DOI 10.37882/2223-2966.2025.02.27

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ ТЕСТИРОВАНИЯ МАСШТАБИРУЕМОСТИ СИСТЕМЫ МОНИТОРИНГА И УПРАВЛЕНИЯ СЕТЕВЫМИ ЭЛЕМЕНТАМИ НА ОСНОВЕ ПОДХОДА «ИНФРАСТРУКТУРА КАК КОД»

PLATFORM FOR TESTING THE SCALABILITY OF MONITORING AND MANAGEMENT SYSTEM BASED ON «INFRASTRUCTURE AS CODE»

V. Rogov
A. Kharitonov

Summary. This article discusses the issues of testing the scalability of network element monitoring and management systems. A custom-developed tool based on the Infrastructure as Code (IaC) approach is proposed as a solution. This tool offers ease and flexibility of configuration, as well as significant reduction in required testing resources.

Keywords: EMS, scalability testing, virtual machines, Ansible.

Рогов Владислав Дмитриевич
Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
rogvlad91@gmail.com
Харитонов Антон Юрьевич
Кандидат технических наук, инженер, доцент практики, Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
akharitonov@itmo.ru

Аннотация. В данной статье рассматривается проблематика тестирования масштабируемости систем мониторинга и управления сетевыми элементами. В качестве решения предлагается разработанная авторами платформа для тестирования на основе подхода инфраструктура как код. Программное обеспечение, предлагаемое автором, обладает лёгкостью и гибкостью конфигурации, а также позволяет существенно снизить необходимые ресурсы для тестирования.

Ключевые слова: система мониторинга и управления сетевыми элементами, тестирование масштабируемости, виртуальные машины, Ansible.

Введение

Системы мониторинга и управления элементами сети [1, с. 129] играют ключевую роль в сетевой инфраструктуре, обеспечивая эффективное и надёжное управление разнообразными сетевыми устройствами. В современных телекоммуникационных и информационных сетях, где доступность, производительность и безопасность имеют первостепенное значение, системы управления сетевыми элементами становятся критической компонентой, обеспечивающей управление элементами сети и обеспечивающей их бесперебойную работу.

Тестирование масштабируемости — это вид нефункционального тестирования, направленного на проверку производительности системы, сети или компонента путём увеличения или уменьшения нагрузки на определенных уровнях. Масштабируемые тесты можно проводить на основе требований к аппаратным средствам, программному обеспечению или базам данных. Основная цель тестирования масштабируемости — убедиться, что система способна справиться с ожидаемым увеличением нагрузки и количества пользователей [2].

Сбои в работе системы управления сетевыми элементами могут привести к серьёзным последствиям, включая простои сети и потерю связи. Тестирование масштабируемости помогает выявить проблемы до их возникновения в реальной эксплуатации, что позволяет предотвратить сбои и обеспечить надёжную работу системы.

Для тестирования масштабируемости систем управления сетевыми элементами авторы проводят сравнительный анализ рынка программного обеспечения для тестирования масштабируемости.

Требования к разрабатываемому программному обеспечению

Для тестирования масштабируемости системы управления сетевыми элементами программное обеспечение должно удовлетворять следующим требованиям:

1. поддержка специфических сетевых протоколов для системы управления сетевыми элементами, такие как NETCONF [3] — Network Configuration Protocol — и SSH [4] — Secure Shell;
2. универсальность конфигурации для множества пользователей и сетевых элементов.

Было рассмотрено различное программное обеспечение для тестирования масштабируемости для определения возможности использования. Результаты можно увидеть в Таблице 1.

Таблица 1.

