

## **CLOUD DEPLOYMENT STRATEGIES AND ARCHITECTURAL INNOVATIONS IN LEARNING MANAGEMENT SYSTEMS**

*Moiseienko A., Kuznetsova Yu.*

*The article provides a comprehensive analysis of modern technologies for deploying learning management systems (LMS) in cloud environments, with special attention to containerization, microservice architecture, and multi-tenant SaaS models. Practical examples of implementing Moodle, Canvas, Blackboard Learn, and Open edX in leading cloud platforms (AWS, Azure, GCP) are considered, including technical aspects of the infrastructure: scaling, ensuring fault tolerance, using Kubernetes, Docker, Terraform, CI/CD pipelines, and managed services. A comparative analysis of on-premise, SaaS, and containerized deployment models is conducted based on key criteria (flexibility, reliability, maintenance cost, scalability). Practical recommendations are offered for choosing the optimal LMS architecture depending on the implementation context and resources of the educational institution. The results of the study demonstrate the advantages of cloud technologies for developing educational web systems and increasing their availability, stability, and efficiency in the context of the digitalization of education.*

### **Introduction**

Over the past decade, the digital transformation of education has developed rapidly, which was made possible by the spread of modern information and communication technologies. One of the key innovations in this area was the introduction of learning management systems (LMS), which provide centralized organization, implementation, and monitoring of the educational process in an online environment [1, 2]. However, traditional LMSs, which operate on local servers of educational institutions, have a number of limitations related to scalability, reliability, maintenance costs, and the implementation of new functionalities [3, 4].

The search for solutions to overcome these problems has led to the active introduction of cloud technologies in the field of education. Cloud computing, in particular, the SaaS, PaaS, and IaaS models, open up new opportunities for flexible, secure, and efficient deployment of SNS in the environment of public or hybrid clouds [13]. This topic has become particularly relevant in the context of the COVID-19 pandemic and martial law in Ukraine, when the need for continuous access to educational resources has become critical, regardless of the geographical location or physical condition of the infrastructure of the educational institution [2, 4].

However, the practical implementation of a cloud-based CMS is accompanied by a number of challenges, from issues of data security and confidentiality to the difficulties of scaling and integration with other services. A particularly important problem is the optimal distribution of CMS components in a cloud environment, taking into account limited resources, performance requirements, fault tolerance, and cost [11, 12].

*The object of research* is modern learning management systems in the context of cloud technologies.

*Subject of study* – methods, models, and tools for implementing and optimizing cloud deployment of learning management systems, in particular, using microservice architecture, containerization, and formal optimization models.

*The purpose of the study* is the development, analysis, and generalization of effective approaches to implementing cloud-based learning systems with an emphasis on automated planning and resource optimization, which allows for flexible scaling and reliable operation of educational web systems.

To achieve the goal, it is necessary to solve the following tasks:

- systematize current approaches to cloud-based deployment of SNS;
- identify shortcomings and limitations of existing solutions;
- explore the potential of microservice and container architectures;
- to formulate a mathematical model of the problem of optimal placement of SNS in the cloud;
- to propose directions for further improvement of such systems.

*Practical significance.* The research results can be used by educational institutions, IT companies providing EdTech solutions, and researchers to improve the efficiency of cloud-based learning systems implementation. The proposed models contribute to reducing costs, increasing resilience to failures, and improving the user experience in online learning.

## **1. Overview of cloud deployment practices learning management systems**

To analyze modern approaches to implementing and deploying learning management systems in cloud environments, it is necessary to consider practical examples of deploying popular LMS platforms in the infrastructures of leading cloud providers (AWS, Azure, GCP), as well as the architecture of multi-tenant SaaS solutions (multi-tenant) and containerized microservice deployments [5–7, 13].

As a result, the technical aspects of the architecture, deployment models, and the use of tools such as Kubernetes, Docker, Terraform, Helm, and DevOps pipelines for automation will be described in detail. Examples of the implementation of SNS with scalability, reliability, and performance in the cloud will be given, a comparative analysis of the models (on-premise vs. SaaS vs. containerized) will be performed in the form of a table, and conclusions will be formulated regarding the practical advantages and disadvantages of each approach [14].

