Установка MinIO Tenant в Nova Container Platform

1. Предварительные условия

- ✓ У вас есть доступ к кластеру с учетной записью, имеющей роль `cluster-admin` в Kubernetes.
- ✓ Вы установили утилиту `kubectl` для работы с Kubernetes.
- ✓ Вы установили утилиту `helm` для работы с чартами.

2. Шаг первый. Установка MinIO Operator

1. Добавьте Helm репозиторий MinIO.

```bash
helm repo add minio-operator https://operator.min.io
```

2. Установите MinIO Operator.

```bash
helm install --namespace minio-operator --create-namespace operator minio-operator/operator
```

3. Шаг второй. Установка MinIO Tenant

1. Скачайте файл с переменными.

```bash
curl -sLo values.yaml https://raw.githubusercontent.com/minio/operator/master/helm/tenant/values.yaml
```

2. В файле с переменными измените значение *storageClassName* на `ovirt-csi-sc` или любой другой Storage Class, который можно использовать для *Persistent Volume*. Также, если нужно, измените остальные параметры, например, *volumesPerServer* и *servers*.
3. Установите MinIO Tenant.

```bash
helm install --namespace minio-tenant --create-namespace --values values.yaml minio-tenant minio-operator/tenant
```

4. Добавьте *Ingress* ресурс для доступа к MinIO консоли.

```yaml
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: minio-console
  namespace: minio-tenant
  annotations: ①
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  ingressClassName: nginx-public ②
  tls:
    - hosts:
        - minio.apps.nova.test.local
      secretName: minio.apps.nova.test.local
  rules:
    - host: minio.apps.nova.test.local ③
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: myminio-console ④
                port:
                  number: 9443
```

① Настройка перенаправления с HTTP на HTTPS.

② Указание, что нужно использовать встроенный nginx.

③ Ссылка для подключения. Можно изменить на более подходящую.

④ Имя сервиса, на который настроен *Ingress*.

5. Добавьте *Ingress* ресурс для доступа к MinIO API.

```yaml
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: minio-api
  namespace: minio-tenant
  annotations: ①
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  ingressClassName: nginx-public ②
  tls:
    - hosts:
```

```
        - minioapi.apps.nova.test.local
      secretName: minioapi.apps.nova.test.local
  rules:
    - host: minioapi.apps.nova.test.local ③
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: myminio-hl ④
                port:
                  number: 9000
```

① Настройка перенаправления с HTTP на HTTPS.

② Указание, что нужно использовать встроенный nginx.

③ Ссылка для подключения. Можно изменить на более подходящую.

④ Имя сервиса, на который настроен *Ingress*.

4. Проверка

1. Получите доступ к MinIO консоли по ссылке `https://minio.apps.nova.test.local`. Данные для подключения:

```
Username: minio
Password: minio123
```

2. Скачайте MinIO Client `mc` и подключитесь к MinIO API:

```
mc alias set myminio https://minioapi.apps.nova.test.local minio minio123 --
insecure
```

Добавление oVirt CSI в платформе установленной методом UPI

В руководстве описан процесс добавления oVirt CSI в платформе, но только на Worker узле, который располагается в системе виртуализации zVirt.

1. Предварительные условия

Nova Container Platform установлена методом UPI в минимальной конфигурации с двумя Worker узлами. Один из узлов является виртуальной машиной в системе виртуализации zVirt, а второй - физическим сервером.

2. Описание процесса добавления oVirt CSI

1. Настройка zVirt.
2. Настройка StarVault.
3. Настройка Nova Container Platform.
4. Проверка.

3. Настройка zVirt

Настройте пользователя и получите данные для настройки интеграции с zVirt согласно статье Интеграция с zVirt.

4. Настройка StarVault

1. Зайдите в веб-консоль StarVault с использованием root токена.
2. На вкладке **Secrets** перейдите в **nova-secrets** > **credentials** и создайте секрет с именем `ovirt-csi`.
3. Добавьте следующие ключи-значения в новый секрет:
   - `ovirt_ca_bundle` — цепочка корневых TLS-сертификатов для подключения к интерфейсу zVirt API.

