УДК 004.42

Рындов Сергей Николаевич,
Нижегородский государственный технический
университет им. Р.Е. Алексеева
Ryndov Sergey Nikolaevich,
R.E. Alekseev Nizhny Novgorod State Technical University

АВТОМАТИЗАЦИЯ НАСТРОЙКИ ВИРТУАЛЬНЫХ МАШИН
AUTOMATING THE CONFIGURATION OF VIRTUAL MACHINES

Аннотация. Автоматизация настройки виртуальных машин является важной частью работы с инфраструктурой.

В данной статье рассматривается методы и инструменты автоматизации виртуальных машин, их преимущества и недостатки. Необходимо определить наиболее подходящий метод подготовки вычислительных ресурсов, в зависимости требований и доступных ресурсов.

Abstract. Automating the configuration of virtual machines is an important part of working with the infrastructure.

This article discusses methods and tools for automating virtual machines, their advantages and disadvantages. It is necessary to determine the most appropriate method of preparing computing resources, depending on the requirements and available resources.

Ключевые слова: Ansible, AWX, Terraform, Jenkins, инфраструктура, виртуальные машины.

Keywords: Ansible, AWX, Terraform, Jenkins, pipeline, infrastructure, virtual machines.

Введение

Одна из основных архитектурных потребностей современных IT компаний связана с созданием процесса выдачи виртуальных вычислительных ресурсов в виде виртуальных машин. Виртуальные машины позволяют разработчикам и инженерам создать изолированную и удобную с точки зрения администрирования среду для разворачивания сервисов, способных удовлетворить как внутренние, так и внешние запросы бизнеса. Кроме того, виртуальные машины позволяют протестировать изменения программного кода или конфигурации готового программного продукта в безопасном окружении, перед этапом поставки изменений на главные сервера. Один из наиболее важных аспектов создания ВМ – это настройка системы, включая установку операционной системы, конфигурацию сети, настройку безопасности и установку необходимых базовых приложений.

Существует множество методов автоматизации настройки виртуальных машин, и выбор конкретного метода зависит от уникальных потребностей и требований каждой компании.

В ситуациях, когда возникают единичные запросы на создание виртуальных машин, настройка может осуществляться вручную. Такой подход может быть оправдан в небольших масштабах инфраструктуры или в случаях, когда требуется быстрая настройка без необходимости углубления в сложные процессы автоматизации. Однако, когда речь идет о более крупных масштабах инфраструктуры или регулярном создании виртуальных машин, ручная настройка становится менее эффективной и может привести к ошибкам, связанным с человеческим фактором.

В таких случаях автоматизация настройки виртуальных машин становится не только предпочтительным, но и необходимым шагом. Автоматизированные процессы позволяют значительно сократить время на развертывание, минимизировать вероятность ошибок и повысить общую надежность системы.

Помимо создания и настройки виртуальных машин, разработчикам и инженерам часто требуется уже предустановленные и предварительно настроенные сервисы. В современном мире, где скорость и эффективность являются ключевыми факторами успеха, важно не только предоставить разработчикам виртуальные машины, но и обеспечить их необходимыми инструментами и сервисами, которые позволят им сосредоточиться на решении задач, а не на конфигурации окружения.

Для достижения этой цели процесс автоматизации связывается с инструментами непрерывной интеграции. Эти инструменты предоставляют разработчикам и инженерам гибкие возможности выбора параметров, что позволяет настроить виртуальные машины и сопутствующие сервисы в соответствии с конкретными требованиями проекта.

«Ручная и автоматизированная настройка виртуальных машин»

Ручная настройка виртуальных машин представляет собой важный этап подготовки безопасной, гибкой и удобной виртуальной среды [1], который актуален в различных сценариях и практиках. В первую очередь, такая настройка становится необходимой в тех случаях, когда существуют специфические требования к конфигурации. Это может включать необходимость в определенных аппаратных ресурсах [2], операционных системах или программном обеспечении, что позволяет точно настроить параметры виртуальной машины под конкретные нужды пользователя или организации.

Такой способ настройки виртуальных машин особенно актуальна в контексте тестирования и разработки. Для разработчиков и тестировщиков ручная настройка виртуальных машин предоставляет возможность создания конкретных тестовых сред, которые могут включать в себя установку различных версий операционных систем для проверки совместимости, а также настройку программного обеспечения, сети [3] и библиотек, необходимых для разработки. Ручная настройка виртуальных машин позволяет экспериментировать с конфигурациями без риска повредить основную систему, изучать настройки сети, безопасности и другие аспекты виртуализации.

Однако, в условиях современных потребностей ИТ компаний, автоматизация процесса настройки виртуальных машин приобретает все большее значение [4]. Автоматизация настройки виртуальных машин позволяет значительно ускорить процесс развертывания. Вместо того чтобы тратить значительное время на выполнение рутинных операций вручную, можно использовать скрипты и инструменты автоматизации, которые выполняют необходимые действия в кратчайшие сроки. Это особенно важно в условиях, когда требуется быстрое развертывание среды для тестирования [5] или разработки [6].

