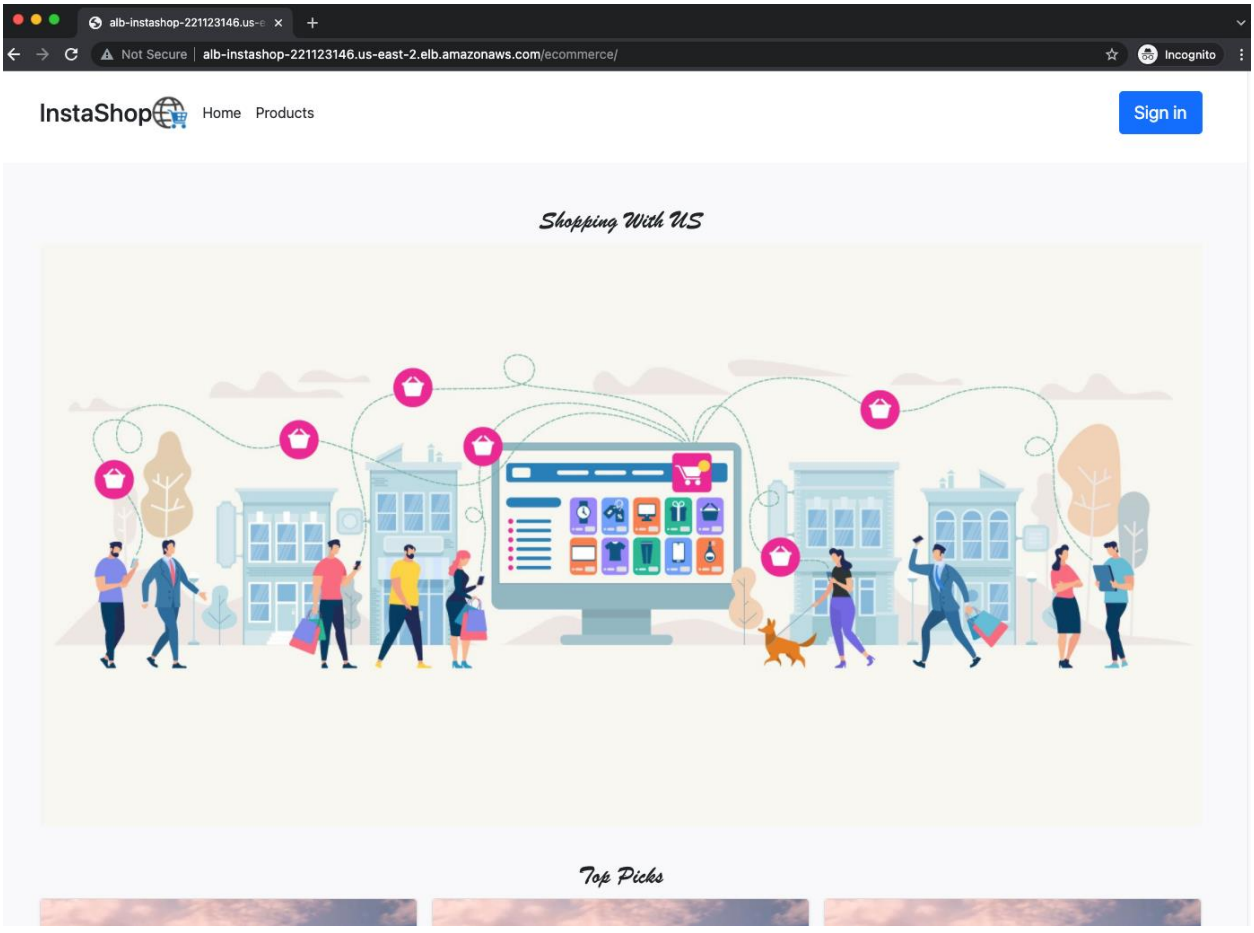
CS6620 Fundamentals of Cloud Computing

E-Commerce SaaS Web Application -- InstaShop

Team Members:

Guan Wang
Yu Feng
Qichen An
Wenhao Ge

https://github.com/singinzrain/cloud_computing_ecommerce



Project Description:

For years, on-premises solutions were synonymous with ecommerce. These solutions were installed on-site on company-owned servers. They promised innovation and customization — often by adding on hardware. This is how many companies built high-growth digital businesses, but they proved rigid. Over time, more IT resources were needed to support legacy systems. What started as a strength — the ability to customize — became a weakness. Years of one-off customizations made platforms less agile and inflexible. Unfortunately, ecommerce is the last place any company can afford to lose out. Businesses need to meet ever-changing customer expectations, security threats, and regulatory requirements. Ecommerce technology is a significant area of innovation. Difficult and costly product upgrades were another big challenge with legacy on-premises solutions. Even after investing considerable time, money, and resources, upgrades added little business value.

In the mid-2000s, cloud-based SaaS ecommerce platforms became available. In the following years, many ecommerce vendors shifted to a cloud strategy. And, new entrants emerged. A SaaS e-commerce platform is an e-commerce software delivered as a cloud-based system that can be accessed through all types of web browsers. Cloud-based SaaS ecommerce platforms could provide Scalability, Flexibility, Maintenance, Updates and Security, which push SaaS-based e-commerce to come to the forefront as exactly the solution businesses need.

Therefore, under such a background, our team collaborated on the final project (SaaS Ecommerce web application) by designing, implementing, and thoroughly testing on AWS. And finally, we ultimately finished the project – InstaShop.

InstaShop is a SaaS e-commerce website. Nowadays, the Internet has changed the traditional financial model. Along with it, there are various applications that facilitate people's lives. For example, delivery software has changed the traditional restaurant supply model and e-commerce platforms have changed the offline sales model. The design of InstaShop is to help people stay at home but could buy a variety of snacks, fruits, groceries, take-out, household appliances and so on. In an ideal world, we'll have our own delivery people and would partner with UberEats, Ikea, Warlmart and so on. But for Instashop under the concept of CS6620 cloud engineering, our team will only do an e-commerce web application on the web side for the following use cases.

Use Case:

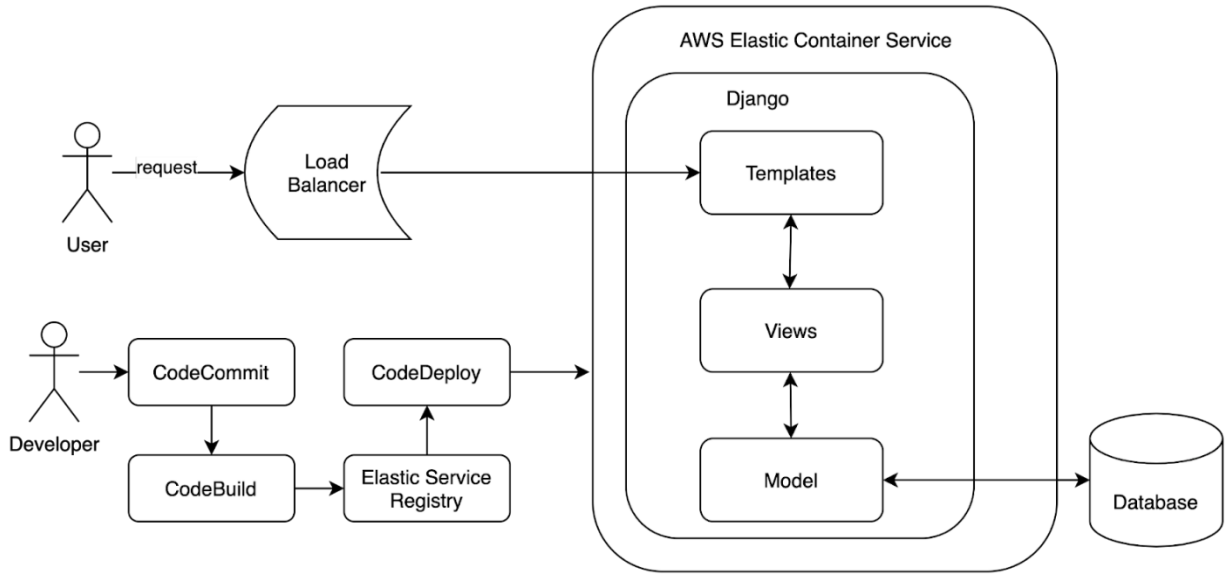
As customers go to our website, they will first see our home page which welcomes our customers and informs them with top picked products. They can view all the products as going to the products page. For each listed product, they can click on it which will direct them to a product detail page. From the navigation bar, they can sign up and login with their accounts. After they login, they can add products into the shopping cart, edit the shopping cart and then check out their orders. They can revisit their orders in the order history page.

Key Features:

- **Home page**: welcome customers and inform them with top 3 picked products
- **Product page**: showing all the products
- **Product details page**: showing more detail information for the product along with reviews
- **Shopping cart page**: showing a list of customer wanted products and the customer can increase or decrease the product count. The customer can also delete a product from the shopping cart
- **Check out page**: asking customers to fill out needed information and allowing them to check-out
- **History orders**: showing a list of history orders
- **Sign In/Sign Up**: allowing customers to create their account and login

Please note, we mention a search product feature in our project proposal. But, we think Sign In/Sign Up is a more important feature. Thus, we decided to implement the Sign In/Sign Up feature rather than the search product feature.

Architecture overview diagram (AOD) and design description



For a high-level view, users will send out requests which would be sent through a load balancer and directed to an AWS instance. In the AWS instance, our web application will be implemented with Django where requests would go through related templates, views, and models.

From the perspective of developers, AWS Codepipeline is leveraged. The code update will be pushed to CodeCommit repo, then CodeBuild is triggered to build a docker image and the image will be pushed to Elastic Service Registry. Finally, CodeDeploy is triggered to start the container from the image and deploy the service to ECS cluster.

Cloud Features:

For the implementation, our group use following AWS features:

1. AWS Elastic Container Service

   We deploy our containerized application service through ECS.

2. Load Balancer

   Route incoming end-users' traffic to applications based in the AWS cloud.

3. Amazon RDS

   We use AWS RDS to store our structured data.

4. Complete CI/CD

   Automate build and deployment with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline.

5. AWS Lambda

   In our deployment process, we tried to use zappa to deploy our web app which involves AWS Lambda.

