Обучающие материалы для IT

19 июля 2021

Подготовлено для студентов курсов

Содержание

Содержание 2

0. Очень важное замечание 4

1. Контейнерезация и виртуализация (Vagrant,Docker,Packer,Kubernetes) 4

Материалы и литература 4

Замечание к практическому заданию 5

Практические задания 5

2. Jenkins/TeamCity 7

Материалы и литература 7

Практическое задание: 7

3. Ansible 9

Материалы и литература 9

Замечание к практическому заданию 9

Практическое задание 9

4. Linux 11

Материалы и литература 11

Замечание к практическому заданию 12

Практическое задание 12

5. Amazon 14

Материалы и литература 14

Практические задания 14

6. CloudFormation and Terraform. IaaC (Infrastructure as a Code). 17

Материалы и литература 17

Практические задания: 17

7. Azure 18

Материалы и литература 18

Практические задания 18

8. Раздел со звездочкой: сборка сайтов в AWS 19

Terms of use 20

1. Очень важное замечание

Перед началом выполнения заданий, необходимо зарегистрироваться на GitHub и создать новый репозиторий, в который вы будете загружать все свои выполненые задания.

Результатом выполнения здачи будет либо код (Dockerfile, Bash скрипт, JSON файлы и т.д.), либо скриншоты из вашего браузера или другого места где видно, что у вас все работает как требует задание.

2. Контейнерезация и виртуализация (Vagrant,Docker,Packer,Kubernetes)

Материалы и литература

- Vagrant:

  Оригинал:

  Vagrant Documentation

  - цикл статей на русском:

  Системы виртуализации

  Vagrant. Установка и первый запуск

  Vagrant. Создание собственного box-a

  - блог на русском

  Начало работы с Vagrant

  - Еще небольшая статья по старту

  Vagrant. Начало работы. Часть 1 из 2

  Vagrant. Начало работы. Часть 2 из 2

- Docker:

  Часть 1: основы

  Часть 2: термины и концепции

  Часть 3: файлы Dockerfile

  Часть 4: уменьшение размеров образов и ускорение их сборки

  Часть 5: команды

  Часть 6: работа с данными

- Packer:

  Introduction to Packer

  Быстрая сборка образов ОС с помощью Packer

  How to use Packer to create Ubuntu 18.04 Vagrant boxes

  Vagrant Builder

- Kubernetes:

  Документация по Kubernetes

  Шпаргалка по kubectl

  Основы Kubernetes

  Установка Kubernetes с помощью Minikube

- видео:

  Vagrant Beginner (Part 1)

  Vagrant and Packer (part 1)

  Vagrant Crash Course: Vagrant for beginners

Замечание к практическому заданию

Перед выполнением заданий необходимо ознакомиться с документацией. Практически все задания можно выполнить более чем одним способом, любой рабочий вариант будет верным.

Практические задания

2.1. Создать базовый box с Ubuntu 18.04 с помощью packer, выходной формат - vagrant (virtualbox)

2.2. Запустить свежую виртуальную машину в vagrant

2.3. Добавить chef-рецепты для установки на виртуальной машине следующего ПО:

- Apache/Nginx,
- Java
- Tomcat
- MySQL (client & server) - последних версий.

В качестве альтернативы chef вы можете использовать Ansible или Puppet.

2.4. Пробросить порты следующим образом: хост:22022->гость:22, хост:22080->гость:80, хост:22443->гость:443, хост:22306->гость:3306

2.5. С использованием шаблонов виртуальной машины (для virtualbox) создать новый шаблон - с предустановленным mariadb вместо mysql

2.6. Работа с Docker

2.6.1. Создайте папку для проекта и перейдите в нее.
2.6.2. Установите докер на машину.
2.6.3. Запустите контейнер с nginx скачаный из docker hub. Контейнер должен быть запущен из командной строки с параметром `–expose` или `-p` для того чтобы после запуска стартовая страница nginx была доступна из браузера вашего компьютера.
2.6.4. Запустите контейнер с MySQL скачаный из докер хаба, используйте `--volume , -v` при запуске контейнера для того чтобы сохранить базу данных жестком диске хоста.
2.6.5. После запуска контейнера подключитесь к базе, создайте нового польщователя и новую базу.

