Contents

[Section1: Data Pipelines 2](#_Toc120392007)

[Section2: Databases 2](#_Toc120392008)

[Setting up post-gres Docker 2](#_Toc120392009)

[ERD 3](#_Toc120392010)

[Queries 5](#_Toc120392011)

[Section3: System Design 6](#_Toc120392012)

[Design 1 6](#_Toc120392013)

[Design 2 7](#_Toc120392014)

[Section 4: Charts & APIs 11](#_Toc120392015)

[**Section 5: Machine Learning** 12](#_Toc120392016)

[Results 12](#_Toc120392017)

## Section1: Data Pipelines

Refer to Git Hub Section 1 Folder

## Section2: Databases

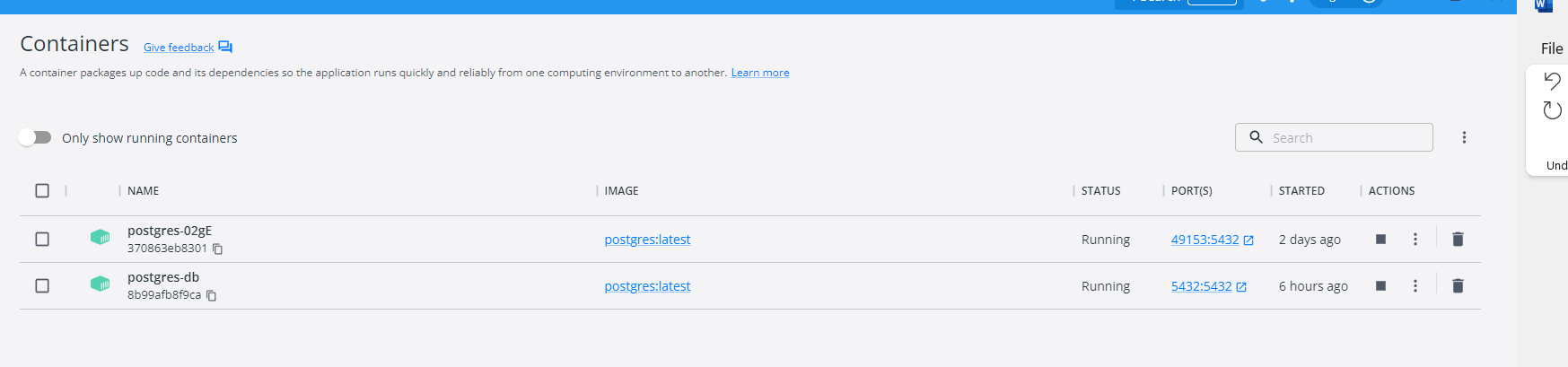
### Setting up post-gres Docker

1. Download Docker Hub on Windows

Graphical user interface, text, application, email

Description automatically generated

1. Download the postgres container.



1. On command prompt, run the following command:

docker run --name postgres-db -e POSTGRES\_PASSWORD=docker -p 5432:5432 -d postgres

* -d means that you enable Docker to run the container in the background
* -p plus the port numbers means you map the containers port 5432 to the external port 5432 - this allows you to connect to it from the outside
* POSTGRES\_PASSWORD sets the password to docker. This is the password that gives you access to your database
* the —name property gives your container a name and means you can easily find it back

1. We can connect to this docker instance using the following:

* Host: localhost
* Port: 5432

User: postgres

* Password: docker

1. Create a docker file:”Dockerfile” and add the following to it:
2. Ddl\_ecommerce.sql contains the ddl commands to create tables.

FROM postgres

ENV POSTGRES\_PASSWORD docker

ENV POSTGRES\_DB world

COPY ddl\_ecommerce.sql /docker-entrypoint-initdb.d/

Graphical user interface, application

Description automatically generated

1. Create the image using the following:

docker build -t my-postgres-db ./

The name of the image is “my-postgres-db”

docker images -a

1. Run the image using the following:

docker run -d --name my-postgresdb-container -p 5432:5432 my-postgres-db

1. After use remove the image using the below command:

docker image rm 'my-postgres-db '

### ERD

Table

Description automatically generated with low confidence

|  |  |
| --- | --- |
| Table Name: Product | |
| Column Name | Data Type |
| Item\_no | NUMBER(UNIQUE KEY) |
| item\_name | VARCHAR(50) |
| Manufacturer\_name | VARCHAR(50) |
| cost | FLOAT(2) |
| weight | NUMBER |

|  |  |
| --- | --- |
| Table Name: Orders | |
| Column Name | Data Type |
| membership\_id | NUMBER(UNIQUE KEY) |
| Order\_id | NUMBER |
| Date\_ordered | TIMESTAMP |
| Date\_required | TIMESTAMP |
| Status | VARCHAR(10) |
| Total\_cost | FLOAT(2) |

|  |  |
| --- | --- |
| Table Name: OrderDetails | |
| Column Name | Data Type |
| Order\_id | NUMBER(UNIQUE KEY) |
| Item\_no | NUMBER |
| quantity | NUMBER |
| price | FLOAT(2) |
| Total\_price\_per\_item | FLOAT(2) |
| Total\_weight | NUMBER |

