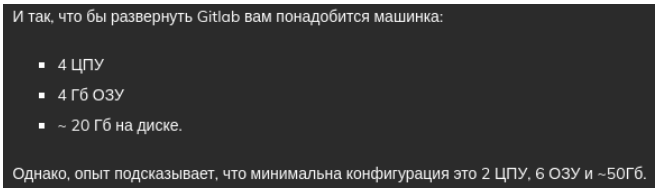
В мире современной разработки программного обеспечения доступ к современным инструментам становится все более критически важным для успешной работы. Один из таких инструментов — GitLab, мощная платформа для управления репозиториями и непрерывной интеграции. Однако, чтобы максимально эффективно использовать все её возможности, иногда бывает необходимо развернуть локальную копию GitLab на своем компьютере или сервере. В этой мини-статье мы рассмотрим, как быстро и легко выполнить это задание, открыв перед вами дверь к миру управления кодом с помощью собственной локальной установки GitLab.

docker-compose файл с запуском gitlab-ce и раннером
(*В этом файле необходимо убрать проброс log файлов (not the best practice) и желательно сделать так, чтобы все нужные файлы пробрасывались в директорию с файлом docker-compose.yml, а также ВАЖНО в git-lab-runner* **/var/run/docker.sock**:/var/run/docker.sock)

полезный сайт(*полезная шпора, но в видео все есть и ниже тоже много чего*)

минимальные требования устройства для разворачивания gitlab local:



Команды для работы с local gitlab

git config --global user.name "Administrator"
git config --global user.email "admin@example.com"

git clone ssh://git@192.168.55.119:50022/root/myfirsthardproj.git
cd myfirsthardproj
git switch --create main
touch README.md
git add README.md
git commit -m "add README"
git push --set-upstream origin main

cd existing_folder
git init --initial-branch=main
git remote add origin ssh://git@192.168.55.119:50022/root/myfirsthardproj.git
git add .
git commit -m "Initial commit"
git push --set-upstream origin main

cd existing_repo
git remote rename origin old-origin
git remote add origin ssh://git@192.168.55.119:50022/root/myfirsthardproj.git
git push --set-upstream origin --all
git push --set-upstream origin --tags

Git global setup

```
git config --global user.name "Administrator"
git config --global user.email "admin@example.com"
```

Create a new repository

```
git clone ssh://git@192.168.55.119:50022/root/myfirsthardproj.git
cd myfirsthardproj
git switch --create main
touch README.md
git add README.md
git commit -m "add README"
git push --set-upstream origin main
```

Push an existing folder

```
cd existing_folder
git init --initial-branch=main
git remote add origin ssh://git@192.168.55.119:50022/root/myfirsthardproj.git
git add .
git commit -m "Initial commit"
git push --set-upstream origin main
```

Push an existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin ssh://git@192.168.55.119:50022/root/myfirsthardproj.git
git push --set-upstream origin --all
git push --set-upstream origin --tags
```

