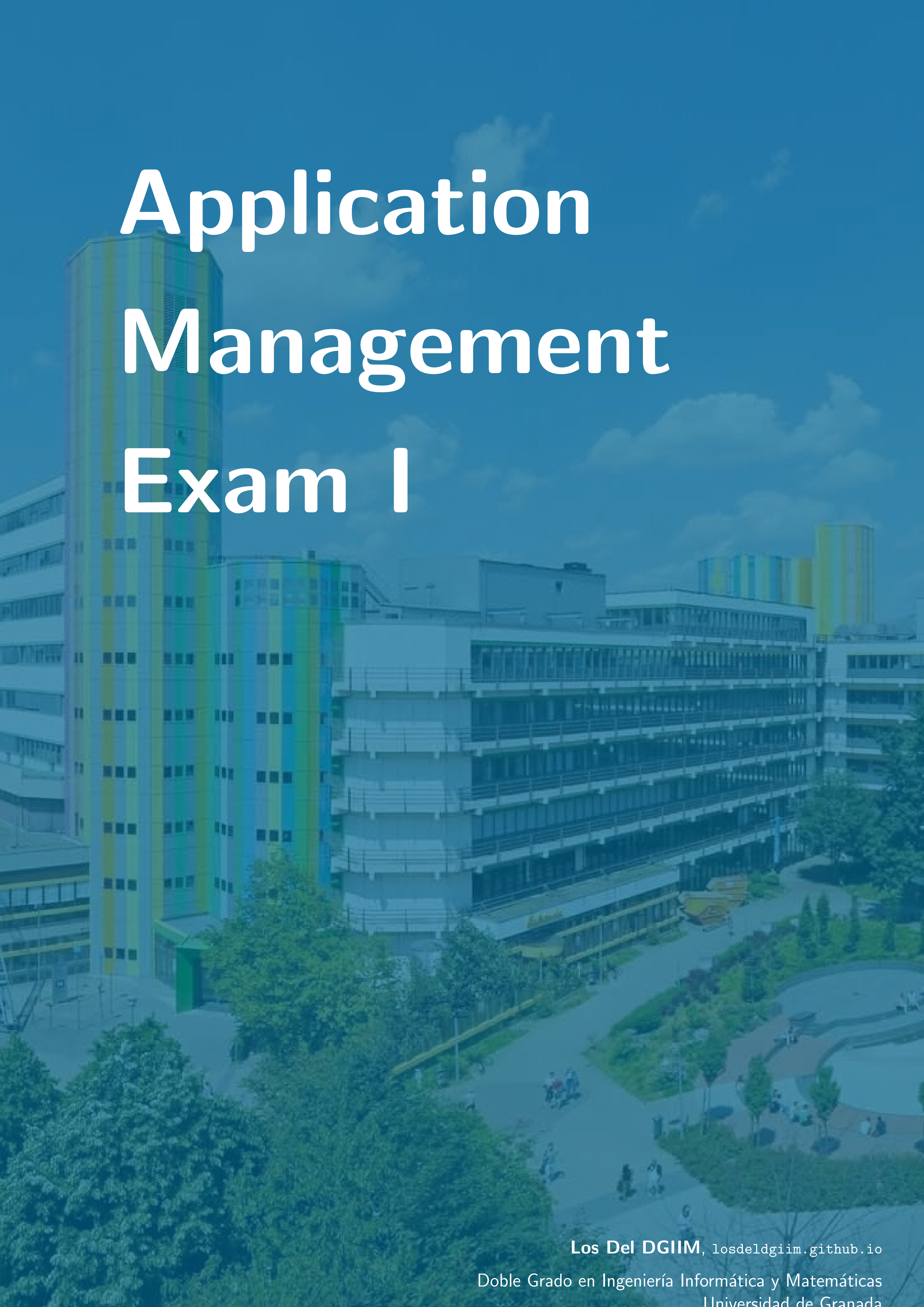
Application Management Exam I



Los Del DGIIM, `losdeldgiim.github.io`

Doble Grado en Ingeniería Informática y Matemáticas

Universidad de Granada

Application Management Exam I

Los Del DGIIM, losdeldgiim.github.io

Arturo Olivares Martos

Granada, 2025-2026

Asignatura Application Management.

