Добрый день, уважаемые студенты! Я рад вас видеть на очередной лекции. Сегодня мы с вами поговорим про Networking и Content Delivery. Итак, давайте начнем.

Сегодняшняя наша лекция поделена на шесть частей. В первой части мы поговорим про основы сетей и рассмотрим базовые понятия, т.е. о Networking basics. Далее, в следующие три части, поговорим про сервис Amazon Virtual Private Cloud или чаще вы можете встретить как Amazon VPC. Затронем темы VPC networking и VPC security. Последние две темы это отдельные два сервиса, первый это – сервис Amazon Route 53, а второй это – сервис Amazon CloudFront.

Мы начинаем первую секцию, это основы сетей. Мы рассмотрим основы сетей для того, чтобы облегчить вам понимание следующих секций, а именно секции, когда мы говорим про сети в AWS, в частности сервис Amazon VPC. Что такое сеть? Сеть – это два или более клиентских машин, соединенных между собой для обмена информации. Сеть может быть поделена на так называемые логические части, которые называются subnets. Для того, чтобы соединить машины между собой, нам необходимо сетевое оборудование. Как сетевое оборудование может выступать switch или router. У каждой машины внутри сети есть уникальный IP-адрес, который идентифицирует это устройство внутри определенной сети.

IP-адрес – это некоторое числовое обозначение в десятичной системе счисления, которое состоит из четырех частей. Далее машина переводится в двоичную систему счисления, как вы видите, и таким образом программа на низшем уровне работает с этим IP-адресом. Мы же его видим в десятичной системе счисления, и как пример вы видите 192.0.2.0. Каждый отсек, раздел внутри IP-адреса это какое-то число, которое может принимать значения от нуля до 255, т.е. 256 уникальных значений. 256 это 2 в степени 8, что говорит о том, что в двоичной системе счисления для обозначения чисел от нуля до 255 нам нужно выделить 8 бит. Начиная от нулей 8 раз (00000000), заканчивая единичками 8 раз (11111111). Таким образом IP-адрес в бинарном формате представляет из себя 8 умноженное на 4 = 32 бита.

Мы с вами проговорили, что IP-адрес состоит из 32 битов. Такой адрес называется IPv4-адрес. У нас также есть другой вид IP-адреса, это IPv6, который состоит из 128 битов, и оно может сгенерировать еще большее количество уникальных IP-адресов для конкретной сети. Когда мы говорим про IPv6-адрес, то оно представлено в виде 8 групп. В каждой группе могут выступать буквы и цифры. Мы видим этот адрес в шестнадцатеричной системе счисления и каждая группа из себя представляет 16 битов, и может принимать значение от нуля до FFFF. Это опять же повторюсь в шестнадцатеричной системе счисления. В работе вы не так часто будете сталкиваться напрямую с IPv6-адресами. В основном будете работать с IPv4-адресами. Поэтому давайте рассмотрим ее по подробнее.

В локальной сети для того, чтобы определить список возможных значений IPv4-адресов, мы используем так называемый CIDR блок. CIDR расшифруется как Classless Inter-Domain Routing. Он записывается в следующем формате: это стартовый, т.е. самый первый IP-адрес в формате IPv4, далее слэш и какой-то номер, который может принимать значение от 0 до 32. И означает, какое количество битов у нас зафиксировано. Мы с вами помним, что IPv4-адрес состоит из 32 битов. Поэтому если мы видим, что в CIDR блоке написано 24, это говорит о том, что 24 бита из 32 зафиксировано и меняться не может. Таким образом, 8 битов могут меняться. Когда мы говорим 8 битов, то это 2 в 8 степени количества уникальных IP-адресов, 2 в степени 8 это 256. Таким образом, конкретно вот CIDR блок, указанный на нашем слайде, может принимать 256 уникальных IP-адресов. Самый первый IP-адрес, который относится к этой сети, это 192.0.2.0. Таким образом, дальше идет от 0 последняя часть, которая у нас может меняться, 0.1.2.3.4. И так до самого последнего, который будет иметь IP-адрес 192.0.2.255. Хорошо. Я надеюсь здесь понятно, а для того чтобы закрепить материал, мы с вами сделаем следующую активность.

Давайте представим, что у нас есть CIDR блок, у которого первая часть совпадает, то есть 192.0.2.0 слэш вместо 24 будет 16. Давайте остановим это видео и попробуем посчитать, какое количество уникальных адресов будет в этой сети с таким CIDR блоком. Хорошо, я думаю, все из вас справились. Правильный ответ это 65536. Как мы вышли на это число? Мы с вами говорили, что CIDR блок у нас 192.0.2.0/16. Это говорит о том, что из 32 битов IP-адреса 16 у нас зафиксированы. Значит, ровно 16 битов у нас flexible, то есть могут меняться. Когда мы говорим 16 битов, это значит 2 в степени 16 уникальных IP-адресов. Если мы посчитаем 2 в степени 16, это 65536. Хорошо, я думаю, здесь стало более понятнее.

Давайте разберем еще два граничных случая. Представим, что у нас CIDR блок следующего вида 192.0.2.0/32. Давайте сделаем еще одну активность и попробуйте посчитать, какое количество IP-адресов есть в этом CIDR блоке. Можете остановить видео. Хорошо, я очень надеюсь, что каждый из вас смог правильно посчитать и получить верный ответ. Правильный ответ – это 1. CIDR блок, где окончание /32 идентифицирует одну машину. Как это посчитать? Когда мы говорим /32, это говорит о том, что 32 бита у нас зафиксированы. А так как у нас всего в IPv4 адресе 32 бита, то чтобы посчитать, какое количество уникальных адресов у нас есть, мы отнимаем от 32 количество зафиксированных битов, а они у нас тоже 32. 32-32 получается 0, а 2 в степени 0 у нас 1. Таким образом, вы запомните, что в Amazon, когда вы работаете в AWS Management Console, для того чтобы указать конкретный IP-адрес, вы знаете что CIDR блок – это /32, ну и указываете, соответственно, ваш IP-адрес.