Одним из ключевых преимуществ автоматизации является возможность легкой повторяемости процессов [7]. Инструменты автоматизации позволяют создавать идентичные виртуальные среды для тестирования, разработки или других нужд IT компаний, что критически важно для обеспечения стабильности и предсказуемости результатов.

Автоматизация упрощает управление изменениями в конфигурациях [8] виртуальных машин. При возникновении новых требований, обновлений или изменений в инфраструктуре, автоматизированные процессы позволяют быстро адаптироваться без необходимости повторной ручной настройки. Это значительно повышает гибкость и адаптивность виртуальной среды.

Автоматизация также облегчает масштабирование инфраструктуры. В ситуациях, когда необходимо развернуть большое количество виртуальных машин с одинаковыми настройками [9], автоматизированные процессы позволяют реализовать это быстро и эффективно, что невозможно или крайне затруднительно при ручной настройке.

Одна из наиболее значимых возможностей, которые предоставляет автоматизация настройки виртуальных машин – это подход "Инфраструктура как код". Данный подход

используют применяют большинство инструментов автоматизации. Подход "Инфраструктура как код" [10] стал важным элементом современного управления настройкой конфигурации, особенно в контексте автоматизации настройки виртуальных машин. Этот метод позволяет разработчикам и операционным командам управлять и развертывать инфраструктуру с помощью программного кода, что значительно упрощает процессы, повышает их эффективность и снижает вероятность ошибок.

В большинстве случаев реализация данного подхода основывается на декларативном подходе, что означает, что пользователи описывают желаемое состояние, а инструменты автоматизации заботятся о том, чтобы это состояние было достигнуто. Это позволяет избежать рутинных задач и сосредоточиться на более важных аспектах работы [11].

Использование подхода "Инфраструктура как код" позволяет применять методы управления версиями для конфигурационных файлов, что делает возможным отслеживание изменений и возврат к предыдущим версиям при необходимости. Это особенно важно в условиях быстро меняющегося бизнеса, где требования могут изменяться на лету. Кроме того, благодаря читаемости кода, команды могут эффективно работать над одной и той же конфигурацией, улучшая совместную работу и уменьшая вероятность конфликтов.

Важным аспектом автоматизации настройки виртуальных машин с использованием подхода "Инфраструктура как код" является управление конфиденциальной информацией, такой как пароли и ключи доступа. Современные инструменты автоматизации предлагают интеграцию с системами управления секретами, что позволяет безопасно хранить и использовать эти данные [12], минимизируя риски утечек и обеспечивая безопасность.

«Автоматизация при помощи Ansible»

Одним из наиболее часто применяемым инструментом для автоматизации рутинных задач на удалённых хостах является Ansible – это инструмент автоматизации [13], разработанный для упрощения процессов развертывания, управления конфигурациями и оркестрации приложений. Он использует декларативный подход, это означает, что пользователи описывают желаемое состояние системы, а Ansible заботится о том, чтобы это состояние было достигнуто. Ansible работает по принципу "agentless", что означает, что для его работы не требуется установка дополнительных агентов на целевых машинах. Вместо этого Ansible использует SSH для подключения к удаленным системам, что упрощает процесс настройки и управления.

Декларативный подход, которым оперирует инструмент Ansible [14] позволяет пользователям сосредоточиться на конечной цели, а не на деталях выполнения. Ansible автоматически определяет, какие изменения необходимо внести, чтобы привести систему в соответствие с описанным состоянием, что снижает вероятность ошибок.

Ansible использует формат файлов YAML (Yet Another Markup Language) [15]. Этот формат легко читаем и понятен, что позволяет даже новичкам быстро освоить его. Читаемость кода упрощает совместную работу в команде.

Для использования инструмента Ansible необходимо создать два ключевых элемента: файлы инвентори и плейбук [16]. Файл инвентори играет важную роль, так как в нем описываются хосты, которые будут приводиться в требуемое состояние. Для успешного подключения к хостам, перечисленным в инвентори файле, необходимо обеспечить возможность подключения по протоколу SSH. Это важное условие [17], так как именно через протокол SSH инструмент Ansible взаимодействует с удаленными системами. Данные для подключения, такие как адреса хостов, имена пользователей и пароли, могут быть указаны как в самом файле инвентори, так и в плейбук файле, что предоставляет гибкость в настройке и управлении.

Плейбук, в свою очередь, является основным инструментом для описания параметров работы и запуска Ansible ролей. В нем подробно прописываются все необходимые шаги и

параметры, которые будут использоваться при выполнении автоматизации. Плейбук определяет последовательность действий, которые Ansible должен выполнить на целевых хостах, а также может включать переменные и условия, что делает его мощным инструментом для управления конфигурацией.

