Research on the optimization model for building an efficient IT infrastructure using the AWS platform

Simkin Andrii¹,
Kopp Andrii²,
Olkhovy Oleksii³

¹ Second year Master’s Student; National Technical University «Kharkiv Polytechnic Institute»; Ukraine
² Ph.D. in Computer Science, Associate Professor, Associate Professor of the Department of Software Engineering and Management Intelligent Technology; National Technical University «Kharkiv Polytechnic Institute»; Ukraine
³ Senior Lecturer of the Department of Information Systems and Technologies; National Technical University «Kharkiv Polytechnic Institute»; Ukraine

Abstract. The analysis of IT infrastructure processes has shown that their manual execution without automation can be time-consuming and labor-intensive, as well as increases the risk of errors. Designing an IT infrastructure using Amazon Web Services (AWS) allows to identify the required resources and components for building an efficient infrastructure. Therefore, this study aims to improve the processes of building and managing IT infrastructure by automating them using the AWS platform. Hence, in this paper we propose a mathematical model to determine the IT infrastructure desired configuration. The proposed model assumes solving a linear programming problem with integer constraints on the optimization variables. The Branch-and-Bound algorithm is used to study the proposed mathematical model, by considering different AWS server input data and defining optimal IT infrastructure configurations.

Keywords: IT infrastructure, Process Automation, Infrastructure as Code, Ansible, Amazon Web Services, Optimization Model

This work is distributed under the terms of the Creative Commons Attribution-ShareAlike 4.0 International License (https://creativecommons.org/licenses/by-sa/4.0/).
Introduction. One of the most relevant problems of information technology in recent times is the use of information systems based on cloud technologies. In order to make information systems available to clients any computer or mobile device, cloud platforms can help to solve this problem [1].

One of the most popular cloud platforms is Amazon Web Services (AWS). AWS is the world’s most widely used and feature-rich cloud platform, providing more than 200 full-featured services for data centers around the globe. Millions of customers, including startups, corporations, and government agencies, use AWS to reduce costs, increase flexibility, and accelerate innovation [2].

Thus, the object of this study is the process of automating the building of IT infrastructure on the AWS platform. The subject of the study is a software solution for automating the building of IT infrastructure on the AWS platform. The study aims to improve the processes of building and managing IT infrastructure by automating them using the AWS platform.

Infrastructure as Code. Infrastructure as Code (IaC) is a technique for delivering and managing computing and network resources by describing them in the form of source code, as opposed to configuring the necessary infrastructure manually or using interactive tools. When creating all the resources manually, the probability of making a mistake in several hundred manual operations is close to 100%. Considering these problems, infrastructure automation by using IaC practices is becoming not just a fashion trend, but a necessity [3].

One of the most famous methodologies for working with IaC is “The Twelve-Factor App” [4], by one of the cloud providers, Heroku. The goals of this methodology are the following:

- ensure maximum portability between various environments, reducing the risk of differences and bugs and enabling Continuous Deployment (CD);
- make automation as simple as possible, so that developers do not spend a lot of time getting started on a project.

IaC scripts help to create and manage automated deployment pipelines for rapid software deployment. But similar to software source code, IaC scripts may contain defects. Defects
in the IaC scripts can have dire consequences, for example, GitHub had a DNS failure caused by a defect in the IaC script. Predicting defects in software modules helps the software team to prioritize inspection efforts. Predicting faulty IaC scenarios can help organizations make informed decisions about allocating inspection efforts to those IaC scenarios that are likely to be defective [5].

IaC is one of the fundamental pillars for DevOps implementation, but just like software source code, IaC scripts are susceptible to defects. Defect prediction models for IaC can help software teams prioritize inspection efforts [5].

**IaC-based solutions.** There are already a lot of various Infrastructure as Code (IaC) software solutions that attempt to automate IT infrastructure deployment on the AWS platform. Ansible and Terraform are two of the most powerful and unique open source IT tools that are often compared in competitive discussions. Both tools perform better together and can work in harmony to create a better experience for developers and operations teams.

If compare Ansible [6] and Terraform [7], Ansible has the following advantages:
- easier and faster installation – the biggest advantage of Ansible is its simple and fast setup and installation;
- easier management – Ansible uses YAML as its configuration language, which makes it easier for management even by non-engineers;
- easier to learn – one of the best features of Ansible is that it was designed to be learnable by anyone – while people with a background in DevOps, and programming will be able to learn it faster, literally anyone can master it because it provides such a simple user interface.