И второй граничный случай, я не буду его задавать как активность, давайте сразу отвечу. Когда мы говорим, что у нас CIDR блок 0.0.0.0/0, в этом случае мы понимаем, что мы начинаем с самого первого IP-адреса, так как у нас 0 количество зафиксированных битов, то все биты могут меняться, т.е. flexible. Это говорит о том, что у нас количество уникальных адресов в этой сети 2 в степени 32, и это представляет из себя весь интернет. Также в AWS Management Console, когда вам нужно указать, что доступ открываете всему интернету, либо route путь прокладываете до интернета, либо запрещаете выход в интернет, то как CIDR блок вы всегда можете указывать все нули через точку слеш 0, это и будет интернетом.

Следующая модель, с которой нам нужно познакомиться, возможно вы ее уже прошли, это OSI модель, т.е. Open Systems Interconnection Model. Основная идея этой модели в том, что она концептуально описывает, как данные передаются по сети, т.е. у нас есть 7 уровней, начиная от application, заканчивая физическим уровнем, когда мы работаем на уровне последовательности единичек и нулей. А самый верхний уровень application, это когда мы работаем по протоколу HTTP, HTTPS и другие. Таким образом, когда мы делаем HTTP request на какой-то сайт, то наш запрос, он с верхнего уровня опускается до самого нижнего, и в итоге последовательность нулей и единичек от нас идет к нашему получателю. Как только наша последовательность единичек и нулей, то есть наши данные или наш запрос дойдет до получателя, он обратно расшифровывает вот эту последовательность и поднимается на верхний уровень application для того, чтобы прочитать, понять и возможно обратно ответить каким-то сообщением. Ответ также будет проходить, опускаться с верхнего уровня до нижней, и как только придет вам, вы для того, чтобы прочитать эту последовательность, будете от самого нижнего уровня подниматься до самого верхнего и получите тот ответ в исходном виде, который вам был отправлен. Этой информации вам достаточно для того, чтобы понимать, как функционирует сеть, как данные проходят по сети и в целом легче понять, как работает сервис Amazon VPC.

Мы с вами переходим ко второй части нашей лекции и подробнее познакомимся с сервисом Amazon VPC. Amazon VPC расшифровывается как Amazon Virtual Private Cloud. Это сервис, который позволяет вам настроить логически изолированную сеть в облаке. У вас есть полный контроль над этой сетью, и вы можете создавать ресурсы в этой вашей сети. Более того, вы можете применять любые кастомные настройки в этой сети, а также есть несколько уровней безопасности, это Security Group и Network Access Control Lists, либо чаще вы встречаете как Network ACLs.

Здесь вы можете видеть схему, которая объясняет, как VPC может быть расположена в облаке. Когда мы говорим VPC, она присутствует только в одном AWS аккаунте. Также VPC полностью должна находиться в одном регионе. Если вам нужно использовать несколько регионов, то значит у вас будет по одному VPC на каждом из регионов. Мы с вами помним, что в рамках региона у нас может быть несколько availability zone. Таким образом, одна VPC может находиться в нескольких availability zone-ах.

VPC идентифицируется CIDR блоком, который мы ранее поговорили. Также внутри VPC мы можем делать некоторые логические группы, которые называются subnets. Subnets тоже идентифицируются CIDR блоком меньшего размера. И range, т.е. возможные значения IP-адресов subnet, они должны входить в возможные значения VPC. Касательно subnets, они могут быть двух видов. Первый это private, второй это public. Public это те subnets, которые доступны из интернета. А private это те subnets, которые недоступны из интернета.

Пару слов хотелось бы сказать о том, в какой размерности могут быть CIDR блоки у VPC. Самый максимальный размер это 65536 уникальных IP-адресов, что равняется CIDR блоку /16. Самые маленькие это 16 IP-адресов. И в CIDR блоке она записывается как /28, т.е. 4 вида у нас могут быть flexible, т.е. меняться. Когда мы говорим про CIDR блок для VPC, очень нужно аккуратно и ответственно подойти к размеру этого CIDR блока, а также к начальному IP-адресу. Связано это с тем, что CIDR блок, который вы назначите для VPC, позже уже не может быть изменен. Таким образом, если вам нужно поменять CIDR блок, вам необходимо создать новый VPC. Если у вас в старом VPC есть какие-то ресурсы, которые работают, вам необходимо выделить время, силы, деньги, людей для того, чтобы все эти ресурсы перенести в новый VPC с новым CIDR блоком. Это может быть очень дорого и неприятно, поэтому правило такое, что вы с каким-то достаточным запасом выбираете размер VPC. Более того, всегда помните, что VPC между собой пересекаться не могут. Даже если у вас VPC находится в разных AWS аккаунтах, то такая рекомендация на будущее – создавать CIDR блоки не пересекающимися, так как есть разные сервисы в AWS, которые помогают VPC из разных аккаунтов соединять между собой. Если CIDR блоки этих VPC будут пересекаться, то вы их между собой не сможете соединить, поэтому учитывайте всегда этот момент.

Другой момент – это касательно subnet-ов. Как уже говорилось ранее, CIDR блок subnet-а должен входить в CIDR блок VPC. Размер subnet-а максимально может быть равен размеру самого VPC. Таким образом, в этом VPC будет только один subnet. Если же мы внутри VPC создаем несколько subnet-ов, нам надо убедиться в том, что CIDR блоки в рамках одного VPC также не пересекаются.

Когда мы создаем CIDR блок неважно большой или маленький, так же неважно для VPC либо для сsubnet-а в облаке AWS, AWS резервирует 5 IP-адресов для системного использования. Вы можете видеть, что самый первый IP-адрес в range CIDR блока зарезервирован под network address, второй – для internal communication, третий – для DNS resolution, четвертый – для будущего использования, и самый последний – он используется как network broadcast address. Таким образом, для CIDR блока, у которого размерность – /24, т.е. 256 уникальных адресов, фактически для вашего использования доступны минус 5, это значит 251 IP-адресов.

Давайте здесь делаем очередную активность. И вопрос следующий. Представьте у вас следующий CIDR блок: 10.0.0.0/28. Вопрос. Какое количество IP-адресов внутри этого CIDR блока доступно для вашего использования? Хорошо, я думаю, все вы справились. Давайте попробуем вместе посчитать. CIDR блок с размером /28 говорит о том, что 28 битов в 30-битном IPv4 CIDR блоке зафиксированы, значит, количество битов, которое может меняться, 32 минус 28 равно 4. Какое количество IP-адресов для 4 битов мы можем посчитать следующим образом. Для этого мы считаем 2 в степени, число которое вышло, в нашем случае 4, 2 в степени 4 равняется 16. Как только что мы сказали, 5 адресов AWS резервирует для системного использования. Таким образом, 16 минус 5 равняется 11. И ответ у нас для CIDR блока с размером /28 у нас не 16, а 11 уникальных IP-адресов, которые доступны для нашего пользования.

