true.code
software solutions

Руководство по установке на сервер

сервиса
«Моментальные платежи - Кредит в корзине»
B2B

1. Общие сведения и подготовка к установке 3

2. Окружения и структура настроек 4

3. Добавление сервера и настройка SSH 5

4. Настройки сервера и ключи хостов 6

5. Настройка секретов Ansible Vault 7

6. Первичная установка приложения 8

7. Обновление приложения вручную 9

8. Развертывание через GitLab CI/CD 10

9. Проверка установки и первичная настройка 11

10. Возможные затруднения 12

Контакты 13

1. Общие сведения и подготовка к установке

1.1. Настоящее руководство описывает установку и обновление сервиса B2B «Моментальные платежи - Кредит в корзине» на сервере. Руководство предназначено для специалиста, выполняющего развертывание приложения.

1.2. Для каждого клиента используются отдельный сервер и домен. Серверная часть работает на Node.js 20, NestJS, TypeORM и SQLite. Клиентская часть использует React 19, TypeScript и Vite. Компоненты поставляются в Docker-образах backend и web и запускаются через Docker Compose.

1.3. Ansible подготавливает сервер, устанавливает Docker, nginx и certbot, формирует файлы .env и docker-compose.yml, выполняет вход в registry и запускает приложение. Nginx раздает статику и проксирует API. Данные и журналы размещаются в томах.

1.4. До начала установки необходимо подготовить доступ к репозиторию проекта, серверу по SSH, GitLab Container Registry и паролю Ansible Vault. На машине управления должны быть доступны Ansible, SSH и Bash. Все команды в руководстве выполняются из корня репозитория, если не указано иное.

1.5. Подготовьте доменное имя, направленное на сервер, и доступ для выпуска HTTPS-сертификата. Убедитесь, что сервер может обращаться к registry, API Сбербанка и используемым внешним сервисам. Пользователь Ansible должен иметь права для установки пакетов и настройки служб.

1.6. Первичная установка выполняется скриптом init-env.sh. Последующие обновления выполняются локально или через GitLab CI/CD. Задания CI не устанавливают Docker, nginx и certbot на новый сервер.

2. Окружения и структура настроек

2.1. Выбор окружения

2.1.1. Для тестового окружения используются ветка разработчика, группа Ansible test и инвентарь ansible/inventory/test/hosts.yml. Для промышленного окружения используются главная ветка, группа prod и инвентарь ansible/inventory/prod/hosts.yml.

2.1.2. В командах обозначение <env> заменяется на test или prod. Параметр target_group должен совпадать с именем группы в hosts.yml. Значения в угловых скобках необходимо заменить своими данными, сами скобки в команду включать не нужно.

2.2. Файлы инвентаря

2.2.1. Файл ansible/inventory/<env>/hosts.yml содержит список серверов среды. Файл ansible/inventory/<env>/known_hosts содержит SSH-ключи хостов, в CI он копируется в ~/.ssh/known_hosts.

2.2.2. Общие настройки находятся в файле ansible/inventory/<env>/group_vars/all/main.yml. В нем задаются project_name, env_name, deploy_path, docker_project_name, registry_url, registry_project_path, backend_image и web_image.

2.2.3. В этом же файле задаются image_tag (по умолчанию latest), jwt_expires_in, b2b_auth_url и letsencrypt_email, формируются полные имена образов backend_image_full и web_image_full.

2.2.4. Настройки отдельного сервера находятся в файле ansible/inventory/<env>/host_vars/<hostname>/main.yml. Значение <hostname> соответствует имени сервера в инвентаре.

2.3. Тег образов

2.3.1. Параметр image_tag определяет устанавливаемую версию образов. Скрипт init-env.sh по умолчанию использует latest. В CI в bootstrap.yml передается тег CI_COMMIT_SHORT_SHA. До запуска убедитесь, что образы backend и web с выбранным тегом доступны в registry.

3. Добавление сервера и настройка SSH

3.1. Добавьте сервер с помощью скрипта. При стандартном SSH-порте 22 последний аргумент можно не указывать:

```
bash ansible/scripts/add-host.sh <env> <host_name> <ip> <domain> [ssh_port]
```

Пример для промышленного окружения и нестандартного порта:

```
bash ansible/scripts/add-host.sh prod client-x \
  203.0.113.20 client.example.com 2266
```

3.2. Скрипт создает host_vars/<host_name>/main.yml, при необходимости добавляет сервер в hosts.yml и дописывает SSH-ключ в known_hosts через ssh-keyscan. Для нестандартного порта используется ssh-keyscan -p <ssh_port>.

