

Научная статья

УДК 681.3.06

DOI:10.31854/1813-324X-2023-9-3-68-73



# Development of an Accessibility Testing System for the Virtual Machine Deployment Service in the Cloud

Andrey Marchenko✉, mar4enko.ag@gmail.com

Dmitry Shchemelinin, dshchmel@gmail.com

Intel Corporation,  
Santa Clara, United States of America

**Abstract:** This paper presents a developed system for testing the accessibility of the virtual machine deployment service in the cloud. The methods used for monitoring cloud systems based on open-source solutions such as Kubernetes, Prometheus, and Selenium are discussed. The key stages of the system design are described, including requirements gathering and analysis, architecture, and implementation features. This system allows for the prevention of potential issues before they arise, as well as increasing IT service reliability. Integration with open-source systems helps to reduce the cost of system development and operation and speeds up implementation time. Additionally, this system can be quickly adapted and customized to meet specific needs. To analyze the feasibility of building the system, production statistics of virtual machine reservations were collected using the Prometheus monitoring system.

**Keywords:** information technology, monitoring, information processing, monitoring metrics, cloud technologies, application testing, Prometheus, Selenium, Kubernetes, Cronjobs, Python

**For citation:** Marchenko A., Shchemelinin D. Development of an Accessibility Testing System for the Virtual Machine Deployment Service in the Cloud. *Proc. of Telecommun. Univ.* 2023;9(3):68–73. DOI:10.31854/1813-324X-2023-9-3-68-73

## Разработка системы тестирования доступности сервиса развертывания виртуальных машин в облаке

Андрей Геннадьевич Марченко✉, mar4enko.ag@gmail.com

Дмитрий Александрович Щемелинин, dshchmel@gmail.com

Корпорация Intel,  
Санта-Клара, Соединенные Штаты Америки

**Аннотация:** В данной статье представлена разработанная система тестирования доступности сервиса развертывания виртуальных машин в облаке. Рассмотрены методы, используемые для мониторинга облачных систем на основе решений с открытым исходным кодом (от англ. Open source), такие как Kubernetes, Prometheus и Selenium. Описаны ключевые этапы проектирования системы, а именно сбор и анализ требований, архитектура, а также особенности реализации. Данная система позволяет предупредить возможные проблемы до их возникновения, а также повысить надежность сервиса. Интеграция с системами на основе открытых исходных кодов позволяет уменьшить стоимость разработки и экс-

платации системы, а также ускорить сроки внедрения. Данная система также может быть быстро адаптирована и настроена с учетом конкретных потребностей. Для анализа целесообразности построения системы была собрана производственная статистика резервирований виртуальных машин с помощью системы мониторинга Prometheus.

**Ключевые слова:** информационные технологии, мониторинг, обработка информации, метрики мониторинга, облачные технологии, тестирование приложений, Prometheus, Selenium, Kubernetes, Cronjobs, Python

**Ссылка для цитирования:** Марченко А.Г., Щемелинин Д.А. Разработка системы тестирования доступности сервиса развертывания виртуальных машин в облаке // Труды учебных заведений связи. 2023. Т. 9. № 3. С. 68–73. DOI:10.31854/1813-324X-2023-9-3-68-73

## Introduction

Cloud providers offer various types of services, such as hosting virtual machines, data storage, databases, analytics, and more. Typically, access to these services is provided both through a web interface and an API (Application Programming Interface) – a set of interfaces for software integration with services via various protocols. Monitoring the availability and performance of cloud systems and services is one of the most important aspects of ensuring their stable operation, as well as identifying and fixing issues that arise during operation. Specialized software is used to monitor the availability of cloud services. One popular open-source solution is the Prometheus monitoring service [1]. This monitoring system allows you to collect and store performance and availability data in the form of time series – sequences of values that change over time. The collected data can be analyzed using various methods, such as visualization, statistical analysis, and machine learning.

