Secure Systems Design and Technology Development. Veliko Tarnovo, Bulgaria
Proceedings of the International Scientific Conference, 2026. Volume III, 13-17

An Approach for Preventing Overload of the EC2 Service

Evgeni Andreev
Department of Information Technologies
Nikola Vaptsarov Naval Academy
Varna, Bulgaria
e.andreev@naval-acad.bg

Desislava Valcheva
Department of Information Technologies
Nikola Vaptsarov Naval Academy
Varna, Bulgaria
d.valcheva@naval-acad.bg

***Abstract*— The aim of this article is to present an approach for preventing service overload in Elastic Compute Cloud (EC2) within Amazon Web Services (AWS). The study analyzes methods for preventing overload of virtual machines in AWS EC2 by comparing vertical and horizontal scaling. Developed block diagrams are presented to visually illustrate the architecture and processes involved in dynamic resource management. A practical implementation of horizontal scaling of virtual machines is also provided. For this purpose, a Python script using the boto3 library has been developed, which fully automates the configuration of key cloud components such as Launch Template, Auto Scaling Group, and Scaling Policy. The proposed solution minimizes the need for manual intervention, protects cloud-based systems from failures during peak loads, and ensures the development of a resilient, flexible, and cost-effective infrastructure. Compared to the manual configuration through the AWS console, the proposed approach groups all required steps into a single Boto3 script that creates the Launch Template, the Auto Scaling Group and the Scaling Policy in one run. The block diagrams visualise both scaling workflows in a unified form, and the scripted procedure can be executed on a schedule through cron, which makes the solution suitable for use in the training of cadets and students at the Nikola Vaptsarov Naval Academy.**

Keywords*— *Auto Scaling, Boto3 library, Cloud technologies, Virtual machines.

I. INTRODUCTION

Cloud technologies offer flexible solutions based on the "on-demand service" principle and are widely adopted thanks to their efficiency, scalability, and high availability [1], [2], [3]. Among the main advantages of cloud services are the low initial investment, high availability, and easy scalability of resources. Cloud services enable rapid deployment, offer flexible pricing based on actual usage, ensure greater reliability through automatic updates and recovery from failures, and provide for shared responsibility regarding security between the user and the provider [1], [2], [4], [5].

The use of AWS for creating virtual machines (instances) makes it possible to operate a variety of operating systems that support the processing and storage of data [6], [7]. Through these virtual machines, different types of services can be launched for use either in a local environment or across the public Internet. The virtualization services available in AWS can be automatically scaled both vertically and horizontally, which is of fundamental importance for the design of modern network architectures that require specific analysis of the data exchange protocols [8].

The present article extends the already published AWS configuration procedures in three directions. First, vertical and horizontal scaling in EC2 are compared and described through original block diagrams (Figs. 4 to 6), which summarize the configuration workflow in a form that is not available in this consolidated form in the official documentation. Second, a Boto3 script is developed which performs the creation of the Launch Template, the Auto Scaling Group and a CPU-based Scaling Policy in a single execution, replacing the multi-step procedure that has to be carried out manually through the AWS console as described in [7]. Third, automation will be included in practical training for cadets and students at the Nikola Vaptsarov Naval Academy, in which the script will be executed on a schedule and used as a training configuration, not as a one-time configuration task.

II. MATERIALS AND METHODS

In this work, both horizontal and vertical scaling approaches for active instances are applied. Their use requires the creation of virtual images of pre-installed and pre-configured AWS services - EC2, Auto Scaling and Launch Templates [7]. For access to a Linux-based instance, the secure Secure Shell Protocol (SSH) is used together with the PuTTY client application. To add additional protection (a password) to the private key, PuTTYgen is applied.

Online ISSN 3044-7771
https://doi.org/10.68302/std2026.vol3.118

The automatic creation of virtual machines for preventing system overload and failure through automation of the Auto Scaling service is implemented in the Python programming language using the Boto3 library [7].

