Thought Process

The architecture of the application is as follows- Elastic beanstalk was used with an autoscaling group and a load balancer. Elastic Beanstalk was used as I've had experience with it in the past and it seemed like the most straightforward way to deploy this web application, it also allows an easy way for deployment and management of the application- in addition to this, the autoscaling group and load balancer automatically handles capacity provisioning, load balancing if multiple users try to connect at the same time, scaling, and monitoring the health of the application.

The auto scaling group ensures scalability and cost optimization, as only the minimum required amount of instances are spun up based upon user demand. The Load balancer distributes traffic and ensures that high availability is in place. S3 buckets with versioning enabled allow for a simple and straightforward way to rollout updates and for rollbacks to take place.

Maintaining Security:

The Security Groups are defined to only allow necessary traffic (in the form of HTTP/HTTPS requests) to the load balancer, and SSH to the EC2 instances from a specific IP range if needed.
The IAM roles will be given only the minimum required permissions that are necessary for the elastic beanstalk environment to access resources on AWS

Maintaining the Implemented Solution:

In order to update the application, updates would need to be pushed to the S3 bucket, and then updating the elastic beanstalk environment in order to use the required version pushed to the S3 bucket.

Reusability:

The Terraform script can be easily reused for deploying other applications, minimal changes are needed for this.
In order to make the script more robust, parametrization can be used for different environments (using elastic beanstalk or without), and different configurations.

Improvements:

Due to the timeframe of this task, I wasn't able to implement a full CI/CD solution as this was too complex to implement. If the time was allowed, here is what I would do:

Using Github Actions with a connection to AWS and the terraform CLI installed, i would implement the following script to the GitHub repository where the code would be hosted (this is

available in the `add-yaml` branch)::

```yaml
name: Deploy FLask App

on:
  push:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: us-east-1

      - name: Set up Terraform
        uses: hashicorp/setup-terraform@v1
        with:
          terraform_version: 1.0.0

      - name: Terraform Init
        run: terraform init

      - name: Terraform Apply
        run: terraform apply -auto-approve
        env:
          TF_VAR_some_variable: ${{ secrets.TF_ENV }}
```

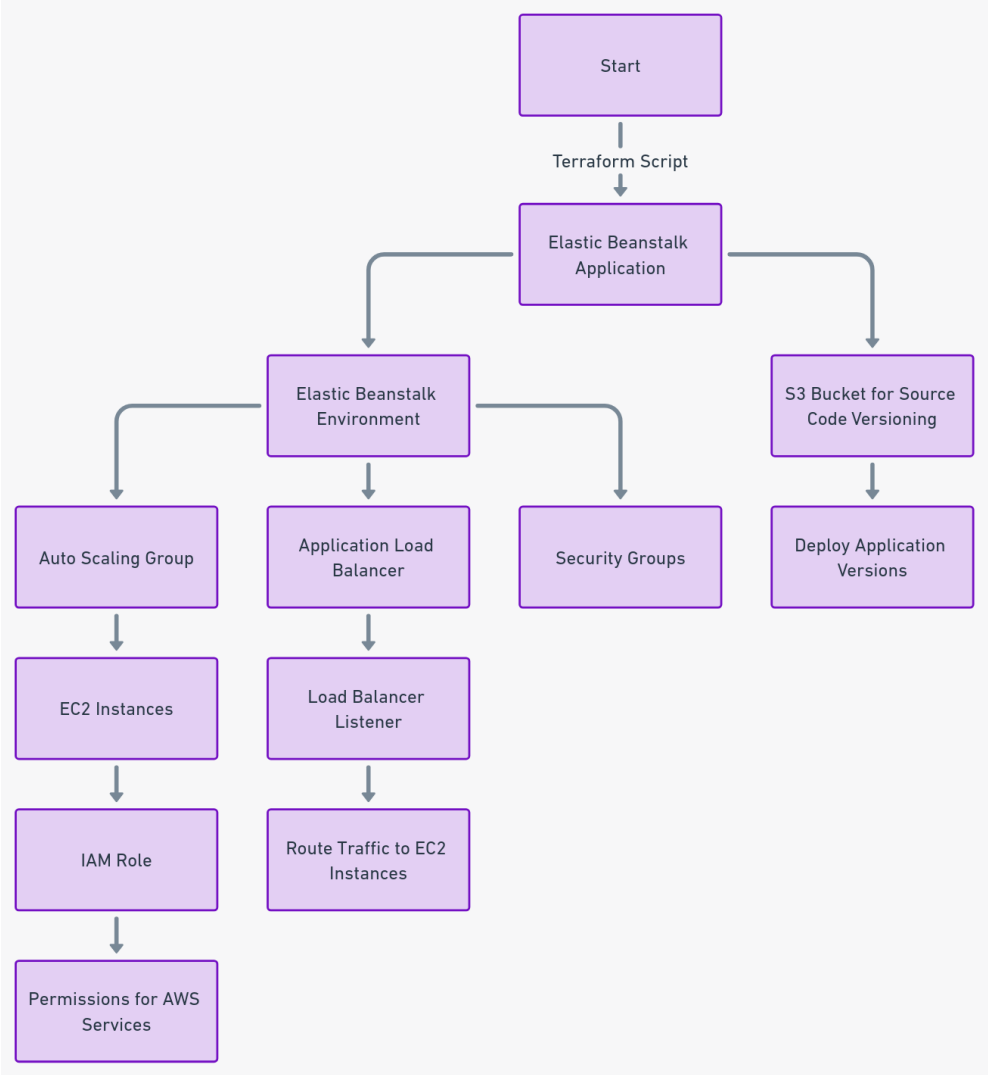
Points of Importance for the above script:

- **AWS Credentials**: This script uses the action aws-actions/configure-aws-credentials@v1 to configure AWS credentials. Ideally, the AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY would be stored under repository secrets.

- **Terraform State**: Terraform would be configured to use a remote backend like S3 with state locking via DynamoDB to manage the state file securely and allow for teams to use this easily.

- **Terraform Version**: The terraform version used in the script would need to be the same version that is being used on my local machine..

Terraform Overview



The terraform script I have made does the following:

1. Creates the elastic beanstalk application
2. Creates the elastic beanstalk environment
3. Creates an autoscaling group within the elastic beanstalk environment
4. Creates an application load balancer
5. Creates the necessary Security Groups to be used by the elastic beanstalk environment
6. An IAM role is created to be used by the EC2 instances to ensure they have the correct permissions
7. An S3 bucket is created for versioning of the application

The script is split into modules to keep everything concise and readable, the format follows the below diagram (obtained via tree):

```
.
├── main.tf
├── modules
│   ├── s3
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   ├── eb_application
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   └── security_group
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
└── variables.tf
```

Other Information:

In the Terraform configuration files provided above, you will need to replace placeholder values with actual values that correspond to your specific AWS environment and setup. Here are the values that you would need to replace:

VPC ID:

In main.tf at the root level and in modules/security_group/variables.tf, replace "vpc-12345678" with your actual AWS VPC ID where you want to deploy the resources.

AWS Region:

The AWS region is set to "us-east-1" in main.tf at the root level. If your infrastructure needs to be in a different region, update this to the desired AWS region.
Bucket Name:

In main.tf at the root level under the module "s3_bucket" block, "my-flask-app-source" is a placeholder for the S3 bucket name. S3 bucket names need to be globally unique, so you'll have to provide a unique name here.

Elastic Beanstalk Solution Stack Name:

Ensure that the solution stack name ("64bit Amazon Linux 2 v3.3.6 running Python 3.8") in modules/eb_application/main.tf is the correct and latest one supported by Elastic Beanstalk for

the Python environment.

Instance Type:

In modules/eb_application/main.tf, "t2.micro" is set as the instance type. Depending on your application's requirements and AWS's current offerings, you may want to select a different instance type that is more suitable for production environments.
Min and Max Size:

In modules/eb_application/main.tf, the min_size and max_size are set to 2 and 4, respectively. You should adjust these numbers based on the expected traffic and scaling needs of your application.