Syllabus for the Course "Containerization and Kubernetes"

Volume: 7 ECTS credits (252 hours)
Semesters: 1–2 Form of final assessment: exam (in each semester)

Abstract of the course

The course is aimed at developing engineering thinking in the field of containerization, orchestration , and operation of distributed software systems.
The development of technologies such as Docker , Kubernetes , Helm (Kubernetes package manager), CI/CD (continuous integration and delivery), GitOps (operations via Git), Observability and Production Security is carried out within the logic of systemic problem statement, architecture design and ensuring the operational sustainability of the solution.
The discipline is implemented with the instrumental use of LLM (Large Language Model) as:

- reference source,
- educational intelligent assistant,
- architectural analysis tools,
- automated decision verification tool.

1. General characteristics of the discipline

The course is an instrumental engineering course that develops competencies in the field of Cloud Native (cloud native architecture) and DevOps engineering.
The course covers:

- understanding the architecture of container environments,
- distributed systems design,
- application lifecycle management,
- ensuring scalability and fault tolerance,
- Ensuring the security and operational maturity of the system.

The training is built from containerization of an individual service to the design of a production-ready cluster architecture.
LLM integrates as an intelligent tool for engineering reflection, configuration analysis and operational scenario modeling.

2. The place of the discipline in the structure of the educational program

The course is part of a professional module and is implemented in semesters 1–2 of the master's degree program.
It is the instrumental basis for the disciplines:

- information systems architecture,
- distributed systems,
- artificial intelligence,
- big data,
- project activities,

- operational engineering.

Mastering this discipline provides a practical basis for deploying microservice and ML systems in a cluster environment.

3. Objectives of mastering the discipline

The objectives of the discipline are:

- developing a systemic understanding of containerization and orchestration ;
- Mastering Docker and Kubernetes tools for engineering design;
- development of skills in the operation and sustainability of distributed systems;
- DevOps engineering competencies;
- development of skills in applying LLM to analyze, generate and verify configurations and architectural solutions.

4. Objectives of the discipline

- developing an understanding of process isolation mechanisms (namespaces , cgroups);
- mastering the principles of application containerization;
- designing microservice architecture;
- Kubernetes resources (Pod , Deployment , Service, StatefulSet);
- implementation of CI/CD (continuous integration and delivery);
- mastering Helm (Kubernetes package manager);
- development of incident diagnostic skills (Root Cause Analysis (root cause analysis);
- security (RBAC - role-based access control, Network Policies);
- Using LLMs in reference, tutorial, and controller modes.

5. Planned learning outcomes

The student must:

Know:

- principles of containerization;
- Docker and Kubernetes architecture ;
- scaling mechanisms (Horizontal Pod Autoscaler - horizontal autoscaler);
- CI/CD and GitOps basics ;
- principles of observability ;
- basic principles of production security (operational safety).

Be able to:

- develop correct Dockerfiles (image build files);
- containerize microservice applications;
- create and manage Kubernetes YAML configurations;
- diagnose failures (CrashLoopBackOff , etc.);
- design Helm chart (Helm chart);
- design a pipeline (assembly conveyor);
- use LLM to analyze architectures and validate solutions.

Be proficient in:

- skills in deploying applications to a cluster;
- operational analysis practices;
- methodology for architectural justification of decisions;
- instrumental use of LLM as an engineering assistant.

6. Methodological concept of the discipline

The training is based on the principle of an engineering trajectory:
Container → Cluster → Scaling → Automation → Operational Maturity
Each thematic block includes:

1. System analysis of the problem.
2. Architectural design of the solution.
3. Practical implementation.
4. Engineering reflection.
5. Audit of the solution using LLM.

LLM is used in three modes:
1. Directory
Explanation of terms, syntax, architectural mechanisms.
2. Educational mode
Interactive dialogue with individualized questions depending on the student's level of understanding.
3. Controller
Analysis of Dockerfiles , YAML configurations, architectural diagrams, and reports.
Formulation of recommendations for solution improvement.

