ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «ДИДЖИТРЕК» (ООО «ДТ»)
Адрес: 115114, Москва г., Муниципальный Округ Замоскворечье вн. тер. г., Шлюзовая набережная, д. 8, стр. 1
ИНН: 9705200012, КПП: 770501001, ОГРН: 1237700339788
Тел.: +7 985 770-45-12, Email: info@digi-track.ru

Инструкция по установке программного обеспечения «DigiTrack Confidential Computing»

г. Москва – 2026

Оглавление

1. Введение

Настоящий документ предназначен для администраторов и инженеров, выполняющих развертывание программного обеспечения (ПО) DigiTrack Confidential Computing, реализующего процессы вертикального федеративного обучения (Vertical Federated Learning, VFL). Система позволяет организовать совместное обучение моделей машинного обучения на данных, распределенных между несколькими участниками, без необходимости раскрытия исходных данных друг другу.

Документ содержит пошаговые инструкции по установке и настройке всех компонентов системы:

- Брокера сообщений RabbitMQ для обеспечения коммуникации между узлами;
- Серверов обучения (активного и пассивного) для реализации SecureBoost;
- PSI-сервера (Private Set Intersection) для безопасного определения пересечений наборов данных.

Развертывание производится с использованием Docker-контейнеров, что обеспечивает изолированность, воспроизводимость и простоту управления компонентами.

2. Глоссарий

Термин	Определение
VFL	Vertical Federated Learning — вертикальное федеративное обучение, подход к обучению моделей на данных с одинаковыми объектами, но разными признаками, распределенными между участниками
Активный узел (Active Party)	Участник федеративного обучения, который инициирует процесс обучения и содержит целевую переменную (метки)
Пассивный узел (Passive Party)	Участник федеративного обучения, который предоставляет признаки для объектов, но не имеет доступа к целевой переменной
PSI	Private Set Intersection — криптографический протокол для нахождения пересечения множеств без раскрытия элементов, не вошедших в пересечение
SecureBoost	Алгоритм градиентного бустинга, адаптированный для федеративного обучения
RabbitMQ	Брокер сообщений, используемый для асинхронной коммуникации между компонентами системы
Docker	Платформа для контейнеризации приложений, обеспечивающая изоляцию и переносимость
vflnet	Виртуальная сеть Docker, создаваемая для взаимодействия контейнеров системы
Хэширование	Процесс преобразования идентификаторов в хэш-значения фиксированной длины (в системе используется SHA-512)

3. Описание системы

Архитектура VFL включает следующие основные компоненты:

3.1. Брокер сообщений (RabbitMQ)

Обеспечивает надежную асинхронную коммуникацию между всеми компонентами системы. RabbitMQ управляет очередями сообщений, гарантирует доставку и маршрутизацию между серверами обучения и PSI-серверами.

3.2. Серверы обучения (Training Servers)

Реализуют алгоритм SecureBoost для вертикального федеративного обучения:

- **Активный сервер (training_active):** Содержит целевую переменную, инициирует обучение, агрегирует градиенты
- **Пассивный сервер (training_passive):** Предоставляет признаки, вычисляет локальные градиенты по запросу активного сервера

3.3. PSI-сервер (Private Set Intersection)

Обеспечивает конфиденциальное нахождение пересечения идентификаторов между участниками без раскрытия полных наборов данных. Использует протокол на основе хэширования SHA-512 и работу с чанками данных.

3.4. Сетевое взаимодействие

Компоненты взаимодействуют через:

- **Сеть vflnet:** внутренняя сеть Docker для безопасного взаимодействия контейнеров
- **Порты:** проброшенные порты для внешнего доступа к API сервисов

4. Требования к среде развертывания

Перед началом установки необходимо убедиться, что серверная инфраструктура соответствует требованиям для развертывания программного обеспечения **«DigiTrack Confidential Computing»**.

4.1. Аппаратные требования

Для развертывания одного экземпляра Системы (на физическом сервере или виртуальной машине) должны быть обеспечены следующие минимальные аппаратные ресурсы:

- **Процессор:** не менее **16 вычислительных ядер (CPU / vCPU)**;
- **Оперативная память:** не менее **32 ГБ RAM**;
- **Дисковое пространство:** не менее **500 ГБ** свободного пространства;
- **Пропускная способность сетевого канала:** не менее **10 Мбит/с**.

