CodeContender

Table of Contents

1. Introduction

CodeContender is a platform for learning and practicing coding. It is a web application that provides a set of coding courses with diffrend tasks and questions. Content is divided into different categories and levels. Users get information in form of text, images, and videos. They can also practice coding in an online code editor. The platform is designed to be user-friendly and easy to use. It is suitable for beginners as well as advanced users. Goal after finishing a course is to pass a final test and get a certificate.

2. Problem Analysis

Software Setup:

- is a cluster already running wtih the required software? (e.g. docker, kubernetes, etc.)
- Installation scripts like ansible, puppet, etc. to install the required software
- Installation scripts for the application
- Configuration files for the application
- Configuration files for the cluster

2.1. Requirements

- Authentication, Authorization, User Management

2.1.1. Content

- Courses, Categories, Levels
    - Tasks, Questions
    - Text, Images, Videos
    - Folder structure for content or standerdized format for content like IMS QTI
    - Versioning of content (e.g. git)
    - Admin Panel for managing content
    - Progress Tracking, evaluation, scoring of users
    - Submitting the solution separately from the evaluation (error timeout etc.)
    - Final Test, Certificate
    - Programming Language independent
    - Courses should have diffrent forms of questions (e.g. multiple choice, fill in the blank, etc.)
    - own definable docker images for the evaluation of the submissions (e.g. python, java, golang, etc.)
    - Possible ML or AI tasks (e.g. image recognition, video generation, etc.)

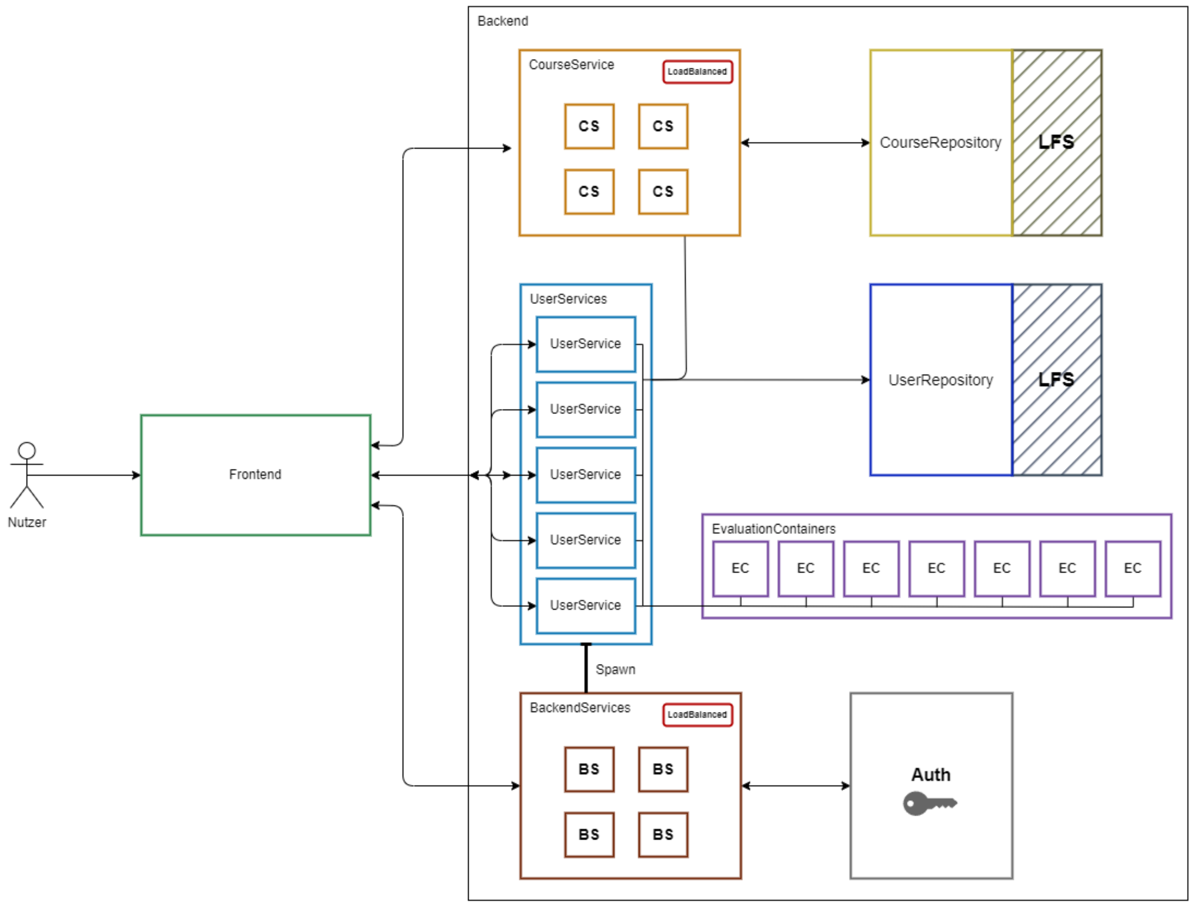
2.1.2. Evaluation of submission

- Management container per user for evaluation (security, isolation, resource management)
    - Management of containers solution (e.g. docker, kubernetes, etc.) or a extra own container management service
    - Extrecting of diffrent responses from the container (extraction and saving results)
- Submission
    - directed to extera container for user session
    - hook in datamanagement for submissions to run the evaluation → Management Container
- Evaluation
    - Evaluation of the submission
    - Running the code on extra container for evaluation (security, isolation, resource management)
    - Saving the results
    - Sending the results back to the user

2.1.3. Code Editor

- Code Editor
    - Syntax Highlighting
    - Code Completion
    - Code Execution
    - Code Testing
    - Code Sharing

3. Architecture



3.1. Components

3.1.1. Frontend

This is the user interface of the application. It is a web application that users interact with. It is responsible for displaying content, managing user interactions, and sending requests to the backend. It could be loadbalanced and scaled horizontally to handle more users. Its stateless and can be easily replaced.