6.1. Logic of the two-semester engineering trajectory

Stage 1. Containerization and Basic Orchestration (Semester 1)

Understanding is formed:

- Docker ;
- microservice architecture;
- Kubernetes architectures ;
- basic scaling;
- failure diagnostics.

The result of this stage is the ability to containerize the system and correctly deploy it in a Kubernetes cluster.

Stage 2. Production Kubernetes and DevOps (Semester 2)

The focus shifts to:

- operational maturity;
- safety;
- CI/CD;
- Helm ;

- GitOps ;
- observability ;
- engineering audit of the system.

The result of this stage is the ability to design and maintain a production-ready system.

7. Educational technologies

An AI-augmented learning model is used.
The educational process includes:

- lectures with architectural analysis;
- master classes demonstrating techniques;
- laboratory work;
- project activities;
- using LLM as an intellectual partner.

8. Differentiated assessment model and the principle of humane ambition

The assessment is based on:

- engineering correctness of the solution;
- depth of architectural analysis;
- independence;
- ability to justify decisions;
- correctness of application of LLM.

The following levels of development are distinguished:
Basic level
Correct implementation and understanding of the architecture.
Advanced level
Analysis of alternatives, identification of risks, argumentation of trade-offs.
Research level
Optimization, architectural improvement, critical evaluation of LLM recommendations.

9. Final certification

The exam includes:

- architectural case analysis;
- Kubernetes configuration diagnostics ;
- fixing Dockerfile or YAML;
- production incident analysis ;
- oral defense of decisions.

LLM is used for:

- generation of individual variants;
- primary automatic verification;
- preparing an analytical report for the teacher.

Curriculum schedule for the Course:

SDS - Self-Directed Study

1st SEMESTER

Module 1: Containerization and Basic Orchestration

Week	Topic	Lect.	Pract.	Lab.	SDS
1	**Cloud-Native Architecture. Containerization and Virtualization.** Lecture: Virtual Machine, Container , Linux namespaces, cgroups (Resource Control). Workshop: Docker Installation , Docker run , image layer analysis. SDS: comparative analysis of virtual machines and containers. LLM use: glossary lookup; training mode — isolation questions; controller — table verification.	2	2	0	4
2	**Docker Architecture. Dockerfile (image build file).** Lecture : Image , Layer , Container. Workshop: Dockerfile writing , error analysis. SDS: backend service containerization , layer optimization. LLM use: syntax reference; learning mode — layer logic explanation; controller — Dockerfile audit.	2	2	0	4
3	**Multi-stage build (Building). Image Security.** Lecture: builder stage , runtime stage , image scanning. Workshop: Rewriting Dockerfiles . SDS: Image minimization, vulnerability analysis. LLM: Reference; Training mode - Alternative strategies; Controller - Structure audit.	2	2	2	4
4	**Docker Compose (orchestration) Docker). Networking . Volume (storage volume).** Workshop: Backend + PostgreSQL . SDS: Multi-container application, Redis . LLM: Docker Compose reference. network ; controller docker-compose.yml .	2	2	2	4
5	**Microservices architecture architecture). 12-Factor App (12 principles of application).** Masterclass: decomposition of a monolith. CRS (Current Research Study): designing a 3-service system. LLM: training mode - questions about service boundaries; controller - schema audit.	2	2	0	4
6	**Introduction to Kubernetes. Cluster Architecture.** **Lecture** : Cluster, Control Plane , Worker Node (worker	2	2	0	4

