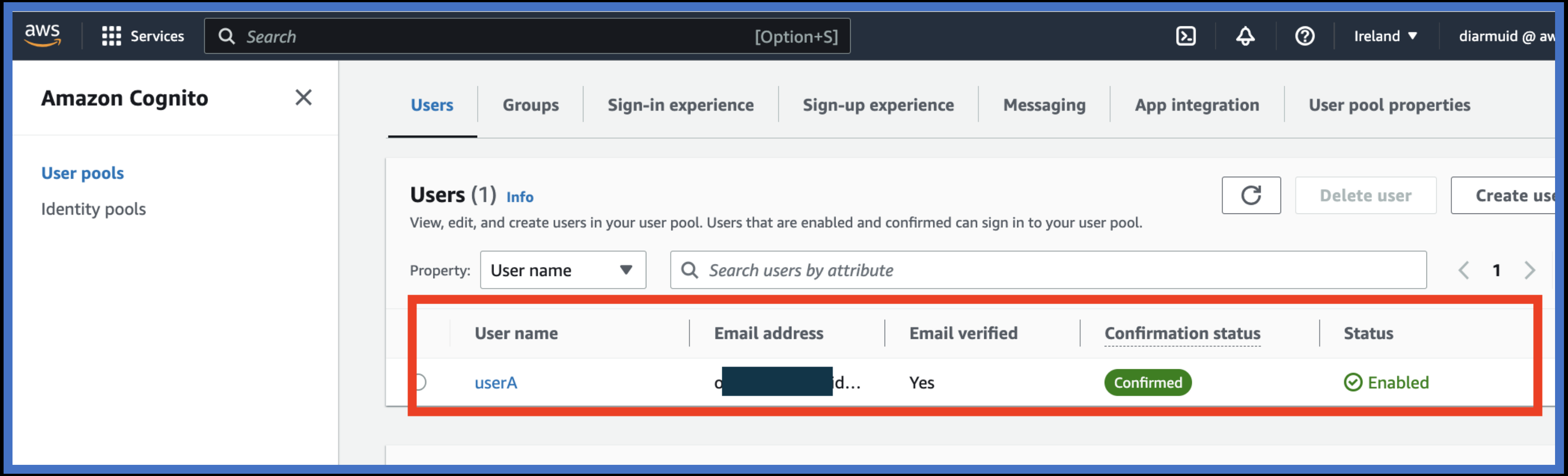
https://87jqug6skd.execute-api.eu-west-1.amazonaws.com/prod/auth/confirm_signup

```
{
"username": "userA",
 "code":"105002"
}
```

200 OK 864 ms 374 B

```
{
    "message": "User userA successfully confirmed",
    "confirmed": true
}
```

Demo – Confirm SignUp



Users (1)

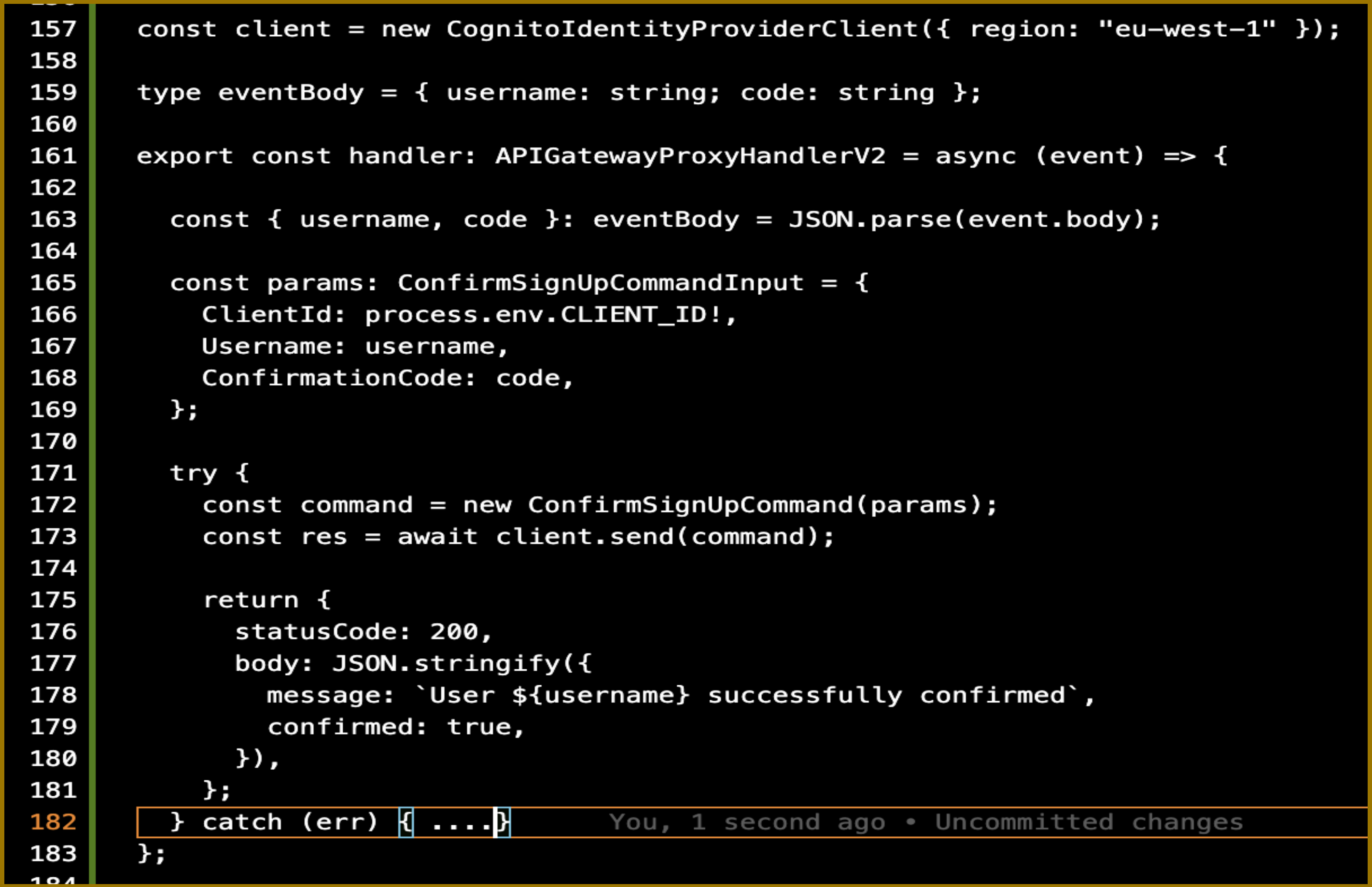
View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name