Когда мы говорим про публичные IP-адреса, выделяют два вида. Первый – это публичный IPv4-адрес и ElasticIP-адрес. Чем они отличаются? Когда мы создаем VPC внутри облака AWS, то все инстансы внутри этого VPC автоматически получают внутренний IP-адрес, который называется private IP address. Но параллельно с этим вы можете поставить галочку в пункте Auto assign public IP address во время создания VPC. Таким образом, помимо private IP address, все инстансы будут получать public IPv4 address, который является динамическим public IP address. Если же вам необходим статический публичный IP-адрес, в этом случае вы можете воспользоваться функционалом Elastic IP address, который эту возможность предоставляет. Вы этот IP-адрес можете привязать к ресурсу, а именно к инстанcу либо к интерфейсу внутри вашего VPC. Следует помнить, что услуга Elastic IP address также является платной. Таким образом, следует взять за правило все Elastic IP address, которые не привязаны к ресурсам, высвобождать для того, чтобы не платить впустую.

Пару слов хотелось бы сказать касательно Elastic Network Interface. Это отдельный ресурс, который вы можете привязать к инстанcу внутри вашего VPC, который может обслуживать некоторый IPv4-адрес. Помимо Elastic Network Interface, у каждого инстанcа внутри VPC есть так называемый Default Network Interface. К этому Default Network Interface привязывается Private IPv4-адрес, который выбирается из CIDR блока вашего VPC. К инстанcу вы можете привязывать определенное количество Network Interface-ов, и это количество зависит от типа инстанcа, чем она выше, тем она больше, тем большее количество Network Interface вы можете подключить к конкретному инстанcу.

Давайте подробнее поговорим, что такое Route table. Route table – это некая таблица, которая состоит из набора правил, rule или routes. Каждое правило идентифицируется местом отправки и местом назначения, или английскими терминами destination and target. Как destination у нас выступает некоторый CIDR блок, он может описывать наш subnet, а как target указывается уже некий ресурс. В любом Route table есть первый, самый основной rule. Это rule, который со всех наших subnets как target указывает local. Это нужно для того, чтобы все инстансы между нашими subnet-ами могли между собой взаимодействовать.

У VPC есть Route table по умолчанию, который называется Main Route table. Он автоматически создается и привязывается к нашему VPC. Важный момент – это то, что каждый subnet должен быть ассоциирован, т.е. привязан к одному из Route table. Но у каждого Route Table может быть несколько subnet-ов. Более того, у каждого VPC может быть не один, а несколько Route table. Таким образом вы гибко можете настроить все ваши routes между вашими subnet.

Мы завершаем вторую секцию. Давайте пройдемся по основным моментам, которые необходимо запомнить. VPC – это логически изолированная часть облака AWS, в которой мы можем создавать необходимые для нас ресурсы. VPC относится только к одному аккаунту и к одному определенному региону. Но VPC может находиться в нескольких availability зонах в рамках одного региона. VPC идентифицируется одним определенным CIDR блоком. Что такое CIDR блок мы проговорили в предыдущей секции. Каждый VPC может быть поделен на логические группы, которые называются subnet-ами. Каждый subnet также идентифицируется CIDR блоком, который должен входить в CIDR блок VPC. Также у нас есть такое понятие как Route table, это некий ресурс, который контролирует трафик между нашими subnet-ами внутри нашего VPC. У нас есть внутри каждого Route table встроенный путь Local Route, который нельзя удалить и который помогает нам предоставить возможность всем инстансам внутри разных subnet-ов общаться между собой в рамках одного VPC. Вы можете добавлять дополнительные routes, rules, или правила в ваши Route table для того, чтобы гибко настроить сетевые потоки внутри вашего VPC.

Мы с вами добрались до третьей секции. На третьей секции мы с вами поговорим про особенности настройки роутинга трафика внутри VPC. Одним из важных компонентов VPC является Internet gateway. Internet gateway используется для того, чтобы связать интернет с вашими инстансами внутри вашего VPC. Для того, чтобы ваш public subnet сделать публичным, то есть доступным из интернета, нам необходимо добавить Internet gateway, а также добавить один route в нашем Route Table, в котором как destination выступает интернет, то есть это 0.0.0.0/0, как target выступает наш Internet gateway.

Другой частый случай это когда для инстансов внутри private subnet необходимо выходить в интернет. При этом мы должны запретить доступ с интернета добираться до инстансов внутри private subnet. Для этих целей используется так называемый Network Address Translation gateway, чаще вы его будете встречать как NAT gateway. Для того, чтобы правильно настроить gateway, вам в первую очередь необходимо настроить public subnet. Как настроить public subnet мы с вами знаем. Повторюсь еще раз. Представим, у вас есть route table специально для вашего public subnet, и в нем как destination вы указываете весь интернет, как target указываете Internet gateway. Таким образом ваш subnet становится public. Далее внутри public subnet мы создаем NAT gateway. Теперь нам необходимо в route table для private subnet добавить route с интернета, то есть destination является интернет, а target является NAT gateway. После этого все инстансы внутри private subnet смогут выходить в интернет через NAT gateway. При этом NAT gateway обратно впускать уже запросы из интернета в private subnet давать не будет. Помимо NAT gateway есть еще вариант настроить NAT instance, то есть поднимается отдельный EC2 инстанс и на нем настраивается NAT gateway. Отличие в том, что если инстанс сломается, с ним что-то произойдет, то интернет у вас пропадет, и его нужно будет заменить. В случае, когда мы говорим про NAT gateway, это managed service от Amazon, поэтому все возможные поломки, недоступность обеспечивается со стороны Amazon и в случае каких-то поломок прозрачно для вас заменяется на новый рабочий ресурс. Таким образом, есть рекомендация от Amazon как best practice использовать NAT gateway вместо NAT instance.