Результат сравнительного анализа программного обеспечения для тестирования масштабируемости

Требования к ПО	Программное обеспечение			
	JMeter[5]	LoadRunner[6]	Gatling[7]	Locust[8]
Поддержка SSH и NETCONF	–	+	–	–
Универсальность в конфигурация при множестве пользователей и сетевых элементов	–	+	+	+
Открытый код	+	–	+	+
Стоимость лицензии	Бесплатно	Платно	Бесплатно	Бесплатно

Как можно видеть из таблицы, по всем техническим пунктам нам подходит лишь LoadRunner. Однако данное программное обеспечение является проприетарным и легально использовать его на данный момент в Российской Федерации не представляется возможным, вследствии чего автором данной статьи было разработано собственное программное обеспечение для тестирования масштабируемости.

Исходя из вышеперечисленного, схема разрабатываемого программного обеспечения и его взаимодействия с тестируемой системой управления сетевыми элементами должна быть примерно следующей (см. рис. 1), где присутствуют тестируемая система управления элементами, контейнеры с сетевыми элементами, представляющими нагрузку и симулирующими действия реального пользователя сети, которых в одном условном контейнере может быть от 1 до N, в тоже время контейнеров может быть от 1 до M, компонент управления элементами нагрузки — некий элемент, который управляет запуском и управлением жизненным циклом контейнеров сетевых элементов.

Подход для разработки платформы для тестирования масштабируемости EMS на основе рекурсивных виртуальных машин

Изначально авторами рассматривался подход с использованием виртуальной машины, однако у него имеется существенный недостаток, а именно — неоптималь-

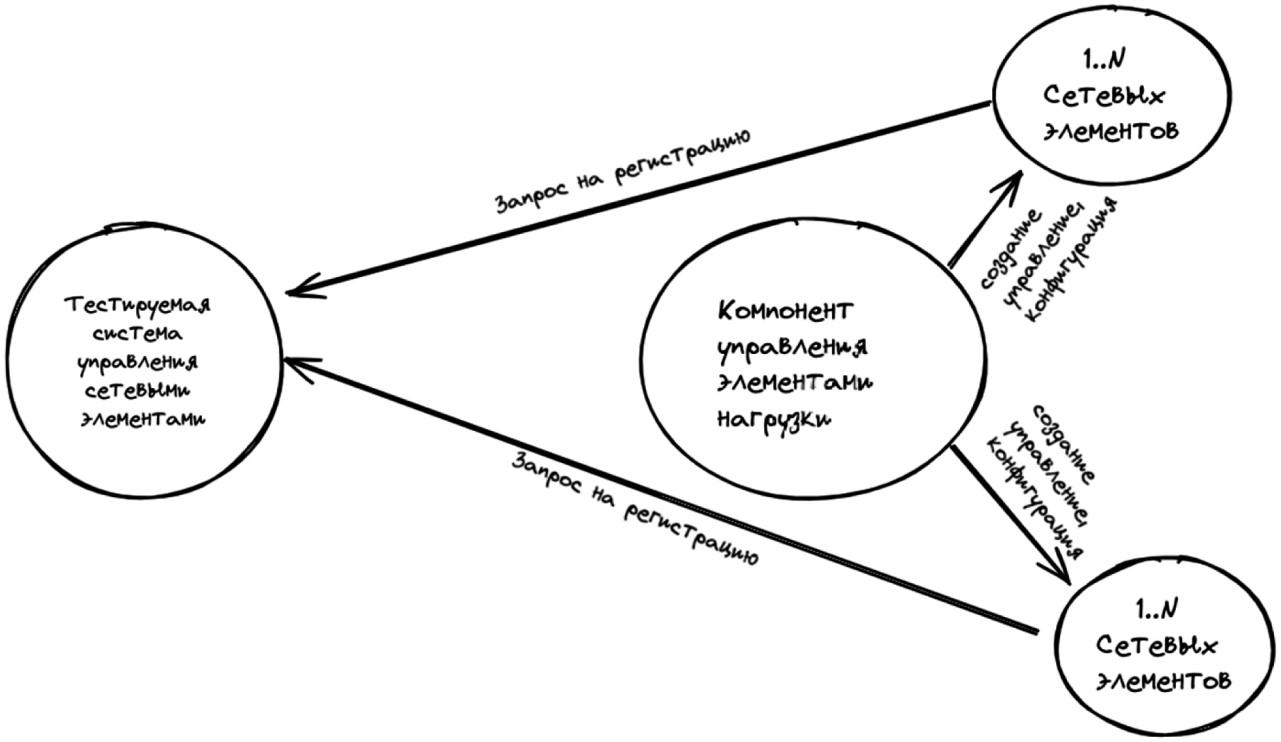

Рис. 1. Структура разрабатываемого автором статьи программного обеспечения с тестируемой системой управления сетевыми элементами