```
cd existing_repo
git remote add origin http://192.168.55.119/timofey/pyproject.git
git branch -M main
git push -uf origin main
```

Runner, pipeline

Runner и pipeline - это два разных концепта в GitLab, которые взаимодействуют друг с другом.

Pipeline - это серия шагов, которые описываются в файле .gitlab-ci.yml и выполняются последовательно для каждого коммита в вашем репозитории. Pipeline позволяет автоматизировать процесс разработки и тестирования вашего приложения и включает в себя шаги, такие как сборка, тестирование и деплой.

Runner - это компонент, который выполняет каждый шаг pipeline на физической или виртуальной машине. Runner получает задачу из GitLab и выполняет ее в соответствии с описанием из .gitlab-ci.yml.

Таким образом, pipeline - это описание шагов, которые должен выполнить Runner. Runner же - это компонент, который фактически выполняет каждый шаг pipeline.

Отличие между Runner и pipeline можно сравнить с отличием между планом и исполнением. Pipeline описывает план действий, а Runner выполняет этот план.

В GitLab runner есть два типа:

Shared runners: это runner, который доступен для всех проектов в GitLab. Он обычно находится на удаленном сервере.

Specific runners: это runner, который вы можете настроить только для вашего проекта. Он может быть развернут на физическом сервере или в виртуальной
машине.

> **Стоит учесть**
>
> Переменные лучше описывать не в pipeline, а в CI/CD variables, если необходимо в качестве переменной передать ssh-key, то необходимо выбрать тип переменной FILE

Запуск gitlab-runner

Запускать gitlab-runner локально без docker-compose
`docker run -d --name gitlab-runner --restart always \ -v /srv/gitlab-runner/config:/etc/gitlab-runner \ -v /var/run/docker.sock:/var/run/docker.sock \ gitlab/gitlab-runner:latest`
(***НО** лучше все-таки запускать с помощью docker-compose**)*

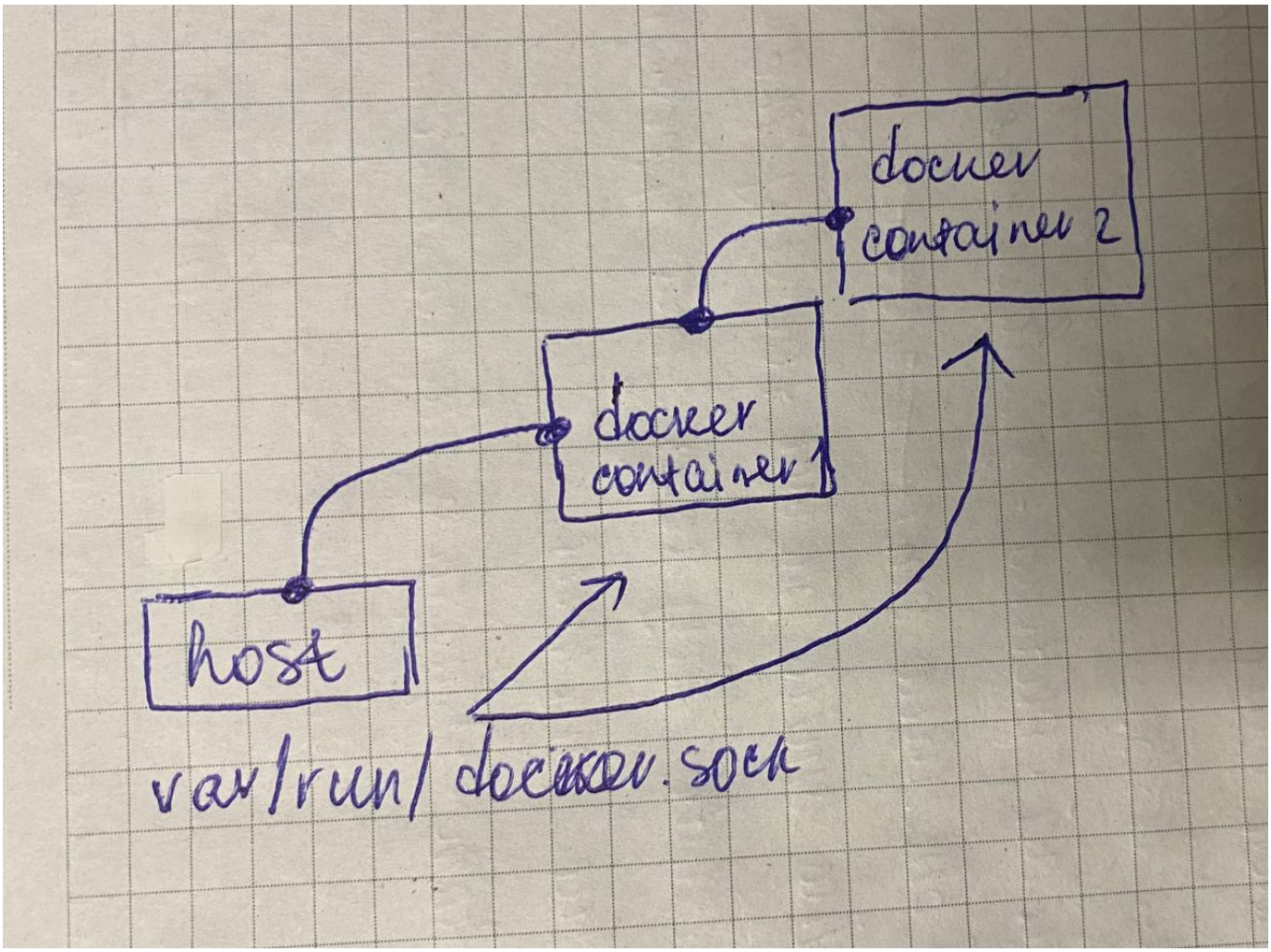
`gitlab-runner verify` - можно посмотреть какие логические runners есть и какие из них работают

`gitlab-runner register` - зарегистрировать gitlab-runner
*(**НО** лучше зайти в панель администратора и там выбрать в Admin Area панель с созданием runner и затем скопировать команду уже с токеном и hostip)*

> **⚠ ВАЖНО**

Когда мы пробрасываем docker.sock с хоста на контейнер, нужно обязательно пробрасывать так! **/var/run/docker.sock:/var/run/docker.sock**

`/etc/gitlab-runner/config.toml` - тут хранятся конфигурации runner (в этом файле мы можем сделать проброс сокета с хостовой машины на докер, который запущен в докере **dockerindocker**)



LDAP

LDAP (Lightweight Directory Access Protocol) - это протокол для доступа к службам каталогов. Он используется для управления и поиска информации в структурированных каталогах, таких как Active Directory, OpenLDAP и других службах каталогов.

`/etc/gitlab/gitlab.rb` - файл с конфигурацией git-lab (*тут можно найти конфигурацию ldap, примерно на 510 строчке*)

`gitlab-ctl reconfigure` - применить конфигурацию после редактирования файла /etc/gitlab/gitlab.rb

`gitlab-rake gitlab:ldap:check` - проверить состояние LDAP

