Blue-Green Deployment of a Node.js Application

Yuvan Raj Krishna (Reg No. 22011102127)

November 8, 2025

Objective

Implement a zero-downtime blue-green deployment workflow for a Node.js service by containerizing the app, publishing images to Docker Hub, and automating deployment via a Jenkins pipeline that alternates between blue and green environments.

1 System Overview

- **Application**: Express server exposing ‘/‘ and ‘/health‘.
- **Containerization**: Dockerfile based on ‘node:20-alpine‘ plus ‘.dockerignore‘.
- **Orchestration**: ‘docker-compose.bluegreen.yml‘ launches two app containers and an NGINX proxy that reads the active color from ‘.env‘.
- **CI/CD**: Jenkins pipeline builds, tests, publishes, deploys to the idle color, smoke-tests, and optionally flips traffic.

2 Implementation Steps

2.1 Build the Node.js service

```
npm init -y
npm install express
cat > src/server.js <<'JS'
  // Express app uses APP_COLOR/APP_VERSION to render status
JS
npm start
```

2.2 Containerize the service

```
docker build -t bluegreen-demo:dev .
docker run -p 3000:3000 bluegreen-demo:dev
```

2.3 Compose blue/green stacks

```
cp .env.sample .env  # fill Docker Hub values
ACTIVE_COLOR=blue docker compose -f docker-compose.bluegreen.yml up -d
   --build
open http://localhost:8080
```

2.4 Automate release helpers

- `scripts/build-and-push.sh <tag>` builds/pushes `DOCKERHUB_USERNAME/APP_NAME:<tag>`.
- `scripts/deploy-color.sh <blue|green> <image> <version>` updates `.env` entries and restarts the chosen color.
- `scripts/switch-color.sh <blue|green>` rewrites `ACTIVE_COLOR` and recreates the proxy.

3 Jenkins Pipeline

The declarative pipeline (Listing 1) contains parameters for repository, tag, smoke-test URL, deploy color, and whether to switch traffic. Pipeline stages:

1. Checkout code and install dependencies with ‘npm ci‘.
2. Run placeholder unit tests (ready for future expansion).
3. Build Docker image tagged with either the provided `IMAGE_TAG` or `build-$BUILD_NUMBER`.
4. Push to Docker Hub using the `dockerhub-creds` credential.
5. Deploy the idle color via ‘scripts/deploy-color.sh‘.
6. Smoke-test the ‘/health‘ endpoint.
7. Optionally switch the proxy to route users to the freshly deployed color.

Listing 1: Jenkinsfile excerpt

```
stage('Deploy␣to␣idle␣color') {
  steps {
    sh "./scripts/deploy-color.sh␣${params.DEPLOY_COLOR}␣$IMAGE_NAME␣
       $RELEASE_TAG"
  }
}
```

4 Testing and Verification

4.1 Local preparation

Figure 1 captures the completed ‘.env‘ file populated with the Docker Hub coordinates, and Figure 2 shows ‘npm install‘ succeeding prior to containerization.

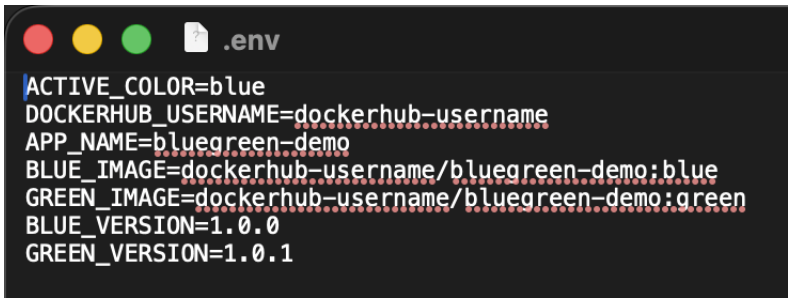

Figure 1: Environment file populated with blue/green image references.

```
(base) ➜  BLUEGREEN deployment npm install

up to date, audited 69 packages in 657ms

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) ➜  BLUEGREEN deployment
```

Figure 2: Dependencies installed locally before containerizing.

