1 Что такое облако

Хотите —
буду от мяса бешеный
— и, как небо, меняя тона —
хотите —
буду безукоризненно нежный,
не мужчина, а — облако в штанах!

Владимир Маяковский.
Облако в штанах

1.1. Общие сведения

Облачные технологии появились совсем недавно: в 2006 году один из крупнейших американских интернет-магазинов Amazon предоставил свои неиспользуемые вычислительные ресурсы (а к тому времени их объем стал огромным) совершенно новым образом. Традиционно для аренды ресурсов в дата-центрах необходимо было составить договор и внести плату за определенный срок. Линейка типоразмеров серверов (объем оперативной памяти, количество ядер, размер дискового пространства и др.) достаточно обширна и выбирается заранее, до подписания договора. Можно арендовать много серверов, связать их высокопроизводительной сетью, подключить балансировщик нагрузки и получить систему, обрабатывающую большую нагрузку. У подобной модели использования ресурсов есть существенные неудобства. При создании приложений зачастую неизвестно, какая потребуется нагрузка, на какой срок арендовать серверы приложения. Или такой пример: создается стартап, арендуются серверы и до завершения срока аренды этот стартап «умирает». Что делать со ставшими ненужными арендованными серверами? Еще сложнее дело обстоит с покупкой физических серверов. Ведь их, помимо администрирования операционной системы и установленных приложений, необходимо обслуживать физически. Сюда входит подбор помещения, электропитания, системы охлаждения, вентиляции... Все эти проблемы можно решить с помощью эластичных вычислительных ресурсов, предоставляемых облачными провайдера-

ми. (Платформа Amazon Web Services называет эти ресурсы EC2 — Elastic Cloud Computers.) Ключевые преимущества облачной модели таковы:

- ресурсы предоставляются по требованию и таким же образом освобождаются;
- плата начисляется за фактическое время использования ресурсов;
- предоставление и освобождение ресурсов производится самим потребителем ресурсов через веб-портал, без всякой бумажной волокиты с договорами.

Помимо виртуальных машин, в облачных средах предоставляются различные сервисы, позволяющие строить различные архитектуры: сервисы виртуальных сетей, подсетей, балансировщики нагрузки, списки контроля доступа (Access Control Lists, ACL)), выделенные IP-адреса и др. Эти сервисы составляют основу инфраструктуры как сервиса (Infrastructure as a Service, IaaS). Конечно, *прямая* стоимость годовой аренды физического сервера может быть меньше, чем стоимость аренды облачного сервера с почасовой оплатой с такими же характеристиками, но многие облачные провайдеры (например, AWS) предоставляют возможность долгосрочной аренды виртуальных машин по ценам существенно меньшим, чем при почасовой оплате. Если же серверы требуются на небольшое время и заранее не известно, какого размера должна быть виртуальная машина, то эластичные виртуальные машины могут стать единственным приемлемым выбором. В случае же прямой покупки физических серверов задача выбора, приобретения, настройки и обслуживания, а также их продажи после применения становится весьма непростой. Чтобы обеспечить возможность выделения пользователям ресурсов, облачные провайдеры имеют крупные, географически разнесенные дата-центры, веб-порталы для получения доступа к их ресурсам, а также API для программного доступа. Это позволяет сделать то, что нельзя выполнить с помощью любой другой традиционной технологии: код программы может сам себе выделять столько ресурсов, сколько ему нужно. Или же программы могут создавать инфраструктуру, на которой они будут выполняться.