3.1.2. Backend

This is the core of the application. It is responsible for handling requests from the frontend, processing data, and sending responses back. It is also responsible for managing the database, handling authentication, and managing user sessions.

3.1.3. Database

For saving progress, user data, and other information, a database is needed. It could be a traditional SQL database or a NoSQL database. It should be scalable, reliable, and secure.

No traditional database is needed for course and user data. All content is stored in a git repository. The backend fetches the content from the repository and serves it to the frontend. This makes it easy to update content, manage versions, and collaborate with others. There are diffrent git repositories for user generated content and the content of the courses. Both utilize a LFS (Large File

Storage) for storing images and videos and the ability to pull the content from the repository without the need to clone the whole repository.

3.1.4. CourseService

This is a microservice that is responsible for managing courses. It fetches content from the git repository, processes it, and serves it. The course content is cached and linked with the git revision. This makes it easy to update content and manage versions.

3.1.5. UserService

This is a microservice that is responsible for managing users input and output. It fetches the users input data for courses from the git repository, processes it, and serves it. The user content is cached and linked with the git revision. This makes it easy to update content and manage versions.

3.1.6. UserRepository

This is the git server that stores the user input data for solving tasks in the courses. It is a separate git repository that is linked with the user input data. This makes it easy to manage user input data and track changes. It follows a predefined structure for storing user input data. Each user has a separate repository with a unique ID. Inside the repository, there are diffrent branches for each course. Each branch contains the user input data for the course. This makes it easy to manage user input data and track changes. The repository makes it possible to work in your own environment and push the changes to the repository. The backend fetches the changes and updates the user input data. This makes it easy to work offline and collaborate with others.

Repositorys need to be secured and limited in size. That users can’t upload large files or malicious content. The repository should be monitored and checked for changes. It should be possible to revert changes and restore the repository to a previous state.

4. Existing Solutions

4.1. Code Grade

CodeGra.de

Codegrade is a commercial solution for grading code submissions in an external tool, integratable via the open LTI interface. It supports the automatic grading of code submissions through unit tests, I/O, error code, and linter. The platform additionally provides a code review feature with inline comments, that can be used to give feedback to the students. There is also a plagiarism detection feature to check the similarity between submissions.

[1]

4.2. CS50 with Check50

CS50 The Harvard MOOC which is build upon the edX platform doesn’t use a built-in code grading system. Instead, there is a separate tooling called CS50 with multiple checking tools for i.e. linting (`style50`), plagiarism detection (`compare50`), and unit tests (`check50`).

The check tool is written in Python and uses plain annotated Python functions to define tests, for other programming languages, extensions are used. Therefore this tool in itself only supports C, Python, and Flask and has limited support for other languages. Whoever wants to use this toolset in any language will have to get used to the Python syntax. [2]

The scalability of this solution consists of the use of GitHub repositories. The git-Wrapper `submit50` is used to store the solutions in a local repository and push them to GitHub, where CS50 staff can access them. [3]

4.3. openHPI and CodeOcean

CodeOcean

The openHPI platform of the Hasso Plattner Institute in Potsdam (HPI) is an xMOOC (Massive Open Online Course) offer that focuses on video lectures, which are reused from existing recordings in form of podcasts at the HPI. It builds upon the Canvas LMS, which was chosen for its "modern user interface and the availability of crucial functional components [...]" [4].

The implemented solution for automatic grading of programming tasks in openHPI is CodeOcean, a web based platform separately developed at the HPI. It allows learners to write, execute and test code directly in their browser without the need to set up a development environment. Features include syntax highlighting, server-side code execution and automated feedback through unit tests. Additionally, learners can ask context-related questions, which makes it a collaborative learning environment.

The highlighted goals and requirements of this work are:

- Versatility: Use of different programming languages possible
- User-friendliness: Simple user interface
- Scalability: Support for many users
- Security: Execution of foreign code in a sandbox
- Interoperability: Integration into learning management systems

The chosen approach is therefore most similar to the desired one. A predetermined number of Docker containers are provided as execution environments for the programming tasks and assigned to the users to work on their programming tasks. CodeOcean facilitates integration into other learning management systems via the open LTI interface, as is the case with most comparable solutions. [5]

There have been new developments to this platform which in include a collaboration service called CodeHarbor to exchange tasks between instances of code ocean. This could be interesting for future uses in the context of the HTWK Leipzig, depending on how the platform is used, e.g. for the use in internal courses. [6]

4.4. EmpowrOrg Coppin

EmpowrOrg Coppin is an assignment creation and assessment tool used in conjunction with **Doctor** and **CodeEditorXblock**. By combining these three tools, any organization using Open Edx can teach and assess programming assignments. Coppin integrates with **Doctor** to enable assessment of assignments and uses **CodeEditorXblock** as a code editor and executor.

Python and Swift code, can be executed and tested locally. The results are displayed in the **Doctor** interface. Evaluation is performed by automatically executing unit tests defined in the task

5. edX Platform

5.1. General information about the open source learning platform edX

Foundation and history: edX was started in 2012 by Harvard University and the Massachusetts Institute of Technology (MIT). The platform is run as a non-profit organization to make quality education accessible to all and create a community of learners and educators worldwide. Since its inception, edX has become one of the largest platforms for online learning, offering courses from universities and institutions worldwide.