2.7. Работа с Dockerfile
2.7.1. Создайте файл Dockerfile в корневой папке проекта.
В качестве базового образа используйте Ubuntu 20.04
2.7.2. Дополните Dockerfile инструкциями из которого при выполнении команды docker build соберется docker образ с установленым Ruby 2.7.2
2.7.3. После успешной сборки образа, запустите контейнер для выполнения команды
ruby -v, для проверки работоспособности ruby.

2.8. Работа с Docker Compose
2.8.1. Установите docker-compose на хост
2.8.2. С помощью docker-compose установите и запустите сайт на Wordpress.
По мимо docker-compose.yml файла у вас могут быть другие файлы необхдимые для работы Wordpress, например nginx.conf и другие.

2.9. Работа с Kubernetes
2.9.1. Установите minikube согласно инструкции на официальном сайте.
2.9.2. Создайте namespace для деплоя простого веб приложения.
2.9.3. Напишите deployments файл для установки в Kubernetes простого веб приложения например https://github.com/crccheck/docker-hello-world.
2.9.4. Установите в кластер ingress контроллер
2.9.5. Напишите и установите Ingress rule для получения доступа к своему приложению.

В качестве результата работы сделайте скриншоты результата выполнения команд:

```
kubectl get pods -A

kubectl get svc

kubectl get all
```

а также все написаные вами фалы конфигурации

3. Jenkins/TeamCity

В данном разделе рассматриваются инструменты для continuous integration на примере TeamCity или Jenkins. Они примерно в одинаковой степени используются на проектах и

почти не отличаются друг от друга по своему назначению. Использовать можно любой инструмент. В зависимости от ментора на лекциях и практических занятиях может рассматриваться какой-то конкретный инструмент.

Материалы и литература

Установка и настройка

Jenkins:

- https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-16-04
- https://www.digitalocean.com/community/tutorials/how-to-set-up-jenkins-for-continuous-development-integration-on-centos-7
- https://www.digitalocean.com/community/tutorials/how-to-set-up-continuous-integration-pipelines-in-jenkins-on-ubuntu-16-04

TeamCity:

- https://www.jetbrains.com/help/teamcity/installing-and-configuring-the-teamcity-server.html
- https://www.tutorialspoint.com/continuous_integration/continuous_integration_creating_project_teamcity.htm
- https://www.jetbrains.com/ru-ru/teamcity/documentation/
- https://www.jetbrains.com/help/teamcity/getting-started-with-teamcity.html#3.+Run+your+First+Build

Практическое задание:

1. Установить Jenkins или Teamcity server. Это может быть установка на ваш локальный компьютер или на инстансе в облаке, это не имеет значение, как не имеет значение и метод уставки (с использованием docker контейнера, playbook или установка вручную из репзитория и пр.).
2. Создать новый проект “Staging”, в нем добавить задачу для сборки простого приложения, например
    a. .net: https://github.com/chaitalidey6/HelloWorldAspNetCore/tree/master/HelloWorldAspNetCore
    b. Java: https://github.com/jenkins-docs/simple-java-maven-app
    c. Node JS: https://github.com/jenkins-docs/simple-node-js-react-npm-app

    Замечания:
    - Вы можете использовать любое привычное приложение на любом языке (.net, java, js, python, php).
    - Код приложения должен быть размещен в вашем собственном git-репозитории.
    - Должна использоваться ветка “staging”.
    - Приложение может быть собрано в контейнере (предпочтительный способ).
    - Задача по сборке должна запускаться с параметрами.
    - Результатом сборки обязательно должен быть артифакт (архив, docker-контейнер), который вы дальше будете использовать.

   - Необходимо самостоятельно подумать над тем, каким образом Jenkins/TeamCity получит доступ к git-репозиторию, при этом необходимо придумать наиболее безопасный на ваш взгляд способ.
3. Создать задачу в Jenkins /Teamcity для деплоя вашего артифакта на сервер и перезапуск приложения.
   Замечания:
   - Здесь артефакт может доставляться на удаленный сервер (например, на EC2 инстанс в AWS), либо на контейнер (при работе локально в Docker), либо на локальный сервер (при работе с Vagrant/VirtualBox).
   - Необходимо самостоятельно подумать над тем, каким образом будет организован доступ из Jenkins/Teamcity на сервер (дря загрузки артефактов), при этом необходимо придумать наиболее безопасный на ваш вгляд способ.
4. Настроить зависимость задачи деплоя от задачи сборки.
5. Настроить деплой артифакта в место где он будет работать и запуск приложения.
6. Добавить задачу создания бэкапа артефактов на сервере.
7. Настроить пайплайн, где должны быть включены шаги: сборка, бэкап и деплой (опционально: тестирование).
8. Настроить автоматический запуск деплоя при добавлении нового commit'а в ветке "staging" git.
   ** При запуске локально – здесь могут быть проблемы с настройкой webhook, потому используйте другой метод взаимодействия с git.*