Search users by attribute

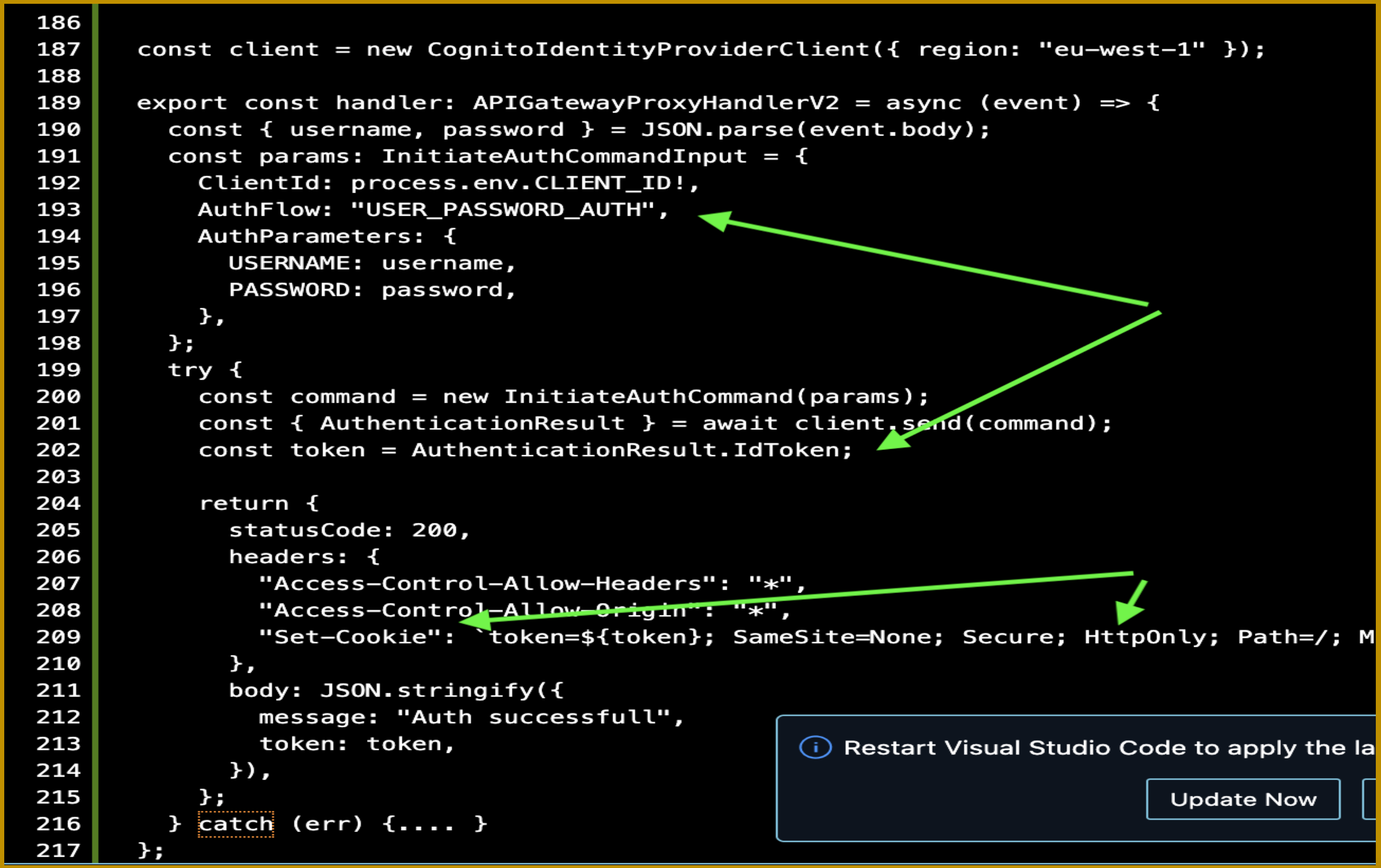
User name	Email address	Email verified	Confirmation status	Status
userA	c[redacted]d...	Yes	Confirmed	✓ Enabled