4.2 Application preview

Figures 3 and 4 document the HTML served by the Node.js process and the styled UI indicating the blue stack metadata.

```
(base) ➜  BLUEGREEN deployment curl http://localhost:3000

    <html>
      <head>
        <title>Blue-Green Demo</title>
        <style>
          body { font-family: Arial, sans-serif; background: #0f172a; color: #e2e8f0; margin: 0; }
          main { min-height: 100vh; display: flex; flex-direction: column; justify-content: center; align-items: center; }
          .card { background: rgba(15, 23, 42, 0.9); padding: 2rem 3rem; border-radius: 0.5rem; box-shadow: 0 20px 35px rg
ba(15, 23, 42, 0.5); }
          .pill { padding: 0.35rem 0.75rem; border-radius: 999px; background: #3b82f6; color: #0f172a; font-weight: 600; }
          table { width: 100%; margin-top: 1rem; border-collapse: collapse; }
          td { padding: 0.5rem 0; border-bottom: 1px solid rgba(148, 163, 184, 0.2); }
          td:first-child { color: #94a3b8; text-transform: uppercase; font-size: 0.8rem; }
        </style>
      </head>
      <body>
        <main>
          <div class="card">
            <div class="pill">BLUE (blue)</div>
            <h1>Blue-Green Deployment Demo</h1>
            <p>Seamless deployments without downtime.</p>
            <table>
              <tr><td>Version</td><td>1.0.0</td></tr>
              <tr><td>Host</td><td>Yuvans-MacBook-Pro.local</td></tr>
              <tr><td>Timestamp</td><td>11/8/2025, 12:11:21 PM</td></tr>
            </table>
          </div>
        </main>
      </body>
    </html>
(base) ➜  BLUEGREEN deployment
```

Figure 3: ‘curl http://localhost:3000‘ returning the rendered HTML/CSS.

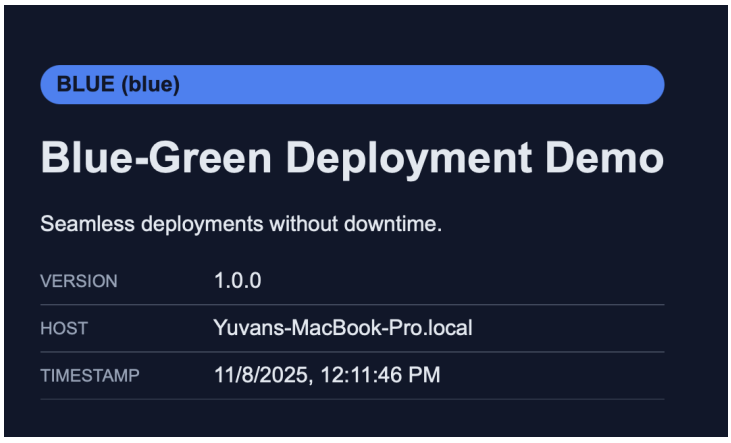

Figure 4: Browser view of the blue stack showing version and host info.

4.3 Container build and registry proof

Figure 5 shows the helper script building and pushing ‘yuvan4525/bluegreen-demo:blue‘, while Figure 6 confirms the tag in Docker Hub.

