Балансировка TCP\UDP трафика ingress-контроллером

У Ingress-контроллера nginx существует функция опубликования не только HTTP портов, но любых других TCP\UDP портов: https://kubernetes.github.io/ingress-nginx/user-guide/exposing-tcp-udp-services/

Настройка данного функционала требует изменения нескольких компонентов кластера, которые будут рассмотрены далее.

1. Тестовая задача

Допустим, у нас есть Deployment, поды которого слушают TCP порты 1540 и 1541. У нас в кластере уже созданы сервисы для каждого пода и теперь нам нужно опубликовать данные порты наружу с помощью ingress-контроллера.

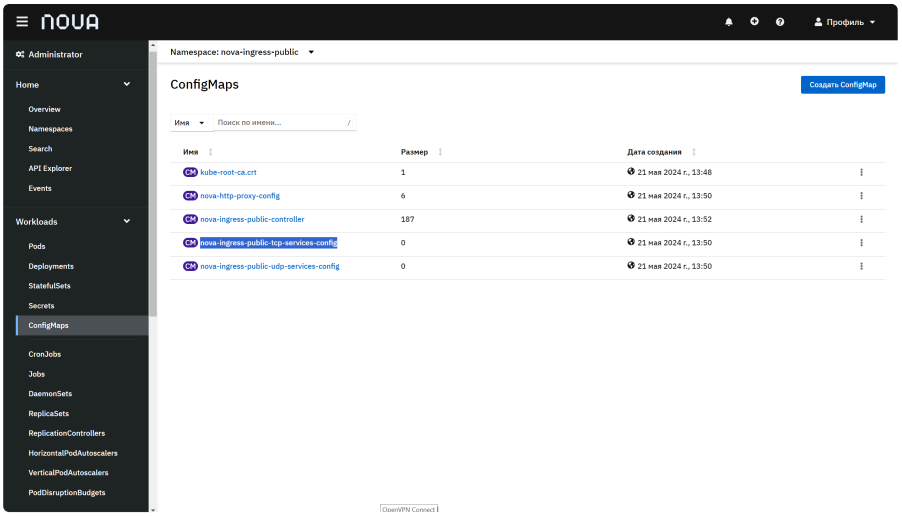
2. Реализация

Для выполнения данной задачи нам потребуется внести изменения в 2 объекта:

- ConfigMap - `nova-ingress-public-tcp-services-config`
- Service - `nova-ingress-public-controller`

Так как service `nova-ingress-public-controller` управляется с помощью FluxCD, изменения будут вноситься не напрямую в объект Service, а через объект Kustomization `nova-release-ingress-public-main`

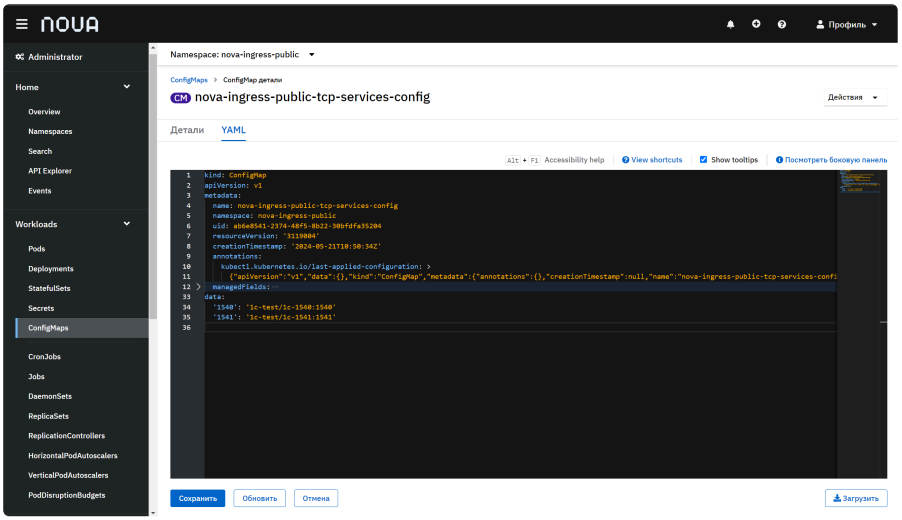
1. Откройте в браузере веб-интерфейс Nova Console и перейдите в раздел Workloads → ConfigMaps. Переключите фильтр Namespace на `nova-ingress-public` и из списка ConfigMaps выберите `nova-ingress-public-tcp-services-config`