Demo – Confirm SignUp



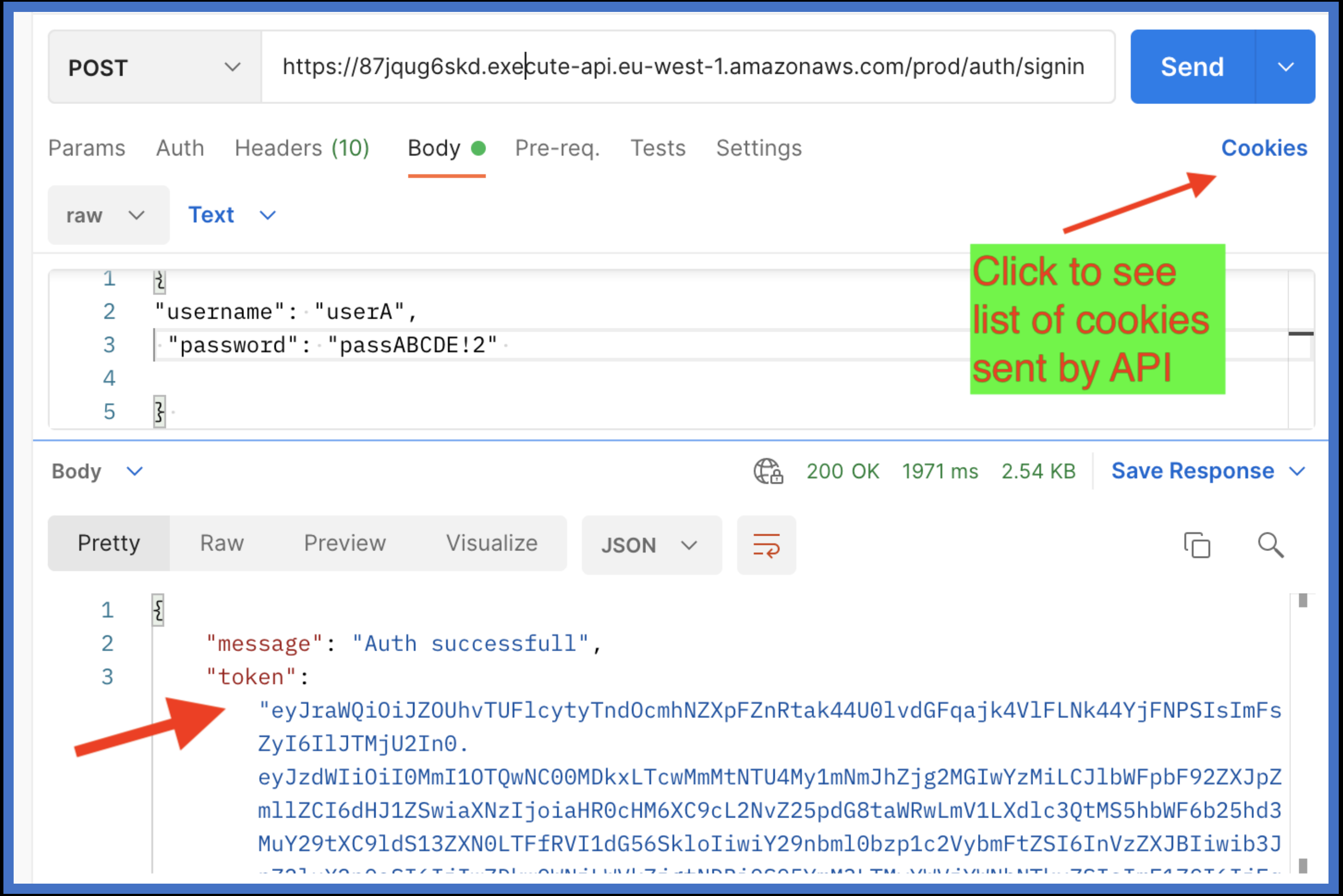
```
const client = new CognitoIdentityProviderClient({ region: "eu-west-1" });

type eventBody = { username: string; code: string };

export const handler: APIGatewayProxyHandlerV2 = async (event) => {

  const { username, code }: eventBody = JSON.parse(event.body);

  const params: ConfirmSignUpCommandInput = {
    ClientId: process.env.CLIENT_ID!,
    Username: username,
    ConfirmationCode: code,
  };

  try {
    const command = new ConfirmSignUpCommand(params);
    const res = await client.send(command);

    return {
      statusCode: 200,
      body: JSON.stringify({
        message: `User ${username} successfully confirmed`,
        confirmed: true,
      }),
    };
  } catch (err) { ....}      You, 1 second ago • Uncommitted changes
};
```

Demo – Sign In



```
const client = new CognitoIdentityProviderClient({ region: "eu-west-1" });

export const handler: APIGatewayProxyHandlerV2 = async (event) => {
  const { username, password } = JSON.parse(event.body);
  const params: InitiateAuthCommandInput = {
    ClientId: process.env.CLIENT_ID!,
    AuthFlow: "USER_PASSWORD_AUTH",
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
  };
  try {
    const command = new InitiateAuthCommand(params);
    const { AuthenticationResult } = await client.send(command);
    const token = AuthenticationResult.IdToken;

    return {
      statusCode: 200,
      headers: {
        "Access-Control-Allow-Headers": "*",
        "Access-Control-Allow-Origin": "*",
        "Set-Cookie": `token=${token}; SameSite=None; Secure; HttpOnly; Path=/; M
      },
      body: JSON.stringify({
        message: "Auth successfull",
        token: token,
      }),
    };
  } catch (err) {.... }
};
```

