Interview Presentation

Shivika Sodhi

About Me

- Born in Delhi (capital of India).

- I like playing board games, baking, hiking and skiing.

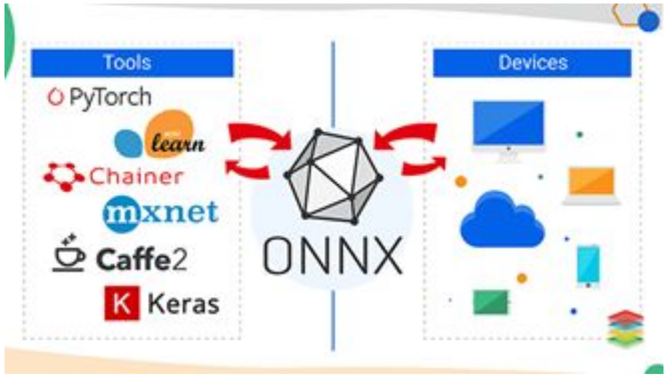
- Is also a proud aunt of two!

About Me

- 3 years in software industry
  - 1 year at Infosys India
  - 2 years at Intel, USA

- Currently working on WindowsML and ONNX analysis in collaboration with MSFT

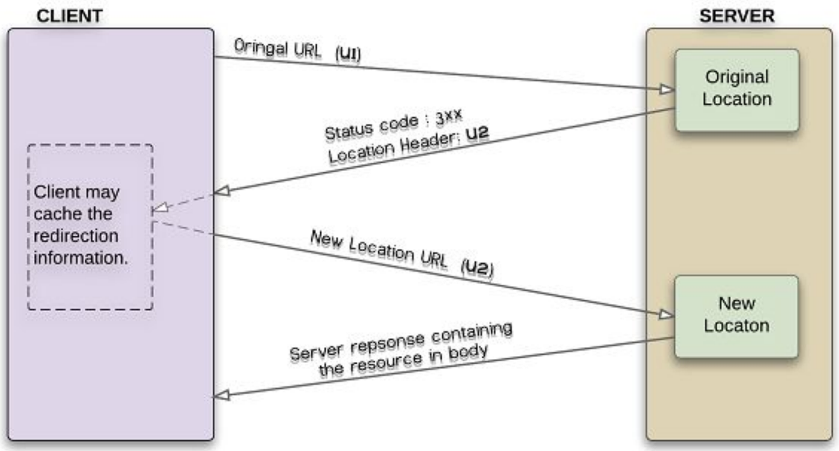
- Did bachelors and masters in computer science



URL Redirection Service

Objective: Build a service that redirects users from the website's home page to their last activity page (maintained by sessions in the DB).

My Role: I was the project owner. I designed, implemented and launched this service independently.



URL Redirection Service

Outcome:

The service is up and running in production. Used by ~40,000 employees on an average of ~10 times per day.

What did I learn?

Java, Rest API, Microservice architecture, OSGI (Open Source Gateway Initiative), Apache ACE, JBoss, SQL, working with production (testing, deployment etc)

Why I selected this achievement ?

- First ever project as a Software Engineer at Intel!

- Great feeling to see your product being used by co-workers.

- Lots of learning !!

Online Document Collaborating System

Objective: To allow architects at intel to share and modify SPEC documents online.

My Role:

- Part of 3 people team.
- I Designed the overall project and implemented a quick POC.
- I implemented the frontend and REST API's in the final product.

Problem:

- Architects need to collaborate on shared docs.
- Docs need special structure (Google Docs etc can't provide).
- Docs should be backward compatible with MS Word

Online Document Collaborating System

Outcome:

The module is currently live in production and is being used by various teams within Intel (~50 teams).

What did I learn:

System design , Rest API, Angular, DB (SQL), Microservices, Java, dealing with permissions

Why this achievement?

- High impact
  - Major source of inconvenience at Intel (people used to coordinate over email)
- My first major design
  - I designed the entire system end-to-end
- Challenging problem
  - Keeping docs in sync, conflict resolving etc.
- Learnt System Design
  - Flexible architecture to support new feature requests.

Case Study Recap

Goal: Design a Stock Financial Service

End user: See your stocks and trends. See your gains/loss per day. See notification if a stock has been the price for the past X days.

