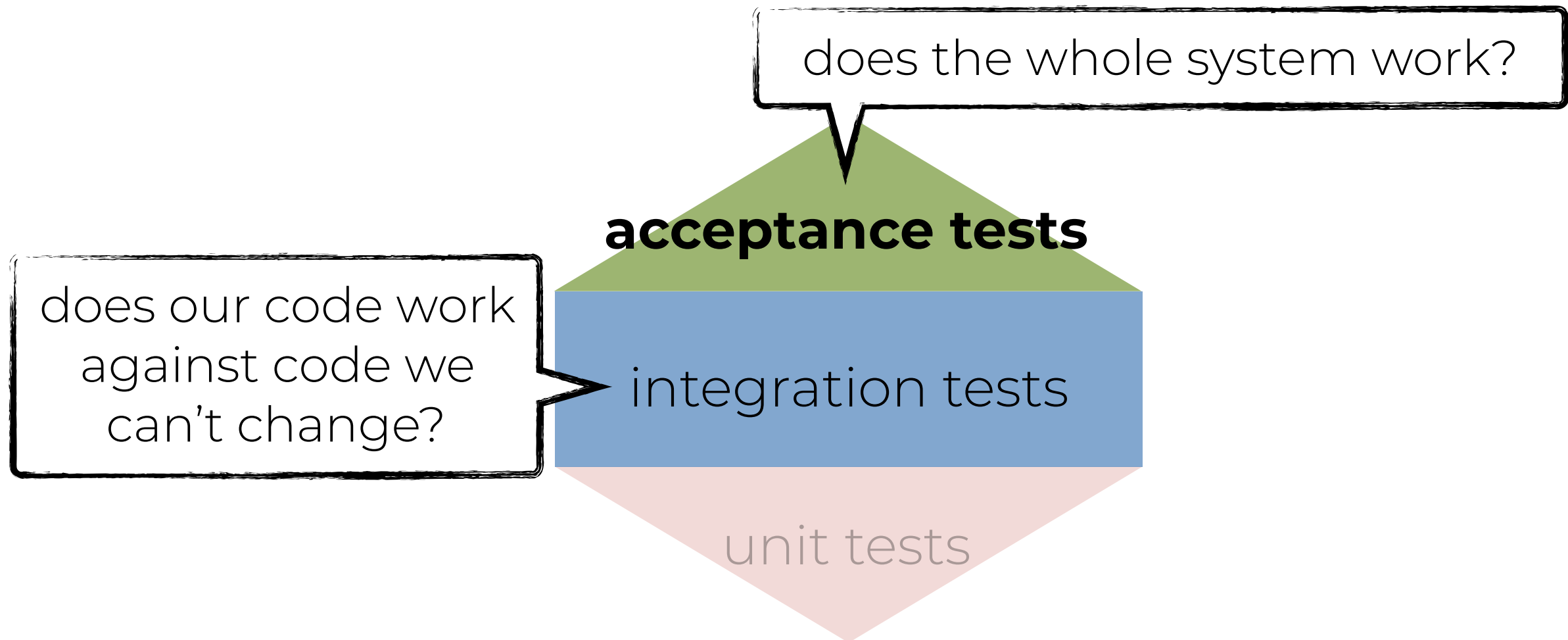