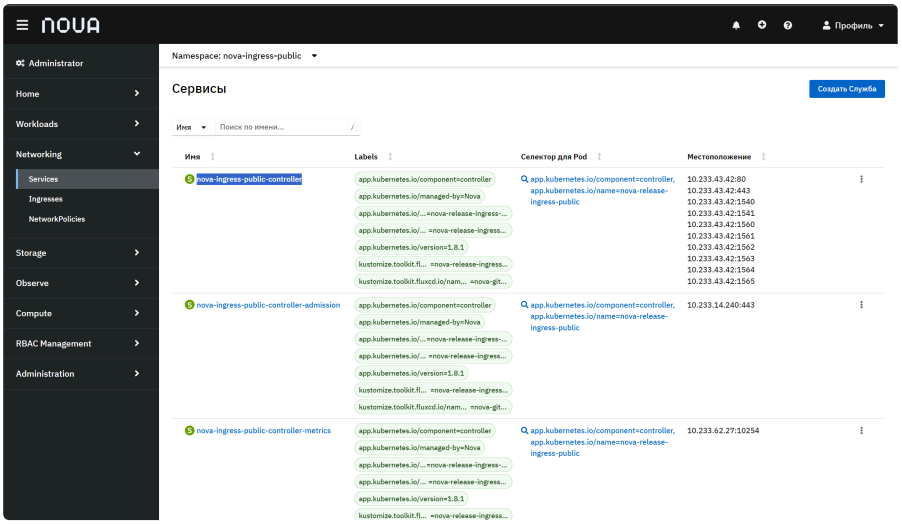
2. На открывшейся странице перейдите на вкладку YAML и вставьте следующие строки в редактор:

```yaml
data:
  1540: "1c-test/1c-1540:1540"
  1541: "1c-test/1c-1541:1541"
```

В данном примере строка имеет следующий формат: `<порт_который_слушает_nginx>: "<namespace>/<service>:<порт_который_слушает_service>"` После внесения изменений нажмите "Сохранить":

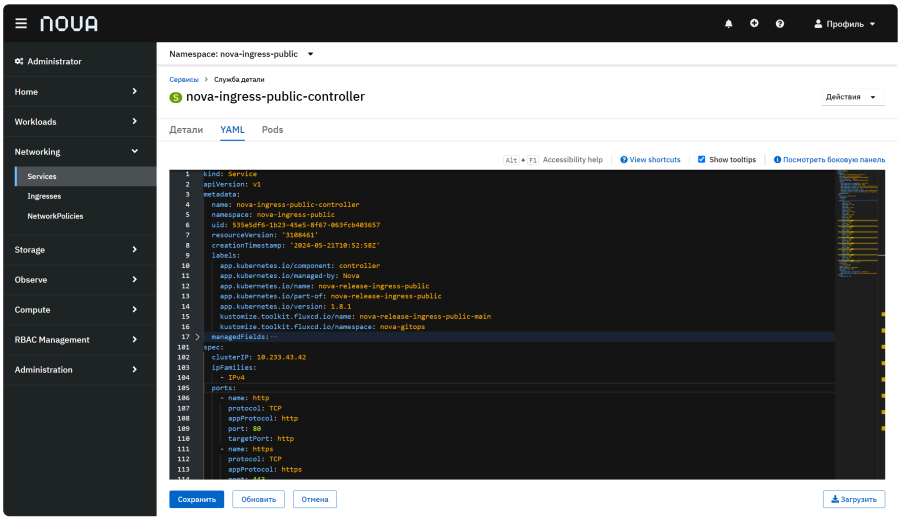


3. Перейдите в раздел Networking → Services, переключите фильтр Namespace на `nova-ingress-public` и из списка выберите `nova-ingress-public-controller`

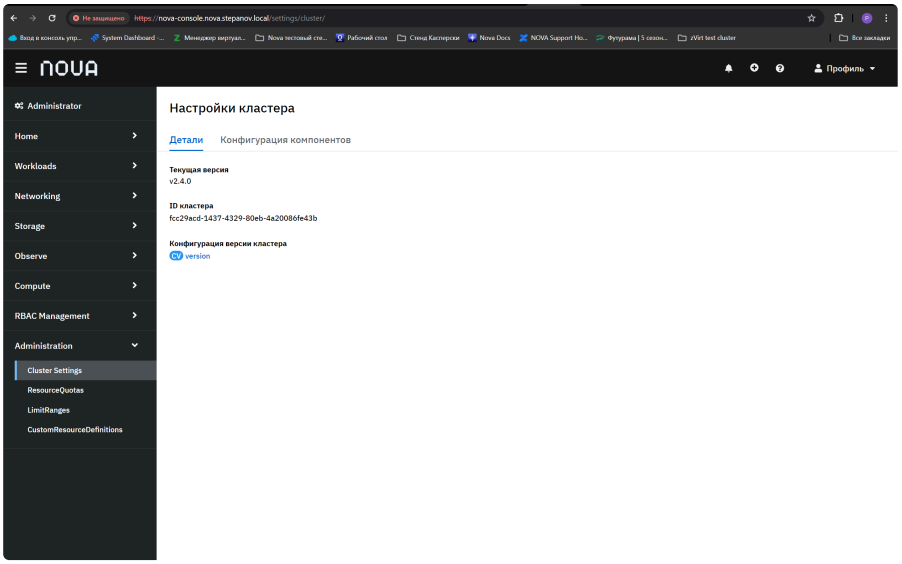


4. На открывшейся странице перейдите на вкладку YAML. Так как у данного объекта присутствуют следующие лейблы, это означает что service `nova-ingress-public-controller` управляется с помощью FluxCD, изменения будут вноситься не напрямую в объект Service, а через объект Kustomization `nova-release-ingress-public-main`:

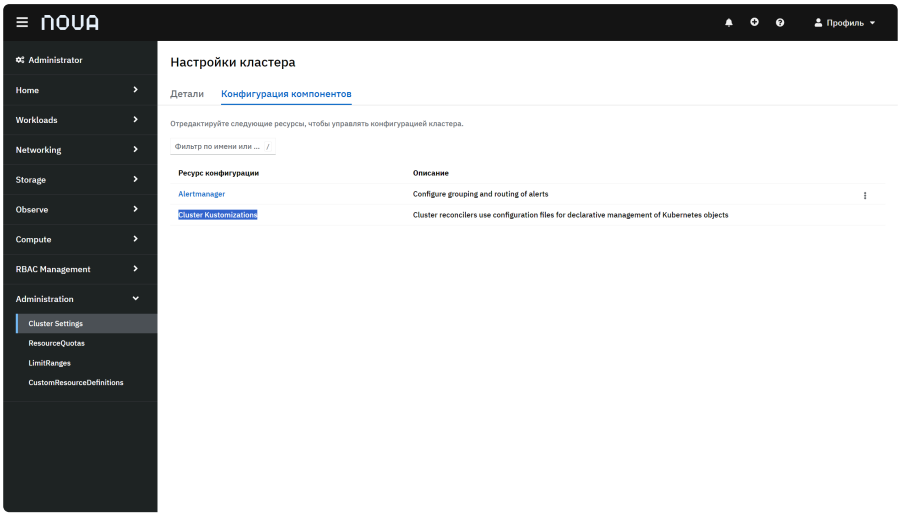
```yaml
app.kubernetes.io/managed-by: Nova
kustomize.toolkit.fluxcd.io/name: nova-release-ingress-public-main
kustomize.toolkit.fluxcd.io/namespace: nova-gitops
```



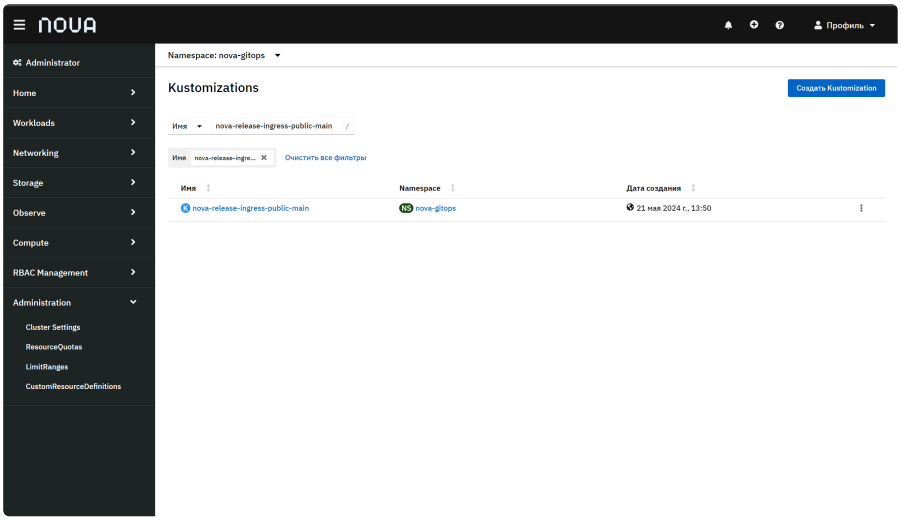
5. Откройте еще одну вкладку браузера с веб-интерфейсом Nova Console и перейдите в раздел Administration → ClusterSettings



6. На открывшейся странице перейдите на вкладку "Конфигурация компонентов" и выберите из списка "Cluster Kustomizations"



7. На открывшейся странице найдите имя объекта Kustomization из п.4 (nova-release-ingress-public-main)



8. На открывшейся странице перейдите на вкладку YAML. Здесь нам нужно применить патч для изменения объекта `Service\nova-ingress-public-controller`. Существует несколько способов применения патчей к объектам Kustomization, которые подробно

описаны в документации FluxCD
https://fluxcd.io/flux/components/kustomize/kustomizations/#patches
В данном примере мы будем использовать формат патча JSON6902. Для этого нам нужно добавить к текущему манифесту раздел patches, в котором мы опишем наш patch. Наш патч имеет следующий вид:

```yaml
  patches:
    - patch: |
        - op: add ①
          path: /spec/ports/0 ②
          value: { "name": "1c-1540", "port": 1540, "targetPort": 1540,
"protocol": "TCP"} ③
        - op: add ①
          path: /spec/ports/1 ②
          value: { "name": "1c-1541", "port": 1541, "targetPort": 1541,
"protocol": "TCP"} ③
      target: ④
        kind: Service
        name: nova-ingress-public-controller
        namespace: nova-ingress-public
```