Curso Académico Winter Semester 2024-25.

Fecha 13 de Febrero de 2025.

Ejercicio 1 (Application Lifecycle Management (ALM))**.**

1. (1 Punkt) Nennen Sie vier Phasen des ALM, welche auch dem Wasserfallmodell zugeordnet werden können.

   Requirements, Design, Implementation, Testing, Deployment or Maintenance. Basically, all the phases of the software development lifecycle (SDLC) can be considered part of ALM, but the most commonly referenced ones are those mentioned above.

2. (2 Punkte) Welche(n) Vorteil(e) bietet die Wahl einer permissiven Lizenz wie Apache 2.0 oder MIT?

   a) Förderung der breiten Nutzung und Akzeptanz.

   b) Strikte Kontrolle über alle Modifikationen.

   c) Geringere Hürden für kommerzielle Nutzung.

   d) Verpflichtung zur Offenlegung aller Änderungen.

3. (2 Punkte) Welche der folgenden Aussagen zum Application-Lifecycle-Management (ALM) trifft/treffen zu?

   a) ALM umfasst neben Entwicklung auch Betriebs- und Wartungsprozesse.

   b) ALM ist nur relevant für agile Softwareprojekte.

   c) ALM setzt sich häufig aus kontinuierlichem Monitoring und Feedback zusammen.

   d) Ein zentrales Ziel von ALM ist die Transparenz über die gesamte Lebensdauer einer Software.

   The first and last statements are obviously correct. The second statement is also obviously wrong, as ALM is relevant for all types of software projects, not just agile ones. The third statement is also correct, as continuous monitoring and feedback are often key components of effective ALM practices. **Therefore, the correct answers are: a), c), and d).**

4. (2 Punkte) Welche der folgenden Aussagen trifft/treffen auf die Shift-Left-Teststrategie zu?

   a) Tests möglichst spät durchführen, damit mehr Zeit für die Entwicklung bleibt.

   b) Probleme möglichst früh erkennen und beheben, um Kosten zu senken.

   c) Den Testaufwand im Betrieb auf ein Minimum reduzieren.

   d) Nur kritische Funktionen in einer frühen Phase testen.

   The first and third statements are obviously incorrect, as the Shift-Left-Teststrategie emphasizes the importance of testing early in the development process to identify and address issues as soon as possible. The second statement is also obviously correct, as early detection and resolution of problems can indeed help reduce costs. The fourth statement is also incorrect, as the Shift-Left-Teststrategie encourages testing of all functions, not just critical ones, in an early phase to ensure overall software quality. **Therefore, the correct answer is: b).**

5. (2 Punkte) Warum ist es wichtig, den gesamten Lebenszyklus einer Software zu betrachten?

   It is important to consider the entire lifecycle of software because it allows for developping a software that not only is of a higher quality, but also is more maintainable, scalable, and adaptable to changing requirements.

6. (2 Punkte) Was versteht man unter 'Flakiness' bei Tests?

   "Flaky" Tests are tests that can produce different results (pass or fail) when run multiple times under the same conditions. This can be due to various factors such as timing issues, dependencies on external systems, or non-deterministic behavior in the code being tested. Flaky tests can lead to unreliable test results and can make it difficult to identify real issues in the software.

7. (2 Punkte) Beschreiben Sie, welche Einfluss 'flaky' Tests auf eine CI/CD haben.

   Flaky tests can have a significant negative impact on CI/CD pipelines, as they can lead to false positives (tests that pass when there is actually a problem). This can therefore lead to a lack of trust in the test suite, making it harder for developers to identify and fix real issues in the code. Additionally, flaky tests can cause delays in the development process, as developers may need to spend time investigating and addressing the flaky tests instead of focusing on actual code changes.

8. (4 Punkte) Nennen Sie vier Stakeholder, welche typischerweise in den ALM-Prozess involviert sind und beschreiben Sie diese kurz.

   - Developper: They are responsible for writing the source code and implementing the features of the software.
   - QA Team: They are responsible for testing the software to ensure it meets the required quality standards and is free of bugs. They should not be the same as the developers, as they need to have an independent perspective to effectively identify issues.
   - Client: They are the ones who have commissioned the software and will be using it. They provide requirements and feedback throughout the development process to ensure the final product meets their needs.
   - Bussiness Analyst: They study the market and the business needs to help define the requirements for the software.

