Введение

Поздравляем с завершением курса «DevOps. Docker»! Надеемся, он стал для вас интересным и полезным источником знаний. Теперь вы на финишной прямой.

При прохождении курса вы многое узнали о контейнеризации Docker:

- разобрались в теоретических основах и в том, как работает Docker;
- изучили особенности сборки и запуска контейнеров;
- познакомились с Docker Trust, Docker Compose и Docker Swarm.

Сейчас вам предстоит выполнить финальную работу, в которой вы сможете применить полученные знания и решить задачи, максимально приближенные к реальным.

Вы пройдёте путь специалиста, начинающего строить свою инфраструктуру.

Последовательно изучите каждую часть финальной работы. Периодически обращайтесь к актуальным для вас разделам в этом документе.

Будет интересно, приступим!

Тема финальной работы:

«Инфраструктура разработки на docker-контейнерах»

В финальной работе вам предстоит развернуть GitLab и Rocket Chat. Однако для удобства работы в такой инфраструктуре надо реализовать централизованное управление учётными записями пользователей. Проще говоря, в результате работы должно появиться три веб-интерфейса. В первом можно создать учётную запись пользователя, используя которую можно выполнять вход в GitLab и Rocket Chat.

Описание кейса

Вы работаете в начинающей фирме по разработке приложений для внешних заказчиков. У фирмы уже есть первый заказ, теперь предстоит нанять разработчиков для разработки приложения.

При этом пока решено не вкладываться в инфраструктуру разработки, а реализовать всё на бесплатных продуктах.

Для начала разработки требуется GitLab и любой чат, чтобы разработчикам было куда пушить код и где обсуждать баги. Но чат должен быть размещён на собственном сервере, чтобы минимизировать риски утечки данных.

Также руководитель организации с оптимизмом смотрит в будущее и верит, что это будет не единственный заказ. А значит, инфраструктура должна быть готова к регулярному принятию новых работников и предоставлению им необходимых доступов на основе централизованного управления. Следовательно, нужен каталог пользователей, в который будут смотреть остальные системы и предоставлять доступ для учётных записей в нём.

По понятным причинам развёртывание инфраструктуры разработки требуется осуществить как можно быстрее.

Вы изучили курс Docker и в своё время рассказали руководителю, насколько удобно и быстро можно с его помощью разворачивать новые сервисы.

Руководитель ставит вам задачу — продемонстрировать простоту и быстроту развёртывания инфраструктуры разработки при помощи Docker Engine и Docker Compose.

В качестве каталога пользователей вы выбираете OpenLDAP. Его сильные стороны — открытость протокола и широкая распространённость.

В качестве чата — Rocket Chat, так как он бесплатный и интегрируется с LDAP.

В работе DevOps-инженера именно интеграция разных систем играет ключевую роль. Сборка приложений тоже важна, но проблемы при сборке обычно решаются совместно с разработчиками. И именно процессы организации взаимодействия разных сервисов друг с другом занимают значительную часть времени.

Вам предстоит выстроить такое взаимодействие. Вы решаете не терять времени и сразу приступаете к работе.

Рекомендации по выполнению

Рекомендации по организации работы

- Распланируйте выполнение финальной работы по дням, как на реальной работе с задачами на спринт. Исходя из нашего опыта, продуктивнее выделять на финальную работу по 2-3 часа несколько дней в неделю, чем делать тот же объём за один подход. Лучше придерживаться такого графика и обязательно выделять время на отдых.
- Отмечайте свой прогресс по мере выполнения плана. Это полезно по нескольким причинам:
    - вы будете держать ритм — такой подход дисциплинирует;
    - сможете контролировать ситуацию: если что-то пойдёт не так, вы будете точно знать, в какой очерёдности выполняли задачи до этого.

