УДК 004.942

СРАВНИТЕЛЬНЫЙ ОБЗОР СРЕДСТВ УПРАВЛЕНИЯ КОНФИГУРАЦИЯМИ РЕСУРСОВ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ ФУНКЦИОНИРОВАНИЯ ЦИФРОВЫХ ДВОЙНИКОВ

Костромин Роман Олегович
м.н.с., e-mail: kostromin@icc.ru,
Институт динамики систем и теории управления СО РАН,
664033, г. Иркутск, ул. Лермонтова, 134.

Аннотация. Статья посвящена проблемам автоматизации конфигурирования ресурсов распределенной вычислительной среды. Среда может использоваться как для запуска научных приложений, так и для задач исследования функционирования цифровых двойников инфраструктурных объектов. Для такой среды характерно динамическое изменение состава ее ресурсов, их характеристик и требований к ним. Ресурсы среды могут включать классические вычислительные машины, серверы, виртуальные машины и микрокомпьютеры. Необходимы универсальные средства автоматизации настройки таких ресурсов. В статье выполнен сравнительный анализ средств управления конфигурациями ресурсов среды (Chef, Ansible, Puppet, SaltStack), которые позволяют автоматизировать процесс настройки узлов. Благодаря такой автоматизация сокращается время их подготовки и повышается надежность вычислений за счет уменьшения отказов программного и аппаратного обеспечения, связанных с человеческим фактором в процессе настройки в ручном режиме. Рассмотренные средства являются основой для дальнейшего развития инструментального комплекса для построения среды, обеспечивающей условия функционирования цифровых двойников природосберегающего оборудования. Исходя из результатов сравнительного анализа и требований, предъявляемых к инструментальным комплексам, выбрана система Ansible для внедрения в цепочку автоматизации процессов непрерывной интеграции прикладного и системного программного обеспечения приложений. Практические эксперименты показали преимущества использования Ansible в сравнении с другими системами аналогичного назначения.

Ключевые слова: инструментальные средства, управление ресурсами, распределенная вычислительная среда, виртуализированные ресурсы, автоматизация управления конфигурациями, цифровой двойник

Цитирование: Костромин Р. О. Сравнительный обзор средств управления конфигурациями ресурсов вычислительной среды функционирования цифровых двойников //Информационные и математические технологии в науке и управлении. 2021. № 1 (21). С. 131-145. DOI:10.38028/ESI.2021.21.1.011

Введение. В настоящее время разработка программного обеспечения (ПО) является сложным и многоэтапным процессом [1]. Как правило, эти процессы включают этапы тестирования, отладки и доставки программ конечным пользователям. Разработка автоматизированной цепочки эффективной доставки программного обеспечения в целом или его отдельных компонентов (модулей) по-прежнему является актуальной проблемой. На практике, значимым подходом для решения данной проблемы является непрерывная интеграция (Continuous Integration, CI) программного обеспечения [2].

Средства CI позволяют организовать автоматизированную цепочку от получения обновленного программного кода из репозитория до развертывания готового научного приложения на стороне пользователя. В рамках CI пользователи получают программный продукт как сервис. Они регулярно разрабатывают новые или модифицируют существующие версии приложений, отлаживают их и тестируют.

Отметим, что средства CI при доставке и тестировании могут использовать только уже существующую вычислительную инфраструктуру. Это справедливо как для физических, так и виртуализированных ресурсов. Таким образом, необходимы дополнительные инструментальные средства для автоматизации создания среды и управления конфигурациями ее узлов.

Разработчики программного обеспечения вынуждены сопровождать как разрабатываемое приложение, так и вычислительную инфраструктуру, необходимую для его отладки и тестирования в процессе CI [3]. Как результат, значительное время при подготовке приложения к выпуску занимает настройка такой инфраструктуры.

Ручное управление инфраструктурой замедляет процессы разработки и сопровождения программного обеспечения. Использование собственных скриптов автоматизации усложняет ее развитие и обслуживание. В частности, это относится к средствам Parallel SSH для параллельного выполнения скриптов [4]. Вышеперечисленные проблемы возникают как в коммерческой сфере разработки ПО, так и в научной сфере. При этом автоматизация конфигурирования инфраструктуры для отладки и тестирования ПО по-прежнему является вызовом [5]. В частности, остается необходимость подготовки и хранения разных версий образов виртуальных машин (ВМ) для разных операционных систем (ОС), версий этих ОС, и версий ПО. В этом случае, дисковое пространство быстро исчерпывается. Кроме того, это усложняет сопровождение инфраструктуры.

Автоматизация процессов создания ресурсов, их автоматизированная настройка, поставка на них модулей ПО может быть применена практически во всех сферах разработки ПО [6]. Однако особое значение это имеет при организации среды функционирования цифровых двойников (ЦД) [7].

Под ЦД понимается виртуальная программная сущность, которая отражает наиболее важные компоненты жизненного цикла исследуемого объекта или системы. В качестве исходных данных для функционирования ЦД используются как реальные показатели работы объекта, таки ретроспективные. ЦД может быть представлен в виде взаимосвязанных имитационных моделей, приложений и сервисов [8].

На практике ЦД все шире внедряются в различных сферах человеческой деятельности на различных этапах создания и использования сложных объектов. Например, такой сферой является поддержка принятия решений при управлении социально-экономическими территориями и расположенными на них инфраструктурными объектами. При этом особое внимание уделяется использованию природосберегающего оборудования [7-8], ему соответствует оборудование, которое в процессе своей работы использует экологически чистые технологии и возобновляемые источники энергии. Например тепловые насосы, ветряные электростанции, солнечные панели и т.д.