Front End

- Show the top stock and able to let users add/remove to their profile.
- Show trend of the stocks and ability to refresh.

Back End:

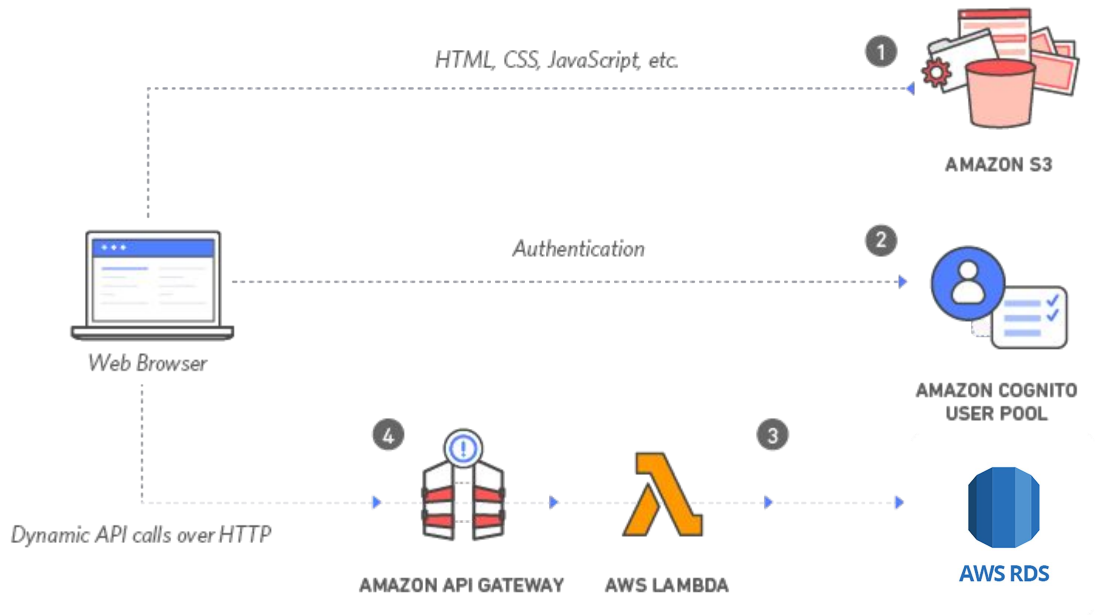
- Create a Rest or GraphQL API with the correct end points needed for the UI with a persistent database.
- Future proofing

V1 Design

Features supported:

- Securely Login
- Buy/Sell stocks (in a quantity)
- See price trends for a stock
- See total profit/loss
- See top and lowest performing stocks
- Get notifications if the stock price has been same for X days

Architecture Diagram



Working Demo

LINK : http://stockfin-storage.s3-us-west-2.amazonaws.com/index.html?

- Nothing is mocked
- Some of the cases have been simplified
- Not all the features mentioned in V1, have been implemented

Serverless Architecture

- ➢ Reduced Setup Cost
- ➢ No need to worry about scaling or load balancing

Alternatives

- EC2 beanstalk
- EC2
- CloudFront

REST API's

- REST vs Graph ?
  - Decided to use REST
    - Complexity of the application
      - This application has straightforward interactions with the server, so extra flexibility of GraphQl, might not provide value.
    - Market Support
      - REST API's are used lot more
    - Familiarity

- API's
  - stockfin/v1/users
    - Manage users
  - stockfin/v1/users/stocks
    - Manage stocks of users
  - stockfin/v1/stocks
    - Manage stocks

REST API’s

stockfin/v1/stocks

Instead of storing the data for all the stocks in our database, we are getting this data on the fly from [Markit On Demand API’s]

`http://dev.markitondemand.com/Api/v2/Quote`

REST API's

stockfin/v1/users/

POST

- Create new User

GET

- Fetch a User
- Returns profit, top stocks etc

Put

- Update User details
- Add Money, change profile

DELETE

- Delete a User

REST API's

stockfin/v1/users/stocks

POST
- Buy stock

GET
- Get profit/loss, or past transactions for a particular stock

DELETE
- Sell stock

Database

MySQL vs NoSQL

- Financial data is transactional, need to be atomic and support strong consistency (ACID)
- Stocks data has a lot of structure to it, which MySQL can leverage for lower latency
- *storing time-series data at scale is a challenge
    - Data can be easily sharded (with user, stock etc), elevating scalability concerns.
    - My current design does not store any time-series data.

