Программные системы и вычислительные методы
Правильная ссылка на статью:
Хлесткин А.Ю., Райков А.В., Казанцев А.А., Емелин Д.П., Ларин Д.В. Роль операционных систем и оболочек в облачных вычислениях: анализ ОС и оболочек, используемых в облачных платформах и их влияние на облачную инфраструктуру // Программные системы и вычислительные методы. 2024. № 4. DOI: 10.7256/2454-0714.2024.4.70626 EDN: KYNBQH URL: https://nbpublish.com/library_read_article.php?id=70626

Роль операционных систем и оболочек в облачных вычислениях: анализ ОС и оболочек, используемых в облачных платформах и их влияние на облачную инфраструктуру

Хлесткин Андрей Юрьевич

кандидат технических наук

доцент; кафедра информатика и вычислительная техника; Поволжский государственный университет телекоммуникаций и информатики

443010, Россия, Самарская область, г. Самара, ул. Льва Толстого, 23

a.hlestkin@psuti.ru

Райков Александр Вячеславович

ORCID: 0009-0005-0033-8524

студент, кафедра информатики и вычислительной техники, Поволжский государственный университет телекоммуникаций и информатики

443010, Россия, Самарская область, г. Самара, ул. Льва Толстого, 23

sraikov7@mail.ru

Казанцев Андрей Алексеевич

студент, кафедра информатики и вычислительной техники, Поволжский государственный университет телекоммуникаций и информатики

443010, Россия, Самарская область, г. Самара, ул. Льва Толстого, 23

NuclearAndGoner@gmail.com

Емелин Даниил Павлович

студент, кафедра информатики и вычислительной техники, Поволжский государственный университет телекоммуникаций и информатики

443010, Россия, Самарская область, г. Самара, ул. Льва Толстого, 23

demelin163@gmail.com

Ларин Денис Вячеславович

студент, кафедра информатики и вычислительной техники, Поволжский государственный университет телекоммуникаций и информатики

443010, Россия, Самарская область, г. Самара, ул. Льва Толстого, 23

denlar1989@gmail.com

Статья из рубрики "Models and methods of information security management"

DOI:

10.7256/2454-0714.2024.4.70626

EDN:

KYNBQH

Дата направления статьи в редакцию:

01-05-2024

Аннотация: Основное внимание в статье уделяется операционным системам, облачным вычисления и командным оболочкам, которые активно развиваются несколько десятков лет и уже являются частью жизни, как обычного пользователя, так и профессионала компьютерных технологий. Эти объекты рассматриваются как отдельные составляющие информационных технологий, так и их взаимосвязь и результаты этой взаимосвязи. Операционные системы в облачных серверах выполняют управленческую роль. Если быть точнее, то они управляют ресурсами физических серверов. Операционные системы или же ОС в данном случает определяют несколько параметров. К таким параметрам мы может отнести то, как операционные системы могут использовать память, хранилище для различных виртуальных машин и управлять ими. Командные оболочки в свою очередь представляются приложением, которое предоставляет пользователю некий интерфейс командной строки, в которой тот вводит команды как по отдельности, так и запускает скрипты, состоящие из списка команд. Методы исследования включают теоретические (классификация, сравнительный анализ, анализ литературы) и практические (эксперимент, моделирование) подходы. Это позволяет провести комплексный анализ функционирования операционных систем и командных оболочек в облачных вычислениях. Научной новизной нашего исследования служит приведение скриптов для выполнения той или иной задачи в области облачных вычислений на определённой операционной системе с использованием вышеописанных командных оболочек. Таким образом, авторами были приведены теоретические данные об операционные системах и командных оболочках. Авторы привели примеры скриптов и анализ безопасности для командных оболочек Bash и Bourne Shell (sh) для операционной системы Linux и скрипты для командных оболочек Command Prompt (cmd.exe) и Windows PowerShell для операционной системы Microsoft Windows. В результате проведенной работы были составлены таблицы с влиянием операционной системы и командной оболочки на выполнение облачных вычислений. Анализ таблиц позволил охарактеризовать авторам объекты исследования данной научной работы и сделать соответствующие выводы.

Ключевые слова:

операционные системы, командные оболочки, облачные вычисления, облачные платформы, облачная инфраструктура, Linux, Microsoft Windows, Bash, Bourne Shell, Hyperfine

Введение

Операционные системы и командные оболочки по своей природе являются основополагающими элементами облачной инфраструктуры. Эти элементы обеспечивают стабильность, безопасность, эффективное использование ресурсов и удобное управление облачной инфраструктурой. Немаловажным является и правильный выбор, и настройка операционной системы и оболочки в успешности вычислений.

Облачными вычислениями можно назвать процесс предоставления удаленных вычислительных мощностей конечным потребителем. Данные мощности могут использоваться различным образом – начиная от использования рабочих мест с помощью виртуальных машин и других технологий виртуализации, и заканчивая майнингом криптовалют [1].