```
root@192:/var/log/gitlab/gitlab-rails# gitlab-rake gitlab:ldap:check
Checking LDAP ...

LDAP: ... Server: ldapmain
LDAP authentication... Success
LDAP users with access to your GitLab server (only showing the first 100 results)
       DN: cn=tnalekseev,ou=dcadmins,ou=gitlab,dc=rosatom,dc=ru         sAMAccountName: TNAlekseev
       DN: cn=administrator,ou=dcadmins,ou=gitlab,dc=rosatom,dc=ru      sAMAccountName: Administrator

Checking LDAP ... Finished
```

Перед началом нужно раскомментировать строку **ldap_enabled** и, если не стоит поставить флаг **true**

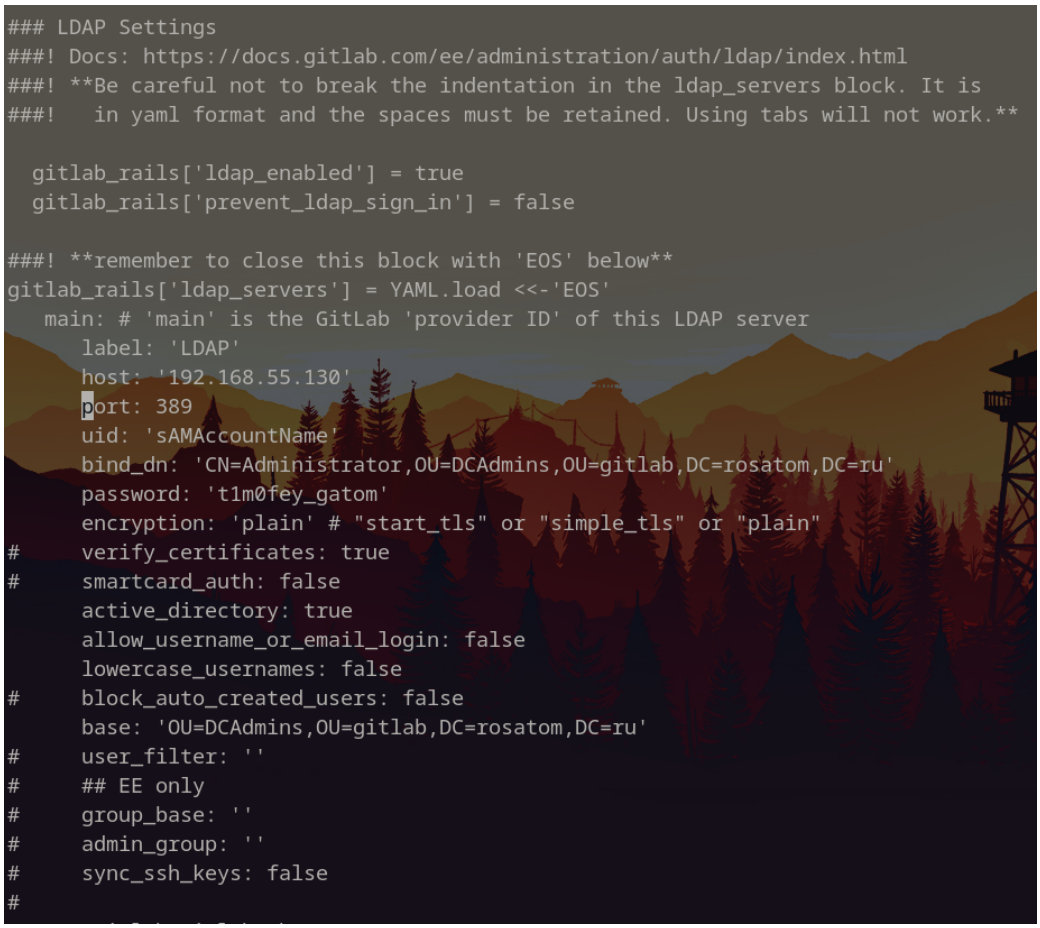
Параметр `gitlab_rails['prevent_ldap_sign_in']` в конфигурационном файле `gitlab.rb` определяет, будет ли предотвращено вход через учетные данные LDAP в GitLab.

Когда параметр установлен на значение `false`, это означает, что GitLab разрешает пользователям аутентифицироваться с помощью их учетных данных LDAP. Пользователи смогут войти в систему, используя свои учетные данные из Active Directory или другой LDAP-системы, если они успешно прошли аутентификацию на LDAP-сервере.

Однако, если параметр установлен на значение `true`, вход через учетные данные LDAP будет предотвращен, и пользователи не смогут использовать свои учетные данные из LDAP для аутентификации в GitLab. В этом случае пользователи смогут входить в систему только с помощью учетных данных, установленных в GitLab (локальных учетных данных).

1. `main`: Это идентификатор (провайдер ID) LDAP-сервера в GitLab. В данном случае, `main` используется для указания, что это основной LDAP-сервер.
2. `label`: Указывает метку или имя для LDAP-провайдера, чтобы его можно было идентифицировать. В вашем примере метка установлена как `'LDAP'`.
3. `host`: Это IP-адрес или имя хоста вашего LDAP-сервера. В примере, указан IP-адрес `'192.168.55.130'`. (*адрес удаленного хоста, с active directory*)
4. `port`: Порт, на котором работает LDAP-сервер. Обычно, порт 389 используется для нешифрованных соединений LDAP. (*защищенный 636*)
5. `uid`: Указывает атрибут, который будет использоваться как уникальный идентификатор пользователя. В данном случае, `'sAMAccountName'` предполагается, что вы используете Active Directory, и учетные записи пользователей идентифицируются атрибутом `sAMAccountName`. (*бывают еще uid, cn)
6. `bind_dn`: Это Distinguished Name (DN) пользователя, который будет использоваться для привязки (bind) или подключения к LDAP-серверу. В вашем примере, указан DN `'DC=rosatom,DC=ru'`.
