Добрый день, уважаемые студенты! Я рад вас всех видеть на очередной лекции. Тема сегодняшней лекции это – Compute, т.е. вычисления в облаке. Эта тема достаточно важная и хотелось бы ее максимально подробно с вами разобрать. Поэтому мы эту тему поделили на две недели и сегодня у нас первая часть. Итак, давайте начнем.

Сегодняшняя лекция поделена на три части. В первой части мы с вами поговорим про существующие сервисы AWS, которые предоставляют нам вычислительные ресурсы. Далее вторая и третья часть, она посвящена сервису Amazon EC2, т.е. AmazonElastic Compute Cloud. Сервис Amazon EC2 является одним из самых основных и важных для понимания с точки зрения подготовки к экзамену AWS, а также к вашей будущей работе, так как очень большая вероятность того, что вы будете использовать этот сервис при построении своей IT-инфраструктуры в облаке.

Первая секция. Обзор существующих сервисов вычислений. На этом слайде вы видите основной список сервисов, предоставляющих нам вычислительные ресурсы, и выделены те сервисы, которые мы более-менее, что-то более подробно, что-то менее подробно, насмотрим на нашем курсе. Самый первый и основной – это Amazon EC2, это тот сервис, который предоставляет нам виртуальные машины. Далее, есть так называемый Amazon EC2 Auto Scaling, это тот сервис, который помогает настроить автоскейлинг для наших Amazon EC2 инстансов. Следующие три – это Amazon Elastic Container Registry, далее Amazon Elastic Container Service, и третий – это Amazon Elastic Kubernetes Service, т.е. аббревиатуры Amazon ECR, Amazon ECS и Amazon EKS. Чем они между собой отличаются? Первый Amazon ECR – это хранилище Docker images, т.е. образов. Далее Amazon ECS – это сервис, который нам помогает работать с Docker-образами. Следующий – это Amazon EKS, который позволяет нам работать с Kubernetes. Kubernetes – это абстрактное приложение поверх для более удобной продвинутой оркестрации Docker images, т.е. если вы используете Docker и в какой-то момент количество этих образов, с которыми вам приходится работать растет или достаточно большое, то обычно переходят на использование Kubernetes, т.е. K8S. И также связанный сервис – это AWS Fargate, тоже сервис, который работает с контейнерами и который является managed сервисом. Идея в том, что большую часть административной работы AWS берет на себя, при этом основные моменты, определенные входные данные для работы с контейнерами предоставляем мы как владельцы IT-инфраструктуры в облаке.

Следующие сервисы – это AWS Elastic Beanstalk и Amazon LightSail. Они в каком-то смысле также похожи, я специально рассказываю сервисы в сгруппированном виде для того, чтобы у вас структурированнее это запомнилось. AWS Elastic Beanstalk – идея в том, что вы загружаете в этот сервис ваш код, код веб-приложения, и этот сервис за вас поднимает инфраструктуру в облаке AWS, таким образом забирает на себя большую часть административной работы. То же самое относится к Amazon LightSail, в этом случае мы говорим, что это сервис, который помогает нам строить приложения либо веб-сайты.

Следующий сервис – это AWS Lambda, про него мы уже несколько раз проговаривали. Теперь на этих лекциях разберем более подробно. Это Serverless Compute Service, идея в том, что вы пишете код, все остальное с точки зрения администрирования и обслуживания этого кода происходит на стороне AWS. Идеальный сервис для разработчиков, так как максимально уменьшает все то, что связано с администрированием инфраструктуры.

Также хотелось бы отметить AWS Batch, это тот сервис, который помогает нам проводить batch jobs, т.е. некие работы в больших объемах или группами. И еще один сервис, который также хотелось бы отметить, это AWS Outposts, это тот сервис, который позволяет нам управлять локальной инфраструктурой через интерфейс AWS. Таким образом, мы взаимодействуем как будто с AWS, но под капотом используются вычислительные ресурсы, которые у нас в дата-центре.

Мы можем поделить определенную часть сервисов вычисления на четыре группы. Первая группа – это сервисы по модели IaaS, infrastructure as a service, в нашем случае это Amazon EC2. Атомарной единицей в рамках этого сервиса, т.е. ресурсом, с которым мы работаем, является виртуальная машина. Так как этот сервис предоставляется по модели infrastructure as a service, у нас абсолютно полный контроль над этими виртуальными машинами. Мы можем устанавливать любую операционную систему, любые приложения. Единственный момент, нужны специалисты, которые все это проделывают. Если есть специалисты и есть определенные уникальные требования к IT-инфраструктуре, то Amazon EC2 это наш выбор.

Следующий вариант – это Serverless ресурсы, яркий представитель AWS Lambda. Когда мы пишем код, этот код загружаем в облако и все то, что связано с администрированием, обслуживанием этого кода, оно происходит на стороне AWS. Это идеальный выбор в случае, когда есть недостаток специалистов, которые будут обслуживать вашу инфраструктуру.

Третья группа – это сервисы, которые связаны с контейнерами. Мы чуть подробнее далее поговорим, что такое контейнеры. Здесь есть четыре сервиса, которые работают с контейнерами. Это Amazon ECS, Amazon EKS, Amazon ECR и AWS Fargate.

Четвертая группа – это сервис, который предоставляется по модели Platform as a Service, в минимальной единице, атомарной единице, ресурсом в этом случае является один web application, и называется сервис AWS Elastic Beanstalk. Идея в том, что мы пишем код, этот код загружаем в облако, дальше AWS Elastic Beanstalk самостоятельно разворачивает необходимую инфраструктуру. Естественно, мы передаем дополнительные параметры, которые прописывают некоторые ограничения, условия либо мощность этой инфраструктуры, но всю ту работу по поднятию и поддержке осуществляет этот сервис AWS Elastic Beanstalk, так как оно предоставляется по модели Platform as a Service.

Вы видите, что даже когда мы говорим про вычислительные ресурсы, то AWS предоставляет несколько вариантов сервисов, которые в зависимости от вашей бизнес-задачи могут вам идеально подойти. Здесь единственный момент, который я бы хотел отметить, это то, что компания всегда находится в различных стадиях развития. Таким образом, в начале представим, что вы какой-то маленький стартап, у вас небольшие нагрузки. В этом случае, возможно, вам идеально подойдут сервисы, которые serverless, например, AWS Lambda. Когда у вас нет штата, у вас нет штатного системного администратора, который этим всем занимается, более того, у вас нет больших нагрузок или постоянных нагрузок на ваш IT-продукт, в этом случае достаточно использовать AWS Lambda. Когда у вас растут нагрузки, у вас растет штат, далее у вас появляются некоторые уникальные требования к IT-инфраструктуре, то вам нужно больше контроля над вашими же IT-ресурсами в облаке. В этом случае вы постепенно будете перебираться от моделей, когда вам предоставляется сервис, когда вы лишь малую часть выполняете, а большую часть делает AWS, к сервисам, которые дают вам полный контроль над IT-ресурсами, самый максимальный случай, который возможен, это IaaS и, в нашем случае, Amazon EC2.