Рекомендации к финальной работе

- Используйте IaC-подход. Работающая конфигурация = коммит в репозиторий.
- Если вы думаете, что всё сделали правильно, но система не работает так, как ожидалось, то возможно всего два варианта:
    - вы не всё сделали правильно;
    - на самом деле работает, но вы неправильно проверяете.
- Используйте идеи и наработки из интернета — так вы нарабатываете навык правильного поиска и расширяете кругозор. Однако всегда проверяйте, что вы понимаете, что вставляете. Не делайте это бездумно.

Ваши задачи

Задача 1. Развернуть LDAP-сервер

Вы начинаете с каталога пользователей на основе OpenLDAP.

LDAP (англ. Lightweight Directory Access Protocol) — легковесный протокол доступа к объектам каталога. В каталоге могут храниться пользователи, группы пользователей, почтовые аккаунты, компьютеры, принтеры и тому подобное — в общем, объекты любой инфраструктуры.

В частности, на корпоративном уровне в больших организациях он используется для доступа к объектам Microsoft Active Directory, в которой хранятся учётные записи пользователей организации. Из-за того, что LDAP широко распространён, многие программы имеют функции интеграции с LDAP прямо «из коробки».

Ваша инфраструктура не станет исключением. Вы будете создавать учётные записи пользователей в LDAP-каталоге.

В отличие от Active Directory, ваша компания решила не разворачивать отдельный сервер на базе OS Windows Server с закупкой лицензий и прочими проблемами, связанными с установкой обновлений и обслуживанием отдельного сервера. Вместо этого решили использовать легковесную программу OpenLDAP, которая поставляется в docker-контейнере.

Что нужно сделать

1. Разверните виртуальную Ubuntu Server 22.04.
2. Установите последние обновления, Docker Engine и Docker Compose Plugin.
3. Сделайте скриншоты команд docker version и docker compose version для проверки готовности сервера.
4. Создайте отдельную папку проекта финальной работы. Например, /opt/docker_final.
5. Создайте файл compose.yaml в папке проекта.
6. Добавьте первый сервис со следующими параметрами:
    a) имя сервиса: ldap;
    b) используемый образ: osixia/openldap;

c) переменные среды:

- LDAP_DOMAIN=mardeev.ru (здесь вместо mardeev укажите произвольное имя домена)*
- LDAP_ADMIN_PASSWORD=admin (здесь вместо admin укажите произвольный пароль);

d) монтируемые папки:

- ./ldap/data/slapd/database:/var/lib/ldap,
- ./ldap/data/slapd/config:/etc/ldap/slapd.d.

7. Сохраните файл и запустите проект командой docker compose up -d.
8. Выполните docker compose ps.
9. Сделайте скриншот того, что сервис запущен.

Советы и рекомендации

- В качестве дополнительного источника информации используйте открытые источники в интернете.
- Полное руководство по docker-openldap находится здесь.
- LDAP_DOMAIN — это имя домена организации. Обычно оно используется в качестве имени основного сайта компании, а также постфиксом в адресах электронной почты.
- В указанные пути можно монтировать и docker volumes. В этих папках будет находиться вся создаваемая информация. Так достигается сохранность данных при перезапуске контейнера.
- В следующем задании будет добавляться веб-интерфейс для администрирования LDAP-каталога, поэтому здесь никакие порты не публикуются.
- Старайтесь руководствоваться подходом IaC. Сразу создайте Git-репозиторий. Только не отправляйте туда папку с данными ./ldap.
- Пароли также старайтесь не отправлять в git. Переменную LDAP_ADMIN_PASSWORD лучше указать через ENV-файл.

Критерии готовности

- Развёрнута виртуальная Ubuntu Server.
- Установлен Docker Engine и Docker Compose Plugin.
- Запущен сервис LDAP.

Артефакты блока

Скриншоты результата выполнения команд:

- docker version,
- docker compose version,
- docker compose ps.

Задача 2. Добавить веб-интерфейс для созданного сервиса LDAP

Отлично! LDAP поднят, но в него надо как-то входить. Конечно, существуют консольные утилиты из пакета ldap-utils, но для удобства есть подходящий веб-интерфейс.