Technical details:

Techstack:

- **Programming languages:** Python (mainly Django for the backend)
- **Frontend:** JavaScript, React
- **Databases:** MySQL, MongoDB
- **Containerization:** Docker
- **Continuous Integration/Continuous Deployment (CI/CD):** GitHub Actions, Jenkins

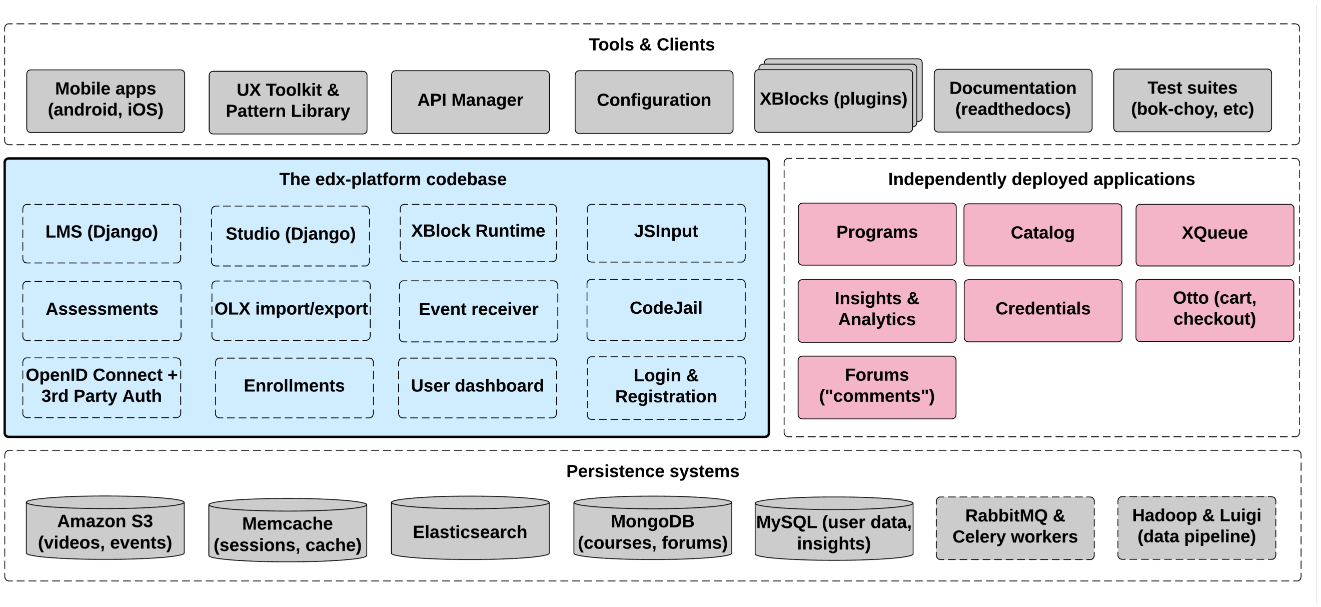
Modules and functions:

- **LMS (Learning Management System):** Management of course content, users and enrollments.
- **CMS (Content Management System):** Creation and management of course content.
- **XBlock:** Extensible module for implementing various learning components such as videos, quizzes, discussions and more.
- **LTI (Learning Tools Interoperability):** Enables the integration of external tools and resources.
- **Open edX Studio:** An authoring tool for course creation and management.
- **Analysis tools:** Provides data analysis and reports on learning performance.

Architecture:

- **Microservices architecture:** edX uses a microservices architecture to separate and independently scale different functions and modules.

- **APIs:** Extensive RESTful APIs to integrate and extend the platform.
- **Scalability:** Use of cloud services to scale and manage user loads.



Tools & Clients
Mobile apps (android, iOS)
UX Toolkit & Pattern Library
API Manager
Configuration
XBlocks (plugins)
Documentation (readthedocs)
Test suites (bok-choy, etc)
The edx-platform codebase
LMS (Django)
Studio (Django)
XBlock Runtime
JSInput
Assessments
OLX import/export
Event receiver
CodeJail
OpenID Connect + 3rd Party Auth
Enrollments
User dashboard
Login & Registration
Independently deployed applications
Programs
Catalog
XQueue
Insights & Analytics
Credentials
Otto (cart, checkout)
Forums ("comments")
Persistence systems
Amazon S3 (videos, events)
Memcache (sessions, cache)
Elasticsearch
MongoDB (courses, forums)
MySQL (user data, insights)
RabbitMQ & Celery workers
Hadoop & Luigi (data pipeline)

Main components:

- edx/edx-platform repo contains the code for the edX platform.
- edx/edx-analytics-dashboard repo contains the code for edX Insights.
- edx/configuration repo contains scripts to set up and operate the edX platform.

