UDC 004.4
DOI: 10.20535/SRIT.2308-8893.2025.2.02

# STREAMLINED MANAGEMENT OF PHYSICAL AND CLOUD INFRASTRUCTURE THROUGH A CENTRALIZED WEB INTERFACE

**I. BYZOV, S. YAKOVLEV**

**Abstract**. The article focuses on the development of a centralized management system for both physical and virtual servers via a web interface. This system enables server infrastructure administration through various tools. Key components of the system, integration with server management tools, the use of Paramiko for remote access via SSH, and the potential application of other technologies for managing cloud and physical servers through a web interface are discussed. The system enables centralized management of various server types through a web interface, supporting both basic and advanced administrative operations. Integration with Docker and support for cloud APIs provide convenient automation and simplify operations with both cloud and physical servers. As a result, the system serves as a versatile tool for DevOps engineers. The study highlights the relevance and importance of a centralized approach to server infrastructure management, thereby enhancing the efficiency and reliability of server operations in modern environments.

**Keywords:** centralized management, SSH, web interface, cloud services, automation, DevOps.

## INTRODUCTION

With the rapid growth of information technology and the increasing volume of data processed by server infrastructures, effective management of physical and virtual servers has become a critical necessity. Hybrid infrastructures, which combine physical servers and cloud computing resources, are widely adopted for their flexibility and scalability in handling data processing and storage. However, the expansion of these infrastructures introduces significant complexity into administrative tasks, demanding more advanced and streamlined management solutions.

Server administrators frequently encounter challenges in managing diverse server types—both physical and virtual—across multiple tools and protocols. This fragmentation leads to inefficiencies, as routine tasks such as updates, monitoring, configuration, and troubleshooting must be performed using disparate platforms and interfaces.

This study addresses these challenges by presenting a centralized management system for physical and virtual servers via a web interface. The proposed solution aims to automate server administration processes by integrating various modern tools and technologies. Specifically, it incorporates Docker for containerization, Paramiko for secure SSH-based remote access, and cloud APIs for managing virtual servers across different providers. This unified approach not only optimizes administrative workflows but also minimizes errors and improves overall efficiency.

The primary goal of this research is to develop a universal solution for automating the management of hybrid server infrastructures, including both physical servers and cloud instances. The proposed system offers a centralized web interface for managing diverse server environments, integrating technologies such as Docker, Terraform, Ansible, and cloud APIs. This integration simplifies routine administrative tasks, enhances productivity, and reduces complexity and error rates associated with fragmented tools.

The article is structured as follows. The next section provides a detailed review of existing tools and technologies for managing physical and virtual servers, with particular emphasis on solutions such as Docker, Paramiko, and cloud APIs. Section 3 formulates the main research objectives. Section 4 describes the architecture of the proposed centralized management system, highlighting its design and the integration of various server management tools within a unified web interface.Subsequent sections focus on the system's implementation, detailing the integration of Docker, Terraform, Ansible, and Paramiko into a cohesive solution. A comparison with existing management tools is also included.

Finally, the effectiveness of the developed system is evaluated through performance testing and automation analysis. The conclusion summarizes the findings and discusses potential avenues for further development and enhancement of the system.

## CURRENT STATE OF THE PROBLEM

Various scientific studies highlight the importance of automation and centralization in server infrastructure management. One of the primary trends in this field is the use of containerization to ensure efficient resource allocation and process isolation. Technologies such as Docker and Kubernetes are among the most prominent in this area. Docker facilitates the containerization of software applications, while Kubernetes orchestrates container operations, enabling the automation of deployment, scaling, and management processes.

In [1], the authors showed that Docker significantly simplifies infrastructure management through application isolation and streamlined deployment processes. This fact is confirmed by studies [2] emphasize Kubernetes' advantages, particularly in cloud environments. Kubernetes offers a high degree of automation, allowing efficient resource management and infrastructure scaling while also highlighting opportunities to enhance Kubernetes (K8s) for deployment platforms.

Another critical aspect of server management is the use of the SSH protocol for remote administration of physical servers. Many automation tools, such as Ansible, Chef, and libraries like Paramiko, leverage SSH to execute commands on remote servers. Among these, Paramiko has been identified as one of the most efficient and user-friendly tools. A study [3] showed that Paramiko reduces server configuration time by 4.14 times compared to Netmiko.

On the other hand, numerous researchers focus on managing cloud resources via APIs. For example, in [4], explores methods of integrating with cloud services such as AWS, Azure, and Google Cloud. These integrations enable automation in managing cloud instances, significantly reducing administrative overhead.

Most studies conclude that centralizing hybrid infrastructure management is an effective approach to reducing the complexity of administration and improving server management productivity. In [5] it is emphasized that unifying tools for cloud infrastructure management into a single interface greatly simplifies administrative processes, reduces error rates, and enhances overall productivity.

A review of the literature underscores the critical relevance of centralized management for hybrid infrastructures in modern IT systems. Existing research emphasizes the need for solutions that integrate essential management tools while modernizing algorithms to enable efficient centralized administration. These approaches simplify infrastructure management and enhance overall server operation efficiency, marking an important direction for further research and implementation in the DevOps domain.

## RESEARCH OBJECTIVES AND TASKS

Managing server infrastructures, both physical and cloud-based, has become increasingly complex and demanding in modern IT environments. Numerous tools exist to address specific tasks, such as containerization, remote server access, and cloud resource management. However, effective management of hybrid infrastructures requires a centralized system that integrates all these tools into a single interface, simplifying administration and automating routine tasks.

The primary objective of this research is to design the architecture and develop a web application that provides a unified interface for managing physical and cloud servers, integrating key tools and technologies.

To achieve this objective, the following tasks are defined:

- *Designing the architecture of a centralized management system.* Develop the architecture of a web application capable of integrating various tools for managing physical and cloud servers within a unified interface.
- *Implementing integration with Docker, SSH, and cloud services.* Enable interaction with Docker for containerization, utilize Paramiko for secure remote access to physical servers via SSH, and integrate cloud services through APIs.
- *Testing and evaluating the system's performance.* Conduct experimental testing of the developed system on various infrastructures, including physical servers and cloud platforms, to assess its efficiency and performance in real-world scenarios.
- *Analyzing results and identifying future development prospects.* Based on the testing results, draw conclusions about the feasibility of implementing the system in production environments and identify potential directions for further development and enhancement.

This structured approach ensures the development of an efficient, centralized solution that simplifies hybrid infrastructure management and optimizes server administration processes.

## METHODS AND TOOLS FOR INFRASTRUCTURE MANAGEMENT

Automation of server tasks through specialized modules using the SSH protocol significantly reduces the time required for routine tasks, such as installing updates, restarting services, scheduling tasks, or managing Docker containers. The proposed system also facilitates rapid infrastructure scaling by enabling server configuration with IaC tools like Ansible and Terraform via preconfigured templates and scripts accessible through a web interface. This approach minimizes human error and improves server reliability.