Мы с вами добрались до второй части нашей сессии, с этого момента мы начнем подробное знакомство с сервисом Amazon EC2. Amazon EC2 расшифруется как Amazon Elastic Compute Cloud. Это сервис, который предоставляет нам виртуальные машины. Эта виртуальная машина ничем не отличается от выделенного сервера, который мы заказываем от локального hosting провайдера, либо который мы настраиваем и заказываем на локальном дата-центре. Отличие лишь в том, что предоставляются все преимущества облачных технологий по сравнению с локальным развёртыванием. А именно это то, что вы можете за минуты запросить необходимое количество вычислительных мощностей, в любой момент, когда вам эти вычислительные мощности не нужны, вы можете их отозвать и вернуть обратно и оплата будет производиться только за то время и за то количество инстансов, типов инстансов, которые вы использовали. Уже вы сразу можете видеть, что по сравнению с локальной инфраструктурой и облачной инфраструктурой, абсолютно такой же с использованием Amazon EC2 сервиса, мы уже получаем некоторые выгоды, для нас она выходит выгодной, потому как большинство IT-проектов, нагрузка на IT-инфраструктуру динамическая. Например, если это какой-то веб-сайт по продаже одежды, то обычно люди занимаются покупками либо в выходные, либо в так называемые prime hours, т.е. часы, когда идет пиковая нагрузка. Это время либо утром до 9, либо это обеденное время, или это время после 6-7 часов вечера до 10 часов вечера, когда люди свободны от работы и у них появляется время делать какие-то покупки. В этом случае, когда у вас есть серверы на локальном дата-центре, то для того, чтобы обслужить все нагрузки, которые приходят, максимальное время, чтобы не потерять ни одного клиента, у вас должен быть какой-то запас серверов, которые постоянно там находятся. Потому как быстро в течение минуты получить новый сервер – это нереальная задача, когда вы хоститесь локально. Обычно заказ сервера занимает несколько дней, а более реалистичный вариант – это несколько недель. Таким образом, у вас должен быть какой-то запас. Другой момент – это то, что те серверы, которые вы уже заказали, вы не можете использовать временно или быстро вернуть обратно. Поэтому, раз вы купили, вы уже понесли основные ваши расходы в большом объеме. И в случае, когда у вас нет нагрузки, например, в 4 часа утра, никто не использует ваш веб-сайт, в этом случае все ваши вычислительные мощности простаивают, фактически, вы теряете ваши деньги. Когда мы работаем с облачными сервисами, в этом случае мы можем настроить скейлинг, автоскейлинг, когда, в зависимости от нагрузки на нашу инфраструктуру, мы получаем необходимое количество Amazon EC2 инстансов. И это уже позволяет нам экономить, потому как в те часы, когда нагрузки небольшие, мы отдаем большую часть серверов обратно Amazon и за эти часы не оплачиваем и сохраняем только минимальное количество инстансов для того, чтобы обслуживать клиентов в эти непопулярные часы. В тот момент, когда у нас происходят часы пика, т.е. когда люди в большом количестве посещают ваш сайт, в этом случае, необходимое количество инстансов запрашивается у облачного провайдера и в течение нескольких минут она доступна и на него можно направлять трафик. Все происходит автоматически, так как это некий живой организм в облаке. И это позволяет, даже если у вас внезапно увеличатся нагрузки в 10 раз, то все те новые пользователи, которые к вам пришли поверх, они будут обслужены успешно, потому как вы всего-навсего запросите в 10 раз больше инстансов. Естественно, вы что-то заработаете, это будет для вас очень выгодно и в тот момент, когда пиковые часы пройдут, вы вернете обратно эти инстанcы и у вас не будет случаев, когда ваша инфраструктура простаивает. Примерами использования Amazon EC2 инстансов могут быть все те примеры, которые могут быть развернуты в локальном сервере, а именно, это может быть сервер, обслуживающий приложение, веб-сайт, базы данных, может быть сервер какой-то игры, который обслуживает игру по сети, далее, это может быть почтовый сервер, медиа-сервер, когда вы раздаете картинки, может быть аудио либо видео материалы, стримите, например, какое-то видео. И другой вариант – это файл-сервер, когда вы загружаете туда файлы и даете возможность скачивать и работать с этими файлами, как, например, Dropbox, это может быть какой-то proxy-сервер и все те примеры, которые применимы на локальный сервер, т.е. сервер, который хостится в локальном дата-центре, они также применимы для Amazon EC2 инстансов в облаке. Резюмируя, мы говорим, что Amazon EC2 – это сервис, который предоставляет нам виртуальные машины, эти виртуальные машины называются Amazon EC2 инстансами. EC2 расшифруется как Elastic Compute Cloud.

Также хотелось бы отметить, что Amazon EC2 поддерживает все популярные операционные системы Windows, Linux, а именно Windows версии 8, 12, 16, 19, Red Hat, SUSE, Ubuntu и есть такая операционная система Amazon Linux, которая была создана, поддерживается и постоянно улучшается cамим Amazon-ом называется Amazon Linux. На сегодняшний день существует две версии, т.е. Amazon Linux первая версия и Amazon Linux вторая версия. Рекомендуется переходить на следующую версию, так как за ту же цену вы получаете еще больше возможностей и преимуществ по сравнению с предыдущей версией. В целом также рекомендуется использовать операционную систему от Amazon, потому как она максимально интегрирована с AWS, а также все те новинки, которые выходят, они могут выходить только для этой операционной системы, либо если появляется некое обновление для всех операционных систем, то в первую очередь она появляется внутри Amazon Linux.

Давайте пройдемся по процессу создания Amazon EC2 инстанса. Оно состоит из 9 различных шагов. Самый первый – это выбор AMI. AMI – это Amazon Machine Instance, либо простыми словами, это слепок либо шаблон Amazon EC2 инстанса. В этот шаблон входит операционная система, все возможные настройки, установленные приложения и все то, что вы можете проделать в рамках этого Amazon EC2 инстанса. Вы как будто бы замораживаете в определенном состоянии этот Amazon EC2 инстанс со всеми необходимыми, правильно проделанными настройками и это состояние можете наложить на новый Amazon EC2 инстанс. Таким образом, вы экономите свое же время, исключаете человеческий фактор и все те новые инстансы, которые создаются, они создаются намного быстрее, нежели вам пришлось бы все эти настройки проделывать с нуля для каждого Amazon EC2 инстанса.

Существуют три источника AMI. Первый – это делать самостоятельно, второй – это AWS Marketplace. Это то место, где компания подготавливает от своего имени AMI со специфическими настройками, вы можете эти AMI там покупать. И третий вариант – это Community AMI, это то место, где специалисты со всего мира подготавливают самостоятельные AMI и бесплатно публикуют в общем доступе. Так как AWS не проверяет содержимое этих AMI, то использование AMI с Community AMI полностью под вашей личной ответственности, т.е. на ваш страх и риск.

Я со своей стороны не рекомендую использовать AMI с Community AMI, в некоторых случаях это является оправданным, если покупать AMI внутри AWS Marketplace, но самым лучшим вариантом является подготовка AMI самостоятельно.

Здесь вы можете видеть на диаграмме процесс создания нового AMI, т.е. вы можете импортировать состояние уже существующей виртуальной машины, либо можете начать с самого начала. Следующим шагом вы получаете немодифицированный инстанс, далее вы производите все необходимые настройки, апгрейды, обновления приложения, патчи безопасности и так далее, и получаете уже модифицированный инстанс, который в таком же состоянии сохраняете как новый AMI. Следует помнить, что AMI создается в рамках определенного региона, если ваша инфраструктура развернута в нескольких регионах, то вам необходимо будет скопировать AMI с одного региона на второй, для того чтобы он был доступен уже в другом регионе. Его ID будет отличаться, но содержимое будет такое же.

Следующий пункт – это выбор типа инстанса. Когда мы говорим про тип инстанса, имеется в виду мощность этого виртуального сервера. Мощность определяется количеством оперативной памяти, количеством ядер процессора, информация о возможности подключения жестких дисков и объем этих жестких дисков, а также информация по каналу передачи данных по сети. На этом слайде представлена информация по характеристикам серверов в зависимости от изменения его размера. Самый маленький размер внутри AWS это nano, дальше micro, small и т.д. Также в названии инстанса вы видите как префикс идет, буквы, в нашем случае T и цифра 3. Буква – это семейство инстансов, а цифра – это поколение. Про семейство мы подробнее поговорим на следующем слайде. Когда мы говорим про поколение, это цифра, которая с каждым годом увеличивается, на текущий момент последними поколениями являются 5 и 6.