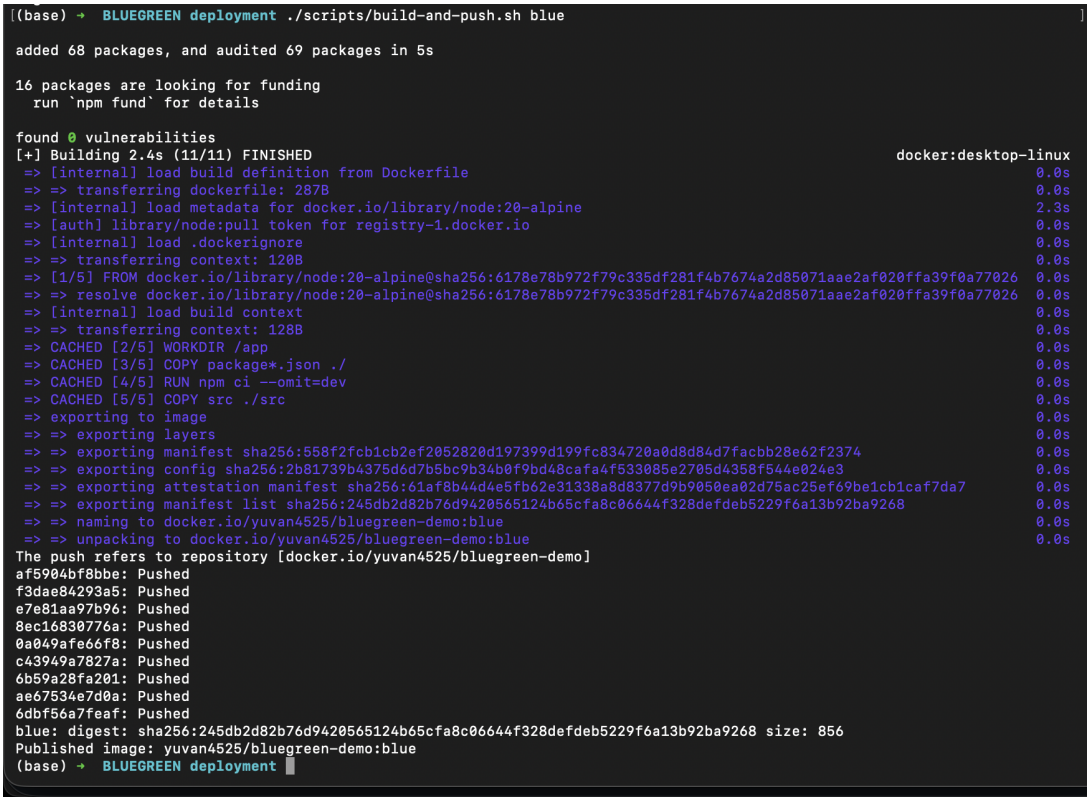

Figure 5: ‘./scripts/build-and-push.sh blue‘ output including digest.

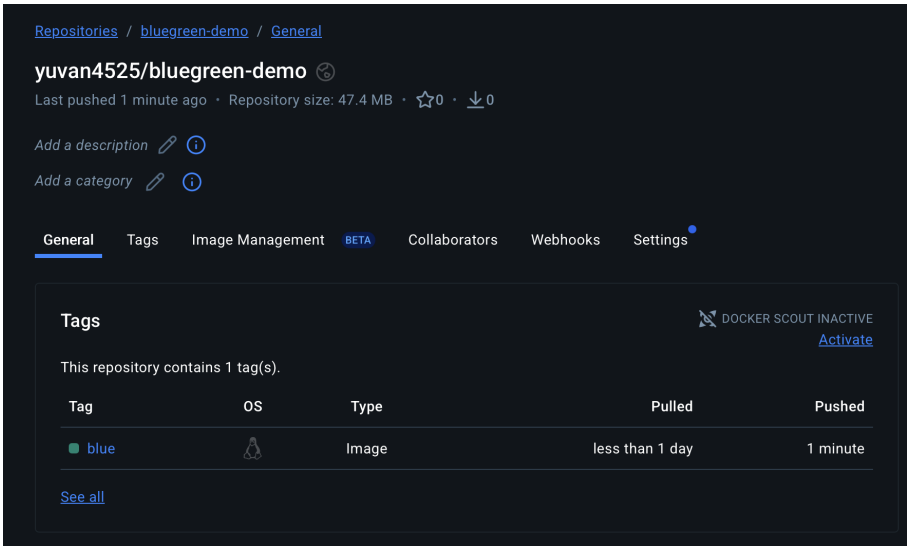

Figure 6: Docker Hub repository showing the freshly pushed tag.

4.4 Blue/green stack validation

Figure 7 captures ‘docker compose ps‘ with both app containers plus the proxy, while Figure 8 shows the proxy ‘/health‘ endpoint reporting the blue environment.

