Lernzettel

Praktische Umsetzung im Tutorium: kleine Programmieraufgaben, Containerisierung (Docker), Orchestrierung (Kubernetes), API-Design- und Integrationsübungen

Universität:	Technische Universität Berlin
Kurs/Modul:	Architektur von Anwendungssystemen
Erstellungsdatum:	September 19, 2025

Lernzettel: Praktische Umsetzung im Tutorium: kleine Programmieraufgaben, Containerisierung (Docker), Orchestrierung (Kubernetes), API-Design- und Integrationsübungen

(1) Überblick und Zielsetzung. Dieses Teilthema behandelt die praktische Umsetzung im Tutorium zu Architekturen von Anwendungssystemen. Es umfasst:

- kleine Programmieraufgaben zur Implementierung von Microservices,
- Containerisierung der Anwendungen mit Docker,
- Orchestrierung der Container mit Kubernetes,
- API-Design und Integrationsübungen zur verteilten Kommunikation.

Ziel ist es, die in der Vorlesung eingeführten Architekturstile, Middleware-Technologien und Web-Standards in praktischen Tasks anzuwenden, zu bewerten und zu vergleichen. Die Übungen erfolgen je nach Teilnehmerzahl in Einzel- oder Kleingruppenarbeit.

(2) Praktische Aufgabenübersicht. Die Übungen gliedern sich in vier Schwerpunkte: Programmierung, Containerisierung, Orchestrierung sowie API-Design und Integration. Im Folgenden sind Beispielaufgaben und minimale Beispiele skizziert, wie typische Aufgaben aufgebaut sein können.

(2A) Kleine Programmieraufgaben. Ziel: Einen einfachen Microservice implementieren und testen. Ein kurzes Beispiel mit einem REST-Endpunkt:

```
# app.py
from fastapi import FastAPI
app = FastAPI()

@app.get("/ping")
def ping():
    return {"status":"ok"}
```

(2B) Containerisierung (Docker). Ziel: Den Microservice als Container betreiben. Beispiel für eine Docker-Datei:

```
# Dockerfile
FROM python:3.11-slim as builder
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .

FROM python:3.11-slim
WORKDIR /app
COPY --from=builder /app /app
CMD ["uvicorn","app:app","--host","0.0.0.0","--port","8000"]
EXPOSE 8000
```

(2C) Orchestrierung (Kubernetes). Ziel: Deployment und Service für den Container in Kubernetes. Beispiel-YAML-Dateien:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myservice
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myservice
  template:
    metadata:
      labels:
        app: myservice
    spec:
      containers:
      - name: myservice
        image: myorg/myservice:0.1
        ports:
        - containerPort: 8000
```

```
apiVersion: v1
kind: Service
metadata:
  name: myservice-svc
spec:
  selector:
    app: myservice
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8000
  type: ClusterIP
```

(2D) API-Design- und Integrationsübungen. Ziel: Entwurf einer gut gestalteten API samt Spezifikation und Anbindung an weitere Dienste.

```
openapi: 3.0.0
info:
  title: Example API
  version: 1.0.0
paths:
  /items:
    get:
      summary: List items
      responses:
        '200':
          description: OK
```

```
content:
  application/json:
    schema:
      type: array
      items:
        type: object
        properties:
          id:
            type: string
          name:
            type: string
```

Zusätzliches Designwissen:

- API-Versionierung und Kompatibilität sicherstellen.
- REST- oder gRPC-Interfaces klar definieren und konsistent nutzen.
- Authentifizierung, Autorisierung und sichere Übertragung (TLS).
- Fehlerbehandlung, Statuscodes und konsistente Fehlertexte.
- Logging, Monitoring und Tracing (z. B. OpenTelemetry).

(3) Integrationsübungen – Beispielszenarien.

- Dienste über HTTP oder asynchrone Nachrichten verbinden (z. B. Kafka, RabbitMQ).
- Schemas und Verträge (API-Spezifikationen) gegen Vertrauens- und Kompatibilitätsprobleme absichern.
- End-to-end Testfälle zur Verfügbarkeit und Fehlertoleranz erstellen.

Hinweise zur Umsetzung.

- Verwende sinnvolle Namensgebungen für Images, Deployments und Services.
- Behalte klare Versionskontrolle und Dokumentation der REST-Schnittstellen.
- Berücksichtige nichtfunktionale Eigenschaften wie Skalierbarkeit, Zuverlässigkeit und Sicherheit.