УДК 004.45

ИНФРАСТРУКТУРА УПРАВЛЯЕМАЯ КОДОМ: МЕТОДОЛОГИЯ И ИНСТРУМЕНТЫ

М.А. Овчинников, Е.В. Овчинникова

INFRASTRUCTURE AS A CODE: METHOLODGY AND TOOLS

M.A. Ovchinnikov, E.V. Ovchinnikova

Аннотация. За счет развития облачных технологий и средств автоматизации в датацентрах, у разработчиков программного обеспечения появились новые возможности по организации программного окружения для разрабатываемых проектов. В статье рассмотрены предпосылки к появлению подхода «Инфраструктура управляемая кодом», основные принципы и приведены примеры используемых инструментов.

Ключевые слова: *инфраструктура управляемая кодом; системное администрирование; DevOps; управление проектами; разработка программного обеспечения*.

Abstract. Development of cloud and automation technologies in data centers, allows software developers to use new approaches for organization of environments for their projects. The article describes the background of approach called "Infrastructure managed code," as well as the basic principles and examples of tools used.

Keywords: *infrastructure as a code; system administration; DevOps; project management; software development.*

Введение

Термин “Инфраструктура управляемая кодом” (Infrastructure as a Code) появился относительно недавно. Отметим, что под термином «инфраструктура» имеются в виду не физические сервера, сетевая инфраструктура и другие аппаратные компоненты, а здесь и далее, это наборы программного обеспечения, библиотек и компонентов операционной системы которые необходимы для полноценного функционирования приложений предприятия. Развитые технологии виртуализации позволяют абстрагироваться от особенностей построения физической инфраструктуры, используя лишь удобные программные интерфейсы. За счет развития облачных технологий и средств автоматизации в датацентрах, поднять десятки, сотни и даже тысячи виртуальных серверов стало возможным лишь запуском единственной команды.

С появлением таких возможностей, изменились и подходы к системному администрированию. Изменения в инфраструктуре крупных предприятий всегда проходили медленно и болезненно, а знания об ее организации хранились чаще в головах системных администраторов, либо в большом количестве документов, многие из которых очень быстро утрачивали актуальность. Зачастую увольнение одного системного администратора и приход нового могли означать полное перестроение всего рабочего окружения в рамках вверенных ему полномочий.

Источником любых изменений всегда являются изменения в бизнес-процессах. Увеличивается количество клиентов, появляются новые партнеры и поставщики, системами которых необходимо осуществить интеграцию, приходится подстраиваться под меняющееся законодательство, наконец, просто поддерживать парк используемых технологий на современном уровне. Долгое время требования об изменениях приходили в ИТ-отделы в виде абстрактной документации, которая интерпретировалась каждым специалистом по-

своему. Изменения инфраструктуры проходили медленно, с большим количеством проблем, таких как несоответствие версий операционных систем, отсутствие нужных программных компонентов на ключевых серверах инфраструктуры и т.д.

Неудивительно, что последние 15-20 лет, ИТ-отрасль была очень сильно отделена от бизнеса. Ситуация стала меняться только в последнее десятилетие с появлением так называемых “стартапов” — технологичных, как правило малых, предприятий, направленных на быстрый рост и развитие. Технически подкованные бизнесмены, либо инженеры с деловой жилкой, уставшие от неэффективной разработки продуктов и услуг в больших корпорациях, стали объединяться в небольшие группы, чтобы выпускать программное обеспечение, которое соответствовало их видению и внутренним ценностям, а главное, они смогли начать применять инновационные подходы и технологии недоступные в консервативном корпоративном секторе.

Финансовые ограничения малого бизнеса заставляют отказываться от подходов присущих корпоративному миру. Так появляются гибкие (Agile) методологии управления проектами (Scrum, Kanban), приходят новые подходы к программированию (парное программирование, TDD), наконец появляется DevOps (*Development & Operations* - методология разработки программного обеспечения, нацеленная на активное взаимодействие и интеграцию специалистов по разработке и специалистов по информационно-технологическому обслуживанию), который распространяет практики разработчиков на системное администрирование, фактически стирая грань между этими двумя областями ИТ.