9. Создать новый проект "Production", добавить задачу для сборки приложения, выполнить те же настройки, что и в Staging (п. 2), но с небольшими изменениями: должна использоваться ветка "master".
10. Создать задачу для деплоя Production артефактов на сервер (здесь может использоваться тот же сервер, но приложения должны быть различными: «висеть» на разных портах или под разными доменами).
11. Настроить зависимость задачи деплоя от задачи сборки.
12. Настроить автоматический запуск деплоя при подтверждении pull request'а в ветке "master" в git.

4. Ansible

Материалы и литература

- установка Ansible:

  https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-an-ubuntu-12-04-vps

  https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-14-04

- работа с playbooks

  https://www.digitalocean.com/community/tutorials/how-to-create-ansible-playbooks-to-automate-system-configuration-on-ubuntu

- настройка

  https://www.digitalocean.com/community/tutorials/how-to-configure-apache-using-ansible-on-ubuntu-14-04

  https://www.digitalocean.com/community/tutorials/how-to-deploy-a-basic-php-application-using-ansible-on-ubuntu-14-04

Замечание к практическому заданию

Практические задания по Ansible (раздел 3 и далее) не связаны с заданиями Vagrant/Docker-Jenkins (разделы 1 и 2). В случае досрочного завершения выполнения заданий из раздела 1 – можно перейти к разделу 3 или 4.

Практическое задание

4.1 Создать 3 репозитория в github или bitbucket с названием ab-haproxy, ab-logstash, ab-webui (все дальнейшие действия подразумевают выгрузку результатов в эти репозитории).
Важно: в решении необходимо использовать playbooks, а не ссылаться на репозиторий Galaxy.

4.2 Создать в Vagrant виртуальную машину Ubuntu 18.04

4.3 ab-haproxy

4.3.1 Создать следующие роли в Ansible (можно пользоваться репозиторием Galaxy):

- apt (добавление необходимых пакетов, обновление из репозиториев установленных по умолчанию пакетов)
- ntp (обновить время, настроить синхронизацию времени по cron 1 раз в сутки с любого общедоступного сервера времени)
- monit (установить и настроить monit для само-мониторинга виртуальной машины, правила можно составить любые, например: перезапуск haproxy)
- haproxy (можно адаптировать роли из https://galaxy.ansible.com/list#/roles, haproxy будет использоваться в качестве балансировщика для веб-иинтерфейса 4.5.)

4.3.2 На выходе Ansible так же должен быть сформирован ini-файл, содержащий любой уникальный ID виртуалки, например Packer ID, следующего формата:

```
[general]
```

```
uniqueID=
```

4.4 ab-logstash

4.4.1 Создать следующие роли в Ansible (можно пользоваться репозиторием Galaxy):

- apt (добавление необходимых пакетов, обновление из репозиториев установленных по умолчанию пакетов)
- ntp (обновить время, настроить синхронизацию времени по cron 1 раз в сутки с любого общедоступного сервера времени)
- monit (установить и настроить monit для само-мониторинга виртуальной машины)
- java (smola.java role (https://galaxy.ansible.com/list#/roles/1209) для установки openjdk-7-jdk)
- logstash (https://www.elastic.co/downloads/logstash без локального syslog и web интерфейса)
- elasticsearch (https://www.elastic.co/downloads/elasticsearch), со следующими опциями:
  - установленный плагин royrusso/elasticsearch-HQ
  - установленный плагин mobz/elasticsearch-head
  - установленный плагин elasticsearch/elasticsearch-cloud-aws/2.4.2 (данной версии)
  - имя кластера "logstash"
  - CORS enabled

4.4.2 На выходе Ansible так же должен быть сформирован ini-файл, содержащий любой уникальный ID виртуалки, например Packer ID

4.5 ab-webui

4.5.1 Создать следующие роли в Ansible (можно пользоваться репозиторием Galaxy):