Программно-аппаратный комплекс функционирования ЦД как правило состоит из оборудования, которое осуществляет мониторинг текущих показателей работы ЦД и ПО, обеспечивающего агрегацию получаемых данных, моделирование работы исследуемого объекта и визуализацию. Моделирование связано с многократными запусками многовариантных расчетов, поэтому невозможно без применения высокопроизводительных вычислительных ресурсов.

С одной стороны, разработчикам ЦД необходимо управлять конфигурациями оснащенных датчиками микрокомпьютеров, которые осуществляют сбор и предварительную обработку показателей работы оборудования инфраструктурных объектов [9]. С другой стороны – управлять узлами в распределенной вычислительной среде, на которых осуществляется многократный запуск многовариантных расчетов в рамках имитационного моделирования. Поэтому характеристики ПО для обеспечения функционирования ЦД природосберегающего оборудования инфраструктурных объектов требуют применения специализированных инструментальных средств.

Таким образом, можно сформулировать следующие технические и функциональные требования к средствам управления конфигурациями:

- поддержка управления широким спектром популярных ОС,

- возможность полного контроля настройки узлов вычислительной инфраструктуры на каждом шаге,
- простота установки с минимальной подготовкой настраиваемых узлов,
- свободное распространение,
- инициализация изменений в управляемых узлах сервером (в рамках цепочки CI),
- управление узлами посредством протокола SSH для управления маломощными периферийными устройствами, используемыми в рамках туманных вычислений.

В статье рассматриваются распространенные средства управлениям конфигурациями узлов вычислительной среды. Обсуждаются их важные свойства и общая схема управления конфигурациями.

Статья структурирована следующим образом. В разделе 1 выполнен обзор свойств систем управления конфигурациями, выделены две признанные перспективными системы, их характеристики сравниваются в разделе 2. Сравнительный анализ различных способов управлениями конфигурациями узлов в экспериментальной среде представлен в разделе 3. В заключении представлены выводы по материалам статьи.

1. Обзор средств управления конфигурациями. В настоящее время известен широкий спектр программных средств для управления конфигурациями (System Configuration Management, SCM) [10]. Они предназначены для автоматизации процессов создания и развертывания вычислительной инфраструктуры с последующей доставкой приложений в ее узлы.

Кроме того, такие системы позволяют улучшить как управляемость процессов настройки и сопровождения узлов инфраструктуры, так и их надежность. Однако, подготовка и запуск ВМ не поддерживаются в рамках SCM. Данный процесс реализуется средствами гипервизоров или специальными надстройками [11].

Предполагается, что средства SCM запускаются на «чистых» ОС, хотя и допускается возможность изменения в работающих системах. Они осуществляют настройку и подготовку узлов к работе. Таким образом, разработчик программного обеспечения должен сформировать требования к конфигурации узла. В процессе работы узлов, доступно отслеживание и корректировка их конфигураций в соответствии с требованиями. Общая схема управления конфигурациями представлена на рис. 1.

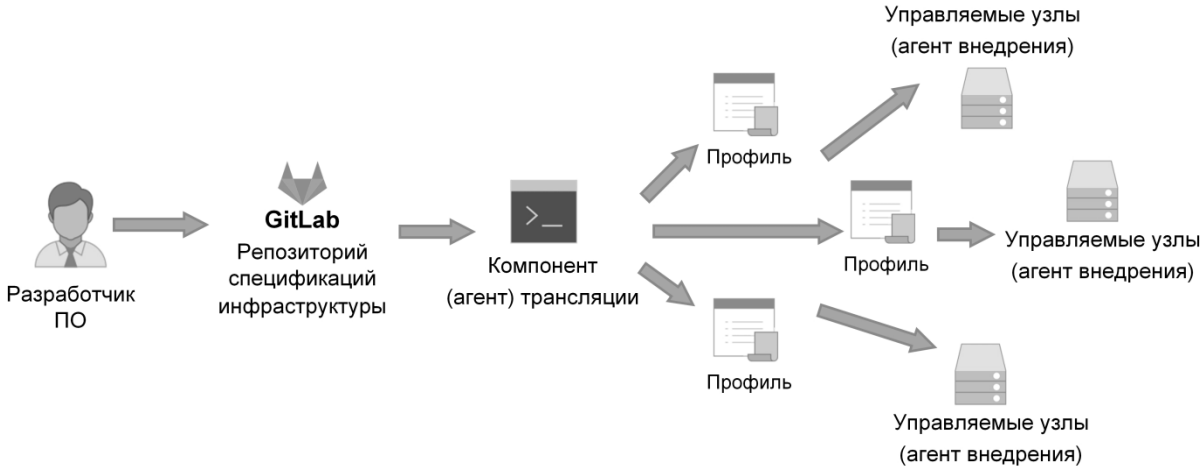

Рис. 1. Схема управления конфигурациями

При помощи выполнения специального программного кода (скриптов), средства SCM позволяют привести вычислительную инфраструктуру в целевое состояние автоматически [12]. Такой код удобно разрабатывать, модифицировать и поддерживать с помощью систем контроля версий, таких как Git [13] и Mercurial [14]. Разработчик программного обеспечения

создает и сохраняет в репозитории спецификацию инфраструктуры. Затем компонент (агент) трансляции самостоятельно или по сигналу администратора передает спецификацию (профиль) на управляемые узлы. В узлах данные профили представляются в виде набора исполняемых команд с помощью компонентов (агентов) внедрения.

Наиболее популярными системами управления конфигурациями являются Puppet [15], Chef [16], Ansible [17], и SaltStack [18]. Их ключевые характеристики представлены в табл. 1.

