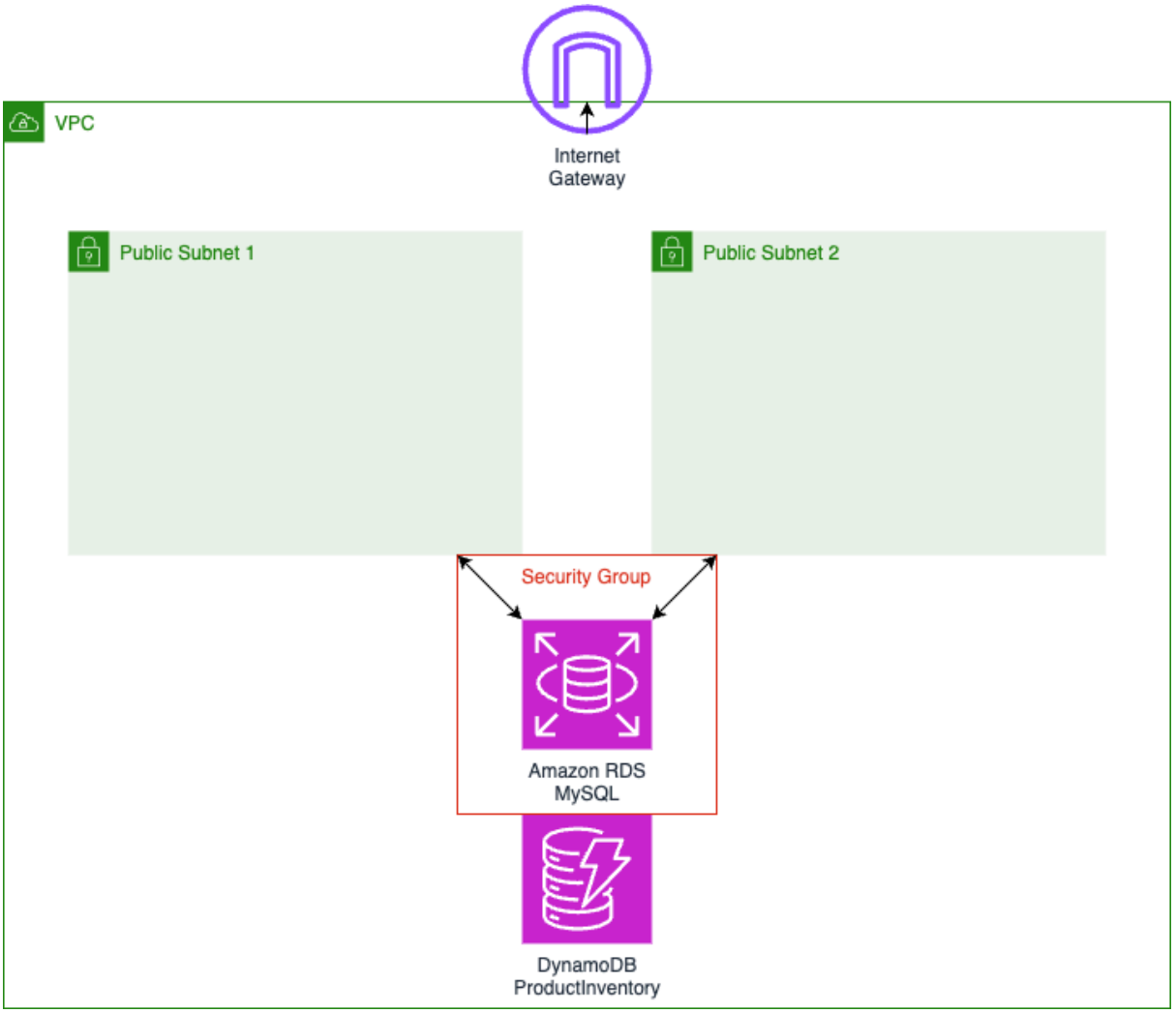
Day 4 real flow

Prepare Prod.json for parameters,database.yaml for CloudFormation template.



