

# Shopping Website

---

Team: Cloud Players  
Juntian Sun: js5940  
Zijian Zhang: zz2994  
Yi Zhang: yz4339  
Yifan Wang: yw3914  
Lihui Yan: ly2593

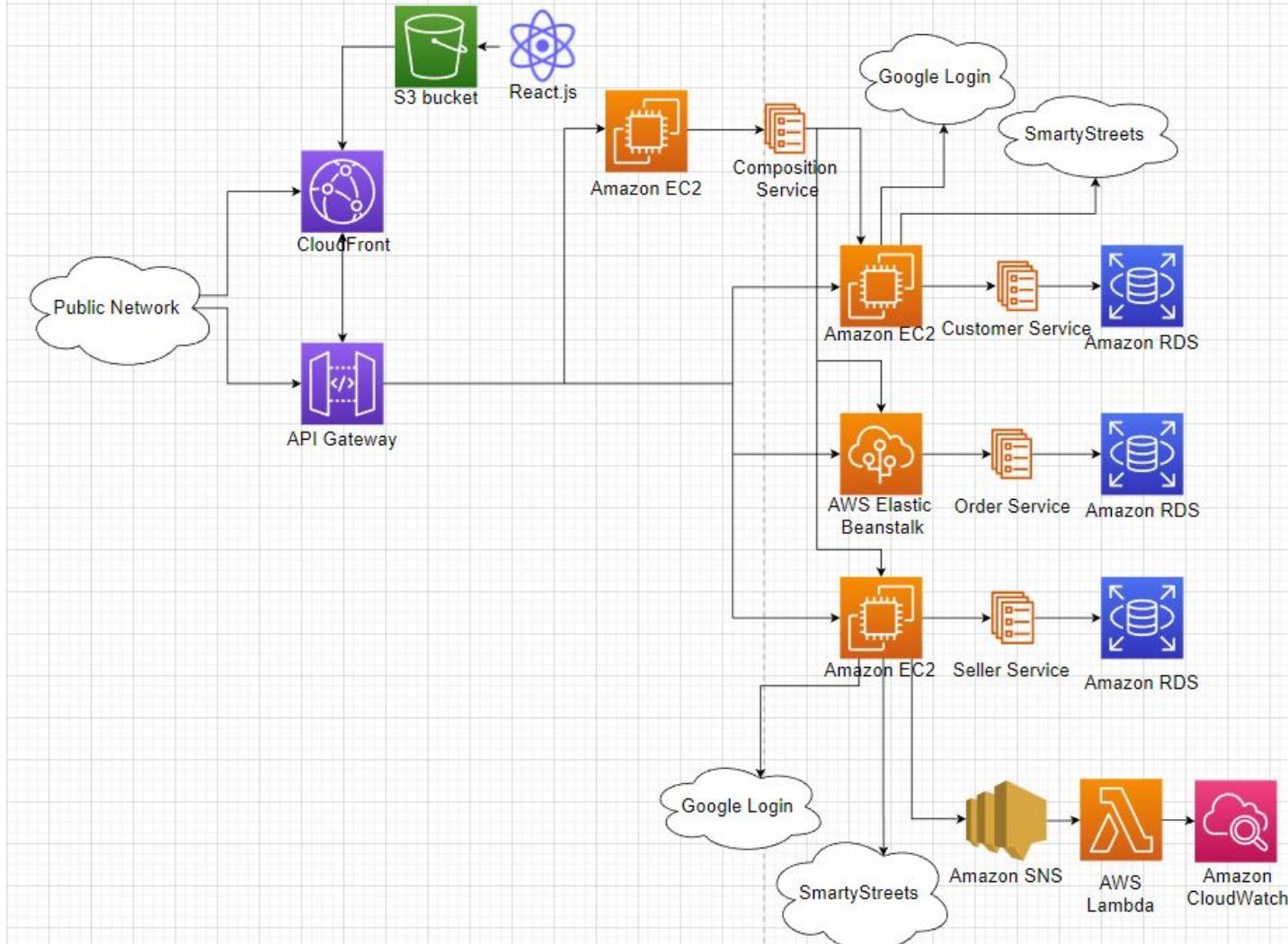
# UI demo

Github for frontend <https://github.com/sqwy/FullStackReact.git>

# Our project

- Customer Service
- Seller Service
- Order service
- Composite Service

# Our Project: Architecture



# CloudFront

The screenshot shows the AWS CloudFront console interface. At the top, there's a navigation bar with 'View metrics' and tabs for 'General', 'Origins', 'Behaviors', 'Error pages', 'Geographic restrictions', 'Invalidations', and 'Tags'. Below this, the main content area has a title 'EC281AQHOOAMF' and a 'Details' section. The 'Origins' tab is selected in the left sidebar, which also includes 'General' and 'Behaviors'.

**Details**

Distribution domain name	ARN	Last modified
d14ohravlws9nk.cloudfront.net	arn:aws:cloudfront::703007965469:distribution/E C281AQHOOAMF	December 21, 2022 at 6:21:31 PM UTC

**Origins**

Filter origins by property or value

Origin name	Origin domain	Origin path	Origin type	Origin Shield reg...
react0001.s3-websit...	react0001.s3-website-us-east-1.amazonaws.com		Custom Origin	-

Link for cloudfont: <https://d14ohravlws9nk.cloudfront.net>

# CloudFront Behaviors

CloudFront > Distributions > EC281AQHOOAMF

## EC281AQHOOAMF

[View metrics](#)

General

Origins

Behaviors

Error pages

Geographic restrictions

Invalidations

Tags

### Behaviors

[Save](#)

[Move up](#)

[Move down](#)

[Edit](#)

[Delete](#)

[Create behavior](#)

Filter behaviors by property or value



Preced...	Path pattern	Origin or origin group	Viewer protocol policy	Cache policy name	Origin request policy n...	Response headers polic...
0	/index.html	react0001.s3-website-us...	HTTP and HTTPS	Managed-CachingOptimized	-	-
1	Default (*)	react0001.s3-website-us...	HTTP and HTTPS	Managed-CachingOptimized	-	-