Указанные ресурсы должны обеспечивать:

- запуск контейнерных компонентов Системы;
- хранение локальных наборов данных;
- выполнение сценариев безопасной сверки данных (**PSI**);
- выполнение сценариев вертикального федеративного обучения (**VFL**);
- выполнение сценариев исполнения модели (**Inference**);
- хранение журналов, промежуточных результатов и служебных артефактов.

4.2. Требования к операционной системе

Экземпляр Системы должен быть развёрнут на сервере под управлением операционной системы семейства **Linux**, совместимой с контейнерной средой исполнения Docker.

Рекомендуемая операционная система:

- **Ubuntu 24**.

Операционная система должна обеспечивать:

- поддержку запуска Docker-контейнеров;
- работу сетевых сервисов и контейнерного взаимодействия;
- доступ к файловой системе для размещения локальных данных и служебных файлов;
- возможность выполнения административных операций, необходимых для установки и сопровождения Системы.

Установка программного обеспечения должна выполняться пользователем с правами администратора (**root**) либо с эквивалентными привилегиями.

Операционная система должна обеспечивать:

- поддержку запуска Docker-контейнеров;
- работу сетевых сервисов и контейнерного взаимодействия;
- доступ к файловой системе для размещения локальных данных и служебных файлов;
- возможность выполнения административных операций, необходимых для установки и сопровождения Системы.

Установка программного обеспечения должна выполняться пользователем с правами администратора (**root**) либо с эквивалентными привилегиями.

4.3. Программные требования

На сервере, предназначенном для установки экземпляра Системы, должны быть установлены следующие программные компоненты:

- **Docker Engine** версии **20.10** или выше;
- **Docker Compose** — при использовании сценария запуска нескольких контейнеров;
- **Git** — при необходимости получения конфигурационных или вспомогательных файлов;
- системные утилиты администрирования и диагностики:
  - ss
  - grep
  - curl

Установка и настройка указанных программных компонентов осуществляется средствами операционной системы и/или в соответствии с официальной документацией соответствующего программного обеспечения.

4.4. Сетевые требования

Для корректной работы Системы должна быть обеспечена сетевая связность между экземплярами, развёрнутыми у участников взаимодействия.

На сетевом уровне должны быть обеспечены:

- возможность обмена сообщениями между экземплярами Системы;
- возможность установления защищённых соединений между участниками;
- доступность прикладных сервисов, используемых компонентами Системы;

- пропускная способность канала связи между взаимодействующими узлами не менее **10 Мбит/с**.

При типовой конфигурации развертывания используются следующие сетевые порты:

- **5672**, **15672** — для компонентов очередей сообщений (**RabbitMQ**);
- **50000**, **50001** — для сценариев обучения (**Training**);
- **50050**, **50051** — для сценариев исполнения модели (**Prediction / Inference**);
- **5051** — для сценария безопасной сверки данных (**PSI**);
- **22/TCP** — для административного доступа по **SSH**.

На межсетевых экранах (firewall**)** должны быть разрешены входящие и исходящие соединения для портов, используемых в согласованной схеме развертывания.

Конкретный перечень открываемых портов может уточняться в зависимости от архитектуры размещения и конфигурации экземпляра Системы.

4.5. Требования к поставочным материалам

Для выполнения установки должны быть доступны поставочные материалы программного обеспечения, включая:

- контейнерные образы компонентов Системы;
- конфигурационные файлы;
- установочные и эксплуатационные документы;
- при необходимости — архивные файлы и дополнительные артефакты развертывания.

Поставочные материалы должны предоставляться одним из следующих способов:

- в составе дистрибутива программного обеспечения;
- в виде архивных файлов, передаваемых заказчику;
- по защищённой ссылке для скачивания из доверенного источника поставки.

Использование внутренних инфраструктурных ресурсов разработчика или поставщика, не входящих в поставочный контур заказчика, **не является обязательным условием установки и эксплуатации Системы**.