III. RESULTS AND DISCUSSION

A. Building a virtual image and accessing a Linux machine

Creating a virtual machine image involves seven steps, as shown in Fig. 1:

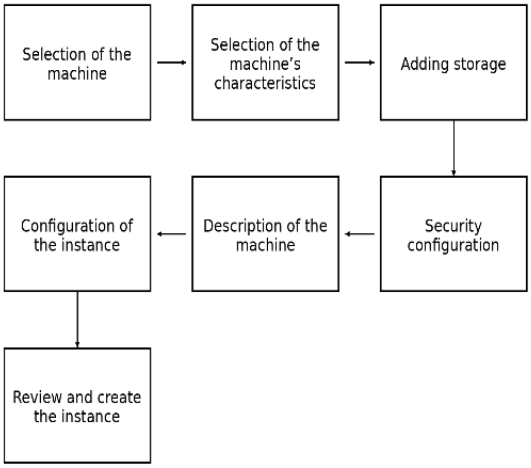

Fig. 1. Sequence for creating a virtual machine image.

One of the key steps in creating an instance is the generation of a cryptographic key pair — public and private — which ensures secure remote access through protocols such as SSH and Remote Desktop Protocol (RDP) [3], [6]. The public key is generated and stored on the AWS side and may be associated with one or more instances, whereas the private key is downloaded only once by the user and remains under their control. Access to the instance can be performed only by the user holding the corresponding private key.

The private key may be in Privacy Enhanced Mail (PEM) or PuTTY Private Key (PPK) format. The PPK format is used directly for SSH access through software such as PuTTY, while the PEM format is used with Windows-based instances to decrypt the initial administrator password. After successful use in the AWS "Password recovery" service and entering the private key, the username and password for accessing the virtual machine are displayed. The PEM and PPK formats can be converted to add additional protection through a passphrase, which raises the level of security [3], [9]. This functionality is implemented through the PuTTYgen tool. Access to virtual machines is granted only when both a valid private key and the corresponding passphrase, if configured, are provided.

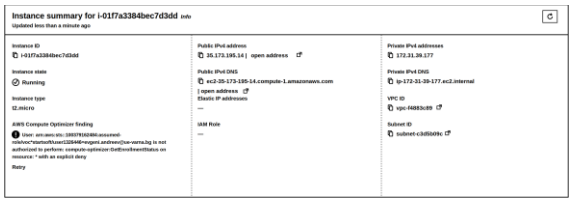

Fig. 2. Linux instance created in AWS with the main instance details shown.

Fig. 2 shows access to a Linux-based instance, which is performed via the SSH protocol using a client application such as PuTTY. When the remote connection between the client and the virtual machine is established, a private key is presented and is validated against the corresponding public key associated with the instance; when the keys match, administrative access is granted. After the session is established, the user can manage the operating system. Before installing additional services or configurations, it is recommended to update the operating system using the command: "sudo yum update -y". Fig. 3 presents the result of executing the command sudo yum update -y.

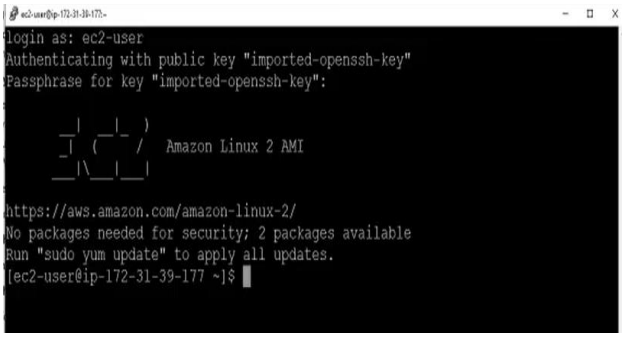

Fig. 3. Running Linux instance with the distribution updated.

B. Building and accessing a virtual image of a Windows Server 2025 machine

Creating a Windows Server 2025 instance is analogous to creating a Linux-based instance (Fig. 3); the main difference is the protocol used for remote access — RDP [3].

Different software tools may be used for remote management of the instance, regardless of the client operating system in use (Windows / Windows Server, Linux-based systems, or macOS). For example, with Windows-based virtual images the Remote Desktop Connection application included with the Windows operating system is used by default.

