**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

**Факультет программной инженерии и компьютерной техники (ПИКТ)**

**Направление подготовки (специальность) – 09.04.04 (Нейротехнологии и программная инженерия)**

**КУРСОВАЯ РАБОТА**

по дисциплине «Методология разработки программного обеспечения DevOps»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тема: | Исследование возможностей VirtualBox для построения и эксплуатации DevOps методов | | | | | |
| Обучающийся: | | Бубенцова А. О. | |  | P42222 |  |
|  | | **(Фамилия И.О.)** | |  | **(номер группы)** |  |
| Руководитель: | | Ильин Д. Г., старший инженер-программист, АО «Нэксайн» | | | | |
|  | | | **(Фамилия И.О., должность и место работы)** | | | |

|  |  |
| --- | --- |
| Работа защищена с оценкой |  |
| Подписи членов комиссии |  |
|  | (подпись) |
|  | (подпись) |
|  | (подпись) |
| Дата |  |

Санкт-Петербург, 2022

Содержание

[Введение 3](#_Toc122536936)

[1 Анализ предметной области 5](#_Toc122536937)

[1.1 Jenkins 5](#_Toc122536938)

[1.2 Ansible 6](#_Toc122536939)

[1.3 Веб-серверы Apache и Nginx 7](#_Toc122536940)

[2 Практическая часть 9](#_Toc122536941)

[2.1 Конфигурация web-приложения 9](#_Toc122536942)

[2.2 Создание Ansible playbooks 9](#_Toc122536943)

[2.3 Создание сборки в Jenkins 15](#_Toc122536944)

[2.5 Обеспечение доступа к web-приложению и проверка работы 18](#_Toc122536945)

[Заключение 20](#_Toc122536946)

[Литература 21](#_Toc122536947)

[Задание на курсовую работу 22](#_Toc122536948)

# Введение

С развитием технологических компаний и IT продуктов, с увеличением степени интегрирования и применения их в повседневной жизни возникла проблема рассогласования рабочих процессов, при которой нормальная работа программного продукта нарушена из-за функционального и организационного разделения департаментов разработки, внедрения и сопровождения [8]. У разработчиков и специалистов по эксплуатации продукта часто бывают разные и даже противоречащие друг другу цели, руководители подразделений и ключевые показатели эффективности. В связи с чем встал вопрос о поиске решения, способного нивелировать конфликт интересов, чем впоследствии и стал DevOps [4].

DevOps – это совместная и междисциплинарная организационная работа по автоматизации непрерывной доставки новых обновлений программного обеспечения, c гарантией корректности и надежности [5]. DevOps объединяет два мира разработки и эксплуатации, используя автоматизированную разработку, развертывание и мониторинг инфраструктуры. Это организационный сдвиг, при котором вместо распределенных изолированных групп, выполняющих функции по отдельности, кросс-функциональные команды работают над непрерывными поставками функциональных возможностей. Такой подход помогает быстрее и непрерывно создавать ценность, уменьшая проблемы из-за недопонимания между членами команды и ускоряя решение проблем.

DevOps означает сдвиг культуры в сторону сотрудничества между разработкой, обеспечением качества и операциями. Вместо того чтобы придерживаться высокоразвитых концепций процессов, которые никогда не изменятся, организации настраивают непрерывную доставку с небольшими обновлениями. DevOps может применяться к самым разным моделям доставки, но должен быть адаптирован к среде и архитектуре продукта. Не все продукты обеспечивают непрерывную доставку — например, в системах, критически важных для безопасности. Тем не менее, даже в таких ограниченных условиях обновления могут быть спланированы и доставлены быстро и надежно, как показывает недавняя эволюция автомобильного программного обеспечения в прямом эфире. Помимо высокозащищенной облачной доставки, такие модели доставки требуют специальной архитектуры и аппаратных изменений [3].

Актуальность исследования возможностей VirtualBox для построения и эксплуатации DevOps методов обусловлена тем, что DevOps практики внедряются повсеместно и являются одним из основополагающих подходов в разработке программного обеспечения, поскольку существует тенденция к расширению возможностей команд, их взаимодействию и сотрудничеству, а также автоматизации технологий.

Целью данной работы является получение теоретических сведений и практических навыков в использовании DevOps практик на примере работы с виртуальными машинами посредством VirtualBox, а именно реализация автоматизации приложения для балансировки доступа к web-ресурсу.

