How the Flask Application Works with the AWS Resources

The Flask application is deployed using a set of Terraform scripts that provision and configure various AWS resources. Here's a detailed explanation of how the application interacts with these resources:

Overview

1. **Application Code (`app.py`)**:
    - This is the main Flask application that interacts with DynamoDB.
    - It includes endpoints for reversing and storing IP addresses and a health check endpoint.
2. **Dockerfile**:
    - Used to build a Docker image for the Flask application.
    - The image is pushed to a Docker registry and used by the EC2 instances.
3. **Terraform Scripts**:
    - These scripts provision AWS resources such as VPC, subnets, security groups, EC2 instances, IAM roles, DynamoDB, and an Application Load Balancer (ALB).

Detailed Resource Interaction

1. **VPC and Networking (`vpc.tf`, `subnets.tf`, `route_tables.tf`, `internet_gateway.tf`)**:
    - **VPC**: Creates a Virtual Private Cloud to host the resources.
    - **Subnets**: Defines public and private subnets for different availability zones.
    - **Route Tables**: Configures routing within the VPC.
    - **Internet Gateway**: Provides internet access to resources within the VPC.
2. **Security Groups (`security_groups.tf`)**:
    - **Web Security Group**: Allows inbound traffic on ports 80 (HTTP), 443 (HTTPS), 22 (SSH), and 5000 (application port).
    - **Database Security Group**: Configures access rules for database traffic (if applicable).
3. **IAM Roles and Policies (`iam.tf`)**:
    - **IAM Role for EC2**: Allows EC2 instances to access DynamoDB.
    - **Policy Attachments**: Attaches policies such as AmazonDynamoDBFullAccess to the EC2 IAM role.
4. **DynamoDB Table (`dynamodb.tf`)**:
    - **DynamoDB Table**: Defines the table where reversed IP addresses are stored.
5. **EC2 Instances (`ec2.tf`)**:
    - **Launch Template**: Configures the EC2 instances, including the AMI, instance type, security groups, and user data script.
    - **Auto Scaling Group**: Manages the number of EC2 instances based on demand.
6. **Load Balancer (`elb.tf`)**:
    - **ALB**: Distributes incoming traffic across the EC2 instances.
    - **Listeners and Target Groups**: Configures how the ALB routes traffic to the instances and performs health checks.
7. **User Data Script (`user_data.sh`)**:
    - Installs Docker and starts the Flask application container on EC2 instances.

- Ensures the application is running before the instance is marked as healthy.

Application Flow

1. **User Requests**:
   - Users send HTTP requests to the ALB, which forwards the traffic to the EC2 instances.
2. **Flask Application**:
   - The Flask application handles incoming requests, processes them, and interacts with DynamoDB to store and retrieve data.
3. **DynamoDB Interaction**:
   - The application uses the `boto3` library to interact with DynamoDB.
   - Stores reversed IP addresses in the DynamoDB table.
4. **EC2 Instances and ALB**:
   - EC2 instances run the Docker container with the Flask application.
   - The ALB performs health checks and ensures only healthy instances receive traffic.