1. Create a simple HTML/CSS website
    a. Add a Procfile
        i. web: nginx -g "daemon off;"
    b. Add appspace.yml

```
version: 0.0
os: linux
files:
 - source: /
   destination: /var/www/html

hooks:
 BeforeInstall:
   - location: scripts/before_install.sh
     timeout: 300
     runas: ec2-user
 AfterInstall:
   - location: scripts/after_install.sh
     timeout: 300
     runas: ec2-user
 ApplicationStart:
   - location: scripts/start_application.sh
     timeout: 300
     runas: ec2-user
 ValidateService:
   - location: scripts/validate_service.sh
     timeout: 300
     runas: ec2-user
```

    c. scripts/after_install.sh

```
#!/bin/bash
# Copy new application files to the target directory
sudo cp -r * /var/www/html/

sudo systemctl restart apache2  # or `nginx` if using Nginx
```

   d. scripts/before_install.sh

```
#!/bin/bash
# Ensure the scripts are executable
chmod +x /opt/codedeploy-agent/deployment-root/*/scripts/before_install.sh
chmod +x /opt/codedeploy-agent/deployment-root/*/scripts/after_install.sh

# Create or clear the target directory
sudo mkdir -p /var/www/html
sudo rm -rf /var/www/html/*
```

2. Upload the website to GitHub (please ensure that it is visible to the public)
3. Deploy the application to EC2
   a. Launch an EC2 Amazon Linux 2023 Instance
   b. Add your key pair
   c. Add any key and value for tag
   d. Connect to it via SSH
   e. Install nginx
      i. sudo yum update -y
      ii. sudo yum install -y git
      iii. sudo amazon-linux-extras install nginx1.12 -y # Example: Nginx for Amazon Linux
      iv. git clone <your-repo-url>
   f. In the terminal, go to /etc/nginx/nginx.conf and edit the file so that root leads to your cloned repo

```
server_name  ec2-35-95-118-174.us-west-2.compute.amazonaws.com;
        root         /home/ec2-user/COMP-4964-Assignment-04;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
            try_files $uri $uri/ =404;  # This will try to serve the file or return a 404 error if not found
```

ix.
```
        }
```

g. Open permissions to file to prevent 403 Forbidden + turn off SELinux
i. sudo chown -R nginx:nginx <path-to-repo>
ii. sudo chmod -R 755 <path-to-repo>
iii. sudo chmod 644 <path-to-repo>/index.html
iv. sudo chmod +x /home/ec2-user
v. sudo setenforce 0

h. Test configurations
i. sudo nginx -t

i. Restart nginx
i. sudo systemctl restart nginx

4. Link EC2 to CodePipeline

a. Create IAM role
i. Attach the following policies:
1. AmazonEC2RoleforAWSCodeDeploy
2. AmazonS3ReadOnlyAccess
3. CloudWatchLogsFullAccess

b. Attach to EC2 Instance
i. EC2 Instance > Actions > Security > Modify IAM role

c. Install EC2 to CodeDeploy
i. In the terminal of the EC2 Instance
1. sudo yum update
2. sudo yum install ruby
3. sudo yum install wget
4. cd /home/ec2-user
5. wget https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/install
6. chmod +x ./install
7. systemctl status codedeploy-agent
8. sudo systemctl start codedeploy-agent
9. sudo systemctl status codedeploy-agent

ii. Create a new CodeDeploy application
1. Put in any name
2. Compute platform: EC2/On-premises
3. Create Deployment groups
a. Enter any name
b. Use CodeDeployRole for service role
c. Deployment type: In-place
d. Environment configuration: Amazon EC2 Instances

i. Choose your instance via tags

e. Disable Load balancer

f. Create deployment group

5. Follow this for create CodeBuild: https://aws.amazon.com/getting-started/hands-on/create-continuous-delivery-pipeline/module-three/
6. Follow this for creating CodePipeline: https://aws.amazon.com/getting-started/hands-on/create-continuous-delivery-pipeline/module-four/
    a. Change the Deploy to CodeDeploy instead of Elastic Beanstalk