### Queries

1. Which are the top 10 members by spending

select

o.membership\_id

sum(od.quantity \* od.total\_price\_per\_item) as total\_spend

from

Orderdetails od

INNER JOIN

Orders o

ON od.order\_id = o.order\_id

group by o.membership\_id

order by sum(od.quantity \* od.total\_price\_per\_item) desc

limit 10

1. Which are the top 3 items that are frequently brought by members

select

p.item\_name ,

count(o.membership\_id) as user\_count

from

Orderdetails od

INNER JOIN

Orders o

ON od.order\_id = o.order\_id

INNER JOIN

product p

ON od.item\_no = p.item\_no

group by p.item\_name

order by count(o.membership\_id) desc

limit 3

## Section3: System Design

### Design 1

**SYSTEM DESIGN**

**DATABASE**

**Requests**

Web Server 3

Web Server 2

Platform Server 3

Platform Server 2

Platform Server 1

Web Server 1

Order Details

Order

Product

Load Balancer

**Requests**

**Requests**

Logistics

Analytics

Sales

**Logistics**

* For the logistics team, it can be a batch job running hourly/daily to update the total items bought ad the completed transactions in the Order table (Fields:status)
* Provide read and write access to database.

**Sales**

* The database design must be normalised and concurrent updates must be taken care of.
* Provide read and write access to database.

**Analytics**

* Provide tools such as Tableau etc to visualize the data. Indexing must be performed to the tables to perform efficient retrieval of data.
* Only provide Read access to the database

**Design Considerations:**

* Database must be Simple and functional.
* High performance and the Database queries execute quickly to facilitate efficient retrieval without latency
* Database should have good indexing and performance optimization options.
* High availability and scalability: The database design should be highly available with automatic snapshots and enables automatic scaling to support future platform growth as well as sudden traffic spikes.

### Design 2

Diagram

Description automatically generated

1. The Web application containing the Image processor will upload the images to S3 bucket. The API Gateway is used to invoke the API, a lambda function can be written to upload the images to S3.
2. The Kafka Web App pushes the images to the Amazon MSK.
3. The Kafka Consumer consumes the images.
4. The images are stored in S3. The EC2 instance processes the images using the already written code.
5. Using S3 lifecycle management, after 7 days the images moved into S3 Glacier and then purges from the S3 bucket.
6. Using Analytics in AWS we can create dashboard and company’s analysts can access and perform analytical computation on the stored data. AWS Quicksight can also be used to perform analysis of the data.

**AWS Best Practices:**

**Securing access to the environment and its resources as the company expands**

1. When you created an AWS account, you specified an email address and password you use to sign in to the [AWS Management Console](https://console.aws.amazon.com/console/home). To prevent your root account from being compromised, which is an important safeguard because your root account has access to *all*services and resources in your account.Create a strong password for your AWS resources.
2. Set a strong password with a combination of letters, numbers, and special characters.
3. Use a group email alias with your AWS account. If for any reason you are unavailable to respond to an AWS notification or manage your AWS Cloud workloads, using a group email alias with your AWS account means other trusted members of your organization can manage the account in your absence
4. Enable multi-factor authentication. [Multi-factor authentication](https://aws.amazon.com/iam/details/mfa/) (MFA) is a security capability that provides an additional layer of authentication on top of your user name and password. When using MFA, after you sign in with your user name and password (*what you know*), you must also provide an additional piece of information that only you have physical access to (*what you have*), which can come from a dedicated MFA hardware device or an app on a phone.
5. Set up AWS IAM users, groups, and roles for daily account access
6. To manage and control access and permissions to your AWS resources, use AWS Identity and Access Management (IAM) to create [users](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html), [groups](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_groups.html), and [roles](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html). When you create an IAM user, group, or role, it can access only the AWS resources to which you explicitly grant permissions, which is also known as [least privilege](http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege).
7. If you are the account owner, AWS recommends that you create an IAM user for yourself for daily use of your resources. See [How do I set up an IAM user and sign in to the AWS Management Console using IAM credentials?](https://www.youtube.com/watch?v=XMi5fXL2Hes) and [Now Create and Manage Users More Easily with the AWS IAM Console](https://aws.amazon.com/blogs/security/now-create-and-manage-users-more-easily-with-the-aws-iam-console/).
8. Delete your account’s access key. You can allow programmatic access to your AWS resources from the command line or for use with AWS APIs. However, AWS recommends that you do not create or use the *access keys* associated with your root account for programmatic access. In fact, if you still have access keys, delete them. Instead, create an IAM user and grant that user only the permissions needed for the APIs you are planning to call. You can then use that IAM user to issue access keys.
9. Enable CloudTrail in all AWS regions. You can track all activity in your AWS resources by using [AWS CloudTrail](https://aws.amazon.com/cloudtrail/). Even if you initially do not know how to use CloudTrail, turning it on now can help AWS Support and your AWS solutions architect later if they need to troubleshoot a security or configuration issue. To enable CloudTrail logging in all AWS regions, see [AWS CloudTrail Update – Turn On in All Regions and Use Multiple Trails](https://aws.amazon.com/blogs/aws/aws-cloudtrail-update-turn-on-in-all-regions-use-multiple-trails/). To learn more about CloudTrail.