   (*В этом контексте, `bind_dn` - это учетные данные пользователя, которые будут использоваться GitLab для привязки к LDAP-серверу, чтобы выполнить операции аутентификации и поиска пользователей и групп. Обычно, это специальный учетный запись в LDAP с достаточными правами для выполнения этих операций.*)
   ***МОЖНО ПОСМОТРЕТЬ** на сервере с помощью команды ldifde -f output.ldif, затем открыть этот файл и уже там найти нужного пользователя, например Administrator*)
7. Параметр `base` в настройках LDAP определяет базовый DN (Distinguished Name) для выполнения поисковых запросов к LDAP-серверу.
   (*`base` и `user_filter`: Определяют базовый DN (Distinguished Name) для поиска пользователей и фильтр для поиска пользователей на LDAP-сервере.*)
8. `password`: Пароль для пользователя, чьи учетные данные указаны в параметре `bind_dn`. В вашем примере, пароль установлен как `'t1m0fey_gatom'`.
9. `encryption`: Определяет метод шифрования для соединения с LDAP-сервером. В вашем примере, установлено значение `'plain'`, что означает, что соединение не будет шифроваться.
   (*9.1. `start_tls` (по умолчанию): Это означает, что GitLab будет устанавливать обычное незашифрованное соединение (LDAP) с LDAP-сервером, а затем запрашивать установку шифрованного соединения через STARTTLS. STARTTLS - это команда для переключения на шифрованное соединение внутри незашифрованного соединения.*
   9.2. `simple_tls`: Это означает, что GitLab будет устанавливать прямое шифрованное соединение (LDAPS) с LDAP-сервером. При использовании `simple_tls`, соединение сразу же устанавливается с использованием шифрования.
   9.3. `plain`: Это означает, что GitLab будет устанавливать незашифрованное (обычное) соединение с LDAP-сервером без использования шифрования. Важно заметить, что передача учетных данных в незашифрованном виде представляет потенциальную угрозу для безопасности.)
10. `active_directory`: Устанавливается в `true`, если используется Active Directory, иначе `false`.
11. `allow_username_or_email_login`: Устанавливается в `true`, если вы хотите разрешить пользователям входить с помощью как имени пользователя, так и электронной почты.
12. `lowercase_usernames`: Устанавливается в `true`, если вы хотите преобразовать имена пользователей в нижний регистр при аутентификации.
13. `block_auto_created_users`: Устанавливается в `true`, чтобы блокировать пользователей, созданных автоматически при первой аутентификации через LDAP.
14. `verify_certificates`: Устанавливает, требуется ли проверка сертификата сервера при использовании защищенного соединения.w

(**ЧАСТО** base это bind_dn без CN, в параметре base описывается, где хранятся пользователи, а в bind_dn тот пользователь под которым у нас будет первая авторизация*)