Задачами на курсовую работу являются:

1. изучить предметную область и проанализировать рекомендованную литературу;
2. выбрать web-приложения, отображающего уникальный index.html.
3. создать Ansible playbook'и для развертывания необходимых пакетов для web-приложения;
4. создать сборку в Jenkins;
5. разместить playbook'и и сборку в Git репозитории;
6. обеспечить доступ к web-приложению внутри и проверить его работу;
7. составить отчет о проделанной работе.

# 1 Анализ предметной области

# 1.1 Jenkins

Jenkins – программная система, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения, устанавливается на сервере, на котором будет происходить центральная сборка. Jenkins реализован на Java, распространяется с открытым исходным кодом, для установки требуется также JDK и Java.

Непрерывная интеграция (continuous integration) – это практика разработки, при которой разработчики c определенной частотой (несколько раз в день) объединяют изменения своего кода в общий репозиторий. Данная практика позволяет обнаруживать и разрешать конфликты на ранней стадии, избегать необходимости координировать крупномасштабные изменения кода и обеспечивать стабильность кодовой базы. CI – это ключевая практика для гибких команд разработчиков.

За уровнем непрерывной интеграции следует непрерывная доставка (continuous delivery). Непрерывная доставка – это практика разработки программного обеспечения, при которой изменения кода автоматически создаются, тестируются и развертываются в рабочей среде. CD позволяет автоматизировать весь цикл тестирования программного обеспечения и уменьшить объем ручной работы, что приводит к снижению, быстрому обнаружению и устранению ошибок, увеличению продуктивности разработки, быстрой доставки обновлений пользователям.

Использование вышеописанных практик в совокупности позволяет автоматизировать жизненный цикл DevOps, поскольку они направлены на борьбу с противоречиями между частыми обновлениями программного продукта и стабильностью при его эксплуатации.

CI/CD-конвейеры используются в случаях, когда необходимо часто вносить изменения в приложения с надежным процессом поставки. Помимо стандартизации сборки, разработки тестов и автоматизации развертываний практики CI/CD позволяют реализовать целостный производственный процесс по развертыванию изменений кода. Внедрение CI/CD позволяет разработчикам сосредоточиться на улучшении кода и реализации изменений в нем и уделять меньше времени и сил развертыванию этого кода.

# 1.2 Ansible

Ansible — система управления конфигурациями, написанная на языке программирования Python, с использованием декларативного языка разметки для описания конфигураций. Для установки требуются следующие модули: python-yaml, python-jinja2; распространяется по лицензии открытого исходного кода. Ansible применяется для автоматизации настройки и развёртывания программного обеспечения.

Ansible автоматизирует задачи на управляемых узлах, используя список или группу списков этих узлов, называемую inventory. Файл inventory (расположение – /etc/ansible/hosts) определяет хосты и группы хостов, на которых работают команды, модули и задачи в playbook’е. Как уже было отмечено, в файле inventory могут быть записаны не только одиночные хосты, но и группы хостов, что впоследствии позволяет автоматизировать запуск playbook’ов для конкретной группы. После определения хостов внутри файла inventory становится возможным использование шаблонов [6].

Необъемлемой частью работы с Ansible является использование упомянутых выше playbook’ов. Playbook представляет собой повторяемую, многократно используемую, простую систему управления конфигурацией и развертывания на нескольких хостах, которая позволяет автоматизировать разворачивание приложений, в т. ч. сложных. Playbook’и используются в случае, если требуется многократное (больше одного раза) выполнение задачи средствами Ansible. Playbook, загруженный в систему контроля версий, может быть использован для отправки новой конфигурации или подтверждения конфигурации удаленных систем [2].

Содержимое playbook’ов реализовывается в формате YAML (расширение .yml) с минимальным синтаксисом. Playbook выполняется по порядку сверху вниз, один за одним выполняя сценарии, из которых он состоит. Сценарий представляет собой список задач для конкретного хоста или группы хостов. Каждая задача связана с модулем, отвечающим за действие, и параметрами конфигурации. Playbook’и, содержащие несколько сценариев, могут организовать развертывание на нескольких хостах, которые в этих сценариях прописаны. Каждый сценарий должен содержать, как минимум, один узел, вызываемый по его псевдониму, и как минимум, одну задачу [2].