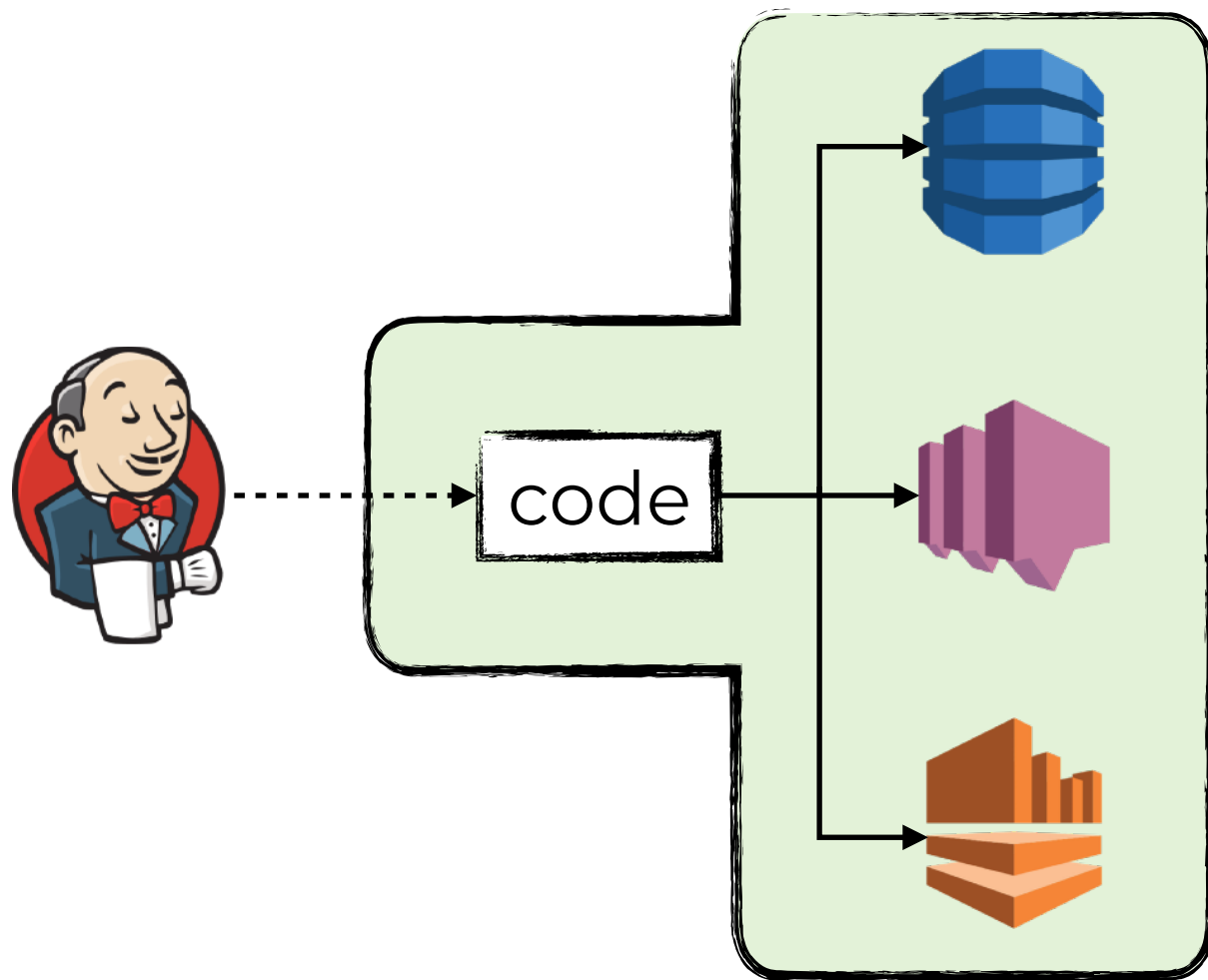