Disadvantages of Ansible [6]:
- fewer features (however it releases new features every month);
- poor user interface;
- Ansible does not have a concept of “state” and, accordingly, does not track it;
- weak Windows compatibility;
- weaker support (the cheapest version of Ansible does not provide any support, while the mid-range option only
The Ansible is not as complicated as Puppet [8] and Chef [9], but it provides sufficiently wide capabilities and high management flexibility. Ansible can be used on almost any UNIX-like machine with Python 3.9 or later installed. This includes Red Hat, Debian, Ubuntu, macOS, BSD, and Windows as part of the Windows Subsystem for Linux (WSL). Windows without WSL is not supported as a control node [6].

Ansible, Chef, Puppet, and Terraform (Table 1) were designed to simplify the setup and maintenance of tens, hundreds, or even thousands of servers. But that does not mean that smaller IT companies cannot benefit from using these tools, because automation generally makes it easier to operate any size infrastructure.

<table>
<thead>
<tr>
<th>Features</th>
<th>Chef</th>
<th>Puppet</th>
<th>Ansible</th>
<th>Terraform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Type</td>
<td>Configuration Management</td>
<td>Configuration Management</td>
<td>Configuration Management</td>
<td>Orchestration</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Mutable</td>
<td>Mutable</td>
<td>Mutable</td>
<td>Immutable</td>
</tr>
<tr>
<td>Language</td>
<td>Procedural</td>
<td>Declarative</td>
<td>Procedural</td>
<td>Declarative</td>
</tr>
<tr>
<td>Architecture</td>
<td>Client/Server</td>
<td>Client/Server</td>
<td>Client</td>
<td>Client</td>
</tr>
<tr>
<td>Lifecycle Management</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Virtual Machine Provisioning</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>+</td>
</tr>
<tr>
<td>Networking</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td></td>
</tr>
<tr>
<td>Storage Management</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td></td>
</tr>
<tr>
<td>Packaging</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td></td>
</tr>
<tr>
<td>Templating</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>Partial</td>
</tr>
<tr>
<td>Service Provisioning</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 1

Therefore, Ansible can be used to manage IT infrastructure on AWS or other cloud platforms more easily. However, if more complex IT infrastructure deployment is needed, such as Kubernetes clusters or others, then Terraform should be considered. Nevertheless, Ansible allows to solve such tasks as well.

Problem statement. The rapid development of virtualization, coupled with the increase in server capacity...
that meets industry standards, as well as the availability of cloud computing, has led to a significant increase in the number of servers that need to be managed, both inside and outside the organization [10].

Thus, the analysis of recent research papers [1–3; 5] has shown that configuration management systems are gaining more and more popularity every year as the number of servers that need to be maintained increases.

Therefore, the study of automation in the field of IT infrastructure building is a relevant problem. Configuration management systems available on the market have different capabilities and have certain specifics of operation and require different qualifications of the service personnel.

Hence, this study is intended to facilitate the work of IT/DevOps engineers and administrators involved in building IT infrastructure on the AWS platform. This will help to reduce time and effort, provide more accurate and efficient resource planning, and ensure optimal infrastructure configuration.

Thus, the following research objectives should be addressed to improve the IT infrastructure building and management processes:

1) analyze the processes of building IT infrastructure;
2) develop the mathematical model to determine the desired parameters of the IT infrastructure;
3) implement the solution for automating the building of IT infrastructure on the AWS platform;
4) present and discuss the results of automation of IT infrastructure building.

**IT infrastructure processes.** The processes of building the IT infrastructure on the AWS platform may include such stages as requirements definition, analysis and design, hardware and software procurement, infrastructure deployment, configuration and testing, launch and operation [2].

The difference between processes that are automated with Ansible and AWS [11] and those processes that are not is in the way infrastructure deployment tasks are performed.

Processes without automation assume performing most of the tasks manually. This means that engineers perform various steps and operations to ensure that the IT infrastructure is
built and configured.

For example, manually creating servers, configuring network components, and more. This process can be time-consuming, labor-intensive, and prone to human error.

Fig. 1 shows the deployment process model performed without automation.

In processes are automated with Ansible and AWS [11], part of the tasks are automated using the Ansible tool and the capabilities provided by the AWS platform.

Ansible allows to describe the IT infrastructure and
configuration of applications as code, and then automatically execute this code to deploy and configure various components. This ensures faster and more reliable operations, reduces the risk of errors, and simplifies IT infrastructure management.