In this study, a new method for testing and monitoring a cloud service was proposed. This service allows users to rent virtual machines in the cloud and choose one of the offered types of virtual machines with a specified amount of RAM, processors, and storage. This type of service is very popular for both enterprises and individual users and allows significant resources to be saved by providing compute resources on demand.

## The object of scientific research

The object of the research is the internal cloud platform of the Intel company [2], which allows developers and researchers to use computing resources to test

and optimize their applications and algorithms. The platform also provides extensive capabilities for managing infrastructure for computing and efficient resource management and is used for various tasks, such as processing big data and machine learning. Intel is one of the world's leading manufacturers of electronic devices and computer components, including microprocessors and chipsets for client computing systems and data centers, chips for artificial intelligence systems and the Internet of Things, and non-volatile memory.

One of the key services of the cloud platform is VMaaS (Virtual Machine as a Service), which provides virtual infrastructure. By analyzing the reservation statistics of virtual machines within the internal cloud (Figure 1), one can infer that there is a continuous increase in the demand for virtual infrastructure, thereby necessitating a greater emphasis on ensuring the service's stability and reliability.

The aim of the research is to develop a method and system for testing the availability of virtual machine deployment service in the cloud based on Prometheus and Selenium [3], characterized by the following scientific novelty:

- Implementation of a service for testing virtual machine deployment in the cloud;
- Integration of Prometheus and Selenium tools with the deployment service and defining scenarios for monitoring and testing cloud resources;
- A new method for collecting and analyzing data on availability using Prometheus and Selenium, including identifying performance issues and optimizing the operation of the virtual machine deployment service in the cloud.

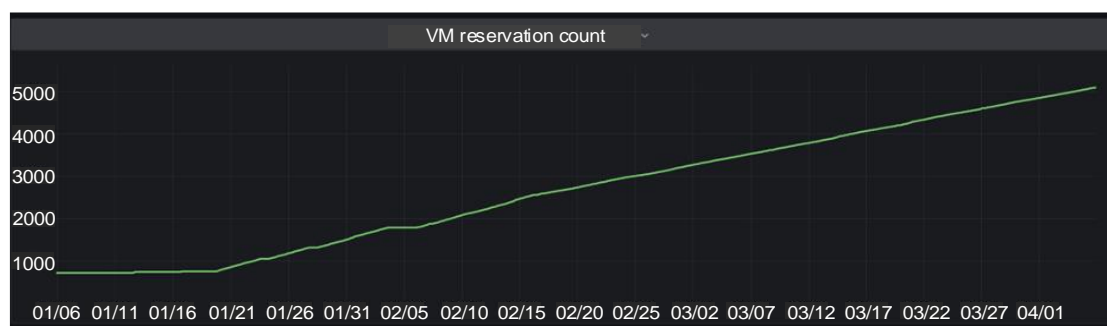


Fig. 1. The Number of Reservations for Virtual Machines in the VMaaS System from 06th January 2023 to 01st April 2023

## Research Methods

The main functional requirement for testing a cloud subsystem is the ability to create and delete a virtual machine with specified characteristics such as memory, processor, and disk size. As a part of this research a user interface was chosen for interaction with the testing system. This allowed not only to ensure the maximum correspondence of the test scenario with the end user behavior in the system, but also to organize the collection of metrics for analyzing the quality of services that provide functionality, with the aim of identifying potential problems before users report them.

The following functional requirements were formulated for the developed subsystem:

- 1) Implementation of login to the cloud system;
- 2) Navigation through the interface, localization, and selection of the virtual machine deployment service;
- 3) Creation and deletion of virtual machines;
- 4) Checking the readiness status of the virtual machine;
- 5) Calculation of key service metrics:
  - Tracking login time;
  - Tracking virtual machine creation time;
  - Tracking the overall scenario execution time.