6. edX Kubernetes Deployment

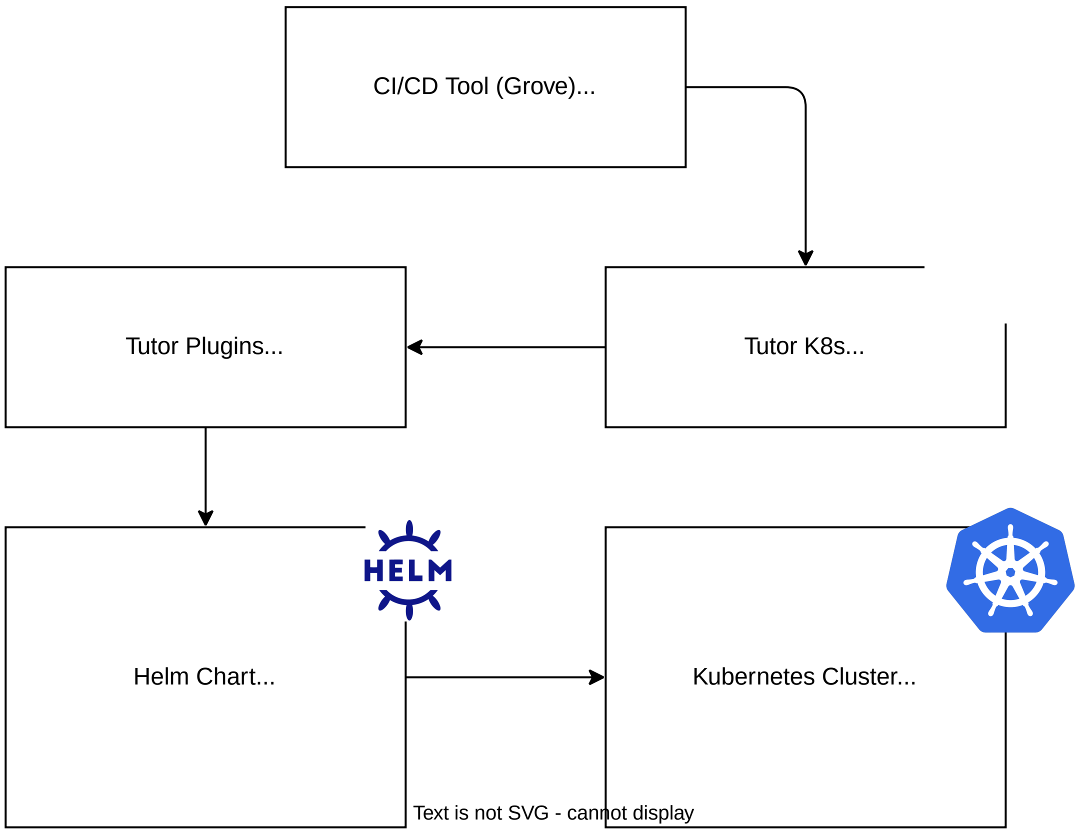
6.1. Kubernetes Cluster Setup for edX

This document describes the steps to set up a Kubernetes cluster for hosting the edX platform. Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. The repository with all informations and the helm charts is available at openedx/openedx-k8s-harmony.

In particular, this project aims to provide the following benefits to Open edX operators:

- Ease of use and rapid deployment: This project aims to provide an Open edX hosting environment that just works out of the box, that can be easily upgraded, and that follows best practices for monitoring, security, etc.
- Lower costs by sharing resources where it makes sense. For example, by default Tutor's k8s feature will deploy a separate load balancer and ingress controller for each Open edX instance, instead of a shared ingress controller for all the instances in the cluster. Likewise for MySQL, MongoDB, ElasticSearch, and other resources. By using shared resources by default, costs can be dramatically reduced and operational monitoring and maintenance is greatly simplified.
    - For setups with many small instances, this shared approach provides a huge cost savings with virtually no decrease in performance.

    - For larger instances on the cluster that need dedicated resources, they can easily be configured to do so.
- Scalable hosting for instances of any size. This means for example that the default configuration includes autoscaling of LMS pods to handle increased traffic.
- Flexibility: this project aims to be "batteries included" and to support setting up all the resources that you need, with useful default configurations, but it is carefully designed so that operators can configure, replace, or disable any components as needed. [7]



6.1.1. Setup Kubernetes Cluster

For setting up a Kubernetes cluster, you can use any cloud provider like AWS, GCP, Azure, or a self-hosted solution like Ubuntu with MicroK8s. The following steps are for setting up a Kubernetes cluster on Ubuntu with MicroK8s.

6.2. Setup Kubernetes

In this guide we are using microk8s to setup a Kubernetes cluster. To setup a Micro K8S Cluster on you need ubuntu maschines with static IP addresses, a working internet connection and optional for metallb "loadbalancer" an extra IP range for the services.

The network configuration for ubuntu 20.04+ is done with netplan. Config in /etc/netplan/00-installer-config.yaml and apply with sudo netplan apply

Example Configuration:

```
# This is the network config written by 'subiquity'
```

```
network:
  ethernets:
    ens18:
      addresses:
      - 141.57.9.180/22
      nameservers:
        addresses:
        - 141.57.1.94
        search:
        - imn.htwk-leipzig.de
      routes:
      - to: default
        via: 141.57.11.253
  version: 2
```

Name	IP	SSH Port	Proxmox Node	Config
k8s-nfs	141.57.9.171	23505	yprox	4 Core + 4 GB RAM
rancher	141.57.9.170	23505	xprox	4 Core + 4 GB RAM
k8s-node1	141.57.9.180	23505	yprox	4 Core + 8 GB RAM
k8s-node2	141.57.9.181	23505	zprox	4 Core + 8 GB RAM
k8s-node3	141.57.9.182	23505	yprox	8 Core + 32 GB RAM

We installed Ubuntu 22.04 on the nodes and configured the network with netplan. The current snap version for microk8s is 1.28.0 wich is supported by the current rancher release.

```
snap install microk8s --classic --channel=1.28/stable
```

Join the groups and rights so without root

```
sudo usermod -a -G microk8s $USER

sudo mkdir -p ~/.kube

sudo chown -f -R $USER ~/.kube

newgrp microk8s # to reload shell groups without close and open
```

Check the status while Kubernetes starts

```
microk8s status --wait-ready
```

Turn on the services you want

```
microk8s enable dns
```

```
microk8s enable registry
```

```
microk8s enable rbac
```

```
microk8s enable ingress
```