Таблица 1. Характеристики систем управления конфигурациями.

Система	Ansible	Chef	Puppet	SaltStack	MAS-SCM
Архитектура	Без-агентная	Клиент-серверная	Клиент-серверная	Без-агентная	Мультиагентная
Мультиплатформенность	+	+	+	+	+
Протокол доставки конфигураций	SSH	RabbitMQ	Mcollective	ZeroMQ	SSH / Jade ACL
Способ доставки конфигураций	Push	Pull (Push only in the corporate version)	Push, Pull	Pull	Push, Pull (Agent request)
Пошаговая установка	+	+	-	-	+
База данных	-	PostgreSQL	PuppetDB	-	MariaDB
Язык разработки	Python	Ruby	C++ & Clojure	Python	Java

Среди них язык разработки системы, архитектура системы и поддерживаемая ОС. В рамках выполнения распределенных пакетов прикладных программ важным требованием является поддержка управления ОС семейств Windows и Linux. В статье проприетарные системы и те, разработка которых приостановлена или прекращена, не рассматриваются.

В клиент - серверной архитектуре, агент - это программное обеспечение на стороне узла, которое управляет его конфигурацией в рамках SCM. Без-агентная архитектура позволяет управлять узлом без установки дополнительного программного обеспечения через протокол SSH. Наличие специального (проприетарного) агента в средствах SCM ограничивает типы ОС, конфигурациями которых можно управлять. Среди рассматриваемых систем только Ansible является системой с полностью без-агентной архитектурой. При этом, SaltStack тоже позволяет использовать без-агентный режим. Однако, он работает значительно медленнее в таком режиме в сравнении с использованием проприетарного протокола. В целом, каждая система имеет агентов для поддержки большинства популярных ОС.

Каждая из рассматриваемых систем активно развивается. На их официальных сайтах, размещена полная документация, описывающая все процессы, начиная от установки систем и до их применения при разработке программного обеспечения. Вокруг этих систем сформировано международное сообщество пользователей и энтузиастов, что позитивно сказывается на их развитии.

Все четыре системы предоставляют веб-интерфейс для управления конфигурациями. Его наличие в данных системах не является обязательным требованием. Однако он может использоваться для построения отчетов и наглядного отображения конфигурации инфраструктуры при ее мониторинге. Все системы имеют средства подключения к внешним средствам мониторинга. SaltStack использует собственные средства подобного назначения.

Рассмотрим подробнее особенности работы рассматриваемых систем.

Базы данных. При развертывании систем Chef и Puppet используются системы управления базами данных (СУБД) PostgreSQL и PuppetDB соответственно. Эти СУБД обеспечивают централизованное хранение конфигураций. Производительность и масштабируемость каждой из систем напрямую зависит от используемой СУБД. Кроме того, сервер базы данных требует дополнительного сопровождения.

Корпоративная версия Ansible Tower использует PostgreSQL. Дополнительно она допускает установку MongoDB для построения высоконадежных архитектур.

Доставка конфигураций. Каждая из рассматриваемых систем использует свой собственный способ доставки (транспортировки) конфигураций из репозитория в узел. В то же время, только Ansible использует SSH (Powershell в Windows) для доставки конфигураций. Теоретически это позволяет управлять любым устройством, поддерживающим SSH, с помощью Ansible.

Управляющий узел. Ansible обеспечивает поддержку широкого диапазона ОС на управляющем узле. Теоретически, поддерживаются также любые ОС, имеющие поддержку Python 2.6. SaltStack работает как на Linux, так и на BSD. Управляющие узлы Puppet и Chef работают только на ОС с ядром Linux. Среди них RHEL, SLES, и последние версии Ubuntu.

Пошаговая установка. Только в Ansible и Chef используется пошаговый подход (step-by-step) к настройке узлов. В рамках SCM под пошаговым выполнением понимается последовательное выполнение всех действий, описанных в конфигурационном файле. Данный подход называется императивной конфигурацией. В Puppet и SaltStack используется декларативная конфигурация. Это значит, что конечное состояние узла описывается в конфигурационном файле. При этом система сама выбирает наилучший путь для достижения этого состояния. Декларативная конфигурация является предпочтительной для однотипных серверов. В то же время, императивная конфигурация обеспечивает полный контроль за процессом изменения состояния узла.

Способ доставки изменений в конфигурации от управляющего узла на управляемые узлы. Рассматриваются следующие два способа доставки: Push и Pull. В случае Push, сервер сам отслеживает изменения и при необходимости инициализирует обновление у клиентов. В случае Pull, клиентские агенты периодически опрашивают сервер об обновлениях.

Оперативный учет изменений в конфигурациях является важной характеристикой при построении экспериментальной и рабочей вычислительных инфраструктур. В рамках Push, оперативность достигается в доставке конфигураций на узлы. Только Puppet и Ansible поддерживают Push. Chef в данном случае не рассматривается, так как использование Push доступно только в платной версии. В связи с этим, только системы Puppet и Ansible рассматриваются далее в деталях. Обе эти системы имеют средства интеграции с Docker, Kubernetes, и Jenkins, которые активно применяются в рамках CI.

2. Сравнение Puppet и Ansible. *Особенности управления узлами в Puppet и Ansible.* Первым этапом при настройке Ansible является выбор управляющего узла. Таким узлом может быть любой узел, доступный в репозитории пакетов дистрибутива. В рамках этого этапа необходимо установить Ansible и указать перечень IP-адресов управляемых узлов. При этом нет необходимости в настройке клиентского программного обеспечения. Единственным необходимым условием является поддержка доступа по SSH с управляющего узла на управляемые узлы. Схема управления узлами в Ansible представлена на рис. 2. Описание конфигурации узлов в Ansible представлено в виде PlayBook.