Сейчас вы как раз займётесь его подключением к существующему сервису LDAP.

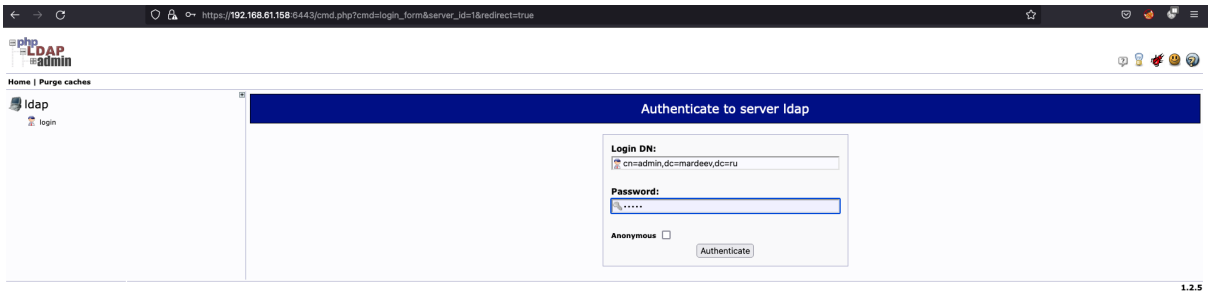
Что нужно сделать

1. Добавьте сервис в существующий compose.yaml со следующими параметрами:
   a) имя сервиса: ldap_admin;
   b) используемый образ: osixia/phpldapadmin;
   c) опубликованные порты:
      - 6443:443;
   d) переменные среды:
      - PHPLDAPADMIN_LDAP_HOSTS=ldap.
2. Снова выполните docker compose up -d и docker compose ps. Сделайте скриншот.
3. Зайдите браузером на порт 6443 сервера по протоколу https.
4. Выполните вход в консоль администрирования. Сделайте скриншот.

Советы и рекомендации

- В качестве дополнительного источника информации используйте открытые источники в интернете.

- Полное руководство по docker-phpLDAPadmin находится по ссылке.
- При первом входе в веб-интерфейс администрирования появится сообщение о недоверенном сертификате. Проигнорируйте предупреждение. Для Firefox, например, нажмите Advanced → Accept the risk and continue.
- Для входа в веб-интерфейс в качестве логина используйте следующую строку: cn=admin,dc=mardeev,dc=ru (без пробелов). mardeev замените на имя вашего домена.



- В качестве пароля используйте пароль, указанный в переменной LDAP_ADMIN_PASSWORD в задании 1.

Критерии готовности

- Запущены два сервиса: ldap и ldap_admin.
- Выполнен вход в веб-интерфейс ldap_admin.

Артефакты блока

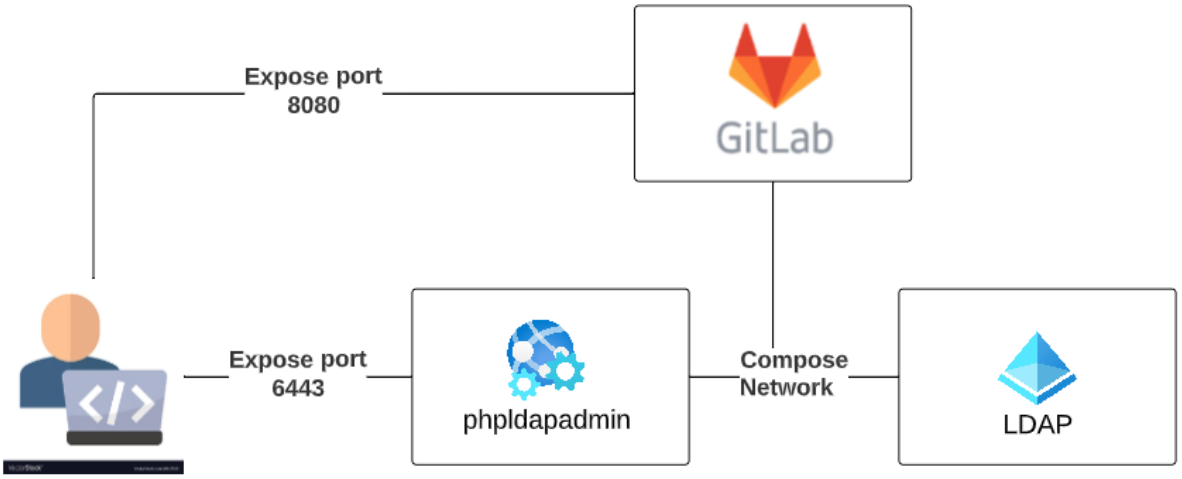
- Скриншот результата выполнения команды docker compose ps.
- Скриншот выполненного входа в веб-интерфейс phpLDAPadmin.