Когда мы говорим про размеры в таблице, вы видите, что с увеличением размера инстанса логично увеличивается количество ядер процессора, а также количество оперативной памяти. Здесь вы видите информацию по существующим семействам типов инстансов. Не нужно запоминать каждую букву, достаточно запомнить самые основные, которые потенциально могут прийти на реальном экзамене AWS.

Когда мы говорим про семейство общего назначения General Purprose, черное, это семейство a, m и t. Необходимо запомнить t. Дальше идет Compute optimized, т.е. для того же размера, например, large, у compute optimized инстанcа будет большее количество ядер процессоров по сравнению с large инстанcом семейства general purpose, т.е. какой-нибудь t3 large и c3 large. Здесь необходимо запомнить, что это семейство c.

Дальше идет у нас Memory optimized, это r и x. Также есть z, но чаще всего и самым популярным является семейство r. Здесь также по сравнению с general purpose тот же, например, t3 large и тот же r3 large, в этом случае у r3 large будет как минимум в два раза больше оперативной памяти по сравнению с тем же t3 large, при этом количество ядер процессора будет таким же.

Следующие они достаточно редко используемы в зависимости от вашей бизнес-задачи, есть Accelerated computing, когда вы тренируете некоторые модели машинного обучения, искусственный интеллект в этом направлении. Есть также типы инстансов, которые позволяют вам развернуть distributed file systems, т.е. распределенные файловые системы, Hadoop и т.д. В этом случае есть семейство d, h и i.

Я вам рекомендую остановить это видео, перейти на страницу Pricing сервиса Amazon EC2. Там вы увидите, какие на текущий момент есть семейства типов инстансов, какие есть размерности, сколько они стоят, а также какое количество оперативной памяти, ядер процессора предоставляется, какие есть возможности в плане storage и networking, вы можете сравнить несколько разных семейств, чтобы получить полное представление.

Когда мы говорим про networking, когда учитывать возможности networking. В случае, если у вас передается небольшое количество данных внутри вашей инфраструктуры, будь то между Amazon EC2 инстансами, либо между вашими Amazon EC2 инстансами и базой данных, в этом случае не так критично информация по сети, она будет вам предоставлена. В случае, когда вы работаете с большим объемом информации, данные передаются по сети, в этом случае этот момент может быть для вас критичным, поэтому обратите на это внимание. И следует также учитывать, что есть специальные выделенные семейства типов инстенсов, которые оптимизированы для работы по сети, т.е. у них может быть не такое большое количество оперативной памяти, либо ядер процессора, но канал для передачи данных по сети, он предоставляется больше по сравнению с другими семействами.

Следующий пункт, третий, это – Network settings, в котором мы указываем всю информацию, где будет расположен наш Amazon EC2 инстанс. А именно мы указываем информацию, в каком VPC, в какой availability зоне и в каком subnet-e будет находиться новый Amazon EC2 инстанс. Я здесь не говорю касательно региона, потому что до того, как вы нажмете в самом начале кнопку Create Instance или Launch Instance, нужный регион уже должен быть выбран, потому как внутри Launch Wizard вы не сможете поменять регион. Это тот момент, который следует учитывать. В случае, когда вы делаете эту программу, то скорее всего команда не сработает, если вы не укажете какой-нибудь из регионов. Еще один момент, который следует учитывать, это то, что все инстансы, которые создаются в дефолтовом VPC, они получают Public IP-адрес. Мы с вами помним, что это динамический IP-адрес, но благодаря этому адресу ваш инстанс может быть доступен из интернета. В случае, когда мы говорим не дефолтовый VPC, то флажок выставлять, выдавать публичное IP-адреса будет убрано и публичные IP-адреса не выдаются. Если вам нужно выдать публичный IP-адрес, то вы должны явно это указать в настройках создания вашего VPC либо subnet.

Следующий четвертый шаг – это привязка IAM role к Amazon EC2 инстансу, и этот шаг является опциональным. Идея ее в том, что представим, что ваш инстанс, в нем крутится некоторый код, логика, которая использует и обращается к другим AWS сервисам. В этом случае ему необходимо выдать соответствующие доступы, права. AWS предоставляет удобную и безопасную возможность это сделать, а именно создается IAM role, дальше создается instance profile с этой role. В случае, когда вы делаете это все через AWS Management Console, то в момент привязки IAM role к инстансу, прозрачно для вас создается instance profile с таким же названием, как ваша IAM role.

Фактически instance profile это некий контейнер, который содержит в себе эту role, а instance profile вы можете привязать к инстансу для того, чтобы предоставить ему доступ к другим AWS сервисам. В случае, когда вы создаете role и пытаетесь привязать к Amazon EC2 инстансу программным путем, то помните, что вам необходимо также создать instance profile. На слайде проиллюстрирован пример, когда у нас есть role. Role предоставляет доступ к Amazon S3 bucket, и в тот момент, когда мы привязываем эту role к инстансу через instance profile, у инстанса, точнее у кода, который работает внутри инстанса, появляется возможность обратиться к Amazon S3 bucket-ам и как-то взаимодействовать с данными, которые там находятся. Аналогично будет происходить процесс выдачи прав к инстансу к любому другому сервису AWS.

Пятый шаг – это User data script, и этот шаг является опциональным. Идея ее в том, что вместе с созданием вашего инстанса вы можете передать некий набор команд, и эти команды запустятся в самый первый запуск вашего инстанса. Зачастую вы здесь передаете список команд, которые могут быть выполнены только после создания инстанса. Возможно, это команды, которые обновляют существующие приложения, может быть, это команды, которые привязаны к metadata вашего инстанса. Как metadata инстанса мы можем сказать, что есть private IP-адрес, public IP-адрес и другая информация.

Здесь у вас может возникнуть вопрос, почему нам нужно использовать User data, если есть уже готовый AMI, в котором мы предварительно установим все необходимые программы, запустим необходимые команды и уже все будет готово. Да, это абсолютно правильный ход мыслей, и так и нужно поступать, т.е. все то, что вы можете обернуть и сохранить в шаблоне AMI, это нужно сделать. Все то, что не получится по тем или иным причинам положить внутрь AMI, вы можете сделать в User data.

User Data доступно не только для Linux операционных систем, так же для Windows. Следует только учесть, что когда вы запускаете в Windows, набор команд должен быть совместимым либо с Windows Command Prompt, т.е. CMD, либо с Windows PowerShell. Другой немаловажный момент, это то, что все команды, которые вы запускаете, будь то Windows, будь то Linux операционная система, они запускаются под admin правами, либо root пользователем.

Следующий шаг, это шестой, Specify storage, т.е. мы здесь указываем информацию по нашим жестким дискам. Здесь мы указываем основной жесткий диск, т.е. root volume, в котором мы устанавливаем операционную систему. Так же можем указать дополнительные жесткие диски, в которых будем хранить наши данные. Если вы уже настроили ваши жесткие диски внутри AMI, то этот шаг пропускается. Когда мы говорим про жесткие диски, мы говорим про три момента: первый – это ее размер, дальше – это тип, и третий – это информация по шифрованию этих жестких дисков.

Когда мы говорим про хранилища для Amazon EC2 инстанса, то упоминают четыре возможных варианта: первые два – это Amazon Elastic Block Store, т.е. Amazon EBS, и Amazon EC2 Instance Store. Это фича сервиса Amazon EC2. Оба эти варианты являются блочным хранилищем, таким образом, что первый и второй вариант подходят для установки операционной системы и соответственно полноценного запуска нашего виртуального сервера. Есть некоторые отличия, когда-то давно, когда не было сервиса Amazon EBS, использовался вариант только Amazon EC2 Instance Store. Это те жесткие диски, которые в дата-центре физически привязаны, точнее подключены к нашей оперативной памяти и к нашему процессору, т.е. это одно целое, один сервер. Таким образом, когда что-то происходит не так, ломается, поломка и автоматически заменяется ваш инстанс, то все то, что находится в жестких дисках с этим инстансом, оно для вас становится недоступным. Так как есть такое неудобство, был создан альтернативный вариант, это сервис Amazon Elastic Block Store. Идея в том, что жесткие диски, они больше не привязаны к инстансам, т.е. к вычислительным мощностям и являются чем-то отдельно стоящим. Таким образом, вы через Amazon EBS создаете ресурсы, т.е. жесткие диски, которые привязываете к вашему инстансу. В случае, когда происходит какая-то поломка либо с процессором, либо с оперативной памятью в целом, с этим инстансом физически, то вы можете ваши жесткие диски, которые к нему не относятся, приаттачить, т.е. привязать к другому инстансу, и все будет работать, вы никакие данные не потеряете. В этом и есть отличие Amazon EBS от Amazon EC2 Instance Store.