Demo – Sign In request / response



https://87jqug6skd.execute-api.eu-west-1.amazonaws.com/prod/auth/signin

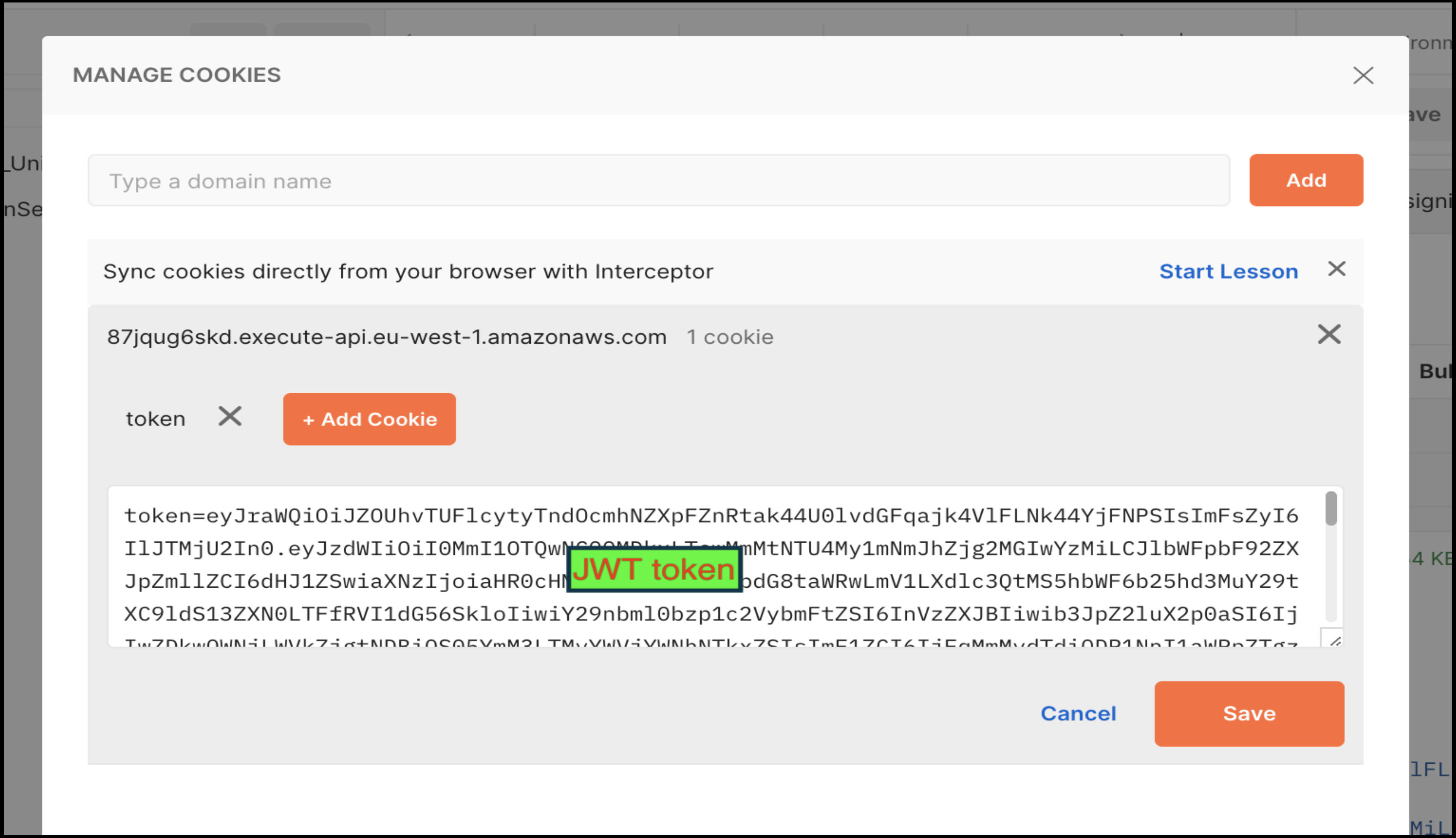
```
{
"username": "userA",
 "password": "passABCDE!2"

}
```

Click to see list of cookies sent by API

200 OK 1971 ms 2.54 KB

```
{
    "message": "Auth successfull",
    "token":
        "eyJraWQiOiJZOUhvTUFlcytyTndOcmhNZXpFZnRtak44U0lvdGFqajk4VlFLNk44YjFNPSIsImFs
        ZyI6IlJTMjU2In0.
        eyJzdWIiOiI0MmI1OTQwNC00MDkxLTcwMmMtNTU4My1mNmJhZjg2MGIwYzMiLCJlbWFpbF92ZXJpZ
        mllZCI6dHJ1ZSwiaXNzIjoiaHR0cHM6XC9cL2NvZ25pdG8taWRwLmV1LXdlc3QtMS5hbWF6b25hd3
        MuY29tXC9ldS13ZXN0LTFfRVI1dG56SkloIiwiY29nbml0bzp1c2VybmFtZSI6InVzZXJBIiwib3J
```