- apt (добавление необходимых пакетов, обновление из репозиториев установленных по умолчанию пакетов)
- ntp (обнвоить время, настроить синхронизацию времени по cron 1 раз в сутки с любого общедоступного сервера времени)
- monit (установить и настроить monit для само-мониторинга виртуальной машины)
- rsyslog (для трансляции любых стандартных системных логов)
- kibana 3.x (нужна версия 3.x!)
- nginx (настроить таким образом, чтобы по запросу к сайту по умолчанию отдавался интерфейс kibana, важно: т.к. kibana 3.x состоит из статических файлов, то не нужно устанавливать интерпретаторы, все должно работать только со статическим контентом)

4.5.2 На выходе Ansible так же должен быть сформирован ini-файл, содержащий любой уникальный ID виртуалки, например Packer ID

4.6 Настроить конфигурации следующим образом:

- logstash должен собирать логи с rsyslog на ВМ WebUI
- kibana должен предоставлять интерфейс для просмотра логов из logstash
- доступ к kibana должен быть возможен через ВМ haproxy

5. Linux

Материалы и литература

- описание дистрибутивов:

  Какой linux выбрать?

- начальная установка

  Разметка жёсткого диска при установке linux

- работа с менеджером пакетов aptitude, dpkg

  Работа с пакетами при помощи dpkg

  Aptitude

  AptPreferences

  (справочник): https://debian-handbook.info/browse/ru-RU/stable/apt.html

  (справочник): https://www.debian.org/doc/obsolete.ru.html

- GNU core utils

  https://ru.wikipedia.org/wiki/GNU_Coreutils

  https://github.com/uran1980/web-dev-blog/blob/master/Linux/linux-commands.md

  (справочник): https://www.gnu.org/software/coreutils/manual/coreutils.pdf (все читать не нужно, но можно активно обращаться как к справочнику в процессе разработки скриптов). Что нужно знать: cat, head, wc, sort, uniq, ls, cp, dd, mv, rm, mkdir, chown, chmod, touch, du, df, echo, tee, pwd, date, hostname

- bash

  Chapter 1. Bash and Bash scripts

  Chapter 2. Bash and Bash scripts

  Chapter 3. Bash and Bash scripts

  Chapter 4. Bash and Bash scripts

  Chapter 5. Bash and Bash scripts

  sed + awk - можно изучать из этих статей или на примерах в Интернете.

  Искусство командной строки

- curl или wget

  wget - руководство GNU Wget

  Wget - насос для Интернета

- network utilities

LINUX: КОМАНДЫ ДЛЯ НАСТРОЙКИ СЕТИ ИЗ КОНСОЛИ

Замечание к практическому заданию

При выполнении заданий из данного раздела, пожалуйста пользуйтесь следующими ресурсами:
https://explainshell.com/ - поможет вам понять что означает команда или набор команд в скрипте.

https://www.shellcheck.net/ - поможет проверить ваши скрипты на наличие багов и ошибок.

Практическое задание

1.1. Написать скрипт определяющий запущен ли скрипт пользователем root.
1.2. Написать скрипт который после запуска ожидает ввод строи от пользователя, а затем выводит ее в стандартный вывод. Используйте команду read.
1.3. Написать скрипт который после запуска ожидает ввод строи от пользователя, а затем выводит ее в стандартный вывод, но ограничить время ожидания пользовательского ввода 5 секундами. В случае если пользователь не успел ввести строку, остановить программу и напечатать в стандартный вывод сообщение об этом.
1.4. Напишите скрипт, возвращающий
1.4.1. количество дней, прошедших с начала года
1.4.2. количество секунд, прошедших с начала "эпохи UNIX".
1.5. Написать скрипт, который выводит в стандартный вывод все аргументы который были ему переданы. Используйте команду shift.
1.6. Напишите скрипт который ожидает пользовательского ввода одного символа с клавиатуры, проверяет что символ был введен один. Далее скрипт определяет что за символ был введен - буква в нижнем регистре, буква в верхнем регистре, цифра, знак пунктуации, пробел или что-то другое. Испольйте команду case.
1.7. Написать скрипт преобразующий метры в мили. В качестве входящего аргумента должна быть цифра - метры. В стандартный вывод вывести количество миль.
1.8. Напишите скрипт который будет проводить симуляцию 700 бросков 6 гранного кубика. Вывод должен быть в следующем формате:

```
echo "единиц   =   $ones"
echo "двоек    =   $twos"
echo "троек    =   $threes"
echo "четверок =   $fours"
echo "пятерок  =   $fives"
echo "шестерок =   $sixes"
```

1.2.1. Написать скрипт, который рандомно сгенерирует на диске структуру файлов и папок. Входным параметром будут: директория, в которой нужно генерировать данные, глубину вложенности поддиректорий, максимальный размер файлов, которые будут создаваться в этих директориях и максимальное количество итераций по созданию объектов в директории (т.е. общее количество файлов и директорий, которые будут

созданы). Максимальная длина имени - 8 символов, содержимое файлов - рандом или нули. Проверку на доступное место делать не нужно.