Поэтому общая рекомендация как best practice, использовать Amazon EBS там, где это возможно. В некоторых случаях оправданно использование Amazon EC2 Instance Store, это все зависит от вашей бизнес-задачи и нужно детально это рассматривать.

Следующий вариант – это Amazon Elastic File System. Опять же, это файловая система, блочное хранилище. Отличие лишь в том, что вы не можете на нем установить операционную систему, но она отличается тем, и используются в тех случаях, когда вам нужен отдельно стоящий жесткий диск, который доступен для нескольких Amazon EC2 инстансов, и является некоторым общим жестким диском. Опять же, в зависимости от вашей бизнес-задачи вам может понадобиться хранить в одном месте некоторые данные, для того, чтобы ваши инстансы могли туда добираться и совместно с этими данными работать.

Также другой вариант для совместной работы с несколькими инстансами, это сервис уже знакомый нам, Amazon S3, т.е. Amazon Simple Storage Service. Этот сервис предоставляет не блочное, но объектное хранилище. В случае, когда мы работаем с объектами, мы также можем воспользоваться сервисом Amazon S3 для того, чтобы совместно с несколькими инстансами обращаться и работать с одними и теми же данными.

Здесь представлен наглядный пример инстанса, который использует Amazon EBS как root volume, а также инстанс второй, которую используют как root volume, нашей Ephemeral volume, т.е. instance store. Давайте пройдемся по отличию либо схожести этих двух вариантов в зависимости от действий или состояния, которое происходит с инстансом. Первое – это перезагрузка, в случае, когда мы перезагружаем инстанс, для обоих вариантов данные у нас сохраняются, ничего не теряется. Когда мы говорим про остановку, т.е. stop инстанса, мы можем инстанс, который работает с Amazon EBS остановить, остановленный инстанс мы не оплачиваем, мы оплачиваем только за наши данные, которые лежат внутри Amazon EBS, а за Amazon EC2, который является намного дороже, мы не оплачиваем. Это может быть удобно, если некоторые наши серверы простаивают и там проще их приостановить либо терминейтнуть для того, чтобы не терять деньги. Когда мы говорим про instance store, то функционал остановить инстанс недоступен, вы не можете его остановить. Таким образом, есть вариант либо перезагрузить, либо третий вариант это терминейтнуть, т.е. уничтожить сервер. Когда мы терминейтим инстанс с Amazon EBS жесткими дисками, то терминейтится сам инстанс, а жесткие диски они сохраняются и остаются для нас доступными. При удалении мы также можем указать, есть определенная опция с флажком, если мы выставляем этот флажок, то наши жесткие диски совместно с Amazon EC2 инстансом удалятся, либо мы можем не выставлять этот флажок и наши жесткие диски сохранятся. И эти жесткие диски мы после можем привязать к другому инстансу, другого типа и все данные, которые были, они в таком же состоянии останутся доступными в таком же виде.

Когда мы терминейтим инстанс, который работает с Amazon EC2 Instance Store, то данные, которые находятся в Ephemeral volume полностью удаляются, включая операционную систему. Поэтому хранить некоторые данные, которые вы не сможете восстановить, либо важные данные не рекомендуется хранить в Amazon EC2 Instance Store. Там вы можете хранить какие-то данные для кэширования, либо любые другие данные, которые временные или вы в любой момент можете восстановить.

Следующий шаг – это возможность добавить теги. Здесь хотелось бы отметить, что теги –это функционал, который доступен для любого ресурса, любого AWS сервиса, поэтому оно не привязано конкретно к Amazon EC2 инстансам. Что такое теги? Теги – это некий набор пар, ключ и значения. Приведу примеры ключей. Самый популярный – это name, и когда вы заполняете неким значением, то это значение зачастую отображается внутри AWS Management Console вместе с ID этого ресурса. Это в большинстве случаев помогает идентифицировать ресурс. Другие варианты ключей – это environment, т.е. среда. Оно может быть тестовой средой для разработчиков, либо боевой средой. Соответственно, для ключа key environment будут значения внутри value поля, либо dev, либо test, либо prod, либо любые другие. Другой вариант ключа тега это owner, т.е. владелец. И как значение может указываться либо определенный человек, специалист, либо некоторый департамент управления внутри вашей компании. Идея тегов это в том, чтобы идентифицировать однозначно этот ресурс и в случае чего получить ответы, если не получить ответы, то по крайней мере получить некоторую информацию, которая поможет найти нужных людей, нужную группу людей, нужный отдел, структуру вашей организации, которая отвечает за этот ресурс. Причины подобных вопросов или поиска людей могут быть разными. Может быть такое, что этот ресурс работает долгое время и ежемесячно потребляет очень много денег, уходит, например, тысячу долларов только на один Amazon EC2 инстанс, он большой. И люди, которые занимаются оптимизацией расходов внутри AWS, нашли этот ресурс и уже хотят получить ответы, почему он крутится, работает, не работает, нужно, не нужно и т.д. Другой вариант может быть такой, что есть некоторые сервисы, под сервисом, естественно, у нас разный набор ресурсов и что-то пошло не так, что-то сломалось, никто восстановить не может, не может найти специалиста, кому можно обратиться. И в этом случае нам могут помочь теги для того, чтобы понять, куда задавать вопросы, чтобы уточнить, этот сервис кому-то нужен, что-то серьезное сломалось, либо нет, нужно восстанавливать или не нужно терять на это время, потому что потраченное время специалисту это тоже деньги для компании.

Все теги, что ключ, что значение, они являются case sensitive, это значит, что они чувствительны к регистру. Простыми словами вы можете создать ключ Name с большой буквой, заглавной буквой, а также такой же ключ name, только все буквы с прописными, маленькими буквами. И никакой ошибки в этом случае не будет, потому как внутри в системе эти ключи воспринимаются как отдельные.

Восьмой шаг – это настройка Security group. Мы можем привязать одну либо несколько security group к нашему инстансу. Что такое security группа, мы с вами ранее проговаривали. Вкратце, это – некий виртуальный firewall, который фильтрует трафик до нашего инстанса. Мы указываем в security группе набор рулов, т.е. правил, которые разрешают либо входящий, либо исходящий трафик. Вы можете видеть на слайде пример предоставления доступа по SSH, т.е. это протокол TCP, порт 22 и как Source источник вы выбираете пункт My IP. Когда вы его выбираете, то копируется ваш текущий адрес и вставляется в соответствующее поле. После чего у вас появляется возможность по SSH подключиться ко всем инстансам, к которым привязана конкретная security группа. Для того, чтобы успешно подключиться к инстансу, дополнительно вам также нужны соответствующие SSH ключи.

Самый последний шаг во время создания Amazon EC2 инстанса – это выбрать либо создать новый key pair. Key pair – это связка public key и private key, т.е. публичный ключ и приватный ключ. Публичный ключ сохраняется внутри Amazon EC2 инстанса, а private key вы загружаете для локального хранения. В момент подключения по SSH вы предоставляете ваш private key, и после чего происходит криптографическая операция, которая сверяется, действительно ли ваш private key относится к public key, хранящийся внутри Amazon EC2 инстанса. Если все совпадает, то доступ по SSH вам предоставляется.

