Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **69**, *3*, pp. 347–352, Bucarest, 2024

AUTOMATION IMPROVEMENT FOR GIS-BASED APPLICATIONS DEPLOYMENT IN FAST-GROWING HIGH SCALABILITY DATA-ROOMS

CRISTIAN ALEXANDRU CAZAN[1], CONSTANTIN VIOREL MARIAN[2]

Keywords: Automation; Software deployment; Geographic information system (GIS) conversion; Software resources configuration.

This paper presents the viability and importance of automation in deploying and managing software systems to reduce costs. Automation allows for optimizing time spent on resource deployment and configuration and simplifying the upgrade process. A real-world geographic information system (GIS) project illustrates these improvements by automating WGS84 & STEREO70 conversion modules for software systems.

1. INTRODUCTION

The Industry 4.0 concept introduces smart manufacturing through the digital transformation of manufacturing companies. The introduction of automation, the Internet of Everything, and education significantly impact increased productivity and flexibility [1]. The evolution of the Internet of Things (IoT) concept over time [2] is part of the Internet of Everything (alongside the Internet of Things, Internet of Data, Internet of Services, and Internet of People).

Industry 4.0 impacts significant life aspects, such as communication, social media, management, and digital learning methods [3].

Software deployment automation has become essential with the continuous evolution of modern software and modern software development.

The increasing complexity of software and rising cybernetic threats compel companies to invest in processes that increase the efficiency of software development and deployment in fast-paced, real-time environments.

Modern software has become highly complex, and with the advent of cloud computing and cloud providers, it has grown to leverage the dynamic and scalable infrastructures those technologies provide. Companies begin to migrate their products to the cloud. They must adapt their deployment and provisioning processes to leverage this new environment, using load-balancing algorithms to distribute the workload evenly between virtual machines and obtain better performance [4].

As a result, infrastructure can be provisioned and de-provisioned dynamically. Owing to this newfound agility, new efficient ways to manage the infrastructure have become imperative for any engineering department and its respective companies. Manual, one-off configurations are proven to be inefficient but also error-prone and time-consuming, resulting in time lost that could be better spent developing and improving the product itself.

Infrastructure as Code (IaC) [5] has emerged as a response to these needs and challenges. IaC is a modern practice in software development that allows developers and operations teams to manage the provisioning process and the infrastructure through readable definition files rather than physical hardware or interactive tools. Specifically, it treats infrastructure as software, enabling versioning, testing, or testing automation [6] and deployment that follows the same rigor as normal application software.

2. ARCHTERR PROJECT

ArchTerr [7] is a geographic information system (GIS)-based integrated cultural and archaeological heritage protection system. GIS [8] are computer-assisted systems for capturing, storing, retrieving, analyzing, and displaying spatial data.

ArchTerr is an interactive digital map containing information regarding archaeological sites throughout Romania (freely available at www.archterr.ro).

ArchTerr is a working tool available to the Romanian Ministry of Culture's territorial branches. It provides a record-keeping database system and procedures for enforcing legislation surrounding archaeological heritage protection.

One of the main features of this integrated system is the ability to display both STEREO70 (Stereographic 1970) [9] and WGS84 (World Geodetic System 1984) [10] standard coordinates for all documented archeological sites and allow users to convert their coordinates.

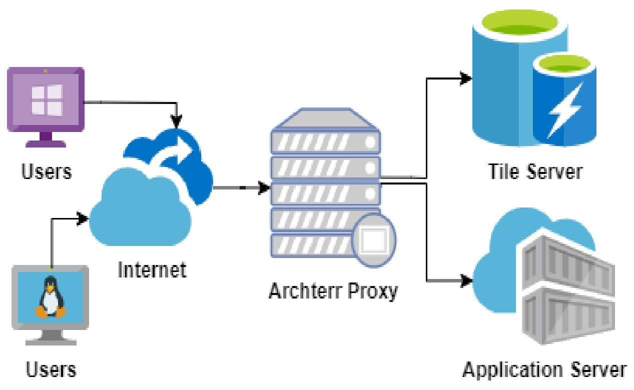

Fig. 1 – ArchTerr basic architecture.