MySQL Database

User		
idUser	username	cash

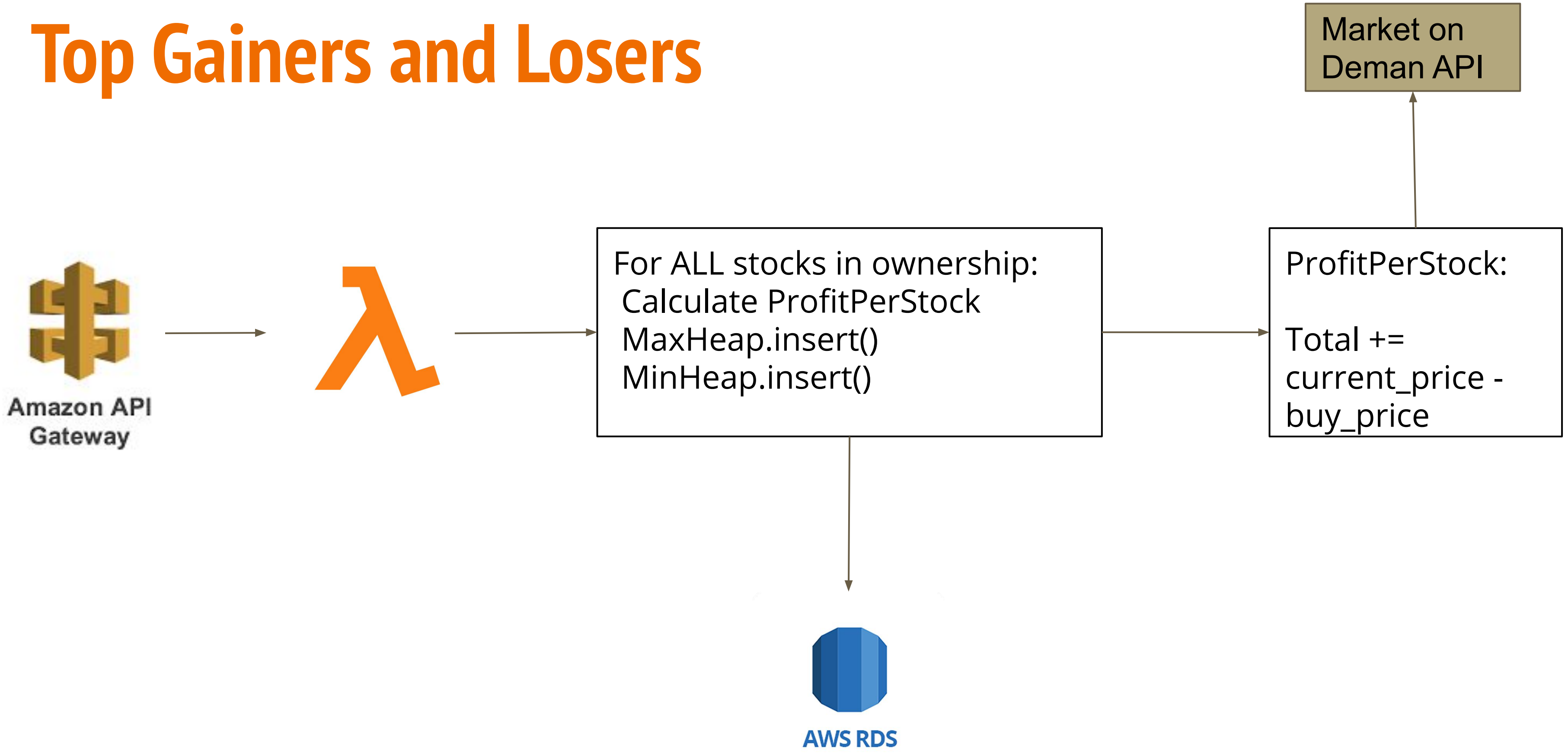
Stock	
idStock	ticker

Favourites		
idFav	idUser	idStock

Ownership				
idOwn	isUser	idStock	price	qty

History					
idHist	isUser	idStock	price	qty	op

Top Gainers and Losers

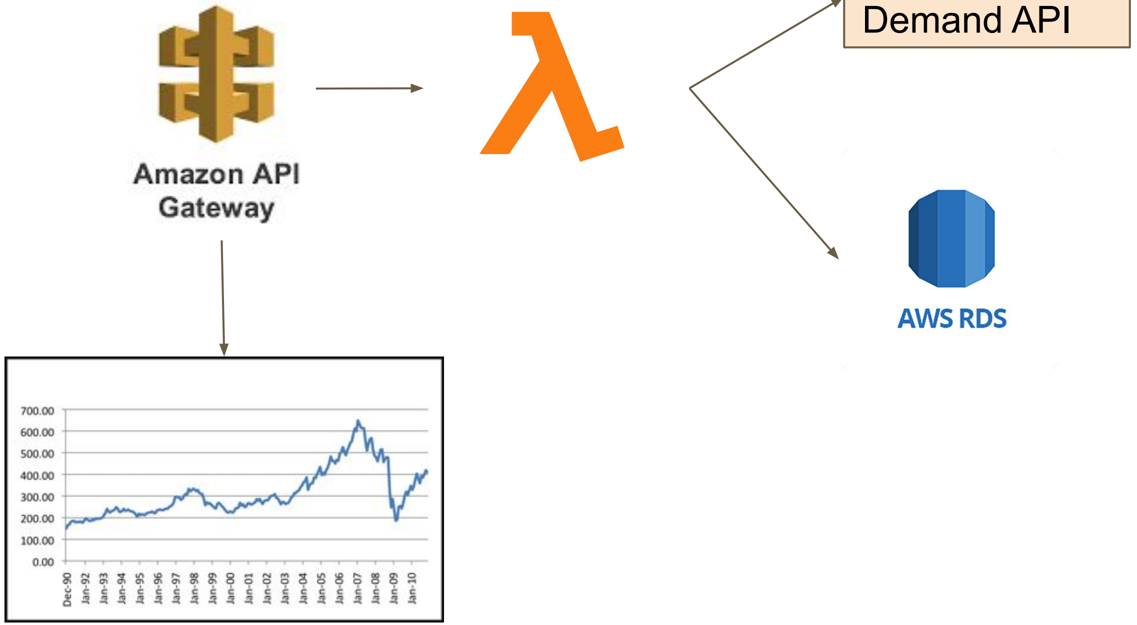


```
For ALL stocks in ownership:
 Calculate ProfitPerStock
