COMP 4350 - Software Engineering II

Lecture 5 - Dr. Shaowei Wang

Administrative Items

- Meet TA next week for sprint 1 evaluation.
- Submit project proposal by next Friday to UMLearn (Assignment -> project proposal).
- Presentation to introduce your project next week.
    - Find schedule in the project google spreadsheet.
    - Submit slides before your presentation to UMLearn (Assignments -> Project proposal presentation).

Agenda

- Regression Testing
- Continuous Deployment
- Tutorial on Docker

Regression Testing

- **Definition**: Ensures an application functions as expected after changes. Aimed to maintain existing functionality and uncover defects early after alterations.
- **Goals**:
    - Prevent code changes from breaking existing system requirements.
    - Detect defects at an early stage.
- **Levels**:
    - Unit testing
    - Integration testing
    - System testing
    - Acceptance testing
- **Approaches**:
    - Complete Regression: Done when core of the code, multiple changes, or substantial codebase updates are made.
    - Partial Regression: Selected based on recent changes, areas with frequent bugs, or critical functionalities.
- **Regression Testing Tips**:
    - Tightly link code changes with test cases.
    - Utilize defect frequency heatmap and prediction models (e.g., Google's method scoring files based on bug-fix commits' age and frequency).
    - Ensure tests cover various levels and critical functionalities.

Continuous Deployment

- **Definition**: Automated process where changes are deployed to production without manual intervention.
- **Strategies**:
  - Blue-Green Deployments: Simultaneous running of two identical production setups to reduce risk and downtime.
  - Canary Releasing: Gradually rolling out changes to a subset of users before full deployment to manage risk.
  - A/B Testing: Analyzes two versions to determine better performance.
- **Tools**:
  - nginx for user requests distribution.
  - Docker Hub for sharing images.

Docker Tutorial

- **Containerization**: Technique for deploying software in a way that's isolated, consistent, and replicable across different environments.
- **Containers vs. Virtual Machines**:
  - Containers share the host OS kernel, lighter, and suited for isolated apps.
  - VMs include a full OS and are more resource-intensive, suited for running multiple apps.
- **Benefits of Containers**:
  - Less overhead and more efficient.
  - Increased portability across systems.
  - Consistent operation environment conducive to DevOps.
  - Accelerated development cycles.
- **Docker Basics**:
  - Installation and basic commands: `docker --version`, `docker ps`, `docker pull`, `docker run`.
  - Running a simple use case with MySQL.
- **Creating & Managing Docker Images**:
  - Building an image manually and with a Dockerfile.
  - Testing locally before pushing to Docker Hub.
  - Tagging and sharing images on Docker Hub.

References

- Continuous Delivery (Chapter 7 & 8) by Jez Humble and David Farley.
- Regression Testing articles (testsigma.com)
- Docker Official Documentation and Docker Hub.

Class Notes

Slide 1

Title: Comp 4350 Software Engineering II Lecture 5
Dr. Shaowei Wang

Slide 2

Title: Administrative items

- Meet TA next week for sprint 1 evaluation.
- Submit project proposal by next Friday to UMLearn (Assignment -> project proposal)
- Presentation to introduce your project next week. Find the schedule in the project google spreadsheet.
- Submit slides before your presentation to UMLearn (Assignments-> Project proposal presentation)

Slide 3

Title: Agenda

- Regression testing
- Continuous deployment
- Tutorial on Docker

Slide 4

System Building and Release Pipelines

Slide 5

Title: Regression testing

Slide 6

Title: Regression testing Body:

- A software testing practice that ensures an application functions after any code changes, updates, or improvements.
- Goal:
    - Ensure changes don't break existing requirements.
    - Find defects early.

Slide 7

Title: Earlier features get more exercise!

Slide 8

Title: How to do regression testing? Body:

- Uses test cases re-executed post-changes to check functionality.
- Can be done in different levels:
    - Unit testing
    - Integration testing

    - System testing
    - Acceptance testing

Slide 9

Title: Complete vs partial ?

Slide 10

- Quality (e.g., coverage)
- Cost (e.g., time/resource)

Slide 11

Slide 12

Title: Complete regression - expensive Body:

- Conducted when:
    - Software updates affect the code foundation.
    - Multiple changes are added.
    - The update greatly affects the codebase.
- (Gp:) No free lunch!

Slide 13-14

Slide 15

Title: Tips for test cases selection/prioritization

- Select test cases based on recent code/functional changes.
- Requires a link between code and test cases.

