WHITE PAPER

НАСТРОЙКА ИНТЕГРАЦИИ ПЛАТФОРМЫ ДЛЯ РАБОТЫ С КОДОМ GITFLIC И TATLIN.OBJECT

О WHITE PAPER

Данный документ описывает архитектурные принципы и подходы, а также рекомендации по их применению при построении корпоративных решений на основе продуктов производства компании YADRO.

Документ предназначен для широкой аудитории, но также может включать материалы, требующие специальной профессиональной подготовки, ориентированные на системных архитекторов, администраторов, сервисных инженеров и других ИТ-специалистов, участвующих в планировании, проектировании и внедрении комплексных решений.

! Представленная информация основана на результатах исследований, а также экспертизе специалистов YADRO. Документ носит не предписывающий, а информационный характер, целью которого является информирование получателей документа о предмете произведенного исследования и о его результатах. При этом конкретные результаты и эффективность конечных решений зависят от множества факторов, которые могут различаться в зависимости от особенностей инфраструктуры и условий эксплуатации. Ни при каких обстоятельствах YADRO не несет ответственности за последствия применения получателем документа рекомендаций, инструкций, подходов и решений, указанных в документе.

Запрещается копировать, воспроизводить, изменять, публиковать, передавать, распространять, продавать информацию (часть информации), содержащуюся в документе, или создавать производные работы на основе содержания документа без предварительного письменного разрешения YADRO.

Любая информация или утверждения относительно возможностей, емкости, производительности или других характеристик продукции, указанной в документе, предоставляются исключительно на условиях «как есть» и приведена исключительно в целях описания произведенного исследования. YADRO оставляет за собой право вносить изменения и обновлять информацию в документе без уведомления.

«YADRO» является зарегистрированным товарным знаком ООО «КНС ГРУПП». Другие названия продуктов и компаний, упомянутые здесь, могут быть товарными знаками или торговыми наименованиями их соответствующих владельцев.

Актуальные версии документации публикуются на информационном ресурсе YADRO по адресу service.yadro.com.

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ 5
2. GITFLIC: УСТАНОВКА И НАСТРОЙКА В DOCKER 5
   - 2.1. Установка GitFlic в Docker 5
   - 2.2. Настройка application.properties для подключения к S3 6
   - 2.3. Применение настроек S3 в Docker-контейнере 7
   - 2.4. Команды для переноса данных в S3 8
3. ДОБАВЛЕНИЕ КОРНЕВОГО СЕРТИФИКАТА S3-ХРАНИЛИЩА В ДОВЕРЕННЫЕ 8
   - 3.1. Получение корневого сертификата S3 8
   - 3.2. Копирование сертификата в GitFlic volume 8
   - 3.3. Установка сертификата в контейнер GitFlic 9
   - 3.4. Автоматическая установка сертификата при запуске контейнера 10
4. ПРИМЕРЫ РАБОТЫ И ЗАГРУЗКИ ДАННЫХ 11
   - 4.1. Загрузка файлов и работа с Git LFS 11
   - 4.2. Работа с CI/CD артефактами 12
   - 4.3. Управление релизами и пакетами 13
5. ПРИМЕЧАНИЯ 14

ОБ АВТОРАХ

Антон Морозов — ведущий ИТ-инженер и системный архитектор с более чем 15-летним опытом проектирования и внедрения высоконагруженных отказоустойчивых решений.

Начинал карьеру с глубокого погружения в Windows-экосистемы, что позволило сформировать уникальную экспертизу в построении сложных гибридных инфраструктур. В рамках сотрудничества с Microsoft и Ericsson реализовывал масштабные проекты по развертыванию IPTV-платформ для международных телеком-операторов (Vodafone Greece, TDC Denmark, Oi Brazil), где особое внимание уделял аспектам высокой доступности и бесперебойной миграции сервисов.

Сейчас в YADRO занимает позицию ведущего специалиста отдела объектных систем хранения данных, где фокусируется на:

- Оптимизации интеграции Tatlin Object Storage со сторонним ПО;
- внедрении best practices по работе с распределенными системами хранения.

ВВЕДЕНИЕ

Этот документ описывает установку GitFlic в Docker, подключение S3-хранилища, миграцию данных и добавление корневого сертификата в доверенные.

