Программные системы и вычислительные методы
Правильная ссылка на статью:
Марченко А.Г., Щемелинин Д.А. — Разработка автоматизированной системы тестирования облачного сервиса развертывания виртуальных машин с использованием современных средств мониторинга // Программные системы и вычислительные методы. – 2023. – № 2. DOI: 10.7256/2454-0714.2023.2.40755 EDN: TEVFBN URL: https://nbpublish.com/library_read_article.php?id=40755

Разработка автоматизированной системы тестирования облачного сервиса развертывания виртуальных машин с использованием современных средств мониторинга

Марченко Андрей Геннадьевич

ORCID: 0009-0001-9276-3907

Архитектор облачных программных систем, Компания Интел

193318, Россия, Санкт-Петербург область, г. Санкт-Петербург, ул. Бадаева, 6к1

mar4enko.ag@gmail.com

Щемелинин Дмитрий Александрович

ORCID: 0000-0003-3032-130X

доктор технических наук

Вице-президент, Компания Интел

195251, Россия, г. Санкт-Петербург, ул. Политехническая, 29

dshchmel@gmail.com

Статья из рубрики "Телекоммуникационные системы и компьютерные сети"

DOI:

10.7256/2454-0714.2023.2.40755

EDN:

TEVFBN

Дата направления статьи в редакцию:

15-05-2023

Аннотация: Объектом данного исследования является сервис по управлению виртуальными машинами в облачной среде. При разработке и эксплуатации такого сервиса возникает необходимость оценки его доступности и надежности на соответствие выбранному уровню качества, на который может рассчитывать клиент. В данной работе представлена разработанная система, позволяющая тестировать доступность облачного сервиса по управлению виртуальными машинами. Рассмотрен метод интеграции с

существующей системой мониторинга на предприятии с использованием открытого программного обеспечения с целью уменьшения стоимости разработки и эксплуатации. Разработан и реализован тест-кейс по развертыванию и удалению виртуальной машины с использованием графического интерфейса пользователя, а также определены критерии срабатывания триггеров. Собраны и проанализированы требования к архитектуре и реализации системы на основе производственной статистики сервиса создания виртуальных машин с помощью системы мониторинга Prometheus. Новизна исследования заключается в разработке нового метода тестирования облачного сервиса по управлению виртуальными машинами с целью повышения его надежности и доступности. На основе данного метода описана и реализована система тестирования виртуальных машин а также метод интеграции в систему мониторинга облачного сервиса компании Intel. В процессе эксплуатации облачных сред с помощью данной системы были выявлены проблемные места в архитектуре сервиса создания виртуальных машин, что позволило своевременно оптимизировать работу системы. Описанный метод является эффективным способом тестирования облачных сервисов, и также может использоваться для анализа и повышения надежности и доступности.

Ключевые слова:

информационные технологии, мониторинг, prometheus, selenium, kubernetes, python, тестирование приложений, метрики, обработка информации, пороговое значение

Введение

Облачные провайдеры предоставляют различные виды услуг, такие как размещение виртуальных машин, хранение данных, базы данных, аналитику и многое другое. Как правило, доступ к этим услугам осуществляется как через веб интерфейс, так и через API (англ. Application Programming Interface) – набор готовых интерфейсов для программной интеграции с сервисами посредством различных протоколов. Мониторинг доступности и производительности облачных систем и сервисов является одним из важнейших аспектов обеспечения их стабильной работы, а также позволяет выявлять и исправлять проблемы возникающие в процессе эксплуатации. Для мониторинга доступности облачных сервисов используется специализированное программное обеспечение. Одним из популярных решений с открытым исходным кодом является сервис Prometheus [1]. Эта система мониторинга позволяет собирать и хранить данные о производительности и доступности в виде временных рядов – последовательности значений, которые изменяются во времени. Собранные данные в дальнейшем можно анализировать с использованием различных методов, таких как визуализация, статистический анализ и машинное обучение.

