CMSC388T

Deployment

Today's Lecture

1

Deployment
Introduction and Basics

2

Deployment with Github
Deployment with Terpconnect & Github

1. Deployment

Introduction & Basics

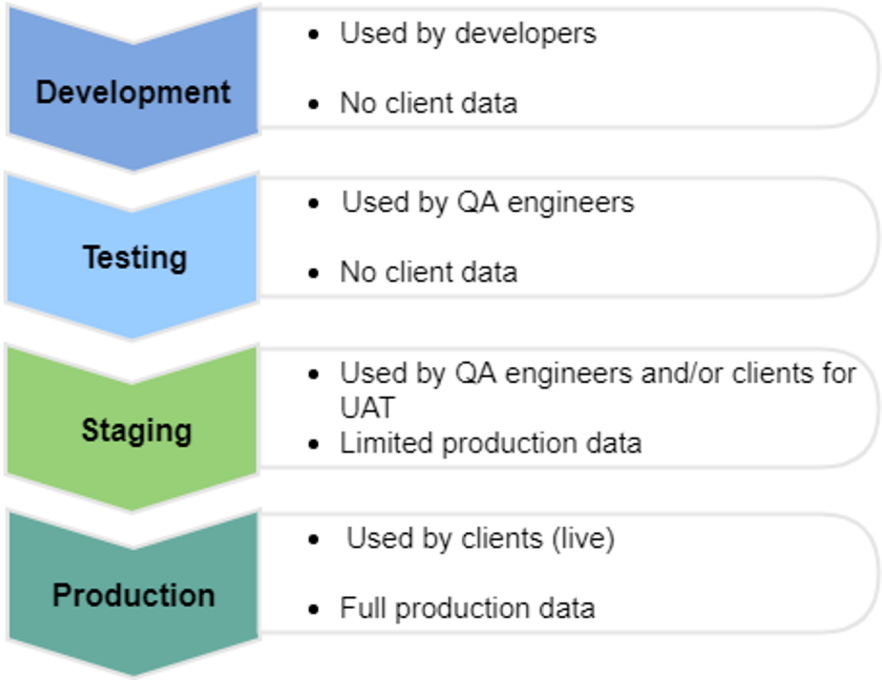
Deployment

- **Deployment -** push changes or updated from one environment to another
    - Example: Website might have a “live” website - the one customers use and a “production” website - the one programmers use to make edits and changes before the customer sees
- We have different staging environments because it allows code to be thoroughly tested as it goes through each stage
    - For example, the testing stage allows the creator of the script to receive feedback on their code

Typical process for Deployment

Different Staging Environments

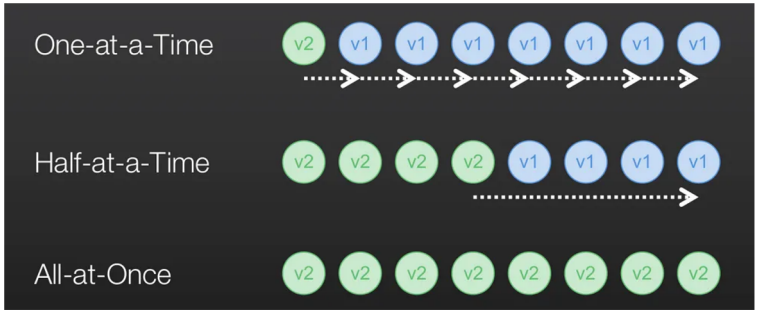
1. **Development:** changes to software are developed
1. **Testing:** testing the developed code
1. **Staging:** a pre-production environment for testing that mirrors the actual production environment: Insures reliability in the code.
1. **Production:** the environment has direct interactions with users, and is considered a live project



All-at-Once Deployment

- A pattern of deployment in which an update is sent out all at once

- **Pros:**
  - Cost effective
  - Very straightforward
- **Cons:**
  - Downtime for when you add changes
  - Downtime for when you roll back