### ***1.1 Moodle in cloud environments AWS, Azure, GCP***

Moodle on AWS is deployed using EC2 (web servers), Amazon RDS/Aurora (DB), ElastiCache (Redis), EFS (files), and ALB + CloudFront (load balancing and CDN). Auto Scaling groups provide dynamic scaling, and managed services increase reliability. Integration with AWS CodePipeline and CodeDeploy allows for CI/CD for continuous updates of Moodle [5].

Moodle deployment on Azure is based on ARM templates. The basic configuration uses a single VM, MySQL, and NFS. For larger workloads, scenarios with multiple web hosts, Redis cache, and GlusterFS for storage are used. Azure Database for MySQL simplifies administration, but compared to AWS, more components require manual management (VMs, file systems) [6].

Moodle on Google Cloud is implemented as a cloud-native solution: containerized in Docker, orchestrated in GKE (Kubernetes). Cloud SQL (DB), Filestore (files), Cloud Memorystore (Redis), Cloud Load Balancer, CDN and WAF (Cloud Armor) are used. The solution is fully managed and automated: no manual work with VMs, with CI/CD support via Cloud Build and Artifact Registry [7].

It can be concluded that all three platforms support scalable and reliable Moodle deployment. AWS provides the most out-of-the-box managed components. Azure is suitable for flexible scenarios, but requires more manual configuration. GCP offers the most modern cloud-native approach with deep containerization.

### ***1.2 Architecture of the Canvas LMS multi-user SaaS solution***

Canvas LMS [8] is a cloud-native platform with a multi-tenant SaaS architecture, where each client (university or school) is a separate «tenant» in a shared environment. Isolation is achieved by sharding the database: each client has its own shard, although all users work through a common web application.

The Canvas infrastructure runs on AWS and is built on Ruby on Rails for scalability. Inbound traffic passes through AWS WAF and is distributed to web

servers by Elastic Load Balancer, data is cached via Redis/Memcached. Data is stored in a DBMS cluster (Aurora or PostgreSQL), and exchange rate information is stored in S3 storage with delivery via CDN (CloudFront).

For background tasks (reports, conversions) queue servers are used that scale automatically. The platform is deployed in multiple AWS regions to meet data storage requirements. Administration, updates, scaling, and backups are handled by the provider (Instructure).

Canvas demonstrates a combination of multi-tenancy, horizontal scaling, and automated resource management that allows it to serve millions of users with a high level of reliability [8].

### ***1.3 Managed services and integration with DevOps practices in Blackboard Learn SaaS***

Serving over 100 million users, Blackboard Learn has evolved from an on-premise solution to a modern SaaS platform deployed in the AWS cloud. While historically the system was monolithic (Java), the SaaS version implements a multi-tenant architecture that allows educational institutions to operate in a single environment while maintaining data isolation [9].

The cloud infrastructure includes web hosts in multiple AWS availability zones, load balancers, caching (Redis/Memcached), and file storage based on Amazon S3 (in partnership with NetApp Cloud Volumes ONTAP). This allowed Blackboard to dynamically scale, including during the pandemic when the load increased by 3000%.

The databases have been migrated from Oracle/MSSQL to PostgreSQL on Amazon RDS, which has reduced costs, increased automation, and enabled the use of Read Replicas for analytics without compromising performance. The company is also reportedly using ElastiCache, CloudWatch, and AWS Lambda.

DevOps practices enable continuous integration and deployment (CI/CD). Blackboard implemented a fully automated release pipeline with containerization, infrastructure as code, and blue-green releases. Automated UI testing (powered by AWS Lambda) reduced testing time from days to seconds. With IaC, the company can scale resources in hours instead of weeks [9].

As a result, we can conclude that the platform is capable of serving millions of users with high SLAs, without downtime, with quick response to changes and updates «on the fly».