Ключевым компонентом Ansible является роль, которая представляет собой способ организации и структурирования декларированного кода [18]. Роли позволяют разбивать плейбуки на более мелкие, легко управляемые и повторно используемые компоненты. Это делает код более модульным и упрощает его поддержку. Все требуемые состояния систем описываются с помощью модулей, которые предоставляют конкретные функции и действия, необходимые для достижения желаемого результата.

Ansible роли могут объявлять зависимости от других ролей, что упрощает управление сложными проектами. Тестирование созданных ролей может происходить отдельно друг от друга, что упрощает непосредственно процесс тестирования и отладки.

Главное преимущество инструмента Ansible заключается в его обширной библиотеке модулей, которые предоставляют возможность приводить в необходимое состояние различные системные компоненты, а также приложения, установленные на различных системах. Эта многообразная коллекция модулей позволяет пользователям эффективно управлять инфраструктурой, автоматизируя рутинные задачи и конфигурируя приложения с минимальными усилиями.

Ansible поддерживается активным международным сообществом, которое не только постоянно работает над улучшением самого инструмента, но и создает новые модули, расширяющие его функциональность. Это сообщество является важным ресурсом, который способствует обмену знаниями и опытом, а также поддерживает пользователей на всех этапах работы с Ansible.

В случае, если в существующей библиотеке модулей не удается найти подходящий для настройки виртуальной машины или приложения, установленного на ней, Ansible предоставляет пользователям возможность создавать собственные модули [19].

«Автоматизация при помощи Terraform»

Альтернативным, но не менее востребованным инструментом для настройки виртуальных машин является Terraform [20] – инструмент, разработанный для автоматизации развертывания и управления облачными ресурсами. Terraform использует декларативный подход. Это обеспечивает высокую степень предсказуемости и надежности в управлении ресурсами.

Одной из ключевых особенностей Terraform является его способность работать с множеством облачных платформ [21], таких как AWS, Google Cloud, Azure и многие другие. Интеграция с облачными платформами предоставляет разработчикам возможность более гибкой настройки виртуальных машин, на уровне управления вычислительными ресурсами. Это делает инструмент Terraform универсальным решением [22] для управления ресурсами многих IT компаний занимающимся предоставлением виртуальных машин.

Terraform использует язык конфигурации HashiCorp Configuration Language, который отличается простотой и читаемостью. Благодаря этому даже новички могут быстро освоить основы работы с инструментом. Читаемость конфигурационных файлов облегчает совместную работу в команде и упрощает процесс ревью кода. Данный язык имеет сходство с языком YAML. Они используют интуитивно понятный синтаксис, что облегчает работу с конфигурационными файлами. Данные языки одинаково поддерживают структурирование данных в виде вложенных объектов, списков и ключ-значение пар. Это позволяет быстро освоится с синтаксисом языка HCL для логичной и понятной организации нужной конфигураций.

Для эффективного использования Terraform необходимо создать конфигурационные файлы [23], в которых описываются ресурсы, которые необходимо создать или изменить. Эти файлы могут включать в себя не только описание самих ресурсов, но и их взаимосвязи, что позволяет Terraform автоматически определять порядок выполнения операций. Это значительно сокращает время и усилия, необходимые для управления сложными инфраструктурами.

Terraform также поддерживает концепцию модулей, что позволяет пользователям организовывать и повторно использовать код. Модули представляют собой наборы ресурсов, которые можно использовать в различных проектах, что способствует модульности и упрощает поддержку кода. Кроме того, модули могут иметь свои зависимости, что позволяет управлять сложными проектами более эффективно.

Главное преимущество Terraform заключается в его способности управлять состоянием инфраструктуры. Инструмент хранит текущее состояние ресурсов в специальном файле состояния, что позволяет отслеживать изменения и предотвращать конфликты. При каждом применении конфигурации Terraform сравнивает текущее состояние с желаемым, автоматически определяя необходимые изменения и минимизируя риск ошибок.

Terraform поддерживается активным сообществом пользователей и разработчиков, которое постоянно работает над улучшением инструмента и созданием новых провайдеров и модулей. Это сообщество является важным ресурсом для обмена знаниями и опытом, а также для поддержки пользователей на всех этапах работы с Terraform.

В случае, если в существующих провайдерах не удается найти необходимый ресурс, Terraform предоставляет возможность создания собственных провайдеров [24].

Инструмент Terraform также предлагает возможность интеграции с другими инструментами и системами, что делает его еще более мощным в контексте DevOps практик. Например, Terraform может быть интегрирован с CI/CD системами [25], такими как Jenkins, GitLab CI и GitHub Actions, что позволяет автоматизировать процесс развертывания приложений и управления инфраструктурой. Это позволяет командам быстрее реагировать на изменения и улучшать качество развертывания.

