# Aufgabenblatt: Teamprojekt "Homepage Deployment mit Jenkins, Kubernetes und Config-Management"

## Ziel

Ihr arbeitet als gesamtes Team gemeinsam an dieser Aufgabe.

## Aufgabenbeschreibung

✅ Folgende Schritte sind umzusetzen:

* 1️⃣ Dockerfile erstellen:  
  - Homepage containerisieren mit httpd.
* 2️⃣ Image lokal bauen
* 3️⃣ Image in Azure Container Registry pushen
* 4️⃣ Kubernetes Deployments:
* - 20.125.24.149  
  - Erst Deployment auf kurs-2-dev.  
  - Nach erfolgreichem Test identisches Deployment auf kurs-2-prod.
* 5️⃣ Konfigurationsverwaltung:  
  - Ihr erhaltet mehrere Kubeconfig-Dateien (kurs2-dev.yaml, kurs2-prod.yaml).  
  - Diese sind korrekt zu mergen, sodass ein konsistenter Zugriff auf alle Cluster möglich ist.  
    
  👉 Dafür müssen folgende Befehle verwendet werden:  
  export KUBECONFIG=~/.kube/config:~/.kube/kurs2-dev.yaml:~/.kube/kurs2-prod.yaml  
  kubectl config view --flatten > ~/.kube/merged.yaml  
  mv ~/.kube/merged.yaml ~/.kube/config  
    
  Diese Vorgehensweise ist verpflichtend und Bestandteil der Abgabe!
* 6️⃣ Jenkins-Pipeline:  
  - Automatisiert Docker-Build, Push nach ACR und Deployment in kurs-2-dev und kurs-2-prod.

- secretfile für Jenkins, um config für Kubernetes mitzunehmen, ohne diese zu leaken

- optional: dynamische versionierung als bash/python-Skript implementieren

* 7️⃣ Projektdokumentation:  
  - Alles in ein GitHub-Repository pushen: Dockerfile, YAMLs, README.

## Abgabe

- GitHub-Link mit vollständigem Projekt  
- Funktionierende Jenkins-Pipeline  
- Screenshots:  
 - kubectl get pods und kubectl get svc in beiden Namespaces  
 - Browser-Test der Homepage

## Regeln

- Keine Hilfestellungen von Trainer oder KI erlaubt  
- Eigenverantwortliche Teamarbeit (alle beteiligt)  
- Dokumentation aller Arbeitsschritte im Repository

## Kurz gesagt

👉 Containerisieren. Jenkins automatisieren. Kubernetes deployen. Config-Dateien korrekt mergen. Alles in GitHub dokumentieren. Als Team, ohne Hilfe, jeder trägt Verantwortung!