3.3. Проверьте, что сервер находится в нужной группе hosts.yml. В host_vars проверьте ansible_host, ansible_user и ansible_port. Если ansible_port не указан, используется порт 22.

3.4. Добавьте публичный ключ машины управления на сервер. Пользователь должен совпадать с ansible_user. Пример для нестандартного порта:

```
ssh-copy-id -p <ssh_port> -i ~/.ssh/id_ed25519_ansible.pub \
  <ansible_user>@<ip>
```

3.5. Проверьте подключение с соответствующим приватным ключом:

```
ssh -p <ssh_port> -i ~/.ssh/id_ed25519_ansible \
  <ansible_user>@<ip>
```

3.6. Для порта 22 параметр -p можно опустить. При запуске Ansible используйте тот же приватный ключ, через ansible_ssh_private_key_file, параметр --private-key или ssh-agent.

Обратите внимание, при ручном заполнении known_hosts используйте команду из следующего раздела. Сообщение add-host.sh о host_vars/.../vault.yml относится к необязательным секретам отдельного сервера.

4. Настройки сервера и ключи хостов

4.1. Заполнение host_vars

4.1.1. Откройте файл ansible/inventory/<env>/host_vars/<hostname>/main.yml и проверьте параметры подключения: ansible_host - адрес сервера, ansible_user - пользователь SSH, ansible_port - порт SSH при отличии от 22.

4.1.2. Укажите app_domain - домен приложения, web_port и backend_port - порты компонентов, database_path - путь к данным базы и log_dir - каталог журналов.

4.1.3. Заполните server_url, client_mail, payment_base_url и support_api_key. Проверьте URL сервисов, почту клиента, базовый URL оплаты и ключ поддержки в соответствии с настройками конкретного клиента.

4.2. Заполнение known_hosts вручную

4.2.1. Если ключ не был добавлен скриптом, получите его для стандартного порта:

```
ssh-keyscan -H <ip> >> ansible/inventory/<env>/known_hosts
```

Для нестандартного порта выполните:

```
ssh-keyscan -p <ssh_port> -H <ip> \
  >> ansible/inventory/<env>/known_hosts
```

4.2.2. Значение <ssh_port> должно совпадать с ansible_port в host_vars. Иначе проверка ключа при подключении к фактическому порту не пройдет. Перед использованием сверяйте отпечаток ключа с доверенными данными сервера.

4.2.3. Сохраните и закоммитьте обновленный known_hosts вместе с изменениями инвентаря. В CI используется этот файл: для test - из ansible/inventory/test/, для prod - из ansible/inventory/prod/.

5. Настройка секретов Ansible Vault

5.1. Подготовьте зашифрованные секреты для выбранной среды. В описанной конфигурации test они находятся в ansible/inventory/test/group_vars/all/vault.yml и являются общими для всех экземпляров тестовой среды.

5.2. В vault задаются registry_username и registry_password для входа в registry, а также internal_api_key, jwt_secret и encryption_key для файла .env приложения. Примеры имен переменных приведены в комментариях group_vars/all/main.yml тестовой среды.

5.3. Для просмотра содержимого тестового vault выполните:

```
ansible-vault view \
  ansible/inventory/test/group_vars/all/vault.yml
```

Если пароль хранится в локальном файле, используйте:

```
ansible-vault view \
  ansible/inventory/test/group_vars/all/vault.yml \
  --vault-password-file .vault_pass
```

5.4. Для промышленной среды подготовьте соответствующий vault до установки. Состав переменных должен соответствовать playbook-файлам и шаблону ansible/templates/app.env.j2 выбранной среды.

5.5. При ручном запуске playbook-файлов параметр --ask-vault-pass запрашивает пароль интерактивно. Вместо него можно использовать --vault-password-file .vault_pass.

5.6. В CI пароль передается переменной ANSIBLE_VAULT_PASSWORD. Задание записывает его в .vault_pass и передает этот файл каждому вызову ansible-playbook.