Кроме того, Terraform поддерживает концепцию "планирования" [26] изменений перед их применением. Команда может использовать команду terraform plan, чтобы увидеть, какие изменения будут внесены в инфраструктуру, прежде чем они будут фактически применены.

Несмотря на все преимущества, работа с Terraform требует определенных знаний и навыков. Пользователи должны понимать основные принципы работы с облачными провайдерами, а также быть знакомыми с концепциями инфраструктуры как кода. Однако благодаря обширной документации, примерам и активному сообществу, новички могут быстро освоить основы и начать эффективно использовать Terraform в своих проектах.

«AWX»

Для обеспечения пользователей возможностью регулярной настройки виртуальных машин требуется использование дополнительных приложений, способных упростить данный процесс. Пользователям, не знакомым с особенностями автоматизации настройки виртуальных машин, важно иметь четкое представление о том, какой результат они могут ожидать от работы созданной автоматизированной систмы. Важно, чтобы они имели возможность управлять параметрами процесса настройки виртуальных машин через интуитивно понятный и удобный графический интерфейс.

Одним из таких приложений является AWX – гибкое веб-приложение, которое предоставляет графическую оболочку для работы с Ansible. Основная логика AWX построена на базе Django, что не только обеспечивает высокую надежность, но и масштабируемость приложения, позволяя ему эффективно справляться с растущими требованиями пользователей.

Приложение AWX позволяет пользователям легко управлять инвентаризациями [27], которые представляют собой списки управляющихся узлов. Пользователи могут импортировать инвентаризации из различных источников, таких как динамические инвентаризации AWS, Azure и другие облачные провайдеры.

AWX разворачивается на сервере [28], и все взаимодействие с этим приложением осуществляется через графический интерфейс в браузере. Это позволяет пользователям легко и быстро настраивать виртуальные машины без необходимости погружаться в сложные технические детали. Кроме того, AWX поддерживает интеграцию с LDAP-системами, что позволяет синхронизировать учетные данные сотрудников с системой AWX.

Для запуска автоматизированного процесса настройки виртуальных машин в AWX создаётся проект, который может быть привязан к Git репозиторию [29], содержащему актуальные версии Ansible ролей. Этот функционал позволяет пользователям легко управлять версиями и обновлениями своих конфигураций, обеспечивая актуальность используемых инвентори и плейбук файлов.

В рамках созданного проекта формируются шаблоны заданий (job templates) [30], которые представляют собой графическую интерпретацию Ansible плейбуков. Эти шаблоны служат удобным инструментом для автоматизации задач, позволяя пользователям без необходимости погружаться в код плейбуков запускать необходимые операции. Кроме того, job шаблоны могут быть синхронизированы с Ansible плейбуками, хранящимися в Git репозитории, что обеспечивает согласованность между визуальными настройками и фактическими конфигурациями.

При настройке job-шаблона пользователю необходимо выбрать инвентори. В AWX инвентори представлен как отдельная сущность, что позволяет ей быть гибкой и многофункциональной. Инвентаризация может быть синхронизирована с файлом Ansible инвентаризации, хранящимся в Git репозитории, что упрощает управление списками узлов и их параметрами. Такой подход обеспечивает не только удобство, но и повышает надежность и согласованность процессов автоматизации, позволяя пользователям легко адаптировать свои настройки в зависимости от изменяющихся требований и условий работы.

Для подключения к хостам при запуске Ansible ролей через приложение AWX существует возможность указать данные для подключения непосредственно в job шаблонах. Однако приложение AWX предлагает более безопасный и продуманный подход – создание и конфигурирование полномочий (credentials) [31]. Полномочия представляют собой набор информации, необходимой для аутентификации и авторизации при выполнении задач, обеспечивая надежный доступ к управляющим узлам и другим системам.

Учетные данные могут включать в себя различные элементы, такие как пароли, SSH-ключи, токены API и другие данные, которые позволяют AWX устанавливать соединение с облачными провайдерами или удалёнными хостами. Созданные полномочия можно легко выбрать в настройках соответствующего job шаблона, что упрощает процесс конфигурации и повышает безопасность.

Одним из главных преимуществ такого подхода является возможность создания отдельной логики, ориентированной на сбор секретной информации на уровне сервиса, без необходимости вовлечения пользователя. Это позволяет минимизировать риск утечки конфиденциальных данных и обеспечивает более высокий уровень безопасности. Поскольку AWX поддерживает интеграцию с сервисами хранения секретов, такими как HashiCorp Vault, построение логики сбора и управления секретами становится гибким и удобным. Это открывает возможности для автоматизации, позволяя пользователям сосредоточиться на выполнении задач, не беспокоясь о безопасности своих учетных данных.

«Jenkins»

Потребности IT-компаний зачастую выходят за рамки простых и единичных случаев настройки виртуальных машин. Многие из таких компаний специализируются на выдачи вычислительных ресурсов своих серверов, что подразумевает регулярный ввод и вывод из эксплуатации виртуальных машин.