# S3 bucket

Amazon S3 > Buckets > react0001

## react0001 Info

Publicly accessible

Objects Properties Permissions Metrics Management Access Points

### Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Actions ▾

Find objects by prefix < 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	asset-manifest.json	json	December 22, 2022, 21:07:00 (UTC-05:00)	369.0 B	Standard
<input type="checkbox"/>	index.html	html	December 22, 2022, 21:07:00 (UTC-05:00)	233.0 B	Standard
<input type="checkbox"/>	static/	Folder	-	-	-

# S3 static website hosting

## Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Edit

Static website hosting

Enabled

Hosting type

Bucket hosting

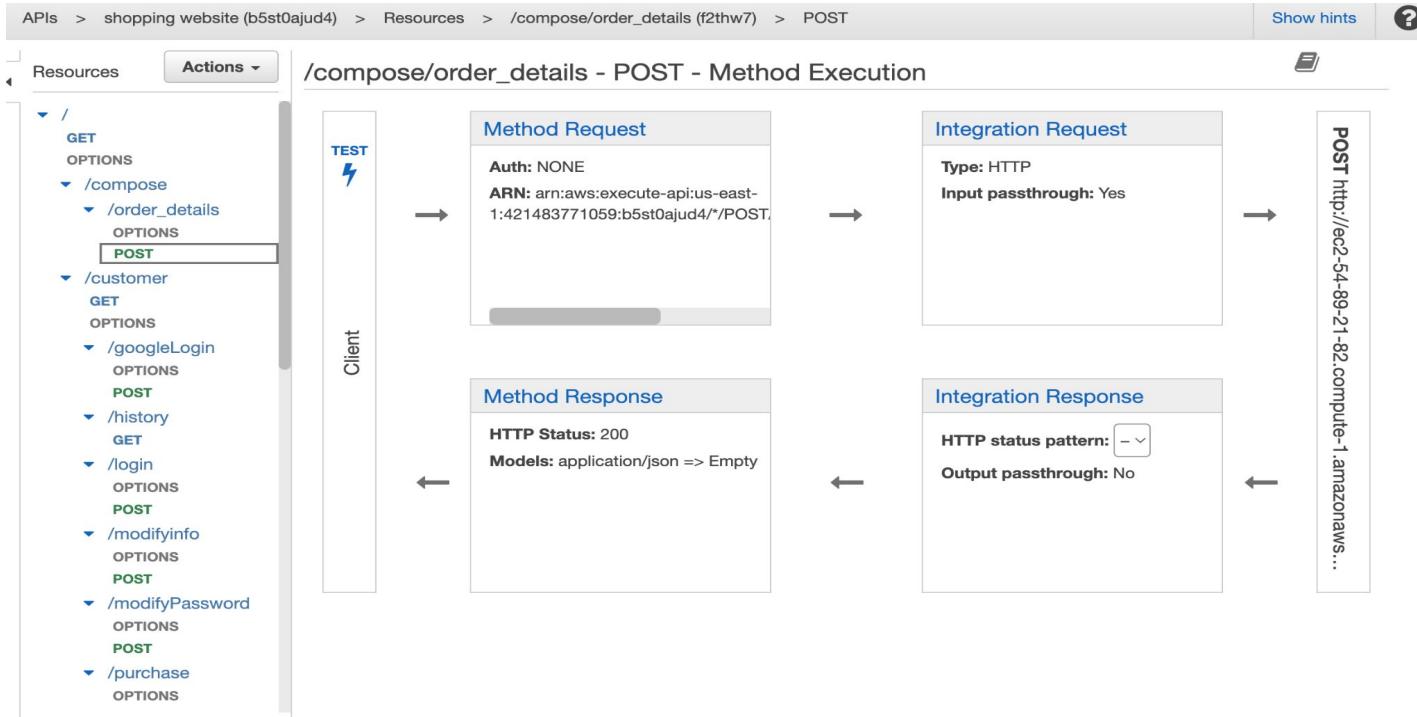
Bucket website endpoint

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

 <http://react0001.s3-website-us-east-1.amazonaws.com> 

Link for S3: <http://react0001.s3-website-us-east-1.amazonaws.com>

# API Gateway



Link for API Gateway <https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping>

# Customer service

---

# Customer Service Overview

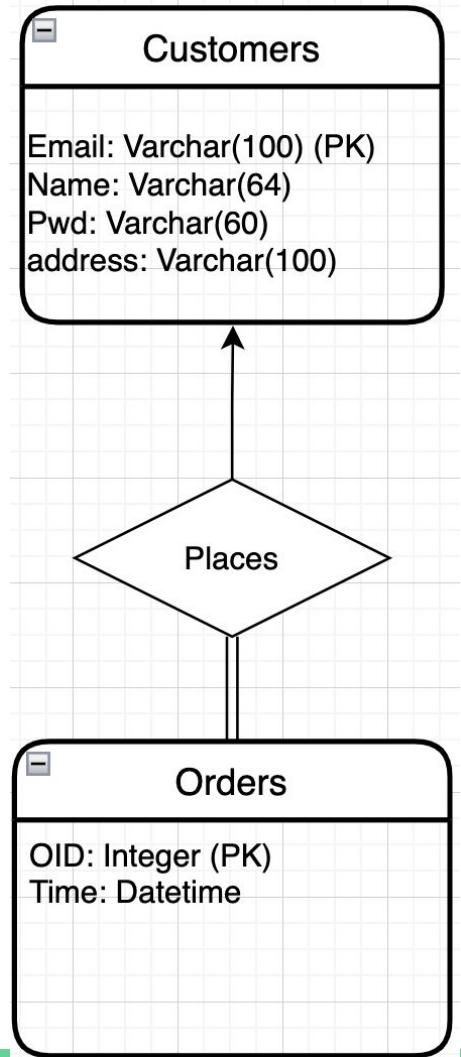
- Accept and respond with JSON
- Google login
- Connect a cloud database and browser application
- Interact with cloud database
- Smarty.com to validate and correct mailing addresses
- Middleware to check before signing in and after registering successfully
- Notification with SNS and Lambda functions after registering
- Allows pagination

```
    POST  
    ▼ /customer  
        GET  
        OPTIONS  
        □ /googleLogin  
            OPTIONS  
            POST  
    ▼ /history  
        GET  
        ▼ /{proxy+}  
            OPTIONS  
            POST  
    ▼ /login  
        OPTIONS  
        POST  
    ▼ /modifyinfo  
        OPTIONS  
        POST  
    ▼ /modifyPassword  
        OPTIONS  
        POST  
    ▼ /purchase  
        OPTIONS  
        POST
```

# Customer Service ER Diagram And Schema

Github for Customer

[https://github.com/ywang289/  
customer.git](https://github.com/ywang289/customer.git)



A hierarchical tree structure representing the database schema:

- customer
  - tables 3
    - Customers
      - columns 4
        - Email varchar(100)
        - Name varchar(64)
        - Pwd varchar(20)
        - address varchar(100)
      - keys 1
      - indexes 1
    - Orders
      - columns 2
        - OID int
        - Time datetime
      - keys 1
      - indexes 1
    - Places
      - columns 2
        - Email varchar(100)
        - OID int
      - keys 1
      - foreign keys 2
      - indexes 2

# Google Login Oauth

The image shows two browser windows illustrating the Google OAuth login process.