Помимо «голой» инфраструктуры, облачные провайдеры предоставляют наиболее типовые приложения в виде веб-сервисов. В качестве примера можно привести облачное хранилище данных (cloud storage), сервис предоставления учетных записей (identity provider), сервис хостинга веб-приложений, базу данных как сервис, брокер сообщений, концентратор сообщений и др. Все эти сервисы, кажущиеся на первый взгляд разрозненным набором, предоставляются как общая платформа. Доступ к ним унифицируется в виде единообразных API, SDK, возможны их «соединение» между собой, общий мониторинг логов и событий и пр. Это иной уровень применения ресурсов облака: платформа как сервис (Platform as a Service). PaaS позволяет пользователям создавать не просто программные продукты в рамках одной операционной системы, веб-платформы и др., но целые информационные системы, компонентами которых будут экземпляры облачных сервисов. Подобно IaaS, сервисы PaaS обычно допускают масштабирование (как ручное, путем выбора соответствующего их размера, так и автоматическое, с помощью различных метрик и событий). Как правило, сервисы PaaS предоставляют

гораздо меньшие права для доступа к вычислительным ресурсам инфраструктуры, лежащей в их основе. Например, сервисы хостинга веб-приложений не позволяют установить специфические программы, COM-компоненты, поменять библиотеку DLL в GAC, изменить запись в реестре и др., поскольку отсутствует root-доступ. Но взамен они предоставляют удобные порталы администрирования, интеграцию с другими сервисами, встроенные средства логирования и мониторинга, доступность 99,99 % времени и др.

В настоящее время крупнейшими облачными провайдерами являются Amazon Web Services (AWS), Microsoft Azure, Google Cloud, IBM Bluemix, Oracle. В книге приведены описания сервисов двух облачных провайдеров: AWS и Microsoft Azure. AWS — первый в истории облачный провайдер, а Microsoft Azure — облачный провайдер от корпорации Microsoft, обеспечивающий интеграцию практически со всеми сервисами Microsoft.

1.2. Способы создания ресурсов в облаке

> В каюте первого класса Остап, лежа с башмаками на кожаном диване и задумчиво глядя на пробочный пояс, обтянутый зеленой парусиной, допрашивал Ипполита Матвеевича:
>
> — Вы умеете рисовать? Очень жалко. Я, к сожалению, тоже не умею.
>
> Он подумал и продолжал:
>
> — А буквы вы умеете рисовать? Тоже не умеете? Совсем нехорошо! Ведь мы-то попали сюда как художники. Ну, дня два можно будет мотать, а потом выкинут.
>
> *Ильф и Петров. Двенадцать стульев*

Прежде чем начать описывать способы создания ресурсов, поясню, что это такое. Как отмечалось выше, облачные провайдеры имеют в основе своих сервисов огромные дата-центры, чьи вычислительные ресурсы с помощью системы виртуализации разделяются на небольшие части: голые виртуальные машины различных размеров с установленной операционной системой (IaaS) и группы виртуальных машин с установленным софтом, предоставляющим доступ только к своим возможностям (PaaS). Так вот, создать облачный ресурс — значит отправить запрос контроллеру ресурсов, размещенному в облачном ЦОДе, на выделение требуемых вычислительных ресурсов из пула доступных. То есть, по сути, ресурс не создается из ничего, а только выделяется по требованию. И тут

возможна ситуация (редко, но бывает), когда пользователь запросил ресурсы у контроллера, а они не появились. Это случается из-за того, что физические ресурсы, на которых размещаются виртуальные, уже задействованы другими пользователями. Задачу оптимального распределения доступных ресурсов между пользователями целиком решает контроллер. И если пользователь при создании ресурсов столкнулся с проблемой, он должен повторить попытку, прибегнув к различным вариациям (повторить через некоторое время, сменить регион и повторить, сменить аккаунт и повторить и пр.).

С точки зрения пользователя, выделение ресурсов выглядит как создание ресурсов: он выполнил ряд действий на веб-портале, и в последнем появились ресурсы. На самом деле, конечно же, они были выделены, и об этом не стоит забывать. Однако для простоты и наглядности я буду применять термин «создание».

