WEBVTT

00:00:00.000 --> 00:00:02.880

Добрый день, уважаемые студенты!

00:00:02.880 --> 00:00:06.120

Я рад вас всех видеть на очередной сессии Демо.

00:00:06.120 --> 00:00:16.640

Сегодня мы с вами проделаем абсолютно то же задание, что и предыдущее Демо, то есть создадим VPC и все ее необходимые компоненты в облаке AWS.

00:00:16.640 --> 00:00:27.280

Отличие лишь в том, что мы будем для этого использовать не AWS Management Console, а AWS CLI, то есть командную строку для работы с облаком AWS.

00:00:27.280 --> 00:00:32.200

Итак, давайте начнем.

00:00:32.200 --> 00:00:34.400

Давайте вспомним какое же было задание.

00:00:34.400 --> 00:00:42.680

Необходимо было в AWS аккаунте создать в регионе North Virginia, это us-east-1 VPC.

00:00:42.680 --> 00:00:48.400

VPC будет называться DEMO\_VPC с указанным CIDR блоком.

00:00:48.400 --> 00:00:54.040

Все ресурсы в VPC будут в одной availability zone это us-east-1a.

00:00:54.040 --> 00:00:58.560

В VPC у нас будет два subnet, один public, другой private.

00:00:58.560 --> 00:01:06.020

Мы с вами помним, что public subnet это лишь только название и необходимо сделать определенные шаги, чтобы он фактически стал public.

00:01:06.020 --> 00:01:13.800

Для этого у нас будут отдельные два route table, один для public subnet, другой для private subnet.

00:01:13.800 --> 00:01:20.680

Мы подключим Internet gateway к нашему VPC, настроим NAT gateway и поработаем с Elastic IP адресом.

00:01:20.680 --> 00:01:29.200

Все работы мы будем проводить в sandbox.

00:01:29.200 --> 00:01:42.560

В нижней части списка модулей в AWS Academy вы можете найти пункт Sandbox и если по нему перейдете, будет аналогичный интерфейс с лабораторной работой.

00:01:42.560 --> 00:01:55.400

Здесь нам необходимо будет открыть окно Terminal и далее стартовать лабораторную работу, чтобы для нас создался временный AWS account.

00:01:55.400 --> 00:02:02.760

Как только вы ее нажмете необходимо будет подождать некоторое время, после чего статус лабораторной работы перейдет в Ready.

00:02:02.760 --> 00:02:11.280

Мы можем закрывать это всплывающее окно и после того как вы нажмете на кнопку AWS, вас направить на AWS Management Console.

00:02:11.280 --> 00:02:24.200

Все команды мы будем вводить на странице лабораторной работы в Terminal, а проверять будем позже в AWS Management Console.

00:02:24.200 --> 00:02:29.480

Мы начинаем первую часть нашего Демо и сейчас будем создавать VPC.

00:02:29.480 --> 00:02:33.600

Также разберем определенные нюансы при работе с командной строкой.

00:02:33.600 --> 00:02:36.760

Самая первая команда - это создание VPC.

00:02:36.760 --> 00:02:39.280

VPC без дополнительных компонентов.

00:02:39.280 --> 00:02:43.320

Вы видите команду, давайте разберем команду по частям.

00:02:43.320 --> 00:02:50.120

Первая часть - это ключевое слово AWS, говорит о том, что мы начинаем использовать AWS CLI.

00:02:50.120 --> 00:02:54.720

Далее через пробел идет второе ключевое слово - это название сервиса.

00:02:54.720 --> 00:02:58.200

В нашем случае это EC2.

00:02:58.200 --> 00:03:02.880

Третье ключевое слово - это название API или операции.

00:03:02.880 --> 00:03:05.440

В нашем случае, это Create VPC.

00:03:05.440 --> 00:03:20.160

Далее, вы видите через двойное тире перечисленные параметры с теми значениями, которые мы хотели бы присвоить создаваемому ресурсу.