Rolling Deployment

- A type of release model that releases updates to servers in a manner that prioritizes their functionality. EX: If I have 10 servers, then I update one server at a time.
- **Pros:**
  - Allows you to keep most of your servers functional even while others update
- **Cons:**
  - Overall, a slow transition to an update

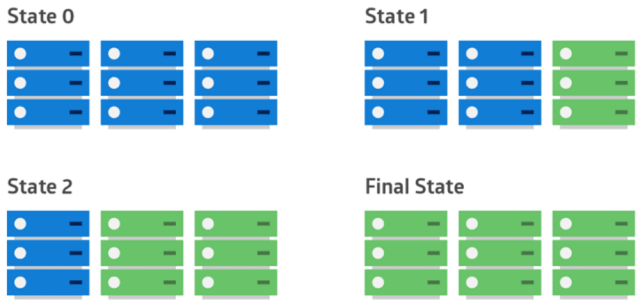


Image Source: dev.to

Rolling With Additional Batch Deployment

- Unlike Rolling, this release model always has an extra server on standby so that no servers are ever down.
    - EX: You have 11 servers always running (10 always running and 1 for updating)
    - Now, instead of having one server down you always have full capacity servers up
- **Pros:**
    - Allows you to keep all of your servers functional during updates
- **Cons:**
    - not as cost effective because you need an additional server
    - Still slow to rollback and update

Image Source: dev.to

Canary Deployment

- Canary deployment is a pattern of deployment for rolling out releases in the Production Staging environment to a subset of users or servers
- **Pros:**
  - Application features can be updated and tested independently
  - Safety net for errors that get to Production
- **Cons:**
  - Cannot be used for all applications *(Consider an update for an application that controls a nuclear reactor. The app cannot be updated with a canary release)*

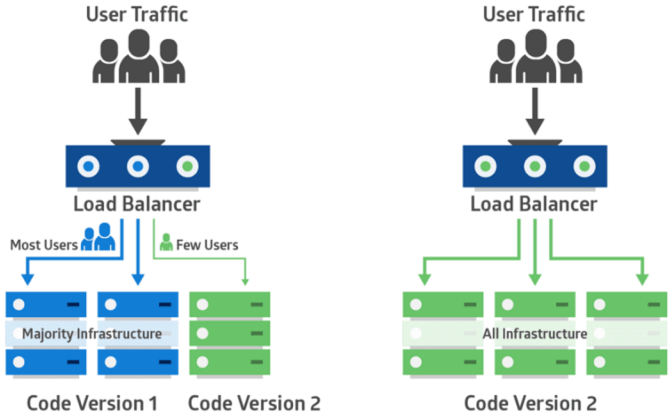


Image Source: dev.to

Blue-Green Deployment

- Blue-Green Deployment is a type of release model to help transfer new versions of an app to production
  - Old version is called blue environment
  - Newer versions are called green environment
- **Pros:**
  - Can Test in Live Environment
  - Minimize Deployment Downtime
- **Cons:**
  - Complex delivery at large scale
  - Not compatible with database based applications
  - Costly

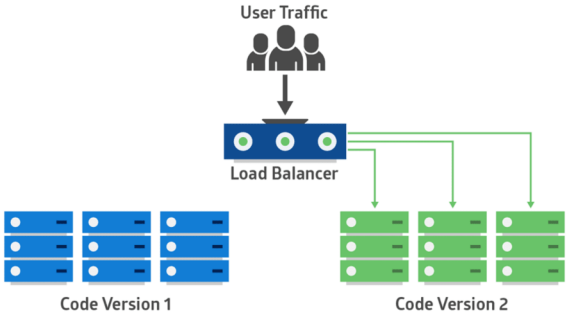


Image Source: dev.to

Deployment to Live Environment

- Live Environment - version that clients and customers use and have access too
- Final version that has been tested and checked
- Comes after Production phase
- Usual changes are made in a local environment first
- Git Version Control is useful to monitor changes and bug fixes

Clicker Quiz

What is blue-green deployment?

a) a type of release model to simulate tests for a new app
b) a type of release model that encourages the CI of a workflow with the production line
c) a type of release model to help with file upload during production
d) a type of release model to help transfer new versions of an app to production