В данном исследовании был предложен новый метод тестирования и мониторинга облачного сервиса, предоставляющего пользователю возможность арендовать виртуальные машины в облаке, а также выбрать один из предложенных типов виртуальной машины с заданным объемом оперативной памяти, процессоров и хранилища. Данный тип сервиса очень популярен как для предприятий, так и для индивидуальных пользователей и позволяет сэкономить значительные ресурсы предоставляя вычислительные мощности по запросу.

Объект научного исследования

Объектом исследования является внутренняя облачная платформа компании Intel [2],

которая позволяет разработчикам и исследователям использовать ресурсы вычислительных систем для тестирования и оптимизации своих приложений и алгоритмов. Платформа также предоставляет широкие возможности по управлению инфраструктурой для вычислений, а также эффективному управлению ресурсами и используется для выполнения различных задач, таких как обработка больших данных и машинное обучение. Компания Intel является одним из ведущих мировых производителей электронных устройств и компьютерных компонентов: микропроцессоров и наборов системной логики (чипсетов) для клиентских вычислительных систем и для дата-центров, чипов для систем искусственного интеллекта и для интернета вещей, энергонезависимой памяти.

Одним из ключевых сервисов облачной платформы является VMaaS (англ. Virtual Machine as a Service), предоставляющий виртуальную инфраструктуру. Изучив статистику резервирований виртуальных машин во внутреннем облаке (рис 1), можно сделать вывод о том, что спрос на виртуальную инфраструктуру постоянно растет, и следовательно, повышаются требования к стабильности и надежности сервиса.

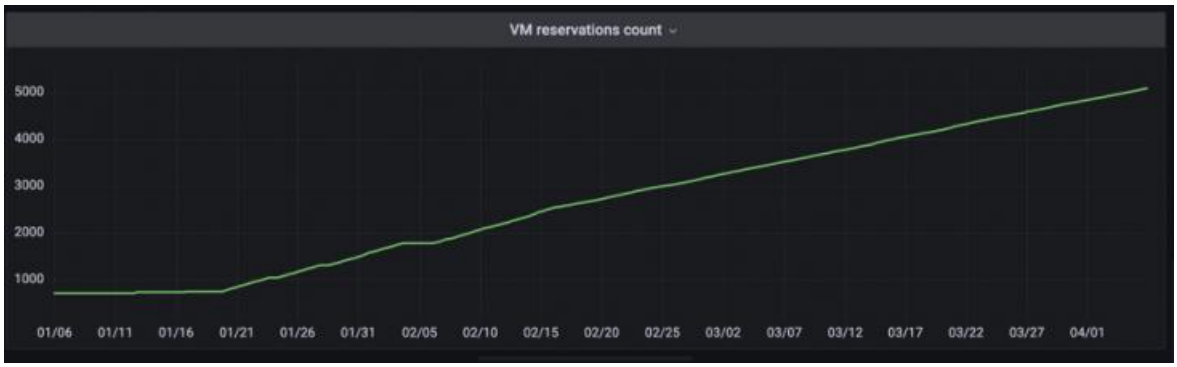

Рисунок 1. Количество резервирований виртуальных машин в системе VMaaS с 06.01.2023 по 01.04.2023

Цель исследования

Предметом данного исследования является разработка метода и системы тестирования доступности сервиса развертывания виртуальных машин в облаке на базе Prometheus и Selenium [3], отличающиеся следующей научной новизной:

- Реализация сервиса тестирования развертывания виртуальной машины в облаке
- Интеграция инструментов Prometheus и Selenium с сервисом развертывания и определение сценариев мониторинга и тестирования облачных ресурсов
- Новый метод сбора и анализа данных о доступности с помощью Prometheus и Selenium, включая определение проблем с производительностью и оптимизация работы сервиса развертывания виртуальных машин в облаке

Методы Исследования

Основным функциональным требованием для подсистемы тестирования является возможность создания и удаления виртуальной машины с заданными характеристиками – память, процессор, объем диска.

Для взаимодействия с системой был выбран пользовательский интерфейс. Это позволило не только обеспечить максимальное соответствие тестового сценария с поведением конечного пользователя в системе, но и организовать сбор метрик для анализа качества сервисов, обеспечивающих функционал, целью которого является выявление потенциальных проблем до того, как о них начнут сообщать пользователи.