00:03:20.160 --> 00:03:25.640

В нашем случае мы создаем VPC.

00:03:25.640 --> 00:03:32.440

Первый параметр - это CIDR блок с тем значением, которое мы видели на картинке в самом начале.

00:03:32.440 --> 00:03:34.960

Далее, второй параметр - это регион.

00:03:34.960 --> 00:03:39.400

И здесь мы указываем значение регион North Virginia, us-east-1.

00:03:39.400 --> 00:03:45.120

Как только вы наберете эту команду и нажмете Enter, вы в ответе получите JSON.

00:03:45.120 --> 00:03:48.840

Этот JSON описывает ресурс, который был создан.

00:03:48.840 --> 00:03:52.160

Вы видите, что уже присвоился VPC ID.

00:03:52.160 --> 00:04:01.120

Также видите CIDR блок, который мы с вами ввели в входных данных и другие метаданные.

00:04:01.120 --> 00:04:07.960

Давайте теперь проверим, что этот VPC действительно появился и откроем AWS Management Console.

00:04:07.960 --> 00:04:13.540

Здесь нам необходимо воспользоваться строкой поиска сервисов и начать вводить VPC.

00:04:13.540 --> 00:04:19.440

Как только вы перейдете на нужный сервис, обязательно проверьте, что вы находитесь в нужном регионе.

00:04:19.440 --> 00:04:25.560

И в левом навигационном меню необходимо воспользоваться пунктом Your VPCs.

00:04:25.560 --> 00:04:36.440

Вы увидите, что у нас создался VPC с тем же VPC ID и с CIDR блоком, который мы назначили.

00:04:36.440 --> 00:04:38.040

Давайте теперь удалим этот VPC.

00:04:38.040 --> 00:04:43.600

Я бы хотел вам продемонстрировать возможности командной строки.

00:04:43.600 --> 00:04:51.840

Нам необходимо будет подтвердить удаление VPC и увидеть сообщение, что VPC действительно удалено.

00:04:51.840 --> 00:05:07.080

Мы теперь обратно переходим в командную строку и вы видите здесь, что мы результат работы команды, которая создает VPC, присваиваем переменные.

00:05:07.080 --> 00:05:11.120

Но обратите внимание, у нас добавились дополнительные параметры.

00:05:11.120 --> 00:05:13.240

Это output, которое равно текст.

00:05:13.240 --> 00:05:17.280

То есть он в виде текста возвращает результат работы команды.

00:05:17.280 --> 00:05:19.440

И второй это query, то есть фильтрация.

00:05:19.440 --> 00:05:26.520

Мы хотим в ответе вернуть значение по пути Vpc.VpcId.

00:05:26.520 --> 00:05:32.520

И это значение присвоить переменной vpc\_id.

00:05:32.520 --> 00:05:40.320

Далее как только команда отработает, мы можем попробовать вывести на экран значение переменной VPC ID.

00:05:40.320 --> 00:05:52.840

Вы увидите, что мы получили другой уникальный VPC ID, но с теми же параметрами, которые мы указывали в команде.

00:05:52.840 --> 00:05:59.320

Давайте теперь перейдем обратно в AWS Management Console и проверим, что действительно все корректно отработало.

00:05:59.320 --> 00:06:04.400

Мы видим, что CIDR блок тот, который мы указывали, и совпадает VPC ID.

00:06:04.400 --> 00:06:16.680

Отличие этого действия от предыдущего в том, что в командной строке мы уже сохранили VPC ID в переменной и можем его использовать в следующих наших командах.

00:06:16.680 --> 00:06:32.080

Это исключает необходимость ввода различных ID-шников, исключает человеческий фактор и упрощает работу с облаком AWS через командную строку.

00:06:32.080 --> 00:06:43.760

Мы возвращаемся обратно в командную строку и давайте здесь попробуем заполнить тег с ключом Name, значением Demo VPC.