**Left Window:** A Microsoft Edge window titled "Sign in - Google Accounts - Personal - Microsoft Edge". The URL is [https://accounts.google.com/o/oauth2/auth/identifier?redirect\\_uri=storagerelay...](https://accounts.google.com/o/oauth2/auth/identifier?redirect_uri=storagerelay...). The page displays the "Sign in with Google" logo and a "Sign in" form. The email input field contains "js5940@columbia.edu". Below the form, a message states: "To continue, Google will share your name, email address, language preference, and profile picture with d14ohravlws9nk.cloudfront.net.". At the bottom are "Create account" and "Next" buttons.

**Right Window:** A Microsoft Edge window showing the successful login to a web application. The URL is <https://d14ohravlws9nk.cloudfront.net/customer>. The page header includes "Shopping" and a back button. The main content area displays a welcome message: "Hello, Juntian Sun!". It also features links for "Search new products!", "Your Cart", and "history". Above the content area, a log message in the developer tools console reads:

```
successfully  
14.212.182.221 - - [21/Dec/2022 18:23:36] "POST /customer/googleLogin HTTP/1.1" 200 -  
successfully
```

# EC2

EC2 > Instances > i-0c379753324eff1f8

## Instance summary for i-0c379753324eff1f8 (customer) [Info](#)

Updated less than a minute ago

Instance ID  
[i-0c379753324eff1f8 \(customer\)](#)

Public IPv4 address  
[44.201.86.144 | open address](#)

Private IPv4 addresses  
[172.31.93.123](#)

IPv6 address

Instance state

[Running](#)

Public IPv4 DNS

[ec2-44-201-86-144.compute-1.amazonaws.com | open address](#)

Hostname type  
IP name: ip-172-31-93-123.ec2.internal

Private IP DNS name (IPv4 only)  
[ip-172-31-93-123.ec2.internal](#)

Elastic IP addresses  
-

Answer private resource DNS name  
IPv4 (A)

Instance type

t2.micro

AWS Compute Optimizer finding

[Opt-in to AWS Compute Optimizer for recommendations.](#)  
[Learn more](#)

IAM Role

Subnet ID

[subnet-018fed174e0ff90bb](#)

Auto Scaling Group name  
-

[Details](#) | [Security](#) | [Networking](#) | [Storage](#) | [Status checks](#) | [Monitoring](#) | [Tags](#)

### ▼ Instance details [Info](#)

Platform

[Amazon Linux \(Inferred\)](#)

AMI ID

[ami-09d3b3274b6c5d4aa](#)

Monitoring

disabled

Platform details

[Linux/UNIX](#)

AMI name

[amzn2-ami-kernel-5.10-hvm-2.0.20221004.0-](#)

Termination protection

Disabled

aws

Services

Search

[Option+S]

Last login: Wed Dec 21 04:00:22 2022 from ec2-18-206-107-28.compute-1.amazonaws.com

Amazon Linux 2 AMI

<https://aws.amazon.com/amazon-linux-2/>  
34 package(s) needed for security, out of 53 available  
Run "sudo yum update" to apply all updates.  
ec2-user@ip-172-31-93-123 ~]\$ ls  
156\_server customer falsk  
ec2-user@ip-172-31-93-123 ~]\$ cd customer  
ec2-user@ip-172-31-93-123 customer]\$ python3 customer.py  
\* Serving Flask app 'customer'  
\* Debug mode: on

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server.**  
\* Running on all addresses (0.0.0.0)  
\* Running on http://127.0.0.1:8080  
\* Running on http://172.31.93.123:8080  
Press CTRL+C to quit  
\* Restarting with stat  
\* Debugger is active!  
\* Debugger PIN: 829-398-309  
successfully

14.206.4.95 - - [21/Dec/2022 18:22:38] "POST /customer/googleLogin HTTP/1.1" 200 -  
successfully

14.212.182.221 - - [21/Dec/2022 18:23:36] "POST /customer/googleLogin HTTP/1.1" 200 -  
successfully

14.206.6.66 - - [21/Dec/2022 18:23:44] "POST /customer/googleLogin HTTP/1.1" 200 -  
successfully

3.216.142.72 - - [21/Dec/2022 18:24:01] "POST /customer/googleLogin HTTP/1.1" 200 -  
successfully

14.206.4.109 - - [21/Dec/2022 18:27:42] "POST /customer/googleLogin HTTP/1.1" 200 -

i-0c379753324eff1f8 (customer)

Public IPs: 44.201.86.144 Private IPs: 172.31.93.123

Customer service: <https://ec2-44-201-86-144.compute-1.amazonaws.com>

# Middleware

- Check email and password before logging in
- Check address before registration
- Send email to manager after user registration successfully

```
@app.before_request
def run():
    if request.path == '/customer/register':
        auth_id = "c832f79c-cc29-3a15-c118-51733a6b5929"
        auth_token = "CoGbY7DbR8qNdUPu3aZh"
        data = json.loads(request.get_data())
        address = data['address']
        zipcode= data["zipcode"]

        credentials = StaticCredentials(auth_id, auth_token)

        client = ClientBuilder(credentials).build_us_street_api_client()

        lookup = Lookup()

        lookup.street = address
        lookup.zipcode = zipcode

    try:
        client.send_lookup(lookup)
    except exceptions.SmartyException as err:
        print(err)
        return {"message": "input error", "state": False}

    result = lookup.result

    if not result:
        print("No candidates. This means the address is not valid.")
        return {"message": "invalid address", "state": False}

@app.after_request
def af3(response):
    if request.path == '/customer/register':
        # print(request.environ.get('SERVER_PROTOCOL'))
        # print(len((response.data.decode('utf-8')).split(":")))
        # print(json.loads(response.get_data())["state"])

You, 1小时前 • Uncommitted changes
```