Deployment Process

Create Dockerfile for the application

FROM python:3

ENV PYTHONDONTWRITEBYTECODE=1

ENV PYTHONUNBUFFERED=1

WORKDIR /code

COPY requirements.txt /code/

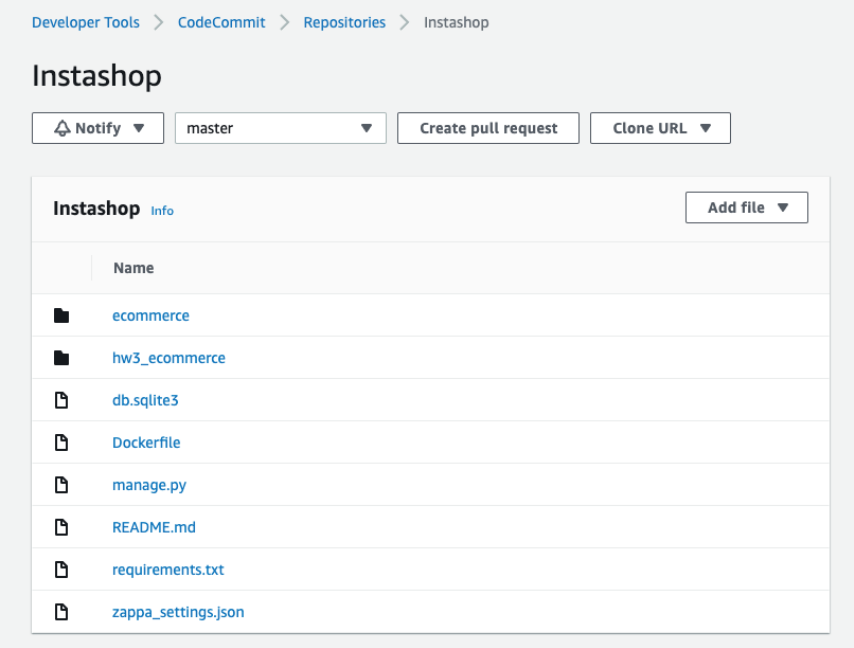
RUN pip install -r requirements.txt

COPY . /code/

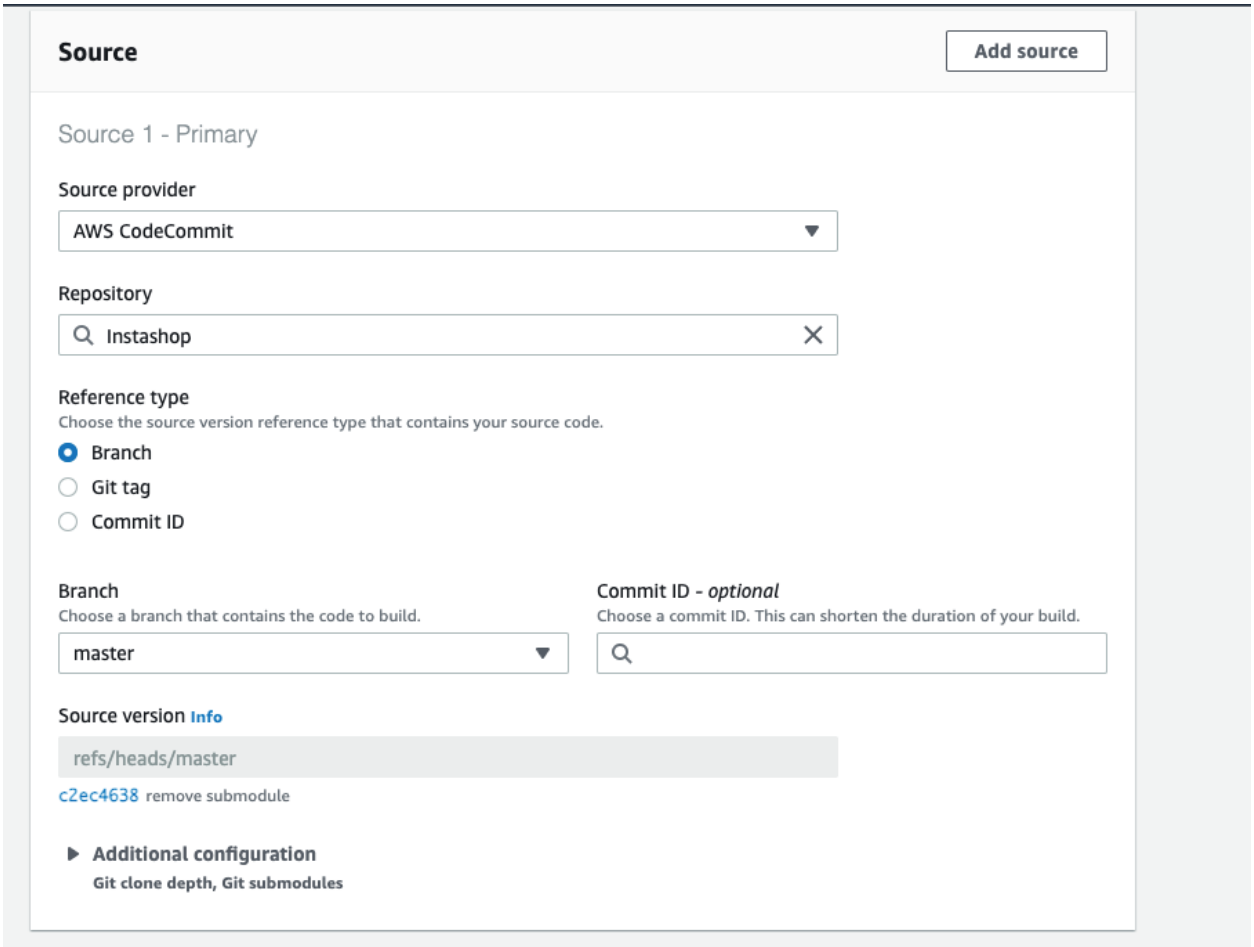
EXPOSE 8000

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

Create CodeCommit and upload the code files



Create Codebuild and make the source as the Instashop from CodeCommit, make sure the codebuild environment image is privileged



Add buildspec.yaml

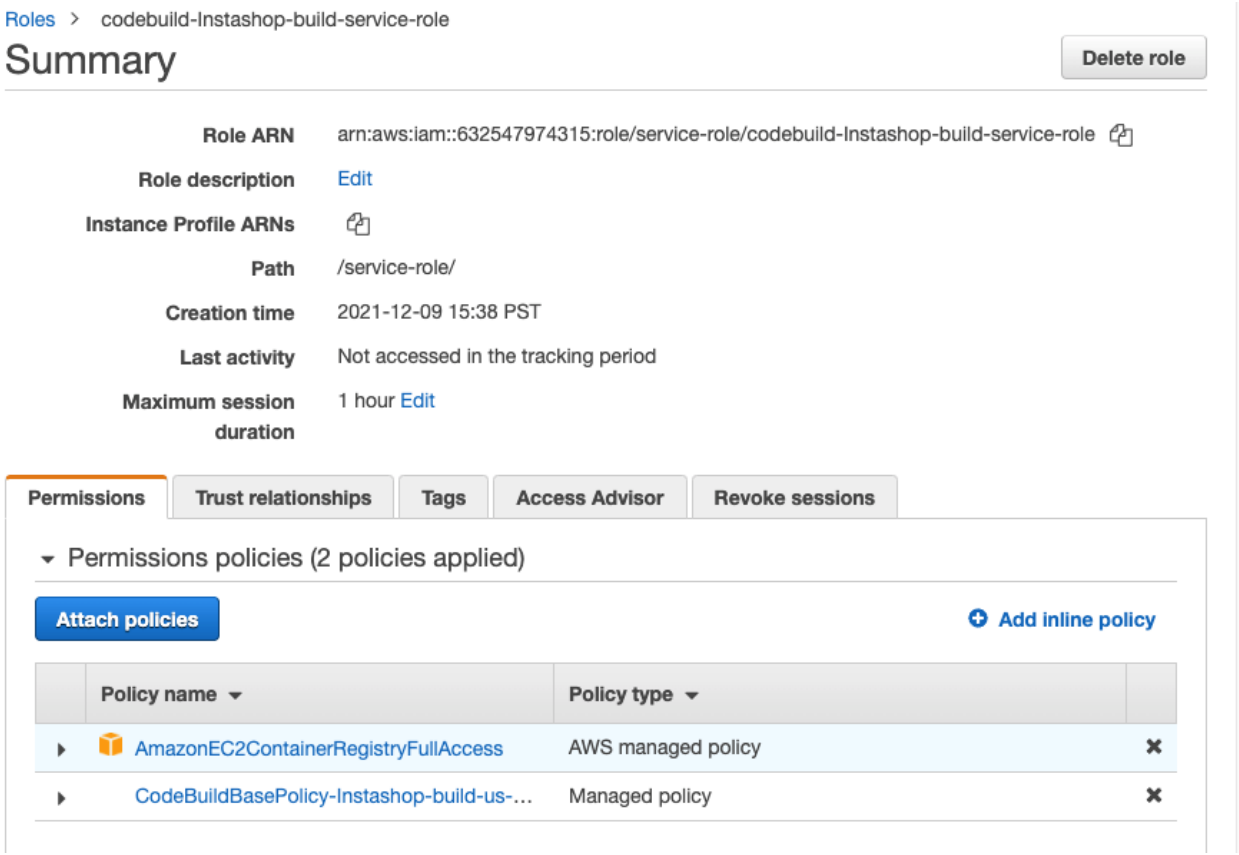
```
version: 0.2

phases:
pre_build:
  commands:
    - echo Logging in to Amazon ECR...
    - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
build:
  commands:
    - echo Build started on `date`
```

```
    - echo Building the Docker image...
    - docker build -t instashop:1 .
    - docker tag instashop:1 632547974315.dkr.ecr.us-east-2.amazonaws.com/yufengcloud
post_build:
  commands:
    - echo Build completed on `date`
    - echo Pushing the Docker image...
    - docker push 632547974315.dkr.ecr.us-east-2.amazonaws.com/yufengcloud
artifacts:
files: imagedefinition.json
```

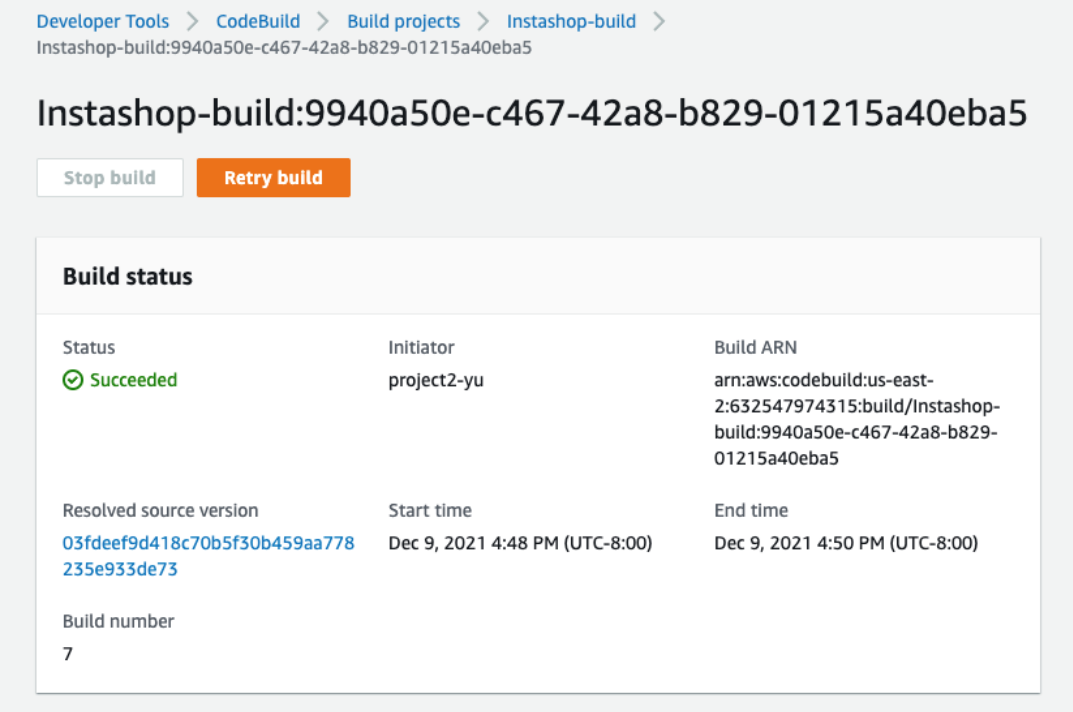
Add imagedefinition.json

```
[
{
  "name": "instashop",
  "imageUri": "632547974315.dkr.ecr.us-east-2.amazonaws.com/yufengcloud:latest"
}
]
```

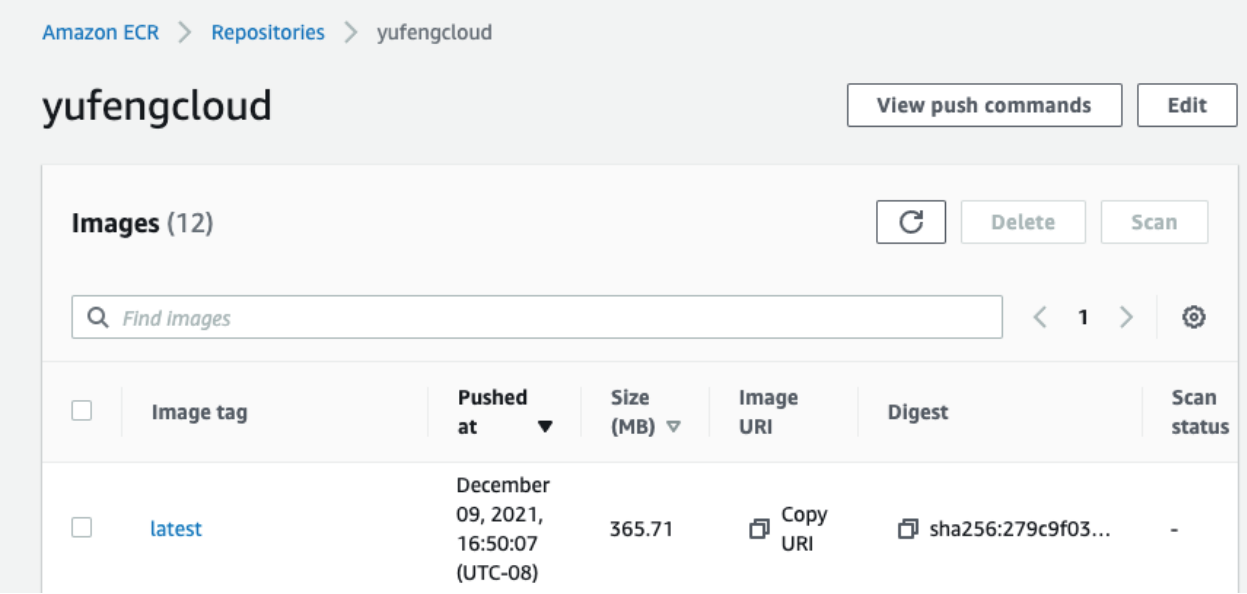
Add policy to the codebuild role



Successfully built image



Verify the image in ECR



Create task

Task Definitions > instashop-task > 1

Task Definition: instashop-task:1

View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the required changes to the task definition

Create new revision | Actions

Builder | JSON | Tags

Task definition name instashop-task

Task role *None*

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the IAM Console

Network mode awsvpc

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

Operating system family Linux

Compatibilities EC2, FARGATE

Requires compatibilities FARGATE

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.

