Introduction

Welcome to Module 3! This week, we will learn about Continuous Integration and Continuous Development (CI/CD). CI/CD is enabled by tools and processes. Git is one of the tools and agile development is one of the processes that enables CI/CD. Virtualization environments enable a developer to create development environments to support different library or API versions.

Topics and Learning Objectives

Topics

- The “DevOps movement” and its importance for modern web development
    - Continuous integration
    - Continuous delivery

- Managing multiple environments for your application
- Dev/Prod (Development/Productivity) parity
- Deploying a Python/Flask web application to Heroku

Learning Objectives

By the end of the module, you should be able to:

1. Explain the concept of application development in a team.
2. Describe how to develop software, test software, and keep track of development history.
3. Describe how to use development methodology that ensures software quality.
4. Create, commit, and deploy a web application to the cloud.

Readings & Resources

Reading

Required

- Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media, Inc.
    - Chapter 17: Deployment

- Wiggins, A. (2017). Dev/prod parity. The Twelve-Factor App.
- Humble, J. (2017). Patterns. Continuous Delivery.

CI/CD Introduction

- Continuous integration (CI) and continuous development (CD) are aspects of Agile software methodology.
- Agile development is a flexible alternative to traditional software management practices such as waterfall and SDLC (software development life cycle).
- CI/CD bridges gaps between development and operation activities by enforcing automated building and testing of applications.
- CI/CD is an ongoing process where code is constantly built and tested for feature quality.
- The aim of CI/CD is to discover defects early and provide a faster release cycle.

Rationale

- When embarking on a change, a developer takes a copy of the current code base on which to work. As other developers submit changed code to the source code repository, this copy gradually ceases to reflect the repository code. Not only can the existing code base change, but new code can be added as well as new libraries, and other resources that create dependencies and potential conflicts.
- The longer development continues on a branch without merging back to the mainline, the greater the risk of multiple integration conflicts and failures when the developer branch is eventually merged back. When developers submit code to the repository they must first update their code to reflect the changes in the repository since they took their copy. The more changes the repository contains, the more work developers must do before submitting their own changes.
- Eventually, the repository may become so different from the developers' baselines that they enter what is sometimes referred to as "merge hell", or "integration hell", where the time it takes to integrate exceeds the time it took to make their original changes.

Continuous Integration

- CI is an agile software process where developers are required to integrate code into a main repository during various times of the day. This enables simultaneous updates and reduces costs of integration.
- Unlike traditional cycles where developers have to build software in total isolation and integrate it

after developing all modules, CI offers frequent integration during development.

- CI requires a high level of automation that includes testing and automatic builds.
- Primary objectives of CI are:
    a. Reduce cost of integration
    b. Discover conflicts and bugs early
    c. Fast-tracking development process

Continuous Development

- CD is a process where CI is followed by automated testing and automated daily development for production.
- Any code changes are automatically tested for quality and deployed.
- The above two ensures that applications are ready for usage with the assumption that tests pass.
- Everything from integration, quality testing, delivery, and deployment are automated for each iteration.
- Manual intervention is required only at the time of final push to market.
- Primary objectives for CD are
    a. Product is updated in real-time
    b. Application is ready for market
    c. Highest quality is maintained via robustly automated test suites

CI/CD Key Concepts

1. Iterations
2. Trunk-based development
3. Fast track build and test
4. Consistent development

Iterations

- Iterations encourage smaller updates
- Developers break complex work into modules and commit them as early as possible.
- Small changes are faster to develop easy to integrate.

- Minor updates reduce the probability of errors resulting in a reduced cost of integration.
- When integration problems surface, they can be fixed on a daily basis.
- Development techniques used are branch by abstraction and form-in progress code upgrades

Trunk-Based Development

- Trunk-based development makes sure development is always in the main branch. Alternatively, it can be integrated back to main branch in frequent intervals.
- Smaller branches may be used, provided they are modularized and integrated into the main branch.
- All branches are short lived. Main branch lives forever.
- Main objective is to avoid large changes. Large changes might result in errors.
- For release purposes, a dedicated branch is allowed where no development takes place.

Fast Track Build and Test

- Changes are handled by automated build and test suites.
- Every change has to go through build and testing. Since this process is repeated many times it is important to automate and streamline.
- Focus is on unit tests and integration testing for quality as they are faster to implement rather than acceptance testing.

Consistent Deployment

- As each change is implemented the same build and testing process is used. It is necessary to maintain consistency in your development in order to ensure high quality.
- Features are designed, coded and tested in that order. They may be additional debug and test cycles. Once a feature is accepted, its code may not be changed at any intervals after integration.
- An automated build and testing should make use of the same tools and procedures for deploying of all changes.