В данной специфике работы, простое логирование процесса настройки виртуальных машин оказывается недостаточным. Крайне важно собирать и анализировать более широкий спектр информации. Систематический сбор подобной информации позволяет не только оптимизировать процесс управления виртуальными машинами, но и улучшить качество обслуживания клиентов, обеспечивая более гибкое и эффективное распределение ресурсов.

Для технической реализации таких комплексных потребностей IT компаний необходим полноценный инструмент непрерывной интеграции [32]. Одним из таких инструментов является Jenkins – инструмент для автоматизации процессов разработки, развертывания программного обеспечения и управления жизненным циклом приложений, который широко используется в методологиях непрерывной интеграции.

Инструмент Jenkins имеет большое активное международное сообщество, которое специализируется в различных сферах автоматизации. Сообщество Jenkins не только активно разрабатывает и поддерживает плагины, расширяющие функциональность инструмента Jenkins, но и делится опытом, знаниями и лучшими практиками, что делает его ценным ресурсом для разработчиков и администраторов.

В основе работы инструмента непрерывной интеграции Jenkins лежат пайплайны [33], настроенные на автоматизацию рутинных задач. Автоматизированные пайплайны Jenkins позволяют быстро реагировать на изменения в запросах пользователей, обеспечивая более гибкое и эффективное распределение ресурсов.

Инструмент Jennkins позволяет расширить потенциал автоматизации различных процессов, применяя модули. С помощью модулей можно интегрировать Jenkins с различными системами, инструментами, библиотеками и фреймворками, а также настраивать его поведение под нужды конкретного проекта или команды. инструмента непрерывной интеграции Jenkins имеет более 2000 доступных к использованию модулей, которые позволяют настраивать созданные пайплайны, улучшать взаимодействие с системами контроля версий, интегрировать инструменты для тестирования, развертывания и мониторинга, а также расширять возможности визуализации и управления.

Главным и наиболее часто применяемым модулем является Pipeline Plugin [34]. Данный плагин предоставляет инструментарий для создания сложных мультиэтапных пайплайнов. Он позволяет описывать логику автоматизации процессов настройки виртуальных машин в виде кода, что обеспечивает высокую гибкость и простоту в управлении.

Для создания пайплайна необходимо разработать и оформить логику автоматизации процессов в специальном конфигурационном Jenkinsfile файле. Данный файл является основным элементом, позволяющим Jenkins осуществлять построение, тестирование и развертывание приложений в автоматизированном режиме. Внутри Jenkinsfile программный код пишется на языке Groovy, который представляет собой динамический язык программирования, синтаксически схожий с JavaScript. Благодаря этому сходству, разработчики, обладающие опытом работы с JavaScript, быстро адаптируются к спецификациям Groovy. Это делает Groovy удобным инструментом для профессионалов в области автоматизации.

Другим не менее важным модулем, предоставляющий необходимый для автомазиции настройки виртуальных машин функцианал является модуль Docker Plugin [35]. С помощью этого модуля разработчики могут интегрировать инструмент непрерывной интеграции Jenkins

с Docker, что предоставляет возможность контейнеризации приложений для автоматизации настройки виртуальных машин.

Docker – это платформа и инструмент, предназначенные для разработки, развертывания и управления контейнеризованными приложениями. Основная концепция Docker заключается в использовании контейнеров [36], которые предоставляют изолированные среды для запуска приложений, обеспечивая при этом удобство и гибкость. Контейнеры представляют собой легковесные, самостоятельные единицы, в которых находятся не только приложения, но и все их зависимости, что позволяет избежать проблем совместимости между различными средами. Для создания контейнеров в Docker используют образы – готовые для использования шаблоны, которые содержат все необходимые компоненты для запуска приложения. Эти образы создаются на основе инструкции, прописанных в файлах Dockerfile, где указываются все необходимые шаги для сборки приложения, включая установку библиотек, настройку окружения и другие важные параметры.

Интеграция инструментов Jenkins и Docker в процесс автоматизации открывает новые возможности для разработчиков. Благодаря интеграции данных инструментов, разработчики могут сосредоточиться на создании и тестировании своих приложений в удобной среде. Процесс начинается с того, что разработчики создают приложение и, прежде чем протестировать его, оборачивают его в Docker контейнер. Это позволяет им обеспечить изоляцию и переносимость приложения, что значительно упрощает его развертывание и тестирование. Тем временем, специалисты, ответственные за автоматизацию, берут на себя задачу настройки виртуальных машин. Используя заранее подготовленный Docker образ, они интегрируют процесс развертывания созданного приложения в автоматизированный рабочий пайплайн настройки виртуальных машин.