Demo – Sign In JWT token

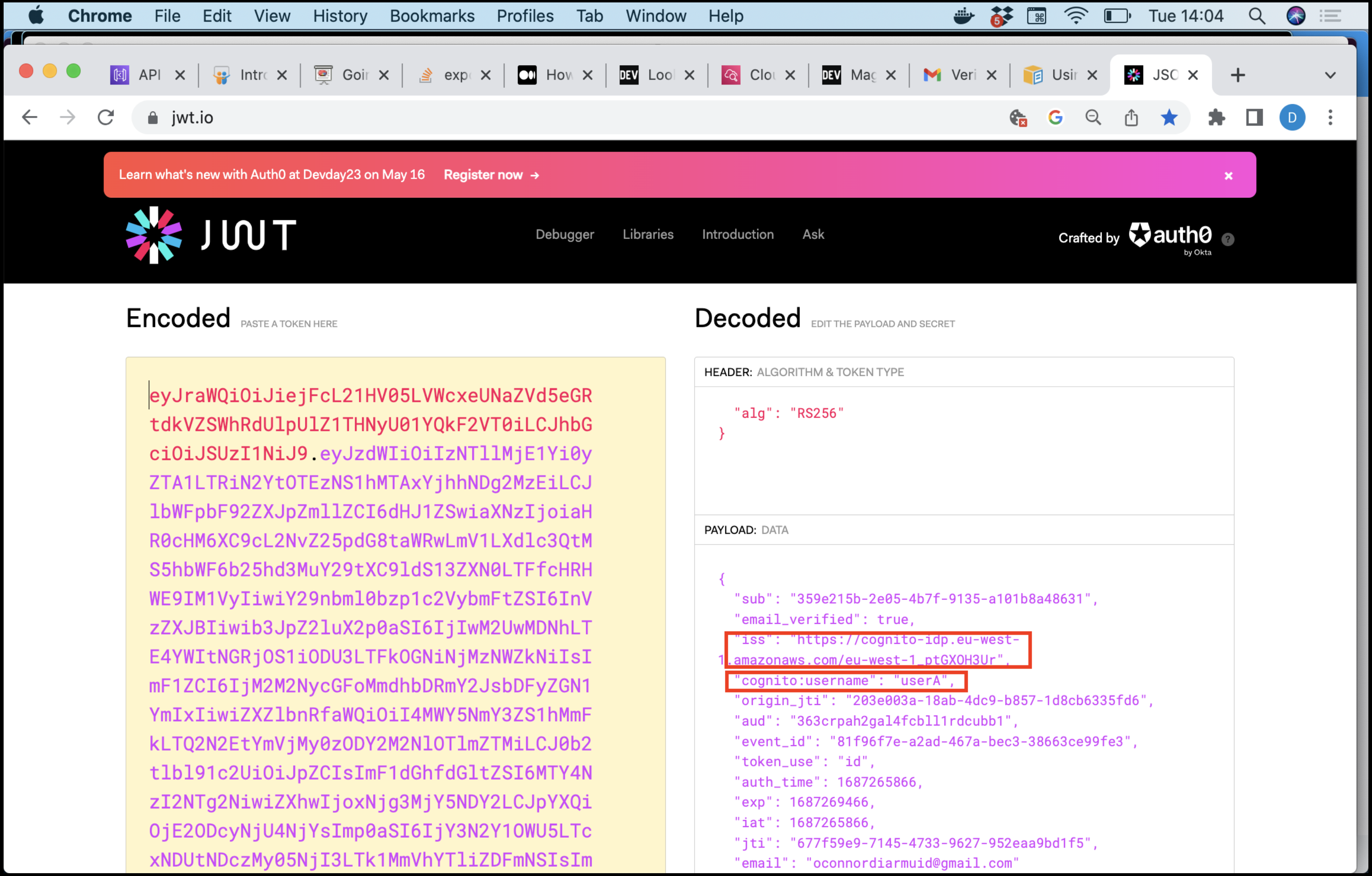


Sync cookies directly from your browser with Interceptor

87jqug6skd.execute-api.eu-west-1.amazonaws.com 1 cookie

```
token=eyJraWQiOiJZOUhvTUFlcytyTndOcmhNZXpFZnRtak44U0lvdGFqajk4VlFLNk44YjFNPSIsImFsZyI6
IlJTMjU2In0.eyJzdWIiOiI0MmI1OTQw[illegible]mMtNTU4My1mNmJhZjg2MGIwYzMiLCJlbWFpbF92ZX
JpZmllZCI6dHJ1ZSwiaXNzIjoiaHR0cHM[illegible]dG8taWRwLmV1LXdlc3QtMS5hbWF6b25hd3MuY29t
XC9ldS13ZXN0LTFfRVI1dG56SkloIiwiY29nbml0bzp1c2VybmFtZSI6InVzZXJBIiwib3JpZ2luX2p0aSI6Ij
[illegible]
```