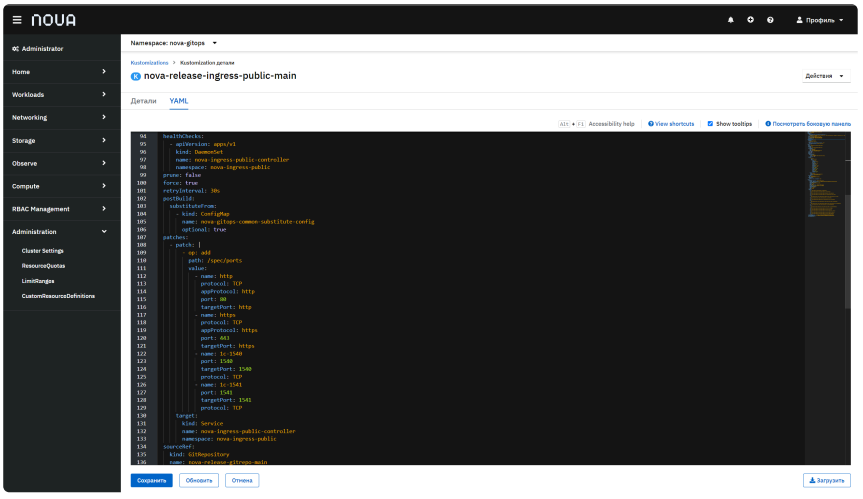
① Параметр `op` описывает действие `add` (добавить).

② Параметр `path` описывает место в манифесте, куда стоит внести изменения. В данном случае указывается список портов, поэтому можно добавить новые порты в уже существующий список указывая номер списка. Данные которые уже присутствуют в списке не удаляются, а сдвигаются дальше по списку.

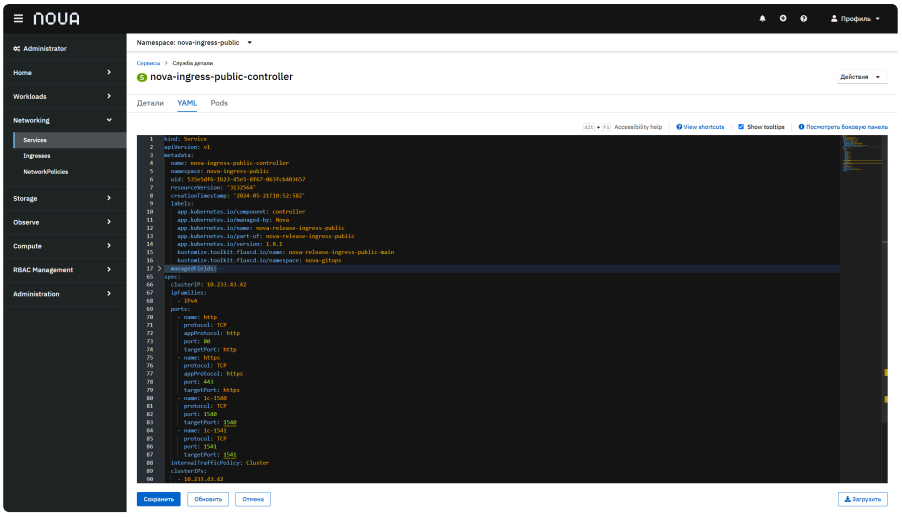
③ Параметр `value` описывает значение, которое нужно добавить в данное место.

④ Параметр `target` описывает к какому объекту будет применен патч, в нем мы указываем тип объекта, имя и namespace.

Вставьте данный патч в редактор в секцию spec и нажмите "Сохранить"



9. Вернитесь на вкладку с объектом `service/nova-ingress-public-controller` из п.4 и убедитесь, что в YAML манифесте объекта появилось описание новых портов (возможно понадобится нажать кнопку "Обновить"):



Теперь ingress-контроллер слушает указанные порты на worker-узлах (или выделенных ingress-узлах) и перенаправляет запросы в соответствии с конфигурацией.

Архитектура и концепции модуля Neuvector

В данном разделе представлено описание концепций и архитектуры платформы обеспечения безопасности Neuvector.

1. Компоненты модуля

Модуль Neuvector в кластере Nova Container Platform включает компоненты, представленные в таблице ниже:

Компонент	**Категория**	**Количество**
Controller	Кластерный компонент	1 или 3
Enforcer	Хостовой компонент (агент)	1 на узел
Scanner	Кластерный компонент	1 и более
Manager	Кластерный компонент	1 и более
Registry Adapter	Кластерный компонент	1 и более
Updater	Кластерный компонент	1
API docs	Кластерный компонент	1 и более
Provisioner	Кластерный компонент	1
Prometheus Exporter	Кластерный компонент	1

Компоненты Neuvector являются контейнерами, которые взаимодействуют между собой, так и с кластером Kubernetes и его узлами. Все компоненты Neuvector запускаются и функционируют в среде Kubernetes.