ВЕРСИИ ИСПОЛЬЗУЕМОГО ПО	
TATLIN.OBJECT	1.8.1
GitFlic	4.4.1

GITFLIC: УСТАНОВКА И НАСТРОЙКА В DOCKER

Установка GitFlic в Docker

1. Скачайте архив с приложением GitFlic и распакуйте его.
2. Создайте сертификат для SSH-сервера и ключевую пару RSA:

```
ssh-keygen -t ed25519 -N "" -q -f ./key.pem
   openssl genpkey -algorithm RSA -out ./private_key.pem -pkeyopt rsa_keygen_bits:2048
   openssl rsa -pubout -in private_key.pem -out ./public_key.pem
```

3. Создайте volume `gitflic_cert` и поместите в него созданные ключи. Точку монтирования можно найти командой:

```
docker volume inspect gitflic_cert | grep Mountpoint
```

Затем создайте volume и скопируйте ключи:

```
docker volume create gitflic_cert
  cp private_key.pem public_key.pem key.pem $(docker volume inspect gitflic_cert -f '{{.Mountpoint}}')
```

4. Перейдите в директорию `docker` внутри распакованного архива и при необходимости отредактируйте файл `.env`.
5. Запустите систему:

```
docker compose --detach -f docker-compose.yaml --env-file .env up
```

Настройка application.properties для подключения к S3

Добавьте в `application.properties`, с которым запускается бинарный файл GitFlic (в Docker обычно лежит в `volume gitflic_etc`), следующие параметры:

```
properties
# Активация работы с S3
gitflic.object-storage.enabled=true

# Конфигурация подключения к S3
# Регион (пример: eu-west-3)
gitflic.object-storage.connection.region=eu-west-3
# Бакет по умолчанию
gitflic.object-storage.default-bucket=YourBucketName
# Endpoint S3
# Если endpoint не указан, будет использовано значение http://localhost:4566/
gitflic.object-storage.connection.endpoint=https://data0.indierock-node1.indierock.object.set.yadro.com
# Статические ключи доступа
gitflic.object-storage.connection.awsAccessKeyId=YourAccessKeyId
gitflic.object-storage.connection.awsSecretAccessKey=YourSecretAccessKey

# Данные, которые будут храниться в S3.
# Все старые данные, настроенные на локальную ФС до подключения S3, останутся на прежнем месте.

# Пакеты
gitflic.object-storage.objects.packages.enabled=true
gitflic.object-storage.objects.packages.provider=s3
#gitflic.object-storage.objects.packages.bucket=custom-packages-bucket

# Релизы и загрузки
gitflic.object-storage.objects.uploads.enabled=true
gitflic.object-storage.objects.uploads.provider=s3
#gitflic.object-storage.objects.uploads.bucket=custom-uploads-bucket

# CI/CD артефакты
gitflic.object-storage.objects.artifacts.enabled=true
gitflic.object-storage.objects.artifacts.provider=s3
#gitflic.object-storage.objects.artifacts.bucket=custom-artifacts-bucket

# LFS-файлы
gitflic.object-storage.objects.lfs.enabled=true
gitflic.object-storage.objects.lfs.provider=s3
#gitflic.object-storage.objects.lfs.bucket=custom-lfs-bucket
```

! Примечание: если отдельные bucket-параметры не заданы, будет использоваться `default-bucket`.

Применение настроек S3 в Docker-контейнере

Шаг 1. Остановите контейнер с GitFlic (если запущен).
Шаг 2. Если базы данных поднимаются в контейнерах, убедитесь, что контейнеры с **Postgres** и **Redis** запущены и доступны.
Шаг 3. Перейдите в директорию `cli` внутри распакованного архива GitFlic.
Шаг 4. В файле `docker/.env` раскомментируйте переменные `DATA_VOLUME` и `ETC_VOLUME`, указав имена соответствующих volume. Если используется внешний `.env`, добавьте:

```
env
   DATA_VOLUME=gitflic_data
   ETC_VOLUME=gitflic_etc
   CERT_VOLUME=gitflic_cert
```

Шаг 5. Создайте контейнер (команда выполняется из директории `cli/docker`):

```
docker compose --env-file docker/.env --file cli/docker/docker-compose.yaml create
```

Шаг 6. Войдите в созданный контейнер. Убедитесь, что указан корректный `docker network` и контейнер имеет доступ к PostgreSQL и Redis:

```
docker run -ti --env-file docker/.env --network git_net --entrypoint=bash docker-fixcli
```