#### ***1.4 Cloud deployment of Open edX based on containerization and orchestration in Kubernetes***

Open edX is a scalable online learning platform that includes a number of interconnected services: LMS, CMS (Studio), MySQL, MongoDB, ElasticSearch, analytics, etc. Previously, its deployment required complex configuration via Ansible, but today the community has moved to containerization and orchestration in Kubernetes [10].

*Containerization of components.* Each Open edX component is packaged in a separate Docker container using the «one service, one container» principle. This simplifies deployment, provides isolation, scalability, and reduces dependencies between services. Persistent volumes are used for stateful components [10].

*Orchestration in Kubernetes.* In production environments, Open edX is deployed as a set of pods in a Kubernetes cluster. YAML specifications are created for the LMS, CMS, DB, and other services, specifying environment variables, recovery policies, and state retention (via PersistentVolumeClaim). For simplicity, an official Helm Chart is used to automate the deployment of the entire platform [10].

*Scalability and continuous operation.* Kubernetes provides horizontal and vertical auto-scaling. For example, during a large course, the number of LMS pods automatically increases. Rolling updates are also supported, which allows you to update the system without downtime. The EFK stack and probes (liveness/readiness) are used for monitoring and logging [10].

*Safety and isolation.* Containers provide service isolation. Kubernetes allows you to restrict network communication through Network Policies. This is critical for multi-tenant deployments, where different clients use separate instances on a shared cluster [10].

Therefore, the modernized approach to deploying Open edX significantly increases the flexibility, scalability, and reliability of the system. The use of Kubernetes and Helm ensures efficient management of a multi-component LMS and makes it suitable for high loads and dynamic usage scenarios.

## **2. Comparative analysis of cloud implementation models learning management systems**

To summarize, let's look at the key characteristics of different approaches to deploying learning management systems – from traditional local (on-premise) to modern multi-cloud. Table 1 compares three basic models [11, 14]:

1. Local/self-contained deployment – when the institution itself installs the LMS on its servers or rented VMs (monotenant architecture);

2. Cloud SaaS (multi-tenant) – when the LMS is provided as an online service by the provider and serves many clients in one shared environment;

3. Cloud containerized deployment – when the institution or provider deploys the LMS in a cloud infrastructure using containerization and microservices (can be either a separate instance for one institution or multi-instance on a shared cluster).

This classification covers most of the practices described above, i.e. Moodle traditionally implements the first and third models (standalone deployment on VM or modern containerized on GCP) [5–7], Canvas and Blackboard represent the second model (SaaS) [8, 9], and Open edX is used in all three variants (there are institutions that install Open edX locally; there are SaaS providers such as EdX/Open edX; there are containers in the cloud) [10]. The main comparison criteria: scalability, support requirements, financial model, customization capabilities, and reliability/fault tolerance are given in Table 1.

Also, it should be noted that each of the models has its own optimal use scenarios. On-premises deployment is appropriate where complete autonomy and control over data are critical (for example, in military or closed corporate networks). The SaaS model is best suited for a quick start and minimizing technical risks, which is especially important for small institutions with limited IT staff. Containerized cloud deployment, in turn, allows for a combination of scalability and control, which makes it advantageous for universities with large audiences or for national educational platforms.

Additionally, security and regulatory compliance (GDPR, FERPA, etc.) are important. In a SaaS model, much of the responsibility lies with the provider, while in on-premise and containerized systems, the institution independently defines access and security policies. This can be a decisive factor in choosing an architecture.

Equally important is the issue of integration with other services: on-premise solutions are often more difficult to integrate with modern analytical tools or video conferencing systems, while SaaS and Kubernetes-based models usually offer ready-made APIs and modules for interaction. Finally, one should consider the risk of «vendor lock-in» – dependence on a single cloud service provider, which can limit an institution’s flexibility in the future.