«Cloud computing» или же «Облачные вычисления» являются реализацией определённого метафорического образа Internet в виде облака, благодаря которому пользователи получают доступ ко всем возможным сервисам и ресурсам. Основы вычислений заключается в использовании виртуализированных или масштабируемых ресурсах.

В данной статье осуществлена попытка проведения структурного анализа операционных систем и оболочек и их влияние на облачную инфраструктуру.

Практическая часть

В современном мире существует множество специализированных операционных систем для запуска контейнеров. Авторами статьи в качестве примера такой операционной системы хотелось бы отметить CoreOS Container Linux. Данная операционная система автоматически обновляется, а также обеспечивает высокую доступность в использовании контейнеризованных приложений.

Выбор данной операционной системы был сделан из-за её особенностей. Так, главным преимуществом CoreOS Container Linux является концепт immutable infrastructure (неизменяемая инфраструктура). То есть операционная система и приложения запускаются в контейнерах и не подвергаются изменениям во время выполнения. Любые изменения в системе осуществляются путём замены всего образа операционной системы на новую версию.

CoreOS Container Linux предоставляет некую систему управления обновлениями, которая в свою очередь обновляет операционную систему без перезагрузки. Если сравнивать CoreOS Container Linux с операционной системой Microsoft Windows, то у второй в большинстве случаев после установки обновления требуется дополнительное действие - перезагрузка. Отсутствие перезагрузки после обновления у CoreOS Container Linux обеспечивает её безопасностью и надёжностью обновлений в кластерной среде без простоев.

В CoreOS Container Linux отсутствует избыточность, как у многих дистрибутивов Linux. CoreOS Container Linux предоставляет минимальный набор компонентов, в следствии чего идёт упрощение конфигурации, а так же обеспечение надёжной работой в распределённых средах.

Container-Optimized OS — это образ операционной системы для виртуальной машины Google Compute Engine, оптимизированный для запуска контейнеров Docker, предоставляемые Google Cloud Platform (GCP) (набор облачных служб, которая

представляет компания Google).

Container-Optimized OS обладает следующими функциями:

· Запуск контейнеров «из коробки»: экземпляры операционной системы, оптимизированные для контейнеров, поставляются с предустановленными средами выполнения Docker и containerd и cloud-init. С помощью экземпляра операционной системы, оптимизированного для контейнеров, существует возможность запускать контейнер одновременно с созданием виртуальной машины, не требуя настройки со стороны хоста.

· Меньшая поверхность атаки: ОС, оптимизированная для использования в контейнерах, занимает меньше места, что снижает потенциальную поверхность атаки для созданного экземпляра.

· Заблокировано по умолчанию: экземпляры операционной системы, оптимизированные для контейнеров, по умолчанию включают заблокированный брандмауэр и другие параметры безопасности.

· Автоматические обновления: экземпляры операционной системы, оптимизированные для контейнеров, настроены на автоматическую загрузку еженедельных обновлений в фоновом режиме; для использования последних обновлений требуется только перезагрузка.

Пользователь может выбрать данную операционную систему, если ему необходимо решать следующие задачи:

· Требуется поддержка контейнеров или Kubernetes с минимальной настройкой.

· Нужна операционная система, занимающая мало места и защищенная для контейнеров.

· Нужна операционная система, которая протестирована для запуска Kubernetes на экземплярах Google Compute Engine.

Авторами было решено сделать акцент на операционных системах Linux, Microsoft Windows и на оболочках Bourne Again Shell, Bourne Shell (sh), Command Prompt (cmd.exe), Windows PowerShell.

Финский студент (заметьте, студент) Линус Торвальд в 1991 году создал клон Unix и назвал его **Linux** (версия 0.0.1). Вся ОС размещалась в ядре, была монолитной (9300 строк кода на C, 950 - на Assembler). Работала на процессоре Intel (386). Окончательно датой выпуска считается 1994 год, версия 1.0 (165 тысяч строк кода, новая файловая система, сетевое программное обеспечение). 2-я версия - 1996 год (470 тысяч строк кода, добавлены новые драйверы устройств) [2].

Linux входит в число крупнейших хорошо документированных линеек программных продуктов, изучаемых до сих пор. Сообщество разработчиков Linux включает как добровольцев, так и платных разработчиков из более чем 200 компаний, включая Red Hat, IBM, Intel, Oracle, Google и Microsoft среди прочих[4]. Зрелость проекта проявляется по нескольким показателям, таким как размер кодовой базы (более 8 миллионов строк), количество активных разработчиков (600–1200 на каждый релиз и далее) и уровень активности (до 10000 исправлений на выпуск) [5].