# Login Authentication

```
@app.before_request
def check_login():
    if request.path == '/seller/login':
        data = json.loads(request.get_data())
        password = data['password']
        email = data['email']

        try:
            sql = "SELECT * FROM Sellers where email = '{}' ".format(email)
            result = db.session.execute(sql).fetchone()
        except Exception as err:
            return {"state": False, "message": "error! input error"}
        if result:
            stored_password= result[2]
            if stored_password != password:

                print("unmatch")
                return {"state": False,"message":"password unmatch"}

        else:
            return {"state": False, "message": "please register"}
```

# Customer service: Address Verification with SmartyStreets

The image shows two side-by-side Postman request panels. Both requests are to the same endpoint: `https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping/customer/register`. The first request is for a user with address "400W". The second request is for a user with address "400w 113th". The responses are different based on the address format.

**Request 1 (Left):**

- Method: POST
- URL: `https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping/customer/register`
- Body (JSON):

```
{"username": "ywang", "password": "0002", "email": "wg@gmail.com", "address": "400W", "zipcode": "10000"}
```

**Request 2 (Right):**

- Method: POST
- URL: `https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping/customer/register`
- Body (JSON):

```
{"username": "ywang", "password": "0002", "email": "ssoh@gmail.com", "address": "400w 113th", "zipcode": "10025"}
```

**Responses:**

**Request 1 Response (Left):**

- Status: 200 OK
- Body (Pretty JSON):

```
{  "message": "register successfully",  "state": true}
```

**Request 2 Response (Right):**

- Status: 200 OK
- Body (Pretty JSON):

```
{  "message": "invalid address",  "state": false}
```

# Customer service: Notification with SNS and Lambda functions

Hello External Inbox x



email no-reply@sns.amazonaws.com via amazonses.com  
to me ▾

a new user  
ywang register successfully

--  
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:

<https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsns.us-east-1.amazonaws.com%2Funsubscribe.html%3FSubscriptionArn%3Dam%3Aaws%3Asns%3Aus-east-1%3A421483771059%3Aemail%3A3d50f2a8-f5c6-4739-873c-ea58ba0e9408%26Endpoint%3Dywang289%40buffalo.edu&data=05%7C01%7Cywang289%40g-mail.buffalo.edu%7Cec7e1c175f6b4dfcecc1f08dae388c6f6%7C96464a8af8ed40b199e25fb50a20250%7C0%7C638072473833503597%7CUnknown%7CTWFPbGZsb3d8eyJWjoiMC4wLjAwMDA1LCJQjoIV2lMzIiLCJBTiI6lk1haWwlCJXVCi6Mn0%3D%7C3000%7C%7C%7C&sdata=ltHoxvbaWn79pRY%2Bani6RT8MYQj1tvtYT0ky3YIAU%3D&reserved=0>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at  
<https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Faws.amazon.com%2Fsupport&data=05%7C01%7Cywang289%40g-mail.buffalo.edu%7Cec7e1c175f6b4dfcecc1f08dae388c6f6%7C96464a8af8ed40b199e25fb50a20250%7C0%7C638072473833503597%7CUnknown%7CTWFPbGZsb3d8eyJWjoiMC4wLjAwMDA1LCJQjoIV2lMzIiLCJBTiI6lk1haWwlCJXVCi6Mn0%3D%7C3000%7C%7C%7C&sdata=yE%2FSkrBqxR2i3NdFa04Cbaz0NkJeAG1YljbjpmhHXJY%3D&reserved=0>

The screenshot shows the AWS Lambda console interface for a function named 'infoname'. The function has two layers listed under 'Layers' (0). An 'API Gateway' trigger is present with '(2)' triggers. A button '+ Add destination' is available. The 'Code source' tab is selected, showing the following code in 'lambda\_function.py':

```
1 import boto3
2
3 def lambda_handler(event, context):
4     # TODO implement
5
6     client = boto3.client('sns')
7     username = event["username"]
```

# Pagination

POST https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping/customer/history/1 APIs > shopping website (b5st0ajud4) > Resources > /customer/history/{proxy+} (qkm879) > POST Show hints ?

Authorization Headers (1) Body Pre-request Script Tests

Body (1) form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {"email": "wg@gmail.com"}
```

Resources Actions /

- GET
- OPTIONS
- /compose
- /order\_details
- OPTIONS
- POST
- /customer
- GET
- OPTIONS
- /googleLogin
- OPTIONS
- POST
- /history
- GET
- {proxy+}
- OPTIONS
- POST
- /login
- OPTIONS
- POST
- /modifyinfo
- OPTIONS
- POST

/customer/history/{proxy+} - POST - Method Execution

Client

Method Request  
Auth: NONE  
ARN: arn:aws:execute-api:us-east-1:421483771059:b5st0ajud4/\*/POST

Integration Request  
Type: HTTP\_PROXY  
Paths: proxy

Method Response  
HTTP Status: Proxy  
Models: application/json => Empty

Integration Response  
Proxy integrations cannot be configured to transform responses.

ANY http://ec2-44-201-86-144.compute-1.amazonaws...

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```
1 {  
2   "history": [  
3     [  
4       2,  
5       "Sun, 11 Dec 2022 17:20:00 GMT"  
6     ],  
7     [  
8       3,  
9       "Sun, 11 Dec 2022 17:23:00 GMT"  
10    ],  
11    [  
12      5,  
13      "Sun, 11 Dec 2022 17:30:00 GMT"  
14    ]  
15  ]  
}
```

# Customer Service: SwaggerHub Verified

Aa ⚡ SAVE SYNC

```
1 openapi: 3.0.0
2 info:
3   version: '1.0'
4   title: test0001
5   description: ''
6 paths:
7   /customer/login:
8     post:
9       summary: Login for a customer
10      description: If successfully login, it will return the address
11        and the username of the customer
12      parameters:
13        - name: email
14          in: query
15          description: Email of the login customer
16          schema:
17            type: string
18        - name: password
19          in: query
20          description: Password of the login customer
21          schema:
22            type: string
23      responses:
24        '200':
25          description: Successfully login
26          content:
27            application/json:
28              schema:
29                type: array
30                items:
31                  $ref: '#/components/schemas/userlogin'
32        '404':
33          description: path not found
```

Last Saved: 2:45:47 pm - Dec 21, 2022 VALID

Code Details

200 Response headers  
`content-length: 20`  
`content-type: application/json; charset=utf-8`

Request duration  
`300 ms`

Responses

Code	Description	Links
200	Successfully login	No links

Media type  
`application/json`

Controls `Accept` header.