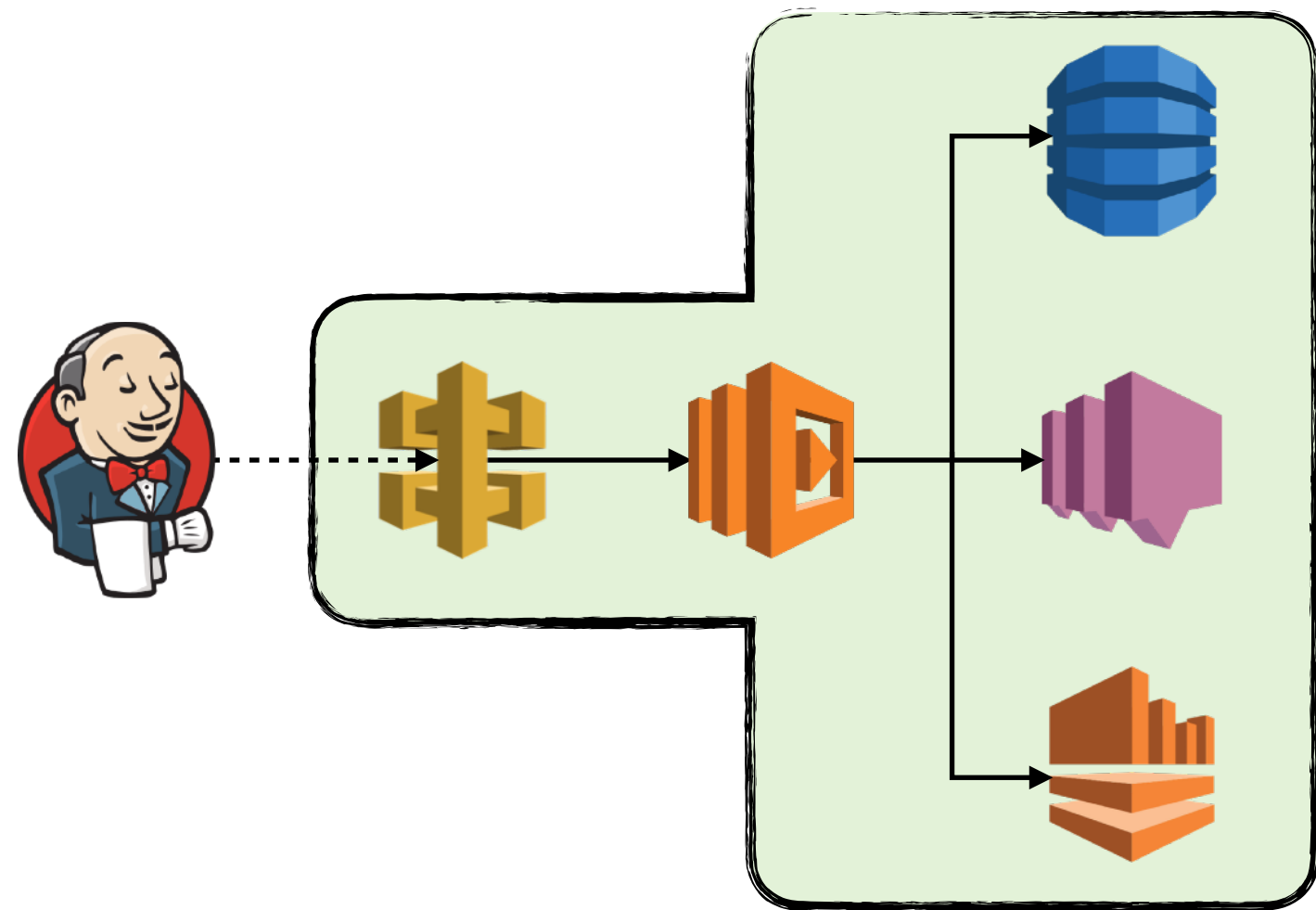
# test pyramid



# reuse test cases



integration tests exercise system's  
**Integration** with its external dependencies