5. Подготовка образов

Порядок выполнения команд

Все команды, приведённые в настоящем разделе, должны выполняться **непосредственно на сервере**, на котором планируется развертывание соответствующего экземпляра программного обеспечения **«DigiTrack Confidential Computing»**.

Требования к среде выполнения команд:

- сервер должен работать под управлением операционной системы **Ubuntu 24**;
- на сервере должна быть **предварительно установлена и настроена контейнерная среда Docker**;
- пользователь, выполняющий команды установки, должен обладать **правами администратора** (root) либо эквивалентными привилегиями, достаточными для запуска контейнеров и выполнения операций с образами;
- команды должны выполняться **в командной оболочке Linux** непосредственно на сервере установки.

Если развертывание выполняется на нескольких серверах (например, отдельно для **координатора** и **партнёра данных**), действия по загрузке, настройке и запуску контейнерных образов выполняются **на каждом соответствующем сервере отдельно** в соответствии с назначенной ролью.

5.1. Загрузка образов PSI-сервера

Предполагается, что образы psi-server сохранены в файлы:
bash
Сохранение образа (выполняется на машине с доступом к образу)
docker save -o psi-server.tar psi-server
Скопируйте эти файлы в директорию запуска (например, ~/psi-deploy/):
bash
Копирование на целевую машину
scp psi-server.tar user@target-server:~/psi-deploy/
Загрузка образов из директории с *.tar:
bash
cd ~/psi-deploy/
docker load -i psi-server.tar
Проверьте наличие образов:
bash
docker images | grep psi-server

5.2. Загрузка образа VFL-сервера

Если у вас есть архив образа vfl_server-1.0.0.tgz, выполните:
bash
gunzip -c vfl_server-1.0.0.tgz | docker load

Пояснение: gunzip -c распаковывает содержимое на stdout и передаёт поток в docker load для загрузки образа в локальный Docker.
Проверьте, что образ появился:
bash
docker images | grep vfl_server

5.3. Альтернативный способ: загрузка из реестра

При наличии доступа к реестру образов:
bash
Логин в реестр (если требуется)
docker login nexus.ucb.infra

Загрузка образов
docker pull nexus.ucb.infra/conf-compute/vfl:latest
docker pull psi-server:latest

6. Настройка сети Docker (vflnet)

Для изолированного и безопасного взаимодействия контейнеров необходимо создать выделенную сеть Docker.

6.1. Создание сети

bash
Создание сети vflnet
docker network create vflnet

Проверка создания сети
docker network ls | grep vflnet

6.2. Альтернативные варианты настройки сети

Вариант А: Использование сети с драйвером bridge (по умолчанию)
bash
docker network create --driver bridge vflnet
Вариант Б: Сеть с заданной подсетью
bash
docker network create --driver bridge --subnet=172.20.0.0/16 --ip-range=172.20.0.0/24 vflnet
Вариант В: Использование сети host (для отладки, не рекомендуется для продуктивной среды)
bash
Контейнеры будут использовать сеть хоста
В этом случае создание отдельной сети не требуется
Вариант Г: Удаление и пересоздание сети (при необходимости)
bash
Остановка контейнеров перед удалением сети
docker stop $(docker ps -q --filter network=vflnet) **2**>/dev/null

Удаление сети
docker network rm vflnet

Создание заново
docker network create vflnet

6.3. Создание сети для PSI (опционально)

При необходимости изолировать PSI-сервер в отдельной сети:
bash
docker network create psi-network
Примечание: Для упрощения управления рекомендуется использовать единую сеть vflnet для всех компонентов.