When choosing key metrics, the need for further predicting the service availability using mathematical models and event forecasting algorithms to achieve business goals was considered [4–6]. Creating a virtual machine through the user interface is a process with many steps, each of which must be considered from the perspective of potential errors and their corresponding handling, including ensuring reliable and stable system behavior in case the task unexpectedly fails.

Therefore, the following non-functional requirements were formulated:

- Running the scenario at a specified frequency and schedule;
- Automatically retrying unsuccessful operations during scenario execution. Configure the waiting interval between retries and the number of retries;
- Global time limit for the entire scenario. If the scenario exceeds the specified limit, it should be forcibly terminated.

To implement the test scenario with the formulated functional and non-functional requirements, the Python programming language, and the selenium-python library [7] based on the Selenium WebDriver were chosen. Python is a widely used language [8] that allows applications of varying complexity to be implemented in a short time. It has simple and understandable syntax, lack of additional performance requirements, and ease of debugging and testing make it an ideal candidate for this research.

Selenium WebDriver [9], together with the selenium-python library, is widely used for testing web ap-

plications and allows for emulating user interaction with the browser.

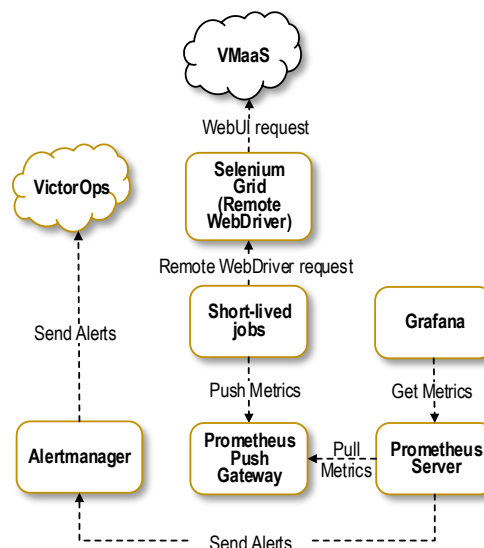


Fig. 2. System Architecture

Based on the formulated functional and non-functional requirements, a corresponding architecture was developed (Figure 2). Kubernetes, an open-source software for automating deployment, scaling, and management of containerized applications, was chosen as the platform for implementation, along with a special type of Kubernetes resource – Cronjob. This type of resource enables the scheduling and execution of tasks at specific times, controlling the schedule and frequency, managing retries in case of errors, and setting a global limit on task execution. The Selenium Grid subsystem [10] was used to interact with the user interface of the virtual machine deployment cloud service. This component provides a platform for remote script execution using WebDriver, enabling parallel execution of test scenarios by running them on different instances of remote browsers. VMAaS is a cloud service that provides a platform for deploying virtual machines. Prometheus was chosen as the main system for collecting and storing metrics, which also handles metric processing and notification generation. Alertmanager processes notifications generated by Prometheus and forwards them to VictorOps. This component manages and processes incidents in real-time and allows for notifying relevant teams in case of problems. Prometheus Pushgateway serves as an intermediate layer between short-lived tasks and the main Prometheus server. This component is necessary because Prometheus operates on a pull model where it periodically scrapes target objects for metric collection via a short interval, while tasks run on their own schedule, and for efficient metric collection, a push model needs to be implemented. During task execution, the task sends an HTTP (Hypertext transfer protocol) request to Prometheus Pushgateway, which temporarily stores metrics and serves as the target object for Prometheus. To create and configure graphs,

diagrams, and other metric visualization panels, Grafana [11] was along with Prometheus.