Example Value | Schema

```
[ { "state": true, "username": "Juntian Sun", "address": "788 Columbus Ave", "message": "login successfully" } ]
```

POST /customer/login Login for a customer

If successfully login, it will return the address and the username of the customer

Parameters

Name Description

email Email of the login customer  
`test@gmail.com`

password Password of the login customer  
`test`

Cancel

# Seller service

---

# Seller Service Overview

- Accept and respond with JSON
- Google login
- Connect a cloud database and browser application
- Interact with cloud database
- Smarty.com to validate and correct mailing addresses
- Middleware to check before signing in and after registering successfully
- Notification with SNS and Lambda functions after registering
- Allows pagination

```
▼ /seller
  GET
  ▼ /delete_item
    OPTIONS
    POST
  ▼ /googleLogin
    OPTIONS
    POST
  ▼ /insert_item
    OPTIONS
    POST
  ▼ /insert_merchandise
    OPTIONS
    POST
  ▼ /login
    OPTIONS
    POST
  ▼ /register
    OPTIONS
    POST
  ▼ /show_item
    OPTIONS
    POST
  ▼ /update_item
    OPTIONS
    POST
```

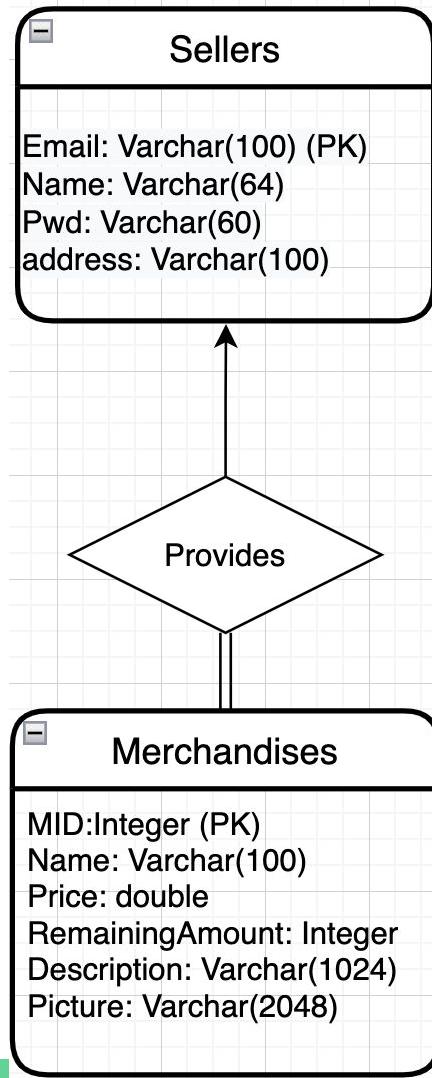
# Seller Service

## ER Diagram

### And Schema

Github for Seller

<https://github.com/ywang289/seller.git>



Seller	tables 3
Merchantises	columns 8
MID int	
Name varchar(100)	
Price double	
RemainingAmount int	
Description varchar(1024)	
Picture1 varchar(2048)	
Picture2 varchar(2048)	
Picture3 varchar(2048)	
keys 1	
indexes 1	
Provides	columns 2
Email varchar(100)	
MID int	
keys 1	
foreign keys 2	
indexes 2	
Sellers	columns 4
Email varchar(100)	
Name varchar(64)	
Pwd varchar(20)	
address varchar(100)	
keys 1	
indexes 1	

# EC2

EC2 > Instances > i-01838c8e3d10cbfe2

### Instance summary for i-01838c8e3d10cbfe2 (seller) [Info](#)

Updated less than a minute ago

Instance ID <a href="#">i-01838c8e3d10cbfe2 (seller)</a>	Public IPv4 address <a href="#">52.55.10.164</a>   open address	Private IPv4 addresses <a href="#">172.31.88.37</a>
IPv6 address -	Instance state <a href="#">Running</a>	Public IPv4 DNS <a href="#">ec2-52-55-10-164.compute-1.amazonaws.com</a>   open address
Hostname type IP name: ip-172-31-88-37.ec2.internal	Private IP DNS name (IPv4 only) <a href="#">ip-172-31-88-37.ec2.internal</a>	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a> <a href="#">Learn more</a>
Auto-assigned IP address <a href="#">52.55.10.164 [Public IP]</a>	VPC ID <a href="#">vpc-0d82b169fad21879f</a>	Auto Scaling Group name -
IAM Role -	Subnet ID <a href="#">subnet-018fed174e0ff90bb</a>	-

[Details](#) [Security](#) [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

**Instance details** [Info](#)

Platform <a href="#">Amazon Linux (Inferred)</a>	AMI ID <a href="#">ami-0b0dcdb5067f052a63</a>	Monitoring disabled
Platform details -	AMI name -	Termination protection

aws Services Search [Option+S]

Last login: Wed Dec 21 04:02:32 2022 from ec2-18-206-107-28.compute-1.amazonaws.com

```
|_ _| ( _ _ / ) Amazon Linux 2 AMI
__| \_\_|_\_|

https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 53 available
Run "sudo yum update" to apply all updates.
ec2-user@ip-172-31-88-37:~$ ls
seller
ec2-user@ip-172-31-88-37:~$ cd seller
ec2-user@ip-172-31-88-37:~/seller]$ python3 seller.py
* Serving Flask app 'seller'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8081
* Running on http://172.31.88.37:8081
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 742-428-642
<class 'float'>
14.212.177.254 - - [21/Dec/2022 19:06:19] "POST /customer/search HTTP/1.1" 200 -
<class 'float'>
14.210.65.240 - - [21/Dec/2022 19:35:22] "POST /customer/search HTTP/1.1" 200 -
```

i-01838c8e3d10cbfe2 (seller)

Public IPs: 52.55.10.164 Private IPs: 172.31.88.37

# Seller service: Address Verification with SmartyStreets

The image shows two separate Postman API requests side-by-side.