ное использования ресурсов — на одну виртуальную машину запускается один экземпляр симулятора сетевого элемента.

Данную проблему предлагается нивелировать путем применения рекурсивных виртуальных машин. Суть этого подхода, предлагаемого автором — внутри каждой виртуальной машины запускаются другие виртуальные машины, на которых и разворачиваются симуляторы сетевых элементов. Таких уровней рекурсии может быть несколько.

В таком случае схема программного обеспечения может выглядеть следующей (см. рис. 2).

В данном подходе управление и конфигурация виртуальных машин с элементами нагрузки осуществляется при помощи программного обеспечения по запуску и конфигурации виртуальных машин VirtualBox[9] или Vmware [10]. Число симуляторов в таком случае будет зависеть от числа виртуальных машин и слоёв рекурсии. Однако данный подход не был применен автором статьи для разработки программного обеспечения вследствии следующих недостатков:

1. Ресурсоемкость:

Запуск множества виртуальных машин внутри других виртуальных машин приведет к серьезному потреблению ресурсов, таких как мощность процессора, память и дисковое пространство, что снизит производительность и эффективность.

2. Отсутствие гибкости в конфигурации управляемыми ресурсами, такими как мощность процессора, память и дисковое пространство:

Вложенные виртуальные машины могут сталкиваться с низкой производительностью из-за эффекта «контейнеризации», когда ресурсы неэффективно распределяются между различными слоями виртуализации.

3. Отсутствие изоляции:

Использование рекурсивных виртуальных машин приводит к отсутствию изоляции между различными экземплярами, что недопустимо при тестировании сетевых элементов, особенно если требуется изолированное окружение.

Вследствие данных проблем использование рекурсивных виртуальных машин не является эффективной основой для программного обеспечения для тестирования масштабируемости.

Подход для разработки платформы для тестирования масштабируемости EMS на основе Kubernetes

Kubernetes [11] является программным обеспечением для создания и управления контейнерами. Схе-