Последним модулем, предоставляющий функционал для реализации комплексных запросов современных IT компаний в сфере автоматики настройки виртуальных машин, является Git Plugin [37]. Данный модуль позволяет Jenkins интегрироваться с системами контроля версий, такими как Git.

Многие IT-компании, стремящиеся эффективно управлять версиями своих разработок и заботящиеся о безопасности своих данных, решают развернуть системы управления версиями на основе Git на собственных серверах. Это позволяет им не только организовать хранение и отслеживание изменений в коде, но и предоставляет разработчикам возможность тестирования разрабатываемых приложений непосредственно из своих веток.

Jenkins позволяет выстроить паплайн так, что при настройке виртуальных машин происходит выгрузка репозитория с необходимой веткой, что обеспечивает доступ к актуальной версии кода. После успешной выгрузки репозитория приложение собирается на виртуальной машине. Этот этап включает в себя компиляцию, сборку зависимостей и подготовку к тестированию.

Процесс тестирования программного обеспечения можно эффективно автоматизировать не только на этапе разработки, но и во время слияния ветки разработчика с главной веткой внутреннего Git-репозитория [38]. В этом контексте инструмент Jenkins выступает в роли надежного помощника, предоставляя возможность запускать заранее настроенные пайплайны в момент инициализации запроса на слияние, инициируемого разработчиком. Когда разработчик создает запрос на слияние, Jenkins автоматически активирует соответствующий пайплайн, который включает в себя целый ряд тестов, направленных на проверку целостности, производительности и функциональности кода. После успешного прохождения всех тестов в рамках созданного Jenkins пайплайна, разработчики получают уверенность в том, что их изменения будут стабильно интегрированы в основную ветку.

«Практическая эффективность автоматизации процесса подготовки виртуальных машин»

Для получения количественных показателей эффективности каждого описанного подхода, были проведены замеры времени подготовки виртуальных машин с использованием шести подходов: традиционного ручного метода, автоматизации на основе Ansible, Terraform, а также комплексного решения с применением AWX, Jenkins в совокупности с Docker. В качестве критерия эффективности выступает время, затраченное на подготовку виртуальной машины.

В качестве платформы, на которой проводится замер эффективности, выступает виртуальная машина, созданная на серверной платформе виртуализации Proxmox. Характеристики виртуальной машины описаны в таблице 1.

Таблица 1.

Описание аппаратной составляющих виртуальной машины.

Описание аппаратной составляющей	Численная характеристика
CPU	4 Core
RAM	8 Gbyte
SYS Disk	60 Gbyte

Операционной системой, установленной на виртуальной машине, является OS Rocky Linux. Объектом автоматизации использовались следующие настройки виртуальный машины:

- Настойка hostname, DNS, chrony,
- Замена стандартных конфигурационных файлов с метаданными rpm репозиториев на конфигурации с частным репозиторием,
- Установка и настройка базового ПО: Node exporter, fluenbit, S.M.A.R.T tools,
- Создание и заполнение sudoers.d файла с распределением по сервисам.

При подготовки виртуальной машины следует учитывать специфику её использования. В зависимости от специфики конфигурация выгульной машины будет отличатся. Фиксация времени выполнения настройки виртуальной машины происходило при помощи инструмента time. Получение результаты описаны в таблице 2.

Таблица 2.

Время методов настройки виртуальной машины.

Описание метода	Время настройки виртуальной машины
Ручная настройка	80 мин
Ansible	20 мин
Terraform	21 мин
AWX	31 мин
Ansible + Docker + Jenkins	27 мин
Terraform + Docker + Jenkins	29 мин

Практические результаты демонстрируют, что применение базовых инструментов автоматизации Ansible и Terraform существенно сокращают время подготовки виртуальной машины, в сравнение с ручным методом. Но в тоже время, выстраивание дополнительной инфраструктуры над базовыми инструментами автоматизации способствует повышению времени подготовки виртуальной машины, предоставляя при этом возможность создания дополнительной логики управления процессом подготовки.

Заключение

Описанные в статье практики и инструменты автоматизации настройки виртуальных машин находят широкое применение на практике для решения задач, подготовки вычислительных ресурсов. В зависимости от специфики работы, а также доступных ресурсов, процесс настройки виртуальных машин может осуществляться различными способами. Если потребность в использовании ресурсов виртуальной машины возникает не часто, то настройка может быть выполнена вручную. В этом случае инженеры могут подключаться к виртуальной машине по протоколу SSH и осуществлять конфигурирование системы в соответствии с необходимыми требованиями. Такой подход, хотя и требует больше времени и усилий, может быть вполне приемлем для единичных случаев.