Playbook’и в Ansible запускаются командой   
ansible-playbook playbook.yml -i hosts.ini, где playbook.yml – имя файла, содержащего playbook’, hosts.ini – файл inventory.

# 1.3 Веб-серверы Apache и Nginx

Apache – программное обеспечение для размещения веб-сервера. Apache является безопасным, эффективным и расширяемым сервером; он предоставляет службы HTTP в соответствии с текущими стандартами HTTP. Среди преимуществ Apache следует выделить Apache гибкость, мощность и широкую распространенность. Он может быть расширен с помощью системы динамически загружаемых модулей и исполнять программы на большом количестве интерпретируемых языков программирования без использования внешнего программного обеспечения [1].

NGINX – это веб-сервер и почтовый прокси. Nginx эффективно потребляет ресурсы, и отзывчив под нагрузкой, а также, как уже было отмечено, предоставляет возможность использовать его и как веб-сервер, и как прокси.

Механизм обработки входящих сигналов у Nginx отличается упрощением обработки запроса пользователя путем декомпозиции этого запроса на ряд более мелких. В терминологии Nginx они получили название рабочее соединение. После обработки каждое соединение собирается в одном виртуальном контейнере, чтобы трансформироваться в единый первоначальный запрос, а после отправляется пользователю. Одно соединение может одновременно обрабатывать до 1024 запросов конечного пользователя.

Nginx используется:

* для обработки запросов с сайтов, где много статического неизменного контента;
* обслуживания серверов, на которые поступает много запросов одновременно;
* в качестве прокси, почтового сервера или для распределения нагрузки на серверную часть.

# 2 Практическая часть

# 2.1 Конфигурация web-приложения

Приложение состоит из следующих хостов:

* mng – хост, с которого будет осуществляться управление конфигурацией системы, должен содержать Ansible;
* lb – балансировщик нагрузки. Данный хост будет отвечать за распределением нагрузки между хостами apps1 и apps2 посредством Nginx;
* apps1 – сервер приложения, необходима установка Apache;
* apps2 – сервер приложения, наряду с apps1 будет являться субъектом балансировки нагрузки, необходим Apache.

Таблица 1 содержит ip-адреса всех существующих в рамках данной системы хостов.

Таблица 1 – ip-адреса хостов

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Public IP | Internal IP | Software |
| lb | 10.0.2.15 | 192.168.56.122 | Nginx |
| apps1 | 10.0.2.16 | 192.168.56.121 | Apache |
| apps2 | 10.0.2.17 | 192.168.56.123 | Apache |
| mng | 10.0.2.19 | 192.168.56.125 | Ansible |

# 2.2 Создание Ansible playbooks

Для того, чтобы создать и запустить playbook Ansible, необходимо создать файл inventory и прописать в нем хосты, с которыми планируется работа в playbook’ах. На рисунке 1 изображено содержимое файла /etc/ansible/hosts.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Файл inventory

Команда ansible all –list-hosts позволяет вывести все хосты, доступные Ansible, результат ее работы приведен на рисунке 2.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Список доступных хостов

Листинг 1 представляет собой содержимое файла install\_httpd.yml. Данный файл содержит сценарии Ansible для установки Apache на apps1 и apps2.

Листинг 1 – Playbook установки Apache

---

- hosts: apps1

tasks:

- name: Install httpd

yum: name=httpd state=present

- name: Starting httpd

service: name=httpd state=started

- hosts: apps2

tasks:

- name: Install httpd

yum: name=httpd state=present

- name: Starting httpd

service: name=httpd state=started

На рисунке 3 представлен результат выполнения playbook’а установки Apache на хосты apps1 и apps2.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 3 – Результат выполнения playbook’а install\_httpd.yml

Сценарий установки Nginx на хост lb представлен в листинге 2.

Листинг 2 – Playbook установки Nginx

---

- hosts: lb

tasks:

- name: Install nginx

yum: name=nginx state=present

- name: Starting nginx

service: name=nginx state=started

Результат выполнения playbook’а установки Nginx на хост lb – файл install\_nginx.yml – представлен на рисунке 4.