Ejercicio 2 (Versionskontrolle)**.**

1. (2 Punkte) Welche der folgenden Vorteile bietet ein verteiltes Versionskontrollsystem?

   a) Jeder Entwickler hat eine vollständige Kopie des Repositories.

   b) Änderungen werden automatisch mit allen anderen synchronisiert.

   c) Es gibt immer einen zentralen Server, der alle Änderungen speichert.

   d) Arbeiten an Branches können unabhängig voneinander erfolgen.

   The first and fourth statements are obviously correct. The second is obviously false, as changes are not automatically synchronized with all other developers in a distributed version control system (`push` and `pull` operations are required for synchronization). The third statement is also false, as distributed version control systems do not rely on a central server to store all changes; instead, each developer has a complete copy of the repository. **Therefore, the correct answers are: a) and d).**

2. (2 Punkte) Welche Rolle spielen Hashes bei Git?

   a) Sie identifizieren eindeutig jede Version einer Datei oder eines Commits.

   b) Sie verhindern Datenverlust bei Netzwerkausfällen.

   c) Sie ermöglichen es Entwicklern, Konflikte automatisch zu lösen.

   d) Sie gewährleisten die Integrität der gespeicherten Daten.

   The first and fourth statements are correct. The second statement is incorrect, as hashes have nothing to do with network failures. The third statement is also incorrect, as conflicts in Git should be resolved manually by developers, not automatically. **Therefore, the correct answers are: a) and d).**

3. (6 Punkte) Diskutieren Sie die Vor- und Nachteile von Rebase vs. Merge in Bezug auf:

   - Der Übersichtlichkeit der Historie

     In terms of history clarity, merge commits can make the history more complex and harder to read, especially if there are many branches and merges. On the other hand, rebasing creates a cleaner, linear history by applying commits on top of the target branch, which can make it easier to understand the sequence of changes.

   - Des Konfliktmanagements

     Merge can lead to more complex conflict resolution, as it combines the histories of two branches, which may have diverged significantly. Rebase can simplify conflict resolution by applying commits one at a time, allowing developers to address conflicts in a more controlled manner.

   - Der Teamabsprachen (z. B. wann force push nötig wird).

     Rebase can require force pushing if the branch has already been shared with others, which can lead to complications and potential data loss if

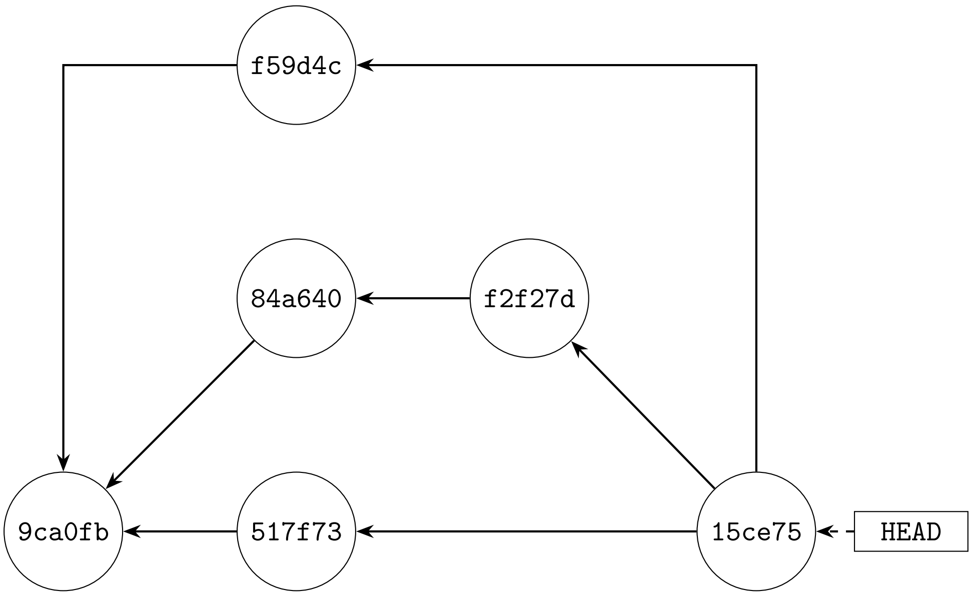


Figura 1: Commit-Historie

```
HEAD Historie:
15ce75 merge f2f27d, f59d4c
517f73 checkout
f2f27d commit
84a640 checkout
f59d4c commit
9ca0fb checkout
```