**Security of data at rest and in transit**

Amazon S3 and DynamoDB are tightly integrated with IAM. It can be managed by using ACL-type permissions to individual resources at the platform level, or permissions based on user identity or user responsibility at the IAM user/group level.

Platform-provided encryption can be used for data at rest, or platform-provided HTTPS encapsulation for the payloads for protecting your data in transit to and from the service.

**Scaling to meet user demand while keeping costs low**

Autoscaling or On-demand is possible for every resource in AWS.

**Maintenance of the environment and assets (including processing scripts)**

For Amazon S3 File Gateway, you configure file shares that are mapped to selected S3 buckets or S3 prefixes, using IAM roles. You can monitor the status of your data transfer and your storage interfaces through the AWS Management Console.

Amazon S3 Lifecycle can be used to manage S3. An S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects(Images in our case).

S3 Bucket’s Transition actions can be defineed when objects transition to another storage class. For example, we need to transition objects to the S3 Standard-IA storage class after 7 days after creating them, or archive objects to the S3 Glacier Flexible Retrieval storage.

For Aws Glacier S3’s Expiration actions define when objects expire. Amazon S3 deletes expired objects on your behalf.

The following are the additional best practices while using the cloud:

|  |  |
| --- | --- |
| **Design Property** | **Description** |
| Manageability | The AWS Management & Governance product suite allows you to enable, provision, and operate AWS resources to determine the health and predictability of your cloud workloads. |
| Scalability | AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, it's easy to setup application scaling for multiple resources across multiple services in minutes. |
| Secure | Safeguard your passwords and access keys.  Activate multi-factor authentication (MFA) on the AWS account root user and any users with interactive access to AWS Identity and Access Management (IAM)  Limit AWS account root user access to your resources.  Audit IAM users and their policies frequently. |
| High Availability | Amazon S3 [storage classes](https://aws.amazon.com/s3/storage-classes/) replicate their data on more than three Availability Zone (except for S3 One Zone-Infrequent Access). Amazon S3 maintains redundancy even within one of the facilities in a single Availability Zone. For example, Amazon S3 replicates data across multiple disks, so even if one of them fails, customers can still access their data with no downtime. |
| Elastic | Elasticsearch accomplishes its super-fast search capabilities through the use of a Lucene-based distributed reverse index. When a document is loaded to Elasticsearch, it creates a reverse index of all the fields in that document. A reverse index is an index where each of the entries is mapped to a list of documents that contains them. Data is stored in JSON form and can be queried using the proprietary query language. |
| Fault Tolerant and Disaster Recovery |  |
| Efficient | Using AWS, you can go global in minutes and deploy resources in multiple locations across the globe to be closer to your end users. You can also dynamically add read-only replicas to information stores (such as database systems) to reduce the load on the primary database. |
| Low Latency | S3 Multi-Region Access Points in Action  We can provide lower latency in AWS with the help of cross-region replication because it uses to replicate the same object to all regions so the end-user can access the object by nearest region. |
| Least Privilege | [AWS Identity and Access Management (IAM)](http://aws.amazon.com/iam) [policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html) [Least privilege](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege) is a principle of granting only the permissions required to complete a task. Least privilege is also one of many [Amazon Web Services (AWS)](http://aws.amazon.com/) [Well-Architected](https://aws.amazon.com/architecture/well-architected/) best practices that can help you build securely in the cloud. For example, if you have an [Amazon Elastic Compute Cloud (Amazon EC2)](http://aws.amazon.com/ec2) instance that needs to access an [Amazon Simple Storage Service (Amazon S3)](http://aws.amazon.com/s3) bucket to get configuration data, you should only allow read access to the specific S3 bucket that contains the relevant data. |

## Section 4: Charts & APIs

Refer to Git Hub Section 4 Folder for Code

Chart, line chart

Description automatically generated

Chart, histogram

Description automatically generated

**Section 5: Machine Learning**

Refer to Git Hub Section 5 Folder for Code

### Results

|  |  |
| --- | --- |
| **Algorithm** | **Accuracy** |
| Logistic Regression | 0.69 |
| Naïve Bayes | 0.70 |
| NearestNeighborClassifier | 0.92 |
| MLP Classifier | 0.93 |
| DecisionTreeClassifier | 0.83 |
| RandomForestClassifier | 0.84 |
| XGBClassifier | 0.84 |