1.2.2. Написать скрипт, который принимает 2 входных параметра: source_dir и destination_dir - две директории на диске, сравнивает, что они ни одна из них не является родительской для другой, у них не совпадают имена и т.п. Далее скрипт должен подсчитать место, занимаемое source_dir, и место, доступное на диске, где находится директория destination_dir, если есть доступное место - скопировать (с наследованием всех атрибутов: времени создания, доступа и модификации, владельца и т.п.), если места нет - вывести предупреждение на экран и предложить пользователю продолжить (C или Y) или прервать (N или A).

1.2.3. Модернизировать скрипт, чтобы он упаковывал данные из source_dir и складывал в destination_dir. При этом в самом начале работы скрипта нужно предусмотреть диалог с пользователем: предложить дописывать текущую дату и время в формате YYMMDD_HHSS.gz к имени файла или выполнять ротацию архивов, т.е. дописывал порядковый номер архива (0.gz, 1.gz, 2.gz). В случае выбора второго варианта - так же нужно запросить максимальное количество копий, при этом номер 0 присваивается самому последнему созданному архиву, а номер N-1 - самому старому, на каждой итерации архив с номером N - удаляется.

1.2.4. Модернизировать скрипт, чтобы все сообщения об ошибках подавлялись, а стандартные - выводились на консоль + записывались в лог-файл в текущей директории с именем out_YYMMDD_HHSS.log. Т.к. сообщения об ошибках подавляются, то в случае возникновения ошибок в процессе работы скрипта (проблемы с пермишенами, нехватка места, отсутствие какой-либо утилиты и т.п.) необходимо в конце работы скрипта вывести сообщение в консоль красным цветом "Warning: X error(s) occurred!", где X - количество перехваченных ошибок или количество вызовов команд, которые повлекли ошибки (как удобнее, на выбор).

1.3.1. Написать скрипт, который бы выводил на экран наиболее выгодный курс валют конвертации USD->BYR и EUR->BYR. Способ работы скрипта: либо передавать входным параметром "USD" или "EUR" и выводить курс, либо, если входных параметров нет - а консольном диалоге запросить на выбор желаемый курс. Получать можно с любого сайта (в идеале - с finance.tut.by).

1.3.2. Написать скрипт, который мог бы пройти авторизацию на Яндексе, сохранить сессию в cookie-файл, и показать кол-во писем в папке Inbox.

6. Amazon

Материалы и литература

1. https://aws.amazon.com/training/intro_series/ - вводные демо-видео по основным сервисам.
2. https://qwiklabs.com/ - Amazon рекомендует этот источник как официальный сервис для самообучения.
3. Architecting for The Cloud: Best Practices – общее описание сервисов Amazon и для чего их следует использовать.
4. AWS Best Practices: five key approaches to get you started – Рекомендации по использованию ключей
5. https://aws.amazon.com/products/security/
6. http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
7. http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3.html
8. http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_docker_ecs.html
9. http://codepany.com/blog/rails-5-and-docker-puma-nginx/
10. http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_Ruby.html
11. https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html
12. https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_GetStarted_EC2.html
13. FAQ разделы для каждого конкретного сервиса