Week	Topic	Lect.	Pract.	Lab.	SDS
	node), etcd . Masterclass: Minikube , kubectl . SDS: cluster diagram. LLM: component role reference; controller - diagram analysis.				
7	**Pod, ReplicaSet, Deployment. Rolling Update** Workshop: Application Deployment. SDS: 3 replicas, update strategy analysis. LLM: Training mode – availability analysis; YAML checker.	2	2	2	4
8	**Service. ClusterIP, NodePort, LoadBalancer loads)** Master class: service publication. SDS: implement external access. LLM: routing reference; configuration checker.	2	2	2	4
9	**ConfigMap (configuration) and Secret** Workshop: Injecting environment variables. SDS: Extract configuration. LLM: Educational mode - threat analysis; security controller.	2	2	0	4
10	**Persistent Volume (persistent volume). StorageClass (storage class)** Workshop: DB connection. SDS: node loss modeling. LLM use: learning mode — resilience analysis; YAML configuration checker.	2	2	2	4
11	**Horizontal Pod Autoscaler (horizontal autoscaler). Scaling** Masterclass: CPU autoscale . SDS: load testing. LLM: behavior prediction; report controller.	2	2	0	4
12	**Diagnostics and Debugging. CrashLoopBackOff (a Kubernetes error state)** Workshop: Incident Analysis. SDS: RCA (Root Cause Analysis) LLM: incident generation; report verification.	2	2	0	4
13	**Mini-project: Containerization of a microservice system** Workshop: architectural consultation. SDS: project implementation. LLM: architectural reviewer; YAML checker.	0	2	2	6
14	**Deploying a project to Kubernetes** Workshop: Configuration audit. SDS: Project refinement. LLM: Controller and error expert.	0	2	2	6
15	**Analysis of common mistakes. Exam preparation**	0	2	0	6
16	**Exam: 32 hours of preparation and 4 hours of assessment**	-	-	-	32

2nd SEMESTER

Module 2: Production Kubernetes and DevOps

Week	Topic	Lect.	Pract.	Lab.	SDS
1	**StatefulSet (state), DaemonSet (service pods), Job (task)** Workshop: Deploying a database via StatefulSet . SDS: Compare Deployment and StatefulSet . LLM: Reference; YAML controller.	2	2	0	2
2	**Ingress (router for incoming traffic). TLS (encryption)** Masterclass: Configuring HTTPS. SDS: Domain Routing. LLM: TLS Reference; Configuration Checker.	2	2	2	2
3	**RBAC (role-based access control)** Workshop: access restriction. SDS: multi-tenant model . LLM: rights audit; role controller.	2	2	0	2
4	**Network Policies. Zero-trust model** Masterclass: Namespace isolation . SDS: Attack modeling. LLM: Learning mode; security controller.	2	2	2	2
5	**Helm (Kubernetes package manager). Helm chart (chart). Values.yaml** Master class: creating a chart. SDS: configuration parameterization. LLM: template reference; structure checker.	2	2	2	2
6	**CI/CD (continuous integration and delivery). Pipeline (build pipeline)** Workshop: automated Docker build . SDS: pipeline design . LLM: architectural consultant; logic controller.	2	2	0	2
7	**GitOps (operations via Git). Infrastructure as Code** Masterclass: Autodeploy via Git . SDS: GitOps Strategy . LLM: Educational Mode; Declarativity Checker.	2	2	0	2
8	**Observability. Monitoring. Logging** Masterclass: Prometheus . SDS: Load Graph Analysis. LLM: Metrics Analysis; Report Controller.	2	2	2	2
9	**Distributed Tracing (distributed tracing). Latency** Workshop: Bottleneck analysis. SDS: Latency report. LLM: training mode; analysis controller.	2	2	0	2
10	**Production Security (operational) security). Image scanning. Secrets Management (secret management)**	2	2	0	2

Week	Topic	Lect.	Pract.	Lab.	SDS
	Master class: Image audit. SDS: security review . LLM: Security audit; Recommendation checker.				
11	**Final project: architectural design of a production system**	0	2	2	3
12	**Helm + CI/CD + HPA (autoscaler) implementation .**	0	2	2	3
13	**Engineering audit of the system** LLM: architectural opponent and controller.	0	2	2	3
14	**Preparing for Defense. Operational Risk Analysis**	0	1	1	2
15	**Exam: 32 hours of preparation and 4 hours of assessment**	-	-	-	32