Задача 3. Добавить GitLab

Плацдарм для централизованного управления учётными записями пользователей готов. Но вы помните, что главная задача — создание простейшей инфраструктуры разработки. GitLab здесь подойдёт лучше всего.

К счастью, он поставляется также в виде образа контейнера Docker.

Вы решаете добавить его в проект.



Что нужно сделать

1. Добавьте сервис в существующий compose.yaml со следующими параметрами:
    a) имя сервиса: gitlab;
    b) используемый образ: gitlab/gitlab-ce;
    c) опубликованные порты:
        - 8080:80;
    d) примонтированные тома:
        - ./gitlab/gitlab_home/config:/etc/gitlab;
        - ./gitlab/gitlab_home/logs:/var/log/gitlab;
        - ./gitlab/gitlab_home/data:/var/opt/gitlab.
2. Выполните docker compose up -d и docker compose ps. Сделайте скриншот.
3. Войдите браузером в порт 8080 сервера по протоколу http.
4. Выполните первый вход при помощи учётной записи root. Сделайте скриншот.

Советы и рекомендации

- В качестве дополнительного источника информации используйте открытые источники в интернете.
- Полное руководство по установке GitLab находится по ссылке.
- Полная загрузка GitLab может занять продолжительное время — до 10 минут. Интерфейс входа во время загрузки не будет доступен.
- При входе используйте пароль, полученный командой sudo docker compose exec -it gitlab grep 'Password:' /etc/gitlab/initial_root_password.

Критерии готовности

- Развёрнуты сервисы ldap, ldap_admin и gitlab.
- Выполнен вход в GitLab.

Артефакты блока

- Скриншот результата выполнения команды docker compose ps.
- Скриншот выполненного входа в GitLab.

Задача 4. Интеграция GitLab и LDAP

Итак, теперь у вас есть централизованное управление пользователями и GitLab. Пора сделать так, чтобы в GitLab можно было входить с учётными записями из LDAP-каталога.

Для интеграции используется файл ./gitlab/gitlab_home/config/gitlab.rb.

Вы не теряете времени и приступаете к задаче.

Что нужно сделать

1. Найдите в файле ./gitlab/gitlab_home/config/gitlab.rb раздел ### LDAP Settings.
2. Добавьте в него следующую конфигурацию:

```
          gitlab_rails['ldap_enabled'] = true
###! **remember to close this block with 'EOS' below**
 gitlab_rails['ldap_servers'] = YAML.load <<-'EOS'
   main: # 'main' is the GitLab 'provider ID' of this LDAP server
     label: 'LDAP'
     host: 'ldap'
     port: 389
     uid: 'uid'
     bind_dn: 'cn=admin,dc=mardeev,dc=ru'
     password: 'admin'
     encryption: 'plain' # "start_tls" or "simple_tls" or "plain"
     active_directory: false
     allow_username_or_email_login: true
     lowercase_usernames: true
     block_auto_created_users: false
     base: 'dc=mardeev,dc=ru'
 EOS
```

3. Замените значения на релевантные LDAP-каталогу вашего домена в добавляемых параметрах bind_dn, password и base.
4. Сохраните файл и выполните перезагрузку GitLab командой docker compose restart gitlab.
5. Дождитесь перезагрузки сервера и убедитесь, что в окне входа появилась опция LDAP. Сделайте скриншот.