Если говорить про достоинства данной операционной системы, то стоит отметить open

source. Отвечая на вопрос, что это может значить, можно сказать, что ядро любого дистрибутива Linux имеет открытый код, что позволяет улучшить любую программу или само ядро этой операционной системы.

Linux чаще всего взаимодействует с оболочкой Bash или как её ещё называют Bourne Again Shell. На данный момент данная оболочка всё чаще установлена по умолчанию во многих дистрибутивах Linux.

Оболочка Bash включает такие инструменты, как:

· Редактирование командной строки;

· Управление заданиями;

· Неограниченный размер истории команд;

· Функции и псевдонимы оболочек;

· Индексированные массивы неограниченного размера;

· Целочисленная арифметика в любой базе от двух до шестидесяти четырех [5].

Авторами статьи, были написаны следующие коды на операционной системе Linux с использованием Bash для визуализации их влияния на облачную инфраструктуру Mail.

· *Скрипт для автоматического создания нового виртуального сервера:*

```
#!/bin/bash

# Аутентификация в Облаке Mail.ru

mcloud login

# Создание нового виртуального сервера

mcloud vm create --name my-server --flavor b2.micro --image centos_7
```

· *Скрипт для запуска виртуального сервера по его ID:*

```
#!/bin/bash

# Аутентификация в Облаке Mail.ru

mcloud login

# Запуск виртуального сервера по его ID

mcloud vm start --id <server_id>
```

· *Скрипт для остановки виртуального сервера по его ID:*

```
#!/bin/bash

# Аутентификация в Облаке Mail.ru

mcloud login

# Остановка виртуального сервера по его ID

mcloud vm stop --id <server_id>
```

Проанализировав данные коды, можно сделать вывод, что Bash позволяет пользователю создать скрипт, который упростит, а главное, автоматизирует операции с какими-либо облачными ресурсами, так же скрип позволяет быстро создать виртуальный сервер и настроить его параметры.

Кроме того, основываясь на второй скрипт, можно сказать, что Bash-скрипт позволяет без привлечения особо трудоёмких и затратных ресурсов управлять виртуальным сервером, выполняя различные операции. Например, операцию, приведённую в скрипте, запуск и остановка виртуального сервера по его ID.

Не стоит оставлять без внимания оболочку **Bourne Shell**(sh). История данной оболочки начинается с Shell, вошедшего в поставку первой редакции UNIX, разработанную Кеном Томпсоном. Но данная оболочка была не совсем удобна для работы, так как имела многочисленные ограничения. В результате Стивен Борн переписал стандартный Shell UNIX.

Популярность оболочка завоевала благодаря компактности и преимущественно высокой скоростью работы. Благодаря этим преимуществам она стала оболочкой по умолчанию для ОС Solaris.

Но у оболочки Bourne есть ряд существенных недостатков:

· Нет встроенных функций для обработки логических и арифметических операций.

· Не может запоминать ранее выполненные команды, в отличие от большинства других командных оболочек.

· Не хватает комплексных функций для удобного интерактивного использования [6].

Чтобы просмотреть как происходит взаимодействие Bourne Shell (sh) с облачной платформой Mail на операционной системе Linux можно просмотреть следующие примитивные скрипты кодов:

· *Скрипт для автоматического создания нового виртуального сервера:*

```
#!/bin/sh

# Аутентификация в Облаке Mail.ru

mcloud login

# Создание нового виртуального сервера

mcloud vm create --name my-server --flavor b2.micro --image centos_7
```

· *Скрипт для запуска виртуального сервера по его ID:*

```
#!/bin/sh

# Аутентификация в Облаке Mail.ru

mcloud login

# Запуск виртуального сервера по его ID

mcloud vm start --id <server_id>
```

· *Скрипт для остановки виртуального сервера по его ID:*

```
#!/bin/sh

# Аутентификация в Облаке Mail.ru

mcloud login

# Остановка виртуального сервера по его ID

mcloud vm stop --id <server_id>
```

Авторами отмечено, что скрипт позволяет автоматизировать процесс создания и управления облачными ресурсами. Написание скриптов на Bourne Shell (sh) даёт возможность для создание виртуального сервера, а главное для его настройки. Такие опции позволяют администратору ускорить и оптимизировать рабочий процесс.

Гибкость и управляемость облачными ресурсами через запуск скриптов на Bourne Shell (sh) становится простым и доступным действием, как создание и управление объектными хранилищами.

Microsoft Windows — семейство проприетарных операционных систем корпорации Microsoft, направленных на использование графического интерфейса при управлении. Первоначально Windows представляла графическую надстройку для MS-DOS. Сейчас Windows занимает подавляющую долю на мировом рынке операционных систем [7].

История операционной системы Microsoft Windows начинается с ноября 1985 года. По настоящее время Microsoft Windows выпускает обновления. На декабрь 2024 года самой свежей операционной системой является Windows 11 версии 24H2.