Важно: файл .vault_pass содержит пароль в открытом виде. Не включайте его в коммит и ограничьте доступ к нему. Данные для подключения к API Сбербанка вводятся отдельно в административном разделе после установки.

6. Первичная установка приложения

6.1. Перед установкой убедитесь, что сервер добавлен в инвентарь, host_vars заполнен, SSH-подключение работает, ключ хоста сохранен, vault подготовлен и выбранные образы доступны в registry.

6.2. Запустите полную инициализацию среды:

```
bash ansible/scripts/init-env.sh <env> [image_tag]
```

Например, для тестового окружения с тегом latest:

```
bash ansible/scripts/init-env.sh test latest
```

6.3. Второй аргумент необязателен; по умолчанию используется latest. Скрипт выбирает инвентарь ansible/inventory/<env>/hosts.yml. При запросе введите пароль Ansible Vault.

Важно: скрипт работает с инвентарем среды. Перед запуском проверьте состав целевой группы, команда не содержит отдельного аргумента для выбора одного сервера.

6.4. Последовательность установки

6.4.1. bootstrap.yml - подготовка каталогов и шаблонов .env и docker-compose.yml с параметрами target_group=<env> и image_tag=<tag>.

6.4.2. install_docker.yml, install_nginx.yml и install_certbot.yml - установка Docker, nginx и certbot.

6.4.3. configure_nginx.yml и issue_certificate.yml - настройка nginx и выпуск сертификата Let’s Encrypt.

6.4.4. registry_login.yml и deploy_app.yml - вход в registry, получение образов и запуск приложения через Docker Compose.

6.4.5. Все перечисленные playbook-файлы находятся в ansible/playbooks/. Для bootstrap, настройки nginx, выпуска сертификата, входа в registry и развертывания используется доступ к vault. После завершения выполните проверку по разделу 9.

7. Обновление приложения вручную

7.1. Обновление через Ansible

7.1.1. Для уже подготовленного сервера выполните три команды последовательно: обновление шаблонов и каталогов, вход в registry, развертывание образов. Пример для test:

```
ansible-playbook -i ansible/inventory/test/hosts.yml \
  ansible/playbooks/bootstrap.yml --ask-vault-pass \
  -e "target_group=test image_tag=latest"

ansible-playbook -i ansible/inventory/test/hosts.yml \
  ansible/playbooks/registry_login.yml --ask-vault-pass \
  -e "target_group=test"

ansible-playbook -i ansible/inventory/test/hosts.yml \
  ansible/playbooks/deploy_app.yml --ask-vault-pass \
  -e "target_group=test"
```

7.1.2. Для prod замените путь инвентаря на ansible/inventory/prod/hosts.yml и укажите target_group=prod. Для установки конкретной версии замените latest на требуемый тег образов. При необходимости используйте --vault-password-file вместо --ask-vault-pass.

7.2. Обновление промышленной среды скриптом

7.2.1. Из корня репозитория выполните:

```
./deploy-prod.sh [<image_tag>]
```

7.2.2. Скрипт выполняет ту же последовательность, что и deploy_prod в CI. Если тег не указан, внутри git-дерева используется результат git rev-parse --short HEAD. Вне git-дерева тег обязателен.

7.2.3. Пароль vault можно ввести интерактивно, указать в файле .vault_pass в корне репозитория либо передать путь к файлу через переменную ANSIBLE_VAULT_PASSWORD_FILE.

7.2.4. После обновления проверьте доступность страницы и состояние приложения по разделу 9.

8. Развертывание через GitLab CI/CD

8.1. Подготовка переменных

8.1.1. В настройках CI/CD проекта задайте SSH_PRIVATE_KEY - приватный SSH-ключ для доступа к серверам, и ANSIBLE_VAULT_PASSWORD - пароль Ansible Vault. Настройте masked/protected в соответствии с политикой команды и доступом нужных веток.

8.1.2. В задании деплоя SSH_PRIVATE_KEY записывается в ~/.ssh/id_rsa с правами 600. В ~/.ssh/known_hosts копируется файл соответствующего окружения. Публичная часть ключа должна быть добавлена на целевые серверы пользователю ansible_user.