Шаг 7. Запустите приложение и дождитесь окончания загрузки:

```
./gitflic-cli.sh ./application.properties
```

Пример лога при запуске:

```
2024-02-22 14:36:32.630 INFO 7 --- [   main] o.s.i.channel.PublishSubscribeChannel  : Channel 'application.errorChannel' has 1 subscriber(s).
2024-02-22 14:36:32.631 INFO 7 --- [   main] o.s.i.endpoint.EventDrivenConsumer     : started bean '_org.springframework.integration.errorLogger'
shell:>
```

Шаг 8. Выполните нужную команду, например фикс релизов:

```
fix releases
```

Пример лога в начале:

```
 2024-02-14 14:37:50.291  INFO 70237 --- [          main] c.gitflic.cli.service.ReleaseFixService  : --- Подсчет количества релизов ---
 2024-02-14 14:37:50.457  INFO 70237 --- [          main] c.gitflic.cli.service.ReleaseFixService  : Всего релизов 24
 2024-02-14 14:37:50.458  INFO 70237 --- [          main] c.gitflic.cli.service.ReleaseFixService  : ----------------------------------------
```

Пример лога после выполнения:

```
2024-02-14 14:37:51.004  INFO 70237 --- [          main] c.gitflic.cli.service.ReleaseFixService  : Мигрировано: 24/24 100,00% релизов
2024-02-14 14:37:51.004  INFO 70237 --- [          main] c.gitflic.cli.service.ReleaseFixService  : Задача выполнена
```

Шаг 9. После успешного выполнения нажмите `Ctrl+C` для выхода из приложения и удалите временные контейнеры/образы, созданные на шагах 5–6.

Команды для переноса данных в S3

Перенос пакетов реестра:

```
s3 migrate-to packages
```

Перенос релизов:

```
s3 migrate-to releases
```

Перенос LFS-файлов:

```
s3 migrate-to lfs
```

Перенос картинок (аватарок команд, компаний, OAuth-приложений, а также аватарок и баннеров пользователей):

```
fix images
s3 migrate-to images
```

ДОБАВЛЕНИЕ КОРНЕВОГО СЕРТИФИКАТА S3-ХРАНИЛИЩА В ДОВЕРЕННЫЕ

Если ваше S3-хранилище использует самоподписанный SSL-сертификат или корпоративный CA, необходимо добавить корневой сертификат в доверенные для GitFlic.

Получение корневого сертификата S3

Получите корневой сертификат вашего S3-хранилища одним из способов:

Способ 1: Скачать с помощью openssl

```
# Замените HOSTNAME на адрес вашего S3-endpoint
openssl s_client -connect data0.indierock-node1.indierock.object.set.yadro.com:443 -showcerts
</dev/null 2>/dev/null | \
  openssl x509 -outform PEM > s3-root-ca.crt
```

Способ 2: Экспортировать из браузера

1. Откройте в браузере адрес S3-endpoint (например, https://data0.indierock-node1.indierock.object.set.yadro.com)
2. Нажмите на замок в адресной строке → Сертификат → Подробности
3. Найдите корневой сертификат (Root CA) в цепочке
4. Экспортируйте в формате PEM с именем `s3-root-ca.crt`

Способ 3: Получить у администратора хранилища
Запросите файл корневого сертификата у администратора S3-хранилища.

Копирование сертификата в GitFlic volume

Скопируйте полученный сертификат в Docker volume gitflic_cert:

```
# Скопировать сертификат в volume gitflic_cert
cp s3-root-ca.crt $(docker volume inspect gitflic_cert -f '{{.Mountpoint}}')

# Проверить, что файл скопирован
ls -la $(docker volume inspect gitflic_cert -f '{{.Mountpoint}}')
```

Установка сертификата в контейнер GitFlic

Шаг 1. Войдите в запущенный контейнер GitFlic

```
# Найдите имя контейнера GitFlic
docker ps | grep gitflic

# Войдите в контейнер (замените gitflic-app на имя вашего контейнера)
docker exec -it gitflic-app bash
```

Шаг 2. Добавьте сертификат в системное хранилище сертификатов

```
# Внутри контейнера выполните:
cp /app/cert/s3-root-ca.crt /usr/local/share/ca-certificates/s3-root-ca.crt
update-ca-certificates
```

Вывод команды должен содержать:

```
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
```