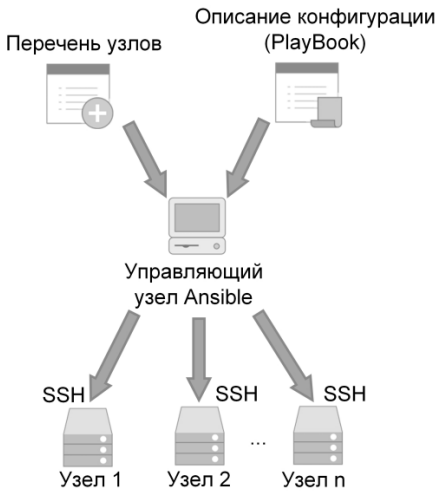

Рис. 2. Схема управления узлами в Ansible

Схема управления узлами в Puppet представлена на рис. 3. Подготовка Puppet для работы происходит сложнее. Во-первых, необходимо синхронизировать время и часовой пояс на всех узлах. Далее, необходимо установить серверное и клиентское ПО. Кроме того, необходимо выполнить следующие операции:

- настроить конфигурационный файл для каждого управляемого узла на сервере Puppet,
- открыть порт 8140,
- перевести службу PuppetServer в рабочий режим,
- указать IP-адрес сервера и сгенерировать SSL-сертификат на управляемых узлах.

Необходимость генерации SSL-сертификата обуславливается тем, что для связи используется протокол HTTPS.

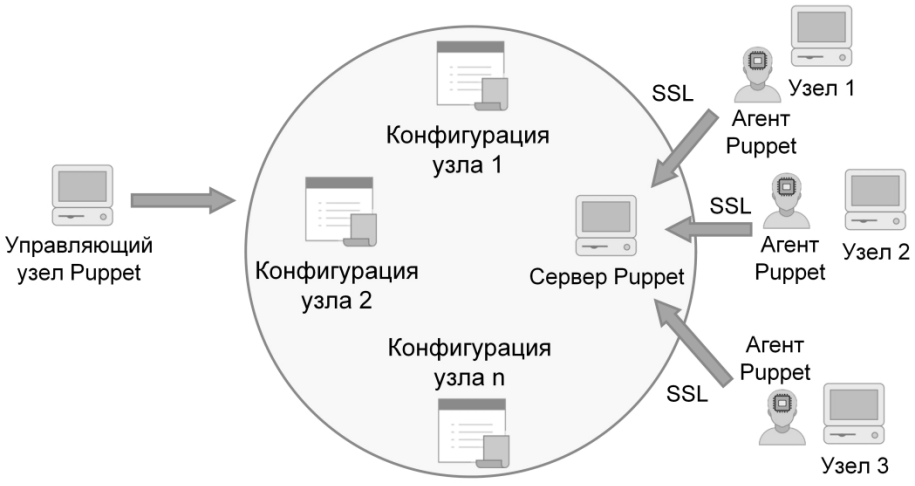

Рис. 3. Схема управления узлами в Puppet

Описание конфигураций в Ansible. Рис. 4 демонстрирует пример фрагмента кода Ansible PlayBook на языке YAML из официального репозитория Ansible examples [19]. В данном примере, установка веб-сервера Tomcat в Centos производится на управляемые узлы. На первом этапе, осуществляется установка Java JDK 1.7. После этого выполняются создание и настройка пользователя tomcat. В частности, ему добавляются права супер-пользователя. Далее, при загрузке исходного кода Tomcat из репозитория выполняются его распаковка, настройка и установка. Код на языке YAML хорошо читаем, что упрощает процесс освоения Ansible.

```
1.  ---
2.  - name: Install Java 1.7
3.  yum: name=java-1.7.0-openjdk state=present
4.  - name: add group "tomcat"
5.  group: name=tomcat
6.  - name: add user "tomcat"
7.  user: name=tomcat group=tomcat home=/usr/share/tomcat createhome=no
8.  become: True
9.  become_method: sudo
10.  - name: Download Tomcat
11.  get_url: url=http://archive.apache.org/dist/tomcat/tomcat-
7/v7.0.61/bin/apache-tomcat-7.0.61.tar.gz dest=/opt/apache-tomcat-
7.0.61.tar.gz
12.  - name: Extract archive
13.  command: chdir=/usr/share /bin/tar xvf /opt/apache-tomcat-
7.0.61.tar.gz -C /opt/ creates=/opt/apache-tomcat-7.0.61
14.  - name: Symlink install directory
15.  file: src=/opt/apache-tomcat-7.0.61 path=/usr/share/tomcat
state=link
16.  - name: Change ownership of Tomcat installation
17.  file: path=/usr/share/tomcat/ owner=tomcat group=tomcat
state=directory recurse=yes
18.  …
19.  - name: Install Tomcat init script
20.  copy: src=tomcat-initscript.sh dest=/etc/init.d/tomcat mode=0755
21.  - name: Start Tomcat
22.  service: name=tomcat state=started enabled=yes
23.  - name: deploy iptables rules
24.  template: src=iptables-save dest=/etc/sysconfig/iptables
```

Рис. 4. Фрагмент кода Ansible PlayBook на языке YAML
из официального репозитория «Ansible examples».

Описание конфигураций в Puppet. Описание конфигураций (манифест Puppet) создается на предметно-ориентированном Ruby-подобном языке в Puppet. Рис. 5 демонстрирует пример из репозитория PuppetLabs MySQL Puppet Module, в котором рассматриваются установка и настройка MySQL [20].