Table 1

### Comparison of SSN deployment models

Criterion	Local / self-deployment (on-premise or own VMs)	Cloud SaaS (multi-tenant model)	Containerized cloud deployment (cloud-native microservices)
1	2	3	4
<b>Scalability</b>	Limited by local resources; scaling requires purchasing and manually configuring new hardware or VMs. High peak loads are difficult to handle without redundancy.	Almost unlimited horizontal scalability due to provider resources; servers/capacity are added dynamically according to demand (auto-scaling in the cloud). Customers do not notice the increase in load - this is the provider's problem.	Flexible scalability thanks to Kubernetes and cloud services: the system can automatically scale individual components (pods) based on load metrics. It is easy to scale different services independently (for example, only LMS web servers or only DB by increasing resources or shards).
<b>Costs and payment model</b>	Large capital costs at the beginning: purchase of servers or long-term lease of VMs, purchase of software licenses. Ongoing costs for electricity, cooling, IT staff. Scaling requires additional investments.	OPEX model: predictable subscription or payment per user/year. Low initial costs (no need to buy equipment). The provider distributes infrastructure costs among many customers, which reduces TCO for each. Scaling means subscription growth, but does not require direct customer action.	Pay-as-you-go. No capital costs for servers, but DevOps team costs. Optimal resource utilization through auto-scaling can make maintenance cheaper (minimum resources during off-hours). However, the complexity of the platform may require experts, which is also worth it.
<b>Support and updates</b>	The responsibility lies entirely with the institution's IT department: they must independently apply patches, update the LMS to new versions, and monitor server security. Downtime is possible during updates.	All updates are performed by the SaaS provider, often continuously (Continuous Delivery) and with minimal disruption. The customer always receives the latest version without any effort. There is no need to have a large IT staff to support it – «LMS as a Service».	Automation greatly simplifies support: CI/CD pipelines can deploy new versions of containers themselves. Kubernetes allows rolling updates without downtime. But the responsibility for configuring updates lies with the team that implemented the system. Requires high competence of DevOps engineers to maintain the infrastructure.

Continuation of the table 1

1	2	3	4
<b>Customization and control</b>	Maximum customization: the institution has full control over the servers and software, can modify the LMS source code, install non-standard plugins, configurations, integrate with internal systems at a deep level. This is important for organizations with special needs (e.g., specific authentication).	Limited flexibility: A multi-user SaaS system typically does not allow for customizing the source code for a specific client. Customization is limited to configuration options and adding supported plugins. Deep integrations are only possible through the provided APIs. In return, the client gets stability and proven solutions, no «manual» intervention.	High flexibility: the system is deployed under the control of the customer's (or provider's) team, you can modify the container configuration, add your own services (e.g. additional analytical modules), choose component versions. A Kubernetes-oriented system easily integrates with other cloud services (e.g., you can add an external analytics system, Lambda server, etc.). Customization is almost as broad as on-premise, but requires skills.
<b>Reliability and scale</b>	Depends on redundancy in the local infrastructure. High reliability can be achieved, but it is expensive: server duplication, database clustering, backup power. For many institutions, a typical configuration is one application server and one database server, which are points of failure. Fault tolerance and disaster recovery are entirely the responsibility of the institution.	Deployed on a professional infrastructure with redundancy at all levels (multiple datacenters, database replication, daily backups, etc.). The provider guarantees an SLA (usually 99.9% and above uptime). The scale is global: users around the world receive acceptable performance through CDN and geo-distribution. On the other hand, a general SaaS failure (although unlikely) affects all customers at once.	Built on cloud capabilities, such a system can be very reliable with the right configuration: multi-AZ deployment, use of managed databases with automatic failover, regular backups, etc. Kubernetes itself provides service recovery in the event of container or node failure. Disaster recovery and reserves on the customer side: you need to think about, for example, a backup cluster or a backup strategy in the cloud. Scaling to very large loads is possible, but you need to follow the limits of the cloud provider and optimize the architecture (for example, sharding the database when growing, distributing components across microservices).

### **3. Analysis of cloud deployment models learning management systems**

The review showed that there is no clear universal model for deploying a cloud computing system – each approach has its strengths and weaknesses (summarized in Table 1), and the choice depends on the needs and capabilities of a particular organization. Here are some practical conclusions regarding the main models of cloud deployment.

