Modul “Cloud Computing und Big Data”

Studiengang "Data Science und Künstliche Intelligenz", Sommersemester 2025

Dennis Pfisterer, dennis.pfisterer@dhbw.de, Version vom 08.05.2025

Prüfungsleistung: Portfolio

Der Lernerfolg wird anhand einer Portfolio-Prüfung evaluiert, die in Gruppen von jeweils ca. fünf Personen bearbeitet wird. Das Portfolio soll den individuellen Lern- und Entwicklungsprozess widerspiegeln und zeigen, wie die erlernten Konzepte angewendet und verstanden werden.

Es ist aus mehreren studienbegleitenden Prüfungsteilen zusammengesetzt, sodass der Fortschritt kontinuierlich verfolgt wird und somit nicht am Ende eine große Aufgabe zu lösen ist ohne eine fachliche Begleitung in der Veranstaltung. Es erfordert aber auch eine kontinuierliche Beschäftigung mit den Technologien.

Anforderungen an die Abgabe

- Gesamte Abgabe als eine ZIP-Datei in Moodle (es wird eine Zusammenarbeit z.B. via git empfohlen)
- Keine externen Dateien und Verweise auf externe Inhalte (wie z.B. YouTube, Google Drive, Github, etc.)
- Eine README.md ist enthalten
- Alle eingesetzten Technologien sind Open Source

Übergeordnete, wesentliche Bewertungskriterien

- Die README.md dient als Gesamtdokumentation aller Abgaben in einem Dokument
- Nachvollziehbarkeit ist gegeben und es wird auf relevante Code-Abschnitte verwiesen oder diese werden eingebettet in die Dokumentation
- Der vollständige Code und alle Befehle sind enthalten, kommentiert, formatiert und verständlich
- Es sind Zielsetzung, Architektur, Entwurf sowie Screencasts enthalten (kleine Dateien mit hoher Kompression)
- Es ist eine klare Trennung der Aufgaben erkennbar
- Abbildungen sind von hoher Qualität
- Rechtschreibung und Grammatik sind einwandfrei
- Fachliche Richtigkeit und Tiefe des Verständnisses
- Qualität der praktischen Umsetzung
- Kreativität und Innovationsgrad der Anwendungsidee
- Struktur und Klarheit der Dokumentation

Aufgaben

Aufgabe 1: Immutable Infrastructure

Es soll eine Infrastruktur nach dem Prinzip der „Immutable Infrastructure“ entworfen und mit Ansible oder einer selbstgewählten Technologie (kann einen Bonus erzielen) bereitgestellt werden.

- Zuerst ist eine Technologie für die Infrastrukturautomatisierung auszuwählen und die Wahl kurz zu begründen.
- Anschließend wird eine einfache Immutable-Komponente entworfen und deren Konfiguration sowie die Sicherstellung der Unveränderlichkeit beschrieben.
- Die Implementierung mit der gewählten Technologie inklusive Dokumentation der Schritte und der Demonstration eines „Immutable Updates“ folgt.

Abschließend werden Technologie, Entwurf, Implementierung und Demonstration in einer kurzen Dokumentation inkl. Screencast zusammengefasst.

Die Bewertung erfolgt nach Verständnis von „Immutable Infrastructure“, Technologieauswahl und -anwendung, Qualität von Entwurf und Implementierung, Vollständigkeit der Dokumentation sowie Klarheit der Präsentation. Bonuspunkte können für unkonventionelle Technologien, innovative Entwürfe, Stateful Applications oder effiziente Implementierungen vergeben werden. Ziel ist die praktische Anwendung des Konzepts und die Förderung der Technologiekompetenz.

Aufgabe 2: Configuration Management and Deployment

Aufbauend auf der vorherigen Aufgabe zur „Immutable Infrastructure“ soll nun die Installation und Versionierung einer Anwendung auf der bereitgestellten Infrastruktur betrachtet werden. Ziel ist es, das Zusammenspiel zwischen Infrastruktur und Anwendung zu demonstrieren und die Möglichkeit zu schaffen, bestimmte Versionen der Infrastruktur zu installieren und alte Versionen zu verwalten. Die Aufgabenstellung erweitert die vorherige Aufgabe und umfasst folgende Punkte:

- Erweiterung der Infrastrukturdefinition: Die bestehende Infrastrukturdefinition (z.B. Terraform-Konfiguration) wird um die Installation und Konfiguration einer Beispielanwendung erweitert. Dies kann eine einfache Webanwendung (z.B. Node.js, Python, etc.) oder ein anderer Dienst sein. Die Anwendung soll über die Infrastrukturdefinition installiert und gestartet werden.
- Versionierung der Anwendung: Es soll demonstriert werden, wie verschiedene Versionen der Anwendung bereitgestellt und verwaltet werden können. Dies kann durch Mechanismen zur Anwendungs-Versionierung (z.B. verschiedene Branches im Code-Repository) erfolgen.
- Infrastruktur-Versionierung und Rollback: Die erstellte Infrastrukturdefinition soll versioniert werden (z.B. mit Git). Es soll demonstriert werden, wie eine bestimmte Version der Infrastruktur (inklusive der Anwendung) installiert und bei Bedarf auf eine ältere Version zurückgerollt werden kann. Dabei soll auch gezeigt werden, wie alte Infrastrukturversionen entfernt werden können, um Ressourcen zu sparen.
- Dokumentation: Die erweiterte Infrastrukturdefinition, die Schritte zur Anwendungs-Bereitstellung und Versionierung sowie die Vorgehensweise zum Rollback auf ältere Infrastrukturversionen sind detailliert zu dokumentieren.
- Präsentation: Abschließend werden Technologie, Entwurf, Implementierung und Demonstration in einer kurzen Dokumentation inkl. Screencast zusammengefasst und als Präsentation dem Kurs vorgeführt.