В табл. 2 представлены ключевые преимущества и недостатки систем Puppet и Ansible.

Таблица 2. Преимущества и недостатки систем Puppet и Ansible.

	Puppet	Ansible
Преимущества	Хорошо известный, стабильно развивающийся продукт. Имеет удобный графический интерфейс пользователя. Все основные ОС поддерживаются в Puppet.	Высокая производительность. Без-агентная установка и развертывание. Низкие накладные расходы. Использование Python для разработки этой системы. Простота в освоении.
Недостатки	Низкая производительность работы системы, написанной на Ruby (по сравнению с системами, написанными на Python). Необходимость в изучении языка Puppet для описания конфигураций.	Это относительно новая система. Поддержка Windows осуществляется только через PowerShell.

```
1. # @summary
2. #     Installs and configures the MySQL client.
3. #
4. # @example Install the MySQL client
5. #     class {'::mysql::client':
6. #         package_name => 'mysql-client',
7. #         package_ensure => 'present',
8. #         bindings_enable => true,
9. #     }
10. #
11. # @param bindings_enable
12. #     Whether to automatically install all bindings. Valid values are
    `true`, `false`. Default to `false`.
13. # @param install_options
14. #     Array of install options for managed package resources. You must
    pass the appropriate options for the package manager.
15. # @param package_ensure
16. #     Whether the MySQL package should be present, absent, or a spe-
    cific version. Valid values are 'present', 'absent', or 'x.y.z'.
17. # @param package_manage
18. #     Whether to manage the MySQL client package. Defaults to `true`.
19. # @param package_name
20. #     The name of the MySQL client package to install.
21. #
22. class mysql::client (
23. $bindings_enable = $mysql::params::bindings_enable,
24. $install_options = undef,
25. $package_ensure = $mysql::params::client_package_ensure,
26. $package_manage = $mysql::params::client_package_manage,
27. $package_name = $mysql::params::client_package_name,
28. )  inherits mysql::params {
29. include '::mysql::client::install'
30. if $bindings_enable {
      a. class { 'mysql::bindings':
           i.  java_enable => true,
          ii.  perl_enable => true,
         iii.  php_enable => true,
          iv.  python_enable => true,
           v.  ruby_enable => true,
      b. }
31. }
32. # Anchor pattern workaround to avoid resources of
    mysql::client::install to
33. # "float off" outside mysql::client
34. anchor { 'mysql::client::start': }
                  => Class['mysql::client::install']
                  => anchor { 'mysql::client::end': }
35. }
```

Рис. 5. Описание конфигураций на предметно-ориентированном Ruby-подобном языке в Puppet.

В целом, каждая из систем обладает своими преимуществами и недостатками. Исходя из требований, сформулированных во введении, принято решение внедрить в цепочку CI систему Absible. Это обусловлено следующими ключевыми факторами:

- Поддержка управления большинством популярных ОС.
- Использование без-агентного подхода, что упрощает подготовку системы к работе.
- Использование императивной конфигурации узлов, обеспечивающей полный контроль за установкой программного обеспечения на каждом этапе.
- Бесплатное распространение по лицензии GPLv3+.

- Использование Push для доставки конфигураций.
- Управление узлами посредством протокола SSH, что позволяет управлять практически любым устройством, поддерживающим данный протокол.

3. Сравнительный анализ. В рамках сравнительного анализа оценивается время, затрачиваемое на подготовку 10 узлов экспериментальной вычислительной среды, обеспечивающей проведение имитационного моделирования для ЦД. 10 ВМ запущены на 10 узлах среды с помощью специальной гипервизорной надстройки, разработанной в дополнение к OpenStack [11]. ВМ запущены с использованием их образов, подготовленных заранее.

Экспериментальная среда создана на базе ресурсов Иркутского суперкомпьютерного центра. На пяти узлах установлена Centos 7, на остальных узлах используется Ubuntu 18.04. В каждой ОС обеспечен беспарольный ключевой доступ по протоколу SSH.

Все ВМ должны быть готовы для разворачивания экземпляров исполнительной системы. При этом, Web-сервер Apache, PHP версии 7.2, СУБД Maria DB версии 10, и прочее системное программное обеспечение необходимы для работы исполнительной системы.

Для конфигурирования узлов необходимо выполнить следующие действия:

- Установка необходимого системного программного обеспечения на ресурсы и конфигурирование среды.
- Запуск мультиагентной платформы.
- Запуск агентов на ресурсах и их регистрация.
- Распределение вычислительных работ.
- Развертывание исполняемых файлов на ресурсах агентов.
- Запуск системной части исполнительной системы.
- Запуск модулей.
- Сбор и анализ результатов выполнения модулей.

Есть несколько путей подготовки экспериментальной вычислительной среды без использования Ansible:

- Подготовка полностью настроенных образов ВМ для всех возможных случаев.
- Установка и настройка каждой запущенной ВМ вручную.
- Автоматизация установки с помощью параллельного выполнения команд (скриптов) с применением Parallel SSH.

Хранение настроенных образов ВМ для всех возможных случаев является нецелесообразным, так как требования к версиям системного и прикладного программного обеспечения могут часто изменяться. Кроме того, свободное дисковое пространство очень быстро заканчивается при таком хранении.

На рис. 6. представлена гистограмма, отражающая время конфигурирования одного узла в ручном режиме и с помощью Ansible. Из рисунка видно, что настройка одного узла вручную занимает больше времени на каждом из этапов, чем с использованием средств, базирующихся на применении Ansible.