Шаг 3. Импортируйте сертификат в Java KeyStore (для Java-приложения GitFlic)

```
# Внутри контейнера выполните:
keytool -import -noprompt \
  -storepass changeit \
  -trustcacerts \
  -alias s3-root-ca \
  -file /app/cert/s3-root-ca.crt \
  -keystore "$JAVA_HOME/lib/security/cacerts"
```

Вывод команды должен содержать:

```
Certificate was added to keystore
```

Шаг 4. Проверьте установку сертификата

```
# Проверить в системном хранилище
ls -la /etc/ssl/certs/ | grep s3-root-ca

# Проверить в Java KeyStore
keytool -list -keystore "$JAVA_HOME/lib/security/cacerts" -storepass changeit | grep s3-root-ca
```

Шаг 5. Выйдите из контейнера и перезапустите GitFlic

```
# Выйти из контейнера
exit

# Перезапустить контейнер GitFlic
docker restart gitflic-app
```

Автоматическая установка сертификата при запуске контейнера

Для автоматической установки сертификата при каждом запуске контейнера добавляйте в docker-compose.yaml в секцию контейнера GitFlic:

```
services:
  gitflic:
    # ... остальные параметры ...
    volumes:
      - gitflic_cert:/app/cert
      - gitflic_data:/app/data
      - gitflic_etc:/app/etc
    # Добавьте entrypoint script
    entrypoint: /app/scripts/entrypoint.sh
```

Создайте файл entrypoint.sh в volume или в образе:

```
#!/bin/bash
set -e

# Установка сертификата S3 в системное хранилище
if [ -f /app/cert/s3-root-ca.crt ]; then
  echo "Installing S3 root certificate..."
  cp /app/cert/s3-root-ca.crt /usr/local/share/ca-certificates/
  update-ca-certificates

  # Импорт в Java KeyStore
  keytool -import -noprompt \
    -storepass changeit \
    -trustcacerts \
    -alias s3-root-ca \
    -file /app/cert/s3-root-ca.crt \
    -keystore "$JAVA_HOME/lib/security/cacerts" 2>/dev/null || true

  echo "S3 root certificate installed successfully"
fi

# Запуск основного приложения
exec java -jar /app/gitflic.jar "$@"
```

Не забудьте сделать скрипт исполняемым:

```
chmod +x entrypoint.sh
```

ПРИМЕРЫ РАБОТЫ И ЗАГРУЗКИ ДАННЫХ

Загрузка файлов и работа с Git LFS

1. Базовая работа с репозиторием

```
# Клонировать репозиторий
git clone https://gitflic.example.com/project/my-repo.git
cd my-repo

# Добавить файлы и запушить
git add .
git commit -m "feat: initial commit"
git push origin main
```

2. Настройка и использование Git LFS

```
# Установить Git LFS
apt install git-lfs   # Debian/Ubuntu
git lfs install

# Отслеживать крупные файлы (бинарники, модели, архивы)
git lfs track "*.bin"
git lfs track "*.zip"
git lfs track "*.tar.gz"
git lfs track "*.psd"

# Убедиться, что .gitattributes добавлен
git add .gitattributes
git commit -m "chore: configure LFS tracking"

# Добавить крупный файл — LFS подхватит автоматически
cp /path/to/large-model.bin .
git add large-model.bin
git commit -m "feat: add model binary"
git push origin main
```

3. Проверка LFS-объектов

```
# Список LFS-файлов в репозитории
git lfs ls-files

# Статус LFS
git lfs status

# Проверить, что файл ушёл в S3 (через CLI GitFlic)
s3 migrate-to lfs
Ожидаемый лог:
INFO --- Подсчет количества LFS-файлов ---
INFO Всего LFS-файлов: 38
INFO Мигрировано: 38/38 100,00% LFS-файлов
INFO Задача выполнена
```

Работа с CI/CD артефактами

1. Пример .gitflic-ci.yml с сохранением артефактов

```
stages:
  - build
  - test
  - release

build-job:
  stage: build
  script:
    - mkdir -p dist
    - echo "Building project..."
    - tar -czf dist/app-${CI_COMMIT_SHORT_SHA}.tar.gz src/
  artifacts:
    paths:
      - dist/
    expire_in: 7 days

test-job:
  stage: test
  script:
    - echo "Running tests..."
    - mkdir -p reports
    - echo "<testsuite tests='5' failures='0'/>" > reports/junit.xml
  artifacts:
    reports:
      junit: reports/junit.xml
    paths:
      - reports/
    expire_in: 30 days
```