Task execution role ecsTaskExecutionRole

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (MiB) 512

Task CPU (unit) 256

Task memory maximum allocation for container memory reservation

0 — 512 shared of 512 MiB

Task CPU maximum allocation for containers

0 — 256 shared of 256 CPU units

Task Placement

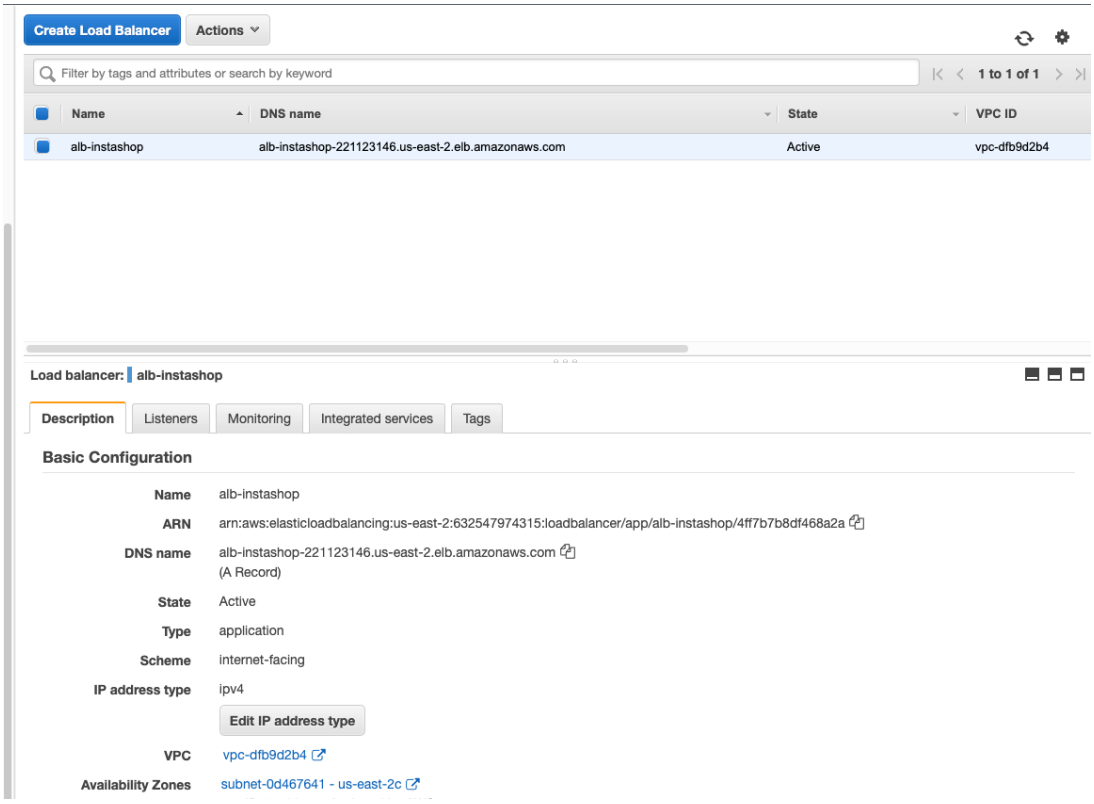
Constraint No constraints

Container definitions

Container Na…	Image	CPU Units	GPU	Inference Acc…	Hard/Soft memory limits (MiB)	Essential
instashop	632547974315…	0			--/--	true

Volumes

Create ALB


Create Load Balancer
Actions
Filter by tags and attributes or search by keyword
1 to 1 of 1
Name
DNS name
State
VPC ID
alb-instashop
alb-instashop-221123146.us-east-2.elb.amazonaws.com
Active
vpc-dfb9d2b4
Load balancer: alb-instashop
Description
Listeners
Monitoring
Integrated services
Tags
Basic Configuration
Name
alb-instashop
ARN
arn:aws:elasticloadbalancing:us-east-2:632547974315:loadbalancer/app/alb-instashop/4ff7b7b8df468a2a
DNS name
alb-instashop-221123146.us-east-2.elb.amazonaws.com
(A Record)
State
Active
Type
application
Scheme
internet-facing
IP address type
ipv4
Edit IP address type
VPC
vpc-dfb9d2b4
Availability Zones
subnet-0d467641 - us-east-2c

Create Cluster

Clusters > instashop-cluster

Cluster : instashop-cluster

Update Cluster | Delete Cluster

Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:us-east-2:632547974315:cluster/instashop-cluster
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	2 Fargate, 0 EC2, 0 External
Active service count	1 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Run new Task | Stop | Stop All | Actions | Last updated on December 9, 2021 10:11:22 PM (0m ago)

Desired task status: Running | Stopped

Filter in this page | Launch type ALL | 1-2 | Page size 50

Task	Task defini...	Container i...	Last status...	Desired st...	Started at ...	Started By ...	Group	Launch typ...	Platform v...
716bb69d2f...	instashop-t...	--	RUNNING	RUNNING	2021-12-09...	ecs-svc/08...	service:inst...	FARGATE	1.4.0
eba5d2010...	instashop-t...	--	RUNNING	RUNNING	2021-12-09...	ecs-svc/08...	service:inst...	FARGATE	1.4.0

Clusters > instashop-cluster

Cluster : instashop-cluster

Update Cluster | Delete Cluster

Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:us-east-2:632547974315:cluster/instashop-cluster
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	2 Fargate, 0 EC2, 0 External
Active service count	1 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

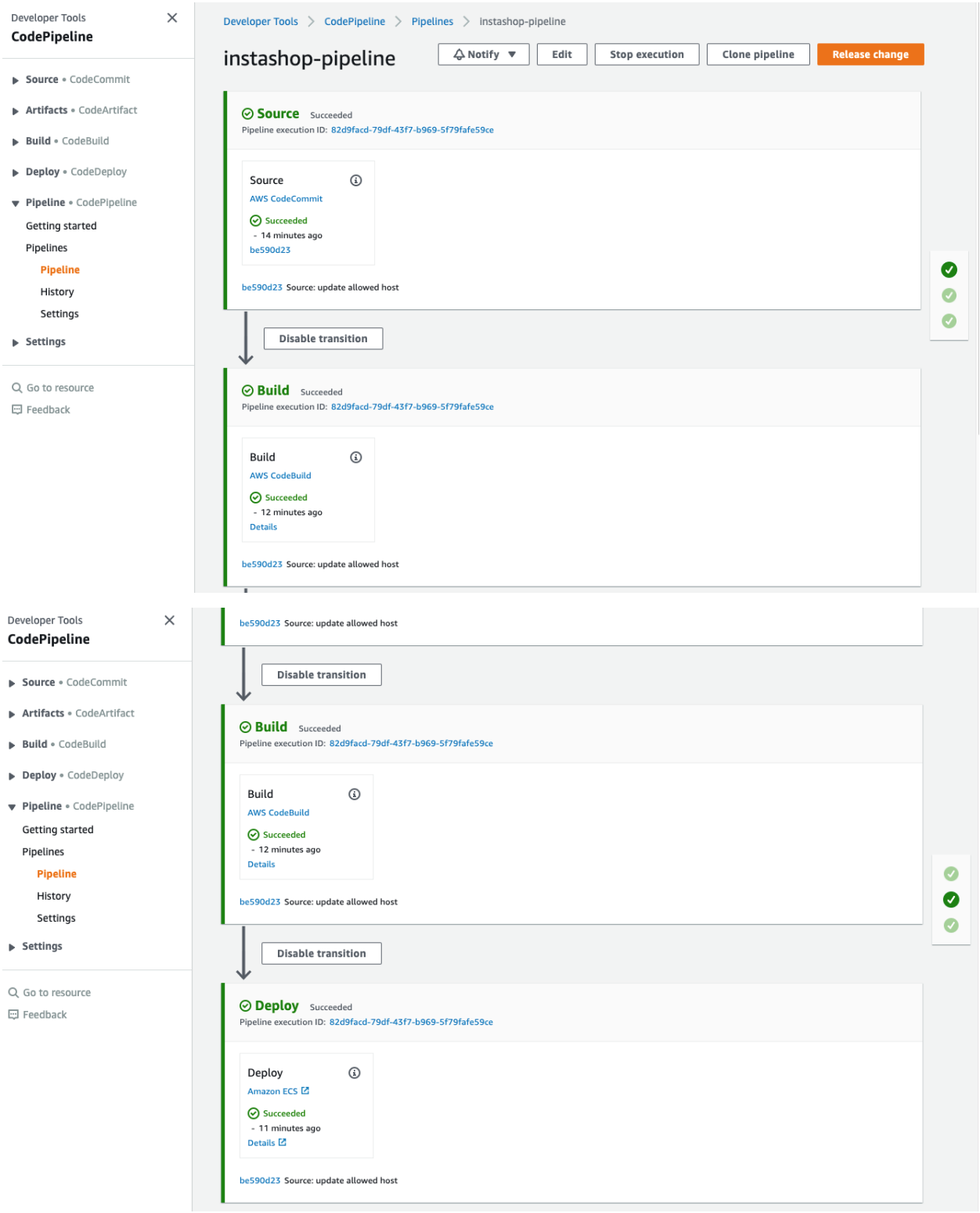
Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Create | Update | Delete | Actions | Last updated on December 9, 2021 10:11:03 PM (0m ago)

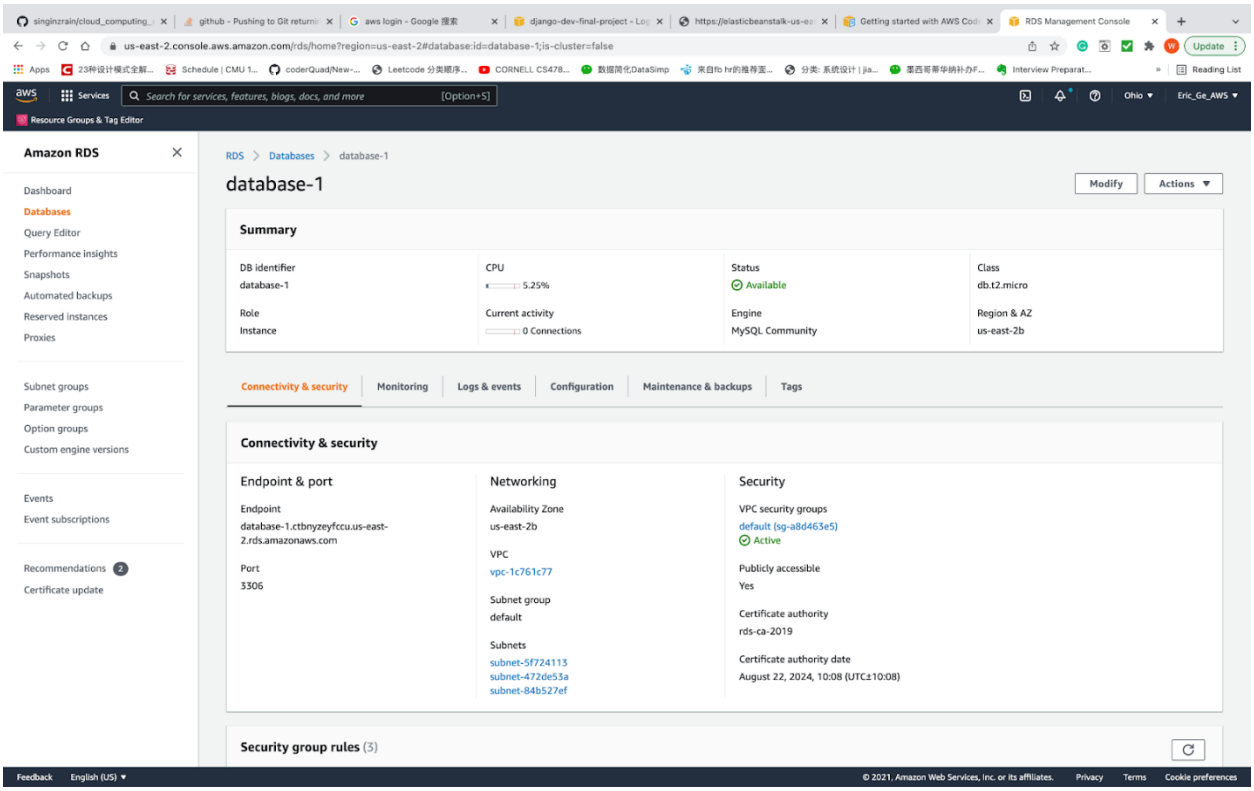
Filter in this page | Launch type ALL | Service type ALL | 1-1

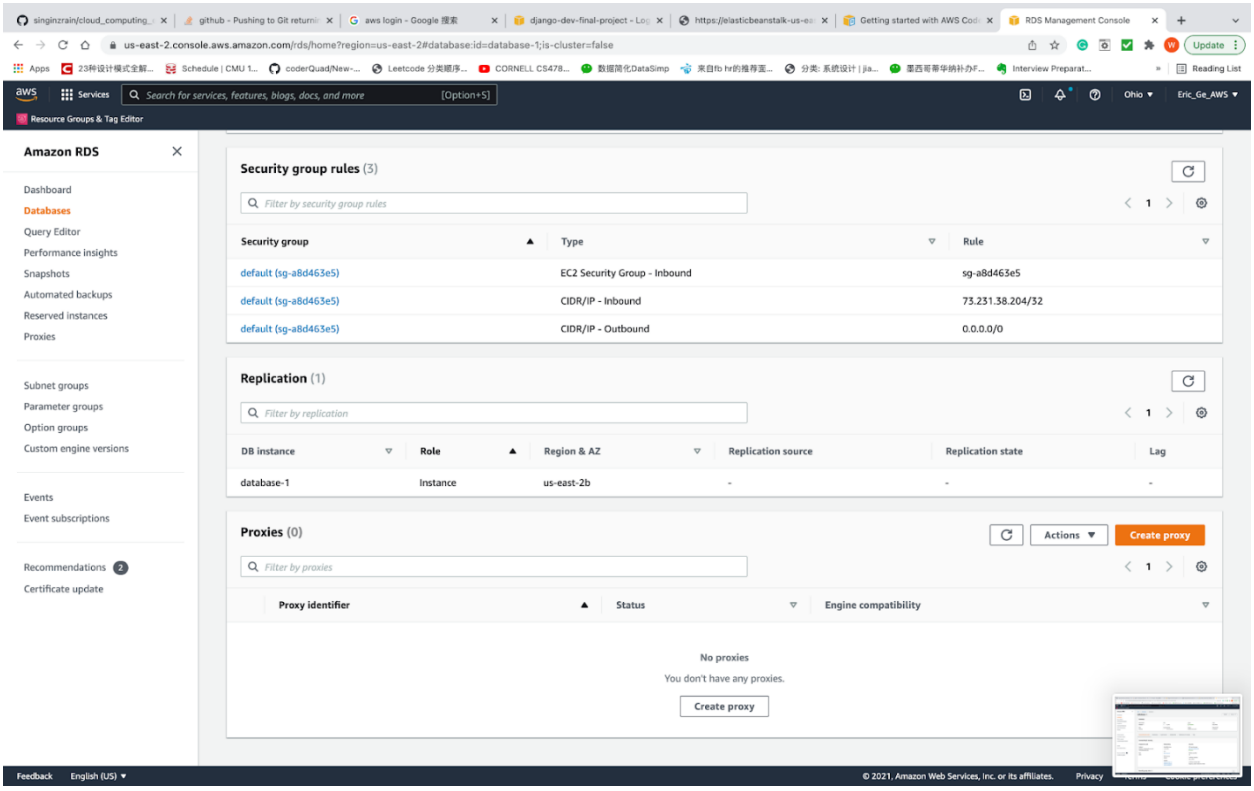
Service Name	Status	Service ty...	Task Defi...	Desired t...	Running t...	Launch ty...	Platform ...
instashop-service	ACTIVE	REPLICA	instashop-...	2	2	FARGATE	LATEST(1....