C. Scalability of cloud systems through horizontal and vertical scaling

a) Vertical scaling approach

In vertical scaling, a single instance, or a limited number of instances, is used; instead of adding or removing instances, the computing resources of the existing ones (CPU, RAM, etc.) are increased. This approach typically involves higher costs, as larger resource configurations are paid for regardless of the current load. In practice, situations are often observed in

which the available resources are insufficient under high instance load, or, conversely, a significant share of them remain unused at low request intensity, which leads to inefficiency from an economic point of view. Increasing resources usually requires temporarily stopping the instance, which causes a brief service interruption. In addition, decreasing resources after they have been increased is often limited or even impossible in certain configurations. Although this approach provides scalability, it is characterized by greater complexity and higher costs; nevertheless, it remains the preferred option in several scenarios [6], [10].

b) Horizontal scaling approach

Another AWS service used for automatic scaling is Auto Scaling. It relies on a previously prepared template for creating virtual machines. The basic requirements for creating new virtual machines are the template name, and the choice of initial cloud computing resources such as CPU cores, RAM, and Internet connection bandwidth. Further requirements for creating the template include defining scalability parameters based on increasing or decreasing the average load of the virtual CPU cores. For example, the following criterion may be selected: if the average load exceeds 60%, a new virtual machine is to be created. The new instance can reduce the load on a previously created virtual machine by distributing the load between the two machines through the load-balancing service, which is also provided by AWS. Other criteria for creating new machines may include high inbound or outbound network traffic, and others [11].

The approach of preventing service failure through automation of Auto Scaling is referred to as horizontal because it allows multiple virtual machines to be created in order to compensate for the load [7], [12]. These machines have identical baseline settings in terms of computing power, based on the previously created template. For this functionality to operate correctly, three values are defined in the template:

- Minimum number of machines required by the customer to be running.
- Desired number of machines to be running when the service is first activated.
- Maximum number of machines to be running under peak load.

In this way, the service administrator can decide how many machines should run as the minimum, calculate the financial resources to be allocated, and determine the maximum number of machines accordingly. A larger number of machines than the chosen maximum cannot be created automatically. The advantages of this type of automatic horizontal scaling can be illustrated by the following example: if a user of the AWS service selects a desired count of two machines, a minimum of one and a maximum of three, then two machines will be created based on the prepared template. If, however, there is insufficient load on them, the count will be automatically reduced to one. If the load exceeds the values defined in the template, new machines will be created automatically up to the selected maximum threshold. For automating scaling, the horizontal approach is more appropriate than the vertical one, because the load is distributed across many machines with smaller computing capabilities.

Creating a launch template for instances:

Fig. 4 visualizes the process of creating a single, uniformly configured virtual machine with specific parameters, which is subsequently used as a template for launching identical machines.

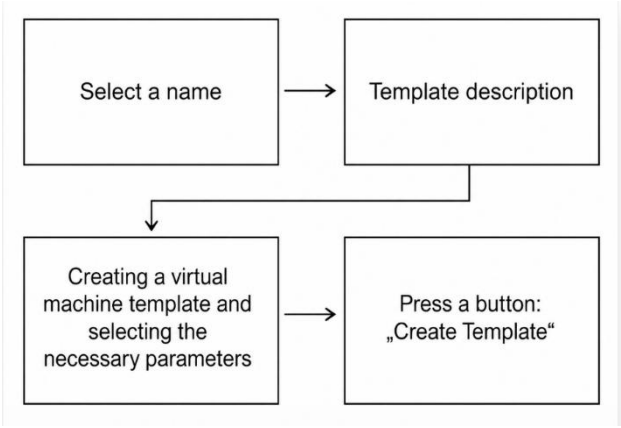

Fig. 4. The process of creating a template with parameters chosen by the user.

Creating a launch template for identical instances is part of configuring the automatic scaling service.

Configuring the Auto Scaling service

Configuring the automatic scaling service involves seven steps, as shown in Fig. 5:

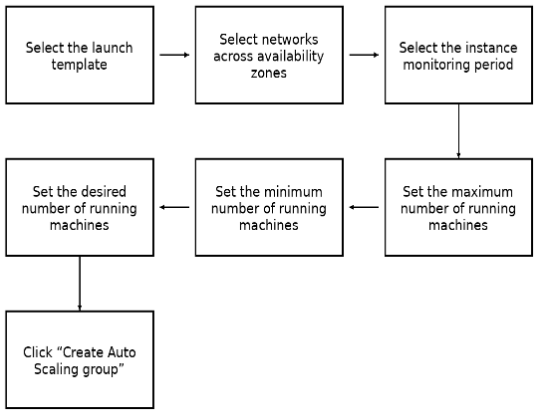

Fig. 5. The process of configuring automatic scaling of machines.