Python Testing Tools

- Selenium – used for browser UI testing
  - pip install selenium
- PyTest – a simple and efficient unit testing utility
  - pip install pytest

- PyUnit – easy way to create unit testing programs out of the box.
    - PyUnit

Git and CI/CD

- Note the three branches: master, development and test (see Figure 3.1 below).
- Note that bug fixes happen in the test branch. Once tests are passed, the content from the test branch is merged into master branch and a version tag is added.
- After a version tag is added to master branch a merge happens with the development branch. This is needed to add bug fixes that were checked into test branch but were never checked into development branch.

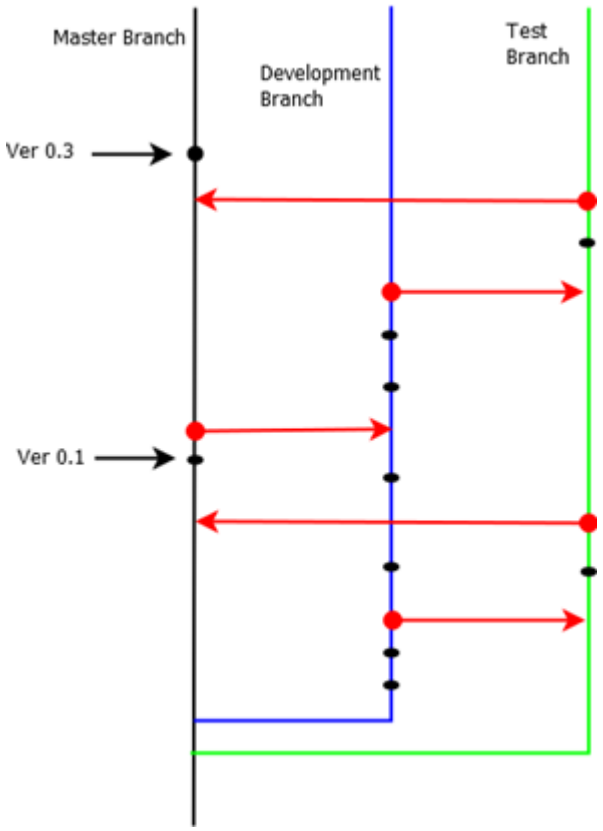

Figure 3.1. A Git Repository exhibiting three branches: master, development and test branch. The development team will check the source code into the development branch. At the same time, the quality assurance team will operate on the test branch. Clean code resides in the master branch.

Source: Mihal Miu

Long Description +

A Git Repository exhibiting three branches: master, development and test branch. The development team will check the source code into the development branch. At the same, the time quality assurance team will operate on the test branch. Clean code resides in the master branch. When the development team has the source code ready for testing, the source code will be moved to the test branch. This enables the developers to keep working while the quality assurance team performs tests. Once the quality assurance team has determined that their branch contains code that meets the current milestone, then the code is moved into the master branch.

Video

Please watch the following video, Git History Browsing. This video demonstrates how a Git repository houses source code. Source code history may be viewed with git command line tools or using Git GUI tools such as Gittyup.

Git History Browsing
TMU Video

VirtualEnv

- VirtualEnv, a virtual environment, is an isolated working copy of Python which allows you to work

on your project without affecting other projects

- VirtualEnv enables multiple side-by-sde installations of Python and modules for each project.
- VirtualEnv keeps different project environments isolated.
- VirtualEnv is available through CLI using the virtualenv and source command
    - `virtualenv –version`
    - `source activate`
    - `source deactivate`

Working with VirtualEnv

- To activate VirtualEnv simply navigate into your project directory and activate it using the following command:
    - `source activate`
- Once VirtualEnv has been activated the command prompt changes. Any modules and libraries installed for this project will not affect any other project
- To exit out of VirtualEnv you simply deactivate VirtualEnv using the following command:
    - `source deactivate`
- Once VirtualEnv has been deactivated the command prompt returns to the bash default.

Dev/Prod Parity

- In a development environment there are multiple development machines and testing servers.
- An application may exhibit bugs on one development machine but not on another. The questions is why is this occurring. The answer may be that the two development machines may have the same libraries installed but they differ in the installed version.
- To avoid the issue caused above, the same development environment will have to be used by everyone. There are two possible solutions for this: virtualization and containers.
- Keep development, staging and production as simple as possible.
- Note development is performed on individual machines while production is where the application runs on a server and is accessed by many users. Servers are more powerful than development machines and may contain different software stacks.
- There are gaps between development and production. These gaps manifest in three areas
    a. Time gap – a developer may work on code that takes hours, days, or weeks before it goes into production
    b. Personnel gap – developers write code and ops engineers deploy it

c. Tools gap – development stacks may be different than production stacks

Virtualization vs Containers

- Virtualization is traditionally supported by Virtual Machines. In this class we all have the same development environment distributed by the VirtualBox virtual machine that was installed in Lab 1.
- The disadvantage to virtual machines is that they provide an excellent sandbox at cost (memory and cpu usage).
- An alternative to virtual machines are containers. Containers are a lighter solution than virtual machines. A container is a sandbox for an application (your Python/Flask application).
- Containers have the assumption that your applications are virtualized by the operating system running on your computer. What if we all run different operating systems (Linux, MacOS, MS Windows)? We cannot use containers, but we can run Virtual Machines.