Существует четыре способа управления облачной инфраструктурой (рис. 1.1). Первый, самый простой и очевидный — задействовать веб-портал. При этом пользователь должен иметь соответствующие права на создание ресурсов. Ручной способ очень прост: у всех облачных провайдеров есть удобные порталы, обширная документация, видеоинструкции и др. Не нужны никакие дополнительные сервисы, SDK и пр.

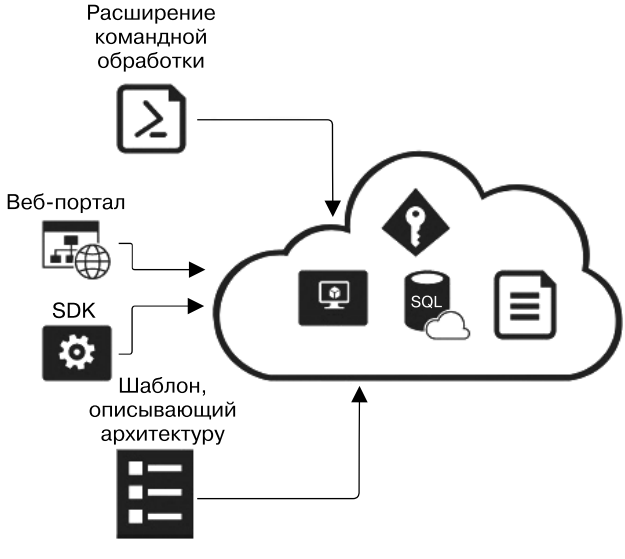

Рис. 1.1. Способы управления облачной инфраструктурой

Однако у данного способа есть недостатки:

- длительное время создания инфраструктуры;
- недостаточная надежность (в случае проблем с ресурсами их придется пересоздавать вручную, со всеми ручными настройками, конфигурированием и пр.);

- трудность переноса инфраструктуры в новый регион или аккаунт — ее понадобится вручную клонировать или копировать (данный недостаток частично сглаживается тем, что облачные провайдеры позволяют копировать или клонировать ресурсы, но эта процедура все равно требует ручного инициирования);
- процесс создания ресурсов в этом случае невозможно автоматизировать.

Второй способ — применить программные библиотеки (Software Development Kit, SDK), обеспечивающие доступ к ресурсам облака из кода пользовательских программ. Как правило, SDK представляет собой набор классов и методов, облегчающих программные операции с ресурсами облака. Чтобы обеспечить доступ к таким ресурсам, программа с облачным SDK должна содержать ключи учетной записи, которая будет иметь доступ к облаку. В облаке эти ключи зарегистрированы в виде пользователя в активном каталоге облачного аккаунта, обладающего правами выполнять программное манипулирование ресурсами облака (такой пользователь называется принципалом — service principal). И управление этими учетными записями происходит таким же образом, как и учетными записями пользователей облачного веб-портала. В числе достоинства такого подхода — возможность создания программ, которые сами себе создают облачные ресурсы, а также автоматизированного управления облачным аккаунтом.

К третьему способу создания облачных ресурсов относят специализированные расширения для языков командной строки — shell, CMD и др. (например Azure PowerShell, AWS CLI и пр.), работающие в ней напрямую. Для подключения этих расширений к облачным ресурсам необходимо импортировать ключи или выполнить вход в аккаунт через форму ввода логина/пароля. Как и в случае SDK для сценарных языков программирования, SDK для командной оболочки позволяет описывать облачную инфраструктуру в виде набора команд, каждая из которых создает или конфигурирует соответствующий облачный сервис.

И SDK, и команды оболочки оперируют в конечном итоге с API облачного провайдера (как правило, REST API), доступ к которым также позволит манипулировать ресурсами облака.

Четвертый способ создания облачных ресурсов — применить шаблоны. В этом случае все требуемые ресурсы и связи между ними описываются с помощью текстового файла в формате YAML или JSON. Такой шаблон может быть загружен в соответствующий облачный сервис напрямую через веб-портал или через CLI-команды.

