Prerequisites:

1. **Terraform** installed on your local machine.
2. **AWS CLI** configured with your AWS credentials.
3. **Docker** installed on your local machine.
4. A basic understanding of **Terraform**, **AWS**, and **Docker**.

Project Outline:

1. Terraform will provision an AWS EC2 instance.
2. Docker will be installed on the EC2 instance using a shell script.
3. A simple Python Flask application will be containerized using Docker.
4. Terraform will push this Docker container to an EC2 instance and run the application.

Step-by-Step Guide:

1. Create the Terraform configuration files:

2. Create the Python Flask Application:

1. *Inside the `app` folder, create a file `app.py` with a simple Python Flask app:*

```
python
# app/app.py
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return "Hello, World from Flask inside Docker!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

2. *Create the **`Dockerfile`** to containerize the Python Flask app:*

```
Dockerfile

# app/Dockerfile
FROM python:3.8-slim

WORKDIR /app

COPY app.py /app

RUN pip install Flask
```

```
CMD ["python", "app.py"]
```

This `Dockerfile` specifies the base image, sets the working directory, copies the Python app, installs the required dependencies, and defines the command to run the Flask app.

---

3. Create the Shell Script to Install Docker on EC2 Instance:

Create a script `install_docker.sh` inside the `script/` folder that will install Docker on the EC2 instance:

```
bash
# script/install_docker.sh
#!/bin/bash
sudo apt-get update -y
sudo apt-get install -y apt-transport-https ca-certificates curl software-
properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update -y
sudo apt-get install -y docker-ce
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker ubuntu
```

This script will install Docker and add the user to the Docker group for permission.

---

4. Terraform Configuration:

1. **`main.tf`**: This file contains the main configuration for provisioning AWS resources.

```
hcl
# terraform/main.tf
provider "aws" {
  region = "us-east-1"  # Update with your preferred region
}

resource "aws_instance" "web_server" {
  ami           = "ami-0c55b159cbfafe1f0" # Ubuntu 20.04 AMI
  instance_type = "t2.micro"

  key_name      = "your-key-name" # Add your EC2 key name

  user_data = <<-EOF
              #!/bin/bash
              curl -O https://raw.githubusercontent.com/your-
username/dockerized-web-server/main/script/install_docker.sh
              chmod +x install_docker.sh
```

```
              ./install_docker.sh
              docker build -t flask-app /app
              docker run -d -p 5000:5000 flask-app
              EOF

  tags = {
    Name = "DockerizedWebServer"
  }
}

resource "aws_security_group" "web_sg" {
  name        = "web_sg"
  description = "Allow HTTP traffic"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 5000
    to_port     = 5000
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

output "web_server_public_ip" {
  value = aws_instance.web_server.public_ip
}
```

This `main.tf` script:

- Provisions an EC2 instance.
- Uses `user_data` to install Docker and start the Flask application in a Docker container.
- Creates a security group to allow inbound HTTP and SSH traffic.
- Outputs the public IP of the EC2 instance once it's provisioned.

2. **`variables.tf`**: This file contains variables that can be used in the main configuration file.

```
hcl
# terraform/variables.tf
variable "aws_region" {
  description = "The AWS region to deploy resources"
```

```
  default     = "us-east-1"
}

variable "instance_type" {
  description = "The EC2 instance type"
  default     = "t2.micro"
}
```

3. **`outputs.tf`**: This file defines the output variables.

```
hcl
# terraform/outputs.tf
output "instance_ip" {
  value = aws_instance.web_server.public_ip
  description = "The public IP address of the EC2 instance"
}
```

---

5. Terraform Workflow:

1. **Initialize Terraform:**

Run the following command to initialize the Terraform project:

```
bash
terraform init
```

2. **Apply Terraform Configuration:**

Run the following command to create the infrastructure:

```
bash

terraform apply
```

- Terraform will ask for confirmation to create the resources. Type `yes` and press Enter.

3. **Access the Web Server:**

Once the EC2 instance is provisioned, Terraform will output the public IP address of the EC2 instance. Open a browser and navigate to the public IP of the EC2 instance, and "Hello, World from Flask inside Docker!" .