Практические задания

Часть 1 - Основы AWS

1. Создать выделенную сеть Amazon Virtual Private Cloud с тремя подсетями, как минимум в двух разных зонах (например, обычно обозначаются как us-west-1a, us-west-1c). Две подсети (расположенные в разных зонах) должны быть публичными. Третья подсеть – приватная, ограничить доступ на ACL.
2. Создать Security Group (назовем ее web-sg):
   - Разрешить входящий SSH трафик только со своего IP (или доверенных IPs).
   - Разрешить входящий HTTP/HTTPS трафик со своего IP (или доверенных IPs).
   - Разрешить весь исходящий трафик во все 3 подсети.
   - Остальной трафик запретить.
3. Сгенерировать собственный RSA ключ (Key Pairs) для использования в дальнейшем при создании инстансов (необходимо для подключения по SSH).
4. EC2:
   a. Создать один t2.micro инстанс в созданной в п 6.1 VPC и одной из публичных подсетей. Использовать Security Group из п 6.1.1.
   b. Создать второй t2.micro инстанс в созданной в п 6.1 VPC и второй публичной подсети. Использовать Security Group из п 6.1.1.
   c. На оба инстанса установить Nginx и создать простую страницу-заглушку (index.html) на 80-ом порту.
5. ELB:
   a. Создать один ELB с поддержкой созданных Availability Zones.
   b. Разрешить HTTP трафик на ELB с любого IP адреса.
   c. Добавить в ELB оба инстанса.
   d. Настроить Health Check на протокол HTTP, порт 80, страница index.html с минимальными интервалами проверки.

e. Обновить security group созданную в пункте 6.1.1, так чтобы доступ по http/https был возможен только с ELB.
f. Принудительно остановить веб-сервер на одном из инстансов и проверить доступность сайта.

6. RDS:
   a. Создать инстанс PostgreSQL в выделенной VPC и приватной подсети с типом хранилища как General Purpose и объёмом в 20 Гб. Использовать Security Group из п 6.1.2.
   b. Разрешить входящий трафик только от web-sg. Как результат должны продемонстрировать возможность подключения к RDS как минимум с двух исходных точек (серверов)
7. ElastiCache:
   a. Создать один инстанс ElastiCache (Redis) в выделенной VPC.
   b. Разрешить трафик только внутри выделенной VPC. Как результат: должны продемонстрировать возможность подключения к Redis как минимум с двух исходных точек (серверов)
   c. Создать один инстанс ElastiCache (Memcached) в выделенной VPC.
   d. Разрешить трафик только от серверов созданных в пункте 5.3. Как результат: должны продемонстрировать возможность подключения к Memcached как минимум с двух исходных точек (серверов)
8. Создать CloudFront Distribution с параметрами по умолчанию.
   a. Сгенерировать 100 небольших файлов (< 512 Kb) и заполнить ими созданный бакет в S3. К файлам сторонние лица не должны иметь доступ.
   b. Настроить политику хранения объектов в данном бакете S3 следующим образом:
   - по истечении 30 дней – отправлять объекты в Glacier;
   - после 6 месяцев хранения – полностью удалять с Glacier.
9. *На одном из серверов созданных в пункте 6.3. подготовьте скрипт (с использование aws cli) для загрузки/выгрузки/удаления файлов в S3 бакете созданном в пункте 6.7.1 в соответствии с best practice и наибольшим уровнем безопасности окружения.
10. *Заменить инстансы (ELB должно будет ссылаться на новые инстансы созданные через autoscaling group) созданные в пункте 6.3 на autoscalinng group, со следующими правилами:
    - Если CPU Utilization > 70%, то добавить инстанс
    - Если CPU Utilization < 15%, то удалить инстанс
    - Минимальное количество запущенных инстансов = 1
    - Максимальное количество запущенных инстансов = 4
    - Autoscaling group должна разворачивать инстанс вместе с установленным nginx и страницей заглушкой из пункта 6.3

Часть 2 — Работа с контейнерами

11. Установите и настройте EB CLI
    a. Создайте приложение и окружение, используя Dockerrun.aws.json v2 создайте 2 контейнера:
    - Nginx

- Rails (можно использовать приложение на другом языке программирования, который поддерживает EB, но следующие подпункты обязательны)

b. Установите на rails порт 3000 и настройте nginx на фронт
c. Создайте простое Rails приложение выводящее строку "Hello world"
d. Проверьте используя URL: <имя_приложения>.<регион>.elasticbeanstalk.com работу beanstalk, вы должны увидеть содержимое index.html
e. Используя созданные в 6.9.3 контейнеры, настройте контейнер balancer на балансировку нагрузки round-robin между nginx контейнером beanstalk и локальным контейнером web

12. *Создать аналогичную пункту 6.9.1 структуру средствами ECS
    a. Настроить на ELB балансировку между инстансами созданными в пункте 6.3 и сайтом в ECS созданным в пункте 6.10
13. **Настроить EKS кластер в полном соответствии с пунктом 6.10

7. CloudFormation and Terraform. IaaC (Infrastructure as a Code).

Материалы и литература

1. Best Practices for Creating Amazon CloudFormation Templates – рекомендации по использованию CloudFormation
2. AWS CloudFormation (UserGuide) — официальная документация по AWS CF
3. Hashicorp Terraform - Getting started-AWS
4. Terraform Registry

Практические задания:

1. Подготовить шаблон CloudFormation используя Nested Stacks для автоматизации создания ресурсов, знакомство с которыми было в части «Основы AWS».
2. Подготовить шаблон Terraform для автоматизации создания ресурсов, знакомство с которыми было в части «Основы AWS».