В целом, автоматизация ручной установки с помощью Parallel SSH позволяет сократить время настройки узлов. При этом необходимо разрабатывать несколько версий таких скриптов для каждой ОС и ее версий. Число таких скриптов растет очень быстро, т.к. для каждого отдельного случая необходимо подготовить соответствующий скрипт, который приведет узел в рабочее состояние. Очевидно, что использование Parallel SSH является неприемлемым при большом числе разнородных конфигураций узлов в экспериментальной среде. Кроме того, использование Parallel SSH может привести к дополнительным ошибкам в разработке, сопровождении и отладке большого числа скриптов.

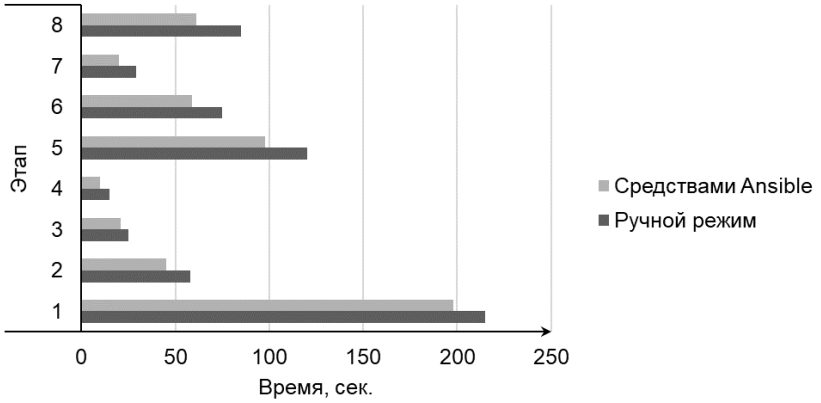

Рис. 6. Время конфигурирования одного узла.

Гистограмма на рис. 7 демонстрирует значительное уменьшение времени автоматической настройки узлов по сравнению с ручной настройкой, даже с применением Parallel SSH. При этом, небольшое время настройки узлов с помощью скриптов нивелируется сложностью сопровождения и разворачивания собственных скриптов. В Ansible такие проблемы отсутствуют.

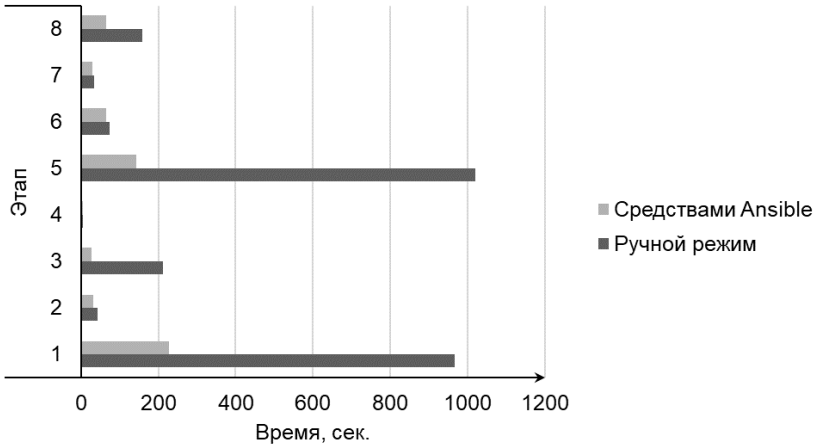

Рис. 7. Время конфигурирования десяти узлов.

Использование средств Ansible позволило существенно сократить время конфигурирования узлов для обеспечения работы ЦД. В случае Ansible мы затрачиваем приблизительно 582 секунды против 2624 секунд в ручном режиме. Таким образом, используя Ansible, мы сокращаем накладные расходы в 4 раза по сравнению с ручным режимом настройки узлов.

Заключение. В статье выполнен сравнительный анализ средств управления конфигурациями вычислительных сред. Такие средства предназначены для автоматизации настройки узлов среды. В результате автоматизации время, затрачиваемое на настройку, существенно сокращается. Дополнительно надежность узлов повышается за счет уменьшения ошибок в настройке, связанных с человеческим фактором.

Достоинства SCM особенно очевидны при их внедрении в рамках CI. При этом требуемое число ВМ запускается с помощью средств виртуализации. Дальнейшая настройка ВМ выполняется автоматически средствами SCM.

Рассмотренные средства являются основой для дальнейшего создания и развития инструментального комплекса для построения среды, обеспечивающей условия функционирования ЦД природосберегающего оборудования. В целом, подход к автоматизации организа-

ции среды функционирования ЦД на базе средств SCM является новым. Дополнительная интеграция поддержки мультиагентных технологий позволит повысить научную и практическую значимость применения данных средств [21]. В частности, это касается улучшения показателей качества конфигурирования узлов среды на примере задач эффективного распределения исполняемых модулей ЦД между разнородными вычислительными ресурсами [22]. Встраивание мультиагентной системы в такие системы, как Ansbile, позволяет реализовать их недостающие функциональные возможности.

Результаты экспериментального анализа подготовки экспериментальной вычислительной среды продемонстрировали преимущества системы Ansible и позволяют обоснованно выбрать эту систему в качестве базового средства для автоматизации управления конфигурациями ресурсов. Автоматизация обеспечивает улучшение качества разработки, тестирования и распространения исполняемых модулей цифровых двойников и других научных приложений.

Благодарности. Работа выполнена при финансовой поддержке РФФИ (проект № 19-07-00097), а также Российского фонда фундаментальных исследований и Правительства Иркутской области (проект № 20-47-380002-р_а). Исследования, связанные с моделированием вычислительной среды, выполнены при поддержке Министерства науки и высшего образования Российской Федерации, проект № 121032400051-9.