     Пример: "LS0k1JSURxekNDQXBPZ0F3SUJBZ0lWUS0tLQo=…"

- `ovirt_password` — пароль от учётной записи пользователя zVirt
- `ovirt_url` — URL-адрес API среды виртуализации zVirt.

  Пример: "https://zvirt.mycompany.local/ovirt-engine/api"
- `ovirt_username` — имя пользователя zVirt

4. Перейдите на вкладку **Policies** и создайте новую политику с именем `nova-system-ovirt-csi` и следующей настройкой:

```
path "nova-secrets/data/credentials/ovirt-csi" { capabilities = ["read"] }
```
BASH

5. Перейдите на вкладку **Access** в раздел **Auth Methods** > **nova-kubernetes** и создайте новую роль со следующими параметрами:
   - **Name:** `nova-system-ovirt-csi`
   - **Alias name source:** `serviceaccount_uid`
   - **Bound service account names:**
     - `ovirt-csi-driver-controller-sa`
     - `ovirt-csi-driver-node-sa`
   - **Bound service account namespaces:** `nova-csi-drivers`
   - Раскройте блок `Tokens`
     - Включите опцию `Generated Token's Maximum TTL` и поставьте значение в 1 час
     - Включите опцию `Do Not Attach 'default' Policy To Generated Tokens`
     - В поле **Generated Token's Policies** добавьте имя политики созданной ранее
     - Включите опцию `Generated Token's Initial TTL` и поставьте значение в 1 час
     - В поле `Generated Token's Type` поставьте значение `default`

5. Настройка Nova Container Platform

1. В веб-консоли Nova Container Platform перейдите на вкладку **Узлы кластера** > **Nodes**.
2. Повторите следующие действия для всех **Worker** узлов на zVirt:

   a. Нажмите на выбранный узел

   b. Перейдите на вкладку YAML и добавьте метку:

```yaml
  labels:
    ovirt: 'true'
```

3. Создайте сервисные аккаунты и роли для них.

▸ **Манифесты**

4. Установите контроллер и агенты.

▸ **Манифесты**

Если платформа была установлена используя Universe, то замените в манифестах `hub.nova-platform.io/registry` на `hub.universe.mycompany.local/nova-universe`.
Не забудьте заменить `universe.mycompany.local` на FQDN вашего Universe сервера. Также вы можете взять это значение из поля `hubRegistryURL` в файле `nova-deployment-conf.yaml`.

5. Установите манифест для StorageClass

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ovirt-csi-sc
  namespace: nova-csi-drivers
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.ovirt.org
reclaimPolicy: "Delete"
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  storageDomainName: "zvirtCsiStorageDomainName" ①
  thinProvisioning: "true"
  fsType: "xfs"
```

1. Измените на имя вашего домена хранения в zVirt.

6. Проверка

1. Создайте новое пространство имён, PVC и под.

▸ **Пример**

2. Зайдите в консоль пода и создайте файл в директории `/usr/local/apache2/htdocs`

3. Удалите под и создайте его заново.
4. Убедитесь, что файл существует в директории `/usr/local/apache2/htdocs`
5. Перейдите на вкладку **Узлы кластера** > **Nodes**.
6. Повторите следующие действия для любого **Worker** узла, который является физическим сервером:
   a. Нажмите на выбранный узел.
   b. Перейдите на вкладку YAML и добавьте метку:

```yaml
  labels:
    ovirt: 'false'
```

7. Удалите под и создайте новый со следующим манифестом:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  namespace: test
spec:
  nodeSelector:
    ovirt: "false"
  containers:
  - name: httpd
    image: httpd:latest
    volumeMounts:
    - mountPath: /usr/local/apache2/htdocs
      name: httpd-storage
  volumes:
  - name: httpd-storage
    persistentVolumeClaim:
      claimName: httpd-pvc
```

8. Зайдите в под и перейдите на вкладку **События**. Убедитесь, что под не может быть создан из-за того, что на узле не установлен CSI.
9. Зайдите в веб-консоль zVirt и найдите созданный диск. Его имя будет начинаться с `pvc`.
10. В веб-консоли платформы удалите под и PVC.
11. Убедитесь, что диск удалился из zVirt.
12. Проверка завершена.