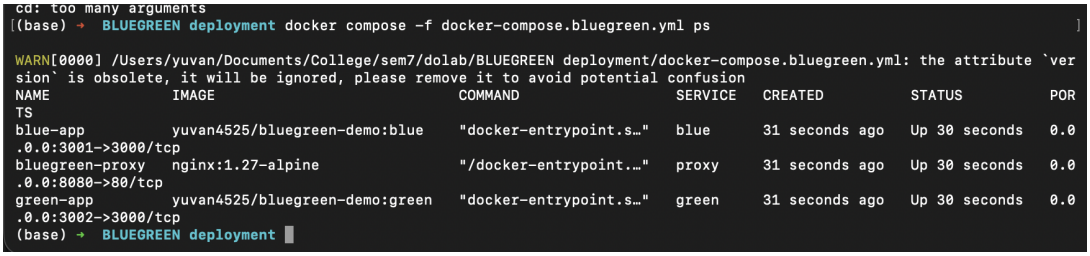

Figure 7: Compose stack running blue, green, and proxy services locally.

Figure 8: Proxy health endpoint exposing blue metadata before the switch.

4.5 Green deployment and traffic switch

Figures 9, 10, and 11 document the CLI deployment to the idle color, the '/health' check on port 3002, and the subsequent switch that routes users to the green UI.

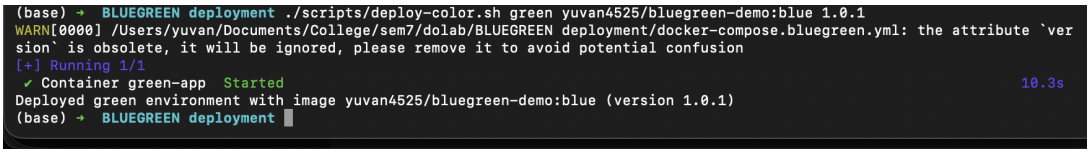

Figure 9: 'deploy-color.sh' rolling out the green environment.

Figure 10: Green stack '/health' response confirming version 1.0.1.

Figure 11: Proxy-backed UI after switching traffic to the green environment.

4.6 Jenkins pipeline evidence

Figures 12–14 summarize the Jenkins success: the stage view, permalinks widget, and console log snippet covering docker login/push.

```
12:57:51  + curl -fsSL http://localhost:8090/health
12:57:51  {"status":"ok","env":"green","color":"green","version":"1.0.1","host":"019cc6fa9906","timestamp":"2025-11-08T07:27:51.055Z"}
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Switch traffic (optional))
Stage "Switch traffic (optional)" skipped due to when conditional
[Pipeline] getContext
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // timestamps
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Figure 12: Blue Ocean stage visualization showing every stage green.

Figure 13: Job permalinks confirming build #11 as the latest stable run.

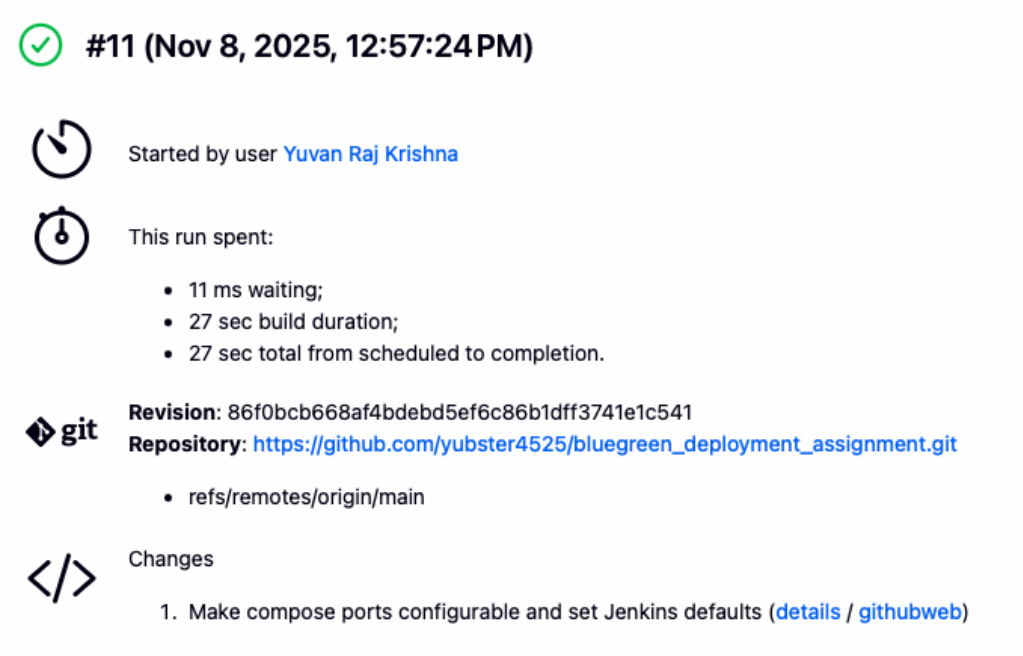

Figure 14: Console log excerpt covering the docker push executed by Jenkins.

5 Results

The lab delivers a reusable workflow:

- Deterministic Node.js container image published to Docker Hub.
- Docker Compose stack that keeps blue and green environments warm while an NGINX proxy steers production traffic.
- Jenkins pipeline that automates build, push, deploy, smoke test, and promotion with a single click.

Appendix: Key Configuration Files

Jenkinsfile