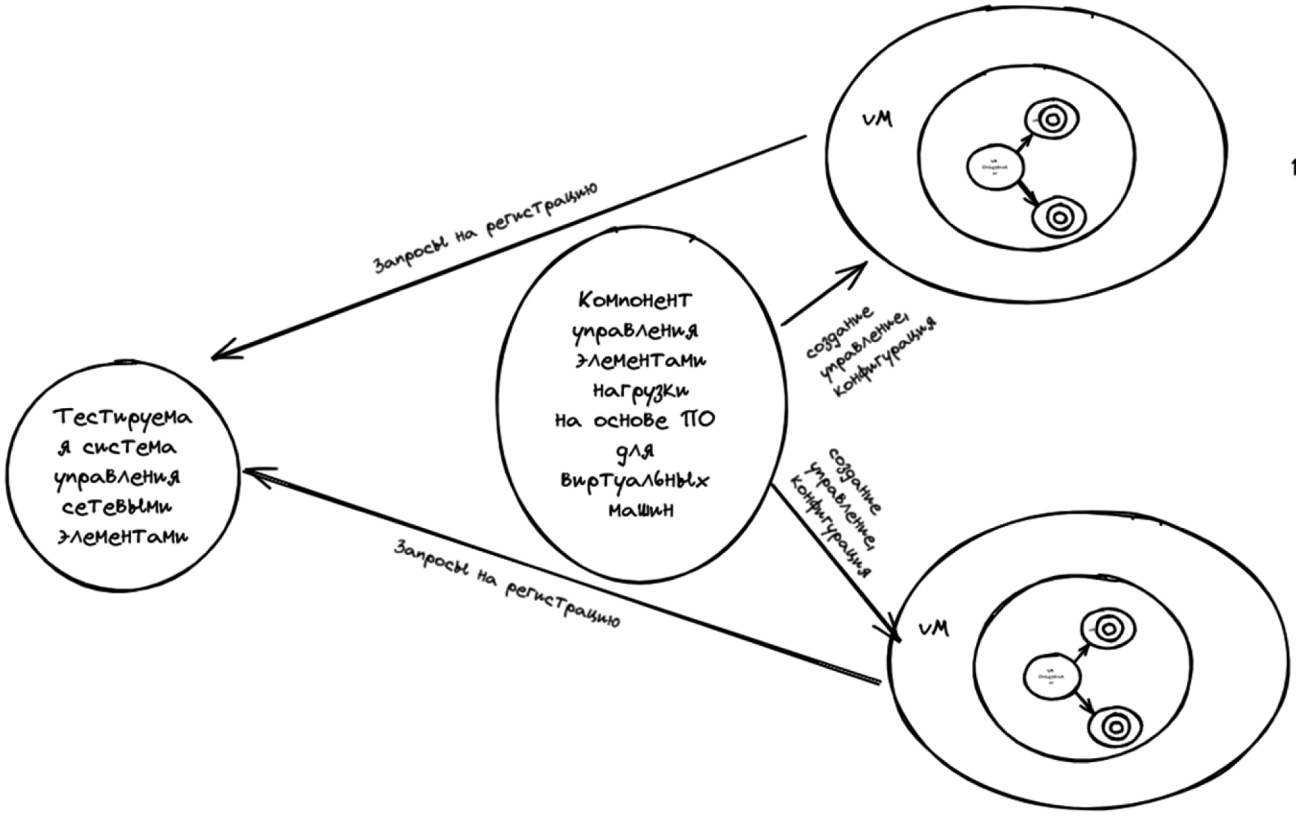

Рис. 2. Структура программного обеспечения с использованием рекурсивных виртуальных машин