JWT token decoding



Learn what's new with Auth0 at Devday23 on May 16 Register now →

Debugger Libraries Introduction Ask

Crafted by auth0 by Okta

Encoded PASTE A TOKEN HERE

```
eyJraWQiOiJiejFcL21HV05LVWcxeUNaZVd5eGR
tdkVZSWhRdUlpUlZ1THNyU01YQkF2VT0iLCJhbG
ciOiJSUzI1NiJ9.eyJzdWIiOiIzNTllMjE1Yi0y
ZTA1LTRiN2YtOTEzNS1hMTAxYjhhNDg2MzEiLCJ
lbWFpbF92ZXJpZmllZCI6dHJ1ZSwiaXNzIjoiaH
R0cHM6XC9cL2NvZ25pdG8taWRwLmV1LXdlc3QtM
S5hbWF6b25hd3MuY29tXC9ldS13ZXN0LTFfcHRH
WE9IM1VyIiwiY29nbml0bzp1c2VybmFtZSI6InV
zZXJBIiwib3JpZ2luX2p0aSI6IjIwM2UwMDNhLT
E4YWItNGRjOS1iODU3LTFkOGNiNjMzNWZkNiIsI
mF1ZCI6IjM2M2NycGFoMmdhbDRmY2JsbDFyZGN1
YmIxIiwiZXZlbnRfaWQiOiI4MWY5NmY3ZS1hMmF
kLTQ2N2EtYmVjMy0zODY2M2NlOTlmZTMiLCJ0b2
tlbl91c2UiOiJpZCIsImF1dGhfdGltZSI6MTY4N
zI2NTg2NiwiZXhwIjoxNjg3MjY5NDY2LCJpYXQi
OjE2ODcyNjU4NjYsImp0aSI6IjY3N2Y1OWU5LTc
xNDUtNDczMy05NjI3LTk1MmVhYTliZDFmNSIsIm
```

Decoded EDIT THE PAYLOAD AND SECRET

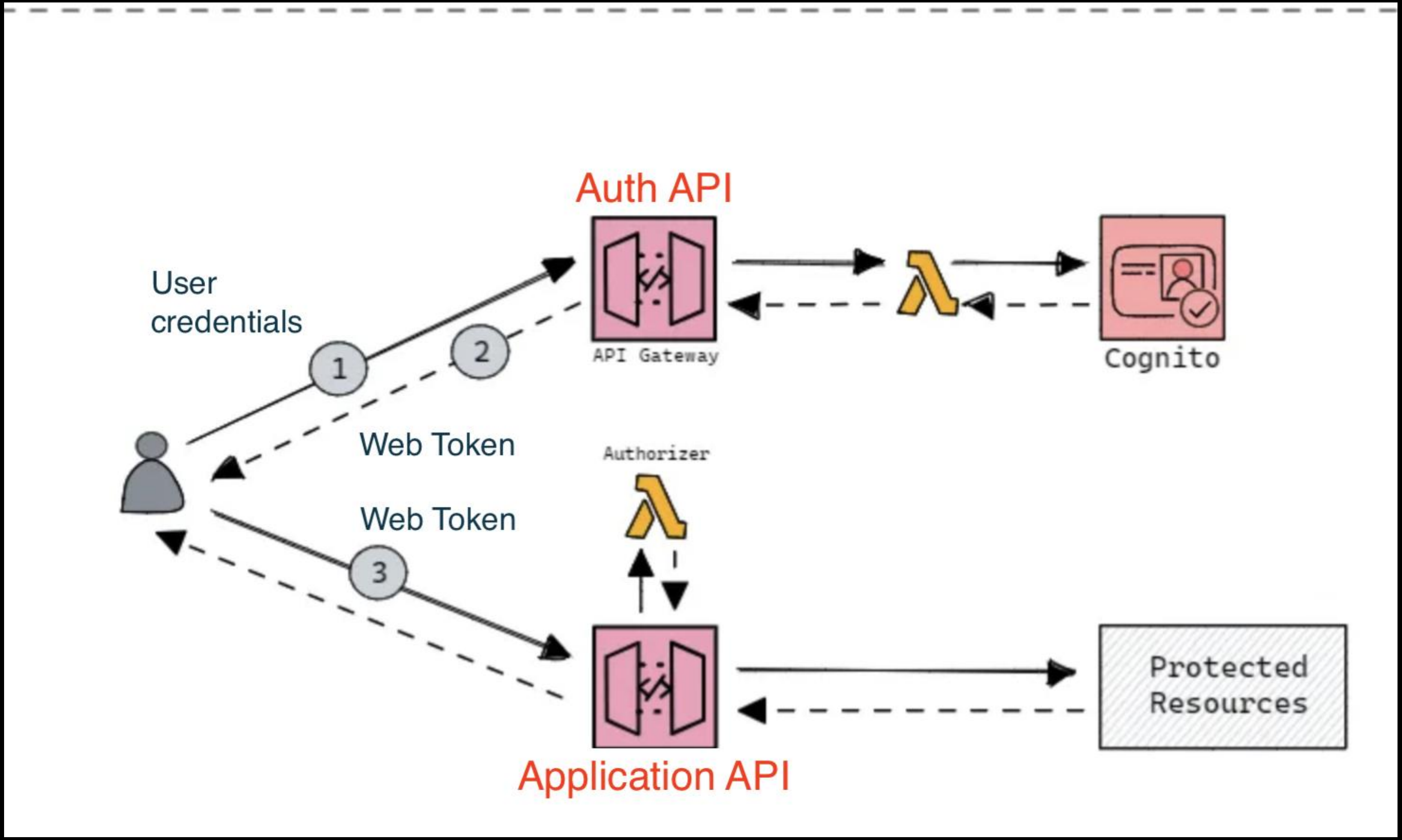
HEADER: ALGORITHM & TOKEN TYPE

```
  "alg": "RS256"
}
```