Помимо этого, Microsoft Windows распространяет свои операционные системы на планшеты и смартфоны. Результатом такого решения стала операционная система Microsoft Windows 8, которая была выпущена в октябре 2012 года. Новая система стала значительно быстрее загружаться, но были некоторые проблемы с драйверами и запуском игр.

В Windows 10 Microsoft собирает множество данных об использовании компьютера. Примерами таких данных являются имя, адрес электронной почты и другие. Из-за этого на Microsoft обрушился шквал критики [8].

Одной из старейших приложений и инструментов командной строки, поставляемых с операционной системой Windows, — это командная строка, обычно называемая CMD [9].

Ниже представлен фрагмент стартового окна Command Prompt (cmd.exe) для удобства понимания читателями.

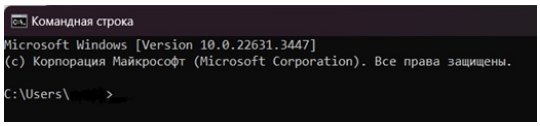

Рисунок 1. Пример Command Prompt (cmd.exe)

Для того чтобы открыть Command Prompt (cmd.exe) нужно выполнить следующие шаги:

1. Нажмите на меню «Пуск» на панели инструментов рабочего стола.

2. Введите команду «cmd» и нажмите Enter.

Наиболее распространённое применение использования Command Prompt (cmd.exe) администраторов заключается в администрировании задач, например, форматирование или управление разделами диска. Помимо этого, Command Prompt (cmd.exe) удобна для работы с просмотром файлов каталогов и управления ими.

Ниже приведены примеры скриптов для работы с облаком Mail.ru, которые можно выполнить в командной строке Command Prompt (cmd.exe) в операционной системе Microsoft Windows:

· *Скрипт для автоматического создания нового виртуального сервера:*

```
mcloud vm create --name my-server --flavor b2.micro --image centos_7
```

· *Скрипт для запуска виртуального сервера по его ID:*

```
mcloud vm start --id <server_id>
```

· *Скрипт для остановки виртуального сервера по его ID:*

```
mcloud vm stop --id <server_id>
```

Использование администраторами командной строки Command Prompt (cmd.exe) обусловлено удобством использования, так как Command Prompt (cmd.exe) позволяет выполнять операции с облачными вычислениями напрямую из стандартной командной оболочки Microsoft Windows, а значит не нужно устанавливать дополнительные инструменты.

Последней рассмотренной в данной статье командной оболочкой будет **Windows PowerShell**.

Windows PowerShell используется администраторами для автоматизации рутинных задач. Так же Windows PowerShell как и предыдущие командные оболочки позволяет менять настройки, останавливать и запускать сервера, обслуживать установленные приложения.

Windows PowerShell позволяет:

· Менять настройки операционной системы;

· Управлять службами и процессами;

· Настраивать роли и компоненты сервера;

· Устанавливать программное обеспечение;

· Управлять установленным ПО через специальные интерфейсы;

· Встраивать исполняемые компоненты в сторонние программы;

· Создавать сценарии для автоматизации задач администрирования;

· Работать с файловой системой, реестром Windows, хранилищем сертификатов и т.д. [10].

Windows PowerShell – командная оболочка, первоначально построенная на базе Microsoft .NET Framework, а позднее на .NET, и интегрирована с ними. В данной

командой оболочке используются разнообразные хранилища, которые напоминают файловую систему, для доступа к которым созданы поставщики.

Стоит рассмотреть и скрипты для выполнения облачный вычислений:

· *Скрипт для автоматического создания нового виртуального сервера:*

```
mcloud vm create --name my-server --flavor b2.micro --image centos_7
```

· *Скрипт для запуска виртуального сервера по его ID:*

```
mcloud vm start --id <server_id>
```

· *Скрипт для остановки виртуального сервера по его ID:*

```
mcloud vm stop --id <server_id>
```

PowerShell предоставляет для использования наиболее широкий набор функций и возможностей, если сравнивать его с командной строкой Command Prompt (cmd.exe).

Синтаксис у данной командной оболочки будет удобнее, что делает его оптимальным при выборе инструмента для автоматизации и написания сложных скриптов. Поддержка автодополнения и подсказки позволяют начинающему пользователю с лёгкостью познать управление данной командной оболочкой.

Анализ производительности с использованием утилиты Hyperfine

Для проведения анализа производительности различных оболочек в облачных вычислениях была использована утилита **Hyperfine** [11] — инструмент командной строки, предназначенный для тестирования скорости выполнения команд. Эта утилита предоставляет простой способ оценки времени работы различных скриптов и команд, обеспечивая точность измерений. В рамках исследования были проведены тесты, в которых скрипты выполняли аналогичные задачи, такие как создание, запуск и остановка виртуальных машин в облачной среде Mail.ru. Сравнение времени выполнения скриптов в различных оболочках позволило выявить ключевые различия в их производительности.