- **Controller**: основной компонент (контроллер), реализующий функции управления и координации действий других компонентов. В Nova Container Platform контроллер разворачивается в виде сущности StatefulSet. В зависимости от количества инфраструктурных узлов в кластере автоматически устанавливается необходимое количество реплик. Также контроллер предоставляет REST API для управления сущностями Neuvector.

- **Enforcer**: хостовый компонент (агент), который разворачивается в виде сущности DaemonSet, тем самым размещаясь на каждом узле кластера Kubernetes. Задача агента Enforcer состоит в том, чтобы отслеживать потоки сетевого трафика, процессы и файловые системы контейнеров и применять установленные политики безопасности.
- **Scanner**: компонент (сканер) выполняет задачи по сканированию образов контейнеров, используя встроенную базу уязвимостей (CVE). Сканеров может быть от 1 и более в кластере Kubernetes для увеличения скорости при сканировании большого количество образов. Сканер имеет политику загрузки собственного образа (ImagePullPolicy) в Kubernetes с типом *Always* и установленный тег *latest*. Тем самым, перезапускаясь, сканер всегда запускается с последней доступной в хранилище версией образа и базой CVE.
- **Manager**: компонент предоставляет веб-интерфейс управления сущностями Neuvector и взаимодействует с контроллером через REST API.
- **Registry Adapter**: адаптер для использования во внешних хранилищах образов (например, Harbor) с целью осуществлять сканирование образов по базе CVE Neuvector.
- **Updater**: компонент реализует процесс обновления базы CVE с помощью запуска CronJob, выполняющей поочередный перезапуск всех сканеров.
- **API docs**: веб-портал с документацией к REST API Neuvector.
- **Provisioner**: компонент, запускаемый однократно в виде сущности Job для первоначальной настройки Neuvector.
- **Prometheus Exporter**: компонент для сборка метрик со всех компонентов Neuvector.

2. Архитектурная схема

На рисунке далее представлена архитектурная схема платформы Neuvector.

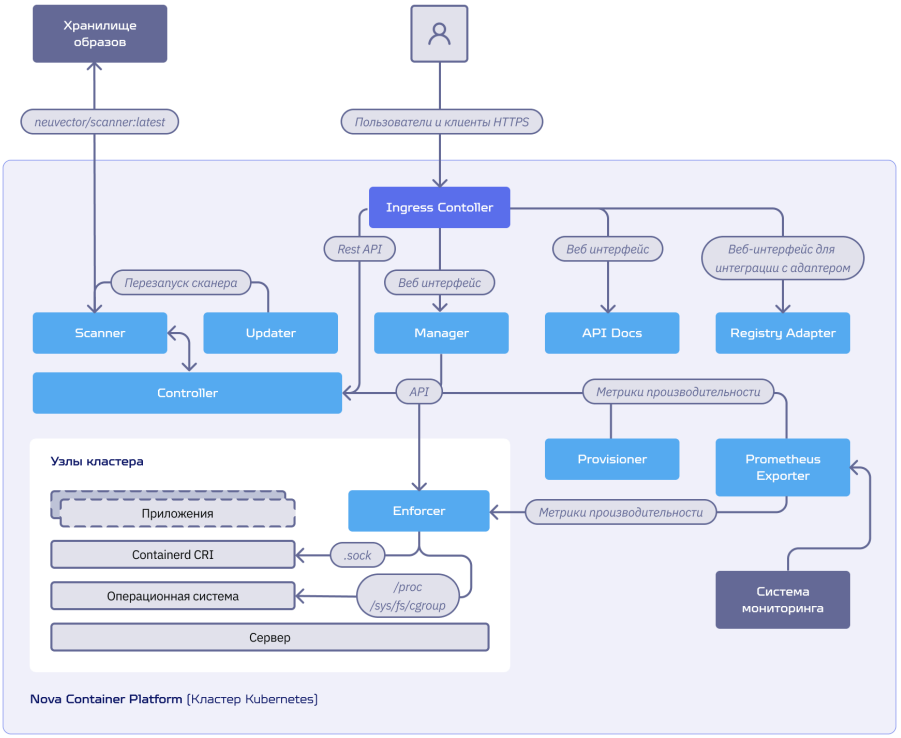


Рисунок 1. Архитектура платформы Neuvector

В кластере Kubernetes модуль Neuvector развертывается с помощью службы непрерывного развертывания FluxCD, которая также обеспечивает дальнейшую консистентность конфигураций компонентов Neuvector в Kubernetes и возможности их кастомизации.

На рисунке далее представлена схема развертывания платформы Neuvector в Nova Container Platform.

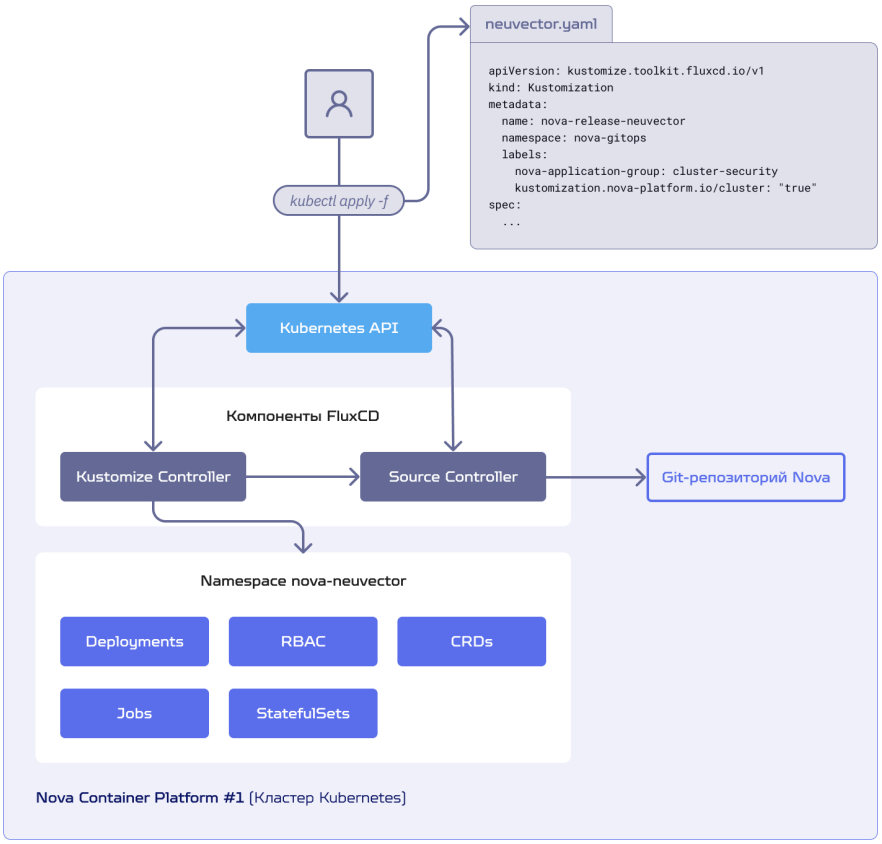


Рисунок 2. Схема развертывания платформы Neuvector

Используемый в Nova Container Platform Git-репозиторий Gitea содержит все необходимые манифесты для развертывания платформы Neuvector. Пользователю платформы необходимо применить в кластере манифесты *Kustomizations*, предназначенные для установки Neuvector. Для этого можно воспользоваться как утилитой `kubectl`, так и веб-консолью Nova.

Модуль Neuvector по умолчанию не установлен в кластере Kubernetes с целью экономии вычислительных ресурсов. Ознакомьтесь с разделом Планирование установки и системные требования для подготовки Nova Container Platform к установке модуля Neuvector.

3. Высокая доступность и непрерывность работы

Под высокой доступностью и непрерывностью работы компонентов Neuvector предполагается использование количества реплик компонентов более 1 для того, чтобы избежать прерывания работы сервисов в период недоступности части узлов платформы или перезапуска части сервисов.

Далее для каждого компонента Neuvector рассмотрены особенности его работы в режиме высокой доступности.

Controller

Для повышения доступности контроллеров необходимо использовать строго нечетное количество реплик. Рекомендуемое и достаточное количество реплик - 3. Контроллеры имеют встроенные механизмы выбора лидера и синхронизируют конфигурацию друг с другом.

Enforcer

Данный компонент разворачивается в виде сущности DaemonSet, тем самым размещаясь на каждом узле кластера Kubernetes. Сценарий с увеличением реплик в данном случае не применим.

Scanner

Компонент требует обязательного увеличения количества реплик в случаях, когда вы работаете с большим количеством образов или время сканирования образов неудовлетворительное. Для сканера в Neuvector поддерживается автоматическое увеличение количества реплик.

Manager

Веб-интерфейс управления сущностями Neuvector взаимодействует с контроллером через REST API и не влияет на работу ключевых сервисов Neuvector. Для повышения доступности веб-интерфейса количество реплик может быть увеличено. Постоянство пользовательских сессий обеспечивается уже установленной директивой метода балансировки в Ingress Controller `nginx.ingress.kubernetes.io/upstream-hash-by: "$binary_remote_addr"`.

Registry Adapter

Количество реплик *Registry Adapter* может быть увеличено при необходимости повышения доступности адаптера(ов), процессов сканирования образов во внешних хранилищах.