Bonuspunkte können auch hier für unkonventionelle Technologien (abseits der in der Veranstaltung vorgestellten), innovative Entwürfe oder effiziente Implementierungen vergeben werden. Die Bewertung erfolgt anhand der Funktionalität der Implementierung, der Qualität der Dokumentation, des Verständnisses der Konzepte und der Präsentation.

Aufgabe 3: Microservice-Architektur

Im Rahmen der Portfolio-Prüfung soll eine Multi-Node-Kubernetes-Infrastruktur auf der OpenStack-Umgebung bereitgestellt werden, auf der eine containerisierte und versionierbare Anwendung mit Skalierbarkeit und externer Erreichbarkeit deployet wird.

- Zunächst ist eine geeignete Technologie zur Installation der Kubernetes-Umgebung in OpenStack zu wählen (effektive Wahl wird ggf. mit Bonuspunkten honoriert).
- Anschließend wird eine Anwendung containerisiert und versionierbar auf dem Kubernetes-Cluster deployet. Die Anwendung soll skalierbar sein und über Ingress von außen erreichbar gemacht werden.
- Optional kann die Performance der Anwendung per Prometheus überwacht werden (Bonus).

- Die Abgabe umfasst die Dokumentation der Installationsschritte, der Containerisierung, der Kubernetes-Konfigurationen (Deployment, Service, Ingress), der Versionierungs- und Skalierungsstrategien sowie (optional) des Monitorings. Die Präsentation demonstriert die Funktionalität der Kubernetes-Umgebung, des Anwendungsdeployments, der Versionierung, der Skalierbarkeit und der externen Erreichbarkeit sowie (optional) des Performance-Monitorings.

Bewertet werden Funktionalität, Konzepte, Dokumentation und Präsentation. Bonuspunkte gibt es für effektive Technologien, innovative Ansätze und erfolgreiches Performance-Monitoring.

Aufgabe 4: Data Lake / Big Data-Processing

In diesem Teil steht die Installation eines verteilten Data Lakes und die Verarbeitung von Big Data in der zuvor aufgebauten Multi-Node-Infrastruktur im Fokus.

- Zunächst soll ein verteilter Data Lake (z.B. Apache Hadoop) in der bestehenden Multi-Node-Umgebung installiert werden. Die Wahl einer alternativen Technologie für den Data Lake wird mit Bonuspunkten honoriert.
- In diesem Data Lake sind anschließend verschiedene Datensätze von nicht-trivialer Größe zu speichern.
- Die gespeicherten Daten sollen dann mithilfe einer Big Data-Verarbeitungstechnologie (z.B. Apache Spark) verarbeitet werden. Die Ergebnisse der Verarbeitung sind wiederum im Data Lake zu speichern. Die Verwendung einer alternativen Processing-Engine (z.B. Apache Flink) oder der Einsatz einer darauf aufbauenden Machine Learning-Bibliothek (z.B. Spark MLlib) kann mit zusätzlichen Bonuspunkten gewürdigt werden.

Wichtige Bewertungskriterien sind die Komplexität und die Kreativität der gewählten Lösung für den Data Lake und die Datenverarbeitungspipeline. Die Dokumentation der Installation, der Datenhaltung und der Verarbeitungsschritte sowie eine Präsentation der Ergebnisse sind erforderlich.

Aufgabe 5: Big Data-Stream Processing

Nach der Batch-Verarbeitung von Daten soll nun der Fokus auf gestreamte Daten aus einem Data Ingestion-Cluster (wie z.B. einem Kafka-Cluster) gelegt werden. Wichtig ist hierbei die horizontale Skalierbarkeit. Die Wahl einer alternativen Technologie kann wieder Bonuspunkte erzielen.

Aufbauend auf der bestehenden Multi-Node-Infrastruktur ist zunächst ein Kafka-Cluster zu installieren und zu konfigurieren. Anschließend soll eine Lösung implementiert werden, die Daten aus diesem Kafka-Cluster gestreamt verarbeitet. Die horizontale Skalierbarkeit der Verarbeitungspipeline ist dabei ein zentrales Kriterium.

Für die Verarbeitung der gestreamten Daten kann eine geeignete Stream-Processing-Engine gewählt werden (z.B. Apache Kafka Streams, Apache Flink, Spark Streaming). Die Verwendung einer nicht in der Vorlesung vorgestellten Technologie wird mit Bonuspunkten honoriert. Ebenfalls mit Bonuspunkten kann der Einsatz einer Machine Learning-Bibliothek (z.B. Spark MLlib) zur Echtzeit-Analyse oder -Modellierung der gestreamten Daten gewürdigt werden.

Die Aufgaben umfassen die Installation und Konfiguration des Kafka-Clusters, die Implementierung der Stream-Processing-Pipeline unter Berücksichtigung der horizontalen Skalierbarkeit sowie gegebenenfalls die Integration einer Machine Learning-Komponente. Die Dokumentation der Architektur, der Konfigurationen und der Skalierungsmechanismen sowie eine Präsentation der funktionalen Skalierbarkeit und der Ergebnisse sind erforderlich. Die Komplexität und Kreativität der gewählten Lösung sind wesentliche Bewertungskriterien.