00:06:43.760 --> 00:06:47.240

Для этого воспользуемся следующей командой.

00:06:47.240 --> 00:06:49.400

Мы воспользуемся сервисом EC2.

00:06:49.400 --> 00:06:54.640

Далее у нас есть API Create Tags и следующие параметры.

00:06:54.640 --> 00:07:00.440

Обратите внимание, что для параметра Resources мы передаем нашу переменную VPC ID.

00:07:00.440 --> 00:07:10.920

И в параметре Tags мы передаем в определенном формате, что для тега с ключом Name мы хотим присвоить значение Demo VPC.

00:07:10.920 --> 00:07:18.600

Как только вы запустите эту команду, если она не вернет ошибку, это говорит о том, что команда успешно отработала.

00:07:18.600 --> 00:07:30.040

Теперь, если мы вернемся обратно в AWS Management Console, мы увидим, что столбец Name для нашего VPC заполнился, ее значение Demo VPC.

00:07:30.040 --> 00:07:39.440

Отлично, мы с вами возвращаемся обратно в командную строку и здесь я хотел бы обратить ваше внимание на следующее.

00:07:39.440 --> 00:07:46.680

Наличие тегов в ресурсе позволяет нам фильтровать и запрашивать метаданные об этих ресурсах.

00:07:46.680 --> 00:07:58.720

Пример, мы можем сейчас получить информацию о нашем VPC, отфильтровав его по тегу Name и по совпадению значения Demo VPC.

00:07:58.720 --> 00:08:07.760

Для этого воспользуемся API describe VPCs и введем параметр в определенном формате, который называется filters.

00:08:07.760 --> 00:08:11.240

В результате вы видите, нам вернулась JSON.

00:08:11.240 --> 00:08:20.760

Есть единственный ключ VPCs, это массив из JSON и каждый JSON описывает определенный ресурс.

00:08:20.760 --> 00:08:33.000

Так как мы в фильтрах указали определенный конкретный VPC, то и в массиве был возвращен один JSON и описывает наш VPC.

00:08:33.000 --> 00:08:54.240

Здесь бы я также хотел обратить ваше внимание, что работая с командной строкой, вы не только пишете независимые друг от друга команды, это не только про присваивание значений переменным и работе с этими переменными, вы можете писать отдельные bash скрипты.

00:08:54.240 --> 00:09:03.640

Идея в том, что все эти команды могут быть объединены в одном определенном файле и этот файл выполняет определенную задачу.

00:09:03.640 --> 00:09:06.920

Не маленькую, это не одна команда, это набор из нескольких команд.

00:09:06.920 --> 00:09:14.480

В тот момент, когда вы запускаете этот bash скрипт, то последовательно выполняются все команды из этого скрипта.

00:09:14.480 --> 00:09:24.840

Таким образом, вы через bash скрипты можете определенную часть своей ежедневной работы, работы Cloud инженера автоматизировать.

00:09:24.840 --> 00:09:34.160

Там также вы можете использовать переменные, более того, вы можете использовать более сложные конструкции, то есть циклы, if else конструкции и так далее.

00:09:34.160 --> 00:09:40.760

В рамках нашего Демо мы с вами будем использовать простой вариант.

00:09:40.760 --> 00:10:05.640

Написание bash скриптов не входит в рамках темы нашего Демо, поэтому мы будем писать независимые команды, но при этом максимально будем стараться не привязываться к конкретным значениям, а значение ID-шники ресурсов будем присваивать переменным, чтобы потом их можно было переиспользовать.

00:10:05.640 --> 00:10:12.320

Мы с вами двигаемся к следующим компонентам нашего VPC и первым делом создадим Internet gateway.

00:10:12.320 --> 00:10:15.520

Для этого мы воспользуемся следующей командой.

00:10:15.520 --> 00:10:24.040

Здесь мы сохраним значения по пути InternetGateway.InternetGatewayId в переменную ig\_id.