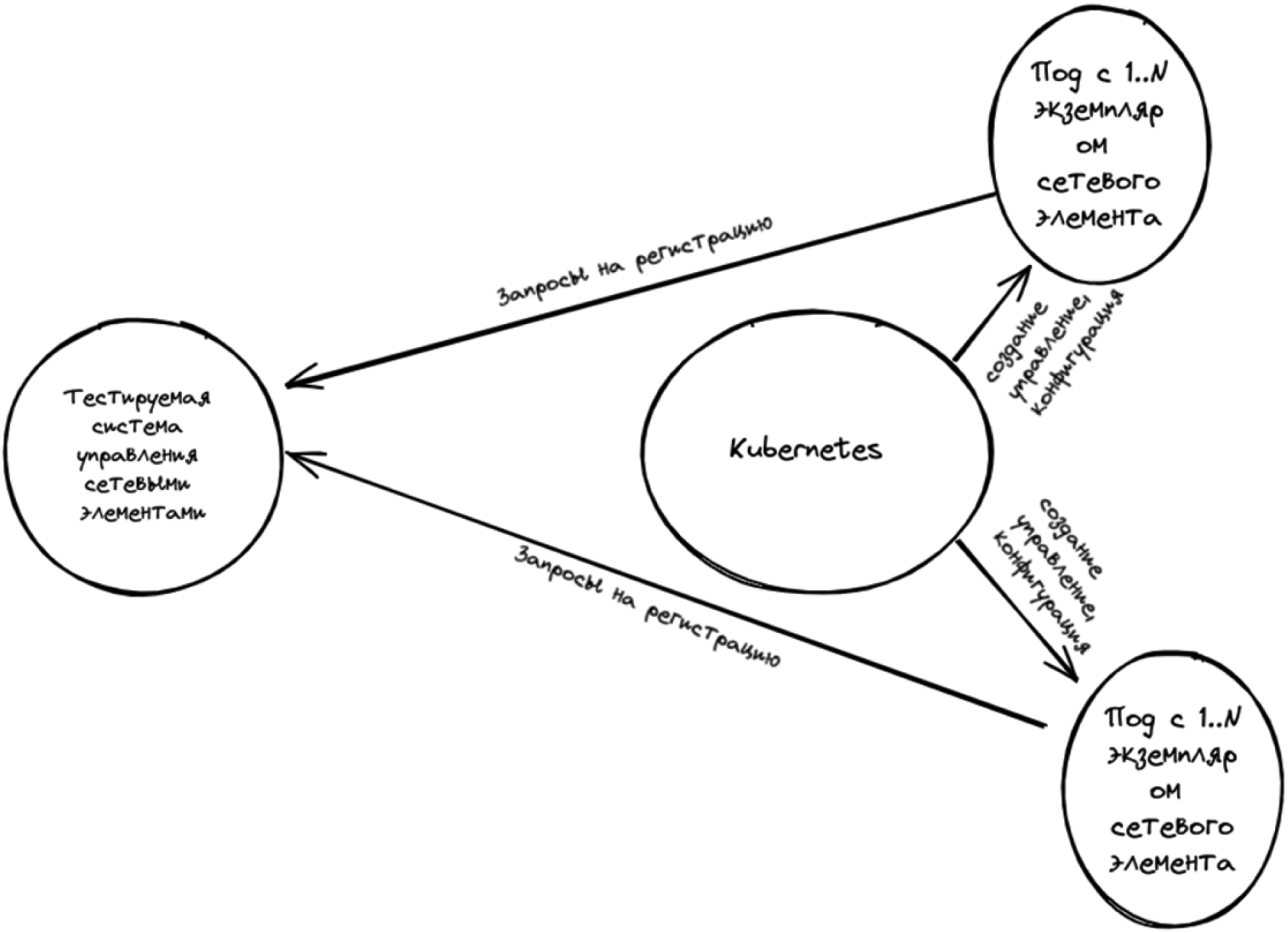

Рис. 3. Схема программного обеспечения с использованием Kubernetes

ма программного обеспечения на основе Kubernetes и docker-compose [12] представлена на рис. 3.

Автором предлагается следующий подход: на одном устройстве, разворачивается кластер Kubernetes. Внутри этого кластера будут разворачиваться поды (pods) — самая маленькая и базовая вычислительная единица, которую можно создать и управлять. Под представляет собой один экземпляр контейнера в кластере Kubernetes.

Программа на языке bash [13] будет создавать yaml deployment файлы с нужным количеством подов, задаваемым в конфигурационном файле, с помощью которых будут запускаться элементы нагрузки, разработанные автором данной статьи.

Элементы нагрузки будут иметь свой уникальный IP-адрес и уникальный идентификатор.

Следовательно, для запуска 100000 подобных элементов будет необходимо 100 подов, внутри которых будет развернуто 100 сервисов. Автором данной статьи было разработано собственное программное обеспечение для тестирования масштабируемости, поскольку решение на основе Kubernetes обладает рядом недостатков:

1. Отсутствие изоляции между контейнерами внутри пода:

Если контейнер внутри пода перейдет в нерабочее состояние, например, из-за сбоя приложения, автоматический перезапуск будет применён ко всем контейнерам внутри этого пода. Это может привести к потере доступности не только одного, но и всех сервисов, работающих в этом поде.

2. Сложность изучения и лишние функции:

Для решения конкретной задачи по запуску множества инстансов сетевых элементов придётся столкнуться с лишней сложностью, представляемой Kubernetes для широкого спектра использования.

В связи с приведенными выше недостатками автором было принято решение не использовать данное программное обеспечение в качестве основы для разрабатываемой платформы по тестированию масштабируемости.