Updater

Поскольку компонент Updater является сущностью типа *CronJob* в Kubernetes и запускается только для перезапуска компонента *Scanner*, то высокая доступность и большое количество реплик компоненту не требуются.

API docs

Веб-портал с документацией к REST API не является критичным сервисом для работы Neuvector, поэтому достаточно иметь одну реплику в кластере Kubernetes.

Provisioner

Поскольку компонент Provisioner является сущностью типа *Job* в Kubernetes и запускается однократно при первоначальной установке, то высокая доступность и большое количество реплик компоненту не требуются.

Prometheus Exporter

В кластере Kubernetes в большинстве случаев достаточно иметь одну реплику Prometheus Exporter для сбора метрик Neuvector.

4. Персистентность данных в Neuvector

Конфигурации Neuvector и зарегистрированные события синхронизируются между доступными контроллерами. В случае, если все контроллеры будут недоступны одновременно, то некоторые данные могут быть потеряны, а именно:

- Параметры и конфигурации, политики безопасности установленные вручную через API или веб-интерфейс Neuvector
- Данные о сетевых соединениях, сканированиях
- Зарегистрированные события безопасности

Доступность данных (состояния) обеспечивает как минимум один контроллер Neuvector.

Для возможности постоянного хранения собственных конфигураций Neuvector поддерживает использование сущностей *ConfigMap* в Kubernetes. При этом, Neuvector не сохраняет данные в *ConfigMap* самостоятельно, а только загружает их при первом и каждом последующем запуске.

4.1. Отличие Neuvector в Nova Container Platform

По умолчанию, для хранения данных архитектура Neuvector предполагает использование постоянного тома в Kubernetes с типом доступа RWX (Read Write Many). Данный том подключается в директорию `/var/neuvector/` каждого контроллера, куда сохраняются все данные Neuvector.

Поскольку организация файлового хранилища с множественным доступом (RWX) несет дополнительные накладные расходы (например, отсутствие поддержки в публичном или частном облаке, необходимость организации отдельного NFS-хранилища и т.п.), в Nova Container Platform изменен подход к организации персистентности данных согласно следующим принципам:

1. Контроллеры Neuvector разворачиваются в виде сущности StatefulSet вместо Deployment с привязкой к инфраструктурным узлам.
2. Каждый контроллер Neuvector имеет отдельный смонтированный в директорию `/var/neuvector/` постоянный том из хранилища Local Path.
3. Для всех контроллеров установлен параметр `CTRL_PERSIST_CONFIG`, разрешающий выполнять сохранение всех данных в `/var/neuvector/`.

4. Вместо базовых объектов *ConfigMap* для хранения конфигураций, используются единый объект *Secret*, который инжектируется напрямую в Neuvector с помощью StarVault. Это позволяет безопасно использовать конфигурационные файлы Neuvector в Kubernetes, а также сохраняет возможность декларативного описания данных конфигураций.

4.2. Рекомендации по обеспечению доступности данных в Neuvector

Для обеспечения сохранности данных в Neuvector рекомендуется выполнить следующие шаги:

- Настроить периодическое резервное копирование конфигураций и политик Neuvector.
- Настроить интеграцию с Syslog-сервером и SIEM-системой для экспорта информации о событиях безопасности, системных событиях, атаках, уязвимостях.

В качестве Syslog-сервера и хранилища журналов событий вы можете использовать подключаемый модуль логирования в Nova Container Platform.

5. Интеграция с StarVault

5.1. Единый вход с помощью Nova Oauth

После установки модуля Neuvector возможен вход с помощью учетной записи по умолчанию `kubeadmin`. Для этого между Neuvector и StarVault настроена интеграция по протоколу OIDC.

5.2. Конфигурации контроллера Neuvector

Каждый Pod контроллера Neuvector включает Sidecar-контейнер с StarVault, работающий в режиме агента. Перед запуском основного контейнера Neuvector, Sidecar-контейнер получает из StarVault необходимую информацию и генерирует конфигурационные файлы, а именно:

- `oidcinitcfg` - конфигурация параметров OIDC
- `passwordprofileinitcfg` - конфигурация параметров профилей паролей
- `roleinitcfg` - конфигурация базовых ролей
- `samlinitcfg` - конфигурация параметров SAML
- `ldapinitcfg` - конфигурация параметров LDAP
- `sysinitcfg` - системные параметры Neuvector

- `userinitcfg` - конфигурация системных пользователей Neuvector

Шаблоны конфигурационных файлов без чувствительной информации находятся в объекте *Secret* с именем `neuvector-init-template` в пространстве имен `nova-neuvector`.

6. Следующие шаги

- Планирование установки и системные требования

Проверка уязвимостей в кластере

1. Общие сведения

Компонент Scanner в системе безопасности NeuVector (далее - сканер) отвечает за сканирование уязвимостей и проверку соответствия стандартам безопасности образов контейнеров и узлов кластера.