```
pipeline {
  agent any
  tools {
    nodejs 'Node18'
  }
  options {
    timestamps()
  }

  environment {
    REGISTRY_CREDENTIALS = 'c15b784c-6898-4020-8619-6d9aac49c3bc'
    PATH = "/opt/homebrew/bin:/usr/local/bin:${env.PATH}"
  }

  parameters {
    choice(name: 'DEPLOY_COLOR', choices: ['blue', 'green'],
       description: 'Idle environment that should receive the new
       version first')
    string(name: 'DOCKERHUB_REPO', defaultValue: 'yuvan4525/bluegreen-
       demo', description: 'Docker Hub repository (e.g. user/app)')
    string(name: 'IMAGE_TAG', defaultValue: '', description: 'Optional
       image tag override. Leave blank to use build number')
    string(name: 'SMOKE_TEST_URL', defaultValue: 'http://localhost
       :8180/health', description: 'URL hit after deployment to verify
       the new color')
    booleanParam(name: 'SWITCH_TRAFFIC', defaultValue: false,
       description: 'Switch NGINX proxy to the new color after smoke
       tests pass?')
  }

  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }

    stage('Install dependencies') {
      steps {
        sh 'npm ci'
      }
```

```
    }

    stage('Unit␣tests') {
      steps {
        sh 'npm␣test␣||␣echo␣"No␣automated␣tests␣yet"'
      }
    }

    stage('Build␣image') {
      steps {
        script {
          env.RELEASE_TAG = params.IMAGE_TAG?.trim() ? params.IMAGE_TAG
              : "build-${env.BUILD_NUMBER}"
          env.IMAGE_NAME = "${params.DOCKERHUB_REPO}:${env.RELEASE_TAG}
             "
        }
        sh 'docker␣build␣-t␣$IMAGE_NAME␣.'
      }
    }

    stage('Push␣image') {
      steps {
        withCredentials([usernamePassword(credentialsId: env.
           REGISTRY_CREDENTIALS, usernameVariable: 'DOCKER_USER',
           passwordVariable: 'DOCKER_PASS')]) {
          sh '''
␣␣␣␣␣␣␣␣␣␣␣␣␣␣echo␣$DOCKER_PASS␣|␣docker␣login␣-u␣$DOCKER_USER␣--password
   -stdin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣docker␣push␣$IMAGE_NAME
␣␣␣␣␣␣␣␣␣␣␣␣␣␣docker␣logout
␣␣␣␣␣␣␣␣␣␣␣␣'''
        }
      }
    }