graph TB
subgraph "VPC (10.0.0.0/16)"
subgraph "Public Subnets"
PS1[Public Subnet 1
10.0.7.0/24] --> |3306| RDS
PS2[Public Subnet 2
10.0.8.0/24] --> |3306| RDS
end

```
graph TB
   subgraph "VPC (10.0.0.0/16)"
```

```
    subgraph "Public Subnets"
      PS1[Public Subnet 1<br>10.0.7.0/24] --> |3306| RDS
      PS2[Public Subnet 2<br>10.0.8.0/24] --> |3306| RDS
    end

    subgraph "Database Layer"
      RDS[RDS MySQL<br>db.t3.micro<br>20GB Storage] --> |Backup| S
3[Daily Backups<br>7 Day Retention]
      SG[Security Group<br>Port 3306<br>Source: 0.0.0.0/0]
      RDS --> SG
    end

    subgraph "NoSQL Layer"
      DDB[DynamoDB<br>ProductInventory<br>On-Demand Capacity]
      DDB --> |Stream| DDBStream[DynamoDB Stream<br>NEW_AND_OL
D_IMAGES]
    end
  end
```

prod.json

```
[
  {
    "ParameterKey": "Environment",
    "ParameterValue": "prod"
  },
  {
    "ParameterKey": "DBPassword",
    "ParameterValue": "YourSecurePassword123!"
  },
  {
    "ParameterKey": "VpcCidr",
    "ParameterValue": "10.0.0.0/16"
  }
 ]
```

1. Then create a new stack with the updated template:

```
aws cloudformation create-stack \\
  --stack-name prod-database-stack \\
  --template-body file://templates/database.yaml \\
  --parameters file://parameters/prod.json \\
  --capabilities CAPABILITY_IAM
```

Here's the updated template with proper deletion policies:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Day 4 - Database Infrastructure'

Parameters:
  DBPassword:
    Type: String
    NoEcho: true
    Description: Database admin password
    MinLength: 8

  Environment:
    Type: String
    Default: prod
    AllowedValues: [dev, staging, prod]

  VpcCidr:
    Type: String
    Default: 10.0.0.0/16
    Description: CIDR block for VPC

Resources:
  # VPC Resources
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCidr
      EnableDnsHostnames: true
      EnableDnsSupport: true
      Tags:
        - Key: Name
```

```
        Value: !Sub ${Environment}-vpc

# Public Subnets
PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.7.0/24  # Changed CIDR
    AvailabilityZone: !Select [0, !GetAZs '']
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${Environment}-public-subnet-1

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.8.0/24  # Changed CIDR
    AvailabilityZone: !Select [1, !GetAZs '']
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${Environment}-public-subnet-2

# Internet Gateway
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
      - Key: Name
        Value: !Sub ${Environment}-igw

AttachGateway:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
```

```
# Route Tables
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${Environment}-public-rt

PublicRoute:
  Type: AWS::EC2::Route
  DependsOn: AttachGateway
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet1
    RouteTableId: !Ref PublicRouteTable

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable

# DB Security Group
DBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Database security group
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
```

```
        FromPort: 3306
        ToPort: 3306
        CidrIp: 0.0.0.0/0
      Tags:
       - Key: Name
         Value: !Sub ${Environment}-db-sg

# DB Subnet Group
DBSubnetGroup:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: Subnet group for RDS
    SubnetIds:
      - !Ref PublicSubnet1
      - !Ref PublicSubnet2
    Tags:
      - Key: Name
        Value: !Sub ${Environment}-db-subnet-group

# RDS Instance
PrimaryDB:
  Type: AWS::RDS::DBInstance
  DependsOn: AttachGateway
  Properties:
    DBInstanceIdentifier: !Sub ${Environment}-product-db
    Engine: mysql
    DBInstanceClass: db.t3.micro
    AllocatedStorage: 20
    MasterUsername: admin
    MasterUserPassword: !Ref DBPassword
    BackupRetentionPeriod: 7
    MultiAZ: false
    VPCSecurityGroups:
      - !Ref DBSecurityGroup
    DBSubnetGroupName: !Ref DBSubnetGroup
    PubliclyAccessible: true
    DeletionProtection: false
    Tags:
```