Подход для разработки платформы для тестирования масштабируемости системы управления сетевыми элементами на основе подхода «Инфраструктура как Код»

«Инфраструктуру как код» (Infrastructure as Code, IaC) [14, с. 159] можно охарактеризовать как автоматизацию создания инфраструктуры с помощью программ или же конфигурационных файлов.

Для решения на основе подхода IaC в данной работе использовался инструмент Ansible [15, с. 127].

С помощью Ansible сценария — описания состояния ресурсов системы, в котором она должна находиться в конкретный момент времени, включая установленные пакеты, запущенные службы, созданные файлы и многое другое — будет производиться запуск симуляций сетевых элементов на виртуальных машинах.

Пример программы для тестирования масштабируемости

Основой данного программного обеспечения будет являться Ansible сценарий, написанный авторами и частично изображённый на рисунке 4.

Данный текст сценария описывает следующее:

1. подготовку таргетной виртуальной машины — установка необходимых зависимостей, добавление текущего пользователя в docker группу прав, установка docker-compose;
2. копирование приложение симулятора сетевого элемента;
3. исполнение Python программы по созданию docker-compose.yaml файла с необходимым количеством сервисов, задаваемым в .env файле. На распечатке 1 изображена фрагмент текста файла данного файла:

Распечатка 1

«version»: «3»
«services»:
«image»: «golang:alpine»,
«build»:
«context»: «.»,
«dockerfile»: «Dockerfile»,
«container_name»: service_name,
«ports»: [f»{service_port}: {service_port}»],
«volumes»: [«. /:/app»]

4. запуск команды docker-compose up.

Python программа представляет собой создание docker-compose файла с шаблонным описанием необходимых сервисов — симуляций сетевых элементов. Данные сетевые элементы будут с помощью HTTP посылать запросы на хост-машину, где их будет принимать, считывать и записывать в текстовый файл симуляция EMS.

```
---
- hosts: target_vm
  become: true
  environment:
    NUM_SIMS: 5
  tasks:
    - name: Remove outdated Docker GPG key
      apt_key:
        id: 0EBFCD88
        state: absent
    - name: Add GPG key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Update apt cache
      ansible.builtin.apt:
        update_cache: yes
        allow_unauthenticated: yes

    - name: Install aptitude
      apt:
        name: aptitude
        state: latest
        update_cache: true

    - name: Install required system packages
      apt:
        pkg:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
          - python3-pip
          - virtualenv
          - python3-setuptools
        state: latest
        update_cache: true

    - name: Add docker repository to apt
      apt_repository:
        repo: deb https://download.docker.com/linux/ubuntu focal stable
        state: present
    - name: Install docker Python library
      pip:
        name: docker
        version: 6.1.3
```

Рис. 4. Фрагмент текста программы, представляющей Ansible сценарий

Результаты

В данной работе был разработано программное обеспечение для тестирования масштабируемости на основе подхода «Инфраструктура как код», а также собран демонстрационный стенд. Демонстрационный стенд включает в себя приложения-симуляторы сетевого элемента и EMS, написанные на языке Golang [16].

Также для развертывания необходимого числа сетевых элементов были сконфигурированы две виртуальные машины. После чего на каждой из виртуальных машин с помощью Ansible Playbook было запущено необходимое количество симуляторов сетевых элементов и продемонстрировано их взаимодействие с EMS, развернутой на хост-машине.