Create CodePipeline and execute successfully



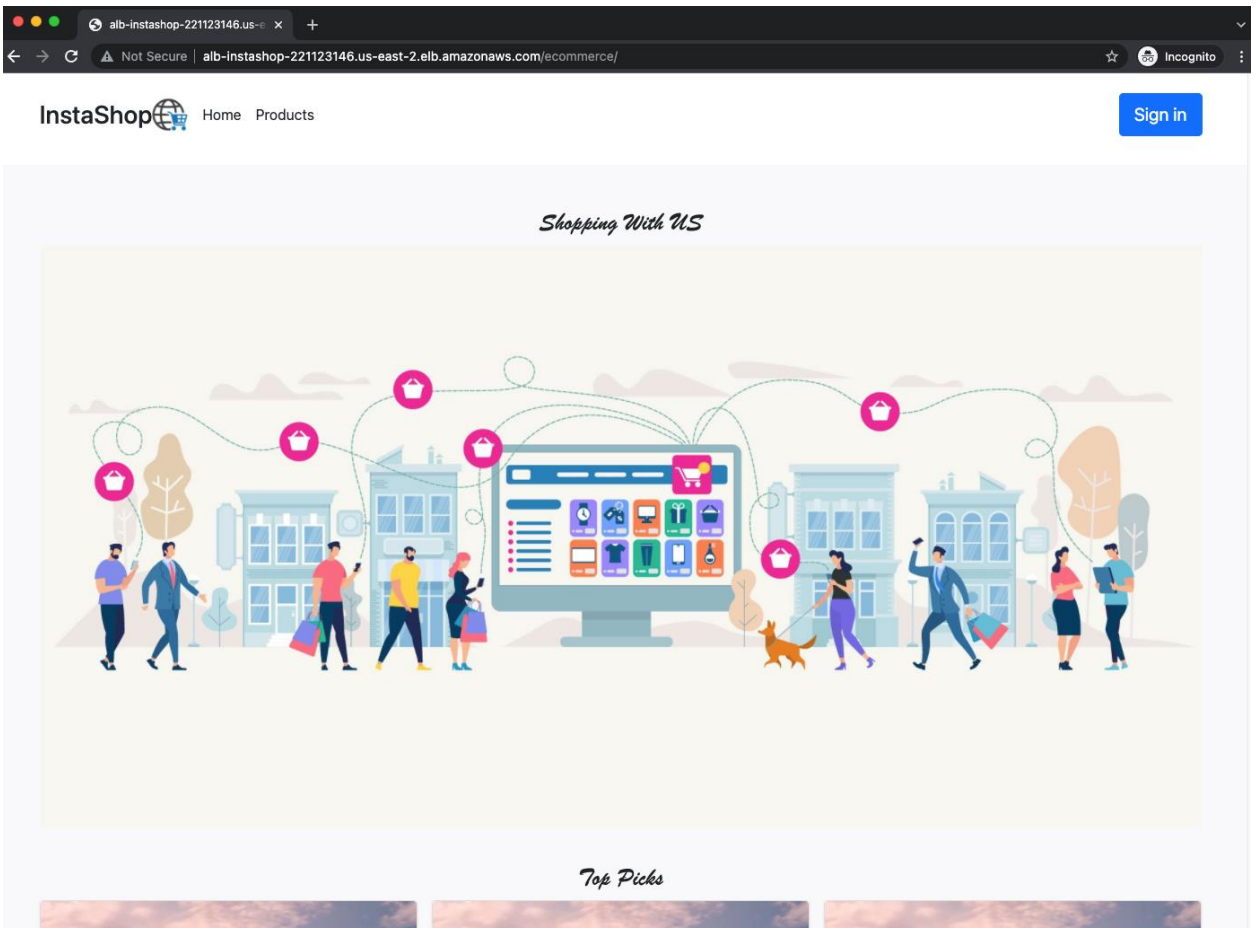
AWS Relational Database Service





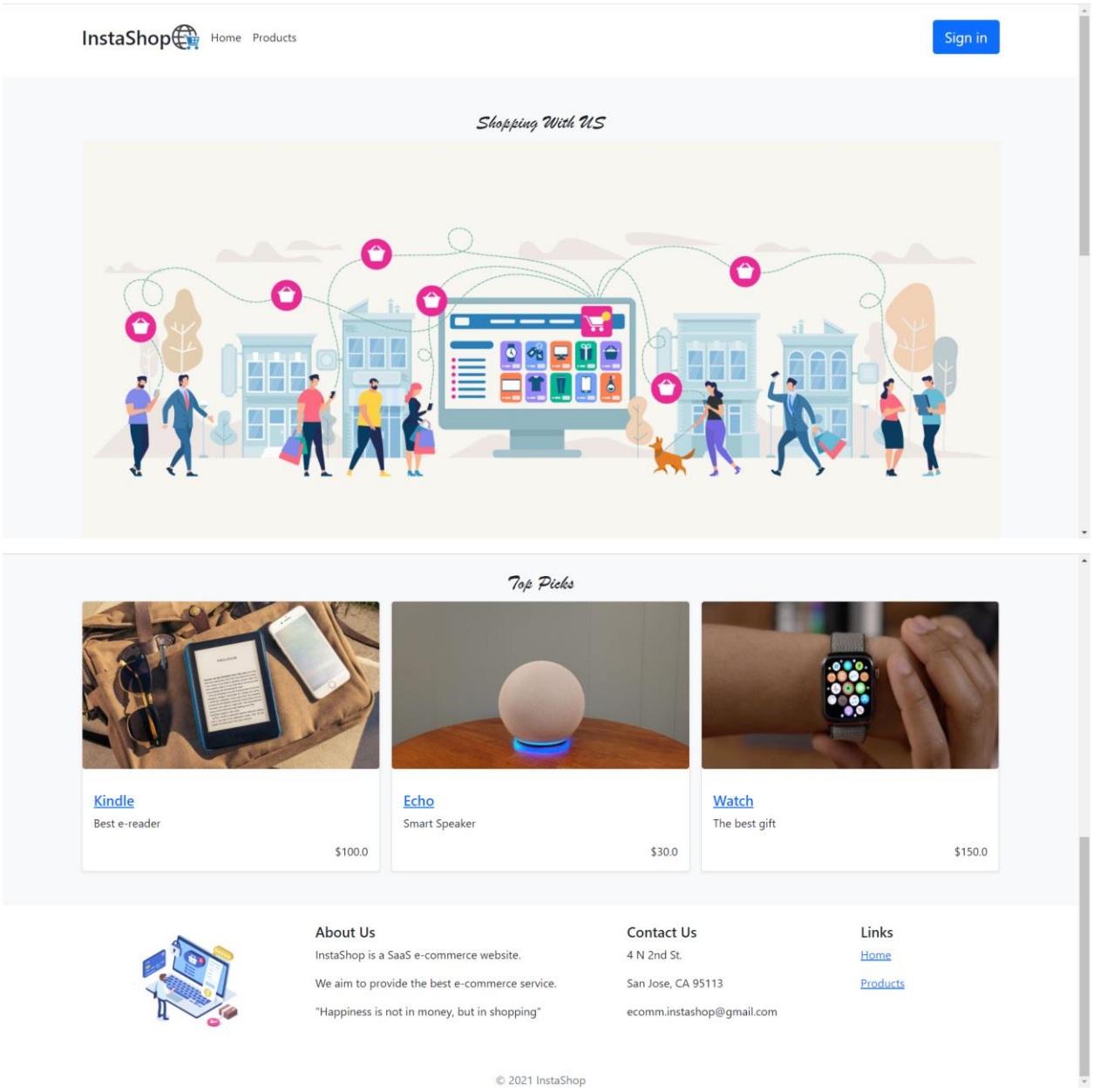
Verify the deployed web app is working

http://alb-instashop-221123146.us-east-2.elb.amazonaws.com/ecommerce/

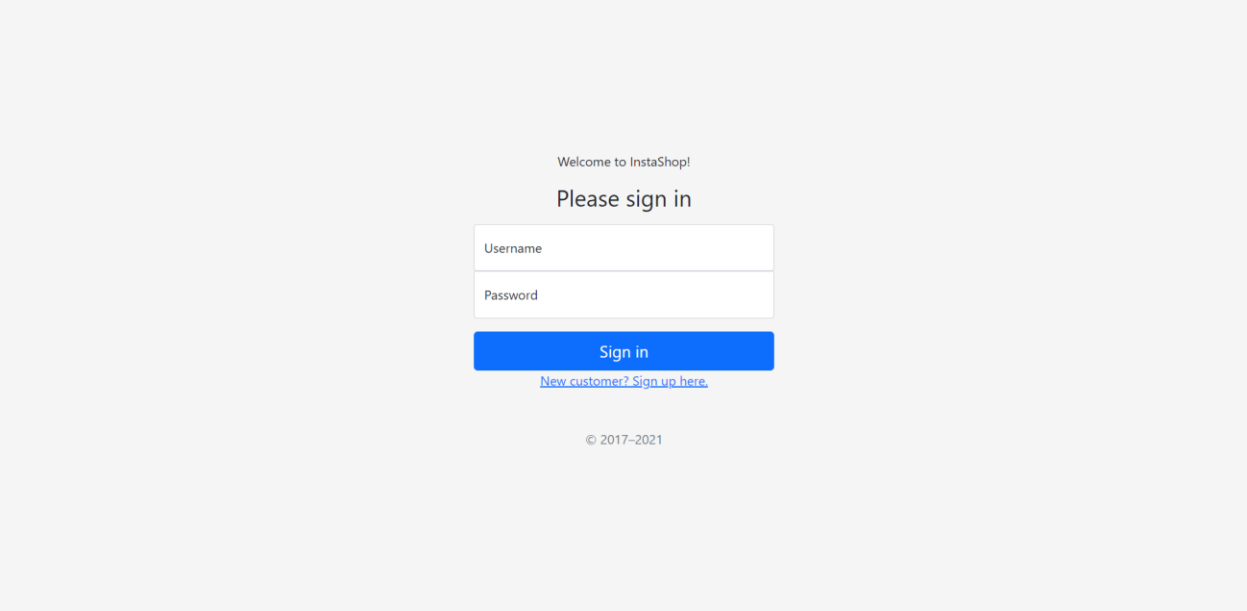


UI & HCI

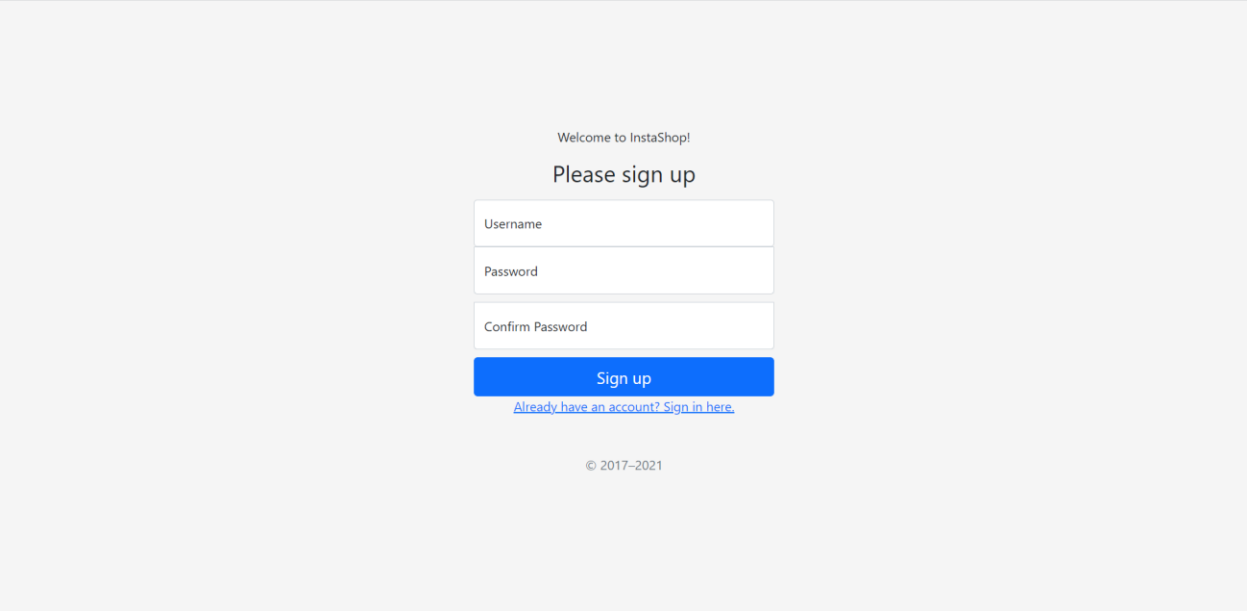
Home Page (without user logged in)