Описание инфраструктуры через шаблон — очень мощный механизм, широко применяемый для конфигурирования различных инфраструктур (например, в системах Ansible, Chef, Puppet и др.). Как уже указывалось, шаблоны представляют собой текстовые файлы, которые могут храниться в репозитории шаблонов или чаще всего в репозитории GitHub. Для облака AWS сервис создания ресурсов с помощью шаблонов называется CloudFormation (поддерживает YAML и JSON), у Azure это ARM Template (в настоящее время поддерживает только JSON).

На веб-портале AWS имеется специальный редактор, упрощающий создание и конфигурирование шаблона. Последний может быть загружен в файловое хранилище S3, репозиторий CodeCommit или любое другое место, доступное для сервиса CloudFormation. Этот сервис создает стек — набор ресурсов, управляемых совместно (создание, удаление и обновление).

Сервис CloudFormation очень удобен в применении со сторонними сервисами конфигурирования — например, Ansible. Это широко используемое приложение, задействующее YAML для создания конфигурационных шаблонов, которые служат для администрирования группы серверов (преимущественно Linux, но есть расширения и для Windows), не требуя инсталляции на этих серверах «агентов». Для работы Ansible необходимы только ключи доступа к ресурсам (SSH-ключи для Linux-хостов, сертификат для PowerShell-доступа к Windows-хостам или ключи доступа к AWS). Шаблон CloudFormation для Ansible представлен в виде JINJA, допускающего передачу параметров через переменные Ansible.

1.3. Безопасность облачных ресурсов

> В мире существует нежелательный парадокс: чем больше власти, тем меньше ответственности.
>
> *Валентин Пикуль. Битва железных канцлеров*

> Воруют так, что печку раскаленную нельзя без присмотра оставить.
>
> Отвернись только — и печку голыми руками вынесут...
>
> *Валентин Пикуль. На задворках великой империи*

Наряду с неоспоримыми преимуществами, хранение и обработка данных в облачных средах потенциально может доставить ряд проблем, которых нет (или, вернее, они проявляются не так отчетливо) в случае размещения и обработки данных в собственных дата-центрах. Это обусловлено рядом причин. Во-первых, облачные среды сами по себе публично доступны и все сервисы, если явно не сконфигурировано иное, доступны для всех в Интернете. Во-вторых, защита данных и инфраструктуры от непреднамеренных действий пользователей лежит вне компетенции облачного провайдера. Кроме того, облачные инфраструктуры, работающие с большими данными, часто содержат в своем составе большие кластеры виртуальных машин, что требует применения специальных мер для обеспечения надежной работы всей системы. Помимо этого, информация физически будет передаваться

по незащищенным каналам и существует угроза ее перехвата. Рассмотрим подробнее все перечисленные и некоторые другие аспекты безопасности облачных сред.

Наиболее распространенный способ защиты конечных точек облачных сервисов — ограничение доступа к ним с помощью механизмов аутентификации и создания списков разрешенных IP-адресов, с которых можно получить доступ к точкам. Рассмотрим прежде всего различные способы обеспечения доступа из заданного адресного пространства.

Сервисы, относящиеся к IaaS, а также в ряде случаев к PaaS, требуют для своего создания сконфигурированной облачной виртуальной частной сети (VNet, VPC), разбитой на подсети. Доступ к конечным точкам сервисов, расположенным в этих подсетях, можно регулировать с помощью конфигурирования сетевых групп безопасности (Network Security Group, NSG) (рис. 1.2), которые представляют собой списки контроля доступа, ACL.