00:10:24.040 --> 00:10:38.000

Вы видите, после исполнения команды, мы вывели на экран значение в переменной ig\_id и здесь отображается ID InternetGateway.

00:10:38.000 --> 00:10:45.920

Также мы можем перепроверить в AWS Management Console такой же ID, он появился в списке InternetGateway.

00:10:45.920 --> 00:10:54.360

Теперь нам необходимо для этого InternetGateway заполнить тег с ключом Name значение Demo IGW.

00:10:54.360 --> 00:11:07.720

Запускается абсолютно та же команда, которую мы запускали ранее, отличие лишь в том, что в параметр Resources мы передаем ID InternetGateway.

00:11:07.720 --> 00:11:10.680

Оно у нас сохранено в переменной ig\_id.

00:11:10.680 --> 00:11:19.200

Как только мы это сделаем, вы можете вернуться в AWS Management Console и проверить, что оно успешно применилось.

00:11:19.200 --> 00:11:20.480

Двигаемся дальше.

00:11:20.480 --> 00:11:24.560

Созданный InternetGateway теперь нам необходимо привязать к VPC.

00:11:24.560 --> 00:11:30.440

Для этого есть API Attach InternetGateway.

00:11:30.440 --> 00:11:35.280

Обязательными параметрами являются InternetGateway и VPCID.

00:11:35.280 --> 00:11:49.520

Что тот, что этот у нас сохранены в наших переменных, поэтому мы соответственно для InternetGatewayID указываем значение переменной ig\_id, а для VPCID переменной vpc\_id.

00:11:49.520 --> 00:12:04.080

После того, как вы запустите команду, в случае если не было выдано ошибки, то мы предполагаем, что команда была выполнена успешно.

00:12:04.080 --> 00:12:06.000

Теперь мы двигаемся дальше.

00:12:06.000 --> 00:12:08.920

Мы будем создавать subnets.

00:12:08.920 --> 00:12:20.120

Первым делом нам необходимо проверить действительно ли availability zone us-east-1a существует.

00:12:20.120 --> 00:12:38.380

На самом деле это очевидно, так как мы знаем системное название региона это us-east-1 и далее мы знаем, что availability zone, если их три например, то они обозначаются в алфавитном порядке.

00:12:38.380 --> 00:12:42.360

Это us-east-1a, 1b и 1c.

00:12:42.360 --> 00:12:54.640

Идея этой команды была в том, что вы можете внутри программы заложить запрос существующих availability zone для указанного региона.

00:12:54.640 --> 00:13:11.080

В этом случае как value вы указываете не us-east-1a, а us-east-1 и тогда вам вернется в ответе массив availability zone с JSON, в которых будет содержаться информация каждой availability zone.

00:13:11.080 --> 00:13:25.160

Так как мы отфильтровали по нужной нам availability zone и в ответе что-то вернулось, мы для себя дополнительно подтвердили, что написание этой availability zone верное.

00:13:25.160 --> 00:13:27.680

Теперь мы будем создавать public subnet.

00:13:27.680 --> 00:13:37.460

Я не буду детально останавливаться на частях команды, так как идея она абсолютно такая же, что и в предыдущих командах.

00:13:37.460 --> 00:13:40.520

А предыдущие команды мы с вами уже подробно разобрали.

00:13:40.520 --> 00:13:49.440

Здесь мы будем использовать API create subnet, введем необходимые параметры это availability zone, CIDR block и VPC ID.

00:13:49.440 --> 00:13:55.320

Далее это все мы присваиваем переменной.

00:13:55.320 --> 00:14:00.280

Нужно нам значение по пути Subnet.SubnetId.

00:14:00.280 --> 00:14:04.640

Мы сразу проверяем значение в переменной public\_subnet\_id.

00:14:04.640 --> 00:14:07.040

Есть некоторая ID.

00:14:07.040 --> 00:14:08.040

Отлично.

00:14:08.040 --> 00:14:09.040

Двигаемся дальше.