Для тестирования инструмента было развернуто две виртуальные машины с операционной системой Ubuntu

```
{"time":"2024-01-15T10:28:34.000227004+03:00","level":"INFO","msg":"Listening on","port":8080}
{"time":"2024-01-15T10:33:02.254042928+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:42846","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:03.50844208+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:50156","data":"Unique ID: 1705303968
{"time":"2024-01-15T10:33:04.249422024+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:38832","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:04.380735949+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:55026","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:04.666395549+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:32846","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:04.811040993+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:59278","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:05.449616503+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:50302","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:17.251527084+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:42846","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:18.508423632+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:50156","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:19.247832106+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:38832","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:19.377385582+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:55026","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:19.666256954+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:32846","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:19.982289245+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:56062","data":"Unique ID: 170530396
{"time":"2024-01-15T10:33:20.114487672+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:59800","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:20.198440954+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:33830","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:20.404469499+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.131:54494","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:20.461006402+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:50302","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:20.739455147+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:38706","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:20.823025839+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:49906","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:20.956866609+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:33156","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:21.046979935+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:35284","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:21.047074107+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:50038","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:21.208987678+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:47704","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:21.389283191+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:44818","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:21.416246427+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:57890","data":"Unique ID: 170530397
{"time":"2024-01-15T10:33:21.438593961+03:00","level":"INFO","msg":"Received connection from ","addr":"192.168.241.132:37326","data":"Unique ID: 170530397
```


Рис. 5. Логи о полученных запросах от сетевых элементов

22.04 [17]. На каждой из них было запущено по 10 симуляций сетевого элемента, соответственно количество уникальных запросов, зарегистрированных на хост-машине EMS должно быть 20. После запуска playbook на каждой из машин развернулось необходимое количество контейнеров с одним сетевым элементом в каждом.

На компьютере, с которого был запущен playbook, также появились логи от EMS, которые продемонстрированы на рис. 5.

Как можно видеть из логов, EMS регистрировала запросы с разных IP-адресов. Для того чтобы подсчитать количество уникальных адресов, напишем простую Python-программу, код которой продемонстрирован на рис. 6.

```python
import json


def count_unique_ips(log_file_path):
    unique_ips = set()

    with open(log_file_path, 'r') as file:
        for line in file:
            log_entry = json.loads(line)
            if 'addr' in log_entry:
                unique_ips.add(log_entry['addr'])

    return len(unique_ips) - 1


log_file_path = 'logfile.txt'
unique_ips_count = count_unique_ips(log_file_path)
print(f"Количество уникальных IP-адресов: {unique_ips_count}")
```

Рис. 6. Текст программы по подсчёту уникальных IP-адресов

Эта программа открывает лог-файл, читает каждую строку как JSON, извлекает IP-адрес из поля ‘addr’ и подсчитывает уникальные IP-адреса с использованием множества. В конце выводится количество уникальных IP-адресов. На рис. 7 представлен результат выполнения программы.

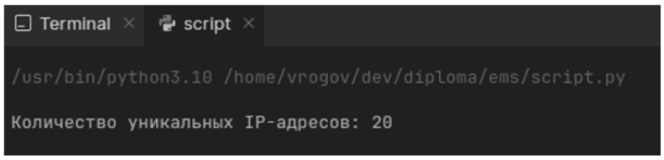

Рис. 7. Результат выполнения кода программы по подсчету уникальных IP-адресов

Как можно видеть, количество уникальных IP-адресов равно 20, что соответствует количеству запущенных симуляторов и виртуальных машин.

Обсуждение

Предложенный подход для тестирования масштабируемости EMS является более предпочтительным и удобным, нежели рассмотренные существующие инструменты, однако также обладает рядом недостатков, среди которых можно выделить необходимость создания и конфигурации виртуальных машин вручную. Также отсутствие механизма управления и мониторинга контейнеров может сделать систему менее отказоустойчивой, так как не будет легко обнаруживать и восстанавливать сервисы в случае их сбоев.

Заключение

Системы управления сетевыми элементами играют очень большую роль в построении инфраструктуры,

обеспечивая управление элементами, мониторинг, настройку и диагностику. Именно поэтому важно осуществлять тестирование масштабируемости для систем управления сетевыми элементами. Это поможет понять пределы возможностей системы, её точки отказа и слабые места.