    stage('Prepare␣env␣file') {
      steps {
        script {
          def repoParts = params.DOCKERHUB_REPO.tokenize('/')
          def dockerUser = repoParts ? repoParts[0] : 'dockerhub-user'
          def appName = repoParts.size() > 1 ? repoParts[1] : '
             bluegreen-demo'
          def envText = """
␣␣␣␣␣␣␣␣␣␣␣␣ACTIVE_COLOR=blue
␣␣␣␣␣␣␣␣␣␣␣␣DOCKERHUB_USERNAME=${dockerUser}
␣␣␣␣␣␣␣␣␣␣␣␣APP_NAME=${appName}
␣␣␣␣␣␣␣␣␣␣␣␣BLUE_IMAGE=${params.DOCKERHUB_REPO}:blue
␣␣␣␣␣␣␣␣␣␣␣␣GREEN_IMAGE=${params.DOCKERHUB_REPO}:green
␣␣␣␣␣␣␣␣␣␣␣␣BLUE_VERSION=1.0.0
␣␣␣␣␣␣␣␣␣␣␣␣GREEN_VERSION=1.0.1
␣␣␣␣␣␣␣␣␣␣␣␣STACK_PREFIX=jenkins-
␣␣␣␣␣␣␣␣␣␣␣␣BLUE_PORT=3101
␣␣␣␣␣␣␣␣␣␣␣␣GREEN_PORT=3102
␣␣␣␣␣␣␣␣␣␣␣␣PROXY_PORT=8180
␣␣␣␣␣␣␣␣␣␣""".stripIndent().trim() + "\n"
          writeFile file: '.env', text: envText
        }
      }
```

```
    }

    stage('Deploy to idle color') {
      steps {
        sh "./scripts/deploy-color.sh ${params.DEPLOY_COLOR}
           $IMAGE_NAME $RELEASE_TAG"
      }
    }

    stage('Smoke test') {
      steps {
        sh "curl -fsSL ${params.SMOKE_TEST_URL}"
      }
    }

    stage('Switch traffic (optional)') {
      when {
        expression { params.SWITCH_TRAFFIC }
      }
      steps {
        sh "./scripts/switch-color.sh ${params.DEPLOY_COLOR}"
      }
    }
  }
}
```

Dockerfile

```
FROM node:20-alpine AS base
WORKDIR /app
ENV NODE_ENV=production

COPY package*.json ./
RUN npm ci --omit=dev

COPY src ./src

EXPOSE 3000
ENV PORT=3000 \
    APP_COLOR=blue \
    APP_ENV=blue \
    APP_VERSION=1.0.0

CMD ["node", "src/server.js"]
```

Docker Compose Definition

```
services:
  blue:
    build: .
    image: ${BLUE_IMAGE}
    container_name: "${STACK_PREFIX:-}blue-app"
    environment:
      - APP_COLOR=blue
      - APP_ENV=blue
```

```
      - APP_VERSION=${BLUE_VERSION}
    ports:
      - "${BLUE_PORT:-3001}:3000"
  green:
    build: .
    image: ${GREEN_IMAGE}
    container_name: "${STACK_PREFIX:-}green-app"
    environment:
      - APP_COLOR=green
      - APP_ENV=green
      - APP_VERSION=${GREEN_VERSION}
    ports:
      - "${GREEN_PORT:-3002}:3000"
  proxy:
    image: nginx:1.27-alpine
    container_name: "${STACK_PREFIX:-}bluegreen-proxy"
    depends_on:
      - blue
      - green
    environment:
      - ACTIVE_COLOR=${ACTIVE_COLOR}
    ports:
      - "${PROXY_PORT:-8080}:80"
    volumes:
      - ./nginx/default.conf.template:/etc/nginx/templates/default.conf
          .template:ro
    command: /bin/sh -c "envsubst␣'$$ACTIVE_COLOR'␣<␣/etc/nginx/
        templates/default.conf.template␣>␣/etc/nginx/conf.d/default.conf
        ␣&&␣nginx␣-g␣'daemon␣off;'"
```

Deployment Script