```
        - Key: Environment
          Value: !Ref Environment

  # DynamoDB Table
  ProductInventory:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: !Sub ${Environment}-ProductInventory
      BillingMode: PAY_PER_REQUEST
      AttributeDefinitions:
        - AttributeName: product_id
          AttributeType: S
        - AttributeName: warehouse_id
          AttributeType: S
      KeySchema:
        - AttributeName: product_id
          KeyType: HASH
        - AttributeName: warehouse_id
          KeyType: RANGE
      StreamSpecification:
        StreamViewType: NEW_AND_OLD_IMAGES
      Tags:
        - Key: Environment
          Value: !Ref Environment

Outputs:
  VpcId:
    Description: VPC ID
    Value: !Ref VPC

  PrimaryDBEndpoint:
    Description: Primary DB Endpoint
    Value: !GetAtt PrimaryDB.Endpoint.Address

  DynamoDBTableName:
    Description: DynamoDB Table Name
    Value: !Ref ProductInventory
```

Create Stack

```
aws cloudformation create-stack \
  --stack-name prod-database-stack \
  --template-body file://templates/database.yaml \
  --parameters file://parameters/prod.json \
  --capabilities CAPABILITY_IAM
```

```
# Validate template
aws cloudformation validate-template \
  --template-body file://templates/database.yaml
```

Current Architecture

```
graph TB
    subgraph VPC[10.0.0.0/16]
        subgraph Public Subnets
            PS1[Public Subnet 1<br>10.0.7.0/24] --> RDS
            PS2[Public Subnet 2<br>10.0.8.0/24] --> RDS
        end
        RDS[RDS MySQL<br>db.t3.micro] --> SG[Security Group<br>Port 330
6]
        DDB[DynamoDB<br>ProductInventory]
    end
    IG[Internet Gateway] --> VPC
```

1. Get RDS Endpoint

```
aws cloudformation describe-stacks \\
  --stack-name prod-database-stack \\
  --query 'Stacks[0].Outputs[?OutputKey==`PrimaryDBEndpoint`].OutputVal
ue' \\
  --output text
```

2. Database Schema Deployment

Connect to your RDS instance and create the schema:

```
mysql -h <your-rds-endpoint> -u admin -p
```

Once connected, execute:

```
-- Create database
CREATE DATABASE IF NOT EXISTS productdb;
USE productdb;

-- Create products table
CREATE TABLE products (
    product_id VARCHAR(36) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    stock INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE C
URRENT_TIMESTAMP
);

-- Create categories table
CREATE TABLE categories (
    category_id VARCHAR(36) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    parent_id VARCHAR(36),
    FOREIGN KEY (parent_id) REFERENCES categories(category_id)
);

-- Create product_categories table
CREATE TABLE product_categories (
    product_id VARCHAR(36),
    category_id VARCHAR(36),
    PRIMARY KEY (product_id, category_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id),
    FOREIGN KEY (category_id) REFERENCES categories(category_id)
);
```

3. Test Database Connectivity and Operations

Test RDS:

```
-- Insert test category
INSERT INTO categories (category_id, name)
VALUES ('cat-001', 'Electronics');

-- Insert test product
INSERT INTO products (product_id, name, description, price, stock)
VALUES ('prod-001', 'Test Product', 'Description', 99.99, 100);

-- Insert product category relationship
INSERT INTO product_categories (product_id, category_id)
VALUES ('prod-001', 'cat-001');

-- Verify data
SELECT p.*, c.name as category_name
FROM products p
JOIN product_categories pc ON p.product_id = pc.product_id
JOIN categories c ON pc.category_id = c.category_id;
```

Test DynamoDB:

```
# Insert test item
aws dynamodb put-item \\
    --table-name prod-ProductInventory \\
    --item '{
        "product_id": {"S": "prod-001"},
        "warehouse_id": {"S": "wh-001"},
        "quantity": {"N": "50"},
        "location": {"S": "Tokyo"}
    }'

# Query item
aws dynamodb get-item \\
    --table-name prod-ProductInventory \\
    --key '{
```

```
    "product_id": {"S": "prod-001"},
    "warehouse_id": {"S": "wh-001"}
}'
```

4. Monitoring Setup

Monitor these metrics in CloudWatch:

- RDS CPU Utilization
- RDS Free Storage Space
- RDS DB Connections
- DynamoDB Read/Write Capacity
- DynamoDB Throttled Requests

5.clean up

aws cloudformation delete-stack --stack-name prod-database-stack