Código fuente 1: `HEAD` Historie

not handled carefully. Merge does not require force pushing, as it creates a new commit that combines the changes from both branches without altering the existing commits. Therefore, teams need to have clear agreements on when to use rebase and how to handle force pushes to avoid issues in collaborative workflows.

4. (5 Punkte) Es wurde folgender Befehl ausgeführt: `git checkout HEAD^2`. Auf welchen Commit zeigt `HEAD`? Begründen Sie Ihre Antwort. In Figur 1 ist die Commit-Historie dargestellt, und in Listing 1 die Historie von `HEAD`.

   The `HEAD` is currently pointing to the commit `15ce75`, which is a merge commit that has three parents:

   a) Father 1: `517f73`, which is the commit that was checked out before the merge.

   b) Father 2: `f2f27d`, given that it is written first in the HEAD history

   c) Father 3: `f59d4c`, given that it is written second in the HEAD history

   Therefore, the command `git checkout HEAD^2` will move the `HEAD` to the second parent of the current commit, which is `f2f27d`. Thus, after executing the command, `HEAD` will point to the commit `f2f27d`.

```
1. git revert [--no-edit] [-n] [-m <parent-number>] [-s] [-S[keyid]] <commit>...
2. git revert [--continue | --skip | --abort | --quit]

> Wenn ein oder mehrere bestehende Commits angegeben werden, werden die durch
die zugehörigen Patches eingeführten Änderungen rückgängig gemacht. Dabei
entstehen neue Commits, die diesen Vorgang festhalten. Voraussetzung dafür ist,
dass dein Working Tree sauber ist.
```

Código fuente 2: Git Revert Befehl

5. (4 Punkte) Ein Entwicklerteam verwendet Git für ein Webprojekt. Aus Versehen hat ein Teammitglied in einem früheren Commit eine Datei `.env` eingecheckt, in der sensible Zugangsdaten (API-Keys, Datenbankpasswörter) enthalten sind. Das ist dem Team erst nach einigen Tagen aufgefallen, nachdem bereits weitere Commits und Branches erstellt wurden. Das Repository ist öffentlich auf GitHub verfügbar. Die Git-Dokumentation beschreibt den Befehl `git revert` wie in Listing 2 dargestellt. Begründen Sie, wie und ob dieser Befehl dazu beitragen kann, das Problem zu beheben. Sollten der Befehl nützlich sein, geben Sie außerdem den abgeätzten Befehl an, welcher das Problem behebt.

   It depens on the final goal of the team.

   - The only thing that `git revert` can do is to create a new commit that undoes the changes introduced by the commit that added the `.env` file. It should be done by executing the command `git revert <commit-hash>`, where `<commit-hash>` is the hash of the commit that added the `.env` file. This will create a new commit that removes the `.env` file from the repository's history.
   - However, it should be noted that, even though the `.env` file will be removed from the current state of the repository, it will still be present in the commit history. Therefore, if the goal is to completely remove the sensitive data from the repository, `git revert` alone will not be sufficient. With the `git checkout` command, the team can move to the commit before the one that deleted the `.env` file, and then obtain the file from there. This means that the sensitive data will still be accessible through the commit history, which can be a security risk if the repository is public.

6. (2 Punkte) Reicht das Entfernen aus Git aus, um die Sicherheit vollständig wiederherzustellen? Begründen Sie Ihre Antwort.

   It depends on what does "removing from Git" mean.

   - If it means just deleting the file from the current state of the repository, then it is not sufficient to fully restore security, as the sensitive data will still be present in the commit history.
   - If it means actually deleting the file from the entire commit history (deleting also the blob that contains the file's content), then even though it

would remove the sensitive data from the repository, it may not be sufficient to fully restore security, as other people could have gained access to the sensitive data while it was still present in the repository, and they could have made copies of it. Additionally, if the repository is public, there may be other copies of the repository (e.g., forks) that still contain the sensitive data, which can also pose a security risk.