```
### LDAP Settings
###! Docs: https://docs.gitlab.com/ee/administration/auth/ldap/index.html
###! **Be careful not to break the indentation in the ldap_servers block. It is
###!   in yaml format and the spaces must be retained. Using tabs will not work.**

  gitlab_rails['ldap_enabled'] = true
  gitlab_rails['prevent_ldap_sign_in'] = false

###! **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: '192.168.55.130'
      port: 389
      uid: 'sAMAccountName'
      bind_dn: 'CN=Administrator,OU=DCAdmins,OU=gitlab,DC=rosatom,DC=ru'
      password: 't1m0fey_gatom'
      encryption: 'plain' # "start_tls" or "simple_tls" or "plain"
#     verify_certificates: true
#     smartcard_auth: false
      active_directory: true
      allow_username_or_email_login: false
      lowercase_usernames: false
#     block_auto_created_users: false
      base: 'OU=DCAdmins,OU=gitlab,DC=rosatom,DC=ru'
#     user_filter: ''
#     ## EE only
#     group_base: ''
#     admin_group: ''
#     sync_ssh_keys: false
#
```

> ⚠ **СУПЕР МЕГА ВАЖНо**
> **Нужно убрать # перед EOS в конце данного списка**

Логи можно посмотреть в файле */var/log/gitlab/production.log* или в ***/var/log/gitlab/production_json.log***

как пользоваться гит лаб в условиях санкций

Домен можно настроить локально /etc/hosts

`gitlab-rake "gitlab:password:reset"` - изменить пароль пользователя

docker
linux
Linux (в частности Fedora)