ArchTerr architecture (see Fig. 1) consists of two major components: the application server and the map servers. Requests are routed to each element through a reverse web proxy.

[1] Doctoral School on Automatic Control and Computers, National University of Science and Technology "Politehnica" Bucharest, Bucharest, Romania
[2] National University of Science and Technology Politehnica Bucharest, Bucharest, Romania (Corresponding).
Emails: cristian.cazan1997@outlook.com, cristian.cazan2908@upb.ro, constantinvmarian@gmail.com , constantin.marian@upb.ro

DOI: 10.59277/RRST-EE.2024.69.3.16

On a hardware level, the two components are deployed on two different virtual machines on a physical server. This allows for easy backup and restoration of each virtual machine in case of failure.

In this system, the tile server has only one purpose: to generate the tiles used by the application to display geographic information stored within the local PostgreSQL database (Fig. 2).

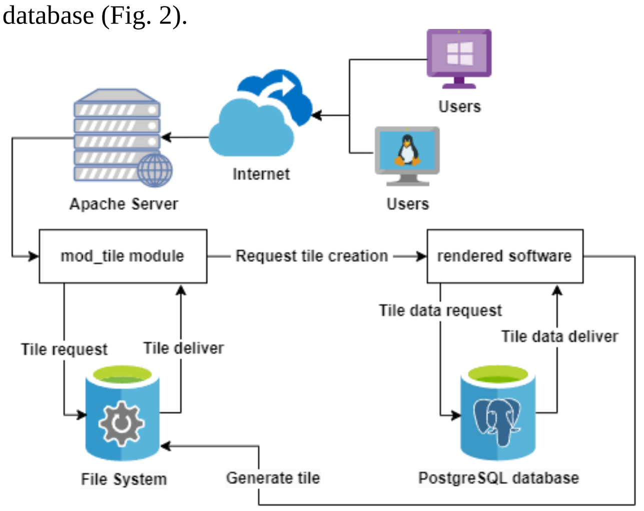

Fig. 2 – ArchTerr components (Tile server).

Thus, the tile server needs to be able to render the raster graphics files using the geographic information stored in the database and then deliver the said renders to the client. This is achieved using a daemon called rendered and an Apache module called mod_tile to manage the file transfer.

All the information the TileServer provides is only useful with the web (Fig. 3) ApplicationServer, which manages the interactive application. It is a web application built using the open-source Leaflet [11] library. Using this library, the application will interface with the TileServer and display several zoom (detail) levels based on the chosen tile layers.

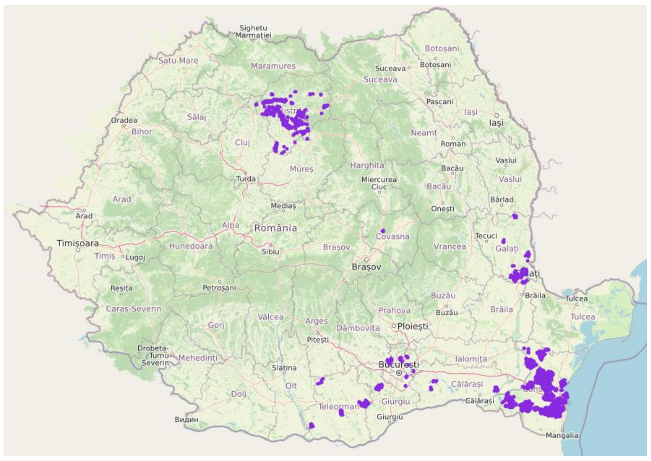

Fig. 3 – Web view.

Additionally, specific information, such as archaeological sites, documentation, and user information, will be stored in the local PostgreSQL database and used to generate the relevant reports.