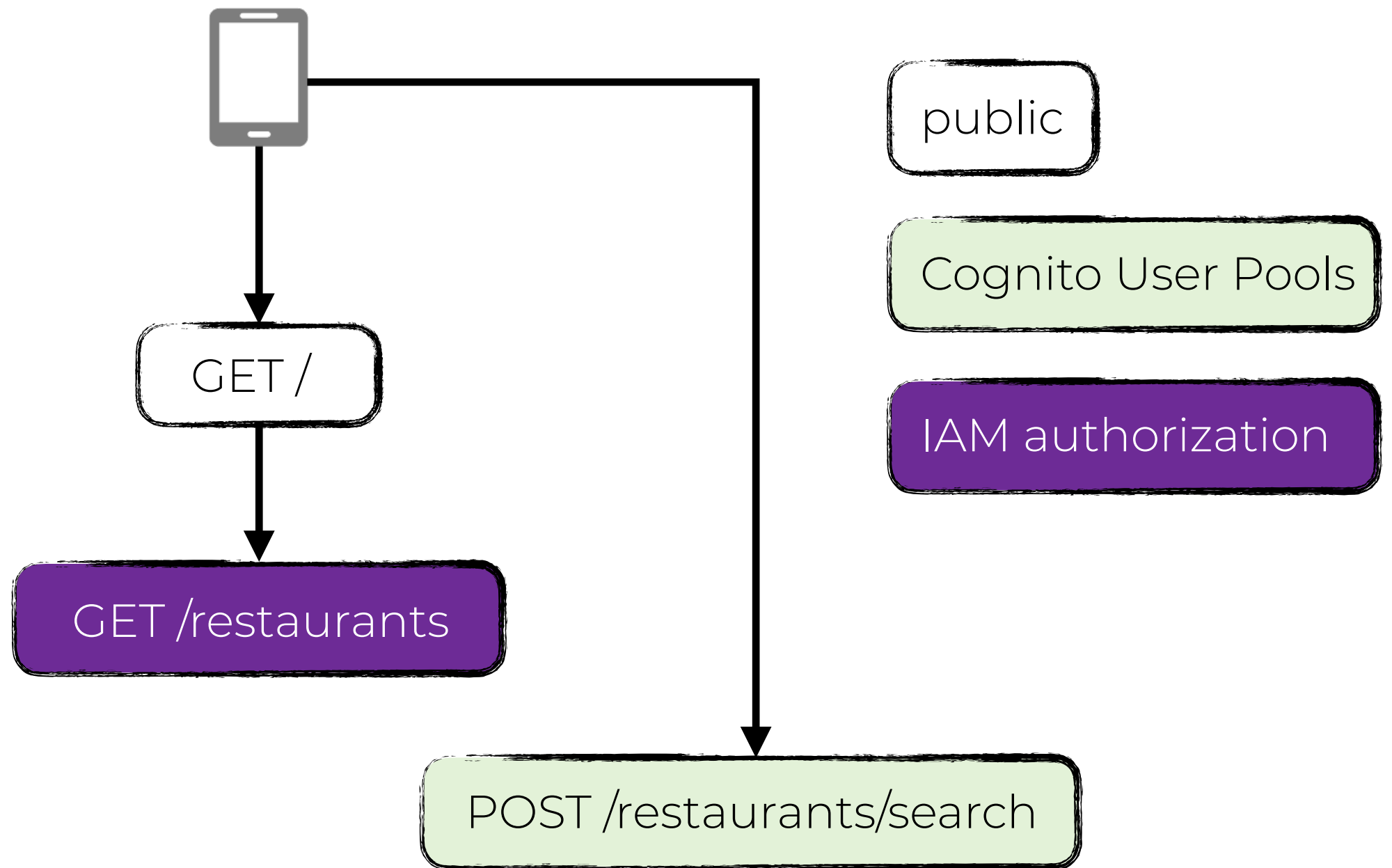
acceptance tests exercise system  
**End-to-End** from the outside

