

**Intro:** Side Project. Use AWS CDK in TypeScript, Lambda code and web scraping in Python, and UI tooling in Java to webscrape Amazon and allow caching and tracking of Amazon pricing information

### ***Project Structure***

#### *Main Internal APIs - Just Lambdas*

void updateProductPrices()

- updates products tracked by DynamoDB#1 if last updated over an week ago, configurable
- Python script that webscrapes Amazon with Selenium

void sendScheduledMessage()

- Scheduled to run weekly
- Checks DynamoDB#2 and sends out emails to all users attached to SES
- Calls updateProductPrices then getTrackedPrices

#### *Main External APIs - APIs as well*

void updateUserDB(username=, password=, \*add\_track=, \*remove\_track=, \*email=)

- Update DynamoDB#2 for added and removed tracking links and emails
- Add new username/password hash to DynamoDB#2 if needed
- Runs updateProductPrices() if any new tracking links added
- PUT Request

table getTrackedPrices(\*username=, \*password=, \*userhash=)

- Pull info from DynamoDB#2 for what is being tracked
- Pull info for prices from DynamoDB#1
- Make a version that accepts just hash value as well as an optional hidden value
- GET Request

#### *AWS Infrastructure*

- *AWS/DynamoDB*
  - DynamoDB #1
    - Only need to update if last updated over hour ago
    - Master Product Tracking Database
      - Key isProduct Link
      - Product Name

- Original Price
  - Current Price
  - Historic Low
  - Last Updated (Web Cache)
- DynamoDB #2
  - User Product Tracking Database
    - Key = Hash(Hash(User) | Hash>Password))
    - Links User Is Tracking
    - Emails To Send Tracked Info To (Also keep encrypted in table)
- *AWS/IAM-API Gateway-Lambda-SES*
  - APIs
  - Role to call external APIs via HTTP Queries
  - Role to call all internal APIs, limited perms
  - SES for emails
- *AWS/S3-CloudFormation-CDK*
  - Setup/Bootstrap for all the AWS infrastructure in TypeScript
- *AWS/CloudWatch*
  - Logs for Lambdas

#### *User Interface - Java*

- Java JFrame for UI
  - Allow Selecting Products To Track via URL
    - Adds just link to table and keeps added links in array until refresh done
  - Allow Deselecting Products via Delete Button
    - Delete just removes from table and keeps deleted links in array until refresh done
  - Allow Email Updating
    - Also leaves value until refresh
  - Pricing Information On Selected Products
    - Keeps obtained information cached until refresh
- Login Information
  - Parameters On Program Start
- Refresh values on Ctrl-R
  - Check/Update DynamoDB/JTable on Refresh

- Run on Start As Well

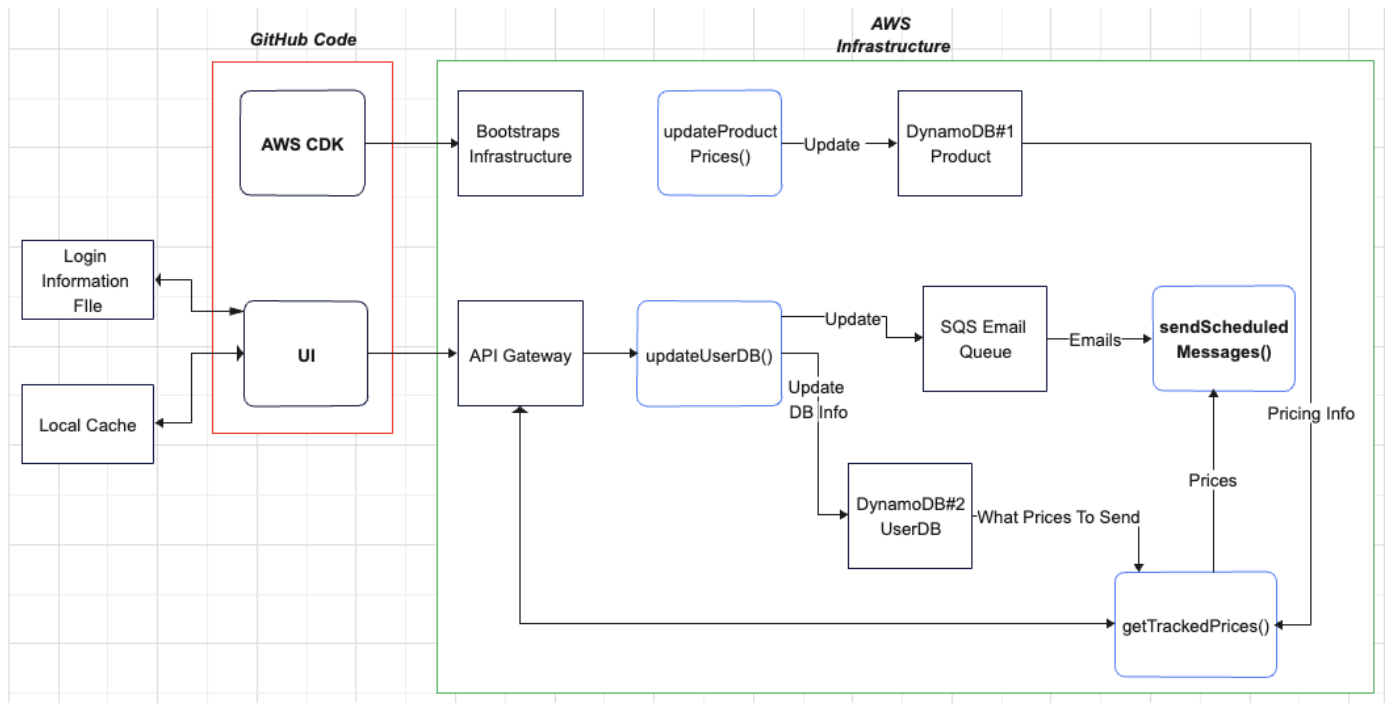
### *Testing*

- Proper Integration Tests for all APIs

### *Building*

- Use Maven to build Java app
- Use CDK to build and deploy CDK app

## Architecture Diagram



**Bolded** are architecture starting points. **Blue** are APIs/Lambdas.

## UI Mock

Note:  
Ctrl-R to Refresh Product Tracking Info Or  
Will Refresh Every 5 Minutes

Amazon Price Tracker - Username: {}

| #  | Product Name | Orig Price | Curr Price | Hist Low | Del |
|----|--------------|------------|------------|----------|-----|
| 1  | Product Name | Price      | Price      | Price    |     |
| 2  | Product Link |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
|    |              |            |            |          |     |
| 1u | 6u           | 2u         | 2u         | 2u       | <1u |



Product:

Track

Email:

Update

## *Timeline*

- 1) Create API Plan 
- 2) Create Architecture Diagrams 
- 3) Create GitHub Repo 
- 4) Web Scraping Script for Amazon Python 
- 5) Create AWS Resources (IAM, S3, CloudFormation, CloudWatch, Lambda, DynamoDB, SES, API Gateway) with AWS CDK - TypeScript
  - a) Bootstrap Stack (S3, CloudFormation) 
  - b) IAM Roles Stack (Internal Role and External Role) 
  - c) Database Stack (DynamoDB) 
  - d) API Stack (API Gateway, Lambda, SES) 
    - i) Blank Lambdas set up
    - ii) Setup actual lambda functions
      - (1) updateUserDB
      - (2) getTrackedPrices
      - (3) sendScheduledMessages
      - (4) updateProductPrices
- 6) Create UI to interact with APIs - Java **Oct 22**
- 7) Api Documentation and Integration Tests