Советы и рекомендации

- Обратите внимание на отступы. Добавленная конфигурация — это YAML-документ.
- Полное руководство по интеграции GitLab с LDAP находится по ссылке.
- В качестве дополнительного источника информации используйте открытые источники в интернете.

Критерии готовности

- Корректно настроена интеграция GitLab с LDAP.

Артефакты блока

- Скриншот окна входа в GitLab с двумя входами: Standard и LDAP.

Задача 5. Добавление пользователя в LDAP. Вход в GitLab

Теперь надо проверить интеграцию.

Для компании важна интеграция всех систем с единым каталогом пользователей. Если создавать пользователей везде отдельно (в GitLab свой логин и пароль, в чате — свой и так далее), то:

- сотрудникам будет неудобно,
- они станут чаще забывать пароли,
- как следствие — больше времени тратится на восстановление аккаунтов, а компания, соответственно, теряет деньги.

Именно поэтому было принято решение не сразу разворачивать GitLab, а вначале создать каталог пользователей на основе OpenLDAP.

Что нужно сделать

1. Создайте учётную запись в LDAP-каталоге в соответствии с инструкцией. Сделайте скриншот.
2. Выполните вход в GitLab с созданной учётной записью. Сделайте скриншот.

Советы и рекомендации

- При создании учётной записи обязательно должно быть заполнено поле email. Без него GitLab не пустит.

- Для входа в GitLab в поле Username можно использовать uid пользователя либо email пользователя.
- UID пользователя можно посмотреть в веб-интерфейсе ldap_admin. Надо нажать на созданного пользователя и нажать на кнопку Show Internal Attributes. Он будет в поле User Name.

Критерии готовности

- В LDAP-каталоге создан пользователь.
- В GitLab выполняется вход этим пользователем.