Одним интересным подходом, который вы можете реализовать внутри AWS, является VPC sharing. Идея ее в том, что вы можете свои subnets в рамках вашего VPC расшарить для VPC из другого аккаунта. Но этот аккаунт должен быть внутри общего AWS Organizations. Есть несколько нюансов. Это то, что вы можете видеть, а также изменять все те ресурсы внутри этого shared subnet, но при этом вы не видите и не можете изменять ресурсы других аккаунтов, которые находятся в этом subnet. То же самое касается всех участников, которые совместно работают внутри этого shared subnet. Этот подход очень удачно будет применен для тех IT-инфраструктур, у которых приложения внутри этой IT-инфраструктуры сильно взаимосвязаны. Таким образом, помимо того, что упрощается связь между приложениями, вы также можете сэкономить на некоторых общих ресурсах, таких как NAT gateway, VPC Interface Endpoints и так далее.

Еще одним интересным подходом, который вы можете воспользоваться внутри AWS, является VPC peering. Это та возможность, которая позволяет вам соединять между собой разные VPC. Эти VPC могут находиться как в одном аккаунте, так и в нескольких разных аккаунтах. Для того, чтобы произвести соединение, то есть peering connection между двумя VPC, нам необходимо, во-первых, создать peering connection ресурс, как вы видите по центру pcx-id, также внести для каждого VPC route-ы в соответствующих route tables.

Если говорить более подробно, то в route table-е VPC A нам необходимо добавить route, где destination является CIDR блок VPC B, а как target указать наш peering connection ресурс, то есть мы через него доходим до нашего VPC B. А в route table-е VPC B сделать обратную route, когда как destination указывается CIDR блок VPC A, а target является peering connection ресурс.

Когда мы говорим про VPC peering, у нас есть несколько ограничений. Самое первое и самое важное, это то, что IP-адреса, то есть CIDR блоки, они не могут пересекаться. Если они пересекаются, то вы не сможете настроить peering connection. Это как раз то, что я вам говорил на предыдущих слайдах, и вам всегда нужно иметь в виду, даже если VPC не планируется соединять между собой, то на всякий случай должен быть центральный список со всеми CIDR блоками, для того чтобы создавать их такими, чтобы не было пересечений. Другой момент, это то, что VPC peering нетранзитивный. Это говорит о том, что если мы настроим связь между VPC A и VPC B, далее между VPC B и VPC C, то связь между VPC A и VPC C не появится. Если вам нужно VPC A также соединить с VPC C, то необходимо настроить отдельный peering connection. И другим моментом, который также важно помнить, это то, что между двумя VPC вы можете настроить только один peering connection.

Все это время мы говорили с вами, как соединить сети внутри облака. Если же нам необходимо подключиться к облаку с нашего локального офиса, либо с локального дата-центра, такая возможность тоже есть. Для этого существует сервис AWS Site-to-Site VPN, который позволяет нам это все настроить. Давайте пройдемся по самым основным моментам, что необходимо сделать, чтобы предоставить доступ с локального офиса на ваш VPC в облаке. Самым первым нам необходимо создать так называемый Virtual Private Network Gateway или Virtual Gateway. Вы видите по центру экрана Virtual Gateway ID. Он привязывается к нашему VPC. Следующее, нам необходимо сконфигурировать так называемый Customer Gateway. Customer Gateway не является неким ресурсом, это является AWS ресурсом, который предоставляет информацию AWS о вашем VPN-девайсе, т.е. оборудование. Третье – нам необходимо настроить, например, для Private Subnet, route table, добавить route, где destination является наш локальный офис, то есть CIDR блок нашего локального офиса, а target является vgw-id, то есть это Virtual Gateway ID. После этого нам необходимо воспользоваться сервисом AWS Site-to-Site VPN для того, чтобы соединить между собой две системы. Это основные моменты, которые необходимо проделать для того, чтобы соединить локальную инфраструктуру с инфраструктурой в облаке AWS.

Следующий похожий сервис, который позволяет нам соединить нашу инфраструктуру в облаке с локальной инфраструктурой, является сервис AWS Direct Connect. Отличие ее в том, что в случае с предыдущим сервисом AWS Site-to-Site VPN мы настраиваем VPN-подключение через интернет. Таким образом, связь у нас есть, но скорость передачи данных и канал сети определяется размером канала вашего интернета, если у вас слабое интернет-подключение, это говорит о том, что у вас подключение к облаку через сервис AWS Site-to-Site VPN будет также небольшим. В случае, если вам необходимо передавать большой объем данных, либо передавать это все быстрее, то есть вариант в обход интернета подключиться к так называемым DX-locations, проще говоря, это глобальная инфраструктура AWS, и мы напрямую подключаемся к ней. В этом случае нет необходимости выходить в интернет, а напрямую подключать нашу локальную инфраструктуру с инфраструктурой в облаке.

Часто бывает такое, что вам необходимо вызвать некий AWS сервис из VPC, т.е. не все сервисы AWS могут работать внутри VPC, но тем не менее есть решение, которое позволяет нам, не покидая внутренней сети AWS, не выходя в интернет, добираться от VPC до необходимого нам сервиса AWS. Это так называемые VPC endpoints, выделя.т два вида, это Interface VPC endpoint, либо Interface endpoint, который поддерживает AWS Private Link. Другой вариант – это Gateway endpoint. В зависимости от того, какой сервис вам нужен, вы можете посмотреть в документации, какой из вариантов endpoint-а поддерживается и его соответственно использовать.

Давайте посмотрим на примере сервиса Amazon S3, что необходимо настроить для того, чтобы из VPC иметь возможность работать с сервисом Amazon S3. Для этого нам необходимо создать ресурс VPC Endpoint, а также в route table для нашего subnet прописать дополнительный route, где destination является Amazon S3 ID, а target является наш VPC endpoint ресурс.

Чуть ранее мы с вами поговорили про VPC peering, это когда нам необходимо два отдельных VPC, даже если они находятся в разных AWS аккаунтах, соединить между собой. Одним из ограничений этого подхода было то, что нет транзитивности, т.е. вы первый VPC соединяете со вторым, второй соединяете с третьим, и это не говорит о том, что появляется связь от первого до третьего. Если вам необходимо было первый VPC соединить с третьим, то необходимо настраивать отдельный VPC peering connection для того, чтобы связь появилась.