В данной работе был проведен краткий сравнительный анализ существующего программного обеспечения для тестирования масштабируемости. В результате выяснилось, что для поставленной задачи можно применить LoadRunner, но его проприетарная природа и платное распространение являются преградой для этого.

В процессе поиска основы для инструмента для тестирования были рассмотрены рекурсивные виртуальные машины и программного обеспечения Kubernetes. Однако рекурсивные виртуальные машины являются слишком сложными в конфигурации и не оптимизированными в плане ресурсоемкости, а Kubernetes сложен в изучении и погружении, а также требует платных кластеров.

В результате было выбрано решение на основе подхода IaC и Ansible. Был собран демонстрационный стенд из хост-машины и двух виртуальных машин, на которых с помощью Ansible Playbook поставлялось необходимое ПО и запускались симуляции сетевого элемента.

Данные симуляции совершали запросы к запущенной на хост-машине EMS.

Разработанное решение легко масштабируется путем добавления новых виртуальных машин, запуска большего числа сетевых элементов. Благодаря возможностям Ansible, разработанное программное обеспечение можно будет адаптировать под специфику EMS, что будет продемонстрировано в дальнейшей работе.

ЛИТЕРАТУРА

1. Kim D.W., & Kim J.S. (2003). Element management systems for telecom networks. IEEE Communications Magazine, 41(12), 122–130.
2. Myers G.J., Sandler C., Badgett T. The art of software testing. — John Wiley & Sons, 2011.
3. Network Configuration Protocol (NETCONF). [Официальный сайт]. URL: https://www.rfc-editor.org/rfc/rfc6241 (дата обращения: 10.03.2024).
4. The Secure Shell (SSH) Transport Layer Protocol. [Официальный сайт]. URL: https://www.rfc-editor.org/rfc/rfc4253 (дата обращения: 10.03.2024).
5. V. Tiwari, S. Upadhyay, J.K. Goswami and S. Agrawal, «Analytical Evaluation of Web Performance Testing Tools: Apache JMeter and SoapUI,» 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 2023, pp. 519–523, doi: 10.1109/CSNT57126.2023.10134699.
6. R. Abbas, Z. Sultan, and S.N. Bhatti, «Comparative analysis of automated load testing tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege,» 2017 International Conference on Communication Technologies (ComTech), Rawalpindi, Pakistan, 2017, pp. 39–44, doi: 10.1109/COMTECH.2017.8065747.
7. Gatling. [Официальный сайт]. URL: https://gatling.io (дата обращения: 10.03.2024).
8. Locust. [Официальный сайт]. URL: https://locust.io/ (дата обращения: 10.03.2024).
9. VirtualBox. [Официальный сайт]. URL: https://www.virtualbox.org/ (дата обращения: 10.03.2024).
10. VMware. [Официальный сайт]. URL: https://www.vmware.com/ (дата обращения: 10.03.2024).
11. Kubernetes. [Официальный сайт]. URL: https://kubernetes.io/ (дата обращения: 10.03.2024).
12. Docker Compose overview. [Официальный сайт]. URL: https://docs.docker.com/compose/ (дата обращения: 10.03.2024).
13. Bash Reference Manual. [Официальный сайт]. URL: https://www.gnu.org/software/bash/manual/bash.html (дата обращения: 10.03.2024).
14. Artac M. et al. Infrastructure-as-code for data-intensive architectures: a model-driven development approach //2018 IEEE international conference on software architecture (ICSA). — IEEE, 2018. — С. 156–15609.
15. Mallett A., Mallett A. Managing services using ansible //Red Hat Certified Engineer (RHCE) Study Guide: Ansible Automation for the Red Hat Enterprise Linux 8 Exam (EX294). — 2021. — С. 123–133.
16. The Go Programming Language [Официальный сайт]. URL: https://go.dev/ (дата обращения: 10.03.2024).
17. Canonical Ubuntu. [Официальный сайт]. URL: https://ubuntu.com/ (дата обращения: 10.03.2024).

Журнал «Современная наука: актуальные проблемы теории и практики»