Integration with cloud provider APIs adds another layer of convenience, allowing real-time monitoring and management of cloud resources. Users can fetch data from multiple providers into a single web application and perform necessary operations across providers through a unified platform.

The developed system thus provides an efficient solution for centralized management of large-scale server infrastructures, reducing maintenance costs and improving the productivity of administrators and DevOps engineers.

## Tools for Infrastructure Management (Ansible, Terraform, Docker)

Modern automation and infrastructure management tools, such as Ansible, Terraform, and Docker, play a crucial role in establishing and maintaining effective DevOps processes. Ansible is a powerful tool for configuration management and automation, supporting both physical and virtual servers. Its modular architecture and user-friendly interface make it ideal for scaling infrastructure management processes, automating repetitive tasks, and reducing the risk of errors.

Terraform specializes in Infrastructure as Code (IaC), enabling administrators to automate the provisioning and management of infrastructure using declarative configurations. Its support for multiple cloud providers and modules makes it particularly effective for building cloud infrastructure from scratch using codified scripts.

Docker provides lightweight containerization for isolating applications on servers. It allows multiple applications or services to run on a single server in isolated environments, enhancing resource utilization and simplifying deployment and application management. Integrating Docker with the web-based management system adds a new level of automation and flexibility for both physical and cloud environments.

By integrating these tools into a centralized management system with an intuitive web interface, the proposed solution enhances the efficiency of administration and automation processes, making it a vital asset for modern IT operations.

## System Architecture

The architecture of the proposed system is based on a web application developed in Python using the Django framework, which enables centralized management of hybrid infrastructures. The system's modular structure provides flexibility for easy expansion, allowing new features to be integrated, such as support for additional cloud providers or other infrastructure management tools.

At its core, the architecture includes a server-side component that handles SSH requests via the Paramiko library, integration with Docker for managing containers on remote servers, and API interactions with various cloud providers for managing their infrastructure. The key components of the system are the backend server, the web interface for user interaction, and authentication and authorization mechanisms to ensure security.

By utilizing the paramiko.SSHClient() function and intermediate servers, the web application can execute all the functions shown in Figs. 1, 2, enabling seamless interaction and operation management.

To ensure that all functions can be performed, several variations of sending commands to the server have been developed, including:

- d connection through the terminal to the user's server;
- sending a command via a request with a response.

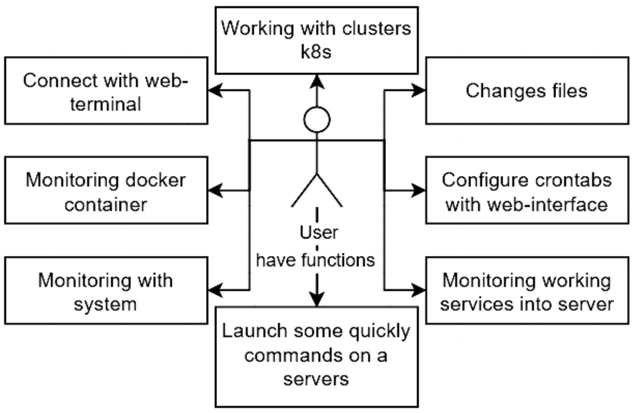

Working with clusters k8s
Connect with web-terminal
Changes files
Monitoring docker container
Configure crontabs with web-interface
User have functions
Monitoring with system
Monitoring working services into server
Launch some quickly commands on a servers

*Fig. 1.* User options

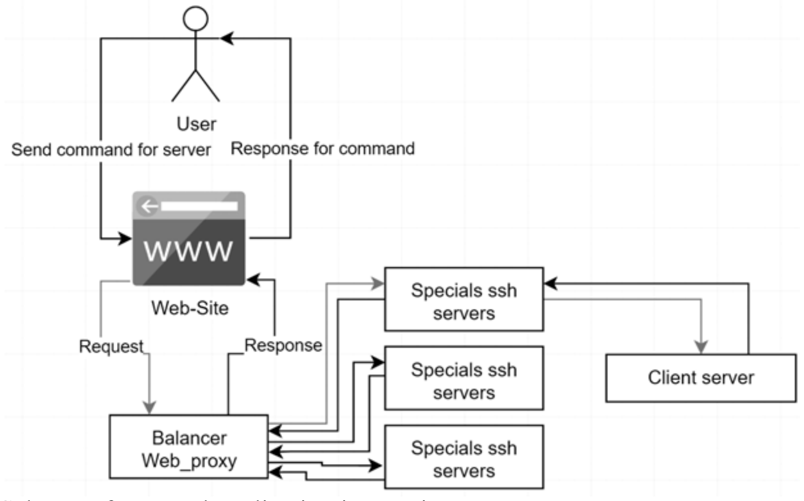
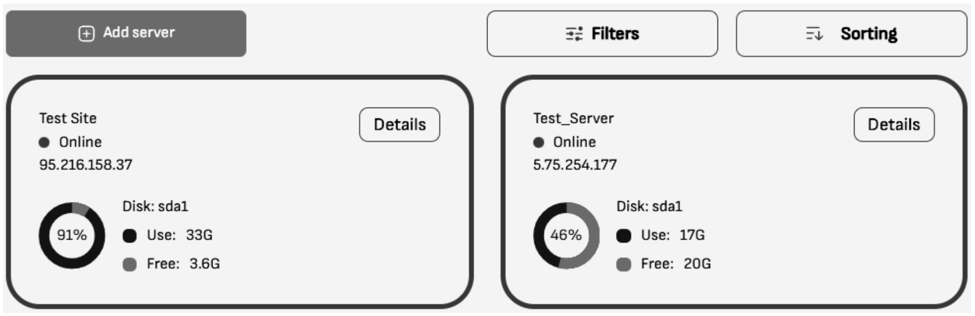


*Fig. 2.* Scheme of user and application interaction



Test Site
● Online
95.216.158.37
91%
Disk: sda1
● Use: 33G
● Free: 3.6G

Test_Server
● Online
5.75.254.177
46%
Disk: sda1
● Use: 17G
● Free: 20G

*Fig. 3.* Servers display page

The primary principle underlying this approach is that users can perform operations on their own servers from other servers, even in conditions where power outages periodically occur in Ukraine and the quality of the Internet connection can be unstable. The utilization of Paramiko for remote server access through a web application offers several key advantages. In the event of an unreliable connection, a request-response cycle is employed, and establishing a connection via a terminal will also function correctly, as it does not necessitate a full connection to the target server.The application can both automatically receive data about cloud servers, having received all the necessary information from the provider's API, and provide the ability to add a separate server to the site, which can be either physical or another cloud provider, which allows you to quickly switch and start working with it (Fig. 3).

## Automation of server operations

Secure Shell (SSH) is a standard protocol for secure server access that plays a crucial role in automating server infrastructure management processes. The utilization of SSH facilitates remote server management, command execution, service management, and automation of diverse operations, obviating the need for physical access to the servers. The ability to execute commands via terminal and transfer files renders SSH a versatile tool for managing all types of servers [7].