 MaxHeap.insert()
 MinHeap.insert()
```

```
ProfitPerStock:

Total +=
current_price -
buy_price
```

Trends of Profit/Loss

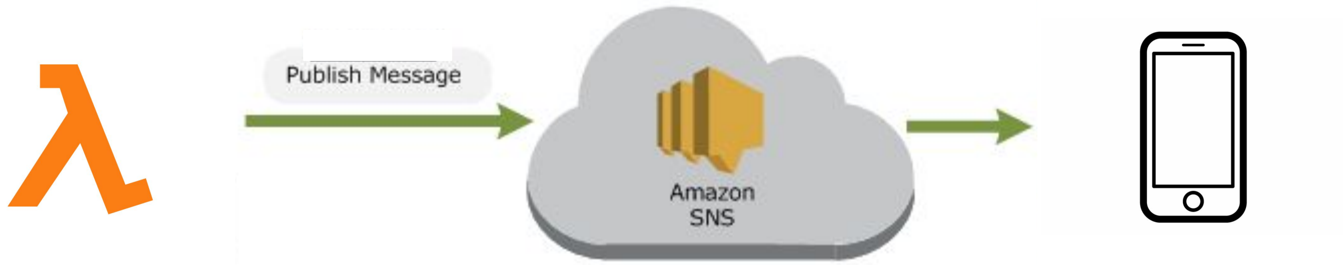


- For all stocks in the history table of the user, fetch stock price history from MarketOnDemand

- Use the purchase price in DB and price history from Market to compute time-series of profit/loss.