СПИСОК ЛИТЕРАТУРЫ

1. KimG., Debois P., Willis J., Humble J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press. 2016. 480 с.
2. Shahin M., Babar M.A., Zhu L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices // IEEE Access. 2017. Т. 5. С. 3909–3943. DOI: 10.1109/ACCESS.2017.2685629.
3. Morris K. Infrastructure as Code. O'Reilly Media. 2016. 362 с.
4. Parallel-SSH. Режим доступа: https://parallel-ssh.org (дата обращения: 09.04.2021).
5. Feoktistov A., Gorsky S., Sidorov I., Tchernykh A. Continuous Integration in Distributed Applied Software Packages // Proceedings of the 42th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2019). IEEE. 2019. С. 1775–1780. DOI: 10.23919/MIPRO.2019.8757002.
6. Феоктистов А.Г., Сидоров И.А., Горский С.А. Автоматизация разработки и применения распределенных пакетов прикладных программ // Проблемы информатики. 2017. № 4. С. 61–78.
7. Медведев А.В. Цифровые двойники территорий для поддержки принятия решений в сфере регионального социально-экономического развития // Современные наукоемкие технологии. 2020. № 6-1. С. 61–66. DOI: 10.17513/snt.38072.
8. Krawczyk J.B., Lifran R., Tidball M. Use of coupled incentives to improve adoption of environmentally friendly technologies // Journal of Environmental Economics and Management. 2005. Т. 49. № 2. С. 311–329. DOI: 10.1016/j.jeem.2004.04.007.
9. Sidorov I., Kostromin R., Feoktistov A. System for monitoring parameters of functioning infrastructure objects and their external environment // Proceedings of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments. CEUR-WS Proceedings. 2020. Т. 2638. С. 252–264. DOI: 10.47350/ICCS-DE.2020.23.
10. Quigley J.M., Robertson K.L. Configuration Management: Theory, Practice, and Application. Auerbach Publications. 2015. 438 с.

11. Feoktistov A., Sidorov I., Sergeev V., Kostromin R., Bogdanova V. Virtualization of Heterogeneous HPC-clusters Based on OpenStack Platform // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2017. Т. 6. № 2. С. 37–48.
12. Delaet T., Joosen W., Vanbrabant B. A survey of system configuration tools // Proceedings of the 24th International Conference on Large installation system administration. USENIX Association. USA. 2010. С 1–8.
13. GitHub. Режим доступа: https://github.com (дата обращения: 09.04.2021).
14. Mercurial. Режим доступа: https://www.mercurial-scm.org (дата обращения: 09.04.2021).
15. Puppet. Режим доступа: https://puppet.com/ (дата обращения: 09.04.2021).
16. Chef. Режим доступа: https://www.chef.io/products/chef-infra (дата обращения: 09.04.2021).
17. Ansible. Режим доступа: https://www.ansible.com (дата обращения: 09.04.2021).
18. SaltStack. 2021. Режим доступа: https://www.saltstack.com (дата обращения: 09.04.2021).
19. Ansible PlayBook examples repository. Режим доступа: https://github.com/ansible/ansible-examples/blob/master/tomcat-standalone/roles/tomcat/tasks/main.yml (дата обращения 09.04.2021).
20. Puppetlabs manifest examples repository. Режим доступа: https://github.com/puppetlabs/puppetlabs-mysql (дата обращения 09.04.2021).
21. Ильясов Б.Г., Макарова Е.А., Закиева Е.Ш., Габдуллина Э.Р. Методологические основы моделирования и интеллектуального управления промышленным комплексом как сложным динамическим многоагентным объектом // Современные наукоемкие технологии. 2020. № 11-2. С. 288–293. DOI: 10.17513/snt.38376.
22. Черных А.Н., Бычков И.В., Феоктистов А.Г., Горский С.А., Сидоров И.А., Костромин Р.О., Еделев А.В., Зоркальцев В.И., Аветисян А.И. Смягчение неопределенности при разработке и применении научных приложений в интегрированной вычислительной среде // Труды ИСП РАН. 2021. Т. 33. № 1. С. 151–172. DOI: 10.15514/ISPRAS-2021-33(1)-11.

UDK 004.942

COMPARATIVE ANALYSIS OF THE RESOURCE CONFIGURATION MANAGEMENT SYSTEMS FOR THE COMPUTING ENVIRONMENT OF DIGITAL TWINS FUNCTIONING

Roman O. Kostromin
junior researcher, e-mail:kostromin@icc.ru,
Matrosov Institute for System Dynamics and Control Theory
of the Siberian Branch of the Russian Academy of Sciences,
134, Lermontov St., 664033, Irkutsk, Russia.

Annotation. The paper discusses the problem of preparing a computing environment for large-scale scientific experiments in the process of continuous integration of applied and system software. The environment is used both for launching scientific applications and for studying the functioning of digital twins of infrastructure objects. Such an environment is characterized by dynamic changes in the composition of resources, their characteristics, and requirements for them. Environment resources include computers,

servers, virtual machines, and microcomputers. We need universal tools to automate the configuration of such resources. A comparative analysis of software configuration management tools (such as Chef, Ansible, Puppet, and SaltStack) of computational nodes in a heterogeneous environment is being performed. These tools are intended for automating the configuration of different nodes. Such automation reduces the setup time of nodes and increases the reliability of computations by minimizing the number of software and hardware failures, associated with the human factor in the manual configuration process. The considered tools are the basis for the further development of the instrumental complex for building an environment that provides the conditions for the functioning of digital twins of nature protection equipment. Based on the results of the comparative analysis and requirements of this framework, the Ansible framework was selected for further integration into the chain of continuous integration of applied and system software. Practical experiments have shown the advantages of using Ansible in comparison with other systems of a similar purpose.