In contemporary environments, the implementation of SSH for the automation of operations substantially enhances the flexibility and speed of administration, particularly in distributed infrastructures where servers are dispersed across multiple geographical regions. Integrating the SSH protocol into a website enables administrators to perform operations on remote servers via terminal, thereby simplifying management and automation processes.

The utilization of SSH as a transport mechanism by prominent automation tools such as Ansible and Chef underscores its pivotal role in contemporary infrastructure automation solutions. Another key aspect is the security provided by SSH through encrypted connections and key-based authentication [8].

## Tools for Centralized Management of Cloud and Physical Servers

In the contemporary landscape, numerous solutions have emerged that facilitate the centralized management of both cloud and physical servers. A prominent example is Mirohost, and Red Hat Satellite is an infrastructure management solution designed to administer and monitor servers based on Red Hat Enterprise Linux (RHEL) and other Red Hat products. The MiroHost website facilitates the management of physical and virtual servers within a centralized system, though this functionality is exclusive to Mirohost's servers (Figs. 4, 5, 6) [9].

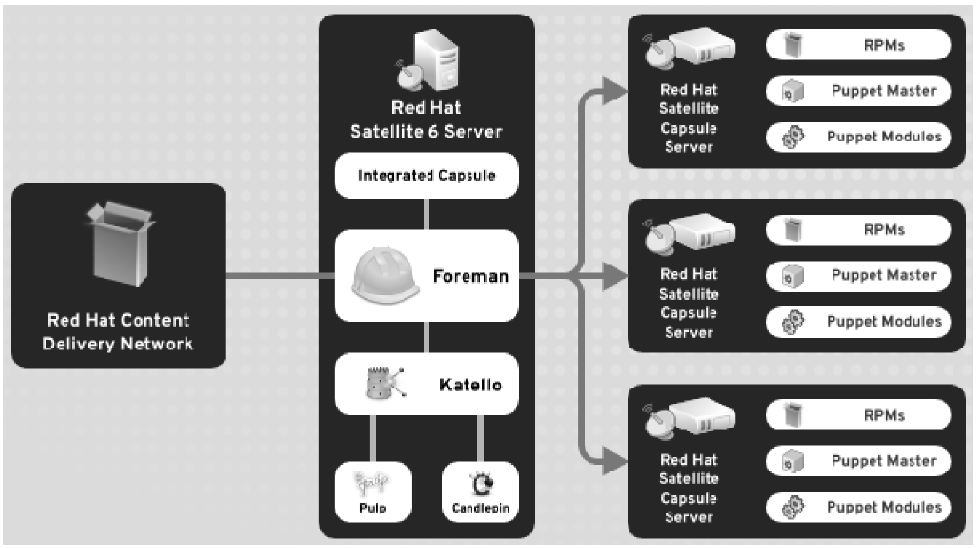

Red Hat Content Delivery Network
Red Hat Satellite 6 Server
Integrated Capsule
Foreman
Katello
Pulp
Candlepin
Red Hat Satellite Capsule Server
RPMs
Puppet Master
Puppet Modules
Red Hat Satellite Capsule Server
RPMs
Puppet Master
Puppet Modules
Red Hat Satellite Capsule Server
RPMs
Puppet Master
Puppet Modules

*Fig. 4*. Red Hat Satellite interaction diagram

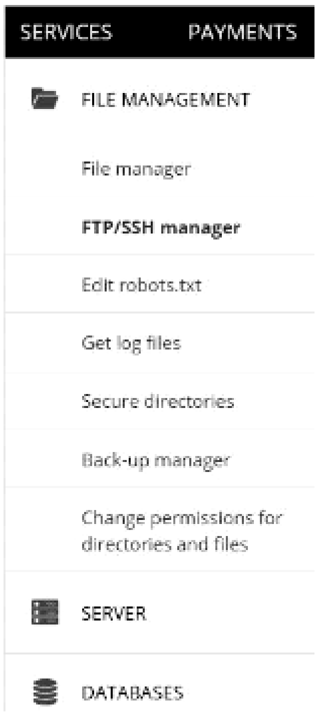
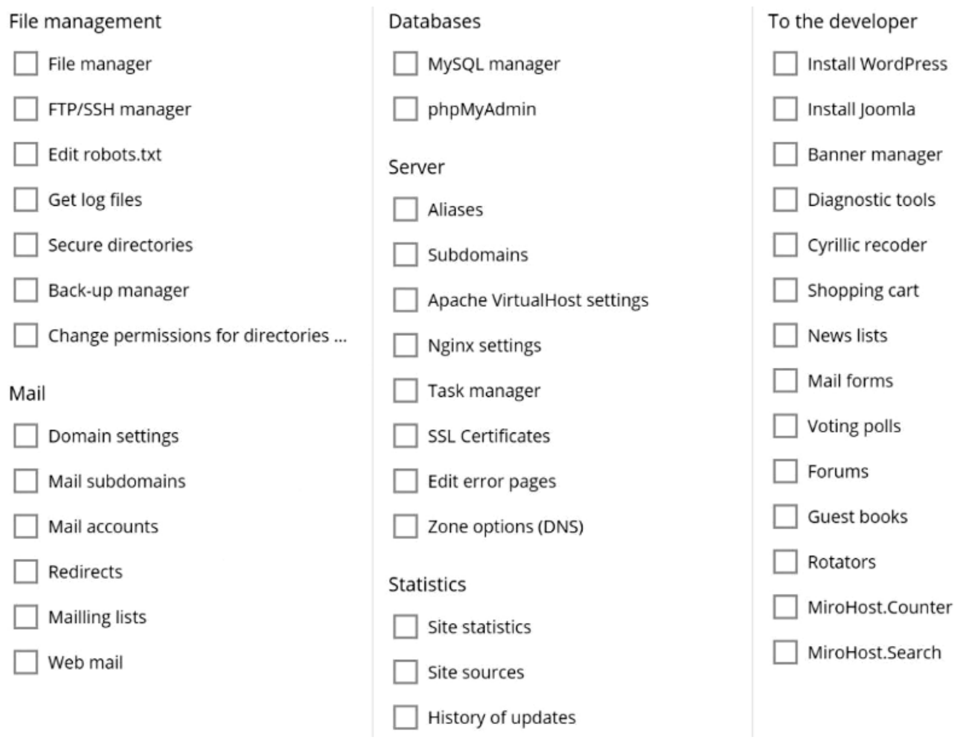


*Fig. 5*. Admin panel for Mirohost interaction


File management

- ☐ File manager
- ☐ FTP/SSH manager
- ☐ Edit robots.txt
- ☐ Get log files
- ☐ Secure directories
- ☐ Back-up manager
- ☐ Change permissions for directories ...