In this case, the team should also consider revoking any compromised credentials and generating new ones, as well as monitoring for any unauthorized access that may have occurred while the sensitive data was exposed.

7. (2 Punkte) Welche Maßnahmen würden Sie dem Team vorschlagen, um solche Vorfälle in Zukunft zu vermeiden? Nennen Sie mindestens zwei.

   There are several measures that the team can take to prevent such incidents in the future:

   - Implementing a pre-commit hook that checks for sensitive files (e.g., `.env` files) and prevents them from being committed to the repository.
   - Adding these sensitive files to the `.gitignore` file to ensure that they are not accidentally committed to the repository.
   - Providing training and awareness to the team about the importance of keeping sensitive data out of version control and the potential risks associated with exposing such data.
   - Using secret management tools to securely store and manage sensitive information, rather than keeping it in files that could be accidentally committed to version control.
   - Using 2FA (Two-Factor Authentication) for accessing the repository to add an extra layer of security in case credentials are compromised.

Ejercicio 3 (Deployment und Delivery)**.**

1. (2 Punkte) Welche der folgenden Aussagen trifft/treffen auf Continuous Deployment zu?

    a) Jede Codeänderung durchläuft manuelle Tests, bevor sie deployed wird.

    b) Jede erfolgreich getestete Codeänderung wird automatisch in Produktion deployed.

    c) Continuous Deployment erfolgt nur bei erfolgreichen Software-Releases.

    d) Continuous Deployment setzt keine vollständige Automatisierung voraus.

    The first statement is incorrect, as Continuous Deployment emphasizes automation and does not require manual testing for every code change. The second statement is correct, as Continuous Deployment involves automatically deploying every successfully tested code change to production. The third statement is also correct because, if any of the code changes fail the tests, they will not be deployed to production. The fourth statement is incorrect, as Continuous Deployment relies heavily on automation to ensure that code changes are deployed efficiently and reliably. **Therefore, the correct answers are: b) and c).**

2. (2 Punkte) Welche der folgenden Aussagen zu Canary Releases ist korrekt?

    a) Alle Benutzer erhalten sofort die neue Version.

    b) Die neue Version wird erst einer kleinen Nutzergruppe zur Verfügung gestellt.

    c) Ein Rollback ist nicht möglich.

    d) Canary Releases sind nicht für sicherheitskritische Systeme geeignet.

    The first and third statements are incorrect, and the second statement is correct. Regarding the fourth statement, Canary Releases are indeed suitable for secure systems, as they allow for controlled rollouts and quick rollbacks in case of issues. **Therefore, the correct answer is: b).**

3. (3 Punkte) Welche Herausforderungen müssen Unternehmen bei der Umstellung auf automatisiertes Deployment berücksichtigen?

    Enterprises need to consider several challenges when transitioning to automated deployment, including:

    - Ensuring that there are enough and robust automated tests in place to catch any issues before they reach production.
    - Rapid Incremental Deployment: Using also the agile principles to the deployment process, letting the team not stop the development process to wait for the automated scripts to be ready.
    - Cultural Shift: Encouraging a culture of collaboration and shared responsibility for the deployment process among development, operations, and other teams.

- Security Concerns: Implementing proper security measures to protect the deployment pipeline and the production environment from potential threats.
- Tooling and Infrastructure: Investing in the right tools and infrastructure to support automated deployment, which may require significant changes to existing processes and systems.

4. (3 Punkte) Continuous Integration, Continuous Delivery und Continuous Deployment sind zentrale Prinzipien moderner Softwareentwicklung. Erklären Sie die Unterschiede dieser Konzepte.

    - Continuous Integration (CI) focuses on automatically integrating code changes from multiple developers into a shared repository. The main goal is to detect integration conflicts early and ensure that the codebase remains in a healthy state. In addition, it focuses on building and testing the code automatically whenever changes are made, to catch issues as early as possible.
    - Continuous Delivery (CD) builds upon CI by ensuring that the code is always in a deployable state. It involves automating the release process so that code can be deployed to production at any time with minimal manual intervention. However, the actual deployment to production may still require manual approval, allowing for additional checks and controls before the code goes live.
    - Continuous Deployment takes Continuous Delivery a step further by automatically deploying every successfully tested code change to production without requiring manual approval. This approach emphasizes rapid and frequent releases, allowing for faster feedback and iteration. However, it also requires a high level of confidence in the automated testing and deployment processes to ensure that only stable and reliable code is deployed to production.