This procedure for configuring the automatic scaling service takes more time than using Software Development Kit (SDK) code in AWS. In this paper, a solution to the scaling automation task is proposed using the Python programming language and the Boto3 library.

D. An approach to automating Auto Scaling Group using the Boto3 library

Automating the Auto Scaling Group using the boto3 library enables interaction with AWS services. Examples of applications of Boto3 include the creation of virtual machines, storage, automatic creation of virtual machines, and others [7], [10].

The block diagram visualized in Fig. 6 shows the steps for creating a script that automates the Auto Scaling Group.

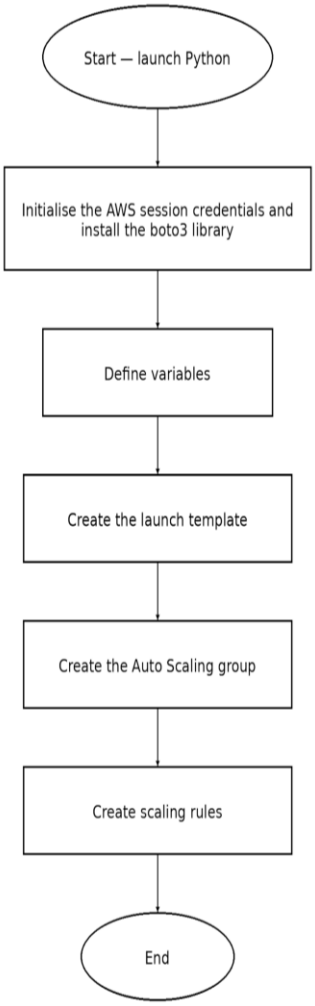

Fig. 6. Steps for creating a script that automates the Auto Scaling Group.

The script automates the creation of a Launch Template, an Auto Scaling Group and a Scaling Policy in AWS through the boto3 library. At the start, clients are initialized for EC2 and Auto Scaling. Next, the basic parameters are defined (AMI ID, instance type, key, zones, etc.). An EC2 Launch Template is created and is used to launch instances. After that, an Auto Scaling Group is created with a minimum and a maximum number of instances. Finally, an automatic scaling policy is added that increases the number of instances on demand. An example of such automation is shown in Fig. 7.

Fig. 7 presents the code of a script for creating a Launch Template and an Auto Scaling Group in AWS with automatic scaling at 50% CPU utilisation of the machines. Including a cron job [13] for executing the script on a schedule will improve the efficiency of repetitive tasks in AWS.

The presented methods for EC2 automation - with a graphical interface and using a Python script, will be used to create an automated training system for cadets and students at the Nikola Vaptsarov Naval Academy.

```
import boto3

REGION = "eu-central-1"
VPC_ID = "vpc-xxxxxxxx"

ec2 = boto3.client("ec2", region_name=REGION)
asg = boto3.client("autoscaling", region_name=REGION)
ssm = boto3.client("ssm", region_name=REGION)

ami = ssm.get_parameter(
    Name="/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64"
)["Parameter"]["Value"]

subs = [
    s["SubnetId"]
    for s in ec2.describe_subnets(
        Filters=[{"Name": "vpc-id", "Values": [VPC_ID]}]
    )["Subnets"][:3]
]

lt = ec2.create_launch_template(
    LaunchTemplateName="demo-lt",
    LaunchTemplateData={
        "ImageId": ami,
        "InstanceType": "t3.micro",
    },
)["LaunchTemplate"]

asg.create_auto_scaling_group(
    AutoScalingGroupName="demo-asg",
    MinSize=1,
    DesiredCapacity=2,
    MaxSize=3,
    VPCZoneIdentifier=",".join(subs),
    LaunchTemplate={
        "LaunchTemplateId": lt["LaunchTemplateId"],
        "Version": str(lt["LatestVersionNumber"]),
    },
)

asg.put_scaling_policy(
    AutoScalingGroupName="demo-asg",
    PolicyName="cpu50",
    PolicyType="TargetTrackingScaling",
    TargetTrackingConfiguration={
        "PredefinedMetricSpecification": {
            "PredefinedMetricType": "ASGAverageCPUUtilization"
        },
        "TargetValue": 50.0,
    },
)

print("DONE")
```

Fig. 7. Python (boto3) script for creating a Launch Template and an Auto Scaling Group.