Mail

- ☐ Domain settings
- ☐ Mail subdomains
- ☐ Mail accounts
- ☐ Redirects
- ☐ Mailling lists
- ☐ Web mail

Databases

- ☐ MySQL manager
- ☐ phpMyAdmin

Server

- ☐ Aliases
- ☐ Subdomains
- ☐ Apache VirtualHost settings
- ☐ Nginx settings
- ☐ Task manager
- ☐ SSL Certificates
- ☐ Edit error pages
- ☐ Zone options (DNS)

Statistics

- ☐ Site statistics
- ☐ Site sources
- ☐ History of updates

To the developer

- ☐ Install WordPress
- ☐ Install Joomla
- ☐ Banner manager
- ☐ Diagnostic tools
- ☐ Cyrillic recoder
- ☐ Shopping cart
- ☐ News lists
- ☐ Mail forms
- ☐ Voting polls
- ☐ Forums
- ☐ Guest books
- ☐ Rotators
- ☐ MiroHost.Counter
- ☐ MiroHost.Search

*Fig. 6*. Mirohost functions for interacting with the server

However, the majority of existing web-based applications lack seamless integration for concurrent management of physical and cloud resources through a unified web interface. Other applications can be cumbersome to utilize or necessitate specialized configuration for various providers and servers.The study demonstrates that these limitations can be addressed by integrating the management capabilities of both physical and virtual servers within a single system. This approach has the potential to enhance the processes of backup, monitoring, and updating the infrastructure. The system of operation of the web application is very similar to the system described in (Fig. 7) [10–12].

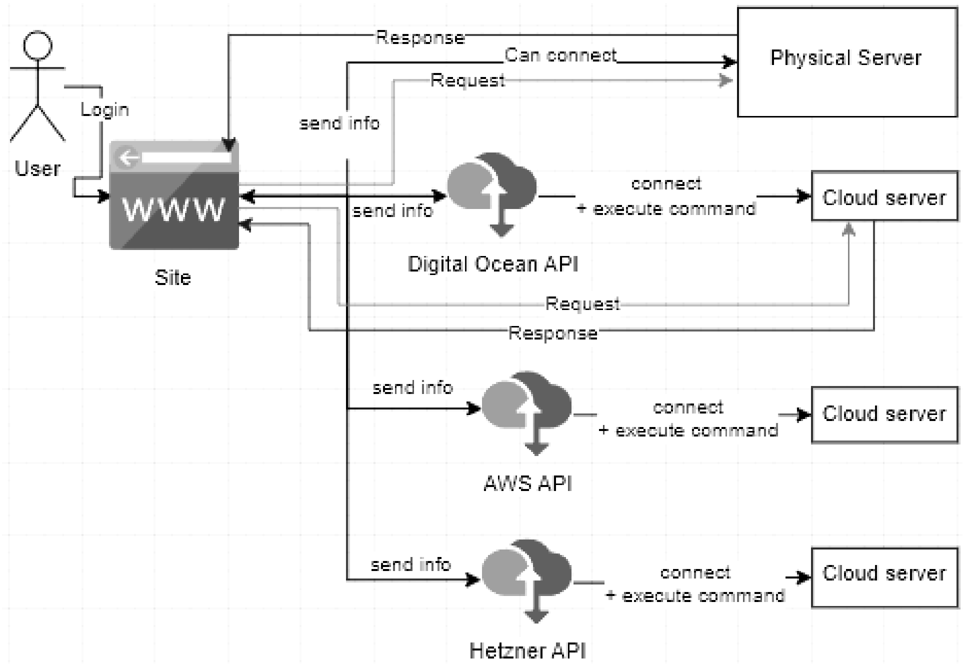


*Fig. 7*. Scheme of user interaction with the application

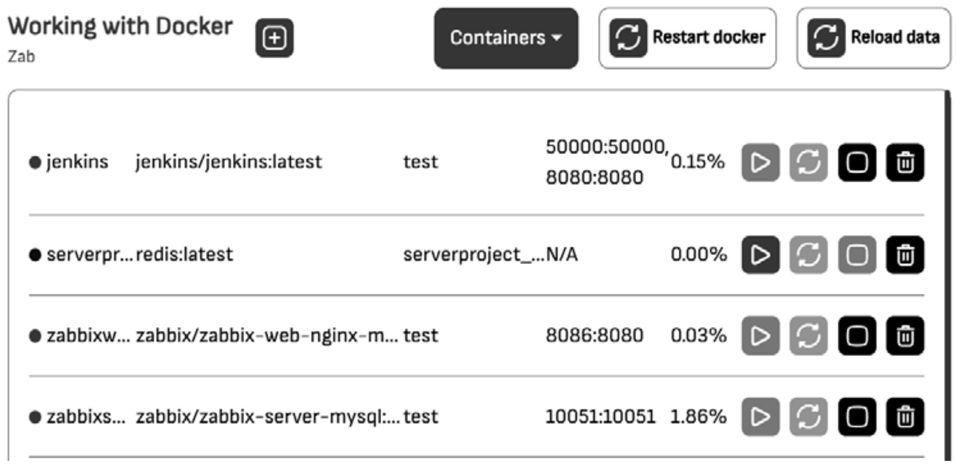
## Use of modern technologies and tools

A pivotal technology in the implementation of the system is the Paramiko library, which facilitates Secure Shell (SSH) connections to all servers [13; 14]. The utilization of SSH ensures flexibility and security in the management of disparate server types, irrespective of their geographical location or categorization.

The system incorporates a user data encryption mechanism, ensuring that added passwords and keys are encrypted using a unique algorithm. This feature facilitates secure connections to servers via the terminal and enables the execution of commands remotely through the web interface, thereby eliminating the necessity for direct interaction with the terminal.The Paramiko library plays a pivotal role in establishing reliable and secure connections to servers, as well as in the transmission of commands to administrators. This includes the management of system services and the configuration files of added servers.

The system also includes the ability to manage Docker containers (Fig. 8), which allows users to start, stop, and view the status of containers through a web interface. This is achieved with the help of Paramiko and additional modules integrated into the server side.

This module enables specialists to efficiently manage containers on remote servers. The integration of this module significantly simplifies the process of managing container environme



| | | | | |
|---|---|---|---|---|
| ● jenkins | jenkins/jenkins:latest | test | 50000:50000, 8080:8080 | 0.15% |
| ● serverpr... | redis:latest | serverproject_... | N/A | 0.00% |
| ● zabbixw... | zabbix/zabbix-web-nginx-m... | test | 8086:8080 | 0.03% |
| ● zabbixs... | zabbix/zabbix-server-mysql:... | test | 10051:10051 | 1.86% |

*Fig. 8*. Docker management page (containers, icons, network, volumes)

**Manage system services and task scheduler through the web interface**

Another salient feature is the capacity to oversee system services and the Task Scheduler via the web interface (Figs. 9, 10). This functionality enables users to view active services, initiate or halt their operation, and configure the task scheduler to automate the execution of tasks on servers. This solution facilitates the management of server operations, reduces the time required for configuration, and enhances productivity by centrally controlling all processes on disparate servers. Users can perform tasks such as adding, editing, and deleting scheduled tasks directly through interaction with the interface.