Для наглядности авторами разработана таблица (Рис.2) с необходимыми в результате тестирования критериями при использовании командных оболочек:

· Windows PowerShell под управлением ОС Microsoft Windows;

· Bash, под управлением ОС Linux;

· Command Prompt под управлением ОС Microsoft Windows;

· Bourne Shell под управлением ОС Linux.

Представлены результаты выполнения критериев:

· Время запуска (в секундах);

· Время отключения (в секундах).

Оболочка	Операционная система	Время запуска, с	Время отключения, с
Windows PowerShell	Windows	13	14
Bash	Linux	15	17
Command Prompt (cmd.exe)	Windows	24	29
Bourne Shell (sh)	Linux	19	20

Рисунок 2. Сравнительная таблица

***Безопасность** при **использовании** различных операционных систем и оболочек*

PowerShell:

Управление доступом**:** PowerShell поддерживает детализированное управление доступом, что позволяет ограничивать права пользователей и групп, а также настраивать роли для обеспечения безопасности.

Execution Policy**:** Эта настройка регулирует выполнение скриптов в PowerShell. Включение режимов "Restricted" или "AllSigned" помогает предотвратить запуск неподписанных или вредоносных скриптов.

Шифрование и криптография**:** PowerShell использует различные механизмы для защиты данных и аутентификации, включая поддержку SSL/TLS.

Уязвимости и атаки**:** PowerShell является популярной целью для атак, таких как использование вредоносных скриптов, что требует правильной настройки безопасности и применения многофакторной аутентификации (MFA).

Удаленное управление**:** PowerShell Remoting предоставляет возможность удаленного выполнения команд, что может быть использовано злоумышленниками. Поэтому важно защищать этот канал с помощью сильной аутентификации и настройки ограничений доступа.

Bash:

Переменные среды**:** Конфиденциальные данные, такие как пароли, могут храниться в переменных среды, что повышает риск утечек, если они не защищены.

Командные инъекции**:** Bash уязвим к атакам типа Command Injection, если ввод не проверяется должным образом. Поэтому важно тщательно фильтровать все данные, поступающие от пользователя.

Привилегии sudo и root**:** Использование прав суперпользователя может привести к серьезным уязвимостям, если эти права не ограничены или не контролируются должным образом.

Обновления безопасности**:** Так как Bash имеет долгую историю, важно регулярно обновлять систему для защиты от известных уязвимостей, таких как Shellshock.

Command Prompt (cmd.exe):

Ограниченная безопасность**:** Command Prompt имеет ограниченные возможности по работе с криптографией и безопасностью, что делает его менее гибким в плане защиты данных по сравнению с PowerShell.

Устаревшие команды**:** Использование старых команд в cmd.exe может привести к уязвимостям, таким как утечка данных через команды, которые предназначены для отображения содержимого конфиденциальных файлов.

Уязвимости в BAT-файлах**:** Скрипты в формате .bat могут быть легко выполнены в cmd.exe, что делает систему уязвимой для внедрения вредоносного кода, если эти файлы не проверяются должным образом.

Интеграция с PowerShell**:** Поскольку cmd.exe поддерживает интеграцию с PowerShell, уязвимости в PowerShell могут быть использованы через cmd.exe.

Bourne Shell (sh):

Инъекции команд**:** Подобно Bash, Bourne Shell уязвим к инъекциям команд, если данные не проверяются корректно, что может привести к выполнению произвольного кода.

Скрипты с привилегиями**:** Скрипты, выполняющиеся с правами суперпользователя, могут стать источником уязвимостей, особенно если они содержат ошибки или неправильно настроены.

Переменные среды**:** В Bourne Shell также используются переменные среды для хранения конфиденциальных данных, что повышает риск утечек, если они не защищены должным образом.

Аутентификация и шифрование**:** Современные версии Bourne Shell могут использовать инструменты, такие как SSH, для улучшения безопасности и аутентификации в облачных системах.

Общие угрозы для всех оболочек:

Удаленный доступ: Все оболочки подвергаются риску удаленного выполнения команд, что требует надежной настройки межсетевых экранов, использования шифрования и многофакторной аутентификации.

Уязвимости в скриптах**:** Ошибки в скриптах могут привести к утечке данных или выполнению вредоносного кода.

Инъекции команд**:** Командные инъекции являются одной из самых распространенных угроз, что может повлиять на безопасность всей системы.

Привилегированные права**:** Все оболочки требуют строгого контроля за использованием суперпользовательских прав, чтобы предотвратить утечку данных или компрометацию системы.

Перспективы дальнейших исследований

В рамках дальнейших исследований можно рассмотреть следующие направления:

Изучение новых оболочек и операционных систем: Оценка производительности новых или менее популярных оболочек и операционных систем в контексте облачных вычислений, а также их безопасности.