8.2. Сборка образов

8.2.1. Pipeline состоит из стадий build и deploy. Задания build_backend и build_web собирают образы по файлам docker/backend.Dockerfile и docker/frontend.Dockerfile и публикуют их в GitLab Container Registry:

```
$CI_REGISTRY_IMAGE/backend:$CI_COMMIT_SHORT_SHA
$CI_REGISTRY_IMAGE/web:$CI_COMMIT_SHORT_SHA
```

8.2.2. Для входа в registry на стадии build используются встроенные переменные GitLab: CI_REGISTRY_USER, CI_REGISTRY_PASSWORD и CI_REGISTRY.

8.3. Запуск развертывания

8.3.1. После push в ветку разработчика выполняется сборка и автоматический deploy_test с target_group=test и image_tag=$CI_COMMIT_SHORT_SHA.

8.3.2. После push в главную ветку дождитесь успешной сборки и вручную запустите deploy_prod. Задание доступно только для main, имеет when: manual и использует target_group=prod и тот же тег коммита.

8.3.3. В CI выполняются только bootstrap.yml, registry_login.yml и deploy_app.yml. Новый сервер предварительно подготовьте через init-env.sh по разделу 6.

9. Проверка установки и первичная настройка

9.1. Убедитесь, что Ansible или задание CI завершились без ошибок. При ошибке определите playbook и шаг, на котором остановилось выполнение, прежде чем повторять запуск.

9.2. Подключитесь к серверу и перейдите в каталог, заданный переменной deploy_path. Проверьте состояние контейнеров командой:

```
cd <deploy_path>
docker compose ps
```

9.3. При необходимости просмотрите сообщения контейнеров:

```
docker compose logs --tail=100
```

9.4. Откройте https://<app_domain> в браузере. Проверьте доступность страницы и HTTPS-сертификат. Дополнительно проверьте файлы журналов в log_dir и размещение данных по настройке database_path.

9.5. Откройте административный раздел и выполните настройку по «Руководству администратора». Укажите реквизиты получателя, ставку НДС, параметры API и сертификаты для используемых способов оплаты. Выберите нужный контур Сбербанка и сохраните настройки.

9.6. Выполните авторизацию владельца через СберБизнес ID отдельно для каждого подключаемого способа оплаты. Проверьте статус авторизации и отображение настроенных способов на платежной странице.

9.7. Проверьте ввод данных вручную и загрузку PDF-счета. В согласованном тестовом контуре выполните проверку платежного сценария, возврата после авторизации и страницы статуса.

Обратите внимание: успешный запуск контейнеров подтверждает развертывание приложения. Для приема платежей также необходимы корректные банковские настройки, авторизация владельца и действующий тариф клиента.

10. Возможные затруднения

10.1. Не удается подключиться по SSH. Проверьте ansible_host, ansible_user, ansible_port, доступность порта и соответствие приватного ключа публичному ключу на сервере.

10.2. Ошибка проверки ключа хоста. Проверьте запись known_hosts и порт, для которого она получена. При смене ключа сначала подтвердите его подлинность, затем обновите файл инвентаря и коммит для CI.

10.3. Не удается прочитать vault. Проверьте пароль, расположение vault для выбранной среды и параметр --vault-password-file. В CI проверьте доступность ANSIBLE_VAULT_PASSWORD заданию.

10.4. Образы не загружаются. Проверьте registry_url, registry_project_path, имена backend_image и web_image, выбранный image_tag, учетные данные registry и результат стадии build.

10.5. Страница недоступна или сертификат не выпущен. Проверьте домен и его адресацию, настройки nginx, доступность сервера и результат configure_nginx.yml и issue_certificate.yml. Убедитесь, что первичная установка выполнена полностью.

10.6. Контейнеры запущены, но приложение работает с ошибкой. Проверьте сформированный .env, host_vars и журналы. Сопоставьте ошибку с параметрами внешних сервисов и банковскими настройками.

10.7. Способ оплаты не отображается. Проверьте его настройку и авторизацию владельца в административном разделе. При предупреждении «Сервис временно недоступен» проверьте доступность банковского сервиса. При блокировке интерфейса проверьте тариф клиента.

10.8. Если устранить ошибку не удалось, передайте в поддержку описание, время возникновения и этап установки. Не включайте в обращение приватные ключи, пароли, токены или содержимое vault.

Контакты

ИТ-компания true.code

Менеджер проектов:
Синицын Вячеслав

email: pay@t-code.ru
https://t-code.ru/