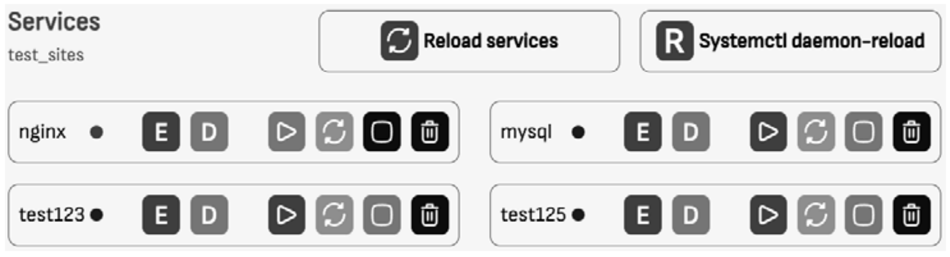


*Fig. 9.* Service management page

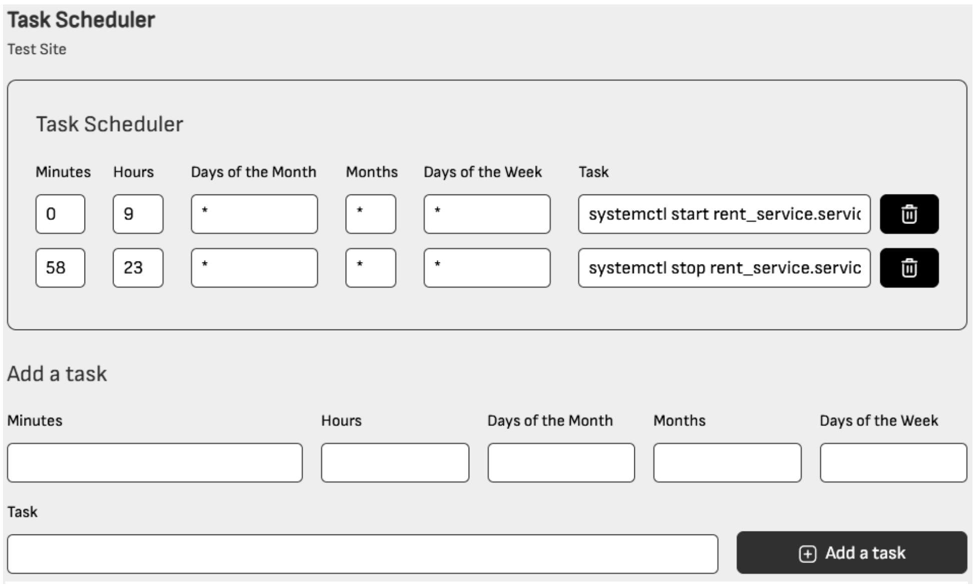
**Advantages of Centralized Server Management**

Centralized server management through a single web interface offers several significant benefits:

- *No Need for Constant Connectivity.* One of the primary advantages of using Paramiko through a web application is that it eliminates the need for a constant, stable connection to the server from the user side. Instead of a direct SSH connection from each device, administrators can work through a web interface located on a server in a region with reliable power supply and internet connection. This ensures that even in areas with intermittent internet connectivity, users can still send commands to the server.
- *Optimized Connectivity in Poor Internet Conditions.* Since the connection to the server is made via the centralized web application, users only need a short, stable connection to send commands. Even in poor internet conditions or frequent disconnections, the web application continues to process and send commands to the server, minimizing the need for constant user intervention. This is especially

beneficial in areas with unreliable connectivity, allowing administrators to execute critical tasks without the need for prolonged direct SSH connections.

- *Web Interface for Server Management.* Paramiko ensures SSH connectivity, while the web application serves as the interface between the user and the remote server. This allows administrators to remotely connect to the server's terminal through the web interface, ensuring stable connections and reducing reliance on the quality of the local network.
- *Faster Command Execution via the Web Application.* Sending commands through the web interface optimizes server operations even in low-speed internet environments. Since the web application is hosted in regions with better connectivity, commands are processed and dispatched more quickly. This reduces downtime and increases productivity by allowing administrators to quickly address various issues, even when local connections are unstable or limited.


Task Scheduler
Test Site

Task Scheduler

| Minutes | Hours | Days of the Month | Months | Days of the Week | Task |
|---|---|---|---|---|---|
| 0 | 9 | * | * | * | systemctl start rent_service.servic |
| 58 | 23 | * | * | * | systemctl stop rent_service.servic |

Add a task

Minutes, Hours, Days of the Month, Months, Days of the Week

Task

*Fig. 10.* Task Scheduler configuration

Administrators can manage the entire server infrastructure from a single place, which increases efficiency and reduces the need to constantly switch between different tools to manage various types of servers. This is especially useful in hybrid environments where both physical and cloud servers are used. Moreover, the centralized approach helps standardize administrative processes, lowering the risk of errors during routine tasks.

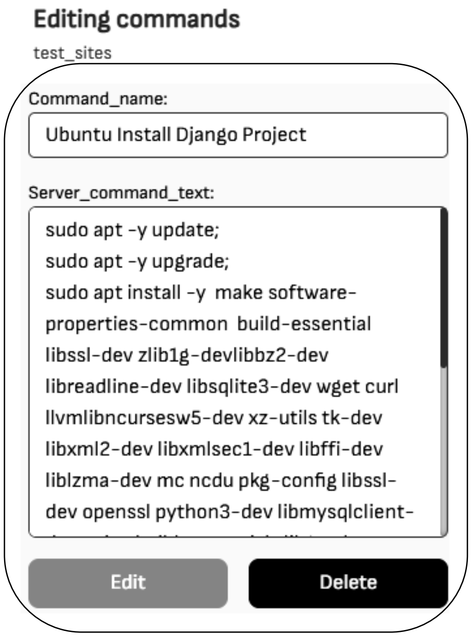
This approach also allows you to standardize administration processes, reducing the number of possible errors during routine operations. The interface provides an intuitive way to manage server configuration, task scheduler, services, updates, as well as server health monitoring and container management, which significantly reduces the level of technical knowledge required for infrastructure management.

## Convenience of server process automation

The implemented system has effectively streamlined the automation of routine server processes through a unified web interface, thereby facilitating the management of heterogeneous servers. The system's integration with the APIs of the Hetzner cloud platform and AWS exemplifies its user-centric design, significantly simplifying the configuration and management of providers' infrastructure [16; 17]. The interface's comprehensive functionality enables administrators to execute all requisite actions without the need for dedicated applications or tools for each provider. The UI/UX design of the system prioritizes minimizing administrator actions, with all essential functions accessible within a few clicks in the web interface. For instance, instead of manually configuring the server through individual providers, the system enables the utilization of the "Ubuntu install Django Project" button, which employs a prescriptive configuration template, thereby reducing the time required for setting up the infrastructure by 44%.