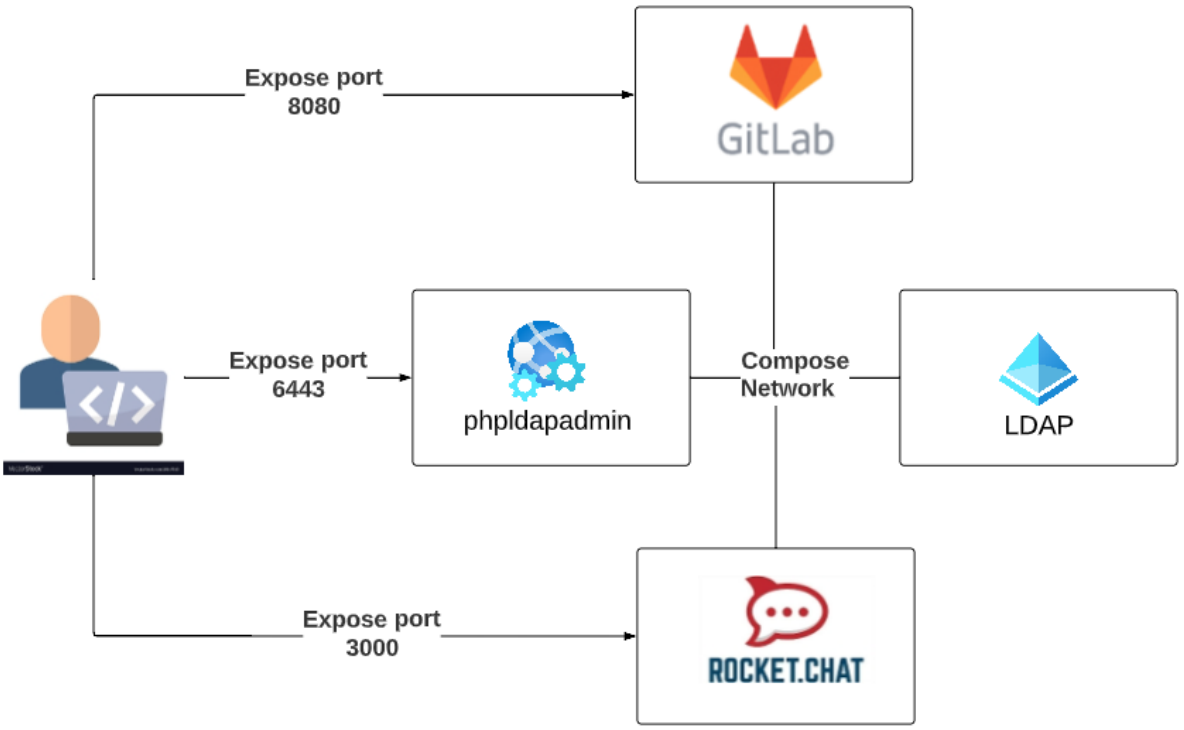
Артефакты блока

- Скриншот окна веб-интерфейса ldap_admin с созданной учётной записью.
- Скриншот выполненного входа в GitLab учётной записью из LDAP-каталога.

Задача 6. Rocket Chat

Вы уже продемонстрировали, что целую инфраструктуру можно поднимать только на основе Docker и Docker Compose. При этом поднятые сервисы позволят создавать учётные записи для нанимаемых разработчиков и других членов команды, а также централизованно управлять созданными учётными записями.

Но для минимального взаимодействия обязательно нужен чат для команды разработки. Сейчас вы займётесь добавлением Rocket Chat в созданную структуру.



Что нужно сделать

1. Добавьте сервис в существующий compose.yaml со следующими параметрами:
    a) имя сервиса: rocketchat;
    b) используемый образ: registry.rocket.chat/rocketchat/rocket.chat;
    c) опубликованные порты:
        - 3000:3000;
    d) переменные среды:
        - MONGO_URL=mongodb://mongodb:27017/rocketchat?replicaSet=rs0;
        - MONGO_OPLOG_URL=mongodb://mongodb:27017/local?replicaset=rs0;
        - ROOT_URL=http://localhost:3000;
        - PORT=3000;
        - DEPLOY_METHOD=docker;
    e) зависимый от: mongodb.
2. Добавьте сервис в существующий compose.yaml со следующими параметрами:
    a) имя сервиса: mongodb;
    b) используемый образ: docker.io/bitnami/mongodb:4.4;
    c) монтируемый том mongodb_data:/bitnami/mongodb;
    d) переменные среды:
        - MONGODB_REPLICA_SET_MODE=primary;
        - MONGODB_REPLICA_SET_NAME=rs0;
        - MONGODB_PORT_NUMBER=27017;
        - MONGODB_INITIAL_PRIMARY_HOST=mongodb;

        - MONGODB_INITIAL_PRIMARY_PORT_NUMBER=27017;
        - MONGODB_ADVERTISED_HOSTNAME=mongodb;
        - MONGODB_ENABLE_JOURNAL=true;
        - ALLOW_EMPTY_PASSWORD=yes;
3. Выполните docker compose up -d и docker compose ps. Сделайте скриншот.
4. Браузером войдите в порт 3000 сервера по протоколу http.
5. Выполните шаги мастера первоначальной настройки.
6. Сделайте скриншот окна приветствия Welcome to Rocket.Chat.

Советы и рекомендации

- В качестве дополнительного источника информации используйте открытые источники в интернете.
- Репозиторий от вендора находится по ссылке.
- Это демостенд. Можно использовать вымышленные email-адреса.

Критерии готовности

- Установлен и настроен Rocket Chat.
- Выполнен вход.

Артефакты блока

- Скриншот результата выполнения команды docker compose ps.
- Скриншот выполненного входа в Rocket Chat.