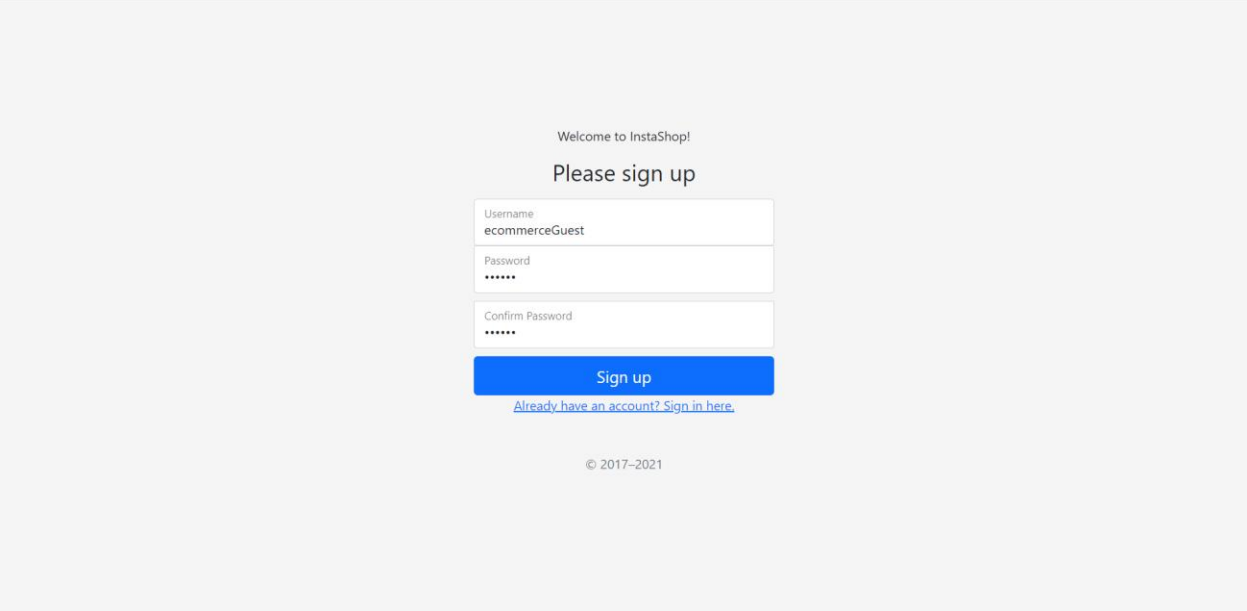
Sign In Page

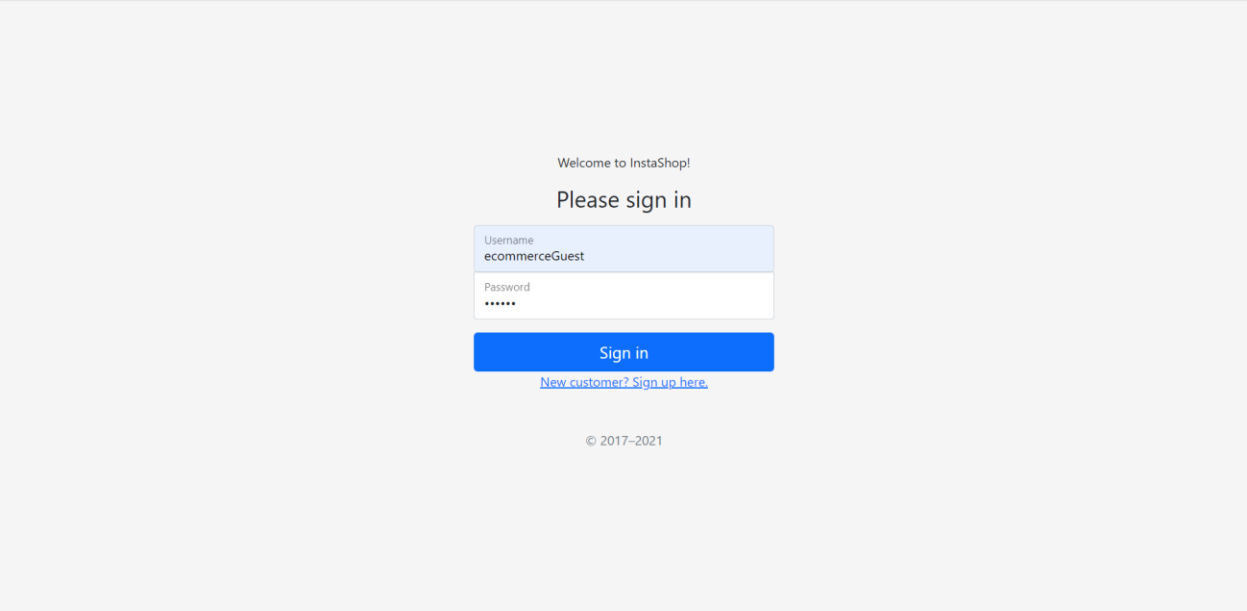


Sign Up Page

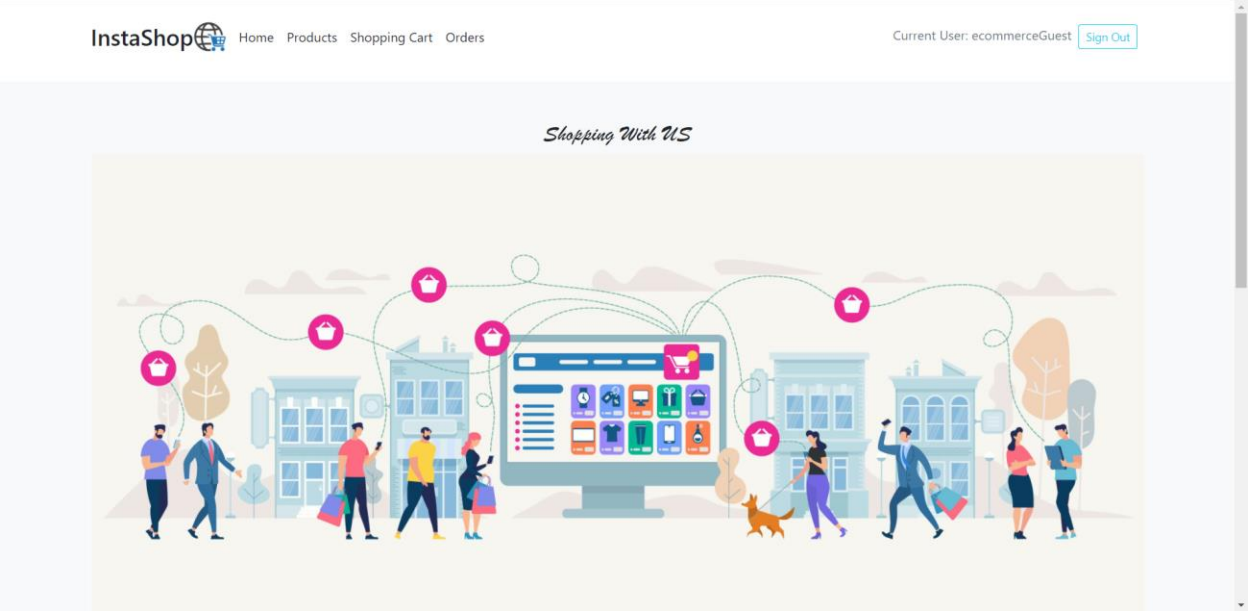


Here is an example, we create a guest for this system.

