Установка стенда Pilot

Комплект разворачивает Pilot-Server в монолитном режиме. Внешние Pilot, GitLab, PostgreSQL и S3 можно подключить отдельно.

1. Как устроена установка

Документ для администратора, который ставит стек. Он описывает установку на одну машину и на несколько машин. Правила доступа и требования безопасности находятся в документе «Безопасность», работа без интернета подробно разобрана в «GitLab без интернета».

Папка Install копируется на сервер целиком. Перед установкой нужно принять три решения:

Решение	Вариант А	Вариант Б
Сколько машин	Одна машина: всё ставится одной командой	Несколько машин: на каждой свои роли, см. раздел 4
Какие данные	С текущими данными: копия стенда из папки data с теми же паролями	Чистая установка: пустые серверы, свои или случайные пароли, ключ -NoData
Есть ли интернет	Есть: Docker скачает недостающие образы	Нет: нужен комплект с образами для закрытого контура. Его готовит поставщик, см. раздел 5

2. Что нужно на сервере

Комплект работает на Windows и на Linux, команды для обеих систем приведены рядом. Для серверов, которые работают круглосуточно, особенно для прода, лучше подходит Linux: Docker Desktop запускается только после входа пользователя в Windows, не поддерживает Windows Server и в крупных организациях требует платной подписки. Установка проверена на Linux (Ubuntu 24.04, одна машина и две). Скрипт для Windows проверен по коду и самопроверкам, полный запуск на Docker Desktop не выполнялся.

Что поддерживается. Git-хостинг - только GitLab: из комплекта или свой. GitHub, Bitbucket, GitBucket и GitVerse не поддерживаются. Pilot-Server ставится в монолитном режиме, по одному серверу на контур. Кластерный режим Pilot-Server (несколько узлов и Redis) комплект не разворачивает, но стенд с ним работает: как перевести контур в кластер, описано в docs/Pilot-Server_cluster.md. Серверным модулям нужен ServerHost 1.2.0 или новее и общий каталог модулей на всех узлах, настройки раздела «Серверы» вступают в силу после перезапуска всех узлов контура.

- **Docker.** На Windows это Docker Desktop. На Linux это Docker Engine с плагином compose версии 2.20 или новее.
- **Место на диске и память.** Около 10 ГБ под образы и данные и не меньше 10 ГБ под GitLab, дальше объём растёт вместе с репозиториями и артефактами. При установке с данными добавьте размер папки data. Артефакты и загрузки GitLab, данные Grafana, Loki и Prometheus лежат на томах Docker вне DATA_ROOT. Память: GitLab требует не меньше 6 ГБ, все роли

вместе с GitLab на одной машине не меньше 16 ГБ (на машине с 10 ГБ при дополнительной нагрузке начинался своп).

- **Интернет, если образы скачиваются.** Нужен доступ к registry.ascon.ru, registry-1.docker.io, gcr.io и registry.gitlab.com (служебный образ runner), напрямую или через прокси. Образы заданий CI берутся ещё и с mcr.microsoft.com. Docker Hub использует и вспомогательные адреса (авторизация, раздача слоёв): точный перечень смотрите в документации Docker или по журналу прокси.

На сервере используются две папки, их легко перепутать:

Папка	Windows	Linux	Что в ней
Установочная	C:\PilotProAdmin\Install	/opt/cicd/Install	Скрипты, docker-compose.yml и .env со всеми паролями. Отсюда запускаются команды docker compose. После установки папку не удаляйте
Данные (DATA_ROOT)	C:\Ascon-Docker	/opt/ascon-docker	Базы PostgreSQL, лицензии и настройки Pilot, GitLab, данные админки. Эту папку нужно бэкапить

Подготовка Windows

1. Установите Docker Desktop с WSL2.
2. Docker Desktop с WSL2 берёт память по настройке WSL, а не по ползунку Settings / Resources: лимит задаётся в файле %UserProfile%\.wslconfig (раздел [wsl2], параметр memory), по умолчанию это часть памяти компьютера, обычно половина. Для GitLab на этой же машине нужно не меньше 6 ГБ, для всех ролей вместе не меньше 16 ГБ. После правки выполните wsl --shutdown и перезапустите Docker Desktop. Прокси настраивается в Settings / Resources, раздел Proxies.
3. В разделе Settings / General включите запуск при входе в систему. Пока пользователь не вошёл в Windows, сервисы не работают.
4. Запускайте установку из PowerShell от имени администратора при запущенном Docker Desktop.

Подготовка Linux

Установите Docker и включите его автозапуск. Официальный скрипт подходит для Ubuntu, Debian и RHEL-совместимых систем:

> curl -fsSL https://get.docker.com | sudo sh
>
> sudo systemctl enable --now docker
>
> docker compose version

На дистрибутивах, которые скрипт не поддерживает, ставьте Docker из репозитория дистрибутива и проверьте версию compose: с версией ниже 2.20 установка не запустится. Без интернета принесите пакеты docker-ce, docker-ce-cli, containerd.io и docker-compose-plugin с download.docker.com.

Скопируйте комплект, например командой scp -r Install user@10.0.0.5:/tmp/, затем выполните sudo mkdir -p /opt/cicd && sudo mv /tmp/Install /opt/cicd/. Бит исполнения после копирования с Windows не нужен: скрипт запускается через bash.

Если нужен прокси, создайте файл /etc/systemd/system/docker.service.d/http-proxy.conf:

> [Service]
>
> Environment="HTTP_PROXY=http://proxy:3128" "HTTPS_PROXY=http://proxy:3128" "NO_PROXY=localhost,127.0.0.1"

Примените настройки:

> sudo systemctl daemon-reload
>
> sudo systemctl restart docker

Чем отличаются Windows и Linux

Что	Windows	Linux
Запуск установки	powershell -ExecutionPolicy Bypass -File .\install.ps1 -Roles prod	sudo bash install.sh --roles prod
Ключи	-NoData, -GitLabHost, -FromBackup	--no-data, --gitlab-host, --from-backup
Папка данных по умолчанию	C:/Ascon-Docker	/opt/ascon-docker. Если в .env остался путь с C: или DATA_ROOT пуст, скрипт запишет этот путь сам
Запуск после перезагрузки	После входа пользователя, если включено в настройках Docker Desktop	Сам, через systemd, вход пользователя не нужен
Память	Ограничена настройкой WSL2 (файл .wslconfig), а не только Docker Desktop	Без ограничения, доступна вся память машины
Права на папки данных	Права на DATA_ROOT установка не меняет: закройте доступ средствами Windows. Права на .env и PASSWORDS.md она ограничивает: доступ у пользователя, SYSTEM и администраторов	Админка и миграция работают от пользователя с UID 1654, скрипт сам выдаёт ему нужные каталоги, в том числе databases контуров test и stage. Каталоги с секретами в DATA_ROOT закрываются от остальных пользователей, .env и PASSWORDS.md получают права 600. Данные PostgreSQL принадлежат UID 999, поэтому читать их и делать бэкап нужно через sudo

Что	Windows	Linux
Кто запускает docker	Пользователь Docker Desktop	sudo или группа docker. Участие в группе равносильно правам root
Закрытие портов	Брандмауэр Windows	**Правила ufw и firewalld не закрывают опубликованные порты Docker.** Нужна цепочка DOCKER-USER, см. ниже
Сетевые папки для миграции	Docker Desktop штатно их не подключает	Подключаются через mount, см. раздел 4
Мониторинг	node-exporter и cAdvisor показывают виртуальную машину WSL, а не сам компьютер. Журнал входов в систему не собирается	Показывают настоящую машину. Promtail читает /var/log/auth.log или /var/log/secure. cAdvisor 0.49.1 на Docker с хранилищем образов containerd (по умолчанию в Docker 29) не определяет имена контейнеров: дашборд «Нагрузка контейнеров» пуст, нагрузка машины и логи работают
SELinux	Нет	В режиме enforcing контейнеры не получат доступ к папкам, см. ниже
Зависание при установке	Скрипт ждёт PostgreSQL до трёх минут и GitLab до 30 минут и завершается с ошибкой по таймауту	Скрипт ждёт PostgreSQL без ограничения, GitLab до 30 минут. Если шаг идёт слишком долго, смотрите docker logs gitlab во втором терминале

Порты на Linux. Docker добавляет свои правила раньше правил ufw и firewalld, поэтому порт останется открытым, даже если он запрещён в них. Ограничивайте доступ в цепочке DOCKER-USER. Пример: PostgreSQL прода (порт 5433) на внешнем интерфейсе eth0 доступен только машине 10.0.0.2 (test, stage) и машине 10.0.0.3 (admin, metrics, gitlab). Правила вставляются в начало цепочки, поэтому сначала запрещающее, затем разрешающие: они сработают первыми. Имя интерфейса возьмите из вывода ip -br addr.

> sudo iptables -I DOCKER-USER -i eth0 -p tcp -m conntrack --ctorigdstport 5433 --ctdir ORIGINAL -j DROP
>
> sudo iptables -I DOCKER-USER -i eth0 -p tcp -m conntrack --ctorigdstport 5433 --ctdir ORIGINAL -s 10.0.0.2 -j RETURN
>
> sudo iptables -I DOCKER-USER -i eth0 -p tcp -m conntrack --ctorigdstport 5433 --ctdir ORIGINAL -s 10.0.0.3 -j RETURN

Сохраните правила средствами дистрибутива, например через пакет iptables-persistent, и повторите их для остальных служебных портов (RustFS, миграция, Prometheus, Loki, 9100, 9101) с источниками из таблицы раздела 4. Если задан INTERNAL_BIND, в конце установки печатается список служебных портов машины. Проверяйте результат подключением с другой машины.

Готовые правила. Вместо ручных команд используйте файл, который установка пишет сама. Если в .env задан INTERNAL_BIND, в установочной папке появляется firewall-linux.sh (на Windows firewall-windows.ps1) с правилами для всех служебных портов этой машины: они пускают на порты только адреса из PROD_HOST, TEST_HOST, STAGE_HOST, METRICS_HOST, ADMIN_HOST (адрес машины admin, если она другая), GITLAB_HOST_IP и саму машину, остальным закрывают. Порты админки, Grafana, Pilot и GitLab файл не трогает. Установка файл только создаёт и ничего не применяет: просмотрите его, проверьте командой sh firewall-linux.sh --test, примените командой sudo sh firewall-linux.sh (Windows: PowerShell от администратора, .\firewall-windows.ps1). Снять правила можно ключом --remove (Windows: -Remove). В правила входят только IP-адреса: имена хостов установка пропускает и предупреждает. Правила Linux пропадают после перезагрузки: запускайте файл при старте системы, после Docker.

HTTPS для админки и Grafana (необязательно). По умолчанию они работают по HTTP, и пароль входа виден в сети. Чтобы включить HTTPS, положите в установочную папку файлы certs/tls.crt (сертификат) и certs/tls.key (закрытый ключ без пароля), оба в формате PEM, и запустите установку: админка и Grafana ответят по HTTPS на тех же портах, HTTP на них перестанет работать. Нет файлов: всё работает по HTTP, как раньше. Самоподписанный сертификат для закрытой сети создаётся так (подставьте свои имена и IP):

```
openssl req -x509 -newkey rsa:2048 -nodes -days 825 -keyout certs/tls.key -out certs/tls.crt -subj "/CN=cicd" -addext "subjectAltName=DNS:localhost,DNS:host.docker.internal,IP:10.0.0.3"
```

В сертификате должны быть имена и адреса, по которым открывают админку и Grafana. После включения HTTPS задайте в .env ADMIN_PUBLIC_URL=https://…, в GitLab (Admin / Applications) смените Redirect URI приложения админки на https://<админка>/GitLab/Callback, а в переменных групп модулей задайте ADMIN_URL=https://<админка>:5300. Самоподписанный сертификат задания CI не знают: добавьте в переменные группы переменную типа File CURL_CA_BUNDLE (для dotnet ещё и SSL_CERT_FILE) с содержимым tls.crt. GitLab и Pilot-Server комплект на HTTPS не переключает: GitLab настраивается по его документации (external_url 'https://…', сертификат в config/ssl), а порядок настройки сертификата Pilot-Server в его образе не описан, уточните его у поставщика.

SELinux. Проверьте режим командой getenforce. При значении Enforcing разметьте папку данных и установочную папку:

```
sudo chcon -Rt container_file_t /opt/ascon-docker /opt/cicd/Install
```

Эта команда размечает только папки данных и установочную. Сервисы мониторинга (promtail, cadvisor, node-exporter) и runner монтируют docker.sock, /proc, /sys и /var/log хоста: при SELinux в режиме enforcing им, вероятно, понадобится отдельная политика. Комплект на такой системе не проверялся, решение согласуйте с ИБ.

Разные системы на разных машинах. Их можно смешивать. Файл .env при копировании подходит как есть, но на Windows путь DATA_ROOT нужно поправить вручную: на Linux скрипт исправляет только путь с C:.

3. Установка на одну машину

С текущими данными

Вариант работает, если в комплекте есть папка data. Допустим, комплект лежит в C:\PilotProAdmin\Install, а IP сервера 10.0.0.5. Откройте PowerShell от имени администратора и запустите установку:

```
cd C:\PilotProAdmin\Install

powershell -ExecutionPolicy Bypass -File .\install.ps1 -GitLabHost 10.0.0.5
```

На Linux:

```
cd /opt/cicd/Install

sudo bash install.sh --gitlab-host 10.0.0.5
```

Файл .env создавать не нужно: скрипт соберёт его из data. Пароли обязаны совпасть с бэкапом, потому что они записаны в settings.xml, ролях PostgreSQL и данных RustFS. Логины и пароли лежат в файле PASSWORDS.md. Установка занимает от 15 до 30 минут, на медленном диске дольше. Самая долгая часть это GitLab: установка ждёт его настоящей готовности (запущен puma и отвечает /-/readiness), до 30 минут, затем останавливает puma и sidekiq и восстанавливает копию. После восстановления проверяются код возврата gitlab-backup restore, строки об ошибках схемы (предупреждение) и число проектов в базе: ноль проектов считается ошибкой установки. Затем GitLab корректно останавливается (docker stop -t 600), запускается и снова ждётся до готовности, и только после этого печатается «Готово». Журнал восстановления лежит в DATA_ROOT/gitlab/logs/restore.log.

- **Пустая папка данных.** DATA_ROOT должна быть пустой. После восстановления появляется метка .installed, поэтому повторно данные не загружаются. Перед восстановлением GitLab ставится метка .restore-started: если установка оборвалась после неё, повторный запуск откажется восстанавливать данные второй раз (изменённые данные были бы затёрты). Очистите DATA_ROOT и запустите установку снова. Если .installed есть, а .env рядом со скриптом нет, установка остановится, а не создаст .env из шаблона: верните .env прежней установки. Данные Grafana из копии не распаковываются поверх непустого тома (имя тома берётся из COMPOSE_PROJECT_NAME, по умолчанию cicd): установка остановится с подсказкой. Файлы RustFS установка разворачивает сама: архив rustfs\rustfs_<контур>.tgz из копии она проверяет и распаковывает в том <проект>_rustfs_<контур> до запуска Pilot-Server и до создания бакета. Том должен быть пустым: если в нём уже есть данные, установка остановится и ничего не распакует. То же верно для папки data из комплекта.

- **Свежий бэкап.** Чтобы взять данные не из data, укажите каталог ночного бэкапа: -FromBackup D:\Backup\2026-11-03_0300, на Linux --from-backup. Работает и на Linux без PowerShell. Файлы RustFS из такого бэкапа установка разворачивает сама, поэтому для новой машины это рекомендуемый путь. Ручной restore.ps1 -Parts rustfs после установки потребует ключ -ConfirmOverwrite rustfs: том уже не пуст.

- **Другая папка данных.** Скопируйте .env.example в .env и перед запуском поменяйте строку DATA_ROOT.

Если IP отличается от исходной машины, проверьте вручную:

1. **OAuth-приложение GitLab.** Его Redirect URI хранится в базе GitLab: Admin / Applications. Укажите там адрес админки с путём /GitLab/Callback и задайте в .env ADMIN_PUBLIC_URL (например, http://10.0.0.5:5300): из него админка строит адрес возврата, иначе берётся адрес запроса. На машине с ролью gitlab это делает шаг GitLab после установки: если в .env заданы GITLAB_OAUTH_CLIENT_ID и ADMIN_PUBLIC_URL (или GITLAB_OAUTH_REDIRECT_URI), он добавляет адрес возврата админки в приложение, прежние адреса остаются. Иначе в конце установки с -GitLabHost печатается напоминание. Адрес админки и адрес GitLab должны открываться из браузеров пользователей: при входе браузер переходит на страницу GitLab и возвращается в админку.

2. **Адреса в CI.** Скрипт заменяет старый IP в .env, настройках админки, config.toml runner и external_url GitLab. На машине с ролью gitlab шаг после установки заменяет его ещё и в nuget.config репозиториев групп модулей и группы отчётов: в ветках dev, main, master, release/* и hotfix/* меняется только адрес в значениях с /api/v4/, коммит идёт с [skip ci]. В базе GitLab (например, в вебхуках), истории заданий, тегах, ветках задач и в собранных архивах модулей адрес остаётся: проверьте вебхуки, переменные групп, .gitlab-ci.yml и nuget.config в этих местах.

3. **Клоны у разработчиков.** Нужны git remote set-url на новый адрес и git pull: nuget.config в ветках, которые поправил установщик, обновится сам, в остальных ветках поправьте адрес вручную.

4. **Лицензии Pilot.** Копируется содержимое папок license из данных, если оно там есть. В присланном комплекте эти папки могут быть пустыми: проверьте license каждого контура. Если лицензия привязана к машине, понадобится новая.

5. **Адреса в модулях.** Статистика и отправка отчётов в GitLab продолжат обращаться к старому IP, см. раздел 6.

Затем проверьте настройки админки и статистику, см. раздел 6.

Чистая установка

Файл .env можно не создавать: установка возьмёт .env.example и при установке на одну машину сама заменит пароли-заглушки (Pilot, PostgreSQL, RustFS, Grafana, root GitLab) случайными. Она же создаст пустые ключи (MIGRATION_API_KEY, COLLECTOR_API_KEY, ключи выкладки). Сгенерированные значения лежат в .env, их список печатается в конце установки. Если нужны свои пароли, создайте .env и задайте их до запуска:

```
cd C:\PilotProAdmin\Install

copy .env.example .env

notepad .env
```

Задайте эти параметры (имена MINIO_USER и MINIO_PASSWORD исторические: это ключ и секрет доступа RustFS):

```
DATA_ROOT=C:/Ascon-Docker

PILOT_ADMIN_PASSWORD=<пароль администратора Pilot>

PG_SUPERUSER_PASSWORD=<пароль postgres>
```

> PILOT_PG_PASSWORD=<пароль pilotuser>
>
> MINIO_PASSWORD=<не короче 8 символов>
>
> GRAFANA_ADMIN_PASSWORD=<пароль Grafana>
>
> GITLAB_ROOT_PASSWORD=<пароль root GitLab, если ставите GitLab>
>
> Комментарий пишите только на отдельной строке: в строке со значением всё после # станет частью пароля. Используйте буквы, цифры и -_.. Символы $ # " ' и пробел ломают .env.

На нескольких машинах заглушки сами не заменяются: пока в .env остались значения из .env.example, установка остановится и назовёт эти параметры. Задайте пароли одинаково на всех машинах, см. раздел 4.

Запустите установку. Если своего GitLab нет, добавьте -WithGitLab. Для Pilot-myAdmin Web на test и stage добавьте -WithMyAdmin.

> powershell -ExecutionPolicy Bypass -File .\install.ps1 -NoData

На Linux:

> sudo cp .env.example .env
>
> sudo nano .env
>
> sudo bash install.sh --no-data

Права на .env (на Linux 600, на Windows доступ только у пользователя, SYSTEM и администраторов) и на PASSWORDS.md установка выставляет сама после каждой записи.

После установки выполните три шага:

1. **Лицензии Pilot.** Скопируйте файл лицензии в папку контура и перезапустите контейнер docker restart pilot-server-prod. На Windows это C:\Ascon-Docker\prod\license\Pilot Server\License\, на Linux /opt/ascon-docker/prod/license/Pilot Server/License/. Для test и stage действия те же.

2. **База Pilot.** Создайте или подключите pilot-ice_ru на каждом контуре. Порядок описан в docs/Pilot-Server_Docker_Windows.md или docs/Pilot-Server_Docker_Linux.md: разделы 5 и 7.3-7.7, затем настройка S3 (разделы 8.3, 9 или 10).

3. **GitLab.** Создайте группы автоматизаций, отчётов и модулей и пропишите их в .env. Зарегистрируйте runner командой ниже. Примените настройки повторным запуском установки: она выполнит docker compose up -d и шаг GitLab после установки. На машине с ролью gitlab этот шаг задаст в группах модулей переменные DEPLOY_KEY_TEST (Masked), DEPLOY_KEY_STAGE и DEPLOY_KEY_PROD (Masked и Protected) значениями MODULES_DEPLOY_KEY_TEST, _STAGE, _PROD из .env, защитит ветки main, master, release/* и dev в проектах этих групп (см. «Защита веток» ниже) и закроет саморегистрацию; шаг идемпотентен. Ему нужны ключи MODULES_DEPLOY_KEY_* в .env машины с GitLab (при нескольких машинах скопируйте их с машины admin) и пути групп (MODULES_GITLAB_GROUP_URL, SERVER_MODULES_GITLAB_GROUP_URL). Если установка предупредила, что шаг пропущен или не удался, или GitLab стоит не на машине

установки, задайте то же вручную: переменные групп модулей, защиту веток и закрытие регистрации (Admin / Settings / General / Sign-up restrictions).

Токен для регистрации: Admin / CI/CD / Runners / New instance runner, Docker, токен glrt-…. Команда выполняется в установочной папке. Ключ --docker-extra-hosts host.docker.internal:host-gateway нужен, чтобы задания CI находили админку по адресу по умолчанию; образ задания по умолчанию лежит на runner (if-not-present):

> docker compose exec gitlab-runner gitlab-runner register --non-interactive --url http://gitlab --clone-url http://gitlab --token <токен> --executor docker --docker-image alpine:3.20 --docker-network-mode gitlab_gitlab-net --docker-pull-policy if-not-present --docker-extra-hosts host.docker.internal:host-gateway --description local-docker-runner

Затем заполните настройки админки и статистики, см. раздел 6, и установите модули, см. ниже.

Модули при чистой установке

В пустом GitLab нет исходников модулей, а в новых базах Pilot нет самих модулей. Без них не работают отчёты, статистика клиентских модулей и серверные модули автоматизаций. Всё нужное лежит в папке modules комплекта, её заполняет сборка комплекта. Если папки нет, запросите у поставщика комплект с ней: без неё чистая установка не получит ни загрузчика серверных модулей, ни исходников, ни пакетов:

- **ServerHost.zip.** Загрузчик серверных модулей. Это единственный модуль, который ставится в Pilot вручную: остальные серверные модули он получает от админки и обновляет без перезапуска сервера.
- **sources\Pilot.ClientSdk.bundle.** Исходники общего клиентского модуля: ветка main с тегами версий. Остальные клиентские модули работают через него.
- **sources\Reports.bundle.** Исходники модуля отчётов, тоже ветка main с тегами.
- **Папка packages.** Пакеты Pilot.* в файлах .nupkg, без которых CI модулей не соберётся. В новом GitLab их нет.

Сначала закончите раздел 6: в админке должны быть заданы адреса серверов Pilot и групп GitLab, а в .env заполнены MODULES_DEPLOY_KEY_TEST, _STAGE, _PROD (пустые установка создаёт сама) и учётки MODULES_DEPLOY_<КОНТУР>_*. Те же ключи внесите в переменные группы клиентских модулей GitLab: DEPLOY_KEY_TEST (Masked), DEPLOY_KEY_STAGE и DEPLOY_KEY_PROD (Masked и Protected). Понадобятся git на машине администратора и учётная запись GitLab с правами Owner в группе клиентских модулей. В примерах GitLab находится на 10.0.0.3, группа модулей называется client-modules, комплект лежит в C:\PilotProAdmin\Install. На Linux команды те же, путь /opt/cicd/Install/modules/….

1. **ServerHost.** Войдите в клиент Pilot администратором базы контура, откройте «Модули расширения» и добавьте modules\ServerHost.zip как расширение Pilot-Server. Pilot-Server загружает его только при старте базы, поэтому перезапустите контейнер командой docker restart pilot-server-test. Повторите на stage и prod.
2. **Проекты в GitLab.** В группе клиентских модулей создайте два пустых проекта без README: pilot-clientsdk и module-for-report. Имена могут быть другими, тогда поправьте их в командах ниже.

3. **Исходники Pilot.ClientSdk.** Загрузите ветку main с тегами и создайте от неё ветки dev и release. Параметр ci.skip не даёт CI запуститься раньше времени:

> git clone C:\PilotProAdmin\Install\modules\sources\Pilot.ClientSdk.bundle Pilot.ClientSdk
>
> cd Pilot.ClientSdk
>
> git describe --tags --abbrev=0
>
> git remote set-url origin http://10.0.0.3:8080/client-modules/pilot-clientsdk.git
>
> git push -o ci.skip origin main --tags
>
> git push -o ci.skip origin main:dev main:release/<версия>

Команда git describe покажет версию, например v2.3.2: тогда ветка называется release/2.3.2.

1. **Защита веток Pilot.ClientSdk.** До запуска pipeline откройте в проекте Settings / Repository / Protected branches и защитите ветки так, как описано в абзаце «Защита веток» ниже. Без защиты main и release/* Protected-ключи стейджа и прода задания не получат, и deploy_stage и deploy_prod завершатся с ошибкой. Если проект уже лежал в группе модулей, когда выполнялся шаг GitLab после установки, защита стоит; в проекте, созданном позже, повторите запуск установки или задайте защиту вручную.

2. **Выкладка Pilot.ClientSdk.** В проекте откройте Build / Pipelines / Run pipeline и запустите pipeline по очереди для веток dev, release/<версия> и main. Задания deploy_test, deploy_stage и deploy_prod создадут модуль на тесте, стейдже и проде: каждое шлёт ключ своего контура и токен задания. Задания nuget_dev, nuget_rc и nuget_release тем же pipeline публикуют пакет Pilot.ClientSdk в реестр проекта (dev: X.Y.Z-dev.<номер pipeline>, release/*: X.Y.Z-rc.<номер>, main: X.Y.Z после deploy_prod). Предупреждение check_version на main о том, что версия не выросла, здесь ожидаемо. Новый модуль создаётся никому не назначенным: на каждом контуре назначьте его пользователям в Pilot-myAdmin.

3. **Пакеты NuGet.** Загрузите каждый файл из modules\packages в реестр проекта pilot-clientsdk. Нужен Personal Access Token со scope api. Вместо <ID> подставьте Project ID проекта, он указан в его настройках:

> curl.exe --user <логин>:<токен> -X PUT --form "package=@C:\PilotProAdmin\Install\modules\packages\\\<пакет>.nupkg" http://10.0.0.3:8080/api/v4/projects/\<ID>/packages/nuget/

Затем откройте заданиям CI доступ к реестру: в проекте pilot-clientsdk раздел Settings / CI/CD / Job token permissions, добавьте группу клиентских модулей. Заданиям нужно разрешить и читать пакеты, и публиковать их: задания nuget_* отправляют пакет с CI_JOB_TOKEN. Результат виден в разделе Deploy / Package registry.

1. **Исходники Reports.** Загрузите их так же, но без ветки release:

> git clone C:\PilotProAdmin\Install\modules\sources\Reports.bundle Reports
>
> cd Reports
>
> git remote set-url origin http://10.0.0.3:8080/client-modules/module-for-report.git

> git push -o ci.skip origin main --tags
>
> git push -o ci.skip origin main:dev

1. **Адреса в Reports.** Как есть модуль выкладывать нельзя: в поставляемом Reports.config.json адреса пусты, а источник cicd в nuget.config указан образцом (gitlab.example.local, проект 9). Пока statsUrl пуст, статистика выключена, пока пуст reportsGitLabGroupUrl, отправка отчётов в GitLab недоступна, без gitLabOAuthClientId нет кнопки «Войти через браузер». Выпущенные ранее версии модуля собраны с адресами исходной машины: выпустите новую версию со своими адресами. Сделайте обычную задачу (после защиты веток, см. ниже): ветку от dev, merge request в dev, затем выпуск на стейдж и прод на вкладке «Модули» админки. В ветке поправьте два файла:

- nuget.config: адрес источника cicd http://10.0.0.3:8080/api/v4/projects/\<ID>/packages/nuget/index.json, где <ID> это проект pilot-clientsdk.
- Reports.config.json: statsUrl, reportsGitLabGroupUrl и gitLabOAuthClientId. Что в них писать и как создать OAuth-приложение, описано в разделе 6, «Файл настроек Reports».

Модуль, которого не было на контуре, админка тоже создаёт никому не назначенным: назначьте его в Pilot-myAdmin.

1. **Защита веток.** В обоих проектах откройте Settings / Repository / Protected branches. main: Allowed to push - No one, Allowed to merge - Maintainers. release/*: Allowed to push - Maintainers, Allowed to merge - Maintainers (админка сама создаёт ветку release/X.Y.Z и коммитит в неё версию и CHANGELOG.md, при «No one» GitLab это запретит). dev: Allowed to push - Maintainers, Allowed to merge - Maintainers (при подключении модуля админка коммитит .gitlab-ci.yml в dev, при «No one» GitLab это запретит). Защита main и release/* обязательна: на ней держатся Protected-переменные ключей стейджа и прода, их получают только задания защищённых веток. В проектах групп модулей на машине с ролью gitlab защиту main, master и release/* ставит установщик (main и master: push No one, merge Maintainers; release/*: push и merge Maintainers); остальные проекты защитите вручную. В проекте Reports защитите ветки сразу после загрузки main и dev, до первого pipeline. Проекты, которые создаёт админка, получают настройку Pipelines must succeed; в проектах, созданных вручную, включите её в Settings / Merge requests сами.

Проверка:

1. В админке на вкладке «Модули» у Pilot.ClientSdk и Reports на всех контурах стоит версия из main.
2. На вкладке «Серверные модули» каждый контур показывает версию ServerHost.
3. В клиенте Pilot у назначенного пользователя в меню трея есть пункты «Модули …», например «Модули версии», а отчёт строится.

4. Установка на разные машины

Сервисы можно разнести по машинам, например прод на одной, тест на другой, админку на третьей. На каждой машине ставится тот же комплект, а ключ -Roles выбирает её сервисы.

Роли

Роль	Что запускается	Порты для людей	Служебные порты
prod	Pilot-Server, PostgreSQL, RustFS, Pilot-myAdmin Web	Pilot 5546, myAdmin 5601	PostgreSQL 5433, RustFS 9000 и 9001
test	То же, что на prod, и сборщик статистики автоматизаций. myAdmin ставится по ключу -WithMyAdmin	Pilot 5547, myAdmin 5602	PostgreSQL 5434, RustFS 9010 и 9011
stage	Как у test	Pilot 5549, myAdmin 5603	PostgreSQL 5435, RustFS 9020 и 9021
admin	Веб-админка и сервис миграции	Админка 5300	Миграция 5320
metrics	Grafana, Prometheus, Loki, сборщик статистики отчётов и модулей	Grafana 3000, сборщик 5310	Prometheus 9090, Loki 3100
gitlab	GitLab CE и GitLab Runner	GitLab 8080, ssh 2224	-

Роли можно совмещать, но один контур нельзя делить между машинами: его Pilot-Server, PostgreSQL и RustFS всегда работают вместе. Скрипт сам добавляет агенты мониторинга (node-exporter на порту 9100, cAdvisor на порту 9101, Promtail), а также мосты к сервисам на других машинах.

Пример на три машины

	Машина 1	Машина 2	Машина 3
Адрес	10.0.0.1	10.0.0.2	10.0.0.3
Роли	prod	test,stage	admin,metrics,gitlab
Кто работает	Все пользователи Pilot	Разработчики и тестировщики	Разработчики, администраторы, руководители
Система	Windows или Linux	Windows или Linux	Linux, если нужны миграция и раздел «Серверы»

Как машины связаны

Сервисы обращаются друг к другу по именам контейнеров, например postgres-prod:5432. Если контейнер находится на другой машине, это имя принимает маленький контейнер-мост link-prod (nginx в режиме TCP-прокси) и пересылает соединение на порт той машины. Поэтому строки подключения и настройки миграции менять не нужно.

Откуда	Куда, порт	Зачем
Машина admin	Контуры: Pilot 5546, 5547, 5549	Админка и миграция работают с Pilot
Машина admin	Контуры: PostgreSQL 5433-5435, RustFS 9000, 9010, 9020	Миграция копирует данные

Откуда	Куда, порт	Зачем
Машина admin	Контуры: сетевая папка данных (SMB 445 или NFS 2049)	Миграция и раздел «Серверы» читают настройки контуров
Машина admin	GitLab 8080	Ветки, merge request и pipeline; проверка токена задания при выкладке из CI
GitLab Runner	Админка 5300	Выкладка модулей из CI: ключ контура и токен задания
Машины test, stage, metrics	Прод: PostgreSQL 5433	Статистика хранится в базе pilot_metrics на проде
Машина metrics	Все машины: 9100 и 9101	Prometheus снимает нагрузку
Все машины	Машина metrics: Loki 3100	Promtail отправляет логи
Контуры и рабочие места	Машина metrics: сборщик 5310	Статистика отчётов и модулей

Порядок установки

1. **Скопируйте папку** Install на каждую машину. Если установка с данными, заполните .env на каждой машине до запуска: скрипт дополнит его паролями из data, а адреса сохранит.
2. **Создайте файл настроек** .env из .env.example и задайте адреса машин. Они одинаковы на всех машинах, мост строится только там, где нужной роли нет. INTERNAL_BIND на каждой машине свой, это IP этой машины, на нём слушают служебные порты:

> PROD_HOST=10.0.0.1
>
> TEST_HOST=10.0.0.2
>
> STAGE_HOST=10.0.0.2
>
> METRICS_HOST=10.0.0.3
>
> GITLAB_HOST_IP=10.0.0.3
>
> INTERNAL_BIND=10.0.0.1
>
> При чистой установке на нескольких машинах пароли-заглушки сами не заменяются: пока в .env остались значения из .env.example (Pilot, PostgreSQL, RustFS, Grafana, root GitLab), установка остановится и назовёт их. Задайте пароли в .env первой машины, затем скопируйте его на остальные и поменяйте только INTERNAL_BIND. Пустой MIGRATION_API_KEY скрипт создаёт при первом запуске, пустые ключи выкладки MODULES_DEPLOY_KEY_TEST, _STAGE, _PROD только на машине admin. COLLECTOR_API_KEY на нескольких машинах не создаётся: задайте его заранее, иначе приём статистики остаётся без ключа, и установка предупредит об этом.

1. **Запустите установку** на каждой машине. Начните с контуров, затем metrics, в конце admin и gitlab.

Windows, машина 1:

> powershell -ExecutionPolicy Bypass -File .\install.ps1 -Roles prod -GitLabHost 10.0.0.3

Linux, машины 2 и 3:

> sudo bash install.sh --roles test,stage --gitlab-host 10.0.0.3
>
> sudo bash install.sh --roles admin,metrics,gitlab --gitlab-host 10.0.0.3

При чистой установке добавьте -NoData или --no-data. Роли запоминаются в .env, при обновлении их можно не указывать.

Что настроить между машинами

1. **Сетевые папки для админки.** Миграции нужны pilot-server/settings, databases и server-settings каждого контура, а разделу «Серверы» только server-settings. Откройте их для машины admin. Миграция в прод ничего не пишет (её тома прода смонтированы только для чтения), но раздел «Серверы» сохраняет server-settings в том числе прода: если настройки прода будут править из админки, смонтируйте прод без ro, иначе сохранение закончится ошибкой записи. Пример для Linux:

> sudo mount -t cifs //10.0.0.1/prod /mnt/cicd/prod -o uid=1654,gid=1654,credentials=/root/.smb
>
> sudo mount -t cifs //10.0.0.2/test /mnt/cicd/test -o uid=1654,gid=1654,credentials=/root/.smb
>
> sudo mount -t cifs //10.0.0.2/stage /mnt/cicd/stage -o uid=1654,gid=1654,credentials=/root/.smb

Параметры uid и gid нужны потому, что админка и миграция работают не от root. Чтобы папки подключались после перезагрузки, добавьте их в /etc/fstab с параметром _netdev. Если контур на Linux раздаёт папки по NFS, у владельца на сервере должен быть UID 1654.

Пропишите пути в .env машины admin и примените их командой docker compose up -d:

> PROD_ROOT=/mnt/cicd/prod
>
> TEST_ROOT=/mnt/cicd/test
>
> STAGE_ROOT=/mnt/cicd/stage
>
> Docker Desktop на Windows штатно не подключает сетевые папки в контейнеры, поэтому для миграции и раздела «Серверы» ставьте admin на Linux. Без сетевых папок задайте SERVER_SETTINGS_ROOT= с пустым значением: раздел «Серверы» отключится, а настройки сервера придётся править в server-settings.conf на машине контура. Миграция в этом режиме не работает, остальные функции админки работают.

1. **Нагрузка других машин в Grafana.** В monitoring/prometheus/prometheus.yml на машине metrics добавьте адреса в targets: 10.0.0.1:9100 и 10.0.0.2:9100 для node-exporter, 10.0.0.1:9101 и 10.0.0.2:9101 для cAdvisor. Затем перечитайте конфигурацию командой docker kill -s SIGHUP prometheus. Управление Prometheus по HTTP отключено, команда curl -X POST http://localhost:9090/-/reload не работает. Папка monitoring/prometheus смонтирована в контейнер целиком, поэтому правку файла любым редактором эта команда подхватывает. В логах Loki у каждой строки есть метка host: LOG_HOST из .env, иначе INTERNAL_BIND; по ней различаются одноимённые контейнеры разных машин, например {host="10.0.0.2", container="pilot-server-test"}.

2. **Настройки админки и адрес сборщика статистики.** Они описаны в разделе 6.
3. **Адрес админки в CI.** В группах модулей GitLab задайте переменную ADMIN_URL=http://10.0.0.3:5300. Для входа через GitLab OAuth задайте в .env машины admin ADMIN_PUBLIC_URL=http://10.0.0.3:5300. Админка проверяет токен задания обращением к GitLab из своего контейнера: адрес группы GitLab в настройках должен быть доступен оттуда.
4. **Сеть.** Откройте порты из таблицы выше и закройте остальные служебные порты. Служебные порты слушают сеть без аутентификации: при заданном INTERNAL_BIND установка пишет файл правил брандмауэра (firewall-linux.sh или firewall-windows.ps1), печатает путь к нему и ничего не применяет; как его проверить и применить, описано в разделе 2. Если машина admin не та, на которой вы запускаете установку, задайте её IP в ADMIN_HOST, иначе правила закроют ей доступ. Полная матрица находится в документе для ИБ.

Проверка

- На каждой машине docker compose ps показывает только её сервисы, контейнеры в состоянии running.
- В админке вход проходит на всех контурах, а раздел «Миграция / Подключения» при проверке не показывает ошибок.
- В Grafana видны нагрузка и логи всех машин, а после построения отчёта появляется его статистика.

Бэкап делается на каждой машине отдельно. Остановите сервисы командой docker compose down, скопируйте DATA_ROOT и .env, на Linux через sudo, и запустите их снова. Скрипт ночного бэкапа не входит в комплект и рассчитан на одну машину.

5. Установка без интернета

Комплект для работы без интернета готовит поставщик: в нём лежат все образы, в том числе GitLab и заданий CI. Скриптов сборки в комплекте нет, поэтому собрать такой комплект на месте нельзя. Проверьте вторую строку файла IMAGES.txt. В комплекте без интернета там написано:

> Комплект для установки без интернета: все образы, в том числе GitLab и заданий CI, лежат в images\.

Если там написано «с интернетом» или второй строки нет (комплект собран старой версией сборки), в конце файла есть раздел «Из интернета»: эти образы docker скачал бы при установке. Запросите у поставщика комплект без интернета или перенесите эти образы сами. На машине с интернетом, где включено хранилище containerd (см. ниже), для каждой строки раздела выполните:

> docker pull <образ из IMAGES.txt>
>
> docker save -o <имя>.tar.gz <образ из IMAGES.txt>

Файлы положите в папку images комплекта: установка загружает из неё все файлы *.tar.gz, сжатие не обязательно. Образы заданий CI переносятся так же, см. документ «GitLab без интернета», шаг 1.

На целевой машине нужно хранилище образов containerd, иначе docker load теряет digest и compose не находит образы. Включите его до установки:

- **Docker Desktop:** Settings / General / Use containerd for pulling and storing images.
- **Linux:** в /etc/docker/daemon.json добавьте {"features": {"containerd-snapshotter": true}} и перезапустите Docker.
- **Проверка:** команда docker info должна показывать driver-type: io.containerd.snapshotter.v1.

Дальше установка идёт как обычно. Как подготовить GitLab, пакеты NuGet и рабочие места разработчиков, описано в документе «GitLab без интернета».

6. Настройка после установки

Этот раздел нужен при любом варианте установки. Без него админка не видит серверы и GitLab, а статистика не доходит до Grafana. На нескольких машинах сначала закончите раздел 4. При чистой установке сначала создайте базы Pilot и группы GitLab, см. раздел 3.

Общие настройки админки

1. **Войдите в админку:** http://<машина admin>:5300. Логин и пароль те же, что у администратора Pilot (вход возможен только учётной записью администратора сервера Pilot), затем нужен вход в GitLab. Если OAuth не настроен, вставьте Personal Access Token с правом api. Адрес Pilot на странице входа выбрать нельзя: вход проверяется на серверах из настроек (поле для ручного ввода остаётся только при первом запуске, пока серверы не заданы). Для входа достаточно, чтобы отвечал хотя бы один настроенный сервер. После 10 неудачных входов с одного адреса за 5 минут вход блокируется на 5 минут.
2. **Откройте вкладку «Настройки»** и заполните поля из таблицы ниже.
3. **Нажмите «Проверить подключения»,** исправьте ошибки и нажмите «Сохранить». Стейдж эта кнопка не проверяет: его проверьте на вкладке «Модули».

Поле	Что указать	Пример
Серверы Pilot: Тест, Прод, Стейдж	Адрес Pilot-Server **с именем базы**: форма настроек не сохранит адрес без имени базы. Если контур на этой же машине, можно указать localhost	http://10.0.0.2:5547/pilot-ice_ru, прод на порту 5546, стейдж на порту 5549
Ветка GitLab	Какая ветка соответствует серверу. Контур без ветки доступен только для просмотра	Тест dev, прод main
Задача в коммите	Обязателен ли номер задачи в коммите для этого контура	Включено на проде
Группа с автоматизациями	Адрес группы GitLab. Из него админка берёт и сервер GitLab для входа. Указывайте адрес, доступный из браузеров пользователей, а не localhost	http://10.0.0.3:8080/automation
Группа с отчётами	Группа с репозиториями отчётов. Пусто отключает вкладку «Отчёты»	http://10.0.0.3:8080/reports

Поле	Что указать	Пример
Группа с клиентскими модулями	Пусто отключает вкладку «Модули»	http://10.0.0.3:8080/client-modules
Группа с серверными модулями	Нужна только для таблицы «Активные работы» на вкладке «Серверные модули»	http://10.0.0.3:8080/server-extensions
Репозиторий с типами	Один проект GitLab, а не группа. Пусто оставляет на вкладке «Типы» только сравнение серверов	http://10.0.0.3:8080/pilot-ice/types
Job сборки модуля в CI	Имя задания CI, артефакт которого админка выкладывает на серверы. Допустимо до 64 знаков: латинская буква или _, затем буквы, цифры, _ и -	build_module
Номер задачи и ссылка на задачу	Регулярное выражение для номера и шаблон ссылки на трекер. Ссылка необязательна, допускаются только схемы http и https	KTN-\d+
Code Review	Вливать в dev только после одобрения не автором. Если разработчик один, выключите	
Код лицензии для отчётов	Код лицензии Pilot, под которым админка входит в базу для работы с отчётами и модулями	100

Откуда берутся начальные значения. При установке с данными они восстанавливаются из бэкапа, ключ -GitLabHost заменяет в них старый IP, а адреса контуров на других машинах скрипт подставляет сам. При чистой установке при первом запуске берутся PILOT_*_URL и *_GITLAB_*_URL из .env, имя базы в адресах Pilot нужно дописать. Всё сохраняется в services/proadmin/settings.json в папке данных. После первого запуска значения из .env больше не действуют: меняйте адреса только на этой странице.

Кто может менять настройки. Сохранять общие настройки, пользоваться разделом «Миграция» и сохранять «Серверы» может только сессия, вошедшая администратором Pilot контура prod (если прод в настройках не задан, администратором любого настроенного сервера, до первой настройки любым). Остальные видят эти страницы только для чтения с замаскированными секретами, запись даёт ошибку 403. Изменения в GitLab и выкладка на серверы (в том числе все операции серверных модулей) запрещены, если GitLab не подключён, не задана группа или не удалось определить роль: нужна роль не ниже Developer в группе.

Ключи и учётки для выкладки из CI. На этой странице их нет. Выкладка из CI (/api/ci/modules/deploy, /api/ci/server-modules/deploy) требует два доказательства: заголовок X-Job-Token с токеном задания $CI_JOB_TOKEN, который админка проверяет в GitLab (GET /api/v4/job), и заголовок X-Deploy-Key с ключом контура, выведенного из проверенной ветки (dev это test, release/* stage, main и master prod). Ветка, коммит, pipeline, проект и автор берутся из ответа GitLab: несовпавшие поля запроса дают 400, проект вне группы модулей этого вида 403. Ключи разные для каждого контура и задаются в .env установочной папки машины admin. Модули на серверы Pilot админка выкладывает под служебной учёткой Pilot контура, она тоже задаётся там. Если ключ контура пуст, выкладка из CI на этот контур выключена, если пуст логин, она не выполняется. При установке с данными значения учёток переносятся из бэкапа, при чистой установке заполните их сами:

> MODULES_DEPLOY_KEY_TEST=<ключ test, пустой создаст установка>

> MODULES_DEPLOY_KEY_STAGE=<ключ stage, пустой создаст установка>
>
> MODULES_DEPLOY_KEY_PROD=<ключ prod, пустой создаст установка>
>
> MODULES_DEPLOY_TEST_LOGIN=<пользователь Pilot>
>
> MODULES_DEPLOY_TEST_PASSWORD=<пароль>
>
> MODULES_DEPLOY_STAGE_LOGIN=<пользователь Pilot>
>
> MODULES_DEPLOY_STAGE_PASSWORD=<пароль>
>
> MODULES_DEPLOY_PROD_LOGIN=<пользователь Pilot>
>
> MODULES_DEPLOY_PROD_PASSWORD=<пароль>

Нужен пользователь базы Pilot с правами на папку расширений. Для серверных модулей это должен быть администратор базы: ServerHost принимает команды только от администраторов. Лучше завести отдельного пользователя, а не использовать admin. После правки .env выполните docker compose up -d proadmin. Ручная выкладка из админки этой учёткой не пользуется: она идёт под учёткой, введённой на вкладке «Модули», а если её не вводили, под учёткой, с которой выполнен вход.

Ключи в GitLab. На машине с ролью gitlab значения ключей вносит в переменные групп модулей шаг GitLab после установки (он же удаляет старую DEPLOY_KEY). Вручную их вносят, если шаг пропущен или не удался либо GitLab стоит на другой машине: DEPLOY_KEY_TEST (Masked), DEPLOY_KEY_STAGE и DEPLOY_KEY_PROD (Masked и Protected); ветки main и release/* должны быть защищены, иначе Protected-ключи защиты не дают. Старый общий MODULES_DEPLOY_KEY принимается только для test и только если ключ test не задан. При установке с данными он переносится в MODULES_DEPLOY_KEY_TEST и очищается, для stage и prod создаются новые ключи. Напоминание внести ключи в GitLab вручную печатается на машине admin без GitLab при каждом запуске, а на машине с GitLab только если шаг пропущен или не удался. Для stage и prod старый ключ не принимается никогда: ответ 503 называет переменную, которую нужно задать. Шаблон CI без X-Job-Token получает 401: обновите .gitlab-ci.yml модулей из ci-template. Модули, у которых в .gitlab-ci.yml остались переменные ключа с прежним префиксом (вида <префикс>_DEPLOY_KEY_TEST; MODULES_DEPLOY_KEY_* не в счёт) или нет X-Job-Token, шаг GitLab после установки перечисляет в предупреждении: в админке на странице «Модули» нажмите «Обновить CI у всех модулей» и влейте MR в dev. Проверка библиотек (check_libraries) использует только токен задания. После 10 неверных ключей или токенов с одного адреса за 5 минут админка отвечает 429 на 5 минут; адрес клиента виден, если админка не стоит за обратным прокси или включено ADMIN_TRUST_PROXY=true.

Статистика и мониторинг

Вся статистика попадает в базу pilot_metrics на проде, Grafana читает её оттуда. Адрес сборщика настраивается отдельно для каждого источника. В примерах сборщик находится на машине 10.0.0.3. Если сервер другой, замените адрес.

Источник	Где задан адрес	Что сделать
Клиентские модули и отчёты	Общие настройки Pilot, раздел «Модули», хранятся в базе контура	Задать на каждом контуре, пересборка не нужна

Источник	Где задан адрес	Что сделать
Серверные модули	Переменная METRICS_HOST в .env	На одной машине ничего, на нескольких заполнить METRICS_HOST
Автоматизации	Адрес не нужен: сборщик читает лог Pilot-Server и пишет прямо в базу	Для test и stage логирование включено по умолчанию (для prod выключено); проверьте его на вкладке «Серверы»
Логи всех машин	Адрес не нужен: Promtail отправляет их через мост к Loki	Заполнить METRICS_HOST
Нагрузка других машин	monitoring/prometheus/prometheus.yml на машине metrics	Добавить адреса, см. раздел 4

> При установке с данными модули в базах Pilot собраны с адресами исходной машины. Если адрес сборщика изменился, задайте его в общих настройках на каждом контуре, иначе события будут уходить на старый адрес и теряться.

Клиентские модули и отчёты. Адрес задаётся в одном месте. Клиентские модули отправляют статистику через Pilot.ClientSdk. Reports в текущем коде берёт адрес сервера из этого же раздела, если он заполнен, иначе из Reports.config.json, и сам отправляет отчёты в /api/report-events; в журнале изменений эта возможность значится как «Не выпущено», номер версии ей не назначен, а выпущенная версия 1.4.0 читает адрес только из файла. На каждом контуре выполните:

1. Войдите в клиент Pilot администратором этого контура.
2. Откройте «Сервис / Настройки / Общие настройки», раздел «Модули общие настройки».
3. В поле «Адрес сервиса статистики» укажите http://10.0.0.3:5310/api/module-events.
4. Нажмите «Проверить»: кнопка обращается к сборщику с этого компьютера. Сохраните настройки.

Клиенты получают новый адрес без перезапуска. Действующий адрес показывает пункт «Модули версии» в меню трея.

Файл настроек Reports. Адрес GitLab модуль берёт только из файла Reports.config.json в корне своего репозитория. Поставляемый файл содержит пустые statsUrl, reportsGitLabGroupUrl и gitLabOAuthClientId: администратор вписывает свои значения до сборки или в файл рядом с dll. Выпущенные версии (до 1.4.0 включительно) берут из файла и адрес статистики:

> "statsUrl": "http://10.0.0.3:5310/api/report-events",
>
> "reportsGitLabGroupUrl": "http://10.0.0.3:8080/reports",

Если адрес GitLab изменился, измените файл как обычное изменение модуля: в ветке задачи увеличьте Version в Reports.csproj, влейте в dev и выпустите на стейдж и прод на вкладке «Модули». Если GitLab установлен заново, создайте для модуля новое OAuth-приложение: не confidential, scope api, Redirect URI http://127.0.0.1:7890/callback. Его Application ID запишите в gitLabOAuthClientId.

Серверные модули. Pilot-Server получает адрес из .env при создании контейнера: http://<METRICS_HOST>:5310/api/module-events. Если METRICS_HOST пуст, используется host.docker.internal:5310, то есть сборщик на этой же машине. После изменения .env выполните

docker compose up -d. Отдельному контуру адрес можно задать файлом stats.json, но он действует, только если METRICS_HOST пуст:

> <DATA_ROOT>/<контур>/databases/.server-modules/stats.json
>
> {"enabled": true, "statsUrl": "http://10.0.0.3:5310/api/module-events", "apiKey": ""}

После создания или изменения файла перезапустите контейнер: docker restart pilot-server-test.

Ключ сборщика. Если задан COLLECTOR_API_KEY, события без него отклоняются, а сводки GET /api/report-events/summary и /api/module-events/summary требуют заголовок X-Api-Key; /health открыт. При первой установке на одну машину ключ создаётся сам, в остальных случаях пустое значение остаётся, установка и сам сборщик при старте предупреждают, что приём идёт без ключа. Серверные модули получают ключ из .env сами. Для клиентских модулей и отчётов укажите его в поле «API-ключ» общих настроек.

Контур в статистике. Сборщик определяет контур по адресу сервера Pilot: порт 5549 это stage, 5547 это test, остальное prod. Если порты или адреса другие, поправьте ENV_MAP в .env на машине metrics и выполните docker compose up -d. Формат: подстрока=контур;...;*=контур по умолчанию.

Проверка статистики

1. С рабочего места Pilot откройте http://10.0.0.3:5310/health. Если адрес не открывается, порт 5310 закрыт в сети.
2. Постройте любой отчёт и запустите клиентский модуль.
3. В Grafana на машине metrics, порт 3000, откройте папку «CI/CD Pilot» и дашборды отчётов и модулей. Новые записи должны появиться в течение минуты с правильным контуром.

Если данных нет, смотрите на рабочем месте логи Pilot.ClientSdk.log и Reports.log в %AppData%\ASCON\Units Logs\Logs. Адрес серверных модулей виден в журнале контейнера: docker logs pilot-server-test | grep "Статистика серверных", при условии что ServerHost установлен на этом контуре.

7. Справочник

Ключи установки

Windows	Linux	Что делает
-Roles <роли>	--roles <роли>	Ставит на машину только эти сервисы. По умолчанию все, кроме GitLab: он добавляется ключом -WithGitLab или автоматически при установке с данными, если в них есть копия GitLab
-GitLabHost <ip>	--gitlab-host <ip>	Заменяет IP исходной машины при установке с данными
-FromBackup <каталог>	--from-backup <каталог>	Берёт данные из указанного бэкапа

Windows	Linux	Что делает
-NoData	--no-data	Чистая установка, папка data не используется
-WithGitLab	--with-gitlab	Добавляет GitLab CE и runner
-WithMyAdmin	--with-myadmin	Добавляет Pilot-myAdmin Web для test и stage
-SkipLoad	--skip-load	Не загружает images, если образы уже загружены

Параметры .env

Параметр	Что это
DATA_ROOT	Папка данных
ROLES, COMPOSE_PROFILES	Роли машины и включённые части стека. Скрипт записывает их сам
PROD_HOST, TEST_HOST, STAGE_HOST, METRICS_HOST	Адреса машин с контурами и мониторингом, одинаковые на всех машинах: мост к сервису строится только там, где такой роли нет. Если роли контура на машине нет, а адрес пуст, установка предупреждает: сервисы до него не достучатся. По METRICS_HOST серверные модули отправляют статистику
COLLECTOR_API_KEY, ENV_MAP	Ключ сборщика статистики и правила определения контура по адресу сервера Pilot
INTERNAL_BIND	IP для служебных портов. Пусто означает только эту машину (127.0.0.1)
ADMIN_HOST	IP машины с админкой, если она не на этой машине. Нужен только правилам брандмауэра, которые установка пишет при INTERNAL_BIND
TLS_CERT_FILE, TLS_KEY_FILE	Необязательно: пути к сертификату и ключу HTTPS, по умолчанию certs/tls.crt и certs/tls.key в установочной папке. Если оба файла есть, админка и Grafana работают по HTTPS
PROD_ROOT, TEST_ROOT, STAGE_ROOT	Сетевые папки контуров на машине admin
SERVER_SETTINGS_ROOT	Пустое значение отключает раздел «Серверы»
GITLAB_HOST_IP	IP машины с GitLab. С localhost GitLab недоступен с других машин
GITLAB_GROUP_URL, REPORTS_GITLAB_GROUP_URL, MODULES_GITLAB_GROUP_URL, SERVER_MODULES_GITLAB_GROUP_URL, TYPES_GITLAB_PROJECT_URL	Группы и проект в GitLab для первого запуска админки: автоматизации, отчёты, модули и типы. Позже они меняются на вкладке «Настройки»
GITLAB_OAUTH_CLIENT_ID, _SECRET, _REDIRECT_URI	OAuth-приложение для входа в админку. Без него вход работает по Personal Access Token. Если _REDIRECT_URI пуст, адрес возврата строится из ADMIN_PUBLIC_URL (…/GitLab/Callback), а если пуст и он, из адреса запроса

Параметр	Что это
ADMIN_PUBLIC_URL, ADMIN_TRUST_PROXY	Адрес админки для браузеров (из него строится адрес возврата OAuth; https-адрес включает Secure-cookie) и признак доверия заголовкам X-Forwarded-* от обратного прокси перед админкой (по умолчанию false)
COMPOSE_PROJECT_NAME	Имя проекта compose, по умолчанию cicd; от него зависят имена томов, например cicd_grafana_data
MODULES_DEPLOY_KEY_TEST, _STAGE, _PROD, MODULES_DEPLOY_<КОНТУР>_LOGIN, _PASSWORD	Ключи контуров и служебные учётки Pilot для выкладки модулей из CI, см. раздел 6. Пустой ключ контура отключает выкладку на него. Старый общий MODULES_DEPLOY_KEY принимается только для test
PILOT_TEST_URL, PILOT_PROD_URL, PILOT_STAGE_URL	Адреса Pilot-Server для первого запуска админки. На машине admin скрипт заполняет их по адресам машин. Позже они меняются на вкладке «Настройки»
PILOT_SERVER_IMAGE	Другой образ Pilot-Server
*_PORT	Порты сервисов: админка, миграция, сборщик, Grafana, Prometheus, GitLab

Где хранятся пароли

На установленном сервере все пароли лежат в .env установочной папки, доступ к файлу ограничен (на Linux права 600). При установке с данными у контуров свои пароли с префиксами PROD_, TEST_ и STAGE_, значение с префиксом важнее общего.

Что	Логин	Пароль
Pilot-Server	PILOT_ADMIN_LOGIN, PROD_PILOT_ADMIN	PILOT_ADMIN_PASSWORD, PROD_PILOT_PASSWORD
Админка	логин Pilot, затем вход в GitLab	пароль Pilot
Grafana	GRAFANA_ADMIN_USER	GRAFANA_ADMIN_PASSWORD
Консоль RustFS	MINIO_USER, PROD_MINIO_USER	MINIO_PASSWORD, PROD_MINIO_PASSWORD
PostgreSQL, суперпользователь	postgres	PG_SUPERUSER_PASSWORD, PROD_PG_SUPERUSER_PASSWORD
PostgreSQL, Pilot и метрики	pilotuser	PILOT_PG_PASSWORD, PROD_PILOT_PG_PASSWORD
GitLab	root	GITLAB_ROOT_PASSWORD, только начальный
Выкладка модулей, служебная учётка Pilot	MODULES_DEPLOY_<КОНТУР>_LOGIN	MODULES_DEPLOY_<КОНТУР>_PASSWORD
Ключ миграции	-	MIGRATION_API_KEY

Что	Логин	Пароль
Ключи выкладки из CI	-	MODULES_DEPLOY_KEY_TEST, MODULES_DEPLOY_KEY_STAGE, MODULES_DEPLOY_KEY_PROD
Ключ сборщика статистики	-	COLLECTOR_API_KEY

> **Смена паролей.** Пароли PostgreSQL и RustFS записываются в базы при первом запуске (ключи RustFS в .env по-старому называются MINIO_USER и MINIO_PASSWORD). Если поменять их только в .env, сервисы перестанут подключаться. Пароли Pilot, Grafana и GitLab меняются в их интерфейсах, после этого поправьте .env, чтобы он не вводил в заблуждение.

Пароли в данных. Пароли администраторов Pilot и секретные ключи S3 для миграции лежат в DATA_ROOT/services/migration/config.json открытым текстом. На Linux файл создаётся с правами 600; на томе Windows права не действуют, закройте каталог data средствами хоста. Страница «Подключения» показывает пароль и ключ маской: они сохраняются как есть, пока не менялись адрес Pilot и логин (для пароля) либо адрес S3 и адрес для сервиса (для ключа); при смене сервис их сбрасывает, введите заново.

Где что лежит

> <DATA_ROOT>
>
> .installed метка: данные повторно не восстанавливаются
>
> .restore-started метка: начато восстановление GitLab; после обрыва очистите DATA_ROOT
>
> prod\ контур prod; test\ и stage\ устроены так же
>
> license\Pilot Server\License\ файл лицензии Pilot
>
> pilot-server\settings\settings.xml настройки Pilot-Server
>
> databases\ файлы баз Pilot и серверные модули
>
> postgres\ данные PostgreSQL
>
> server-settings\server-settings.conf настройки сервера из админки
>
> logs\ логи Pilot-Server
>
> services\proadmin\, services\migration\ данные админки и миграции
>
> import\, export\ присланные базы и выгрузки
>
> gitlab\ GitLab; журнал восстановления gitlab\logs\restore.log

Вне DATA_ROOT, на томах Docker проекта compose: файлы Pilot (S3) в RustFS (<проект>_rustfs_<контур>, например cicd_rustfs_prod), данные Grafana, Loki и Prometheus и артефакты и загрузки GitLab (<проект>_gitlab_artifacts, <проект>_gitlab_uploads). Копия DATA_ROOT их не содержит: артефакты и загрузки входят в gitlab-backup, том RustFS снимает архив тома (backup.ps1, часть rustfs), остальное копируйте отдельно. На новой машине том RustFS разворачивает установка с -FromBackup.

Обновление

Пример: стоит C:\PilotProAdmin\Install, новая версия комплекта пришла в D:\Distr\Install. Перенесите .env и запустите установку из новой папки:

```
copy C:\PilotProAdmin\Install\.env D:\Distr\Install\.env

cd D:\Distr\Install

powershell -ExecutionPolicy Bypass -File .\install.ps1
```

На Linux:

```
sudo cp /opt/cicd/Install/.env /opt/cicd/Install-new/.env

cd /opt/cicd/Install-new

sudo bash install.sh
```

Данные в DATA_ROOT не меняются, даже если в новой папке есть data. Контейнеры Pilot-Server и PostgreSQL при переезде в другую папку пересоздаются, даже если образы не менялись (в compose относительные пути монтирования ./init, ./monitoring, ./wait-for-postgres.sh): планируйте короткий простой. Остальные пересоздаются, только если изменились образы или настройки. Рабочей становится новая папка, старую можно удалить. На нескольких машинах обновляйте сначала контуры test и stage, затем prod.

Файловый архив Pilot (S3). Его хранит RustFS (образ rustfs/rustfs:1.0.5, лицензия Apache-2.0). Контейнер называется rustfs-<контур>, адрес S3 в Pilot Admin: http://rustfs-<контур>:9000, бакет pilot-ice-ru. Имена MINIO_USER и MINIO_PASSWORD в .env исторические: значение - ключ и секрет доступа RustFS.

Где лежат данные. Файлы RustFS хранятся в именованном томе Docker <проект>_rustfs_<контур> (проект из COMPOSE_PROJECT_NAME, по умолчанию cicd), а не в DATA_ROOT. Причина: на Windows папка хоста, подключённая к контейнеру, ломает составную загрузку больших файлов (RustFS отвечает NoSuchKey), на томе всё работает; том к тому же быстрее и не требует настройки прав. Список томов: docker volume ls. Том не входит в копию DATA_ROOT: снимайте его скриптом backup.ps1 (архив тома) или командой docker run --rm -v <том>:/v:ro -v <каталог>:/b alpine tar czf /b/rustfs.tgz -C /v .. Не удаляйте том и не запускайте docker compose down -v без копии.

Права на данные RustFS. Настраивать их не нужно ни на Linux, ни на Windows: том создаётся с владельцем 10001 (пользователь RustFS), установка не выполняет chown и не создаёт папку для этих данных.

Ограничения. RustFS 1.0.x вышел недавно: поведение с Pilot-Server под вашей нагрузкой проверьте на стенде test. Совместимость с формой работы Pilot-Server и поддержку вендором Pilot хранилищ, отличных от MinIO, мы не проверяли: уточните у вендора. Запуск RustFS, создание бакета, учёток и политик, ротация паролей и составная загрузка на томе проверены на живом Docker Desktop (Windows). Распаковка тома из бэкапа при установке проверена на заглушках docker, а сохранение владельца 10001 при распаковке от root на образе RustFS. Не проверены на живом Docker: запуск RustFS на восстановленных данных, работа Pilot-Server поверх RustFS; ветка восстановления RustFS в install.sh на Linux-хосте не запускалась.

Полезные команды

Выполняйте в установочной папке. На Linux, если пользователь не входит в группу docker, добавляйте sudo:

> docker compose ps # что запущено
>
> docker compose logs -f pilot-server-prod # логи сервиса
>
> docker compose restart proadmin # перезапуск сервиса
>
> docker compose down # остановить всё, данные остаются
>
> docker compose up -d # запустить снова

Посмотреть пароли и ключи на Linux можно командой sudo grep -E 'PASSWORD|KEY|SECRET' /opt/cicd/Install/.env, на Windows командой Select-String 'PASSWORD|KEY|SECRET' C:\PilotProAdmin\Install\.env.