00:14:09.040 --> 00:14:15.120

Теперь нам необходимо для этого subnet заполнить тег с ключом name.

00:14:15.120 --> 00:14:18.520

Заполним значением Demo Public Subnet.

00:14:18.520 --> 00:14:20.080

Переходим к следующему шагу.

00:14:20.080 --> 00:14:25.520

Теперь нам необходимо создать NAT gateway в нашем public subnet.

00:14:25.520 --> 00:14:30.600

Для этого в первую очередь нам необходимо создать Elastic IP address.

00:14:30.600 --> 00:14:35.240

У нас есть отдельный API для создания Elastic IP адресов.

00:14:35.240 --> 00:14:37.800

Называется Allocate address.

00:14:37.800 --> 00:14:52.280

Здесь мы дополнительных параметров не указываем, а лишь сохраняем в переменную eip значение в JSON по пути AllocationId.

00:14:52.280 --> 00:14:58.200

Вы видите, что в переменной eip появился ID elastic IP адреса.

00:14:58.200 --> 00:15:02.400

Теперь мы можем двигаться дальше, а именно создадим NAT gateway.

00:15:02.400 --> 00:15:06.920

В этом случае мы используем API create NAT gateway.

00:15:06.920 --> 00:15:25.560

Передаем сюда параметры subnet ID, allocation ID и сохраняем значение по пути Natgateway.NatgatewayId в параметр ngw\_id.

00:15:25.560 --> 00:15:38.600

Следующий шаг, чтобы сделать наш subnet фактически public, нам необходимо создать route table и в этом route table произвести дополнительные настройки.

00:15:38.600 --> 00:15:56.680

Итак, для создания route table мы воспользуемся API create route table, вводим параметры VPC ID и как результат в переменной public\_rt сохраняем значение из JSON по пути RouteTable.RouteTableId.

00:15:56.680 --> 00:16:00.040

Мы видим, что в переменную записалось некоторое значение.

00:16:00.040 --> 00:16:01.040

Отлично.

00:16:01.040 --> 00:16:18.160

Теперь следующим шагом мы заполним тег с ключом name, значение будет Public Subnet RouteTable и теперь для нашего route table добавим routing rule.

00:16:18.160 --> 00:16:31.680

Как мы с вами помним, нам нужно добавить routing rule, который как destination использует интернет и как target у нас используется Internet gateway.

00:16:31.680 --> 00:16:37.240

Соответственно, мы заполняем параметры destination cidr block и gateway ID.

00:16:37.240 --> 00:16:51.040

Как только мы это все проделаем, мы получим ответ в виде JSON, где ключ Return заполнен значением true.

00:16:51.040 --> 00:17:03.880

Мы сейчас создали все необходимые routing rule, теперь не забываем, что route table это отдельно независимый ресурс и чтобы он использовался, нам необходимо его привязать к subnet.

00:17:03.880 --> 00:17:14.560

Мы помним, что у каждого subnet может быть привязан только один route table, но любой route table может использоваться несколькими subnet.

00:17:14.560 --> 00:17:24.040

Окей, дальше мы будем использовать API associate route table, здесь укажем параметр route table и subnet ID.

00:17:24.040 --> 00:17:37.120

После того как мы это сделаем, мы увидим в ответе JSON, где у нас создалась ассоциация, у нее есть тоже ресурс ID, видите ее на экране.

00:17:37.120 --> 00:17:44.600

То, что мы не получили ошибку, говорит о том, что операция была успешно выполнена.

00:17:44.600 --> 00:17:52.880

Мы с вами добрались до третьей части нашего Демо и будем создавать компоненты, связанные с private subnet.

00:17:52.880 --> 00:17:56.240

Первым делом нам необходимо создать этот subnet ID.

00:17:56.240 --> 00:18:06.920

Вы видите команду, мы используем API create subnet, указываем availability zone, CIDR block, VPC ID и это все сохраняем в переменную private\_subnet\_id.