reuse test cases

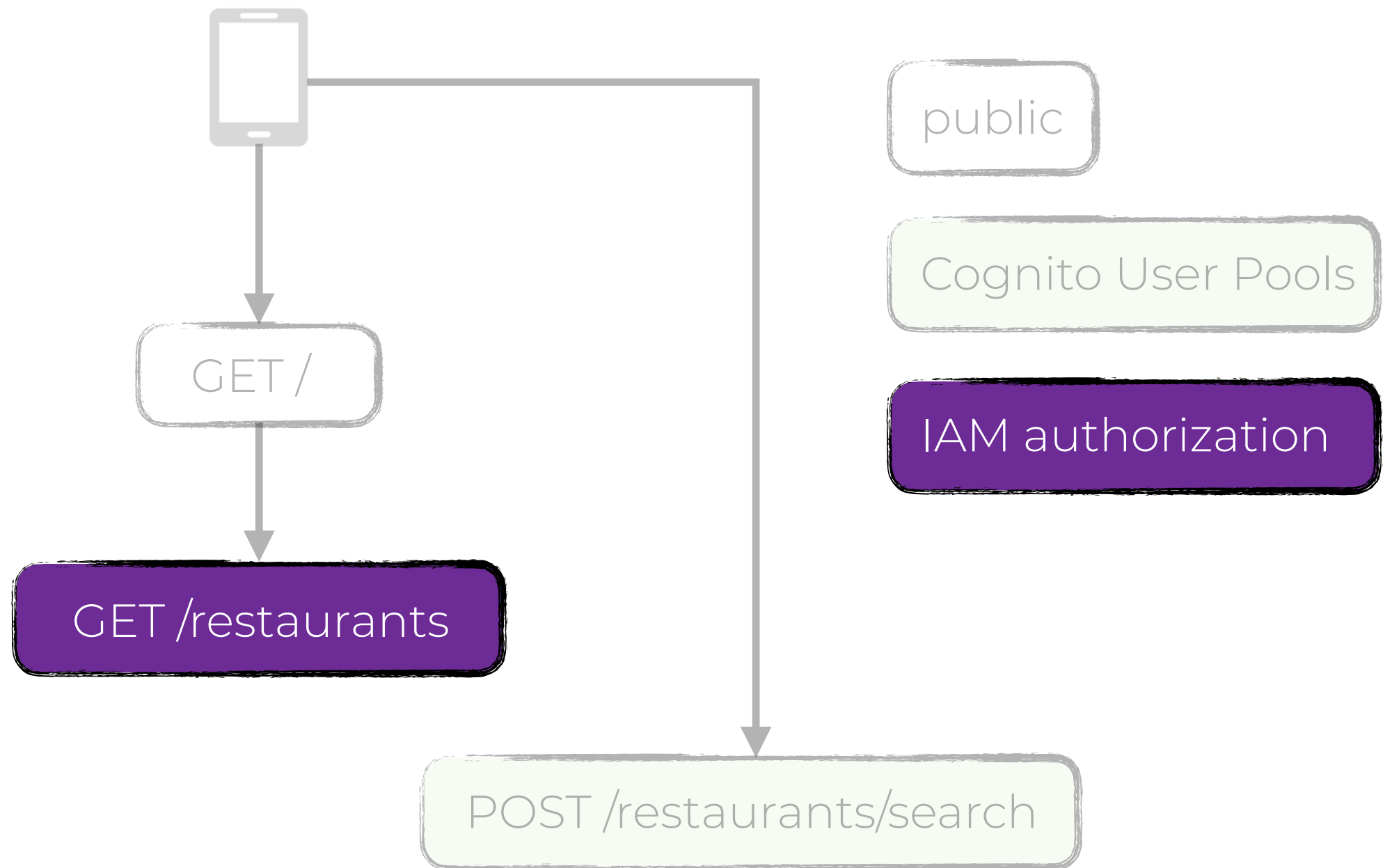
observation

integration tests differ from  
acceptance tests only in **HOW** the  
Lambda functions are invoked