To evaluate the availability of the virtual machine deployment service, key metrics were selected. The primary metric was chosen as the user login time, which is the time during which the user is authenticated and authorized using registered credentials. The measurement was taken from the moment the login form appeared until the user was shown the main screen with the ability to launch a virtual machine. The following `time_monitor` function was proposed for calculating time metrics.

```
@contextmanager
def time_monitor(
    self, metric name: str, start time:
Optional[datetime] = None
):
    start_time = start_time or datetime.now()
    try:
        yield
    finally:
        self.metrics[metric name] =
(datetime.now() - start_time).seconds
```

The usage of this function for the login process is as follows:

```
wrapped_login = retry(
    on exception=WebDriverException,
    to exception=LoginError,
    attempts=self.login_retry_count,
    sleep interval=self.login_retry_interval
)(self.login)
with self.time_monitor('login'):
    wrapped_login()
```

The next key metric selected was the instance launch time, which determines the acceptable limit for the time we expect the virtual machine to be ready.

```
with self.time_monitor('launch_instance',
start_time):
    self.driver.find_element(
        By.XPATH,
        "//button[@class='btn btn-primary' and
text() = 'Launch Instance']"
    ).click()
    logger.info(f'Instance launched:
{start_time}')
    self.wait_running_instance()
```

If this limit is exceeded, the script will terminate with an error.

```
deadline = datetime.now() + timedelta(
seconds=self.wait_deadline_seconds)
while True:
    if datetime.now() > deadline:
        raise VmProvisioningTimeout(
            f'VM provisioning exceeds wait dead-
line.'
            '{self.wait_deadline_seconds}'
        )
```

The last key metric identified was the total time it took to create a virtual machine, including processing expected exceptional situations, repeating operations, waiting for the machine to be ready, and deleting it. Machine deletion is an important part of completing

the testing scenario, as all resources allocated during the process must be freed up.

The virtual machine launch time metric was analyzed, and a threshold value was determined, which was used in Prometheus to create notifications. To select the optimal method for determining the threshold value, it was necessary to check if the distribution was normal. To do this, the Shapiro-Wilk test was used [12]. The basis of the test is to check the null hypothesis – the data is normally distributed. The alternative hypothesis is that the data does not have a normal distribution. The result of the Shapiro-Wilk criterion is a statistical value and a p-value. In most cases, the threshold value is taken as 0.05. If the p-value is less than 0.05, the null hypothesis is rejected. The SciPy library of the Python language was used for implementation. The input data was an array of virtual machine launch time metrics from Prometheus over the past 7 days.

```
from scipy.stats import shapiro
from datetime import timedelta, datetime
from prometheus import PrometheusClient

p = PrometheusClient (PROMETHEUS ADDRESS,
PROMETHEUS USER, PROMETHEUS PASSWORD)

query = 'vm test provisioning seconds'
end = datetime.now()
start = end - timedelta(days=7)
step = 600 # 10 min

input_list =
p.make_range_query(start.timestamp(),
end.timestamp(), step, query)
statistic, p_value = shapiro(input_list)

print('Statistic:', statistic)
print('p-value:', p_value)

alpha = 0.05
if p_value > alpha:
    print('fail to reject H0')
else:
    print('reject H0')
```

As a result, a low p-value of 2.098965063816978e-31 was obtained, based on which the null hypothesis was rejected, and it was concluded that the data did not have a normal distribution.

To calculate the threshold value, the following method was used – the sum of the median and three absolute deviations [13]:

$$\text{Threshold} = \text{Median} + 3 * \text{MAD}$$

The median of the data was calculated using the Numpy library in Python, and the mean absolute deviation was computed.

```
import numpy as np
median = np.median(input_list)
mad = np.median(np.abs(input_list - median))
threshold = median + 3 * mad

print("Median: ", median)
print("MAD: ", mad)
print("Threshold: ", threshold)
```

As a result, the median was calculated to be 131.5, the mean absolute deviation was 12.5, and the threshold value was determined to be 169.

## Results and discussion

A system that collects various key parameters of a cloud virtual machine deployment service was developed, based on a real-world scenario that simulates user behavior.

After analyzing the collected data (Figure 3) using the method of summing the median and three absolute deviations, a threshold value of 169 was calculated, which can be used in Prometheus in an expression describing a potential issue in the virtual machine deployment service. Exceeding this threshold indicates a deviation from the expected values and requires immediate escalation and technical investigation of the causes by the relevant support team [14].