3. MODELING THE DEPLOYMENT PROCESS

Before the advent of IaC and automation processes, deploying a software system was rather straightforward. Hardware resources, in this case, one or more physical machine servers, are provisioned by a system administrator and then configured to fit their respective roles, the application and tile servers.

Software is installed on each machine individually and configured to use the relevant networks, certificates, and secrets. At the end of the process, the system is up and running. But what happens in the event of a necessary update? What if a disaster occurs, and the system must be deployed again? A system administrator needs to manage these actions manually.

We will consider that a physical machine has already been provisioned. This machine will have the physical machine hosting the virtual machines that comprise the ArchTerr system. To ultimately deploy ArchTerr, we will require two virtual machines. This will be the first step of the new process, the provisioning of virtual machines. All necessary software will be deployed to the machines and configured after provisioning. Once configured, a smoke test is performed to ensure the system functions correctly.

A preliminary model has been created to describe the steps necessary to automate the provisioning and deployment of the ArchTerr system (Fig. 4). However, this model needs to consider existing infrastructure and configuration. As stated before, with the appearance of IaC, infrastructure should be versioned as any standard application software.

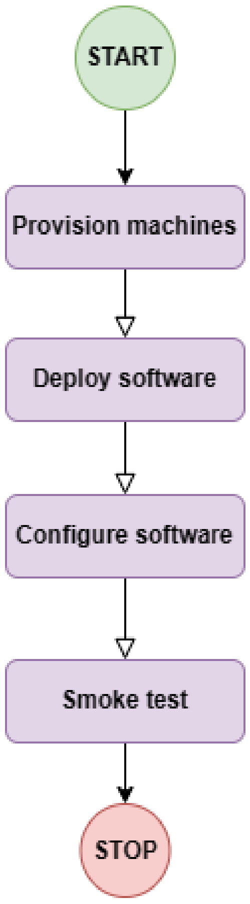

Fig. 4 – Preliminary process.

Versioning is important when continuously deploying and redeploying infrastructure. The process must consider the system's current state and the desired target. This is especially relevant when performing updates or upgrades on existing infrastructure.

The functionalities implemented for server management and automation include the following:

- Configuration management to automate the creation and maintenance of system configurations. It ensures server consistency by defining desired states (installing packages, managing users, configuring services).
- Deployment to automate application installation and updates. It streamlines pushing code changes into production (deploying web applications and updating databases).
- Backup and Restore to automate data backup and recovery. It ensures data integrity and disaster recovery (regular backup of databases, files, and configurations).
- Monitoring and alerts to automate monitoring tasks to detect problems. It sets alerts for abnormal conditions (monitoring CPU usage, disk space, and network traffic).
- Security and compliance to automate security-related tasks. It enforces compliance policies across servers (applying security patches and managing firewall rules).