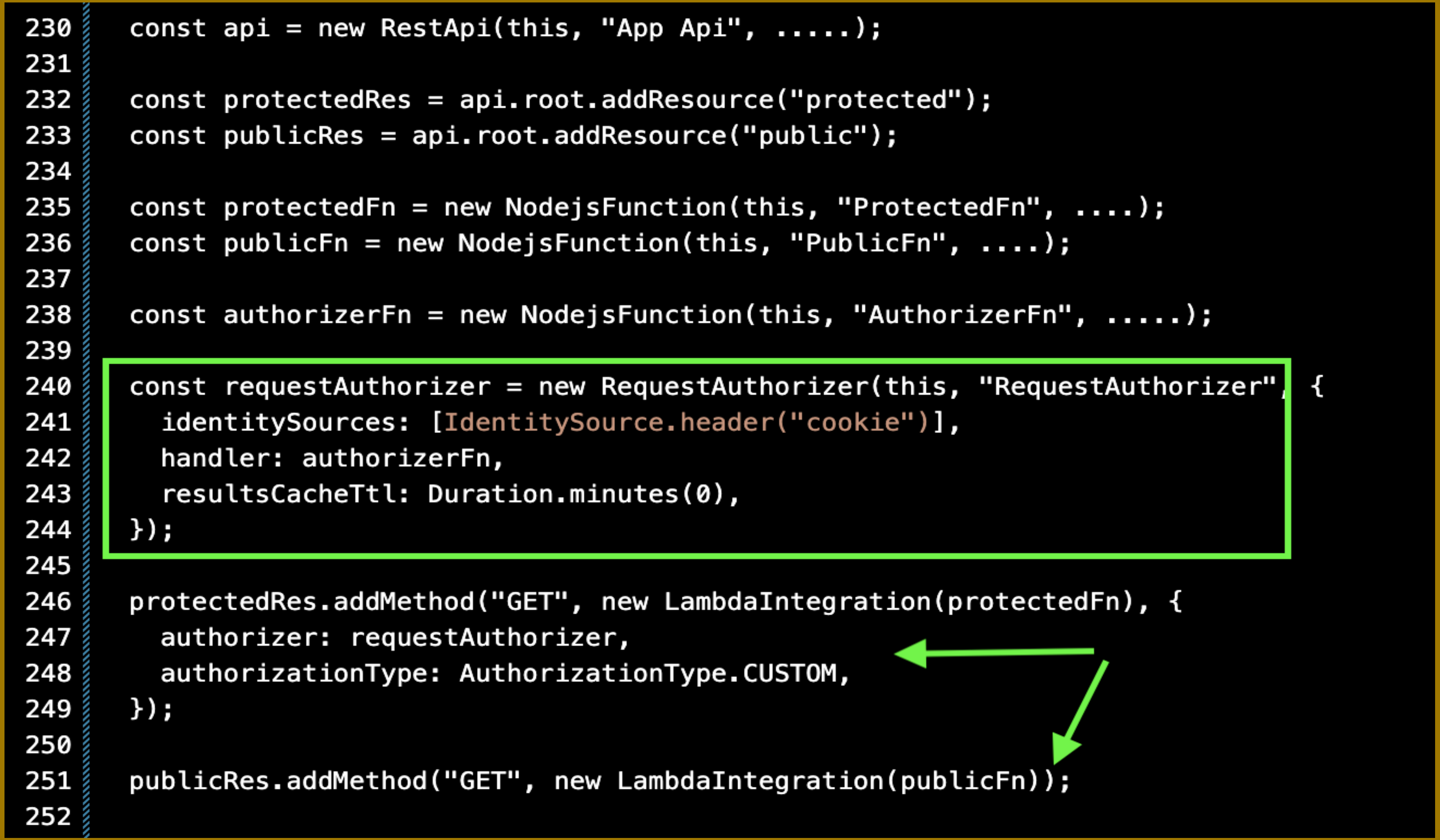
PAYLOAD: DATA

```
{
  "sub": "359e215b-2e05-4b7f-9135-a101b8a48631",
  "email_verified": true,
  "iss": "https://cognito-idp.eu-west-
1.amazonaws.com/eu-west-1_ptGXOH3Ur",
  "cognito:username": "userA",
  "origin_jti": "203e003a-18ab-4dc9-b857-1d8cb6335fd6",
  "aud": "363crpah2gal4fcbll1rdcubb1",
  "event_id": "81f96f7e-a2ad-467a-bec3-38663ce99fe3",
  "token_use": "id",
  "auth_time": 1687265866,
  "exp": 1687269466,
  "iat": 1687265866,
  "jti": "677f59e9-7145-4733-9627-952eaa9bd1f5",
  "email": "oconnordiarmuid@gmail.com"
```