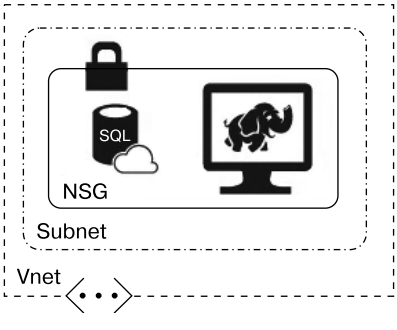

Рис. 1.2. Ограничение доступа к конечным точкам облачных сервисов с помощью сетевых групп безопасности

Итак, виртуальная часть сети — один из базовых сервисов IaaS. Он представляет собой облачный аналог локальной сети и служит для предоставления диапазона IP-адресов для размещения в них ресурсов. Виртуальную частную сеть можно разделить на подсети (subnet), а между ними — установить правила маршрутизации IP-пакетов. Кроме того, на подсети можно установить списки контроля доступа, которые именуются сетевыми группами безопасности. Это позволяет логически разделять архитектуры информационных систем на различные уровни (например, уровень данных, бизнес-логики, фронтенд) путем размещения каждого уровня в своей подсети и установления правил маршрутизации.

NSG представляет собой список доступа, содержащий набор записей. Каждая запись состоит из таких элементов, как:

- название;
- число, определяющее приоритет просмотра списка записей;
- диапазон IP-адресов (для одного конкретного адреса это /32);
- номер порта;

- действие — ALLOW или DENY («Позволить» или «Отклонить») по отношению к запросу, поступившему с данного адреса.

Кроме того, указывается протокол, к которому применимо действие ALLOW или DENY (TCP, UDP, ICMP и пр.). Безопасность конечных точек в данном случае обеспечивается ограничением к ним доступа извне. Помимо NSG, ряд облачных сервисов, не требующих виртуальной частной сети (например, Azure SQL), имеют фаерволы — списки «разрешенных» и «запрещенных» диапазонов. Хорошей практикой является повсеместное использование NSG и фаерволов. При этом необходимо, чтобы все порты, относящиеся к удаленному доступу/управлению (например, 22 для SSH, 3388 для RDP) или непосредственно к сервису (скажем, 1433 для MS SQL), были недоступны из Интернета вне диапазона адресов виртуальной частной сети. Для получения же доступа к сервисам из «разрешенной» локальной сети или с разрешенного компьютера следует установить VPN-шлюз из локальной сети или с компьютера в виртуальную частную сеть или непосредственно к экземпляру сервиса. Помимо шлюза, при соединении локальной сети с виртуальной частной сетью необходимо применить промежуточный хост, прокси-хост (рис. 1.3), который будет транслировать запросы и разрешать частные адреса облачных ресурсов из локальной сети. Прокси-хост в различных реализациях можно разместить как в облачной сети, так и в локальной. В последней может располагаться контроллер домена, сервер БД с данными, которые не могут быть размещены в облаке, а также прочие серверы, которые могут быть размещены только в локальной сети.