Notifications

Requirement: Notify the user if stock has been the same price for last Y days.

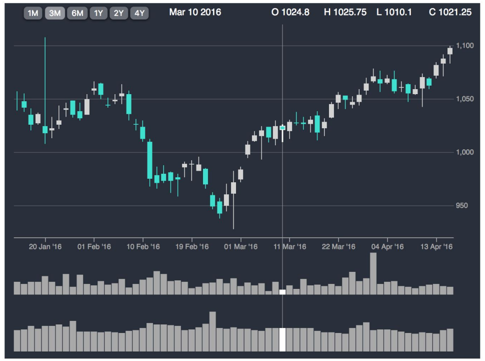


Scheduled Lambda,
Runs every day.
Check if the stock
price is same for last
Y days.

Frontend

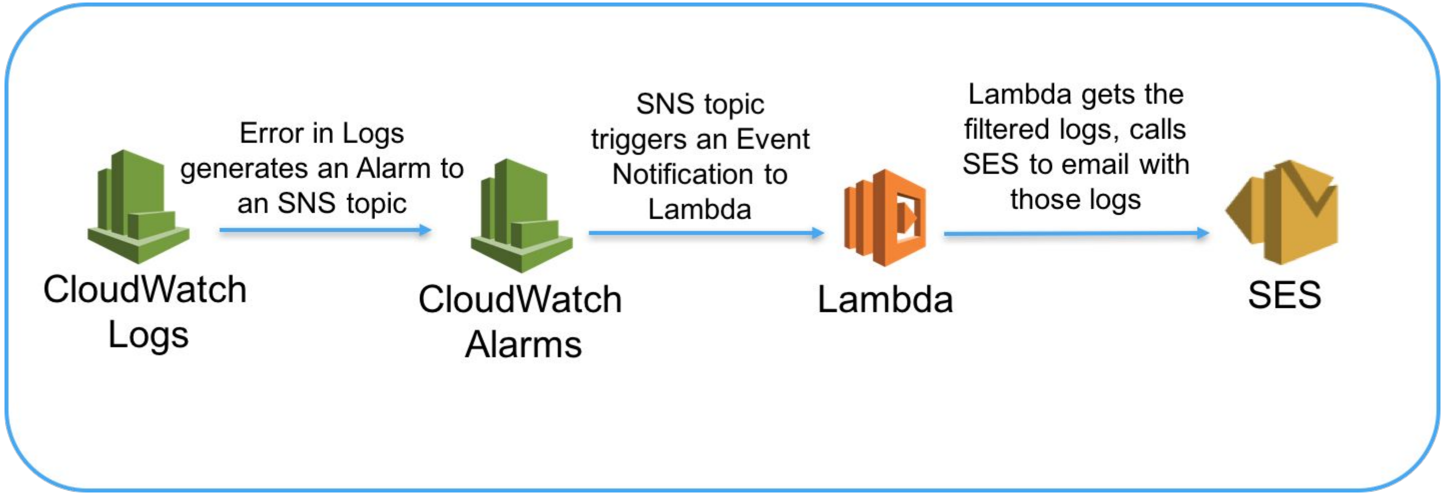
- Libraries to display Candlestick charts (Highcharts, D3.js etc.)

- React JS : To refresh the data without manually reloading the page.
    - Optimal for fetching rapidly changing data



Monitoring & Alerting

- Use SNS (and cloudwatch) to alert on key metrics
  - Error rate
  - Latency


CloudWatch Logs
Error in Logs generates an Alarm to an SNS topic
CloudWatch Alarms
SNS topic triggers an Event Notification to Lambda
Lambda
Lambda gets the filtered logs, calls SES to email with those logs
SES

Possible V2 Features

- Star stocks
- View your past transactions (Date ranges)
- Set Automatic buy/sells
- Get Recommendations
    - Similar stocks to track (people who buy this stock also buy these one, collaborative filtering)
- Customer support
    - automated support, phone call support etc

Challenges

- Encountered while building the prototype
  - CORS, CORB
  - VPC settings
  - Connecting local systems to DB