# **Integrating IFTTT with Lambda + API Gateway**

## Phase 1: Create Platform

Requirements:

1. Custom API for handling IFTTT
2. AWS Lambda functions (get\_user, action, etc.) + API Gateway Trigger
3. Contact Sales for publishing SMC as a searchable service (ex. Phillips Hue)

## Phase 2: Implement OAuth 2.0 Authentication

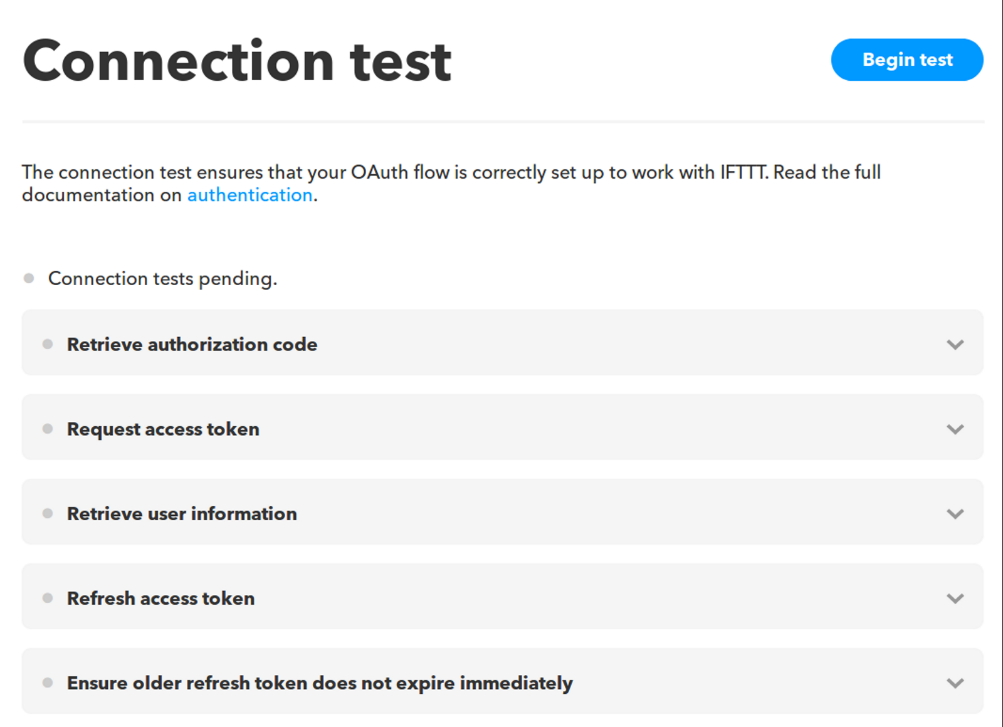
* Use any OAuth 2.0 service (Auth0 for example)
* Required Properties: Client ID, Client secret, Authorization URL, Token URL
* Required Scopes (only for Auth0): openid, profile, offline\_access
  + Default scope is “ifttt”, but can be overridden under Authorization URL
  + Ex: https://seichang00.auth0.com/authorize?scope=profile+openid
* Get User Info
  + GET Requests to API: <https://APIURLHERE/ifttt/v1/user/info>
  + API Path is fixed, must create resource under API Gateway
  + For handling, refer to “UserData” lambda function