7. Установка и настройка RabbitMQ

7.1. Запуск RabbitMQ контейнера

bash
```
docker run -d \
  --name rabbitmq \
  --hostname rabbitmq \
  --network vflnet \
  -p 5672:5672 \
  -p 15672:15672 \
  rabbitmq:3-management
```
Параметры запуска:

- -d — запуск в фоновом режиме
- --name rabbitmq — имя контейнера
- --hostname rabbitmq — hostname внутри контейнера
- --network vflnet — подключение к сети vflnet
- -p 5672:5672 — проброс порта для AMQP протокола
- -p 15672:15672 — проброс порта для веб-интерфейса управления

Дайте контейнеру время на запуск:

bash
```
sleep 10
```
7.2. Создание пользователя и настройка прав

bash
```
# Создание пользователя
docker exec rabbitmq rabbitmqctl add_user rabbitmqadmin 12345
docker exec rabbitmq rabbitmqctl set_user_tags rabbitmqadmin administrator

sleep 10

# Создание виртуальных хостов
docker exec rabbitmq rabbitmqctl add_vhost arbiter
docker exec rabbitmq rabbitmqctl add_vhost guest
docker exec rabbitmq rabbitmqctl add_vhost host

# Настройка прав доступа
docker exec rabbitmq rabbitmqctl set_permissions -p arbiter rabbitmqadmin ".*" ".*" ".*"
docker exec rabbitmq rabbitmqctl set_permissions -p guest rabbitmqadmin ".*" ".*" ".*"
docker exec rabbitmq rabbitmqctl set_permissions -p host rabbitmqadmin ".*" ".*" ".*"
```
7.3. Проверка работоспособности RabbitMQ

bash
```
# Проверка статуса
docker exec rabbitmq rabbitmqctl status

# Проверка доступности веб-интерфейса
curl -I http://localhost:15672
```
Ожидаемый результат: HTTP-статус 200 OK

7.4. Важные замечания по RabbitMQ

ВНИМАНИЕ: RabbitMQ является **критически важным и обязательным компонентом** системы. Без работающего RabbitMQ невозможно:

- Взаимодействие между активным и пассивным серверами обучения
- Координация процессов PSI
- Обмен градиентами и параметрами моделей
- Асинхронная обработка задач

8. Установка серверов обучения (Training Servers)

8.1. Подготовка рабочих директорий

Создайте директории для монтирования данных:
bash
```
# Создание структуры директорий
mkdir -p ./example/workdir/active
mkdir -p ./example/workdir/passive

# Проверка прав доступа
chmod 755 ./example/workdir/active
chmod 755 ./example/workdir/passive
```

8.2. Клонирование репозитория (при необходимости)

bash
```
# Клонирование репозитория VFL
git clone https://git.ucb.local/scm/cc/vfl.git

# Переход в директорию проекта
cd vfl
```

8.3. Запуск пассивного сервера обучения

bash
```
docker run -d \
  --name ucbvfl-server-training_passive \
  --network vflnet \
  -p 50001:50001 \
  -v "$(pwd)/example/workdir/passive":/mnt/ext \
  nexus.ucb.infra/conf-compute/vfl training_passive \
    --work_dir=/mnt/ext \
    --listening_address=0.0.0.0:50001
```

Параметры запуска:

- -p 50001:50001 — проброс порта для внешнего доступа
- -v — монтирование директории с данными
- training_passive — режим работы пассивного сервера
- --work_dir — рабочая директория внутри контейнера
- --listening_address — адрес для прослушивания

8.4. Запуск активного сервера обучения

bash
```
docker run -d \
  --name ucbvfl-server-training_active \
  --network vflnet \
  -p 50000:50000 \
  -v "$(pwd)/example/workdir/active":/mnt/ext \
  nexus.ucb.infra/conf-compute/vfl training_active \
    --work_dir=/mnt/ext \
    --listening_address=0.0.0.0:50000 \
```

--passive_server_address=ucbvfl-server-training_passive:50001

Параметры запуска:

- training_active — режим работы активного сервера
- --passive_server_address — адрес пассивного сервера (используется имя контейнера в сети vflnet)

9. Установка PSI-сервера

9.1. Подготовка директорий данных

bash
Создание структуры директорий для PSI
mkdir -p extdir/server/data/input
mkdir -p extdir/server/data/output
mkdir -p extdir/client/data/input
mkdir -p extdir/client/data/output

Установка прав
chmod -R 755 extdir

9.2. Подготовка входных данных

В extdir/server/data/input/ поместите файл server_dataset_hashes.txt с хэшами идентификаторов серверных данных:
bash
Пример создания тестового файла
echo "hash1" > extdir/server/data/input/server_dataset_hashes.txt
echo "hash2" >> extdir/server/data/input/server_dataset_hashes.txt

9.3. Запуск PSI-сервера

bash
docker run -d \
  --name psi-server \
  --network vflnet \
  -p 5051:5051 \
  -v "$(pwd)/extdir/server":/mnt/ext \
  psi-server \
  --input_file=/mnt/ext/data/input/server_dataset_hashes.txt \
  --output_dir=/mnt/ext/data/output \
  --listening=0.0.0.0:5051

Параметры запуска:

- --network vflnet — подключение к общей сети (можно также использовать psi-network при необходимости изоляции)
- -p 5051:5051 — проброс порта PSI-сервера
- --input_file — путь к файлу с хэшами внутри контейнера
- --output_dir — директория для сохранения результатов
- --listening — адрес для прослушивания запросов

10. Конфигурация после установки

10.1. Параметры PSI

Параметр	Описание	Значение по умолчанию	Диапазон
Хэш-функция	Алгоритм хэширования идентификаторов	SHA-512	-
Максимальный размер файла	Макс. количество строк в ID-файле	10 000	-
batch_size	Размер чанка ID-файла	1024	128–10 000
max_retries	Число попыток повторной передачи	3	1–10
parallelism	Число параллельных потоков	4	1–16

10.2. Параметры SecureBoost

Параметр	Описание	Примеры значений
num_trees	Количество деревьев (итераций бустинга)	100, 200
max_depth	Максимальная глубина дерева	3, 6, 10
learning_rate	Скорость обучения (shrinkage)	0.01, 0.1, 0.2
objective	Функция потерь(оптимизации)	'binary:bce', 'regression:mse'
task_type	Тип задачи	'classification', 'regression'
min_child_weight	Минимальный вес дочернего узла	1.0, 5.0
subsample	Доля случайных образцов	0.6, 0.8, 1.0
colsample_bytree	Доля признаков для дерева	0.6, 0.8, 1.0
random_state	Фиксация уровня случайности	42, 2025

11. Проверка установки

11.1. Статус контейнеров

bash
Проверка всех контейнеров системы
docker ps --filter "name=psi-server" --filter "name=ucbvfl" --filter "name=rabbitmq"
Ожидаемый результат: Все контейнеры имеют статус Up (работают).

11.2. Прослушиваемые порты

bash
Проверка открытых портов
ss -tln | grep -E "5051|5672|15672|50000|50001"
Ожидаемый результат: Все порты отображаются в состоянии LISTEN.

11.3. Проверка сетевого взаимодействия

bash
Проверка доступности контейнеров по именам в сети vflnet
docker run --rm --network vflnet busybox ping -c 2 rabbitmq
docker run --rm --network vflnet busybox ping -c 2 ucbvfl-server-training_active
docker run --rm --network vflnet busybox ping -c 2 psi-server

11.4. Проверка томов и прав доступа

bash
Проверка монтирования директорий
docker exec psi-server ls -la /mnt/ext/data/input
docker exec psi-server ls -la /mnt/ext/data/output
docker exec ucbvfl-server-training_active ls -la /mnt/ext

Проверка прав доступа
docker exec psi-server stat -c "%U:%G %a" /mnt/ext
docker exec ucbvfl-server-training_active stat -c "%U:%G %a" /mnt/ext

11.5. Проверка логов

bash
Просмотр последних записей логов
docker logs psi-server --tail 20
docker logs ucbvfl-server-training_active --tail 20
docker logs ucbvfl-server-training_passive --tail 20
docker logs rabbitmq --tail 20
Ищите маркеры готовности:
- PSI-сервер: Listening on 0.0.0.0:5051
- VFL-серверы: Waiting for tasks on RabbitMQ или Connected to rabbitmq
- RabbitMQ: Server startup complete
- Отсутствие ошибок: ERROR, Traceback, Exception

11.6. Комплексная проверка

bash
Скрипт быстрой проверки

```
echo "=== Проверка контейнеров ==="
docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}" | grep -E
"rabbitmq|ucbvfl|psi"

echo -e "\n=== Проверка портов ==="
for port in 5051 5672 15672 50000 50001; do
  if ss -tln | grep -q ":$port"; then
    echo "Порт $port: OK"
  else
    echo "Порт $port: НЕ ПРОСЛУШИВАЕТСЯ"
  fi
done
```