**Request 1 (Left):**

- Method:** POST
- URL:** https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping/seller/register
- Body (JSON):** {"username": "ywang", "password": "0002", "email": "ssoh@gmail.com", "address": "400w 113th", "zipcode": "10025"}
- Status:** 200 OK
- Body (Pretty):**

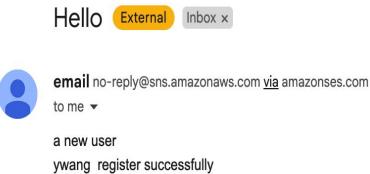
```
1 {  
2   "message": "register successfully",  
3   "state": true  
4 }
```

**Request 2 (Right):**

- Method:** POST
- URL:** https://b5st0ajud4.execute-api.us-east-1.amazonaws.com/shopping/seller/register
- Body (JSON):** {"username": "ywang", "password": "0002", "email": "wg@gmail.com", "address": "400w", "zipcode": "10000"}
- Status:** 200 OK
- Body (Pretty):**

```
1 {  
2   "message": "invalid address",  
3   "state": false  
4 }
```

# Seller service: Notification with SNS and Lambda functions



If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:  
<https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsns.us-east-1.amazonaws.com%2Funsubscribe.html%3FSubscriptionArn%3Dam%3Aaws%3Asns%3Aus-east-1%3A421483771059%3Aemail%3A3d50f2a8-f5c6-4739-873c-ea58ba0e9408%26Endpoint%3Dywang289%40buffalo.edu&data=05%7C01%7Cywang289%40g-mail.buffalo.edu%7Cec7e1c175f6b4dfcecf108dae388c6f6%7C96464a8af8ed40b199e25fb50a20250%7C0%7C638072473833503597%7CUnknown%7CTWFPbGZsb3d8eyJWjpoIMC4wlAwMDA1LCJQijoV2luMzIiLCJBTiI6lk1haWw1LCJXVCi6Mn0%3D%7C3000%7C%7C%7C&sdata=ltH0xvbaWn79pRY%2Bani6RT8MYQj1tvtYT0ky3YIAU%3D&reserved=0>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at  
<https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Faws.amazon.com%2Fsupport&data=05%7C01%7Cywang289%40g-mail.buffalo.edu%7Cec7e1c175f6b4dfcecf108dae388c6f6%7C96464a8af8ed40b199e25fb50a20250%7C0%7C0%7C638072473833503597%7CUnknown%7CTWFPbGZsb3d8eyJWjpoIMC4wlAwMDA1LCJQijoV2luMzIiLCJBTiI6lk1haWw1LCJXVCi6Mn0%3D%7C3000%7C%7C%7C&sdata=E%2FSkrBqxR2i3NdFa04Cbaz0NkJeAG1YljbjpmhHXJY%3D&reserved=0>

The screenshot shows the AWS Lambda function configuration interface. The function name is "infoname". It has two triggers associated with an API Gateway. The "Code source" tab is selected, showing the following Python code in the "lambda\_function.py" file:

```
1 import boto3
2
3 def lambda_handler(event, context):
4     # TODO implement
5
6     client = boto3.client('sns')
7     username = event["username"]
```

# Like customer service...

Pagination

Google login

Notification with sns and lambda functions

Middlewire

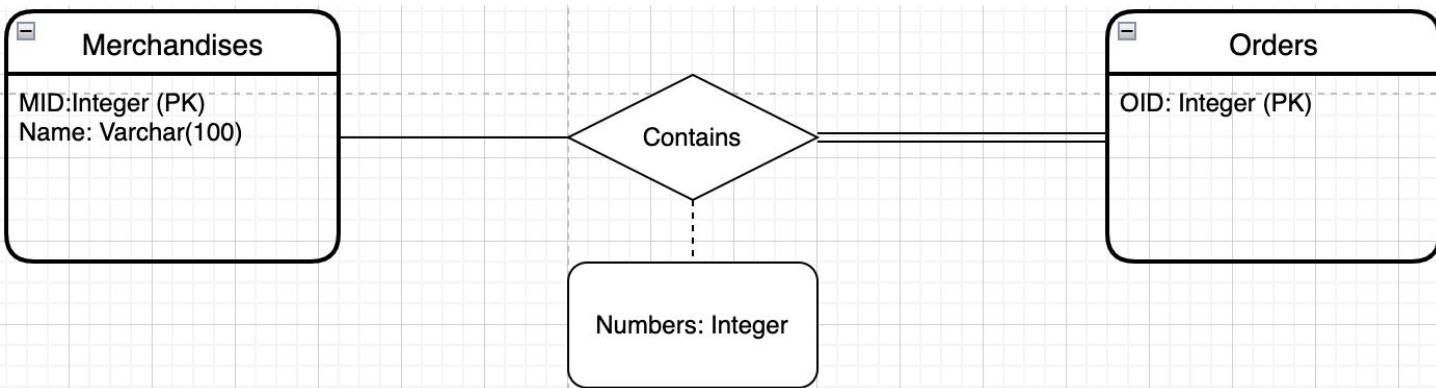
# Order service

---

# Order Service Overview

- Accept and respond with JSON
- Deploy on Elastic Beanstalk
- Connect a cloud database and browser application
- Interact with cloud database
  - ▼ /order
    - ▼ /order/add\_merchandise
    - OPTIONS**
    - POST**
  - ▼ /order/place\_order
  - OPTIONS**
  - POST**
  - ▼ /order/update\_merchandi
  - OPTIONS**
  - POST**

# Order Service ER Diagram And Schema



< Purchase
< tables 3
< Contains
< columns 3
OID int
MID int
Numbers int
> keys 1
> foreign keys 2
> indexes 2
< Merchandise
< columns 2
MID int
Name varchar(100)
> keys 1
> indexes 1
< Orders
< columns 1
OID int
> keys 1
> indexes 1

Github for order <https://github.com/ywang289/order.git>

# Order Service: Elastic Beanstalk Deployment

Order-env

[Order-env.eba-nqc9rr5s.us-east-1.elasticbeanstalk.com](http://Order-env.eba-nqc9rr5s.us-east-1.elasticbeanstalk.com) (e-wsmxubjxf2)