Fig. 2 demonstrates the deployment process model performed using automation with Ansible.

Therefore, the main difference is that the automated process uses tools that allow actions to be performed automatically and require less human intervention. This allows to increase the process execution efficiency, reduce the risk of errors, and facilitate IT infrastructure management.

Proposed model. Mathematical modeling is an important step in automating the building of IT infrastructure on the AWS platform. This stage involves the creation of a mathematical model that will optimize and automate the processes of building and managing IT infrastructure.

One of the main tasks of developing a mathematical model is to find an optimal solution to the problem of constructing the IT infrastructure on the AWS platform. This includes determining the set of resources required for the efficient
functioning of the system, allocating these resources, taking into account the constraints and requirements of the project, and maximizing the resource utilization efficiency.

One of the mathematical problems that can be used to develop a mathematical model to automate the construction of IT infrastructure on the AWS platform is linear programming. Linear programming is an optimization method that allows to find the optimal strategy or solution for a problem that has a linear objective function and linear constraints [12].

To determine the required parameters of the IT infrastructure, the following linear programming problem with integer optimization variables is proposed:

\[
D \cdot H \cdot (\sum_{i=1}^{n} c_i \cdot x_i) \rightarrow \min_{x_i = 1,n} \quad (1)
\]

\[
\sum_{i=1}^{n} x_i \geq T, \quad (2)
\]

\[
D \cdot H \cdot (\sum_{i=1}^{n} c_i \cdot x_i) \leq C, \quad (3)
\]

\[
\sum_{i=1}^{n} r_i \cdot x_i \geq R, \quad (4)
\]

\[
\sum_{i=1}^{n} p_i \cdot x_i \geq P, \quad (5)
\]

\[
\sum_{i=1}^{n} s_i \cdot x_i \geq S, \quad (6)
\]

\[
x_i - \text{integer numbers}, i = 1,n, \quad (7)
\]

where

- \(D\) - the number of work days, for a period of one month; \(D = 30\);
- \(H\) - is the number of work hours per day, for continuous operation; \(H = 24\);
- \(n\) - the number of servers considered to build the IT infrastructure;
- \(i\) - the index of the server and its characteristics, varies from 1 to \(n\);
- \(c_i\) - the cost of the \(i\)-th server (USD per hour);
- \(x_i\) - the integer number of the \(i\)-th server instances to be used;
The Simplex Method [12] is an effective algorithm for solving integer linear optimization problems. There are several approaches to using the Simplex Method for integer optimization [12], and one of them is the Branch-and-Bound algorithm.

The basic idea of the Branch-and-Bound algorithm is to find an optimal solution by dividing the space of possible solutions into many smaller subspaces and restricting the search in each subspace [13]. The basic algorithm (Fig. 3) includes the following steps:

1. The standard Simplex Method is used to find a continuous optimal solution to a linear optimization problem.
2. After finding a continuous optimal solution, it is checked whether all variables are integer. If so, this is the optimal solution to the problem.
3. If there is at least one variable that is not an integer, one of these variables is selected, and the problem is branched into two new problems in which constraints are added to exclude invalid values for this variable.
4. For each new problem, steps 1-3 are repeated until the optimal integer solution is found or a certain stopping criterion function (e.g., upper or lower bound) is reached.

This procedure (Fig. 3) generates the tree of branches...
and bounds, where each branch corresponds to a different optimization problem. The algorithm continues to branch and constrain the search space until an optimal integer solution is found or a stopping criterion function is reached.

Figure 3
Branch-and-Bound basic algorithm

**Results and discussion.** After developing a mathematical model, the process of constructing the IT infrastructure on the Amazon Web Services platform can be automated. This process includes steps of automatic resource allocation, configuration of AWS services, security configuration, and infrastructure monitoring. This will help to ensure efficient and reliable IT infrastructure construction, as well as reduce the effort and time required for this process.

To study the model for constructing the optimal IT
infrastructure, let us consider the following initial data of AWS servers (Table 2) [14].