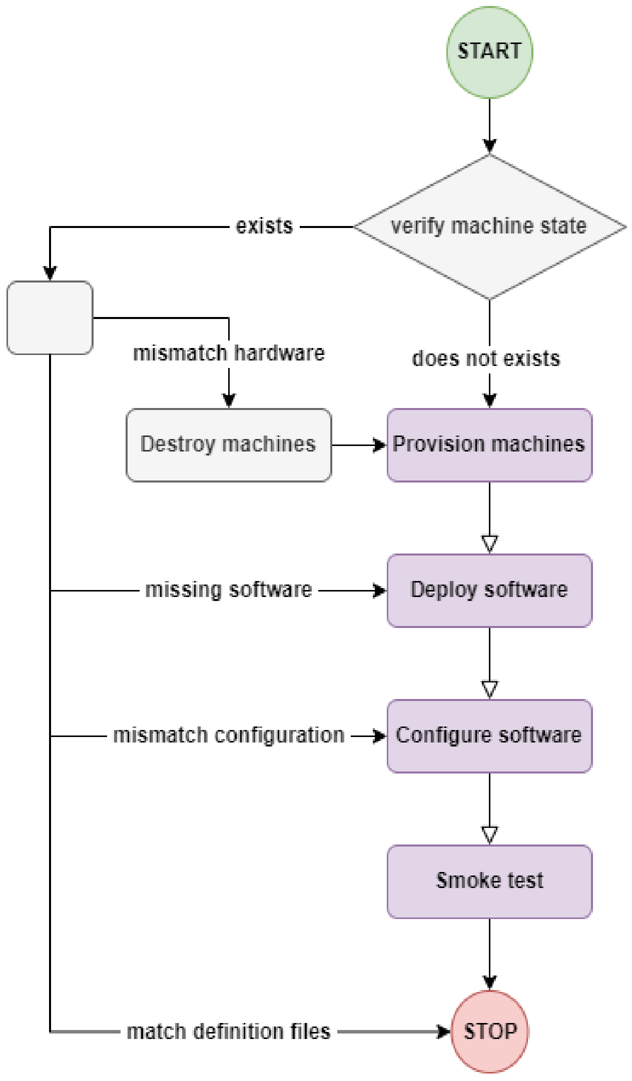

Fig. 5 – Final process.

As evidenced by Fig. 5, a few scenarios arose. Referring to the definition files at the core of the IaC practice, several checks must be performed to ensure consistency between the system and the state definition files.

1. Machine state. If the machines are no longer provisioned or visible due to a disaster, begin the provisioning process.

2. System state. If the machine exists, gather system information: hardware configuration, existing software, and relevant versions, and configuration relating to secrets, keys, and environment variables.

3. If the hardware does not match the most recent definitions, destroy the machines and recreate them with the correct resources.

4. If the hardware matches definitions but the software is missing, install the missing software and configure it.

5. If the hardware and software match definitions, but some configurations do not, overwrite all incorrect configurations.

6. If the hardware, software, and configuration match the definition, terminate execution.

With the process now clearly defined, choosing the tools necessary for automating the deployment process is possible.

4. AUTOMATING THE ARCHTERR DEPLOYMENT

Some base assumptions need to be made to automate the deployment and configuration of the ArchTerr integrated system.

1. Servers are based on a Linux distribution. In this case, it will be Debian 11.

2. Servers will be virtual machines hosted on the same physical machine.

3. The physical machine (server) has already been installed and is accessible to the hosting network.

Vagrant [12] is a software product developed by HashiCorp that allows to build complete, reproducible development environments. For the demonstration, the tool will be used to automate the deployment of the two servers. This is achieved through the creation of a Vagrantfile, where the configuration is defined as such:

```
Vagrant.configure("2") do |config|
 config.vm.define "app" do | app |
  web.vm.box = "ubuntu/focal64"
  web.vm.hostname = app
  web.vm.network :private_network, ip: "192.168.56.101"
 end

 config.vm.define "tile" do | tile |
  db.vm.box = "debian/buster64"
  db.vm.hostname = tile
  db.vm.network :private_network, ip: "192.168.56.102"
 end
end
```

Any other static configuration could be deployed similarly if lacking access to a secret vault that would allow the process to pull the information from a more secure environment directly.

With the static configuration now being deployed, another important step is the database backup automation. In this case, a daily backup would suffice. This could be implemented as follows:

```
cp "$PJDIR/automation/backup.sh" /opt/backup.sh
chown postgres:postgres /opt/backup.sh
chmod 554 /opt/backup.sh # RX RX R
mkdir -p /var/backups/postgres
chown postgres:postgres /var/backups/postgres
```

```
chmod 604 /var/backups/postgres # RW - R
(crontab -u postgres -l ; echo "0 3 * * * /opt/backup.sh") |
                    crontab -u postgres -
```

A small script has already been provided for the backup process in the above code. However, what is important to illustrate is the proper assignment of ownership and permissions to the Postgres user and the execution scheduling. At 03:00 every day, the backup script will secure the entire database.

Now, it is necessary to configure the application to start as a service. This can be achieved with a definition file in the system daemon. Configuring the service to run with the users created explicitly for the application is important. Other relevant configurations are the restart policy and the application's working directory.