Keywords. software tools, resource management, distributed computing environment, virtualized resources, configuration management automation, digital twin

Acknowledgements. The study was supported by the Russian Foundation of Basic Research (project no. 19-07-00097) and by the Russian Foundation of Basic Research and Government of Irkutsk Region (project no. 20-47-380002). The study related to modeling the computing environment was supported by the Ministry of Science and Higher Education of the Russian Federation, project no. 121032400051-9.

REFERENCES

1. KimG., Debois P., Willis J., Humble J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press. 2016. 480 c.
2. Shahin M., Babar M.A., Zhu L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices // IEEE Access. 2017. T. 5. C. 3909–3943. DOI: 10.1109/ACCESS.2017.2685629.
3. Morris K. Infrastructure as Code. O'Reilly Media. 2016. 362 c.
4. Parallel-SSH. Режим доступа: https://parallel-ssh.org (дата обращения 09.04.2021).
5. Feoktistov A., Gorsky S., Sidorov I., Tchernykh A. Continuous Integration in Distributed Applied Software Packages // Proceedings of the 42th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2019). IEEE. 2019. C. 1775–1780. DOI: 10.23919/MIPRO.2019.8757002.
6. Feoktistov A.G., Sidorov I.A, Gorsky S.A. Avtomatizatsiya razrabotki i primene-niya raspredelennykh paketov prikladnykh programm [Automation of development and application of distributed applied software packages] // Problemy informatiki = Problems of Informatics. 2017. № 4. P. 61–78.
7. Medvedev A.V. Tsifrovyye dvoyniki territoriy dlya podderzhki prinyatiya resheniy v sfere regional'nogo sotsial'no-ekonomicheskogo razvitiya [Digital twins of territories to support decision-making in the field of regional socio-economic development] // Sovremennyye naukoyemkiye tekhnologii = Modern high technologies. 2020. No. 6-1. P. 61–66. DOI: 10.17513/snt.38072.
8. Krawczyk J.B., Lifran R., Tidball M. Use of coupled incentives to improve adoption of environmentally friendly technologies // Journal of Environmental Economics and Management. 2005. Vol. 49, No. 2. P. 311–329. DOI: 10.1016/j.jeem.2004.04.007.
9. Sidorov I., Kostromin R., Feoktistov A. System for monitoring parameters of functioning infrastructure objects and their external environment // Proceedings of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environ-

ments. CEUR-WS Proceedings. 2020. Vol. 2638. P. 252–264. DOI: 10.47350/ICCS-DE.2020.23.

10. Quigley J.M., Robertson K.L. Configuration Management: Theory, Practice, and Application. Auerbach Publications, 2015. 438 p.
11. Feoktistov A., Sidorov I., Sergeev V., Kostromin R., Bogdanova V. Virtualization of Heterogeneous HPC-clusters Based on OpenStack Platform // South Ural State University Bulletin: Computational Mathematics and Software Engineering series. 2017. Vol. 6, No. 2. P. 37–48.
12. Delaet T., Joosen W., Vanbrabant B. A survey of system configuration tools // Proceedings of the 24th International Conference on Large installation system administration. USENIX Association, USA. 2010. P. 1–8.
13. GitHub. Режим доступа: https://github.com (дата обращения 09.04.2021).
14. Mercurial. Режим доступа: https://www.mercurial-scm.org (дата обращения 09.04.2021).
15. Puppet. Режим доступа: https://puppet.com (дата обращения 09.04.2021).
16. Chef. Режим доступа: https://www.chef.io/products/chef-infra (дата обращения 09.04.2021).
17. Ansible. Режим доступа: https://www.ansible.com (дата обращения 09.04.2021).
18. SaltStack. Режим доступа: https://www.saltstack.com (дата обращения 09.04.2021).
19. Ansible PlayBook examples repository. Режим доступа: https://github.com/ansible/ansible-examples/blob/master/tomcat-standalone/roles/tomcat/tasks/main.yml (дата обращения 09.04.2021).
20. Puppetlabs manifest examples repository. Режим доступа: https://github.com/puppetlabs/puppetlabs-mysql (дата обращения 09.04.2021).
21. Ilyasov B.G., Makarova E.A., Zakieva E.Sh., Gabdullina E.R. Metodologicheskiye osnovy modelirovaniya i intellektual'nogo upravleniya promyshlennym kompleksom kak slozhnym dinamicheskim mnogoagentnym ob"yektom [Methodological foundations of modeling and intelligent management of an industrial complex as a complex dynamic multi-agent object] // Sovremennyye naukoyemkiye tekhnologii = Modern high technologies. 2020. № 11-2. P. 288–293. DOI: 10.17513/snt.38376.
22. Tchernykh A., Bychkov I., Feoktistov A., Gorsky S., Sidorov I., Kostromin R., Edelev A., Zorkalzev V., Avetisyan A. Smyagcheniye neopredelennosti pri razrabotke i primenenii nauchnykh prilozheniy v integrirovannoy vychislitel'noy srede [Mitigating Uncertainty in Developing and Applying Scientific Applications in an Integrated Computing Environment] // Trudy ISP RAN = Proceedings of ISP RAS. 2021. Vol. 33. № 1. P. 151–172. DOI: 10.15514/ISPRAS-2021-33(1)-11.