#### *1. Traditional local deployment (on-premise/self-hosted)*

The main advantages include: full control over the system, data, and update process; the ability to deeply customize the LMS to your requirements [10]; and no dependence on external providers (relevant for closed networks or strict security policies).

Disadvantages: high costs for infrastructure and IT personnel – investments in servers, their maintenance, and backup are necessary; limited scalability – it is difficult to respond quickly to a sharp increase in the number of users or load (you need to purchase enough capacity in advance); risk of downtime and difficulty in ensuring high availability – the administrator is responsible for backup, fault tolerance, updates without stopping the system, etc. In modern conditions, an on-premise solution is justified only for small institutions with very specific requirements or where the use of the cloud is impossible for security reasons [11].

#### *2. Multi-user cloud SaaS solution*

The main advantages are minimal hassle for the customer – the deployed system is available immediately after signing the contract, all aspects of technical support (updates, patches, monitoring) are provided by the provider; high reliability and scalability «out of the box» – the provider provides the necessary resources, geographical distribution, and backup, which would be very expensive at individual cost; budget predictability – subscription payment that correlates with the number of users or other metrics, without capital expenditures.

Disadvantages: less flexibility and customization options – you mostly have to work within the framework of standard functionality, since the common platform does not allow you to change it for only one client (it is impossible to make changes to the LMS code if they are not provided for everyone); dependence on the provider – both technical (in the event of a service failure, the client will not fix anything himself, he must wait for a support response) and strategic (changes in prices, policies, or termination of product support by the provider directly affect the institution). Also, the issue of data security requires trust in the provider: educational

data is stored in the cloud, and the institution must be confident in its security and compliance with regulatory requirements (GDPR, FERPA, etc.).

In general, the SaaS model is optimal for most cases, especially when the priority is rapid implementation and reduced IT costs [5-9].

3. *Containerized cloud deployment (microservice architecture in IaaS/PaaS)* combines the advantages of the cloud and independent control

The customer (or the cloud operator providing the service) can flexibly configure the system using modern DevOps tools: automate deployments, scale components in real time, and optimally use resources.

Containerization allows you to isolate LMS services, which simplifies updates and ensures stability (the failure of one component will not «bring down» the entire system). Kubernetes orchestration brings self-healing to the system (automatic restart of crashed containers) and makes it easier to achieve high availability without manual intervention. In addition, the cloud-native approach provides portability: you can relatively easily move containers to another cloud platform or deploy a hybrid solution (part of the services run locally, part in the cloud) – this reduces the risk of being tied to a single vendor.

Disadvantages: significant complexity of implementation – experts in containerization, CI/CD setup, cluster security, etc., are required; the entry threshold for small teams is high. In fact, deploying a Kubernetes cluster with Open edX or Moodle is within the power of only a fairly mature team, while small organizations are easier to choose SaaS or use third-party integrators. It should also be noted that migration to microservices does not completely eliminate the problems of database scaling or transactional integrity – with a very high load, you will still have to implement database sharding or distribute services across multiple clusters (which requires additional efforts in architecture design). Therefore, the container model is most effective for medium and large implementations, where there are resources to support it and the need for flexibility/scale, and is less justified for small installations [11–14].

## **Conclusions**

Cloud technologies have opened up new horizons for learning management systems, enabling rapid scaling to any audience and high availability of educational platforms around the world.

The practices discussed in this section – from hosting Moodle on AWS/Azure/GCP to SaaS architectures of Canvas and Blackboard and containerization

of Open edX – demonstrate that the competent use of cloud services and DevOps approaches can significantly improve the effectiveness of an LMS.

In particular, the transition to a SaaS model allows you to focus on the educational process, leaving the technical aspects to the provider, while the microservices approach provides maximum flexibility and the potential for optimization to your needs. The choice of model depends on the context: for small institutions, SaaS is often optimal (minimum hassle and high reliability), for large universities or consortia with their own IT resources - containerized deployment can provide the desired control and economies of scale, and some specific scenarios may still require on-premise (for example, military academies or remote closed networks).