Start using Kubernetes

```
microk8s kubectl get all --all-namespaces
```

Um weitere Nodes mit microk8s zu verbinden auf dem main node:

```
microk8s add-node
```

Example output:

```
From the node you wish to join to this cluster, run the following:

microk8s join 192.168.122.210:25000/b346782cc8956830924c04f2cf1b1745/dadf654db615



Use the '--worker' flag to join a node as a worker not running the control plane, eg:

microk8s join 192.168.122.210:25000/b346782cc8956830924c04f2cf1b1745/dadf654db615
--worker



If the node you are adding is not reachable through the default interface you can use
one of the following:

microk8s join 192.168.122.210:25000/b346782cc8956830924c04f2cf1b1745/dadf654db615

microk8s join 192.168.123.1:25000/b346782cc8956830924c04f2cf1b1745/dadf654db615

microk8s join 172.17.0.1:25000/b346782cc8956830924c04f2cf1b1745/dadf654db615
```

Alias for easy setup copy paste commands:

```
alias helm="microk8s helm"
```

```
alias kubectl="microk8s kubectl"
```

If cattle-cluster-agent in cattle-system reboots multiple times and has dns resolving problems change the config.

```
kubectl edit deployment cattle-cluster-agent -n cattle-system
...
dnsPolicy: Default
```

6.2.1. Deploy Open edX on Kubernetes

Caution: This project is still in development and documentation is being updated. Please refer to the repository for the latest information. openedx/openedx-k8s-harmony/README.md

6.3. Step 1: Use Helm to provision a kubernetes cluster using this chart

6.3.1. Option 1a: Setting up Harmony Chart on a cloud-hosted Kubernetes Cluster (recommended)

For this recommended approach, you need to have a Kubernetes cluster in the cloud **with at least 12GB of usable memory** (that's enough to test 2 Open edX instances).

1. Make sure you can access the cluster from your machine: run `kubectl cluster-info` and make sure it displays some information about the cluster (e.g. two URLs).
2. Copy `values-example.yaml` to a new `values.yaml` file and edit it to put in your email address and customize other settings. The email address is required for Lets Encrypt to issue HTTPS certificates. It is not shared with anyone else. For a full configuration reference, see the `charts/harmony-chart/values.yaml` file.
3. Install [Helm](https://helm.sh/) if you don't have it already.
4. Add the Harmony Helm repository:

   ~~~
   ```shell
   helm repo add openedx-harmony https://openedx.github.io/openedx-k8s-harmony
   helm repo update
   ```
   ~~~

5. Install the cert-manager CRDs if using cert-manager:

   ~~~
   ```shell
   ~~~

```
kubectl apply -f https://github.com/cert-manager/cert-
manager/releases/download/v1.10.1/cert-manager.crds.yaml --namespace=harmony
```
```

```
   You can check the version of cert-manager that is going to be installed by the
chart by checking the corresponding
   line in the `charts/harmony-chart/Chart.yaml` file.
6. Install the Harmony chart by running:
```

```
```shell
helm install harmony --namespace harmony --create-namespace -f values.yaml openedx-
harmony/harmony-chart
```
```

Note: in the future, if you apply changes to `values.yaml`, please run this command to update the deployment of the chart:

```
helm upgrade harmony --namespace harmony -f values.yaml openedx-harmony/harmony-chart
```

6.3.2. Option 1b: Setting up Harmony Chart locally on Minikube

Note: if possible, it's preferred to use a cloud-hosted cluster instead (see previous section). But if you don't have a cluster available in the cloud, you can use minikube to try this out locally. The minikube version does not support HTTPS and is more complicated due to the need to use tunnelling.

1. First, [install `minikube`](https://minikube.sigs.k8s.io/docs/start/) if you don't have it already.
2. Run `minikube start` (you can also use `minikube dashboard` to access the Kubernetes dashboard).
3. Add the Helm repository and install the Harmony chart using the `values-minikube.yaml` file as configuration:

```
```shell
helm repo add openedx-harmony https://openedx.github.io/openedx-k8s-harmony
helm repo update
helm install harmony --namespace harmony --create-namespace -f values-minikube.yaml
openedx-harmony/harmony-chart
```
```

4. Run `minikube tunnel` (you may need to enter a password), and then you should be able to access the cluster (see "External IP" below). If this approach is not working, an alternative is to run\ `minikube service harmony-ingress-nginx-controller -n harmony`\ and then go to the URL it says, e.g. `http://127.0.0.1:52806` plus `/cluster-echo-test` (e.g. `http://127.0.0.1:52806/cluster-echo-test`)
```

5. In this case, skip step 2 ("Get the external IP") and use `127.0.0.1` as the external IP. You will need to remember to include the port numbers shown above when accessing the instances.

## 6.4. Step 2: Get the external IP

The [ingress NGINX Controller](https://kubernetes.github.io/ingress-nginx/) is used to automatically set up an HTTPS reverse proxy for each Open edX instance as it gets deployed onto the cluster. There is just one load balancer with a single external IP for all the instances on the cluster. To get its IP, use:

```
kubectl get svc -n harmony harmony-ingress-nginx-controller
```

To test that your load balancer is working, go to `http://<the external ip>/cluster-echo-test` . You may need to ignore the HTTPS warnings, but then you should see a response with some basic JSON output.

7. Kubernetes vs. Nomad

Kubernetes and Nomad are both platforms for the management and orchestration of containers, but the underlying concepts and architectures differ. To compare the two systems, especially with regard to terms such as deployment, job, pod, and their implementation in both systems, it is important to understand the respective architecture and concepts in detail.

7.1. Basic architecture

- **Kubernetes** consists of a master node (control plane) and several worker nodes. The master node is responsible for controlling, scheduling and managing the cluster resources. The worker nodes execute the containers (pods).
- **Nomad** has a centralized architecture in which a Nomad cluster consists of servers and clients. The servers manage the state and scheduling of jobs, while the clients actually run the jobs. Nomad is slightly more flexible as it can orchestrate not only containers but also other types of workloads (e.g. non-container applications, batch jobs).

7.2. Comparison of the most important concepts

Konzept	Kubernetes	Nomad
Pod	A pod is the smallest deployment unit in Kubernetes and can contain one or more containers. All containers in a pod share network and storage.	Nomad does not have an exact equivalent to pods. A Nomad job can contain multiple "tasks" that can work closely together like containers in a pod.

Konzept	Kubernetes	Nomad
Deployment	A deployment in Kubernetes describes how many instances of a pod are to be provided and how they are updated. It ensures self-healing (replications).	In Nomad, this is referred to as a "job", whereby a job can also have multiple instances of tasks. Updates and rollouts can be controlled by update strategies in the job.
Job	Jobs in Kubernetes (e.g. `CronJob` or `Job`) are intended for one-time or recurring tasks where pods should only run for a limited time.	A job in Nomad is the central construct that comprises a collection of tasks or services. It describes the entire deployment process, which can be similar to a deployment in Kubernetes. Nomad jobs can also include batch-like processes.
Service	Kubernetes uses service resources to abstract network access to pods and provide load balancing.	Nomad uses "Service Discovery" to register services and provide load balancing, often in conjunction with Consul.
Namespace	A Kubernetes namespace is a logical partitioning within a cluster used to organize resources.	Nomad also supports namespaces, but to separate different applications or teams in a cluster.
Scheduler	The Kubernetes scheduler assigns pods to available nodes based on resource requirements and settings.	Nomad also uses a central scheduler that assigns jobs to clients based on resource requirements and scheduling strategies.
StatefulSet	Kubernetes uses StatefulSets for stateful applications that require individual, persistent storage and ordered starts and stops.	Nomad supports stateful applications through the use of volumes and persistent data. However, there is no direct equivalent to Kubernetes' StatefulSet.

7.3. Correspondences for a mapping from Nomad to Kubernetes

The following concepts are most relevant for mapping Nomad to Kubernetes:

- **Nomad "Job" → Kubernetes "Deployment" or "Job"**
- A Nomad job that runs continuously corresponds to a **Kubernetes deployment**, while a batch-type Nomad job can be better assigned to a **Kubernetes job** or **CronJob**.
- If a Nomad job contains multiple tasks, it is possible to implement it using a **Pod** in Kubernetes, where each task is a separate container in the pod.
- **Nomad "Task" → Kubernetes "Container"**
- A task in a Nomad job is comparable to a container in a Kubernetes pod. Each task in Nomad has its own definition and can use Docker or a different runtime. In Kubernetes, multiple

containers can be defined within a pod.

- **Nomad update strategies → Kubernetes deployments**
- Nomad jobs support update strategies such as rolling updates. These can be implemented in Kubernetes directly with a deployment and its rollout strategies (e.g. `rollingUpdate` in Kubernetes).
- **Nomad "Task Group" → Kubernetes "Pods"**
- In Nomad, there are "Task Groups", which represent a group of tasks within a job that are deployed together. This is comparable to Kubernetes pods, which contain several containers.

7.4. Special deployment scenarios

- **Stateful applications**:
- In Kubernetes, StatefulSets are used to deploy stateful applications, while Nomad works with individual jobs and persistent volumes instead.
- **Service Mesh and Service Discovery**:
- Kubernetes uses services by default and optionally additional service mesh solutions like Istio to control network traffic. Nomad often uses Consul for service discovery and can be integrated into a service mesh.

7.5. Example: Nomad Job as Kubernetes Deployment

7.5.1. Nomad Job Definition:

```
job "example" {
  group "web" {
    count = 3
    task "nginx" {
      driver = "docker"
      config {
        image = "nginx:latest"
        port_map {
          http = 80
        }
      }
      resources {
        cpu    = 500
        memory = 256
      }
    }
  }
}
```

7.5.2. Corresponding Kubernetes Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "500m"
            memory: "256Mi"
```

7.6. Most important differences and similarities

- **Pods in Kubernetes** correspond to **Task Groups in Nomad**, with each task in Nomad corresponding to a container in Kubernetes.
- **Nomad Jobs** are more flexible and can include both batch processes and services, while in Kubernetes batch jobs are distinguished from deployments (for continuously running services).
- Kubernetes offers extensive **network and service options** (e.g. Services, Ingress) as well as special features such as **StatefulSets** and **DaemonSets** for specific use cases.

Rewriting from Nomad to Kubernetes seems a very difficult task considering how many little differences come up when comparing the two systems. However, on a basic level the concepts are similar and can be mapped to each other with some effort.

8. Dockerized Codeocean

One problem in the use of CodeOcean is the deployment. The installation instructions for CodeOcean are not comprehensive and only feature a development branch install, with a lot of installation steps to get the software working. They include Nomad as container orchestration instead of Kubernetes, which in itself doesn't run in Docker and therefore needs to be installed on the host system. Everything else, which is Codeocean with it's database and Poseidon, an interface between Codeocean and Nomad, was rewritten to run in Docker to simplify the deployment process. How these containers are built and deployed is described in the following sections.

9. Deployment

To start up the HTWK MOOC platform there are a few steps to follow, which will be layed out in the following sections. As container orchestration we need Nomad first, as it has to be installed on the host system. Once this precondition is met, we can start with the deployment using Docker compose.