Теперь представьте такую ситуацию, что у вас слева на слайде несколько разных VPC, а также есть VPN соединения, и все они между собой взаимосвязаны. Для того, чтобы все VPC связать между собой, опять же это зависит от вашей бизнес потребности, то вам необходимо создать вот такое количество VPC peering. Поддерживать это может быть очень неудобно, и создается буквально большое количество VPC peering с каждым последующим разом. Для того, чтобы решить подобную проблему в существующих IT-инфраструктурах, предлагается использовать сервис AWS Transit Gateway. Идея ее в том, что она работает по принципу Hub and Spoke Model, то есть AWS Transit Gateway выступает как центральным хабом, и все VPC, которые соединяются к этому Transit Gateway, автоматически получают доступ ко всем VPC, которые уже подключены к AWS Transit Gateway. Таким образом, вы видите с правой стороны, как можно упростить вид вашей инфраструктуры, используя AWS Transit Gateway.

Мы с вами добрались до конца третьей секции. Давайте вкратце пройдемся о том, что мы прошли. Первый VPC компонент – это Internet Gateway, который помогает соединить интернет с нашими ресурсами в public subnet-е. Далее есть компонент NAT Gateway либо NAT Instance, который позволяет уже инстанcам из Private Subnet добираться до интернета, при этом запрещать доступ из интернета, подключаться до инстанcов внутри private subnet. Далее мы поговорили про VPC Endpoint. Идея в том, что не все сервисы бывают внутри VPC, и в зависимости от ваших бизнес требований, если вам нужно использовать этот сервис AWS, есть возможность, не выходя в интернет, а внутри сети AWS, локально, т.е. подключиться от этого сервиса к вашему VPC, для того чтобы это было безопасно, быстрее, и в целом вы получили доступ из VPC к необходимому вам сервису. Следующее – это VPC peering, это когда вам необходимо попарно соединить между собой ваши VPC, при этом VPC могут находиться в разных AWS аккаунтах. Следующее – это VPC sharing. Идея в том, что вы можете ваши subnets расшарить с другими VPC, с VPC с других AWS аккаунтов. Основной момент, который надо учитывать, это чтобы все аккаунты были в одном AWS Оrganizations. Далее мы поговорили с вами про AWS Site-to-Site VPN. Идея в том, что вы можете вашу облачную инфраструктуру соединить с локальной инфраструктурой через интернет, прокинув специальное VPN-подключение. Есть и плюсы и минусы такого подхода. Другой вариант – это AWS Direct Connect. Идея в том, что вы не через интернет подключаетесь к облаку, а подключаетесь к глобальной инфраструктуре AWS. Таким образом выходить в интернет не нужно,и при этом вы получаете больше канал. И этот вариант лучше, если вам необходимо передавать большой объем информации с вашего локального офиса в облако либо обратно. И последнее то, что мы поговорили – это AWS Transit Gateway. Идея в том, что если ваши VPC должны быть тесно связаны между собой, и когда вы используете VPC peering, то создаете большое количество этих VPC peering-ов, то как решением может быть AWS Transit Gateway, когда у вас есть центральный ресурс Transit Gateway, к которому подключаются все ваши VPC. Таким образом все подключенные VPC получают доступ ко всем другим VPC, которые уже подключены к этому Transit Gateway.

Мы добрались до четвертой секции. И на этой секции мы поговорим про безопасность VPC. Давайте начнем. Начнем мы с Security Groups. Security Groups – это некий виртуальный firewall для ваших инстансов, который позволяет контролировать входящий и исходящий трафик. Важно запомнить, что Security Groups работают на уровне инстанса. Таким образом вы можете в рамках одного subnet-а для каждого инстанса внутри него настроить специальный набор Security Groups. Когда мы говорим про Security Groups, следует понимать это как набор правил, которые контролируют входящий и исходящий трафик. В только что созданной Security Groups у вас будет отсутствовать Inbound Rules, то есть входящий трафик. Таким образом сторонний хост не сможет обратиться и добраться до инстанса с новой созданной Security Group. Если мы говорим про Outbound Rules, то есть исходящий трафик, то по умолчанию он полностью разрешен и ничем не ограничен. Если вам необходимо ограничить исходящий трафик, то вы можете удалить rule, то есть правило, которое создается по умолчанию, и задать свои правила, которые в каком-то объеме ограничены.

Здесь также следует отметить, что Security Groups являются stateful, т.е. это означает, что когда вы с вашего инстанса делаете запрос к другому ресурсу, если у вас есть Outbound traffic, то запрос разрешается. Далее, когда вы получаете ответ на ваш запрос, вне зависимости от того, Inbound Rule разрешен или запрещен, вы его получаете, т.е. запоминается состояние. Обратное тоже верно, т.е. если разрешен Inbound Rule от внешнего ресурса, то запоминается состояние, что он пришел со стороны, так как он уже был разрешен. Мы запоминаем это состояние, и уже вне зависимости от того, разрешен ли исходящий трафик, он разрешается в любом случае и передает ответ обратно.

Давайте рассмотрим пример Security Groups. Вы здесь видите, есть настроенные rules для исходящего и входящего трафика. Если посмотреть на исходящие rules, то мы видим, что разрешены порты 80 и 443, т.е. HTTP и HTTPS доступ со всего интернета, т.е. любой IPv4 адрес с интернета может подключаться к нашему инстансу. Другое, третье, это то, что как в source указан CIDR блок нашей сети, таким образом мы с локальной сети по порту 22, т.е. SSH, можем подключаться к инстансу, к которой привязана эта Security Group. Если мы говорим про исходящий трафик, то мы можем как destination указать Security Group ID наших баз данных и выдать разрешение на порт 1433, таким образом, инстанс этой Security Group может подключаться к базам данных по порту 1433. Это порт по умолчанию для Microsoft SQL Server баз данных.

Следующим компонентом, который позволяет обеспечить безопасность VPC, является Network Access Control Lists. Чаще вы ее будете встречать как Network ACLs. Network ACLs выступает как firewall и контролирует входящий и исходящий трафик на уровне subnet-ов. Оно является опциональным, поэтому использовать или нет это на ваше усмотрение. Amazon со своей стороны рекомендует ее использовать и продублировать входящие и исходящие rules для всех ваших VPC.

Когда мы говорим про Network ACLs, каждый subnet должен быть привязан к одному из Network ACLs. Если вы явно не указываете связь, то subnet-ы привязываются к дефолтовому Network ACLs. Здесь важный момент, это то, что каждый subnet может быть соединен только с одним Network ACL, тогда как несколько subnet-ов могут быть соединены с одним Network ACL. Когда вы один subnet перепривязываете к другому Network ACL, то предыдущая привязка удаляется.