5. (2 Punkte) Nennen Sie zwei Vorteile und zwei Herausforderungen von CI/CD Pipelines:

    - Advantages:
        - Faster Feedback: CI/CD pipelines provide rapid feedback on code changes, allowing developers to identify and address issues quickly.
        - Improved Quality: By automating testing and deployment processes, CI/CD pipelines can help ensure that only high-quality code is released to production.
    - Challenges:
        - Initial Setup Complexity: Setting up a CI/CD pipeline can be complex and time-consuming, requiring significant effort to configure the necessary tools and processes.
        - Maintenance Overhead: CI/CD pipelines require ongoing maintenance to ensure they continue to function effectively, which can be resource-intensive and may require dedicated personnel.

```yaml
services:
    magicweb1:
        image: MagicWeb
        ports:
            - "8081:8080"
        container_name: magicweb_container1
    magicweb2:
        image: MagicWeb
        ports:
            - "8082:8080"
        container_name: magicweb_container2
```

Código fuente 3: Docker-Compose Datei

6. (6 Punkte) Sie haben eine Binary (MagicWeb), welche einen Webserver auf Port 8080 startet. Diese Binary haben Sie frisch erstellt. Da diese Binary Ihre kompletten Geheimnisse enthält, möchten Sie nicht, dass die Binary im Internet verfügbar ist, daher haben Sie dafür gesorgt, dass Ihr Rechner nicht mit dem Internet verbunden ist. Sie möchten diese Binary aber zweimal ausführen, deshalb haben Sie Docker gerade frisch installiert. Dazu wollen Sie die Docker-Compose Datei wie in Listing 3 nutzen.

   Wenn Sie nun `docker compose up` ausführen, beschreiben Sie was nun passiert. Sollte der Befehl nicht funktionieren, beschreiben Sie, welche Schritte man vornehmen müsste, um das Problem zu beheben.

   When you run `docker compose up`, Docker will attempt to start two containers based on the `MagicWeb` image.

   a) It will first check if the `MagicWeb` image is available locally. Given that you have just installed Docker and have not pulled or built the image, it will not be available locally.

   b) Docker will try to pull the `MagicWeb` image from a Docker registry (e.g., Docker Hub). However, since your computer is not connected to the internet, this will fail, and you will receive an error message indicating that the image cannot be found or pulled.

   Therefore, the `docker compose up` command will not work as expected due to the lack of internet connectivity and the absence of the `MagicWeb` image locally. To fix this issue, you need to previously build the `MagicWeb` image locally using a Dockerfile that defines how to build the image from your binary. This can be done by creating a Dockerfile with the necessary instructions to copy the binary into the image and set up the environment for it to run. Once you have built the image locally, you can then run `docker compose up` successfully, as it will find the `MagicWeb` image in your local Docker registry and start the containers as defined in the Docker-Compose file.

Ejercicio 4 (Secure Software Development)**.**

1. (2 Punkte) Welche der folgenden Aussagen trifft/treffen auf Netflix Chaos Monkey zu?

   a) Es simuliert und testet zufällige Ausfälle von Systemen, um die Infrastruktur zu verbessern.

   b) Es dient dazu, sensible Daten in Cloud-Umgebungen zu verschlüsseln und vor Bedrohungen zu schützen.

   c) Es ist ein Tool zur automatisierten Schwachstellenanalyse in Webanwendungen.

   d) Es analysiert den Softwarecode auf unsichere Abhängigkeiten und veraltete Bibliotheken.

2. (2 Punkte) Welche der folgenden Aussagen trifft/treffen zu?

   a) Mutational Fuzzing ist eine Technik, bei der Eingaben durch gezielte synthetische Constraints generiert werden.

   b) Symbolic Execution eignet sich besonders für Programme mit vielen Verzweigungen.

   c) Coverage-Guided Fuzzing nutzt Feedback über erreichte Code-Pfade, um Eingaben systematisch zu generieren.

   d) Symbolic Execution kann auch auf Binärprogramme angewendet werden, ohne zusätzliche Modellierung.