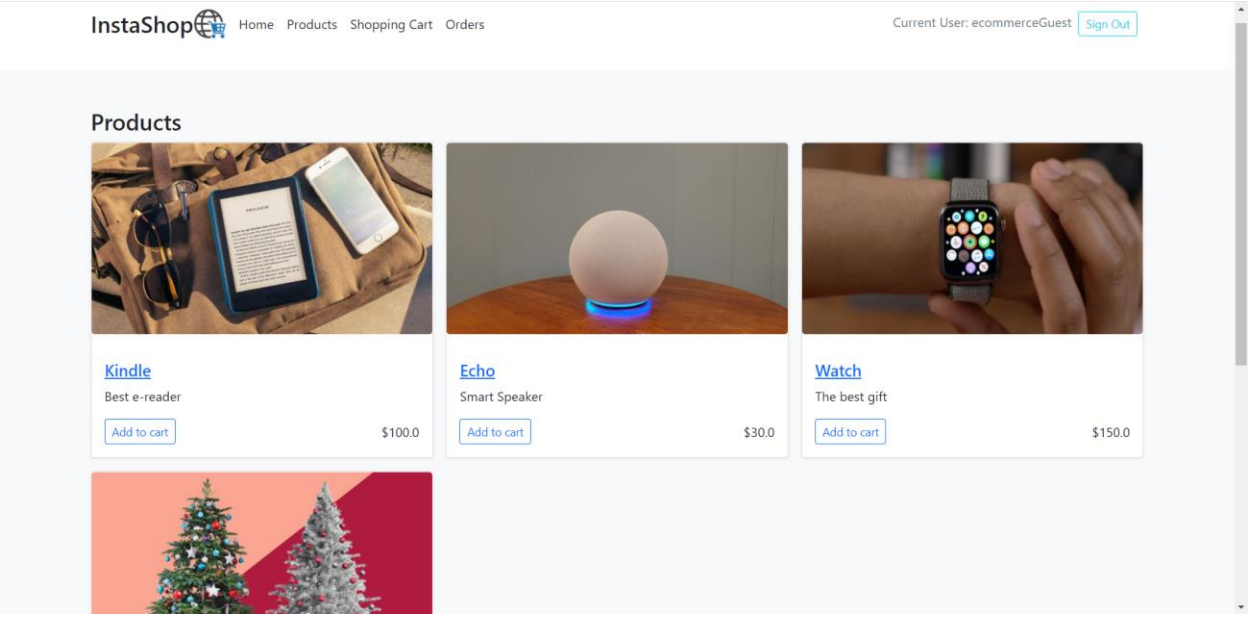


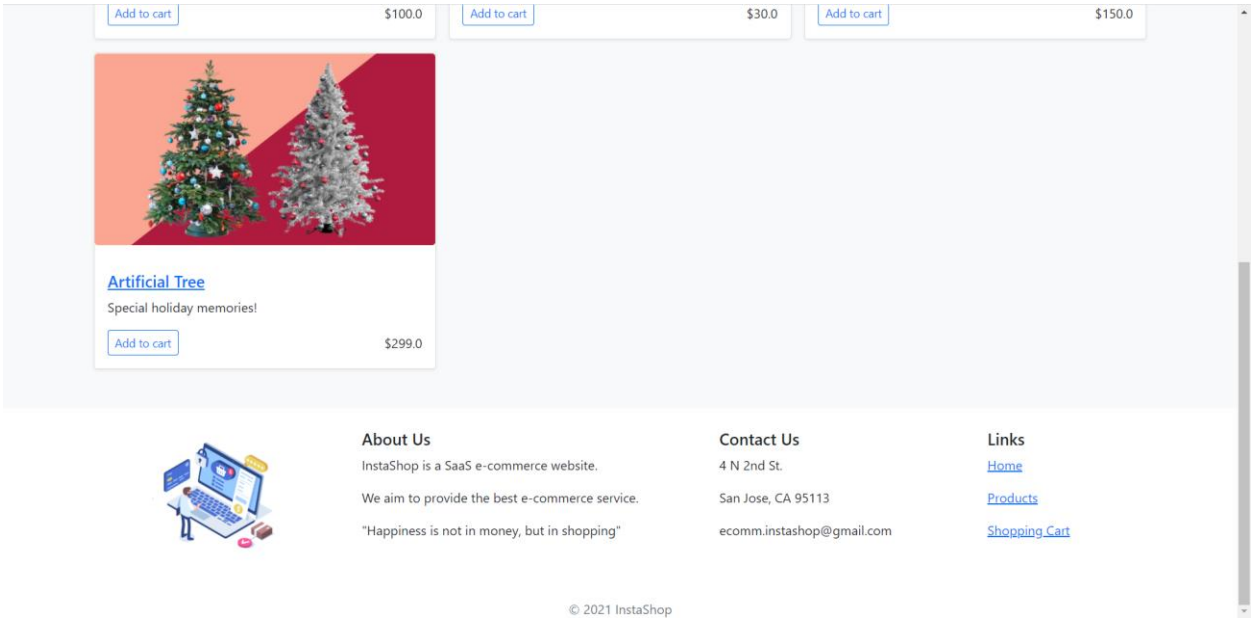


Home Page with user logged in

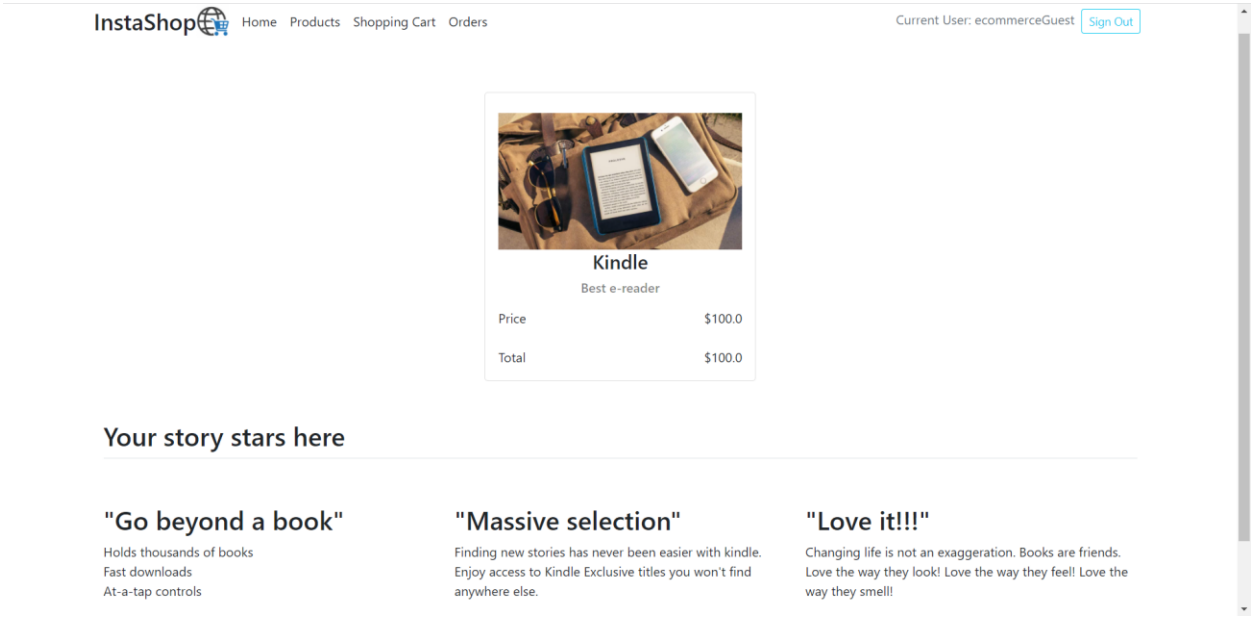


Products Page





Product Details Page



InstaShop Home Products Shopping Cart Orders

Current User: ecommerceGuest Sign Out

Echo

Smart Speaker

Price $30.0

Total $30.0

Meet Echo

"Alexa, play top hits"

Echo combines premium sound, a built-in Zigbee smart home hub, and a temperature sensor. Powerful speakers deliver clear highs, dynamic mids, and deep bass for rich,

"Designed to protect your privacy"

Alexa and Echo devices are built with multiple layers of

"Alexa is happy to help!"

Make your life easier at home. Set timers, ask questions, add items to lists, and create calendar events and reminders. Check the traffic and weather, or play the

InstaShop Home Products Shopping Cart Orders

Current User: ecommerceGuest Sign Out

Watch

The best gift

Price $150.0

Total $150.0

The ultimate device for life

"Full screen ahead"

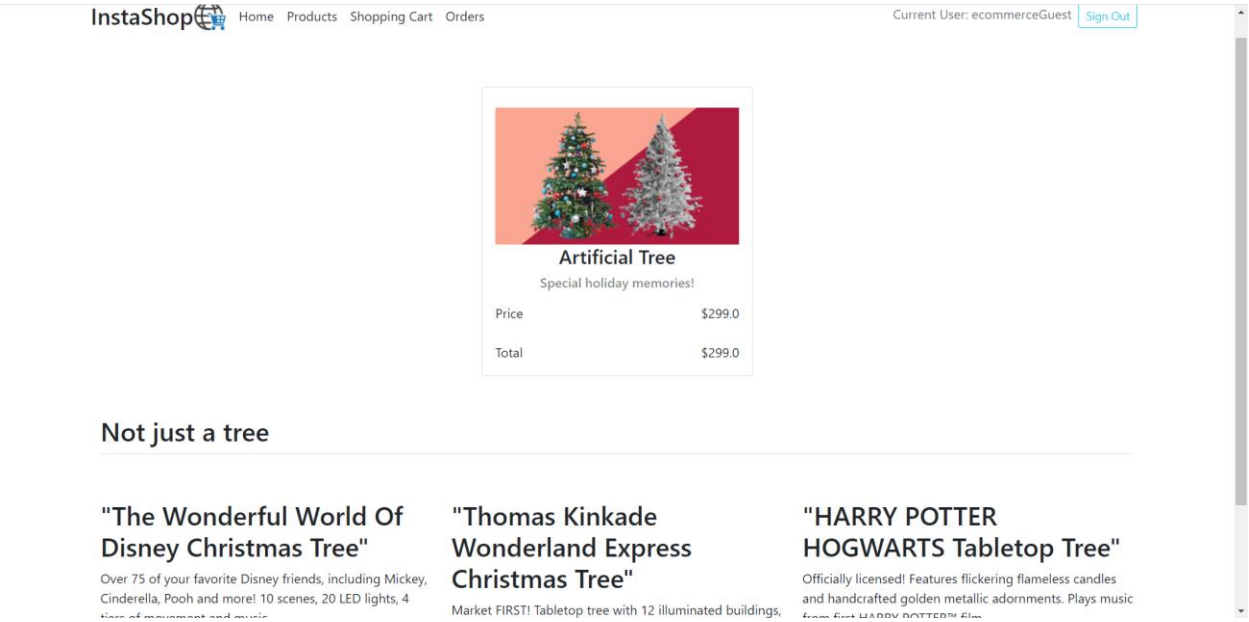
Apple Watch can do what your other devices can't because it's on your wrist. When you wear it, you get a fitness partner that measures all the ways you move,

"Track all the ways you're active"

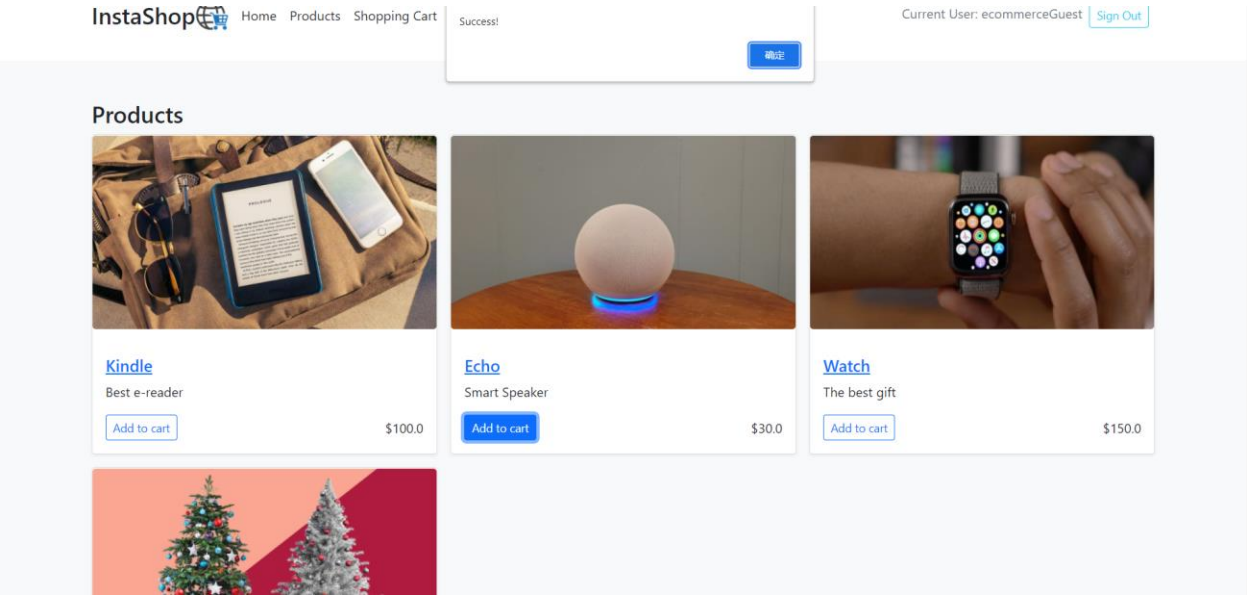
Activity rings show your daily activity. Make it your goal

"Stay in touch with just a tap"

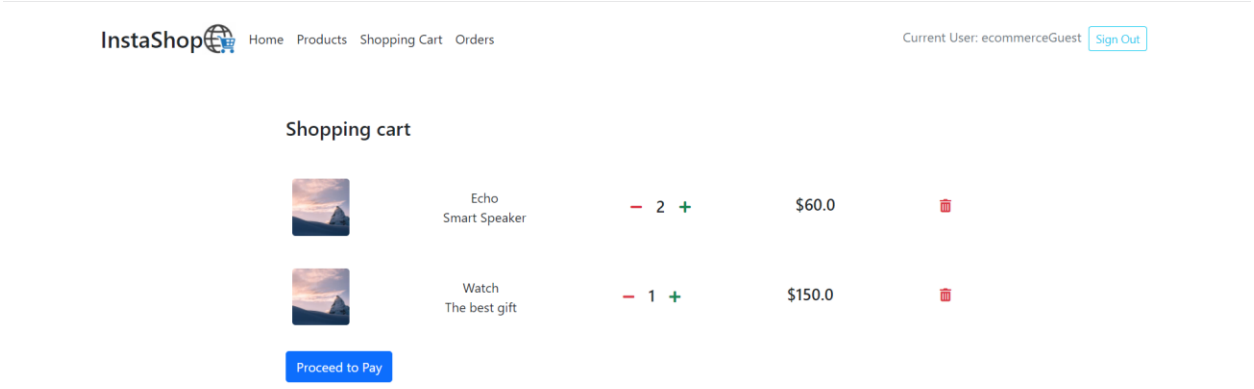
Text, make calls, and listen to music with ease. And with



If the user wants to buy one product, he or she just need to click ‘add to cart’ button and then receive a prompt for success.