Clicker Quiz

What is blue-green deployment?

a) a type of release model to simulate tests for a new app
b) a type of release model that encourages the CI of a workflow with the production line
c) a type of release model to help with file upload during production
d) a type of release model to help transfer new versions of an app to production

2. Github + Deployment

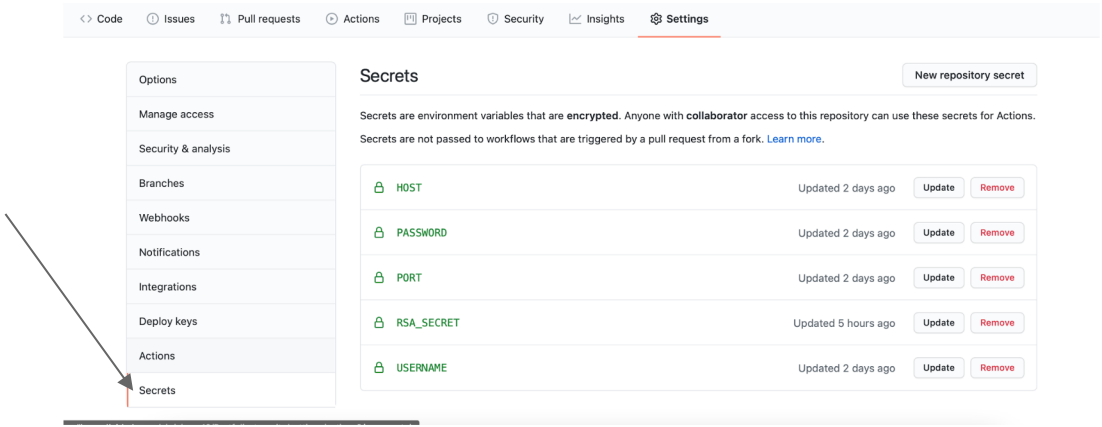
Using Github for Deployment

Setup ‘Github Secrets’

Github Secrets allow you to use private and valuable information such as usernames and passwords without their contents being displayed publicly for others to see

Notice how the title displays rather than the contents

Use this to add your
username - directory ID
password
host - terpconnect.umd.edu
port - 22



Create .yml file to Clone

- In a separate workflow, create a .yml
- Replace:
  - “on: push” with “on: workflow_dispatch”
  - change “git pull to git clone (repo)”

```yaml
name: clone workflow
on:
  workflow_dispatch:
jobs:
  job_one:
    name: Clone
    runs-on: ubuntu-latest
    steps:
      - name: testing
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.HOST }}
          username: ${{ secrets.USERNAME }}
          password: ${{ secrets.PASSWORD }}
          port: ${{ secrets.PORT }}
          script: |
              git clone
```

clone.yml

Create .yml file to Deploy

- Set up .yml file such that every time changes are pushed or merged to the master branch, it will update and deploy to the terp connect server
- The script signifies what should be done in the server

```yaml
name: deployment workflow
on:
  push:
    branches: [master]
jobs:
  job_one:
    name: Deploy
    runs-on: ubuntu-latest
    steps:
      - name: testing
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.HOST }}
          username: ${{ secrets.USERNAME }}
          password: ${{ secrets.PASSWORD }}
          port: ${{ secrets.PORT }}
          script: |
              cd TestRepo
              git pull
```

deploy.yml

Create .yml file to Cleaning Up

- Similar to before up .yml file such that every time changes are pushed or merged to the master branch, it will update and deploy to the terp connect server
- The script signifies what should be done in the server

```yaml
name: Clean workflow
on:
  workflow_dispatch:
jobs:
  job_one:
    name: Clean
    runs-on: ubuntu-latest
    steps:
      - name: testing
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.HOST }}
          username: ${{ secrets.USERNAME }}
          password: ${{ secrets.PASSWORD }}
          port: ${{ secrets.PORT }}
          script: |
             unalias rm
             rm -rf <Repo-Name>
```

clean.yml