For a quick installation, all necessary instructions are printed in bold. The text gives a detailed explanation of the steps.

9.1. Installation of Nomad

To install Nomad there should be a guide on the HashiCorp website: Nomad Installation Instructions. After a successful installation (for example using a package repository), Nomad needs to be running on localhost using port 4646. Additionally, when enabled as systemd service it should be started automatically on system boot. Both should be set up by default.

To check if Nomad is running correctly, the following command can be used and should return the hostname of the instance:

```
curl localhost:4646/v1/status/leader
```

Later on, this could be scaled to a cluster of multiple instances to keep the platform running on a larger scale.

The CodeOcean local setup guide advices to turn on Memory Oversubscription for Nomad now and with every subsequent restart of nomad with the following command:

```
curl -X POST -d '{"SchedulerAlgorithm": "spread", "MemoryOversubscriptionEnabled":
true}' http://localhost:4646/v1/operator/scheduler/configuration
```

This command is automatically executed when the Docker containers are started, so it is not necessary if the platform is started with Docker Compose.

9.2. Pulling the CodeContender repository

The repository for the HTWK MOOC platform is hosted on the GitLab instance of the faculty DIT. It consists of submodules for Poseidon and Codeocean, which have to be initialized after cloning the repository.

The repository can be cloned with the following commands:

```
git clone https://gitlab.dit.htwk-leipzig.de/htwkmooc/codecontender.git
cd codecontender
git checkout submodules
git submodule update --init
```

Because the HPI repositories contain additional submodules, that are for internal use only and not necessary for the HTWK platform, the init command does not need to be recursive.

9.3. Deployment with Docker Compose

To build and start the platform, Docker Compose is used. Default username and password for the admin can be changed in the `codeocean/docker-compose.yml` file. The default values are `support@htwkalender.de` and `htwkalender`. **The following command downloads the necessary images, builds the containers and starts them:**

```
docker-compose up --build
```

All data used by CodeOcean is stored in a Postgres database which is mounted

This doesn't work yet, we have to manually go into the `poseidon` *folder and copy* `configuration.example.yaml` *to* `configuration.yaml` *and adjust the following:*

- `server.address` to `0.0.0.0`
- `nomad.address` to `host.docker.internal`

For embedding and Usage of CodeOcean as LTI Integration in edX you have to configure the Content Security Policy in the `codeocean/config/initializers/content_security_policy.rb` file. The following lines have to be added:

```
policy.frame_ancestors      :self, 'https://*.htwk-leipzig.de'
```

You can define the allowed domains for embedding CodeOcean in other websites. The `:self` option allows embedding in the same domain. The `*.htwk-leipzig.de` option allows embedding in all subdomains of htwk-leipzig.de. This step has to be done before building the Docker containers.

The rest seems to be left unchanged…

9.3.1. Checking access to CodeOcean

After starting up the Docker containers, the platform should be running on port 443 of the host system. This can be checked by navigating to https://localhost in a web browser on the host machine.

9.4. Installing openEdx

The openEdx platform itself is not containerized, but there is a containerized distribution called Tutor, which can be deployed to a Kubernetes. Following the short installation process for a local deployment. Kubernetes instructions are following in the next section.

```
pip install tutor[full]
tutor local launch
```

After inputting the necessary information, all services should be running and the platform should be accessible on localhost. This can be checked by navigating to http://[studio.]local.edly.io on the host. This only works, when the port mapping for the CodeOcean container is changed. The domain points to localhost and is interpreted by a reverse proxy. Problem is, that logging to CodeOcean doesn't work at a different port, as 443 is hardcoded in the frontend.

10. CodeOcean Docker Images

To work with codeocean and execute programming and automatic grading tasks custom docker images are used that serve file copy/paste and logging functionality customized for codeocean. The images are built from the Dockerfiles in their example https://github.com/openHPI/dockerfiles repository.

The following images are used by CodeOcean:

Image	Version
co_execenv_java	8
co_execenv_julia	1.8/1.10
co_execenv_node	0.12
co_execenv_python	3.4-rpi-web; 3.4; 3.7-ml; 3.8
co_execenv_r	4
co_execenv_ruby	2.5
docker_exec_phusion	
ubuntu-base	Old ubuntu images no longer in use
ubuntu-coffee	
ubuntu-cpp	
ubuntu-exec	
ubuntu-jruby	
ubuntu-python	
ubuntu-sinatra	
ubuntu-sqlite	

Supported images

- docker_exec_phusion: Base image for all execution environments
- co_execenv_<langauge>: An image used by CodeOcean for the specific programming language and version

All supported images are built for the following architectures:

- amd64
- arm64

To create own images, the Dockerfiles can be used as a template. The repository contains a `Dockerfile` for each supported language and version. The Dockerfiles are based on the `docker_exec_phusion` image and install the necessary dependencies for the respective language. The Dockerfiles also contain the necessary configuration for the CodeOcean environment.

For example the `Dockerfile` for the Java 17 image looks like this:

```
FROM openhpi/docker_exec_phusion
MAINTAINER openHPI Team <openhpi-info@hpi.de>
LABEL description='This image provides a Java 17.0 environment with JUnit 5 support.'
LABEL run_command='make run'
LABEL test_command='make test CLASS_NAME="%{class_name}" FILENAME="%{filename}"'

# Get target architecture from Docker buildx
ARG TARGETARCH

# Install latest OpenJDK version 17.0
RUN install_clean openjdk-17-jdk make

# Add JUnit 5 Platform
ARG JUNIT_VERSION=1.11.0
RUN mkdir -p /usr/java/lib
RUN curl -fsSL -O --output-dir /usr/java/lib
https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-
standalone/$JUNIT_VERSION/junit-platform-console-standalone-$JUNIT_VERSION.jar

# Adjust Path variables
ENV JAVA_HOME /usr/lib/jvm/java-17-openjdk-$TARGETARCH
ENV JUNIT /usr/java/lib/junit-platform-console-standalone-$JUNIT_VERSION.jar
ENV CLASSPATH .:$JUNIT
```

11. Content Creation and Linking

Once CodeOcean and Edly are setup, there are certain configuration steps to do, to provide an own course on this platform. The following steps show common steps used by content creators.