Figure 1: Connection test for ensuring correct OAuth flow

## Phase 3: Creating Custom Action

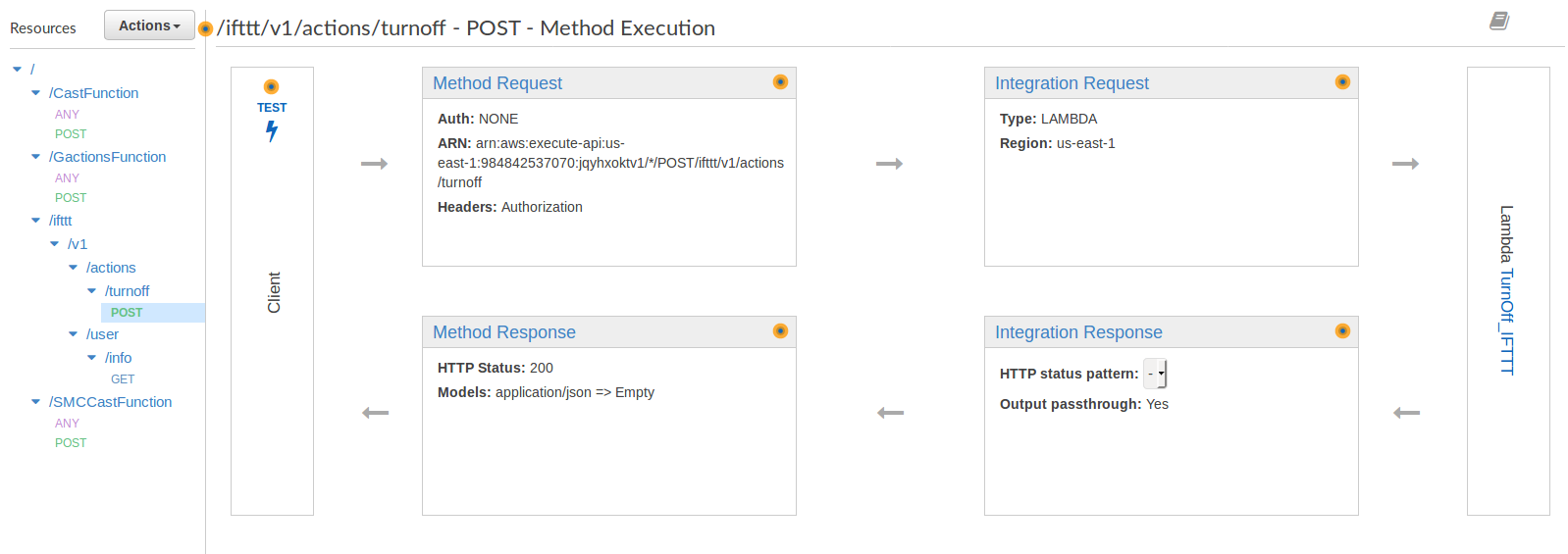
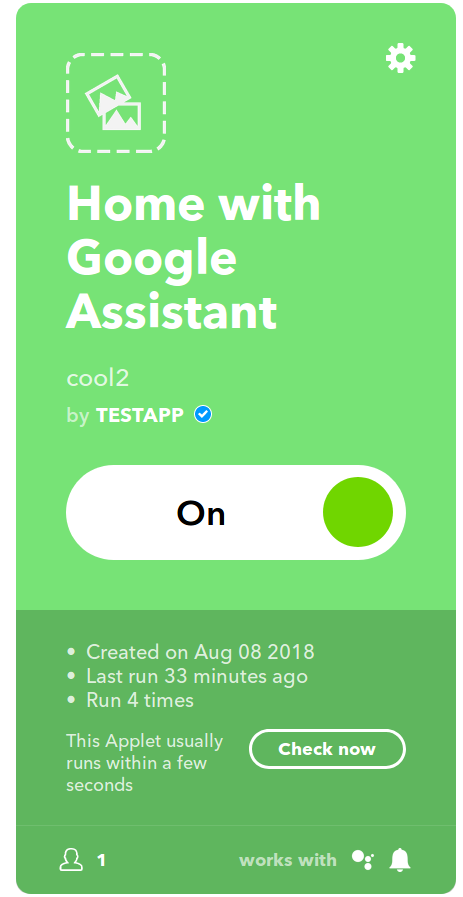
* Create custom action under API
* Sends POST Request to API: <https://APIURLHERE/ifttt/v1/actions/ACTIONNAME>
* “TurnOff\_IFTTT” function (POST Method)
  + Retrieves token from Authorization header
  + Performs GET request to retrieve user profile from Auth0
  + Uses Auth0 UserID to access user data from DynamoDB
  + Updates DynamoDB so cameras are turned off
  + Returns UserID under “*data”*

Figure 2: POST Method of Action through API Gateway



## Phase 4: Creating IFTTT Applet

* Applets involve two aspects: Trigger + Action
* Possible Triggers
  + Location (exiting or entering)
  + Custom Google Assistant phrases (I am home now)
  + Custom Trigger (if necessary)
* Possible Actions
  + Custom Action (refer to Phase 3)
  + Notifications (from IFTTT app)
  + Call/SMS/Email (perhaps emergency)
  + Controlling other smart appliances

Figure 3: Sample IFTTT Applet