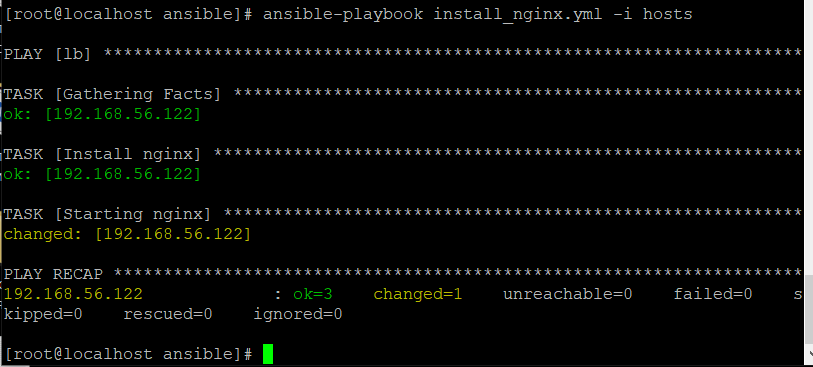


Рисунок 4 – Результат выполнения playbook’а install\_nginx.yml

После установки необходимо внести изменения в конфигурационный файл Nginx для внесения в него ip-адресов серверов приложений. Изменения, внесенные в файл nginx.conf, представлены в листинге 3.

Листинг 3 – Изменения в файле nginx.conf

upstream backend {

least\_conn;

server 192.168.56.121;

server 192.168.56.123;

}

location / {

proxy\_pass http://backend;

}

Данный playbook предусматривает хранение правильного конфигурационного файла Nginx в папке Ansible под именем new\_conf. Playbook изменения файла nginx.conf представлен в листинге 4.

Листинг 4 – Playbook изменения файла nginx.conf

---

- hosts: lb

tasks:

- name: remove config

file:

state: absent

path: "/etc/nginx/nginx.conf"

- copy:

src: "new\_conf"

dest: "/etc/nginx/nginx.conf"

- name: restart nginx

service: name=nginx state=restarted

Рисунок 5 содержит результат выполнения playbook’а внесения изменений в конфигурационный файл Nginx.

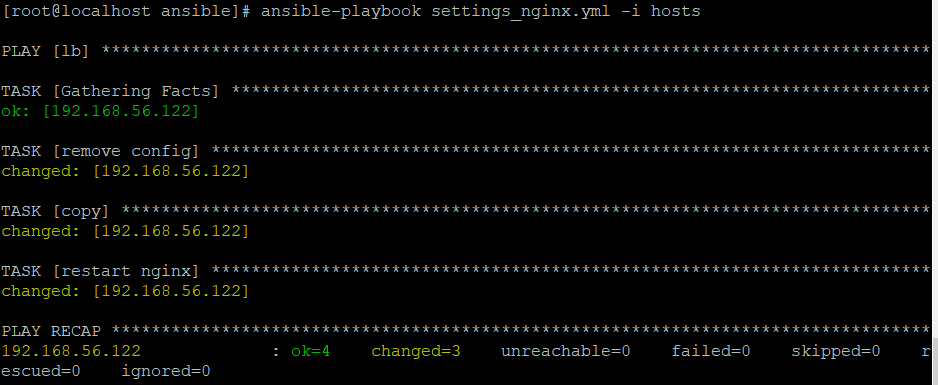


Рисунок 5 – Результат выполнения playbook’а settings\_nginx.yml

Playbook для создания приложений на хостах apps1 и apps2 содержится в файле index­­­\_html.yml и представлен в листинге 5. На хостах apps1 и apps2 необходимо создать страницу, которая будет отображаться в веб-браузере, index.html (расположение /var/www/html/index.html). Для демонстрации работы балансировщика нагрузки страницы для каждого хоста должны отличаться, например, содержать номер хоста apps.

Листинг 5 – Playbook создания приложений на хостах apps1 и apps2

---

- hosts: apps1

tasks:

- name: create index.html

file:

state: absent

path: "/var/www/html/index.html"

- copy:

src: "index\_html\_apps1.html"

dest: "/var/www/html/index.html"

- hosts: apps2

tasks:

- name: create index.html

file:

state: absent

path: "/var/www/html/index.html"

- copy:

src: "index\_html\_apps2.html"

dest: "/var/www/html/index.html"

На рисунке 6 отображен результат выполнения playbook’а index­­­\_html.yml.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 6 – Результат выполнения playbook’а index\_html.yml

Все созданные playbook’и необходимо выгрузить в репозиторий. В рамках данной работы в соответствии с заданием был использован Gitea, куда впоследствии сборка Jenkins будет обращаться за playbook’ами.