Kubernetes CLI (kubectl)

1. Kubectl

Для работы с Nova Container Platform вы можете использовать стандартный инструмент CLI `kubectl`. С помощью `kubectl` вы сможете развертывать приложения, проверять и управлять ресурсами кластера, а также получать диагностическую информацию.

Вы можете установить последнюю версию `kubectl`, руководствуясь официальной документацией Kubernetes по установке и настройке kubectl для разных ОС.

2. Kubelogin

Kubelogin является плагином инструмента `kubectl` и предоставляет возможность автоматизированного получения доступа к кластеру Kubernetes с помощью персональной учетной записи по протоколу OAuth (OpenID Connect).

Kubelogin удобно использовать, когда необходимо обеспечить доступ к кластеру множеству пользователей. При этом, нет необходимости подготавливать пользователям персональные `kubeconfig` конфигурации.

Nova Container Platform предоставляет универсальную конфигурацию (`kubeconfig`) для инструмента `kubectl`.

> **Дополнительная информация**
>
> Подробнее о работе Kubernetes API Server с токенами OpenID Connect (OIDC) можно узнать в официальной документации Kubernetes OpenID Connect Tokens.

2.1. Установка kubelogin

Вы можете установить последнюю версию `kubelogin` следующими способами:

- с помощью Homebrew для ОС macOS и Linux,
- используя менеджер плагинов `kubectl` Krew,
- используя менеджер пакетов для Windows Chocolatey
- из релизов GitHub

Пример установки `kubelogin`

▶ macOS, Linux

▶ Krew

▶ Chocolatey (Windows)

Github-релизы `kubelogin` также доступны в репозиториях Nova Container Platform:

▶ macOS

▶ Linux

▶ Windows

Для установки `kubelogin` из GitHub-релиза необходимо выполнить следующие действия:

1. Загрузить подходящую для вашей ОС версию `kubelogin`.
2. Переименовать бинарный файл `kubelogin` в файл `kubectl-oidc_login`.
3. Добавить путь к бинарному файлу `kubectl-oidc_login` в переменную окружения `PATH`.

Пример проверки установки `kubelogin`

BASH

```
$ which kubectl-oidc_login
/Users/nova/.krew/bin/kubectl-oidc_login

$ echo $PATH
/Users/nova/.krew/bin:/opt/homebrew/bin:/opt/homebrew/sbin
```

Дополнительная информация

Подробнее о работе плагинов `kubectl` можно узнать в официальной документации Kubernetes Extend kubectl with plugins.

3. Подключение к кластеру

Для управления кластером Kubernetes вам необходимо пройти процесс аутентификации с использованием зарегистрированной учетной записи.

Необходимые условия

- У вас есть сетевой доступ к кластеру Kubernetes
- Вы установили инструмент управления кластером `kubectl` и плагин `kubelogin`

> **Информация**
>
> Если ваш кластер доступен только через HTTP-прокси, вы можете установить переменные окружения `HTTP_PROXY`, `HTTPS_PROXY` и `NO_PROXY`. Эти переменные учитываются инструментом `kubectl` и обеспечат доступ к кластеру Kubernetes через ваш HTTP-прокси.