Demo - Architecture



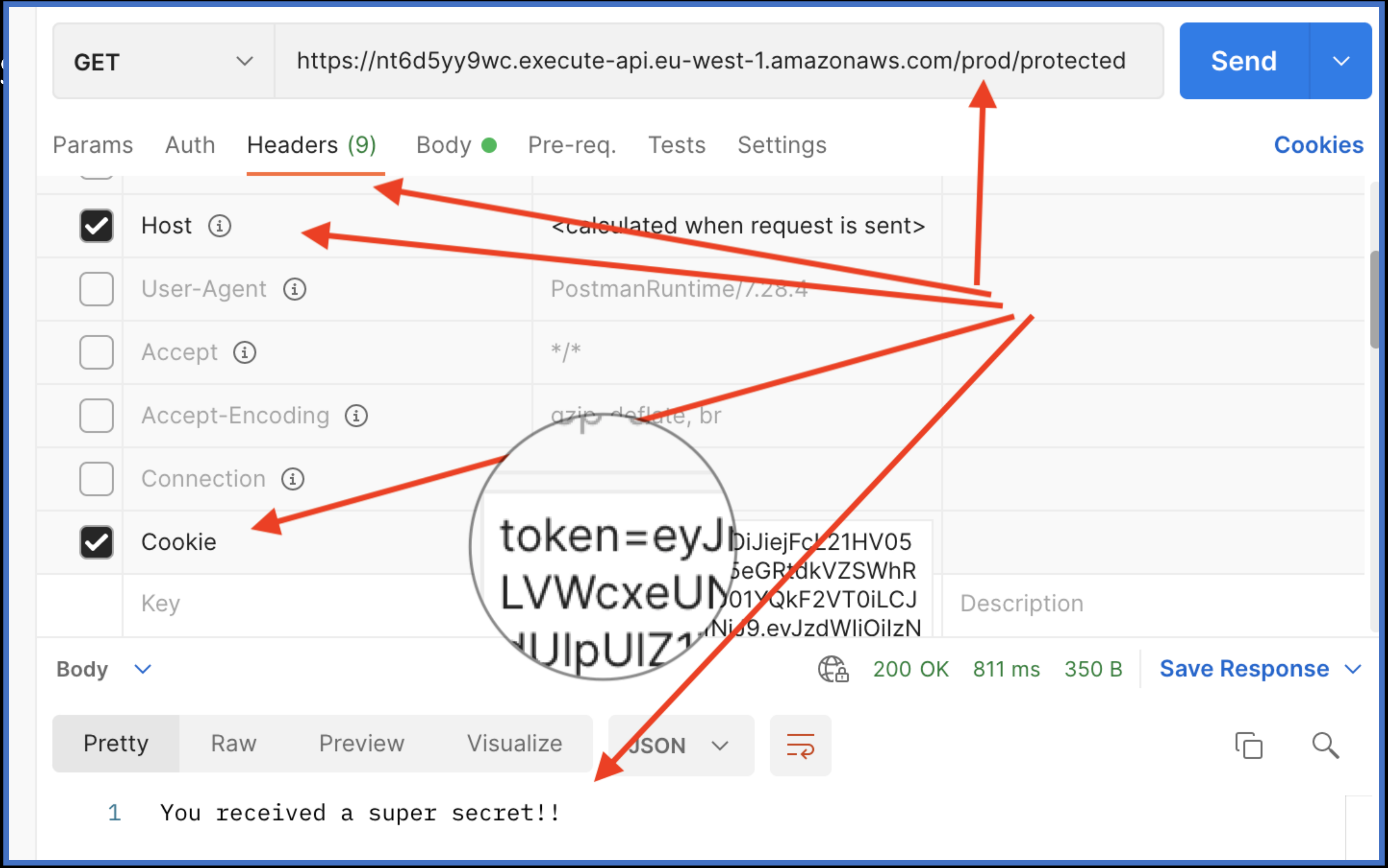
Demo – App API infrastructure.



```
const api = new RestApi(this, "App Api", .....);

const protectedRes = api.root.addResource("protected");
const publicRes = api.root.addResource("public");

const protectedFn = new NodejsFunction(this, "ProtectedFn", ....);
const publicFn = new NodejsFunction(this, "PublicFn", ....);

const authorizerFn = new NodejsFunction(this, "AuthorizerFn", .....);

const requestAuthorizer = new RequestAuthorizer(this, "RequestAuthorizer", {
  identitySources: [IdentitySource.header("cookie")],
  handler: authorizerFn,
  resultsCacheTtl: Duration.minutes(0),
});

protectedRes.addMethod("GET", new LambdaIntegration(protectedFn), {
  authorizer: requestAuthorizer,
  authorizationType: AuthorizationType.CUSTOM,
});

publicRes.addMethod("GET", new LambdaIntegration(publicFn));

```

Demo – Protected route HTTP request

- For AWS



GET https://nt6d5yy9wc.execute-api.eu-west-1.amazonaws.com/prod/protected

Send

Params Auth Headers (9) Body Pre-req. Tests Settings Cookies

	Key	Value	Description
✓	Host	<calculated when request is sent>	
	User-Agent	PostmanRuntime/7.28.4	
	Accept	*/*	
	Accept-Encoding	gzip, deflate, br	
	Connection		
✓	Cookie	token=eyJhbGciOiJiejFcL21HV05LVWcxeUN5eGRtdkVZSWhR01YQkF2VT0iLCJJUlpUIZ1NiJ9.evJzdWliOilzN	
	Key		Description

Body 200 OK 811 ms 350 B Save Response

Pretty Raw Preview Visualize JSON

```
You received a super secret!!
```

Demo – Authorizer

- Steps performed by authorizer:

1. Parse HTTP Request Cookie header value & store in a local Map data structure.
2. Find ‘token’ key in the Map.
3. If not found:
    - Return an IAM policy Denying use of the App Web API.
4. Verify the token - Decode and check user exists in the User pool.
5. If successful verification:
    - Return IAM policy that Allows execution of the App Web API.
    - Else
    - Return policy Denying execution of the App Web API.