The interface facilitates not only the utilization of predefined templates but also the creation of bespoke scripts or command sets for expedited setup. A segment of the recorded code for the command line interface (CLI) to install the requisite dependencies on a pristine server, along with an update and the installation of Python version 3.10.5 via pyenv, is presented below.

The addition of a command to the "General custom commands" group ensures its availability across all servers, irrespective of their location or type. However, for specific tasks, "Individual custom commands" have been developed, which are displayed exclusively within a single server (Figs. 11, 12).


Editing commands

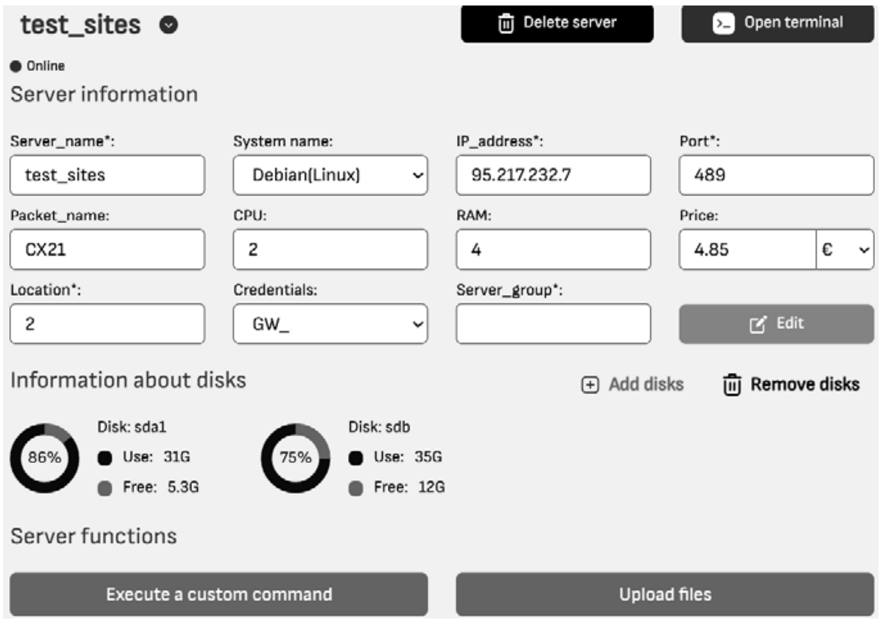
test_sites

Command_name: Ubuntu Install Django Project

Server_command_text:

```
sudo apt -y update;
sudo apt -y upgrade;
sudo apt install -y  make software-
properties-common  build-essential
libssl-dev zlib1g-devlibbz2-dev
libreadline-dev libsqlite3-dev wget curl
llvmlibncursesw5-dev xz-utils tk-dev
libxml2-dev libxmlsec1-dev libffi-dev
liblzma-dev mc ncdu pkg-config libssl-
dev openssl python3-dev libmysqlclient-
```

*Fig. 11.* Adding a new general custom command


test_sites

● Online

Server information

| | | | |
|---|---|---|---|
| Server_name*: test_sites | System name: Debian(Linux) | IP_address*: 95.217.232.7 | Port*: 489 |
| Packet_name: CX21 | CPU: 2 | RAM: 4 | Price: 4.85 € |
| Location*: 2 | Credentials: GW_ | Server_group*: | |

Information about disks

Disk: sda1 — 86% — Use: 31G — Free: 5.3G

Disk: sdb — 75% — Use: 35G — Free: 12G

Server functions

*Fig. 12*. Interaction page with the server

## CASE STUDY

This section presents an analysis of the server infrastructure configuration process, comparing manual and automated methods. The purpose of this evaluation is to determine the effectiveness of automation in managing server environments. The study involved two types of servers: physical servers based on Raspberry Pi 4 and Hetzner CX21 cloud servers, both running Ubuntu 24.04. The time required to configure the infrastructure manually and using an automated system was measured and compared, highlighting the advantages and limitations of each approach. The evaluation focuses on the configuration processes, tools used, and the impact of automation on efficiency and accuracy.

### Server Specifications

To evaluate the effectiveness of automating server infrastructure configuration, a case study was conducted using five Hetzner CX21 cloud servers and five physical servers based on Ubuntu 24.04 running on Raspberry Pi 4. Each physical server has its own IP address and SSH port for access. The time required for configuring the infrastructure manually and using the automated system was measured.

*Characteristics of Physical Servers:* Processor — Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz; RAM: 8 GB SDRAM; Disk: 32 GB; Operating System: Ubuntu 24.04.

*Characteristics of Hetzner CX21 Cloud Servers:* Processor -2 vCPU (virtual cores) AMD EPYC 7002 series; RAM: 4 GB RAM; Disk: 40 GB SSD; Operating System: Ubuntu 24.04.

### Manual Configuration Process

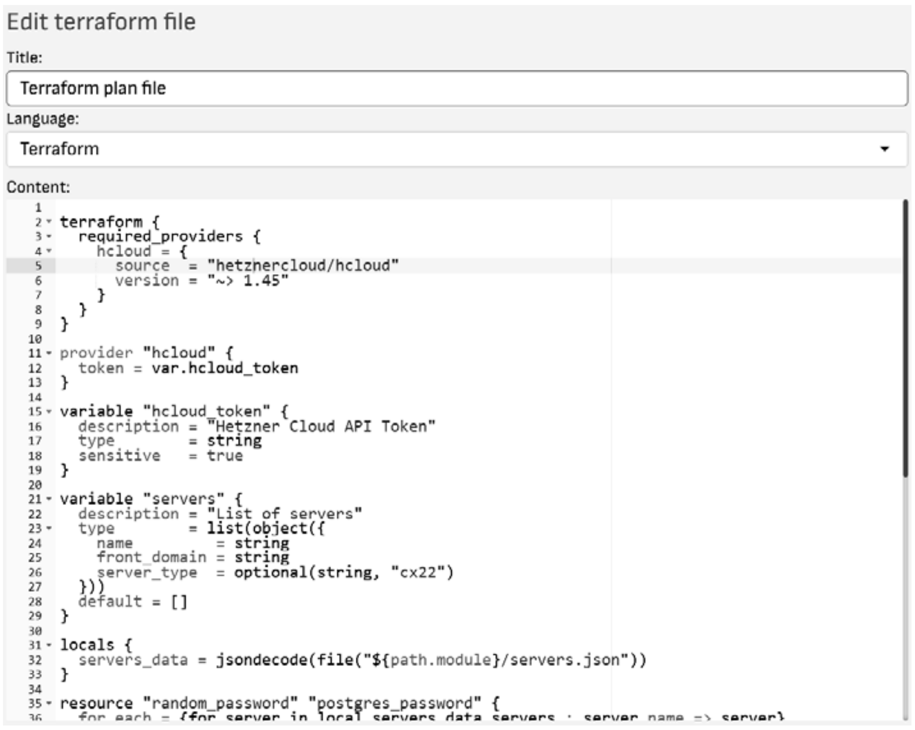
In this case, the process involved running commands through the terminal without using Ansible or Terraform, as these tools require additional knowledge and expertise. Working with Ubuntu, the server contains only minimal pre-installed programs, which requires the engineer to manually execute the following steps:

- Connect to the server.
- Update the system.
- Install all necessary packages (web server, database server, etc.).
- Prepare directories and create a virtual environment.
- Copy the project.
- Configure the web server.
- Check the results.

