Android Authentication on AWS with AWS Amplify

Kurtis Kemple
Developer Advocate
AWS Mobile

What to expect from this session

- Learn how to implement **authentication** for Android apps using
- Amazon Cognito User Pools
- AWS Amplify

Identity Management

"Enables the right individuals to access the right resources at the right times and for the right reasons"

— Wikipedia

Data access patterns

- Public data access
- Private data access
- Custom data access

Public data access

- Data is not user specific
- No restriction is imposed on the data

Private data access

- Data can be private to a specific user
- Access to data is privileged/restricted

Custom data access

- Data can be private/public
- Access to data can be privileged/restricted
- Access to data can be further guarded by application logic

Identity Management

AuthN

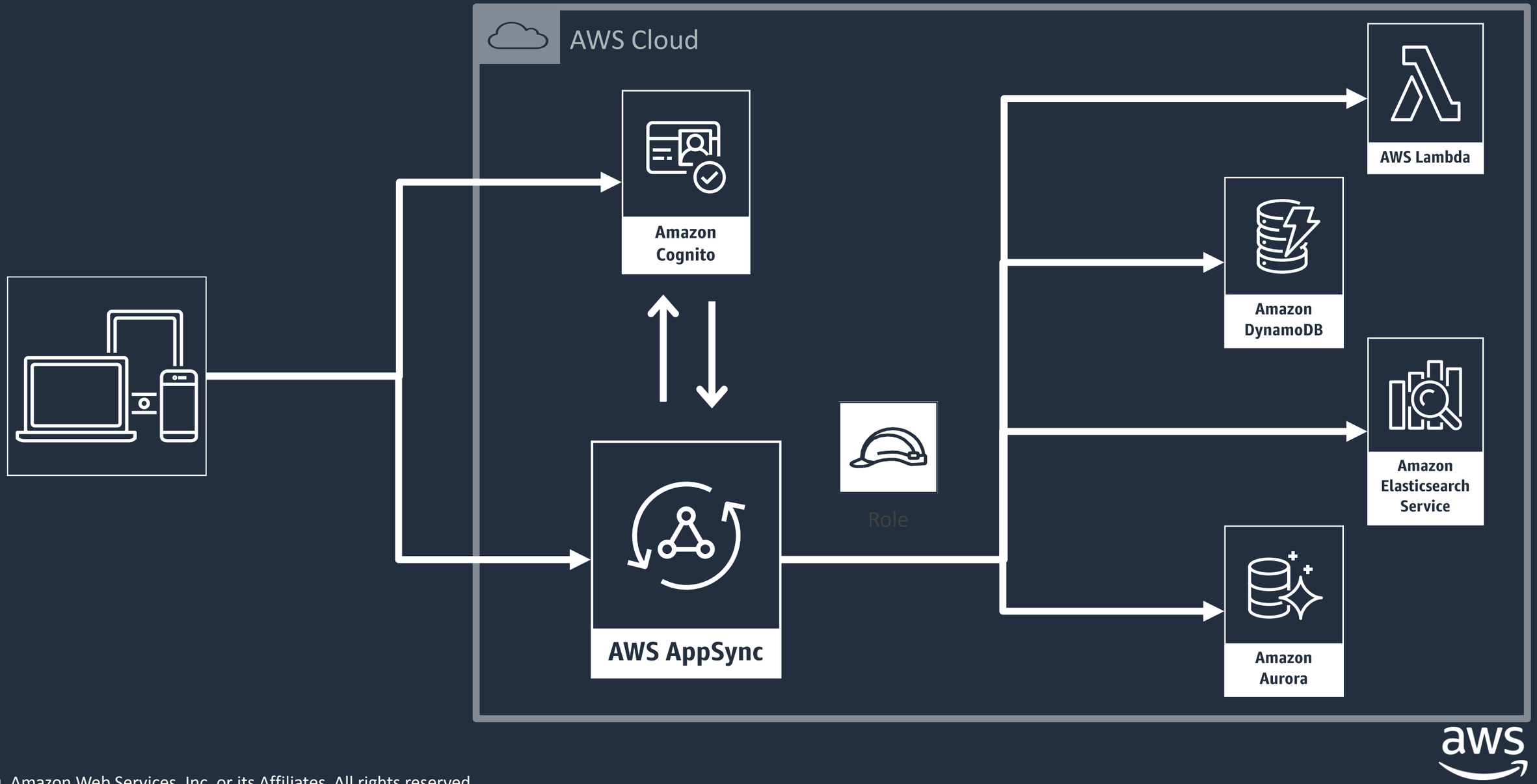
AuthZ

AWS Amplify: Four types of authorization

No authentication → API key

Private/custom data access →
- Amazon Cognito User Pools
- OpenID Connect authorization
- AWS IAM authorization

Amazon Cognito User Pools



Amplify Framework

- ✓ CLI
- ✓ Client Libraries (Native & JS Support)
- ✓ Prebuilt UI Components
- ✓ Toolchain

Amplify Framework

Common Workflows

1. Deploying a Lambda function with API Gateway
2. Adding Authentication
3. Deploying a GraphQL API
4. Adding analytics

Amplify Framework

CLI

- Create, update & delete cloud services.
- Manage multiple environments.
- GraphQL Tranform
- GraphQL Codegen

Amplify CLI

Categories

Authentication

Analytics

API (REST)

API (GraphQL)

Storage

Push Notifications

Amplify CLI

```
$ amplify add auth

$ amplify push

$ amplify configure auth

$ amplify delete

$ amplify codegen add
```

AWS Amplify Native Client SDK

- Opinionated
- Declarative
- Client focused
- High-level abstractions
- Best practices built-in
- Built with Amplify CLI in mind

Adding Authentication

1. Create the authentication service

```
$ amplify add auth

// choose either default or custom configuration
```

Adding Authentication

2. Import dependencies

```
//For AWSMobileClient only:
implementation 'com.amazonaws:aws-android-sdk-mobile-client:2.13.+’
// Cognito UserPools for SignIn
implementation 'com.amazonaws:aws-android-sdk-auth-userpools:2.13.+’
//For the drop-in UI also:
implementation 'com.amazonaws:aws-android-sdk-auth-ui:2.13.+'
```

Adding Authentication

3. Set Up Authentication Activity

```
AWSMobileClient.getInstance().initialize(getApplicationContext(), new
        Callback<UserStateDetails>() {

  @Override

  public void onResult(UserStateDetails userStateDetails) {

     switch (userStateDetails.getUserState()){

        case SIGNED_IN: Intent i = new Intent(AuthenticationActivity.this,
          MainActivity.class); startActivity(i); break;

        case SIGNED_OUT: showSignIn(); break;

        …

} });
```


aws

Adding Authentication

4. Set Up Sign In

```
private void showSignIn() {
    try {
        AWSMobileClient.getInstance().showSignIn(this,
            SignInUIOptions.builder().nextActivity(MainActivity.class).build());
    } catch (Exception e) {
        Log.e(TAG, e.toString());
    }
}
```

DEMO

Adding Authentication

API Methods

```
AWSMobileClient.getInstance()
      - signUp
      - confirmSignUp
      - signIn
      - confirmSignIn
      - signOut
```

Thank you!

Kurtis Kemple