In general, the global trend is clearly moving towards the cloud: cloud implementation and deployment technologies for E-learning are becoming an integral part of the development of educational web systems, ensuring their readiness for the challenges of mass and continuous learning in the digital age.

Prospects for further work in this direction are associated with the development of adaptive and intelligent LMSs that will use artificial intelligence methods to personalize the learning process. An important vector will be the use of hybrid architectures that combine local resources and public clouds to ensure greater security and reliability. In addition, the implementation of unified interoperability standards is promising, which will simplify the integration of LMSs with external educational services.

## References

1. Al-Busaidi KA, Al-Shihi H. Key factors to learning management system implementation success: A case study of Sultan Qaboos University // *Education and Information Technologies*. – 2019. – Vol. 24, No. 2. – P. 1313–1335. – DOI: 10.1007/s10639-018-9831-y
2. Pappas IO, Pateli AG, Giannakos MN, Chrissikopoulos V. The interplay of online learning motivations and cloud-based LMS quality on students' satisfaction // *Computers & Education*. – 2019. – Vol. 132. – P. 75–92. – DOI: 10.1016/j.compedu.2019.01.004
3. Kumar A., Sharma R. Cloud computing for education: A modern approach to teaching and learning // *Journal of Cloud Computing: Advances, Systems and Applications*. – 2021. – Vol. 10, No. 15. – P. 1–17. – DOI: 10.1186/s13677-021-00234-6
4. Singh G., Hardaker G. Cloud-based learning management systems: A review on features and architecture // *Education and Information Technologies*. – 2020. - Vol. 25. – P. 5131–5154. – DOI: 10.1007/s10639-020-10234-7
5. Amazon Web Services. Deploying Moodle on AWS [Electronic resource]. – Access mode: <https://aws.amazon.com/quickstart/architecture/moodle/> (date of application: 08/23/2025).
6. Microsoft Azure. Deploy Moodle on Azure [Electronic resource]. – Access mode: <https://azuremarketplace.microsoft.com/en-us/marketplace/apps/bitnami.moodle> (date of application: 08/23/2025).

7. Google Cloud Platform. Moodle on GCP using GKE [Electronic resource]. – Access mode: <https://cloud.google.com/solutions/moodle> (date of application: 08/23/2025).
8. Instructure. Canvas LMS architecture overview [Electronic resource]. – Access mode: <https://www.instructure.com/canvas> (date of application: 08/23/2025).
9. Blackboard Inc. Blackboard Learn SaaS: Technical Architecture [Electronic resource]. – 2021. – Access mode: <https://help.blackboard.com/Learn/Administrator/SaaS/Deployment> (date of application: 08/23/2025).
10. Open edX. Deploying Open edX using Tutor and Kubernetes [Electronic resource]. – 2023. – Access mode: <https://docs.tutor.overhang.io/> (date of application: 08/23/2025).
11. Bernstein D., Ludvigson E., Sankar K., Diamond S., Morrow M. Blueprint for the Intercloud – Protocols and Formats for Cloud Computing Interoperability // Proc. of the 2009 4th Int. Conf. on Internet and Web Applications and Services. – 2009. – P. 328–336. – DOI: 10.1109/ICIW.2009.55
12. Zaharia M., Chowdhury M., Franklin MJ, Shenker S., Stoica I. Spark: Cluster computing with working sets // Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing. – 2010.
13. Mell P., Grance T. The NIST definition of cloud computing. – National Institute of Standards and Technology, US Department of Commerce. – 2011. – [Electronic resource]. – Access mode: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf> (date of application: 08/23/2025).
14. European Union. General Data Protection Regulation (GDPR) // Official Journal of the European Union. – 2016. – L119.
15. EDUCAUSE. 7 Things You Should Know About SaaS LMS [Electronic resource]. – 2020. – Access mode: <https://library.educause.edu/> (date of application: 08/23/2025).