A Bash script for Ubuntu servers was written in advance. After the cloud servers were created and the operating system was installed on the physical servers, the administrator needed to log into each server via SSH (without using third-party software, just the standard admin console and SSH tools), transfer the script to the server, set the appropriate permissions, and run it, hoping that all syntax and indentation were correct. Time was spent switching between the browser and terminal, looking for IP addresses, and more. As a result, the manual configuration of 10 servers took 18 minutes using the terminal and a script file.

### Automation Configuration Process

For efficiency evaluation, the server configuration process was carried out using the developed web application. The web application includes a page for file editing and running scripts via an integrated code editor that supports a set of popular programming languages. This allows users to quickly edit configuration files and scripts, such as Terraform or Ansible, and run them directly from the browser on their servers. Syntax checking plays a crucial role in improving code-writing speed, as errors are immediately visible (Fig. 13).



Edit terraform file

Title: Terraform plan file

Language: Terraform

Content:

```

terraform {
  required_providers {
    hcloud = {
      source  = "hetznercloud/hcloud"
      version = "~> 1.45"
    }
  }
}

provider "hcloud" {
  token = var.hcloud_token
}

variable "hcloud_token" {
  description = "Hetzner Cloud API Token"
  type        = string
  sensitive   = true
}

variable "servers" {
  description = "List of servers"
  type        = list(object({
    name         = string
    front_domain = string
    server_type  = optional(string, "cx22")
  }))
  default = []
}

locals {
  servers_data = jsondecode(file("${path.module}/servers.json"))
}

resource "random_password" "postgres_password" {
  for_each = {for server in local.servers_data.servers : server.name => server}
```

*Fig. 13*. Integrated file editor with syntax highlighting (programming languages)

For the time measurement in this case study, the configuration process was performed using the web application's quick command tools without utilizing Ansible or Terraform. However, the web application already provides templates for working with these tools to facilitate interaction with them.

Once all the servers were created and the operating system was installed on the new physical servers, the following steps were performed within the web application:

- Add server authorization details.
- Add physical servers (cloud instances are displayed automatically via the cloud provider's API).
- Open the first server (of any type) and use the "add custom commands" function to specify the script for execution (the newly created command will appear on all server pages, both existing and new).
- On each server page, click the newly created command and wait for the execution result.

As a result, the automatic configuration of 10 servers took 10 minutes using the web application and the script file. By using the web application, no additional time was spent on connecting, transferring script files, or entering commands manually.

To calculate how the automated system reduces the time required to configure server infrastructure, the following formula was used:

$$E_t = \frac{(T_m - T_a)}{T_m} * 100\% \ ,$$

where $E_t$ — time savings percentage; $T_m$ — manual configuration time (without automation); $T_a$ — configuration time using the automated system.

The results obtained, measured using a timer, indicate that 18 minutes were spent manually configuring the servers. This time was obtained from the actual process of configuring the servers in a real environment, involving the following steps: manually connecting to each server, passing the configuration script, setting the appropriate permissions and executing the script, while frequently switching between the terminal and browser to look up IP addresses and other details. The total duration of the manual configuration of all ten servers was thus 18 minutes.Consequently, $T_m = 18$ minutes signifies the actual time taken to manually configure a server without automation. The 10 minutes required for the automated configuration were determined using the developed web application, with the automated system simplifying the process by executing the script on all servers, thereby greatly reducing the need for manual intervention. The total duration from the addition of servers to the execution of the customization script was 10 minutes. Therefore, $T_a = 10$ minutes signifies the time required to complete the server configuration using the automated system through the web application.

As a result, by arithmetic calculation we have $E_t = 44\%$.

Thus, the system reduces the configuration time by 44%, demonstrating the efficiency of automation:

- *Before automation:* 18 minutes for manual server configuration.
- *After automation:* 10 minutes using the web application (44% time savings).

The implementation of the automated solution, utilizing configuration templates, reduces the server configuration time by 44%.

## DISCUSSION: RESULTS AND FUTURE RESEARCH

The study demonstrated that centralized management of both physical and virtual servers through a web interface significantly impacts the daily operations of administrators and DevOps specialists. Using such a system can considerably reduce the occurrence of errors during manual operations, as well as minimize unexpected server failures that require human intervention. It simplifies the management of complex infrastructures and reduces the time spent on routine tasks. Automation of these processes helps improve the overall efficiency of DevOps teams, enabling them to focus on critical aspects of software development and maintenance. Centralized solutions provide not only operational control over infrastructure but also promote rapid implementation of changes in response to dynamic task requirements, making this approach highly relevant to modern workflows and enhancing IT operations productivity.

Current solutions have certain limitations, most notably the lack of an integrated approach that allows centralized management of both physical and cloud-based servers through a single web interface. Although individual solutions allow partial integration of such servers, the absence of a unified platform capable of effectively managing both types of resources presents a significant drawback.

For instance, cloud providers offer built-in tools to manage only their services, but they do not provide solutions for managing physical servers or servers from other providers. This creates the need to use multiple tools to manage different infrastructures, complicating the process and reducing administrative efficiency. An integrated solution could streamline the management of hybrid infrastructures, reducing the time spent on administration and improving the overall reliability of the systems.

## CONCLUSION

The study's development of a centralized hybrid infrastructure management system is notable for its provision of integrated administration of physical servers and cloud instances through a single web interface. A key feature of the system is its support for remote access via SSH using the Paramiko library, which facilitates convenient management of servers regardless of their location. The relevance of the proposed solution is evidenced by the contemporary requirements for server infrastructure, which necessitate a high degree of automation, convenient monitoring and management of a substantial number of cloud and physical resources.The presented system offers advanced administration capabilities, rendering it a versatile and powerful tool for DevOps professionals.

The study's findings indicate that the implementation of centralized solutions enhances the reliability of server infrastructure, streamlines maintenance operations, and mitigates the risk of human errors during manual management.

In the future, the centralized management system will be enhanced to include integration of additional cloud providers, such as Google Cloud Platform and DigitalOcean. This will extend administrative capabilities to a larger number of cloud platforms. The system may include improved monitoring features, such as tools for automatic notification of potential problems, load analysis tools, or detection of security vulnerabilities. It is expected to support automatic deployment of servers and containers using Kubernetes, which will provide greater scalability without increasing administrative costs. It is also of interest to optimize algorithms for more cost-effective and faster provisioning of servers in cloud providers using IaC (infrastructure as code) tools that support scalability.