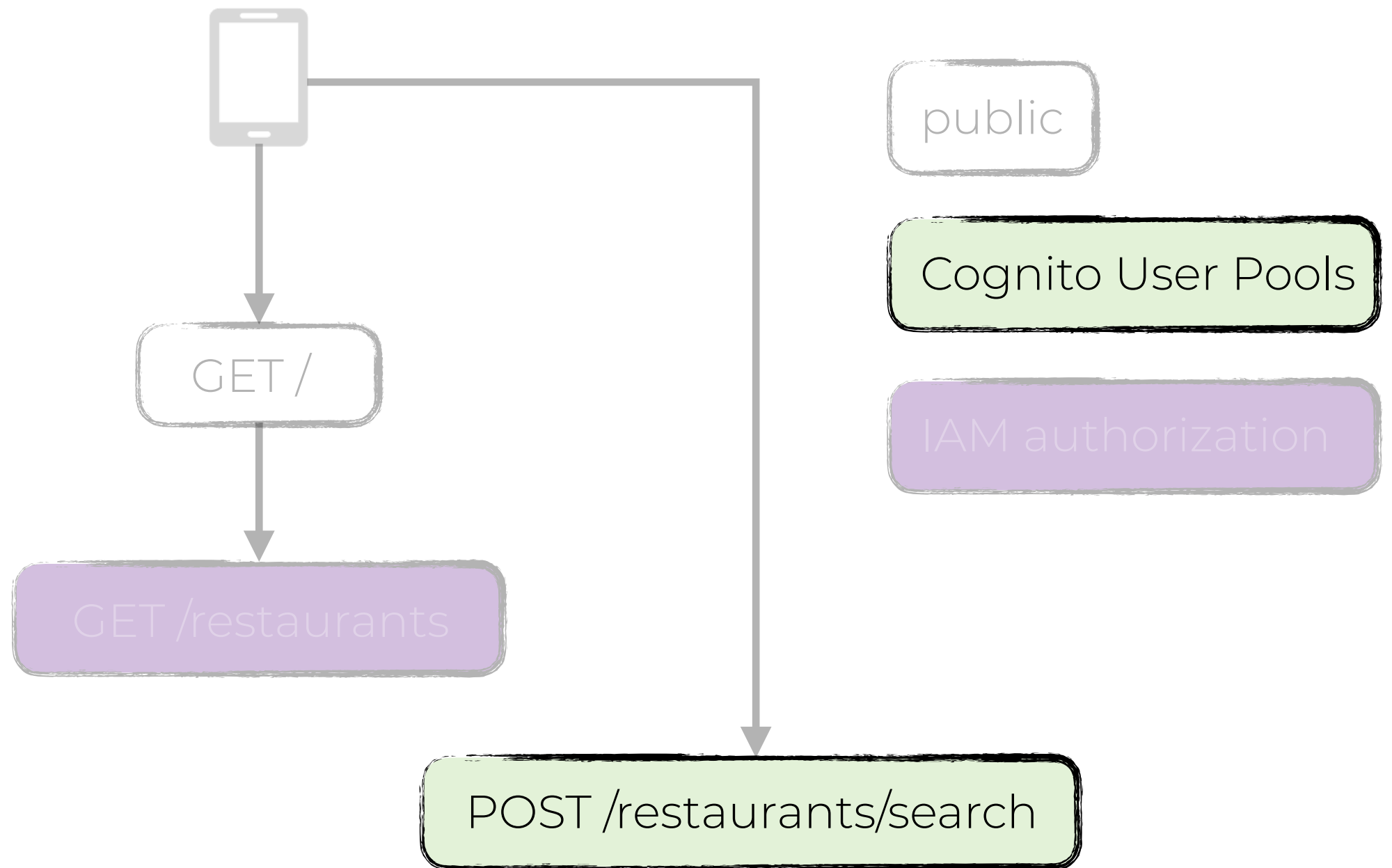
# acceptance tests



# acceptance tests



# acceptance tests



# acceptance tests

## Class List

Classes | Methods | Properties | Files

Search:

▶ CloudwatchLogs < Service

▶ CodeBuild < Service

▶ CodeCommit < Service

▶ CodeDeploy < Service

▶ CodePipeline < Service

▶ CodeStar < Service

▶ CognitoIdentity < Service

▼ CognitoIdentityServiceProvider

2016-04-18

▶ CognitoSync < Service

▶ Comprehend < Service

▶ ConfigService < Service

▶ CostExplorer < Service

▶ CUR < Service

▶ DataPipeline < Service

▶ DAX < Service

▶ DeviceFarm < Service

▶ DirectConnect < Service

▶ DirectoryService < Service

▶ Discovery < Service

▶ DMS < Service

▶ DynamoDB < Service

`adminConfirmSignUp(params = {}, callback) ⇒ AWS.Request`

Confirms user registration as an admin without using a confirmation code.

`adminCreateUser(params = {}, callback) ⇒ AWS.Request`

Creates a new user in the specified user pool.

`adminDeleteUser(params = {}, callback) ⇒ AWS.Request`

Deletes a user as an administrator.

`adminDeleteUserAttributes(params = {}, callback) ⇒ AWS.Request`

Deletes the user attributes in a user pool as an administrator.

`adminDisableProviderForUser(params = {}, callback) ⇒ AWS.Request`

Disables the user from signing in with the specified external (SAML or social) identity provider.

`adminDisableUser(params = {}, callback) ⇒ AWS.Request`

Disables the specified user as an administrator.

`adminEnableUser(params = {}, callback) ⇒ AWS.Request`

Enables the specified user as an administrator.

`adminForgetDevice(params = {}, callback) ⇒ AWS.Request`

Forgets the device, as an administrator.

`adminGetDevice(params = {}, callback) ⇒ AWS.Request`

Gets the device, as an administrator.

`adminGetUser(params = {}, callback) ⇒ AWS.Request`

Gets the specified user by user name in a user pool as an administrator.

`adminInitiateAuth(params = {}, callback) ⇒ AWS.Request`

Initiates the authentication flow, as an administrator.

<http://amzn.to/2nMu3QN>

# acceptance tests

## Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

App client name

server

Refresh token expiration (days)

30

☐ Generate client secret

☒ Enable sign-in API for server-based authentication (ADMIN\_NO\_SRP\_AUTH) [Learn more.](#)

☐ Only allow Custom Authentication (CUSTOM\_AUTH\_FLOW\_ONLY)

[Set attribute read and write permissions](#)

Cancel

Create app client

we need this option to be disabled as JS AWS SDK doesn't support client secret (yet)

we need this option to be enabled to use the admin API