<table>
<thead>
<tr>
<th>Server</th>
<th>Cost, USD/hour</th>
<th>RAM, GB</th>
<th>CPU, GHz</th>
<th>Disk, GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1.micro</td>
<td>0.02</td>
<td>0.612</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>t2.nano</td>
<td>0.0058</td>
<td>0.9</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>t2.micro</td>
<td>0.0116</td>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>t2.small</td>
<td>0.023</td>
<td>2</td>
<td>1</td>
<td>120</td>
</tr>
<tr>
<td>t2.medium</td>
<td>0.0464</td>
<td>4</td>
<td>2</td>
<td>240</td>
</tr>
<tr>
<td>t2.large</td>
<td>0.0928</td>
<td>8</td>
<td>2</td>
<td>480</td>
</tr>
<tr>
<td>t2.xlarge</td>
<td>0.1856</td>
<td>16</td>
<td>4</td>
<td>960</td>
</tr>
<tr>
<td>t2.2xlarge</td>
<td>0.3712</td>
<td>32</td>
<td>8</td>
<td>1920</td>
</tr>
<tr>
<td>t3.nano</td>
<td>0.0052</td>
<td>0.5</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>t3.micro</td>
<td>0.0104</td>
<td>1</td>
<td>2</td>
<td>60</td>
</tr>
</tbody>
</table>

Using the proposed optimization model, let us identify optimal configurations of the IT infrastructure on the AWS platform. The corresponding constraints and obtained suggestions on the IT infrastructure configuration are illustrated in Table 3.

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
<th>Option 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum number of servers</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Maximum cost per month, USD</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Minimum amount of RAM, GB</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Minimum CPU clock speed, GHz</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Minimum disk size, GB</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Solutions</td>
<td>1 × t3.micro</td>
<td>2 × t3.nano</td>
<td>3 × t3.nano</td>
<td>2 × t3.micro</td>
</tr>
</tbody>
</table>

To conduct a test load for Option 1 (Table 3) with the single “t3.micro” server configuration, we used 100 clients...
requesting a website page at 2-second intervals for 5 minutes. By analyzing the resulting graphs (Fig. 4 and Fig. 5), we noticed that the server could not handle the CPU load.

![Figure 4](image1.png)

**Figure 4**

CPU utilization monitoring for 1 × t3.micro server configuration

![Figure 5](image2.png)

**Figure 5**

CPU usage monitoring for 1 × t3.micro server configuration

After identifying the problem with CPU load processing in Option 1 (Table 3), we decided to scale the IT infrastructure horizontally. Hence, we considered Option 2 (Table 3), which had the different server configuration – two "t3.nano" instances.

After running the test load for Option 2 with the specified parameters, the obtained results (Fig. 6 and Fig. 7) indicate the success of horizontal scaling in the context of automating the construction of IT infrastructure on the AWS platform.
Depending on the load, it becomes obvious that at least 2 servers are required for such load parameters. If more clients need to be processed, the number of servers should be increased to ensure that there is a capacity margin for optimal performance. This is an important step in solving the load processing problem, as reducing the CPU load improves system performance and ensures more efficient resource utilization.

Using the graphs shown in Fig. 8 and Fig. 9, let us demonstrate how the use of horizontal scaling has led to a 40% reduction in CPU load.
Options 3 and 4 (Table 3) suggest server characteristics and number of servers for horizontal scaling. Hence, Fig. 8 and Fig. 9 show how different combinations of servers can provide the necessary performance and capacity to ensure reliability and scalability in the process of automating the construction of IT infrastructure on the Amazon Web Services platform. This allows to deploy the optimized and efficient IT infrastructure that meets business needs and ensures the successful implementation of various IT projects, especially in critical domains where availability is vital.

**Conclusion.** As a result of this study, the existing research on the problem of a software solution for automating the construction of IT infrastructure on the AWS platform was analyzed. Also, the existing software solutions aimed at
automating the process of deploying the IT infrastructure in the cloud were studied and their main advantages and disadvantages were identified and discussed.

We analyzed the processes of constructing the IT infrastructure and found that their manual execution without automation is time-consuming and labor-intensive, as well as increasing the risk of errors.

Designing a target IT infrastructure using AWS allows to identify the necessary resources and components for efficient operations. Hence, the proposed mathematical model helps to determine the optimal configuration for automating the deployment of IT infrastructures on the AWS platform, which makes it easier to configure and manage IT solutions. Also, determining the optimal IT infrastructure parameters helps to ensure system performance and reliability by reducing the CPU utilization and usage by 40% as the performed experiments have demonstrated.

There are many benefits of implementing IT process automation on the AWS platform, including reducing the time and cost of building IT infrastructure, improving process accuracy and consistency, simplifying management and support, and ensuring velocity and scalability. This allows to increase the efficiency and flexibility of the IT infrastructure, which is important for business success.

The findings of this study may be useful for organizations planning to automate their IT processes on the AWS platform.

References:


