WHITE PAPER

НАСТРОЙКА ИНТЕГРАЦИИ ПЛАТФОРМЫ УПРАВЛЕНИЯ ИСХОДНЫМ КОДОМ GITLAB И ОБЪЕКТНОЙ СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ 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/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. ОБ ЭКСПЕРТАХ 4
2. ВВЕДЕНИЕ 5
3. АРХИТЕКТУРА РЕШЕНИЯ 6
   - 3.1. Схема потоков данных 6
   - 3.2. Требования к сети 6
4. РАЗВЕРТЫВАНИЕ GITLAB 6
   - 4.1. Подготовка системы 7
   - 4.2. Установка Postfix 7
   - 4.3. Установка GitLab CE 8
5. НАСТРОЙКА TLS/HTTPS 8
   - 5.1. Генерация самоподписанного сертификата 8
   - 5.2. Размещение сертификатов 9
   - 5.3. Настройка HTTPS в gitlab.rb 9
6. ИНТЕГРАЦИЯ СЕРТИФИКАТОВ TATLIN.OBJECT 10
   - 6.1. Структура цепочки сертификатов 10
   - 6.2. Получение и конвертация сертификата TATLIN.OBJECT 10
   - 6.3. Размещение сертификатов 10
7. ИНТЕГРАЦИЯ GITLAB С TATLIN.OBJECT 11
   - 7.1. Резервные копии (Backups) 11
   - 7.2. Git LFS 12
   - 7.3. Uploads 12
   - 7.4. Job Artifacts 12
   - 7.5. Merge Request Diffs 13
   - 7.6. CI Secure Files 13
8. ПРОВЕРКА ИНТЕГРАЦИИ 14
   - 8.1. Проверка S3-интеграции 14
   - 8.2. Проверка TLS-сертификата 14
   - 8.3. Функциональный тест LFS 15
9. РЕКОМЕНДАЦИИ ПО БЕЗОПАСНОСТИ 15
   - 9.1. Управление ключами доступа 15
   - 9.2. TLS 15
   - 9.3. Сетевая изоляция 15
10. ДИАГНОСТИКА И ТИПИЧНЫЕ ПРОБЛЕМЫ 15
11. ЗАКЛЮЧЕНИЕ 16
12. ИСТОЧНИКИ ДАННЫХ 16

ОБ ЭКСПЕРТАХ

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 со сторонним ПО
- Внедрении best practices по работе с распределенными системами хранения

ВВЕДЕНИЕ

Документ описывает полный цикл интеграции платформы управления исходным кодом GitLab Community Edition с программно-определяемым объектным хранилищем TATLIN.OBJECT.

Документ охватывает развертывание GitLab на Ubuntu, настройку TLS/HTTPS, интеграцию сертификатов СХД, конфигурацию S3-совместимого хранения для Backups, LFS, Uploads, Artifacts, Merge Request Diffs и CI Secure Files.

Современные DevOps-практики предполагают централизованное хранение артефактов жизненного цикла разработки: исходных кодов, бинарных зависимостей (Git LFS), артефактов сборки CI/CD, резервных копий и вспомогательных файлов. Размещение этих данных на локальных дисках сервера GitLab создает риски потери данных, усложняет масштабирование и резервное копирование.

Объектное хранилище TATLIN.OBJECT предоставляет S3-совместимый интерфейс, позволяющий выгрузить хранение данных за пределы GitLab-инстанса без изменения рабочих процессов разработчиков.

ВЕРСИИ ИСПОЛЬЗУЕМОГО ПО		
GitLab Community Edition	16.x и выше	Платформа DevOps
Ubuntu Server	22.04 LTS	ОС для GitLab
TATLIN.OBJECT	1.9.3	Объектное хранилище
OpenSSL	3.x	Генерация сертификатов
Nginx (встроенный)	GitLab Omnibus	HTTPS-терминация

АРХИТЕКТУРА РЕШЕНИЯ

Интеграция строится по следующей схеме: GitLab Omnibus взаимодействует с TATLIN.OBJECT через S3 API (HTTPS).

Каждый тип данных изолирован в отдельном бакете. Это упрощает управление политиками жизненного цикла объектов и IAM-правами.

Схема потоков данных

ТИП ДАННЫХ	БАКЕТ (ПРИМЕР)	ОПИСАНИЕ
Резервные копии	gitlab-backup	Полные и инкрементальные бэкапы GitLab
Git LFS	gitlab-lfs	Крупные бинарные файлы репозиториев
Uploads	gitlab-uploads	Вложения Wiki, Issues, Avatars
Job Artifacts	gitlab-artifacts	Артефакты CI/CD пайплайнов
MR Diffs	gitlab-diffs	Внешнее хранение diffs MR
CI Secure Files	gitlab-ci-secure-files	Сертификаты, ключи для CI

Требования к сети

- GitLab-сервер должен иметь IP-связность с data-интерфейсами кластера TATLIN.OBJECT.
- Порт 443 (HTTPS/S3) должен быть открыт между GitLab и TATLIN.OBJECT.
- DNS-разрешение имени кластера TATLIN.OBJECT должно совпадать с CN в его TLS-сертификате. При несоответствии необходимо добавить запись в /etc/hosts.
- Для HTTPS GitLab: порты 80 (redirect) и 443.

РАЗВЕРТЫВАНИЕ GITLAB

> **!** В данном разделе описана установка GitLab Community Edition на ОС Ubuntu 22.04 LTS. В качестве примера доменной A-записи в данной статье используется example.ru. Замените ее на актуальную A-запись своей инфраструктуры.

Подготовка системы

Шаг 1. Разверните виртуальную машину с ОС Ubuntu 22.04 LTS и создайте DNS A-запись, указывающую на IP-адрес машины.
Шаг 2. Обновите систему:

```
sudo apt-get update
```

Шаг 3. Установите зависимости:

```
sudo apt-get install -y curl openssh-server ca-certificates tzdata perl
```

Установка Postfix:

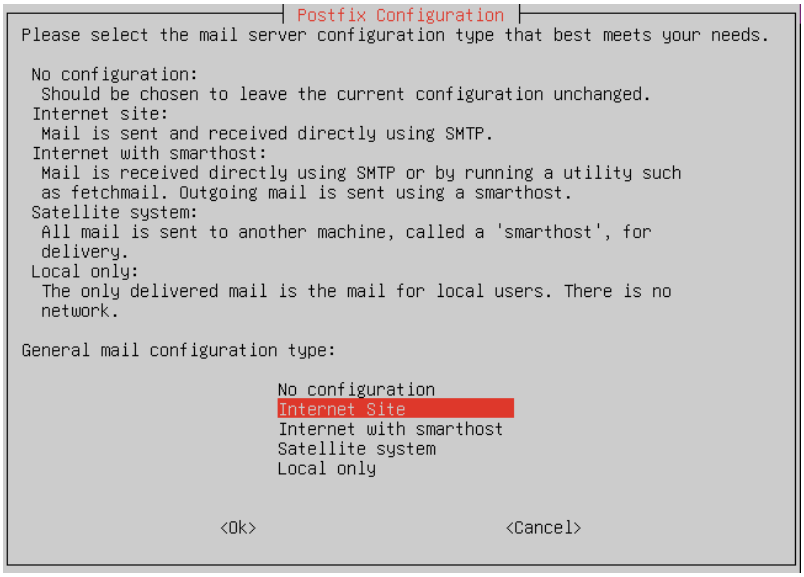
GitLab использует Postfix для отправки уведомлений по электронной почте.

Для установки Postfix выполните:

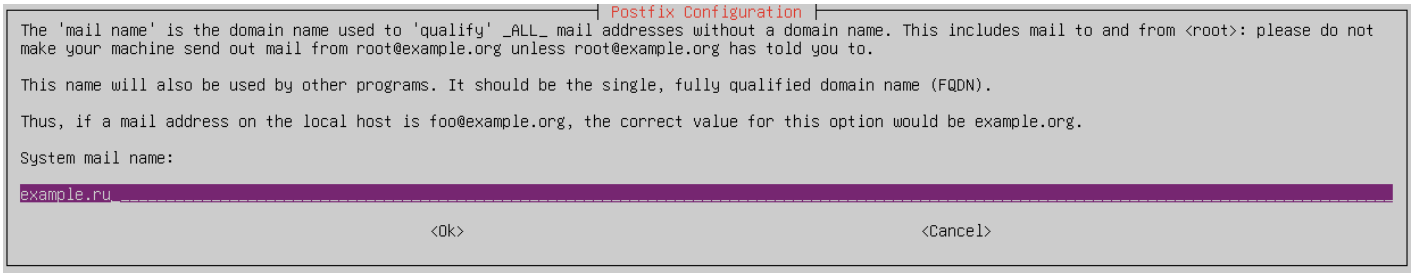
```
sudo apt-get install -y postfix
```

В процессе установки настройте:

General type of mail configuration: Internet Site.



System mail name: A-запись сервера (например, example.ru).



Установка GitLab CE

> **!** После завершения установки GitLab будет доступен по HTTP. Доступ по HTTPS настраивается на следующем этапе.

Шаг 1. Скачайте установочные файлы GitLab:

```
curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh
| sudo bash
```

Шаг 2. Запустите установку GitLab:

```
sudo EXTERNAL_URL="http://<A-RECORD>" apt-get install gitlab-ce
```

где <A-RECORD> — доменная A-запись.

Пример

```
sudo EXTERNAL_URL="http://example.ru" apt-get install gitlab-ce
```

НАСТРОЙКА TLS/HTTPS

> ! Для обеспечения безопасности передачи данных и корректной работы ряда функций (OAuth, webhooks, Git over HTTPS) GitLab должен работать по HTTPS.

Генерация самоподписанного сертификата

Если у вас нет сертификата от доверенного CA (Let's Encrypt, корпоративный PKI), сгенерируйте самоподписанный сертификат:

```
openssl req \
  -subj "/C=<COUNTRY>/ST=<REGION>/L=<CITY>/O=<COMPANY>/OU=<DEPT>/CN=<FQDN>" \
  -newkey rsa:4096 -nodes -sha256 \
  -keyout ./<CERT_NAME>.key \
  -x509 -days 3650 \
  -out ./<CERT_NAME>.crt
```

ПАРАМЕТР	ОПИСАНИЕ
<COUNTRY>	Двухбуквенный код страны (ISO 3166), например RU
<REGION>	Регион регистрации компании, например Moscow
<CITY>	Город регистрации компании
<COMPANY>	Наименование организации
<DEPT>	Подразделение, например IT
<FQDN>	FQDN GitLab-сервера (должен совпадать с A-записью)
<CERT_NAME>	Базовое имя файлов ключа и сертификата

Пример

```
openssl req \
-subj "/C=RU/ST=Moscow/L=Moscow/O=Example/OU=IT/CN=example.ru" \
-newkey rsa:4096 -nodes -sha256 \
-keyout ./example.ru-ss.key \
-x509 -days 3650 \
-out ./example.ru-ss.crt
```

Размещение сертификатов

```
sudo mkdir -p /etc/gitlab/trusted-certs /etc/ssl/gitlab
sudo cp ./*.key ./*.crt /etc/gitlab/trusted-certs/
sudo cp ./*.key ./*.crt /etc/ssl/gitlab/
sudo update-ca-certificates
```

Настройка HTTPS в gitlab.rb

Шаг 1. Откройте конфигурационный файл:

```
sudo nano /etc/gitlab/gitlab.rb
```

> ! Для поиска в nano используйте Ctrl+W. Для повторного поиска — Alt+W. Закомментированные строки необходимо раскомментировать (закомментированные строки начинаются с #).

Шаг 2. Установите параметры:

```
external_url "https://example.ru"

nginx['redirect_http_to_https'] = true
nginx['ssl_certificate'] = "/etc/ssl/gitlab/example.ru-ss.crt"
nginx['ssl_certificate_key'] = "/etc/ssl/gitlab/example.ru-ss.key"

letsencrypt['enable'] = false
```

Шаг 3. Сохраните файл **(Ctrl+X → Y → Enter)** и примените конфигурацию:

```
sudo gitlab-ctl reconfigure
```

ИНТЕГРАЦИЯ СЕРТИФИКАТОВ TATLIN.OBJECT

> ! Для установления доверенного TLS-соединения между GitLab и TATLIN.OBJECT необходимо добавить сертификаты СХД в хранилище доверенных сертификатов GitLab.

Структура цепочки сертификатов

Серверный, промежуточный и корневой сертификаты необходимо разместить как отдельные .crt-файлы в директории /etc/gitlab/trusted-certs/.

Для этого выполните:

```
/etc/gitlab/trusted-certs/
  external-service.tatlin_object_ca.crt              # серверный сертификат
  external-service.tatlin_object_ca_intermediate.crt # промежуточный CA
  external-service.tatlin_object_ca_root.crt         # корневой CA
```

Получение и конвертация сертификата TATLIN.OBJECT

Для замены <ENDPOINT_URL> на FQDN/IP data-интерфейса кластера выполните:

```
wget https://<ENDPOINT_URL>/public/tatlin_object_ca.pem --no-check-certificate
```

Чтобы получить серверный сертификат из TLS-рукопожатия, выполните:

```
openssl s_client -showcerts -connect <ENDPOINT_URL>:443 </dev/null 2>/dev/null \
  | openssl x509 -outform PEM > external-service.tatlin_object_ca.crt
```

Размещение сертификатов

Скопируйте корневой сертификат:

```
cp tatlin_object_ca.pem external-service.tatlin_object_ca_root.crt
```

Скопируйте все сертификаты в trusted-certs:

```
sudo cp external-service.tatlin_object_ca*.crt /etc/gitlab/trusted-certs/
```

Примените конфигурацию:

```
gitlab-ctl reconfigure
```

!

Убедитесь, что FQDN кластера TATLIN.OBJECT совпадает с CN сертификата. При несоответствии добавьте статическую запись:

```
echo '<IP> <FQDN>' | sudo tee -a /etc/hosts
```

ИНТЕГРАЦИЯ GITLAB С TATLIN.OBJECT

!

Все типы данных GitLab настраиваются в файле /etc/gitlab/gitlab.rb. Каждый тип требует создания отдельного бакета на кластере TATLIN.OBJECT и соответствующего блока конфигурации.
Общая процедура для каждого типа:
создайте бакет → отредактируйте gitlab.rb → выполните sudo gitlab-ctl reconfigure
Параметры подключения:
provider — всегда 'AWS' (S3-совместимый);
region — любое значение (например, eu-west-1);
path_style: true — обязательно для TATLIN.OBJECT.

Резервные копии (Backups)

Шаг 1. Создайте бакет gitlab-backup на кластере TATLIN.OBJECT.
Шаг 2. В gitlab.rb в разделе ### Backup settings укажите:

```
gitlab_rails['backup_upload_connection'] = {
  'provider'              => 'AWS',
  'region'                => 'eu-west-1',
  'aws_access_key_id'     => '<ACCESS_KEY>',
  'aws_secret_access_key' => '<SECRET_KEY>',
  'aws_signature_version' => 4,
  'endpoint'              => 'https://<ENDPOINT_URL>',
  'path_style'            => true
}

gitlab_rails['backup_upload_remote_directory'] = 'gitlab-backup'
```

Шаг 3. Примените изменения:

```
sudo gitlab-ctl reconfigure
```

Git LFS

Шаг 1. Создайте бакет gitlab-lfs.
Шаг 2. В gitlab.rb в разделе ### Git LFS укажите:

```
gitlab_rails['lfs_enabled']                       = true
gitlab_rails['lfs_object_store_enabled']          = true
gitlab_rails['lfs_object_store_proxy_download']   = true
gitlab_rails['lfs_object_store_remote_directory'] = 'gitlab-lfs'

gitlab_rails['lfs_object_store_connection'] = {
  'provider'              => 'AWS',
  'region'                => 'eu-west-1',
  'aws_access_key_id'     => '<ACCESS_KEY>',
  'aws_secret_access_key' => '<SECRET_KEY>',
  'aws_signature_version' => 4,
  'endpoint'              => 'https://<ENDPOINT_URL>',
  'path_style'            => true,
  'secure'                => true,
  'encrypt'               => false
}
```

Шаг 3. sudo gitlab-ctl reconfigure

Uploads

Шаг 1. Создайте бакет gitlab-uploads.
Шаг 2. В gitlab.rb в разделе ### GitLab uploads укажите:

```
gitlab_rails['uploads_object_store_enabled']           = true
gitlab_rails['uploads_object_store_proxy_download']    = false
gitlab_rails['uploads_object_store_remote_directory']  = 'gitlab-uploads'

gitlab_rails['uploads_object_store_connection'] = {
  'provider'              => 'AWS',
  'region'                => 'eu-west-1',
  'aws_access_key_id'     => '<ACCESS_KEY>',
  'aws_secret_access_key' => '<SECRET_KEY>',
  'aws_signature_version' => 4,
  'endpoint'              => 'https://<ENDPOINT_URL>',
  'path_style'            => true,
  'secure'                => true,
  'encrypt'               => false
}
```

Шаг 3. Примените изменения:

```
sudo gitlab-ctl reconfigure
```

Job Artifacts

Шаг 1. Создайте бакет `gitlab-artifacts`.

Шаг 2. В gitlab.rb в разделе ### Job Artifacts uploads укажите:

```
gitlab_rails['artifacts_enabled']                      = true
gitlab_rails['artifacts_object_store_enabled']         = true
gitlab_rails['artifacts_object_store_proxy_download']  = false
gitlab_rails['artifacts_object_store_remote_directory'] = 'gitlab-artifacts'

gitlab_rails['artifacts_object_store_connection'] = {
  'provider'              => 'AWS',
  'region'                => 'eu-west-1',
  'aws_access_key_id'     => '<ACCESS_KEY>',
  'aws_secret_access_key' => '<SECRET_KEY>',
  'aws_signature_version' => 4,
  'endpoint'              => 'https://<ENDPOINT_URL>',
  'path_style'            => true,
  'secure'                => true,
  'encrypt'               => false
}
```

Шаг 3. Примените изменения:

```
sudo gitlab-ctl reconfigure
```

Merge Request Diffs

Шаг 1. Создайте бакет `gitlab-diffs`.
Шаг 2. В gitlab.rb в разделе ### External merge request diffs укажите:

```
gitlab_rails['external_diffs_enabled']                      = true
gitlab_rails['external_diffs_when']                         = nil
gitlab_rails['external_diffs_object_store_enabled']         = true
gitlab_rails['external_diffs_object_store_proxy_download']  = false
gitlab_rails['external_diffs_object_store_remote_directory'] = 'gitlab-diffs'

gitlab_rails['external_diffs_object_store_connection'] = {
  'provider'              => 'AWS',
  'region'                => 'eu-west-1',
  'aws_access_key_id'     => '<ACCESS_KEY>',
  'aws_secret_access_key' => '<SECRET_KEY>',
  'aws_signature_version' => 4,
  'endpoint'              => 'https://<ENDPOINT_URL>',
  'path_style'            => true,
  'secure'                => true,
  'encrypt'               => false
}
```

Шаг 3. Примените изменения:

```
sudo gitlab-ctl reconfigure
```

CI Secure Files

Шаг 1. Создайте бакет `gitlab-ci-secure-files`.

Шаг 2. В gitlab.rb в разделе ### CI Secure Files укажите:

```
gitlab_rails['ci_secure_files_enabled']                        = true
gitlab_rails['ci_secure_files_object_store_enabled']           = true
gitlab_rails['ci_secure_files_object_store_remote_directory'] = 'gitlab-ci-secure-files'

gitlab_rails['ci_secure_files_object_store_connection'] = {
  'provider'              => 'AWS',
  'region'                => 'eu-west-1',
  'aws_access_key_id'     => '<ACCESS_KEY>',
  'aws_secret_access_key' => '<SECRET_KEY>',
  'aws_signature_version' => 4,
  'endpoint'              => 'https://<ENDPOINT_URL>',
  'path_style'            => true
}
```

Шаг 3. Примените изменения:

```
sudo gitlab-ctl reconfigure
```

ПРОВЕРКА ИНТЕГРАЦИИ

Проверка S3-интеграции

Для проверки статусов сервисов выполните:

```
sudo gitlab-ctl status
```

Для проверки логов на ошибки S3 выполните:

```
sudo gitlab-ctl tail gitlab-rails | grep -i 's3\|object\|storage\|error'
```

Для выполнения теста резервного копирования (создание бэкапа и его выгрузка в S3) выполните:

```
sudo gitlab-backup create SKIP=db
```

Проверка TLS-сертификата

Для проверки цепочки доверия до TATLIN.OBJECT выполните:

```
openssl s_client -connect <ENDPOINT_URL>:443 -CApath /etc/ssl/certs
```

Для проверки срока действия сертификата GitLab выполните:

```
openssl x509 -in /etc/ssl/gitlab/example.ru-ss.crt -noout -dates
```

Функциональный тест LFS

Загрузите тестовый файл через Git LFS:

```
git lfs track "*.bin"
dd if=/dev/urandom of=test.bin bs=1M count=10
git add test.bin .gitattributes && git commit -m 'test: lfs upload'
git push
```

Убедитесь, что файл появился в бакете `gitlab-lfs`.

РЕКОМЕНДАЦИИ ПО БЕЗОПАСНОСТИ

Управление ключами доступа

- Создайте выделенного IAM-пользователя на TATLIN.OBJECT с минимально необходимыми правами (`ListBuckets`, `GetObject`, `PutObject`, `DeleteObject`).
- Не используйте root-ключи кластера в конфигурации GitLab.
- Ротируйте ключи доступа не реже одного раза в год.
- Рассмотрите хранение ключей в HashiCorp Vault или GitLab Secrets с внедрением через переменные окружения.

TLS

- Для продуктивных сред используйте сертификаты от доверенного CA (Let's Encrypt или корпоративный PKI).
- Используйте самоподписанные сертификаты в изолированных тестовых средах.

Сетевая изоляция

- Ограничьте доступ к S3-эндпоинту TATLIN.OBJECT только с IP-адреса GitLab-сервера через ACL или firewall.
- Рассмотрите выделенный VLAN для трафика между GitLab и СХД.

ДИАГНОСТИКА И ТИПИЧНЫЕ ПРОБЛЕМЫ

СИМПТОМ	ВЕРОЯТНАЯ ПРИЧИНА	РЕШЕНИЕ
SSL certificate verify failed	Сертификат TATLIN.OBJECT не добавлен в trusted-certs	Выполните шаги раздела 6
hostname does not match	FQDN в /etc/hosts отличается от CN сертификата	Добавьте корректную запись в /etc/hosts
AccessDenied при загрузке в S3	Неверные или недостаточные права ключей	Проверьте IAM-политики на кластере TATLIN.OBJECT
NoSuchBucket	Бакет не создан или указано неверное имя	Создайте бакет и проверьте remote_directory
reconfigure завершается с ошибкой	Синтаксическая ошибка в gitlab.rb	Введите: sudo ruby -c /etc/gitlab/gitlab.rb
LFS-объекты не выгружаются в S3	Proxy download отключён, прямой доступ недоступен	Установите: lfs_object_store_proxy_download: true

ЗАКЛЮЧЕНИЕ

Интеграция GitLab с объектным хранилищем TATLIN.OBJECT позволяет:

- устранить риск потери данных при отказе локального диска GitLab-сервера;
- масштабировать хранилище артефактов независимо от вычислительных ресурсов;
- централизованно управлять политиками хранения (lifecycle policies) для каждого типа данных;
- упростить процедуру резервного копирования: бэкапы автоматически выгружаются в S3-бакет.

После выполнения всех шагов данного руководства GitLab будет использовать TATLIN.OBJECT в качестве единого бэкенда хранения для всех поддерживаемых типов данных, что обеспечивает надежность, масштабируемость и гибкость управления жизненным циклом артефактов разработки.

ИСТОЧНИКИ ДАННЫХ

1. Руководство администратора TATLIN.OBJECT
2. GitLab Omnibus: Object Storage
3. GitLab Backup and Restore