Инфраструктура также не смогла остаться без изменений, что связано с рядом ранее обозначенных проблем:

- ручной, малоэффективный труд системных администраторов;
- документация с описанием инфраструктуры, которая редко поддерживается в актуальном состоянии;
- высокий порог вхождения для новых членов команды;
- как следствие: низкая скорость изменения инфраструктуры и медленная реакция на изменения бизнес-процессов.

В связи с тем, что от использования реальных серверов осуществляется переход к виртуальным машинам, то, к примеру, такие операции, как увеличение оперативной памяти или дискового пространства сводятся лишь к передаче определенных команд гипервизору. Таким образом, если описать все эти команды в виде программного кода, можно будет легко автоматизировать любые требуемые операции. При таком подходе становится реально производить мониторинг и реагировать на изменения в автоматическом режиме. При увеличении нагрузки можно легко поднимать дополнительные машины, так как их конфигурация уже определена. В этом случае отпадает и проблема написания документации: все уже описано в конфигурационных файлах.

Отдельным преимуществом становится, то, что всю конфигурацию инфраструктуры теперь можно поместить в систему контроля версий (Git, SVN), видеть историю изменений, автоматически обновлять инфраструктуру при помощи средств непрерывной интеграции (Jenkins, TeamCity), т.е. работать с ней точно таким же способом, как и с кодом самого приложения.

Рассмотрим несколько типичных инструментов используемых при автоматизации инфраструктуры.

Vagrant [1] — надстройка над платформой виртуализации VirtualBox, которая позволяет легко и быстро создавать виртуальные машины по шаблону. Vagrant нужен для одной простой цели — тестировать выкатку и изменение конфигурации.

Cobbler [2] — Linux installation server. Универсальная система для организации сетевой загрузки и обновления группы машин, поддерживающей наиболее популярные Linux дистрибутивы.

Cobbler позволяет, используя простой набор команд, сконфигурировать систему для бездисковой загрузки, переустановки, установки по сети, инсталляции в виртуальное окружение Xen, qemu, KVM или VMware Server.

Puppet [3] — кроссплатформенное клиент-серверное приложение, которое позволяет централизованно управлять конфигурацией операционных систем и программ, установленных на нескольких компьютерах. Puppet написан на языке программирования Ruby.

Puppet позволяет просто настроить и впоследствии быстро управлять практически любой сетью на базе любой операционной системы Red Hat, CentOS, Fedora, Debian, Ubuntu, OpenSUSE, Solaris, BSD, Mac OS X и Microsoft Windows (через cygwin).

Узлы сети, управляемые с помощью Puppet, периодически опрашивают сервер, получают и применяют внесённые администратором изменения в конфигурацию. Конфигурация описывается на специальном декларативном языке.

Использование инструментов управления конфигурацией, таких как Puppet, позволяет больше не задумываться над деталями конкретной реализации и сосредоточиться на функциональности и взаимодействии компонентов. Основным преимуществом такого подхода является то, что система всегда находится в известном состоянии, которое описано в коде, исключается человеческий фактор, при разворачивании сложных систем и экономится время за счет полного исключения ручных операций. Puppet — не единственный инструмент подобного рода, существуют также Chef, Ansible, Salt и многие другие, которые определяют различные подходы к описанию конфигурации инфраструктуры, но служат аналогичным целям.

Программный код, которым описывается инфраструктура, может быть частью программного обеспечения, либо распространяться отдельно, но требования к его написанию и оформлению будут оставаться аналогичными, не смотря на то, что зачастую он может иметь лишь вспомогательное назначение.