**Acknowledgments.** The authors express their gratitude to the staff of the Institute of Computer Science and Artificial Intelligence at V.N. Karazin Kharkiv National University for their valuable discussions and insightful comments, which significantly contributed to improving the quality of the presented results.

## REFERENCES

1. Hernandez L., C.E. Uc Rios, "Docker Optimization of an Automotive Sector Virtual Server Infrastructure," *Knowledge Engineering and Data Science*, 7(1), pp. 71–85, 2024. doi: https://doi.org/10.17977/um018v7i12024p71-85

2. X. Song, Y. Wu, Y. Yao, B. Li, “Research on Distribution Automation Testing Platform Based on Kubernetes Technology,” *J. Phys.: Conf. Ser.*, 2777 012005, 2024. Available: https://iopscience.iop.org/article/10.1088/1742-6596/2777/1/012005/pdf
3. K. Nugroho, A.D. Abrariansyah, S. Ikhwan, “Perbandingan Kinerja Library Paramiko Dan Netmiko Dalam Proses Otomasi Jaringa,” *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 5, no. 1, 2020. Available: https://jurnal.uisu.ac.id/index.php/infotekjar/article/download/2758/pdf
4. P. Kaushik, A.M. Rao, D.P. Singh, S. Vashisht, S. Gupta, “Cloud Computing and Comparison based on Service and Performance between Amazon AWS, Microsoft Azure, and Google Cloud,” *International Conference on Technological Advancements and Innovations* (*ICTAI*), pp. 268–273, 2021. doi: 10.1109/ ICTAI53825.2021.9673425
5. M. Spichkova, J. Bartlett, R. Howard, A. Seddon, X. Zhao, Y. Jiang, “SMI: Stack Management Interface,” *23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 156–159, 2018. doi: 10.1109/ ICECCS2018.2018.00024
6. S. Bhatia, C. Gabhane, “Terraform: Infrastructure as Code,” in *Reverse Engineering with Terraform*. Apress, Berkeley, CA, 2024, pp. 1–36. doi: https:// doi.org/10.1007/979-8-8688-0074-0_1
7. P. Kirkbride, “Using SSH,” in *Basic Linux Terminal Tips and Tricks*. Apress, Berkeley, CA, 2020. https://doi.org/10.1007/978-1-4842-6035-7_5
8. A. Witt, S.Westling, *Ansible in Different Cloud Environments*, 2023. Available: http://www.diva-portal.org/smash/get/diva2:1765141/FULLTEXT01.pdf
9. A. Chan, D. Macpherson, *Red Hat Satellite 6 System Architecture*. Accessed on: 05.01.2024. [Online]. Available: https://docs.redhat.com/en/documentation/red_hat_ satellite/6.0/html-single/user_guide/index#sect-Red_Hat_Satellite-User_Guide-Red_Hat_Satellite_6_System_Architecture
10. K.I. Mantilla, A. Flórez, “Development of a Monitoring Module for Physical and Virtual Servers: Advanced Computing Center Case of the Universidad Pontificia Bolivariana, Bucaramanga, Colombia,” *Journal of Physics: Conference Series*, vol, 1513, 2020. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1513/ 1/012006
11. Almeida Wilfred, “SC Connect: Secure Server Access from Mobile Device,” *IEEE Bombay Section Signature Conference (IBSSC)*, pp. 1–3, 2022. doi: 10.1109/IBSSC56953.2022.10037430
12. B. Donald, D. Quirk, “Managing Infrastructure: DCIM: Is it a Product or a Solution?” *ASHRAE Journal*, vol. 57, pp. 82–84, 2015. Available: https://www.researchgate.net/publication/285366410_DGIM_Is_it_a_Product_or_a_ Solution_Managing_Infrastructure
13. M. Zadka, “Paramiko,” in *DevOps in Python*. Apress, Berkeley, CA, 2019. doi: https://doi.org/10.1007/978-1-4842-4433-3_9
14. A. Florez, L. Serrano, U. Gomez, L. Suarez, H. Villalarga, H. Rodriguez, “Analysis of Dynamic Complexity of the Cyber Security Ecosystem of Colombia,” *Future Internet*, 8(3), 33, 2016. doi: https://doi.org/10.3390/fi8030033
15. K. Maurice et al., “Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud,” *NDSS*, vol. 17, pp. 8–11, 2017. doi: http://dx.doi.org/10.14722/ndss.2017.23294
16. C. Tao, G. Xiaofeng, C. Guihai, “The Features, Hardware, and Architectures of Data Center Networks: A Survey,” *Journal of Parallel and Distributed Computing*, 96, 45, 2016. doi: 10.1016/j.jpdc.2016.05.009
17. Y. Su, T. Zhou, W. Zheng, L. Zhao, H. Zhen, G. Huang, “Construction of Power System Computing and Analysis Platform Based on Cloud Computing,” *Southern Power System Technology*, 16(07), pp. 67–75, 2022.

*Received 27.01.2025*

## INFORMATION ON THE ARTICLE

**Ivan S. Byzov,** ORCID: 0009-0004-2950-7814, V.N. Karazin Kharkiv National University, Ukraine, e-mail: utelephona@gmail.com

**Sergiy V. Yakovlev,** ORCID: 0000-0003-1707-843X, V.N. Karazin Kharkiv National University, Ukraine, e-mail: s.yakovlev@karazin.ua

**СПРОЩЕНЕ КЕРУВАННЯ ФІЗИЧНОЮ ТА ХМАРНОЮ ІНФРАСТРУКТУРОЮ ЧЕРЕЗ ЦЕНТРАЛІЗОВАНИЙ ВЕБ-ІНТЕРФЕЙС** / І.С. Бизов, С.В. Яковлев

**Анотація.** Присвячено розробленню системи централізованого управління фізичними та віртуальними серверами через веб-інтерфейс. Система дозволяє адмініструвати серверну інфраструктуру за допомогою різних інструментів. Означено ключові компоненти системи, інтеграція з інструментами управління серверами, використання Paramiko для віддаленого доступу через SSH, а також можливості застосування інших технологій для управління хмарними і фізичними серверами через веб-інтерфейс. Система дає змогу централізовано керувати різними типами серверів через веб-інтерфейс, підтримуючи як базові, так і розширені адміністративні операції. Інтеграція з Docker і підтримання хмарних API забезпечують зручну автоматизацію і спрощують операції із хмарними і фізичними серверами. Система слугує універсальним інструментом для DevOps-інженерів. Дослідження підкреслює актуальність і важливість централізованого підходу до управління серверною інфраструктурою, що підвищує ефективність і надійність роботи серверів у сучасних умовах.

**Ключові слова:** централізоване управління, SSH, веб-інтерфейс, хмарні сервіси, автоматизація, DevOps.