Slide 16-18

Title: Tips for test cases selection/prioritization Body:

- Select test cases for Regression testing where there are recent code changes or functional changes.
- Select test cases for Regression testing in areas with frequent bugs/defects.
- What do we need?
    - Requires the link between code and test cases
    - Requires the history of the bug fixing

Slide 19

Title: Defects frequency heatmap

Slide 20

Title: Defect prediction at Google

- Based on bug occurrence and age for scoring files.

- Score (f) = [Equation]
- Source: Google's Bug Prediction

Slide 21-23

Title: Tips for test cases selection/prioritization Body:

- Select test cases for Regression testing where there are recent code changes or functional changes.
- Select test cases for Regression testing in areas with frequent bugs/defects.
- Choose test cases with critical functionalities.

Slide 24

Title: Why Is Automated Acceptance Testing Essential? Body:

- Rapid release for modern software systems.
- Manual testing is expensive and slow.
- Protects applications during large-scale changes.

Slide 25

Title: Automate Functional testing tool - TESTIM Link: TESTIM - Regression Testing

Slide 26

Title: Drawback Body:

- Won't work properly if UI is modified (UI perspective)
- Apply it once your UI is stable

Slide 27

Title: Agenda

- Regression testing
- Continuous deployment
- Tutorial on Docker

Slide 28

Title: Continuous Deployment

- Software production process deploying changes automatically to production.

Slide 29

Title: Typical deployment pipeline

- Production environment

Slide 30

Title: Releasing strategies

- Blue-green deployments
- Canary Releasing
- A/B Testing

Slide 31

Title: Blue-green deployments Body:

- Run two identical production environments (Blue and Green).
- Only one live at any time.
- Source: Martin Fowler - BlueGreenDeployment

Slide 32-34

Title: Canary Releasing Body:

- Gradual rollout of new software versions to a subset of users.
- Increases confidence in the new version before full release.
- Source: Martin Fowler - CanaryRelease

Slide 35

Title: A/B Testing Body:

- A method to test features for usability, popularity, etc.
- Compares versions (A vs B) to determine better functionality.

Slide 36

Title: Use nginx to distribute user requests Link: KubeSphere Ingress Canary

Slide 37

Title: Agenda

- Regression testing
- Continuous deployment
- Tutorial on Docker

Slide 38

Title: Good practice in DevOps Body: A container is a standard unit of software packaging code and its dependencies for quick and reliable application runs across computing environments.

Slide 39

Title: What is container? Body: An isolated, lightweight silo for running an application on the host operating system, sharing the kernel.

Slide 40

Title: What is a virtual machine (VM)? In contrast to containers, VMs run a complete operating system, including its own kernel.

Slide 41

Link: VM vs Containers - Backblaze

Slide 42

Title: Benefits of containers

- Less overhead
- Increased portability
- More consistent operation
- Better application development
- Greater efficiency

Slide 43

Title: Uses for VMs vs Uses for Containers

- VMs for running apps requiring full OS resources.
- Containers for maximizing apps on minimal servers.

Slide 44-45

Slide 46

Title: Outline

- Step 00 - Installing Docker
- Step 01 - A simple Docker use case - Run mysql
- Step 02 - Playing with Docker - Containers and Images
- Step 03 - Manually creating a docker image
- Step 04 – Push image to docker hub

Slide 47

Title: Step 00 - Installing Docker Link: Docker Installation Guide Check installation:

- `docker --version`

Slide 48-49

Title: Step 01 - A simple Docker use case - Run mysql

- Pull the mysql image.
- Start a MySQL instance.

Slide 50

- Get into the interaction mode of the container.

- Play with MySQL commands.

Slide 51

Title: Step 01 - A simple Docker use case - Run mysql

Slide 52

Title: Step 02 - Playing with Docker - Containers and Images

- List running containers.
- Restart/Stop/Remove containers.

Slide 53

Title: Step 02 - Playing with Docker - Containers and Images

Slide 54

Title: Step 03 - Manually creating a docker image

- Prepare a jar package.
- Test the jar in the local machine.

Slide 55

Title: Step 03 - Manually creating a docker image

- Copy the jar into a Java container.
- Run the jar in the container.

Slide 56

Title: Step 03 - Manually creating a docker image

- Create an image from the running container.
- Run a container from the created image.

Slide 57

Title: Step 04 – Push image to docker hub

- Create an account in docker hub.
- Create a repo.
- Retag and push the image.

Slide 58

Title: Reference

- Chapter 7 & 8, Jez Humble and David Farley, Continuous Delivery.
- Test Case Selection References
- Tips for Test Case Selection

Slide 59

Title: Create your image based on existing OS

- Pull Ubuntu image.
- Install packages within the container.