Подход к написанию кода будет варьироваться от команды к команде, тем не менее, можно выделить некоторые общие практики:

- код находится под контролем версий. В качестве системы контроля версий, как правило, используется Git, реже Mercurial или SVN;

- используется единый стиль оформления кода (именование переменных, оступы, а также распространенные практики характерные для конкретного языка). Большинство инфраструктурного кода разрабатывается на языках Ruby или Python. Для Python существует стандарт официальный стандарт PEP8 и одноименное средство статического анализа кода. Для языка Ruby нет официального стандарта, однако компании и независимые разработчики предлагают несколько открытых стандартов, среди которых наибольшую популярность набрал Ruby Style Guide разработанный болгарским программистом Божидаром Бацовым и его инструмент для статического анализа — Rubocop;

- код покрыт тестами. Для тестирования инфраструктуры разработаны специальные инструменты: Test Kitchen [4] — для автоматического создания виртуальных сред для запуска и тестирования инфраструктурного кода, ServerSpec — фреймворк для проверки результата работы систем управления конфигурацией;

- тесты регулярно запускаются в системе непрерывной интеграции (CI). Стандартом де-факто среди таких систем является Jenkins, также активно используются TeamCity, TravisCI и многие другие.

Заключение

Появление у разработчиков программного обеспечения новых возможностей по организации программного окружения для разрабатываемых проектов за счет развития облачных технологий и средств автоматизации в датацентрах позволило создать условия для реализации подхода «Инфраструктура управляемая кодом», который обладает рядом преимуществ:

- конфигурация инфраструктуры для приложения документирована программным кодом, который её разворачивает, каждый член команды с минимальными навыками программирования способен разобраться как она устроена;
- все находится под системой контроля версий, что позволяет быстро понять какие конкретно изменения были внесены и кем;
- система всегда находится в известном состоянии, которое однозначно определяется программным кодом;
- существуют инструменты позволяющие абстрагироваться от используемой операционной системы и описывать поведение системы универсальными методами, что упрощает миграцию между системами;
- так как ручные операции полностью исключаются, ощутимо сокращается время на развертывание и подготовку среды и исключается влияние человеческого фактора.

Однако, необходимо учитавать определенные трудности, которые может повлечь такой подход:

- потребуется время на обучение персонала и внедрение соответствующей подходу культуры;
- потребуется существенная переработка и переосмысление текущей инфраструктуры, что может вылиться в существенные расходы;
- инфраструктура на основе ОС Windows и продуктов Microsoft, существенно осложняют задачу, так как гибкость таких решений, как правило, существенно ниже, чем у продуктов на базе открытых технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Vagrant. Информ.-справочный портал [Электронный ресурс]. URL: http://evtuhovich.ru/blog/2012/01/10/vagrant/ (дата обращения: 25.10.2015).
2. Cobbler. Информ.-справочный портал [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Cobbler_%28software%29 (дата обращения: 25.10.2015).
3. What is Puppet? Информ.-справочный портал [Электронный ресурс]. URL: https://puppetlabs.com/puppet/what-is-puppet (дата обращения: 25.10.2015);
4. Тестируем инфраструктуру как код. Информационно-справочный портал [Электронный ресурс]. URL: http://habrahabr.ru/company/express42/blog/256725/ (дата обращения: 25.10.2015).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Овчинников Михаил Андреевич
ФГБОУ ПО «Рязанский государственный университет имени С.А. Есенина», г. Рязань, Россия, аспирант,
E-mail: michail@ovchinnikov.cc
Ovchinnikov Mikhail Andreevich
FSEI PE «Ryazan State University named for S.Yesenin», Ryazan, Russia, graduate student,
E-mail: michail@ovchinnikov.cc

Овчинникова Елена Вадимовна
ФГБОУ ПО «Рязанский государственный университет имени С.А. Есенина», г. Рязань, Россия, кандидат технических наук, доцент кафедры общей и теоретической физики и МПФ,
E-mail: e.ovchinnikova@rsu.edu.ru

Ovchinnikova Elena Vadimovna
FSEI PE «Ryazan State University named for S.Yesenin», Ryazan, Russia, Ph.D., Associate Professor
E-mail: e.ovchinnikova@rsu.edu.ru