   The first statement is incorrect, as Mutational Fuzzing is a technique that generates inputs by mutating existing valid inputs, rather than using targeted synthetic constraints. The second statement is also incorrect given the Path Explosion problem, which makes Symbolic Execution less effective for programs with many branches. The third statement is correct, as Coverage-Guided Fuzzing uses feedback about which code paths have been executed to systematically generate new inputs that explore untested paths. The fourth statement is also incorrect, as applying Symbolic Execution to binary programs typically requires additional modeling to handle the complexities of binary code and its execution environment. **Therefore, the correct answer is: c).**

3. (2 Punkte) Warum ist Fuzzing ein effektives Mittel zur Identifizierung von Sicherheitslücken?

   Fuzzing is an effective method for identifying security vulnerabilities because it can automatically generate a large number of test inputs, including unexpected or malformed inputs, that may trigger edge cases or hidden bugs in the software. This can help uncover vulnerabilities that may not be easily found through manual testing or code review. Additionally, fuzzing can be used to test the robustness and resilience of software against various types of input, which is crucial for identifying potential security weaknesses.

4. (3 Punkte) Beschreiben Sie, wie Symbolische Ausführung (`SymExe`) mit einem SMT-Solver wie Z3 systematisch Pfade in einem Programm analysiert.

   Symbolic Execution (SymExe) is a technique used to systematically analyze the paths in a program by treating inputs as symbolic variables rather than concrete values. When a program is executed symbolically, it generates path constraints based on the conditions encountered during execution. These constraints represent the conditions under which certain paths in the program would be taken. A SMT (Satisfiability Modulo Theories) solver like Z3 can then be used to solve these constraints to determine if there are any inputs that would lead to specific paths being executed. For example, if SymExe encounters a conditional statement (e.g., `if (x > 5)`), it would generate a path constraint (e.g., `x > 5`) for the true branch and (e.g., `x <= 5`) for the false branch. Z3 can then be used to find concrete values for `x` that satisfy these constraints, allowing the analysis to explore different execution paths and identify potential vulnerabilities or bugs in the program.

5. (2 Punkte) Falls die symbolische Ausführung einen kritischen Pfad entdeckt, welche Informationen würde Z3 dazu typischerweise liefern?

   If symbolic execution discovers a critical path, Z3 would typically provide information about the specific input values that would trigger that path. This could include the values of symbolic variables that satisfy the path constraints, which can help developers understand how to reproduce the issue and identify potential vulnerabilities in the code.

6. (4 Punkte) Nennen Sie zwei Security-Testing-Methoden und geben Sie für jede ein Beispiel.

7. (4 Punkte) Wie verbessert eine mehrstufige Zertifikatskette die Sicherheit?

   A multi-level certificate chain improves security by providing a hierarchical structure of trust. In a certificate chain, each certificate is signed by a trusted authority, and the chain of trust extends from the end-entity certificate (e.g., a website's SSL certificate) up to a root certificate authority (CA). This structure allows for better management of trust and revocation. If a certificate in the chain is compromised or needs to be revoked, it can be done without affecting the entire chain, as long as the root CA remains secure. Additionally, multi-level certificate chains can provide additional layers of validation and authentication, making it more difficult for attackers to forge certificates or impersonate legitimate entities.

8. Was ist binary provenance und was kann passieren wenn dies nicht gewährt ist? Nennen Sie zwei Fälle.

   Binary provenance refers to the ability to trace the origin and history of a binary file, including information about how it was built, what source code it was derived from, and any dependencies it may have. If binary provenance is not provided, it can lead to several issues:

   - Security Risks: Without binary provenance, it becomes difficult to verify the authenticity and integrity of a binary file. This can lead to security risks, as malicious actors could potentially distribute tampered or malicious binaries without detection.

- Compliance Issues: In certain industries, there may be regulatory requirements for software supply chain transparency. Without binary provenance, organizations may struggle to meet these compliance requirements, which could result in legal consequences or loss of trust from customers and partners.