Network ACLs выглядит следующим образом. Это пример Network ACLs, созданного по умолчанию. По умолчанию разрешается весь входящий и исходящий IPv4 traffic. Если это применимо, то IPv6 traffic также разрешается. Здесь следует отметить, что Network ACLs является stateless, т.е. не сохраняется состояние. Таким образом, когда вы делаете, представим, исходящий запрос на некий ресурс, если у вас разрешен outbound traffic, то запрос проходит. Далее, так как Network ACLs является stateless, то при возвращении ответа он также проверяет, если через inbound traffic он разрешен, тогда вы получаете ответ. Если же в inbound traffic прописано, что он не пропускает этот входящий трафик, то вы не получите ответа, даже если он дойдет до вашего получателя. Обратное тоже верно, т.е. для того, чтобы внешний ресурс получил от вас ответ, то должны быть разрешены и outbound, и inbound traffic. Иначе вы получите запрос от ресурса, но ответить этому ресурсу уже не сможете, так как не пропустит Network ACLs и нет соответствующего rule внутри outbound traffic.

Здесь вы можете видеть пример Network ACLs, уже заполненного. Когда мы создаем кастомный Network ACLs, то весь трафик входящий и исходящий, он запрещен. Поэтому по необходимости вам нужно добавить rules, т.е. правила, которые разрешают входящий либо исходящий трафик. Здесь вы можете видеть, что входящий трафик и исходящий трафик разрешен для HTTPS и SSH соединения. При этом по HTTPS могут подключаться любой IPv4-адрес, т.е. с интернета есть доступ, а с SSH могут подключаться только IP-адреса с CIDR блока 192.0.2.0/24, т.е. здесь вы можете указывать вашу локальную сеть.

Возможно, сейчас вам немного непонятно, чем же все-таки отличается Security groups от Network ACLs, поэтому давайте сравним в табличном виде, чем же они все-таки отличаются. Когда мы говорим про Security groups, то Security groups работают на уровне инстанса, тогда как Network ACLs работают на уровне subnet, т.е. привязывается к subnet. Когда мы говорим про правила, Security groups поддерживают только правила, которые что-то разрешают, т.е. разрешают трафик. Когда мы говорим про Network ACLs, там присутствуют не только правила, которые что-то разрешают, а также можно настроить правила, которые запрещают трафик. Когда мы говорим про Security groups, то оно является stateful, т.е. запоминает состояние. И это значит, что трафик, который смог войти, он обязательно выйдет, несмотря на Outbound rules. И обратное тоже верно. Если у вас есть Outbound traffic, который прошел, то вне зависимости от того, какие Inbound rules у вас прописаны, оно вернется обратно. Когда мы говорим про Network ACLs, она является stateless, т.е. она не запоминает состояние, и для того, чтобы запрос прошел успешно через Network ACLs, оно должно быть разрешено как в Inbound rules, так и в Outbound rules, иначе в какой-то момент она не пройдет. Четвертый момент, который отличается, это то, что Security groups для того, чтобы принять решение, разрешать трафик или нет, просматривает все имеющиеся rules, после чего принимает решение. А когда мы говорим про Network ACLs, он по приоритету сверху вниз просматривает все rules. В тот момент, когда мы находим правила, которые разрешают трафик, все оставшиеся правила не рассматриваются, и трафик разрешается. Мы заканчиваем четвертую секцию. Здесь мы с вами рассмотрели более подробно, что же такое Security groups и что такое Network ACLs. Оба эти компонента помогают нам защитить наши VPC.

Мы с вами добрались до пятой секции. И здесь мы поговорим про сервис Amazon Route 53. Это сервис, который предоставляет нам DNS сервис. Самыми простыми словами это значит то, что в тот момент, когда пользователи открывают сайт www.example.com, то этот сервис перенаправляет трафик на соответствующие IP-адреса, а именно IP-адреса нашей инфраструктуры, на наши инстансы. Amazon Route 53 поддерживает большой выбор сервисов AWS. Это может быть не только Amazon EC2 инстанcы, также это может быть Amazon S3 buckets, Elastic Load Balancing и другие сервисы. Внутри Amazon Route 53 можно гибко настроить трафик для того, чтобы реагировать на состояние таргет IP-адресов, т.е. IP-адресов, куда перенаправляет этот сервис и соответственно действовать, т.е. таким образом можно настроить перенаправление трафика в случае возникновения проблем у принимающей стороны. Также следует отметить, что черезтAmazon Route 53 мы можем покупать домены. Какие домены верхнего уровня доступны, вы можете посмотреть и ознакомиться на странице сервиса Amazon Route 53.

На этом слайде вы можете видеть, как обрабатывается запрос пользователя, когда он открывает определенный сайт. Когда пользователь вводит, например, www.example.com, название сайта, то этот запрос уходит на DNS resolver. Это такие сервера, которые сохраняют связки доменное имя и какой IP-адрес его обслуживает, либо информации о том, где, на каком источнике можно получить этот IP-адрес. В случае, если у DNS resolver этой информации нет, но он знает, куда пойти, он идет в сервис Amazon Route 53, который смотрит в своих настройках и соответственно возвращает IP-адрес этому DNS resolver. DNS resolver дальше возвращает этот IP-адрес пользователю, и это происходит прозрачно для пользователя. Когда он вводит в строке браузера, он видит уже открытый сайт. Но здесь есть два шага. Первый шаг мы с вами увидели. И второй шаг, как только браузер получает IP-адрес, он делает уже этот запрос на соответствующий IP-адрес и получает оттуда ответ, отображает пользователю. В строке ввода, в строке поиска у пользователя все так же останется то доменное имя, которое он ввел вначале.

Amazon Route 53 поддерживает несколько различных policy, которые помогают нам направлять и гибко настраивать наш трафик. Самый первый это Simple routing. Идея ее в том, что вы в настройках Amazon Route 53 указываете один или несколько IP-адресов, которые обслуживают ваше доменное имя. В тот момент, когда приходит запрос, то этот сервис в рандомном порядке, в случайном порядке отдаёт один из IP-адресов, и уже браузер на стороне пользователя переходит на этот IP-адрес, чтобы получить ответ на запрос пользователя.

Следующий более продвинутый policy – это Weighted round robin routing. Идея ее в том, что вы можете для списка ваших IP-адресов, серверов, которые обслуживают трафик, задать некоторые веса. И представим, у вас есть два сервера, один мощный, другой менее мощный. В этом случае вы можете, например, 75% трафика направить на большой сервер, а оставшиеся 25% на маленький.