After creating the file, the daemon must be reloaded, and the service must be enabled to start at boot.

```
touch /etc/systemd/system/archterr.service
printf "[Unit]" \
  "Description=API for starting ArchTerrGIS application" \
  "\n[Service]" \
  "ExecStart=npm start" \
  "Restart=always" \
  "WorkingDirectory=/opt/archterrgis" \
  "User=archterr" \
  "Group=nrchterr" \
  "\n[Install]" \
  "WantedBy=multi-user.target" >
                    /etc/systemd/system/archterr.service
systemctl daemon-reload
systemctl restart archterr
systemctl enable archterr
```

There is more configuration to do for the database and map services on the Tileserver itself. As this machine will solely generate and publish the raster graphics, some database optimization can be done. In this scenario, the data will be stored in a PostgreSQL database.

```
total_mem=$(cat /proc/meminfo | grep MemTotal | awk '{print
          int($2 / 1024 / 1000 + 0.5)}')
sed -i 's/^shared_buffers = .*$/shared_buffers =
          $((total_mem/4))GB/' \
          /etc/postgresql/11/main/postgresql.conf
sed -i 's/^work_mem = .*$/work_mem = 1GB/' \
          /etc/postgresql/11/main/postgresql.conf
sed -i 's/^maintenance_work_mem =
          .*$/maintenance_work_mem = $((total_mem/6))GB/'
          \/etc/postgresql/11/main/postgresql.conf
sed -i 's/^effective_cache_size = .*$/effective_cache_size =
          $((total_mem/4))GB/'
          \/etc/postgresql/11/main/postgresql.conf
systemctl restart postgresql
```

This allows the configuration to be dynamically generated based on the system's resources. If the machine is upscaled or downscaled, the configuration will be automatically altered during the configuration process.

Considering a different deployment environment, such as a cloud environment, all the automation presented above can be deployed with a configuration management system (CMS) such as Ansible. This would allow for a more transparent configuration and eliminate potential differences between distributions and distribution versions due to the internal mechanics of Ansible [13]. Alternatively, the usage of containers and the implicit containerization of the application could prove fruitful. By switching to a container approach, versioning the deployment of the application becomes rudimentary, and most of the machine's configuration is replaced by the creation of the image. This can be taken a step forward when utilized in a cloud environment. As such, we could integrate with Kubernetes for cloud computing or high-performance computing, ultimately leveraging the service-oriented resource management system and scheduling mechanism [14].

The logical schema in Fig. 6 presents the relationship between a central Configuration Management Server (CMS) (Ansible) and the virtual machines within its inventory. Through this abstraction, the administrator only needs to maintain one version of the same script, without being impacted by the differences between distributions: yum/dnf (CentOS) vs apt (Ubuntu & Debian); selinux (CentOS) vs apparmor (Debian) [15].

Since the project began, we have observed an exponential increase in registered archeological sites and users. We are designing the new platform, which will be based on a cluster [16] and potentially migrate to microservices.

In the future, along with clustering implementation, we'll create scaling and load-balancing scripts to automate server scaling based on demand. This will distribute traffic efficiently using load balancers (auto-scaling instances, load balancing web servers).

Another module will focus on orchestration to coordinate complex workflows involving multiple servers (automating sequences of tasks across systems) such as rolling software updates and failover procedures).