```
#!/usr/bin/env bash
set -euo pipefail

COLOR=${1:-}
IMAGE_OVERRIDE=${2:-}
VERSION_OVERRIDE=${3:-}

if [[ -z "$COLOR" ]]; then
  echo "Usage:␣$0␣<blue|green>␣[image]␣[version]"
  exit 1
fi
if [[ "$COLOR" != "blue" && "$COLOR" != "green" ]]; then
  echo "Color␣must␣be␣'blue'␣or␣'green'"
  exit 1
fi

ENV_FILE=.env
if [[ ! -f $ENV_FILE ]]; then
  echo "Missing␣.env␣file"
  exit 1
fi

source $ENV_FILE

IMAGE_VAR_NAME=$(echo "${COLOR}_IMAGE" | tr '[:lower:]' '[:upper:]')
```

```
VERSION_VAR_NAME=$(echo "${COLOR}_VERSION" | tr '[:lower:]' '[:upper:]'
    )
SERVICE_NAME=$COLOR

CURRENT_IMAGE=${IMAGE_OVERRIDE:-${!IMAGE_VAR_NAME}}
CURRENT_VERSION=${VERSION_OVERRIDE:-${!VERSION_VAR_NAME}}

if [[ -z "$CURRENT_IMAGE" ]]; then
  echo "Define␣$IMAGE_VAR_NAME␣in␣.env␣or␣pass␣an␣image␣reference"
  exit 1
fi

python3 - "$ENV_FILE" "$IMAGE_VAR_NAME" "$CURRENT_IMAGE" "
    $VERSION_VAR_NAME" "$CURRENT_VERSION" <<'PY'
import sys
from pathlib import Path
path = Path(sys.argv[1])
image_var, image_value, version_var, version_value = sys.argv[2:]
lines = path.read_text().splitlines()
out = []
seen = {image_var: False, version_var: False}
for line in lines:
    if '=' not in line:
        out.append(line)
        continue
    key, value = line.split('=', 1)
    if key == image_var:
        out.append(f"{key}={image_value}")
        seen[image_var] = True
    elif key == version_var:
        out.append(f"{key}={version_value}")
        seen[version_var] = True
    else:
        out.append(line)
if not seen[image_var]:
    out.append(f"{image_var}={image_value}")
if not seen[version_var]:
    out.append(f"{version_var}={version_value}")
path.write_text('\n'.join(out) + '\n')
PY

env "${IMAGE_VAR_NAME}=$CURRENT_IMAGE" "${VERSION_VAR_NAME}=
    $CURRENT_VERSION" docker compose -f docker-compose.bluegreen.yml up
    -d $SERVICE_NAME

echo "Deployed␣$SERVICE_NAME␣environment␣with␣image␣$CURRENT_IMAGE␣(
    version␣$CURRENT_VERSION)"
```

Traffic Switch Script

```
#!/usr/bin/env bash
set -euo pipefail

COLOR=${1:-}
if [[ -z "$COLOR" ]]; then
  echo "Usage:␣$0␣<blue|green>"
  exit 1
```

```
fi

if [[ "$COLOR" != "blue" && "$COLOR" != "green" ]]; then
  echo "Color must be 'blue' or 'green'"
  exit 1
fi

ENV_FILE=.env
if [[ ! -f $ENV_FILE ]]; then
  echo ".env not found. Copy .env.sample and fill in required values
     first."
  exit 1
fi

python3 - "$ENV_FILE" "$COLOR" <<'PY'
import sys
from pathlib import Path
path = Path(sys.argv[1])
color = sys.argv[2]
lines = path.read_text().splitlines()
out = []
updated = False
for line in lines:
    if line.startswith('ACTIVE_COLOR='):
        out.append(f'ACTIVE_COLOR={color}')
        updated = True
    else:
        out.append(line)
if not updated:
    out.append(f'ACTIVE_COLOR={color}')
path.write_text('\n'.join(out) + '\n')
PY

docker compose -f docker-compose.bluegreen.yml up -d proxy

echo "Proxy now routing traffic to $COLOR environment"
```