Однако для проектов, которые регулярно предоставляют ресурсы в виде виртуальных машин для нужд разработки и тестирования, существует необходимость в более эффективных решениях. Инструменты, такие как Ansible и Terraform, предоставляют возможность автоматизировать процесс настройки виртуальных машин, значительно упрощая управление и снижая вероятность ошибок. Данные инструменты позволяют задать необходимое состояние системы в виде кода на языке YAML, что позволяет легко и быстро вносить изменения и поддерживать консистентность конфигураций. Настройка таких хостов может быть выполнена на любом сервере, который имеет возможность подключения ко всем необходимым виртуальным машинам по протоколу SSH. Это обеспечивает гибкость и масштабируемость, позволяя командам сосредоточиться на более важных аспектах разработки и тестирования, а не на рутинных задачах по настройке инфраструктуры.

В случае, когда работа с виртуальными машинами становится неотъемлемой частью инфраструктуры IT-компании, а результаты взаимодействия с машинами напрямую влияют на прогресс разработки программных продуктов целых отделов, автоматизация настройки виртуальных машин приобретает особую значимость. В таких случаях использование полноценных инструментов непрерывной интеграции становится необходимым условием для обеспечения эффективного и бесперебойного рабочего процесса. Такие инструменты позволяют интегрировать множество сервисов инфраструктуры, создавая автоматизированные системы для выделения, настройки, вывода из работы виртуальных машин. Для разработчиков, которым требуются вычислительные ресурсы, но которые не имеют глубоких знаний в области автоматизации настройки виртуальных машин, инструменты непрерывной поставки предоставляют возможность гибкой и интуитивно понятной настройки процесса подготовки виртуальных машин через удобный графический интерфейс.

Список литературы:

1. Кондрашин Михаил Алексеевич, Арсенов Олег Юрьевич, Козлов Илья Викторович применение технологии виртуализации и облачных вычислений при построении сложных распределенных моделирующих систем // труды маи. – 2016. – №89. – С. 8 – 10

2. Федотов В. А. Работа с программой vm virtualbox от Oracle // Форум молодых ученых – 2021. – №2. – С. 3-4

3. Гурьев Дмитрий Евгеньевич опыт дистанционного проведения практических занятий по построению сегмента сети интернет // Современные информационные технологии и ИТ-образование. – 2020. – №4. – С. 3-5.

4. Костромин Роман Олегович сравнительный обзор средств управления конфигурациями ресурсов вычислительной среды функционирования цифровых двойников // Информационные и математические технологии в науке и управлении. – 2021. – №1. – С.133-134.

5. Красов Андрей Владимирович, Штеренберг Станислав Игоревич, Москальчук Андрей Игоревич Методология создания виртуальной лаборатории для тестирования безопасности распределенных информационных систем // Транспортное машиностроение. – 2020. – №3. – С. 39-41.

6. Дмитрий Сергеевич Фомин, Александр Витальевич Бальзамов, Анастасия Васильевна Савкина, Сергей Алексеевич Федосин Проблемы взаимодействия разработчиков с облачными инфраструктурами на базе Kubernetes в тестовых средах // Известия вузов. Поволжский регион. Технические науки. – 2023. – №3. – С. 33-35.

7. Прилепко Максим Анатольевич Модель создания лабораторной работы при использовании системы автоматизации проектирования виртуальных тренажеров Network Lab // ОНВ. 2014. – №2. – С. 225-226.

8. Лазарева Н. Б. Автоматизация развертывания kubernetes-кластеров на базе Ubuntu Oc B Rancher на инфраструктуре vmware vsphere // ивд. – 2023. – №4. – С. 4-6.

9. Ефимов В. Ю., Беззубиков А. А., Богомолов Д. А., Горемыкин О. В., Падарян В. А. Автоматизация разработки моделей устройств и вычислительных машин для qemu // Труды ИСП РАН. – 2017. – №6. – С. 9-11.

10. Тюменцев Д. В. Devops в эпоху облачных технологий: современные практики и перспективы развития // Вестник науки. – 2023. – №8. – С. 192.

11. Бушуев С. А. Сокращение операционных издержек бизнеса с помощью infrastructure as code при управлении высоконагруженными системами // Вестник науки. – 2024. – №8. – С. 6-9.

12. Aleksei Kabarukhin Methodology "infrastructure as code" in the operation of it infrastructure // EESJ. –2022. –№6. – С. 1-2.

13. Unleashing Full Potential of Ansible Framework: University Labs Administration / P. Masek, M. Stusek, Ja. Krejci [et al.] // Conference of Open Innovations Association, FRUCT. – 2018. – No. 22. – P. 144-150. – EDN XPBLNJ.

14. Документация Ansible Module format and documentation [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible/latest/dev_guide/developing_modules_documenting.html (дата обращения 26.11.2024).

15. Документация Ansible YAML Syntax [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html (дата обращения 26.11.2024).

16. Мулик, Д. И. Система управления конфигурацией Ansible / Д. И. Мулик, Д. А. Замотайлова // Цифровизация экономики: направления, методы, инструменты: Сборник материалов II всероссийской научно-практической конференции, Краснодар, 20–24 января 2020 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2020. – С. 250-252. – EDN PDIHMN.