Здесь вы можете видеть пример списка инстансов внутри вашего AWS аккаунта, т.е. это сервис Amazon EC2. В левом навигационном меню открыта вкладка instances, и на основной странице указывается список всех инстансов. Мы видим, что у нас отфильтрован определенный инстанс, он и выбран. В нижней части страницы вы видите несколько вкладок, таких как Description, Status check, Monitoring Tags и, возможно, любые другие вкладки. В этих вкладках предоставляется информация о конкретном выбранном инстансе.

Мы также с вами можем создавать инстансы, либо проделывать любые другие действия внутри облака AWS программным путем. Для этого существует AWS CLI, т.е. Command Line Interface, и второй вариант – это AWS Software Development Kit, т.е. SDK. Здесь мы видим пример AWS CLI команды, которая создает Amazon EC2 инстанс. Обратите внимание, что она состоит из нескольких частей. Самое первое – это AWS, говорит о том, что мы обращаемся к AWS CLI. Далее название сервиса – это Amazon EC2, и внутри Amazon EC2 мы указываем, какую операцию нам необходимо произвести. В нашем случае это операция run instances. После чего в той же строке мы указываем параметры, которые относятся к этой операции. Здесь мы видим, что указан Image ID, т.е. наш AMI. Далее указываем количество инстансов для создания, тип инстансов. Key name – это название публичного SSH ключа, security группы, а также информацию об AWS регионе, где создаются наши инстансы. Как только запускаем эту команду, мы получаем ответ. Ответ может быть успешен либо не успешен. В случае ошибки предоставляется информация о коде ошибки, а также ошибка в текстовом формате для того, чтобы получить представление, что же пошло не так. В случае, если мы получаем успешный ответ, то в нем как минимум будет указан ID Amazon EC2 инстанса, а также другая информация, которая существует и доступна для этой операции.

На этом слайде представлена информация по жизненному циклу любого Amazon EC2 инстанса. В квадратных фигурах указано состояние инстанса, а название стрелок – это то действие, которое мы производим. Начнем с самого начала. Есть у нас AMI. Когда мы запускаем instance, то instance переходит от AMI в состояние в pending. Pending состояние означает, что AMI устанавливается на железо внутри дата-центра. Как только все настройки произведены и instance готов к бою, т.е. он может начать принимать трафик, вы можете подключаться по SSH, и в целом он доступен для работы, то он переходит от состояния pending в состояние running. Мы с вами помним, что операционная система, т.е. root volume Amazon EC2 инстанса может быть Amazon EBS, back-up, т.е. жесткие диски, созданные внутри сервиса Amazon EBS, либо используется опция Amazon EC2 instance store, когда мы работаем с жесткими дисками, напрямую привязанные к нашему железу внутри дата-центра. И для обоих вариантов у нас доступен вариант reboot, т.е. перезагрузить сервер, в этом случае сервер перезагружается, и данные, которые там находятся в обоих случаях, они сохраняются. Теперь в случае, когда мы работаем с Amazon EBS жесткими дисками, то у нас есть вариант остановить наш instance, она от состояния running, как только мы стартовали действие stop, переходит в состояние stopped через состояние stopping. В состоянии stopped мы не оплачиваем за инстансы, она в состоянии гибернации, мы оплачиваем только за Amazon EBS. Здесь я напомню, что основная часть оплаты, она не за жесткие диски, а именно за инстансы, таким образом остановив инстанс, мы экономим наши деньги. В тот момент, когда нам необходимо наш инстанс стартовать еще раз, то мы можем из состояния stopped его запустить, он переходит в состояние pending, и обратно возвращается в состояние running, т.е. готов к бою. Далее, мы можем terminate, т.е. уничтожить, либо высвободить ресурсы, высвободить железо, и более не оплачивать. Таким образом происходит переход от состояния running в состояние terminated, промежуточное состояние shutting down. Как только сервер у нас остановлен, то какое-то время у нас в списке инстансов наш остановленный, т.е. отключенный инстанс будет отображаться и статус будет terminated. Как только удаление, высвобождение этого инстанса полностью заканчивается, то из списка этот инстанс полностью пропадает.

Мы с вами поговорили подробнее про жизненный цикл Amazon EC2 инстанса. Также в зависимости от состояния EC2 инстанса есть некоторые нюансы работы с публичными IP-адресами. У нас есть два варианта публичного адреса, это public IP адрес, который выдаётся временно для наших инстансов, и эти инстансы доступны через интернет. Второй вариант – это вариант static IP адрес, т.е. постоянный IP адрес через сервис Elastic IP адрес, вы можете отдельно его заказать, если вам нужна такая опция. Так вот, когда мы говорим про public IP адрес, публичный IP адрес это некоторый pool, т.е. некоторый набор IP адресов, не привязанных конкретно к вашему аккаунту, это общий набор IP адресов, которые доступны для Amazon. В зависимости от занятости того или иного IP адреса, вам во временное пользование вашим инстансам выдаётся некоторое значение. И здесь важный момент, когда мы терминейтим инстанс и создаём новый инстанс, здесь понятно, мы получаем случайный публичный IP адрес, и никак не влияем на её значение. В случае, когда мы останавливаем инстанс, также следует запомнить, что публичный IP адрес обратно возвращается AWS, для того, чтобы оно могло назначиться Amazon EC2 инстансу в другом AWS аккаунте. Поэтому, учитывайте этот момент, и знайте, что когда сервер у нас останавливается и запускается снова, для него выдаётся новый публичный IP адрес, который доступен на тот момент, на момент запуска.

Если возвращаясь к статическому IP адресу, то это Elastic IP адрес, выделенный сервис, который вам даёт статический IP адрес. По умолчанию, внутри AWS аккаунта вам доступны пять IP адресов, если же вам нужно больше, то вы можете сделать соответствующий запрос. Этот лимит он soft limit, поэтому может быть увеличен в зависимости от вашей необходимости через обращение в support. Как только ваш инстанс запущен, то у этого инстанса появляются некие метаданные. Как метаданные выступают публичный IP адрес, приватный IP адрес, информация о том, на каком регионе запущен этот инстанс, к какой availability zone относится, какие security группы привязаны к этому инстансу, т.е. вся возможная информация об этом инстансе, она доступна в AWS Management Console в соответствующих вкладках в сгруппированном виде. А также эта же информация доступна из самого инстанса, если обратиться по IP адресу 169.254.169.254. Если мы идем по пути latest meta-data, то мы выходим на метаданные по этому конкретному инстансу. Если мы идем по пути latest и user-data, то мы открываем список команд, которые запускаются при первом запуске этого инстанса. Вы можете использовать эти данные, эта ссылка она постоянная. И в случае, если вам нужна какая-то информация либо с user-data, либо с meta-data конкретного инстанса, то вы можете во время настройки обращаться по этому пути, это JSON документ, в которым соответствующие ключи, этот путь внутри JSON файла тоже постоянный, и вы можете эти данные использовать для финальной настройки вашего инстанса. Также я вам рекомендую запомнить оба URL адреса, так как они бывает приходят на реальном АWS экзамене.

Когда мы говорим про мониторинг нашего Amazon EC2 инстанса, то подразумевается сервис Amazon CloudWatch, это тот сервис, который сохраняет метрики, и вы в графическом виде можете отобразить изменения ваших метрик и предпринимать некоторые шаги либо вручную, либо опять же настроить автоматическое реагирование на соответствующее значение метрик. Как метрики могут выступать: нагруженность ваших ядер процессора, нагрузка на ваши жесткие диски, т.е. операция записи отдельно, операция чтения отдельно, также информация по загрузке канала сети, т.е. Network. Здесь важно отметить, что метрика RAM, т.е. загрузка оперативной памяти по умолчанию недоступна. Для того, чтобы эти метрики тоже передавать, необходимо выполнить некоторые дополнительные действия, и эти метрики также будут для вас доступны. Более того, в рамках CloudWatch есть два варианта мониторинга. Это Basic Monitoring и Detailed Monitoring. При варианте Basic Monitoring метрики передаются каждые 5 минут, и эта опция бесплатная, доступная по умолчанию. Вторая опция, это если ваши метрики нужно передавать чаще, либо вам нужно видеть большую детализацию. В этом случае Detailed Monitoring позволяет вам передавать и хранить метрики по каждой минуте, т.е. ежеминутно. Следует отметить, что Detailed Monitoring, она оплачивается отдельно, и информацию по оплате, по стоимости вы можете посмотреть на соответствующей странице.