Были сформулированы следующие функциональные требования к разработанной подсистеме:

- Реализация входа в облачную систему
- Навигация по интерфейсу, локализация и выбор сервиса развертывания виртуальной машины
- Создание и удаление виртуальной машины
- Проверка статуса готовности виртуальной машины
- Вычисление ключевых метрик сервиса:
  - Отслеживание времени входа в систему
  - Отслеживание времени создания виртуальной машины
  - Отслеживание времени работы всего сценария

При выборе ключевых метрик была учтена необходимость дальнейшего прогнозирования доступности сервиса с применением математических моделей и алгоритмов прогнозирования событий для достижения целевых бизнес-показателей [4, 5, 6].
Также были рассмотрены нефункциональные требования. Задача создания виртуальной машины с помощью пользовательского интерфейса — это процесс с большим количеством шагов, каждый из которых необходимо рассматривать с точки зрения возникновения потенциальных ошибок и их соответствующей обработки, включая обеспечение надежного и стабильного поведения системы в случае, если задача неожиданно завершается с ошибкой. Следовательно, были сформулированы следующие нефункциональные требования:

- Запуск сценария с заданной периодичностью и по расписанию
- Автоматический повтор неуспешной операции в ходе выполнения сценария. Предусмотреть конфигурацию интервала ожидания между повторами, и количество повторов
- Глобальный лимит времени на весь сценарий. В случае если сценарий не укладывается в отведенный лимит, он должен быть завершен принудительно

Для реализации тестового сценария с учетом сформулированных функциональных и нефункциональных требований был выбран язык программирования Python и библиотека selenium-python [7] на основе Selenium WebDriver. Язык Python очень распространен [8] и позволяет в короткие сроки реализовывать приложения разного уровня сложности. Простой и понятный синтаксис, отсутствие дополнительных требований к производительности, а также простота отладки и тестирования делает его идеальным кандидатом для данного исследования. Selenium WebDriver [9] вместе с библиотекой selenium-python широко используется в целях тестирования веб-приложений и позволяют эмулировать взаимодействие пользователя с браузером.