Задача 7. Интеграция Rocket Chat и LDAP

Отлично! Можно приступать к разработке. Благодаря GitLab у компании есть единое хранилище репозиториев кода. Наличие чата даст возможность разработчикам удобно коммуницировать друг с другом в процессе работы. А то, что он работает на собственных мощностях, минимизирует риски утечки данных.

Но вы, конечно, помните про важность интеграции всех систем с LDAP. Поэтому и в RocketChat решаете настроить её так, чтобы пользователи могли входить в него с теми же учётными записями из LDAP-каталога.

Что нужно сделать

1. Изучите документ по интеграции Rocket Chat и LDAP.
2. Настройте LDAP-аутентификацию в Rocket Chat следующим образом:
    a) перейдите в **Administration → Workspace → Setting → LDAP;**
    b) во вкладке Connection заполните поля:
        - Server Type = Other;
        - Host = ldap;
        - Port = 389;
        - User DN = cn=admin,dc=mardeev,dc=ru;
        - Password = <ldap_admin_password>;
    c) во вкладке User Search в разделе Search Filter укажите BaseDN для поиска пользователей; обычно это корень домена (dc=mardeev,dc=ru);
    d) нажмите Save changes.
3. Выполните вход в Rocket Chat той же учётной записью, которую использовали для входа в GitLab. Сделайте скриншот
4. При желании скачайте с официального сайта десктоп-приложение Rocket Chat и подключитесь к серверу.
5. Запушьте финальную конфигурацию файла compose.yaml. Добавьте проверяющего в проект.

Советы и рекомендации

- При входе LDAP-пользователем в качестве логина используйте uid — это поле User Name в веб-интерфейсе ldap_admin.
- При входе LDAP-пользователем появится окно Two Factor Authentication via Email. Чтобы его отключить, снова войдите в Rocket Chat администратором и отключите двухфакторную аутентификацию: **Administration → Workspace → Settings → Accounts → Two Factor Authentication.**
- В качестве дополнительного источника информации используйте открытые источники в интернете и официальный сайт.

Критерии готовности

- Настроена интеграция Rocket Chat с LDAP.

Артефакты блока

- Скриншот окна выполненного входа в Rocket Chat с использованием учётной записи из LDAP-каталога.
- Ссылка на Git-репозиторий с файлом compose.yaml и всеми скриншотами финальной работы.

Формат сдачи материалов и оценивание

Список материалов

- Скриншоты команд docker version и docker compose version.
- Скриншоты команд docker compose ps с сервисами:
  - ldap,
  - ldap и ldap_admin,
  - ldap, ldap_admin, gitlab,
  - ldap, ldap_admin, gitlab, rocketchat и mongodb.
- Скриншот выполненного входа в веб-интерфейс phpLDAPadmin.
- Скриншот результата выполненного входа в GitLab.
- Скриншот окна входа в GitLab с двумя входами: Standard и LDAP.
- Скриншот окна веб-интерфейса ldap_admin с созданной учётной записью.
- Скриншот выполненного входа в GitLab учётной записью из LDAP-каталога.
- Скриншот выполненного входа в Rocket Chat.
- Скриншот окна выполненного входа в Rocket Chat с использованием учётной записи из LDAP-каталога.
- Ссылка на Git-репозиторий с файлом compose.yaml и всеми скриншотами финальной работы.

Формат сдачи

Финальную работу следует сдать в виде ссылки на Git-репозиторий с документами и артефактами. В этом случае документом с набором ссылок и пояснениями может быть readme-файл.

Критерии оценки работы

Принято:

- все созданные сервисы работают корректно;
- настроена интеграция всех сервисов;
- предоставлены все необходимые скриншоты и файл compose.yaml.

На доработку:

- допущены ошибки в конфигурации компонентов;
- предоставлены не все скриншоты;
- не предоставлен файл compose.yaml.

Заключение

Поздравляем с окончанием финальной работы!

Вы отлично поработали: объединили полученные знания и построили хорошую и расширяемую инфраструктуру. Теперь вам будет легче повторить это и в реальной компании. Удачи в работе!