Сканер развернут в Kubernetes в составе системы Neuvector в виде отдельного ресурса Deployment и выполняет следующие функции:

- **Сканирование образов в реестрах**: подключается к указанным реестрам образов контейнеров, извлекает список доступных образов и проводит их анализ на наличие уязвимостей. Для повышения производительности и масштабируемости для сканнера поддерживается автоматическое масштабирование количества его реплик, которые могут параллельно сканировать образы в реестрах.
- **Сканирование на этапе сборки**: интегрируется с процессами CI/CD, позволяя сканировать образы на наличие уязвимостей еще на этапе сборки, до их размещения в реестре или развертывания в среде выполнения.
- **Сканирование узлов и контейнеров в реальном времени**: автоматически сканирует запущенные контейнеры и узлы на наличие уязвимостей и соответствие стандартам безопасности.

2. Источники уязвимостей

Сканер в Neuvector содержит в себе базу данных уязвимостей (CVE), которая использует следующие источники:

- Общие источники информации об уязвимостях:
  - Банк данных угроз безопасности информации ФСТЭК
  - РЕД ОС
  - NVD (National Vulnerability Database)
  - Mitre
- Операционные системы:
  - Astra Linux, РЕД ОС

    - Alpine, Amazon, Debian, Microsoft Mariner, Oracle, Rancher OS, Red Hat, SUSE Linux, Ubuntu
- Приложения и языки программирования:
    - .NET, Apache, BusyBox, GoLang, Java, Maven, Kubernetes, Nginx, npm/Node.js, Python, OpenSSL, Ruby.

Актуализация баз уязвимостей выполняется один раз в сутки. По умолчнию в Nova Container Platform для сканнера установлен автоматический перезапуск каждый день в 00:00. При перезапуске загружается новая версия сканера с актуальной базой уязвимостей.

3. Обновление базы уязвимостей

3.1. Обновление при онлайн-установке

Если ваш кластер Nova Container Platform установлен в онлайн-режиме, то для обновления баз уязвимостей (сканера) дополнительные настройки не требуются. Актульная версия сканера будет загружаться ежедневно из публичных репозиториев Nova.

3.2. Обновление при офлайн-установке

Для регулярного обновления сканера в кластерах, установленных без доступа к сети Интернет с использованием Nova Universe вы можете использовать варианты ниже.

3.3. Кеширование образа сканера в корпоративном реестре

Вы можете настроить кеширование публичного репозитория Nova или образа сканера в частности в собственном корпоративном реестре образов и использовать его для обновления. Для настройки следуйте процедуре ниже:

1. Настройте кеширование публичного репозитория Nova или образа сканера. Актуальная версия сканера доступна по ссылке: `hub.nova-platform.io/registry/neuvector/scanner-nova:latest`.

    Для получения токена доступа к публичному репозиторию Nova при офлайн-установке обратитесь в техническую поддержку.

2. Если доступ в корпоративный реестр осуществляется с обязательным использованием учетной записи, подготовьте секрет Kubernetes:

```bash
kubectl create secret docker-registry custom-registry-credentials -n nova-neuvector \
--docker-server=<Адрес реестра> \
--docker-username=<Имя пользователя> \
--docker-password=<Пароль или токен> \
--docker-email=<Почтовый адрес пользователя>
```

3. В веб-консоли Nova Container Platform перейдите на вкладку **Administration** > **CustomResourceDefinitions**, найдите ресурс Kustomization, перейдите на вкладку **Экземпляры**, найдите `nova-release-neuvector-main`.
4. На вкладке **YAML** добавьте блок `patches`:

```yaml
spec:
  patches:
    - patch: |-
        apiVersion: apps/v1
        kind: Deployment
        metadata:
          name: neuvector-scanner-pod
          namespace: nova-neuvector
        spec:
          template:
            spec:
              imagePullSecrets: ①
              - name: custom-registry-credentials
              containers:
              - name: neuvector-scanner-pod
                image: "" ②
      target:
        kind: Deployment
        name: neuvector-scanner-pod
        namespace: nova-neuvector
```

① Блок конфигурации учетной записи для доступа к корпоративному реестру (при необходимости).

② Адрес образа в корпоративном реестре с тегом latest.

3.4. Загрузка образа сканера в корпоративный реестр

Если ваш корпоратинвый реестр не поддерживает кеширование и проксирование запросов, вы можете получить актуальный образ сканера в архиве, обратившись в техническую поддержку. Вы также можете самостоятельно выгрузить актуальный образ сканера вручную, используя утилиты docker, podman и аналогичные, из публичного репозитория Nova. Далее вам необходимо будет самостоятельно загрузить данный образ в ваш реестр образов.

Обратите внимание, что для корректного использования собственного реестра образов в кластере Nova, необходимо добавить соответствующие TLS- сертификаты в доверенные. Это можно сделать на этапе установке кластера, используя параметр спецификации `caTrustBundle` . Подробную информацию можно получить в разделе документации по ссылке.
