АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ ПО ТЕХНОЛОГИИ ЕДИНОГО ВХОДА

Руководство по установке

Версия 10.3.0

nexign

АО «Нэксайн»

Россия, 199155, Санкт-Петербург, ул. Уральская, д.4 лит.Б, помещение 22Н

Тел.: + 7 (812) 326-12-99; факс: + 7 (812) 326-12-98.

office@nexign.com; www.nexign.com

Содержание

1. Предварительные условия

Для развёртывания функциональных модулей продукта SSO требуются серверы, на которых установлены:

- ОС Linux RedHat 8.0, CENTOS 8.1 или РЕД ОС версии 7.2 или выше, с запущенной службой NTP;
- OpenJDK Java.

Также требуется подготовить серверы со следующим программным обеспечением:

- СУБД PostgreSQL;
- «Сервер кэшей» (COUCHBASE);
- RabbitMQ – (опционально) для отправки нотификаций;
- «Шлюз доступа к API» (API_GATEWAY) – (опционально) для работы веб-интерфейса.

На сервере, на котором планируется запускать инсталлятор, должны быть установлены следующие компоненты:

- операционная система Linux RedHat 8.0 (с запущенной службой NTP), CENTOS 8.1 или РЕД ОС версии 7.2 или выше;
- Ansible 2.10.7;
- OpenJDK;
- Jinja2 2.11.3;
- Python 3.6 с пакетами:
  - jmespath 0.9.3;
  - lxml 4.2.5;
  - kazoo 2.5.0;
  - pywinrm 0.3.0;
  - natsort 6.0.0;
  - PyYAML 5.4.1;

Актуальные версии программного обеспечения см. в файле releasenotes, поставляемом в составе дистрибутива.

2. Установка продукта

Для установки продукта выполните следующие действия:

1. Скачайте и разархивируйте дистрибутив продукта.
2. Создайте схему развертывания.
3. Настройте общие параметры.
4. Настройте хосты.
5. Выполните конфигурацию продукта.
6. Запустите автоинсталлятор.
7. Переключитесь на новую версию продукта и зафиксируйте ее.
8. При необходимости выполните откат на предыдущую версию продукта.

2.1. Создание схемы развертывания

В каталоге дистрибутива install/inventory/localhost содержится пример с настройками схемы развёртывания.

Чтобы создать собственную схему развёртывания, создайте копию каталога localhost и переименуйте её (например, staging).

2.2. Настройки общих параметров установки

1. Задайте местоположение запускаемых компонентов, файлов журналирования, мониторинга в файле install/inventory/<схема_развертывания>/group_vars/all.yml.
2. При необходимости переопределите общие настройки установки каждого компонента и настройки конфигурации в файлах install/inventory/<схема_развертывания>/group_vars/<component_name>.yml.

Файлы журналирования в формате <component_name>.current.log для каждого запущенного компонента хранятся в каталоге, который определяет параметр `logs_dir` в файле install/inventory/<схема_развертывания>/group_vars/all.yml.

Подробнее о настройках inventory см. документацию Ansible.

2.3. Файлы Playbook

Настройте параметры установки общего playbook в файле install/install/group_vars/sso.yml для каждого компонента продукта или по отдельности для используемых компонентов продукта в соответствующих файлах yml.

Перечень файлов playbook:

- sso-check-environment.yml – проверка окружения;
- sso-db-couchbase.yml – установка Couchbase;
- sso-db-grants-postgresql.yml – выдача прав (PostgreSQL);
- sso-db-liquibase.yml – установка Liquibase;
- sso-db-sysdba-postgresql.yml – создание табличных пространств и пользователей базы данных PostgreSQL;
- sso-deploy-db.yml – установка компонентов базы данных;
- sso-deploy-finalize.yml – завершение установки и перезапуск приложения;
- sso-deploy-full.yml – общий скрипт установки;
- sso-deploy-rollback.yml – откат на последнюю успешную версию;
- sso-deploy-rsa-keys.yml – генерация новых пар RSA-ключей для подписи и шифрования токенов и загрузка их в ZooKeeper; время начала действия ключей устанавливается в

соответствии с параметром inventory `ps.sso.jwt.rsa_start_time`; при генерации новых ключей рекомендуется устанавливать время начала действия со смещением в будущее на несколько часов, чтобы сервисы, интегрированные с `SSO`, могли успеть обновить локальные кэши ключей;
- sso-deploy-success.yml – отметка текущей версии приложения как успешной;
- sso-deploy.yml – установка Java-based компонентов;
- sso-mbus-cm.yml – регистрация сервиса и конфигурации `SSO` в `MBUS_CM`;
- sso-nginx-components-deploy.yml – установка компонентов `NGINX`;
- sso-nginx-deploy.yml – подготовка к установке `NGINX`;
- sso-schema-generate.yml – создание схемы данных;
- sso-validate-parameters.yml – проверка конфигурационных параметров инсталлятора.

2.4. Настройка хостов

Настройте хосты сервера, на котором устанавливается продукт, и опишите группы серверов схемы развертывания в файле install/inventory/<схема_развёртывания>/1-isem-sso.yml.

В каждой группе укажите серверы, на которые необходимо выполнить установку компонентов продукта, и параметры соединения с ними в формате: alias, имя серверной машины `ansible_host` и метод подключения `ansible_connection`.

Пример:

```
sso-product:
   product-alias1:
    ansible_host: <адрес удаленной машины>
    ansible_connection: ssh
# Либо
sso-product:
  product-alias1:
    ansible_host: localhost
    ansible_connection: local
```

Файлы по умолчанию заполнены параметрами для локальной установки и в минимальной конфигурации. Нельзя изменять названия существующих групп. Можно добавлять новые хосты, новые группы и наследования.

2.5. Разделение трафика на группы экземпляров компонентов

Продукт поддерживает возможность конфигурирования установки нескольких групп для одного компонента `SSO` и правила разделения трафика между отдельными группами одного компонента.

Конфигурирование поддерживается для компонентов:

- `Access Control Service`;
- `Authentication Service`;
- `Directory Service`;
- `Provisioning Adapter`.

На примере компонента `Access Control Service` в примерах описано, как сконфигурировать две группы серверов, одна из которых обрабатывает запросы прав, вторая – остальные запросы.

Настройка разделения трафика включает следующие этапы:

- Конфигурирование нескольких групп для одного компонента;
- Конфигурирование правил разделения трафика между группами компонента;
- Кастомизация конфигурации для групп компонента, устанавливаемых на один сервер (опционально).

2.5.1. Конфигурирование нескольких групп для одного компонента

В файле 1-isem-sso.yml сконфигурируйте несколько дочерних групп для группы, соответствующей компоненту.

Для этого удалите ноду hosts внутри конфигурации нужного компонента и добавьте ноду children с конфигурацией нескольких групп компонента по следующему принципу:

```
all:
  children:
    sso:
      children:
        <...>
        <component_name>:
          children:
            <component_group1_name>:
              hosts:
                <component_group1_host1_name>:
                  ansible_host: <host1_1>
                <component_group1_host2_name>:
                  ansible_host: <host1_2>
                <...>
                <component_group1_hostN_name>:
                  ansible_host: <host1_N>
            <component_group2_name>:
              hosts:
                <component_group2_host1_name>:
                  ansible_host: <host2_1>
                <component_group2_host2_name>:
                  ansible_host: <host2_2>
                <...>
                <component_group2_hostM_name>:
                  ansible_host: <host2_M>
            <...>
            <component_groupN_name>:
              hosts:
                <component_groupN_host1_name>:
                  ansible_host: <hostN_1>
                <component_groupN_host2_name>:
                  ansible_host: <hostN_2>
                <...>
                <component_groupN_hostP_name>:
                  ansible_host: <hostN_P>
```

где:

- <component_name> – имя компонента (sso_access_control, sso_authentication, sso_directory, sso_provisioning_adapter);
- <component_group*_name> – имена групп компонента; поддерживается произвольное количество групп;
- <component_group*_host*_name> – имена хостов для групп компонента; поддерживается произвольное количество хостов в группе;
- <host*> – реальные имена хостов.

Пример:

```
all:
  children:
    sso:
      children:
        sso_access_control:
          children:
            sso_access_control_general:
              hosts:
                sso_access_control_general_local:
                  ansible_host: <host1>
            sso_access_control_calculator:
              hosts:
                sso_access_control_calculator_local:
                  ansible_host: <host2>
```

2.5.2. Конфигурирование правил разделения трафика между группами компонента

В файле inventory sso_bln_inner.yml сконфигурируйте правила маршрутизации трафика на хосты групп компонента.

Для этого задайте значение `ps.sso.bln.ports.inner.<component_name>.hostgroups` по следующему принципу:

```
ps:
  sso:
    bln:
      ports:
        inner:
          <component_name>:
            hostgroups:
              - name: "<component_group1_name>"
                rules:
                  - "<condition1_for_group1>"
                  - "<condition2_for_group1>"
                  - <...>
                  - "<conditionN_for_group1>"
              - name: "<component_group2_name>"
                rules:
                  - "<condition1_for_group2>"
                  - "<condition2_for_group2>"
                  - <...>
                  - "<conditionM_for_group2>"
              - <...>
              - name: "<component_groupN_name>"
```

где:

- <component_group*_name> – имена групп компонента (как в разделе «Конфигурирование нескольких групп для одного компонента»);
- <condition*> – множество условий балансировки трафика на хосты группы; синтаксис в соответствии с требованиями к условию внутри `NGINX` if.

Внимание!

Множество описанных в `hostgroups` групп компонента должно строго соответствовать множеству групп, описанных в разделе «Конфигурирование нескольких групп для одного компонента».

Внимание!

Одна из групп компонента, идущая в данном списке последней, не должна иметь условий балансировки, чтобы все запросы, не соответствующие условиям для предыдущих групп, маршрутизировались на неё.

В правилах маршрутизации можно использовать любые данные контекста запроса, доступные в контекстных переменных `NGINX`, например:

- `$uri` – строка запроса;
- `$http_x_ps_login` – логин вызывающего пользователя;

- `$http_x_ps_appl_code` – код приложения, через которое произведена аутентификация пользователя;
- прочие заголовки запроса, проставляемые продуктом `API GATEWAY`.

Пример:

```
ps:
  sso:
    bln:
      ports:
        inner:
          access_control:
            hostgroups:
              - name: "sso_access_control_calculator"
                rules:
                  - "$uri ~ \"^/ps/acc/api/users/grants/?$\""
              - name: "sso_access_control_general"
```

2.5.3. Кастомизация конфигурации для групп компонента, устанавливаемых на один сервер

Если экземпляры разных групп компонента устанавливаются на один сервер, необходимо кастомизировать конфигурации групп компонента в части прослушиваемых портов и названия приложения для supervisord.

Для устанавливаемых на один сервер групп компонента в каталоге group_vars создайте файлы <component_group_name>.yml содержания аналогичного <component_name>.yml, но с другим наименованием приложения (app_name) и значениями прослушиваемых портов (ps.sso.<component_name>.server.port, ps.sso.jvm.remote_debug_port), где:

- <component_group_name> – имя группы серверов компонента;
- <component_name> – имя компонента.

Параметр `app_name` для групп нужно задавать таким образом, чтобы наименование ни одной из групп не являлось наименованием другой группы. Например, неправильно:

`['sso-access-control', 'sso-access-control-calculator'],`

правильно:

`['sso-access-control-general', 'sso-access-control-calculator'].`

Пример для sso_access_control_calculator.yml:

```
app_name: "sso-access-control-calculator"
ps:
  sso:
    access_control:
      server:
        port: 47179
    config_diff:
      config_dir: "config"
    jvm:
      remote_debug_port: 5007

check_running:
  urls_list:
    - url:
"http://localhost:{{ps.sso.access_control.server.port}}/health"
  retry_count: 18
  retry_delay: 10
```

2.6. Запуск автоинсталлятора

Общий вид команды запуска:

```
ansible-playbook sso-deploy.yml -i inventory/localhost -u <user> -k
```

где:

- `-i <inventory/localhost>` – inventory, который будет использоваться при выполнении сценария (если указать каталог, то сценарий пройдется по всем хост-файлам, находящимся в указанном каталоге);
- `-u` – пользователь, от имени которого Ansible будет подключаться к серверам, указанным в файле хостов в `inventory`;
- `-k` – необходимость ввода пароля пользователя, указанного в ключе -u.

При необходимости можно добавить дополнительные параметры:

- `--become-user` – привилегированный пользователь;
- `-K` – необходимость ввода пароля для привилегированного пользователя, указанного в ключе `--become-user`;
- `-b` – необходимость выполнения сценария от имени привилегированного пользователя;
- `-t` – перечисление тегов, которые будут выполняться в сценарии (по умолчанию используется `all`, то есть запуск всех тегов);
- `-l` – перечисление хостов или групп, по которым будет выполнен сценарий (по умолчанию сценарий будет выполнен для всех хостов);
- `-v` – уровень журналирования; максимальное количество v=4 (-vvvv), чем больше v, тем выше уровень журналирования;
- `-e` – дополнительные переменные задаются в виде key=value или YAML/JSON. Если

указывается имя файла, то имя должно начинаться с @.

Полный список ключей можно посмотреть в официальной документации Ansible.

Для установки определенной версии продукта или компонента предусмотрена возможность задания встроенных переменных. Для этого при запуске инсталлятора задайте в параметре `--extra-vars(-e)` одну из следующих переменных:

- `version`;
- `path_version`;
- `<component>_version`;
- `<component>_path_version`;

где `<component>` – имя группы в hosts-файле и соответствующий этому имени файл в group_vars.

В этом случае будет установлена указанная версия.

Пример.

```
ansible-playbook sso-deploy.yml -i inventory/localhost --extra
-vars="path_version=3.2"
```

2.7. Принятие/финализация установленной версии

Установленная версия принимается или финализируется отдельным шагом при запуске роли `common/deploy-finalize`. На этом шаге удаляется файл DEPLOY_UNFINISHED и создается символическая ссылка current на принятую версию.

Переключитесь на новую версию вводом команды:

```
ansible-playbook sso-deploy-finalize.yml -i inventory/localhost
```

Если установленная версия работает корректно, ее можно зафиксировать для возможности отката на нее.

Зафиксируйте установку версии (статус новой версии продукта – успешно установлена) вводом команды:

```
ansible-playbook sso-deploy-success.yml -i inventory/localhost
```

2.8. Откат продукта на предыдущую версию

Для отката на предыдущую версию продукта выполните команду:

```
ansible-playbook sso-deploy-rollback.yml -i inventory/localhost
```

При запуске playbook возврата на предыдущую версию продукта возможны варианты выполнения:

- в случае присутствия на сервере финализированных предыдущих версий продукта

происходит откат на предыдущую версию, то есть переключение текущей ссылки (current) на предыдущую версию;
- в случае отсутствия на сервере предыдущих версий продукта сценарий отката будет остановлен с ошибкой «rollback_path is defined».

2.9. Конфигурация продукта

Конфигурация продукта задается в каталогах:

- install/inventory/default/group_vars;
- install/inventory/<схема_развертывания>/group_vars.

2.9.1. install/inventory/default/group_vars

В каталоге описаны параметры, используемые при конфигурации продукта, со значениями по умолчанию.

Внимание!

Изменение значений параметров в файлах каталога install/inventory/default/group_vars запрещено. Для изменения значений параметров используйте файлы в каталоге inventory/inventory/<схема_развертывания>/group_vars.

2.9.2. install/inventory/<схема_развертывания>/group_vars

В install/inventory/<схема_развертывания>/group_vars находятся настройки, которые могут изменяться. Эти настройки зависят от окружения, на которое будет устанавливаться продукт.

3. Запуск и остановка продукта

Для запуска, остановки и перезапуска продукта можно использовать инсталлятор, указав нужную команду (start, stop или restart):

```
sso-deploy.sh -c <COMMAND> -i <INVENTORY> -u <USER> [-d <DOMAIN>] [-p
<PLAYBOOK>] [-s <PRODUCT USER>] [-e <EXTRA VARS>] [-h]"
```

4. Обновление

Для обновления продукта выполните установку дистрибутива требуемой версии.

На текущий каталог установки указывает символическая ссылка с именем, совпадающим с именем компонента без временной метки.

При установке создается новый каталог с именем компонента, дополненный текущей временной меткой, в который и производится установка. Если установка произошла успешно, символическая ссылка сдвигается и начинает указывать на новый каталог.

5. Проверка работоспособности

Автоинсталлятор продукта в процессе работы проверяет работоспособность компонентов и в случае неработоспособности возвращает код, отличный от `0`.