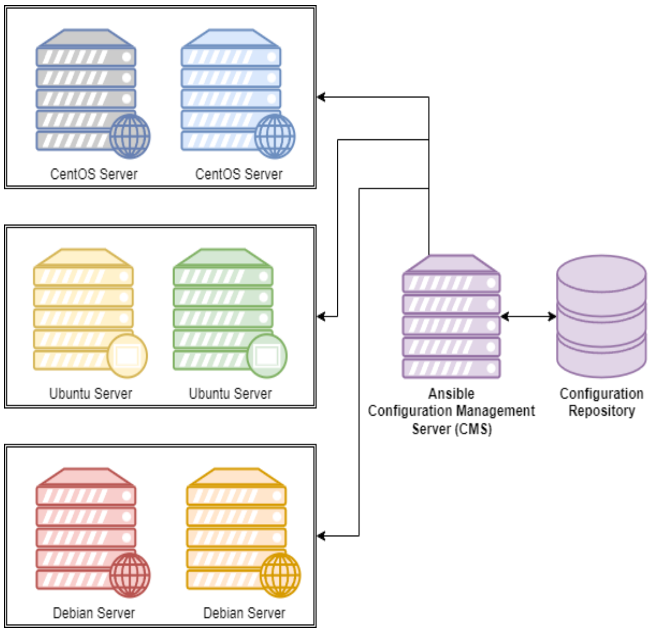

Fig. 6 – Ansible CMS.

Ansible guarantees that tasks are idempotent and will ensure consistency across all systems. Moreover, its rich set of modules can greatly simplify the tasks presented above, particularly surrounding the setup of the databases and the tile server. Furthermore, Ansible will integrate seamlessly with existing infrastructure and tools thanks to its agentless infrastructure. No dependencies must be installed on the targeted machines; the only server requiring setup is the CMS server. Numerous automation types employ disparate methodologies, contingent upon the domain in which the tools are utilized. A comprehensive comparison of automation types is realized in [17]. These include but are not limited to, manufacturing and software deployment.

In the ArchTerr research project context, the automation scripts were selected based on several advantages compared to a commercial off-the-shelf solution. The principal reasons are presented below:

- Commercial tools offer many features in terms of simplicity and efficiency, but they can be intricate to set up and maintain. In contrast, automation scripts offer a more straightforward approach, particularly when tailored to specific tasks.
- In our case, customization was paramount as a research project. Automation scripts allow for the customization of scripts to fit the user's specific needs. In any of the areas above – data analysis, quality assurance, or IT operations – we maintain complete control over the automation process. The scripts permitted the automation process to be tailored to the project's requirements. Commercial tools are constrained in their ability to be customized.
- The cost-effectiveness of automation scripts was also considered. Using open-source languages and libraries resulted in minimal cost, and no licensing fees were incurred. However, it is important to note that any commercial solution will inevitably entail licensing costs, which can accumulate significantly over time.
- The agility and adaptability of automation scripts proved invaluable in facilitating a rapid response to the archeologist team. A significant advantage is the ability to rapidly adapt and modify scripts in response to evolving research requirements. In contrast, commercial tools may require vendor support or updates, which can be slower.
- The project involved creating highly targeted automation scripts (specific use cases) ideal for tasks such as software testing, data analysis, and IT operations and deployment. Commercial solutions often encompass broader functionalities, including many features that are not necessary for our needs.

As a concluding remark, utilizing automation scripts afforded us complete control over the customization process. Although Ansible offers considerable flexibility in terms of customization, its predefined modules and playbooks impose certain limitations.

In its commercial form (the Ansible Tower platform), Ansible has additional costs for enhanced features and support and requires vendor assistance for modifications. Consequently, it is less agile.

5. **CONCLUSIONS**

Due to the need for a nationally mandated information system for archeological preservation, each province must create and maintain its informatics system to handle the documentation and storage of information related to archeological sites.

This creates a financial strain on the province administrations due to the complexity of creating, deploying, and maintaining such a system. Added to this are the licensing costs for the software itself; this cost can be high for GIS software systems.

The ArchTerr information system simplifies adopting a digital information system at the provincial level. Furthermore, by automating the deployment and configuration of ArchTerr, the provincial administration can easily provide the system to its archeological departments at greatly reduced costs compared to other industry solutions such as ESRI and ArcGIS [18].

While the cost of these systems can be justified in some contexts, in the context of an interactive archeological database, a provincial administration may need help justifying the costs of licensing and powerful hardware.