# 2.3 Создание сборки в Jenkins

Для реализации сборки необходимо установить JDK версии 11 или 17 и, собственно, сам Jenkins. Важно соблюдать версии зависимостей, в противном случае Jenkins не запустится. Рисунок 7 содержит версию Java в системе.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Проверка версии Java

После установки необходимых зависимостей становится возможным установить Jenkins. Рисунок 8 содержит результат вывода статуса Jenkins.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Статус Jenkins: активен

Для осуществления сборки необходимо далее перейти в интерфейс Jenkins, что продемонстрировано на рисунке 9.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Переход в интерфейс Jenkins

Следующий шаг – привязка репозитория, в котором находятся playbook’и, к сборке. Для этого необходимо приватный ключ из сгенерированной ранее пары внести в настройки Jenkins, а также в разделе Git Host Key Verification Configuration выбрать опцию Accept first connection.

Для реализации сборки необходимо создать Pipeline. Pipeline – это состояние плагинов, которое поддерживает внедрение и интеграцию конвейеров непрерывной доставки в Jenkins [7]. Другими словами, Pipeline можно охарактеризовать, как перечень заданий для реализации CI средствами Jenkins. На рисунке 10 представлен Pipeline сборки в рамках данной задачи.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Pipeline для запуска playbook’ов Ansible

В Pipeline необходимо указать репозиторий, а также созданный ранее ключ (приватный ключ из сгенерированной пары), а далее прописать команды, которые должны быть выполнены.

Результат выполнения сборки представлен на рисунке 11.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 11 – Результат выполнения сборки

Финальная часть вывода на консоль представлен на рисунке 12.

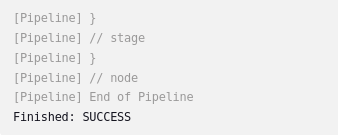


Рисунок 12 – Вывод на консоль

# 2.5 Обеспечение доступа к web-приложению и проверка работы

Доступ к разработанному приложению осуществляется по адресу хоста lb. При многократном обновлении страницы запрос перенаправляется на тот хост (apps1 или apps2), который является менее загруженным. На рисунке 13 продемонстрирована работа приложения.

|  |  |
| --- | --- |
|  |  |

Рисунок 13 – Результат работы приложения

Ansible playbook’и и Jenkins Pipeline, получившиеся в результате данной работы, находятся в репозитории GitHub https://github.com/anastasiabubencova/DevOps.

# Заключение

В рамках данной работы было разработано web-приложение, расположенное на нескольких хостах, а также была произведена автоматизация этого приложения следующей последовательностью этапов:

1. произведена балансировка доступа к web-приложению средствами Nginx;
2. созданы Ansible Playbooks для автоматизации развертывания приложений;
3. локально развернут репозиторий Gitea для хранения ploaybook’ов;
4. средствами Jenkins создана сборка для автоматизации выполнения разработанных Ansible playbooks;
5. протестировано web-приложение на предмет реализации балансировки нагрузки после выполнения автоматизированного развертывания.

В результате выполнения курсовой работы были сделаны следующие выводы:

* DevOps практики позволяют уменьшить ошибки, трудовые и временные затраты на этапе развертывания приложения;
* Ansible playbooks позволяют автоматизировать разворачивание приложений на группе хостах, когда требуется многократное выполнение одного и того же сценария;
* Jenkins предоставляет инструментарий для непрерывной интеграции, в рамках данной задачи позволил автоматизировать процесс запуска Ansible playbooks.