00:18:06.920 --> 00:18:12.800

Из JSON мы используем значение по пути Subnet.SubnetId.

00:18:12.800 --> 00:18:17.480

Далее мы видим, что в переменной private\_subnet\_id появилось значение.

00:18:17.480 --> 00:18:27.680

Теперь нам необходимо для созданного subnet ID заполнить значение ключ name.

00:18:27.680 --> 00:18:36.120

Для этого воспользуемся API create tags и введем параметры resources и tags.

00:18:36.120 --> 00:18:50.560

Далее нам необходимо создать route table, это будет private route table, используем API create route table, заполняем параметры VPC ID, у нас есть уже значение в переменной vpc\_id.

00:18:50.560 --> 00:19:09.960

Далее нам необходимо протегать этот route table, чтобы мы могли его легче различать. Значением тега с ключом name будет private subnet route table и здесь нам необходимо будет произвести два изменения в route table.

00:19:09.960 --> 00:19:19.640

Первое - это ввести routing rule, который трафик в интернет будет пропускать через созданный нами ранее NAT gateway.

00:19:19.640 --> 00:19:23.580

ID NAT gateway у нас хранится в переменной ngw\_id.

00:19:23.580 --> 00:19:34.480

Как только мы запустим эту команду, мы увидим в результате JSON, где ключ Return присвоено значение true.

00:19:34.480 --> 00:19:38.400

Это значит, что routing rule был успешно создан.

00:19:38.400 --> 00:19:43.520

Теперь нам необходимо этот route table привязать к private subnet.

00:19:43.520 --> 00:20:01.000

Для этого воспользуемся API associate route table, заполним route table ID, который хранится в переменной private\_rt и subnet ID, значение нужное лежит в переменной private\_subnet\_id.

00:20:01.000 --> 00:20:12.320

В результате мы увидим ответ в виде JSON, где ключ association ID заполним некоторым ID, что говорит о том, что операция была успешно выполнена.

00:20:12.320 --> 00:20:13.480

Мы двигаемся дальше.

00:20:13.480 --> 00:20:21.680

Теперь нам необходимо перепроверить в AWS Management Console, что все необходимые ресурсы были созданы.

00:20:21.680 --> 00:20:25.960

Самое первое - это Demo VPC, совпадает с CIDR block.

00:20:25.960 --> 00:20:30.080

Далее, у нас есть два subnet, public subnet и private subnet.

00:20:30.080 --> 00:20:41.400

В public subnet у нас есть дополнительный routing rule, который трафик в интернет направляет через Internet gateway и есть private subnet, у нее трафик в интернет идет через NAT gateway.

00:20:41.400 --> 00:20:43.720

Здесь все верно.

00:20:43.720 --> 00:20:44.720

Двигаемся дальше.

00:20:44.720 --> 00:20:57.160

Есть у нас два route table, один для public subnet и он с ним ассоциирован и то же самое для private subnet, который ассоциирован с private subnet.

00:20:57.160 --> 00:21:03.400

Далее, у нас есть один Internet gateway, который привязан к нашему Demo VPC.

00:21:03.400 --> 00:21:22.680

Есть у нас Elastic IP адрес, который используется NAT gateway и собственно сам NAT gateway, который вы видите, у него есть primary public IP адрес, который является Elastic IP адресом.

00:21:22.680 --> 00:21:34.560

На этом мы выполнили все необходимое в рамках этого задания и фактически все те же ресурсы создали, используя командную строку.

00:21:34.560 --> 00:21:36.800

На этом мы завершаем сессию Демо.

00:21:36.800 --> 00:21:43.240

Я очень надеюсь, что вам стало понятнее принципы работы в AWS CLI.

00:21:43.240 --> 00:21:45.540

Если будут вопросы, пожалуйста задавайте.

00:21:45.540 --> 00:21:47.560

На этом спасибо за внимание.

00:21:47.560 --> 00:21:51.840

Увидимся с вами на следующих наших активностях.