Оптимизация существующих решений**:** Разработка методов, направленных на повышение эффективности и безопасности текущих оболочек и операционных систем.

Влияние аппаратного обеспечения**:** Исследование воздействия различных типов

аппаратного обеспечения на производительность и безопасность облачных сервисов.

Использование искусственного интеллекта**:** Применение методов машинного обучения и ИИ для автоматизации задач администрирования и оптимизации процессов управления облачными вычислениями.

Заключение

В данной статье, авторами было рассмотрено влияние операционных систем и командных оболочек на облачные вычисления. Операционные системы, Linux и Microsoft Windows, играют важную роль в облачной инфраструктуре. Они обеспечивают стабильность и эффективное использование ресурсов. Командные оболочки, Bash, Bourne Shell (sh), Command Prompt (cmd.exe) и PowerShell, предоставляют различные средства для автоматизации задач и управления облачными ресурсами.

Авторами был проведён анализ основных особенностей каждой операционной системы и командной оболочки, а также были представлены примеры скриптов для работы с облачными вычислениями. Было показано, что выбор операционной системы и командной оболочки может существенно повлиять на скорость выполнения облачных вычислений и управления облачной инфраструктурой.

Если сравнивать коды по критерием, то самой простой и эффективной оболочкой является Windows PowerShell на операционной системе Microsoft Windows. Она позволяет писать лаконичные и простые для понимания коды.

Если говорить про синтаксически обширную оболочку и операционную систему, то тут преимущество будет у оболочек Bash и Bourne Shell (sh) на операционной системе Linux.

Так же авторами была разработана таблица (Рис.3) с необходимыми в результате анализа критериями с использованием командных оболочек которые использовались в этапах исследования:

· Bash, под управлением ОС Linux;

· Bourne Shell под управлением ОС Linux;

· Windows PowerShell под управлением ОС Microsoft Windows;

· Command Prompt под управлением ОС Microsoft Windows.

Проведена оценка на соответствие следующим критериям в оболочках:

· Лаконичность;

· Простота синтаксиса;

· Объёмность кода;

· Удобство использования.

Критерий/Операционная система, оболочка	Оболочка Bash, операционная система Linux	Оболочка Bourne Shell (sh), операционная система Linux	Оболочка Windows PowerShell, операционная система Microsoft Windows	Оболочка Command Prompt (cmd.exe), операционная система Microsoft Windows
Лаконичность	-	-	+	-
Простота синтаксиса	-	-	+	+
Объёмность кода	+	+	+	+
Удобство использования	-	+	+	+

Рисунок 3. Критерии соответствия

Таким образом, понимание роли операционных систем и командных оболочек в облачных вычислениях имеет важное значение для администраторов систем, разработчиков и всех заинтересованных лиц в сфере информационных технологий.

Библиография

1. Савельев Д.Н., Гаврилов С.В. Роль операционных систем в облачных вычислениях: вызовы и перспективы / Савельев Д.Н., Гаврилов С.В. [Электронный ресурс] // Электронный периодический научный журнал sci-article.ru : [сайт]. – URL: https://sci-article.ru/stat.php/stat.php?i=1697382177 (дата обращения: 15.04.2024).
2. Трубачева С.И. Почему Linux и системы реального времени? [Текст] / Трубачева С.И. // Вестник Волжского университета им. В.Н. Татищева. – 2015. – № 2(24). – С. 99-105.
3. Колосов, Л. С., Умаралиев, И. В. Обзор эволюции функционала операционной системы GNU/Linux при эволюции ядра [Текст] / Л. С. Колосов, И. В. Умаралиев // Научный аспект. – 2023. – № 7. – С. 1390-1393.
4. Passos L., Czarnecki K., Wasowski A. (2012). Towards a catalog of variability evolution patterns: the Linux kernel case. In FOSD '12 Proceedings of the 4th International Workshop on Feature-Oriented Software Development (pp. 62-69). Association for Computing Machinery.
5. Upasana Why do you need the different Linux Shells? / Upasana [Электронный ресурс] // edureka : [сайт]. – URL: https://www.edureka.co/blog/types-of-shells-in-linux/#differenttypesofshells (дата обращения: 18.04.2024).
6. siberianMan О разных командных оболочках Linux и Unix / siberianMan [Электронный ресурс] // Хабр : [сайт]. – URL: https://habr.com/ru/articles/157283/ (дата обращения: 18.04.2024).
7. Леонтьев, В. О., Великосельский, С. А. Семейство Microsoft Windows [Текст] / В. О. Леонтьев, С. А. Великосельский // Аллея Науки. – 2018. – № 3(19). – С. 727-729.
8. Влад Массино, Арам Папоян Три десятилетия Windows / Влад Массино, Арам Папоян [Электронный ресурс] // газета.ru : [сайт]. – URL: https://www.gazeta.ru/tech/2015/11/19/7902437/windows_30th_anniversary.shtml (дата обращения: 18.04.2024).
9. Gaurav Bidasaria Command Prompt vs PowerShell vs Windows Terminal: How They Differ / Gaurav Bidasaria [Электронный ресурс] // techwiser : [сайт]. – URL: https://techwiser.com/command-prompt-vs-powershell-vs-windows-terminal-comparison/ (дата обращения: 18.04.2024).
10. ru_vds Что такое Windows PowerShell и с чем его едят? Часть 1: основные возможности / ru_vds [Электронный ресурс] // Хабр : [сайт]. – URL: https://habr.com/ru/companies/ruvds/articles/487876/ (дата обращения: 18.04.2024)
11. sharkdp Hyperfine releases v.1.19.0 / sharkdp [Электронный ресурс] // github : [сайт]. – URL: https://github.com/sharkdp/hyperfine (дата обращения: 01.12.2024)