# Литература

* 1. Ansible playbooks // Ansible Documentation : [сайт]. – 2022.   
     – URL: https://docs.ansible.com/ansible/latest/playbook\_guide/  
     playbooks\_intro.html (дата обращения 18.12.2022).
  2. Building Ansible inventories // Ansible Documentation : [сайт]. – 2022.   
     – URL: https://docs.ansible.com/ansible/latest/inventory\_guide/  
     intro\_inventory.html (дата обращения 18.12.2022).
  3. Ebert C. DevOps / C. Elbert, G. Gallardo, J. Hernantes, N. Serrano // Ieee Software. – 2016. – №. 3. – P. 94-100.
  4. Garnett A. Apache vs Nginx: Practical Considerations / A. Garnett,   
     J. Ellingwood // DigitalOcean : [сайт]. – 2022. – URL: https://  
     www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations (дата обращения 18.12.2022).
  5. Edwards D. The history of DevOps / D. Edwards // IT Revolution : [сайт]. – 2012. – URL: https://itrevolution.com/articles/the-history-of-devops/ (дата обращения 18.12.2022).
  6. Leite L. A survey of DevOps concepts and challenges / L. Leite, C. Rocha, F. Kon, D. Milojicic, P. Meirelles // ACM Computing Surveys (CSUR). – 2019. – Т. 52. – №. 6. – P. 1-35.
  7. Pipeline // Jenkins User Documentation : [сайт]. – 2022. – URL: https://www.jenkins.io/doc/book/pipeline/
  8. Что такое DevOps? // Atlassian : [сайт]. – 2022. – URL: <https://ru.atlassian.com/devops> (дата обращения: 18.12.2022).

**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

**Факультет программной инженерии и компьютерной техники (ПИКТ)**

**Направление подготовки (специальность) – 09.04.04 (Нейротехнологии и программная инженерия)**

# ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Студенту** | Бубенцовой А.О. | |  | | **Группа** | P42222 |
|  | (фамилия и. о.) | |  |  | |  |
| **Руководитель** | | Ильин Д. Г., старший инженер-программист, АО «Нэксайн» | | | | |
|  | | (фамилия, и. о., место работы, должность) | | | | |

* 1. **Наименование темы:** Исследование возможностей VirtualBox для построения и эксплуатации DevOps методов.
  2. **Срок сдачи студентом законченной работы** 23.12.2022
  3. **Техническое задание и исходные данные к работе:** Развернуть виртуальную машину с операционной системой Linux Centos 8 Stream с неймингом, отключить систему контроля доступа selinux, межсетевой экран или брандмауэр firewall. Создать 4 linked клона данной виртуальной машины: lb, apps1, apps2, mng. Заполнить таблицу назначения статических ip адресов из трех подсетей для каждой виртуальной машины. На машину mng необходимо установить Ansible ad-hoc. Создать playbook’и: по установке Nginx и Httpd, настройке Nginx, созданию index.html с содержанием названия виртуальной машины (для машин apps1 и apps2). Установить на машину mng Gitea, создать проект, в котором необходимо разместить playbook’и, созданные ранее. Установить Jenkins, создать credential с данными пользователя из репозитория Gitea. Создать тестовую сборку в Jenkins, где источником кода является проект в Gitea. Выполнить старт сборки и добиться корректного вывода bash-скрипта. Создать сборки для выполнения Ansible ad-hoc playbook’ов на машинах: lb, apps1, apps2. Откатить виртуальные машины lb, apps1, apps2 и запустить сборки повторно для проверки работоспособности сборок. Проверить работоспособность балансировщика и web-сервисов путем перехода по локальному адресу хоста машины lb. При обновлении страницы должны выводиться разное содержание (apps1, apps2). Загрузить полученные Jenkins Pipeline и Ansible Playbooks в репозиторий на Github.
  4. **Содержание курсовой работы (перечень подлежащих разработке вопросов):** Анализ предметной области: сведения об используемых компонентах и практиках. Реализация автоматизации создания системы виртуальных машин для балансировки доступа к web-ресурсу посредством Nginx, Ansible Jenkins, Gitea. Создание Ansible playbook’ов по автоматизации установки и настройке ПО. Хранение разработанных playbook’ов в локальном репозитории Gitea. Создание сборок Jenkins для автоматизации запуска playbook’ов.
  5. **Перечень графического материала (с указанием обязательного материала):** Результатывыполнения playbook’ов в виде скриншотов, результаты работы сборки Jenkins в виде скриншотов.
  6. **Исходные материалы и пособия:** документация следующих компонентов: CentOS 8 Stream, Nginx, Apache. Ansible ad-hoc, Jenkins, Git, Gitea.

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата выдачи задания** |  | |  |
|  | |  | |
| Руководитель | |  | |
|  | | (подпись) | |
| Задание принял к | |  | |
| исполнению | |  | |
|  | | (подпись) | |