Процедура

1. Определите в консоли переменную `KUBECONFIG`, содержащую путь к полученному после установки файлу `kubeadmin.conf`:

   ```bash
   export KUBECONFIG=<путь>/admin.conf
   ```

2. Получите универсальную конфигурацию `kubeconfig` для доступа к кластеру по протоколу OAuth:

   ```bash
   kubectl get secret kubeconfig-oidc -n nova-authentication -o
   jsonpath='{.data.kubeconfig}' | base64 -d > kubeconfig-oidc.conf
   ```

3. Переопределите в консоли переменную `KUBECONFIG`, содержащую путь к полученному на предыдущем шаге файлу `kubeconfig-oidc.conf`:

   ```bash
   export KUBECONFIG=<путь>/kubeconfig-oidc.conf
   ```

4. Выполните любую пробную команду с помощью `kubectl`, например:

   ```bash
   kubectl get nodes
   ```

   При первом доступе к кластеру Kubernetes, а также при необходимости обновления учетных данных, вы будете автоматически перенаправлены на веб-страницу аутентификации через встроенный провайдер идентификации Nova Container Platform.

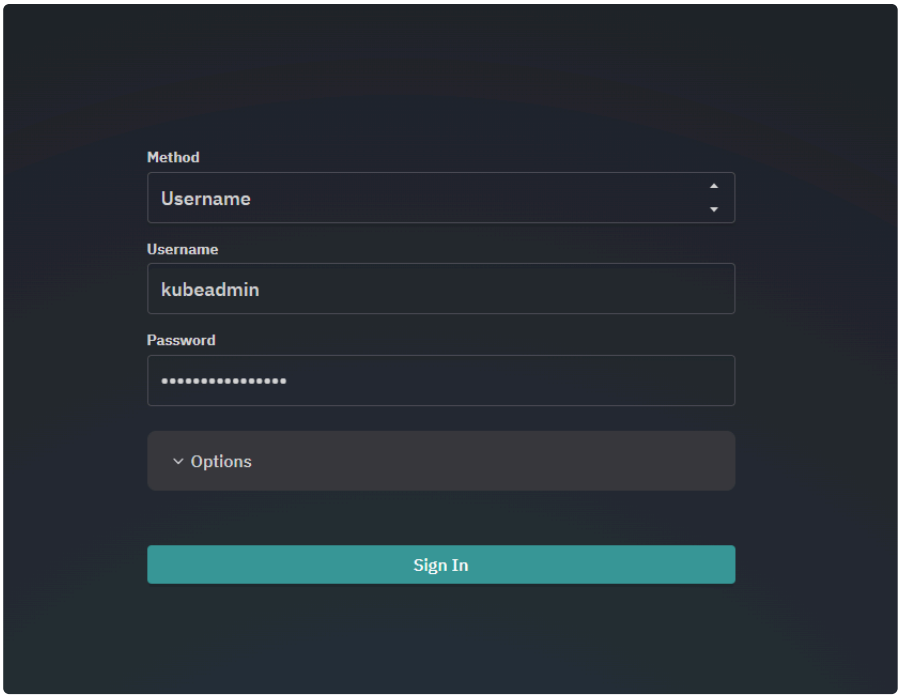

Рисунок 1. Страница аутентификации в кластере

Введите данные учетной записи администратора кластера `kubeadmin` для авторизации по протоколу OAuth выбрав метод аутентификации **Username**.

В случае успешной аутентификации вы будете перенаправлены на страницу с подтверждением возможности входа в кластер Kubernetes. Данную страницу можно закрыть.

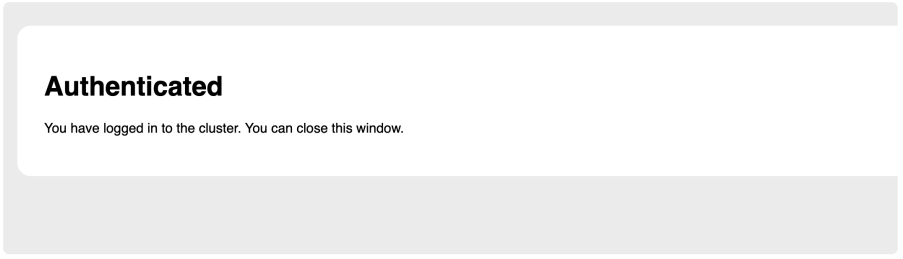

Рисунок 2. Подтверждение успешной аутентификации в кластере

Пробная команда `kubectl`, запущенная ранее, продолжит работу и возвратит запрошенную информацию.