Третий вариант – это Latency routing. Она более сложная по сравнению с предыдущими двумя. Идея ее в том, что она измеряет скорость ответа от определенных IP-адресов и выбирает тот, который быстрее всего отвечает. Давайте я приведу пример. Представим, наша инфраструктура развернута в двух AWS регионах. И по происшествию некоторого времени и сбора необходимой информации, Amazon Route 53 будет выдавать для пользователей тот регион, который для этого пользователя будет работать быстрее.

Следующее – это Geolocation routing. Идея в том, что мы можем для пользователей с определенного региона направлять на указанные нами сервера либо IP-адреса, которые обслуживают этот трафик. Это используется обычно для того, чтобы направлять трафик на соответствующий сайт с нужным языком. Представим, что мы как пользователь подключаемся из страны Центральной Азии, мы знаем русский язык, и в этом случае вы можете настроить сайт таким образом, что он направляет на версию сайта на русском языке. В случае, если подключается пользователь с англоязычной страны, в этом случае, соответственно, пользователя направят на те сервера, которые обслуживают англоязычный трафик.

Следующая policy – это Geoproximity routing. Идея в том, что в отличие от Geolocation routing, когда мы смотрим на локацию пользователей, в Geoproximity routing мы смотрим на расположение нашей инфраструктуры. И, соответственно, выдаем ответ.

Следующий – это Failover routing. Идея в том, что мы передаем IP-адреса active и passive. И в момент, когда с active IP-адресом или ресурсом, который обрабатывает трафик, все хорошо, весь трафик направляется туда. В случае возникновения некоторых проблем, то трафик автоматически перенаправляется на резервный IP-адрес для того, чтобы обработать и не пропустить каждый вопрос. В тот момент, когда active становится снова доступным, то срабатывает переключатель и весь трафик, основной трафик будет перенаправляться обратно на active IP-адреса.

Самый последний – это Multivalue answer routing. Идея ее в том, что во всех предыдущих случаях мы возвращали один конкретный IP-адрес пользователю. В случае с Multivalue answer routing мы передаем несколько IP-адресов, а далее уже браузер в зависимости от настроек либо алгоритма работы определяет, на какой IP-адрес нужно идти.

Здесь вы можете видеть пример использования сервиса Amazon Route 53 и какие выгоды она дает. Например, наша инфраструктура развернута в нескольких регионах. И в случае подключения пользователя с некоторого региона, который ближе к первому региону, а на стороне Amazon Route 53 у нас настроен Latency-based routing, в этом случае пользователь будет направлен на тот регион, который ему ближе. И в целом получит ответ намного быстрее, чем если бы запрос ушел на регион, находящийся немного дальше.

На этом слайде вы видите пример страницы AWS, где происходит настройка Failover routing policy. И пример этот дан для того, чтобы показать, что есть возможности гибко настроить правила направления нашего трафика. Если мы говорим про конкретный пример, то мы видим, что есть у нас Failure threshold 3, то есть если последовательно 3 запроса были неудачными, то у нас происходит переключение и весь трафик уходит на наш резервный IP-адрес, который обслуживает этот трафик. И дальше каждый либо 30 секунд, либо 10 секунд, Request interval, мы проверяем, можем ли мы обратно вернуться на наш active IP-адрес, который является основным.

На этом слайде приведен еще один пример настройки Failover routing policy. Представим, у нас есть инфраструктура, которая обслуживает основной трафик и в Amazon Route 53 эта инфраструктура является основным, то есть Active Route. И соответственно запрос доходит до наших Amazon EC2 инстансов, которые обращаются к базе данных для того, чтобы вернуть какой-то динамический ответ. А на стороне Amazon Route 53 настроена Failover, в случае, если инстансы становятся недоступными, то у нас срабатывает Failover policy, и мы переключаемся на Secondary, то есть Passive IP-адреса. В этом случае мы можем направить трафик на наш статический веб-сайт, который хостится на сервисе Amazon S3. Таким образом мы не сможем обработать всех пользователей как положено, но по крайней мере можем направить на соответствующий веб-сайт, в котором они могут получить более-менее понятный ответ, и вы не потеряете ваших клиентов. В тот момент, когда Amazon Route 53 получит ответ, периодически проверяя, что наши Amazon EC2 инстансы успешно могут принимать запросы, то происходит обратное переключение, и весь трафик уже идет на наши Amazon EC2инстансы.

Мы завершаем пятую секцию. Давайте вкратце остановимся на том, что мы прошли. Первое, это то, что Amazon Route 53 – это высокодоступный и масштабируемый сервис, который предоставляет нам DNS. Простыми словами, это тот сервис, который направляет наш трафик от нашего доменного имени к нашим инстансам, которые обрабатывают этот запрос. Через сервис Amazon Route 53 мы также можем покупать доменные имена. Какие верхнеуровневые домены доступны для нас, мы можем посмотреть на официальной странице этого сервиса. Далее, что важно отметить и что может прийти, и скорее всего придет на реальном экзамене AWS, это какие Routing policy существуют в Amazon Route 53. На самом деле, эти вопросы являются одними из самых простых. Вам достаточно понять, чем является каждая Routing policy, и этого будет достаточно, чтобы правильно ответить на вопрос. С подобными вопросами вы встретитесь на наших Practice test-ах и сможете на них попрактиковаться. После этого мы посмотрели несколько примеров использования сервиса Amazon Route 53 с инфраструктурой в AWS.

Мы добрались до самой последней секции в рамках нашей лекции и поговорим про сервис Amazon CloudFront. Это один из базовых сервисов, который используется практически во всех IT-инфраструктурах, которые так или иначе обрабатывают интернет-трафик. Этот сервис является Content Delivery Network, то есть CDN. Простыми словами, это тот сервис, который переносит все ваши медиафайлы ближе к вашим пользователям, так чтобы это работало быстрее, это было дешевле и в целом безопаснее для вас.