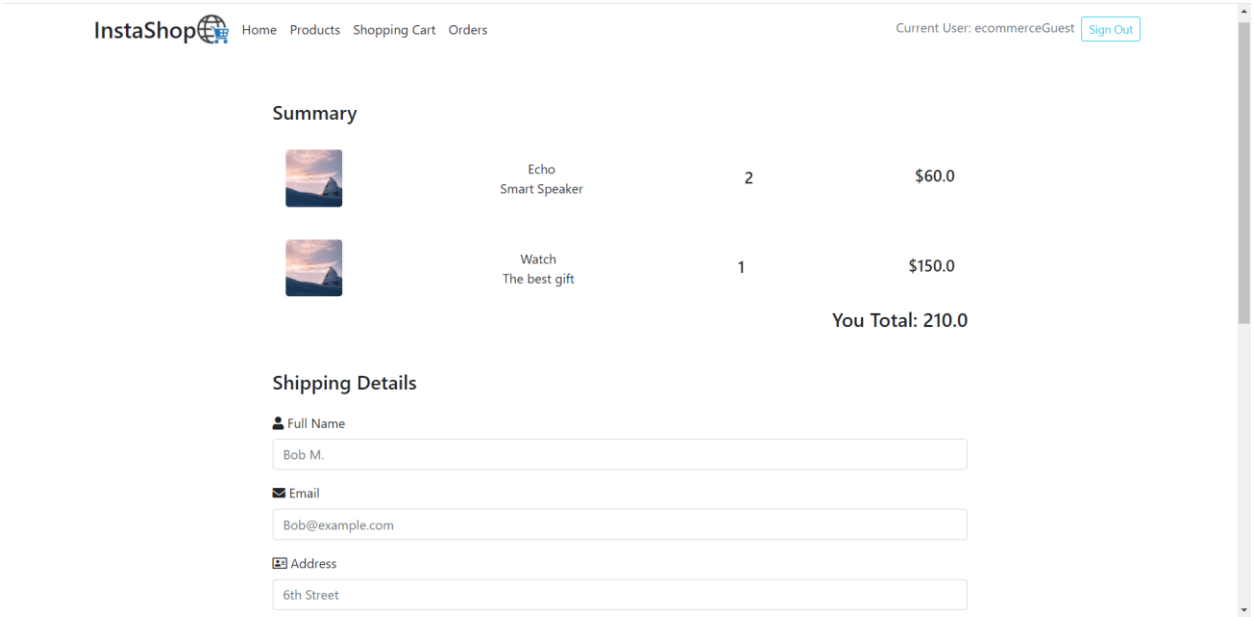


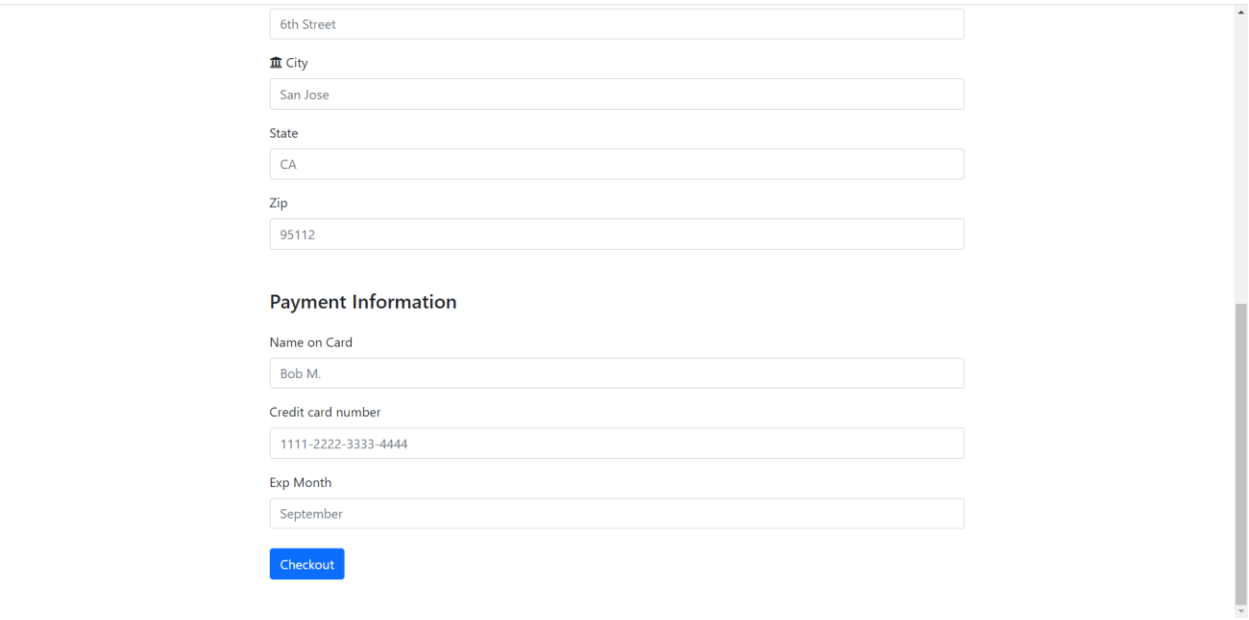
Then, you can see the products are already in the shopping cart.



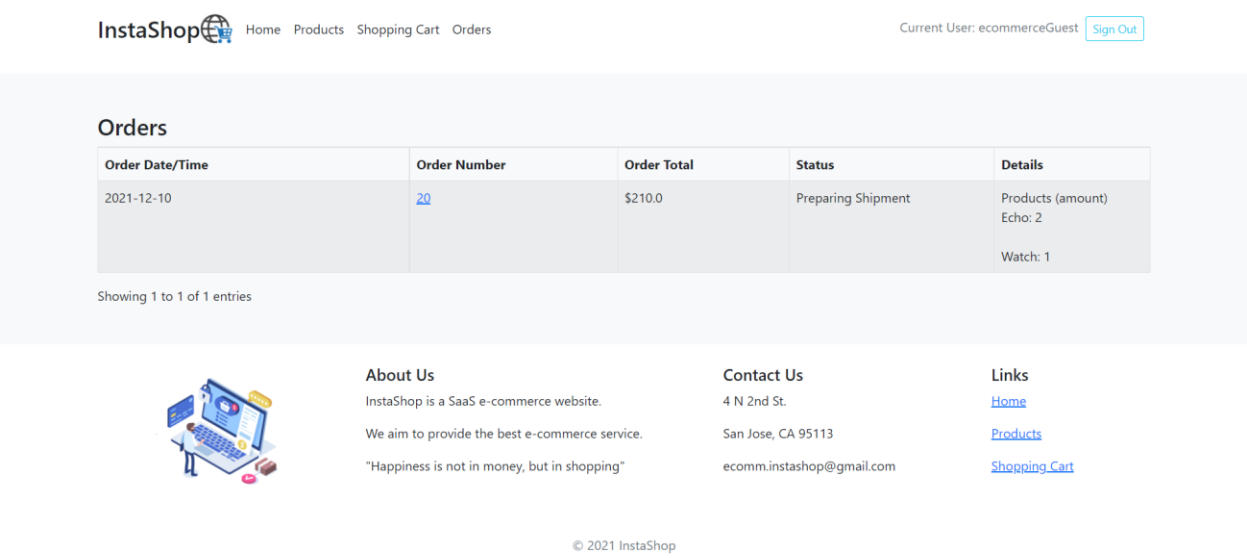
After click ‘Proceed to Pay’ button, you will be redirected to the checkout page.

The checkout page will show you the summary of the order, and require the users filling shipping details and payment information.



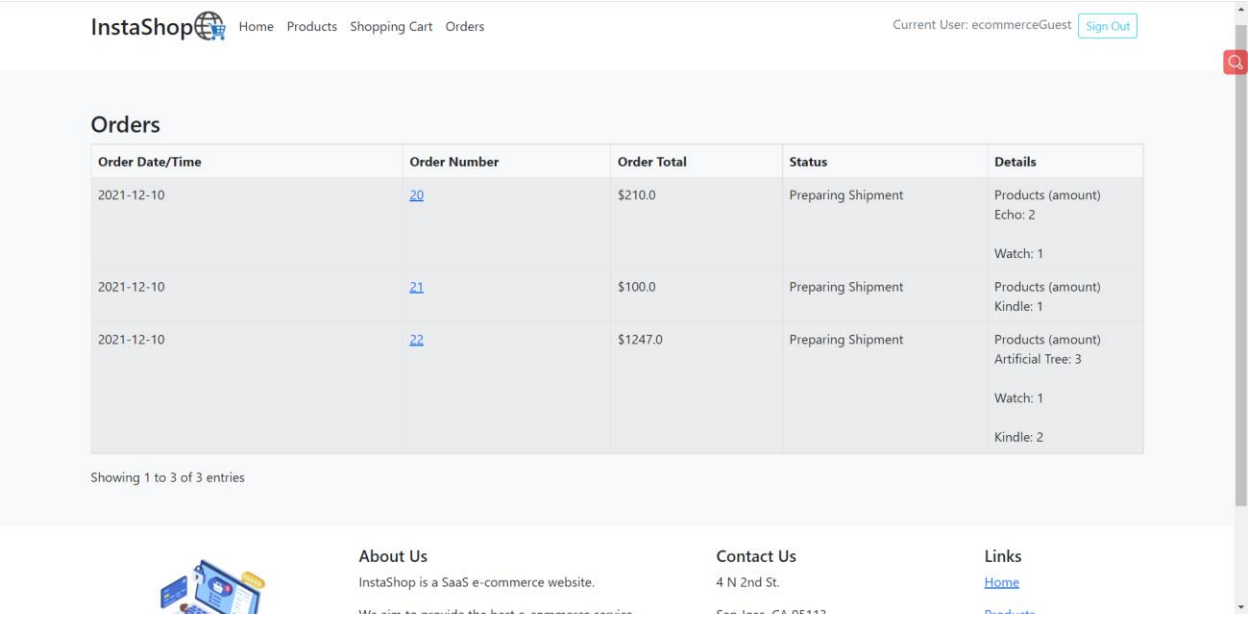


Then, you can see what you have ordered in the order page!



The order page contains the information including order time, order ID, total and the products details for what you have bought.

Enjoy your shopping trip at InstaShop! Add more items to your shopping cart!



Test Plan & Expected results

Test case 1: test the Sign up feature.
expected result: Users can sign up by using their username and password

Test case 2: test the sign in feature
expected result: Users can sign in by using the credentials that they signed up

Test case 3: click ‘Home’, ‘Products’, ‘Shopping Cart’, ‘Orders’ buttons
expected result: These web pages load correctly.

Test case 4: At home page, click each product of top picks
Expected result: Each product information is displayed on the new page and web page is loaded correctly

Test case 5: Click ‘Products’, add each product to the cart
Expected result: all these products can be added to the cart.

Test case 6: Click ‘Shopping Cart’
expected result: shopping cart should have the products that customers chose in the ‘Products’ page. Shopping cart should have the product name, product quantity and the total price of each product. In addition, customers can click the ‘+’ and ‘-’ to increase and decrease the quantity of product.

Test case 7: Click the ‘proceed to pay’ button on the checkout section. Fill out the shipping details and payment information
Expected result: the checkout feature should work properly. Customers click ‘checkout’ button, then this order should be placed successfully

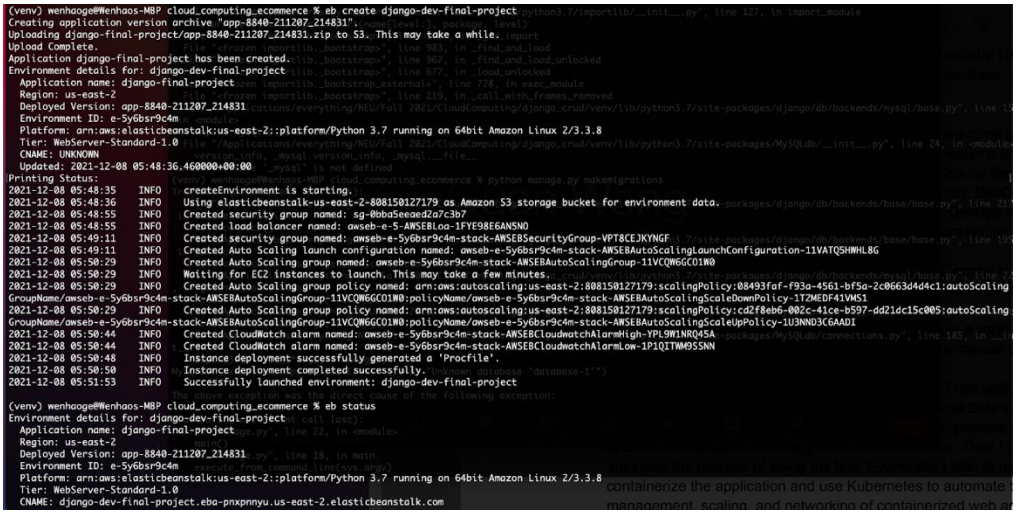
Take away

1. We learned to use Django to develop a web application
2. We learned to use Zappa commands to deploy the web application
3. We learned to use Elastic Beanstalk to deploy the web application
4. We learned to use containerization, amazon rds and build CI/CD pipeline to deploy the web application

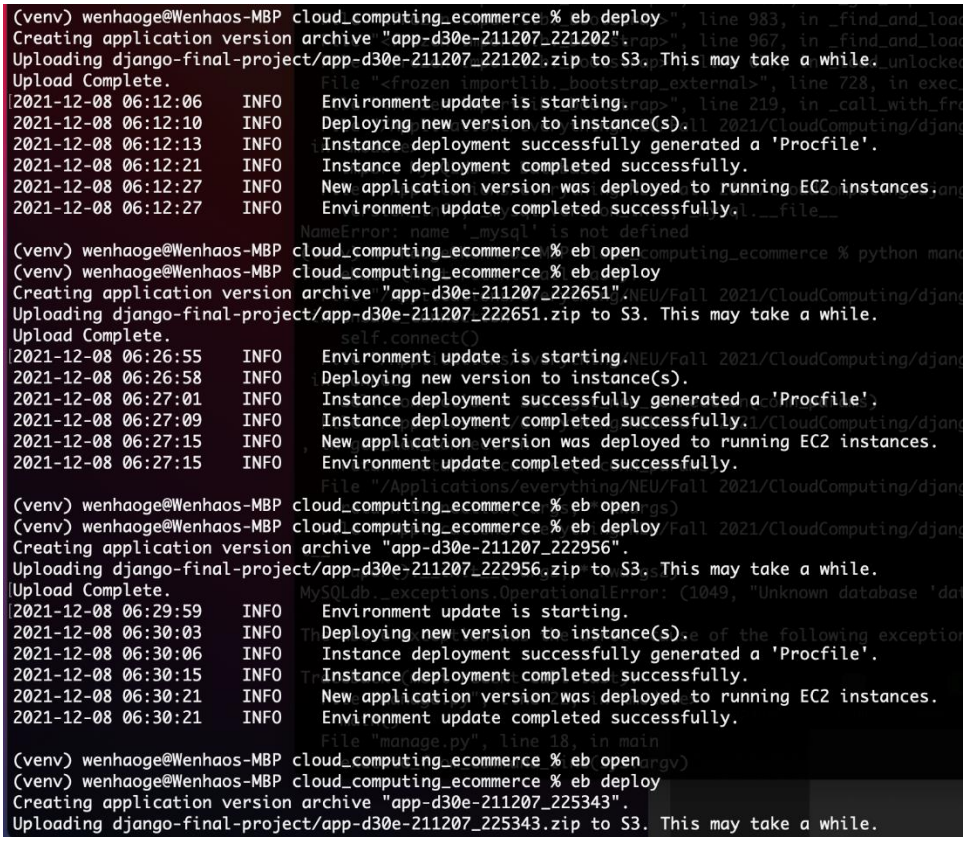
Additional effort

Besides the above-mentioned success deployment, we also tried other approaches to deploy the web app. For instance, we tried to use zappa and Elastic Beanstalk to deploy the web app. We really learned a lot via the overall deployment process. Thus, we also want to mention it here as additional effort.