Задание на отлично: Создать все ресурсы, включая автоматизацию создания сайта из первой части.

Создание CloudFront и S3 можно не использовать в связи с продолжительным временем создания Cloudfront Distribution.

8. Azure

Материалы и литература

Материалы и литература будут выданы на практическом занятии.

Практические задания

1. Вам необходимо развернуть две виртуальные машины Azure с именами VMLU01 и VMLU02 на основе образа Ubuntu. Развертывание должно соответствовать следующим требованиям: Обеспечение SLA 99,95%.Использование managed disks.
2. Вы планируете забэкапить файлы и документы с on-premise Windows file server в хранилище Azure. Бэкап файлы будут храниться в виде блобов.
   Вам необходимо создать storage account с именем CorpStorage01. Решение должно соответствовать следующим требованиям:
   - Убедитесь, что документы доступны через drive mapping с виртуальных машин Azure под управлением Windows Server.
   - Обеспечьте максимально возможное redundancy документов.
   - Минимизируйте затраты на storage account.

3. Вы планируете развернуть Application Gateway с именем AppGw01 для балансировки нагрузки внутреннего IP-трафика на виртуальные машины Azure, подключенных к subnet0.
   Вам необходимо настроить виртуальную сеть с именем VNET01 для поддержки Application Gateway.
4. Вы планируете разместить несколько защищенных веб-сайтов на Web01. Вам необходимо разрешить HTTPS через TCP-порт 443 на Web01 и запретить HTTP через TCP-порт 80 на Web01.
5. Вам нужно создать веб-приложение с именем WabApp01, которое можно горизонтально скалировать. Решение должно использовать самый низкий возможный ценовой уровень app Service Plan.

9. Раздел со звездочкой: сборка сайтов в AWS

Данная задача будет приниматься как курсовой проект.

В случае невыполнения данной задачи в качестве курсового проекта будут приниматься задачи из других разделов (будет проверяться процент выполнения заданий из каждых предыдущих разделов.

Условия:

1. Задачи должны выполняться на Centos или Amazon Linux
2. Необходимо разобраться с установкой Apache + PHP 7.3 + MySQL + NPM, если получится - автоматизировать этот процесс (достаточно все оформить в bash, ansible)
3. Зарегистрировать бесплатную учетную запись в AWS
4. Разобраться со стоимостью сервисов в AWS и все дальнейшие действия делать только на бесплатном окружении!
5. Создать новый Инстанс, автоматически установить на нем софт из п.2, сохранить как AMI и удалить.
6. Используя Cloudfront:
    a. Запустить инстанс, с предустановленным ПО из AMI из п.5.
    b. Создать Application load balancer, добавить маршрутизацию на сервер из п.5
    c. Входные параметры Cloudformation:
        i. Размер инстанса (выпадающий список)
        ii. SSH Ключ
        iii. Использование публичного IP адреса (true/false)
        iv. VPC, где будет размещен инстанс
        v. Имя инстанса, которое будет задано как тэг Name.
    d. Выходные параметры:
        i. Публичный IP инстанса
        ii. DNS, полученный в Load Balancer
7. На сервере сделать два сайта (через виртуальную директорию, или через хосты - на усмотрение)
    a. 1й сайт: установить самый Wordpress без дополнительной кофигурации
    b. 2й сайт: установить статический сайт https://github.com/gatsbyjs/gatsby-starter-hello-world

    Оба сайта должны работать через балансировщик

В качестве успешно выполненного задания необходимо показать:

a. Скрипт по конфигурированию сервере (установке необходимого софта из п.2) – bash или playbook
b. Шаблон cloudformation для разворачивания окружения
c. Сайт на WP
d. Сайт на gatsby

Terms of use

Данный материал может распространяться без каких-либо ограничений. Если вы хотите его использовать или дополнить – пожалуйста, просто дайте знать об этом по адресу devops.mentors@itransition.com