12. Остановка сервисов

12.1. Корректная остановка контейнеров

Для корректного завершения работы рекомендуется останавливать контейнеры в определенном порядке:

bash
Переход в каталог проекта (если требуется)
cd /opt/psi_vfl/

Просмотр запущенных контейнеров
docker ps

Остановка в порядке зависимости (сначала клиенты, затем серверы)
docker stop psi-server
docker stop ucbvfl-server-training_active
docker stop ucbvfl-server-training_passive
docker stop rabbitmq

12.2. Проверка остановки

bash
Убедитесь, что контейнеры остановлены
docker ps | grep -E "rabbitmq|ucbvfl|psi"

Ожидаемый результат: Список пуст или не содержит указанных контейнеров.

12.3. Удаление контейнеров (при необходимости)

bash
Удаление остановленных контейнеров
docker rm psi-server
docker rm ucbvfl-server-training_active
docker rm ucbvfl-server-training_passive
docker rm rabbitmq

12.4. Удаление сети

bash
Удаление сети vflnet
docker network rm vflnet

Удаление дополнительной сети PSI (если создавалась)
docker network rm psi-network

12.5. Полная очистка (при необходимости)

bash
Остановка и удаление всех контейнеров VFL
docker stop $(docker ps -q --filter "name=ucbvfl") **2**>/dev/null
docker stop $(docker ps -q --filter "name=psi") **2**>/dev/null
docker stop rabbitmq **2**>/dev/null

docker rm $(docker ps -aq --filter "name=ucbvfl") **2**>/dev/null

```
docker rm $(docker ps -aq --filter "name=psi") 2>/dev/null
docker rm rabbitmq 2>/dev/null

# Удаление сетей
docker network rm vflnet 2>/dev/null
docker network rm psi-network 2>/dev/null
```

13. Заключение

13.1. Резюме

Данный документ покрывает полный цикл установки и настройки системы вертикального федеративного обучения (VFL):

1. **Подготовка инфраструктуры:**
   - Проверка системных требований
   - Загрузка необходимых Docker-образов
   - Создание сетевой изоляции (vflnet)
2. **Развертывание компонентов:**
   - Установка и настройка RabbitMQ (обязательный компонент)
   - Запуск активного и пассивного серверов обучения
   - Развертывание PSI-сервера для безопасного пересечения множеств
3. **Конфигурация и проверка:**
   - Настройка параметров PSI и SecureBoost
   - Валидация работоспособности всех компонентов
   - Диагностика с использованием логов и сетевых утилит
4. **Эксплуатация:**
   - Корректная остановка сервисов
   - Управление контейнерами и сетями

13.2. Ключевые особенности системы

- **Безопасность данных:** PSI-протокол обеспечивает конфиденциальность идентификаторов участников
- **Масштабируемость:** Поддержка чанковой обработки данных до 10 000 строк с настраиваемым параллелизмом
- **Отказоустойчивость:** Механизм повторных попыток (max_retries) при сетевых сбоях
- **Гибкость настройки:** Широкий выбор параметров SecureBoost для различных задач машинного обучения

13.3. Дальнейшие шаги

После успешного развертывания системы рекомендуется:

1. **Выполнить тестовый запуск PSI** для проверки корректности определения пересечений
2. **Запустить обучение модели** на тестовых данных с использованием SecureBoost
3. **Проверить сохранение результатов** в директориях /data/demo/active/models/ и /data/demo/passive/models/
4. **Выполнить предсказания** с использованием обученной модели
5. **Настроить мониторинг** для отслеживания состояния контейнеров и производительности

13.4. Устранение неполадок

При возникновении проблем:

1. Проверьте логи контейнеров: docker logs <container_name>
2. Убедитесь в доступности RabbitMQ: docker exec rabbitmq rabbitmqctl status
3. Проверьте сетевые настройки: docker network inspect vflnet

4. Проверьте права доступа к монтируемым директориям
5. При необходимости выполните полную перезагрузку: остановка → удаление → повторный запуск