Fig. 3. Virtual Machine Launch Time

## Conclusion

The main result of the work is the developed system for testing the availability of virtual machines and the integration method into the monitoring system of the internal cloud service of Intel company. The main test case was proposed and implemented – creating and deleting a virtual machine using the user interface, and trigger activation criteria were defined. During the operation of cloud environments using this system, problematic areas in the architecture of the virtual machine creation service were identified, which allowed for timely optimization of the cloud system's operation. By analyzing the collected data, problems with the performance of the web interface were identified and eliminated, which significantly improved the user experience of using this interface. The developed system allows for deviations in the behavior of the service to be recorded in a number of key parameters, such as login time and virtual machine launch time. This significantly reduced the response time to incidents. The use of free open-source technologies such as Prometheus and Selenium contributes to increased efficiency and reduced costs for servicing cloud services. The described method is an effective way of testing cloud services and can also be used to analyze and improve reliability and availability – the most important criteria for 24/7 business applications.

tified and eliminated, which significantly improved the user experience of using this interface. The developed system allows for deviations in the behavior of the service to be recorded in a number of key parameters, such as login time and virtual machine launch time. This significantly reduced the response time to incidents. The use of free open-source technologies such as Prometheus and Selenium contributes to increased efficiency and reduced costs for servicing cloud services. The described method is an effective way of testing cloud services and can also be used to analyze and improve reliability and availability – the most important criteria for 24/7 business applications.

## References

1. Prometheus. Monitoring system & time series database URL: <https://prometheus.io/docs/introduction/overview> [Accessed 04th April 2023]
2. Official intel web site. URL: <https://www.intel.com> [Accessed 04th April 2023]
3. Selenium. The selenium browser automation project. URL: <https://www.selenium.dev/documentation> [Accessed 04th April 2023]
4. Shchemelinin D. Mathematical Models and Methods for Monitoring and Predicting the State of Globally Distributed Computing Systems. *Proc. of Telecom. Universities*. 2021;7(3):73–78. (in Russ.) DOI:10.31854/1813-324X-2021-7-3-73-78
5. Shchemelinin D. A method for predicting events in globally distributed computing complexes. *Modern Science: Current Issues in Theory and Practice. Series: Natural Technical Sciences*. 2021;12-2:47–54. (in Russ.) DOI:10.37882/2223-2966.2021.12-2.16
6. Shchemelinin D. Method and algorithm for automatic recovery of information services based on objective predictive monitoring data. *Modern Science: Current Issues in Theory and Practice. Series: Natural Technical Sciences*. 2021;8:140–144. (in Russ.) DOI:10.37882/2223-2966.2021.08.41

7. Selenium with Python. URL: <https://selenium-python.readthedocs.io> [Accessed 04th April 2023]
8. The TIOBE Programming Community index an indicator of the popularity of programming languages. URL: <https://www.tiobe.com/tiobe-index> [Accessed 04th April 2023]
9. Raghavendra S. Python Testing with Selenium: Learn to Implement Different Testing Techniques Using the Selenium WebDriver. Berkeley: Apress; 2020. 196 p.
10. Selenium. When to Use Selenium Grid. URL: <https://www.selenium.dev/documentation/grid/applicability> [Accessed 04th April 2023]
11. Grafana Labs. Grafana documentation. URL: <https://grafana.com/docs/grafana/latest> [Accessed 04th April 2023]
12. Razali N.M., Wah Y.B. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*. 2011;2(1):21–33. URL: <https://www.nrc.gov/docs/ML1714/ML17143A100.pdf> [Accessed 04th April 2023]
13. Leys C, Ley C., Klein O., Bernard P., Licata L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*. 2013;49(4):764–766. URL: <https://www.sciencedirect.com/science/article/pii/S0022103113000668> [Accessed 04th April 2023]
14. Shchemelinin D. System of criteria and algorithm of information processing and decision-making for the software module for displaying the most significant monitoring events in the information system. *XXI century: Results of the Past and Challenges of the Present plus*. 2021;10(3):67–71. (in Russ.) DOI:10.46548/21vek-2021-1055-0012