Результаты процедуры рецензирования статьи

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

Статья представляет интересный анализ операционных систем и командных оболочек. Основная часть статьи начинается с исторического обзора операционной системы Linux, отмечая ее значимость и активное участие сообщества разработчиков. Затем

рассматриваются преимущества Linux, в том числе открытый исходный код и взаимодействие с оболочкой Bash. Также выполнен обзор операционных систем Microsoft Windows и Linux, а также их командных оболочек, с акцентом на их роль в облачных вычислениях. Текст содержит информацию о истории и особенностях различных версий Windows, включая Windows 8, Windows 10 и Windows 11. Также рассматривается командная строка Command Prompt и Windows PowerShell, их функциональность и преимущества.

Приведенные примеры кода на Bash позволяют читателям лучше понять, как данная оболочка может быть использована для автоматизации задач управления облачными ресурсами. Примеры ясно демонстрируют функциональность Bash и его преимущества при работе с облачной инфраструктурой.

Общие замечания:

Во-первых, в статье утверждается, что оболочка Bourne Shell (sh) была разработана Стивом Борном. Однако фактически она была создана Кеном Томпсоном.

Далее, некоторые недостатки оболочки Bourne Shell, перечисленные в статье, могут быть недостоверными или устаревшими, так как она является одной из первых оболочек UNIX и в последующих версиях могли быть устранены некоторые из них.

Также следует отметить, что в разделе о Windows PowerShell неправильно утверждается, что командная оболочка была создана на основе платформы .NET Framework. Фактически, она была разработана с использованием языка программирования C# и базируется на .NET Framework или .NET Core в зависимости от версии.

Раздел "Заключение" содержит общие выводы о выборе операционной системы и командной оболочки, но не предоставляет новых исследовательских данных или выводов, которые бы уточнили значимость рассмотренных аспектов для облачных вычислений.

Отдельно хотелось бы отметить несколько ошибок и неточностей в тексте:

1. В некоторых местах статьи присутствуют ошибки в написании слов и использовании пунктуации, что может затруднить понимание текста: например: "на Bourne Shell (sh) становится простим и …".
2. В разделе о Windows PowerShell некорректно указаны примеры скриптов для работы с облаком Mail.ru. Они дублируются и содержат ошибку в одной из команд - вместо "mcloud vm stop" должно быть "mcloud vm stop".
3. Некоторые изображения, такие как Рис 1, могут быть уменьшены до размеров полезного текста на скриншоте, с целью показа существенной информации без лишнего пространства, заполненного черным цветом или Рис.2 – Логотип PowerShell, который может быть ненужным и не добавлять ценности к тексту.

Заключительное замечание о том, что исследование в области облачных вычислений продолжается, является очевидным и не добавляет информативности.

В целом, статья представляет интересную информацию о роли операционных систем и командных оболочек в облачных вычислениях, но требует доработки с точки зрения правильности изложения и добавления новых выводов на основе проведенного анализа, а также обзора современных технологий, таких как специализированные операционные системы, разработанные для запуска контейнеров, которые становятся все более популярными в облачных средах, и облачные платформы, такие как Amazon Web Services (AWS), Google Cloud Platform (GCP) и Microsoft Azure, которые предлагают разнообразные операционные системы для развертывания в облаке.