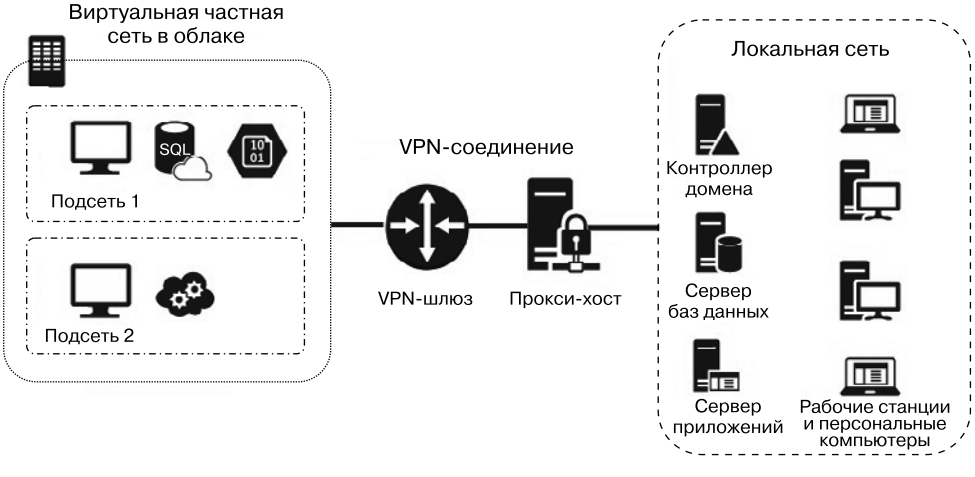

Рис. 1.3. Обеспечение защищенного доступа к облачным ресурсам

Некоторые конечные точки (скажем, API управления самого облачного аккаунта) не оборудованы ни фаерволом, ни NSG. Для их защиты используется как шифрование трафика (HTTPS), так и специальные ключи доступа к ресурсам. Ключи могут генерироваться непосредственно защищаемым сервисом и применяться

в заголовках REST-запросов. Например, сервис хранилища Azure Storage задействует аутентификацию на основе HMAC — Hash-based Message Authentication Code, который должен содержать подписанный алгоритмом SHA-256 токен как заголовок аутентификации. Следующий стандартный механизм аутентификации запросов — применение протокола OAuth 2.0, при котором пользовательские учетные данные хранятся в сервисе хранения учетных данных (в случае Azure это Azure Active Directory, а AWS — Cognito). В общем случае облачный сервис хранения учетных данных включает в себя пользователей, роли и API-ресурсы, защищаемые этим протоколом. Любой запрос к REST API ресурсов, зарегистрированным в этом сервисе (а это, по сути, все REST API конечных точек управления облачными ресурсами), должен в своем заголовке содержать токен, который можно получить, если выполнить процедуру ввода учетных данных в каталог.

Еще один «эшелон» защиты данных в облачных ресурсах — это их шифрование. Распространенный подход в данном случае — так называемое прозрачное шифрование данных (transparent data encryption, TDE). Суть его состоит в том, что все шифрование и дешифрование данных происходит «за кулисами», без участия пользователя, с помощью ключей шифрования, которые генерируются и хранятся самим облачным аккаунтом. В случае Azure эти ключи хранятся в Azure KeyVault, а в случае AWS — в AWS KMS.

Следующее звено защиты — сервисы, отвечающие за мониторинг запросов, поступающих к ресурсам и осуществляющие аудит пользовательской активности. Например, Azure Security Center, который может выполнять мониторинг очень многих ресурсов, выдает рекомендации по их защите и следит за входящим трафиком. На практике в реальной рабочей системе подобный сервис позволил обнаружить и успешно отразить несколько крупных хакерских атак на систему, за которую я отвечал, в том числе с использованием распределенной сети зараженных серверов (ботнет). Поэтому я настоятельно рекомендую применять подобные сервисы для обязательной защиты облачных ресурсов.

И последняя линия защиты данных, встроенная в облачные ресурсы, включает в себя сервисы анализа пользовательской активности, например Audit & Threat Detection для Azure SQL. Этот сервис обеспечивает внутренний аудит всех запросов в базе данных Azure SQL и анализ их на предмет подозрительных действий. У данного вида защиты есть один недостаток: в результате использования в реальной системе при включении максимального уровня аудита и защиты существенно снижается отзывчивость и общая производительность.

1.4. Резюме

В данной главе мы рассмотрели пути создания облачных ресурсов и уровни обеспечения безопасности. Эти ресурсы можно создавать с помощью веб-портала, через SDK (или напрямую с использованием REST API), расширения для интерфейса командной строки и, наконец, применив специальный шаблон, описываю-

щий требуемое состояние облачных ресурсов. Шаблоны автоматизации наиболее удобны при построении больших и сложных систем, поскольку обеспечивают полный контроль над ресурсами и не допускают неконтролируемого разрастания неиспользованных ресурсов и, соответственно, увеличения месячного счета за использованные ресурсы.

Вопросы безопасности информации в облаках являются комплексными. Пользователь облачных ресурсов сам отвечает за защиту своих ресурсов от несанкционированного доступа и должен самостоятельно выбирать для этого стратегию.