Application name: Order

Refresh

Actions ▾

Health



Ok

Causes

Running version

code-pipeline-1671654076394-  
42358b61278fa14ea396cdd6b8f  
bdad5599adef8

Upload and deploy

Platform

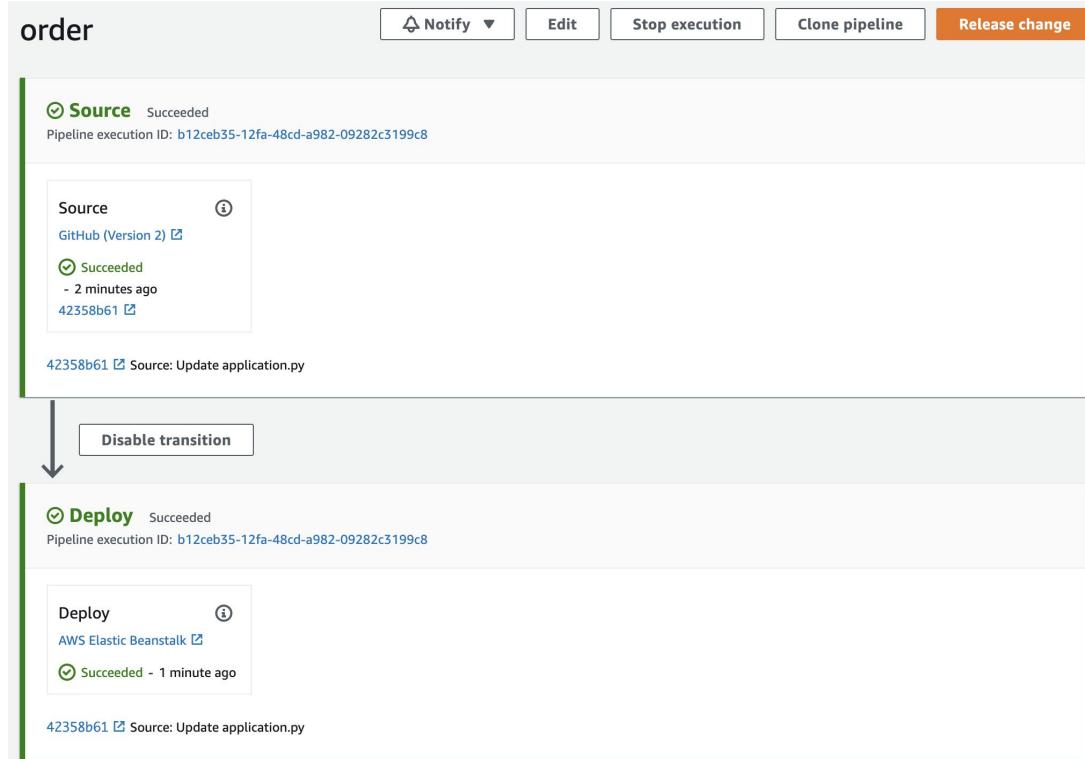


Python 3.8 running on 64bit  
Amazon Linux 2/3.4.2

Change

Link for Elastic Beanstalk: <http://order-env.eba-nqc9rr5s.us-east-1.elasticbeanstalk.com/>

# Elastic Beanstalk Automate Deployment /Redeployment With CodePipeline



Github for CodePipeline: <https://github.com/ZYbunny612/flask-example.git>  
(Same code as order, just add gitignore and change name to application.py for deployment)

# Composite Service

---

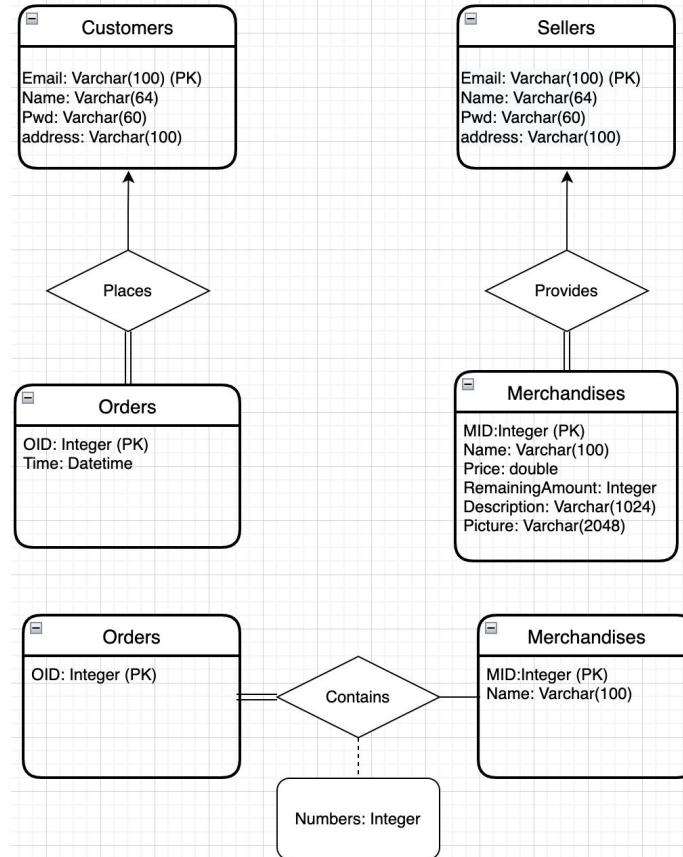
# Composite Service

1. Customer service
2. Seller service
3. Order service

Github for compose:

<https://github.com/ywang289/compose.git>

```
customer_http="http://ec2-44-201-86-144.compute-1.amazonaws.com:8080/"
seller_http="http://ec2-52-55-10-164.compute-1.amazonaws.com:8081/"
order_http="http://order-env.eba-nqc9rr5s.us-east-1.elasticbeanstalk.com/"
```



# Composite Service: Sequential

## 1. Customer Place Order

For each involved database:

`/order/check_amount`

Sellers → check remaining amount for each purchased merchandise:

If all remaining amounts are enough for the order, then update (decrease by the purchased numbers) for each;

If one is not enough, then return warning message;

If check amount is successful:

`/customer/place_order`

Customers → Generate OID; Insert to Orders; Insert to Places; Return OID;

If get OID:

`/order/place_order`

Purchases → Insert OID to Orders; for each purchased merchandise, insert OID, its MID and numbers to Contains

```
@app.route('/customer/purchase', methods=['POST'])
def customer_purchase():
    if request.method == 'POST':
        data = json.loads(request.get_data())
        email = data['email']
        time = data['timestamp']
        items = data["items"]

        sellers_check = requests.post(seller_http+'order/check_amount', data=request.get_data())
        #{response: success or fail}
        # print(json.loads(sellers_check.text)['state'])
        # print(json.loads(sellers_check.text)['message'])
        if json.loads(sellers_check.text)['state']:
            # success
            #{email: string, timestamp: time,( current time),items:dictionary{merchandise id: amount
            print("success")
            s= json.dumps({'email':email, 'timestamp':time, 'items':items})

            get_oid = requests.post(customer_http+'customer/place_order', data=s)
            # true/ false
            oid = json.loads(get_oid.text)['oid']
            if json.loads(get_oid.text)['state']:
                print("successfully")
                # { "email":"test3@gmail.com", "timestamp":"2022-12-14 17:30:00" , "order": {"1": "10",
                s= json.dumps({'email':email, 'timestamp':time, 'items':items, "oid": oid})
                insert_item= requests.post(order_http+'order/place_order', data=s)
    return insert_item.text
```

# Composite Service: Sequential

## 2. Seller Add Merchandise

For each involved database:

`/seller/insert_item`

Sellers → Generate MID; Insert to Merchandises and Provides; Return MID

If get MID:

`/order/add_merchandise`

Purchases → Insert MID, name to Merchandises

```
@app.route('/seller/insert_merchandise', methods=['POST'])
def insert_merchandise():
    if request.method == 'POST':
        data = json.loads(request.get_data())
        email = data['email']
        name= data["name"]
        price= data["price"]
        remaining_amount= data['remaining_amount']
        description= data["description"]
        picture= data['picture']
        sellers_insert= requests.post(seller_http+'seller/insert_item', data=request.get_data())

        if json.loads(sellers_insert.text)['state']:
            # success
            print(json.loads(sellers_insert.text))
            mid = json.loads(sellers_insert.text)[ 'mid' ]

            #{email: string, timestamp: time,( current time),items:dictionary{merchandise id: amount}
            print("success")
            s= json.dumps({'mid':mid, "name": name})
            print ({'mid':mid, "name": name})
            insert_order = requests.post(order_http+'order/add_merchandise', data=s)

            # true/ false
            print("second")
    return insert_order.text
```

# Composite Service: Sequential

## 3. Seller Update Merchandise

For each involved database:

`/seller/update_item`

Sellers → Update related attributes to Merchandises by MID

`/order/update_merchandise`

Purchases → Update name to Merchandise by MID

Note: seller delete merchandise is not composite

We don't delete merchandise in the Purchase database in case that a deleted merchandise was purchased and needed to be checked details through the order

```
# {email, name, price, remaining_amount, description, picture, mid}
@app.route('/compose/update_merchandise', methods=['POST'])
def order_detail():
    if request.method == 'POST':

        order_detail = requests.post(seller_http+'seller/update_item', data=request.get_data())
        order=json.loads(order_detail.text)
        # if can not search order, need to add something
        compose_detail= requests.post(order_http+'/order/update_merchandise', data=request.get_data())

    return json.loads(compose_detail.text)
```

# EC2

EC2 > Instances > i-02195f1535590ed6a

Instance summary for i-02195f1535590ed6a (compose) [Info](#)

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-02195f1535590ed6a (compose)	<a href="#">54.89.21.82</a>   <a href="#">open address</a>	<a href="#">172.31.88.91</a>
IPv6 address	Instance state	Public IPv4 DNS
-	<a href="#">Running</a>	<a href="#">ec2-54-89-21-82.compute-1.amazonaws.com</a>   <a href="#">open address</a>
Hostname type	Private IP DNS name (IPv4 only)	Elastic IP addresses
IP name: ip-172-31-88-91.ec2.internal	<a href="#">ip-172-31-88-91.ec2.internal</a>	-
Answer private resource DNS name IPv4 (A)	Instance type	AWS Compute Optimizer finding
IPv4 (A)	t2.micro	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>
Auto-assigned IP address	VPC ID	<a href="#">Learn more</a>
<a href="#">54.89.21.82</a> [Public IP]	<a href="#">vpc-0d82b169fad21879f</a>	
IAM Role	Subnet ID	Auto Scaling Group name
-	<a href="#">subnet-018fed174e0ff90bb</a>	-

[Details](#) [Security](#) [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

▼ Instance details [Info](#)

```
Last login: Thu Dec 22 16:54:58 2022 from ec2-18-206-107-29.compute-1.amazonaws.com
[ec2-user@ip-172-31-88-91 ~]$ cd
[ec2-user@ip-172-31-88-91 ~]$ ls
compose
[ec2-user@ip-172-31-88-91 ~]$ cd compose
[ec2-user@ip-172-31-88-91 compose]$ python3 compose.py
a
 * Serving Flask app 'compose'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8084
 * Running on http://172.31.88.91:8084
Press CTRL+C to quit
* Restarting with stat
a
 * Debugger is active!
 * Debugger PIN: 843-079-816
success
successfully
44.212.177.207 - - [22/Dec/2022 21:44:30] "POST /customer/purchase HTTP/1.1" 500 -
i-02195f1535590ed6a (compose)
Public IPs: 54.89.21.82 Private IPs: 172.31.88.91
```

Composite service:<https://ec2-54-89-21-82.compute-1.amazonaws.com>

# Contribution

Frontend:

Juntian Sun (js5940):

1. Write the React frontend for the project
2. Create the Github Repo for the frontend
3. Deploy the React files in S3 bucket
4. Connect the frontend to API Gateway
5. Encapsulate the S3 bucket into CloudFront
6. Add Google OAuth for login on frontend

Zijian Zhang (zz2994):

1. Assist implementing the front-end project
2. Help creating queries for front-end connection
3. Use css to simply beautify the UI interface

Backend:

Yifan Wang (yw3914):

1. Write backend of microservers
2. Deploy the customer, seller, compose server in EC2
3. SNS and Lambda functions (middlewire)
4. Api gateway and cloud api for register
5. Pagination and composition service

Yi Zhang (yz4339):

1. Design, create, and manage the databases
2. Write queries needed by the backend
3. Write the backend together with Yifan Wang
4. Write the S3 and Elastic Beanstalk document
5. Deploy the order service on Elastic Beanstalk with CodePipeline

Thank you for watching!

---