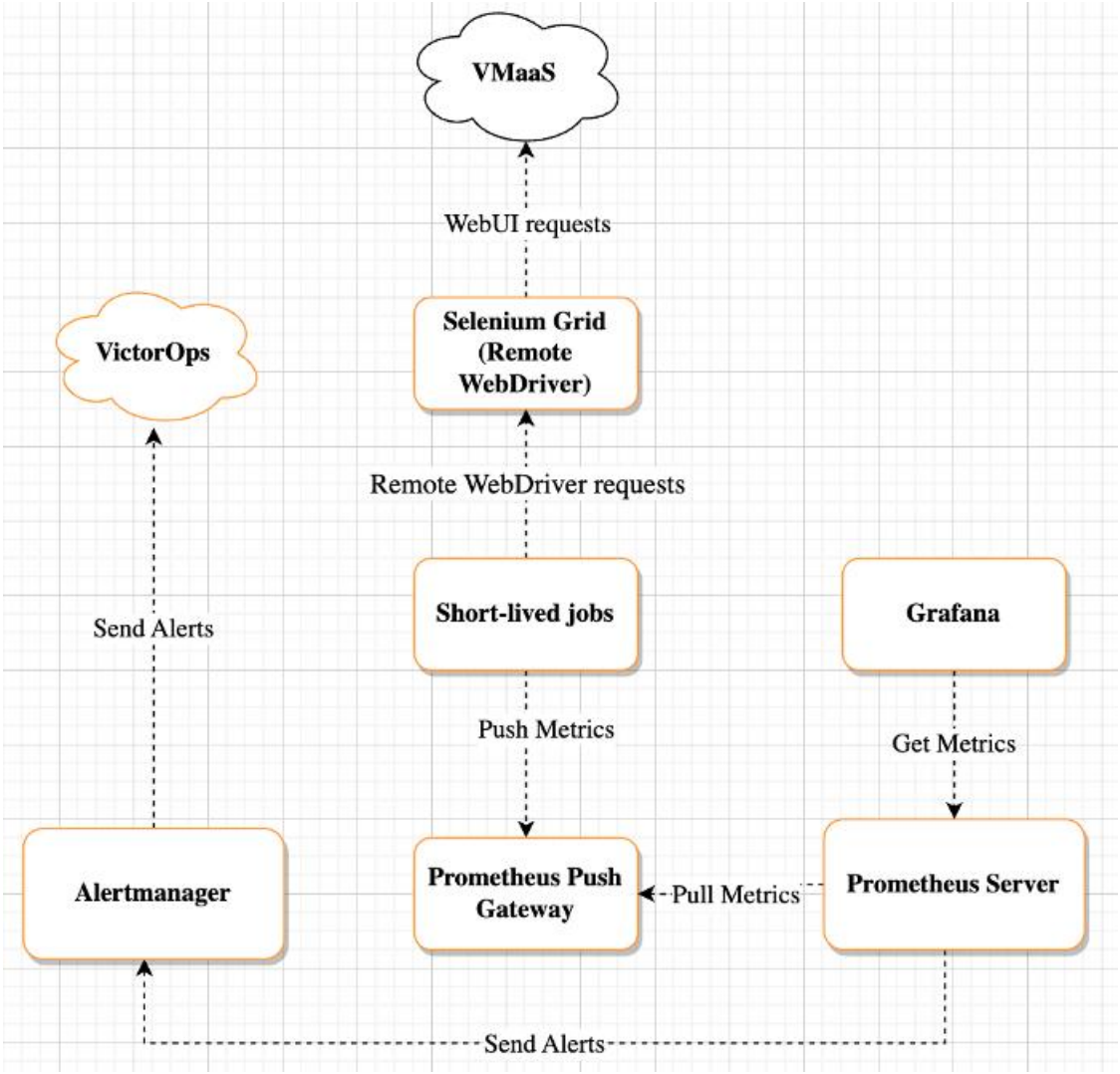

Рисунок 2. Архитектура системы

На основе сформулированных функциональных и нефункциональных требований была разработана соответствующая архитектура (рис. 2). В качестве платформы для реализации был выбран Kubernetes – открытое программное обеспечение для автоматизации развертывания, масштабирования и управления контейнеризированными приложениями, а также специальный тип Kubernetes ресурса – периодическая задача (англ. Cronjob). Данный тип ресурса позволяет планировать и запускать задачи в определенное время, контролировать расписание и периодичность, управлять повторными запусками в случае ошибок, а также задавать глобальный лимит на выполнение задачи. Для взаимодействия с пользовательским интерфейсом облачного сервиса развертывания виртуальных машин была использована подсистема Selenium Grid [10]. Данный компонент представляет собой платформу для удаленного выполнения скриптов с использованием WebDriver. С его помощью обеспечивается параллельный запуск тестовых сценариев путем выполнения их на разных экземплярах удаленных браузеров. VMaaS – это облачный сервис, который предоставляет платформу для развертывания виртуальных машин. В качестве основной системы сбора и хранения метрик был выбран Prometheus, который также отвечает за обработку метрик и генерацию уведомлений. Alertmanager – обрабатывает уведомления, сгенерированные Prometheus и пересылает их в VictorOps. Данный компонент отвечает за управление и обработку инцидентов в режиме реального времени и позволяет оповещать соответствующие команды в случае возникновения проблем. Prometheus Pushgateway - это промежуточный слой между короткоживущими задачами и основным сервером

Prometheus. Данный компонент необходим, так как особенностью работы Prometheus является периодический опрос целевых объектов для сбора метрик через короткий интервал - pull модель (англ. pull model), в то время как задачи запускаются по собственному расписанию и для эффективного сбора метрик необходимо реализовать push модель (англ. Push model). Задача в процессе выполнения отправляет HTTP (англ. Hypertext transfer protocol) запрос в Prometheus Pushgateway, который временно сохраняет метрики и является целевым объектом для Prometheus. Для создания и настройки графиков, диаграмм и других панелей визуализации метрик, совместно с Prometheus был использован сервис Grafana [11].