#### Список источников


1. Monitoring system & time series database // Prometheus. URL: <https://prometheus.io/docs/introduction/overview> [Accessed 04th April 2023]
2. Official intel web site. URL: <https://www.intel.com> [Accessed 04th April 2023]
3. The selenium browser automation project // Selenium. URL: <https://www.selenium.dev/documentation> [Accessed 04th April 2023]
4. Щемелинин Д.А. Математические модели и методы мониторинга и прогнозирования состояния глобально распределенных вычислительных комплексов // Труды учебных заведений связи. 2021. Т. 7. № 3. С. 73–78. DOI:10.31854/1813-324X-2021-7-3-73-78
5. Щемелинин Д.А. Метод прогнозирования событий в глобально распределенных вычислительных комплексах. Современная наука: актуальные вопросы теории и практики. Серия: Естественные и технические науки. 2021. № 12-2. С. 47–54. DOI:10.37882/2223-2966.2021.12-2.16
6. Щемелинин Д.А. Метод и алгоритм автоматического восстановления информационных сервисов на основе объективных предиктивных данных мониторинга. Современная наука: актуальные вопросы теории и практики. Серия: Естественные и технические науки. 2021. № 8. С. 140–144. DOI:10.37882/2223-2966.2021.08.41
7. Selenium with Python. URL: <https://selenium-python.readthedocs.io> [Accessed 04th April 2023]
8. The TIOBE Programming Community index an indicator of the popularity of programming languages. URL: <https://www.tiobe.com/tiobe-index> [Accessed 04th April 2023]
9. Raghavendra S. Python Testing with Selenium: Learn to Implement Different Testing Techniques Using the Selenium WebDriver. Berkeley: Apress, 2020. 196 p.
10. When to Use Selenium Grid // Selenium. URL: <https://www.selenium.dev/documentation/grid/applicability> [Accessed 04th April 2023]
11. Grafana documentation // Grafana Labs. URL: <https://grafana.com/docs/grafana/latest> [Accessed 04th April 2023]
12. Razali N.M., Wah Y.B. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests // *Journal of Statistical Modeling and Analytics*. 2011. Vol. 2. No. 1. PP. 21–33. URL: <https://www.nrc.gov/docs/ML1714/ML17143A100.pdf> [Accessed 04th April 2023]
13. Leys C, Ley C., Klein O., Bernard P., Licata L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median // *Journal of Experimental Social Psychology*. 2013. Vol. 49. Iss. 4. PP. 764–766. URL: <https://www.sciencedirect.com/science/article/pii/S0022103113000668> [Accessed 04th April 2023]
14. Щемелинин Д.А. Система Критериев и алгоритм обработки информации и принятия решений для программного модуля отображения наиболее значимых событий мониторинга в информационной системе // XXI век: итоги прошлого и проблемы настоящего плюс. 2021. Т. 10. № 3(55). С. 67–71. DOI: 10.46548/21vek-2021-1055-0012

Статья поступила в редакцию 20.02.2023; одобрена после рецензирования 27.02.2023; принята к публикации 25.03.2023.


The article was submitted 20.02.2023; approved after reviewing 27.02.2023; accepted for publication 25.03.2023.

## Информация об авторах:

**МАРЧЕНКО**  
Андрей Геннадьевич

архитектор «облачных» приложений Software and Advanced Technology Group, корпорация Intel  
 <https://orcid.org/0009-0001-9276-3907>

**ЩЕМЕЛИНИН**  
Дмитрий Александрович

доктор технических наук, вице-президент Software and Advanced Technology Group, корпорация Intel  
 <https://orcid.org/0000-0003-3032-130X>