Мы с вами добрались до конца второй части нашей лекции. Давайте резюмируем и пройдемся по самым основным моментам. Во время создания Amazon EC2 инстанса, мы можем выбрать операционную систему как Linux, так и Windows. Мы создаем Amazon EC2 инстансы от некоторого AMI. AMI – это Amazon Machine Instance, который в себя включает помимо операционной системы, также дополнительные настройки, установку неких программ и запуск определенных команд, т.е. выступает как снапшот другого инстанса. Также во время создания Amazon EC2 инстанса, мы должны указать в каком VPC он будет создан. С точки зрения безопасности и best practices security рекомендуется не использовать дефолтный VPC, а создать собственный, новый. Для вашего выбора доступны различные типы инстансов, а также есть целые семейства типов инстансов, чтобы подобрать максимально подходящую конфигурацию с точки зрения количества ядер процессора, размера оперативной памяти, возможности хранилищ и сетей. Для того чтобы контролировать доступ к вашим инстансам, вы можете использовать security группы. Security группы работают на уровне инстансов и выступают в роли виртуального firewall. Вы можете во время создания инстанса также передать список команд, который называется user-data, для того, чтобы финализировать настройку и запустить те команды, которые могут быть успешно выполнены только после запуска инстанса. Операционная система инстанса может быть установлена на жесткие диски от Amazon EBS, либо Amazon EC2 Instance Store. Те инстансы, которые использует Amazon EC2 Instance Store не могут быть остановлены. Инстансы, которые используют жесткие диски от Amazon EBS могут быть остановлены. Более того, в остановленном состоянии оплата за инстансы не производится, таким образом, вы экономите деньги. Для того, чтобы мониторить ваши инстансы, вы можете воспользоваться сервисом Amazon CloudWatch и передавать туда соответствующие метрики. На этом мы завершаем вторую часть и переходим к следующей части.

Мы с вами добрались до третьей части сегодняшней нашей лекции. Здесь мы с вами подробнее поговорим про вопросы оплаты и оптимизации оплаты за сервис Amazon EC2. Давайте с вами подробнее разберем, какие модели оплаты существуют за сервис Amazon EC2. Самые первые – это on-demand instances. Это вариант, когда мы оплачиваем ровно за то, за что мы используем. Представим, что мы заказали 10 серверов на 2 часа для того, чтобы произвести некоторые расчеты, после чего эти серверы нам не нужны. Это пример расчетов нагрузки, которые непостоянные или разовые. Таким образом on-demand позволит нам уже в конце месяца получить счет на оплату, где будет сидеть 2 часа, умноженное на 10 инстансов, на 20 часов определенного типа инстансов, не более. Таким образом мы оплатили ровно за то, что мы использовали. Здесь еще важный момент – это то, что в случае, если мы работаем с операционной системой Amazon Linux или Ubuntu, то для нас становится доступны посекундные тарификации. Таким образом, если мы запустили инстанс на 1 минуту и 57 секунд, то ровно за это время мы произведем оплату.

Следующая модель оплаты за Amazon EC2 – это Reserved Instances. Эта модель оплаты идеально подходит для тех нагрузок, которые постоянны 24 на 7. В этом случае мы можем некоторую часть вычислительных мощностей заказать через Reserved Instances и мы получим определенные скидки от AWS. Как это работает? Мы говорим, что мы готовы приобрести определенный объем вычислительных мощностей на год либо на 3 года. Как только заключается договор, эти инстансы со скидкой нам передаются в использование. Есть некоторые нюансы касательно оплаты, а именно 3 варианта, от этого зависит размер скидки. Самую наименьшую скидку мы получаем в случае, когда договор мы заключаем, при этом мы продолжаем оплачивать ежемесячно. Этот вариант называется NURI, т.е. No-upfront Reserved Instances, наперед ничего не оплачиваем. Далее следующий вариант, когда мы 50% периода, год либо 3 года оплачиваем сразу, оставшиеся 50% оплачиваем ежемесячно. Это называется Partial Upfront Reserved Instances либо PURI. И третий вариант, когда мы получаем максимальную скидку в рамках Reserved Instances, это AURI, т.е. All Upfront Reserved Instances. И как вы догадались, мы оплачиваем сразу год либо 3 года работы наших инстансов наперед. Таким образом, ежемесячно нам производить оплату не нужно. Этот вариант идеально подходит для тех случаев, когда вы как бизнес выросли, у вас есть определенный объем нагрузки, постоянный 24 на 7. В этом случае, зная момент, что у вас есть эти постоянные нагрузки, мы можем получить хорошую скидку от AWS через модель Reserved Instances.

Следующая модель оплаты за Amazon EC2 – Spot Instances. Это достаточно интересная модель. Следует запомнить, что Spot Instances дает максимальную возможную прибыль и скидка может достигать до 90%. Это говорит о том, что цены за Spot Instances за единицу времени может быть в 10 раз меньше, чем цены за On-Demand Instances. Это очень интересное предложение, которым следует воспользоваться. Но есть здесь определенные нюансы. Spot Instances, откуда это все появилось? Так как дата-центры AWS достаточно большие помещения, в которых есть достаточно большой объем вычислительных мощностей, частенько бывает, что какая-то часть серверов, она не нагружена, т.е. она не затребована никаким клиентом, пользователем AWS, поэтому зря простаивают. Здесь AWS не растерялся и предложил следующую модель. Он готов делать большие скидки за инстансы, которые не затребованы, не требуются для других моделей оплаты, для того чтобы он что-то мог зарабатывать, пусть и меньше, но при этом зря деньги не терял. Это хорошо и клиентам, так как дает возможность еще больше сэкономить, используя абсолютно то же железо. Момент такой, что Spot Instances по сравнению с On-Demand Instances, они даются во владение, которое контролируется и решается не нами. В случае с On-Demand Instances мы заказываем инстансы, эти инстансы будут доступны для нас ровно до того момента, пока мы сами не решим от них освободиться, т.е. terminated для того, чтобы освободить и вернуть AWS, и больше за них не оплачивать. В случае со Spot Instances мы используем до того момента, пока эти инстансы не затребуются другой моделью оплаты. В этом случае приходит уведомление о том, что этот Spot Instances не может быть дальше обслуживаться по этой модели и через какое-то время отключается, есть такой нюанс. Таким образом, если у вас есть постоянные нагрузки, полностью на Spot Instances она существовать не может. Spot Instances идеально подходят для тех нагрузок, которые являются асинхронными. Представим, что у вас есть какие-то расчеты, эти расчеты могут быть сделаны сейчас, могут быть сделаны в 2 часа ночи, могут быть в любой момент прерваны и продолжены с того места, потому что у вас где-то каким-то образом сохраняется состояние. Для подобных нагрузок Spot Instances идеальный вариант. Есть еще вариант такой, когда вы скейлите, через автоскейлинг настраиваете несколько групп. Часть группы это On-Demand Instances, а часть инстансов это Spot Instances. В тот момент, когда Spot Instances доступны, вы работаете со Spot Instances. Если получается так, что Spot Instances недоступны, то AWS соответственно все Spot Instances забирает себе и автоматически автоскейлинг срабатывает и поднимает уже On-Demand Instances. По цене, естественно, дороже, но как минимум благодаря этой архитектуре есть возможность значительные суммы экономить путем переброски вычислительных мощностей на Spot Instances. Это требует дополнительной настройки, нужен специалист и нужно подходить к задаче с соответствующим расчетом. Если выходит так, что выгоднее это настроить, то бизнесу это нужно предложить и это сделать. Если же выгода от этого небольшая, то рекомендуется оставаться на On-Demand Instances.