Для оценки доступности сервиса развертывания виртуальных машин были выбраны ключевые метрики. За основную метрику было выбрано время входа пользователя в систему. Это время, в течение которого происходит аутентификация и авторизация пользователя, с использованием зарегистрированных учетных данных. Замер был осуществлен с момента появления формы ввода учетных данных, до момента, когда пользователю был показан главный экран с возможностью запуска виртуальной машины. Для подсчета временных метрик было предложено воспользоваться следующей функцией time_monitor.

```
@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
```

Использование данной функции для процесса входа в систему (англ. login) выглядит следующим образом:

```
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()
```

В качестве следующей ключевой метрики было выбрано время запуска виртуальной машины (англ. launch instance). Именно оно определяет допустимый предел, в течение которого мы ожидаем готовность виртуальной машины.

```
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()
```

В случае если данный предел будет превышен, сценарий завершится с ошибкой.

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

Последней ключевой метрикой было обозначено полное время работы сценария создания виртуальной машины, включающее в себя обработку ожидаемых исключительных ситуаций, повторы операций, ожидание готовности машины и последующее ее удаление. Удаление машины является неотъемлемой частью завершения сценария тестирования, так как все ресурсы выделенные в процессе работы должны быть освобождены.

Была проанализирована метрика времени запуска виртуальной машины и определено пороговое значение, которое было использовано в Prometheus для создания нотификаций.

Для выбора оптимального метода определения порогового значения необходимо проверить, является ли распределение нормальным. Для этого было предложено применить тест Шапиро-Уилка [12]. В основе проверки лежит проверка нулевой гипотезы – данные распределены нормально. Альтернативная гипотеза – данные не имеют нормального распределения. Результат критерия Шапиро-Уилка — это значение статистики, а также значение p-value. В качестве порогового значения в большинстве случаев берется 0.05. При значении p-value меньше 0.05 нулевая гипотеза отклоняется. Для реализации была использована библиотека SciPy языка python. В качестве входных данных был использован массив метрик времени запуска виртуальной машины из Prometheus за последние 7 дней.

```
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')
```

В результате было получено низкое значение p-value 2.098965063816978e-31 на основе которого нулевая гипотеза была отклонена и сделан вывод о том, что данные не имеют нормального распределения. Для вычисления порогового значения (англ. Threshold) был использован следующий метод – сумма медианы и трех абсолютных отклонений [13]

Threshold = Median + 3 * MAD

С помощью библиотеки Numpy языка Python была вычислена медиана данных и рассчитано среднее абсолютное отклонение.

```
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)
```

В качестве результата было рассчитана медиана 131.5, среднее абсолютное отклонение 12.5 и пороговое значение 169.

Результаты и обсуждение. Была разработана система позволяющая собирать различные ключевые параметры облачного сервиса развертывания виртуальных машин, на основе реального сценария имитирующего работу пользователя.