17. Документация Ansible Connection methods and detail [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible/latest/inventory_guide/connection_details.html (дата обращения 26.11.2024).

18. Документация Ansible Roles [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html (дата обращения 26.11.2024).

19. Документация Ansible Developing modules [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible/latest/dev_guide/developing_modules_general.html (дата обращения 26.11.2024).

20. Гумеров Булат Зуфарович автоматизация развёртывания георасределенных кластеров splunk с помощью terraform и ansible // Universum: технические науки. – 2022. – №5-2. – С 1-5

21. Долматов, Р. А. Исследование возможностей примения и сравнительный анализ Terraform и Ansible при развертывании приложения в публичное облако / Р. А. Долматов, С. А. Молодяков // Современные технологии в теории и практике программирования: сборник материалов конференции, Санкт-Петербург, 23 апреля 2020 года / Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2020. – С. 150-152.

22. Bhat, K. Multicloud Orchestration using Terraform / K. Bhat // International Journal for Research in Applied Science and Engineering Technology. – 2022. – Vol. 10, No. 6. – P. 3749-3753. – DOI 10.22214/ijraset.2022.44760. – EDN BRZNEJ.

23. Ware, S. M. Garden of Future Blackness: A terraform artwork / S. M. Ware, A. Mitchell // Public. – 2024. – Vol. 35, No. 69. – P. 88-99. – DOI 10.1386/public_00192_1. – EDN LNFBTC.

24. Документация Hashicorp Terraform Writing Custom Terraform Providers [Электронный ресурс]. – Режим доступа: https://developer.hashicorp.com/terraform/tutorials/providers-plugin-framework (дата обращения 28.11.2024).

25. Кражевский, А. И. Исследование и разработка подходов для автоматизации процессов развертывания и мониторинга веб-приложения / А. И. Кражевский // Научные исследования XXI века. – 2024. – № 2 (28). – С. 25-29. – EDN RNJFYL.

26. Документация Hashicorp Terraform Command: plan [Электронный ресурс]. – Режим доступа: https://developer.hashicorp.com/terraform/cli/commands/plan (дата обращения 28.11.2024).

27. Документация Ansible Tower Inventories [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible-tower/latest/html/userguide/inventories.html (дата обращения 30.11.2024).

28. Thiyagarajan S. Automate Provisioning and Orchestration of Cloud Infrastructure using AWX: дис. – Dublin, National College of Ireland, 2022.

29. Bertrand B. et al. ICS Infrastructure Deployment Overview at ESS. – 2019.

30. Документация Ansible Tower Job Templates [Электронный ресурс]. – Режим доступа: https://docs.ansible.com/ansible-tower/latest/html/userguide/job_templates.html (дата обращения 30.11.2024).

31. Sesto V. Managing Large Server Environments //Practical Ansible: Configuration Management from Start to Finish. – Berkeley, CA: Apress, 2022. – С. 293-320.

32. Кадыров, К. А. Сравнение инструментов для CI/Cd Gitlab и Jenkins / К. А. Кадыров, А. М. Сафин // РАЗВИТИЕ СОВРЕМЕННОЙ науки и ОБРАЗОВАНИЯ: АКТУАЛЬНЫЕ ВОПРОСЫ, ДОСТИЖЕНИЯ и ИННОВАЦИИ: сборник статей Международной научно-практической конференции: в 2 ч., Пенза, 20 января 2022 года. Том Часть 1. – Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2022. – С. 123-125. – EDN NCIOIC.

33. Mysari S., Bejgam V. Continuous integration and continuous deployment pipeline automation using Jenkins Ansible //2020 International conference on emerging trends in information technology and engineering (IC-ETITE). – IEEE, 2020. – С. 1-4.

34. Документация Jenkins Pipeline [Электронный ресурс]. – Режим доступа: https://www.jenkins.io/doc/book/pipeline/ (дата обращения 30.11.2024).

35. Документация Jenkins Docker [Электронный ресурс]. – Режим доступа: https://plugins.jenkins.io/docker-plugin/ (дата обращения 30.11.2024).

36. Современные технологии на службе УИС: преимущества контейнерной виртуализации на примере Docker / А. А. Николаев, И. С. Горященко, А. А. Ивановский, И. А. Истомина // Информационные технологии в УИС. – 2021. – № 3. – С. 31-47. – EDN CWKLPS.

37. Документация Jenkins Git [Электронный ресурс]. – Режим доступа: https://plugins.jenkins.io/git/ (дата обращения 30.11.2024).

38. Берьянов, М. С. НАСТРОЙКА JENKINS И BITBUCKET ДЛЯ СБОРКИ ПРОЕКТА ПО PULL-REQUEST / М. С. Берьянов, Е. А. Бакулина, А. А. Ратковский // Вопросы науки. – 2023. – № 2. – С. 20-22. – EDN TMLLAI.