Use Elastic Beanstalk to deploy the web app:
Use ‘eb create’ command to create the elastic Beanstalk environment, ec2 instance and the S3 bucket.



Use ‘eb deploy’ command to deploy the new changes to the environment. Then use the ‘eb open’ command to open the URL in the browser. But the URL did not open.



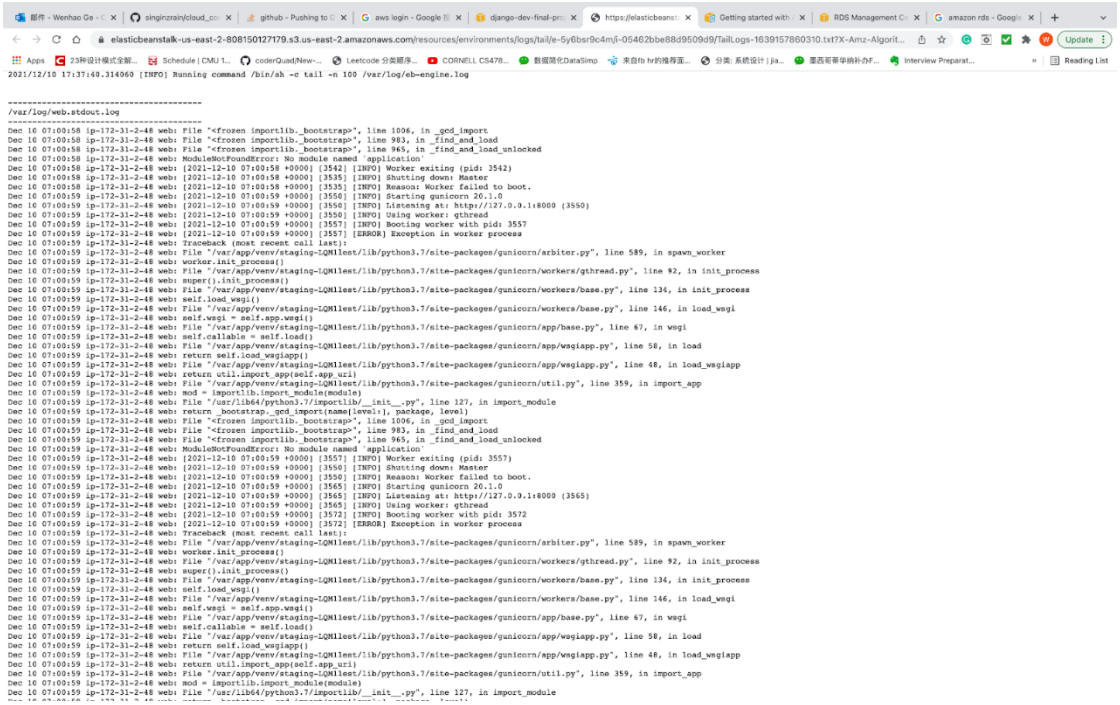
In the logs, we got the errors in the next picture. Then we added a Procfile in the project directory, and added this line into the file:

```
web: gunicorn --bind :8000 --workers 3 --threads 2
```

In addition, we added this line in django.config file:

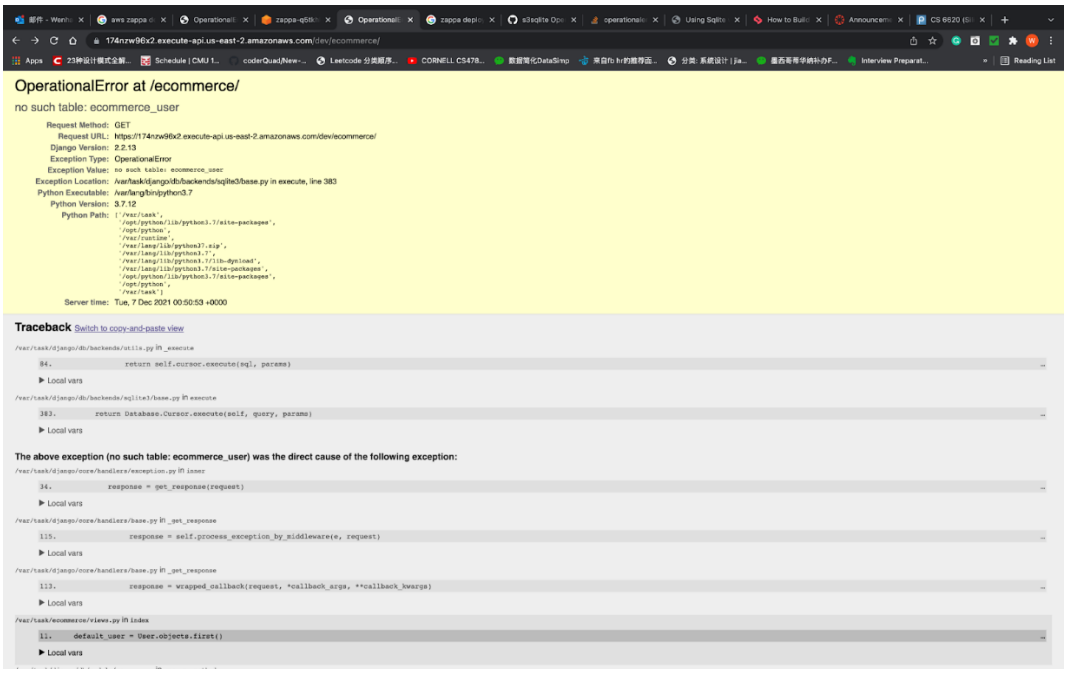
```
WSGIPath: hw3_ecommerce.wsgi.application
```

Then we ran ‘eb deploy’ again, but the problem persisted. When we ran ‘eb open’, there was no URL open in the browser.



Use Zappa to deploy the web app:

We ran ‘zappa init’, ‘zappa update dev’, ‘zappa deploy dev’ to deploy the web app. The deployment is live, and we got a URL. we opened the URL in the browser, then we got this error in the browser.



We think the reason is the database was not deployed successfully. Then we took steps to deploy the database.

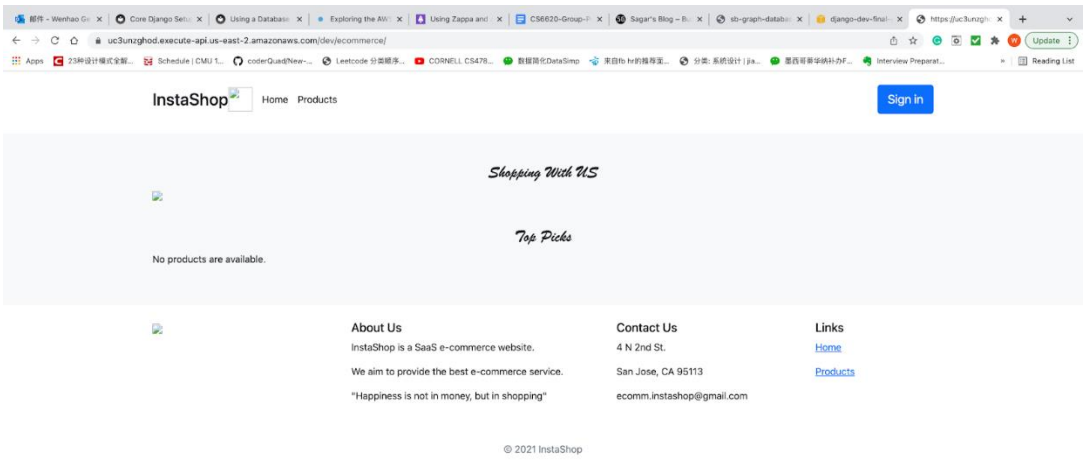
We ran ‘pip install django_s3_sqlite’ and ‘pip install django_s3_storage’. We created a MySQL database using AWS RDS. Then we ran ‘python manage.py makemigrations’, we got an error: NameError: name ‘_mysql’ is not defined. Our solution is to run ‘pip install pymysql’ and ‘brew install mysql’, then this problem was resolved. Later, we met with this kind of error:

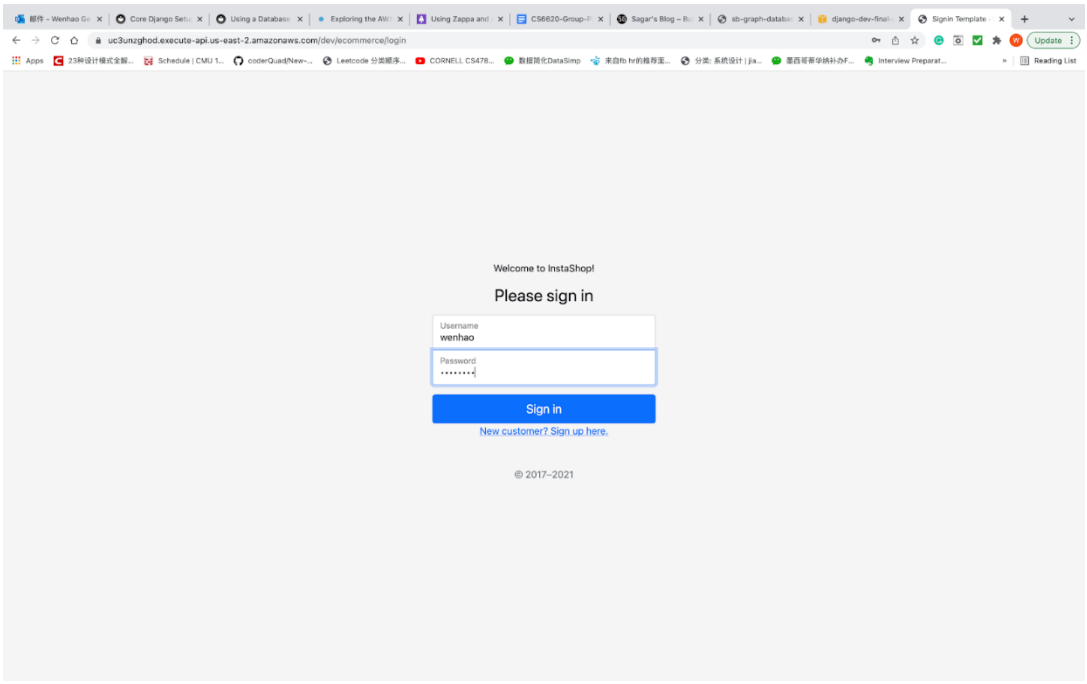
django.db.utils.OperationalError: (1049, "Unknown database"). we fixed this issue too. In addition, when we ran ‘python manage.py makemigrations’ command, we was met with access denied related issues. we rewrote the correct database name, database username and database password in settings.py file. Then we solved these accesses denied issues.

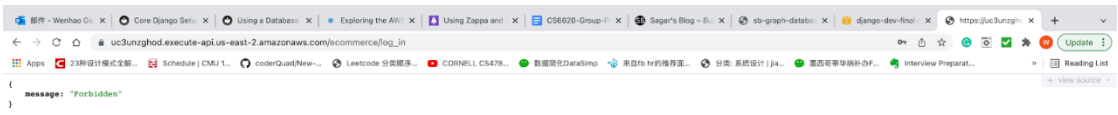
The deployed URL is

https://uc3unzghod.execute-api.us-east-2.amazonaws.com/devhttps://uc3unzghod.execute-api.us-east-2.amazonaws.com/dev

The following images shows that something was not loaded correctly. we clicked the ‘Sign in’ button, we typed in the username and password, but we were not able to sign in. we saw ‘Forbidden’ on the screen.






uc3unzghod.execute-api.us-east-2.amazonaws.com/ecommerce/log_in
Update
Apps
Reading List
{
message: "Forbidden"
}

Executive Summary

As an overview of this project, we would thank our teammates, Professor Prasad and our teaching assistants first. Without their help, no individual could have achieved so much in just one semester. Through this project, our team have a clear division of labor of each task. We split the whole task into some group tasks and some light individual work. In summary, the process of the Ecommerce SaaS web application includes designing, implementing UI section, connecting to the database and deployment. This experience would be helpful for our understanding of Cloud Computing and build long-term memory.

For the learning outcome – Concept, CICD is the combined practices of continuous integration (CI) and either continuous delivery or continuous deployment (CD). CI/CD bridges the gaps between development and operation activities and teams by enforcing automation in building, testing and deployment of applications. The process contrasts with traditional methods where all updates were integrated into one large batch before rolling out the newer version. Modern day DevOps practices involve continuous development, continuous testing, continuous integration, continuous deployment and continuous monitoring of software applications throughout its development life cycle. The CI/CD practice, or CI/CD pipeline, forms the backbone of modern-day DevOps operations. And for the view part, we have a better understanding of how Django works (with SQLite other storage service, for example RDS) and how it is deployed.

Through the process of deployment and some programming, we understand how the ECS, S3, Lambda, git, code pipeline, Load balancer work better. Also, lots of effort could be favorable to SQL operation and the relationships between various of services of AWS. Through the comprehensive application of them, we believe, in the future, we can cooperate with team members, friends or colleagues to complete larger projects in a more efficient way using what we learnt in this course. We’re very grateful for the experience!