VirtualEnv, a VM or Container

- Is VirtualEnv a VM or a container?
- VirtualEnv is not a VM nor a container. It is an environment to facilitate multiple Python environments.
- VirtualEnv can allow for the installation of different version of the same library for different projects.
- VirtualEnv may be used inside a VM or a container. We can use VirtualEnv on our VirtualBox Virtual Machine.

Heroku

- Heroku is a cloud platform as a service (Paas) supporting several programming language that is hosted on Amazon's EC2 cloud-computing platform.
- Heroku may be used to test and deploy production applications supported by Java, Node.js, Python, PHP and Go. Hence it is called a polyglot platform.
- Applications deployed on Heroku are web applications. Hence they are accessed using a web browser.
- Heroku provides a unique domain that may be used to route HTTP requests to you application. In Heroku, your application runs in a container or dyno. Each dyno is spread across a "dyno grid", which consists of several servers.

Heroku Architecture

- Heroku architecture is a layered stack that is accessed by HTTP request from a web browser (see

Figure 3.2 below).

- Scalability is provided by Dynos.
- Dynos are supported by cloud databases (DBaaS based on PostgreSQL), enterprise management tools, add-ons and monitoring utilities.

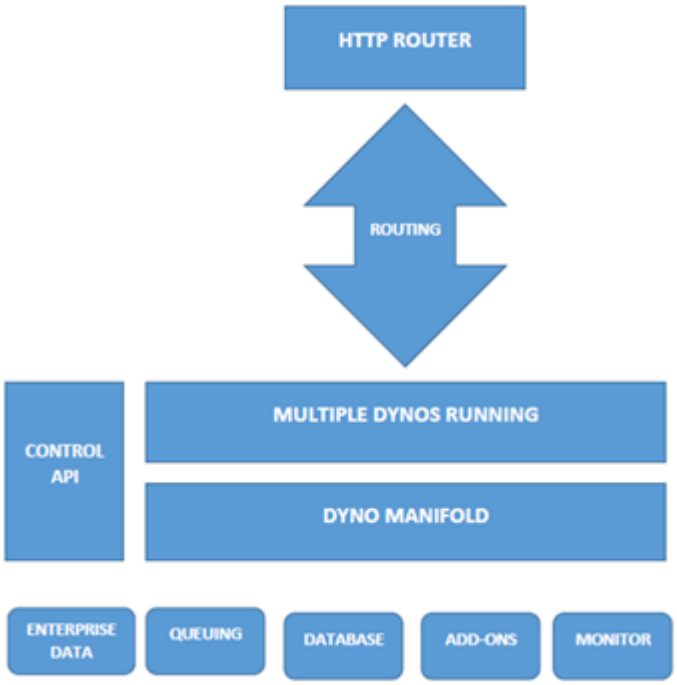

Figure 3.2. An architectural overview of the Heroku Platform.

Source: Wikimedia Commons

Long Description +

An architectural overview of the Heroku Platform. Heroku routes HTTP requests to the appropriate application container or dyno. Each dyno is spread across a dyno grid composed of several servers. The dyno grid is powered by various underlying services such as enterprise data, queuing, database, add-ons and monitoring services.

Heroku Alternatives for Full Stack Hosting

As of November 2022, Heroku no longer provides a free tier for full stack hosting. Fortunately there are alternatives to Heroku that provide a free tier.

1. Render
2. Railway

3. Deta Cloud
4. Fly.io

Summary

CI/CD is enabled by flexible infrastructure that allows developers to work collaboratively across multiple teams. Git is one tool that allows for source code to be maintained by multiple teams (developers and quality assurance). These teams will work independently and at other times they may work collaboratively. A platform such as Heroku allows applications to be deployed and be made available to users over the internet. Heroku is a hosted service, hence the development team does not need to host any physical servers.

Assessments

Each week, you'll find here reminders of assignments or labs that you should be working on. For Week 3,

- Lab 3: CD/CI (Continuous Development / Continuous Integration) (5% of the final grade) is due in 7 days, i.e., the end of day, Friday, of Week 3.

Reminder

You have a weekly Zoom session. The date/timing for your Zoom session can be found in the Course Schedule in the Course Outline. The link to the Zoom session will be provided separately to you by your instructor. The Zoom session will consist of 20–30 minutes of lecture, followed by questions and answers (Q&A), followed by a lab period. You may ask questions about the previous or current week's content or labs.

Please bring any questions to the weekly Zoom session. However, if you have any questions during the week outside of the Zoom session, you may post them in the Discussion Board. Your instructor may respond in the Discussion board or during the weekly Zoom session.

Click-n-reveal: **Where is the Discussion board?**

Click on the "Communication" area at the top main course menu, and select Discussions from the dropdown menu.

References

Wikipedia. (n.d.). Heroku.