

**Программа для ЭВМ
«Виртуальный помощник «СмартБот Техника»
с искусственным интеллектом (ИИ)»**

Руководство по установке и эксплуатации

Листов: 9

Москва, 2026

СОДЕРЖАНИЕ

1	ВВЕДЕНИЕ.....	ERROR! BOOKMARK NOT DEFINED.
1.1	Общие сведения и область применения	Error! Bookmark not defined.
1.2	Термины, сокращения и определения	Error! Bookmark not defined.
2	ПОДГОТОВИТЕЛЬНЫЕ ОПЕРАЦИИ....	ERROR! BOOKMARK NOT DEFINED.
3	СОСТАВ ДИСТРИБУТИВА	ERROR! BOOKMARK NOT DEFINED.
4	УСТАНОВКА	ERROR! BOOKMARK NOT DEFINED.
4.1	Создание домашней директории для Системы.....	Error! Bookmark not defined.
4.2	Установка СУБД	Error! Bookmark not defined.
4.3	Установка	Error! Bookmark not defined.
4.4	Установка	Error! Bookmark not defined.
4.5	Установка из дистрибутива	Error! Bookmark not defined.
4.6	Настройка	Error! Bookmark not defined.
4.7	Настройка временной зоны.....	Error! Bookmark not defined.
4.8	Просмотр журнала и текущего состояния	Error! Bookmark not defined.
5	ЗАПУСК	ERROR! BOOKMARK NOT DEFINED.
	СВЕДЕНИЯ ПО ЭКСПЛУАТАЦИИ:	

1 ВВЕДЕНИЕ

1.1 Общие сведения и область применения

Настоящее руководство содержит описание процедур подготовки среды, установки системных компонентов, настройки CI/CD-агента и запуска программы «Виртуальный помощник «СмартБот Техника» с искусственным интеллектом (ИИ)» (далее — Система) на продуктовом сервере.

Документ предназначен для DevOps-инженеров и системных администраторов, отвечающих за развертывание и ввод Системы в эксплуатацию.

1.2 Термины, сокращения и определения

Продуктовый сервер (Production) — целевой виртуальный или физический сервер, на котором функционирует Система.

DevOps-сервер — выделенный сервер, на котором развернут Self-Hosted GitLab.

GitLab Runner — утилита (агент с открытым исходным кодом), которая запускается на продуктовом сервере и выполняет инструкции из конвейера CI/CD GitLab.

Регистрационный токен (Registration Token) — уникальный ключ безопасности, выпускаемый GitLab для привязки Runner к конкретному репозиторию или группе.

Docker Swarm — встроенный в Docker инструмент оркестрации контейнеров, используемый для управления жизненным циклом Системы.

2 ПОДГОТОВИТЕЛЬНЫЕ ОПЕРАЦИИ

Перед началом установки необходимо убедиться, что продуктовый сервер соответствует минимальным требованиям и подготовлен к сетевому взаимодействию:

Установлена операционная система Ubuntu Server 22.04 LTS.

Настроен статический IP-адрес, сервер доступен по протоколу SSH.

Обеспечен сетевой доступ с продуктового сервера на DevOps-сервер по портам 80/443.

Сгенерирована пара SSH-ключей для административного доступа.

3 СОСТАВ ДИСТРИБУТИВА (ИНФРАСТРУКТУРНЫЕ КОМПОНЕНТЫ)

Ввиду использования GitOps-подхода, Система не поставляется в виде физического архива. Роль дистрибутива выполняет ветка main в удаленном репозитории на Self-Hosted GitLab.

Для развертывания кода из репозитория на продуктовом сервере предварительно разворачивается следующий стек базового ПО:

СУБД (Уровень данных): Redis (разворачивается автоматически в контейнере).

Среда выполнения: Docker Engine + Docker Compose плагин.

Агент доставки (CD): GitLab Runner

4 УСТАНОВКА

4.1 Создание домашней директории для Системы

Для изоляции конфигурационных файлов и хранения бэкапов сессий необходимо создать рабочую директорию в системе:

```
sudo mkdir -p /srv/smartbot/data  
sudo chown -R $USER:$USER /srv/smartbot
```

4.2 Установка среды контейнеризации (Docker Engine)

Установка Docker на ОС Ubuntu выполняется через официальный репозиторий:

```
sudo apt-get update  
sudo apt-get install ca-certificates curl gnupg  
sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://docker.com | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
sudo chmod a+r /etc/apt/keyrings/docker.gpg  
echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]  
https://docker.com \  
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update  
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-  
compose-plugin -y
```

4.3 Инициализация оркестратора Docker Swarm

Для запуска Системы и управления контейнерами необходимо переключить Docker в swarm-режим

```
sudo docker swarm init
```

4.4 Установка GitLab Runner

Для того чтобы продуктовый сервер мог принимать команды от Self-Hosted GitLab, на него устанавливается официальный агент:

```
sudo curl -L --output /usr/local/bin/gitlab-runner https://amazonaws.com  
sudo chmod +x /usr/local/bin/gitlab-runner
```

```
sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
sudo systemctl start gitlab-runner
```

Добавление пользователя gitlab-runner в группу docker, чтобы он мог выполнять сборку:

```
sudo usermod -aG docker gitlab-runner
sudo systemctl restart docker
```

4.5 Связка и регистрация GitLab Runner в Self-Hosted GitLab

Чтобы агент начал слушать команды из репозитория:

Зайдите в ваш веб-интерфейс GitLab -> Откройте проект «Смартбот» -> Settings -> CI/CD -> Разверните вкладку Runners.

Скопируйте URL вашего инстанса и Registration Token.

На продуктовом сервере выполните команду регистрации:

```
sudo gitlab-runner register \
  --non-interactive \
  --url "http://<URL_ВАШЕГО_СЕЛФ_ХОСТЕД_ГИТЛАБА>/" \
  --registration-token "КОД_ТОКЕНА_ИЗ_НАСТРОЕК_ПРОЕКТА" \
  --executor "shell" \
  --description "prod-runner-smartbot" \
  --tag-list "prod,swarm" \
  --run-untagged="false" \
  --locked="true"
```

4.6 Настройка переменных окружения (Конфигурация)

Перед первым запуском в настройках проекта на Self-Hosted GitLab необходимо создать переменную окружения, в который прописываются доступы к мессенджерам, CRM и ИИ.

```
MAX_BOX_TOKEN=Token...
TELEGRAM_BOT_TOKEN=Token...
CRM_BASE_URL=https://yourcompany.ru
CRM_AUTH_TOKEN=Token...
CHAD_API_URL=https://ai-engine.ru
CHAD_API_KEY=Key...
```

```
CHAD_MODEL=gpt-5-mini
```

```
REDIS_URL=redis://redis:6379
```

4.7 Настройка временной зоны

Для корректного логирования процессов и фиксации точного времени активации сертификатов сервер должен быть переведен на единую временную зону (например, Московское время):

```
sudo timedatectl set-timezone Europe/Moscow
```

4.8 Просмотр текущего состояния

Проверка того, что агент успешно запущен, связан с Self-Hosted GitLab и ожидает коммитов:

```
sudo systemctl status gitlab-runner
```

```
sudo gitlab-runner verify
```

5 ЗАПУСК И СВЕДЕНИЯ ПО ЭКСПЛУАТАЦИИ

5.1 Первый запуск через CI/CD

Запуск Системы происходит автоматически без прямого ввода команд на продуктивном сервере.

В корне репозитория проекта находится файл описания конвейера `.gitlab-ci.yml`.

При совершении коммита (Push) в ветку main GitLab отправляет команду зарегистрированному Runner'у на продуктивном сервере.

Runner выполняет шаги скрипта деплоя:

Собирает свежий образ Системы.

Разворачивает стек контейнеров Ядра и Redis в Docker Swarm с использованием файла конфигурации `stack.yml`.

5.2 Сведения по эксплуатации

Мониторинг запущенных контейнеров:

Контроль осуществляется через команду `docker service ls`.

Все заявленные реплики должны находиться в статусе 1/1 (или 3/3 при масштабировании).

Изоляция данных сессий:

Директория `/srv/smartbot/data` на сервере примонтирована к контейнеру Redis (volume).

Это гарантирует, что при обновлении кода Ядра или перезапуске контейнеров данные о текущих регистрациях и открытых сессиях пользователей не сотрутся.

Обновление конфигурации:

При необходимости смены токенов (например, замена API-ключа ИИ) изменения вносятся в настройки CI/CD проекта в Self-Hosted GitLab, после чего служба Ядра перезапускается без прерывания сессий.