11.1. Creating a course in openEdx

A new course can be created on the edit page: `https://hostname/course-authoring`. Course creation needs a naming scheme for course, organization, course number and course run (for example "Python Introduction", "HTWK", "C123.1", "WiSe24").

Before it is possible to link external LTI tools, this has to be enabled for this specific course. The openEdx platform supports LTI 1.1 up to and including LTI 1.3, extended by deep linking, grading and role provisioning services. To do this, the "lti_consumer" entry has to be added to "Advanced Module List" in the Course Settings/Advanced Settings.

A single course consists out of a strict hierarchy of section, subsection and units. There is a "New Section" button in the top right of the course edit page. To create subsections and units, the structure level above must first be expanded using the caret symbol on the left. In the edit page for a single unit, an external grader can be linked using an "Advanced" module and the "LTI Consumer" entry.

11.2. Creating a CodeOcean consumer for edX

Go to Administation → Integrations → Consumer in CodeOcean and add a new entry to get individual secret and key for the LTI integration.

These keys are required for OAuth 1 authentication, which is used by LTI 1.1. The new OAuth 2 standard is not yet supported by CodeOcean, but openEdx supports both standards. In the advanced course settings of openEdx, **go to "Lti passports" and add a new passport with key and secret in the following format:** `unique_id: key:secret` as an entry in a JSON string list. An example configuration looks like this:

```
[
 . "htwkcodeocean:bbebc02185e8e91dedaef54246779516:fc819bdfa4f216ffa525efa87af5579a"
]
```

The id can be chosen freely, but it has to be unique inside the edX platform.

11.3. Linking a CodeOcean exercise

After adding the Lti passport with the required secrets of the Codeocean LTI provider, new exercises can be linked. To get the link, open up an exercise in CodeOcean (Administrator → Exercises → Exercises) by clicking on it’s title. The LTI embedding parameters are shown like this: `locale=en&token=1234abcd`. We are only interested in the last part, the token.

The newly added "LTI Consumer" module has to be configured. In the edit page of the unit, the following settings must be applied:

NOTE:the "LTI Version" needs to be set to "LTI 1.1/1.2 (Default)" and the "LTI ID" has to match the unique_id of the passport, in the aforementioned example "htwkcodeocean".

Field	Value	Comment
LTI Version	LTI 1.1/1.2 (Default)	LTI 1.3 is not yet supported by CodeOcean
LTI ID	`htwkcodeocean`	Has to be identical with the secrets in the LTI passport in advanced course sections
LTI URL	`https://codeocean.htwk-leipzig.de/lti/launch`	Base URL of the CodeOcean platform, with `/lti/launch` as the endpoint
Individuelle Parameter	`["token=1234abcd"]`	This redirects to the correct exercise
LTI Startziel	Inline (Default)	The exercise is shown directly in the course
Schaltflächentext	`To the exercise`	Not necessary when inline exercise is chosen
Erzielte Punkte	True	Necessary to get grading information back from CodeOcean
Weight	1.0	It says default would be 1.0, don't be fooled by this hint. If nothing is entered, the exercise will not be graded.

The rest can be left default. All other necessary LTI parameters are automatically added by the openEdx platform, i. e. to identify the user and the course.

In individual parameters there are a lot more optional parameters, for example to hide parts of the CodeOcean interface. Those can be found in the CodeOcean documentation under `docs/lti_parameters.md`.

For a clean look, the following options can be added:

```
[
    "token=1234abcd",
    "embed_options_hide_navbar=true"
    "embed_options_hide_sidebar=true"
]
```

This hides all unnecessary navigational components, additionally the comment section can be hidden with `disable_rfc=true`.

12. Sources

[1] "The CodeGrade Documentation — CodeGrade QuietStorm.1 documentation." Accessed: Aug. 25, 2024. [Online]. Available: https://docs.codegra.de/.

[2] "check50 Documentation." Accessed: Aug. 25, 2024. [Online]. Available: https://cs50.readthedocs.io/projects/check50/en/latest/.

[3] "submit50." Accessed: Aug. 25, 2024. [Online]. Available: https://cs50.readthedocs.io/submit50/.

[4] M. Totschnig, C. Willems, and C. Meinel, "openHPI: Evolution of a MOOC Platform from LMS to SOA:" in *Proceedings of the 5th International Conference on Computer Supported Education*, Aachen,

Germany, 2013, pp. 593–598, doi: 10.5220/0004416905930598.

[5] T. Staubitz, H. Klement, R. Teusner, J. Renz, and C. Meinel, "CodeOcean - A versatile platform for practical programming excercises in online environments," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, Apr. 2016, pp. 314–323, doi: 10.1109/EDUCON.2016.7474573.

[6] S. Serth, T. Staubitz, R. Teusner, and C. Meinel, "CodeOcean and CodeHarbor: Auto-Grader and Code Repository," *Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice (SPLICE'21)*, 2021.

[7] "Tutor: the Docker-based Open edX distribution designed for peace of mind — Tutor documentation." Accessed: Sep. 20, 2024. [Online]. Available: https://docs.tutor.edly.io/.