Давайте посмотрим на примере, почему важно и чем помогает нам CDN для улучшения пользовательского опыта. Когда пользователь делает запрос, он может проходить через несколько узлов для того, чтобы добраться до оригинального сервера и запросить ваши некоторые данные, медиаданные с вашего сервера. То расстояние от пользователя до наших серверов может быть достаточно большим и оно отражается на скорости работы вашего приложения. Это ухудшает пользовательский опыт. Поэтому есть такой подход как CDN, Content Delivery Network. Идея в том, что в некоторых узлах ваши медиаданные либо любые другие данные кэшируются. Таким образом пользователь не доходя до оригинальных, изначальных источников серверов может раньше получить необходимый контент и это все ускорит время обработки каждого запроса каждого пользователя. Таким образом, Content Delivery Network, то есть CDN является для нас решением. Content Delivery Network это понятие в области IT, не привязанное к Amazon. Поэтому это довольно таки давно существующая технология, подход. Идея в том, что это сеть связанных между собой серверов, которые кэшируют некоторый контент. Как контент может выступать HTML страницы, CSS страницы, JavaScript файлы, картинки, видео, аудиодорожки и т.д., т.е. любые медиафайлы. Более того, продвинутые SDN могут обрабатывать и кэшировать динамический контент. Таким образом, нет необходимости пользователям ждать, когда запрос от него дойдет до оригинального сервиса. Фактически где-то посередине в кэше сохранен или есть готовый ответ для этого пользователя, который он получает. И таким образом улучшается пользовательский опыт. Если мы говорим про AWS, тогда есть специальный сервис, который является CDN, т.е. Content Delivery Network называется Amazon CloudFront.

Мы с вами проговорили ранее, что такое глобальная инфраструктура AWS. Это когда у нас есть AWS регионы, в каждом регионе есть availability zone, в каждой availability zone есть дата-центры, и проговорили, что такое дата-центры. Также есть параллельная инфраструктура AWS, которая относится к CloudFront называется Amazon CloudFront Infrastructure. Состоит она из двух компонентов. Это Edge locations, а также Regional edge caches. Чем отличается? Edge locations – это те сервера, которые находятся наиболее близко к конечным пользователям. Таким образом, в нем сохраняется контент, который самый популярный, самый необходимый, который запрашивается чаще всего. Чуть дальше находится Regional edge cache, в котором сохраняется информация уже чуть большего региона. И в нем есть данные, которые также популярные, но менее популярные, чем те данные, которые находятся на Edge locations. Также это могут быть данные, которые по тем или иным причинам не поместились в Edge locations. Они передаются и сохраняются в кэше Regional Edge cache. После Regional Edge cache у нас уже самое дальнее расстояние – это напрямую обратиться к оригинальным источникам. Таким образом, повторюсь, у нас есть три шага, три узла. Самый дальний – это напрямую обращаться к оригинальным серверам. Чуть быстрее, если мы получаем контент из Regional Edge cache. И самый быстрый контент мы получаем из Edge locations.

Давайте пройдемся по основным преимуществам использования сервиса Amazon CloudFront. Первое – это то, что этот сервис быстрый, глобальный. Далее – это то, что мы можем настроить безопасность на наших конечных кэшируемых серверах. Мы можем дополнительно программировать поведение сервиса Amazon CloudFront, используя AWS Lambda Edge. Этот сервис является CDN, который максимально интегрирован с сервисами AWS. А также это решение, которое намного выгоднее и быстрее, чем если бы мы обслуживали наших интернет-пользователей напрямую с наших серверов.

Amazon CloudFront является платным ресурсом, но также, как и любой другой сервис AWS, мы оплачиваем только за то, что мы использовали. В рамках сервиса Amazon CloudFront мы оплачиваем за объем данных, которые мы передали в кэш-сервера. Также мы оплачиваем за количество запросов HTTP и HTTPS-запросов на эти кэш-сервера. Дополнительно оплачиваются Invalidation requests. Это когда нам необходимо по запросу очистить все кэш-сервера от наших данных. Обычно это требуется, когда мы находим какую-то ошибку и быстро ее исправляем. И хотим, чтобы пользователи получали обновленную версию приложения либо данных. В этом случае мы делаем Invalidation Requests. Другой вариант – это когда у нас большое обновление, и мы хотим, чтобы вот это большое обновление быстрее дошло до наших пользователей. В этом случае нам необходимо очистить кэш, и все пользователи, которые начинают запрашивать, первый пользователь сделает запрос через CloudFront, далее CloudFront, так как этих данных нет, сделает запрос на оригинальные сервера и получит эти данные, после чего закэширует. А все последующие пользователи, которые будут запрашивать эти же данные, уже будут получать не с оригинальных серверов, а быстрее уже с наших Edge серверов.

И еще один момент – это то, что есть возможность настроить так называемый Dedicated IP custom SSL. Это та опция, которая не часто используется пользователями AWS, но вам достаточно запомнить, что такая опция есть. Более подробно вы можете посмотреть на официальной документации сервиса Amazon CloudFront.

Мы закончили шестую последнюю секцию в рамках нашей лекции. Мы познакомились подробно с сервисом Amazon CloudFront, а также узнали, что же такое CDN, для чего оно используется, и также узнали, что есть на стороне AWS сервис, который предоставляет нам CDN. Из преимуществ использования CloudFront – это то, что это сервис глобальный, можно гибко настроить его, программировать его поведение, также есть возможность сделать необходимые настройки с точки зрения безопасности. Это тот сервис, который является сервисом AWS, соответственно, достаточно хорошо проинтегрирован с другими сервисами и является достаточно экономичным решением в случае, когда вы передаете большой объем информации до ваших пользователей ваших приложений.

На этом мы завершаем наше лекционное занятие. Давайте пройдемся по самым основным моментам, которые мы сегодня прошли. Это то, что мы рассмотрели основы сетей, не привязанные к AWS, но это та информация, которая поможет легче понять сервисы, связанные с Networking внутри AWS. Далее мы познакомились с сервисом Amazon VPC, рассмотрели ее дополнительные компоненты, в том числе компоненты, которые помогают обеспечить безопасность, это Security groups и NetworkACLs. После чего мы познакомились с сервисами Amazon CloudFront и Amazon Route 53, рассмотрели примеры использования этих сервисов и как эти сервисы помогают нам лучше построить нашу инфраструктуру в облаке.

Здесь вы можете видеть ссылки, которые могут вам понадобиться, если вы хотите дополнительно ознакомиться со всем тем, что мы прошли сегодня.

На этом мы завершаем наше лекционное занятие. Я надеюсь, вы узнали что-то новое сегодня для себя. И увидимся с вами на следующих наших активностях.