2. Скачать артефакт через curl

```
# Получить артефакты конкретного пайплайна
curl -L \
  --header "Private-Token: YOUR_TOKEN" \
  "https://gitflic.example.com/api/v1/project/{project_id}/pipeline/{pipeline_id}/artifacts" \
  -o artifacts.zip

# Распаковать
unzip artifacts.zip -d ./artifacts
```

3. Миграция артефактов в S3
   После настройки S3-хранилища выполните миграцию существующих артефактов через GitFlic CLI:

```
# Запустить CLI контейнер
docker run -ti --env-file docker/.env --network git_net --entrypoint=bash docker-fixcli

# Внутри контейнера:
./gitflic-cli.sh ./application.properties

# В shell GitFlic выполнить:
s3 migrate-to artifacts
```

   Ожидаемый лог:

```
INFO --- Подсчет количества артефактов ---
INFO Всего артефактов: 142
INFO Мигрировано: 142/142 100,00% артефактов
INFO Задача выполнена
```

Управление релизами и пакетами

1. Создание релиза вручную через API

```
# Создать тег
git tag -a v1.2.0 -m "Release v1.2.0"
git push origin v1.2.0

# Создать релиз через API GitFlic
curl -X POST \
  --header "Private-Token: YOUR_TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "name": "Release v1.2.0",
    "tag_name": "v1.2.0",
    "description": "## Changelog\n- Feature A\n- Bugfix B"
  }' \
  "https://gitflic.example.com/api/v1/project/{project_id}/release"
```

2. Загрузка файла в релиз

```
# Прикрепить бинарник к релизу
curl -X POST \
  --header "Private-Token: YOUR_TOKEN" \
  --form "file=@dist/app-v1.2.0-linux-amd64.tar.gz" \
  "https://gitflic.example.com/api/v1/project/{project_id}/release/v1.2.0/upload"
```

3. Автоматический релиз из CI/CD

```
release-job:
  stage: release
  only:
    - tags
  script:
    - tar -czf app-${CI_COMMIT_TAG}-linux-amd64.tar.gz dist/
    - |
      curl -X POST \
        --header "Private-Token: ${GITFLIC_TOKEN}" \
        --header "Content-Type: application/json" \
        --data "{
          \"name\": \"Release ${CI_COMMIT_TAG}\",
          \"tag_name\": \"${CI_COMMIT_TAG}\",
          \"description\": \"Automated release from CI/CD\"
        }" \
        "${GITFLIC_URL}/api/v1/project/${CI_PROJECT_ID}/release"
    - |
      curl -X POST \
        --header "Private-Token: ${GITFLIC_TOKEN}" \
        --form "file=@app-${CI_COMMIT_TAG}-linux-amd64.tar.gz" \
        "${GITFLIC_URL}/api/v1/project/${CI_PROJECT_ID}/release/${CI_COMMIT_TAG}/upload"
```

4. Миграция релизов и пакетов в S3

```
# В shell GitFlic CLI:
s3 migrate-to releases
s3 migrate-to packages

# Если были проблемы с загрузкой — фикс и миграция изображений:
fix releases
fix images
s3 migrate-to images
Ожидаемый лог для релизов:
INFO --- Подсчет количества релизов ---
INFO Всего релизов: 24
INFO Мигрировано: 24/24 100,00% релизов
INFO Задача выполнена
```

ПРИМЕЧАНИЯ

- Команды выше предполагают наличие Docker и Docker Compose.
- Имена сети/volume могут отличаться от приведенных примеров — используйте свои значения из .env и docker-compose.yaml.
- Если отдельные bucket-параметры не заданы, объекты сохраняются в бакет по умолчанию (default-bucket).
- После установки корневого сертификата GitFlic сможет безопасно подключаться к S3-хранилищу по HTTPS.
- Пароль по умолчанию для Java KeyStore — changeit. Если в вашей системе он изменен, используйте соответствующий пароль в команде keytool.
- Токен (Private-Token) генерируется в профиле GitFlic → Настройки → Access Tokens.
- Для CI/CD используйте переменные окружения (GITFLIC_TOKEN, GITFLIC_URL) — никогда не фиксируйте токен в репозитории.
- Все команды s3 migrate-to выполняются внутри GitFlic CLI (gitflic-cli.sh), а не на хосте.
- Миграция не удаляет данные с локальной ФС — старые файлы остаются на месте до ручной очистки.