Как происходит ценообразование в Spot Instances? Оно организовано неким аукционом, т.е. вы предлагаете какую-то свою цену, другой пользователь AWS также в рамках Spot Instances предлагает свою цену и AWS, естественно, выбирает тот вариант, который максимальный для того, чтобы свою прибыль увеличить, но при этом эта цена может быть в несколько раз меньше, чем On-Demand Instances. Таким образом, все зависит от региона, например, для региона North Virginia, который является самым популярным, самым насыщенным AWS регионом. В нем маловероятно, что будет возможность использовать Spot Instances, а, например, для регионов, которые менее популярны, Spot Instances может быть отличным вариантом, который ввиду того, что очень мало пользователей сидят в этом регионе, может получится так, что ваши постоянные нагрузки большую часть времени отрабатывают по модели Spot Instances, при этом вы получаете максимальные скидки.

Следующие две модели оплаты привязаны к железу Amazon EC2 инстанса, называются Dedicated Hosts и Dedicated Instances. Чем они отличаются? Dedicated Hosts позволяет вам привязаться к некоторым характеристикам этого железа, для того, чтобы применить лицензии. Например, лицензии баз данных Oracle привязываются к количеству ядер на определенном железе. В этом случае, чтобы применить эту лицензию, вы можете использовать Dedicated Hosts. Dedicated Instances – это тот кейс, когда в облаке вам выделяется отдельный сервер железа, в котором нагрузки других пользователей AWS не допускаются. Таким образом, вы являетесь не владельцем, арендуете это железо полностью под себя, и никто другой там не будет. Эти инстансы бывают нужны для некоторых регуляторных требований. Например, требуется, чтобы эти нагрузки, ввиду некоторой специфики данных, либо обработки, должны запускаться на отдельном железе. В этом случае вам необходимо использовать Dedicated Instances. Dedicated Instances, ввиду своей специфики, являются одними из самых дорогих, потому что под вас персонально выделяется конкретное железо, и это достаточно дорогостоящая услуга. Но также следует отметить, что Dedicated Instances достаточно редко требуются, опять же, ввиду некоторых регуляторных требований. Во всех других нормальных случаях Оn-demand instances будет более чем достаточно.

И последний вариант оплаты – это Scheduled Reserved Instances. Здесь следует сразу отметить, что Scheduled Reserved Instances на текущий момент недоступны. Ввиду того, что вопросы внутри реального экзамена AWS обновляются, либо могут обновляться нечасто, то вопросы, связанные с Scheduled Reserved Instances в рамках сервиса Amazon EC2, могут вам еще приходить. Поэтому мы этот момент сейчас разберем, но при этом вы для себя имейте в виду, внутри AWS на текущий момент этой опции на сегодняшний момент не существует. Что же такое Scheduled Reserved Instances? Это та опция, которая позволяет вам каждый день либо определенный день недели, еженедельно, ежемесячно, по какому-то расписанию определенные часы резервировать под себя на один год. Таким образом, вы производите оплату на один год, указываете какое время, какие инстансы должны быть для вас доступны. И в этом случае они в этот момент запускаются. И вы, опять же, так же как и Reserved Instances, по некоторой скидке производите оплату за эти инстансы.

Мы на следующих двух слайдах резюмируем все то, что было сказано по моделям оплаты за сервисы Amazon EC2. Если мы говорим On-demand instances, это тот вариант, который максимально гибкий. Spot instances это тот вариант, который удачно подходит для асинхронных нагрузок. Reserved instances – это хороший вариант модели оплаты, когда у нас есть постоянные нагрузки 24 на 7. И мы используем Dedicated hosts в том случае, когда по некоторым регуляторным требованиям нам необходимо использовать только самим одно определенное железо.

Если мы говорим про use cases, On-demand instances используются для кратковременных динамических нагрузок, которые сложно предугадать. В этом случае On-demand подходят идеально. Другой вариант – это мы используем On-demand instances для среды разработки либо тестирования, т.е. разработка происходит не всегда постоянно, тестирование тоже происходит не всегда постоянно. Поэтому после того, как мы провели этап разработки и тестирования, и эти инстансы зря простаивают, мы можем их отключить и при этом не терять зря деньги. Spot instances – это тот кейс, когда у нас асинхронные нагрузки, мы можем наши расчеты, нагрузки в любой момент остановить и продолжить с того места. Оно может отрабатывать ночью, утром, не имеет значения. И также один из примеров, это когда нам необходимо запустить некоторые расчеты на больших объемах вычислительных мощностей. В этом случае Spot instances даст некоторую выгоду от объемов. Следующий вариант – это Reserved instances, когда у нас есть постоянные нагрузки 24 на 7. Один из вариантов использования – это если компания большая, у нее есть возможность оплатить наперед, при этом знать, что он получит хорошие скидки. В этом случае можно использовать Reserved instances. Другой вариант – это когда мы настраиваем Disaster recovery, т.е. Disaster recovery это когда наша инфраструктура развернута на двух регионах и один основной, другой резервный. В случае когда происходит какая-то проблема с целым AWS регионом, что происходит очень редко. Но мы говорим, что наши нагрузки, наша инфраструктура должна быть 100% всегда доступна для наших клиентов, потому как потеря клиентов ввиду недоступности на некоторое время может в целом обанкротить всю компанию. Поэтому это достаточно критичные нагрузки. В этом случае компания разрабатывает и поддерживает DR, т.е. disaster recovery. И если компании нужно гарантировать, что в основном либо резервном регионе должен быть гарантированный объем вычислительных мощностей, наперед это зная, в этом случае мы можем через reserved instances эти вычислительные мощности забронировать, и как только они понадобятся их запускать. Либо держать в постоянно запущенном состоянии, зная, что мы с какой-то скидкой получили, при этом мы 100% уверены, что в момент переключения эти инстансы для нас будут доступны, и мы сможем обработать все запросы наших клиентов. Последний вариант – это dedicated hosts. Один из примеров использования, если у нас есть уже купленная лицензия, мы можем опять же в зависимости от ситуации использовать свои лицензии, при этом сэкономить на этом. Другой use cases – это когда нам необходимо следовать некоторым регуляторным требованиям. В этом случае работаем на определенном железе, на который не допускаются другие чужие нагрузки.

Давайте резюмируем все то, что было сказано на нашей лекции касательно оптимизации расходов на сервисе Amazon EC2. Оно состоит из четырех направлений, это Right size, Increase elasticity, Optimal pricing model и Optimize storage choices. Давайте подробнее остановимся на каждой из этих направлений. Первое направление – это Right size. Идея в том, что нам необходимо подобрать тот тип инстанса, которые максимально подходит под наши нагрузки. Amazon предоставляет более 60 различных типов инстансов и размеров, поэтому определенно мы сможем найти тот вариант, который устраивает нас. Понятно, что этот процесс не одноразовый, вы не можете угадать либо по каким-либо факторам определить какой вариант определенно ваш, до того как ваши нагрузки начнутся в облаке. Происходит это следующим образом, вы выбираете тот вариант типа инстанса, который максимально дешевый и который будет обслуживать ваши нагрузки, с этим вы начинаете. Далее со временем вы периодически проверяете ваши метрики, т.е. вы передаете ваши метрики в сервис Amazon CloudWatch и видите какая идет нагрузка на процессор, какая идет нагрузка на оперативную память, хранилище и сеть, и делаете выводы.