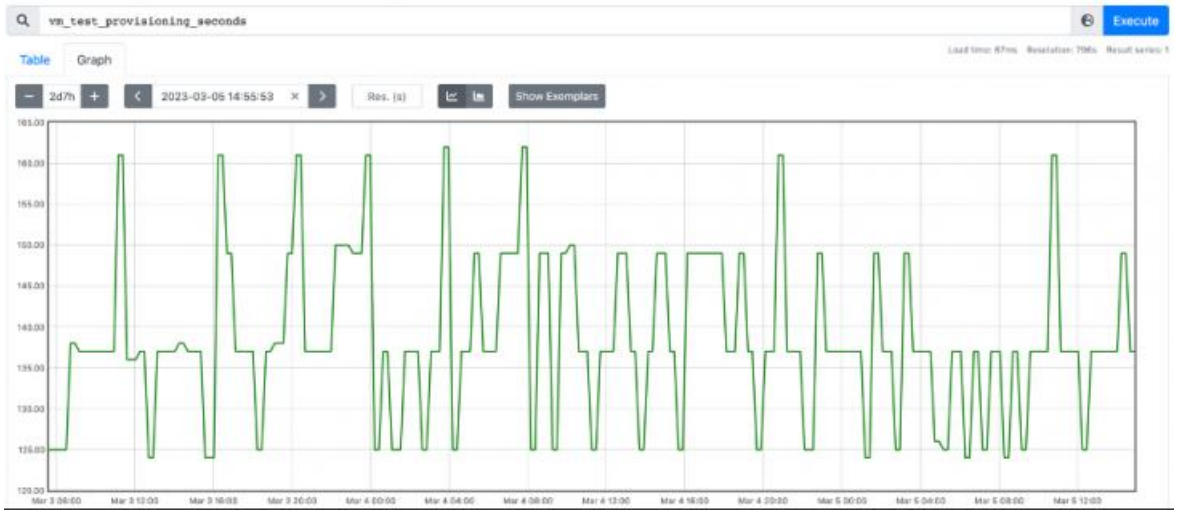

Рисунок 3. Время запуска виртуальной машины

Проанализировав собранные данные (рис. 3) методом суммирования медианы и трех абсолютных отклонений, было вычислено пороговое значение 169, которое можно использовать в Prometheus в выражении описывающим потенциальную проблему в сервисе развертывания виртуальных машин. Превышение данного порога сигнализирует о значительном отклонении от ожидаемых значений, и требует немедленной эскалации и технического расследования причин возникновения со стороны профильной команды поддержки сервиса [14].

Выводы

Основным результатом работы является разработанная система тестирования доступности виртуальных машин и метод интеграции в систему мониторинга внутреннего облачного сервиса компании Intel. Был предложен и реализован основной тест-кейс – создание и удаление виртуальной машины с помощью графического интерфейса пользователя, определены критерии срабатывания триггеров. В процессе эксплуатации облачных сред с помощью данной системы были выявлены проблемные места в архитектуре сервиса создания виртуальных машин, что позволило своевременно оптимизировать работу системы. Путем анализа собранных данных были выявлены и устранены проблемы с производительностью веб-интерфейса, что позволило существенно улучшить пользовательский опыт в использовании данного интерфейса. Разработанная система позволяет регистрировать отклонения в поведении сервиса по ряду ключевых параметров, таких как время входа в систему и время запуска

виртуальной машины. Благодаря этому удалось значительно сократить время реакции на возникающие инциденты. Использование бесплатных технологий с открытым исходным кодом, таких как Prometheus, Selenium способствует повышению эффективности и снижению затрат компании на обслуживание облачных сервисов. Описанный метод является эффективным способом тестирования облачных сервисов, и также может использоваться для анализа и повышения надежности и доступности – важнейших критериев для бизнес-приложений, работающих в режиме 24/7

Библиография

1. Prometheus – Monitoring system & time series database [Электронный ресурс]. URL: https://prometheus.io/docs/introduction/overview/ (дата обращения 04.04.2023).
2. Официальный Интернет-сайт Intel [Электронный ресурс]. URL: https://www.intel.com/ (дата обращения 04.04.2023).
3. The selenium browser automation project [Электронный ресурс]. URL: https://www.selenium.dev/documentation/ (дата обращения 04.04.2023).
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.
5. Щемелинин Д.А. Метод прогнозирования событий в глобально распределенных вычислительных комплексах. Современная наука: актуальные проблемы теории и практики. Серия: Естественные технические науки. 2021. № 12–2. С. 47–54.
6. Щемелинин Д.А. Метод и алгоритм автоматического восстановления информационных сервисов на основе объективных прогностических данных мониторинга. Современная наука: актуальные проблемы теории и практики. Серия: Естественные технические науки. 2021. № 8. С. 140–144.
7. Selenium with python [Электронный ресурс] URL: https://selenium-python.readthedocs.io/ (дата обращения 04.04.2023).
8. The TIOBE Programming Community index an indicator of the popularity of programming languages [Электронный ресурс] URL: https://www.tiobe.com/tiobe-index/ (дата обращения 04.04.2023).
9. Sujay Raghavendra. – Python testing with selenium – Apress Berkeley CA, 2020 – 4 с. ISBN 978-1-4842-6249-8
10. When to use selenium grid [Электронный ресурс] URL: https://www.selenium.dev/documentation/grid/applicability/ (дата обращения 04.04.2023)
11. Grafana documentation [Электронный ресурс]. URL: https://grafana.com/docs/grafana/latest/ (дата обращения 04.04.2023)
12. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. Journal of Statistical Modeling and Analytics, Vol.2, No. I, 21-33, 2011. URL: https://www.nrc.gov/docs/ML1714/ML17143A100.pdf (дата обращения 04.04.2023)
13. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. Journal of Experimental Social Psychology, 49(4), 764–766. URL: https://www.sciencedirect.com/science/article/pii/S0022103113000668 (дата обращения 04.04.2023)
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.