Пример

BASH

```
$ kubectl get nodes

NAME                                          STATUS   ROLES           AGE
VERSION
node-master-ja4k4alk.nova-wp3sx2i4fwa8.local   Ready    control-plane   66m
v1.26.3
node-worker-c36i1ez0.nova-wp3sx2i4fwa8.local   Ready    ingress,worker   66m
v1.26.3
```

```
node-worker-itivaho6.nova-wp3sx2i4fwa8.local   Ready    infra           66m
v1.26.3
```

Ваши учетные данные (OIDC-токены), полученные таким образом, кешируются ограниченное время в файлах директории `~/.kube/cache/oidc-login/`.

4. Подключение к кластеру через промежуточный сервер

Вы можете подключиться к кластеру Kubernetes через промежуточный сервер, используя Nova OAuth.

Необходимые условия

- У вас есть сетевой доступ к промежуточному серверу
- У промежуточного сервера есть сетевой доступ к кластеру Kubernetes
- Вы установили инструмент управления кластером `kubectl` и плагин `kubelogin` на промежуточный сервер

Процедура

1. Получите универсальную конфигурацию `kubeconfig` для доступа к кластеру по протоколу OAuth и сохраните ее на промежуточном сервере:

```bash
kubectl get secret kubeconfig-oidc -n nova-authentication -o
jsonpath='{.data.kubeconfig}' | base64 -d > kubeconfig-oidc.conf
```

2. Добавьте в файл дополнительные параметры промежуточного сервера:

```yaml
apiVersion: v1
clusters:
...
- name: oidc-kubernetes-client
user:
    exec:
    apiVersion: client.authentication.k8s.io/v1beta1
    args:
...
    - --listen-address=< адрес промежуточного сервера >:< порт >
    - --skip-open-browser
    - --oidc-redirect-url-hostname=< адрес промежуточного сервера >
```

Информация

В качестве адреса промежуточного сервера вы можете указать IP-адрес или полное доменное имя (FQDN).

Пример

```yaml
apiVersion: v1
clusters:
...
- name: oidc-kubernetes-client
user:
    exec:
    apiVersion: client.authentication.k8s.io/v1beta1
    args:
...
    - --listen-address=172.31.100.10:8000
    - --skip-open-browser
    - --oidc-redirect-url-hostname=172.31.100.10
```

3. Добавьте адрес перенаправления (Redirect URI) в список разрешенных для клиента Kubernetes:
    - Для этого перейдите выполните вход в веб-интерфейс StarVault с помощью учетной записи администратора. Веб-интерфейс StarVault находится по адресу 'https://nova-oauth.< dnsBaseDomain >/'.
    - Перейдите в раздел Access > OIDC Provider и выберете приложение `oidc-kubernetes-client`.
    - Отредактируйте параметры приложения с помощью кнопки “Edit Application” и добавьте в список `Redirect URIs` адрес вашего промежуточного сервера, например, `http://< адрес промежуточного сервера >:< порт >`.
    - Сохраните конфигурацию приложения.
4. Переопределите в консоли переменную `KUBECONFIG`, содержащую путь к полученному на шаге 2 файлу `kubeconfig-oidc.conf`:

```bash
export KUBECONFIG=<путь>/kubeconfig-oidc.conf
```

5. Выполните любую пробную команду с помощью `kubectl` на промежуточном сервере, например:

```bash
kubectl get nodes

Please visit the following URL in your browser: http://172.31.100.10:8000
```

При первом доступе к кластеру Kubernetes через промежуточный сервер, а также при необходимости обновления учетных данных, вам будет предложено перейти в веб-

интерфейс для прохождения процедуры аутентификации.

После успешного прохождения процедуры ваши учетные данные (OIDC-токены), полученные таким образом, будут сохранены на ограниченное время в файлах домашней директории `~/.kube/cache/oidc-login/` промежуточного сервера.