Приведу простой пример. Представьте, что у вас есть instance type, у которого CPU нагружена на 70%, а оперативная память нагружена на 10%. Здесь мы видим явно, что наше приложение больше использует процессорные мощности, нежели оперативную память. Таким образом, мы можем выбрать тип инстанса, который меньше, но при этом из семейства, который CPU, т.е. Type-C. В этом случае мы можем сэкономить на размере инстанса, потому что он будет меньше, но при этом у него оперативной памяти будет гораздо меньше. Исходя из того, что нагрузки не требует оперативной памяти, этот вариант тип инстанса больше нам подходит. Он справляется полностью с нагрузками нашими текущими, при этом мы оплачиваем меньше, нежели мы оплачивали с предыдущим вариантом инстанса.

Второе направление – Increase elasticity. Здесь говорится про то, что мы должны правильно настроить наш автоскейлинг. Автоскейлинг – это когда наша инфраструктура, это некий живой абстрактный организм, когда в зависимости от нагрузки количество инстансов, которые обрабатывает ваш трафик, либо увеличивается, либо уменьшается. Представим, что вы какой-то интернет-магазин и ваши нагрузки в основном после 6 часов вечера до 10 часов вечера. В этом случае, система, наблюдая за метриками на ваших инстансах, определяет, что сейчас нагрузка растет, нагрузка на сервера растет, поэтому нам необходимо после определенного порога запустить дополнительные сервера, чтобы это все работало верно. В случае, когда мы опускаемся ниже какого-то другого второго порога, например, если нагрузка на наши сервера меньше 10%, то в этом случае мы видим, что нагрузки как таковой нет. Поэтому мы можем лишние инстансы терминейтнуть и вернуть обратно AWS, чтобы за них не платить. Таким образом, размер нашей инфраструктуры уменьшается, либо увеличивается в зависимости от нагрузки. Таким образом, вы оплачиваете только за те реальные нагрузки и у вас нет состояния, когда некоторые ваши ресурсы, инстансы просто так простаивают, и вы фактически впустую оплачиваете за его использование.

Третье направление – это Оptimal pricing model. Мы с вами ранее проговорили, какие модели оплаты существуют для сервиса Amazon EC2. Таким образом, в зависимости от ваших нагрузок, вы можете подобрать ту модель оплаты, которая будет для вас максимально удобной и эффективной. При этом вы несете минимальные расходы на содержание этой инфраструктуры.

Следующий вариант – это оптимизация нашего storage, т.е. хранилища. Здесь подразумевается три момента. Первый момент – это то, что нам необходимо подбирать размер жестких дисков оптимальными, т.е. для инстанса, для которого требуется 20 ГБ на основном жестком диске, нет необходимости делать большой запас и создавать жесткий диск в 500 ГБ. Вы можете создать с небольшим запасом, также вы вспомните, что у вас есть сервис CloudWatch, куда вы пересылаете ваши метрики, и вы видите нагруженность ваших жестких дисков в тот момент, когда достигается некий порог, например, 90% жесткого диска заполнено. В этом случае вы можете произвести какое-то действие, чтобы автоматически увеличить размер жестких дисков. Либо сообщить по любым каналам, будь то e-mail либо SMS, о том, что жесткий диск переполнен, необходимо предпринять какое-то действие. В этом случае, вы можете самостоятельно подключиться, либо увеличить, либо посмотреть, почему такое произошло, если это не ожидалось.

Второй пункт – это тип EBS volume. На следующих лекциях мы с вами пройдем, какие типы EBS volume бывают и в каких случаях какой из типов правильнее выбрать. И третий вариант – это то, что у вас есть некоторые снапшоты, т.е. бэкапы ваших жестких дисков. Это требуется для того, чтобы в случае возникновения каких-то проблем вы могли легко восстановиться. Это нормальная практика – делать бэкапы. Но при этом хранить бэкапы пятилетней давности не всегда рационально. Достаточно хранить бэкапы, опять же, в зависимости от ваших бизнес требований, недельной давности, месячной давности, может быть, года давности, но при этом с некоторыми сдвигами по времени, т.е. вы храните один бэкап, который год назад был создан, один бэкап, который был создан месяц назад, чтобы туда можно было откатиться, и семь бэкапов на последние семь дней. Это максимально оптимальный вариант хранения бэкапов. Все остальные бэкапы должны быть удалены, потому что за бесполезные бэкапы вы также оплачиваете, так как это отдельно созданный ресурс внутри AWS. Чем меньше лишних ресурсов, тем меньше вы, соответственно, оплачиваете.

Следует отметить, что процесс оптимизации расходов в облаке – это не разовый процесс. Это не связано, опять же, напрямую с облаком. Облако лишь дает возможность видеть все ваши ресурсы как на ладони. Но при этом проблема того, что нагрузки меняются, новые ресурсы создаются, она существуют как в облаке, так и на локальном дата-центре. Поэтому идея в том, чтобы, имея в наличии максимальную видимость на стороне облака, у вас есть возможность это все измерить, периодически мониторить и делать некоторые шаги по улучшению. После того, как вы проделали это улучшение, все эти шаги циклически повторять, потому как инфраструктура постоянно меняется, команд у вас разработки может быть много, и вся эта инфраструктура, это как живое организм, она постоянно в динамике, что-то с ней происходит. Поэтому правильно измерять, мониторить, отслеживать изменения, видеть текущее состояние и предпринимать какие-то действия, при этом все эти шаги периодически повторять – это правильный процесс работы по оптимизации расходов в облаке. Все необходимые ресурсы, возможности облака для нас предоставляет.

Одним из моментов, которые следует отметить – это тегирование. Благодаря тегированию вы сможете ваши ресурсы логически подразделять для того, чтобы работать уже с меньшими группами, так как с меньшими группами работать легче. Приведу пример, если вы будете тегировать все ваши ресурсы с ID вашего департамента, то вы будете знать, какому департаменту относятся те или иные проблемные ресурсы. Вы знаете, куда адресно обратиться для того, чтобы эту проблему устранить. Более того, помимо тегинга есть ряд интересных сервисов, которые позволяют вам управлять, оптимизировать ваши расходы. Вы всегда можете одним запросом Google найти эти ресурсы, почитать про них и начать их использовать.

Мы с вами добрались до конца нашей третьей части, а также до конца нашей сегодняшней лекционной сессии. Давайте пройдемся по самым основным моментам. Мы разобрали, какие модели оплаты существуют для сервиса Amazon EC2. Это on-demand instances, reserved instances, spot instances, dedicated instances и dedicated host, а также scheduled reserved instances, которые на текущий момент недоступны. Мы также с вами узнали, что spot instances является наиболее выгодным вариантом в случае, если у нас определенный тип нагрузок. Spot instances могут быть прерваны с двухминутной нотификацией, т.е. уведомлением о том, что через 2 минуты этот инстанс будет отключен. Соответственно, вам нужно строить архитектуру таким образом, чтобы адекватно реагировать на эти отключения. При этом вы можете получить до 90% скидки, используя spot instances. Далее мы разобрали четыре направления, которые следует рассмотреть для оптимизации ваших затрат в облаке, связанных с сервисом Amazon EC2. Это right sizing, дальше это increasing elasticity, т.е. правильное использование автоскейлинга. Далее optimal pricing model, т.е. выбрать ту модель, которая оптимально подходит под наши нагрузки. И четвертое – это оптимизация хранилища, т.е. есть некоторые нюансы по работе с нашими хранилищами. Про сервисы хранения storage мы разберем и более подробно поговорим с вами на следующих наших лекционных занятиях.

На этом мы подошли к концу сегодняшнего лекционного занятия. Мы подробнее разобрали сервис Amazon EC2, а именно посмотрели, какие шаги нужно предпринять для того, чтобы создать новый Amazon EC2 инстанс. Далее рассмотрели, какие модели оплаты существуют, а также познакомились с теми направлениями работ, которые позволяют нам оптимизировать наши расходы на сервис Amazon EC2.

На этом мы завершаем нашу сегодняшнюю сессию. Спасибо за внимание. До встречи! Увидимся с вами на следующих наших лекциях!