Результаты процедуры повторного рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не

раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Предмет исследования. Название статьи свидетельствует о том, что она должна быть посвящена роли операционных систем и оболочек в облачных вычислениях: анализ ОС и оболочек, используемых в облачных платформах и их влияние на облачную инфраструктуру. Содержание статьи не противоречит заявленной теме, но и не раскрывает её полностью, о чём более подробно будет сказано в остальных блоках настоящей рецензии.
Методология исследования базируется преимущественно на изложении общеизвестных фактов и суждений. В отдельных фрагментах статьи применяются методы анализа и синтеза данных. Также автор использует графический инструментарий, однако неясно, что хотел сказать автор, приводя рисунок 1? В тексте статье есть только информация о том, что приведено на рисунке 1, но зачем это представлено и какие выводы делает автор читатель из текущей редакции узнать не сможет. В свою очередь, рисунок 2 представляет, по сути, таблицу, имеющую заголовок «критерии», но неясно критерии осуществления какого действия автор привёл в данной таблице. Также под графическими объектами не приведены источники, что снижает впечатление от ознакомления с рецензируемой статьёй.
Актуальность исследования вопросов, связанных с роль. операционных систем и оболочек в облачных вычислениях: анализ ОС и оболочек, используемых в облачных платформах и их влияние на облачную инфраструктуру, не вызывает сомнения, т.к. это одна из активно обсуждаемых тем в профессиональном сообществе.
Научная новизна в представленном на рецензирование материале не обнаружена.
Стиль, структура, содержание. Стиль изложения научный с точки зрения используемых слов и словосочетаний, но при этом научные методы исследования практически не применяются. Структура статьи автором не выстроена. При проведении доработки статьи рекомендуется использовать следующие компоненты: «Введение», «Постановка проблемы», «Методы и условия исследования», «Результаты исследования», «Обсуждение результатов исследования», «Выводы и дальнейшие направления исследования». Следование данной структуры одновременно решит несколько задач: во-первых, внесёт позитивный вклад в формирование отсутствующей в текущей редакции научной новизны; во-вторых, значительно расширит потенциальную читательскую аудиторию. Также автору рекомендуется обозначить выявленные по итогам исследования проблемы и сформировать комплекс обоснованных рекомендаций по их решению.
Библиография. Библиографический список включает 10 наименований. Ценно, что автор опирается не только на отечественные научные публикации, но и на зарубежные. Это позволяет учесть множество мнений по предмету исследования. Однако 10 источников не является достаточным для всестороннего изучения палитры существующих подходов к исследованию заявленных в заголовке вопросов. Поэтому при проведении доработки статьи рекомендуется увеличить количество источников.
Апелляция к оппонентам. Несмотря на сформированный список источников, какой-либо научной дискуссии в тексте не обнаружено. При проведении доработки и увеличении числа источников рекомендуется обсудить полученные результаты с теми итогами исследований, что содержатся в научных трудах других авторов. Это также окажет позитивное воздействие на формирование отсутствующей в текущей редакции научной новизны.
Выводы, интерес читательской аудитории. В текущей редакции статья не будет востребована у читательской аудитории, но при проведении глубокой содержательной

доработки с учётом высокой актуальности выбранной темы исследования она будет пользоваться спросом у широкого круга лиц.

Результаты процедуры окончательного рецензирования статьи

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

В статье рассматривается влияние различных операционных систем и командных оболочек на облачные вычисления. Авторы анализируют использование ОС Linux, Microsoft Windows и различных оболочек (Bash, Bourne Shell, Command Prompt, Windows PowerShell) в контексте их применения в облачной инфраструктуре. В современных условиях развития информационных технологий облачные вычисления становятся неотъемлемой частью инфраструктуры множества компаний и организаций. Они позволяют оптимизировать затраты на оборудование, упростить управление ИТ-ресурсами и повысить гибкость и масштабируемость систем. Облака предоставляют возможность пользователям получать доступ к вычислительным мощностям и сервисам по запросу, что особенно важно для быстрого реагирования на изменения в бизнесе и для поддержки инноваций. Исследование основано на сравнительном анализе операционных систем и оболочек, с использованием практических примеров скриптов для создания и управления виртуальными серверами в облачной среде. Авторы приводят коды скриптов, что позволяет наглядно оценить возможности и особенности каждой системы и оболочки. Облачные вычисления становятся все более важными в современном ИТ-ландшафте. Правильный выбор операционной системы и командной оболочки может значительно повлиять на производительность, безопасность и удобство управления облачными ресурсами. Статья актуальна для администраторов систем, разработчиков и всех, кто работает с облачными технологиями. Научная новизна статьи заключается в структурированном подходе к сравнению операционных систем и командных оболочек в контексте их использования в облачных вычислениях. Авторы предоставляют конкретные примеры скриптов, что является ценным вкладом в практическое применение теоретических знаний. Стиль изложения научный, с акцентом на практическую применимость представленных данных. Структура статьи логична и последовательна: введение, описание операционных систем и оболочек, примеры использования, заключение. Содержание статьи полное и соответствует заявленной теме. Авторы делают вывод о значительном влиянии выбора операционной системы и командной оболочки на эффективность облачных вычислений. Приведенные примеры скриптов демонстрируют практическую ценность проведенного исследования. Статья представляет интерес для широкой аудитории, связанной с ИТ и облачными технологиями. Это включает системных администраторов, разработчиков, исследователей и студентов.