Результаты процедуры рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Предмет исследования. С учётом указанного наименования научной статьи она должна быть посвящена разработке автоматизированной системы тестирования облачного сервиса развертывания виртуальных машин с использованием современных средств мониторинга. Содержание статьи соответствует заявленной теме.

Методология исследования базируется на разработке системы тестирования доступности виртуальных машин и метод интеграции в систему мониторинга внутреннего облачного сервиса компании Intel. Причём, исследование проведено с использованием Python, что формирует положительное впечатление от качества выбранной методологии. Очень ценно, что по тексту статьи автор активно использует графический метод представления результатов научных исследований, при этом при доработке статьи следует подписать наименования у всех рисунков (в т.ч. относящиеся к коду) и указать источник данных (если разработано автором, то следует так и указать).

Актуальность исследования вопросов, связанных с практическим использованием информационных технологий и достижений научно-технического прогресса, не вызывает сомнения, т.к. это позволяет обеспечить научное обоснование мероприятий по обеспечению технологического суверенитета Российской Федерации, о необходимости которого было заявлено Президентом России.

Научная новизна в представленных на рецензирование материалах содержится и крайне чётко определена: связана с разработанной системой тестирования доступности виртуальных машин и метод интеграции в систему мониторинга внутреннего облачного сервиса компании Intel.

Стиль, структура, содержание. Стиль изложения научной. Статья имеет достаточно чётко выстроенную структуру, что формирует положительное впечатление от ознакомления с нею. Рекомендуется дополнить блоком «Предложения по практическому использованию авторской разработки». Наполнение содержанием данного структурного элемента обеспечит рост интереса к данной научной публикации в геометрической прогрессии. Ценно, что статья базируется на конкретных результатах, полученных автором.

Библиография. Автором сформирован библиографический список из 14 источников. Однако не изучены научные публикации 2022-2023 гг., в т.ч. в зарубежных изданиях. Это позволило бы учесть самые последние тенденции в научной литературе по рассматриваемым вопросам.

Апелляция к оппонентам. Несмотря на сформированный список литературы и, в целом, высокое качество научной статьи, в т.ч. связанное с наличием конкретных авторских суждений и разработок, не осуществлено их обсуждение с итогами исследований, отражённых в трудах других авторов. При доработке статьи необходимо уделить этому важное внимание, т.к. это ещё больше повысит качество научной статьи, представленной на рецензирование, в т.ч. сформирует предпосылки для расширения

читательской аудитории.

Выводы, интерес читательской аудитории. С учётом всего вышеизложенного, следует отметить, что статья выполнена на актуальную тему на достаточно высоком уровне, обладает высоким уровнем потенциального интереса читательской аудитории (как в научном сообществе, так и со стороны органов государственной власти Российской Федерации, субъектов Российской Федерации, субъектов предпринимательства). После проведения указанных в тексте рецензии корректировок (носящих преимущественно формальный характер, но соблюдение правил оформления научной статьи также важно, как и её содержание) может быть одобрена к опубликованию.