CONCLUSIONS

The analysis carried out in the present paper shows that vertical scaling allows the computing resources of existing instances to be increased, but it is associated with higher costs, limited flexibility, and the need for a temporary service interruption. Horizontal scaling, in turn, implemented through Auto Scaling, ensures better resilience and effective distribution of the resource load through the dynamic addition and removal of virtual machines according to current needs.

The presented automation approach using the boto3 library and the Python programming language demonstrates the possibilities for building a flexible and adaptive cloud infrastructure. By defining a Launch Template, an Auto Scaling Group and Scaling policies, automatic resource management is achieved, which minimizes the risk of instance overload and failure. The use of automated mechanisms, including the execution of tasks through a cron job, additionally improves efficiency and reduces the need for manual intervention.

The contribution of the present work is the unification of the Launch Template, an Auto Scaling Group and a Scaling Policy into a single Boto3 script with the goal of using this script within a training environment for cadets and students. Further work is planned in two directions. The first one is to extend the automation with load testing based on different scaling policies. The second one is to introduce indicators for the learning outcomes when the script is applied as a laboratory exercise.

In conclusion, combining horizontal and vertical scaling, together with the automation of processes, represents an effective strategy for building reliable, scalable, and economically optimised cloud systems that meet today's requirements for the high availability and performance of virtual machines.

REFERENCES

[1] T. Erl, R. Puttini, and Z. Mahmood, Cloud Computing: Concepts, Technology & Architecture. Upper Saddle River, NJ, USA: Prentice Hall, 2014. ISBN: 978-0133387520.

[2] S. Cambric and M. T. Ratemo, Cloud Auditing Best Practices: Perform Security and IT Audits across AWS, Azure, and GCP by Building Effective Cloud Auditing Plans. Birmingham, UK: Packt Publishing, 2023. ISBN: 978-1803243771.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, NIST Special Publication 800-145, 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf [Accessed: Apr. 25, 2026], https://doi.org/10.6028/NIST.SP.800-145.

[4] D. Carter, CCSP Certified Cloud Security Professional All-in-One Exam Guide, 3rd ed. New York, NY, USA: McGraw-Hill, 2022. ISBN: 978-1264842209.

[5] C. Bravo and D. Kitchen, Mastering Defensive Security: Effective Techniques to Secure Your Windows, Linux, IoT, and Cloud Infrastructure. Birmingham, UK: Packt Publishing, 2021. ISBN: 978-1800208162.

[6] C. Dotson, Practical Cloud Security: A Guide for Secure Design and Deployment. Sebastopol, CA, USA: O'Reilly Media, 2019. ISBN: 978-1492037514.

[7] Amazon Web Services, "AWS Documentation." [Online]. Available: https://docs.aws.amazon.com/ [Accessed: Apr. 25, 2026].

[8] D. Dimitrova, "An analytical model for multi-criteria evaluation of IoT application layer protocols," in Proc. 8th Int. Symp. Innovative Approaches in Smart Technologies (ISAS), Istanbul, Turkey, 2024. IEEE. ISBN: 9798331540104, https://doi.org/10.1109/ISAS64331.2024.10845381.

[9] D. Dimitrova, "Evaluating the effectiveness of practical exercises in cryptography and authentication courses of cybersecurity education," Mathematics and Education in Mathematics, vol. 54, pp. 118–124, 2025, https://doi.org/10.55630/MEM.2025.54.118-124.

[10] R. Buyya, C. Vecchiola, and S. T. Selvi, Mastering Cloud Computing. New York, NY, USA: McGraw-Hill Education, 2013.

[11] D. Dimitrova, "Selection and justification of criteria for comparative analysis of lightweight ciphers," Mathematics and Informatics, vol. 66, no. 5, pp. 534–542, 2023, https://doi.org/10.53656/math2023-5-8-sel.

[12] "What is Amazon EC2 Auto Scaling?" [Online]. Available: https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html [Accessed: Apr. 25, 2026].

[13] "Cron-job.org." [Online]. Available: https://cron-job.org/en/ [Accessed: Apr. 25, 2026].