Furthermore, with a built-in coordinate conversion system between WGS84 and STEREO70, the relevant archeological departments will not require additional GIS software. Thus, deploying an instance of the ArchTerr system in each province is a feasible solution that is both cost-saving and easy to deploy.

ACKNOWLEDGMENTS

The Archterr project was supported by the Romanian Ministry of Education and Research scientific grant, CCCDI-UEFISCDI, project number PN-III-P2.2.1-PED-2019-5037 contract PED 571/2020 within the PNCDI III area.

Received on 31 March 2024

REFERENCES

1. I.C. Mustata *et al.*, *The evolution of Industry 4.0 and its potential impact on industrial engineering and management education*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **67**, *1*, pp. 73–78 (2022).
2. I.C. Radu, *Architecture considerations for communities of smart objects*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **67**, *3*, pp. 337–341 (2022).
3. E. Lazarou, C. Mustata, C. Dragomirescu, *Working and learning in industry 4.0 environments*, U.P.B. Sci. Bull., Series D, **81**, *4*, pp. 353–366 (2019).
4. M. Jesi, A. Appathurai, M. Kumaran, A. Kumar, *Load balancing in cloud computing via mayfly optimization algorithm*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **69**, *1*, pp. 79–84 (2024).
5. A. Wittig, M. Wittig, *Amazon Web Services in Action*, Manning Press, p. 93 (2016).
6. T.A. Nitescu, A.I. Concea-Prisacaru, V. Sgarciu, *Test automation for continuous integration in software development*, U.P.B. Sci. Bull., Series C, **84**, *4* (2022).
7. C.V. Marian, M. Iacob, *The ArchTerr project – a GIS-based integrated system for cultural and archaeological heritage protection (pilot phase tested in Romania)*, Applied Sciences, **12**, *16*, 8123 (2022).
8. M. DeMers, *Fundamentals of Geographic Information Systems*, John Wiley & Sons Inc., 2009.
9. B. Morosanu, *Deformațiile liniare relative în sistemele de proiecție Stereografic 1970, Gauss-Krüger, UTM și comparații între acestea*, 24 October 2007 [Online]. Available: https://geo-spatial.org/vechi/articole/deformatii-liniare-in-sistemele-proiectie. [Accessed 30 June 2022].
10. U.S. Dept. of Defense, *Department of Defense World Geodetic System 1984*, National Imagery and Mapping Agency, 2000. Available: https://apps.dtic.mil/sti/pdfs/ADA280358.pdf [Accessed 16 August 2023].
11. V. Agafonkin, *Overview* [Online]. Available: https://leafletjs.com/. [Accessed 18 August 2023].
12. HashiCorp, *Overview* [Online]. Available: https://www.vagrantup.com/ [Accessed 18 August 2023].
13. P. Masek, M. Stusek, J. Krejci, K. Zeman, J. Pokorny, M. Kudlacek, *Unleashing full potential of ansible framework: university labs administration*, 22nd Conference of Open Innovations Association (FRUCT), Jyvaskyla, Finland, pp. 144–150 (2018).
14. I.M. Stan, S.D. Ciocirlan, R. Rughinis, *Understanding the opportunities*

of applying Kubernetes scheduling capabilities in high-performance computing, U.P.B. Sci. Bull., Series C, **84**, *4* (2022).

15. C. Cowan, *Securing Linux systems with AppArmor*, DEF CON, **15**, pp. 15–26 (2007).
16. M.E. Mihailescu, D. Mihai, M. Carabas, N. Tapus, *The perspectives of using Freebsd in cluster architectures*, U.P.B. Sci. Bull., Series C, **86**, 1 (2024).
17. V.G. Dogaru, F.D. Dogaru, V. Navrapescu, L.M. Constantinescu, *Analysis of different types of automation with emphasis on second-life battery implementation*, U.P.B. Sci. Bull., Series C, **86**, *2* (2024).
18. Esri, *About Esri* [Online]. Available: https://www.esri.com/en-us/about/about-esri/overview [Accessed 16 August 2023].